LabJack
Measurement & Automation

T-Series Datasheet

Nov 14 2023

Table of Contents

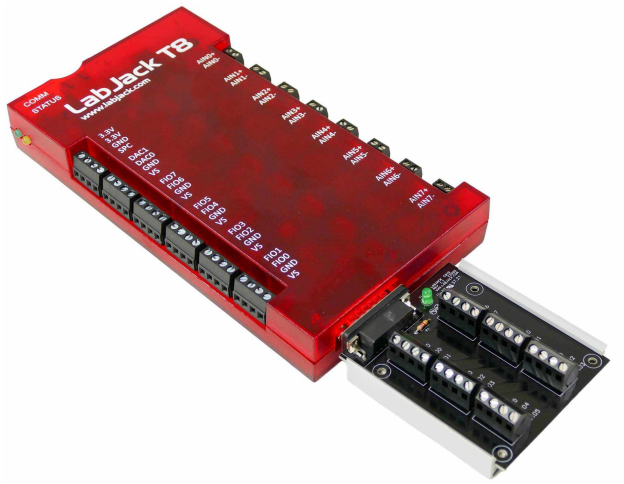
T-Series Datasheet Overview

High performance multifunction DAQ with USB, Ethernet, and WiFi.

This datasheet covers all T-series variants:

- **T4**
- **T4-OEM**
- **T7**
- **T7-OEM**
- **T7-PRO**
- **T7-PRO-OEM**
- **T8**

These pages form the complete datasheet, manual, and user's guide for the T4, T7 and T8. Most information in this datasheet applies to all T4 and T7 variants as well as the T8. Information about WiFi only applies to the T7-Pro variants and the high-resolution ADC (ResolutionIndex = 9-12) applies to the T7-Pro variants in addition to the T8. There is an OEM section with information specific to the build of OEM versions.



Software

T-series devices are supported by the LJM library, which simplifies device communication.

T-series devices use the Modbus protocol, so direct Modbus may be used for communication.

See here for a full list of software options.

Offline PDF

Preface: Warranty, Liability, Compliance [T-Series Datasheet]

For the latest version of this and other documents, go to www.labjack.com.

Warranty

Warranty:

All LabJack hardware is covered by a 5-year limited warranty, covering products and parts against defects in material or workmanship. We will troubleshoot, repair, or replace with like product to make sure you have a device that is operating to specifications.

LabJack products are very robust, but subject to the influence of user connections. The warranty does not apply if damaging mistakes were made in connecting our device, or if inspection reveals obvious signs of improper use, however we will still make a reasonable attempt to fix such devices for free if possible.

The warranty cannot be honored on discontinued / EOL products if replacements are unavailable and repair is not reasonably possible.

LabJack has taken great care of our customers since we sold our first device in 2001, and your satisfaction is our highest priority. If you suspect a problem with a device contact us at support@labjack.com.

LabJack is not liable for any losses, expenses or damages beyond the LabJack device itself. See our Limitation of Liability for more details.

Limitation of Liability

Limitation of Liability:

LabJack designs and manufactures measurement and automation peripherals that enable the connection of a PC to the real world. Although LabJacks have various redundant protection mechanisms, it is possible, in the case of improper and/or unreasonable use, to damage the LabJack and even the PC to which it is connected. LabJack Corporation will not be liable for any such damage.

Except as specified herein, LabJack Corporation makes no warranties, express or implied, including but not limited to any implied warranty or merchantability or fitness for a particular purpose. LabJack Corporation shall not be liable for any special, indirect, incidental or consequential damages or losses, including loss of data, arising from any cause or theory.

LabJacks and associated products are not designed to be a critical component in life support or systems where malfunction can reasonably be expected to result in personal injury. Customers using these products in such applications do so at their own risk and agree to fully indemnify

LabJack Corporation for any damages resulting from such applications.

LabJack assumes no liability for applications assistance or customer product design. Customers are responsible for their applications using LabJack products. To minimize the risks associated with customer applications, customers should provide adequate design and operating safeguards.

T-Series Compliance

Conformity Information (FCC, CE, RoHS):

See the Conformity Page and the text below:

FCC PART 15 STATEMENTS:

This equipment has been tested and found to comply with the limits for a Class A digital device, pursuant to Part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communications. Operation of this equipment in a residential area is likely to cause harmful interference in which case the user will be required to correct the interference at his own expense. The end user of this product should be aware that any changes or modifications made to this equipment without the approval of the manufacturer could result in the product not meeting the Class A limits, in which case the FCC could void the user's authority to operate the equipment.

Declaration of Conformity:

Manufacturers Name: LabJack Corporation
Manufacturers Address: 6900 W. Jefferson Ave Suite 110 Lakewood Colorado 80235 USA

Declares that these products

Product Name: LabJack T4
Model Number: LJT4

Product Name: LabJack T7 (-Pro)
Model Number: LJT7 (-Pro)

Product Name: LabJack T8

Model Number: LJT8

conform to the following Product Specifications:

EMC Directive: 2004/104/EEC

EN 55011 Class A

EN 61326-1: General Requirements

and is marked with CE

RoHS2:

The T4, T8 and T7 (-Pro) are RoHS compliant to Directive 2011/65/EU of the European Parliament on the restriction of the use of certain hazardous substances in electrical and electronic equipment.

REACH:

The T4, T8 and T7 (-Pro) are REACH compliant. REACH Product Compliance Program has been implemented in accordance with Regulation No. 1907/2006 of the European Parliament and the Council of 18 December 2006. LabJack Corporation does not currently have a direct REACH obligation to pre-register substances. LabJack's REACH Product Compliance is determined by a certification from our supply chain. LabJack products are deemed to be REACH compliant when they do not contain Substances of Very High Concern (SVHCs) beyond the specified concentration limits of less than 0.1% by weight as outlined in REACH 1907/2006/EU regulation.

CFM:

LabJack Corporation does not knowingly use these minerals or any by-products, as specified by the Conflict Minerals Trade Act.

Compliance Information for the WiFi module in the T7-Pro and T7-Pro-OEM

FCC:

Contains FCC ID: T9J-RN171
This device complies with Part 15 of the FCC Rules. Operation is subject to the following two conditions: (1) this device may not cause harmful interference, and (2) this device must accept any interference received, including interference that may cause undesired operation.

This equipment has been tested and found to comply with the limits for a Class B digital device, pursuant to part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference in a residential installation. This equipment generates, uses and can radiate radio frequency energy, and if not installed and used in accordance with the instructions, may cause harmful interference to radio communications. However, there is no guarantee that interference will not occur in a particular installation. If this equipment does cause harmful interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures:
• Reorient or relocate the receiving antenna.
• Increase the separation between the equipment and receiver.
• Connect the equipment into an outlet on a circuit different from that to which the receiver is connected.
• Consult the dealer or an experienced radio/TV technician for help.

To satisfy FCC RF Exposure requirements for mobile and base station transmission devices, a separation distance of 20 cm or more should be maintained between the antenna of this device and persons during operation. To ensure compliance, operation at closer than this distance is not recommended. The antenna(s) used for this transmitter must not be co-located or operating in conjunction with any other antenna or transmitter.

Canada:

Contains transmitter module IC: 6514A-RN171

This device complies with Industry Canada license-exempt RSS standard(s). Operation is subject to the following two conditions: (1) this device may not cause interference, and (2) this device must accept any interference, including interference that may cause undesired operation of the device.

Le présent appareil est conforme aux CNR d'Industrie Canada applicables aux appareils radio exempts de licence. L'exploitation est autorisée aux deux conditions suivantes: (1) l'appareil ne doit pas produire de brouillage, et (2) l'utilisateur de l'appareil doit accepter tout brouillage radioélectrique subi, même si le brouillage est susceptible d'en compromettre le fonctionnement.

Under Industry Canada regulations, this radio transmitter may only operate using an antenna of a type and maximum (or lesser) gain approved for the transmitter by Industry Canada. To reduce potential radio interference to other users, the antenna type and its gain should be so chosen that the equivalent isotropically radiated power (e.i.r.p.) is not more than that necessary for successful communication.

Conformément à la réglementation d'Industrie Canada, le présent émetteur radio peut fonctionner avec une antenne d'un type et d'un gain maximal (ou inférieur) approuvé pour l'émetteur par Industrie Canada. Dans le but de réduire les risques de brouillage radioélectrique à l'intention des autres utilisateurs, il faut choisir le type d'antenne et son gain de sorte que la puissance isotrope rayonnée équivalente (p.i.r.e.) ne dépasse pas l'intensité nécessaire à l'établissement d'une communication satisfaisante.

This radio transmitter (identify the device by certification number, or model number if Category II) has been approved by Industry Canada to operate with the antenna types listed below with the maximum permissible gain and required antenna impedance for each antenna type indicated. Antenna types not included in this list, having a gain greater than the maximum gain indicated for that type, are strictly prohibited for use with this device.

Conformément à la réglementation d'Industrie Canada, le présent émetteur radio peut fonctionner avec une antenne d'un type et d'un gain maximal (ou inférieur) approuvé pour l'émetteur par Industrie Canada. Dans le but de réduire les risques de brouillage radioélectrique à l'intention des autres utilisateurs, il faut choisir le type d'antenne et son gain de sorte que la puissance isotrope rayonnée équivalente (p.i.r.e.) ne dépasse pas l'intensité nécessaire à l'établissement d'une communication satisfaisante.

1.0 Device Overview [T-Series Datasheet]

This document contains device-specific information for the following devices:

- T4
- T4-OEM
- T7
- T7-Pro
- T7-OEM
- T7-Pro-OEM
- T8

This family introduces a new line of high-quality analog and Ethernet data acquisition hardware combined with the main traditional advantage of all LabJack data acquisition hardware—namely, high performance and rich feature set at a competitive price point. These features make the T-series a logical choice for many high-performance applications where Ethernet, WiFi, and cost are primary considerations.

Core Features

All T-Series Devices

- On-board Lua scripting for custom, headless operation

T4

High Voltage Analog Inputs

- 4 dedicated high voltage analog inputs (±10V, 12-bit resolution)
- Configurable resolution settings

Flexible I/O

- 8 configurable low voltage analog inputs (0-2.5V, 12-bit resolution) that can function as digital I/O lines

Digital I/O

- 8 dedicated digital I/O lines (EIO4-EIO7 and CIO0-CIO3)

Analog Outputs

- 2 Analog Outputs (10-bit, 0-5 volts)

- Additional analog outputs are possible via the LJTick-DAC

T7

Analog I/O

- 14 Analog Inputs (16-18+ Bits Depending on Speed), expand to 84 with MUX80
- Single-Ended Inputs (14) or Differential Inputs (7)
- Instrumentation Amplifier Inputs
- Software Programmable Gains of x1, x10, x100, and x1000
- Analog Input Ranges of ±10, ±1, ±0.1, and ±0.01 Volts
- 2 Analog Outputs (12-Bit, ~0-5 Volts)

Digital I/O

- 23 Digital I/O
- Supports up to 10 counters
- Supports SPI, I2C, 1-Wire and Asynchronous Serial Protocols (Master Only)
- Supports Software or Hardware Timed Acquisition
- Maximum Input Stream Rate of 100 kHz (Depending on Resolution)
- Capable of Command/Response Times Less Than 1 millisecond

Digital I/O Extended Features

- Simple PWM Output (1-32 bit)
- PWM Output w/ phase control
- Pulse Output w/ phase control
- Positive edge capture
- Negative edge capture
- PWM measure
- Edge capture & compare
- High speed counter
- Software counter
- Software counter w/ debounce
- Quadrature Input
- Easy Frequency Input

Analog Input Extended Features

- User Defined Slope & Offset
- Thermocouple type E, J, K, R, T, and C calculations
- RTDs
- Thermistors

Other highlights

- Built-In CJC Temperature Sensor
- Watchdog system
- Field Upgradable Firmware

- Programmable Startup Defaults
- LJTick Compatible

Fixed Current Outputs

- 200 µA
- 10 µA

T8

Analog I/O

- 8 Analog Inputs (21 bits at 100 Hz)
- Simultaneous Sampling
- Isolated - Each input in fully isolated
- Instrumentation Amplifier Inputs
- Software Programmable Gains of 0.5, 1, 2, 4, 8, 16, 32, 64, 128, 256, 512
- 11 Analog Input Ranges from ±11 Volts, to ±0.019 Volts
- 2 Analog Outputs (16-Bit, 0-10 Volts, up to 20 mA)

Digital I/O

- 20 Digital I/O
- Configurable pull-ups and pull-downs
- Supports up to 7 hardware, or 16 software counters
- Supports SPI, I2C, 1-Wire and Asynchronous Serial Protocols (Master Only)
- Supports Software or Hardware Timed Acquisition
- Maximum Input Stream Rate of 50 kHz (Depending on Resolution)
- Capable of Command/Response Times Less Than 150 microsecond

Digital I/O Extended Features

- Simple PWM Output (1-32 bit)
- PWM Output w/ phase control
- Pulse Output w/ phase control
- Positive edge capture
- Negative edge capture
- PWM measure
- Edge capture & compare
- High speed counter
- Software counter
- Software counter w/ debounce
- Quadrature Input
- Easy Frequency Input
- 100 MHz Clock Source

Analog Input Extended Features

- User Defined Slope & Offset
- Thermocouple type E, J, K, R, T, and C calculations
- RTDs
- Thermistors

Other Highlights

- Built-In CJC Temperature Sensors
- Watchdog system
- Field Upgradable Firmware
- Programmable Startup Defaults
- LJTick Compatible

Fixed Reference Output

- 3.3 V

Device Variants

T-Series devices are built in several configurations:

- **Standard** – The standard device configuration. Comes in a red case with screw terminals and DB connectors installed.
- **OEM** – The device PCB without USB, screw-terminals, or the DB connectors installed. Intended for customers that integrate LabJack devices into their own products.
- **Pro** (not available for all devices) – The Pro option adds hardware upgrades such as higher resolution (ADC) and WiFi. This option can be stacked with any of the Standard or OEM configurations.

For more information about the OEM variants, see Section 22: OEM Versions

Family Variants

All T-series Devices

All T-Series devices use Modbus TCP and are compatible with the LJM Library. Programs designed for one T-Series device will often work on other T-Series devices with minimal changes.

T4 vs T7

T4 characteristics that differ from the T7:

- 12-bit effective resolution on HV and LV lines.
- AIN (0-3) are high voltage (±10V).
- Flexible I/O lines: The T4 I/O lines FIO (4-7) and EIO (0-3) are software-configurable to be either low voltage analog inputs (0-2.5V) or digital I/O lines (3.3V logic level).

T7 vs T7-Pro

The T7-Pro has all features of the normal T7, with the following added:

- Wireless Ethernet 802.11b/g.
- 24-bit low-speed sigma-delta ADC.
- Battery-backed real time clock for stand-alone data logging.
- Factory installed microSD card for stand-alone data logging.

Also see the block diagram in the hardware overview section.

T7-OEM and T7-Pro-OEM

There are also OEM versions of the T7 and T7-Pro. The OEM versions are the same in terms of features, but the enclosure and most connectors are not installed on the OEM versions, allowing customization as needed. See 22.0 OEM Versions for details.

T8

The T8 builds upon the success of the T4 and T7 devices and adds unique features previously unavailable to LabJack customers.

- 8 Isolated Analog Inputs (1000 Vrms channel to channel and channel to ground isolation
- Simultaneous sampling for added timing precision
- 24-bit ΣΔ ADC (up to 40k samples/s/ch) previous LabJack devices with high resolution converters were low speed sampling
- 10+ different Voltage Ranges: ±11V, ±9.6V, ±4.8V, ±2.4V,±1.2V, ±0.6V, ±0.3V, ±0.15V,±0.75V, ±0.36V, and ±0.18V

2.0 Installation [T-Series Datasheet]

LabJack provides simple installation packages which will load the drivers and software necessary to get rolling with a T-series device. Follow the installation instructions in our quickstart guides:

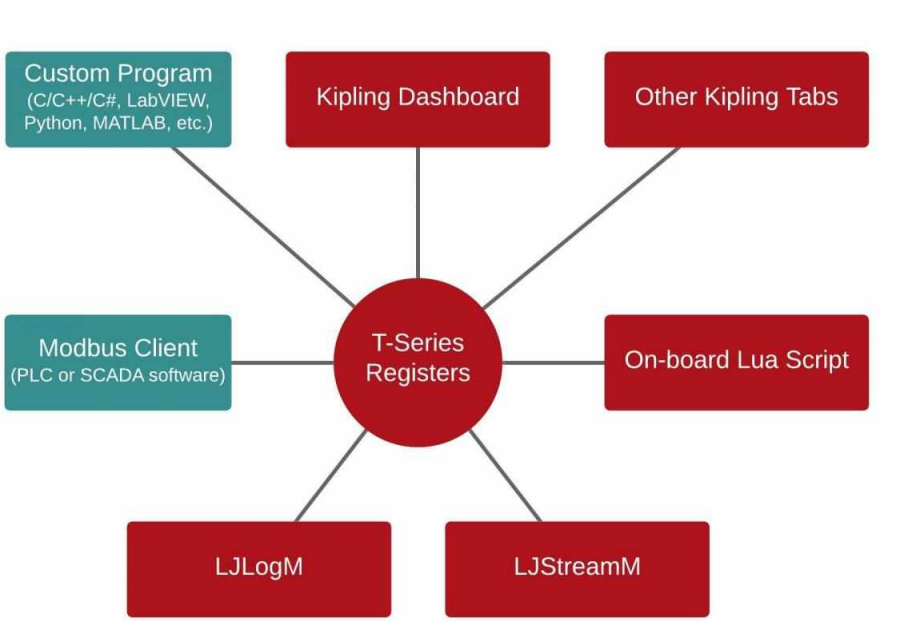
- T4 quickstart tutorial
- T7 quickstart tutorial
- T8 quickstart tutorial

3.0 Communication [T-Series Datasheet]

Overview

T-Series devices are Modbus-TCP servers. Modbus-TCP is the protocol that describes how to interpret the data stream. A server will wait for commands to be sent to it from a client. A computer acts as the client when making requests of the server.

Modbus can be accessed by various clients. In most cases the high-level LJM library and acompanying software should be used to control T-series devices.



T-series devices provide a flexible interface. Modbus registers can be accessed by various clients.

Legend:
•Rectangles are clients
•Circles are servers
•Red boxes are LabJack solutions
•Green boxes are third party options

Subsections

3.1 Modbus Map

3.2 Stream Mode

Communication Options

There are two methods for data transfer to occur:

- Command-response - offers the lowest latency.
- Stream Mode - offers the highest data throughput.

Command-Response

This is the default mode for communication with LabJack devices. It is the simplest communication mode. Communication is initiated by a command from the client which is followed by a response from the device.

- Overall data transfer is paced by the client software.
- Command-response is generally used at 1000 scans/second or slower.
- Command-response mode is generally best for minimum-latency[1] applications such as feedback control.
- See Appendix A-1 for details on command-response data rates.

[1] Latency, in this context, means the time from when a reading is acquired to when it is available in the client software.

Stream Mode

Stream mode is generally best for maximum-throughput applications. Data can be acquired very fast, but to sustain the fast rates it must be buffered and moved from the device to the client in large chunks. The buffers introduce some latency to the data delivery. Because of that latency, streaming is usually not recommended for feedback control operations.

- Data transfer is paced by hardware on the LabJack device.
- Runs at a set interval designated by a configurable scan rate.
- A scan consists of a sample from each channel in the stream scan list.
- Stream samples are placed in a buffer on the LabJack.
- When sufficient samples have been collected on the device, a data packet will be sent to the client.
- Primarily used when command-response is not capable of the desired sampling rate.
- See Appendix-A-1 for details on stream mode data rates.

> For the T7-Pro, stream mode is not supported on the hi-res converter (resolution indices 9-12 are not supported in stream).

Combining Command-response and Stream mode acquisition

Command-response operations can be performed while a stream is active.

If any of the items in the stream scan list is an analog channel, stream will take exclusive control of the analog input system and analog inputs cannot be read via command-response.

If the scan list does not contain any analog inputs, it is what we refer to as a digital only stream. Some T-series devices (T4,T7) will allow analog readings through command-response while performing digital only stream.

> The T8 does not support digital only stream.

Hardware/Software Interface

T-series devices communicate via Modbus TCP and act as Modbus TCP servers. A device must act as the client, using Modbus TCP to interface with the device hardware. This hardware/software interface can be achieved using Modbus TCP client software or by using the high level LJM library. These interface options are described in greater detail below. Further software information can be found in the Software Options appendix.

Modbus Protocol

Modbus TCP is a register based protocol. Registers store values which can be read from or written to. Each register represents a configuration, measurement, or variable. Most T-series device registers are both readable and writable.

All of the Modbus registers that are supported by T-series devices are contained in a list called the Modbus Map. The map allows users to quickly look up any available operations.

The word “register” is often used with two different meanings. The Modbus protocol defines a register as a single 16-bit value which resides at a single address within the Modbus map. A register can also be defined as a logical value. The logical value may require larger data-types such as a 32-bit floating point number. If the data-type associated with a logical value is larger than 16-bits, then multiple Modbus TCP registers will need to be read sequentially. For more information, see the Modbus Map section.

High-level LJM library

LJM is a cross-platform library. It allows users to access device registers by name. For example, "AIN4" for analog input 4. Example code is available for over a dozen different programming languages.

The LJM library makes it easy to integrate T-series devices into a variety of existing software frameworks. The LJM library is the recommended programming interface for T-series devices.

Client Software Options

Prebuilt Programs from LabJack

Prebuilt programs are useful for simple applications such as testing, configuring, and logging data to files. LabJack provides a few prebuilt applications:

- Kipling provides a simple interface for testing and configuring most t-series features.
- LJLogM uses software timing to periodically sample data and save results to a file.

- LJStreamM uses hardware timing to sample data and save results to a file at higher rates. Generally, sampling at more than 500 Hz to 1k Hz requires streaming.

Custom LJM Programs

Custom programs allow all of the functionally of T-series devices to be controlled from many different programming languages. Custom programs are used when an application requires output control, response to stimuli, feedback systems, etc.

Example workflow:

1. Open a connection to the T-series device.
2. Read from and write to Modbus registers.
3. Close the connection.

Prebuilt Programs from third Parties

Programs known as COTS or “Commercial Off The Shelf” can often communicate via Modbus TCP with T-series devices. Some T-series features, such as stream and multiple operations per packet, will not be available when using a COTS program.

Custom Modbus TCP Programs

A custom program could be designed to interface with T-series devices via Modbus TCP, however the high level LJM library should be used in most cases. LJM abstracts away most Modbus TCP details to simplify software development.

Onboard Lua Scripts

Onboard Lua scripts are small programs that run on the t-series device's processor. Scripts have access all the features of the device and do not have to wait for communication delays.

Onboard scripts allow T-Series devices to perform tasks without commands from a client program and are one of the most useful features of T-Series devices. Scripts have several advantages such as lower response times, autonomous operation, simplifying some tasks such as reading from sensors, PID loops, etc.

Script application examples:

- Sensor Reading: A common use for scripts is to read a digital or analog sensor, apply a calibration curve and save the resulting value so that the client application can read that value at its convenience. This remove the complexity of communication protocols, or device specific analog settings from the host application.
- PID: A pid controller can be implemented in a script that will constantly update an output in response to an input. The client application can read the most recent output and input values for P, I, and D as well as start and stop commands.

A device running an onboard script needs to process the Lua byte code. That requires some

processing power, which results in lower maximum data collection rates when running a script.

3.1 Modbus Map [T-Series Datasheet]

Modbus Map Tool

Device:

All Devices

Tags:

All Tags
AIN
AIN_EF
ASYNCH
CONFIG
CORE
DAC
DIO
DIO_EF
ETHERNET

Expand addresses:

Show

10

entries

Search:

name	address	type	access	tags	details
AIN#(0:254)	0	FLOAT32	R	AIN, CORE	
TEMPERATURE#(0:7)	600	FLOAT32	R	AIN, CORE	
AIN#(0:7)_CAPTURE	650	FLOAT32	R	AIN, CORE	
TEMPERATURE#(0:7)_CAPTURE	700	FLOAT32	R	AIN, CORE	
DAC#(0:1)	1000	FLOAT32	R / W	DAC, CORE	
CURRENT_SOURCE_10UA_CAL_VALUE	1900	FLOAT32	R	CONFIG	
CURRENT_SOURCE_200UA_CAL_VALUE	1902	FLOAT32	R	CONFIG	
FIO#(0:7) (also known as: DIO#(0:7))	2000	UINT16	R / W	DIO, CORE	
EIO#(0:7) (also known as: DIO#(8:15))	2008	UINT16	R / W	DIO, CORE	
CIO#(0:3) (also known as: DIO#(16:19))	2016	UINT16	R / W	DIO, CORE	

Showing 1 to 10 of 406 entries

First Previous 1 2 3 4 5 Next Last

The filter and search tool above displays information about the Modbus registers of T-series devices.

- Name: The string name that can be used with the LJM library to access each register.
- Address: The starting address of each register, which can be used through LJM or with direct Modbus.
- Details: A quick description of the register.

- Type: Specifies the datatype, which specifies how many registers each value uses.
- Access: Each register is readable, writable, or both.
- Tags: Used to associate registers with particular functionality. Useful for filtering.

For the U3, U6 and UE9, see the deprecated Modbus system called UD Modbus.

For a printer-friendly version, see the printable Modbus map.

Usage

T-series devices are controlled by reading or writing Modbus registers as described on the Communication page.

Protocol

Modbus TCP is described further on the Protocol Details page.

Address Mapping

T-series devices have a single map of addresses from 0 to 65535. Any type of register can be located anywhere in that address range, regardless of the data type or whether it is read-only, write-only, or read-write.

Some addresses point to buffers that can access more data than what would normally fit within the Modbus address range. See Buffer Registers.

Sequential Addresses

Many registers are sequentially addressed. The Modbus Map gives you the starting address for the first register, and then—depending on whether the data type is 16-bits or 32-bits—you increment the address by 1 or 2 to get the next value:

`Address = StartingAddress + 1*Channel#` (UINT16)

`Address = StartingAddress + 2*Channel#` (UINT32, INT32, FLOAT32)

Note that the term "register" is used 2 different ways throughout documentation:

- A "register" is a location that has a value you might want to read or write (e.g. AIN0 or DAC0).

- The term "Modbus register" generally refers to the Modbus use of the term, which is a 16-bit value pointed to by an address of 0-65535.

Therefore, most "registers" consist of 1 or 2 "Modbus registers".

For example, the first entry (starting address of 0) in the Modbus Map has the name AIN# (0:254), which is shorthand notation for 255 registers named AIN0, AIN1, AIN2, ..., AIN254. The AIN# data type is FLOAT32, so each value needs 2 Modbus registers, thus the address for a given analog input is `channel*2` .

Zero-Based Addressing

Zero-Based Addressing means that the addresses defined in the Modbus Map are the addresses that should be used in the actual Modbus TCP packets.

If you want to read an address documented to be at address 2000, then 2000 should be the address in in the Modbus TCP packet. Some Modbus TCP client software subtracts 1 from all addresses. For example, you tell the client software you want to read address 2000, but it puts 1999 in the actual Modbus TCP packet. That means if you want to read Modbus address 2000 you have to pass 2001 to the client software.

Some Modbus TCP client programs use addresses written as 4xxxx, which can refer to the "Holding Register" address block in some Modbus addressing schemes. However, this addressing scheme is not defined as a part of the Modbus specification and LabJack devices do not use this form for addressing. If 4xxxx addressing is encountered in a client program, the xxxx will often refer to the address used in the Modbus packet, or it might also have 1 subtracted before being put into the Modbus packet. For example, address 2000 on the LabJack may be accessible by passing 42000 or 42001 to the client software.

Big-Endian

Modbus is specified as big-endian, which means the most significant value is at the lowest address. With a read of a 16-bit (single register) value, the 1st byte returned is the MSB (most significant byte) and the 2nd byte returned is the LSB (least significant byte). With a read of a 32-bit (2 register) value, the value is returned MSW then LSW, where each word is returned MSB then LSB, so the 4 bytes come in order from most significant to least significant.

Some Modbus TCP clients expect Modbus to be implemented with big-endian bytes but with the least significant word before the most significant word. In other words, the client software flips the order of the words within a 32-bit value. For example, a read of TEST (address 55100) should return 0x00112233, but the client returns 0x22330011. Most client programs provide options to change the data interpretation.

Data Type Constants

Each entry in the map has an associated data type. The data type tells us know to format data being sent to the device and how to interpret data from the device. The raw data in the map contains the integer values below. The Map above displays the type name instead of the integer value.

Type	Integer Value
UINT16	0
UINT32	1
INT32	2
FLOAT32	3
BYTE	99
STRING	98

ljm_constants.json

LabJack distributes a constants file called `ljm_constants.json` that defines information about the Modbus register map. The filter and search tool above pulls data from that JSON file.

The ljm_constants GitHub repository contains up-to-date text versions of the Modbus register map:

- **JSON** - ljm_constants.json
- **C/C++ header** - LabJackMModbusMap.h

Most device registers are written and/or read by address. Buffer registers are special registers that are used when multiple values must be written or read, but the number of values are able to change. Buffer Registers produce multiple values when being read and consume all values being written, and they allow users to write a sequence of values to a single Modbus address. Buffer registers typically have a companion "_SIZE" register that defines how many sequential values should be sent to or read from a buffer register.

Buffer Registers

Buffer registers could also be called array registers; the array size is defined using a "_SIZE" register, then the array data is accessed or modified using a single specific (Modbus) address.

For example, consider the difference between AIN0 and FILE_IO_PATH_READ:

Normal register:

- AIN0 is at address 0 and is followed by AIN1 at address 2
- AIN0 is a normal register
- Reading an array of 4 registers starting at address 0 would read 2 registers from AIN0 and 2 registers AIN1 (AIN values are FLOAT32, which each consist of 2 registers)

Buffer register:

- FILE_IO_PATH_READ is at address 60652 and is followed by FILE_IO_WRITE at address 60654
- FILE_IO_PATH_READ is a Buffer Register
- Reading an array of 4 registers starting at address 60652 would read 4 registers from FILE_IO_PATH_READ. FILE_IO_WRITE would not be read.
- Note that users would first designate that 8 bytes are about to be read by writing a value of 8 to FILE_IO_PATH_READ_LEN_BYTES.

In practice, the important differences are:

1. You don't need to know what registers follow a Buffer Register, you can simply write / read without worrying about colliding with other registers
2. You can only write / read values sequentially. E.g. you cannot modify previously written values.
3. Define how much data to send/receive to/from the buffer register using the associated _NUM_BYTES, or _SIZE, or _LEN register.
4. Often it is necessary to complete the transaction with an action register, such as _GO, or _OPEN, or _ENABLE.

Buffer Registers, and their size definitions:

Serial Comm Systems

- ASYNCH_DATA_RX
- ASYNCH_DATA_TX
- ASYNCH_NUM_BYTES_RX
- ASYNCH_NUM_BYTES_TX
- I2C_DATA_RX
- I2C_DATA_TX
- I2C_NUM_BYTES_RX
- I2C_NUM_BYTES_TX
- ONEWIRE_DATA_RX
- ONEWIRE_DATA_TX
- ONEWIRE_NUM_BYTES_RX
- ONEWIRE_NUM_BYTES_TX
- SPI_DATA_RX
- SPI_DATA_TX
- SPI_NUM_BYTES

File IO System

- FILE_IO_PATH_READ
- FILE_IO_PATH_WRITE

- FILE_IO_PATH_READ_LEN_BYTES
- FILE_IO_PATH_WRITE_LEN_BYTES
- FILE_IO_READ
- FILE_IO_WRITE
- FILE_IO_SIZE_BYTES

Lua Scripts/Debug Info

- LUA_SOURCE_WRITE
- LUA_SOURCE_SIZE
- LUA_DEBUG_DATA
- LUA_DEBUG_NUM_BYTES

Stream Out System

- STREAM_OUT#(0:3)_BUFFER_F32
- STREAM_OUT#(0:3)_BUFFER_U16
- STREAM_OUT#(0:3)_BUFFER_U32
- STREAM_OUT#(0:3)_BUFFER_ALLOCATE_NUM_BYTES

User RAM FIFOs

- USER_RAM_FIFO#(0:3)_DATA_F32
- USER_RAM_FIFO#(0:3)_DATA_I32
- USER_RAM_FIFO#(0:3)_DATA_U16
- USER_RAM_FIFO#(0:3)_DATA_U32
- USER_RAM_FIFO#(0:3)_ALLOCATE_NUM_BYTES

WIFI

- WIFI_SCAN_DATA
- WIFI_SCAN_NUM_BYTES

Internal Flash

- INTERNAL_FLASH_READ
- INTERNAL_FLASH_WRITE

Buffer registers can be identified using the Modbus Map tool. Expand the details button to see whether a register is a buffer register, and other details.

LJM

When using the LJM library, the ByteArray or Array functions should be used to read or write buffer buffer registers:

- Non-BYTE

registers: LJM_eReadAddressArray, LJM_eReadNameArray, LJM_eWriteAddressArray, or LJM
- BYTE-type registers: LJM_eReadAddressByteArray, LJM_eReadNameByteArray, LJM_eWriteAddressByteArray, or LJM_eWriteNameByteArray

Reading buffer registers with the wrong LJM function can result in error LJME_FUNCTION_DOES_NOT_SUPPORT_THIS_TYPE.

3.1.2 Printable Modbus Map

This page displays common registers for T-series LabJacks. Please allow a few moments for all registers to load. Other registers can be found:

- On the Modbus Map tool
- In the T-series Datasheet

To print the page with formatting preserved, navigate to the bottom of the page and click "Print this section" near the bottom right corner. Once the new printable page loads, use your browser print feature to print the page (control+p is a common shortcut). Using the "Save as PDF" button is not recommended, it will not maintain all of the page formatting.

All AIN TAGS:

Name	Start Address	Type	Access
AIN#(0:253)	0	FLOAT32	R
TEMPERATURE#(0:6)	600	FLOAT32	R
AIN#(0:6)_CAPTURE	650	FLOAT32	R
TEMPERATURE#(0:6)_CAPTURE	700	FLOAT32	R
AIN#(0:148)_EF_READ_A	7000	FLOAT32	R
AIN#(0:148)_EF_READ_B	7300	FLOAT32	R/W
AIN#(0:148)_EF_READ_C	7600	FLOAT32	R/W
AIN#(0:148)_EF_READ_D	7900	FLOAT32	R
AIN#(0:6)_EF_READ_A_CAPTURE	8800	FLOAT32	R
AIN#(0:148)_EF_INDEX	9000	UINT32	R/W
AIN#(0:148)_EF_CONFIG_A	9300	UINT32	R/W
AIN#(0:148)_EF_CONFIG_B	9600	UINT32	R/W
AIN#(0:148)_EF_CONFIG_C	9900	UINT32	R/W
AIN#(0:148)_EF_CONFIG_D	10200	FLOAT32	R/W
AIN#(0:148)_EF_CONFIG_E	10500	FLOAT32	R/W
AIN#(0:148)_EF_CONFIG_F	10800	FLOAT32	R/W
AIN#(0:148)_EF_CONFIG_G	11100	FLOAT32	R/W
AIN#(0:148)_EF_CONFIG_H	11400	FLOAT32	R/W
AIN#(0:148)_EF_CONFIG_I	11700	FLOAT32	R/W
AIN#(0:148)_EF_CONFIG_J	12000	FLOAT32	R/W
AIN#(0:253)_RANGE	40000	FLOAT32	R/W
AIN#(0:253)_NEGATIVE_CH	41000	UINT16	R/W
AIN#(0:253)_RESOLUTION_INDEX	41500	UINT16	R/W
AIN#(0:253)_SETTLING_US	42000	FLOAT32	R/W
AIN_CHANNEL_ENABLE	43700	UINT32	R/W
AIN_SAMPLING_RATE_HZ	43710	FLOAT32	R/W
AIN_SAMPLING_RATE_ACTUAL_HZ	43712	FLOAT32	R
AIN_ALL_RANGE	43900	FLOAT32	R/W
AIN_ALL_NEGATIVE_CH	43902	UINT16	R/W
AIN_ALL_RESOLUTION_INDEX	43903	UINT16	R/W
AIN_ALL_SETTLING_US	43904	FLOAT32	R/W
AIN_ALL_EF_INDEX	43906	UINT32	R/W
POWER_AIN	48005	UINT16	R/W
POWER_AIN_DEFAULT	48055	UINT16	R/W
TEMPERATURE_AIR_K	60050	FLOAT32	R

TEMPERATURE_DEVICE_K	60052	FLOAT32	R
CALIBRATION_CONSTANTS_STATUS	60091	FLOAT32	R
CALIBRATION_SOURCE	60092	UINT16	R/W
AIN#(0:253)_BINARY	50000	UINT32	R
TEMPERATURE#(0:6)_BINARY	50600	UINT32	R
AIN#(0:6)_BINARY_CAPTURE	50650	UINT32	R
TEMPERATURE#(0:6)_BINARY_CAPTURE	50700	UINT32	R

AIN#(0:253)
- Starting Address: 0

Returns the voltage of the specified analog input.

- Data type: FLOAT32 (type index = 3)
- Read-only
- This register may be streamed

Expanded Names	Addresses
AIN0, AIN1, AIN2, AIN3, AIN4, AIN5, AIN6, AIN7, AIN8, AIN9, AIN10, AIN11, AIN12, AIN13, AIN14, AIN15, AIN16, AIN17, AIN18, AIN19, AIN20, AIN21, AIN22, AIN23, AIN24, AIN25, AIN26, AIN27, AIN28, AIN29, AIN30, AIN31, AIN32, AIN33, AIN34, AIN35, AIN36, AIN37, AIN38, AIN39, AIN40, AIN41, AIN42, AIN43, AIN44, AIN45, AIN46, AIN47, AIN48, AIN49, AIN50, AIN51, AIN52, AIN53, AIN54, AIN55, AIN56, AIN57, AIN58, AIN59, AIN60, AIN61, AIN62, AIN63, AIN64, AIN65, AIN66, AIN67, AIN68, AIN69, AIN70, AIN71, AIN72, AIN73, AIN74, AIN75, AIN76, AIN77, AIN78, AIN79, AIN80, AIN81, AIN82, AIN83, AIN84, AIN85, AIN86, AIN87, AIN88, AIN89, AIN90, AIN91, AIN92, AIN93, AIN94, AIN95, AIN96, AIN97, AIN98, AIN99, AIN100, AIN101, AIN102, AIN103, AIN104, AIN105, AIN106, AIN107, AIN108, AIN109, AIN110, AIN111, AIN112, AIN113, AIN114, AIN115, AIN116, AIN117, AIN118, AIN119, AIN120, AIN121, AIN122, AIN123, AIN124, AIN125, AIN126, AIN127, AIN128, AIN129, AIN130, AIN131, AIN132, AIN133, AIN134, AIN135, AIN136, AIN137, AIN138, AIN139, AIN140, AIN141, AIN142, AIN143, AIN144, AIN145, AIN146, AIN147, AIN148, AIN149, AIN150, AIN151, AIN152, AIN153, AIN154, AIN155, AIN156, AIN157, AIN158, AIN159, AIN160, AIN161, AIN162, AIN163, AIN164, AIN165, AIN166, AIN167, AIN168, AIN169, AIN170, AIN171, AIN172, AIN173, AIN174, AIN175, AIN176, AIN177, AIN178, AIN179, AIN180, AIN181, AIN182, AIN183, AIN184, AIN185, AIN186, AIN187, AIN188, AIN189, AIN190, AIN191, AIN192, AIN193, AIN194, AIN195, AIN196, AIN197, AIN198, AIN199, AIN200, AIN201, AIN202, AIN203, AIN204, AIN205, AIN206, AIN207, AIN208, AIN209, AIN210, AIN211, AIN212, AIN213, AIN214, AIN215, AIN216, AIN217, AIN218, AIN219, AIN220, AIN221, AIN222, AIN223, AIN224, AIN225, AIN226, AIN227, AIN228, AIN229, AIN230, AIN231, AIN232, AIN233, AIN234, AIN235, AIN236, AIN237, AIN238, AIN239, AIN240, AIN241, AIN242, AIN243, AIN244, AIN245, AIN246, AIN247, AIN248, AIN249, AIN250, AIN251, AIN252, AIN253	0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42, 44, 46, 48, 50, 52, 54, 56, 58, 60, 62, 64, 66, 68, 70, 72, 74, 76, 78, 80, 82, 84, 86, 88, 90, 92, 94, 96, 98, 100, 102, 104, 106, 108, 110, 112, 114, 116, 118, 120, 122, 124, 126, 128, 130, 132, 134, 136, 138, 140, 142, 144, 146, 148, 150, 152, 154, 156, 158, 160, 162, 164, 166, 168, 170, 172, 174, 176, 178, 180, 182, 184, 186, 188, 190, 192, 194, 196, 198, 200, 202, 204, 206, 208, 210, 212, 214, 216, 218, 220, 222, 224, 226, 228, 230, 232, 234, 236, 238, 240, 242, 244, 246, 248, 250, 252, 254, 256, 258, 260, 262, 264, 266, 268, 270, 272, 274, 276, 278, 280, 282, 284, 286, 288, 290, 292, 294, 296, 298, 300, 302, 304, 306, 308, 310, 312, 314, 316, 318, 320, 322, 324, 326, 328, 330, 332, 334, 336, 338, 340, 342, 344, 346, 348, 350, 352, 354, 356, 358, 360, 362, 364, 366, 368, 370, 372, 374, 376, 378, 380, 382, 384, 386, 388, 390, 392, 394, 396, 398, 400, 402, 404, 406, 408, 410, 412, 414, 416, 418, 420, 422, 424, 426, 428, 430, 432, 434, 436, 438, 440, 442, 444, 446, 448, 450, 452, 454, 456, 458, 460, 462, 464, 466, 468, 470, 472, 474, 476, 478, 480, 482, 484, 486, 488, 490, 492, 494, 496, 498, 500, 502, 504, 506

TEMPERATURE#(0:6)
- Starting Address: 600

T8 Only. Returns the temperature of the specified analog input. And saves AIN and temperature inputs for all channels.

- Data type: FLOAT32 (type index = 3)

- Read-only
- This register may be streamed

Expanded Names	Addresses
TEMPERATURE0, TEMPERATURE1, TEMPERATURE2, TEMPERATURE3, TEMPERATURE4, TEMPERATURE5, TEMPERATURE6	600, 602, 604, 606, 608, 610, 612

AIN#(0:6)_CAPTURE
- Starting Address: 650

T8 Only. Returns the saved voltage of the specified analog input.

- Data type: FLOAT32 (type index = 3)
- Read-only
- This register may be streamed

Expanded Names	Addresses
AIN0_CAPTURE, AIN1_CAPTURE, AIN2_CAPTURE, AIN3_CAPTURE, AIN4_CAPTURE, AIN5_CAPTURE, AIN6_CAPTURE	650, 652, 654, 656, 658, 660, 662

TEMPERATURE#(0:6)_CAPTURE
- Starting Address: 700

T8 Only. Returns the saved temperature of the specified analog input.

- Data type: FLOAT32 (type index = 3)
- Read-only
- This register may be streamed

Expanded Names	Addresses
TEMPERATURE0_CAPTURE, TEMPERATURE1_CAPTURE, TEMPERATURE2_CAPTURE, TEMPERATURE3_CAPTURE, TEMPERATURE4_CAPTURE, TEMPERATURE5_CAPTURE, TEMPERATURE6_CAPTURE	700, 702, 704, 706, 708, 710, 712

AIN#(0:148)_EF_READ_A
- Starting Address: 7000

Function dependent on selected feature index.

- Data type: FLOAT32 (type index = 3)
- Read-only
- Default value: 0
- This register uses system RAM. The maximum RAM is 64KB. For more information, see 4.4 RAM
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0030

Expanded Names	Addresses

AIN0_EF_READ_A, AIN1_EF_READ_A, AIN2_EF_READ_A, AIN3_EF_READ_A, AIN4_EF_READ_A, AIN5_EF_READ_A, AIN6_EF_READ_A, AIN7_EF_READ_A, AIN8_EF_READ_A, AIN9_EF_READ_A, AIN10_EF_READ_A, AIN11_EF_READ_A,	7000, 7002, 7004, 7006,
AIN12_EF_READ_A, AIN13_EF_READ_A, AIN14_EF_READ_A, AIN15_EF_READ_A,	7008, 7010, 7012, 7014,
AIN16_EF_READ_A, AIN17_EF_READ_A, AIN18_EF_READ_A, AIN19_EF_READ_A,	7016, 7018, 7020, 7022,
AIN20_EF_READ_A, AIN21_EF_READ_A, AIN22_EF_READ_A, AIN23_EF_READ_A,	7024, 7026, 7028, 7030,
AIN24_EF_READ_A, AIN25_EF_READ_A, AIN26_EF_READ_A, AIN27_EF_READ_A,	7032, 7034, 7036, 7038,
AIN28_EF_READ_A, AIN29_EF_READ_A, AIN30_EF_READ_A, AIN31_EF_READ_A,	7040, 7042, 7044, 7046,
AIN32_EF_READ_A, AIN33_EF_READ_A, AIN34_EF_READ_A, AIN35_EF_READ_A,	7048, 7050, 7052, 7054,
AIN36_EF_READ_A, AIN37_EF_READ_A, AIN38_EF_READ_A, AIN39_EF_READ_A,	7056, 7058, 7060, 7062,
AIN40_EF_READ_A, AIN41_EF_READ_A, AIN42_EF_READ_A, AIN43_EF_READ_A,	7064, 7066, 7068, 7070,
AIN44_EF_READ_A, AIN45_EF_READ_A, AIN46_EF_READ_A, AIN47_EF_READ_A,	7072, 7074, 7076, 7078,
AIN48_EF_READ_A, AIN49_EF_READ_A, AIN50_EF_READ_A, AIN51_EF_READ_A,	7080, 7082, 7084, 7086,
AIN52_EF_READ_A, AIN53_EF_READ_A, AIN54_EF_READ_A, AIN55_EF_READ_A,	7088, 7090, 7092, 7094,
AIN56_EF_READ_A, AIN57_EF_READ_A, AIN58_EF_READ_A, AIN59_EF_READ_A,	7096, 7098, 7100, 7102,
AIN60_EF_READ_A, AIN61_EF_READ_A, AIN62_EF_READ_A, AIN63_EF_READ_A,	7104, 7106, 7108, 7110,
AIN64_EF_READ_A, AIN65_EF_READ_A, AIN66_EF_READ_A, AIN67_EF_READ_A,	7112, 7114, 7116, 7118,
AIN68_EF_READ_A, AIN69_EF_READ_A, AIN70_EF_READ_A, AIN71_EF_READ_A,	7120, 7122, 7124, 7126,
AIN72_EF_READ_A, AIN73_EF_READ_A, AIN74_EF_READ_A, AIN75_EF_READ_A,	7128, 7130, 7132, 7134,
AIN76_EF_READ_A, AIN77_EF_READ_A, AIN78_EF_READ_A, AIN79_EF_READ_A,	7136, 7138, 7140, 7142,
AIN80_EF_READ_A, AIN81_EF_READ_A, AIN82_EF_READ_A, AIN83_EF_READ_A,	7144, 7146, 7148, 7150,
AIN84_EF_READ_A, AIN85_EF_READ_A, AIN86_EF_READ_A, AIN87_EF_READ_A,	7152, 7154, 7156, 7158,
AIN88_EF_READ_A, AIN89_EF_READ_A, AIN90_EF_READ_A, AIN91_EF_READ_A,	7160, 7162, 7164, 7166,
AIN92_EF_READ_A, AIN93_EF_READ_A, AIN94_EF_READ_A, AIN95_EF_READ_A,	7168, 7170, 7172, 7174,
AIN96_EF_READ_A, AIN97_EF_READ_A, AIN98_EF_READ_A, AIN99_EF_READ_A,	7176, 7178, 7180, 7182,
AIN100_EF_READ_A, AIN101_EF_READ_A, AIN102_EF_READ_A,	7184, 7186, 7188, 7190,
AIN103_EF_READ_A, AIN104_EF_READ_A, AIN105_EF_READ_A,	7192, 7194, 7196, 7198,
AIN106_EF_READ_A, AIN107_EF_READ_A, AIN108_EF_READ_A,	7200, 7202, 7204, 7206,
AIN109_EF_READ_A, AIN110_EF_READ_A, AIN111_EF_READ_A,	7208, 7210, 7212, 7214,
AIN112_EF_READ_A, AIN113_EF_READ_A, AIN114_EF_READ_A,	7216, 7218, 7220, 7222,
AIN115_EF_READ_A, AIN116_EF_READ_A, AIN117_EF_READ_A,	7224, 7226, 7228, 7230,
AIN118_EF_READ_A, AIN119_EF_READ_A, AIN120_EF_READ_A,	7232, 7234, 7236, 7238,
AIN121_EF_READ_A, AIN122_EF_READ_A, AIN123_EF_READ_A,	7240, 7242, 7244, 7246,
AIN124_EF_READ_A, AIN125_EF_READ_A, AIN126_EF_READ_A,	7248, 7250, 7252, 7254,
AIN127_EF_READ_A, AIN128_EF_READ_A, AIN129_EF_READ_A,	7256, 7258, 7260, 7262,
AIN130_EF_READ_A, AIN131_EF_READ_A, AIN132_EF_READ_A,	7264, 7266, 7268, 7270,
AIN133_EF_READ_A, AIN134_EF_READ_A, AIN135_EF_READ_A,	7272, 7274, 7276, 7278,
AIN136_EF_READ_A, AIN137_EF_READ_A, AIN138_EF_READ_A,	7280, 7282, 7284, 7286,
AIN139_EF_READ_A, AIN140_EF_READ_A, AIN141_EF_READ_A,	7288, 7290, 7292, 7294,
AIN142_EF_READ_A, AIN143_EF_READ_A, AIN144_EF_READ_A,	7296
AIN145_EF_READ_A, AIN146_EF_READ_A, AIN147_EF_READ_A,	
AIN148_EF_READ_A	

AIN#(0:148)_EF_READ_B
- Starting Address: 7300

Function dependent on selected feature index.

- Data type: FLOAT32 (type index = 3)
- Readable and writable
- Default value: 0
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0030

Expanded Names	Addresses

AIN0_EF_READ_B, AIN1_EF_READ_B, AIN2_EF_READ_B, AIN3_EF_READ_B, AIN4_EF_READ_B, AIN5_EF_READ_B, AIN6_EF_READ_B, AIN7_EF_READ_B, AIN8_EF_READ_B, AIN9_EF_READ_B, AIN10_EF_READ_B, AIN11_EF_READ_B, AIN12_EF_READ_B, AIN13_EF_READ_B, AIN14_EF_READ_B, AIN15_EF_READ_B, AIN16_EF_READ_B, AIN17_EF_READ_B, AIN18_EF_READ_B, AIN19_EF_READ_B, AIN20_EF_READ_B, AIN21_EF_READ_B, AIN22_EF_READ_B, AIN23_EF_READ_B, AIN24_EF_READ_B, AIN25_EF_READ_B, AIN26_EF_READ_B, AIN27_EF_READ_B, AIN28_EF_READ_B, AIN29_EF_READ_B, AIN30_EF_READ_B, AIN31_EF_READ_B, AIN32_EF_READ_B, AIN33_EF_READ_B, AIN34_EF_READ_B, AIN35_EF_READ_B, AIN36_EF_READ_B, AIN37_EF_READ_B, AIN38_EF_READ_B, AIN39_EF_READ_B, AIN40_EF_READ_B, AIN41_EF_READ_B, AIN42_EF_READ_B, AIN43_EF_READ_B, AIN44_EF_READ_B, AIN45_EF_READ_B, AIN46_EF_READ_B, AIN47_EF_READ_B, AIN48_EF_READ_B, AIN49_EF_READ_B, AIN50_EF_READ_B, AIN51_EF_READ_B, AIN52_EF_READ_B, AIN53_EF_READ_B, AIN54_EF_READ_B, AIN55_EF_READ_B, AIN56_EF_READ_B, AIN57_EF_READ_B, AIN58_EF_READ_B, AIN59_EF_READ_B, AIN60_EF_READ_B, AIN61_EF_READ_B, AIN62_EF_READ_B, AIN63_EF_READ_B, AIN64_EF_READ_B, AIN65_EF_READ_B, AIN66_EF_READ_B, AIN67_EF_READ_B, AIN68_EF_READ_B, AIN69_EF_READ_B, AIN70_EF_READ_B, AIN71_EF_READ_B, AIN72_EF_READ_B, AIN73_EF_READ_B, AIN74_EF_READ_B, AIN75_EF_READ_B, AIN76_EF_READ_B, AIN77_EF_READ_B, AIN78_EF_READ_B, AIN79_EF_READ_B, AIN80_EF_READ_B, AIN81_EF_READ_B, AIN82_EF_READ_B, AIN83_EF_READ_B, AIN84_EF_READ_B, AIN85_EF_READ_B, AIN86_EF_READ_B, AIN87_EF_READ_B, AIN88_EF_READ_B, AIN89_EF_READ_B, AIN90_EF_READ_B, AIN91_EF_READ_B, AIN92_EF_READ_B, AIN93_EF_READ_B, AIN94_EF_READ_B, AIN95_EF_READ_B, AIN96_EF_READ_B, AIN97_EF_READ_B, AIN98_EF_READ_B, AIN99_EF_READ_B, AIN100_EF_READ_B, AIN101_EF_READ_B, AIN102_EF_READ_B, AIN103_EF_READ_B, AIN104_EF_READ_B, AIN105_EF_READ_B, AIN106_EF_READ_B, AIN107_EF_READ_B, AIN108_EF_READ_B, AIN109_EF_READ_B, AIN110_EF_READ_B, AIN111_EF_READ_B, AIN112_EF_READ_B, AIN113_EF_READ_B, AIN114_EF_READ_B, AIN115_EF_READ_B, AIN116_EF_READ_B, AIN117_EF_READ_B, AIN118_EF_READ_B, AIN119_EF_READ_B, AIN120_EF_READ_B, AIN121_EF_READ_B, AIN122_EF_READ_B, AIN123_EF_READ_B, AIN124_EF_READ_B, AIN125_EF_READ_B, AIN126_EF_READ_B, AIN127_EF_READ_B, AIN128_EF_READ_B, AIN129_EF_READ_B, AIN130_EF_READ_B, AIN131_EF_READ_B, AIN132_EF_READ_B, AIN133_EF_READ_B, AIN134_EF_READ_B, AIN135_EF_READ_B, AIN136_EF_READ_B, AIN137_EF_READ_B, AIN138_EF_READ_B, AIN139_EF_READ_B, AIN140_EF_READ_B, AIN141_EF_READ_B, AIN142_EF_READ_B, AIN143_EF_READ_B, AIN144_EF_READ_B, AIN145_EF_READ_B, AIN146_EF_READ_B, AIN147_EF_READ_B, AIN148_EF_READ_B	7300, 7302, 7304, 7306, 7308, 7310, 7312, 7314, 7316, 7318, 7320, 7322, 7324, 7326, 7328, 7330, 7332, 7334, 7336, 7338, 7340, 7342, 7344, 7346, 7348, 7350, 7352, 7354, 7356, 7358, 7360, 7362, 7364, 7366, 7368, 7370, 7372, 7374, 7376, 7378, 7380, 7382, 7384, 7386, 7388, 7390, 7392, 7394, 7396, 7398, 7400, 7402, 7404, 7406, 7408, 7410, 7412, 7414, 7416, 7418, 7420, 7422, 7424, 7426, 7428, 7430, 7432, 7434, 7436, 7438, 7440, 7442, 7444, 7446, 7448, 7450, 7452, 7454, 7456, 7458, 7460, 7462, 7464, 7466, 7468, 7470, 7472, 7474, 7476, 7478, 7480, 7482, 7484, 7486, 7488, 7490, 7492, 7494, 7496, 7498, 7500, 7502, 7504, 7506, 7508, 7510, 7512, 7514, 7516, 7518, 7520, 7522, 7524, 7526, 7528, 7530, 7532, 7534, 7536, 7538, 7540, 7542, 7544, 7546, 7548, 7550, 7552, 7554, 7556, 7558, 7560, 7562, 7564, 7566, 7568, 7570, 7572, 7574, 7576, 7578, 7580, 7582, 7584, 7586, 7588, 7590, 7592, 7594, 7596

AIN#(0:148)_EF_READ_C
- Starting Address: 7600

Function dependent on selected feature index.

- Data type: FLOAT32 (type index = 3)
- Readable and writable
- Default value: 0
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0030

Expanded Names	Addresses

AIN0_EF_READ_C, AIN1_EF_READ_C, AIN2_EF_READ_C, AIN3_EF_READ_C, AIN4_EF_READ_C, AIN5_EF_READ_C, AIN6_EF_READ_C, AIN7_EF_READ_C, AIN8_EF_READ_C, AIN9_EF_READ_C, AIN10_EF_READ_C, AIN11_EF_READ_C, AIN12_EF_READ_C, AIN13_EF_READ_C, AIN14_EF_READ_C, AIN15_EF_READ_C, AIN16_EF_READ_C, AIN17_EF_READ_C, AIN18_EF_READ_C, AIN19_EF_READ_C, AIN20_EF_READ_C, AIN21_EF_READ_C, AIN22_EF_READ_C, AIN23_EF_READ_C, AIN24_EF_READ_C, AIN25_EF_READ_C, AIN26_EF_READ_C, AIN27_EF_READ_C, AIN28_EF_READ_C, AIN29_EF_READ_C, AIN30_EF_READ_C, AIN31_EF_READ_C, AIN32_EF_READ_C, AIN33_EF_READ_C, AIN34_EF_READ_C, AIN35_EF_READ_C, AIN36_EF_READ_C, AIN37_EF_READ_C, AIN38_EF_READ_C, AIN39_EF_READ_C, AIN40_EF_READ_C, AIN41_EF_READ_C, AIN42_EF_READ_C, AIN43_EF_READ_C, AIN44_EF_READ_C, AIN45_EF_READ_C, AIN46_EF_READ_C, AIN47_EF_READ_C, AIN48_EF_READ_C, AIN49_EF_READ_C, AIN50_EF_READ_C, AIN51_EF_READ_C, AIN52_EF_READ_C, AIN53_EF_READ_C, AIN54_EF_READ_C, AIN55_EF_READ_C, AIN56_EF_READ_C, AIN57_EF_READ_C, AIN58_EF_READ_C, AIN59_EF_READ_C, AIN60_EF_READ_C, AIN61_EF_READ_C, AIN62_EF_READ_C, AIN63_EF_READ_C, AIN64_EF_READ_C, AIN65_EF_READ_C, AIN66_EF_READ_C, AIN67_EF_READ_C, AIN68_EF_READ_C, AIN69_EF_READ_C, AIN70_EF_READ_C, AIN71_EF_READ_C, AIN72_EF_READ_C, AIN73_EF_READ_C, AIN74_EF_READ_C, AIN75_EF_READ_C, AIN76_EF_READ_C, AIN77_EF_READ_C, AIN78_EF_READ_C, AIN79_EF_READ_C, AIN80_EF_READ_C, AIN81_EF_READ_C, AIN82_EF_READ_C, AIN83_EF_READ_C, AIN84_EF_READ_C, AIN85_EF_READ_C, AIN86_EF_READ_C, AIN87_EF_READ_C, AIN88_EF_READ_C, AIN89_EF_READ_C, AIN90_EF_READ_C, AIN91_EF_READ_C, AIN92_EF_READ_C, AIN93_EF_READ_C, AIN94_EF_READ_C, AIN95_EF_READ_C, AIN96_EF_READ_C, AIN97_EF_READ_C, AIN98_EF_READ_C, AIN99_EF_READ_C, AIN100_EF_READ_C, AIN101_EF_READ_C, AIN102_EF_READ_C, AIN103_EF_READ_C, AIN104_EF_READ_C, AIN105_EF_READ_C, AIN106_EF_READ_C, AIN107_EF_READ_C, AIN108_EF_READ_C, AIN109_EF_READ_C, AIN110_EF_READ_C, AIN111_EF_READ_C, AIN112_EF_READ_C, AIN113_EF_READ_C, AIN114_EF_READ_C, AIN115_EF_READ_C, AIN116_EF_READ_C, AIN117_EF_READ_C, AIN118_EF_READ_C, AIN119_EF_READ_C, AIN120_EF_READ_C, AIN121_EF_READ_C, AIN122_EF_READ_C, AIN123_EF_READ_C, AIN124_EF_READ_C, AIN125_EF_READ_C, AIN126_EF_READ_C, AIN127_EF_READ_C, AIN128_EF_READ_C, AIN129_EF_READ_C, AIN130_EF_READ_C, AIN131_EF_READ_C, AIN132_EF_READ_C, AIN133_EF_READ_C, AIN134_EF_READ_C, AIN135_EF_READ_C, AIN136_EF_READ_C, AIN137_EF_READ_C, AIN138_EF_READ_C, AIN139_EF_READ_C, AIN140_EF_READ_C, AIN141_EF_READ_C, AIN142_EF_READ_C, AIN143_EF_READ_C, AIN144_EF_READ_C, AIN145_EF_READ_C, AIN146_EF_READ_C, AIN147_EF_READ_C, AIN148_EF_READ_C	7600, 7602, 7604, 7606, 7608, 7610, 7612, 7614, 7616, 7618, 7620, 7622, 7624, 7626, 7628, 7630, 7632, 7634, 7636, 7638, 7640, 7642, 7644, 7646, 7648, 7650, 7652, 7654, 7656, 7658, 7660, 7662, 7664, 7666, 7668, 7670, 7672, 7674, 7676, 7678, 7680, 7682, 7684, 7686, 7688, 7690, 7692, 7694, 7696, 7698, 7700, 7702, 7704, 7706, 7708, 7710, 7712, 7714, 7716, 7718, 7720, 7722, 7724, 7726, 7728, 7730, 7732, 7734, 7736, 7738, 7740, 7742, 7744, 7746, 7748, 7750, 7752, 7754, 7756, 7758, 7760, 7762, 7764, 7766, 7768, 7770, 7772, 7774, 7776, 7778, 7780, 7782, 7784, 7786, 7788, 7790, 7792, 7794, 7796, 7798, 7800, 7802, 7804, 7806, 7808, 7810, 7812, 7814, 7816, 7818, 7820, 7822, 7824, 7826, 7828, 7830, 7832, 7834, 7836, 7838, 7840, 7842, 7844, 7846, 7848, 7850, 7852, 7854, 7856, 7858, 7860, 7862, 7864, 7866, 7868, 7870, 7872, 7874, 7876, 7878, 7880, 7882, 7884, 7886, 7888, 7890, 7892, 7894, 7896

AIN#(0:148)_EF_READ_D
- Starting Address: 7900

Function dependent on selected feature index.

- Data type: FLOAT32 (type index = 3)
- Read-only
- Default value: 0
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0030

Expanded Names	Addresses

AIN0_EF_READ_D, AIN1_EF_READ_D, AIN2_EF_READ_D, AIN3_EF_READ_D, AIN4_EF_READ_D, AIN5_EF_READ_D, AIN6_EF_READ_D, AIN7_EF_READ_D, AIN8_EF_READ_D, AIN9_EF_READ_D, AIN10_EF_READ_D, AIN11_EF_READ_D, AIN12_EF_READ_D, AIN13_EF_READ_D, AIN14_EF_READ_D, AIN15_EF_READ_D, AIN16_EF_READ_D, AIN17_EF_READ_D, AIN18_EF_READ_D, AIN19_EF_READ_D, AIN20_EF_READ_D, AIN21_EF_READ_D, AIN22_EF_READ_D, AIN23_EF_READ_D, AIN24_EF_READ_D, AIN25_EF_READ_D, AIN26_EF_READ_D, AIN27_EF_READ_D, AIN28_EF_READ_D, AIN29_EF_READ_D, AIN30_EF_READ_D, AIN31_EF_READ_D, AIN32_EF_READ_D, AIN33_EF_READ_D, AIN34_EF_READ_D, AIN35_EF_READ_D, AIN36_EF_READ_D, AIN37_EF_READ_D, AIN38_EF_READ_D, AIN39_EF_READ_D, AIN40_EF_READ_D, AIN41_EF_READ_D, AIN42_EF_READ_D, AIN43_EF_READ_D, AIN44_EF_READ_D, AIN45_EF_READ_D, AIN46_EF_READ_D, AIN47_EF_READ_D, AIN48_EF_READ_D, AIN49_EF_READ_D, AIN50_EF_READ_D, AIN51_EF_READ_D, AIN52_EF_READ_D, AIN53_EF_READ_D, AIN54_EF_READ_D, AIN55_EF_READ_D, AIN56_EF_READ_D, AIN57_EF_READ_D, AIN58_EF_READ_D, AIN59_EF_READ_D, AIN60_EF_READ_D, AIN61_EF_READ_D, AIN62_EF_READ_D, AIN63_EF_READ_D, AIN64_EF_READ_D, AIN65_EF_READ_D, AIN66_EF_READ_D, AIN67_EF_READ_D, AIN68_EF_READ_D, AIN69_EF_READ_D, AIN70_EF_READ_D, AIN71_EF_READ_D, AIN72_EF_READ_D, AIN73_EF_READ_D, AIN74_EF_READ_D, AIN75_EF_READ_D, AIN76_EF_READ_D, AIN77_EF_READ_D, AIN78_EF_READ_D, AIN79_EF_READ_D, AIN80_EF_READ_D, AIN81_EF_READ_D, AIN82_EF_READ_D, AIN83_EF_READ_D, AIN84_EF_READ_D, AIN85_EF_READ_D, AIN86_EF_READ_D, AIN87_EF_READ_D, AIN88_EF_READ_D, AIN89_EF_READ_D, AIN90_EF_READ_D, AIN91_EF_READ_D, AIN92_EF_READ_D, AIN93_EF_READ_D, AIN94_EF_READ_D, AIN95_EF_READ_D, AIN96_EF_READ_D, AIN97_EF_READ_D, AIN98_EF_READ_D, AIN99_EF_READ_D, AIN100_EF_READ_D, AIN101_EF_READ_D, AIN102_EF_READ_D, AIN103_EF_READ_D, AIN104_EF_READ_D, AIN105_EF_READ_D, AIN106_EF_READ_D, AIN107_EF_READ_D, AIN108_EF_READ_D, AIN109_EF_READ_D, AIN110_EF_READ_D, AIN111_EF_READ_D, AIN112_EF_READ_D, AIN113_EF_READ_D, AIN114_EF_READ_D, AIN115_EF_READ_D, AIN116_EF_READ_D, AIN117_EF_READ_D, AIN118_EF_READ_D, AIN119_EF_READ_D, AIN120_EF_READ_D, AIN121_EF_READ_D, AIN122_EF_READ_D, AIN123_EF_READ_D, AIN124_EF_READ_D, AIN125_EF_READ_D, AIN126_EF_READ_D, AIN127_EF_READ_D, AIN128_EF_READ_D, AIN129_EF_READ_D, AIN130_EF_READ_D, AIN131_EF_READ_D, AIN132_EF_READ_D, AIN133_EF_READ_D, AIN134_EF_READ_D, AIN135_EF_READ_D, AIN136_EF_READ_D, AIN137_EF_READ_D, AIN138_EF_READ_D, AIN139_EF_READ_D, AIN140_EF_READ_D, AIN141_EF_READ_D, AIN142_EF_READ_D, AIN143_EF_READ_D, AIN144_EF_READ_D, AIN145_EF_READ_D, AIN146_EF_READ_D, AIN147_EF_READ_D, AIN148_EF_READ_D	7900, 7902, 7904, 7906, 7908, 7910, 7912, 7914, 7916, 7918, 7920, 7922, 7924, 7926, 7928, 7930, 7932, 7934, 7936, 7938, 7940, 7942, 7944, 7946, 7948, 7950, 7952, 7954, 7956, 7958, 7960, 7962, 7964, 7966, 7968, 7970, 7972, 7974, 7976, 7978, 7980, 7982, 7984, 7986, 7988, 7990, 7992, 7994, 7996, 7998, 8000, 8002, 8004, 8006, 8008, 8010, 8012, 8014, 8016, 8018, 8020, 8022, 8024, 8026, 8028, 8030, 8032, 8034, 8036, 8038, 8040, 8042, 8044, 8046, 8048, 8050, 8052, 8054, 8056, 8058, 8060, 8062, 8064, 8066, 8068, 8070, 8072, 8074, 8076, 8078, 8080, 8082, 8084, 8086, 8088, 8090, 8092, 8094, 8096, 8098, 8100, 8102, 8104, 8106, 8108, 8110, 8112, 8114, 8116, 8118, 8120, 8122, 8124, 8126, 8128, 8130, 8132, 8134, 8136, 8138, 8140, 8142, 8144, 8146, 8148, 8150, 8152, 8154, 8156, 8158, 8160, 8162, 8164, 8166, 8168, 8170, 8172, 8174, 8176, 8178, 8180, 8182, 8184, 8186, 8188, 8190, 8192, 8194, 8196

AIN#(0:6)_EF_READ_A_CAPTURE
- Starting Address: 8800

(T8 only) Read AIN_EF with the last capture sampled, use AIN#(0:7)_EF_READ_A to read a new sample

- Data type: FLOAT32 (type index = 3)
- Read-only
- Default value: 0
- T8:
    - Minimum firmware version: 1.0014

Expanded Names	Addresses
AIN0_EF_READ_A_CAPTURE, AIN1_EF_READ_A_CAPTURE, AIN2_EF_READ_A_CAPTURE, AIN3_EF_READ_A_CAPTURE, AIN4_EF_READ_A_CAPTURE, AIN5_EF_READ_A_CAPTURE, AIN6_EF_READ_A_CAPTURE	8800, 8802, 8804, 8806, 8808, 8810, 8812

AIN#(0:148)_EF_INDEX
- Starting Address: 9000

Specify the desired extended feature for this analog input with the index value. List of index values: 0=None(disabled); 1=Offset and Slope; 3=Max/Min/Avg; 4=Resistance; 5=Average and Threshold; 10=RMS Flex; 11=FlexRMS; 20=Thermocouple type E; 21=Thermocouple type J; 22=Thermocouple type K; 23=Thermocouple type R; 24=Thermocouple type T; 25=Thermocouple type S; 27=Thermocouple type N; 28=Thermocouple type B; 30=Thermocouple type C; 40=RTD model PT100; 41=RTD model PT500; 42=RTD model PT1000; 50=Thermistor Steinhart-Hart; 51=Thermistor Beta.

- Data type: UINT32 (type index = 1)
- Readable and writable
- Default value: 0
- This register uses system RAM. The maximum RAM is 64KB. For more information, see 4.4 RAM
- T8:
    - Minimum firmware version: 0.0123
- T7:
    - Minimum firmware version: 1.0030
- T4:
    - The T4 does not support thermocouple modes.

Expanded Names	Addresses
AIN0_EF_INDEX, AIN1_EF_INDEX, AIN2_EF_INDEX, AIN3_EF_INDEX, AIN4_EF_INDEX, AIN5_EF_INDEX, AIN6_EF_INDEX, AIN7_EF_INDEX, AIN8_EF_INDEX, AIN9_EF_INDEX, AIN10_EF_INDEX, AIN11_EF_INDEX, AIN12_EF_INDEX, AIN13_EF_INDEX, AIN14_EF_INDEX, AIN15_EF_INDEX, AIN16_EF_INDEX, AIN17_EF_INDEX, AIN18_EF_INDEX, AIN19_EF_INDEX, AIN20_EF_INDEX, AIN21_EF_INDEX, AIN22_EF_INDEX, AIN23_EF_INDEX, AIN24_EF_INDEX, AIN25_EF_INDEX, AIN26_EF_INDEX, AIN27_EF_INDEX, AIN28_EF_INDEX, AIN29_EF_INDEX, AIN30_EF_INDEX, AIN31_EF_INDEX, AIN32_EF_INDEX, AIN33_EF_INDEX, AIN34_EF_INDEX, AIN35_EF_INDEX, AIN36_EF_INDEX, AIN37_EF_INDEX, AIN38_EF_INDEX, AIN39_EF_INDEX, AIN40_EF_INDEX, AIN41_EF_INDEX, AIN42_EF_INDEX, AIN43_EF_INDEX, AIN44_EF_INDEX, AIN45_EF_INDEX, AIN46_EF_INDEX, AIN47_EF_INDEX, AIN48_EF_INDEX, AIN49_EF_INDEX, AIN50_EF_INDEX, AIN51_EF_INDEX, AIN52_EF_INDEX, AIN53_EF_INDEX, AIN54_EF_INDEX, AIN55_EF_INDEX, AIN56_EF_INDEX, AIN57_EF_INDEX, AIN58_EF_INDEX, AIN59_EF_INDEX, AIN60_EF_INDEX, AIN61_EF_INDEX, AIN62_EF_INDEX, AIN63_EF_INDEX, AIN64_EF_INDEX, AIN65_EF_INDEX, AIN66_EF_INDEX, AIN67_EF_INDEX, AIN68_EF_INDEX, AIN69_EF_INDEX, AIN70_EF_INDEX, AIN71_EF_INDEX, AIN72_EF_INDEX, AIN73_EF_INDEX, AIN74_EF_INDEX, AIN75_EF_INDEX, AIN76_EF_INDEX, AIN77_EF_INDEX, AIN78_EF_INDEX, AIN79_EF_INDEX, AIN80_EF_INDEX, AIN81_EF_INDEX, AIN82_EF_INDEX, AIN83_EF_INDEX, AIN84_EF_INDEX, AIN85_EF_INDEX, AIN86_EF_INDEX, AIN87_EF_INDEX, AIN88_EF_INDEX, AIN89_EF_INDEX, AIN90_EF_INDEX, AIN91_EF_INDEX, AIN92_EF_INDEX, AIN93_EF_INDEX, AIN94_EF_INDEX, AIN95_EF_INDEX, AIN96_EF_INDEX, AIN97_EF_INDEX, AIN98_EF_INDEX, AIN99_EF_INDEX, AIN100_EF_INDEX, AIN101_EF_INDEX, AIN102_EF_INDEX, AIN103_EF_INDEX, AIN104_EF_INDEX, AIN105_EF_INDEX, AIN106_EF_INDEX, AIN107_EF_INDEX, AIN108_EF_INDEX, AIN109_EF_INDEX, AIN110_EF_INDEX, AIN111_EF_INDEX, AIN112_EF_INDEX, AIN113_EF_INDEX, AIN114_EF_INDEX, AIN115_EF_INDEX, AIN116_EF_INDEX, AIN117_EF_INDEX, AIN118_EF_INDEX, AIN119_EF_INDEX, AIN120_EF_INDEX, AIN121_EF_INDEX, AIN122_EF_INDEX, AIN123_EF_INDEX, AIN124_EF_INDEX, AIN125_EF_INDEX, AIN126_EF_INDEX, AIN127_EF_INDEX, AIN128_EF_INDEX, AIN129_EF_INDEX, AIN130_EF_INDEX, AIN131_EF_INDEX, AIN132_EF_INDEX, AIN133_EF_INDEX, AIN134_EF_INDEX, AIN135_EF_INDEX, AIN136_EF_INDEX, AIN137_EF_INDEX, AIN138_EF_INDEX, AIN139_EF_INDEX, AIN140_EF_INDEX, AIN141_EF_INDEX, AIN142_EF_INDEX, AIN143_EF_INDEX, AIN144_EF_INDEX, AIN145_EF_INDEX, AIN146_EF_INDEX, AIN147_EF_INDEX, AIN148_EF_INDEX	9000, 9002, 9004, 9006, 9008, 9010, 9012, 9014, 9016, 9018, 9020, 9022, 9024, 9026, 9028, 9030, 9032, 9034, 9036, 9038, 9040, 9042, 9044, 9046, 9048, 9050, 9052, 9054, 9056, 9058, 9060, 9062, 9064, 9066, 9068, 9070, 9072, 9074, 9076, 9078, 9080, 9082, 9084, 9086, 9088, 9090, 9092, 9094, 9096, 9098, 9100, 9102, 9104, 9106, 9108, 9110, 9112, 9114, 9116, 9118, 9120, 9122, 9124, 9126, 9128, 9130, 9132, 9134, 9136, 9138, 9140, 9142, 9144, 9146, 9148, 9150, 9152, 9154, 9156, 9158, 9160, 9162, 9164, 9166, 9168, 9170, 9172, 9174, 9176, 9178, 9180, 9182, 9184, 9186, 9188, 9190, 9192, 9194, 9196, 9198, 9200, 9202, 9204, 9206, 9208, 9210, 9212, 9214, 9216, 9218, 9220, 9222, 9224, 9226, 9228, 9230, 9232, 9234, 9236, 9238, 9240, 9242, 9244, 9246, 9248, 9250, 9252, 9254, 9256, 9258, 9260, 9262, 9264, 9266, 9268, 9270, 9272, 9274, 9276, 9278, 9280, 9282, 9284, 9286, 9288, 9290, 9292, 9294, 9296

AIN#(0:148)_EF_CONFIG_A
- Starting Address: 9300

Function dependent on selected feature index.

- Data type: UINT32 (type index = 1)
- Readable and writable
- Default value: 0
- T8:
    - Minimum firmware version: 0.0123
- T7:
    - Minimum firmware version: 1.0030

Expanded Names	Addresses

AIN0_EF_CONFIG_A, AIN1_EF_CONFIG_A, AIN2_EF_CONFIG_A,	9300, 9302, 9304,
AIN3_EF_CONFIG_A, AIN4_EF_CONFIG_A, AIN5_EF_CONFIG_A,	9306, 9308, 9310,
AIN6_EF_CONFIG_A, AIN7_EF_CONFIG_A, AIN8_EF_CONFIG_A,	9312, 9314, 9316,
AIN9_EF_CONFIG_A, AIN10_EF_CONFIG_A, AIN11_EF_CONFIG_A,	9318, 9320, 9322,
AIN12_EF_CONFIG_A, AIN13_EF_CONFIG_A, AIN14_EF_CONFIG_A,	9324, 9326, 9328,
AIN15_EF_CONFIG_A, AIN16_EF_CONFIG_A, AIN17_EF_CONFIG_A,	9330, 9332, 9334,
AIN18_EF_CONFIG_A, AIN19_EF_CONFIG_A, AIN20_EF_CONFIG_A,	9336, 9338, 9340,
AIN21_EF_CONFIG_A, AIN22_EF_CONFIG_A, AIN23_EF_CONFIG_A,	9342, 9344, 9346,
AIN24_EF_CONFIG_A, AIN25_EF_CONFIG_A, AIN26_EF_CONFIG_A,	9348, 9350, 9352,
AIN27_EF_CONFIG_A, AIN28_EF_CONFIG_A, AIN29_EF_CONFIG_A,	9354, 9356, 9358,
AIN30_EF_CONFIG_A, AIN31_EF_CONFIG_A, AIN32_EF_CONFIG_A,	9360, 9362, 9364,
AIN33_EF_CONFIG_A, AIN34_EF_CONFIG_A, AIN35_EF_CONFIG_A,	9366, 9368, 9370,
AIN36_EF_CONFIG_A, AIN37_EF_CONFIG_A, AIN38_EF_CONFIG_A,	9372, 9374, 9376,
AIN39_EF_CONFIG_A, AIN40_EF_CONFIG_A, AIN41_EF_CONFIG_A,	9378, 9380, 9382,
AIN42_EF_CONFIG_A, AIN43_EF_CONFIG_A, AIN44_EF_CONFIG_A,	9384, 9386, 9388,
AIN45_EF_CONFIG_A, AIN46_EF_CONFIG_A, AIN47_EF_CONFIG_A,	9390, 9392, 9394,
AIN48_EF_CONFIG_A, AIN49_EF_CONFIG_A, AIN50_EF_CONFIG_A,	9396, 9398, 9400,
AIN51_EF_CONFIG_A, AIN52_EF_CONFIG_A, AIN53_EF_CONFIG_A,	9402, 9404, 9406,
AIN54_EF_CONFIG_A, AIN55_EF_CONFIG_A, AIN56_EF_CONFIG_A,	9408, 9410, 9412,
AIN57_EF_CONFIG_A, AIN58_EF_CONFIG_A, AIN59_EF_CONFIG_A,	9414, 9416, 9418,
AIN60_EF_CONFIG_A, AIN61_EF_CONFIG_A, AIN62_EF_CONFIG_A,	9420, 9422, 9424,
AIN63_EF_CONFIG_A, AIN64_EF_CONFIG_A, AIN65_EF_CONFIG_A,	9426, 9428, 9430,
AIN66_EF_CONFIG_A, AIN67_EF_CONFIG_A, AIN68_EF_CONFIG_A,	9432, 9434, 9436,
AIN69_EF_CONFIG_A, AIN70_EF_CONFIG_A, AIN71_EF_CONFIG_A,	9438, 9440, 9442,
AIN72_EF_CONFIG_A, AIN73_EF_CONFIG_A, AIN74_EF_CONFIG_A,	9444, 9446, 9448,
AIN75_EF_CONFIG_A, AIN76_EF_CONFIG_A, AIN77_EF_CONFIG_A,	9450, 9452, 9454,
AIN78_EF_CONFIG_A, AIN79_EF_CONFIG_A, AIN80_EF_CONFIG_A,	9456, 9458, 9460,
AIN81_EF_CONFIG_A, AIN82_EF_CONFIG_A, AIN83_EF_CONFIG_A,	9462, 9464, 9466,
AIN84_EF_CONFIG_A, AIN85_EF_CONFIG_A, AIN86_EF_CONFIG_A,	9468, 9470, 9472,
AIN87_EF_CONFIG_A, AIN88_EF_CONFIG_A, AIN89_EF_CONFIG_A,	9474, 9476, 9478,
AIN90_EF_CONFIG_A, AIN91_EF_CONFIG_A, AIN92_EF_CONFIG_A,	9480, 9482, 9484,
AIN93_EF_CONFIG_A, AIN94_EF_CONFIG_A, AIN95_EF_CONFIG_A,	9486, 9488, 9490,
AIN96_EF_CONFIG_A, AIN97_EF_CONFIG_A, AIN98_EF_CONFIG_A,	9492, 9494, 9496,
AIN99_EF_CONFIG_A, AIN100_EF_CONFIG_A, AIN101_EF_CONFIG_A,	9498, 9500, 9502,
AIN102_EF_CONFIG_A, AIN103_EF_CONFIG_A, AIN104_EF_CONFIG_A,	9504, 9506, 9508,
AIN105_EF_CONFIG_A, AIN106_EF_CONFIG_A, AIN107_EF_CONFIG_A,	9510, 9512, 9514,
AIN108_EF_CONFIG_A, AIN109_EF_CONFIG_A, AIN110_EF_CONFIG_A,	9516, 9518, 9520,
AIN111_EF_CONFIG_A, AIN112_EF_CONFIG_A, AIN113_EF_CONFIG_A,	9522, 9524, 9526,
AIN114_EF_CONFIG_A, AIN115_EF_CONFIG_A, AIN116_EF_CONFIG_A,	9528, 9530, 9532,
AIN117_EF_CONFIG_A, AIN118_EF_CONFIG_A, AIN119_EF_CONFIG_A,	9534, 9536, 9538,
AIN120_EF_CONFIG_A, AIN121_EF_CONFIG_A, AIN122_EF_CONFIG_A,	9540, 9542, 9544,
AIN123_EF_CONFIG_A, AIN124_EF_CONFIG_A, AIN125_EF_CONFIG_A,	9546, 9548, 9550,
AIN126_EF_CONFIG_A, AIN127_EF_CONFIG_A, AIN128_EF_CONFIG_A,	9552, 9554, 9556,
AIN129_EF_CONFIG_A, AIN130_EF_CONFIG_A, AIN131_EF_CONFIG_A,	9558, 9560, 9562,
AIN132_EF_CONFIG_A, AIN133_EF_CONFIG_A, AIN134_EF_CONFIG_A,	9564, 9566, 9568,
AIN135_EF_CONFIG_A, AIN136_EF_CONFIG_A, AIN137_EF_CONFIG_A,	9570, 9572, 9574,
AIN138_EF_CONFIG_A, AIN139_EF_CONFIG_A, AIN140_EF_CONFIG_A,	9576, 9578, 9580,
AIN141_EF_CONFIG_A, AIN142_EF_CONFIG_A, AIN143_EF_CONFIG_A,	9582, 9584, 9586,
AIN144_EF_CONFIG_A, AIN145_EF_CONFIG_A, AIN146_EF_CONFIG_A,	9588, 9590, 9592,
AIN147_EF_CONFIG_A, AIN148_EF_CONFIG_A	9594, 9596

AIN#(0:148)_EF_CONFIG_B
- Starting Address: 9600

Function dependent on selected feature index.

- Data type: UINT32 (type index = 1)
- Readable and writable
- Default value: 0
- T8:
    - Minimum firmware version: 0.0123
- T7:

- Minimum firmware version: 1.0030

Expanded Names	Addresses
AIN0_EF_CONFIG_B, AIN1_EF_CONFIG_B, AIN2_EF_CONFIG_B,	9600, 9602, 9604,
AIN3_EF_CONFIG_B, AIN4_EF_CONFIG_B, AIN5_EF_CONFIG_B,	9606, 9608, 9610,
AIN6_EF_CONFIG_B, AIN7_EF_CONFIG_B, AIN8_EF_CONFIG_B,	9612, 9614, 9616,
AIN9_EF_CONFIG_B, AIN10_EF_CONFIG_B, AIN11_EF_CONFIG_B,	9618, 9620, 9622,
AIN12_EF_CONFIG_B, AIN13_EF_CONFIG_B, AIN14_EF_CONFIG_B,	9624, 9626, 9628,
AIN15_EF_CONFIG_B, AIN16_EF_CONFIG_B, AIN17_EF_CONFIG_B,	9630, 9632, 9634,
AIN18_EF_CONFIG_B, AIN19_EF_CONFIG_B, AIN20_EF_CONFIG_B,	9636, 9638, 9640,
AIN21_EF_CONFIG_B, AIN22_EF_CONFIG_B, AIN23_EF_CONFIG_B,	9642, 9644, 9646,
AIN24_EF_CONFIG_B, AIN25_EF_CONFIG_B, AIN26_EF_CONFIG_B,	9648, 9650, 9652,
AIN27_EF_CONFIG_B, AIN28_EF_CONFIG_B, AIN29_EF_CONFIG_B,	9654, 9656, 9658,
AIN30_EF_CONFIG_B, AIN31_EF_CONFIG_B, AIN32_EF_CONFIG_B,	9660, 9662, 9664,
AIN33_EF_CONFIG_B, AIN34_EF_CONFIG_B, AIN35_EF_CONFIG_B,	9666, 9668, 9670,
AIN36_EF_CONFIG_B, AIN37_EF_CONFIG_B, AIN38_EF_CONFIG_B,	9672, 9674, 9676,
AIN39_EF_CONFIG_B, AIN40_EF_CONFIG_B, AIN41_EF_CONFIG_B,	9678, 9680, 9682,
AIN42_EF_CONFIG_B, AIN43_EF_CONFIG_B, AIN44_EF_CONFIG_B,	9684, 9686, 9688,
AIN45_EF_CONFIG_B, AIN46_EF_CONFIG_B, AIN47_EF_CONFIG_B,	9690, 9692, 9694,
AIN48_EF_CONFIG_B, AIN49_EF_CONFIG_B, AIN50_EF_CONFIG_B,	9696, 9698, 9700,
AIN51_EF_CONFIG_B, AIN52_EF_CONFIG_B, AIN53_EF_CONFIG_B,	9702, 9704, 9706,
AIN54_EF_CONFIG_B, AIN55_EF_CONFIG_B, AIN56_EF_CONFIG_B,	9708, 9710, 9712,
AIN57_EF_CONFIG_B, AIN58_EF_CONFIG_B, AIN59_EF_CONFIG_B,	9714, 9716, 9718,
AIN60_EF_CONFIG_B, AIN61_EF_CONFIG_B, AIN62_EF_CONFIG_B,	9720, 9722, 9724,
AIN63_EF_CONFIG_B, AIN64_EF_CONFIG_B, AIN65_EF_CONFIG_B,	9726, 9728, 9730,
AIN66_EF_CONFIG_B, AIN67_EF_CONFIG_B, AIN68_EF_CONFIG_B,	9732, 9734, 9736,
AIN69_EF_CONFIG_B, AIN70_EF_CONFIG_B, AIN71_EF_CONFIG_B,	9738, 9740, 9742,
AIN72_EF_CONFIG_B, AIN73_EF_CONFIG_B, AIN74_EF_CONFIG_B,	9744, 9746, 9748,
AIN75_EF_CONFIG_B, AIN76_EF_CONFIG_B, AIN77_EF_CONFIG_B,	9750, 9752, 9754,
AIN78_EF_CONFIG_B, AIN79_EF_CONFIG_B, AIN80_EF_CONFIG_B,	9756, 9758, 9760,
AIN81_EF_CONFIG_B, AIN82_EF_CONFIG_B, AIN83_EF_CONFIG_B,	9762, 9764, 9766,
AIN84_EF_CONFIG_B, AIN85_EF_CONFIG_B, AIN86_EF_CONFIG_B,	9768, 9770, 9772,
AIN87_EF_CONFIG_B, AIN88_EF_CONFIG_B, AIN89_EF_CONFIG_B,	9774, 9776, 9778,
AIN90_EF_CONFIG_B, AIN91_EF_CONFIG_B, AIN92_EF_CONFIG_B,	9780, 9782, 9784,
AIN93_EF_CONFIG_B, AIN94_EF_CONFIG_B, AIN95_EF_CONFIG_B,	9786, 9788, 9790,
AIN96_EF_CONFIG_B, AIN97_EF_CONFIG_B, AIN98_EF_CONFIG_B,	9792, 9794, 9796,
AIN99_EF_CONFIG_B, AIN100_EF_CONFIG_B, AIN101_EF_CONFIG_B,	9798, 9800, 9802,
AIN102_EF_CONFIG_B, AIN103_EF_CONFIG_B, AIN104_EF_CONFIG_B,	9804, 9806, 9808,
AIN105_EF_CONFIG_B, AIN106_EF_CONFIG_B, AIN107_EF_CONFIG_B,	9810, 9812, 9814,
AIN108_EF_CONFIG_B, AIN109_EF_CONFIG_B, AIN110_EF_CONFIG_B,	9816, 9818, 9820,
AIN111_EF_CONFIG_B, AIN112_EF_CONFIG_B, AIN113_EF_CONFIG_B,	9822, 9824, 9826,
AIN114_EF_CONFIG_B, AIN115_EF_CONFIG_B, AIN116_EF_CONFIG_B,	9828, 9830, 9832,
AIN117_EF_CONFIG_B, AIN118_EF_CONFIG_B, AIN119_EF_CONFIG_B,	9834, 9836, 9838,
AIN120_EF_CONFIG_B, AIN121_EF_CONFIG_B, AIN122_EF_CONFIG_B,	9840, 9842, 9844,
AIN123_EF_CONFIG_B, AIN124_EF_CONFIG_B, AIN125_EF_CONFIG_B,	9846, 9848, 9850,
AIN126_EF_CONFIG_B, AIN127_EF_CONFIG_B, AIN128_EF_CONFIG_B,	9852, 9854, 9856,
AIN129_EF_CONFIG_B, AIN130_EF_CONFIG_B, AIN131_EF_CONFIG_B,	9858, 9860, 9862,
AIN132_EF_CONFIG_B, AIN133_EF_CONFIG_B, AIN134_EF_CONFIG_B,	9864, 9866, 9868,
AIN135_EF_CONFIG_B, AIN136_EF_CONFIG_B, AIN137_EF_CONFIG_B,	9870, 9872, 9874,
AIN138_EF_CONFIG_B, AIN139_EF_CONFIG_B, AIN140_EF_CONFIG_B,	9876, 9878, 9880,
AIN141_EF_CONFIG_B, AIN142_EF_CONFIG_B, AIN143_EF_CONFIG_B,	9882, 9884, 9886,
AIN144_EF_CONFIG_B, AIN145_EF_CONFIG_B, AIN146_EF_CONFIG_B,	9888, 9890, 9892,
AIN147_EF_CONFIG_B, AIN148_EF_CONFIG_B	9894, 9896

AIN#(0:148)_EF_CONFIG_C
- Starting Address: 9900

Function dependent on selected feature index.

- Data type: UINT32 (type index = 1)
- Readable and writable
- Default value: 0

- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0030

Expanded Names	Addresses
AIN0_EF_CONFIG_C, AIN1_EF_CONFIG_C, AIN2_EF_CONFIG_C, AIN3_EF_CONFIG_C, AIN4_EF_CONFIG_C, AIN5_EF_CONFIG_C, AIN6_EF_CONFIG_C, AIN7_EF_CONFIG_C, AIN8_EF_CONFIG_C, AIN9_EF_CONFIG_C, AIN10_EF_CONFIG_C, AIN11_EF_CONFIG_C, AIN12_EF_CONFIG_C, AIN13_EF_CONFIG_C, AIN14_EF_CONFIG_C, AIN15_EF_CONFIG_C, AIN16_EF_CONFIG_C, AIN17_EF_CONFIG_C, AIN18_EF_CONFIG_C, AIN19_EF_CONFIG_C, AIN20_EF_CONFIG_C, AIN21_EF_CONFIG_C, AIN22_EF_CONFIG_C, AIN23_EF_CONFIG_C, AIN24_EF_CONFIG_C, AIN25_EF_CONFIG_C, AIN26_EF_CONFIG_C, AIN27_EF_CONFIG_C, AIN28_EF_CONFIG_C, AIN29_EF_CONFIG_C, AIN30_EF_CONFIG_C, AIN31_EF_CONFIG_C, AIN32_EF_CONFIG_C, AIN33_EF_CONFIG_C, AIN34_EF_CONFIG_C, AIN35_EF_CONFIG_C, AIN36_EF_CONFIG_C, AIN37_EF_CONFIG_C, AIN38_EF_CONFIG_C, AIN39_EF_CONFIG_C, AIN40_EF_CONFIG_C, AIN41_EF_CONFIG_C, AIN42_EF_CONFIG_C, AIN43_EF_CONFIG_C, AIN44_EF_CONFIG_C, AIN45_EF_CONFIG_C, AIN46_EF_CONFIG_C, AIN47_EF_CONFIG_C, AIN48_EF_CONFIG_C, AIN49_EF_CONFIG_C, AIN50_EF_CONFIG_C, AIN51_EF_CONFIG_C, AIN52_EF_CONFIG_C, AIN53_EF_CONFIG_C, AIN54_EF_CONFIG_C, AIN55_EF_CONFIG_C, AIN56_EF_CONFIG_C, AIN57_EF_CONFIG_C, AIN58_EF_CONFIG_C, AIN59_EF_CONFIG_C, AIN60_EF_CONFIG_C, AIN61_EF_CONFIG_C, AIN62_EF_CONFIG_C, AIN63_EF_CONFIG_C, AIN64_EF_CONFIG_C, AIN65_EF_CONFIG_C, AIN66_EF_CONFIG_C, AIN67_EF_CONFIG_C, AIN68_EF_CONFIG_C, AIN69_EF_CONFIG_C, AIN70_EF_CONFIG_C, AIN71_EF_CONFIG_C, AIN72_EF_CONFIG_C, AIN73_EF_CONFIG_C, AIN74_EF_CONFIG_C, AIN75_EF_CONFIG_C, AIN76_EF_CONFIG_C, AIN77_EF_CONFIG_C, AIN78_EF_CONFIG_C, AIN79_EF_CONFIG_C, AIN80_EF_CONFIG_C, AIN81_EF_CONFIG_C, AIN82_EF_CONFIG_C, AIN83_EF_CONFIG_C, AIN84_EF_CONFIG_C, AIN85_EF_CONFIG_C, AIN86_EF_CONFIG_C, AIN87_EF_CONFIG_C, AIN88_EF_CONFIG_C, AIN89_EF_CONFIG_C, AIN90_EF_CONFIG_C, AIN91_EF_CONFIG_C, AIN92_EF_CONFIG_C, AIN93_EF_CONFIG_C, AIN94_EF_CONFIG_C, AIN95_EF_CONFIG_C, AIN96_EF_CONFIG_C, AIN97_EF_CONFIG_C, AIN98_EF_CONFIG_C, AIN99_EF_CONFIG_C, AIN100_EF_CONFIG_C, AIN101_EF_CONFIG_C, AIN102_EF_CONFIG_C, AIN103_EF_CONFIG_C, AIN104_EF_CONFIG_C, AIN105_EF_CONFIG_C, AIN106_EF_CONFIG_C, AIN107_EF_CONFIG_C, AIN108_EF_CONFIG_C, AIN109_EF_CONFIG_C, AIN110_EF_CONFIG_C, AIN111_EF_CONFIG_C, AIN112_EF_CONFIG_C, AIN113_EF_CONFIG_C, AIN114_EF_CONFIG_C, AIN115_EF_CONFIG_C, AIN116_EF_CONFIG_C, AIN117_EF_CONFIG_C, AIN118_EF_CONFIG_C, AIN119_EF_CONFIG_C, AIN120_EF_CONFIG_C, AIN121_EF_CONFIG_C, AIN122_EF_CONFIG_C, AIN123_EF_CONFIG_C, AIN124_EF_CONFIG_C, AIN125_EF_CONFIG_C, AIN126_EF_CONFIG_C, AIN127_EF_CONFIG_C, AIN128_EF_CONFIG_C, AIN129_EF_CONFIG_C, AIN130_EF_CONFIG_C, AIN131_EF_CONFIG_C, AIN132_EF_CONFIG_C, AIN133_EF_CONFIG_C, AIN134_EF_CONFIG_C, AIN135_EF_CONFIG_C, AIN136_EF_CONFIG_C, AIN137_EF_CONFIG_C, AIN138_EF_CONFIG_C, AIN139_EF_CONFIG_C, AIN140_EF_CONFIG_C, AIN141_EF_CONFIG_C, AIN142_EF_CONFIG_C, AIN143_EF_CONFIG_C, AIN144_EF_CONFIG_C, AIN145_EF_CONFIG_C, AIN146_EF_CONFIG_C, AIN147_EF_CONFIG_C, AIN148_EF_CONFIG_C	9900, 9902, 9904, 9906, 9908, 9910, 9912, 9914, 9916, 9918, 9920, 9922, 9924, 9926, 9928, 9930, 9932, 9934, 9936, 9938, 9940, 9942, 9944, 9946, 9948, 9950, 9952, 9954, 9956, 9958, 9960, 9962, 9964, 9966, 9968, 9970, 9972, 9974, 9976, 9978, 9980, 9982, 9984, 9986, 9988, 9990, 9992, 9994, 9996, 9998, 10000, 10002, 10004, 10006, 10008, 10010, 10012, 10014, 10016, 10018, 10020, 10022, 10024, 10026, 10028, 10030, 10032, 10034, 10036, 10038, 10040, 10042, 10044, 10046, 10048, 10050, 10052, 10054, 10056, 10058, 10060, 10062, 10064, 10066, 10068, 10070, 10072, 10074, 10076, 10078, 10080, 10082, 10084, 10086, 10088, 10090, 10092, 10094, 10096, 10098, 10100, 10102, 10104, 10106, 10108, 10110, 10112, 10114, 10116, 10118, 10120, 10122, 10124, 10126, 10128, 10130, 10132, 10134, 10136, 10138, 10140, 10142, 10144, 10146, 10148, 10150, 10152, 10154, 10156, 10158, 10160, 10162, 10164, 10166, 10168, 10170, 10172, 10174, 10176, 10178, 10180, 10182, 10184, 10186, 10188, 10190, 10192, 10194, 10196

AIN#(0:148)_EF_CONFIG_D
- Starting Address: 10200

Function dependent on selected feature index.

- Data type: FLOAT32 (type index = 3)
- Readable and writable
- Default value: 0
- T8:
    - Minimum firmware version: 0.0123
- T7:
    - Minimum firmware version: 1.0030

Expanded Names	Addresses
AIN0_EF_CONFIG_D, AIN1_EF_CONFIG_D, AIN2_EF_CONFIG_D,	10200, 10202, 10204,
AIN3_EF_CONFIG_D, AIN4_EF_CONFIG_D, AIN5_EF_CONFIG_D,	10206, 10208, 10210,
AIN6_EF_CONFIG_D, AIN7_EF_CONFIG_D, AIN8_EF_CONFIG_D,	10212, 10214, 10216,
AIN9_EF_CONFIG_D, AIN10_EF_CONFIG_D, AIN11_EF_CONFIG_D,	10218, 10220, 10222,
AIN12_EF_CONFIG_D, AIN13_EF_CONFIG_D, AIN14_EF_CONFIG_D,	10224, 10226, 10228,
AIN15_EF_CONFIG_D, AIN16_EF_CONFIG_D, AIN17_EF_CONFIG_D,	10230, 10232, 10234,
AIN18_EF_CONFIG_D, AIN19_EF_CONFIG_D, AIN20_EF_CONFIG_D,	10236, 10238, 10240,
AIN21_EF_CONFIG_D, AIN22_EF_CONFIG_D, AIN23_EF_CONFIG_D,	10242, 10244, 10246,
AIN24_EF_CONFIG_D, AIN25_EF_CONFIG_D, AIN26_EF_CONFIG_D,	10248, 10250, 10252,
AIN27_EF_CONFIG_D, AIN28_EF_CONFIG_D, AIN29_EF_CONFIG_D,	10254, 10256, 10258,
AIN30_EF_CONFIG_D, AIN31_EF_CONFIG_D, AIN32_EF_CONFIG_D,	10260, 10262, 10264,
AIN33_EF_CONFIG_D, AIN34_EF_CONFIG_D, AIN35_EF_CONFIG_D,	10266, 10268, 10270,
AIN36_EF_CONFIG_D, AIN37_EF_CONFIG_D, AIN38_EF_CONFIG_D,	10272, 10274, 10276,
AIN39_EF_CONFIG_D, AIN40_EF_CONFIG_D, AIN41_EF_CONFIG_D,	10278, 10280, 10282,
AIN42_EF_CONFIG_D, AIN43_EF_CONFIG_D, AIN44_EF_CONFIG_D,	10284, 10286, 10288,
AIN45_EF_CONFIG_D, AIN46_EF_CONFIG_D, AIN47_EF_CONFIG_D,	10290, 10292, 10294,
AIN48_EF_CONFIG_D, AIN49_EF_CONFIG_D, AIN50_EF_CONFIG_D,	10296, 10298, 10300,
AIN51_EF_CONFIG_D, AIN52_EF_CONFIG_D, AIN53_EF_CONFIG_D,	10302, 10304, 10306,
AIN54_EF_CONFIG_D, AIN55_EF_CONFIG_D, AIN56_EF_CONFIG_D,	10308, 10310, 10312,
AIN57_EF_CONFIG_D, AIN58_EF_CONFIG_D, AIN59_EF_CONFIG_D,	10314, 10316, 10318,
AIN60_EF_CONFIG_D, AIN61_EF_CONFIG_D, AIN62_EF_CONFIG_D,	10320, 10322, 10324,
AIN63_EF_CONFIG_D, AIN64_EF_CONFIG_D, AIN65_EF_CONFIG_D,	10326, 10328, 10330,
AIN66_EF_CONFIG_D, AIN67_EF_CONFIG_D, AIN68_EF_CONFIG_D,	10332, 10334, 10336,
AIN69_EF_CONFIG_D, AIN70_EF_CONFIG_D, AIN71_EF_CONFIG_D,	10338, 10340, 10342,
AIN72_EF_CONFIG_D, AIN73_EF_CONFIG_D, AIN74_EF_CONFIG_D,	10344, 10346, 10348,
AIN75_EF_CONFIG_D, AIN76_EF_CONFIG_D, AIN77_EF_CONFIG_D,	10350, 10352, 10354,
AIN78_EF_CONFIG_D, AIN79_EF_CONFIG_D, AIN80_EF_CONFIG_D,	10356, 10358, 10360,
AIN81_EF_CONFIG_D, AIN82_EF_CONFIG_D, AIN83_EF_CONFIG_D,	10362, 10364, 10366,
AIN84_EF_CONFIG_D, AIN85_EF_CONFIG_D, AIN86_EF_CONFIG_D,	10368, 10370, 10372,
AIN87_EF_CONFIG_D, AIN88_EF_CONFIG_D, AIN89_EF_CONFIG_D,	10374, 10376, 10378,
AIN90_EF_CONFIG_D, AIN91_EF_CONFIG_D, AIN92_EF_CONFIG_D,	10380, 10382, 10384,
AIN93_EF_CONFIG_D, AIN94_EF_CONFIG_D, AIN95_EF_CONFIG_D,	10386, 10388, 10390,
AIN96_EF_CONFIG_D, AIN97_EF_CONFIG_D, AIN98_EF_CONFIG_D,	10392, 10394, 10396,
AIN99_EF_CONFIG_D, AIN100_EF_CONFIG_D, AIN101_EF_CONFIG_D,	10398, 10400, 10402,
AIN102_EF_CONFIG_D, AIN103_EF_CONFIG_D, AIN104_EF_CONFIG_D,	10404, 10406, 10408,
AIN105_EF_CONFIG_D, AIN106_EF_CONFIG_D, AIN107_EF_CONFIG_D,	10410, 10412, 10414,
AIN108_EF_CONFIG_D, AIN109_EF_CONFIG_D, AIN110_EF_CONFIG_D,	10416, 10418, 10420,
AIN111_EF_CONFIG_D, AIN112_EF_CONFIG_D, AIN113_EF_CONFIG_D,	10422, 10424, 10426,
AIN114_EF_CONFIG_D, AIN115_EF_CONFIG_D, AIN116_EF_CONFIG_D,	10428, 10430, 10432,
AIN117_EF_CONFIG_D, AIN118_EF_CONFIG_D, AIN119_EF_CONFIG_D,	10434, 10436, 10438,
AIN120_EF_CONFIG_D, AIN121_EF_CONFIG_D, AIN122_EF_CONFIG_D,	10440, 10442, 10444,
AIN123_EF_CONFIG_D, AIN124_EF_CONFIG_D, AIN125_EF_CONFIG_D,	10446, 10448, 10450,
AIN126_EF_CONFIG_D, AIN127_EF_CONFIG_D, AIN128_EF_CONFIG_D,	10452, 10454, 10456,
AIN129_EF_CONFIG_D, AIN130_EF_CONFIG_D, AIN131_EF_CONFIG_D,	10458, 10460, 10462,
AIN132_EF_CONFIG_D, AIN133_EF_CONFIG_D, AIN134_EF_CONFIG_D,	10464, 10466, 10468,
AIN135_EF_CONFIG_D, AIN136_EF_CONFIG_D, AIN137_EF_CONFIG_D,	10470, 10472, 10474,
AIN138_EF_CONFIG_D, AIN139_EF_CONFIG_D, AIN140_EF_CONFIG_D,	10476, 10478, 10480,
AIN141_EF_CONFIG_D, AIN142_EF_CONFIG_D, AIN143_EF_CONFIG_D,	10482, 10484, 10486,
AIN144_EF_CONFIG_D, AIN145_EF_CONFIG_D, AIN146_EF_CONFIG_D,	10488, 10490, 10492,
AIN147_EF_CONFIG_D, AIN148_EF_CONFIG_D	10494, 10496

AIN#(0:148)_EF_CONFIG_E

- Starting Address: 10500

  Function dependent on selected feature index.

  - Data type: FLOAT32 (type index = 3)
  - Readable and writable
  - Default value: 0
  - T8:
    - Minimum firmware version: 0.0123
  - T7:
    - Minimum firmware version: 1.0030

Expanded Names	Addresses

AIN0_EF_CONFIG_E, AIN1_EF_CONFIG_E, AIN2_EF_CONFIG_E,	10500, 10502, 10504,
AIN3_EF_CONFIG_E, AIN4_EF_CONFIG_E, AIN5_EF_CONFIG_E,	10506, 10508, 10510,
AIN6_EF_CONFIG_E, AIN7_EF_CONFIG_E, AIN8_EF_CONFIG_E,	10512, 10514, 10516,
AIN9_EF_CONFIG_E, AIN10_EF_CONFIG_E, AIN11_EF_CONFIG_E,	10518, 10520, 10522,
AIN12_EF_CONFIG_E, AIN13_EF_CONFIG_E, AIN14_EF_CONFIG_E,	10524, 10526, 10528,
AIN15_EF_CONFIG_E, AIN16_EF_CONFIG_E, AIN17_EF_CONFIG_E,	10530, 10532, 10534,
AIN18_EF_CONFIG_E, AIN19_EF_CONFIG_E, AIN20_EF_CONFIG_E,	10536, 10538, 10540,
AIN21_EF_CONFIG_E, AIN22_EF_CONFIG_E, AIN23_EF_CONFIG_E,	10542, 10544, 10546,
AIN24_EF_CONFIG_E, AIN25_EF_CONFIG_E, AIN26_EF_CONFIG_E,	10548, 10550, 10552,
AIN27_EF_CONFIG_E, AIN28_EF_CONFIG_E, AIN29_EF_CONFIG_E,	10554, 10556, 10558,
AIN30_EF_CONFIG_E, AIN31_EF_CONFIG_E, AIN32_EF_CONFIG_E,	10560, 10562, 10564,
AIN33_EF_CONFIG_E, AIN34_EF_CONFIG_E, AIN35_EF_CONFIG_E,	10566, 10568, 10570,
AIN36_EF_CONFIG_E, AIN37_EF_CONFIG_E, AIN38_EF_CONFIG_E,	10572, 10574, 10576,
AIN39_EF_CONFIG_E, AIN40_EF_CONFIG_E, AIN41_EF_CONFIG_E,	10578, 10580, 10582,
AIN42_EF_CONFIG_E, AIN43_EF_CONFIG_E, AIN44_EF_CONFIG_E,	10584, 10586, 10588,
AIN45_EF_CONFIG_E, AIN46_EF_CONFIG_E, AIN47_EF_CONFIG_E,	10590, 10592, 10594,
AIN48_EF_CONFIG_E, AIN49_EF_CONFIG_E, AIN50_EF_CONFIG_E,	10596, 10598, 10600,
AIN51_EF_CONFIG_E, AIN52_EF_CONFIG_E, AIN53_EF_CONFIG_E,	10602, 10604, 10606,
AIN54_EF_CONFIG_E, AIN55_EF_CONFIG_E, AIN56_EF_CONFIG_E,	10608, 10610, 10612,
AIN57_EF_CONFIG_E, AIN58_EF_CONFIG_E, AIN59_EF_CONFIG_E,	10614, 10616, 10618,
AIN60_EF_CONFIG_E, AIN61_EF_CONFIG_E, AIN62_EF_CONFIG_E,	10620, 10622, 10624,
AIN63_EF_CONFIG_E, AIN64_EF_CONFIG_E, AIN65_EF_CONFIG_E,	10626, 10628, 10630,
AIN66_EF_CONFIG_E, AIN67_EF_CONFIG_E, AIN68_EF_CONFIG_E,	10632, 10634, 10636,
AIN69_EF_CONFIG_E, AIN70_EF_CONFIG_E, AIN71_EF_CONFIG_E,	10638, 10640, 10642,
AIN72_EF_CONFIG_E, AIN73_EF_CONFIG_E, AIN74_EF_CONFIG_E,	10644, 10646, 10648,
AIN75_EF_CONFIG_E, AIN76_EF_CONFIG_E, AIN77_EF_CONFIG_E,	10650, 10652, 10654,
AIN78_EF_CONFIG_E, AIN79_EF_CONFIG_E, AIN80_EF_CONFIG_E,	10656, 10658, 10660,
AIN81_EF_CONFIG_E, AIN82_EF_CONFIG_E, AIN83_EF_CONFIG_E,	10662, 10664, 10666,
AIN84_EF_CONFIG_E, AIN85_EF_CONFIG_E, AIN86_EF_CONFIG_E,	10668, 10670, 10672,
AIN87_EF_CONFIG_E, AIN88_EF_CONFIG_E, AIN89_EF_CONFIG_E,	10674, 10676, 10678,
AIN90_EF_CONFIG_E, AIN91_EF_CONFIG_E, AIN92_EF_CONFIG_E,	10680, 10682, 10684,
AIN93_EF_CONFIG_E, AIN94_EF_CONFIG_E, AIN95_EF_CONFIG_E,	10686, 10688, 10690,
AIN96_EF_CONFIG_E, AIN97_EF_CONFIG_E, AIN98_EF_CONFIG_E,	10692, 10694, 10696,
AIN99_EF_CONFIG_E, AIN100_EF_CONFIG_E, AIN101_EF_CONFIG_E,	10698, 10700, 10702,
AIN102_EF_CONFIG_E, AIN103_EF_CONFIG_E, AIN104_EF_CONFIG_E,	10704, 10706, 10708,
AIN105_EF_CONFIG_E, AIN106_EF_CONFIG_E, AIN107_EF_CONFIG_E,	10710, 10712, 10714,
AIN108_EF_CONFIG_E, AIN109_EF_CONFIG_E, AIN110_EF_CONFIG_E,	10716, 10718, 10720,
AIN111_EF_CONFIG_E, AIN112_EF_CONFIG_E, AIN113_EF_CONFIG_E,	10722, 10724, 10726,
AIN114_EF_CONFIG_E, AIN115_EF_CONFIG_E, AIN116_EF_CONFIG_E,	10728, 10730, 10732,
AIN117_EF_CONFIG_E, AIN118_EF_CONFIG_E, AIN119_EF_CONFIG_E,	10734, 10736, 10738,
AIN120_EF_CONFIG_E, AIN121_EF_CONFIG_E, AIN122_EF_CONFIG_E,	10740, 10742, 10744,
AIN123_EF_CONFIG_E, AIN124_EF_CONFIG_E, AIN125_EF_CONFIG_E,	10746, 10748, 10750,
AIN126_EF_CONFIG_E, AIN127_EF_CONFIG_E, AIN128_EF_CONFIG_E,	10752, 10754, 10756,
AIN129_EF_CONFIG_E, AIN130_EF_CONFIG_E, AIN131_EF_CONFIG_E,	10758, 10760, 10762,
AIN132_EF_CONFIG_E, AIN133_EF_CONFIG_E, AIN134_EF_CONFIG_E,	10764, 10766, 10768,
AIN135_EF_CONFIG_E, AIN136_EF_CONFIG_E, AIN137_EF_CONFIG_E,	10770, 10772, 10774,
AIN138_EF_CONFIG_E, AIN139_EF_CONFIG_E, AIN140_EF_CONFIG_E,	10776, 10778, 10780,
AIN141_EF_CONFIG_E, AIN142_EF_CONFIG_E, AIN143_EF_CONFIG_E,	10782, 10784, 10786,
AIN144_EF_CONFIG_E, AIN145_EF_CONFIG_E, AIN146_EF_CONFIG_E,	10788, 10790, 10792,
AIN147_EF_CONFIG_E, AIN148_EF_CONFIG_E	10794, 10796

AIN#(0:148)_EF_CONFIG_F
- Starting Address: 10800

Function dependent on selected feature index.

- Data type: FLOAT32 (type index = 3)
- Readable and writable
- Default value: 0
- T8:
    - Minimum firmware version: 0.0123
- T7:

- Minimum firmware version: 1.0030

Expanded Names	Addresses
AIN0_EF_CONFIG_F, AIN1_EF_CONFIG_F, AIN2_EF_CONFIG_F,	10800, 10802, 10804,
AIN3_EF_CONFIG_F, AIN4_EF_CONFIG_F, AIN5_EF_CONFIG_F,	10806, 10808, 10810,
AIN6_EF_CONFIG_F, AIN7_EF_CONFIG_F, AIN8_EF_CONFIG_F,	10812, 10814, 10816,
AIN9_EF_CONFIG_F, AIN10_EF_CONFIG_F, AIN11_EF_CONFIG_F,	10818, 10820, 10822,
AIN12_EF_CONFIG_F, AIN13_EF_CONFIG_F, AIN14_EF_CONFIG_F,	10824, 10826, 10828,
AIN15_EF_CONFIG_F, AIN16_EF_CONFIG_F, AIN17_EF_CONFIG_F,	10830, 10832, 10834,
AIN18_EF_CONFIG_F, AIN19_EF_CONFIG_F, AIN20_EF_CONFIG_F,	10836, 10838, 10840,
AIN21_EF_CONFIG_F, AIN22_EF_CONFIG_F, AIN23_EF_CONFIG_F,	10842, 10844, 10846,
AIN24_EF_CONFIG_F, AIN25_EF_CONFIG_F, AIN26_EF_CONFIG_F,	10848, 10850, 10852,
AIN27_EF_CONFIG_F, AIN28_EF_CONFIG_F, AIN29_EF_CONFIG_F,	10854, 10856, 10858,
AIN30_EF_CONFIG_F, AIN31_EF_CONFIG_F, AIN32_EF_CONFIG_F,	10860, 10862, 10864,
AIN33_EF_CONFIG_F, AIN34_EF_CONFIG_F, AIN35_EF_CONFIG_F,	10866, 10868, 10870,
AIN36_EF_CONFIG_F, AIN37_EF_CONFIG_F, AIN38_EF_CONFIG_F,	10872, 10874, 10876,
AIN39_EF_CONFIG_F, AIN40_EF_CONFIG_F, AIN41_EF_CONFIG_F,	10878, 10880, 10882,
AIN42_EF_CONFIG_F, AIN43_EF_CONFIG_F, AIN44_EF_CONFIG_F,	10884, 10886, 10888,
AIN45_EF_CONFIG_F, AIN46_EF_CONFIG_F, AIN47_EF_CONFIG_F,	10890, 10892, 10894,
AIN48_EF_CONFIG_F, AIN49_EF_CONFIG_F, AIN50_EF_CONFIG_F,	10896, 10898, 10900,
AIN51_EF_CONFIG_F, AIN52_EF_CONFIG_F, AIN53_EF_CONFIG_F,	10902, 10904, 10906,
AIN54_EF_CONFIG_F, AIN55_EF_CONFIG_F, AIN56_EF_CONFIG_F,	10908, 10910, 10912,
AIN57_EF_CONFIG_F, AIN58_EF_CONFIG_F, AIN59_EF_CONFIG_F,	10914, 10916, 10918,
AIN60_EF_CONFIG_F, AIN61_EF_CONFIG_F, AIN62_EF_CONFIG_F,	10920, 10922, 10924,
AIN63_EF_CONFIG_F, AIN64_EF_CONFIG_F, AIN65_EF_CONFIG_F,	10926, 10928, 10930,
AIN66_EF_CONFIG_F, AIN67_EF_CONFIG_F, AIN68_EF_CONFIG_F,	10932, 10934, 10936,
AIN69_EF_CONFIG_F, AIN70_EF_CONFIG_F, AIN71_EF_CONFIG_F,	10938, 10940, 10942,
AIN72_EF_CONFIG_F, AIN73_EF_CONFIG_F, AIN74_EF_CONFIG_F,	10944, 10946, 10948,
AIN75_EF_CONFIG_F, AIN76_EF_CONFIG_F, AIN77_EF_CONFIG_F,	10950, 10952, 10954,
AIN78_EF_CONFIG_F, AIN79_EF_CONFIG_F, AIN80_EF_CONFIG_F,	10956, 10958, 10960,
AIN81_EF_CONFIG_F, AIN82_EF_CONFIG_F, AIN83_EF_CONFIG_F,	10962, 10964, 10966,
AIN84_EF_CONFIG_F, AIN85_EF_CONFIG_F, AIN86_EF_CONFIG_F,	10968, 10970, 10972,
AIN87_EF_CONFIG_F, AIN88_EF_CONFIG_F, AIN89_EF_CONFIG_F,	10974, 10976, 10978,
AIN90_EF_CONFIG_F, AIN91_EF_CONFIG_F, AIN92_EF_CONFIG_F,	10980, 10982, 10984,
AIN93_EF_CONFIG_F, AIN94_EF_CONFIG_F, AIN95_EF_CONFIG_F,	10986, 10988, 10990,
AIN96_EF_CONFIG_F, AIN97_EF_CONFIG_F, AIN98_EF_CONFIG_F,	10992, 10994, 10996,
AIN99_EF_CONFIG_F, AIN100_EF_CONFIG_F, AIN101_EF_CONFIG_F,	10998, 11000, 11002,
AIN102_EF_CONFIG_F, AIN103_EF_CONFIG_F, AIN104_EF_CONFIG_F,	11004, 11006, 11008,
AIN105_EF_CONFIG_F, AIN106_EF_CONFIG_F, AIN107_EF_CONFIG_F,	11010, 11012, 11014,
AIN108_EF_CONFIG_F, AIN109_EF_CONFIG_F, AIN110_EF_CONFIG_F,	11016, 11018, 11020,
AIN111_EF_CONFIG_F, AIN112_EF_CONFIG_F, AIN113_EF_CONFIG_F,	11022, 11024, 11026,
AIN114_EF_CONFIG_F, AIN115_EF_CONFIG_F, AIN116_EF_CONFIG_F,	11028, 11030, 11032,
AIN117_EF_CONFIG_F, AIN118_EF_CONFIG_F, AIN119_EF_CONFIG_F,	11034, 11036, 11038,
AIN120_EF_CONFIG_F, AIN121_EF_CONFIG_F, AIN122_EF_CONFIG_F,	11040, 11042, 11044,
AIN123_EF_CONFIG_F, AIN124_EF_CONFIG_F, AIN125_EF_CONFIG_F,	11046, 11048, 11050,
AIN126_EF_CONFIG_F, AIN127_EF_CONFIG_F, AIN128_EF_CONFIG_F,	11052, 11054, 11056,
AIN129_EF_CONFIG_F, AIN130_EF_CONFIG_F, AIN131_EF_CONFIG_F,	11058, 11060, 11062,
AIN132_EF_CONFIG_F, AIN133_EF_CONFIG_F, AIN134_EF_CONFIG_F,	11064, 11066, 11068,
AIN135_EF_CONFIG_F, AIN136_EF_CONFIG_F, AIN137_EF_CONFIG_F,	11070, 11072, 11074,
AIN138_EF_CONFIG_F, AIN139_EF_CONFIG_F, AIN140_EF_CONFIG_F,	11076, 11078, 11080,
AIN141_EF_CONFIG_F, AIN142_EF_CONFIG_F, AIN143_EF_CONFIG_F,	11082, 11084, 11086,
AIN144_EF_CONFIG_F, AIN145_EF_CONFIG_F, AIN146_EF_CONFIG_F,	11088, 11090, 11092,
AIN147_EF_CONFIG_F, AIN148_EF_CONFIG_F	11094, 11096

AIN#(0:148)_EF_CONFIG_G
- Starting Address: 11100

Function dependent on selected feature index.

- Data type: FLOAT32 (type index = 3)
- Readable and writable
- Default value: 0

- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0030

Expanded Names	Addresses
AIN0_EF_CONFIG_G, AIN1_EF_CONFIG_G, AIN2_EF_CONFIG_G,	11100, 11102, 11104,
AIN3_EF_CONFIG_G, AIN4_EF_CONFIG_G, AIN5_EF_CONFIG_G,	11106, 11108, 11110,
AIN6_EF_CONFIG_G, AIN7_EF_CONFIG_G, AIN8_EF_CONFIG_G,	11112, 11114, 11116,
AIN9_EF_CONFIG_G, AIN10_EF_CONFIG_G, AIN11_EF_CONFIG_G,	11118, 11120, 11122,
AIN12_EF_CONFIG_G, AIN13_EF_CONFIG_G, AIN14_EF_CONFIG_G,	11124, 11126, 11128,
AIN15_EF_CONFIG_G, AIN16_EF_CONFIG_G, AIN17_EF_CONFIG_G,	11130, 11132, 11134,
AIN18_EF_CONFIG_G, AIN19_EF_CONFIG_G, AIN20_EF_CONFIG_G,	11136, 11138, 11140,
AIN21_EF_CONFIG_G, AIN22_EF_CONFIG_G, AIN23_EF_CONFIG_G,	11142, 11144, 11146,
AIN24_EF_CONFIG_G, AIN25_EF_CONFIG_G, AIN26_EF_CONFIG_G,	11148, 11150, 11152,
AIN27_EF_CONFIG_G, AIN28_EF_CONFIG_G, AIN29_EF_CONFIG_G,	11154, 11156, 11158,
AIN30_EF_CONFIG_G, AIN31_EF_CONFIG_G, AIN32_EF_CONFIG_G,	11160, 11162, 11164,
AIN33_EF_CONFIG_G, AIN34_EF_CONFIG_G, AIN35_EF_CONFIG_G,	11166, 11168, 11170,
AIN36_EF_CONFIG_G, AIN37_EF_CONFIG_G, AIN38_EF_CONFIG_G,	11172, 11174, 11176,
AIN39_EF_CONFIG_G, AIN40_EF_CONFIG_G, AIN41_EF_CONFIG_G,	11178, 11180, 11182,
AIN42_EF_CONFIG_G, AIN43_EF_CONFIG_G, AIN44_EF_CONFIG_G,	11184, 11186, 11188,
AIN45_EF_CONFIG_G, AIN46_EF_CONFIG_G, AIN47_EF_CONFIG_G,	11190, 11192, 11194,
AIN48_EF_CONFIG_G, AIN49_EF_CONFIG_G, AIN50_EF_CONFIG_G,	11196, 11198, 11200,
AIN51_EF_CONFIG_G, AIN52_EF_CONFIG_G, AIN53_EF_CONFIG_G,	11202, 11204, 11206,
AIN54_EF_CONFIG_G, AIN55_EF_CONFIG_G, AIN56_EF_CONFIG_G,	11208, 11210, 11212,
AIN57_EF_CONFIG_G, AIN58_EF_CONFIG_G, AIN59_EF_CONFIG_G,	11214, 11216, 11218,
AIN60_EF_CONFIG_G, AIN61_EF_CONFIG_G, AIN62_EF_CONFIG_G,	11220, 11222, 11224,
AIN63_EF_CONFIG_G, AIN64_EF_CONFIG_G, AIN65_EF_CONFIG_G,	11226, 11228, 11230,
AIN66_EF_CONFIG_G, AIN67_EF_CONFIG_G, AIN68_EF_CONFIG_G,	11232, 11234, 11236,
AIN69_EF_CONFIG_G, AIN70_EF_CONFIG_G, AIN71_EF_CONFIG_G,	11238, 11240, 11242,
AIN72_EF_CONFIG_G, AIN73_EF_CONFIG_G, AIN74_EF_CONFIG_G,	11244, 11246, 11248,
AIN75_EF_CONFIG_G, AIN76_EF_CONFIG_G, AIN77_EF_CONFIG_G,	11250, 11252, 11254,
AIN78_EF_CONFIG_G, AIN79_EF_CONFIG_G, AIN80_EF_CONFIG_G,	11256, 11258, 11260,
AIN81_EF_CONFIG_G, AIN82_EF_CONFIG_G, AIN83_EF_CONFIG_G,	11262, 11264, 11266,
AIN84_EF_CONFIG_G, AIN85_EF_CONFIG_G, AIN86_EF_CONFIG_G,	11268, 11270, 11272,
AIN87_EF_CONFIG_G, AIN88_EF_CONFIG_G, AIN89_EF_CONFIG_G,	11274, 11276, 11278,
AIN90_EF_CONFIG_G, AIN91_EF_CONFIG_G, AIN92_EF_CONFIG_G,	11280, 11282, 11284,
AIN93_EF_CONFIG_G, AIN94_EF_CONFIG_G, AIN95_EF_CONFIG_G,	11286, 11288, 11290,
AIN96_EF_CONFIG_G, AIN97_EF_CONFIG_G, AIN98_EF_CONFIG_G,	11292, 11294, 11296,
AIN99_EF_CONFIG_G, AIN100_EF_CONFIG_G, AIN101_EF_CONFIG_G,	11298, 11300, 11302,
AIN102_EF_CONFIG_G, AIN103_EF_CONFIG_G, AIN104_EF_CONFIG_G,	11304, 11306, 11308,
AIN105_EF_CONFIG_G, AIN106_EF_CONFIG_G, AIN107_EF_CONFIG_G,	11310, 11312, 11314,
AIN108_EF_CONFIG_G, AIN109_EF_CONFIG_G, AIN110_EF_CONFIG_G,	11316, 11318, 11320,
AIN111_EF_CONFIG_G, AIN112_EF_CONFIG_G, AIN113_EF_CONFIG_G,	11322, 11324, 11326,
AIN114_EF_CONFIG_G, AIN115_EF_CONFIG_G, AIN116_EF_CONFIG_G,	11328, 11330, 11332,
AIN117_EF_CONFIG_G, AIN118_EF_CONFIG_G, AIN119_EF_CONFIG_G,	11334, 11336, 11338,
AIN120_EF_CONFIG_G, AIN121_EF_CONFIG_G, AIN122_EF_CONFIG_G,	11340, 11342, 11344,
AIN123_EF_CONFIG_G, AIN124_EF_CONFIG_G, AIN125_EF_CONFIG_G,	11346, 11348, 11350,
AIN126_EF_CONFIG_G, AIN127_EF_CONFIG_G, AIN128_EF_CONFIG_G,	11352, 11354, 11356,
AIN129_EF_CONFIG_G, AIN130_EF_CONFIG_G, AIN131_EF_CONFIG_G,	11358, 11360, 11362,
AIN132_EF_CONFIG_G, AIN133_EF_CONFIG_G, AIN134_EF_CONFIG_G,	11364, 11366, 11368,
AIN135_EF_CONFIG_G, AIN136_EF_CONFIG_G, AIN137_EF_CONFIG_G,	11370, 11372, 11374,
AIN138_EF_CONFIG_G, AIN139_EF_CONFIG_G, AIN140_EF_CONFIG_G,	11376, 11378, 11380,
AIN141_EF_CONFIG_G, AIN142_EF_CONFIG_G, AIN143_EF_CONFIG_G,	11382, 11384, 11386,
AIN144_EF_CONFIG_G, AIN145_EF_CONFIG_G, AIN146_EF_CONFIG_G,	11388, 11390, 11392,
AIN147_EF_CONFIG_G, AIN148_EF_CONFIG_G	11394, 11396

AIN#(0:148)_EF_CONFIG_H
- Starting Address: 11400

Function dependent on selected feature index.

- Data type: FLOAT32 (type index = 3)
- Readable and writable
- Default value: 0
- T8:
    - Minimum firmware version: 0.0123
- T7:
    - Minimum firmware version: 1.0030

Expanded Names	Addresses
AIN0_EF_CONFIG_H, AIN1_EF_CONFIG_H, AIN2_EF_CONFIG_H,	11400, 11402, 11404,
AIN3_EF_CONFIG_H, AIN4_EF_CONFIG_H, AIN5_EF_CONFIG_H,	11406, 11408, 11410,
AIN6_EF_CONFIG_H, AIN7_EF_CONFIG_H, AIN8_EF_CONFIG_H,	11412, 11414, 11416,
AIN9_EF_CONFIG_H, AIN10_EF_CONFIG_H, AIN11_EF_CONFIG_H,	11418, 11420, 11422,
AIN12_EF_CONFIG_H, AIN13_EF_CONFIG_H, AIN14_EF_CONFIG_H,	11424, 11426, 11428,
AIN15_EF_CONFIG_H, AIN16_EF_CONFIG_H, AIN17_EF_CONFIG_H,	11430, 11432, 11434,
AIN18_EF_CONFIG_H, AIN19_EF_CONFIG_H, AIN20_EF_CONFIG_H,	11436, 11438, 11440,
AIN21_EF_CONFIG_H, AIN22_EF_CONFIG_H, AIN23_EF_CONFIG_H,	11442, 11444, 11446,
AIN24_EF_CONFIG_H, AIN25_EF_CONFIG_H, AIN26_EF_CONFIG_H,	11448, 11450, 11452,
AIN27_EF_CONFIG_H, AIN28_EF_CONFIG_H, AIN29_EF_CONFIG_H,	11454, 11456, 11458,
AIN30_EF_CONFIG_H, AIN31_EF_CONFIG_H, AIN32_EF_CONFIG_H,	11460, 11462, 11464,
AIN33_EF_CONFIG_H, AIN34_EF_CONFIG_H, AIN35_EF_CONFIG_H,	11466, 11468, 11470,
AIN36_EF_CONFIG_H, AIN37_EF_CONFIG_H, AIN38_EF_CONFIG_H,	11472, 11474, 11476,
AIN39_EF_CONFIG_H, AIN40_EF_CONFIG_H, AIN41_EF_CONFIG_H,	11478, 11480, 11482,
AIN42_EF_CONFIG_H, AIN43_EF_CONFIG_H, AIN44_EF_CONFIG_H,	11484, 11486, 11488,
AIN45_EF_CONFIG_H, AIN46_EF_CONFIG_H, AIN47_EF_CONFIG_H,	11490, 11492, 11494,
AIN48_EF_CONFIG_H, AIN49_EF_CONFIG_H, AIN50_EF_CONFIG_H,	11496, 11498, 11500,
AIN51_EF_CONFIG_H, AIN52_EF_CONFIG_H, AIN53_EF_CONFIG_H,	11502, 11504, 11506,
AIN54_EF_CONFIG_H, AIN55_EF_CONFIG_H, AIN56_EF_CONFIG_H,	11508, 11510, 11512,
AIN57_EF_CONFIG_H, AIN58_EF_CONFIG_H, AIN59_EF_CONFIG_H,	11514, 11516, 11518,
AIN60_EF_CONFIG_H, AIN61_EF_CONFIG_H, AIN62_EF_CONFIG_H,	11520, 11522, 11524,
AIN63_EF_CONFIG_H, AIN64_EF_CONFIG_H, AIN65_EF_CONFIG_H,	11526, 11528, 11530,
AIN66_EF_CONFIG_H, AIN67_EF_CONFIG_H, AIN68_EF_CONFIG_H,	11532, 11534, 11536,
AIN69_EF_CONFIG_H, AIN70_EF_CONFIG_H, AIN71_EF_CONFIG_H,	11538, 11540, 11542,
AIN72_EF_CONFIG_H, AIN73_EF_CONFIG_H, AIN74_EF_CONFIG_H,	11544, 11546, 11548,
AIN75_EF_CONFIG_H, AIN76_EF_CONFIG_H, AIN77_EF_CONFIG_H,	11550, 11552, 11554,
AIN78_EF_CONFIG_H, AIN79_EF_CONFIG_H, AIN80_EF_CONFIG_H,	11556, 11558, 11560,
AIN81_EF_CONFIG_H, AIN82_EF_CONFIG_H, AIN83_EF_CONFIG_H,	11562, 11564, 11566,
AIN84_EF_CONFIG_H, AIN85_EF_CONFIG_H, AIN86_EF_CONFIG_H,	11568, 11570, 11572,
AIN87_EF_CONFIG_H, AIN88_EF_CONFIG_H, AIN89_EF_CONFIG_H,	11574, 11576, 11578,
AIN90_EF_CONFIG_H, AIN91_EF_CONFIG_H, AIN92_EF_CONFIG_H,	11580, 11582, 11584,
AIN93_EF_CONFIG_H, AIN94_EF_CONFIG_H, AIN95_EF_CONFIG_H,	11586, 11588, 11590,
AIN96_EF_CONFIG_H, AIN97_EF_CONFIG_H, AIN98_EF_CONFIG_H,	11592, 11594, 11596,
AIN99_EF_CONFIG_H, AIN100_EF_CONFIG_H, AIN101_EF_CONFIG_H,	11598, 11600, 11602,
AIN102_EF_CONFIG_H, AIN103_EF_CONFIG_H, AIN104_EF_CONFIG_H,	11604, 11606, 11608,
AIN105_EF_CONFIG_H, AIN106_EF_CONFIG_H, AIN107_EF_CONFIG_H,	11610, 11612, 11614,
AIN108_EF_CONFIG_H, AIN109_EF_CONFIG_H, AIN110_EF_CONFIG_H,	11616, 11618, 11620,
AIN111_EF_CONFIG_H, AIN112_EF_CONFIG_H, AIN113_EF_CONFIG_H,	11622, 11624, 11626,
AIN114_EF_CONFIG_H, AIN115_EF_CONFIG_H, AIN116_EF_CONFIG_H,	11628, 11630, 11632,
AIN117_EF_CONFIG_H, AIN118_EF_CONFIG_H, AIN119_EF_CONFIG_H,	11634, 11636, 11638,
AIN120_EF_CONFIG_H, AIN121_EF_CONFIG_H, AIN122_EF_CONFIG_H,	11640, 11642, 11644,
AIN123_EF_CONFIG_H, AIN124_EF_CONFIG_H, AIN125_EF_CONFIG_H,	11646, 11648, 11650,
AIN126_EF_CONFIG_H, AIN127_EF_CONFIG_H, AIN128_EF_CONFIG_H,	11652, 11654, 11656,
AIN129_EF_CONFIG_H, AIN130_EF_CONFIG_H, AIN131_EF_CONFIG_H,	11658, 11660, 11662,
AIN132_EF_CONFIG_H, AIN133_EF_CONFIG_H, AIN134_EF_CONFIG_H,	11664, 11666, 11668,
AIN135_EF_CONFIG_H, AIN136_EF_CONFIG_H, AIN137_EF_CONFIG_H,	11670, 11672, 11674,
AIN138_EF_CONFIG_H, AIN139_EF_CONFIG_H, AIN140_EF_CONFIG_H,	11676, 11678, 11680,
AIN141_EF_CONFIG_H, AIN142_EF_CONFIG_H, AIN143_EF_CONFIG_H,	11682, 11684, 11686,
AIN144_EF_CONFIG_H, AIN145_EF_CONFIG_H, AIN146_EF_CONFIG_H,	11688, 11690, 11692,
AIN147_EF_CONFIG_H, AIN148_EF_CONFIG_H	11694, 11696

AIN#(0:148)_EF_CONFIG_I

- Starting Address: 11700

  Function dependent on selected feature index.

  - Data type: FLOAT32 (type index = 3)
  - Readable and writable
  - Default value: 0
  - T8:
    - Minimum firmware version: 0.0123
  - T7:
    - Minimum firmware version: 1.0030

Expanded Names	Addresses

AIN0_EF_CONFIG_I, AIN1_EF_CONFIG_I, AIN2_EF_CONFIG_I,	11700, 11702, 11704,
AIN3_EF_CONFIG_I, AIN4_EF_CONFIG_I, AIN5_EF_CONFIG_I,	11706, 11708, 11710,
AIN6_EF_CONFIG_I, AIN7_EF_CONFIG_I, AIN8_EF_CONFIG_I,	11712, 11714, 11716,
AIN9_EF_CONFIG_I, AIN10_EF_CONFIG_I, AIN11_EF_CONFIG_I,	11718, 11720, 11722,
AIN12_EF_CONFIG_I, AIN13_EF_CONFIG_I, AIN14_EF_CONFIG_I,	11724, 11726, 11728,
AIN15_EF_CONFIG_I, AIN16_EF_CONFIG_I, AIN17_EF_CONFIG_I,	11730, 11732, 11734,
AIN18_EF_CONFIG_I, AIN19_EF_CONFIG_I, AIN20_EF_CONFIG_I,	11736, 11738, 11740,
AIN21_EF_CONFIG_I, AIN22_EF_CONFIG_I, AIN23_EF_CONFIG_I,	11742, 11744, 11746,
AIN24_EF_CONFIG_I, AIN25_EF_CONFIG_I, AIN26_EF_CONFIG_I,	11748, 11750, 11752,
AIN27_EF_CONFIG_I, AIN28_EF_CONFIG_I, AIN29_EF_CONFIG_I,	11754, 11756, 11758,
AIN30_EF_CONFIG_I, AIN31_EF_CONFIG_I, AIN32_EF_CONFIG_I,	11760, 11762, 11764,
AIN33_EF_CONFIG_I, AIN34_EF_CONFIG_I, AIN35_EF_CONFIG_I,	11766, 11768, 11770,
AIN36_EF_CONFIG_I, AIN37_EF_CONFIG_I, AIN38_EF_CONFIG_I,	11772, 11774, 11776,
AIN39_EF_CONFIG_I, AIN40_EF_CONFIG_I, AIN41_EF_CONFIG_I,	11778, 11780, 11782,
AIN42_EF_CONFIG_I, AIN43_EF_CONFIG_I, AIN44_EF_CONFIG_I,	11784, 11786, 11788,
AIN45_EF_CONFIG_I, AIN46_EF_CONFIG_I, AIN47_EF_CONFIG_I,	11790, 11792, 11794,
AIN48_EF_CONFIG_I, AIN49_EF_CONFIG_I, AIN50_EF_CONFIG_I,	11796, 11798, 11800,
AIN51_EF_CONFIG_I, AIN52_EF_CONFIG_I, AIN53_EF_CONFIG_I,	11802, 11804, 11806,
AIN54_EF_CONFIG_I, AIN55_EF_CONFIG_I, AIN56_EF_CONFIG_I,	11808, 11810, 11812,
AIN57_EF_CONFIG_I, AIN58_EF_CONFIG_I, AIN59_EF_CONFIG_I,	11814, 11816, 11818,
AIN60_EF_CONFIG_I, AIN61_EF_CONFIG_I, AIN62_EF_CONFIG_I,	11820, 11822, 11824,
AIN63_EF_CONFIG_I, AIN64_EF_CONFIG_I, AIN65_EF_CONFIG_I,	11826, 11828, 11830,
AIN66_EF_CONFIG_I, AIN67_EF_CONFIG_I, AIN68_EF_CONFIG_I,	11832, 11834, 11836,
AIN69_EF_CONFIG_I, AIN70_EF_CONFIG_I, AIN71_EF_CONFIG_I,	11838, 11840, 11842,
AIN72_EF_CONFIG_I, AIN73_EF_CONFIG_I, AIN74_EF_CONFIG_I,	11844, 11846, 11848,
AIN75_EF_CONFIG_I, AIN76_EF_CONFIG_I, AIN77_EF_CONFIG_I,	11850, 11852, 11854,
AIN78_EF_CONFIG_I, AIN79_EF_CONFIG_I, AIN80_EF_CONFIG_I,	11856, 11858, 11860,
AIN81_EF_CONFIG_I, AIN82_EF_CONFIG_I, AIN83_EF_CONFIG_I,	11862, 11864, 11866,
AIN84_EF_CONFIG_I, AIN85_EF_CONFIG_I, AIN86_EF_CONFIG_I,	11868, 11870, 11872,
AIN87_EF_CONFIG_I, AIN88_EF_CONFIG_I, AIN89_EF_CONFIG_I,	11874, 11876, 11878,
AIN90_EF_CONFIG_I, AIN91_EF_CONFIG_I, AIN92_EF_CONFIG_I,	11880, 11882, 11884,
AIN93_EF_CONFIG_I, AIN94_EF_CONFIG_I, AIN95_EF_CONFIG_I,	11886, 11888, 11890,
AIN96_EF_CONFIG_I, AIN97_EF_CONFIG_I, AIN98_EF_CONFIG_I,	11892, 11894, 11896,
AIN99_EF_CONFIG_I, AIN100_EF_CONFIG_I, AIN101_EF_CONFIG_I,	11898, 11900, 11902,
AIN102_EF_CONFIG_I, AIN103_EF_CONFIG_I, AIN104_EF_CONFIG_I,	11904, 11906, 11908,
AIN105_EF_CONFIG_I, AIN106_EF_CONFIG_I, AIN107_EF_CONFIG_I,	11910, 11912, 11914,
AIN108_EF_CONFIG_I, AIN109_EF_CONFIG_I, AIN110_EF_CONFIG_I,	11916, 11918, 11920,
AIN111_EF_CONFIG_I, AIN112_EF_CONFIG_I, AIN113_EF_CONFIG_I,	11922, 11924, 11926,
AIN114_EF_CONFIG_I, AIN115_EF_CONFIG_I, AIN116_EF_CONFIG_I,	11928, 11930, 11932,
AIN117_EF_CONFIG_I, AIN118_EF_CONFIG_I, AIN119_EF_CONFIG_I,	11934, 11936, 11938,
AIN120_EF_CONFIG_I, AIN121_EF_CONFIG_I, AIN122_EF_CONFIG_I,	11940, 11942, 11944,
AIN123_EF_CONFIG_I, AIN124_EF_CONFIG_I, AIN125_EF_CONFIG_I,	11946, 11948, 11950,
AIN126_EF_CONFIG_I, AIN127_EF_CONFIG_I, AIN128_EF_CONFIG_I,	11952, 11954, 11956,
AIN129_EF_CONFIG_I, AIN130_EF_CONFIG_I, AIN131_EF_CONFIG_I,	11958, 11960, 11962,
AIN132_EF_CONFIG_I, AIN133_EF_CONFIG_I, AIN134_EF_CONFIG_I,	11964, 11966, 11968,
AIN135_EF_CONFIG_I, AIN136_EF_CONFIG_I, AIN137_EF_CONFIG_I,	11970, 11972, 11974,
AIN138_EF_CONFIG_I, AIN139_EF_CONFIG_I, AIN140_EF_CONFIG_I,	11976, 11978, 11980,
AIN141_EF_CONFIG_I, AIN142_EF_CONFIG_I, AIN143_EF_CONFIG_I,	11982, 11984, 11986,
AIN144_EF_CONFIG_I, AIN145_EF_CONFIG_I, AIN146_EF_CONFIG_I,	11988, 11990, 11992,
AIN147_EF_CONFIG_I, AIN148_EF_CONFIG_I	11994, 11996

AIN#(0:148)_EF_CONFIG_J
- Starting Address: 12000

Function dependent on selected feature index.

- Data type: FLOAT32 (type index = 3)
- Readable and writable
- Default value: 0
- T8:
  - Minimum firmware version: 0.0123
- T7:

- Minimum firmware version: 1.0030

Expanded Names	Addresses
AIN0_EF_CONFIG_J, AIN1_EF_CONFIG_J, AIN2_EF_CONFIG_J,	12000, 12002, 12004,
AIN3_EF_CONFIG_J, AIN4_EF_CONFIG_J, AIN5_EF_CONFIG_J,	12006, 12008, 12010,
AIN6_EF_CONFIG_J, AIN7_EF_CONFIG_J, AIN8_EF_CONFIG_J,	12012, 12014, 12016,
AIN9_EF_CONFIG_J, AIN10_EF_CONFIG_J, AIN11_EF_CONFIG_J,	12018, 12020, 12022,
AIN12_EF_CONFIG_J, AIN13_EF_CONFIG_J, AIN14_EF_CONFIG_J,	12024, 12026, 12028,
AIN15_EF_CONFIG_J, AIN16_EF_CONFIG_J, AIN17_EF_CONFIG_J,	12030, 12032, 12034,
AIN18_EF_CONFIG_J, AIN19_EF_CONFIG_J, AIN20_EF_CONFIG_J,	12036, 12038, 12040,
AIN21_EF_CONFIG_J, AIN22_EF_CONFIG_J, AIN23_EF_CONFIG_J,	12042, 12044, 12046,
AIN24_EF_CONFIG_J, AIN25_EF_CONFIG_J, AIN26_EF_CONFIG_J,	12048, 12050, 12052,
AIN27_EF_CONFIG_J, AIN28_EF_CONFIG_J, AIN29_EF_CONFIG_J,	12054, 12056, 12058,
AIN30_EF_CONFIG_J, AIN31_EF_CONFIG_J, AIN32_EF_CONFIG_J,	12060, 12062, 12064,
AIN33_EF_CONFIG_J, AIN34_EF_CONFIG_J, AIN35_EF_CONFIG_J,	12066, 12068, 12070,
AIN36_EF_CONFIG_J, AIN37_EF_CONFIG_J, AIN38_EF_CONFIG_J,	12072, 12074, 12076,
AIN39_EF_CONFIG_J, AIN40_EF_CONFIG_J, AIN41_EF_CONFIG_J,	12078, 12080, 12082,
AIN42_EF_CONFIG_J, AIN43_EF_CONFIG_J, AIN44_EF_CONFIG_J,	12084, 12086, 12088,
AIN45_EF_CONFIG_J, AIN46_EF_CONFIG_J, AIN47_EF_CONFIG_J,	12090, 12092, 12094,
AIN48_EF_CONFIG_J, AIN49_EF_CONFIG_J, AIN50_EF_CONFIG_J,	12096, 12098, 12100,
AIN51_EF_CONFIG_J, AIN52_EF_CONFIG_J, AIN53_EF_CONFIG_J,	12102, 12104, 12106,
AIN54_EF_CONFIG_J, AIN55_EF_CONFIG_J, AIN56_EF_CONFIG_J,	12108, 12110, 12112,
AIN57_EF_CONFIG_J, AIN58_EF_CONFIG_J, AIN59_EF_CONFIG_J,	12114, 12116, 12118,
AIN60_EF_CONFIG_J, AIN61_EF_CONFIG_J, AIN62_EF_CONFIG_J,	12120, 12122, 12124,
AIN63_EF_CONFIG_J, AIN64_EF_CONFIG_J, AIN65_EF_CONFIG_J,	12126, 12128, 12130,
AIN66_EF_CONFIG_J, AIN67_EF_CONFIG_J, AIN68_EF_CONFIG_J,	12132, 12134, 12136,
AIN69_EF_CONFIG_J, AIN70_EF_CONFIG_J, AIN71_EF_CONFIG_J,	12138, 12140, 12142,
AIN72_EF_CONFIG_J, AIN73_EF_CONFIG_J, AIN74_EF_CONFIG_J,	12144, 12146, 12148,
AIN75_EF_CONFIG_J, AIN76_EF_CONFIG_J, AIN77_EF_CONFIG_J,	12150, 12152, 12154,
AIN78_EF_CONFIG_J, AIN79_EF_CONFIG_J, AIN80_EF_CONFIG_J,	12156, 12158, 12160,
AIN81_EF_CONFIG_J, AIN82_EF_CONFIG_J, AIN83_EF_CONFIG_J,	12162, 12164, 12166,
AIN84_EF_CONFIG_J, AIN85_EF_CONFIG_J, AIN86_EF_CONFIG_J,	12168, 12170, 12172,
AIN87_EF_CONFIG_J, AIN88_EF_CONFIG_J, AIN89_EF_CONFIG_J,	12174, 12176, 12178,
AIN90_EF_CONFIG_J, AIN91_EF_CONFIG_J, AIN92_EF_CONFIG_J,	12180, 12182, 12184,
AIN93_EF_CONFIG_J, AIN94_EF_CONFIG_J, AIN95_EF_CONFIG_J,	12186, 12188, 12190,
AIN96_EF_CONFIG_J, AIN97_EF_CONFIG_J, AIN98_EF_CONFIG_J,	12192, 12194, 12196,
AIN99_EF_CONFIG_J, AIN100_EF_CONFIG_J, AIN101_EF_CONFIG_J,	12198, 12200, 12202,
AIN102_EF_CONFIG_J, AIN103_EF_CONFIG_J, AIN104_EF_CONFIG_J,	12204, 12206, 12208,
AIN105_EF_CONFIG_J, AIN106_EF_CONFIG_J, AIN107_EF_CONFIG_J,	12210, 12212, 12214,
AIN108_EF_CONFIG_J, AIN109_EF_CONFIG_J, AIN110_EF_CONFIG_J,	12216, 12218, 12220,
AIN111_EF_CONFIG_J, AIN112_EF_CONFIG_J, AIN113_EF_CONFIG_J,	12222, 12224, 12226,
AIN114_EF_CONFIG_J, AIN115_EF_CONFIG_J, AIN116_EF_CONFIG_J,	12228, 12230, 12232,
AIN117_EF_CONFIG_J, AIN118_EF_CONFIG_J, AIN119_EF_CONFIG_J,	12234, 12236, 12238,
AIN120_EF_CONFIG_J, AIN121_EF_CONFIG_J, AIN122_EF_CONFIG_J,	12240, 12242, 12244,
AIN123_EF_CONFIG_J, AIN124_EF_CONFIG_J, AIN125_EF_CONFIG_J,	12246, 12248, 12250,
AIN126_EF_CONFIG_J, AIN127_EF_CONFIG_J, AIN128_EF_CONFIG_J,	12252, 12254, 12256,
AIN129_EF_CONFIG_J, AIN130_EF_CONFIG_J, AIN131_EF_CONFIG_J,	12258, 12260, 12262,
AIN132_EF_CONFIG_J, AIN133_EF_CONFIG_J, AIN134_EF_CONFIG_J,	12264, 12266, 12268,
AIN135_EF_CONFIG_J, AIN136_EF_CONFIG_J, AIN137_EF_CONFIG_J,	12270, 12272, 12274,
AIN138_EF_CONFIG_J, AIN139_EF_CONFIG_J, AIN140_EF_CONFIG_J,	12276, 12278, 12280,
AIN141_EF_CONFIG_J, AIN142_EF_CONFIG_J, AIN143_EF_CONFIG_J,	12282, 12284, 12286,
AIN144_EF_CONFIG_J, AIN145_EF_CONFIG_J, AIN146_EF_CONFIG_J,	12288, 12290, 12292,
AIN147_EF_CONFIG_J, AIN148_EF_CONFIG_J	12294, 12296

AIN#(0:253)_RANGE
- Starting Address: 40000

The range/span of each analog input. Write the highest expected input voltage.

- Data type: FLOAT32 (type index = 3)
- Readable and writable
- Default value: 0

- T8:
  - Specify the maximum expected voltage. The T8 will select the best gain for the expected voltage.
- T7:
  - Valid values/ranges: 0.0=Default → ±10V. 10.0 → ±10V, 1.0 → ±1V, 0.1 → ±0.1V, and 0.01 → ±0.01V.
- T4:
  - Valid values/ranges: 0.0=Default → 0-2.5 V on LV lines and ±10 V on HV lines.

Expanded Names	Addresses
AIN0_RANGE, AIN1_RANGE, AIN2_RANGE, AIN3_RANGE, AIN4_RANGE, AIN5_RANGE, AIN6_RANGE, AIN7_RANGE, AIN8_RANGE, AIN9_RANGE, AIN10_RANGE, AIN11_RANGE, AIN12_RANGE, AIN13_RANGE, AIN14_RANGE, AIN15_RANGE, AIN16_RANGE, AIN17_RANGE, AIN18_RANGE, AIN19_RANGE, AIN20_RANGE, AIN21_RANGE, AIN22_RANGE, AIN23_RANGE, AIN24_RANGE, AIN25_RANGE, AIN26_RANGE, AIN27_RANGE, AIN28_RANGE, AIN29_RANGE, AIN30_RANGE, AIN31_RANGE, AIN32_RANGE, AIN33_RANGE, AIN34_RANGE, AIN35_RANGE, AIN36_RANGE, AIN37_RANGE, AIN38_RANGE, AIN39_RANGE, AIN40_RANGE, AIN41_RANGE, AIN42_RANGE, AIN43_RANGE, AIN44_RANGE, AIN45_RANGE, AIN46_RANGE, AIN47_RANGE, AIN48_RANGE, AIN49_RANGE, AIN50_RANGE, AIN51_RANGE, AIN52_RANGE, AIN53_RANGE, AIN54_RANGE, AIN55_RANGE, AIN56_RANGE, AIN57_RANGE, AIN58_RANGE, AIN59_RANGE, AIN60_RANGE, AIN61_RANGE, AIN62_RANGE, AIN63_RANGE, AIN64_RANGE, AIN65_RANGE, AIN66_RANGE, AIN67_RANGE, AIN68_RANGE, AIN69_RANGE, AIN70_RANGE, AIN71_RANGE, AIN72_RANGE, AIN73_RANGE, AIN74_RANGE, AIN75_RANGE, AIN76_RANGE, AIN77_RANGE, AIN78_RANGE, AIN79_RANGE, AIN80_RANGE, AIN81_RANGE, AIN82_RANGE, AIN83_RANGE, AIN84_RANGE, AIN85_RANGE, AIN86_RANGE, AIN87_RANGE, AIN88_RANGE, AIN89_RANGE, AIN90_RANGE, AIN91_RANGE, AIN92_RANGE, AIN93_RANGE, AIN94_RANGE, AIN95_RANGE, AIN96_RANGE, AIN97_RANGE, AIN98_RANGE, AIN99_RANGE, AIN100_RANGE, AIN101_RANGE, AIN102_RANGE, AIN103_RANGE, AIN104_RANGE, AIN105_RANGE, AIN106_RANGE, AIN107_RANGE, AIN108_RANGE, AIN109_RANGE, AIN110_RANGE, AIN111_RANGE, AIN112_RANGE, AIN113_RANGE, AIN114_RANGE, AIN115_RANGE, AIN116_RANGE, AIN117_RANGE, AIN118_RANGE, AIN119_RANGE, AIN120_RANGE, AIN121_RANGE, AIN122_RANGE, AIN123_RANGE, AIN124_RANGE, AIN125_RANGE, AIN126_RANGE, AIN127_RANGE, AIN128_RANGE, AIN129_RANGE, AIN130_RANGE, AIN131_RANGE, AIN132_RANGE, AIN133_RANGE, AIN134_RANGE, AIN135_RANGE, AIN136_RANGE, AIN137_RANGE, AIN138_RANGE, AIN139_RANGE, AIN140_RANGE, AIN141_RANGE, AIN142_RANGE, AIN143_RANGE, AIN144_RANGE, AIN145_RANGE, AIN146_RANGE, AIN147_RANGE, AIN148_RANGE, AIN149_RANGE, AIN150_RANGE, AIN151_RANGE, AIN152_RANGE, AIN153_RANGE, AIN154_RANGE, AIN155_RANGE, AIN156_RANGE, AIN157_RANGE, AIN158_RANGE, AIN159_RANGE, AIN160_RANGE, AIN161_RANGE, AIN162_RANGE, AIN163_RANGE, AIN164_RANGE, AIN165_RANGE, AIN166_RANGE, AIN167_RANGE, AIN168_RANGE, AIN169_RANGE, AIN170_RANGE, AIN171_RANGE, AIN172_RANGE, AIN173_RANGE, AIN174_RANGE, AIN175_RANGE, AIN176_RANGE, AIN177_RANGE, AIN178_RANGE, AIN179_RANGE, AIN180_RANGE, AIN181_RANGE, AIN182_RANGE, AIN183_RANGE, AIN184_RANGE, AIN185_RANGE, AIN186_RANGE, AIN187_RANGE, AIN188_RANGE, AIN189_RANGE, AIN190_RANGE, AIN191_RANGE, AIN192_RANGE, AIN193_RANGE, AIN194_RANGE, AIN195_RANGE, AIN196_RANGE, AIN197_RANGE, AIN198_RANGE, AIN199_RANGE, AIN200_RANGE, AIN201_RANGE,	40000, 40002, 40004, 40006, 40008, 40010, 40012, 40014, 40016, 40018, 40020, 40022, 40024, 40026, 40028, 40030, 40032, 40034, 40036, 40038, 40040, 40042, 40044, 40046, 40048, 40050, 40052, 40054, 40056, 40058, 40060, 40062, 40064, 40066, 40068, 40070, 40072, 40074, 40076, 40078, 40080, 40082, 40084, 40086, 40088, 40090, 40092, 40094, 40096, 40098, 40100, 40102, 40104, 40106, 40108, 40110, 40112, 40114, 40116, 40118, 40120, 40122, 40124, 40126, 40128, 40130, 40132, 40134, 40136, 40138, 40140, 40142, 40144, 40146, 40148, 40150, 40152, 40154, 40156, 40158, 40160, 40162, 40164, 40166, 40168, 40170, 40172, 40174, 40176, 40178, 40180, 40182, 40184, 40186, 40188, 40190, 40192, 40194, 40196, 40198, 40200, 40202, 40204, 40206, 40208, 40210, 40212, 40214, 40216, 40218, 40220, 40222, 40224, 40226, 40228, 40230, 40232, 40234, 40236, 40238, 40240, 40242, 40244, 40246, 40248, 40250, 40252, 40254, 40256, 40258, 40260, 40262, 40264, 40266, 40268, 40270, 40272, 40274, 40276, 40278, 40280, 40282, 40284, 40286, 40288, 40290, 40292, 40294, 40296, 40298, 40300, 40302, 40304, 40306, 40308, 40310, 40312, 40314, 40316, 40318, 40320, 40322, 40324, 40326, 40328, 40330, 40332, 40334, 40336, 40338, 40340, 40342, 40344, 40346, 40348, 40350, 40352, 40354, 40356, 40358, 40360, 40362, 40364, 40366, 40368, 40370, 40372, 40374, 40376, 40378, 40380, 40382, 40384, 40386, 40388, 40390, 40392, 40394, 40396, 40398,

AIN202_RANGE, AIN203_RANGE, AIN204_RANGE, AIN205_RANGE, AIN206_RANGE, AIN207_RANGE, AIN208_RANGE, AIN209_RANGE, AIN210_RANGE, AIN211_RANGE, AIN212_RANGE, AIN213_RANGE, AIN214_RANGE, AIN215_RANGE, AIN216_RANGE, AIN217_RANGE, AIN218_RANGE, AIN219_RANGE, AIN220_RANGE, AIN221_RANGE, AIN222_RANGE, AIN223_RANGE, AIN224_RANGE, AIN225_RANGE, AIN226_RANGE, AIN227_RANGE, AIN228_RANGE, AIN229_RANGE, AIN230_RANGE, AIN231_RANGE, AIN232_RANGE, AIN233_RANGE, AIN234_RANGE, AIN235_RANGE, AIN236_RANGE, AIN237_RANGE, AIN238_RANGE, AIN239_RANGE, AIN240_RANGE, AIN241_RANGE, AIN242_RANGE, AIN243_RANGE, AIN244_RANGE, AIN245_RANGE, AIN246_RANGE, AIN247_RANGE, AIN248_RANGE, AIN249_RANGE, AIN250_RANGE, AIN251_RANGE, AIN252_RANGE, AIN253_RANGE	40400, 40402, 40404, 40406, 40408, 40410, 40412, 40414, 40416, 40418, 40420, 40422, 40424, 40426, 40428, 40430, 40432, 40434, 40436, 40438, 40440, 40442, 40444, 40446, 40448, 40450, 40452, 40454, 40456, 40458, 40460, 40462, 40464, 40466, 40468, 40470, 40472, 40474, 40476, 40478, 40480, 40482, 40484, 40486, 40488, 40490, 40492, 40494, 40496, 40498, 40500, 40502, 40504, 40506

AIN#(0:253)_NEGATIVE_CH
- Starting Address: 41000

Specifies the negative channel to be used for each positive channel. 199=Default=> Single-Ended.

- Data type: UINT16 (type index = 0)
- Readable and writable
- Default value: 199
- T8:
    - Ignored. This register has no meaning on the T8.
- T7:
    - For base differential channels, positive must be an even channel from 0-12 and negative must be positive+1. For extended channels 16-127, see Mux80 datasheet.

Expanded Names	Addresses
AIN0_NEGATIVE_CH, AIN1_NEGATIVE_CH, AIN2_NEGATIVE_CH, AIN3_NEGATIVE_CH, AIN4_NEGATIVE_CH, AIN5_NEGATIVE_CH, AIN6_NEGATIVE_CH, AIN7_NEGATIVE_CH, AIN8_NEGATIVE_CH, AIN9_NEGATIVE_CH, AIN10_NEGATIVE_CH, AIN11_NEGATIVE_CH, AIN12_NEGATIVE_CH, AIN13_NEGATIVE_CH, AIN14_NEGATIVE_CH, AIN15_NEGATIVE_CH, AIN16_NEGATIVE_CH, AIN17_NEGATIVE_CH, AIN18_NEGATIVE_CH, AIN19_NEGATIVE_CH, AIN20_NEGATIVE_CH, AIN21_NEGATIVE_CH, AIN22_NEGATIVE_CH, AIN23_NEGATIVE_CH, AIN24_NEGATIVE_CH, AIN25_NEGATIVE_CH, AIN26_NEGATIVE_CH, AIN27_NEGATIVE_CH, AIN28_NEGATIVE_CH, AIN29_NEGATIVE_CH, AIN30_NEGATIVE_CH, AIN31_NEGATIVE_CH, AIN32_NEGATIVE_CH, AIN33_NEGATIVE_CH, AIN34_NEGATIVE_CH, AIN35_NEGATIVE_CH, AIN36_NEGATIVE_CH, AIN37_NEGATIVE_CH, AIN38_NEGATIVE_CH, AIN39_NEGATIVE_CH, AIN40_NEGATIVE_CH, AIN41_NEGATIVE_CH, AIN42_NEGATIVE_CH, AIN43_NEGATIVE_CH, AIN44_NEGATIVE_CH, AIN45_NEGATIVE_CH, AIN46_NEGATIVE_CH, AIN47_NEGATIVE_CH, AIN48_NEGATIVE_CH, AIN49_NEGATIVE_CH, AIN50_NEGATIVE_CH, AIN51_NEGATIVE_CH, AIN52_NEGATIVE_CH, AIN53_NEGATIVE_CH, AIN54_NEGATIVE_CH, AIN55_NEGATIVE_CH, AIN56_NEGATIVE_CH, AIN57_NEGATIVE_CH, AIN58_NEGATIVE_CH, AIN59_NEGATIVE_CH, AIN60_NEGATIVE_CH, AIN61_NEGATIVE_CH, AIN62_NEGATIVE_CH, AIN63_NEGATIVE_CH, AIN64_NEGATIVE_CH, AIN65_NEGATIVE_CH, AIN66_NEGATIVE_CH, AIN67_NEGATIVE_CH, AIN68_NEGATIVE_CH, AIN69_NEGATIVE_CH, AIN70_NEGATIVE_CH, AIN71_NEGATIVE_CH, AIN72_NEGATIVE_CH, AIN73_NEGATIVE_CH, AIN74_NEGATIVE_CH, AIN75_NEGATIVE_CH, AIN76_NEGATIVE_CH, AIN77_NEGATIVE_CH, AIN78_NEGATIVE_CH, AIN79_NEGATIVE_CH, AIN80_NEGATIVE_CH, AIN81_NEGATIVE_CH, AIN82_NEGATIVE_CH, AIN83_NEGATIVE_CH, AIN84_NEGATIVE_CH, AIN85_NEGATIVE_CH, AIN86_NEGATIVE_CH, AIN87_NEGATIVE_CH, AIN88_NEGATIVE_CH, AIN89_NEGATIVE_CH, AIN90_NEGATIVE_CH, AIN91_NEGATIVE_CH, AIN92_NEGATIVE_CH,	41000, 41001, 41002, 41003, 41004, 41005, 41006, 41007, 41008, 41009, 41010, 41011, 41012, 41013, 41014, 41015, 41016, 41017, 41018, 41019, 41020, 41021, 41022, 41023, 41024, 41025, 41026, 41027, 41028, 41029, 41030, 41031, 41032, 41033, 41034, 41035, 41036, 41037, 41038, 41039, 41040, 41041, 41042, 41043, 41044, 41045, 41046, 41047, 41048, 41049, 41050, 41051, 41052, 41053, 41054, 41055, 41056, 41057, 41058, 41059, 41060, 41061, 41062, 41063, 41064, 41065, 41066, 41067, 41068, 41069, 41070, 41071, 41072, 41073, 41074, 41075, 41076, 41077, 41078, 41079, 41080, 41081, 41082, 41083, 41084, 41085, 41086, 41087, 41088, 41089, 41090, 41091, 41092,

AIN93_NEGATIVE_CH, AIN94_NEGATIVE_CH, AIN95_NEGATIVE_CH,	41093, 41094, 41095,
AIN96_NEGATIVE_CH, AIN97_NEGATIVE_CH, AIN98_NEGATIVE_CH,	41096, 41097, 41098,
AIN99_NEGATIVE_CH, AIN100_NEGATIVE_CH, AIN101_NEGATIVE_CH,	41099, 41100, 41101,
AIN102_NEGATIVE_CH, AIN103_NEGATIVE_CH, AIN104_NEGATIVE_CH,	41102, 41103, 41104,
AIN105_NEGATIVE_CH, AIN106_NEGATIVE_CH, AIN107_NEGATIVE_CH,	41105, 41106, 41107,
AIN108_NEGATIVE_CH, AIN109_NEGATIVE_CH, AIN110_NEGATIVE_CH,	41108, 41109, 41110,
AIN111_NEGATIVE_CH, AIN112_NEGATIVE_CH, AIN113_NEGATIVE_CH,	41111, 41112, 41113,
AIN114_NEGATIVE_CH, AIN115_NEGATIVE_CH, AIN116_NEGATIVE_CH,	41114, 41115, 41116,
AIN117_NEGATIVE_CH, AIN118_NEGATIVE_CH, AIN119_NEGATIVE_CH,	41117, 41118, 41119,
AIN120_NEGATIVE_CH, AIN121_NEGATIVE_CH, AIN122_NEGATIVE_CH,	41120, 41121, 41122,
AIN123_NEGATIVE_CH, AIN124_NEGATIVE_CH, AIN125_NEGATIVE_CH,	41123, 41124, 41125,
AIN126_NEGATIVE_CH, AIN127_NEGATIVE_CH, AIN128_NEGATIVE_CH,	41126, 41127, 41128,
AIN129_NEGATIVE_CH, AIN130_NEGATIVE_CH, AIN131_NEGATIVE_CH,	41129, 41130, 41131,
AIN132_NEGATIVE_CH, AIN133_NEGATIVE_CH, AIN134_NEGATIVE_CH,	41132, 41133, 41134,
AIN135_NEGATIVE_CH, AIN136_NEGATIVE_CH, AIN137_NEGATIVE_CH,	41135, 41136, 41137,
AIN138_NEGATIVE_CH, AIN139_NEGATIVE_CH, AIN140_NEGATIVE_CH,	41138, 41139, 41140,
AIN141_NEGATIVE_CH, AIN142_NEGATIVE_CH, AIN143_NEGATIVE_CH,	41141, 41142, 41143,
AIN144_NEGATIVE_CH, AIN145_NEGATIVE_CH, AIN146_NEGATIVE_CH,	41144, 41145, 41146,
AIN147_NEGATIVE_CH, AIN148_NEGATIVE_CH, AIN149_NEGATIVE_CH,	41147, 41148, 41149,
AIN150_NEGATIVE_CH, AIN151_NEGATIVE_CH, AIN152_NEGATIVE_CH,	41150, 41151, 41152,
AIN153_NEGATIVE_CH, AIN154_NEGATIVE_CH, AIN155_NEGATIVE_CH,	41153, 41154, 41155,
AIN156_NEGATIVE_CH, AIN157_NEGATIVE_CH, AIN158_NEGATIVE_CH,	41156, 41157, 41158,
AIN159_NEGATIVE_CH, AIN160_NEGATIVE_CH, AIN161_NEGATIVE_CH,	41159, 41160, 41161,
AIN162_NEGATIVE_CH, AIN163_NEGATIVE_CH, AIN164_NEGATIVE_CH,	41162, 41163, 41164,
AIN165_NEGATIVE_CH, AIN166_NEGATIVE_CH, AIN167_NEGATIVE_CH,	41165, 41166, 41167,
AIN168_NEGATIVE_CH, AIN169_NEGATIVE_CH, AIN170_NEGATIVE_CH,	41168, 41169, 41170,
AIN171_NEGATIVE_CH, AIN172_NEGATIVE_CH, AIN173_NEGATIVE_CH,	41171, 41172, 41173,
AIN174_NEGATIVE_CH, AIN175_NEGATIVE_CH, AIN176_NEGATIVE_CH,	41174, 41175, 41176,
AIN177_NEGATIVE_CH, AIN178_NEGATIVE_CH, AIN179_NEGATIVE_CH,	41177, 41178, 41179,
AIN180_NEGATIVE_CH, AIN181_NEGATIVE_CH, AIN182_NEGATIVE_CH,	41180, 41181, 41182,
AIN183_NEGATIVE_CH, AIN184_NEGATIVE_CH, AIN185_NEGATIVE_CH,	41183, 41184, 41185,
AIN186_NEGATIVE_CH, AIN187_NEGATIVE_CH, AIN188_NEGATIVE_CH,	41186, 41187, 41188,
AIN189_NEGATIVE_CH, AIN190_NEGATIVE_CH, AIN191_NEGATIVE_CH,	41189, 41190, 41191,
AIN192_NEGATIVE_CH, AIN193_NEGATIVE_CH, AIN194_NEGATIVE_CH,	41192, 41193, 41194,
AIN195_NEGATIVE_CH, AIN196_NEGATIVE_CH, AIN197_NEGATIVE_CH,	41195, 41196, 41197,
AIN198_NEGATIVE_CH, AIN199_NEGATIVE_CH, AIN200_NEGATIVE_CH,	41198, 41199, 41200,
AIN201_NEGATIVE_CH, AIN202_NEGATIVE_CH, AIN203_NEGATIVE_CH,	41201, 41202, 41203,
AIN204_NEGATIVE_CH, AIN205_NEGATIVE_CH, AIN206_NEGATIVE_CH,	41204, 41205, 41206,
AIN207_NEGATIVE_CH, AIN208_NEGATIVE_CH, AIN209_NEGATIVE_CH,	41207, 41208, 41209,
AIN210_NEGATIVE_CH, AIN211_NEGATIVE_CH, AIN212_NEGATIVE_CH,	41210, 41211, 41212,
AIN213_NEGATIVE_CH, AIN214_NEGATIVE_CH, AIN215_NEGATIVE_CH,	41213, 41214, 41215,
AIN216_NEGATIVE_CH, AIN217_NEGATIVE_CH, AIN218_NEGATIVE_CH,	41216, 41217, 41218,
AIN219_NEGATIVE_CH, AIN220_NEGATIVE_CH, AIN221_NEGATIVE_CH,	41219, 41220, 41221,
AIN222_NEGATIVE_CH, AIN223_NEGATIVE_CH, AIN224_NEGATIVE_CH,	41222, 41223, 41224,
AIN225_NEGATIVE_CH, AIN226_NEGATIVE_CH, AIN227_NEGATIVE_CH,	41225, 41226, 41227,
AIN228_NEGATIVE_CH, AIN229_NEGATIVE_CH, AIN230_NEGATIVE_CH,	41228, 41229, 41230,
AIN231_NEGATIVE_CH, AIN232_NEGATIVE_CH, AIN233_NEGATIVE_CH,	41231, 41232, 41233,
AIN234_NEGATIVE_CH, AIN235_NEGATIVE_CH, AIN236_NEGATIVE_CH,	41234, 41235, 41236,
AIN237_NEGATIVE_CH, AIN238_NEGATIVE_CH, AIN239_NEGATIVE_CH,	41237, 41238, 41239,
AIN240_NEGATIVE_CH, AIN241_NEGATIVE_CH, AIN242_NEGATIVE_CH,	41240, 41241, 41242,
AIN243_NEGATIVE_CH, AIN244_NEGATIVE_CH, AIN245_NEGATIVE_CH,	41243, 41244, 41245,
AIN246_NEGATIVE_CH, AIN247_NEGATIVE_CH, AIN248_NEGATIVE_CH,	41246, 41247, 41248,
AIN249_NEGATIVE_CH, AIN250_NEGATIVE_CH, AIN251_NEGATIVE_CH,	41249, 41250, 41251,
AIN252_NEGATIVE_CH, AIN253_NEGATIVE_CH	41252, 41253

AIN#(0:253)_RESOLUTION_INDEX
- Starting Address: 41500

The resolution index for command-response and AIN-EF readings. A larger resolution index generally results in lower noise and longer sample times.

- Data type: UINT16 (type index = 0)
- Readable and writable

- Default value: 0
- T8:
    - Valid values: 0-16. A value of 0 will instruct the T8 to use the best resolution for the rate specified.
- T7:
    - Valid values: 0-8 for T7, 0-12 for T7-Pro. Default value of 0 corresponds to an index of 8 (T7) or 9 (T7-Pro).
- T4:
    - Valid values: 0-5. Default value of 0 corresponds to an index of 5.

Expanded Names	Addresses
AIN0_RESOLUTION_INDEX, AIN1_RESOLUTION_INDEX, AIN2_RESOLUTION_INDEX, AIN3_RESOLUTION_INDEX, AIN4_RESOLUTION_INDEX, AIN5_RESOLUTION_INDEX, AIN6_RESOLUTION_INDEX, AIN7_RESOLUTION_INDEX, AIN8_RESOLUTION_INDEX, AIN9_RESOLUTION_INDEX, AIN10_RESOLUTION_INDEX, AIN11_RESOLUTION_INDEX, AIN12_RESOLUTION_INDEX, AIN13_RESOLUTION_INDEX, AIN14_RESOLUTION_INDEX, AIN15_RESOLUTION_INDEX, AIN16_RESOLUTION_INDEX, AIN17_RESOLUTION_INDEX, AIN18_RESOLUTION_INDEX, AIN19_RESOLUTION_INDEX, AIN20_RESOLUTION_INDEX, AIN21_RESOLUTION_INDEX, AIN22_RESOLUTION_INDEX, AIN23_RESOLUTION_INDEX, AIN24_RESOLUTION_INDEX, AIN25_RESOLUTION_INDEX, AIN26_RESOLUTION_INDEX, AIN27_RESOLUTION_INDEX, AIN28_RESOLUTION_INDEX, AIN29_RESOLUTION_INDEX, AIN30_RESOLUTION_INDEX, AIN31_RESOLUTION_INDEX, AIN32_RESOLUTION_INDEX, AIN33_RESOLUTION_INDEX, AIN34_RESOLUTION_INDEX, AIN35_RESOLUTION_INDEX, AIN36_RESOLUTION_INDEX, AIN37_RESOLUTION_INDEX, AIN38_RESOLUTION_INDEX, AIN39_RESOLUTION_INDEX, AIN40_RESOLUTION_INDEX, AIN41_RESOLUTION_INDEX, AIN42_RESOLUTION_INDEX, AIN43_RESOLUTION_INDEX, AIN44_RESOLUTION_INDEX, AIN45_RESOLUTION_INDEX, AIN46_RESOLUTION_INDEX, AIN47_RESOLUTION_INDEX, AIN48_RESOLUTION_INDEX, AIN49_RESOLUTION_INDEX, AIN50_RESOLUTION_INDEX, AIN51_RESOLUTION_INDEX, AIN52_RESOLUTION_INDEX, AIN53_RESOLUTION_INDEX, AIN54_RESOLUTION_INDEX, AIN55_RESOLUTION_INDEX, AIN56_RESOLUTION_INDEX, AIN57_RESOLUTION_INDEX, AIN58_RESOLUTION_INDEX, AIN59_RESOLUTION_INDEX, AIN60_RESOLUTION_INDEX, AIN61_RESOLUTION_INDEX, AIN62_RESOLUTION_INDEX, AIN63_RESOLUTION_INDEX, AIN64_RESOLUTION_INDEX, AIN65_RESOLUTION_INDEX, AIN66_RESOLUTION_INDEX, AIN67_RESOLUTION_INDEX, AIN68_RESOLUTION_INDEX, AIN69_RESOLUTION_INDEX, AIN70_RESOLUTION_INDEX, AIN71_RESOLUTION_INDEX, AIN72_RESOLUTION_INDEX, AIN73_RESOLUTION_INDEX, AIN74_RESOLUTION_INDEX, AIN75_RESOLUTION_INDEX, AIN76_RESOLUTION_INDEX, AIN77_RESOLUTION_INDEX, AIN78_RESOLUTION_INDEX, AIN79_RESOLUTION_INDEX, AIN80_RESOLUTION_INDEX, AIN81_RESOLUTION_INDEX, AIN82_RESOLUTION_INDEX, AIN83_RESOLUTION_INDEX, AIN84_RESOLUTION_INDEX, AIN85_RESOLUTION_INDEX, AIN86_RESOLUTION_INDEX, AIN87_RESOLUTION_INDEX, AIN88_RESOLUTION_INDEX, AIN89_RESOLUTION_INDEX, AIN90_RESOLUTION_INDEX, AIN91_RESOLUTION_INDEX, AIN92_RESOLUTION_INDEX, AIN93_RESOLUTION_INDEX, AIN94_RESOLUTION_INDEX, AIN95_RESOLUTION_INDEX, AIN96_RESOLUTION_INDEX, AIN97_RESOLUTION_INDEX, AIN98_RESOLUTION_INDEX, AIN99_RESOLUTION_INDEX, AIN100_RESOLUTION_INDEX,	41500, 41501, 41502, 41503, 41504, 41505, 41506, 41507, 41508, 41509, 41510, 41511, 41512, 41513, 41514, 41515, 41516, 41517, 41518, 41519, 41520, 41521, 41522, 41523, 41524, 41525, 41526, 41527, 41528, 41529, 41530, 41531, 41532, 41533, 41534, 41535, 41536, 41537, 41538, 41539, 41540, 41541, 41542, 41543, 41544, 41545, 41546, 41547, 41548, 41549, 41550, 41551, 41552, 41553, 41554, 41555, 41556, 41557, 41558, 41559, 41560, 41561, 41562, 41563, 41564, 41565, 41566, 41567, 41568, 41569, 41570, 41571, 41572, 41573, 41574, 41575, 41576, 41577, 41578, 41579, 41580, 41581, 41582, 41583, 41584, 41585, 41586, 41587, 41588, 41589, 41590, 41591, 41592, 41593, 41594, 41595, 41596, 41597,

AIN101_RESOLUTION_INDEX, AIN102_RESOLUTION_INDEX,	41598, 41599,
AIN103_RESOLUTION_INDEX, AIN104_RESOLUTION_INDEX,	41600, 41601,
AIN105_RESOLUTION_INDEX, AIN106_RESOLUTION_INDEX,	41602, 41603,
AIN107_RESOLUTION_INDEX, AIN108_RESOLUTION_INDEX,	41604, 41605,
AIN109_RESOLUTION_INDEX, AIN110_RESOLUTION_INDEX,	41606, 41607,
AIN111_RESOLUTION_INDEX, AIN112_RESOLUTION_INDEX,	41608, 41609,
AIN113_RESOLUTION_INDEX, AIN114_RESOLUTION_INDEX,	41610, 41611,
AIN115_RESOLUTION_INDEX, AIN116_RESOLUTION_INDEX,	41612, 41613,
AIN117_RESOLUTION_INDEX, AIN118_RESOLUTION_INDEX,	41614, 41615,
AIN119_RESOLUTION_INDEX, AIN120_RESOLUTION_INDEX,	41616, 41617,
AIN121_RESOLUTION_INDEX, AIN122_RESOLUTION_INDEX,	41618, 41619,
AIN123_RESOLUTION_INDEX, AIN124_RESOLUTION_INDEX,	41620, 41621,
AIN125_RESOLUTION_INDEX, AIN126_RESOLUTION_INDEX,	41622, 41623,
AIN127_RESOLUTION_INDEX, AIN128_RESOLUTION_INDEX,	41624, 41625,
AIN129_RESOLUTION_INDEX, AIN130_RESOLUTION_INDEX,	41626, 41627,
AIN131_RESOLUTION_INDEX, AIN132_RESOLUTION_INDEX,	41628, 41629,
AIN133_RESOLUTION_INDEX, AIN134_RESOLUTION_INDEX,	41630, 41631,
AIN135_RESOLUTION_INDEX, AIN136_RESOLUTION_INDEX,	41632, 41633,
AIN137_RESOLUTION_INDEX, AIN138_RESOLUTION_INDEX,	41634, 41635,
AIN139_RESOLUTION_INDEX, AIN140_RESOLUTION_INDEX,	41636, 41637,
AIN141_RESOLUTION_INDEX, AIN142_RESOLUTION_INDEX,	41638, 41639,
AIN143_RESOLUTION_INDEX, AIN144_RESOLUTION_INDEX,	41640, 41641,
AIN145_RESOLUTION_INDEX, AIN146_RESOLUTION_INDEX,	41642, 41643,
AIN147_RESOLUTION_INDEX, AIN148_RESOLUTION_INDEX,	41644, 41645,
AIN149_RESOLUTION_INDEX, AIN150_RESOLUTION_INDEX,	41646, 41647,
AIN151_RESOLUTION_INDEX, AIN152_RESOLUTION_INDEX,	41648, 41649,
AIN153_RESOLUTION_INDEX, AIN154_RESOLUTION_INDEX,	41650, 41651,
AIN155_RESOLUTION_INDEX, AIN156_RESOLUTION_INDEX,	41652, 41653,
AIN157_RESOLUTION_INDEX, AIN158_RESOLUTION_INDEX,	41654, 41655,
AIN159_RESOLUTION_INDEX, AIN160_RESOLUTION_INDEX,	41656, 41657,
AIN161_RESOLUTION_INDEX, AIN162_RESOLUTION_INDEX,	41658, 41659,
AIN163_RESOLUTION_INDEX, AIN164_RESOLUTION_INDEX,	41660, 41661,
AIN165_RESOLUTION_INDEX, AIN166_RESOLUTION_INDEX,	41662, 41663,
AIN167_RESOLUTION_INDEX, AIN168_RESOLUTION_INDEX,	41664, 41665,
AIN169_RESOLUTION_INDEX, AIN170_RESOLUTION_INDEX,	41666, 41667,
AIN171_RESOLUTION_INDEX, AIN172_RESOLUTION_INDEX,	41668, 41669,
AIN173_RESOLUTION_INDEX, AIN174_RESOLUTION_INDEX,	41670, 41671,
AIN175_RESOLUTION_INDEX, AIN176_RESOLUTION_INDEX,	41672, 41673,
AIN177_RESOLUTION_INDEX, AIN178_RESOLUTION_INDEX,	41674, 41675,
AIN179_RESOLUTION_INDEX, AIN180_RESOLUTION_INDEX,	41676, 41677,
AIN181_RESOLUTION_INDEX, AIN182_RESOLUTION_INDEX,	41678, 41679,
AIN183_RESOLUTION_INDEX, AIN184_RESOLUTION_INDEX,	41680, 41681,
AIN185_RESOLUTION_INDEX, AIN186_RESOLUTION_INDEX,	41682, 41683,
AIN187_RESOLUTION_INDEX, AIN188_RESOLUTION_INDEX,	41684, 41685,
AIN189_RESOLUTION_INDEX, AIN190_RESOLUTION_INDEX,	41686, 41687,
AIN191_RESOLUTION_INDEX, AIN192_RESOLUTION_INDEX,	41688, 41689,
AIN193_RESOLUTION_INDEX, AIN194_RESOLUTION_INDEX,	41690, 41691,
AIN195_RESOLUTION_INDEX, AIN196_RESOLUTION_INDEX,	41692, 41693,
AIN197_RESOLUTION_INDEX, AIN198_RESOLUTION_INDEX,	41694, 41695,
AIN199_RESOLUTION_INDEX, AIN200_RESOLUTION_INDEX,	41696, 41697,
AIN201_RESOLUTION_INDEX, AIN202_RESOLUTION_INDEX,	41698, 41699,
AIN203_RESOLUTION_INDEX, AIN204_RESOLUTION_INDEX,	41700, 41701,
AIN205_RESOLUTION_INDEX, AIN206_RESOLUTION_INDEX,	41702, 41703,
AIN207_RESOLUTION_INDEX, AIN208_RESOLUTION_INDEX,	41704, 41705,
AIN209_RESOLUTION_INDEX, AIN210_RESOLUTION_INDEX,	41706, 41707,
AIN211_RESOLUTION_INDEX, AIN212_RESOLUTION_INDEX,	41708, 41709,
AIN213_RESOLUTION_INDEX, AIN214_RESOLUTION_INDEX,	41710, 41711,
AIN215_RESOLUTION_INDEX, AIN216_RESOLUTION_INDEX,	41712, 41713,
AIN217_RESOLUTION_INDEX, AIN218_RESOLUTION_INDEX,	41714, 41715,
AIN219_RESOLUTION_INDEX, AIN220_RESOLUTION_INDEX,	41716, 41717,
AIN221_RESOLUTION_INDEX, AIN222_RESOLUTION_INDEX,	41718, 41719,
AIN223_RESOLUTION_INDEX, AIN224_RESOLUTION_INDEX,	41720, 41721,
AIN225_RESOLUTION_INDEX, AIN226_RESOLUTION_INDEX,	41722, 41723,

AIN227_RESOLUTION_INDEX, AIN228_RESOLUTION_INDEX,	41724, 41725,
AIN229_RESOLUTION_INDEX, AIN230_RESOLUTION_INDEX,	41726, 41727,
AIN231_RESOLUTION_INDEX, AIN232_RESOLUTION_INDEX,	41728, 41729,
AIN233_RESOLUTION_INDEX, AIN234_RESOLUTION_INDEX,	41730, 41731,
AIN235_RESOLUTION_INDEX, AIN236_RESOLUTION_INDEX,	41732, 41733,
AIN237_RESOLUTION_INDEX, AIN238_RESOLUTION_INDEX,	41734, 41735,
AIN239_RESOLUTION_INDEX, AIN240_RESOLUTION_INDEX,	41736, 41737,
AIN241_RESOLUTION_INDEX, AIN242_RESOLUTION_INDEX,	41738, 41739,
AIN243_RESOLUTION_INDEX, AIN244_RESOLUTION_INDEX,	41740, 41741,
AIN245_RESOLUTION_INDEX, AIN246_RESOLUTION_INDEX,	41742, 41743,
AIN247_RESOLUTION_INDEX, AIN248_RESOLUTION_INDEX,	41744, 41745,
AIN249_RESOLUTION_INDEX, AIN250_RESOLUTION_INDEX,	41746, 41747,
AIN251_RESOLUTION_INDEX, AIN252_RESOLUTION_INDEX,	41748, 41749,
AIN253_RESOLUTION_INDEX	41750, 41751,
	41752, 41753

AIN#(0:253)_SETTLING_US
- Starting Address: 42000

Settling time for command-response and AIN-EF readings.

- Data type: FLOAT32 (type index = 3)
- Readable and writable
- Default value: 0
- T8:
  - Ignored. This register has no meaning on the T8.
- T7:
  - 0 = Auto. Max is 50000 (microseconds).
  - Minimum firmware version: 0.9328
- T4:
  - 0 = Auto. Max is 10000 (microseconds).

Expanded Names	Addresses
AIN0_SETTLING_US, AIN1_SETTLING_US, AIN2_SETTLING_US,	42000, 42002, 42004,
AIN3_SETTLING_US, AIN4_SETTLING_US, AIN5_SETTLING_US,	42006, 42008, 42010,
AIN6_SETTLING_US, AIN7_SETTLING_US, AIN8_SETTLING_US,	42012, 42014, 42016,
AIN9_SETTLING_US, AIN10_SETTLING_US, AIN11_SETTLING_US,	42018, 42020, 42022,
AIN12_SETTLING_US, AIN13_SETTLING_US, AIN14_SETTLING_US,	42024, 42026, 42028,
AIN15_SETTLING_US, AIN16_SETTLING_US, AIN17_SETTLING_US,	42030, 42032, 42034,
AIN18_SETTLING_US, AIN19_SETTLING_US, AIN20_SETTLING_US,	42036, 42038, 42040,
AIN21_SETTLING_US, AIN22_SETTLING_US, AIN23_SETTLING_US,	42042, 42044, 42046,
AIN24_SETTLING_US, AIN25_SETTLING_US, AIN26_SETTLING_US,	42048, 42050, 42052,
AIN27_SETTLING_US, AIN28_SETTLING_US, AIN29_SETTLING_US,	42054, 42056, 42058,
AIN30_SETTLING_US, AIN31_SETTLING_US, AIN32_SETTLING_US,	42060, 42062, 42064,
AIN33_SETTLING_US, AIN34_SETTLING_US, AIN35_SETTLING_US,	42066, 42068, 42070,
AIN36_SETTLING_US, AIN37_SETTLING_US, AIN38_SETTLING_US,	42072, 42074, 42076,
AIN39_SETTLING_US, AIN40_SETTLING_US, AIN41_SETTLING_US,	42078, 42080, 42082,
AIN42_SETTLING_US, AIN43_SETTLING_US, AIN44_SETTLING_US,	42084, 42086, 42088,
AIN45_SETTLING_US, AIN46_SETTLING_US, AIN47_SETTLING_US,	42090, 42092, 42094,
AIN48_SETTLING_US, AIN49_SETTLING_US, AIN50_SETTLING_US,	42096, 42098, 42100,
AIN51_SETTLING_US, AIN52_SETTLING_US, AIN53_SETTLING_US,	42102, 42104, 42106,
AIN54_SETTLING_US, AIN55_SETTLING_US, AIN56_SETTLING_US,	42108, 42110, 42112,
AIN57_SETTLING_US, AIN58_SETTLING_US, AIN59_SETTLING_US,	42114, 42116, 42118,
AIN60_SETTLING_US, AIN61_SETTLING_US, AIN62_SETTLING_US,	42120, 42122, 42124,
AIN63_SETTLING_US, AIN64_SETTLING_US, AIN65_SETTLING_US,	42126, 42128, 42130,
AIN66_SETTLING_US, AIN67_SETTLING_US, AIN68_SETTLING_US,	42132, 42134, 42136,
AIN69_SETTLING_US, AIN70_SETTLING_US, AIN71_SETTLING_US,	42138, 42140, 42142,
AIN72_SETTLING_US, AIN73_SETTLING_US, AIN74_SETTLING_US,	42144, 42146, 42148,
AIN75_SETTLING_US, AIN76_SETTLING_US, AIN77_SETTLING_US,	42150, 42152, 42154,
AIN78_SETTLING_US, AIN79_SETTLING_US, AIN80_SETTLING_US,	42156, 42158, 42160,

AIN81_SETTLING_US, AIN82_SETTLING_US, AIN83_SETTLING_US,	42162, 42164, 42166,
AIN84_SETTLING_US, AIN85_SETTLING_US, AIN86_SETTLING_US,	42168, 42170, 42172,
AIN87_SETTLING_US, AIN88_SETTLING_US, AIN89_SETTLING_US,	42174, 42176, 42178,
AIN90_SETTLING_US, AIN91_SETTLING_US, AIN92_SETTLING_US,	42180, 42182, 42184,
AIN93_SETTLING_US, AIN94_SETTLING_US, AIN95_SETTLING_US,	42186, 42188, 42190,
AIN96_SETTLING_US, AIN97_SETTLING_US, AIN98_SETTLING_US,	42192, 42194, 42196,
AIN99_SETTLING_US, AIN100_SETTLING_US, AIN101_SETTLING_US,	42198, 42200, 42202,
AIN102_SETTLING_US, AIN103_SETTLING_US, AIN104_SETTLING_US,	42204, 42206, 42208,
AIN105_SETTLING_US, AIN106_SETTLING_US, AIN107_SETTLING_US,	42210, 42212, 42214,
AIN108_SETTLING_US, AIN109_SETTLING_US, AIN110_SETTLING_US,	42216, 42218, 42220,
AIN111_SETTLING_US, AIN112_SETTLING_US, AIN113_SETTLING_US,	42222, 42224, 42226,
AIN114_SETTLING_US, AIN115_SETTLING_US, AIN116_SETTLING_US,	42228, 42230, 42232,
AIN117_SETTLING_US, AIN118_SETTLING_US, AIN119_SETTLING_US,	42234, 42236, 42238,
AIN120_SETTLING_US, AIN121_SETTLING_US, AIN122_SETTLING_US,	42240, 42242, 42244,
AIN123_SETTLING_US, AIN124_SETTLING_US, AIN125_SETTLING_US,	42246, 42248, 42250,
AIN126_SETTLING_US, AIN127_SETTLING_US, AIN128_SETTLING_US,	42252, 42254, 42256,
AIN129_SETTLING_US, AIN130_SETTLING_US, AIN131_SETTLING_US,	42258, 42260, 42262,
AIN132_SETTLING_US, AIN133_SETTLING_US, AIN134_SETTLING_US,	42264, 42266, 42268,
AIN135_SETTLING_US, AIN136_SETTLING_US, AIN137_SETTLING_US,	42270, 42272, 42274,
AIN138_SETTLING_US, AIN139_SETTLING_US, AIN140_SETTLING_US,	42276, 42278, 42280,
AIN141_SETTLING_US, AIN142_SETTLING_US, AIN143_SETTLING_US,	42282, 42284, 42286,
AIN144_SETTLING_US, AIN145_SETTLING_US, AIN146_SETTLING_US,	42288, 42290, 42292,
AIN147_SETTLING_US, AIN148_SETTLING_US, AIN149_SETTLING_US,	42294, 42296, 42298,
AIN150_SETTLING_US, AIN151_SETTLING_US, AIN152_SETTLING_US,	42300, 42302, 42304,
AIN153_SETTLING_US, AIN154_SETTLING_US, AIN155_SETTLING_US,	42306, 42308, 42310,
AIN156_SETTLING_US, AIN157_SETTLING_US, AIN158_SETTLING_US,	42312, 42314, 42316,
AIN159_SETTLING_US, AIN160_SETTLING_US, AIN161_SETTLING_US,	42318, 42320, 42322,
AIN162_SETTLING_US, AIN163_SETTLING_US, AIN164_SETTLING_US,	42324, 42326, 42328,
AIN165_SETTLING_US, AIN166_SETTLING_US, AIN167_SETTLING_US,	42330, 42332, 42334,
AIN168_SETTLING_US, AIN169_SETTLING_US, AIN170_SETTLING_US,	42336, 42338, 42340,
AIN171_SETTLING_US, AIN172_SETTLING_US, AIN173_SETTLING_US,	42342, 42344, 42346,
AIN174_SETTLING_US, AIN175_SETTLING_US, AIN176_SETTLING_US,	42348, 42350, 42352,
AIN177_SETTLING_US, AIN178_SETTLING_US, AIN179_SETTLING_US,	42354, 42356, 42358,
AIN180_SETTLING_US, AIN181_SETTLING_US, AIN182_SETTLING_US,	42360, 42362, 42364,
AIN183_SETTLING_US, AIN184_SETTLING_US, AIN185_SETTLING_US,	42366, 42368, 42370,
AIN186_SETTLING_US, AIN187_SETTLING_US, AIN188_SETTLING_US,	42372, 42374, 42376,
AIN189_SETTLING_US, AIN190_SETTLING_US, AIN191_SETTLING_US,	42378, 42380, 42382,
AIN192_SETTLING_US, AIN193_SETTLING_US, AIN194_SETTLING_US,	42384, 42386, 42388,
AIN195_SETTLING_US, AIN196_SETTLING_US, AIN197_SETTLING_US,	42390, 42392, 42394,
AIN198_SETTLING_US, AIN199_SETTLING_US, AIN200_SETTLING_US,	42396, 42398, 42400,
AIN201_SETTLING_US, AIN202_SETTLING_US, AIN203_SETTLING_US,	42402, 42404, 42406,
AIN204_SETTLING_US, AIN205_SETTLING_US, AIN206_SETTLING_US,	42408, 42410, 42412,
AIN207_SETTLING_US, AIN208_SETTLING_US, AIN209_SETTLING_US,	42414, 42416, 42418,
AIN210_SETTLING_US, AIN211_SETTLING_US, AIN212_SETTLING_US,	42420, 42422, 42424,
AIN213_SETTLING_US, AIN214_SETTLING_US, AIN215_SETTLING_US,	42426, 42428, 42430,
AIN216_SETTLING_US, AIN217_SETTLING_US, AIN218_SETTLING_US,	42432, 42434, 42436,
AIN219_SETTLING_US, AIN220_SETTLING_US, AIN221_SETTLING_US,	42438, 42440, 42442,
AIN222_SETTLING_US, AIN223_SETTLING_US, AIN224_SETTLING_US,	42444, 42446, 42448,
AIN225_SETTLING_US, AIN226_SETTLING_US, AIN227_SETTLING_US,	42450, 42452, 42454,
AIN228_SETTLING_US, AIN229_SETTLING_US, AIN230_SETTLING_US,	42456, 42458, 42460,
AIN231_SETTLING_US, AIN232_SETTLING_US, AIN233_SETTLING_US,	42462, 42464, 42466,
AIN234_SETTLING_US, AIN235_SETTLING_US, AIN236_SETTLING_US,	42468, 42470, 42472,
AIN237_SETTLING_US, AIN238_SETTLING_US, AIN239_SETTLING_US,	42474, 42476, 42478,
AIN240_SETTLING_US, AIN241_SETTLING_US, AIN242_SETTLING_US,	42480, 42482, 42484,
AIN243_SETTLING_US, AIN244_SETTLING_US, AIN245_SETTLING_US,	42486, 42488, 42490,
AIN246_SETTLING_US, AIN247_SETTLING_US, AIN248_SETTLING_US,	42492, 42494, 42496,
AIN249_SETTLING_US, AIN250_SETTLING_US, AIN251_SETTLING_US,	42498, 42500, 42502,
AIN252_SETTLING_US, AIN253_SETTLING_US	42504, 42506

AIN_CHANNEL_ENABLE
- Address: 43700

T8 Only. This register is a bitmask representing the enable states of the analog inputs.

- Data type: UINT32 (type index = 1)
- Readable and writable
- Default value: 255
- T8:
    - Minimum firmware version: 0.0123

AIN_SAMPLING_RATE_HZ
- Address: 43710

T8 Only. Writing to this register sets the analog sampling rate. Not all rates are possible. Read AIN_SAMPLING_RATE_ACTUAL_HZ to get the actual rate.

- Data type: FLOAT32 (type index = 3)
- Readable and writable
- Default value: 100
- T8:
    - Minimum firmware version: 0.0123

AIN_SAMPLING_RATE_ACTUAL_HZ
- Address: 43712

T8 Only. Returns the AIN system's actual sampling rate.

- Data type: FLOAT32 (type index = 3)
- Read-only
- Default value: 255
- T8:
    - Minimum firmware version: 0.0123

AIN_ALL_RANGE
- Address: 43900

A write to this global parameter affects all AIN. A read will return the correct setting if all channels are set the same, but otherwise will return -9999.

- Data type: FLOAT32 (type index = 3)
- Readable and writable
- Default value: 0
- T8:
    - Minimum firmware version: 0.0123
- T7:
    - Minimum firmware version: 0.9328

AIN_ALL_NEGATIVE_CH
- Address: 43902

A write to this global parameter affects all AIN. Writing 1 will set all AINs to differential. Writing 199 will set all AINs to single-ended. A read will return 1 if all AINs are set to differential and 199 if all AINs are set to single-ended. If AIN configurations are not consistent 0xFFFF will be returned.

- Data type: UINT16 (type index = 0)
- Readable and writable
- Default value: 199
- T8:
    - Ignored. This register has no meaning on the T8.
- T7:
    - Minimum firmware version: 0.9328

AIN_ALL_RESOLUTION_INDEX
- Address: 43903

The resolution index for command-response and AIN-EF readings. A larger resolution index generally results in lower noise and longer sample times. A write to this global parameter affects all AIN. A read will return the correct setting if all channels are set the same, but otherwise will return 0xFFFF.

- Data type: UINT16 (type index = 0)
- Readable and writable
- Default value: 0
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Valid values: 0-8 for T7, 0-12 for T7-Pro. Default value of 0 corresponds to an index of 8 (T7) or 9 (T7-Pro).
  - Minimum firmware version: 0.9328
- T4:
  - Valid values: 0-5. Default value of 0 corresponds to an index of 5.

AIN_ALL_SETTLING_US
- Address: 43904

Settling time for command-response and AIN-EF readings. A write to this global parameter affects all AIN. A read will return the correct setting if all channels are set the same, but otherwise will return -9999. Max is 50,000 us.

- Data type: FLOAT32 (type index = 3)
- Readable and writable
- Default value: 0
- T8:
  - Ignored. This register has no meaning on the T8.
- T7:
  - 0 = Auto. Max is 50000 (microseconds).
  - Minimum firmware version: 0.9328
- T4:
  - 0 = Auto. Max is 10000 (microseconds).

AIN_ALL_EF_INDEX
- Address: 43906

Write 0 to deactivate AIN_EF on all AINs. No other values may be written to this register. Reads will return the AIN_EF index if all 128 AINs are set to the same value. If values are not the same returns 0xFFFF (65535).

- Data type: UINT32 (type index = 1)
- Readable and writable
- Default value: 0
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0115
- T4:
  - The T4 does not support thermocouple modes.

POWER_AIN
- Address: 48005

The current ON/OFF state of the analog input module.

- Data type: UINT16 (type index = 0)
- Readable and writable
- T8:
  - Minimum firmware version: 1.0013
- T7:
  - Minimum firmware version: 0.8600

POWER_AIN_DEFAULT
- Address: 48055

The ON/OFF state of the analog input module after a power-cycle to the device. Provided to optionally reduce power consumption.

- Data type: UINT16 (type index = 0)
- Readable and writable
- T8:
  - Minimum firmware version: 1.0013
- T7:
  - Minimum firmware version: 0.8600

TEMPERATURE_AIR_K
- Address: 60050

Returns the estimated ambient air temperature just outside of the device in its red plastic enclosure. This register is equal to TEMPERATURE_DEVICE_K - 4.3. If Ethernet and/or WiFi is enabled, subtract an extra 0.6 for each.

- Data type: FLOAT32 (type index = 3)
- Read-only
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0105
- T4:
  - Minimum firmware version: 0.2020

TEMPERATURE_DEVICE_K
- Address: 60052

Takes a reading from the internal temperature sensor using range=+/-10V and resolution=8, and applies the formula Volts*-92.6+467.6 to return kelvins.

- Data type: FLOAT32 (type index = 3)
- Read-only
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - The internal temperature sensor, AIN14, is internally connected to an LM94021 (U24) with GS=10 which is physically located on the bottom of the PCB between the AIN0/1 and AIN2/3 screw-terminals.
  - Minimum firmware version: 1.0105
- T4:
  - The internal temperature sensor is internally connected to an LM94021 (U24) with GS=10 which is physically located on the top of the PCB behind the VS screw terminal of the FIO4

and FIO5 screw terminal block.
    - Minimum firmware version: 0.2020

CALIBRATION_CONSTANTS_STATUS
- Address: 60091

T8 Only. Returns the status of the calibration data saved in flash. 1 = calibration looks good. 0 = calibration invalid.

- Data type: FLOAT32 (type index = 3)
- Read-only
- T8:
    - Minimum firmware version: 0.0123

CALIBRATION_SOURCE
- Address: 60092

T8 Only. Sets and reads the source of the current set of calibration constants. 0 = no calibration. 1 = binary calibration, 2 = Calibration set from Flash, 3 = Nominal Calibration. When setting to 2, the constants will be checked and if invalid 1 will be used instead.

- Data type: UINT16 (type index = 0)
- Readable and writable
- T8:
    - Minimum firmware version: 0.0123

AIN#(0:253)_BINARY
- Starting Address: 50000

Returns the 24-bit binary representation of the specified analog input. If you prefer 16-bit representation, simply divide this by 256. This is for command-response only. Stream always returns binary and LJM applies cal constants, so the LJM config flag LJM_STREAM_AIN_BINARY is used to get binary values.

- Data type: UINT32 (type index = 1)
- Read-only

Expanded Names	Addresses
AIN0_BINARY, AIN1_BINARY, AIN2_BINARY, AIN3_BINARY,	50000, 50002, 50004, 50006,
AIN4_BINARY, AIN5_BINARY, AIN6_BINARY, AIN7_BINARY,	50008, 50010, 50012, 50014,
AIN8_BINARY, AIN9_BINARY, AIN10_BINARY, AIN11_BINARY,	50016, 50018, 50020, 50022,
AIN12_BINARY, AIN13_BINARY, AIN14_BINARY, AIN15_BINARY,	50024, 50026, 50028, 50030,
AIN16_BINARY, AIN17_BINARY, AIN18_BINARY, AIN19_BINARY,	50032, 50034, 50036, 50038,
AIN20_BINARY, AIN21_BINARY, AIN22_BINARY, AIN23_BINARY,	50040, 50042, 50044, 50046,
AIN24_BINARY, AIN25_BINARY, AIN26_BINARY, AIN27_BINARY,	50048, 50050, 50052, 50054,
AIN28_BINARY, AIN29_BINARY, AIN30_BINARY, AIN31_BINARY,	50056, 50058, 50060, 50062,
AIN32_BINARY, AIN33_BINARY, AIN34_BINARY, AIN35_BINARY,	50064, 50066, 50068, 50070,
AIN36_BINARY, AIN37_BINARY, AIN38_BINARY, AIN39_BINARY,	50072, 50074, 50076, 50078,
AIN40_BINARY, AIN41_BINARY, AIN42_BINARY, AIN43_BINARY,	50080, 50082, 50084, 50086,
AIN44_BINARY, AIN45_BINARY, AIN46_BINARY, AIN47_BINARY,	50088, 50090, 50092, 50094,
AIN48_BINARY, AIN49_BINARY, AIN50_BINARY, AIN51_BINARY,	50096, 50098, 50100, 50102,
AIN52_BINARY, AIN53_BINARY, AIN54_BINARY, AIN55_BINARY,	50104, 50106, 50108, 50110,
AIN56_BINARY, AIN57_BINARY, AIN58_BINARY, AIN59_BINARY,	50112, 50114, 50116, 50118,
AIN60_BINARY, AIN61_BINARY, AIN62_BINARY, AIN63_BINARY,	50120, 50122, 50124, 50126,
AIN64_BINARY, AIN65_BINARY, AIN66_BINARY, AIN67_BINARY,	50128, 50130, 50132, 50134,
AIN68_BINARY, AIN69_BINARY, AIN70_BINARY, AIN71_BINARY,	50136, 50138, 50140, 50142,
AIN72_BINARY, AIN73_BINARY, AIN74_BINARY, AIN75_BINARY,	50144, 50146, 50148, 50150,
AIN76_BINARY, AIN77_BINARY, AIN78_BINARY, AIN79_BINARY,	50152, 50154, 50156, 50158,
AIN80_BINARY, AIN81_BINARY, AIN82_BINARY, AIN83_BINARY,	50160, 50162, 50164, 50166,
AIN84_BINARY, AIN85_BINARY, AIN86_BINARY, AIN87_BINARY,	50168, 50170, 50172, 50174,
AIN88_BINARY, AIN89_BINARY, AIN90_BINARY, AIN91_BINARY,	50176, 50178, 50180, 50182,

AIN92_BINARY, AIN93_BINARY, AIN94_BINARY, AIN95_BINARY,	50184, 50186, 50188, 50190,
AIN96_BINARY, AIN97_BINARY, AIN98_BINARY, AIN99_BINARY,	50192, 50194, 50196, 50198,
AIN100_BINARY, AIN101_BINARY, AIN102_BINARY, AIN103_BINARY,	50200, 50202, 50204, 50206,
AIN104_BINARY, AIN105_BINARY, AIN106_BINARY, AIN107_BINARY,	50208, 50210, 50212, 50214,
AIN108_BINARY, AIN109_BINARY, AIN110_BINARY, AIN111_BINARY,	50216, 50218, 50220, 50222,
AIN112_BINARY, AIN113_BINARY, AIN114_BINARY, AIN115_BINARY,	50224, 50226, 50228, 50230,
AIN116_BINARY, AIN117_BINARY, AIN118_BINARY, AIN119_BINARY,	50232, 50234, 50236, 50238,
AIN120_BINARY, AIN121_BINARY, AIN122_BINARY, AIN123_BINARY,	50240, 50242, 50244, 50246,
AIN124_BINARY, AIN125_BINARY, AIN126_BINARY, AIN127_BINARY,	50248, 50250, 50252, 50254,
AIN128_BINARY, AIN129_BINARY, AIN130_BINARY, AIN131_BINARY,	50256, 50258, 50260, 50262,
AIN132_BINARY, AIN133_BINARY, AIN134_BINARY, AIN135_BINARY,	50264, 50266, 50268, 50270,
AIN136_BINARY, AIN137_BINARY, AIN138_BINARY, AIN139_BINARY,	50272, 50274, 50276, 50278,
AIN140_BINARY, AIN141_BINARY, AIN142_BINARY, AIN143_BINARY,	50280, 50282, 50284, 50286,
AIN144_BINARY, AIN145_BINARY, AIN146_BINARY, AIN147_BINARY,	50288, 50290, 50292, 50294,
AIN148_BINARY, AIN149_BINARY, AIN150_BINARY, AIN151_BINARY,	50296, 50298, 50300, 50302,
AIN152_BINARY, AIN153_BINARY, AIN154_BINARY, AIN155_BINARY,	50304, 50306, 50308, 50310,
AIN156_BINARY, AIN157_BINARY, AIN158_BINARY, AIN159_BINARY,	50312, 50314, 50316, 50318,
AIN160_BINARY, AIN161_BINARY, AIN162_BINARY, AIN163_BINARY,	50320, 50322, 50324, 50326,
AIN164_BINARY, AIN165_BINARY, AIN166_BINARY, AIN167_BINARY,	50328, 50330, 50332, 50334,
AIN168_BINARY, AIN169_BINARY, AIN170_BINARY, AIN171_BINARY,	50336, 50338, 50340, 50342,
AIN172_BINARY, AIN173_BINARY, AIN174_BINARY, AIN175_BINARY,	50344, 50346, 50348, 50350,
AIN176_BINARY, AIN177_BINARY, AIN178_BINARY, AIN179_BINARY,	50352, 50354, 50356, 50358,
AIN180_BINARY, AIN181_BINARY, AIN182_BINARY, AIN183_BINARY,	50360, 50362, 50364, 50366,
AIN184_BINARY, AIN185_BINARY, AIN186_BINARY, AIN187_BINARY,	50368, 50370, 50372, 50374,
AIN188_BINARY, AIN189_BINARY, AIN190_BINARY, AIN191_BINARY,	50376, 50378, 50380, 50382,
AIN192_BINARY, AIN193_BINARY, AIN194_BINARY, AIN195_BINARY,	50384, 50386, 50388, 50390,
AIN196_BINARY, AIN197_BINARY, AIN198_BINARY, AIN199_BINARY,	50392, 50394, 50396, 50398,
AIN200_BINARY, AIN201_BINARY, AIN202_BINARY, AIN203_BINARY,	50400, 50402, 50404, 50406,
AIN204_BINARY, AIN205_BINARY, AIN206_BINARY, AIN207_BINARY,	50408, 50410, 50412, 50414,
AIN208_BINARY, AIN209_BINARY, AIN210_BINARY, AIN211_BINARY,	50416, 50418, 50420, 50422,
AIN212_BINARY, AIN213_BINARY, AIN214_BINARY, AIN215_BINARY,	50424, 50426, 50428, 50430,
AIN216_BINARY, AIN217_BINARY, AIN218_BINARY, AIN219_BINARY,	50432, 50434, 50436, 50438,
AIN220_BINARY, AIN221_BINARY, AIN222_BINARY, AIN223_BINARY,	50440, 50442, 50444, 50446,
AIN224_BINARY, AIN225_BINARY, AIN226_BINARY, AIN227_BINARY,	50448, 50450, 50452, 50454,
AIN228_BINARY, AIN229_BINARY, AIN230_BINARY, AIN231_BINARY,	50456, 50458, 50460, 50462,
AIN232_BINARY, AIN233_BINARY, AIN234_BINARY, AIN235_BINARY,	50464, 50466, 50468, 50470,
AIN236_BINARY, AIN237_BINARY, AIN238_BINARY, AIN239_BINARY,	50472, 50474, 50476, 50478,
AIN240_BINARY, AIN241_BINARY, AIN242_BINARY, AIN243_BINARY,	50480, 50482, 50484, 50486,
AIN244_BINARY, AIN245_BINARY, AIN246_BINARY, AIN247_BINARY,	50488, 50490, 50492, 50494,
AIN248_BINARY, AIN249_BINARY, AIN250_BINARY, AIN251_BINARY,	50496, 50498, 50500, 50502,
AIN252_BINARY, AIN253_BINARY	50504, 50506

TEMPERATURE#(0:6)_BINARY
- Starting Address: 50600

T8 Only. Returns the 24-bit binary representation of the voltage output of the temperature sensor for the specified analog input. If you prefer 16-bit representation, simply divide this by 256. This is for command-response only. Stream always returns binary and LJM applies cal constants, so the LJM config flag LJM_STREAM_AIN_BINARY is used to get binary values.

- Data type: UINT32 (type index = 1)
- Read-only

Expanded Names	Addresses
TEMPERATURE0_BINARY, TEMPERATURE1_BINARY, TEMPERATURE2_BINARY, TEMPERATURE3_BINARY, TEMPERATURE4_BINARY, TEMPERATURE5_BINARY, TEMPERATURE6_BINARY	50600, 50602, 50604, 50606, 50608, 50610, 50612

AIN#(0:6)_BINARY_CAPTURE
- Starting Address: 50650

T8 Only. Returns the saved 24-bit binary representation of the specified analog input. If you prefer 16-bit representation, simply divide this by 256. This is for command-response only. Stream always returns

binary and LJM applies cal constants, so the LJM config flag LJM_STREAM_AIN_BINARY is used to get binary values.

- Data type: UINT32 (type index = 1)
- Read-only

Expanded Names	Addresses
AIN0_BINARY_CAPTURE, AIN1_BINARY_CAPTURE, AIN2_BINARY_CAPTURE, AIN3_BINARY_CAPTURE, AIN4_BINARY_CAPTURE, AIN5_BINARY_CAPTURE, AIN6_BINARY_CAPTURE	50650, 50652, 50654, 50656, 50658, 50660, 50662

TEMPERATURE#(0:6)_BINARY_CAPTURE
- Starting Address: 50700

T8 Only. Returns the saved 24-bit binary representation of the voltage output of the temperature sensor for the specified analog input. If you prefer 16-bit representation, simply divide this by 256. This is for command-response only. Stream always returns binary and LJM applies cal constants, so the LJM config flag LJM_STREAM_AIN_BINARY is used to get binary values.

- Data type: UINT32 (type index = 1)
- Read-only

Expanded Names	Addresses
TEMPERATURE0_BINARY_CAPTURE, TEMPERATURE1_BINARY_CAPTURE, TEMPERATURE2_BINARY_CAPTURE, TEMPERATURE3_BINARY_CAPTURE, TEMPERATURE4_BINARY_CAPTURE, TEMPERATURE5_BINARY_CAPTURE, TEMPERATURE6_BINARY_CAPTURE	50700, 50702, 50704, 50706, 50708, 50710, 50712

All CORE TAGS:

Name	Start Address	Type	Access
AIN#(0:253)	0	FLOAT32	R
TEMPERATURE#(0:6)	600	FLOAT32	R
AIN#(0:6)_CAPTURE	650	FLOAT32	R
TEMPERATURE#(0:6)_CAPTURE	700	FLOAT32	R
DAC#(0:0)	1000	FLOAT32	R/W
FIO#(0:6)	2000	UINT16	R/W
EIO#(0:6)	2008	UINT16	R/W
CIO#(0:2)	2016	UINT16	R/W
MIO#(0:1)	2020	UINT16	R/W
AIN#(0:148)_EF_READ_A	7000	FLOAT32	R
TDAC#(0:20)	30000	FLOAT32	W
SBUS#(0:21)_TEMP	30100	FLOAT32	R
SBUS#(0:21)_RH	30150	FLOAT32	R
DIO#(0:21)_EF_READ_A	3000	UINT32	R
TEMPERATURE_AIR_K	60050	FLOAT32	R
TEMPERATURE_DEVICE_K	60052	FLOAT32	R
CALIBRATION_CONSTANTS_STATUS	60091	FLOAT32	R
CALIBRATION_SOURCE	60092	UINT16	R/W
LAST_ERR_DETAIL	55000	UINT16	R
LAST_MB_ERR	55001	UINT16	R
LAST_ERR_FRAME	55002	UINT16	R
LAST_ERR_TRANSACTION_ID	55003	UINT16	R

AIN#(0:253)
- Starting Address: 0

Returns the voltage of the specified analog input.

- Data type: FLOAT32 (type index = 3)
- Read-only
- This register may be streamed

Expanded Names	Addresses
AIN0, AIN1, AIN2, AIN3, AIN4, AIN5, AIN6, AIN7, AIN8, AIN9, AIN10, AIN11, AIN12, AIN13, AIN14, AIN15, AIN16, AIN17, AIN18, AIN19, AIN20, AIN21, AIN22, AIN23, AIN24, AIN25, AIN26, AIN27, AIN28, AIN29, AIN30, AIN31, AIN32, AIN33, AIN34, AIN35, AIN36, AIN37, AIN38, AIN39, AIN40, AIN41, AIN42, AIN43, AIN44, AIN45, AIN46, AIN47, AIN48, AIN49, AIN50, AIN51, AIN52, AIN53, AIN54, AIN55, AIN56, AIN57, AIN58, AIN59, AIN60, AIN61, AIN62, AIN63, AIN64, AIN65, AIN66, AIN67, AIN68, AIN69, AIN70, AIN71, AIN72, AIN73, AIN74, AIN75, AIN76, AIN77, AIN78, AIN79, AIN80, AIN81, AIN82, AIN83, AIN84, AIN85, AIN86, AIN87, AIN88, AIN89, AIN90, AIN91, AIN92, AIN93, AIN94, AIN95, AIN96, AIN97, AIN98, AIN99, AIN100, AIN101, AIN102, AIN103, AIN104, AIN105, AIN106, AIN107, AIN108, AIN109, AIN110, AIN111, AIN112, AIN113, AIN114, AIN115, AIN116, AIN117, AIN118, AIN119, AIN120, AIN121, AIN122, AIN123, AIN124, AIN125, AIN126, AIN127, AIN128, AIN129, AIN130, AIN131, AIN132, AIN133, AIN134, AIN135, AIN136, AIN137, AIN138, AIN139, AIN140, AIN141, AIN142, AIN143, AIN144, AIN145, AIN146, AIN147, AIN148, AIN149, AIN150, AIN151, AIN152, AIN153, AIN154, AIN155, AIN156, AIN157, AIN158, AIN159, AIN160, AIN161, AIN162, AIN163, AIN164, AIN165, AIN166, AIN167, AIN168, AIN169, AIN170, AIN171, AIN172, AIN173, AIN174, AIN175, AIN176, AIN177, AIN178, AIN179, AIN180, AIN181, AIN182, AIN183, AIN184, AIN185, AIN186, AIN187, AIN188, AIN189, AIN190, AIN191, AIN192, AIN193, AIN194, AIN195, AIN196, AIN197, AIN198, AIN199, AIN200, AIN201, AIN202, AIN203, AIN204, AIN205, AIN206, AIN207, AIN208, AIN209, AIN210, AIN211, AIN212, AIN213, AIN214, AIN215, AIN216, AIN217, AIN218, AIN219, AIN220, AIN221, AIN222, AIN223, AIN224, AIN225, AIN226, AIN227, AIN228, AIN229, AIN230, AIN231, AIN232, AIN233, AIN234, AIN235, AIN236, AIN237, AIN238, AIN239, AIN240, AIN241, AIN242, AIN243, AIN244, AIN245, AIN246, AIN247, AIN248, AIN249, AIN250, AIN251, AIN252, AIN253	0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42, 44, 46, 48, 50, 52, 54, 56, 58, 60, 62, 64, 66, 68, 70, 72, 74, 76, 78, 80, 82, 84, 86, 88, 90, 92, 94, 96, 98, 100, 102, 104, 106, 108, 110, 112, 114, 116, 118, 120, 122, 124, 126, 128, 130, 132, 134, 136, 138, 140, 142, 144, 146, 148, 150, 152, 154, 156, 158, 160, 162, 164, 166, 168, 170, 172, 174, 176, 178, 180, 182, 184, 186, 188, 190, 192, 194, 196, 198, 200, 202, 204, 206, 208, 210, 212, 214, 216, 218, 220, 222, 224, 226, 228, 230, 232, 234, 236, 238, 240, 242, 244, 246, 248, 250, 252, 254, 256, 258, 260, 262, 264, 266, 268, 270, 272, 274, 276, 278, 280, 282, 284, 286, 288, 290, 292, 294, 296, 298, 300, 302, 304, 306, 308, 310, 312, 314, 316, 318, 320, 322, 324, 326, 328, 330, 332, 334, 336, 338, 340, 342, 344, 346, 348, 350, 352, 354, 356, 358, 360, 362, 364, 366, 368, 370, 372, 374, 376, 378, 380, 382, 384, 386, 388, 390, 392, 394, 396, 398, 400, 402, 404, 406, 408, 410, 412, 414, 416, 418, 420, 422, 424, 426, 428, 430, 432, 434, 436, 438, 440, 442, 444, 446, 448, 450, 452, 454, 456, 458, 460, 462, 464, 466, 468, 470, 472, 474, 476, 478, 480, 482, 484, 486, 488, 490, 492, 494, 496, 498, 500, 502, 504, 506

TEMPERATURE#(0:6)
- Starting Address: 600

T8 Only. Returns the temperature of the specified analog input. And saves AIN and temperature inputs for all channels.

- Data type: FLOAT32 (type index = 3)
- Read-only
- This register may be streamed

Expanded Names	Addresses
TEMPERATURE0, TEMPERATURE1, TEMPERATURE2, TEMPERATURE3, TEMPERATURE4, TEMPERATURE5, TEMPERATURE6	600, 602, 604, 606, 608, 610, 612

AIN#(0:6)_CAPTURE
- Starting Address: 650

T8 Only. Returns the saved voltage of the specified analog input.

- Data type: FLOAT32 (type index = 3)
- Read-only
- This register may be streamed

Expanded Names	Addresses
AIN0_CAPTURE, AIN1_CAPTURE, AIN2_CAPTURE, AIN3_CAPTURE, AIN4_CAPTURE, AIN5_CAPTURE, AIN6_CAPTURE	650, 652, 654, 656, 658, 660, 662

TEMPERATURE#(0:6)_CAPTURE
- Starting Address: 700

T8 Only. Returns the saved temperature of the specified analog input.

- Data type: FLOAT32 (type index = 3)
- Read-only
- This register may be streamed

Expanded Names	Addresses
TEMPERATURE0_CAPTURE, TEMPERATURE1_CAPTURE, TEMPERATURE2_CAPTURE, TEMPERATURE3_CAPTURE, TEMPERATURE4_CAPTURE, TEMPERATURE5_CAPTURE, TEMPERATURE6_CAPTURE	700, 702, 704, 706, 708, 710, 712

DAC#(0:0)
- Address: 1000

Pass a voltage for the specified analog output.

- Data type: FLOAT32 (type index = 3)
- Readable and writable

FIO#(0:6)
- Starting Address: 2000

Read or set the state of 1 bit of digital I/O. Also configures the direction to input or output. Read 0=Low AND 1=High. Write 0=Low AND 1=High.

- Data type: UINT16 (type index = 0)
- Readable and writable
- This register may be streamed

Expanded Names	Addresses
FIO0, FIO1, FIO2, FIO3, FIO4, FIO5, FIO6	2000, 2001, 2002, 2003, 2004, 2005, 2006

EIO#(0:6)
- Starting Address: 2008

Read or set the state of 1 bit of digital I/O. Also configures the direction to input or output. Read 0=Low AND 1=High. Write 0=Low AND 1=High.

- Data type: UINT16 (type index = 0)
- Readable and writable
- This register may be streamed

Expanded Names	Addresses
EIO0, EIO1, EIO2, EIO3, EIO4, EIO5, EIO6	2008, 2009, 2010, 2011, 2012, 2013, 2014

CIO#(0:2)
- Starting Address: 2016

Read or set the state of 1 bit of digital I/O. Also configures the direction to input or output. Read 0=Low AND 1=High. Write 0=Low AND 1=High.

- Data type: UINT16 (type index = 0)
- Readable and writable
- This register may be streamed

Expanded Names	Addresses
CIO0, CIO1, CIO2	2016, 2017, 2018

MIO#(0:1)
- Starting Address: 2020

Read or set the state of 1 bit of digital I/O. Also configures the direction to input or output. Read 0=Low AND 1=High. Write 0=Low AND 1=High.

- Data type: UINT16 (type index = 0)
- Readable and writable
- This register may be streamed

Expanded Names	Addresses
MIO0, MIO1	2020, 2021

AIN#(0:148)_EF_READ_A
- Starting Address: 7000

Function dependent on selected feature index.

- Data type: FLOAT32 (type index = 3)
- Read-only
- Default value: 0
- This register uses system RAM. The maximum RAM is 64KB. For more information, see 4.4 RAM
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0030

Expanded Names	Addresses

AIN0_EF_READ_A, AIN1_EF_READ_A, AIN2_EF_READ_A, AIN3_EF_READ_A, AIN4_EF_READ_A, AIN5_EF_READ_A, AIN6_EF_READ_A, AIN7_EF_READ_A, AIN8_EF_READ_A, AIN9_EF_READ_A, AIN10_EF_READ_A, AIN11_EF_READ_A, AIN12_EF_READ_A, AIN13_EF_READ_A, AIN14_EF_READ_A, AIN15_EF_READ_A, AIN16_EF_READ_A, AIN17_EF_READ_A, AIN18_EF_READ_A, AIN19_EF_READ_A, AIN20_EF_READ_A, AIN21_EF_READ_A, AIN22_EF_READ_A, AIN23_EF_READ_A, AIN24_EF_READ_A, AIN25_EF_READ_A, AIN26_EF_READ_A, AIN27_EF_READ_A, AIN28_EF_READ_A, AIN29_EF_READ_A, AIN30_EF_READ_A, AIN31_EF_READ_A, AIN32_EF_READ_A, AIN33_EF_READ_A, AIN34_EF_READ_A, AIN35_EF_READ_A, AIN36_EF_READ_A, AIN37_EF_READ_A, AIN38_EF_READ_A, AIN39_EF_READ_A, AIN40_EF_READ_A, AIN41_EF_READ_A, AIN42_EF_READ_A, AIN43_EF_READ_A, AIN44_EF_READ_A, AIN45_EF_READ_A, AIN46_EF_READ_A, AIN47_EF_READ_A, AIN48_EF_READ_A, AIN49_EF_READ_A, AIN50_EF_READ_A, AIN51_EF_READ_A, AIN52_EF_READ_A, AIN53_EF_READ_A, AIN54_EF_READ_A, AIN55_EF_READ_A, AIN56_EF_READ_A, AIN57_EF_READ_A, AIN58_EF_READ_A, AIN59_EF_READ_A, AIN60_EF_READ_A, AIN61_EF_READ_A, AIN62_EF_READ_A, AIN63_EF_READ_A, AIN64_EF_READ_A, AIN65_EF_READ_A, AIN66_EF_READ_A, AIN67_EF_READ_A, AIN68_EF_READ_A, AIN69_EF_READ_A, AIN70_EF_READ_A, AIN71_EF_READ_A, AIN72_EF_READ_A, AIN73_EF_READ_A, AIN74_EF_READ_A, AIN75_EF_READ_A, AIN76_EF_READ_A, AIN77_EF_READ_A, AIN78_EF_READ_A, AIN79_EF_READ_A, AIN80_EF_READ_A, AIN81_EF_READ_A, AIN82_EF_READ_A, AIN83_EF_READ_A, AIN84_EF_READ_A, AIN85_EF_READ_A, AIN86_EF_READ_A, AIN87_EF_READ_A, AIN88_EF_READ_A, AIN89_EF_READ_A, AIN90_EF_READ_A, AIN91_EF_READ_A, AIN92_EF_READ_A, AIN93_EF_READ_A, AIN94_EF_READ_A, AIN95_EF_READ_A, AIN96_EF_READ_A, AIN97_EF_READ_A, AIN98_EF_READ_A, AIN99_EF_READ_A, AIN100_EF_READ_A, AIN101_EF_READ_A, AIN102_EF_READ_A, AIN103_EF_READ_A, AIN104_EF_READ_A, AIN105_EF_READ_A, AIN106_EF_READ_A, AIN107_EF_READ_A, AIN108_EF_READ_A, AIN109_EF_READ_A, AIN110_EF_READ_A, AIN111_EF_READ_A, AIN112_EF_READ_A, AIN113_EF_READ_A, AIN114_EF_READ_A, AIN115_EF_READ_A, AIN116_EF_READ_A, AIN117_EF_READ_A, AIN118_EF_READ_A, AIN119_EF_READ_A, AIN120_EF_READ_A, AIN121_EF_READ_A, AIN122_EF_READ_A, AIN123_EF_READ_A, AIN124_EF_READ_A, AIN125_EF_READ_A, AIN126_EF_READ_A, AIN127_EF_READ_A, AIN128_EF_READ_A, AIN129_EF_READ_A, AIN130_EF_READ_A, AIN131_EF_READ_A, AIN132_EF_READ_A, AIN133_EF_READ_A, AIN134_EF_READ_A, AIN135_EF_READ_A, AIN136_EF_READ_A, AIN137_EF_READ_A, AIN138_EF_READ_A, AIN139_EF_READ_A, AIN140_EF_READ_A, AIN141_EF_READ_A, AIN142_EF_READ_A, AIN143_EF_READ_A, AIN144_EF_READ_A, AIN145_EF_READ_A, AIN146_EF_READ_A, AIN147_EF_READ_A, AIN148_EF_READ_A	7000, 7002, 7004, 7006, 7008, 7010, 7012, 7014, 7016, 7018, 7020, 7022, 7024, 7026, 7028, 7030, 7032, 7034, 7036, 7038, 7040, 7042, 7044, 7046, 7048, 7050, 7052, 7054, 7056, 7058, 7060, 7062, 7064, 7066, 7068, 7070, 7072, 7074, 7076, 7078, 7080, 7082, 7084, 7086, 7088, 7090, 7092, 7094, 7096, 7098, 7100, 7102, 7104, 7106, 7108, 7110, 7112, 7114, 7116, 7118, 7120, 7122, 7124, 7126, 7128, 7130, 7132, 7134, 7136, 7138, 7140, 7142, 7144, 7146, 7148, 7150, 7152, 7154, 7156, 7158, 7160, 7162, 7164, 7166, 7168, 7170, 7172, 7174, 7176, 7178, 7180, 7182, 7184, 7186, 7188, 7190, 7192, 7194, 7196, 7198, 7200, 7202, 7204, 7206, 7208, 7210, 7212, 7214, 7216, 7218, 7220, 7222, 7224, 7226, 7228, 7230, 7232, 7234, 7236, 7238, 7240, 7242, 7244, 7246, 7248, 7250, 7252, 7254, 7256, 7258, 7260, 7262, 7264, 7266, 7268, 7270, 7272, 7274, 7276, 7278, 7280, 7282, 7284, 7286, 7288, 7290, 7292, 7294, 7296

TDAC#(0:20)
- Starting Address: 30000

Update a voltage output on a connected LJTick-DAC accessory. Even TDAC# = DACA, Odd TDAC# = DACB. For instance, if LJTick-DAC accessory is connected to FIO2/FIO3 block on main device, TDAC2 corresponds with DACA, and TDAC3 corresponds with DACB.

- Data type: FLOAT32 (type index = 3)
- Write-only
- Default value: 0

Expanded Names	Addresses
TDAC0, TDAC1, TDAC2, TDAC3, TDAC4, TDAC5, TDAC6, TDAC7, TDAC8, TDAC9, TDAC10, TDAC11, TDAC12, TDAC13, TDAC14, TDAC15, TDAC16, TDAC17, TDAC18, TDAC19, TDAC20	30000, 30002, 30004, 30006, 30008, 30010, 30012, 30014, 30016, 30018, 30020, 30022, 30024, 30026, 30028, 30030, 30032, 30034, 30036, 30038, 30040

SBUS#(0:21)_TEMP

- Starting Address: 30100

Reads temperature in Kelvin from an SBUS sensor (EI-1050/SHT1x/SHT7x). SBUS# is the DIO line for the EI-1050 enable. If SBUS# is the same as the value specified for data or clock line, there will be no control of an enable line.

- Data type: FLOAT32 (type index = 3)
- Read-only
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0056

Expanded Names	Addresses
SBUS0_TEMP, SBUS1_TEMP, SBUS2_TEMP, SBUS3_TEMP, SBUS4_TEMP, SBUS5_TEMP, SBUS6_TEMP, SBUS7_TEMP, SBUS8_TEMP, SBUS9_TEMP, SBUS10_TEMP, SBUS11_TEMP, SBUS12_TEMP, SBUS13_TEMP, SBUS14_TEMP, SBUS15_TEMP, SBUS16_TEMP, SBUS17_TEMP, SBUS18_TEMP, SBUS19_TEMP, SBUS20_TEMP, SBUS21_TEMP	30100, 30102, 30104, 30106, 30108, 30110, 30112, 30114, 30116, 30118, 30120, 30122, 30124, 30126, 30128, 30130, 30132, 30134, 30136, 30138, 30140, 30142

SBUS#(0:21)_RH
- Starting Address: 30150

Reads humidity in % from an external SBUS sensor (EI-1050/SHT1x/SHT7x). # is the DIO line for the EI-1050 enable. If # is the same as the value specified for data or clock line, there will be no control of an enable line.

- Data type: FLOAT32 (type index = 3)
- Read-only
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0056

Expanded Names	Addresses
SBUS0_RH, SBUS1_RH, SBUS2_RH, SBUS3_RH, SBUS4_RH, SBUS5_RH, SBUS6_RH, SBUS7_RH, SBUS8_RH, SBUS9_RH, SBUS10_RH, SBUS11_RH, SBUS12_RH, SBUS13_RH, SBUS14_RH, SBUS15_RH, SBUS16_RH, SBUS17_RH, SBUS18_RH, SBUS19_RH, SBUS20_RH, SBUS21_RH	30150, 30152, 30154, 30156, 30158, 30160, 30162, 30164, 30166, 30168, 30170, 30172, 30174, 30176, 30178, 30180, 30182, 30184, 30186, 30188, 30190, 30192

DIO#(0:21)_EF_READ_A
- Starting Address: 3000

Reads an unsigned integer value. The meaning of the integer is dependent on selected feature index.

- Data type: UINT32 (type index = 1)
- Read-only
- This register may be streamed

Expanded Names	Addresses
DIO0_EF_READ_A, DIO1_EF_READ_A, DIO2_EF_READ_A, DIO3_EF_READ_A, DIO4_EF_READ_A, DIO5_EF_READ_A, DIO6_EF_READ_A, DIO7_EF_READ_A, DIO8_EF_READ_A, DIO9_EF_READ_A, DIO10_EF_READ_A, DIO11_EF_READ_A, DIO12_EF_READ_A, DIO13_EF_READ_A, DIO14_EF_READ_A, DIO15_EF_READ_A, DIO16_EF_READ_A, DIO17_EF_READ_A, DIO18_EF_READ_A, DIO19_EF_READ_A, DIO20_EF_READ_A, DIO21_EF_READ_A	3000, 3002, 3004, 3006, 3008, 3010, 3012, 3014, 3016, 3018, 3020, 3022, 3024, 3026, 3028, 3030, 3032, 3034, 3036, 3038, 3040, 3042

TEMPERATURE_AIR_K
- Address: 60050

Returns the estimated ambient air temperature just outside of the device in its red plastic enclosure. This register is equal to TEMPERATURE_DEVICE_K - 4.3. If Ethernet and/or WiFi is enabled, subtract an extra 0.6 for each.

- Data type: FLOAT32 (type index = 3)
- Read-only
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0105
- T4:
  - Minimum firmware version: 0.2020

TEMPERATURE_DEVICE_K
- Address: 60052

Takes a reading from the internal temperature sensor using range=+/-10V and resolution=8, and applies the formula Volts*-92.6+467.6 to return kelvins.

- Data type: FLOAT32 (type index = 3)
- Read-only
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - The internal temperature sensor, AIN14, is internally connected to an LM94021 (U24) with GS=10 which is physically located on the bottom of the PCB between the AIN0/1 and AIN2/3 screw-terminals.
  - Minimum firmware version: 1.0105
- T4:
  - The internal temperature sensor is internally connected to an LM94021 (U24) with GS=10 which is physically located on the top of the PCB behind the VS screw terminal of the FIO4 and FIO5 screw terminal block.
  - Minimum firmware version: 0.2020

CALIBRATION_CONSTANTS_STATUS
- Address: 60091

T8 Only. Returns the status of the calibration data saved in flash. 1 = calibration looks good. 0 = calibration invalid.

- Data type: FLOAT32 (type index = 3)
- Read-only
- T8:
  - Minimum firmware version: 0.0123

CALIBRATION_SOURCE
- Address: 60092

T8 Only. Sets and reads the source of the current set of calibration constants. 0 = no calibration. 1 = binary calibration, 2 = Calibration set from Flash, 3 = Nominal Calibration. When setting to 2, the constants will be checked and if invalid 1 will be used instead.

- Data type: UINT16 (type index = 0)
- Readable and writable
- T8:

    - Minimum firmware version: 0.0123

LAST_ERR_DETAIL
- Address: 55000

Returns the last LabJack error code seen on the device.

- Data type: UINT16 (type index = 0)
- Read-only
- Default value: 0

LAST_MB_ERR
- Address: 55001

Returns the last Modbus TCP error code seen on the device.

- Data type: UINT16 (type index = 0)
- Read-only
- Default value: 0

LAST_ERR_FRAME
- Address: 55002

Returns the frame number that caused the last error.

- Data type: UINT16 (type index = 0)
- Read-only
- Default value: 0

LAST_ERR_TRANSACTION_ID
- Address: 55003

Returns the transaction ID of the Modbus TCP packet that caused the last error.

- Data type: UINT16 (type index = 0)
- Read-only
- Default value: 0

All DAC TAGS:

Name	Start Address	Type	Access
DAC#(0:0)	1000	FLOAT32	R/W
DAC#(0:0)_BINARY	51000	UINT32	W

DAC#(0:0)
- Address: 1000

Pass a voltage for the specified analog output.

- Data type: FLOAT32 (type index = 3)
- Readable and writable

DAC#(0:0)_BINARY
- Address: 51000

Writes binary values to the DACs. Binary values are 16-bit. If the DAC's resolution is less than 16 then the lower bits are ignored. 0 = lowest output, 65535 = highest output.

- Data type: UINT32 (type index = 1)
- Write-only

All DIO TAGS:

Name	Start Address	Type	Access
FIO#(0:6)	2000	UINT16	R/W
EIO#(0:6)	2008	UINT16	R/W
CIO#(0:2)	2016	UINT16	R/W
MIO#(0:1)	2020	UINT16	R/W
FIO_STATE	2500	UINT16	R/W
EIO_STATE	2501	UINT16	R/W
CIO_STATE	2502	UINT16	R/W
MIO_STATE	2503	UINT16	R/W
FIO_EIO_STATE	2580	UINT16	R/W
EIO_CIO_STATE	2581	UINT16	R/W
CIO_MIO_STATE	2582	UINT16	R/W
FIO_DIRECTION	2600	UINT16	R/W
EIO_DIRECTION	2601	UINT16	R/W
CIO_DIRECTION	2602	UINT16	R/W
MIO_DIRECTION	2603	UINT16	R/W
DIO_STATE	2800	UINT32	R/W
DIO_DIRECTION	2850	UINT32	R/W
DIO_PULLDOWN_ENABLE	2870	UINT32	R/W
DIO_PULLUP_DISABLE	2890	UINT32	R/W
DIO_INHIBIT	2900	UINT32	R/W
LED_COMM	2990	UINT16	W
LED_STATUS	2991	UINT16	W
DIO_EF_CLOCK0_ENABLE	44900	UINT16	R/W
DIO_EF_CLOCK0_DIVISOR	44901	UINT16	R/W
DIO_EF_CLOCK0_OPTIONS	44902	UINT32	R/W
DIO_EF_CLOCK0_ROLL_VALUE	44904	UINT32	R/W
DIO_EF_CLOCK1_ENABLE	44910	UINT16	R/W
DIO_EF_CLOCK1_DIVISOR	44911	UINT16	R/W
DIO_EF_CLOCK1_OPTIONS	44912	UINT32	R/W

DIO_EF_CLOCK1_ROLL_VALUE	44914	UINT32	R/W
DIO_EF_CLOCK2_ENABLE	44920	UINT16	R/W
DIO_EF_CLOCK2_DIVISOR	44921	UINT16	R/W
DIO_EF_CLOCK2_OPTIONS	44922	UINT32	R/W
DIO_EF_CLOCK2_ROLL_VALUE	44924	UINT32	R/W
DIO_EF_CLOCK0_COUNT	44908	UINT32	R
DIO_EF_CLOCK1_COUNT	44918	UINT32	R
DIO_EF_CLOCK2_COUNT	44928	UINT32	R
DIO#(0:21)_EF_ENABLE	44000	UINT32	R/W
DIO#(0:21)_EF_INDEX	44100	UINT32	R/W
DIO#(0:21)_EF_OPTIONS	44200	UINT32	R/W
DIO#(0:21)_EF_CONFIG_A	44300	UINT32	R/W
DIO#(0:21)_EF_CONFIG_B	44400	UINT32	R/W
DIO#(0:21)_EF_CONFIG_C	44500	UINT32	R/W
DIO#(0:21)_EF_CONFIG_D	44600	UINT32	R/W
DIO#(0:21)_EF_READ_A	3000	UINT32	R
DIO#(0:21)_EF_READ_A_AND_RESET	3100	UINT32	R
DIO#(0:21)_EF_READ_B	3200	UINT32	R
DIO#(0:21)_EF_READ_A_F	3500	FLOAT32	R
DIO#(0:21)_EF_READ_A_F_AND_RESET	3600	FLOAT32	R
DIO#(0:21)_EF_READ_B_F	3700	FLOAT32	R
DIO#(0:21)_EF_EASY_FREQUENCY_IN	45000	FLOAT32	R/W

FIO#(0:6)
- Starting Address: 2000

Read or set the state of 1 bit of digital I/O. Also configures the direction to input or output. Read 0=Low AND 1=High. Write 0=Low AND 1=High.

- Data type: UINT16 (type index = 0)
- Readable and writable
- This register may be streamed

Expanded Names	Addresses
FIO0, FIO1, FIO2, FIO3, FIO4, FIO5, FIO6	2000, 2001, 2002, 2003, 2004, 2005, 2006

EIO#(0:6)
- Starting Address: 2008

Read or set the state of 1 bit of digital I/O. Also configures the direction to input or output. Read 0=Low AND 1=High. Write 0=Low AND 1=High.

- Data type: UINT16 (type index = 0)
- Readable and writable
- This register may be streamed

Expanded Names	Addresses
EIO0, EIO1, EIO2, EIO3, EIO4, EIO5, EIO6	2008, 2009, 2010, 2011, 2012, 2013, 2014

CIO#(0:2)
- Starting Address: 2016

Read or set the state of 1 bit of digital I/O. Also configures the direction to input or output. Read 0=Low AND 1=High. Write 0=Low AND 1=High.

- Data type: UINT16 (type index = 0)

- Readable and writable
- This register may be streamed

Expanded Names	Addresses
CIO0, CIO1, CIO2	2016, 2017, 2018

MIO#(0:1)
- Starting Address: 2020

Read or set the state of 1 bit of digital I/O. Also configures the direction to input or output. Read 0=Low AND 1=High. Write 0=Low AND 1=High.

- Data type: UINT16 (type index = 0)
- Readable and writable
- This register may be streamed

Expanded Names	Addresses
MIO0, MIO1	2020, 2021

FIO_STATE
- Address: 2500

Read or write the state of the 8 bits of FIO in a single binary-encoded value. 0=Low AND 1=High. Does not configure direction. Reading lines set to output returns the current logic levels on the terminals, not necessarily the output states written. The upper 8-bits of this value are inhibits.

- Data type: UINT16 (type index = 0)
- Readable and writable
- This register may be streamed
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0000
- T4:
  - Minimum firmware version: 0.2000

EIO_STATE
- Address: 2501

Read or write the state of the 8 bits of EIO in a single binary-encoded value. 0=Low AND 1=High. Does not configure direction. Reading lines set to output returns the current logic levels on the terminals, not necessarily the output states written. The upper 8-bits of this value are inhibits.

- Data type: UINT16 (type index = 0)
- Readable and writable
- This register may be streamed
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0000
- T4:
  - Minimum firmware version: 0.2000

CIO_STATE
- Address: 2502

Read or write the state of the 4 bits of CIO in a single binary-encoded value. 0=Low AND 1=High. Does

not configure direction. Reading lines set to output returns the current logic levels on the terminals, not necessarily the output states written. The upper 8-bits of this value are inhibits.

- Data type: UINT16 (type index = 0)
- Readable and writable
- This register may be streamed
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0000
- T4:
  - Minimum firmware version: 0.2000

MIO_STATE
- Address: 2503

Read or write the state of the 3 bits of MIO in a single binary-encoded value. 0=Low AND 1=High. Does not configure direction. Reading lines set to output returns the current logic levels on the terminals, not necessarily the output states written. The upper 8-bits of this value are inhibits.

- Data type: UINT16 (type index = 0)
- Readable and writable
- This register may be streamed
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0000
- T4:
  - Minimum firmware version: 0.2000

FIO_EIO_STATE
- Address: 2580

Read or write the state of the 16 bits of FIO-EIO in a single binary-encoded value. 0=Low AND 1=High. Does not configure direction. Reading lines set to output returns the current logic levels on the terminals, not necessarily the output states written.

- Data type: UINT16 (type index = 0)
- Readable and writable
- This register may be streamed
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0005
- T4:
  - Minimum firmware version: 0.2000

EIO_CIO_STATE
- Address: 2581

Read or write the state of the 12 bits of EIO-CIO in a single binary-encoded value. 0=Low AND 1=High. Does not configure direction. Reading lines set to output returns the current logic levels on the terminals, not necessarily the output states written. As of firmware 1.0172, MIO states are included in the upper nibble of the CIO byte.

- Data type: UINT16 (type index = 0)

- Readable and writable
- This register may be streamed
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0005
- T4:
  - Minimum firmware version: 0.2000

CIO_MIO_STATE
- Address: 2582

Read or write the state of the 12 bits of CIO-MIO in a single binary-encoded value. 0=Low AND 1=High. Does not configure direction. Reading lines set to output returns the current logic levels on the terminals, not necessarily the output states written.

- Data type: UINT16 (type index = 0)
- Readable and writable
- This register may be streamed
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0172
- T4:
  - Minimum firmware version: 0.2000

FIO_DIRECTION
- Address: 2600

Read or write the direction of the 8 bits of FIO in a single binary-encoded value. 0=Input and 1=Output. The upper 8-bits of this value are inhibits.

- Data type: UINT16 (type index = 0)
- Readable and writable
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 0.9402
- T4:
  - Minimum firmware version: 0.2000

EIO_DIRECTION
- Address: 2601

Read or write the direction of the 8 bits of EIO in a single binary-encoded value. 0=Input and 1=Output. The upper 8-bits of this value are inhibits.

- Data type: UINT16 (type index = 0)
- Readable and writable
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 0.9402
- T4:
  - Minimum firmware version: 0.2000

CIO_DIRECTION
- Address: 2602

Read or write the direction of the 4 bits of CIO in a single binary-encoded value. 0=Input and 1=Output. The upper 8-bits of this value are inhibits.

- Data type: UINT16 (type index = 0)
- Readable and writable
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 0.9402
- T4:
  - Minimum firmware version: 0.2000

MIO_DIRECTION
- Address: 2603

Read or write the direction of the 3 bits of MIO in a single binary-encoded value. 0=Input and 1=Output. The upper 8-bits of this value are inhibits.

- Data type: UINT16 (type index = 0)
- Readable and writable
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 0.9402
- T4:
  - Minimum firmware version: 0.2000

DIO_STATE
- Address: 2800

Read or write the state of all digital I/O in a single binary-encoded value. 0=Low AND 1=High. Does not configure direction. A read of an output returns the current logic level on the terminal, not necessarily the output state written. Writes are filtered by the value in DIO_INHIBIT.

- Data type: UINT32 (type index = 1)
- Readable and writable

Constant	Value
Low	0
High	1

DIO_DIRECTION
- Address: 2850

Read or write the direction of all digital I/O in a single binary-encoded value. 0=Input and 1=Output. Writes are filtered by the value in DIO_INHIBIT.

- Data type: UINT32 (type index = 1)
- Readable and writable

Constant	Value
Input	0
Output	1

DIO_PULLDOWN_ENABLE
- Address: 2870

T8 Only. This register will enable pulldowns on DIO lines. This is a binary coded value where bit 0 represent FIO0, and bit 11 represents EIO3, etc. 1 = pulldown enabled, 0 = pulldown disabled. This register only affects flex-lines which can be configured as analog or digital. This register is not affected by the inhibit register.

- Data type: UINT32 (type index = 1)
- Readable and writable

DIO_PULLUP_DISABLE
- Address: 2890

This register will prevent pullups from being enabled on lines set to digital input. This is a binary coded value where bit 0 represent FIO0 and bit 11 represents EIO3. 1 = pullup disabled, 0 = pullup enabled. This register is not affected by the inhibit register.

- Data type: UINT32 (type index = 1)
- Readable and writable
- T8:
  - Disables the pull-up resistors on DIOs.
- T4:
  - This register only affects flex-lines which can be configured as analog or digital.

DIO_INHIBIT
- Address: 2900

A single binary-encoded value where each bit determines whether _STATE, _DIRECTION or _ANALOG_ENABLE writes affect that bit of digital I/O. 0=Default=Affected, 1=Ignored.

- Data type: UINT32 (type index = 1)
- Readable and writable
- Default value: 0

Constant	Value
Affected	0
Ignored	1

LED_COMM
- Address: 2990

Sets the state of the COMM LED when the LEDs are set to manual, see the POWER_LED register.

- Data type: UINT16 (type index = 0)
- Write-only
- Default value: 0
- T8:
  - Minimum firmware version: 1.0013
- T7:
  - Minimum firmware version: 1.0000
- T4:
  - Minimum firmware version: 1.0008

Constant	Value
Off	0
On	1

LED_STATUS
- Address: 2991

Sets the state of the STATUS LED when the LEDs are set to manual, see the POWER_LED register.

- Data type: UINT16 (type index = 0)
- Write-only
- Default value: 0
- T8:
  - Minimum firmware version: 1.0013
- T7:
  - Minimum firmware version: 1.0000
- T4:
  - Minimum firmware version: 1.0008

Constant	Value
Off	0
On	1

DIO_EF_CLOCK0_ENABLE
- Address: 44900

1 = enabled. 0 = disabled. Must be disabled during configuration.

- Data type: UINT16 (type index = 0)
- Readable and writable
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 0.8700

DIO_EF_CLOCK0_DIVISOR
- Address: 44901

Divides the core clock. Valid options: 1,2,4,8,16,32,64,256.

- Data type: UINT16 (type index = 0)
- Readable and writable
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 0.9320

DIO_EF_CLOCK0_OPTIONS
- Address: 44902

Bitmask: bit0: 1 = use external clock. All other bits reserved.

- Data type: UINT32 (type index = 1)
- Readable and writable
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 0.8700

DIO_EF_CLOCK0_ROLL_VALUE

- Address: 44904

The clock count will increment continuously and then start over at zero as it reaches the roll value. DIO_EF_CLOCK0 is a 32-bit clock, valid values are 0 to (2^32 - 1). 0 results in the max roll value (2^32).

- Data type: UINT32 (type index = 1)
- Readable and writable
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 0.8700

DIO_EF_CLOCK1_ENABLE
- Address: 44910

1 = enabled. 0 = disabled. Must be disabled during configuration.

- Data type: UINT16 (type index = 0)
- Readable and writable
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 0.8700

DIO_EF_CLOCK1_DIVISOR
- Address: 44911

Divides the core clock. Valid options: 1,2,4,8,16,32,64,256.

- Data type: UINT16 (type index = 0)
- Readable and writable
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 0.9320

DIO_EF_CLOCK1_OPTIONS
- Address: 44912

Bitmask: bit0: 1 = use external clock. All other bits reserved.

- Data type: UINT32 (type index = 1)
- Readable and writable
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 0.8700

DIO_EF_CLOCK1_ROLL_VALUE
- Address: 44914

The clock will count to this value and then start over at zero. The clock pulses counted are those after the divisor. 0 results in the max roll value possible. This is a 32-bit value (0-4294967295) if using a 32-bit clock, and a 16-bit value (0-65535) if using a 16-bit clock.

- Data type: UINT32 (type index = 1)
- Readable and writable
- T8:

    - Minimum firmware version: 0.0123
- T7:
    - Minimum firmware version: 0.8700

DIO_EF_CLOCK2_ENABLE
- Address: 44920

1 = enabled. 0 = disabled. Must be disabled during configuration.

- Data type: UINT16 (type index = 0)
- Readable and writable
- T8:
    - Minimum firmware version: 0.0123
- T7:
    - Minimum firmware version: 0.8700

DIO_EF_CLOCK2_DIVISOR
- Address: 44921

Divides the core clock. Valid options: 1,2,4,8,16,32,64,256.

- Data type: UINT16 (type index = 0)
- Readable and writable
- T8:
    - Minimum firmware version: 0.0123
- T7:
    - Minimum firmware version: 0.9320

DIO_EF_CLOCK2_OPTIONS
- Address: 44922

Bitmask: bit0: 1 = use external clock. All other bits reserved.

- Data type: UINT32 (type index = 1)
- Readable and writable
- T8:
    - Minimum firmware version: 0.0123
- T7:
    - Minimum firmware version: 0.8700

DIO_EF_CLOCK2_ROLL_VALUE
- Address: 44924

The clock will count to this value and then start over at zero. The clock pulses counted are those after the divisor. 0 results in the max roll value possible. This is a 32-bit value (0-4294967295) if using a 32-bit clock, and a 16-bit value (0-65535) if using a 16-bit clock.

- Data type: UINT32 (type index = 1)
- Readable and writable
- T8:
    - Minimum firmware version: 0.0123
- T7:
    - Minimum firmware version: 0.8700

DIO_EF_CLOCK0_COUNT
- Address: 44908

Current tick count of this clock. Will read between 0 and ROLL_VALUE-1.

- Data type: UINT32 (type index = 1)
- Read-only
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 0.9430

DIO_EF_CLOCK1_COUNT
- Address: 44918

Current tick count of this clock. Will read between 0 and ROLL_VALUE-1.

- Data type: UINT32 (type index = 1)
- Read-only
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 0.9430

DIO_EF_CLOCK2_COUNT
- Address: 44928

Current tick count of this clock. Will read between 0 and ROLL_VALUE-1.

- Data type: UINT32 (type index = 1)
- Read-only
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 0.9430

DIO#(0:21)_EF_ENABLE
- Starting Address: 44000

1 = enabled. 0 = disabled. Must be disabled during configuration. Note that DIO-EF reads work when disabled and do not return an error.

- Data type: UINT32 (type index = 1)
- Readable and writable
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 0.8800

Expanded Names	Addresses
DIO0_EF_ENABLE, DIO1_EF_ENABLE, DIO2_EF_ENABLE, DIO3_EF_ENABLE, DIO4_EF_ENABLE, DIO5_EF_ENABLE, DIO6_EF_ENABLE, DIO7_EF_ENABLE, DIO8_EF_ENABLE, DIO9_EF_ENABLE, DIO10_EF_ENABLE, DIO11_EF_ENABLE, DIO12_EF_ENABLE, DIO13_EF_ENABLE, DIO14_EF_ENABLE, DIO15_EF_ENABLE, DIO16_EF_ENABLE, DIO17_EF_ENABLE, DIO18_EF_ENABLE, DIO19_EF_ENABLE, DIO20_EF_ENABLE, DIO21_EF_ENABLE	44000, 44002, 44004, 44006, 44008, 44010, 44012, 44014, 44016, 44018, 44020, 44022, 44024, 44026, 44028, 44030, 44032, 44034, 44036, 44038, 44040, 44042

DIO#(0:21)_EF_INDEX

- Starting Address: 44100

An index to specify the feature you want.

- Data type: UINT32 (type index = 1)
- Readable and writable
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 0.8800

Expanded Names	Addresses
DIO0_EF_INDEX, DIO1_EF_INDEX, DIO2_EF_INDEX, DIO3_EF_INDEX, DIO4_EF_INDEX, DIO5_EF_INDEX, DIO6_EF_INDEX, DIO7_EF_INDEX, DIO8_EF_INDEX, DIO9_EF_INDEX, DIO10_EF_INDEX, DIO11_EF_INDEX, DIO12_EF_INDEX, DIO13_EF_INDEX, DIO14_EF_INDEX, DIO15_EF_INDEX, DIO16_EF_INDEX, DIO17_EF_INDEX, DIO18_EF_INDEX, DIO19_EF_INDEX, DIO20_EF_INDEX, DIO21_EF_INDEX	44100, 44102, 44104, 44106, 44108, 44110, 44112, 44114, 44116, 44118, 44120, 44122, 44124, 44126, 44128, 44130, 44132, 44134, 44136, 44138, 44140, 44142

DIO#(0:21)_EF_OPTIONS
- Starting Address: 44200

Function dependent on selected feature index.

- Data type: UINT32 (type index = 1)
- Readable and writable
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 0.8800

Expanded Names	Addresses
DIO0_EF_OPTIONS, DIO1_EF_OPTIONS, DIO2_EF_OPTIONS, DIO3_EF_OPTIONS, DIO4_EF_OPTIONS, DIO5_EF_OPTIONS, DIO6_EF_OPTIONS, DIO7_EF_OPTIONS, DIO8_EF_OPTIONS, DIO9_EF_OPTIONS, DIO10_EF_OPTIONS, DIO11_EF_OPTIONS, DIO12_EF_OPTIONS, DIO13_EF_OPTIONS, DIO14_EF_OPTIONS, DIO15_EF_OPTIONS, DIO16_EF_OPTIONS, DIO17_EF_OPTIONS, DIO18_EF_OPTIONS, DIO19_EF_OPTIONS, DIO20_EF_OPTIONS, DIO21_EF_OPTIONS	44200, 44202, 44204, 44206, 44208, 44210, 44212, 44214, 44216, 44218, 44220, 44222, 44224, 44226, 44228, 44230, 44232, 44234, 44236, 44238, 44240, 44242

DIO#(0:21)_EF_CONFIG_A
- Starting Address: 44300

Function dependent on selected feature index.

- Data type: UINT32 (type index = 1)
- Readable and writable
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 0.8800

Expanded Names	Addresses

DIO0_EF_CONFIG_A, DIO1_EF_CONFIG_A, DIO2_EF_CONFIG_A, DIO3_EF_CONFIG_A, DIO4_EF_CONFIG_A, DIO5_EF_CONFIG_A, DIO6_EF_CONFIG_A, DIO7_EF_CONFIG_A, DIO8_EF_CONFIG_A, DIO9_EF_CONFIG_A, DIO10_EF_CONFIG_A, DIO11_EF_CONFIG_A, DIO12_EF_CONFIG_A, DIO13_EF_CONFIG_A, DIO14_EF_CONFIG_A, DIO15_EF_CONFIG_A, DIO16_EF_CONFIG_A, DIO17_EF_CONFIG_A, DIO18_EF_CONFIG_A, DIO19_EF_CONFIG_A, DIO20_EF_CONFIG_A, DIO21_EF_CONFIG_A	44300, 44302, 44304, 44306, 44308, 44310, 44312, 44314, 44316, 44318, 44320, 44322, 44324, 44326, 44328, 44330, 44332, 44334, 44336, 44338, 44340, 44342

DIO#(0:21)_EF_CONFIG_B
- Starting Address: 44400

Function dependent on selected feature index.

- Data type: UINT32 (type index = 1)
- Readable and writable
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 0.8800

Expanded Names	Addresses
DIO0_EF_CONFIG_B, DIO1_EF_CONFIG_B, DIO2_EF_CONFIG_B, DIO3_EF_CONFIG_B, DIO4_EF_CONFIG_B, DIO5_EF_CONFIG_B, DIO6_EF_CONFIG_B, DIO7_EF_CONFIG_B, DIO8_EF_CONFIG_B, DIO9_EF_CONFIG_B, DIO10_EF_CONFIG_B, DIO11_EF_CONFIG_B, DIO12_EF_CONFIG_B, DIO13_EF_CONFIG_B, DIO14_EF_CONFIG_B, DIO15_EF_CONFIG_B, DIO16_EF_CONFIG_B, DIO17_EF_CONFIG_B, DIO18_EF_CONFIG_B, DIO19_EF_CONFIG_B, DIO20_EF_CONFIG_B, DIO21_EF_CONFIG_B	44400, 44402, 44404, 44406, 44408, 44410, 44412, 44414, 44416, 44418, 44420, 44422, 44424, 44426, 44428, 44430, 44432, 44434, 44436, 44438, 44440, 44442

DIO#(0:21)_EF_CONFIG_C
- Starting Address: 44500

Function dependent on selected feature index.

- Data type: UINT32 (type index = 1)
- Readable and writable
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 0.8800

Expanded Names	Addresses
DIO0_EF_CONFIG_C, DIO1_EF_CONFIG_C, DIO2_EF_CONFIG_C, DIO3_EF_CONFIG_C, DIO4_EF_CONFIG_C, DIO5_EF_CONFIG_C, DIO6_EF_CONFIG_C, DIO7_EF_CONFIG_C, DIO8_EF_CONFIG_C, DIO9_EF_CONFIG_C, DIO10_EF_CONFIG_C, DIO11_EF_CONFIG_C, DIO12_EF_CONFIG_C, DIO13_EF_CONFIG_C, DIO14_EF_CONFIG_C, DIO15_EF_CONFIG_C, DIO16_EF_CONFIG_C, DIO17_EF_CONFIG_C, DIO18_EF_CONFIG_C, DIO19_EF_CONFIG_C, DIO20_EF_CONFIG_C, DIO21_EF_CONFIG_C	44500, 44502, 44504, 44506, 44508, 44510, 44512, 44514, 44516, 44518, 44520, 44522, 44524, 44526, 44528, 44530, 44532, 44534, 44536, 44538, 44540, 44542

DIO#(0:21)_EF_CONFIG_D
- Starting Address: 44600

Function dependent on selected feature index.

- Data type: UINT32 (type index = 1)

- Readable and writable
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 0.8800

Expanded Names	Addresses
DIO0_EF_CONFIG_D, DIO1_EF_CONFIG_D, DIO2_EF_CONFIG_D, DIO3_EF_CONFIG_D, DIO4_EF_CONFIG_D, DIO5_EF_CONFIG_D, DIO6_EF_CONFIG_D, DIO7_EF_CONFIG_D, DIO8_EF_CONFIG_D, DIO9_EF_CONFIG_D, DIO10_EF_CONFIG_D, DIO11_EF_CONFIG_D, DIO12_EF_CONFIG_D, DIO13_EF_CONFIG_D, DIO14_EF_CONFIG_D, DIO15_EF_CONFIG_D, DIO16_EF_CONFIG_D, DIO17_EF_CONFIG_D, DIO18_EF_CONFIG_D, DIO19_EF_CONFIG_D, DIO20_EF_CONFIG_D, DIO21_EF_CONFIG_D	44600, 44602, 44604, 44606, 44608, 44610, 44612, 44614, 44616, 44618, 44620, 44622, 44624, 44626, 44628, 44630, 44632, 44634, 44636, 44638, 44640, 44642

DIO#(0:21)_EF_READ_A
- Starting Address: 3000

Reads an unsigned integer value. The meaning of the integer is dependent on selected feature index.

- Data type: UINT32 (type index = 1)
- Read-only
- This register may be streamed

Expanded Names	Addresses
DIO0_EF_READ_A, DIO1_EF_READ_A, DIO2_EF_READ_A, DIO3_EF_READ_A, DIO4_EF_READ_A, DIO5_EF_READ_A, DIO6_EF_READ_A, DIO7_EF_READ_A, DIO8_EF_READ_A, DIO9_EF_READ_A, DIO10_EF_READ_A, DIO11_EF_READ_A, DIO12_EF_READ_A, DIO13_EF_READ_A, DIO14_EF_READ_A, DIO15_EF_READ_A, DIO16_EF_READ_A, DIO17_EF_READ_A, DIO18_EF_READ_A, DIO19_EF_READ_A, DIO20_EF_READ_A, DIO21_EF_READ_A	3000, 3002, 3004, 3006, 3008, 3010, 3012, 3014, 3016, 3018, 3020, 3022, 3024, 3026, 3028, 3030, 3032, 3034, 3036, 3038, 3040, 3042

DIO#(0:21)_EF_READ_A_AND_RESET
- Starting Address: 3100

Reads the same value as DIO#(0:22)_EF_READ_A and forces a reset.

- Data type: UINT32 (type index = 1)
- Read-only
- This register may be streamed

Expanded Names	Addresses
DIO0_EF_READ_A_AND_RESET, DIO1_EF_READ_A_AND_RESET, DIO2_EF_READ_A_AND_RESET, DIO3_EF_READ_A_AND_RESET, DIO4_EF_READ_A_AND_RESET, DIO5_EF_READ_A_AND_RESET, DIO6_EF_READ_A_AND_RESET, DIO7_EF_READ_A_AND_RESET, DIO8_EF_READ_A_AND_RESET, DIO9_EF_READ_A_AND_RESET, DIO10_EF_READ_A_AND_RESET, DIO11_EF_READ_A_AND_RESET, DIO12_EF_READ_A_AND_RESET, DIO13_EF_READ_A_AND_RESET, DIO14_EF_READ_A_AND_RESET, DIO15_EF_READ_A_AND_RESET, DIO16_EF_READ_A_AND_RESET, DIO17_EF_READ_A_AND_RESET, DIO18_EF_READ_A_AND_RESET, DIO19_EF_READ_A_AND_RESET, DIO20_EF_READ_A_AND_RESET, DIO21_EF_READ_A_AND_RESET	3100, 3102, 3104, 3106, 3108, 3110, 3112, 3114, 3116, 3118, 3120, 3122, 3124, 3126, 3128, 3130, 3132, 3134, 3136, 3138, 3140, 3142

DIO#(0:21)_EF_READ_B
- Starting Address: 3200

Reads an unsigned integer value. The meaning of the integer is dependent on selected feature index.

- Data type: UINT32 (type index = 1)
- Read-only
- This register may be streamed

Expanded Names	Addresses
DIO0_EF_READ_B, DIO1_EF_READ_B, DIO2_EF_READ_B, DIO3_EF_READ_B, DIO4_EF_READ_B, DIO5_EF_READ_B, DIO6_EF_READ_B, DIO7_EF_READ_B, DIO8_EF_READ_B, DIO9_EF_READ_B, DIO10_EF_READ_B, DIO11_EF_READ_B, DIO12_EF_READ_B, DIO13_EF_READ_B, DIO14_EF_READ_B, DIO15_EF_READ_B, DIO16_EF_READ_B, DIO17_EF_READ_B, DIO18_EF_READ_B, DIO19_EF_READ_B, DIO20_EF_READ_B, DIO21_EF_READ_B	3200, 3202, 3204, 3206, 3208, 3210, 3212, 3214, 3216, 3218, 3220, 3222, 3224, 3226, 3228, 3230, 3232, 3234, 3236, 3238, 3240, 3242

DIO#(0:21)_EF_READ_A_F
- Starting Address: 3500

Reads a floating point value. The meaning of value is dependent on selected feature index.

- Data type: FLOAT32 (type index = 3)
- Read-only

Expanded Names	Addresses
DIO0_EF_READ_A_F, DIO1_EF_READ_A_F, DIO2_EF_READ_A_F, DIO3_EF_READ_A_F, DIO4_EF_READ_A_F, DIO5_EF_READ_A_F, DIO6_EF_READ_A_F, DIO7_EF_READ_A_F, DIO8_EF_READ_A_F, DIO9_EF_READ_A_F, DIO10_EF_READ_A_F, DIO11_EF_READ_A_F, DIO12_EF_READ_A_F, DIO13_EF_READ_A_F, DIO14_EF_READ_A_F, DIO15_EF_READ_A_F, DIO16_EF_READ_A_F, DIO17_EF_READ_A_F, DIO18_EF_READ_A_F, DIO19_EF_READ_A_F, DIO20_EF_READ_A_F, DIO21_EF_READ_A_F	3500, 3502, 3504, 3506, 3508, 3510, 3512, 3514, 3516, 3518, 3520, 3522, 3524, 3526, 3528, 3530, 3532, 3534, 3536, 3538, 3540, 3542

DIO#(0:21)_EF_READ_A_F_AND_RESET
- Starting Address: 3600

Reads a floating point value and forces a reset. The meaning of value is dependent on selected feature index.

- Data type: FLOAT32 (type index = 3)
- Read-only

Expanded Names	Addresses
DIO0_EF_READ_A_F_AND_RESET, DIO1_EF_READ_A_F_AND_RESET, DIO2_EF_READ_A_F_AND_RESET, DIO3_EF_READ_A_F_AND_RESET, DIO4_EF_READ_A_F_AND_RESET, DIO5_EF_READ_A_F_AND_RESET, DIO6_EF_READ_A_F_AND_RESET, DIO7_EF_READ_A_F_AND_RESET, DIO8_EF_READ_A_F_AND_RESET, DIO9_EF_READ_A_F_AND_RESET, DIO10_EF_READ_A_F_AND_RESET, DIO11_EF_READ_A_F_AND_RESET, DIO12_EF_READ_A_F_AND_RESET, DIO13_EF_READ_A_F_AND_RESET, DIO14_EF_READ_A_F_AND_RESET, DIO15_EF_READ_A_F_AND_RESET, DIO16_EF_READ_A_F_AND_RESET, DIO17_EF_READ_A_F_AND_RESET, DIO18_EF_READ_A_F_AND_RESET, DIO19_EF_READ_A_F_AND_RESET, DIO20_EF_READ_A_F_AND_RESET, DIO21_EF_READ_A_F_AND_RESET	3600, 3602, 3604, 3606, 3608, 3610, 3612, 3614, 3616, 3618, 3620, 3622, 3624, 3626, 3628, 3630, 3632, 3634, 3636, 3638, 3640, 3642

DIO#(0:21)_EF_READ_B_F
- Starting Address: 3700

Reads a floating point value. The meaning of value is dependent on selected feature index.

- Data type: FLOAT32 (type index = 3)
- Read-only

Expanded Names	Addresses
DIO0_EF_READ_B_F, DIO1_EF_READ_B_F, DIO2_EF_READ_B_F, DIO3_EF_READ_B_F, DIO4_EF_READ_B_F, DIO5_EF_READ_B_F, DIO6_EF_READ_B_F, DIO7_EF_READ_B_F, DIO8_EF_READ_B_F, DIO9_EF_READ_B_F, DIO10_EF_READ_B_F, DIO11_EF_READ_B_F, DIO12_EF_READ_B_F, DIO13_EF_READ_B_F, DIO14_EF_READ_B_F, DIO15_EF_READ_B_F, DIO16_EF_READ_B_F, DIO17_EF_READ_B_F, DIO18_EF_READ_B_F, DIO19_EF_READ_B_F, DIO20_EF_READ_B_F, DIO21_EF_READ_B_F	3700, 3702, 3704, 3706, 3708, 3710, 3712, 3714, 3716, 3718, 3720, 3722, 3724, 3726, 3728, 3730, 3732, 3734, 3736, 3738, 3740, 3742

DIO#(0:21)_EF_EASY_FREQUENCY_IN
- Starting Address: 45000

Deprecated. Automatically configures the associated DIO_EF for feature index 11. If already configured for index 11, reads the result and resets the DIO_EF.

- Data type: FLOAT32 (type index = 3)
- Readable and writable
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0105

Expanded Names	Addresses
DIO0_EF_EASY_FREQUENCY_IN, DIO1_EF_EASY_FREQUENCY_IN, DIO2_EF_EASY_FREQUENCY_IN, DIO3_EF_EASY_FREQUENCY_IN, DIO4_EF_EASY_FREQUENCY_IN, DIO5_EF_EASY_FREQUENCY_IN, DIO6_EF_EASY_FREQUENCY_IN, DIO7_EF_EASY_FREQUENCY_IN, DIO8_EF_EASY_FREQUENCY_IN, DIO9_EF_EASY_FREQUENCY_IN, DIO10_EF_EASY_FREQUENCY_IN, DIO11_EF_EASY_FREQUENCY_IN, DIO12_EF_EASY_FREQUENCY_IN, DIO13_EF_EASY_FREQUENCY_IN, DIO14_EF_EASY_FREQUENCY_IN, DIO15_EF_EASY_FREQUENCY_IN, DIO16_EF_EASY_FREQUENCY_IN, DIO17_EF_EASY_FREQUENCY_IN, DIO18_EF_EASY_FREQUENCY_IN, DIO19_EF_EASY_FREQUENCY_IN, DIO20_EF_EASY_FREQUENCY_IN, DIO21_EF_EASY_FREQUENCY_IN	45000, 45002, 45004, 45006, 45008, 45010, 45012, 45014, 45016, 45018, 45020, 45022, 45024, 45026, 45028, 45030, 45032, 45034, 45036, 45038, 45040, 45042

All STREAM TAGS:

Name	Start Address	Type	Access
STREAM_SCANRATE_HZ	4002	FLOAT32	R/W
STREAM_NUM_ADDRESSES	4004	UINT32	R/W
STREAM_SAMPLES_PER_PACKET	4006	UINT32	R/W
STREAM_SETTLING_US	4008	FLOAT32	R/W
STREAM_RESOLUTION_INDEX	4010	UINT32	R/W
STREAM_BUFFER_SIZE_BYTES	4012	UINT32	R/W
STREAM_CLOCK_SOURCE	4014	UINT32	R/W
STREAM_AUTO_TARGET	4016	UINT32	R/W
STREAM_NUM_SCANS	4020	UINT32	R/W
STREAM_EXTERNAL_CLOCK_DIVISOR	4022	UINT32	R/W
STREAM_TRIGGER_INDEX	4024	UINT32	R/W
STREAM_START_TIME_STAMP	4026	UINT32	R
STREAM_AUTORECOVER_DISABLE	4028	UINT32	R/W
STREAM_SCANLIST_ADDRESS#(0:126)	4100	UINT32	R/W
STREAM_OUT#(0:2)	4800	UINT16	R
STREAM_OUT#(0:2)_TARGET	4040	UINT32	R/W
STREAM_OUT#(0:2)_BUFFER_ALLOCATE_NUM_BYTES	4050	UINT32	R/W

STREAM_OUT#(0:2)_LOOP_NUM_VALUES	4060	UINT32	R/W
STREAM_OUT#(0:2)_SET_LOOP	4070	UINT32	W
STREAM_OUT#(0:2)_BUFFER_STATUS	4080	UINT32	R
STREAM_OUT#(0:2)_ENABLE	4090	UINT32	R/W
STREAM_OUT#(0:2)_BUFFER_F32	4400	FLOAT32	W
STREAM_OUT#(0:2)_BUFFER_U32	4410	UINT32	W
STREAM_OUT#(0:2)_BUFFER_U16	4420	UINT16	W
STREAM_DATA_CR	4500	UINT32	R
STREAM_DATA_CAPTURE_16	4899	UINT16	R
STREAM_DEBUG_GET_SELF_INDEX	4898	UINT32	R/W
STREAM_ENABLE	4990	UINT32	R/W

STREAM_SCANRATE_HZ
- Address: 4002

Write a value to specify the number of times per second that all channels in the stream scanlist will be read. Max stream speeds are based on Sample Rate which is NumChannels*ScanRate. Has no effect when using an external clock. A read of this register returns the actual scan rate, which can be slightly different due to rounding. For scan rates >152.588, the actual scan interval is multiples of 100 ns. Assuming a core clock of 80 MHz the internal roll value is (80M/(8*DesiredScanRate))-1 and the actual scan rate is then 80M/(8*(RollValue+1). For slower scan rates the scan interval resolution is changed to 1 us, 10 us, 100 us, or 1 ms as needed to achieve the longer intervals.

- Data type: FLOAT32 (type index = 3)
- Readable and writable
- Default value: 0

STREAM_NUM_ADDRESSES
- Address: 4004

The number of entries in the scanlist

- Data type: UINT32 (type index = 1)
- Readable and writable
- Default value: 0

STREAM_SAMPLES_PER_PACKET
- Address: 4006

Specifies the number of data points to be sent in the data packet. Only applies to spontaneous mode.

- Data type: UINT32 (type index = 1)
- Readable and writable
- Default value: 0

STREAM_SETTLING_US
- Address: 4008

Time in microseconds to allow signals to settle after switching the mux. Does not apply to the 1st channel in the scan list, as that settling is controlled by scan rate (the time from the last channel until the start of the next scan). Default=0. When set to less than 1, automatic settling will be used. The automatic settling behavior varies by device.

- Data type: FLOAT32 (type index = 3)
- Readable and writable
- Default value: 0
- T8:
    - Ignored. The T8's analog input chain does not rely on settling time.
- T7:

        - The T7 will select the settling time based on resolution and gain settings. When the sample rate is above 60 kHz, settling time will be set to ~5 us. Max is 4400.
    - T4:
        - The T4 will default to 10 us. When the scan rate is above 20 kHz, settling time will be set to 5 us.

STREAM_RESOLUTION_INDEX
- Address: 4010

The resolution index for stream readings. A larger resolution index generally results in lower noise and longer sample times.

- Data type: UINT32 (type index = 1)
- Readable and writable
- Default value: 0
- T8:
    - T8 description 0-16. A value of 0 will use the best resolution for the specified data rate.
- T7:
    - Valid values: 0-8. Default value of 0 corresponds to an index of 1.
- T4:
    - Valid values: 0-5. Default value of 0 corresponds to an index of 1.

STREAM_BUFFER_SIZE_BYTES
- Address: 4012

Size of the stream data buffer in bytes. A value of 0 equates to the default value. Must be a power of 2. Size in samples is STREAM_BUFFER_SIZE_BYTES/2. Size in scans is (STREAM_BUFFER_SIZE_BYTES/2)/STREAM_NUM_ADDRESSES. Changes while stream is running do not affect the currently running stream.

- Data type: UINT32 (type index = 1)
- Readable and writable
- Default value: 0
- This register uses system RAM. The maximum RAM is 64KB. For more information, see 4.4 RAM
- T8:
    - Max size is 262144T8. Default size is 4096.
- T7:
    - Max size is 32768. Default size is 4096.
- T4:
    - Max size is 32768. Default size is 8192.

STREAM_CLOCK_SOURCE
- Address: 4014

Controls which clock source will be used to run the main stream clock. 0 = Internal crystal, 2 = External clock source on CIO3. Rising edges will increment a counter and trigger a stream scan after the number of edges specified in STREAM_EXTERNAL_CLOCK_DIVISOR. T7 will expect external stream clock on CIO3. All other values reserved.

- Data type: UINT32 (type index = 1)
- Readable and writable
- Default value: 0
- T8:
    - Minimum firmware version: 0.0123
- T7:
    - Minimum firmware version: 1.0110

STREAM_AUTO_TARGET
- Address: 4016

Controls where data will be sent. Value is a bitmask. bit 0: 1 = Send to Ethernet 702 sockets, bit 1: 1 = Send to USB, bit 4: 1 = Command-Response mode. All other bits are reserved.

- Data type: UINT32 (type index = 1)
- Readable and writable
- Default value: 0
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 0.9108

STREAM_NUM_SCANS
- Address: 4020

The number of scans to run before automatically stopping (stream-burst). 0 = run continuously. Limit for STREAM_NUM_SCANS is 2^32-1, but if the host is not reading data as fast as it is acquired you also need to consider STREAM_BUFFER_SIZE_BYTES.

- Data type: UINT32 (type index = 1)
- Readable and writable
- Default value: 0

STREAM_EXTERNAL_CLOCK_DIVISOR
- Address: 4022

The number of pulses per stream scan when using an external clock.

- Data type: UINT32 (type index = 1)
- Readable and writable
- Default value: 0

STREAM_TRIGGER_INDEX
- Address: 4024

Controls when stream scanning will start. 0 = Start when stream is enabled, 2000 = Start when DIO_EF0 detects an edge, 2001 = Start when DIO_EF1 detects an edge. See the stream documentation for all supported values.

- Data type: UINT32 (type index = 1)
- Readable and writable
- Default value: 0
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0186

STREAM_START_TIME_STAMP
- Address: 4026

This register stores the value of CORE_TIMER at the start of the first stream scan. Note that the first stream scan happens 1 scan period after STREAM_ENABLE is set to 1.

- Data type: UINT32 (type index = 1)
- Read-only
- Default value: 0
- T8:

    - Minimum firmware version: 0.0123
- T7:
    - Minimum firmware version: 1.0214

STREAM_AUTORECOVER_DISABLE
- Address: 4028

When the stream buffer becomes full the stream will be stopped. Same behavior as if STREAM_ENABLE is set to zero.

- Data type: UINT32 (type index = 1)
- Readable and writable
- Default value: 0
- T8:
    - Minimum firmware version: 0.0123
- T7:
    - Minimum firmware version: 1.0260

STREAM_SCANLIST_ADDRESS#(0:126)
- Starting Address: 4100

A list of addresses to read each scan. In the case of Stream-Out enabled, the list may also include something to write each scan.

- Data type: UINT32 (type index = 1)
- Readable and writable
- Default value: 0

Expanded Names	Addresses
STREAM_SCANLIST_ADDRESS0, STREAM_SCANLIST_ADDRESS1,	4100, 4102,
STREAM_SCANLIST_ADDRESS2, STREAM_SCANLIST_ADDRESS3,	4104, 4106,
STREAM_SCANLIST_ADDRESS4, STREAM_SCANLIST_ADDRESS5,	4108, 4110,
STREAM_SCANLIST_ADDRESS6, STREAM_SCANLIST_ADDRESS7,	4112, 4114,
STREAM_SCANLIST_ADDRESS8, STREAM_SCANLIST_ADDRESS9,	4116, 4118,
STREAM_SCANLIST_ADDRESS10, STREAM_SCANLIST_ADDRESS11,	4120, 4122,
STREAM_SCANLIST_ADDRESS12, STREAM_SCANLIST_ADDRESS13,	4124, 4126,
STREAM_SCANLIST_ADDRESS14, STREAM_SCANLIST_ADDRESS15,	4128, 4130,
STREAM_SCANLIST_ADDRESS16, STREAM_SCANLIST_ADDRESS17,	4132, 4134,
STREAM_SCANLIST_ADDRESS18, STREAM_SCANLIST_ADDRESS19,	4136, 4138,
STREAM_SCANLIST_ADDRESS20, STREAM_SCANLIST_ADDRESS21,	4140, 4142,
STREAM_SCANLIST_ADDRESS22, STREAM_SCANLIST_ADDRESS23,	4144, 4146,
STREAM_SCANLIST_ADDRESS24, STREAM_SCANLIST_ADDRESS25,	4148, 4150,
STREAM_SCANLIST_ADDRESS26, STREAM_SCANLIST_ADDRESS27,	4152, 4154,
STREAM_SCANLIST_ADDRESS28, STREAM_SCANLIST_ADDRESS29,	4156, 4158,
STREAM_SCANLIST_ADDRESS30, STREAM_SCANLIST_ADDRESS31,	4160, 4162,
STREAM_SCANLIST_ADDRESS32, STREAM_SCANLIST_ADDRESS33,	4164, 4166,
STREAM_SCANLIST_ADDRESS34, STREAM_SCANLIST_ADDRESS35,	4168, 4170,
STREAM_SCANLIST_ADDRESS36, STREAM_SCANLIST_ADDRESS37,	4172, 4174,
STREAM_SCANLIST_ADDRESS38, STREAM_SCANLIST_ADDRESS39,	4176, 4178,
STREAM_SCANLIST_ADDRESS40, STREAM_SCANLIST_ADDRESS41,	4180, 4182,
STREAM_SCANLIST_ADDRESS42, STREAM_SCANLIST_ADDRESS43,	4184, 4186,
STREAM_SCANLIST_ADDRESS44, STREAM_SCANLIST_ADDRESS45,	4188, 4190,
STREAM_SCANLIST_ADDRESS46, STREAM_SCANLIST_ADDRESS47,	4192, 4194,
STREAM_SCANLIST_ADDRESS48, STREAM_SCANLIST_ADDRESS49,	4196, 4198,
STREAM_SCANLIST_ADDRESS50, STREAM_SCANLIST_ADDRESS51,	4200, 4202,
STREAM_SCANLIST_ADDRESS52, STREAM_SCANLIST_ADDRESS53,	4204, 4206,
STREAM_SCANLIST_ADDRESS54, STREAM_SCANLIST_ADDRESS55,	4208, 4210,
STREAM_SCANLIST_ADDRESS56, STREAM_SCANLIST_ADDRESS57,	4212, 4214,
STREAM_SCANLIST_ADDRESS58, STREAM_SCANLIST_ADDRESS59,	4216, 4218,

STREAM_SCANLIST_ADDRESS60, STREAM_SCANLIST_ADDRESS61,	4220, 4222,
STREAM_SCANLIST_ADDRESS62, STREAM_SCANLIST_ADDRESS63,	4224, 4226,
STREAM_SCANLIST_ADDRESS64, STREAM_SCANLIST_ADDRESS65,	4228, 4230,
STREAM_SCANLIST_ADDRESS66, STREAM_SCANLIST_ADDRESS67,	4232, 4234,
STREAM_SCANLIST_ADDRESS68, STREAM_SCANLIST_ADDRESS69,	4236, 4238,
STREAM_SCANLIST_ADDRESS70, STREAM_SCANLIST_ADDRESS71,	4240, 4242,
STREAM_SCANLIST_ADDRESS72, STREAM_SCANLIST_ADDRESS73,	4244, 4246,
STREAM_SCANLIST_ADDRESS74, STREAM_SCANLIST_ADDRESS75,	4248, 4250,
STREAM_SCANLIST_ADDRESS76, STREAM_SCANLIST_ADDRESS77,	4252, 4254,
STREAM_SCANLIST_ADDRESS78, STREAM_SCANLIST_ADDRESS79,	4256, 4258,
STREAM_SCANLIST_ADDRESS80, STREAM_SCANLIST_ADDRESS81,	4260, 4262,
STREAM_SCANLIST_ADDRESS82, STREAM_SCANLIST_ADDRESS83,	4264, 4266,
STREAM_SCANLIST_ADDRESS84, STREAM_SCANLIST_ADDRESS85,	4268, 4270,
STREAM_SCANLIST_ADDRESS86, STREAM_SCANLIST_ADDRESS87,	4272, 4274,
STREAM_SCANLIST_ADDRESS88, STREAM_SCANLIST_ADDRESS89,	4276, 4278,
STREAM_SCANLIST_ADDRESS90, STREAM_SCANLIST_ADDRESS91,	4280, 4282,
STREAM_SCANLIST_ADDRESS92, STREAM_SCANLIST_ADDRESS93,	4284, 4286,
STREAM_SCANLIST_ADDRESS94, STREAM_SCANLIST_ADDRESS95,	4288, 4290,
STREAM_SCANLIST_ADDRESS96, STREAM_SCANLIST_ADDRESS97,	4292, 4294,
STREAM_SCANLIST_ADDRESS98, STREAM_SCANLIST_ADDRESS99,	4296, 4298,
STREAM_SCANLIST_ADDRESS100, STREAM_SCANLIST_ADDRESS101,	4300, 4302,
STREAM_SCANLIST_ADDRESS102, STREAM_SCANLIST_ADDRESS103,	4304, 4306,
STREAM_SCANLIST_ADDRESS104, STREAM_SCANLIST_ADDRESS105,	4308, 4310,
STREAM_SCANLIST_ADDRESS106, STREAM_SCANLIST_ADDRESS107,	4312, 4314,
STREAM_SCANLIST_ADDRESS108, STREAM_SCANLIST_ADDRESS109,	4316, 4318,
STREAM_SCANLIST_ADDRESS110, STREAM_SCANLIST_ADDRESS111,	4320, 4322,
STREAM_SCANLIST_ADDRESS112, STREAM_SCANLIST_ADDRESS113,	4324, 4326,
STREAM_SCANLIST_ADDRESS114, STREAM_SCANLIST_ADDRESS115,	4328, 4330,
STREAM_SCANLIST_ADDRESS116, STREAM_SCANLIST_ADDRESS117,	4332, 4334,
STREAM_SCANLIST_ADDRESS118, STREAM_SCANLIST_ADDRESS119,	4336, 4338,
STREAM_SCANLIST_ADDRESS120, STREAM_SCANLIST_ADDRESS121,	4340, 4342,
STREAM_SCANLIST_ADDRESS122, STREAM_SCANLIST_ADDRESS123,	4344, 4346,
STREAM_SCANLIST_ADDRESS124, STREAM_SCANLIST_ADDRESS125,	4348, 4350,
STREAM_SCANLIST_ADDRESS126	4352

STREAM_OUT#(0:2)
- Starting Address: 4800

Include one or more of these registers in STREAM_SCANLIST_ADDRESS#(0:127) to trigger stream-out updates. When added to the scan list these do count against the max scan rate just like normal input addresses, but they do not return any data in the stream read.

- Data type: UINT16 (type index = 0)
- Read-only
- Default value: 0
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0012

Expanded Names	Addresses
STREAM_OUT0, STREAM_OUT1, STREAM_OUT2	4800, 4801, 4802

STREAM_OUT#(0:2)_TARGET
- Starting Address: 4040

Channel that data will be written to. Before writing data to _BUFFER_###, you must write to _TARGET so the device knows how to interpret and store values.

- Data type: UINT32 (type index = 1)
- Readable and writable

- Default value: 0
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0012

Expanded Names	Addresses
STREAM_OUT0_TARGET, STREAM_OUT1_TARGET, STREAM_OUT2_TARGET	4040, 4042, 4044

STREAM_OUT#(0:2)_BUFFER_ALLOCATE_NUM_BYTES
- Starting Address: 4050

Size of the buffer in bytes as a power of 2. Should be at least twice the size of updates that will be written and no less than 32. The usable buffer size will be equal to the value of this register minus 2 bytes (one 16-bit sample). Before writing data to _BUFFER_###, you must write to _BUFFER_ALLOCATE_NUM_BYTES to allocate RAM for the data. Max is 16384.

- Data type: UINT32 (type index = 1)
- Readable and writable
- Default value: 0
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 0.9300

Expanded Names	Addresses
STREAM_OUT0_BUFFER_ALLOCATE_NUM_BYTES, STREAM_OUT1_BUFFER_ALLOCATE_NUM_BYTES, STREAM_OUT2_BUFFER_ALLOCATE_NUM_BYTES	4050, 4052, 4054

STREAM_OUT#(0:2)_LOOP_NUM_VALUES
- Starting Address: 4060

The number of values, from the end of the array, that will be repeated after reaching the end of supplied data array.

- Data type: UINT32 (type index = 1)
- Readable and writable
- Default value: 0
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 0.9300

Expanded Names	Addresses
STREAM_OUT0_LOOP_NUM_VALUES, STREAM_OUT1_LOOP_NUM_VALUES, STREAM_OUT2_LOOP_NUM_VALUES	4060, 4062, 4064

STREAM_OUT#(0:2)_SET_LOOP
- Starting Address: 4070

Controls when new data and loop size are used. 1=Use new data immediately. 2=Wait for synch. New data will not be used until a different stream-out channel is set to Synch. 3=Synch. This stream-out# as well as any stream-outs set to synch will start using new data immediately.

- Data type: UINT32 (type index = 1)
- Write-only

- Default value: 0
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 0.9300

Expanded Names	Addresses
STREAM_OUT0_SET_LOOP, STREAM_OUT1_SET_LOOP, STREAM_OUT2_SET_LOOP	4070, 4072, 4074

STREAM_OUT#(0:2)_BUFFER_STATUS
- Starting Address: 4080

The number of values in the buffer that are not currently being used.

- Data type: UINT32 (type index = 1)
- Read-only
- Default value: 0
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 0.9300

Expanded Names	Addresses
STREAM_OUT0_BUFFER_STATUS, STREAM_OUT1_BUFFER_STATUS, STREAM_OUT2_BUFFER_STATUS	4080, 4082, 4084

STREAM_OUT#(0:2)_ENABLE
- Starting Address: 4090

When STREAM_OUT#_ENABLE is enabled, the stream out target is generally updated by one value from the stream out buffer per stream scan. For example, there will generally be one instance of e.g. STREAM_OUT0 in the stream scan list, which will cause one STREAM_OUT0_BUFFER value to be consumed and written to STREAM_OUT0_TARGET for every stream scan. The stream scan list could also contain two instances of STREAM_OUT0, in which case two values from STREAM_OUT0_BUFFER value would be consumed and written for every stream scan.

- Data type: UINT32 (type index = 1)
- Readable and writable
- Default value: 0
- This register uses system RAM. The maximum RAM is 64KB. For more information, see 4.4 RAM
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 0.9300

Constant	Value
DISABLED	0
ENABLED	1

Expanded Names	Addresses
STREAM_OUT0_ENABLE, STREAM_OUT1_ENABLE, STREAM_OUT2_ENABLE	4090, 4092, 4094

STREAM_OUT#(0:2)_BUFFER_F32
- Starting Address: 4400

Data destination when sending floating point data. Appropriate cal constants are used to convert F32 values to 16-bit binary data, and thus each of these values uses 2 bytes of the stream-out buffer. This

register is a buffer.

- Data type: FLOAT32 (type index = 3)
- Write-only
- Default value: 0
- This register is a Buffer Register
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 0.9300

Expanded Names	Addresses
STREAM_OUT0_BUFFER_F32, STREAM_OUT1_BUFFER_F32, STREAM_OUT2_BUFFER_F32	4400, 4402, 4404

STREAM_OUT#(0:2)_BUFFER_U32
- Starting Address: 4410

Not used at this time. There are no U32 registers supported in stream-out. This register is a buffer.

- Data type: UINT32 (type index = 1)
- Write-only
- Default value: 0
- This register is a Buffer Register
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 0.9300

Expanded Names	Addresses
STREAM_OUT0_BUFFER_U32, STREAM_OUT1_BUFFER_U32, STREAM_OUT2_BUFFER_U32	4410, 4412, 4414

STREAM_OUT#(0:2)_BUFFER_U16
- Starting Address: 4420

Data destination when sending 16-bit integer data. Each value uses 2 bytes of the stream-out buffer. This register is a buffer.

- Data type: UINT16 (type index = 0)
- Write-only
- Default value: 0
- This register is a Buffer Register
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 0.9300

Expanded Names	Addresses
STREAM_OUT0_BUFFER_U16, STREAM_OUT1_BUFFER_U16, STREAM_OUT2_BUFFER_U16	4420, 4421, 4422

STREAM_DATA_CR
- Address: 4500

Address to read stream data from when operating in C-R mode.

- Data type: UINT32 (type index = 1)
- Read-only

STREAM_DATA_CAPTURE_16
- Address: 4899

If a channel produces 32-bits of data the upper 16 will be saved here.

- Data type: UINT16 (type index = 0)
- Read-only
- This register may be streamed
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0071

STREAM_DEBUG_GET_SELF_INDEX
- Address: 4898

Returns the index of the channel. The index is the position in the scan list that the channel occupies.

- Data type: UINT32 (type index = 1)
- Readable and writable
- This register may be streamed
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0254

STREAM_ENABLE
- Address: 4990

Write 1 to start stream. Write 0 to stop stream. Reading this register returns 1 when stream is enabled. When using a triggered stream the stream is considered enabled while waiting for the trigger.

- Data type: UINT32 (type index = 1)
- Readable and writable
- This register uses system RAM. The maximum RAM is 64KB. For more information, see 4.4 RAM
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 0.9421

All SPI TAGS:

Name	Start Address	Type	Access
SPI_CS_DIONUM	5000	UINT16	R/W
SPI_CLK_DIONUM	5001	UINT16	R/W
SPI_MISO_DIONUM	5002	UINT16	R/W
SPI_MOSI_DIONUM	5003	UINT16	R/W
SPI_MODE	5004	UINT16	R/W
SPI_SPEED_THROTTLE	5005	UINT16	R/W
SPI_OPTIONS	5006	UINT16	R/W
SPI_GO	5007	UINT16	W
SPI_NUM_BYTES	5009	UINT16	W

SPI_DATA_TX	5010	BYTE	W
SPI_DATA_RX	5050	BYTE	R

SPI_CS_DIONUM
- Address: 5000

The DIO line for Chip-Select.

- Data type: UINT16 (type index = 0)
- Readable and writable
- Default value: 0

SPI_CLK_DIONUM
- Address: 5001

The DIO line for Clock.

- Data type: UINT16 (type index = 0)
- Readable and writable
- Default value: 0

SPI_MISO_DIONUM
- Address: 5002

The DIO line for Master-In-Slave-Out.

- Data type: UINT16 (type index = 0)
- Readable and writable
- Default value: 0

SPI_MOSI_DIONUM
- Address: 5003

The DIO line for Master-Out-Slave-In.

- Data type: UINT16 (type index = 0)
- Readable and writable
- Default value: 0

SPI_MODE
- Address: 5004

The SPI mode controls the clock idle state and which edge clocks the data. Bit 1 is CPOL and Bit 0 is CPHA, so CPOL/CPHA for different decimal values: 0 = 0/0 = b00, 1 = 0/1 = b01, 2 = 1/0 = b10, 3 = 1/1 = b11. For CPOL and CPHA explanations, see Wikipedia article: https://en.wikipedia.org/wiki/Serial_Peripheral_Interface_Bus.

- Data type: UINT16 (type index = 0)
- Readable and writable
- Default value: 0

SPI_SPEED_THROTTLE
- Address: 5005

This value controls the SPI clock frequency. Pass 0-65535. Default=0 corresponds to 65536 internally which results in ~800 kHz. 65500 = ~100 kHz, 65100 = ~10 kHz, 61100 = ~1 kHz, 21000 = ~100 Hz, and 1 = ~67 Hz. Avoid setting too low such that the entire transaction lasts longer than the 250 millisecond timeout of the internal watchdog timer.

- Data type: UINT16 (type index = 0)
- Readable and writable

- Default value: 0

SPI_OPTIONS
- Address: 5006

Bit 0 is Auto-CS-Disable. When bit 0 is 0, CS is enabled. When bit 0 is 1, CS is disabled. Bit 1: 0 = Set DIO directions before starting the SPI operations, 1 = Do not set DIO directions. Bit 2: 0 = Transmit data MSB first, 1 = LSB first. Bits 4-7: This value sets the number of bits that will be transmitted during the last byte of the SPI operation. Default is 8, valid options are 1-8.

- Data type: UINT16 (type index = 0)
- Readable and writable
- Default value: 0

SPI_GO
- Address: 5007

Write 1 to begin the configured SPI transaction.

- Data type: UINT16 (type index = 0)
- Write-only
- Default value: 0
- This register uses system RAM. The maximum RAM is 64KB. For more information, see 4.4 RAM

SPI_NUM_BYTES
- Address: 5009

The number of bytes to transfer. The maximum transfer size is 100 bytes.

- Data type: UINT16 (type index = 0)
- Write-only
- Default value: 0

SPI_DATA_TX
- Address: 5010

Write data here. This register is a buffer.

- Data type: BYTE (type index = 99)
- Write-only
- Default value: 0
- This register is a Buffer Register

SPI_DATA_RX
- Address: 5050

Read data here. This register is a buffer. Underrun behavior - fill with zeros.

- Data type: BYTE (type index = 99)
- Read-only
- Default value: 0
- This register is a Buffer Register

All I2C TAGS:

Name	Start Address	Type	Access
I2C_SDA_DIONUM	5100	UINT16	R/W
I2C_SCL_DIONUM	5101	UINT16	R/W
I2C_SPEED_THROTTLE	5102	UINT16	R/W

I2C_OPTIONS	5103	UINT16	R/W
I2C_SLAVE_ADDRESS	5104	UINT16	R/W
I2C_NUM_BYTES_TX	5108	UINT16	R/W
I2C_NUM_BYTES_RX	5109	UINT16	R/W
I2C_GO	5110	UINT16	R/W
I2C_ACKS	5114	UINT32	R/W
I2C_DATA_TX	5120	BYTE	W
I2C_DATA_RX	5160	BYTE	R

I2C_SDA_DIONUM
- Address: 5100

The number of the DIO line to be used as the I2C data line. Ex: Writing 0 will force FIO0 to become the I2C-SDA line.

- Data type: UINT16 (type index = 0)
- Readable and writable
- Default value: 0

I2C_SCL_DIONUM
- Address: 5101

The number of the DIO line to be used as the I2C clock line. Ex: Writing 1 will force FIO1 to become the I2C-SCL line.

- Data type: UINT16 (type index = 0)
- Readable and writable
- Default value: 0

I2C_SPEED_THROTTLE
- Address: 5102

This value controls the I2C clock frequency. Pass 0-65535. Default=0 corresponds to 65536 internally which results in ~450 kHz. 1 results in ~40 Hz, 65516 is ~100 kHz.

- Data type: UINT16 (type index = 0)
- Readable and writable
- Default value: 0

I2C_OPTIONS
- Address: 5103

Advanced. Controls details of the I2C protocol to improve device compatibility. bit 0: 1 = Reset the I2C bus before attempting communication. bit 1: 0 = Restarts will use a stop and a start, 1 = Restarts will not use a stop. bit 2: 1 = disable clock stretching.

- Data type: UINT16 (type index = 0)
- Readable and writable
- Default value: 0

I2C_SLAVE_ADDRESS
- Address: 5104

The 7-bit address of the slave device. Value is shifted left by firmware to allow room for the I2C R/W bit.

- Data type: UINT16 (type index = 0)
- Readable and writable
- Default value: 0

I2C_NUM_BYTES_TX
- Address: 5108

The number of data bytes to transmit. Zero is valid and will result in a read-only I2C operation.

- Data type: UINT16 (type index = 0)
- Readable and writable
- Default value: 0

I2C_NUM_BYTES_RX
- Address: 5109

The number of data bytes to read. Zero is valid and will result in a write-only I2C operation.

- Data type: UINT16 (type index = 0)
- Readable and writable
- Default value: 0

I2C_GO
- Address: 5110

Writing to this register will instruct the LabJack to perform an I2C transaction.

- Data type: UINT16 (type index = 0)
- Readable and writable
- Default value: 0
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 0.9504

I2C_ACKS
- Address: 5114

A binary encoded value (array of bits) used to observe ACKs from the slave device.

- Data type: UINT32 (type index = 1)
- Readable and writable
- Default value: 0
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 0.9504

I2C_DATA_TX
- Address: 5120

Data that will be written to the I2C bus. This register is a buffer.

- Data type: BYTE (type index = 99)
- Write-only
- Default value: 0
- This register is a Buffer Register

I2C_DATA_RX
- Address: 5160

Data that has been read from the I2C bus. This register is a buffer. Underrun behavior - fill with zeros.

- Data type: BYTE (type index = 99)
- Read-only
- Default value: 0
- This register is a Buffer Register

All ONEWIRE TAGS:

Name	Start Address	Type	Access
ONEWIRE_DQ_DIONUM	5300	UINT16	R/W
ONEWIRE_DPU_DIONUM	5301	UINT16	R/W
ONEWIRE_OPTIONS	5302	UINT16	R/W
ONEWIRE_FUNCTION	5307	UINT16	R/W
ONEWIRE_NUM_BYTES_TX	5308	UINT16	R/W
ONEWIRE_NUM_BYTES_RX	5309	UINT16	R/W
ONEWIRE_GO	5310	UINT16	W
ONEWIRE_ROM_MATCH_H	5320	UINT32	R/W
ONEWIRE_ROM_MATCH_L	5322	UINT32	R/W
ONEWIRE_ROM_BRANCHS_FOUND_H	5332	UINT32	R
ONEWIRE_ROM_BRANCHS_FOUND_L	5334	UINT32	R
ONEWIRE_SEARCH_RESULT_H	5328	UINT32	R
ONEWIRE_SEARCH_RESULT_L	5330	UINT32	R
ONEWIRE_PATH_H	5324	UINT32	R/W
ONEWIRE_PATH_L	5326	UINT32	R/W
ONEWIRE_DATA_TX	5340	BYTE	W
ONEWIRE_DATA_RX	5370	BYTE	R

ONEWIRE_DQ_DIONUM
- Address: 5300

The data-line DIO number.

- Data type: UINT16 (type index = 0)
- Readable and writable
- Default value: 0
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0056

ONEWIRE_DPU_DIONUM
- Address: 5301

The dynamic pullup control DIO number.

- Data type: UINT16 (type index = 0)
- Readable and writable
- Default value: 0
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0056

ONEWIRE_OPTIONS
- Address: 5302

Controls advanced features. Value is a bitmask. bit 0: reserved, bit 1: reserved, bit 2: 1=DPU Enabled 0=DPU Disabled, bit 3: DPU Polarity 1=Active state is high, 0=Active state is low (Dynamic Pull-Up)

- Data type: UINT16 (type index = 0)
- Readable and writable
- Default value: 0
- T8:
    - Minimum firmware version: 0.0123
- T7:
    - Minimum firmware version: 1.0056

ONEWIRE_FUNCTION
- Address: 5307

Set the ROM function to use. 0xF0=Search, 0xCC=Skip, 0x55=Match, 0x33=Read.

- Data type: UINT16 (type index = 0)
- Readable and writable
- Default value: 0
- T8:
    - Minimum firmware version: 0.0123
- T7:
    - Minimum firmware version: 1.0056

ONEWIRE_NUM_BYTES_TX
- Address: 5308

Number of data bytes to be sent.

- Data type: UINT16 (type index = 0)
- Readable and writable
- Default value: 0
- T8:
    - Minimum firmware version: 0.0123
- T7:
    - Minimum firmware version: 1.0056

ONEWIRE_NUM_BYTES_RX
- Address: 5309

Number of data bytes to be received.

- Data type: UINT16 (type index = 0)
- Readable and writable
- Default value: 0
- T8:
    - Minimum firmware version: 0.0123
- T7:
    - Minimum firmware version: 1.0056

ONEWIRE_GO
- Address: 5310

Instructs the device to perform the configured 1-wire transaction.

- Data type: UINT16 (type index = 0)
- Write-only
- Default value: 0
- T8:
    - Minimum firmware version: 0.0123

- T7:
    - Minimum firmware version: 1.0056

ONEWIRE_ROM_MATCH_H
- Address: 5320

Upper 32-bits of the ROM to match.

- Data type: UINT32 (type index = 1)
- Readable and writable
- Default value: 0
- T8:
    - Minimum firmware version: 0.0123
- T7:
    - Minimum firmware version: 1.0056

ONEWIRE_ROM_MATCH_L
- Address: 5322

Lower 32-bits of the ROM to match.

- Data type: UINT32 (type index = 1)
- Readable and writable
- Default value: 0
- T8:
    - Minimum firmware version: 0.0123
- T7:
    - Minimum firmware version: 1.0056

ONEWIRE_ROM_BRANCHS_FOUND_H
- Address: 5332

Upper 32-bits of the branches detected during a search.

- Data type: UINT32 (type index = 1)
- Read-only
- Default value: 0
- T8:
    - Minimum firmware version: 0.0123
- T7:
    - Minimum firmware version: 1.0056

ONEWIRE_ROM_BRANCHS_FOUND_L
- Address: 5334

Lower 32-bits of the branches detected during a search.

- Data type: UINT32 (type index = 1)
- Read-only
- Default value: 0
- T8:
    - Minimum firmware version: 0.0123
- T7:
    - Minimum firmware version: 1.0056

ONEWIRE_SEARCH_RESULT_H
- Address: 5328

Upper 32-bits of the search result.

- Data type: UINT32 (type index = 1)
- Read-only
- Default value: 0
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0056

ONEWIRE_SEARCH_RESULT_L
- Address: 5330

Lower 32-bites of the search result.

- Data type: UINT32 (type index = 1)
- Read-only
- Default value: 0
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0056

ONEWIRE_PATH_H
- Address: 5324

Upper 32-bits of the path to take during a search.

- Data type: UINT32 (type index = 1)
- Readable and writable
- Default value: 0
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0056

ONEWIRE_PATH_L
- Address: 5326

Lower 32-bits of the path to take during a search.

- Data type: UINT32 (type index = 1)
- Readable and writable
- Default value: 0
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0056

ONEWIRE_DATA_TX
- Address: 5340

Data to be transmitted over the 1-wire bus. This register is a buffer.

- Data type: BYTE (type index = 99)
- Write-only
- Default value: 0

- This register is a Buffer Register
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0056

ONEWIRE_DATA_RX
- Address: 5370

Data received over the 1-wire bus. This register is a buffer. Underrun behavior - buffer is static, old data will fill the extra locations, firmware 1.0225 changes this to read zeros.

- Data type: BYTE (type index = 99)
- Read-only
- Default value: 0
- This register is a Buffer Register
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0056

All ASYNCH TAGS:

Name	Start Address	Type	Access
ASYNCH_ENABLE	5400	UINT16	R/W
ASYNCH_BAUD	5420	UINT32	R/W
ASYNCH_RX_DIONUM	5405	UINT16	R/W
ASYNCH_TX_DIONUM	5410	UINT16	R/W
ASYNCH_NUM_DATA_BITS	5415	UINT16	R/W
ASYNCH_RX_BUFFER_SIZE_BYTES	5430	UINT16	R/W
ASYNCH_NUM_BYTES_RX	5435	UINT16	R
ASYNCH_NUM_BYTES_TX	5440	UINT16	R/W
ASYNCH_TX_GO	5450	UINT16	W
ASYNCH_NUM_STOP_BITS	5455	UINT16	R/W
ASYNCH_PARITY	5460	UINT16	R/W
ASYNCH_NUM_PARITY_ERRORS	5465	UINT16	R/W
ASYNCH_DATA_TX	5490	UINT16	W
ASYNCH_DATA_RX	5495	UINT16	R

ASYNCH_ENABLE
- Address: 5400

1 = Turn on Asynch. Configures timing hardware, DIO lines and allocates the receiving buffer.

- Data type: UINT16 (type index = 0)
- Readable and writable
- Default value: 0
- This register uses system RAM. The maximum RAM is 64KB. For more information, see 4.4 RAM

ASYNCH_BAUD
- Address: 5420

The symbol rate that will be used for communication. 9600 is typical. Up to 38400 works, but heavily loads the T7's processor.

- Data type: UINT32 (type index = 1)
- Readable and writable

- Default value: 0

ASYNCH_RX_DIONUM
- Address: 5405

The DIO line that will receive data. (RX)

- Data type: UINT16 (type index = 0)
- Readable and writable
- Default value: 0

ASYNCH_TX_DIONUM
- Address: 5410

The DIO line that will transmit data. (TX)

- Data type: UINT16 (type index = 0)
- Readable and writable
- Default value: 0

ASYNCH_NUM_DATA_BITS
- Address: 5415

The number of data bits per frame. 0-8, 0=8.

- Data type: UINT16 (type index = 0)
- Readable and writable
- Default value: 0

ASYNCH_RX_BUFFER_SIZE_BYTES
- Address: 5430

Number of bytes to use for the receiving buffer. Max is 2048. 0 = 200

- Data type: UINT16 (type index = 0)
- Readable and writable
- Default value: 0

ASYNCH_NUM_BYTES_RX
- Address: 5435

The number of data bytes that have been received.

- Data type: UINT16 (type index = 0)
- Read-only
- Default value: 0

ASYNCH_NUM_BYTES_TX
- Address: 5440

The number of bytes to be transmitted after writing to GO. Max is 256.

- Data type: UINT16 (type index = 0)
- Readable and writable
- Default value: 0
- This register uses system RAM. The maximum RAM is 64KB. For more information, see 4.4 RAM

ASYNCH_TX_GO
- Address: 5450

Write a 1 to this register to initiate a transmission.

- Data type: UINT16 (type index = 0)
- Write-only
- Default value: 0

ASYNCH_NUM_STOP_BITS
- Address: 5455

The number of stop bits. Values: 0 = zero stop bits, 1 = one stop bit, 2 = two stop bits.

- Data type: UINT16 (type index = 0)
- Readable and writable
- Default value: 0

ASYNCH_PARITY
- Address: 5460

Parity setting: 0=none, 1=odd, 2=even.

- Data type: UINT16 (type index = 0)
- Readable and writable
- Default value: 0

ASYNCH_NUM_PARITY_ERRORS
- Address: 5465

The number of parity errors that have been detected. Cleared when UART is enabled. Can also be cleared by writing 0.

- Data type: UINT16 (type index = 0)
- Readable and writable
- Default value: 0

ASYNCH_DATA_TX
- Address: 5490

Write data to be transmitted here. This register is a buffer.

- Data type: UINT16 (type index = 0)
- Write-only
- Default value: 0
- This register is a Buffer Register

ASYNCH_DATA_RX
- Address: 5495

Read received data from here. This register is a buffer. Underrun behavior - fill with zeros.

- Data type: UINT16 (type index = 0)
- Read-only
- Default value: 0
- This register is a Buffer Register

All UART TAGS:

Name	Start Address	Type	Access
ASYNCH_ENABLE	5400	UINT16	R/W
ASYNCH_BAUD	5420	UINT32	R/W
ASYNCH_RX_DIONUM	5405	UINT16	R/W
ASYNCH_TX_DIONUM	5410	UINT16	R/W

ASYNCH_NUM_DATA_BITS	5415	UINT16	R/W
ASYNCH_RX_BUFFER_SIZE_BYTES	5430	UINT16	R/W
ASYNCH_NUM_BYTES_RX	5435	UINT16	R
ASYNCH_NUM_BYTES_TX	5440	UINT16	R/W
ASYNCH_TX_GO	5450	UINT16	W
ASYNCH_NUM_STOP_BITS	5455	UINT16	R/W
ASYNCH_PARITY	5460	UINT16	R/W
ASYNCH_NUM_PARITY_ERRORS	5465	UINT16	R/W
ASYNCH_DATA_TX	5490	UINT16	W
ASYNCH_DATA_RX	5495	UINT16	R

ASYNCH_ENABLE
- Address: 5400

1 = Turn on Asynch. Configures timing hardware, DIO lines and allocates the receiving buffer.

- Data type: UINT16 (type index = 0)
- Readable and writable
- Default value: 0
- This register uses system RAM. The maximum RAM is 64KB. For more information, see 4.4 RAM

ASYNCH_BAUD
- Address: 5420

The symbol rate that will be used for communication. 9600 is typical. Up to 38400 works, but heavily loads the T7's processor.

- Data type: UINT32 (type index = 1)
- Readable and writable
- Default value: 0

ASYNCH_RX_DIONUM
- Address: 5405

The DIO line that will receive data. (RX)

- Data type: UINT16 (type index = 0)
- Readable and writable
- Default value: 0

ASYNCH_TX_DIONUM
- Address: 5410

The DIO line that will transmit data. (TX)

- Data type: UINT16 (type index = 0)
- Readable and writable
- Default value: 0

ASYNCH_NUM_DATA_BITS
- Address: 5415

The number of data bits per frame. 0-8, 0=8.

- Data type: UINT16 (type index = 0)
- Readable and writable
- Default value: 0

ASYNCH_RX_BUFFER_SIZE_BYTES

- Address: 5430

Number of bytes to use for the receiving buffer. Max is 2048. 0 = 200

- Data type: UINT16 (type index = 0)
- Readable and writable
- Default value: 0

ASYNCH_NUM_BYTES_RX
- Address: 5435

The number of data bytes that have been received.

- Data type: UINT16 (type index = 0)
- Read-only
- Default value: 0

ASYNCH_NUM_BYTES_TX
- Address: 5440

The number of bytes to be transmitted after writing to GO. Max is 256.

- Data type: UINT16 (type index = 0)
- Readable and writable
- Default value: 0
- This register uses system RAM. The maximum RAM is 64KB. For more information, see 4.4 RAM

ASYNCH_TX_GO
- Address: 5450

Write a 1 to this register to initiate a transmission.

- Data type: UINT16 (type index = 0)
- Write-only
- Default value: 0

ASYNCH_NUM_STOP_BITS
- Address: 5455

The number of stop bits. Values: 0 = zero stop bits, 1 = one stop bit, 2 = two stop bits.

- Data type: UINT16 (type index = 0)
- Readable and writable
- Default value: 0

ASYNCH_PARITY
- Address: 5460

Parity setting: 0=none, 1=odd, 2=even.

- Data type: UINT16 (type index = 0)
- Readable and writable
- Default value: 0

ASYNCH_NUM_PARITY_ERRORS
- Address: 5465

The number of parity errors that have been detected. Cleared when UART is enabled. Can also be cleared by writing 0.

- Data type: UINT16 (type index = 0)

- Readable and writable
- Default value: 0

ASYNCH_DATA_TX
- Address: 5490

Write data to be transmitted here. This register is a buffer.

- Data type: UINT16 (type index = 0)
- Write-only
- Default value: 0
- This register is a Buffer Register

ASYNCH_DATA_RX
- Address: 5495

Read received data from here. This register is a buffer. Underrun behavior - fill with zeros.

- Data type: UINT16 (type index = 0)
- Read-only
- Default value: 0
- This register is a Buffer Register

All LUA TAGS:

Name	Start Address	Type	Access
LUA_RUN	6000	UINT32	R/W
LUA_SOURCE_SIZE	6012	UINT32	R/W
LUA_SOURCE_WRITE	6014	BYTE	W
LUA_DEBUG_ENABLE	6020	UINT32	R/W
LUA_DEBUG_NUM_BYTES	6022	UINT32	R
LUA_DEBUG_DATA	6024	BYTE	R
LUA_SAVE_TO_FLASH	6032	UINT32	R/W
LUA_LOAD_SAVED	6034	UINT32	W
LUA_SAVED_READ_POINTER	6036	UINT32	R/W
LUA_SAVED_READ	6038	UINT32	R
LUA_NO_WARN_TRUNCATION	6050	UINT16	R/W
LUA_RUN_DEFAULT	6100	UINT32	R/W
LUA_DEBUG_ENABLE_DEFAULT	6120	UINT32	R/W
LUA_DEBUG_NUM_BYTES_DEFAULT	6122	UINT32	R/W
USER_RAM#(0:38)_F32	46000	FLOAT32	R/W
USER_RAM#(0:8)_I32	46080	INT32	R/W
USER_RAM#(0:38)_U32	46100	UINT32	R/W
USER_RAM#(0:18)_U16	46180	UINT16	R/W
USER_RAM_FIFO#(0:2)_DATA_U16	47000	UINT16	R/W
USER_RAM_FIFO#(0:2)_DATA_U32	47010	UINT32	R/W
USER_RAM_FIFO#(0:2)_DATA_I32	47020	INT32	R/W
USER_RAM_FIFO#(0:2)_DATA_F32	47030	FLOAT32	R/W
USER_RAM_FIFO#(0:2)_ALLOCATE_NUM_BYTES	47900	UINT32	R/W
USER_RAM_FIFO#(0:2)_NUM_BYTES_IN_FIFO	47910	UINT32	R
USER_RAM_FIFO#(0:2)_EMPTY	47930	UINT32	W
LUA_NUM_IO_FLOATS	6006	UINT32	R/W
FILE_IO_LUA_SWITCH_FILE	60662	UINT32	R/W
BATTERY_RAM#(0:15)	61200	UINT32	R/W

LUA_RUN

_
- Address: 6000

Writing 1 compiles and runs the Lua script that is loaded in RAM. Writing zero stops the script and begins cleaning up memory. Users may poll the register after writing a value of 0 to verify that the VM is unloaded, and garbage collection is complete. 0 = VM fully unloaded. 1 = Running/in-progress

- Data type: UINT32 (type index = 1)
- Readable and writable
- This register uses system RAM. The maximum RAM is 64KB. For more information, see 4.4 RAM
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0018

LUA_SOURCE_SIZE
- Address: 6012

Allocates RAM for the source code.

- Data type: UINT32 (type index = 1)
- Readable and writable
- This register uses system RAM. The maximum RAM is 64KB. For more information, see 4.4 RAM
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0018

LUA_SOURCE_WRITE
- Address: 6014

Write the source code here. Source will be saved to the RAM allocated by LUA_SOURCE_SIZE. This register is a buffer.

- Data type: BYTE (type index = 99)
- Write-only
- This register is a Buffer Register
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0018

LUA_DEBUG_ENABLE
- Address: 6020

Write 1 to this register to enable debugging.

- Data type: UINT32 (type index = 1)
- Readable and writable
- This register uses system RAM. The maximum RAM is 64KB. For more information, see 4.4 RAM
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0018

LUA_DEBUG_NUM_BYTES
- Address: 6022

The number of data bytes in the debug buffer waiting to be read.

- Data type: UINT32 (type index = 1)
- Read-only
- T8:
    - Minimum firmware version: 0.0123
- T7:
    - Minimum firmware version: 1.0018

LUA_DEBUG_DATA
- Address: 6024

Read debug data from here. This register is a buffer.

- Data type: BYTE (type index = 99)
- Read-only
- This register is a Buffer Register
- T8:
    - Minimum firmware version: 0.0123
- T7:
    - Minimum firmware version: 1.0018

LUA_SAVE_TO_FLASH
- Address: 6032

Write 1 to save the loaded source code to flash.

- Data type: UINT32 (type index = 1)
- Readable and writable
- T8:
    - Minimum firmware version: 0.0123
- T7:
    - Minimum firmware version: 1.0059

LUA_LOAD_SAVED
- Address: 6034

Writing any value reads the script from Flash and loads it into RAM. The script can now be compiled and run.

- Data type: UINT32 (type index = 1)
- Write-only
- T8:
    - Minimum firmware version: 0.0123
- T7:
    - Minimum firmware version: 1.0059

LUA_SAVED_READ_POINTER
- Address: 6036

Address within the saved Lua script in Flash to begin reading from.

- Data type: UINT32 (type index = 1)
- Readable and writable
- T8:
    - Minimum firmware version: 0.0123
- T7:
    - Minimum firmware version: 1.0060

LUA_SAVED_READ
- Address: 6038

Read script saved to flash from here.

- Data type: UINT32 (type index = 1)
- Read-only
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0060

LUA_NO_WARN_TRUNCATION
- Address: 6050

Writing a 1 to this register will prevent truncation warnings from being displayed. This register will be cleared every time Lua is started.

- Data type: UINT16 (type index = 0)
- Readable and writable
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0285
- T4:
  - Minimum firmware version: 1.0028

LUA_RUN_DEFAULT
- Address: 6100

If set to 1 the script saved in flash will be loaded and run when the LabJack boots up.

- Data type: UINT32 (type index = 1)
- Readable and writable
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0078

LUA_DEBUG_ENABLE_DEFAULT
- Address: 6120

This is the value that will be loaded into LUA_DEBUG_ENABLE when the LabJack boots up.

- Data type: UINT32 (type index = 1)
- Readable and writable
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0078

LUA_DEBUG_NUM_BYTES_DEFAULT
- Address: 6122

This is the number of bytes that will be used for the lua debug buffer.

- Data type: UINT32 (type index = 1)

- Readable and writable
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0078

USER_RAM#(0:38)_F32
- Starting Address: 46000

Generic RAM registers. Useful for passing data between a host computer and a Lua script. Will not return an error if alternate data types are used.

- Data type: FLOAT32 (type index = 3)
- Readable and writable
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0023

Expanded Names	Addresses
USER_RAM0_F32, USER_RAM1_F32, USER_RAM2_F32, USER_RAM3_F32, USER_RAM4_F32, USER_RAM5_F32, USER_RAM6_F32, USER_RAM7_F32, USER_RAM8_F32, USER_RAM9_F32, USER_RAM10_F32, USER_RAM11_F32, USER_RAM12_F32, USER_RAM13_F32, USER_RAM14_F32, USER_RAM15_F32, USER_RAM16_F32, USER_RAM17_F32, USER_RAM18_F32, USER_RAM19_F32, USER_RAM20_F32, USER_RAM21_F32, USER_RAM22_F32, USER_RAM23_F32, USER_RAM24_F32, USER_RAM25_F32, USER_RAM26_F32, USER_RAM27_F32, USER_RAM28_F32, USER_RAM29_F32, USER_RAM30_F32, USER_RAM31_F32, USER_RAM32_F32, USER_RAM33_F32, USER_RAM34_F32, USER_RAM35_F32, USER_RAM36_F32, USER_RAM37_F32, USER_RAM38_F32	46000, 46002, 46004, 46006, 46008, 46010, 46012, 46014, 46016, 46018, 46020, 46022, 46024, 46026, 46028, 46030, 46032, 46034, 46036, 46038, 46040, 46042, 46044, 46046, 46048, 46050, 46052, 46054, 46056, 46058, 46060, 46062, 46064, 46066, 46068, 46070, 46072, 46074, 46076

USER_RAM#(0:8)_I32
- Starting Address: 46080

Generic RAM registers. Useful for passing data between a host computer and a Lua script. Will not return an error if alternate data types are used.

- Data type: INT32 (type index = 2)
- Readable and writable
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0162

Expanded Names	Addresses
USER_RAM0_I32, USER_RAM1_I32, USER_RAM2_I32, USER_RAM3_I32, USER_RAM4_I32, USER_RAM5_I32, USER_RAM6_I32, USER_RAM7_I32, USER_RAM8_I32	46080, 46082, 46084, 46086, 46088, 46090, 46092, 46094, 46096

USER_RAM#(0:38)_U32
- Starting Address: 46100

Generic RAM registers. Useful for passing data between a host computer and a Lua script. Will not return an error if alternate data types are used.

- Data type: UINT32 (type index = 1)
- Readable and writable
- T8:
    - Minimum firmware version: 0.0123
- T7:
    - Minimum firmware version: 1.0162

Expanded Names	Addresses
USER_RAM0_U32, USER_RAM1_U32, USER_RAM2_U32, USER_RAM3_U32, USER_RAM4_U32, USER_RAM5_U32, USER_RAM6_U32, USER_RAM7_U32, USER_RAM8_U32, USER_RAM9_U32, USER_RAM10_U32, USER_RAM11_U32, USER_RAM12_U32, USER_RAM13_U32, USER_RAM14_U32, USER_RAM15_U32, USER_RAM16_U32, USER_RAM17_U32, USER_RAM18_U32, USER_RAM19_U32, USER_RAM20_U32, USER_RAM21_U32, USER_RAM22_U32, USER_RAM23_U32, USER_RAM24_U32, USER_RAM25_U32, USER_RAM26_U32, USER_RAM27_U32, USER_RAM28_U32, USER_RAM29_U32, USER_RAM30_U32, USER_RAM31_U32, USER_RAM32_U32, USER_RAM33_U32, USER_RAM34_U32, USER_RAM35_U32, USER_RAM36_U32, USER_RAM37_U32, USER_RAM38_U32	46100, 46102, 46104, 46106, 46108, 46110, 46112, 46114, 46116, 46118, 46120, 46122, 46124, 46126, 46128, 46130, 46132, 46134, 46136, 46138, 46140, 46142, 46144, 46146, 46148, 46150, 46152, 46154, 46156, 46158, 46160, 46162, 46164, 46166, 46168, 46170, 46172, 46174, 46176

USER_RAM#(0:18)_U16
- Starting Address: 46180

Generic RAM registers. Useful for passing data between a host computer and a Lua script. Will not return an error if alternate data types are used.

- Data type: UINT16 (type index = 0)
- Readable and writable
- T8:
    - Minimum firmware version: 0.0123
- T7:
    - Minimum firmware version: 1.0162

Expanded Names	Addresses
USER_RAM0_U16, USER_RAM1_U16, USER_RAM2_U16, USER_RAM3_U16, USER_RAM4_U16, USER_RAM5_U16, USER_RAM6_U16, USER_RAM7_U16, USER_RAM8_U16, USER_RAM9_U16, USER_RAM10_U16, USER_RAM11_U16, USER_RAM12_U16, USER_RAM13_U16, USER_RAM14_U16, USER_RAM15_U16, USER_RAM16_U16, USER_RAM17_U16, USER_RAM18_U16	46180, 46181, 46182, 46183, 46184, 46185, 46186, 46187, 46188, 46189, 46190, 46191, 46192, 46193, 46194, 46195, 46196, 46197, 46198

USER_RAM_FIFO#(0:2)_DATA_U16
- Starting Address: 47000

Generic FIFO buffer. Useful for passing ORDERED or SEQUENTIAL data between various endpoints, such as between a host and a Lua script. Use up to 4 FIFO buffers simultaneously->1 of each data type, all 4 different data types, or a mixture. e.g. FIFO0_DATA_U16 points to the same memory as other FIFO0 registers, such that there are a total of 4 memory blocks: FIFO0, FIFO1, FIFO2 and FIFO3. It is possible to write into a FIFO buffer using a different datatype than is being used to read out of it. This register is a buffer. Underrun behavior - throws an error.

- Data type: UINT16 (type index = 0)
- Readable and writable
- Default value: 0
- This register is a Buffer Register
- T8:
    - Minimum firmware version: 0.0123

- T7:
    - Minimum firmware version: 1.0163

Expanded Names	Addresses
USER_RAM_FIFO0_DATA_U16, USER_RAM_FIFO1_DATA_U16, USER_RAM_FIFO2_DATA_U16	47000, 47001, 47002

USER_RAM_FIFO#(0:2)_DATA_U32
- Starting Address: 47010

Generic FIFO buffer. Useful for passing ORDERED or SEQUENTIAL data between various endpoints, such as between a host and a Lua script. Use up to 4 FIFO buffers simultaneously->1 of each data type, all 4 different data types, or a mixture. e.g. FIFO0_DATA_U16 points to the same memory as other FIFO0 registers, such that there are a total of 4 memory blocks: FIFO0, FIFO1, FIFO2 and FIFO3. It is possible to write into a FIFO buffer using a different datatype than is being used to read out of it. This register is a buffer. Underrun behavior - throws an error.

- Data type: UINT32 (type index = 1)
- Readable and writable
- Default value: 0
- This register is a Buffer Register
- T8:
    - Minimum firmware version: 0.0123
- T7:
    - Minimum firmware version: 1.0163

Expanded Names	Addresses
USER_RAM_FIFO0_DATA_U32, USER_RAM_FIFO1_DATA_U32, USER_RAM_FIFO2_DATA_U32	47010, 47012, 47014

USER_RAM_FIFO#(0:2)_DATA_I32
- Starting Address: 47020

Generic FIFO buffer. Useful for passing ORDERED or SEQUENTIAL data between various endpoints, such as between a host and a Lua script. Use up to 4 FIFO buffers simultaneously->1 of each data type, all 4 different data types, or a mixture. e.g. FIFO0_DATA_U16 points to the same memory as other FIFO0 registers, such that there are a total of 4 memory blocks: FIFO0, FIFO1, FIFO2 and FIFO3. It is possible to write into a FIFO buffer using a different datatype than is being used to read out of it. This register is a buffer. Underrun behavior - throws an error.

- Data type: INT32 (type index = 2)
- Readable and writable
- Default value: 0
- This register is a Buffer Register
- T8:
    - Minimum firmware version: 0.0123
- T7:
    - Minimum firmware version: 1.0163

Expanded Names	Addresses
USER_RAM_FIFO0_DATA_I32, USER_RAM_FIFO1_DATA_I32, USER_RAM_FIFO2_DATA_I32	47020, 47022, 47024

USER_RAM_FIFO#(0:2)_DATA_F32
- Starting Address: 47030

Generic FIFO buffer. Useful for passing ORDERED or SEQUENTIAL data between various endpoints, such as between a host and a Lua script. Use up to 4 FIFO buffers simultaneously->1 of each data type, all 4

different data types, or a mixture. e.g. FIFO0_DATA_U16 points to the same memory as other FIFO0 registers, such that there are a total of 4 memory blocks: FIFO0, FIFO1, FIFO2 and FIFO3. It is possible to write into a FIFO buffer using a different datatype than is being used to read out of it. This register is a buffer. Underrun behavior - throws an error.

- Data type: FLOAT32 (type index = 3)
- Readable and writable
- Default value: 0
- This register is a Buffer Register
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0163

Expanded Names	Addresses
USER_RAM_FIFO0_DATA_F32, USER_RAM_FIFO1_DATA_F32, USER_RAM_FIFO2_DATA_F32	47030, 47032, 47034

USER_RAM_FIFO#(0:2)_ALLOCATE_NUM_BYTES
- Starting Address: 47900

Allocate memory for a FIFO buffer. Number of bytes should be sufficient to store users max transfer array size. Note that FLOAT32, INT32, and UINT32 require 4 bytes per value, and UINT16 require 2 bytes per value. Maximum size is limited by available memory. Care should be taken to conserve enough memory for other operations such as AIN_EF, Lua, Stream etc.

- Data type: UINT32 (type index = 1)
- Readable and writable
- Default value: 0
- This register uses system RAM. The maximum RAM is 64KB. For more information, see 4.4 RAM
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0163

Expanded Names	Addresses
USER_RAM_FIFO0_ALLOCATE_NUM_BYTES, USER_RAM_FIFO1_ALLOCATE_NUM_BYTES, USER_RAM_FIFO2_ALLOCATE_NUM_BYTES	47900, 47902, 47904

USER_RAM_FIFO#(0:2)_NUM_BYTES_IN_FIFO
- Starting Address: 47910

Poll this register to see when new data is available/ready. Each read of the FIFO buffer decreases this value, and each write to the FIFO buffer increases this value. At any point in time, the following equation holds: Nbytes = Nwritten - Nread.

- Data type: UINT32 (type index = 1)
- Read-only
- Default value: 0
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0163

Expanded Names	Addresses

USER_RAM_FIFO0_NUM_BYTES_IN_FIFO, USER_RAM_FIFO1_NUM_BYTES_IN_FIFO, USER_RAM_FIFO2_NUM_BYTES_IN_FIFO	47910, 47912, 47914

USER_RAM_FIFO#(0:2)_EMPTY
- Starting Address: 47930

Write any value to this register to efficiently empty, flush, or otherwise clear data from the FIFO.

- Data type: UINT32 (type index = 1)
- Write-only
- Default value: 0
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0163

Expanded Names	Addresses
USER_RAM_FIFO0_EMPTY, USER_RAM_FIFO1_EMPTY, USER_RAM_FIFO2_EMPTY	47930, 47932, 47934

LUA_NUM_IO_FLOATS
- Address: 6006

Deprecated. Do not need to use this in FW 1.0163+, as 200 bytes of USER_RAM are pre-allocated. Allocates memory for x input-output floats in FW<1.0163.

- Data type: UINT32 (type index = 1)
- Readable and writable
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0023

FILE_IO_LUA_SWITCH_FILE
- Address: 60662

Write any value to this register to instruct Lua scripts to switch to a new file. Lua script should periodically check LJ.CheckFileFlag() to receive instruction, then call LJ.ClearFileFlag() after file switch is complete. Useful for applications that require continuous logging in a Lua script, and on-demand file access from a host.

- Data type: UINT32 (type index = 1)
- Readable and writable
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0168

BATTERY_RAM#(0:15)
- Starting Address: 61200

Saves or recalls data to or from the battery-backed SRAM. Only available on T7-Pros.

- Data type: UINT32 (type index = 1)
- Readable and writable
- Default value: 0
- T8:
  - Minimum firmware version: 0.0123

- T7:
  - Minimum firmware version: 1.0000

Expanded Names	Addresses
BATTERY_RAM0, BATTERY_RAM1, BATTERY_RAM2, BATTERY_RAM3, BATTERY_RAM4, BATTERY_RAM5, BATTERY_RAM6, BATTERY_RAM7, BATTERY_RAM8, BATTERY_RAM9, BATTERY_RAM10, BATTERY_RAM11, BATTERY_RAM12, BATTERY_RAM13, BATTERY_RAM14, BATTERY_RAM15	61200, 61202, 61204, 61206, 61208, 61210, 61212, 61214, 61216, 61218, 61220, 61222, 61224, 61226, 61228, 61230

All AIN_EF TAGS:

Name	Start Address	Type	Access
AIN#(0:148)_EF_READ_A	7000	FLOAT32	R
AIN#(0:148)_EF_READ_B	7300	FLOAT32	R/W
AIN#(0:148)_EF_READ_C	7600	FLOAT32	R/W
AIN#(0:148)_EF_READ_D	7900	FLOAT32	R
AIN#(0:6)_EF_READ_A_CAPTURE	8800	FLOAT32	R
AIN#(0:148)_EF_INDEX	9000	UINT32	R/W
AIN#(0:148)_EF_CONFIG_A	9300	UINT32	R/W
AIN#(0:148)_EF_CONFIG_B	9600	UINT32	R/W
AIN#(0:148)_EF_CONFIG_C	9900	UINT32	R/W
AIN#(0:148)_EF_CONFIG_D	10200	FLOAT32	R/W
AIN#(0:148)_EF_CONFIG_E	10500	FLOAT32	R/W
AIN#(0:148)_EF_CONFIG_F	10800	FLOAT32	R/W
AIN#(0:148)_EF_CONFIG_G	11100	FLOAT32	R/W
AIN#(0:148)_EF_CONFIG_H	11400	FLOAT32	R/W
AIN#(0:148)_EF_CONFIG_I	11700	FLOAT32	R/W
AIN#(0:148)_EF_CONFIG_J	12000	FLOAT32	R/W

AIN#(0:148)_EF_READ_A
- Starting Address: 7000

Function dependent on selected feature index.

- Data type: FLOAT32 (type index = 3)
- Read-only
- Default value: 0
- This register uses system RAM. The maximum RAM is 64KB. For more information, see 4.4 RAM
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0030

Expanded Names	Addresses

AIN0_EF_READ_A, AIN1_EF_READ_A, AIN2_EF_READ_A, AIN3_EF_READ_A, AIN4_EF_READ_A, AIN5_EF_READ_A, AIN6_EF_READ_A, AIN7_EF_READ_A, AIN8_EF_READ_A, AIN9_EF_READ_A, AIN10_EF_READ_A, AIN11_EF_READ_A, AIN12_EF_READ_A, AIN13_EF_READ_A, AIN14_EF_READ_A, AIN15_EF_READ_A, AIN16_EF_READ_A, AIN17_EF_READ_A, AIN18_EF_READ_A, AIN19_EF_READ_A, AIN20_EF_READ_A, AIN21_EF_READ_A, AIN22_EF_READ_A, AIN23_EF_READ_A, AIN24_EF_READ_A, AIN25_EF_READ_A, AIN26_EF_READ_A, AIN27_EF_READ_A, AIN28_EF_READ_A, AIN29_EF_READ_A, AIN30_EF_READ_A, AIN31_EF_READ_A, AIN32_EF_READ_A, AIN33_EF_READ_A, AIN34_EF_READ_A, AIN35_EF_READ_A, AIN36_EF_READ_A, AIN37_EF_READ_A, AIN38_EF_READ_A, AIN39_EF_READ_A, AIN40_EF_READ_A, AIN41_EF_READ_A, AIN42_EF_READ_A, AIN43_EF_READ_A, AIN44_EF_READ_A, AIN45_EF_READ_A, AIN46_EF_READ_A, AIN47_EF_READ_A, AIN48_EF_READ_A, AIN49_EF_READ_A, AIN50_EF_READ_A, AIN51_EF_READ_A, AIN52_EF_READ_A, AIN53_EF_READ_A, AIN54_EF_READ_A, AIN55_EF_READ_A, AIN56_EF_READ_A, AIN57_EF_READ_A, AIN58_EF_READ_A, AIN59_EF_READ_A, AIN60_EF_READ_A, AIN61_EF_READ_A, AIN62_EF_READ_A, AIN63_EF_READ_A, AIN64_EF_READ_A, AIN65_EF_READ_A, AIN66_EF_READ_A, AIN67_EF_READ_A, AIN68_EF_READ_A, AIN69_EF_READ_A, AIN70_EF_READ_A, AIN71_EF_READ_A, AIN72_EF_READ_A, AIN73_EF_READ_A, AIN74_EF_READ_A, AIN75_EF_READ_A, AIN76_EF_READ_A, AIN77_EF_READ_A, AIN78_EF_READ_A, AIN79_EF_READ_A, AIN80_EF_READ_A, AIN81_EF_READ_A, AIN82_EF_READ_A, AIN83_EF_READ_A, AIN84_EF_READ_A, AIN85_EF_READ_A, AIN86_EF_READ_A, AIN87_EF_READ_A, AIN88_EF_READ_A, AIN89_EF_READ_A, AIN90_EF_READ_A, AIN91_EF_READ_A, AIN92_EF_READ_A, AIN93_EF_READ_A, AIN94_EF_READ_A, AIN95_EF_READ_A, AIN96_EF_READ_A, AIN97_EF_READ_A, AIN98_EF_READ_A, AIN99_EF_READ_A, AIN100_EF_READ_A, AIN101_EF_READ_A, AIN102_EF_READ_A, AIN103_EF_READ_A, AIN104_EF_READ_A, AIN105_EF_READ_A, AIN106_EF_READ_A, AIN107_EF_READ_A, AIN108_EF_READ_A, AIN109_EF_READ_A, AIN110_EF_READ_A, AIN111_EF_READ_A, AIN112_EF_READ_A, AIN113_EF_READ_A, AIN114_EF_READ_A, AIN115_EF_READ_A, AIN116_EF_READ_A, AIN117_EF_READ_A, AIN118_EF_READ_A, AIN119_EF_READ_A, AIN120_EF_READ_A, AIN121_EF_READ_A, AIN122_EF_READ_A, AIN123_EF_READ_A, AIN124_EF_READ_A, AIN125_EF_READ_A, AIN126_EF_READ_A, AIN127_EF_READ_A, AIN128_EF_READ_A, AIN129_EF_READ_A, AIN130_EF_READ_A, AIN131_EF_READ_A, AIN132_EF_READ_A, AIN133_EF_READ_A, AIN134_EF_READ_A, AIN135_EF_READ_A, AIN136_EF_READ_A, AIN137_EF_READ_A, AIN138_EF_READ_A, AIN139_EF_READ_A, AIN140_EF_READ_A, AIN141_EF_READ_A, AIN142_EF_READ_A, AIN143_EF_READ_A, AIN144_EF_READ_A, AIN145_EF_READ_A, AIN146_EF_READ_A, AIN147_EF_READ_A, AIN148_EF_READ_A	7000, 7002, 7004, 7006, 7008, 7010, 7012, 7014, 7016, 7018, 7020, 7022, 7024, 7026, 7028, 7030, 7032, 7034, 7036, 7038, 7040, 7042, 7044, 7046, 7048, 7050, 7052, 7054, 7056, 7058, 7060, 7062, 7064, 7066, 7068, 7070, 7072, 7074, 7076, 7078, 7080, 7082, 7084, 7086, 7088, 7090, 7092, 7094, 7096, 7098, 7100, 7102, 7104, 7106, 7108, 7110, 7112, 7114, 7116, 7118, 7120, 7122, 7124, 7126, 7128, 7130, 7132, 7134, 7136, 7138, 7140, 7142, 7144, 7146, 7148, 7150, 7152, 7154, 7156, 7158, 7160, 7162, 7164, 7166, 7168, 7170, 7172, 7174, 7176, 7178, 7180, 7182, 7184, 7186, 7188, 7190, 7192, 7194, 7196, 7198, 7200, 7202, 7204, 7206, 7208, 7210, 7212, 7214, 7216, 7218, 7220, 7222, 7224, 7226, 7228, 7230, 7232, 7234, 7236, 7238, 7240, 7242, 7244, 7246, 7248, 7250, 7252, 7254, 7256, 7258, 7260, 7262, 7264, 7266, 7268, 7270, 7272, 7274, 7276, 7278, 7280, 7282, 7284, 7286, 7288, 7290, 7292, 7294, 7296

AIN#(0:148)_EF_READ_B
- Starting Address: 7300

Function dependent on selected feature index.

- Data type: FLOAT32 (type index = 3)
- Readable and writable
- Default value: 0
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0030

Expanded Names	Addresses

AIN0_EF_READ_B, AIN1_EF_READ_B, AIN2_EF_READ_B, AIN3_EF_READ_B, AIN4_EF_READ_B, AIN5_EF_READ_B, AIN6_EF_READ_B, AIN7_EF_READ_B, AIN8_EF_READ_B, AIN9_EF_READ_B, AIN10_EF_READ_B, AIN11_EF_READ_B, AIN12_EF_READ_B, AIN13_EF_READ_B, AIN14_EF_READ_B, AIN15_EF_READ_B, AIN16_EF_READ_B, AIN17_EF_READ_B, AIN18_EF_READ_B, AIN19_EF_READ_B, AIN20_EF_READ_B, AIN21_EF_READ_B, AIN22_EF_READ_B, AIN23_EF_READ_B, AIN24_EF_READ_B, AIN25_EF_READ_B, AIN26_EF_READ_B, AIN27_EF_READ_B, AIN28_EF_READ_B, AIN29_EF_READ_B, AIN30_EF_READ_B, AIN31_EF_READ_B, AIN32_EF_READ_B, AIN33_EF_READ_B, AIN34_EF_READ_B, AIN35_EF_READ_B, AIN36_EF_READ_B, AIN37_EF_READ_B, AIN38_EF_READ_B, AIN39_EF_READ_B, AIN40_EF_READ_B, AIN41_EF_READ_B, AIN42_EF_READ_B, AIN43_EF_READ_B, AIN44_EF_READ_B, AIN45_EF_READ_B, AIN46_EF_READ_B, AIN47_EF_READ_B, AIN48_EF_READ_B, AIN49_EF_READ_B, AIN50_EF_READ_B, AIN51_EF_READ_B, AIN52_EF_READ_B, AIN53_EF_READ_B, AIN54_EF_READ_B, AIN55_EF_READ_B, AIN56_EF_READ_B, AIN57_EF_READ_B, AIN58_EF_READ_B, AIN59_EF_READ_B, AIN60_EF_READ_B, AIN61_EF_READ_B, AIN62_EF_READ_B, AIN63_EF_READ_B, AIN64_EF_READ_B, AIN65_EF_READ_B, AIN66_EF_READ_B, AIN67_EF_READ_B, AIN68_EF_READ_B, AIN69_EF_READ_B, AIN70_EF_READ_B, AIN71_EF_READ_B, AIN72_EF_READ_B, AIN73_EF_READ_B, AIN74_EF_READ_B, AIN75_EF_READ_B, AIN76_EF_READ_B, AIN77_EF_READ_B, AIN78_EF_READ_B, AIN79_EF_READ_B, AIN80_EF_READ_B, AIN81_EF_READ_B, AIN82_EF_READ_B, AIN83_EF_READ_B, AIN84_EF_READ_B, AIN85_EF_READ_B, AIN86_EF_READ_B, AIN87_EF_READ_B, AIN88_EF_READ_B, AIN89_EF_READ_B, AIN90_EF_READ_B, AIN91_EF_READ_B, AIN92_EF_READ_B, AIN93_EF_READ_B, AIN94_EF_READ_B, AIN95_EF_READ_B, AIN96_EF_READ_B, AIN97_EF_READ_B, AIN98_EF_READ_B, AIN99_EF_READ_B, AIN100_EF_READ_B, AIN101_EF_READ_B, AIN102_EF_READ_B, AIN103_EF_READ_B, AIN104_EF_READ_B, AIN105_EF_READ_B, AIN106_EF_READ_B, AIN107_EF_READ_B, AIN108_EF_READ_B, AIN109_EF_READ_B, AIN110_EF_READ_B, AIN111_EF_READ_B, AIN112_EF_READ_B, AIN113_EF_READ_B, AIN114_EF_READ_B, AIN115_EF_READ_B, AIN116_EF_READ_B, AIN117_EF_READ_B, AIN118_EF_READ_B, AIN119_EF_READ_B, AIN120_EF_READ_B, AIN121_EF_READ_B, AIN122_EF_READ_B, AIN123_EF_READ_B, AIN124_EF_READ_B, AIN125_EF_READ_B, AIN126_EF_READ_B, AIN127_EF_READ_B, AIN128_EF_READ_B, AIN129_EF_READ_B, AIN130_EF_READ_B, AIN131_EF_READ_B, AIN132_EF_READ_B, AIN133_EF_READ_B, AIN134_EF_READ_B, AIN135_EF_READ_B, AIN136_EF_READ_B, AIN137_EF_READ_B, AIN138_EF_READ_B, AIN139_EF_READ_B, AIN140_EF_READ_B, AIN141_EF_READ_B, AIN142_EF_READ_B, AIN143_EF_READ_B, AIN144_EF_READ_B, AIN145_EF_READ_B, AIN146_EF_READ_B, AIN147_EF_READ_B, AIN148_EF_READ_B	7300, 7302, 7304, 7306, 7308, 7310, 7312, 7314, 7316, 7318, 7320, 7322, 7324, 7326, 7328, 7330, 7332, 7334, 7336, 7338, 7340, 7342, 7344, 7346, 7348, 7350, 7352, 7354, 7356, 7358, 7360, 7362, 7364, 7366, 7368, 7370, 7372, 7374, 7376, 7378, 7380, 7382, 7384, 7386, 7388, 7390, 7392, 7394, 7396, 7398, 7400, 7402, 7404, 7406, 7408, 7410, 7412, 7414, 7416, 7418, 7420, 7422, 7424, 7426, 7428, 7430, 7432, 7434, 7436, 7438, 7440, 7442, 7444, 7446, 7448, 7450, 7452, 7454, 7456, 7458, 7460, 7462, 7464, 7466, 7468, 7470, 7472, 7474, 7476, 7478, 7480, 7482, 7484, 7486, 7488, 7490, 7492, 7494, 7496, 7498, 7500, 7502, 7504, 7506, 7508, 7510, 7512, 7514, 7516, 7518, 7520, 7522, 7524, 7526, 7528, 7530, 7532, 7534, 7536, 7538, 7540, 7542, 7544, 7546, 7548, 7550, 7552, 7554, 7556, 7558, 7560, 7562, 7564, 7566, 7568, 7570, 7572, 7574, 7576, 7578, 7580, 7582, 7584, 7586, 7588, 7590, 7592, 7594, 7596

AIN#(0:148)_EF_READ_C
- Starting Address: 7600

Function dependent on selected feature index.

- Data type: FLOAT32 (type index = 3)
- Readable and writable
- Default value: 0
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0030

Expanded Names	Addresses

AIN0_EF_READ_C, AIN1_EF_READ_C, AIN2_EF_READ_C, AIN3_EF_READ_C, AIN4_EF_READ_C, AIN5_EF_READ_C, AIN6_EF_READ_C, AIN7_EF_READ_C, AIN8_EF_READ_C, AIN9_EF_READ_C, AIN10_EF_READ_C, AIN11_EF_READ_C, AIN12_EF_READ_C, AIN13_EF_READ_C, AIN14_EF_READ_C, AIN15_EF_READ_C, AIN16_EF_READ_C, AIN17_EF_READ_C, AIN18_EF_READ_C, AIN19_EF_READ_C, AIN20_EF_READ_C, AIN21_EF_READ_C, AIN22_EF_READ_C, AIN23_EF_READ_C, AIN24_EF_READ_C, AIN25_EF_READ_C, AIN26_EF_READ_C, AIN27_EF_READ_C, AIN28_EF_READ_C, AIN29_EF_READ_C, AIN30_EF_READ_C, AIN31_EF_READ_C, AIN32_EF_READ_C, AIN33_EF_READ_C, AIN34_EF_READ_C, AIN35_EF_READ_C, AIN36_EF_READ_C, AIN37_EF_READ_C, AIN38_EF_READ_C, AIN39_EF_READ_C, AIN40_EF_READ_C, AIN41_EF_READ_C, AIN42_EF_READ_C, AIN43_EF_READ_C, AIN44_EF_READ_C, AIN45_EF_READ_C, AIN46_EF_READ_C, AIN47_EF_READ_C, AIN48_EF_READ_C, AIN49_EF_READ_C, AIN50_EF_READ_C, AIN51_EF_READ_C, AIN52_EF_READ_C, AIN53_EF_READ_C, AIN54_EF_READ_C, AIN55_EF_READ_C, AIN56_EF_READ_C, AIN57_EF_READ_C, AIN58_EF_READ_C, AIN59_EF_READ_C, AIN60_EF_READ_C, AIN61_EF_READ_C, AIN62_EF_READ_C, AIN63_EF_READ_C, AIN64_EF_READ_C, AIN65_EF_READ_C, AIN66_EF_READ_C, AIN67_EF_READ_C, AIN68_EF_READ_C, AIN69_EF_READ_C, AIN70_EF_READ_C, AIN71_EF_READ_C, AIN72_EF_READ_C, AIN73_EF_READ_C, AIN74_EF_READ_C, AIN75_EF_READ_C, AIN76_EF_READ_C, AIN77_EF_READ_C, AIN78_EF_READ_C, AIN79_EF_READ_C, AIN80_EF_READ_C, AIN81_EF_READ_C, AIN82_EF_READ_C, AIN83_EF_READ_C, AIN84_EF_READ_C, AIN85_EF_READ_C, AIN86_EF_READ_C, AIN87_EF_READ_C, AIN88_EF_READ_C, AIN89_EF_READ_C, AIN90_EF_READ_C, AIN91_EF_READ_C, AIN92_EF_READ_C, AIN93_EF_READ_C, AIN94_EF_READ_C, AIN95_EF_READ_C, AIN96_EF_READ_C, AIN97_EF_READ_C, AIN98_EF_READ_C, AIN99_EF_READ_C, AIN100_EF_READ_C, AIN101_EF_READ_C, AIN102_EF_READ_C, AIN103_EF_READ_C, AIN104_EF_READ_C, AIN105_EF_READ_C, AIN106_EF_READ_C, AIN107_EF_READ_C, AIN108_EF_READ_C, AIN109_EF_READ_C, AIN110_EF_READ_C, AIN111_EF_READ_C, AIN112_EF_READ_C, AIN113_EF_READ_C, AIN114_EF_READ_C, AIN115_EF_READ_C, AIN116_EF_READ_C, AIN117_EF_READ_C, AIN118_EF_READ_C, AIN119_EF_READ_C, AIN120_EF_READ_C, AIN121_EF_READ_C, AIN122_EF_READ_C, AIN123_EF_READ_C, AIN124_EF_READ_C, AIN125_EF_READ_C, AIN126_EF_READ_C, AIN127_EF_READ_C, AIN128_EF_READ_C, AIN129_EF_READ_C, AIN130_EF_READ_C, AIN131_EF_READ_C, AIN132_EF_READ_C, AIN133_EF_READ_C, AIN134_EF_READ_C, AIN135_EF_READ_C, AIN136_EF_READ_C, AIN137_EF_READ_C, AIN138_EF_READ_C, AIN139_EF_READ_C, AIN140_EF_READ_C, AIN141_EF_READ_C, AIN142_EF_READ_C, AIN143_EF_READ_C, AIN144_EF_READ_C, AIN145_EF_READ_C, AIN146_EF_READ_C, AIN147_EF_READ_C, AIN148_EF_READ_C	7600, 7602, 7604, 7606, 7608, 7610, 7612, 7614, 7616, 7618, 7620, 7622, 7624, 7626, 7628, 7630, 7632, 7634, 7636, 7638, 7640, 7642, 7644, 7646, 7648, 7650, 7652, 7654, 7656, 7658, 7660, 7662, 7664, 7666, 7668, 7670, 7672, 7674, 7676, 7678, 7680, 7682, 7684, 7686, 7688, 7690, 7692, 7694, 7696, 7698, 7700, 7702, 7704, 7706, 7708, 7710, 7712, 7714, 7716, 7718, 7720, 7722, 7724, 7726, 7728, 7730, 7732, 7734, 7736, 7738, 7740, 7742, 7744, 7746, 7748, 7750, 7752, 7754, 7756, 7758, 7760, 7762, 7764, 7766, 7768, 7770, 7772, 7774, 7776, 7778, 7780, 7782, 7784, 7786, 7788, 7790, 7792, 7794, 7796, 7798, 7800, 7802, 7804, 7806, 7808, 7810, 7812, 7814, 7816, 7818, 7820, 7822, 7824, 7826, 7828, 7830, 7832, 7834, 7836, 7838, 7840, 7842, 7844, 7846, 7848, 7850, 7852, 7854, 7856, 7858, 7860, 7862, 7864, 7866, 7868, 7870, 7872, 7874, 7876, 7878, 7880, 7882, 7884, 7886, 7888, 7890, 7892, 7894, 7896

AIN#(0:148)_EF_READ_D
- Starting Address: 7900

Function dependent on selected feature index.

- Data type: FLOAT32 (type index = 3)
- Read-only
- Default value: 0
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0030

Expanded Names	Addresses

AIN0_EF_READ_D, AIN1_EF_READ_D, AIN2_EF_READ_D, AIN3_EF_READ_D, AIN4_EF_READ_D, AIN5_EF_READ_D, AIN6_EF_READ_D, AIN7_EF_READ_D, AIN8_EF_READ_D, AIN9_EF_READ_D, AIN10_EF_READ_D, AIN11_EF_READ_D, AIN12_EF_READ_D, AIN13_EF_READ_D, AIN14_EF_READ_D, AIN15_EF_READ_D, AIN16_EF_READ_D, AIN17_EF_READ_D, AIN18_EF_READ_D, AIN19_EF_READ_D, AIN20_EF_READ_D, AIN21_EF_READ_D, AIN22_EF_READ_D, AIN23_EF_READ_D, AIN24_EF_READ_D, AIN25_EF_READ_D, AIN26_EF_READ_D, AIN27_EF_READ_D, AIN28_EF_READ_D, AIN29_EF_READ_D, AIN30_EF_READ_D, AIN31_EF_READ_D, AIN32_EF_READ_D, AIN33_EF_READ_D, AIN34_EF_READ_D, AIN35_EF_READ_D, AIN36_EF_READ_D, AIN37_EF_READ_D, AIN38_EF_READ_D, AIN39_EF_READ_D, AIN40_EF_READ_D, AIN41_EF_READ_D, AIN42_EF_READ_D, AIN43_EF_READ_D, AIN44_EF_READ_D, AIN45_EF_READ_D, AIN46_EF_READ_D, AIN47_EF_READ_D, AIN48_EF_READ_D, AIN49_EF_READ_D, AIN50_EF_READ_D, AIN51_EF_READ_D, AIN52_EF_READ_D, AIN53_EF_READ_D, AIN54_EF_READ_D, AIN55_EF_READ_D, AIN56_EF_READ_D, AIN57_EF_READ_D, AIN58_EF_READ_D, AIN59_EF_READ_D, AIN60_EF_READ_D, AIN61_EF_READ_D, AIN62_EF_READ_D, AIN63_EF_READ_D, AIN64_EF_READ_D, AIN65_EF_READ_D, AIN66_EF_READ_D, AIN67_EF_READ_D, AIN68_EF_READ_D, AIN69_EF_READ_D, AIN70_EF_READ_D, AIN71_EF_READ_D, AIN72_EF_READ_D, AIN73_EF_READ_D, AIN74_EF_READ_D, AIN75_EF_READ_D, AIN76_EF_READ_D, AIN77_EF_READ_D, AIN78_EF_READ_D, AIN79_EF_READ_D, AIN80_EF_READ_D, AIN81_EF_READ_D, AIN82_EF_READ_D, AIN83_EF_READ_D, AIN84_EF_READ_D, AIN85_EF_READ_D, AIN86_EF_READ_D, AIN87_EF_READ_D, AIN88_EF_READ_D, AIN89_EF_READ_D, AIN90_EF_READ_D, AIN91_EF_READ_D, AIN92_EF_READ_D, AIN93_EF_READ_D, AIN94_EF_READ_D, AIN95_EF_READ_D, AIN96_EF_READ_D, AIN97_EF_READ_D, AIN98_EF_READ_D, AIN99_EF_READ_D, AIN100_EF_READ_D, AIN101_EF_READ_D, AIN102_EF_READ_D, AIN103_EF_READ_D, AIN104_EF_READ_D, AIN105_EF_READ_D, AIN106_EF_READ_D, AIN107_EF_READ_D, AIN108_EF_READ_D, AIN109_EF_READ_D, AIN110_EF_READ_D, AIN111_EF_READ_D, AIN112_EF_READ_D, AIN113_EF_READ_D, AIN114_EF_READ_D, AIN115_EF_READ_D, AIN116_EF_READ_D, AIN117_EF_READ_D, AIN118_EF_READ_D, AIN119_EF_READ_D, AIN120_EF_READ_D, AIN121_EF_READ_D, AIN122_EF_READ_D, AIN123_EF_READ_D, AIN124_EF_READ_D, AIN125_EF_READ_D, AIN126_EF_READ_D, AIN127_EF_READ_D, AIN128_EF_READ_D, AIN129_EF_READ_D, AIN130_EF_READ_D, AIN131_EF_READ_D, AIN132_EF_READ_D, AIN133_EF_READ_D, AIN134_EF_READ_D, AIN135_EF_READ_D, AIN136_EF_READ_D, AIN137_EF_READ_D, AIN138_EF_READ_D, AIN139_EF_READ_D, AIN140_EF_READ_D, AIN141_EF_READ_D, AIN142_EF_READ_D, AIN143_EF_READ_D, AIN144_EF_READ_D, AIN145_EF_READ_D, AIN146_EF_READ_D, AIN147_EF_READ_D, AIN148_EF_READ_D	7900, 7902, 7904, 7906, 7908, 7910, 7912, 7914, 7916, 7918, 7920, 7922, 7924, 7926, 7928, 7930, 7932, 7934, 7936, 7938, 7940, 7942, 7944, 7946, 7948, 7950, 7952, 7954, 7956, 7958, 7960, 7962, 7964, 7966, 7968, 7970, 7972, 7974, 7976, 7978, 7980, 7982, 7984, 7986, 7988, 7990, 7992, 7994, 7996, 7998, 8000, 8002, 8004, 8006, 8008, 8010, 8012, 8014, 8016, 8018, 8020, 8022, 8024, 8026, 8028, 8030, 8032, 8034, 8036, 8038, 8040, 8042, 8044, 8046, 8048, 8050, 8052, 8054, 8056, 8058, 8060, 8062, 8064, 8066, 8068, 8070, 8072, 8074, 8076, 8078, 8080, 8082, 8084, 8086, 8088, 8090, 8092, 8094, 8096, 8098, 8100, 8102, 8104, 8106, 8108, 8110, 8112, 8114, 8116, 8118, 8120, 8122, 8124, 8126, 8128, 8130, 8132, 8134, 8136, 8138, 8140, 8142, 8144, 8146, 8148, 8150, 8152, 8154, 8156, 8158, 8160, 8162, 8164, 8166, 8168, 8170, 8172, 8174, 8176, 8178, 8180, 8182, 8184, 8186, 8188, 8190, 8192, 8194, 8196

AIN#(0:6)_EF_READ_A_CAPTURE
- Starting Address: 8800

(T8 only) Read AIN_EF with the last capture sampled, use AIN#(0:7)_EF_READ_A to read a new sample

- Data type: FLOAT32 (type index = 3)
- Read-only
- Default value: 0
- T8:
    - Minimum firmware version: 1.0014

Expanded Names	Addresses
AIN0_EF_READ_A_CAPTURE, AIN1_EF_READ_A_CAPTURE, AIN2_EF_READ_A_CAPTURE, AIN3_EF_READ_A_CAPTURE, AIN4_EF_READ_A_CAPTURE, AIN5_EF_READ_A_CAPTURE, AIN6_EF_READ_A_CAPTURE	8800, 8802, 8804, 8806, 8808, 8810, 8812

AIN#(0:148)_EF_INDEX
- Starting Address: 9000

Specify the desired extended feature for this analog input with the index value. List of index values: 0=None(disabled); 1=Offset and Slope; 3=Max/Min/Avg; 4=Resistance; 5=Average and Threshold; 10=RMS Flex; 11=FlexRMS; 20=Thermocouple type E; 21=Thermocouple type J; 22=Thermocouple type K; 23=Thermocouple type R; 24=Thermocouple type T; 25=Thermocouple type S; 27=Thermocouple type N; 28=Thermocouple type B; 30=Thermocouple type C; 40=RTD model PT100; 41=RTD model PT500; 42=RTD model PT1000; 50=Thermistor Steinhart-Hart; 51=Thermistor Beta.

- Data type: UINT32 (type index = 1)
- Readable and writable
- Default value: 0
- This register uses system RAM. The maximum RAM is 64KB. For more information, see 4.4 RAM
- T8:
    - Minimum firmware version: 0.0123
- T7:
    - Minimum firmware version: 1.0030
- T4:
    - The T4 does not support thermocouple modes.

Expanded Names	Addresses
AIN0_EF_INDEX, AIN1_EF_INDEX, AIN2_EF_INDEX, AIN3_EF_INDEX, AIN4_EF_INDEX, AIN5_EF_INDEX, AIN6_EF_INDEX, AIN7_EF_INDEX, AIN8_EF_INDEX, AIN9_EF_INDEX, AIN10_EF_INDEX, AIN11_EF_INDEX, AIN12_EF_INDEX, AIN13_EF_INDEX, AIN14_EF_INDEX, AIN15_EF_INDEX, AIN16_EF_INDEX, AIN17_EF_INDEX, AIN18_EF_INDEX, AIN19_EF_INDEX, AIN20_EF_INDEX, AIN21_EF_INDEX, AIN22_EF_INDEX, AIN23_EF_INDEX, AIN24_EF_INDEX, AIN25_EF_INDEX, AIN26_EF_INDEX, AIN27_EF_INDEX, AIN28_EF_INDEX, AIN29_EF_INDEX, AIN30_EF_INDEX, AIN31_EF_INDEX, AIN32_EF_INDEX, AIN33_EF_INDEX, AIN34_EF_INDEX, AIN35_EF_INDEX, AIN36_EF_INDEX, AIN37_EF_INDEX, AIN38_EF_INDEX, AIN39_EF_INDEX, AIN40_EF_INDEX, AIN41_EF_INDEX, AIN42_EF_INDEX, AIN43_EF_INDEX, AIN44_EF_INDEX, AIN45_EF_INDEX, AIN46_EF_INDEX, AIN47_EF_INDEX, AIN48_EF_INDEX, AIN49_EF_INDEX, AIN50_EF_INDEX, AIN51_EF_INDEX, AIN52_EF_INDEX, AIN53_EF_INDEX, AIN54_EF_INDEX, AIN55_EF_INDEX, AIN56_EF_INDEX, AIN57_EF_INDEX, AIN58_EF_INDEX, AIN59_EF_INDEX, AIN60_EF_INDEX, AIN61_EF_INDEX, AIN62_EF_INDEX, AIN63_EF_INDEX, AIN64_EF_INDEX, AIN65_EF_INDEX, AIN66_EF_INDEX, AIN67_EF_INDEX, AIN68_EF_INDEX, AIN69_EF_INDEX, AIN70_EF_INDEX, AIN71_EF_INDEX, AIN72_EF_INDEX, AIN73_EF_INDEX, AIN74_EF_INDEX, AIN75_EF_INDEX, AIN76_EF_INDEX, AIN77_EF_INDEX, AIN78_EF_INDEX, AIN79_EF_INDEX, AIN80_EF_INDEX, AIN81_EF_INDEX, AIN82_EF_INDEX, AIN83_EF_INDEX, AIN84_EF_INDEX, AIN85_EF_INDEX, AIN86_EF_INDEX, AIN87_EF_INDEX, AIN88_EF_INDEX, AIN89_EF_INDEX, AIN90_EF_INDEX, AIN91_EF_INDEX, AIN92_EF_INDEX, AIN93_EF_INDEX, AIN94_EF_INDEX, AIN95_EF_INDEX, AIN96_EF_INDEX, AIN97_EF_INDEX, AIN98_EF_INDEX, AIN99_EF_INDEX, AIN100_EF_INDEX, AIN101_EF_INDEX, AIN102_EF_INDEX, AIN103_EF_INDEX, AIN104_EF_INDEX, AIN105_EF_INDEX, AIN106_EF_INDEX, AIN107_EF_INDEX, AIN108_EF_INDEX, AIN109_EF_INDEX, AIN110_EF_INDEX, AIN111_EF_INDEX, AIN112_EF_INDEX, AIN113_EF_INDEX, AIN114_EF_INDEX, AIN115_EF_INDEX, AIN116_EF_INDEX, AIN117_EF_INDEX, AIN118_EF_INDEX, AIN119_EF_INDEX, AIN120_EF_INDEX, AIN121_EF_INDEX, AIN122_EF_INDEX, AIN123_EF_INDEX, AIN124_EF_INDEX, AIN125_EF_INDEX, AIN126_EF_INDEX, AIN127_EF_INDEX, AIN128_EF_INDEX, AIN129_EF_INDEX, AIN130_EF_INDEX, AIN131_EF_INDEX, AIN132_EF_INDEX, AIN133_EF_INDEX, AIN134_EF_INDEX, AIN135_EF_INDEX, AIN136_EF_INDEX, AIN137_EF_INDEX, AIN138_EF_INDEX, AIN139_EF_INDEX, AIN140_EF_INDEX, AIN141_EF_INDEX, AIN142_EF_INDEX, AIN143_EF_INDEX, AIN144_EF_INDEX, AIN145_EF_INDEX, AIN146_EF_INDEX, AIN147_EF_INDEX, AIN148_EF_INDEX	9000, 9002, 9004, 9006, 9008, 9010, 9012, 9014, 9016, 9018, 9020, 9022, 9024, 9026, 9028, 9030, 9032, 9034, 9036, 9038, 9040, 9042, 9044, 9046, 9048, 9050, 9052, 9054, 9056, 9058, 9060, 9062, 9064, 9066, 9068, 9070, 9072, 9074, 9076, 9078, 9080, 9082, 9084, 9086, 9088, 9090, 9092, 9094, 9096, 9098, 9100, 9102, 9104, 9106, 9108, 9110, 9112, 9114, 9116, 9118, 9120, 9122, 9124, 9126, 9128, 9130, 9132, 9134, 9136, 9138, 9140, 9142, 9144, 9146, 9148, 9150, 9152, 9154, 9156, 9158, 9160, 9162, 9164, 9166, 9168, 9170, 9172, 9174, 9176, 9178, 9180, 9182, 9184, 9186, 9188, 9190, 9192, 9194, 9196, 9198, 9200, 9202, 9204, 9206, 9208, 9210, 9212, 9214, 9216, 9218, 9220, 9222, 9224, 9226, 9228, 9230, 9232, 9234, 9236, 9238, 9240, 9242, 9244, 9246, 9248, 9250, 9252, 9254, 9256, 9258, 9260, 9262, 9264, 9266, 9268, 9270, 9272, 9274, 9276, 9278, 9280, 9282, 9284, 9286, 9288, 9290, 9292, 9294, 9296

AIN#(0:148)_EF_CONFIG_A
- Starting Address: 9300

Function dependent on selected feature index.

- Data type: UINT32 (type index = 1)
- Readable and writable
- Default value: 0
- T8:
    - Minimum firmware version: 0.0123
- T7:
    - Minimum firmware version: 1.0030

Expanded Names	Addresses

AIN0_EF_CONFIG_A, AIN1_EF_CONFIG_A, AIN2_EF_CONFIG_A,	9300, 9302, 9304,
AIN3_EF_CONFIG_A, AIN4_EF_CONFIG_A, AIN5_EF_CONFIG_A,	9306, 9308, 9310,
AIN6_EF_CONFIG_A, AIN7_EF_CONFIG_A, AIN8_EF_CONFIG_A,	9312, 9314, 9316,
AIN9_EF_CONFIG_A, AIN10_EF_CONFIG_A, AIN11_EF_CONFIG_A,	9318, 9320, 9322,
AIN12_EF_CONFIG_A, AIN13_EF_CONFIG_A, AIN14_EF_CONFIG_A,	9324, 9326, 9328,
AIN15_EF_CONFIG_A, AIN16_EF_CONFIG_A, AIN17_EF_CONFIG_A,	9330, 9332, 9334,
AIN18_EF_CONFIG_A, AIN19_EF_CONFIG_A, AIN20_EF_CONFIG_A,	9336, 9338, 9340,
AIN21_EF_CONFIG_A, AIN22_EF_CONFIG_A, AIN23_EF_CONFIG_A,	9342, 9344, 9346,
AIN24_EF_CONFIG_A, AIN25_EF_CONFIG_A, AIN26_EF_CONFIG_A,	9348, 9350, 9352,
AIN27_EF_CONFIG_A, AIN28_EF_CONFIG_A, AIN29_EF_CONFIG_A,	9354, 9356, 9358,
AIN30_EF_CONFIG_A, AIN31_EF_CONFIG_A, AIN32_EF_CONFIG_A,	9360, 9362, 9364,
AIN33_EF_CONFIG_A, AIN34_EF_CONFIG_A, AIN35_EF_CONFIG_A,	9366, 9368, 9370,
AIN36_EF_CONFIG_A, AIN37_EF_CONFIG_A, AIN38_EF_CONFIG_A,	9372, 9374, 9376,
AIN39_EF_CONFIG_A, AIN40_EF_CONFIG_A, AIN41_EF_CONFIG_A,	9378, 9380, 9382,
AIN42_EF_CONFIG_A, AIN43_EF_CONFIG_A, AIN44_EF_CONFIG_A,	9384, 9386, 9388,
AIN45_EF_CONFIG_A, AIN46_EF_CONFIG_A, AIN47_EF_CONFIG_A,	9390, 9392, 9394,
AIN48_EF_CONFIG_A, AIN49_EF_CONFIG_A, AIN50_EF_CONFIG_A,	9396, 9398, 9400,
AIN51_EF_CONFIG_A, AIN52_EF_CONFIG_A, AIN53_EF_CONFIG_A,	9402, 9404, 9406,
AIN54_EF_CONFIG_A, AIN55_EF_CONFIG_A, AIN56_EF_CONFIG_A,	9408, 9410, 9412,
AIN57_EF_CONFIG_A, AIN58_EF_CONFIG_A, AIN59_EF_CONFIG_A,	9414, 9416, 9418,
AIN60_EF_CONFIG_A, AIN61_EF_CONFIG_A, AIN62_EF_CONFIG_A,	9420, 9422, 9424,
AIN63_EF_CONFIG_A, AIN64_EF_CONFIG_A, AIN65_EF_CONFIG_A,	9426, 9428, 9430,
AIN66_EF_CONFIG_A, AIN67_EF_CONFIG_A, AIN68_EF_CONFIG_A,	9432, 9434, 9436,
AIN69_EF_CONFIG_A, AIN70_EF_CONFIG_A, AIN71_EF_CONFIG_A,	9438, 9440, 9442,
AIN72_EF_CONFIG_A, AIN73_EF_CONFIG_A, AIN74_EF_CONFIG_A,	9444, 9446, 9448,
AIN75_EF_CONFIG_A, AIN76_EF_CONFIG_A, AIN77_EF_CONFIG_A,	9450, 9452, 9454,
AIN78_EF_CONFIG_A, AIN79_EF_CONFIG_A, AIN80_EF_CONFIG_A,	9456, 9458, 9460,
AIN81_EF_CONFIG_A, AIN82_EF_CONFIG_A, AIN83_EF_CONFIG_A,	9462, 9464, 9466,
AIN84_EF_CONFIG_A, AIN85_EF_CONFIG_A, AIN86_EF_CONFIG_A,	9468, 9470, 9472,
AIN87_EF_CONFIG_A, AIN88_EF_CONFIG_A, AIN89_EF_CONFIG_A,	9474, 9476, 9478,
AIN90_EF_CONFIG_A, AIN91_EF_CONFIG_A, AIN92_EF_CONFIG_A,	9480, 9482, 9484,
AIN93_EF_CONFIG_A, AIN94_EF_CONFIG_A, AIN95_EF_CONFIG_A,	9486, 9488, 9490,
AIN96_EF_CONFIG_A, AIN97_EF_CONFIG_A, AIN98_EF_CONFIG_A,	9492, 9494, 9496,
AIN99_EF_CONFIG_A, AIN100_EF_CONFIG_A, AIN101_EF_CONFIG_A,	9498, 9500, 9502,
AIN102_EF_CONFIG_A, AIN103_EF_CONFIG_A, AIN104_EF_CONFIG_A,	9504, 9506, 9508,
AIN105_EF_CONFIG_A, AIN106_EF_CONFIG_A, AIN107_EF_CONFIG_A,	9510, 9512, 9514,
AIN108_EF_CONFIG_A, AIN109_EF_CONFIG_A, AIN110_EF_CONFIG_A,	9516, 9518, 9520,
AIN111_EF_CONFIG_A, AIN112_EF_CONFIG_A, AIN113_EF_CONFIG_A,	9522, 9524, 9526,
AIN114_EF_CONFIG_A, AIN115_EF_CONFIG_A, AIN116_EF_CONFIG_A,	9528, 9530, 9532,
AIN117_EF_CONFIG_A, AIN118_EF_CONFIG_A, AIN119_EF_CONFIG_A,	9534, 9536, 9538,
AIN120_EF_CONFIG_A, AIN121_EF_CONFIG_A, AIN122_EF_CONFIG_A,	9540, 9542, 9544,
AIN123_EF_CONFIG_A, AIN124_EF_CONFIG_A, AIN125_EF_CONFIG_A,	9546, 9548, 9550,
AIN126_EF_CONFIG_A, AIN127_EF_CONFIG_A, AIN128_EF_CONFIG_A,	9552, 9554, 9556,
AIN129_EF_CONFIG_A, AIN130_EF_CONFIG_A, AIN131_EF_CONFIG_A,	9558, 9560, 9562,
AIN132_EF_CONFIG_A, AIN133_EF_CONFIG_A, AIN134_EF_CONFIG_A,	9564, 9566, 9568,
AIN135_EF_CONFIG_A, AIN136_EF_CONFIG_A, AIN137_EF_CONFIG_A,	9570, 9572, 9574,
AIN138_EF_CONFIG_A, AIN139_EF_CONFIG_A, AIN140_EF_CONFIG_A,	9576, 9578, 9580,
AIN141_EF_CONFIG_A, AIN142_EF_CONFIG_A, AIN143_EF_CONFIG_A,	9582, 9584, 9586,
AIN144_EF_CONFIG_A, AIN145_EF_CONFIG_A, AIN146_EF_CONFIG_A,	9588, 9590, 9592,
AIN147_EF_CONFIG_A, AIN148_EF_CONFIG_A	9594, 9596

AIN#(0:148)_EF_CONFIG_B
- Starting Address: 9600

Function dependent on selected feature index.

- Data type: UINT32 (type index = 1)
- Readable and writable
- Default value: 0
- T8:
    - Minimum firmware version: 0.0123
- T7:

- Minimum firmware version: 1.0030

Expanded Names	Addresses
AIN0_EF_CONFIG_B, AIN1_EF_CONFIG_B, AIN2_EF_CONFIG_B,	9600, 9602, 9604,
AIN3_EF_CONFIG_B, AIN4_EF_CONFIG_B, AIN5_EF_CONFIG_B,	9606, 9608, 9610,
AIN6_EF_CONFIG_B, AIN7_EF_CONFIG_B, AIN8_EF_CONFIG_B,	9612, 9614, 9616,
AIN9_EF_CONFIG_B, AIN10_EF_CONFIG_B, AIN11_EF_CONFIG_B,	9618, 9620, 9622,
AIN12_EF_CONFIG_B, AIN13_EF_CONFIG_B, AIN14_EF_CONFIG_B,	9624, 9626, 9628,
AIN15_EF_CONFIG_B, AIN16_EF_CONFIG_B, AIN17_EF_CONFIG_B,	9630, 9632, 9634,
AIN18_EF_CONFIG_B, AIN19_EF_CONFIG_B, AIN20_EF_CONFIG_B,	9636, 9638, 9640,
AIN21_EF_CONFIG_B, AIN22_EF_CONFIG_B, AIN23_EF_CONFIG_B,	9642, 9644, 9646,
AIN24_EF_CONFIG_B, AIN25_EF_CONFIG_B, AIN26_EF_CONFIG_B,	9648, 9650, 9652,
AIN27_EF_CONFIG_B, AIN28_EF_CONFIG_B, AIN29_EF_CONFIG_B,	9654, 9656, 9658,
AIN30_EF_CONFIG_B, AIN31_EF_CONFIG_B, AIN32_EF_CONFIG_B,	9660, 9662, 9664,
AIN33_EF_CONFIG_B, AIN34_EF_CONFIG_B, AIN35_EF_CONFIG_B,	9666, 9668, 9670,
AIN36_EF_CONFIG_B, AIN37_EF_CONFIG_B, AIN38_EF_CONFIG_B,	9672, 9674, 9676,
AIN39_EF_CONFIG_B, AIN40_EF_CONFIG_B, AIN41_EF_CONFIG_B,	9678, 9680, 9682,
AIN42_EF_CONFIG_B, AIN43_EF_CONFIG_B, AIN44_EF_CONFIG_B,	9684, 9686, 9688,
AIN45_EF_CONFIG_B, AIN46_EF_CONFIG_B, AIN47_EF_CONFIG_B,	9690, 9692, 9694,
AIN48_EF_CONFIG_B, AIN49_EF_CONFIG_B, AIN50_EF_CONFIG_B,	9696, 9698, 9700,
AIN51_EF_CONFIG_B, AIN52_EF_CONFIG_B, AIN53_EF_CONFIG_B,	9702, 9704, 9706,
AIN54_EF_CONFIG_B, AIN55_EF_CONFIG_B, AIN56_EF_CONFIG_B,	9708, 9710, 9712,
AIN57_EF_CONFIG_B, AIN58_EF_CONFIG_B, AIN59_EF_CONFIG_B,	9714, 9716, 9718,
AIN60_EF_CONFIG_B, AIN61_EF_CONFIG_B, AIN62_EF_CONFIG_B,	9720, 9722, 9724,
AIN63_EF_CONFIG_B, AIN64_EF_CONFIG_B, AIN65_EF_CONFIG_B,	9726, 9728, 9730,
AIN66_EF_CONFIG_B, AIN67_EF_CONFIG_B, AIN68_EF_CONFIG_B,	9732, 9734, 9736,
AIN69_EF_CONFIG_B, AIN70_EF_CONFIG_B, AIN71_EF_CONFIG_B,	9738, 9740, 9742,
AIN72_EF_CONFIG_B, AIN73_EF_CONFIG_B, AIN74_EF_CONFIG_B,	9744, 9746, 9748,
AIN75_EF_CONFIG_B, AIN76_EF_CONFIG_B, AIN77_EF_CONFIG_B,	9750, 9752, 9754,
AIN78_EF_CONFIG_B, AIN79_EF_CONFIG_B, AIN80_EF_CONFIG_B,	9756, 9758, 9760,
AIN81_EF_CONFIG_B, AIN82_EF_CONFIG_B, AIN83_EF_CONFIG_B,	9762, 9764, 9766,
AIN84_EF_CONFIG_B, AIN85_EF_CONFIG_B, AIN86_EF_CONFIG_B,	9768, 9770, 9772,
AIN87_EF_CONFIG_B, AIN88_EF_CONFIG_B, AIN89_EF_CONFIG_B,	9774, 9776, 9778,
AIN90_EF_CONFIG_B, AIN91_EF_CONFIG_B, AIN92_EF_CONFIG_B,	9780, 9782, 9784,
AIN93_EF_CONFIG_B, AIN94_EF_CONFIG_B, AIN95_EF_CONFIG_B,	9786, 9788, 9790,
AIN96_EF_CONFIG_B, AIN97_EF_CONFIG_B, AIN98_EF_CONFIG_B,	9792, 9794, 9796,
AIN99_EF_CONFIG_B, AIN100_EF_CONFIG_B, AIN101_EF_CONFIG_B,	9798, 9800, 9802,
AIN102_EF_CONFIG_B, AIN103_EF_CONFIG_B, AIN104_EF_CONFIG_B,	9804, 9806, 9808,
AIN105_EF_CONFIG_B, AIN106_EF_CONFIG_B, AIN107_EF_CONFIG_B,	9810, 9812, 9814,
AIN108_EF_CONFIG_B, AIN109_EF_CONFIG_B, AIN110_EF_CONFIG_B,	9816, 9818, 9820,
AIN111_EF_CONFIG_B, AIN112_EF_CONFIG_B, AIN113_EF_CONFIG_B,	9822, 9824, 9826,
AIN114_EF_CONFIG_B, AIN115_EF_CONFIG_B, AIN116_EF_CONFIG_B,	9828, 9830, 9832,
AIN117_EF_CONFIG_B, AIN118_EF_CONFIG_B, AIN119_EF_CONFIG_B,	9834, 9836, 9838,
AIN120_EF_CONFIG_B, AIN121_EF_CONFIG_B, AIN122_EF_CONFIG_B,	9840, 9842, 9844,
AIN123_EF_CONFIG_B, AIN124_EF_CONFIG_B, AIN125_EF_CONFIG_B,	9846, 9848, 9850,
AIN126_EF_CONFIG_B, AIN127_EF_CONFIG_B, AIN128_EF_CONFIG_B,	9852, 9854, 9856,
AIN129_EF_CONFIG_B, AIN130_EF_CONFIG_B, AIN131_EF_CONFIG_B,	9858, 9860, 9862,
AIN132_EF_CONFIG_B, AIN133_EF_CONFIG_B, AIN134_EF_CONFIG_B,	9864, 9866, 9868,
AIN135_EF_CONFIG_B, AIN136_EF_CONFIG_B, AIN137_EF_CONFIG_B,	9870, 9872, 9874,
AIN138_EF_CONFIG_B, AIN139_EF_CONFIG_B, AIN140_EF_CONFIG_B,	9876, 9878, 9880,
AIN141_EF_CONFIG_B, AIN142_EF_CONFIG_B, AIN143_EF_CONFIG_B,	9882, 9884, 9886,
AIN144_EF_CONFIG_B, AIN145_EF_CONFIG_B, AIN146_EF_CONFIG_B,	9888, 9890, 9892,
AIN147_EF_CONFIG_B, AIN148_EF_CONFIG_B	9894, 9896

AIN#(0:148)_EF_CONFIG_C
- Starting Address: 9900

Function dependent on selected feature index.

- Data type: UINT32 (type index = 1)
- Readable and writable
- Default value: 0

- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0030

Expanded Names	Addresses
AIN0_EF_CONFIG_C, AIN1_EF_CONFIG_C, AIN2_EF_CONFIG_C, AIN3_EF_CONFIG_C, AIN4_EF_CONFIG_C, AIN5_EF_CONFIG_C, AIN6_EF_CONFIG_C, AIN7_EF_CONFIG_C, AIN8_EF_CONFIG_C, AIN9_EF_CONFIG_C, AIN10_EF_CONFIG_C, AIN11_EF_CONFIG_C, AIN12_EF_CONFIG_C, AIN13_EF_CONFIG_C, AIN14_EF_CONFIG_C, AIN15_EF_CONFIG_C, AIN16_EF_CONFIG_C, AIN17_EF_CONFIG_C, AIN18_EF_CONFIG_C, AIN19_EF_CONFIG_C, AIN20_EF_CONFIG_C, AIN21_EF_CONFIG_C, AIN22_EF_CONFIG_C, AIN23_EF_CONFIG_C, AIN24_EF_CONFIG_C, AIN25_EF_CONFIG_C, AIN26_EF_CONFIG_C, AIN27_EF_CONFIG_C, AIN28_EF_CONFIG_C, AIN29_EF_CONFIG_C, AIN30_EF_CONFIG_C, AIN31_EF_CONFIG_C, AIN32_EF_CONFIG_C, AIN33_EF_CONFIG_C, AIN34_EF_CONFIG_C, AIN35_EF_CONFIG_C, AIN36_EF_CONFIG_C, AIN37_EF_CONFIG_C, AIN38_EF_CONFIG_C, AIN39_EF_CONFIG_C, AIN40_EF_CONFIG_C, AIN41_EF_CONFIG_C, AIN42_EF_CONFIG_C, AIN43_EF_CONFIG_C, AIN44_EF_CONFIG_C, AIN45_EF_CONFIG_C, AIN46_EF_CONFIG_C, AIN47_EF_CONFIG_C, AIN48_EF_CONFIG_C, AIN49_EF_CONFIG_C, AIN50_EF_CONFIG_C, AIN51_EF_CONFIG_C, AIN52_EF_CONFIG_C, AIN53_EF_CONFIG_C, AIN54_EF_CONFIG_C, AIN55_EF_CONFIG_C, AIN56_EF_CONFIG_C, AIN57_EF_CONFIG_C, AIN58_EF_CONFIG_C, AIN59_EF_CONFIG_C, AIN60_EF_CONFIG_C, AIN61_EF_CONFIG_C, AIN62_EF_CONFIG_C, AIN63_EF_CONFIG_C, AIN64_EF_CONFIG_C, AIN65_EF_CONFIG_C, AIN66_EF_CONFIG_C, AIN67_EF_CONFIG_C, AIN68_EF_CONFIG_C, AIN69_EF_CONFIG_C, AIN70_EF_CONFIG_C, AIN71_EF_CONFIG_C, AIN72_EF_CONFIG_C, AIN73_EF_CONFIG_C, AIN74_EF_CONFIG_C, AIN75_EF_CONFIG_C, AIN76_EF_CONFIG_C, AIN77_EF_CONFIG_C, AIN78_EF_CONFIG_C, AIN79_EF_CONFIG_C, AIN80_EF_CONFIG_C, AIN81_EF_CONFIG_C, AIN82_EF_CONFIG_C, AIN83_EF_CONFIG_C, AIN84_EF_CONFIG_C, AIN85_EF_CONFIG_C, AIN86_EF_CONFIG_C, AIN87_EF_CONFIG_C, AIN88_EF_CONFIG_C, AIN89_EF_CONFIG_C, AIN90_EF_CONFIG_C, AIN91_EF_CONFIG_C, AIN92_EF_CONFIG_C, AIN93_EF_CONFIG_C, AIN94_EF_CONFIG_C, AIN95_EF_CONFIG_C, AIN96_EF_CONFIG_C, AIN97_EF_CONFIG_C, AIN98_EF_CONFIG_C, AIN99_EF_CONFIG_C, AIN100_EF_CONFIG_C, AIN101_EF_CONFIG_C, AIN102_EF_CONFIG_C, AIN103_EF_CONFIG_C, AIN104_EF_CONFIG_C, AIN105_EF_CONFIG_C, AIN106_EF_CONFIG_C, AIN107_EF_CONFIG_C, AIN108_EF_CONFIG_C, AIN109_EF_CONFIG_C, AIN110_EF_CONFIG_C, AIN111_EF_CONFIG_C, AIN112_EF_CONFIG_C, AIN113_EF_CONFIG_C, AIN114_EF_CONFIG_C, AIN115_EF_CONFIG_C, AIN116_EF_CONFIG_C, AIN117_EF_CONFIG_C, AIN118_EF_CONFIG_C, AIN119_EF_CONFIG_C, AIN120_EF_CONFIG_C, AIN121_EF_CONFIG_C, AIN122_EF_CONFIG_C, AIN123_EF_CONFIG_C, AIN124_EF_CONFIG_C, AIN125_EF_CONFIG_C, AIN126_EF_CONFIG_C, AIN127_EF_CONFIG_C, AIN128_EF_CONFIG_C, AIN129_EF_CONFIG_C, AIN130_EF_CONFIG_C, AIN131_EF_CONFIG_C, AIN132_EF_CONFIG_C, AIN133_EF_CONFIG_C, AIN134_EF_CONFIG_C, AIN135_EF_CONFIG_C, AIN136_EF_CONFIG_C, AIN137_EF_CONFIG_C, AIN138_EF_CONFIG_C, AIN139_EF_CONFIG_C, AIN140_EF_CONFIG_C, AIN141_EF_CONFIG_C, AIN142_EF_CONFIG_C, AIN143_EF_CONFIG_C, AIN144_EF_CONFIG_C, AIN145_EF_CONFIG_C, AIN146_EF_CONFIG_C, AIN147_EF_CONFIG_C, AIN148_EF_CONFIG_C	9900, 9902, 9904, 9906, 9908, 9910, 9912, 9914, 9916, 9918, 9920, 9922, 9924, 9926, 9928, 9930, 9932, 9934, 9936, 9938, 9940, 9942, 9944, 9946, 9948, 9950, 9952, 9954, 9956, 9958, 9960, 9962, 9964, 9966, 9968, 9970, 9972, 9974, 9976, 9978, 9980, 9982, 9984, 9986, 9988, 9990, 9992, 9994, 9996, 9998, 10000, 10002, 10004, 10006, 10008, 10010, 10012, 10014, 10016, 10018, 10020, 10022, 10024, 10026, 10028, 10030, 10032, 10034, 10036, 10038, 10040, 10042, 10044, 10046, 10048, 10050, 10052, 10054, 10056, 10058, 10060, 10062, 10064, 10066, 10068, 10070, 10072, 10074, 10076, 10078, 10080, 10082, 10084, 10086, 10088, 10090, 10092, 10094, 10096, 10098, 10100, 10102, 10104, 10106, 10108, 10110, 10112, 10114, 10116, 10118, 10120, 10122, 10124, 10126, 10128, 10130, 10132, 10134, 10136, 10138, 10140, 10142, 10144, 10146, 10148, 10150, 10152, 10154, 10156, 10158, 10160, 10162, 10164, 10166, 10168, 10170, 10172, 10174, 10176, 10178, 10180, 10182, 10184, 10186, 10188, 10190, 10192, 10194, 10196

AIN#(0:148)_EF_CONFIG_D
- Starting Address: 10200

Function dependent on selected feature index.

- Data type: FLOAT32 (type index = 3)
- Readable and writable
- Default value: 0
- T8:
    - Minimum firmware version: 0.0123
- T7:
    - Minimum firmware version: 1.0030

Expanded Names	Addresses
AIN0_EF_CONFIG_D, AIN1_EF_CONFIG_D, AIN2_EF_CONFIG_D,	10200, 10202, 10204,
AIN3_EF_CONFIG_D, AIN4_EF_CONFIG_D, AIN5_EF_CONFIG_D,	10206, 10208, 10210,
AIN6_EF_CONFIG_D, AIN7_EF_CONFIG_D, AIN8_EF_CONFIG_D,	10212, 10214, 10216,
AIN9_EF_CONFIG_D, AIN10_EF_CONFIG_D, AIN11_EF_CONFIG_D,	10218, 10220, 10222,
AIN12_EF_CONFIG_D, AIN13_EF_CONFIG_D, AIN14_EF_CONFIG_D,	10224, 10226, 10228,
AIN15_EF_CONFIG_D, AIN16_EF_CONFIG_D, AIN17_EF_CONFIG_D,	10230, 10232, 10234,
AIN18_EF_CONFIG_D, AIN19_EF_CONFIG_D, AIN20_EF_CONFIG_D,	10236, 10238, 10240,
AIN21_EF_CONFIG_D, AIN22_EF_CONFIG_D, AIN23_EF_CONFIG_D,	10242, 10244, 10246,
AIN24_EF_CONFIG_D, AIN25_EF_CONFIG_D, AIN26_EF_CONFIG_D,	10248, 10250, 10252,
AIN27_EF_CONFIG_D, AIN28_EF_CONFIG_D, AIN29_EF_CONFIG_D,	10254, 10256, 10258,
AIN30_EF_CONFIG_D, AIN31_EF_CONFIG_D, AIN32_EF_CONFIG_D,	10260, 10262, 10264,
AIN33_EF_CONFIG_D, AIN34_EF_CONFIG_D, AIN35_EF_CONFIG_D,	10266, 10268, 10270,
AIN36_EF_CONFIG_D, AIN37_EF_CONFIG_D, AIN38_EF_CONFIG_D,	10272, 10274, 10276,
AIN39_EF_CONFIG_D, AIN40_EF_CONFIG_D, AIN41_EF_CONFIG_D,	10278, 10280, 10282,
AIN42_EF_CONFIG_D, AIN43_EF_CONFIG_D, AIN44_EF_CONFIG_D,	10284, 10286, 10288,
AIN45_EF_CONFIG_D, AIN46_EF_CONFIG_D, AIN47_EF_CONFIG_D,	10290, 10292, 10294,
AIN48_EF_CONFIG_D, AIN49_EF_CONFIG_D, AIN50_EF_CONFIG_D,	10296, 10298, 10300,
AIN51_EF_CONFIG_D, AIN52_EF_CONFIG_D, AIN53_EF_CONFIG_D,	10302, 10304, 10306,
AIN54_EF_CONFIG_D, AIN55_EF_CONFIG_D, AIN56_EF_CONFIG_D,	10308, 10310, 10312,
AIN57_EF_CONFIG_D, AIN58_EF_CONFIG_D, AIN59_EF_CONFIG_D,	10314, 10316, 10318,
AIN60_EF_CONFIG_D, AIN61_EF_CONFIG_D, AIN62_EF_CONFIG_D,	10320, 10322, 10324,
AIN63_EF_CONFIG_D, AIN64_EF_CONFIG_D, AIN65_EF_CONFIG_D,	10326, 10328, 10330,
AIN66_EF_CONFIG_D, AIN67_EF_CONFIG_D, AIN68_EF_CONFIG_D,	10332, 10334, 10336,
AIN69_EF_CONFIG_D, AIN70_EF_CONFIG_D, AIN71_EF_CONFIG_D,	10338, 10340, 10342,
AIN72_EF_CONFIG_D, AIN73_EF_CONFIG_D, AIN74_EF_CONFIG_D,	10344, 10346, 10348,
AIN75_EF_CONFIG_D, AIN76_EF_CONFIG_D, AIN77_EF_CONFIG_D,	10350, 10352, 10354,
AIN78_EF_CONFIG_D, AIN79_EF_CONFIG_D, AIN80_EF_CONFIG_D,	10356, 10358, 10360,
AIN81_EF_CONFIG_D, AIN82_EF_CONFIG_D, AIN83_EF_CONFIG_D,	10362, 10364, 10366,
AIN84_EF_CONFIG_D, AIN85_EF_CONFIG_D, AIN86_EF_CONFIG_D,	10368, 10370, 10372,
AIN87_EF_CONFIG_D, AIN88_EF_CONFIG_D, AIN89_EF_CONFIG_D,	10374, 10376, 10378,
AIN90_EF_CONFIG_D, AIN91_EF_CONFIG_D, AIN92_EF_CONFIG_D,	10380, 10382, 10384,
AIN93_EF_CONFIG_D, AIN94_EF_CONFIG_D, AIN95_EF_CONFIG_D,	10386, 10388, 10390,
AIN96_EF_CONFIG_D, AIN97_EF_CONFIG_D, AIN98_EF_CONFIG_D,	10392, 10394, 10396,
AIN99_EF_CONFIG_D, AIN100_EF_CONFIG_D, AIN101_EF_CONFIG_D,	10398, 10400, 10402,
AIN102_EF_CONFIG_D, AIN103_EF_CONFIG_D, AIN104_EF_CONFIG_D,	10404, 10406, 10408,
AIN105_EF_CONFIG_D, AIN106_EF_CONFIG_D, AIN107_EF_CONFIG_D,	10410, 10412, 10414,
AIN108_EF_CONFIG_D, AIN109_EF_CONFIG_D, AIN110_EF_CONFIG_D,	10416, 10418, 10420,
AIN111_EF_CONFIG_D, AIN112_EF_CONFIG_D, AIN113_EF_CONFIG_D,	10422, 10424, 10426,
AIN114_EF_CONFIG_D, AIN115_EF_CONFIG_D, AIN116_EF_CONFIG_D,	10428, 10430, 10432,
AIN117_EF_CONFIG_D, AIN118_EF_CONFIG_D, AIN119_EF_CONFIG_D,	10434, 10436, 10438,
AIN120_EF_CONFIG_D, AIN121_EF_CONFIG_D, AIN122_EF_CONFIG_D,	10440, 10442, 10444,
AIN123_EF_CONFIG_D, AIN124_EF_CONFIG_D, AIN125_EF_CONFIG_D,	10446, 10448, 10450,
AIN126_EF_CONFIG_D, AIN127_EF_CONFIG_D, AIN128_EF_CONFIG_D,	10452, 10454, 10456,
AIN129_EF_CONFIG_D, AIN130_EF_CONFIG_D, AIN131_EF_CONFIG_D,	10458, 10460, 10462,
AIN132_EF_CONFIG_D, AIN133_EF_CONFIG_D, AIN134_EF_CONFIG_D,	10464, 10466, 10468,
AIN135_EF_CONFIG_D, AIN136_EF_CONFIG_D, AIN137_EF_CONFIG_D,	10470, 10472, 10474,
AIN138_EF_CONFIG_D, AIN139_EF_CONFIG_D, AIN140_EF_CONFIG_D,	10476, 10478, 10480,
AIN141_EF_CONFIG_D, AIN142_EF_CONFIG_D, AIN143_EF_CONFIG_D,	10482, 10484, 10486,
AIN144_EF_CONFIG_D, AIN145_EF_CONFIG_D, AIN146_EF_CONFIG_D,	10488, 10490, 10492,
AIN147_EF_CONFIG_D, AIN148_EF_CONFIG_D	10494, 10496

AIN#(0:148)_EF_CONFIG_E

- Starting Address: 10500

  Function dependent on selected feature index.

  - Data type: FLOAT32 (type index = 3)
  - Readable and writable
  - Default value: 0
  - T8:
    - Minimum firmware version: 0.0123
  - T7:
    - Minimum firmware version: 1.0030

Expanded Names	Addresses

AIN0_EF_CONFIG_E, AIN1_EF_CONFIG_E, AIN2_EF_CONFIG_E,	10500, 10502, 10504,
AIN3_EF_CONFIG_E, AIN4_EF_CONFIG_E, AIN5_EF_CONFIG_E,	10506, 10508, 10510,
AIN6_EF_CONFIG_E, AIN7_EF_CONFIG_E, AIN8_EF_CONFIG_E,	10512, 10514, 10516,
AIN9_EF_CONFIG_E, AIN10_EF_CONFIG_E, AIN11_EF_CONFIG_E,	10518, 10520, 10522,
AIN12_EF_CONFIG_E, AIN13_EF_CONFIG_E, AIN14_EF_CONFIG_E,	10524, 10526, 10528,
AIN15_EF_CONFIG_E, AIN16_EF_CONFIG_E, AIN17_EF_CONFIG_E,	10530, 10532, 10534,
AIN18_EF_CONFIG_E, AIN19_EF_CONFIG_E, AIN20_EF_CONFIG_E,	10536, 10538, 10540,
AIN21_EF_CONFIG_E, AIN22_EF_CONFIG_E, AIN23_EF_CONFIG_E,	10542, 10544, 10546,
AIN24_EF_CONFIG_E, AIN25_EF_CONFIG_E, AIN26_EF_CONFIG_E,	10548, 10550, 10552,
AIN27_EF_CONFIG_E, AIN28_EF_CONFIG_E, AIN29_EF_CONFIG_E,	10554, 10556, 10558,
AIN30_EF_CONFIG_E, AIN31_EF_CONFIG_E, AIN32_EF_CONFIG_E,	10560, 10562, 10564,
AIN33_EF_CONFIG_E, AIN34_EF_CONFIG_E, AIN35_EF_CONFIG_E,	10566, 10568, 10570,
AIN36_EF_CONFIG_E, AIN37_EF_CONFIG_E, AIN38_EF_CONFIG_E,	10572, 10574, 10576,
AIN39_EF_CONFIG_E, AIN40_EF_CONFIG_E, AIN41_EF_CONFIG_E,	10578, 10580, 10582,
AIN42_EF_CONFIG_E, AIN43_EF_CONFIG_E, AIN44_EF_CONFIG_E,	10584, 10586, 10588,
AIN45_EF_CONFIG_E, AIN46_EF_CONFIG_E, AIN47_EF_CONFIG_E,	10590, 10592, 10594,
AIN48_EF_CONFIG_E, AIN49_EF_CONFIG_E, AIN50_EF_CONFIG_E,	10596, 10598, 10600,
AIN51_EF_CONFIG_E, AIN52_EF_CONFIG_E, AIN53_EF_CONFIG_E,	10602, 10604, 10606,
AIN54_EF_CONFIG_E, AIN55_EF_CONFIG_E, AIN56_EF_CONFIG_E,	10608, 10610, 10612,
AIN57_EF_CONFIG_E, AIN58_EF_CONFIG_E, AIN59_EF_CONFIG_E,	10614, 10616, 10618,
AIN60_EF_CONFIG_E, AIN61_EF_CONFIG_E, AIN62_EF_CONFIG_E,	10620, 10622, 10624,
AIN63_EF_CONFIG_E, AIN64_EF_CONFIG_E, AIN65_EF_CONFIG_E,	10626, 10628, 10630,
AIN66_EF_CONFIG_E, AIN67_EF_CONFIG_E, AIN68_EF_CONFIG_E,	10632, 10634, 10636,
AIN69_EF_CONFIG_E, AIN70_EF_CONFIG_E, AIN71_EF_CONFIG_E,	10638, 10640, 10642,
AIN72_EF_CONFIG_E, AIN73_EF_CONFIG_E, AIN74_EF_CONFIG_E,	10644, 10646, 10648,
AIN75_EF_CONFIG_E, AIN76_EF_CONFIG_E, AIN77_EF_CONFIG_E,	10650, 10652, 10654,
AIN78_EF_CONFIG_E, AIN79_EF_CONFIG_E, AIN80_EF_CONFIG_E,	10656, 10658, 10660,
AIN81_EF_CONFIG_E, AIN82_EF_CONFIG_E, AIN83_EF_CONFIG_E,	10662, 10664, 10666,
AIN84_EF_CONFIG_E, AIN85_EF_CONFIG_E, AIN86_EF_CONFIG_E,	10668, 10670, 10672,
AIN87_EF_CONFIG_E, AIN88_EF_CONFIG_E, AIN89_EF_CONFIG_E,	10674, 10676, 10678,
AIN90_EF_CONFIG_E, AIN91_EF_CONFIG_E, AIN92_EF_CONFIG_E,	10680, 10682, 10684,
AIN93_EF_CONFIG_E, AIN94_EF_CONFIG_E, AIN95_EF_CONFIG_E,	10686, 10688, 10690,
AIN96_EF_CONFIG_E, AIN97_EF_CONFIG_E, AIN98_EF_CONFIG_E,	10692, 10694, 10696,
AIN99_EF_CONFIG_E, AIN100_EF_CONFIG_E, AIN101_EF_CONFIG_E,	10698, 10700, 10702,
AIN102_EF_CONFIG_E, AIN103_EF_CONFIG_E, AIN104_EF_CONFIG_E,	10704, 10706, 10708,
AIN105_EF_CONFIG_E, AIN106_EF_CONFIG_E, AIN107_EF_CONFIG_E,	10710, 10712, 10714,
AIN108_EF_CONFIG_E, AIN109_EF_CONFIG_E, AIN110_EF_CONFIG_E,	10716, 10718, 10720,
AIN111_EF_CONFIG_E, AIN112_EF_CONFIG_E, AIN113_EF_CONFIG_E,	10722, 10724, 10726,
AIN114_EF_CONFIG_E, AIN115_EF_CONFIG_E, AIN116_EF_CONFIG_E,	10728, 10730, 10732,
AIN117_EF_CONFIG_E, AIN118_EF_CONFIG_E, AIN119_EF_CONFIG_E,	10734, 10736, 10738,
AIN120_EF_CONFIG_E, AIN121_EF_CONFIG_E, AIN122_EF_CONFIG_E,	10740, 10742, 10744,
AIN123_EF_CONFIG_E, AIN124_EF_CONFIG_E, AIN125_EF_CONFIG_E,	10746, 10748, 10750,
AIN126_EF_CONFIG_E, AIN127_EF_CONFIG_E, AIN128_EF_CONFIG_E,	10752, 10754, 10756,
AIN129_EF_CONFIG_E, AIN130_EF_CONFIG_E, AIN131_EF_CONFIG_E,	10758, 10760, 10762,
AIN132_EF_CONFIG_E, AIN133_EF_CONFIG_E, AIN134_EF_CONFIG_E,	10764, 10766, 10768,
AIN135_EF_CONFIG_E, AIN136_EF_CONFIG_E, AIN137_EF_CONFIG_E,	10770, 10772, 10774,
AIN138_EF_CONFIG_E, AIN139_EF_CONFIG_E, AIN140_EF_CONFIG_E,	10776, 10778, 10780,
AIN141_EF_CONFIG_E, AIN142_EF_CONFIG_E, AIN143_EF_CONFIG_E,	10782, 10784, 10786,
AIN144_EF_CONFIG_E, AIN145_EF_CONFIG_E, AIN146_EF_CONFIG_E,	10788, 10790, 10792,
AIN147_EF_CONFIG_E, AIN148_EF_CONFIG_E	10794, 10796

AIN#(0:148)_EF_CONFIG_F
- Starting Address: 10800

Function dependent on selected feature index.

- Data type: FLOAT32 (type index = 3)
- Readable and writable
- Default value: 0
- T8:
  - Minimum firmware version: 0.0123
- T7:

- Minimum firmware version: 1.0030

Expanded Names	Addresses
AIN0_EF_CONFIG_F, AIN1_EF_CONFIG_F, AIN2_EF_CONFIG_F,	10800, 10802, 10804,
AIN3_EF_CONFIG_F, AIN4_EF_CONFIG_F, AIN5_EF_CONFIG_F,	10806, 10808, 10810,
AIN6_EF_CONFIG_F, AIN7_EF_CONFIG_F, AIN8_EF_CONFIG_F,	10812, 10814, 10816,
AIN9_EF_CONFIG_F, AIN10_EF_CONFIG_F, AIN11_EF_CONFIG_F,	10818, 10820, 10822,
AIN12_EF_CONFIG_F, AIN13_EF_CONFIG_F, AIN14_EF_CONFIG_F,	10824, 10826, 10828,
AIN15_EF_CONFIG_F, AIN16_EF_CONFIG_F, AIN17_EF_CONFIG_F,	10830, 10832, 10834,
AIN18_EF_CONFIG_F, AIN19_EF_CONFIG_F, AIN20_EF_CONFIG_F,	10836, 10838, 10840,
AIN21_EF_CONFIG_F, AIN22_EF_CONFIG_F, AIN23_EF_CONFIG_F,	10842, 10844, 10846,
AIN24_EF_CONFIG_F, AIN25_EF_CONFIG_F, AIN26_EF_CONFIG_F,	10848, 10850, 10852,
AIN27_EF_CONFIG_F, AIN28_EF_CONFIG_F, AIN29_EF_CONFIG_F,	10854, 10856, 10858,
AIN30_EF_CONFIG_F, AIN31_EF_CONFIG_F, AIN32_EF_CONFIG_F,	10860, 10862, 10864,
AIN33_EF_CONFIG_F, AIN34_EF_CONFIG_F, AIN35_EF_CONFIG_F,	10866, 10868, 10870,
AIN36_EF_CONFIG_F, AIN37_EF_CONFIG_F, AIN38_EF_CONFIG_F,	10872, 10874, 10876,
AIN39_EF_CONFIG_F, AIN40_EF_CONFIG_F, AIN41_EF_CONFIG_F,	10878, 10880, 10882,
AIN42_EF_CONFIG_F, AIN43_EF_CONFIG_F, AIN44_EF_CONFIG_F,	10884, 10886, 10888,
AIN45_EF_CONFIG_F, AIN46_EF_CONFIG_F, AIN47_EF_CONFIG_F,	10890, 10892, 10894,
AIN48_EF_CONFIG_F, AIN49_EF_CONFIG_F, AIN50_EF_CONFIG_F,	10896, 10898, 10900,
AIN51_EF_CONFIG_F, AIN52_EF_CONFIG_F, AIN53_EF_CONFIG_F,	10902, 10904, 10906,
AIN54_EF_CONFIG_F, AIN55_EF_CONFIG_F, AIN56_EF_CONFIG_F,	10908, 10910, 10912,
AIN57_EF_CONFIG_F, AIN58_EF_CONFIG_F, AIN59_EF_CONFIG_F,	10914, 10916, 10918,
AIN60_EF_CONFIG_F, AIN61_EF_CONFIG_F, AIN62_EF_CONFIG_F,	10920, 10922, 10924,
AIN63_EF_CONFIG_F, AIN64_EF_CONFIG_F, AIN65_EF_CONFIG_F,	10926, 10928, 10930,
AIN66_EF_CONFIG_F, AIN67_EF_CONFIG_F, AIN68_EF_CONFIG_F,	10932, 10934, 10936,
AIN69_EF_CONFIG_F, AIN70_EF_CONFIG_F, AIN71_EF_CONFIG_F,	10938, 10940, 10942,
AIN72_EF_CONFIG_F, AIN73_EF_CONFIG_F, AIN74_EF_CONFIG_F,	10944, 10946, 10948,
AIN75_EF_CONFIG_F, AIN76_EF_CONFIG_F, AIN77_EF_CONFIG_F,	10950, 10952, 10954,
AIN78_EF_CONFIG_F, AIN79_EF_CONFIG_F, AIN80_EF_CONFIG_F,	10956, 10958, 10960,
AIN81_EF_CONFIG_F, AIN82_EF_CONFIG_F, AIN83_EF_CONFIG_F,	10962, 10964, 10966,
AIN84_EF_CONFIG_F, AIN85_EF_CONFIG_F, AIN86_EF_CONFIG_F,	10968, 10970, 10972,
AIN87_EF_CONFIG_F, AIN88_EF_CONFIG_F, AIN89_EF_CONFIG_F,	10974, 10976, 10978,
AIN90_EF_CONFIG_F, AIN91_EF_CONFIG_F, AIN92_EF_CONFIG_F,	10980, 10982, 10984,
AIN93_EF_CONFIG_F, AIN94_EF_CONFIG_F, AIN95_EF_CONFIG_F,	10986, 10988, 10990,
AIN96_EF_CONFIG_F, AIN97_EF_CONFIG_F, AIN98_EF_CONFIG_F,	10992, 10994, 10996,
AIN99_EF_CONFIG_F, AIN100_EF_CONFIG_F, AIN101_EF_CONFIG_F,	10998, 11000, 11002,
AIN102_EF_CONFIG_F, AIN103_EF_CONFIG_F, AIN104_EF_CONFIG_F,	11004, 11006, 11008,
AIN105_EF_CONFIG_F, AIN106_EF_CONFIG_F, AIN107_EF_CONFIG_F,	11010, 11012, 11014,
AIN108_EF_CONFIG_F, AIN109_EF_CONFIG_F, AIN110_EF_CONFIG_F,	11016, 11018, 11020,
AIN111_EF_CONFIG_F, AIN112_EF_CONFIG_F, AIN113_EF_CONFIG_F,	11022, 11024, 11026,
AIN114_EF_CONFIG_F, AIN115_EF_CONFIG_F, AIN116_EF_CONFIG_F,	11028, 11030, 11032,
AIN117_EF_CONFIG_F, AIN118_EF_CONFIG_F, AIN119_EF_CONFIG_F,	11034, 11036, 11038,
AIN120_EF_CONFIG_F, AIN121_EF_CONFIG_F, AIN122_EF_CONFIG_F,	11040, 11042, 11044,
AIN123_EF_CONFIG_F, AIN124_EF_CONFIG_F, AIN125_EF_CONFIG_F,	11046, 11048, 11050,
AIN126_EF_CONFIG_F, AIN127_EF_CONFIG_F, AIN128_EF_CONFIG_F,	11052, 11054, 11056,
AIN129_EF_CONFIG_F, AIN130_EF_CONFIG_F, AIN131_EF_CONFIG_F,	11058, 11060, 11062,
AIN132_EF_CONFIG_F, AIN133_EF_CONFIG_F, AIN134_EF_CONFIG_F,	11064, 11066, 11068,
AIN135_EF_CONFIG_F, AIN136_EF_CONFIG_F, AIN137_EF_CONFIG_F,	11070, 11072, 11074,
AIN138_EF_CONFIG_F, AIN139_EF_CONFIG_F, AIN140_EF_CONFIG_F,	11076, 11078, 11080,
AIN141_EF_CONFIG_F, AIN142_EF_CONFIG_F, AIN143_EF_CONFIG_F,	11082, 11084, 11086,
AIN144_EF_CONFIG_F, AIN145_EF_CONFIG_F, AIN146_EF_CONFIG_F,	11088, 11090, 11092,
AIN147_EF_CONFIG_F, AIN148_EF_CONFIG_F	11094, 11096

AIN#(0:148)_EF_CONFIG_G
- Starting Address: 11100

Function dependent on selected feature index.

- Data type: FLOAT32 (type index = 3)
- Readable and writable
- Default value: 0

- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0030

Expanded Names	Addresses
AIN0_EF_CONFIG_G, AIN1_EF_CONFIG_G, AIN2_EF_CONFIG_G,	11100, 11102, 11104,
AIN3_EF_CONFIG_G, AIN4_EF_CONFIG_G, AIN5_EF_CONFIG_G,	11106, 11108, 11110,
AIN6_EF_CONFIG_G, AIN7_EF_CONFIG_G, AIN8_EF_CONFIG_G,	11112, 11114, 11116,
AIN9_EF_CONFIG_G, AIN10_EF_CONFIG_G, AIN11_EF_CONFIG_G,	11118, 11120, 11122,
AIN12_EF_CONFIG_G, AIN13_EF_CONFIG_G, AIN14_EF_CONFIG_G,	11124, 11126, 11128,
AIN15_EF_CONFIG_G, AIN16_EF_CONFIG_G, AIN17_EF_CONFIG_G,	11130, 11132, 11134,
AIN18_EF_CONFIG_G, AIN19_EF_CONFIG_G, AIN20_EF_CONFIG_G,	11136, 11138, 11140,
AIN21_EF_CONFIG_G, AIN22_EF_CONFIG_G, AIN23_EF_CONFIG_G,	11142, 11144, 11146,
AIN24_EF_CONFIG_G, AIN25_EF_CONFIG_G, AIN26_EF_CONFIG_G,	11148, 11150, 11152,
AIN27_EF_CONFIG_G, AIN28_EF_CONFIG_G, AIN29_EF_CONFIG_G,	11154, 11156, 11158,
AIN30_EF_CONFIG_G, AIN31_EF_CONFIG_G, AIN32_EF_CONFIG_G,	11160, 11162, 11164,
AIN33_EF_CONFIG_G, AIN34_EF_CONFIG_G, AIN35_EF_CONFIG_G,	11166, 11168, 11170,
AIN36_EF_CONFIG_G, AIN37_EF_CONFIG_G, AIN38_EF_CONFIG_G,	11172, 11174, 11176,
AIN39_EF_CONFIG_G, AIN40_EF_CONFIG_G, AIN41_EF_CONFIG_G,	11178, 11180, 11182,
AIN42_EF_CONFIG_G, AIN43_EF_CONFIG_G, AIN44_EF_CONFIG_G,	11184, 11186, 11188,
AIN45_EF_CONFIG_G, AIN46_EF_CONFIG_G, AIN47_EF_CONFIG_G,	11190, 11192, 11194,
AIN48_EF_CONFIG_G, AIN49_EF_CONFIG_G, AIN50_EF_CONFIG_G,	11196, 11198, 11200,
AIN51_EF_CONFIG_G, AIN52_EF_CONFIG_G, AIN53_EF_CONFIG_G,	11202, 11204, 11206,
AIN54_EF_CONFIG_G, AIN55_EF_CONFIG_G, AIN56_EF_CONFIG_G,	11208, 11210, 11212,
AIN57_EF_CONFIG_G, AIN58_EF_CONFIG_G, AIN59_EF_CONFIG_G,	11214, 11216, 11218,
AIN60_EF_CONFIG_G, AIN61_EF_CONFIG_G, AIN62_EF_CONFIG_G,	11220, 11222, 11224,
AIN63_EF_CONFIG_G, AIN64_EF_CONFIG_G, AIN65_EF_CONFIG_G,	11226, 11228, 11230,
AIN66_EF_CONFIG_G, AIN67_EF_CONFIG_G, AIN68_EF_CONFIG_G,	11232, 11234, 11236,
AIN69_EF_CONFIG_G, AIN70_EF_CONFIG_G, AIN71_EF_CONFIG_G,	11238, 11240, 11242,
AIN72_EF_CONFIG_G, AIN73_EF_CONFIG_G, AIN74_EF_CONFIG_G,	11244, 11246, 11248,
AIN75_EF_CONFIG_G, AIN76_EF_CONFIG_G, AIN77_EF_CONFIG_G,	11250, 11252, 11254,
AIN78_EF_CONFIG_G, AIN79_EF_CONFIG_G, AIN80_EF_CONFIG_G,	11256, 11258, 11260,
AIN81_EF_CONFIG_G, AIN82_EF_CONFIG_G, AIN83_EF_CONFIG_G,	11262, 11264, 11266,
AIN84_EF_CONFIG_G, AIN85_EF_CONFIG_G, AIN86_EF_CONFIG_G,	11268, 11270, 11272,
AIN87_EF_CONFIG_G, AIN88_EF_CONFIG_G, AIN89_EF_CONFIG_G,	11274, 11276, 11278,
AIN90_EF_CONFIG_G, AIN91_EF_CONFIG_G, AIN92_EF_CONFIG_G,	11280, 11282, 11284,
AIN93_EF_CONFIG_G, AIN94_EF_CONFIG_G, AIN95_EF_CONFIG_G,	11286, 11288, 11290,
AIN96_EF_CONFIG_G, AIN97_EF_CONFIG_G, AIN98_EF_CONFIG_G,	11292, 11294, 11296,
AIN99_EF_CONFIG_G, AIN100_EF_CONFIG_G, AIN101_EF_CONFIG_G,	11298, 11300, 11302,
AIN102_EF_CONFIG_G, AIN103_EF_CONFIG_G, AIN104_EF_CONFIG_G,	11304, 11306, 11308,
AIN105_EF_CONFIG_G, AIN106_EF_CONFIG_G, AIN107_EF_CONFIG_G,	11310, 11312, 11314,
AIN108_EF_CONFIG_G, AIN109_EF_CONFIG_G, AIN110_EF_CONFIG_G,	11316, 11318, 11320,
AIN111_EF_CONFIG_G, AIN112_EF_CONFIG_G, AIN113_EF_CONFIG_G,	11322, 11324, 11326,
AIN114_EF_CONFIG_G, AIN115_EF_CONFIG_G, AIN116_EF_CONFIG_G,	11328, 11330, 11332,
AIN117_EF_CONFIG_G, AIN118_EF_CONFIG_G, AIN119_EF_CONFIG_G,	11334, 11336, 11338,
AIN120_EF_CONFIG_G, AIN121_EF_CONFIG_G, AIN122_EF_CONFIG_G,	11340, 11342, 11344,
AIN123_EF_CONFIG_G, AIN124_EF_CONFIG_G, AIN125_EF_CONFIG_G,	11346, 11348, 11350,
AIN126_EF_CONFIG_G, AIN127_EF_CONFIG_G, AIN128_EF_CONFIG_G,	11352, 11354, 11356,
AIN129_EF_CONFIG_G, AIN130_EF_CONFIG_G, AIN131_EF_CONFIG_G,	11358, 11360, 11362,
AIN132_EF_CONFIG_G, AIN133_EF_CONFIG_G, AIN134_EF_CONFIG_G,	11364, 11366, 11368,
AIN135_EF_CONFIG_G, AIN136_EF_CONFIG_G, AIN137_EF_CONFIG_G,	11370, 11372, 11374,
AIN138_EF_CONFIG_G, AIN139_EF_CONFIG_G, AIN140_EF_CONFIG_G,	11376, 11378, 11380,
AIN141_EF_CONFIG_G, AIN142_EF_CONFIG_G, AIN143_EF_CONFIG_G,	11382, 11384, 11386,
AIN144_EF_CONFIG_G, AIN145_EF_CONFIG_G, AIN146_EF_CONFIG_G,	11388, 11390, 11392,
AIN147_EF_CONFIG_G, AIN148_EF_CONFIG_G	11394, 11396

AIN#(0:148)_EF_CONFIG_H
- Starting Address: 11400

Function dependent on selected feature index.

- Data type: FLOAT32 (type index = 3)
- Readable and writable
- Default value: 0
- T8:
    - Minimum firmware version: 0.0123
- T7:
    - Minimum firmware version: 1.0030

Expanded Names	Addresses
AIN0_EF_CONFIG_H, AIN1_EF_CONFIG_H, AIN2_EF_CONFIG_H,	11400, 11402, 11404,
AIN3_EF_CONFIG_H, AIN4_EF_CONFIG_H, AIN5_EF_CONFIG_H,	11406, 11408, 11410,
AIN6_EF_CONFIG_H, AIN7_EF_CONFIG_H, AIN8_EF_CONFIG_H,	11412, 11414, 11416,
AIN9_EF_CONFIG_H, AIN10_EF_CONFIG_H, AIN11_EF_CONFIG_H,	11418, 11420, 11422,
AIN12_EF_CONFIG_H, AIN13_EF_CONFIG_H, AIN14_EF_CONFIG_H,	11424, 11426, 11428,
AIN15_EF_CONFIG_H, AIN16_EF_CONFIG_H, AIN17_EF_CONFIG_H,	11430, 11432, 11434,
AIN18_EF_CONFIG_H, AIN19_EF_CONFIG_H, AIN20_EF_CONFIG_H,	11436, 11438, 11440,
AIN21_EF_CONFIG_H, AIN22_EF_CONFIG_H, AIN23_EF_CONFIG_H,	11442, 11444, 11446,
AIN24_EF_CONFIG_H, AIN25_EF_CONFIG_H, AIN26_EF_CONFIG_H,	11448, 11450, 11452,
AIN27_EF_CONFIG_H, AIN28_EF_CONFIG_H, AIN29_EF_CONFIG_H,	11454, 11456, 11458,
AIN30_EF_CONFIG_H, AIN31_EF_CONFIG_H, AIN32_EF_CONFIG_H,	11460, 11462, 11464,
AIN33_EF_CONFIG_H, AIN34_EF_CONFIG_H, AIN35_EF_CONFIG_H,	11466, 11468, 11470,
AIN36_EF_CONFIG_H, AIN37_EF_CONFIG_H, AIN38_EF_CONFIG_H,	11472, 11474, 11476,
AIN39_EF_CONFIG_H, AIN40_EF_CONFIG_H, AIN41_EF_CONFIG_H,	11478, 11480, 11482,
AIN42_EF_CONFIG_H, AIN43_EF_CONFIG_H, AIN44_EF_CONFIG_H,	11484, 11486, 11488,
AIN45_EF_CONFIG_H, AIN46_EF_CONFIG_H, AIN47_EF_CONFIG_H,	11490, 11492, 11494,
AIN48_EF_CONFIG_H, AIN49_EF_CONFIG_H, AIN50_EF_CONFIG_H,	11496, 11498, 11500,
AIN51_EF_CONFIG_H, AIN52_EF_CONFIG_H, AIN53_EF_CONFIG_H,	11502, 11504, 11506,
AIN54_EF_CONFIG_H, AIN55_EF_CONFIG_H, AIN56_EF_CONFIG_H,	11508, 11510, 11512,
AIN57_EF_CONFIG_H, AIN58_EF_CONFIG_H, AIN59_EF_CONFIG_H,	11514, 11516, 11518,
AIN60_EF_CONFIG_H, AIN61_EF_CONFIG_H, AIN62_EF_CONFIG_H,	11520, 11522, 11524,
AIN63_EF_CONFIG_H, AIN64_EF_CONFIG_H, AIN65_EF_CONFIG_H,	11526, 11528, 11530,
AIN66_EF_CONFIG_H, AIN67_EF_CONFIG_H, AIN68_EF_CONFIG_H,	11532, 11534, 11536,
AIN69_EF_CONFIG_H, AIN70_EF_CONFIG_H, AIN71_EF_CONFIG_H,	11538, 11540, 11542,
AIN72_EF_CONFIG_H, AIN73_EF_CONFIG_H, AIN74_EF_CONFIG_H,	11544, 11546, 11548,
AIN75_EF_CONFIG_H, AIN76_EF_CONFIG_H, AIN77_EF_CONFIG_H,	11550, 11552, 11554,
AIN78_EF_CONFIG_H, AIN79_EF_CONFIG_H, AIN80_EF_CONFIG_H,	11556, 11558, 11560,
AIN81_EF_CONFIG_H, AIN82_EF_CONFIG_H, AIN83_EF_CONFIG_H,	11562, 11564, 11566,
AIN84_EF_CONFIG_H, AIN85_EF_CONFIG_H, AIN86_EF_CONFIG_H,	11568, 11570, 11572,
AIN87_EF_CONFIG_H, AIN88_EF_CONFIG_H, AIN89_EF_CONFIG_H,	11574, 11576, 11578,
AIN90_EF_CONFIG_H, AIN91_EF_CONFIG_H, AIN92_EF_CONFIG_H,	11580, 11582, 11584,
AIN93_EF_CONFIG_H, AIN94_EF_CONFIG_H, AIN95_EF_CONFIG_H,	11586, 11588, 11590,
AIN96_EF_CONFIG_H, AIN97_EF_CONFIG_H, AIN98_EF_CONFIG_H,	11592, 11594, 11596,
AIN99_EF_CONFIG_H, AIN100_EF_CONFIG_H, AIN101_EF_CONFIG_H,	11598, 11600, 11602,
AIN102_EF_CONFIG_H, AIN103_EF_CONFIG_H, AIN104_EF_CONFIG_H,	11604, 11606, 11608,
AIN105_EF_CONFIG_H, AIN106_EF_CONFIG_H, AIN107_EF_CONFIG_H,	11610, 11612, 11614,
AIN108_EF_CONFIG_H, AIN109_EF_CONFIG_H, AIN110_EF_CONFIG_H,	11616, 11618, 11620,
AIN111_EF_CONFIG_H, AIN112_EF_CONFIG_H, AIN113_EF_CONFIG_H,	11622, 11624, 11626,
AIN114_EF_CONFIG_H, AIN115_EF_CONFIG_H, AIN116_EF_CONFIG_H,	11628, 11630, 11632,
AIN117_EF_CONFIG_H, AIN118_EF_CONFIG_H, AIN119_EF_CONFIG_H,	11634, 11636, 11638,
AIN120_EF_CONFIG_H, AIN121_EF_CONFIG_H, AIN122_EF_CONFIG_H,	11640, 11642, 11644,
AIN123_EF_CONFIG_H, AIN124_EF_CONFIG_H, AIN125_EF_CONFIG_H,	11646, 11648, 11650,
AIN126_EF_CONFIG_H, AIN127_EF_CONFIG_H, AIN128_EF_CONFIG_H,	11652, 11654, 11656,
AIN129_EF_CONFIG_H, AIN130_EF_CONFIG_H, AIN131_EF_CONFIG_H,	11658, 11660, 11662,
AIN132_EF_CONFIG_H, AIN133_EF_CONFIG_H, AIN134_EF_CONFIG_H,	11664, 11666, 11668,
AIN135_EF_CONFIG_H, AIN136_EF_CONFIG_H, AIN137_EF_CONFIG_H,	11670, 11672, 11674,
AIN138_EF_CONFIG_H, AIN139_EF_CONFIG_H, AIN140_EF_CONFIG_H,	11676, 11678, 11680,
AIN141_EF_CONFIG_H, AIN142_EF_CONFIG_H, AIN143_EF_CONFIG_H,	11682, 11684, 11686,
AIN144_EF_CONFIG_H, AIN145_EF_CONFIG_H, AIN146_EF_CONFIG_H,	11688, 11690, 11692,
AIN147_EF_CONFIG_H, AIN148_EF_CONFIG_H	11694, 11696

AIN#(0:148)_EF_CONFIG_I

- Starting Address: 11700

  Function dependent on selected feature index.

  - Data type: FLOAT32 (type index = 3)
  - Readable and writable
  - Default value: 0
  - T8:
    - Minimum firmware version: 0.0123
  - T7:
    - Minimum firmware version: 1.0030

Expanded Names	Addresses

AIN0_EF_CONFIG_I, AIN1_EF_CONFIG_I, AIN2_EF_CONFIG_I,	11700, 11702, 11704,
AIN3_EF_CONFIG_I, AIN4_EF_CONFIG_I, AIN5_EF_CONFIG_I,	11706, 11708, 11710,
AIN6_EF_CONFIG_I, AIN7_EF_CONFIG_I, AIN8_EF_CONFIG_I,	11712, 11714, 11716,
AIN9_EF_CONFIG_I, AIN10_EF_CONFIG_I, AIN11_EF_CONFIG_I,	11718, 11720, 11722,
AIN12_EF_CONFIG_I, AIN13_EF_CONFIG_I, AIN14_EF_CONFIG_I,	11724, 11726, 11728,
AIN15_EF_CONFIG_I, AIN16_EF_CONFIG_I, AIN17_EF_CONFIG_I,	11730, 11732, 11734,
AIN18_EF_CONFIG_I, AIN19_EF_CONFIG_I, AIN20_EF_CONFIG_I,	11736, 11738, 11740,
AIN21_EF_CONFIG_I, AIN22_EF_CONFIG_I, AIN23_EF_CONFIG_I,	11742, 11744, 11746,
AIN24_EF_CONFIG_I, AIN25_EF_CONFIG_I, AIN26_EF_CONFIG_I,	11748, 11750, 11752,
AIN27_EF_CONFIG_I, AIN28_EF_CONFIG_I, AIN29_EF_CONFIG_I,	11754, 11756, 11758,
AIN30_EF_CONFIG_I, AIN31_EF_CONFIG_I, AIN32_EF_CONFIG_I,	11760, 11762, 11764,
AIN33_EF_CONFIG_I, AIN34_EF_CONFIG_I, AIN35_EF_CONFIG_I,	11766, 11768, 11770,
AIN36_EF_CONFIG_I, AIN37_EF_CONFIG_I, AIN38_EF_CONFIG_I,	11772, 11774, 11776,
AIN39_EF_CONFIG_I, AIN40_EF_CONFIG_I, AIN41_EF_CONFIG_I,	11778, 11780, 11782,
AIN42_EF_CONFIG_I, AIN43_EF_CONFIG_I, AIN44_EF_CONFIG_I,	11784, 11786, 11788,
AIN45_EF_CONFIG_I, AIN46_EF_CONFIG_I, AIN47_EF_CONFIG_I,	11790, 11792, 11794,
AIN48_EF_CONFIG_I, AIN49_EF_CONFIG_I, AIN50_EF_CONFIG_I,	11796, 11798, 11800,
AIN51_EF_CONFIG_I, AIN52_EF_CONFIG_I, AIN53_EF_CONFIG_I,	11802, 11804, 11806,
AIN54_EF_CONFIG_I, AIN55_EF_CONFIG_I, AIN56_EF_CONFIG_I,	11808, 11810, 11812,
AIN57_EF_CONFIG_I, AIN58_EF_CONFIG_I, AIN59_EF_CONFIG_I,	11814, 11816, 11818,
AIN60_EF_CONFIG_I, AIN61_EF_CONFIG_I, AIN62_EF_CONFIG_I,	11820, 11822, 11824,
AIN63_EF_CONFIG_I, AIN64_EF_CONFIG_I, AIN65_EF_CONFIG_I,	11826, 11828, 11830,
AIN66_EF_CONFIG_I, AIN67_EF_CONFIG_I, AIN68_EF_CONFIG_I,	11832, 11834, 11836,
AIN69_EF_CONFIG_I, AIN70_EF_CONFIG_I, AIN71_EF_CONFIG_I,	11838, 11840, 11842,
AIN72_EF_CONFIG_I, AIN73_EF_CONFIG_I, AIN74_EF_CONFIG_I,	11844, 11846, 11848,
AIN75_EF_CONFIG_I, AIN76_EF_CONFIG_I, AIN77_EF_CONFIG_I,	11850, 11852, 11854,
AIN78_EF_CONFIG_I, AIN79_EF_CONFIG_I, AIN80_EF_CONFIG_I,	11856, 11858, 11860,
AIN81_EF_CONFIG_I, AIN82_EF_CONFIG_I, AIN83_EF_CONFIG_I,	11862, 11864, 11866,
AIN84_EF_CONFIG_I, AIN85_EF_CONFIG_I, AIN86_EF_CONFIG_I,	11868, 11870, 11872,
AIN87_EF_CONFIG_I, AIN88_EF_CONFIG_I, AIN89_EF_CONFIG_I,	11874, 11876, 11878,
AIN90_EF_CONFIG_I, AIN91_EF_CONFIG_I, AIN92_EF_CONFIG_I,	11880, 11882, 11884,
AIN93_EF_CONFIG_I, AIN94_EF_CONFIG_I, AIN95_EF_CONFIG_I,	11886, 11888, 11890,
AIN96_EF_CONFIG_I, AIN97_EF_CONFIG_I, AIN98_EF_CONFIG_I,	11892, 11894, 11896,
AIN99_EF_CONFIG_I, AIN100_EF_CONFIG_I, AIN101_EF_CONFIG_I,	11898, 11900, 11902,
AIN102_EF_CONFIG_I, AIN103_EF_CONFIG_I, AIN104_EF_CONFIG_I,	11904, 11906, 11908,
AIN105_EF_CONFIG_I, AIN106_EF_CONFIG_I, AIN107_EF_CONFIG_I,	11910, 11912, 11914,
AIN108_EF_CONFIG_I, AIN109_EF_CONFIG_I, AIN110_EF_CONFIG_I,	11916, 11918, 11920,
AIN111_EF_CONFIG_I, AIN112_EF_CONFIG_I, AIN113_EF_CONFIG_I,	11922, 11924, 11926,
AIN114_EF_CONFIG_I, AIN115_EF_CONFIG_I, AIN116_EF_CONFIG_I,	11928, 11930, 11932,
AIN117_EF_CONFIG_I, AIN118_EF_CONFIG_I, AIN119_EF_CONFIG_I,	11934, 11936, 11938,
AIN120_EF_CONFIG_I, AIN121_EF_CONFIG_I, AIN122_EF_CONFIG_I,	11940, 11942, 11944,
AIN123_EF_CONFIG_I, AIN124_EF_CONFIG_I, AIN125_EF_CONFIG_I,	11946, 11948, 11950,
AIN126_EF_CONFIG_I, AIN127_EF_CONFIG_I, AIN128_EF_CONFIG_I,	11952, 11954, 11956,
AIN129_EF_CONFIG_I, AIN130_EF_CONFIG_I, AIN131_EF_CONFIG_I,	11958, 11960, 11962,
AIN132_EF_CONFIG_I, AIN133_EF_CONFIG_I, AIN134_EF_CONFIG_I,	11964, 11966, 11968,
AIN135_EF_CONFIG_I, AIN136_EF_CONFIG_I, AIN137_EF_CONFIG_I,	11970, 11972, 11974,
AIN138_EF_CONFIG_I, AIN139_EF_CONFIG_I, AIN140_EF_CONFIG_I,	11976, 11978, 11980,
AIN141_EF_CONFIG_I, AIN142_EF_CONFIG_I, AIN143_EF_CONFIG_I,	11982, 11984, 11986,
AIN144_EF_CONFIG_I, AIN145_EF_CONFIG_I, AIN146_EF_CONFIG_I,	11988, 11990, 11992,
AIN147_EF_CONFIG_I, AIN148_EF_CONFIG_I	11994, 11996

AIN#(0:148)_EF_CONFIG_J
- Starting Address: 12000

Function dependent on selected feature index.

- Data type: FLOAT32 (type index = 3)
- Readable and writable
- Default value: 0
- T8:
  - Minimum firmware version: 0.0123
- T7:

- Minimum firmware version: 1.0030

Expanded Names	Addresses
AIN0_EF_CONFIG_J, AIN1_EF_CONFIG_J, AIN2_EF_CONFIG_J,	12000, 12002, 12004,
AIN3_EF_CONFIG_J, AIN4_EF_CONFIG_J, AIN5_EF_CONFIG_J,	12006, 12008, 12010,
AIN6_EF_CONFIG_J, AIN7_EF_CONFIG_J, AIN8_EF_CONFIG_J,	12012, 12014, 12016,
AIN9_EF_CONFIG_J, AIN10_EF_CONFIG_J, AIN11_EF_CONFIG_J,	12018, 12020, 12022,
AIN12_EF_CONFIG_J, AIN13_EF_CONFIG_J, AIN14_EF_CONFIG_J,	12024, 12026, 12028,
AIN15_EF_CONFIG_J, AIN16_EF_CONFIG_J, AIN17_EF_CONFIG_J,	12030, 12032, 12034,
AIN18_EF_CONFIG_J, AIN19_EF_CONFIG_J, AIN20_EF_CONFIG_J,	12036, 12038, 12040,
AIN21_EF_CONFIG_J, AIN22_EF_CONFIG_J, AIN23_EF_CONFIG_J,	12042, 12044, 12046,
AIN24_EF_CONFIG_J, AIN25_EF_CONFIG_J, AIN26_EF_CONFIG_J,	12048, 12050, 12052,
AIN27_EF_CONFIG_J, AIN28_EF_CONFIG_J, AIN29_EF_CONFIG_J,	12054, 12056, 12058,
AIN30_EF_CONFIG_J, AIN31_EF_CONFIG_J, AIN32_EF_CONFIG_J,	12060, 12062, 12064,
AIN33_EF_CONFIG_J, AIN34_EF_CONFIG_J, AIN35_EF_CONFIG_J,	12066, 12068, 12070,
AIN36_EF_CONFIG_J, AIN37_EF_CONFIG_J, AIN38_EF_CONFIG_J,	12072, 12074, 12076,
AIN39_EF_CONFIG_J, AIN40_EF_CONFIG_J, AIN41_EF_CONFIG_J,	12078, 12080, 12082,
AIN42_EF_CONFIG_J, AIN43_EF_CONFIG_J, AIN44_EF_CONFIG_J,	12084, 12086, 12088,
AIN45_EF_CONFIG_J, AIN46_EF_CONFIG_J, AIN47_EF_CONFIG_J,	12090, 12092, 12094,
AIN48_EF_CONFIG_J, AIN49_EF_CONFIG_J, AIN50_EF_CONFIG_J,	12096, 12098, 12100,
AIN51_EF_CONFIG_J, AIN52_EF_CONFIG_J, AIN53_EF_CONFIG_J,	12102, 12104, 12106,
AIN54_EF_CONFIG_J, AIN55_EF_CONFIG_J, AIN56_EF_CONFIG_J,	12108, 12110, 12112,
AIN57_EF_CONFIG_J, AIN58_EF_CONFIG_J, AIN59_EF_CONFIG_J,	12114, 12116, 12118,
AIN60_EF_CONFIG_J, AIN61_EF_CONFIG_J, AIN62_EF_CONFIG_J,	12120, 12122, 12124,
AIN63_EF_CONFIG_J, AIN64_EF_CONFIG_J, AIN65_EF_CONFIG_J,	12126, 12128, 12130,
AIN66_EF_CONFIG_J, AIN67_EF_CONFIG_J, AIN68_EF_CONFIG_J,	12132, 12134, 12136,
AIN69_EF_CONFIG_J, AIN70_EF_CONFIG_J, AIN71_EF_CONFIG_J,	12138, 12140, 12142,
AIN72_EF_CONFIG_J, AIN73_EF_CONFIG_J, AIN74_EF_CONFIG_J,	12144, 12146, 12148,
AIN75_EF_CONFIG_J, AIN76_EF_CONFIG_J, AIN77_EF_CONFIG_J,	12150, 12152, 12154,
AIN78_EF_CONFIG_J, AIN79_EF_CONFIG_J, AIN80_EF_CONFIG_J,	12156, 12158, 12160,
AIN81_EF_CONFIG_J, AIN82_EF_CONFIG_J, AIN83_EF_CONFIG_J,	12162, 12164, 12166,
AIN84_EF_CONFIG_J, AIN85_EF_CONFIG_J, AIN86_EF_CONFIG_J,	12168, 12170, 12172,
AIN87_EF_CONFIG_J, AIN88_EF_CONFIG_J, AIN89_EF_CONFIG_J,	12174, 12176, 12178,
AIN90_EF_CONFIG_J, AIN91_EF_CONFIG_J, AIN92_EF_CONFIG_J,	12180, 12182, 12184,
AIN93_EF_CONFIG_J, AIN94_EF_CONFIG_J, AIN95_EF_CONFIG_J,	12186, 12188, 12190,
AIN96_EF_CONFIG_J, AIN97_EF_CONFIG_J, AIN98_EF_CONFIG_J,	12192, 12194, 12196,
AIN99_EF_CONFIG_J, AIN100_EF_CONFIG_J, AIN101_EF_CONFIG_J,	12198, 12200, 12202,
AIN102_EF_CONFIG_J, AIN103_EF_CONFIG_J, AIN104_EF_CONFIG_J,	12204, 12206, 12208,
AIN105_EF_CONFIG_J, AIN106_EF_CONFIG_J, AIN107_EF_CONFIG_J,	12210, 12212, 12214,
AIN108_EF_CONFIG_J, AIN109_EF_CONFIG_J, AIN110_EF_CONFIG_J,	12216, 12218, 12220,
AIN111_EF_CONFIG_J, AIN112_EF_CONFIG_J, AIN113_EF_CONFIG_J,	12222, 12224, 12226,
AIN114_EF_CONFIG_J, AIN115_EF_CONFIG_J, AIN116_EF_CONFIG_J,	12228, 12230, 12232,
AIN117_EF_CONFIG_J, AIN118_EF_CONFIG_J, AIN119_EF_CONFIG_J,	12234, 12236, 12238,
AIN120_EF_CONFIG_J, AIN121_EF_CONFIG_J, AIN122_EF_CONFIG_J,	12240, 12242, 12244,
AIN123_EF_CONFIG_J, AIN124_EF_CONFIG_J, AIN125_EF_CONFIG_J,	12246, 12248, 12250,
AIN126_EF_CONFIG_J, AIN127_EF_CONFIG_J, AIN128_EF_CONFIG_J,	12252, 12254, 12256,
AIN129_EF_CONFIG_J, AIN130_EF_CONFIG_J, AIN131_EF_CONFIG_J,	12258, 12260, 12262,
AIN132_EF_CONFIG_J, AIN133_EF_CONFIG_J, AIN134_EF_CONFIG_J,	12264, 12266, 12268,
AIN135_EF_CONFIG_J, AIN136_EF_CONFIG_J, AIN137_EF_CONFIG_J,	12270, 12272, 12274,
AIN138_EF_CONFIG_J, AIN139_EF_CONFIG_J, AIN140_EF_CONFIG_J,	12276, 12278, 12280,
AIN141_EF_CONFIG_J, AIN142_EF_CONFIG_J, AIN143_EF_CONFIG_J,	12282, 12284, 12286,
AIN144_EF_CONFIG_J, AIN145_EF_CONFIG_J, AIN146_EF_CONFIG_J,	12288, 12290, 12292,
AIN147_EF_CONFIG_J, AIN148_EF_CONFIG_J	12294, 12296

All TDAC TAGS:

Name	Start Address	Type	Access
TDAC#(0:20)	30000	FLOAT32	W
TDAC_SERIAL_NUMBER	55200	UINT32	R
TDAC_SPEED_THROTTLE	55202	UINT32	R/W

TDAC#(0:20)
- Starting Address: 30000

Update a voltage output on a connected LJTick-DAC accessory. Even TDAC# = DACA, Odd TDAC# = DACB. For instance, if LJTick-DAC accessory is connected to FIO2/FIO3 block on main device, TDAC2 corresponds with DACA, and TDAC3 corresponds with DACB.

- Data type: FLOAT32 (type index = 3)
- Write-only
- Default value: 0

Expanded Names	Addresses
TDAC0, TDAC1, TDAC2, TDAC3, TDAC4, TDAC5, TDAC6, TDAC7, TDAC8, TDAC9, TDAC10, TDAC11, TDAC12, TDAC13, TDAC14, TDAC15, TDAC16, TDAC17, TDAC18, TDAC19, TDAC20	30000, 30002, 30004, 30006, 30008, 30010, 30012, 30014, 30016, 30018, 30020, 30022, 30024, 30026, 30028, 30030, 30032, 30034, 30036, 30038, 30040

TDAC_SERIAL_NUMBER
- Address: 55200

Returns the serial number of an LJTick-DAC, and forces a re-read of the calibration constants. Which LJTDAC is determined by the last write to TDAC# ... whether it was successful or not.

- Data type: UINT32 (type index = 1)
- Read-only
- Default value: 0

TDAC_SPEED_THROTTLE
- Address: 55202

Sets the I2C clock speed that will be used when communicating with the TDAC. Default value is 65516. See I2C_SPEED_THROTTLE for more detail.

- Data type: UINT32 (type index = 1)
- Readable and writable
- Default value: 65516
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0208
- T4:
  - Minimum firmware version: 0.1000

All SBUS TAGS:

Name	Start Address	Type	Access
SBUS#(0:21)_TEMP	30100	FLOAT32	R
SBUS#(0:21)_RH	30150	FLOAT32	R
SBUS#(0:21)_DATA_DIONUM	30200	UINT16	R/W
SBUS#(0:21)_CLOCK_DIONUM	30225	UINT16	R/W
SBUS#(0:21)_BACKGROUND_ENABLE	30250	UINT16	R/W

SBUS_ALL_DATA_DIONUM	30275	UINT16	R/W
SBUS_ALL_CLOCK_DIONUM	30276	UINT16	R/W
SBUS_ALL_POWER_DIONUM	30277	UINT16	R/W
SBUS_ALL_CLOCK_SPEED	30278	UINT16	R/W

SBUS#(0:21)_TEMP
- Starting Address: 30100

Reads temperature in Kelvin from an SBUS sensor (EI-1050/SHT1x/SHT7x). SBUS# is the DIO line for the EI-1050 enable. If SBUS# is the same as the value specified for data or clock line, there will be no control of an enable line.

- Data type: FLOAT32 (type index = 3)
- Read-only
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0056

Expanded Names	Addresses
SBUS0_TEMP, SBUS1_TEMP, SBUS2_TEMP, SBUS3_TEMP, SBUS4_TEMP, SBUS5_TEMP, SBUS6_TEMP, SBUS7_TEMP, SBUS8_TEMP, SBUS9_TEMP, SBUS10_TEMP, SBUS11_TEMP, SBUS12_TEMP, SBUS13_TEMP, SBUS14_TEMP, SBUS15_TEMP, SBUS16_TEMP, SBUS17_TEMP, SBUS18_TEMP, SBUS19_TEMP, SBUS20_TEMP, SBUS21_TEMP	30100, 30102, 30104, 30106, 30108, 30110, 30112, 30114, 30116, 30118, 30120, 30122, 30124, 30126, 30128, 30130, 30132, 30134, 30136, 30138, 30140, 30142

SBUS#(0:21)_RH
- Starting Address: 30150

Reads humidity in % from an external SBUS sensor (EI-1050/SHT1x/SHT7x). # is the DIO line for the EI-1050 enable. If # is the same as the value specified for data or clock line, there will be no control of an enable line.

- Data type: FLOAT32 (type index = 3)
- Read-only
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0056

Expanded Names	Addresses
SBUS0_RH, SBUS1_RH, SBUS2_RH, SBUS3_RH, SBUS4_RH, SBUS5_RH, SBUS6_RH, SBUS7_RH, SBUS8_RH, SBUS9_RH, SBUS10_RH, SBUS11_RH, SBUS12_RH, SBUS13_RH, SBUS14_RH, SBUS15_RH, SBUS16_RH, SBUS17_RH, SBUS18_RH, SBUS19_RH, SBUS20_RH, SBUS21_RH	30150, 30152, 30154, 30156, 30158, 30160, 30162, 30164, 30166, 30168, 30170, 30172, 30174, 30176, 30178, 30180, 30182, 30184, 30186, 30188, 30190, 30192

SBUS#(0:21)_DATA_DIONUM
- Starting Address: 30200

This is the DIO# that the external sensor's data line is connected to.

- Data type: UINT16 (type index = 0)
- Readable and writable
- Default value: 0
- T8:
  - Minimum firmware version: 0.0123

- T7:
  - Default = FIO0
  - Minimum firmware version: 1.0056
- T4:
  - Default = FIO4

Expanded Names	Addresses
SBUS0_DATA_DIONUM, SBUS1_DATA_DIONUM, SBUS2_DATA_DIONUM, SBUS3_DATA_DIONUM, SBUS4_DATA_DIONUM, SBUS5_DATA_DIONUM, SBUS6_DATA_DIONUM, SBUS7_DATA_DIONUM, SBUS8_DATA_DIONUM, SBUS9_DATA_DIONUM, SBUS10_DATA_DIONUM, SBUS11_DATA_DIONUM, SBUS12_DATA_DIONUM, SBUS13_DATA_DIONUM, SBUS14_DATA_DIONUM, SBUS15_DATA_DIONUM, SBUS16_DATA_DIONUM, SBUS17_DATA_DIONUM, SBUS18_DATA_DIONUM, SBUS19_DATA_DIONUM, SBUS20_DATA_DIONUM, SBUS21_DATA_DIONUM	30200, 30201, 30202, 30203, 30204, 30205, 30206, 30207, 30208, 30209, 30210, 30211, 30212, 30213, 30214, 30215, 30216, 30217, 30218, 30219, 30220, 30221

SBUS#(0:21)_CLOCK_DIONUM
- Starting Address: 30225

This is the DIO# that the external sensor's clock line is connected to.

- Data type: UINT16 (type index = 0)
- Readable and writable
- Default value: 1
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Default = FIO1
  - Minimum firmware version: 1.0056
- T4:
  - Default = FIO5

Expanded Names	Addresses
SBUS0_CLOCK_DIONUM, SBUS1_CLOCK_DIONUM, SBUS2_CLOCK_DIONUM, SBUS3_CLOCK_DIONUM, SBUS4_CLOCK_DIONUM, SBUS5_CLOCK_DIONUM, SBUS6_CLOCK_DIONUM, SBUS7_CLOCK_DIONUM, SBUS8_CLOCK_DIONUM, SBUS9_CLOCK_DIONUM, SBUS10_CLOCK_DIONUM, SBUS11_CLOCK_DIONUM, SBUS12_CLOCK_DIONUM, SBUS13_CLOCK_DIONUM, SBUS14_CLOCK_DIONUM, SBUS15_CLOCK_DIONUM, SBUS16_CLOCK_DIONUM, SBUS17_CLOCK_DIONUM, SBUS18_CLOCK_DIONUM, SBUS19_CLOCK_DIONUM, SBUS20_CLOCK_DIONUM, SBUS21_CLOCK_DIONUM	30225, 30226, 30227, 30228, 30229, 30230, 30231, 30232, 30233, 30234, 30235, 30236, 30237, 30238, 30239, 30240, 30241, 30242, 30243, 30244, 30245, 30246

SBUS#(0:21)_BACKGROUND_ENABLE
- Starting Address: 30250

Currently unsupported.

- Data type: UINT16 (type index = 0)
- Readable and writable
- Default value: 0
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0056

Expanded Names	Addresses

SBUS0_BACKGROUND_ENABLE, SBUS1_BACKGROUND_ENABLE,	30250, 30251,
SBUS2_BACKGROUND_ENABLE, SBUS3_BACKGROUND_ENABLE,	30252, 30253,
SBUS4_BACKGROUND_ENABLE, SBUS5_BACKGROUND_ENABLE,	30254, 30255,
SBUS6_BACKGROUND_ENABLE, SBUS7_BACKGROUND_ENABLE,	30256, 30257,
SBUS8_BACKGROUND_ENABLE, SBUS9_BACKGROUND_ENABLE,	30258, 30259,
SBUS10_BACKGROUND_ENABLE, SBUS11_BACKGROUND_ENABLE,	30260, 30261,
SBUS12_BACKGROUND_ENABLE, SBUS13_BACKGROUND_ENABLE,	30262, 30263,
SBUS14_BACKGROUND_ENABLE, SBUS15_BACKGROUND_ENABLE,	30264, 30265,
SBUS16_BACKGROUND_ENABLE, SBUS17_BACKGROUND_ENABLE,	30266, 30267,
SBUS18_BACKGROUND_ENABLE, SBUS19_BACKGROUND_ENABLE,	30268, 30269,
SBUS20_BACKGROUND_ENABLE, SBUS21_BACKGROUND_ENABLE	30270, 30271

SBUS_ALL_DATA_DIONUM
- Address: 30275

A write to this global parameter sets all SBUS data line registers to the same value. A read will return the correct setting if all channels are set the same, but otherwise will return 0xFF.

- Data type: UINT16 (type index = 0)
- Readable and writable
- Default value: 0
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0056

SBUS_ALL_CLOCK_DIONUM
- Address: 30276

A write to this global parameter sets all SBUS clock line registers to the same value. A read will return the correct setting if all channels are set the same, but otherwise will return 0xFF.

- Data type: UINT16 (type index = 0)
- Readable and writable
- Default value: 1
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0056

SBUS_ALL_POWER_DIONUM
- Address: 30277

Sets the power line. This DIO is set to output-high upon any read of SBUS#_TEMP or SBUS#_RH. Default is FIO6 for the T4 and FIO2 for the T7. An FIO line can power up to 4 sensors while an EIO/CIO/MIO line or DAC line can power up to 20 sensors. Set to 9999 to disable. To use multiple power lines, use a DAC line for power, or otherwise control power yourself, set this to 9999 and then control power using writes to normal registers such as FIO5, EIO0, or DAC0.

- Data type: UINT16 (type index = 0)
- Readable and writable
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Default value: 2
  - Minimum firmware version: 1.0056
- T4:
  - Default value: 6

SBUS_ALL_CLOCK_SPEED
- Address: 30278

Sets the clock speed. The clock is software generated so the resulting frequency is not exact. Valid range is 0-65535. Larger values are faster. 0 is the fastest option and is equivalent to 65536. A value of 0 is ~200 kHz. A value of 65000 is ~9.1 kHz.

- Data type: UINT16 (type index = 0)
- Readable and writable
- Default value: 65000
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0142

All DIO_EF TAGS:

Name	Start Address	Type	Access
DIO_EF_CLOCK0_ENABLE	44900	UINT16	R/W
DIO_EF_CLOCK0_DIVISOR	44901	UINT16	R/W
DIO_EF_CLOCK0_OPTIONS	44902	UINT32	R/W
DIO_EF_CLOCK0_ROLL_VALUE	44904	UINT32	R/W
DIO_EF_CLOCK1_ENABLE	44910	UINT16	R/W
DIO_EF_CLOCK1_DIVISOR	44911	UINT16	R/W
DIO_EF_CLOCK1_OPTIONS	44912	UINT32	R/W
DIO_EF_CLOCK1_ROLL_VALUE	44914	UINT32	R/W
DIO_EF_CLOCK2_ENABLE	44920	UINT16	R/W
DIO_EF_CLOCK2_DIVISOR	44921	UINT16	R/W
DIO_EF_CLOCK2_OPTIONS	44922	UINT32	R/W
DIO_EF_CLOCK2_ROLL_VALUE	44924	UINT32	R/W
DIO_EF_CLOCK0_COUNT	44908	UINT32	R
DIO_EF_CLOCK1_COUNT	44918	UINT32	R
DIO_EF_CLOCK2_COUNT	44928	UINT32	R
DIO#(0:21)_EF_ENABLE	44000	UINT32	R/W
DIO#(0:21)_EF_INDEX	44100	UINT32	R/W
DIO#(0:21)_EF_OPTIONS	44200	UINT32	R/W
DIO#(0:21)_EF_CONFIG_A	44300	UINT32	R/W
DIO#(0:21)_EF_CONFIG_B	44400	UINT32	R/W
DIO#(0:21)_EF_CONFIG_C	44500	UINT32	R/W
DIO#(0:21)_EF_CONFIG_D	44600	UINT32	R/W
DIO#(0:21)_EF_READ_A	3000	UINT32	R
DIO#(0:21)_EF_READ_A_AND_RESET	3100	UINT32	R
DIO#(0:21)_EF_READ_B	3200	UINT32	R
DIO#(0:21)_EF_READ_A_F	3500	FLOAT32	R
DIO#(0:21)_EF_READ_A_F_AND_RESET	3600	FLOAT32	R
DIO#(0:21)_EF_READ_B_F	3700	FLOAT32	R
DIO#(0:21)_EF_EASY_FREQUENCY_IN	45000	FLOAT32	R/W

DIO_EF_CLOCK0_ENABLE
- Address: 44900

1 = enabled. 0 = disabled. Must be disabled during configuration.

- Data type: UINT16 (type index = 0)
- Readable and writable

- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 0.8700

DIO_EF_CLOCK0_DIVISOR
- Address: 44901

Divides the core clock. Valid options: 1,2,4,8,16,32,64,256.

- Data type: UINT16 (type index = 0)
- Readable and writable
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 0.9320

DIO_EF_CLOCK0_OPTIONS
- Address: 44902

Bitmask: bit0: 1 = use external clock. All other bits reserved.

- Data type: UINT32 (type index = 1)
- Readable and writable
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 0.8700

DIO_EF_CLOCK0_ROLL_VALUE
- Address: 44904

The clock count will increment continuously and then start over at zero as it reaches the roll value. DIO_EF_CLOCK0 is a 32-bit clock, valid values are 0 to (2^32 - 1). 0 results in the max roll value (2^32).

- Data type: UINT32 (type index = 1)
- Readable and writable
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 0.8700

DIO_EF_CLOCK1_ENABLE
- Address: 44910

1 = enabled. 0 = disabled. Must be disabled during configuration.

- Data type: UINT16 (type index = 0)
- Readable and writable
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 0.8700

DIO_EF_CLOCK1_DIVISOR
- Address: 44911

Divides the core clock. Valid options: 1,2,4,8,16,32,64,256.

- Data type: UINT16 (type index = 0)
- Readable and writable
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 0.9320

DIO_EF_CLOCK1_OPTIONS
- Address: 44912

Bitmask: bit0: 1 = use external clock. All other bits reserved.

- Data type: UINT32 (type index = 1)
- Readable and writable
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 0.8700

DIO_EF_CLOCK1_ROLL_VALUE
- Address: 44914

The clock will count to this value and then start over at zero. The clock pulses counted are those after the divisor. 0 results in the max roll value possible. This is a 32-bit value (0-4294967295) if using a 32-bit clock, and a 16-bit value (0-65535) if using a 16-bit clock.

- Data type: UINT32 (type index = 1)
- Readable and writable
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 0.8700

DIO_EF_CLOCK2_ENABLE
- Address: 44920

1 = enabled. 0 = disabled. Must be disabled during configuration.

- Data type: UINT16 (type index = 0)
- Readable and writable
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 0.8700

DIO_EF_CLOCK2_DIVISOR
- Address: 44921

Divides the core clock. Valid options: 1,2,4,8,16,32,64,256.

- Data type: UINT16 (type index = 0)
- Readable and writable
- T8:
  - Minimum firmware version: 0.0123

- T7:
    - Minimum firmware version: 0.9320

DIO_EF_CLOCK2_OPTIONS
- Address: 44922

Bitmask: bit0: 1 = use external clock. All other bits reserved.

- Data type: UINT32 (type index = 1)
- Readable and writable
- T8:
    - Minimum firmware version: 0.0123
- T7:
    - Minimum firmware version: 0.8700

DIO_EF_CLOCK2_ROLL_VALUE
- Address: 44924

The clock will count to this value and then start over at zero. The clock pulses counted are those after the divisor. 0 results in the max roll value possible. This is a 32-bit value (0-4294967295) if using a 32-bit clock, and a 16-bit value (0-65535) if using a 16-bit clock.

- Data type: UINT32 (type index = 1)
- Readable and writable
- T8:
    - Minimum firmware version: 0.0123
- T7:
    - Minimum firmware version: 0.8700

DIO_EF_CLOCK0_COUNT
- Address: 44908

Current tick count of this clock. Will read between 0 and ROLL_VALUE-1.

- Data type: UINT32 (type index = 1)
- Read-only
- T8:
    - Minimum firmware version: 0.0123
- T7:
    - Minimum firmware version: 0.9430

DIO_EF_CLOCK1_COUNT
- Address: 44918

Current tick count of this clock. Will read between 0 and ROLL_VALUE-1.

- Data type: UINT32 (type index = 1)
- Read-only
- T8:
    - Minimum firmware version: 0.0123
- T7:
    - Minimum firmware version: 0.9430

DIO_EF_CLOCK2_COUNT
- Address: 44928

Current tick count of this clock. Will read between 0 and ROLL_VALUE-1.

- Data type: UINT32 (type index = 1)
- Read-only
- T8:
    - Minimum firmware version: 0.0123
- T7:
    - Minimum firmware version: 0.9430

DIO#(0:21)_EF_ENABLE
- Starting Address: 44000

1 = enabled. 0 = disabled. Must be disabled during configuration. Note that DIO-EF reads work when disabled and do not return an error.

- Data type: UINT32 (type index = 1)
- Readable and writable
- T8:
    - Minimum firmware version: 0.0123
- T7:
    - Minimum firmware version: 0.8800

Expanded Names	Addresses
DIO0_EF_ENABLE, DIO1_EF_ENABLE, DIO2_EF_ENABLE, DIO3_EF_ENABLE, DIO4_EF_ENABLE, DIO5_EF_ENABLE, DIO6_EF_ENABLE, DIO7_EF_ENABLE, DIO8_EF_ENABLE, DIO9_EF_ENABLE, DIO10_EF_ENABLE, DIO11_EF_ENABLE, DIO12_EF_ENABLE, DIO13_EF_ENABLE, DIO14_EF_ENABLE, DIO15_EF_ENABLE, DIO16_EF_ENABLE, DIO17_EF_ENABLE, DIO18_EF_ENABLE, DIO19_EF_ENABLE, DIO20_EF_ENABLE, DIO21_EF_ENABLE	44000, 44002, 44004, 44006, 44008, 44010, 44012, 44014, 44016, 44018, 44020, 44022, 44024, 44026, 44028, 44030, 44032, 44034, 44036, 44038, 44040, 44042

DIO#(0:21)_EF_INDEX
- Starting Address: 44100

An index to specify the feature you want.

- Data type: UINT32 (type index = 1)
- Readable and writable
- T8:
    - Minimum firmware version: 0.0123
- T7:
    - Minimum firmware version: 0.8800

Expanded Names	Addresses
DIO0_EF_INDEX, DIO1_EF_INDEX, DIO2_EF_INDEX, DIO3_EF_INDEX, DIO4_EF_INDEX, DIO5_EF_INDEX, DIO6_EF_INDEX, DIO7_EF_INDEX, DIO8_EF_INDEX, DIO9_EF_INDEX, DIO10_EF_INDEX, DIO11_EF_INDEX, DIO12_EF_INDEX, DIO13_EF_INDEX, DIO14_EF_INDEX, DIO15_EF_INDEX, DIO16_EF_INDEX, DIO17_EF_INDEX, DIO18_EF_INDEX, DIO19_EF_INDEX, DIO20_EF_INDEX, DIO21_EF_INDEX	44100, 44102, 44104, 44106, 44108, 44110, 44112, 44114, 44116, 44118, 44120, 44122, 44124, 44126, 44128, 44130, 44132, 44134, 44136, 44138, 44140, 44142

DIO#(0:21)_EF_OPTIONS
- Starting Address: 44200

Function dependent on selected feature index.

- Data type: UINT32 (type index = 1)
- Readable and writable
- T8:

    - Minimum firmware version: 0.0123
- T7:
    - Minimum firmware version: 0.8800

Expanded Names	Addresses
DIO0_EF_OPTIONS, DIO1_EF_OPTIONS, DIO2_EF_OPTIONS, DIO3_EF_OPTIONS, DIO4_EF_OPTIONS, DIO5_EF_OPTIONS, DIO6_EF_OPTIONS, DIO7_EF_OPTIONS, DIO8_EF_OPTIONS, DIO9_EF_OPTIONS, DIO10_EF_OPTIONS, DIO11_EF_OPTIONS, DIO12_EF_OPTIONS, DIO13_EF_OPTIONS, DIO14_EF_OPTIONS, DIO15_EF_OPTIONS, DIO16_EF_OPTIONS, DIO17_EF_OPTIONS, DIO18_EF_OPTIONS, DIO19_EF_OPTIONS, DIO20_EF_OPTIONS, DIO21_EF_OPTIONS	44200, 44202, 44204, 44206, 44208, 44210, 44212, 44214, 44216, 44218, 44220, 44222, 44224, 44226, 44228, 44230, 44232, 44234, 44236, 44238, 44240, 44242

DIO#(0:21)_EF_CONFIG_A
- Starting Address: 44300

Function dependent on selected feature index.

- Data type: UINT32 (type index = 1)
- Readable and writable
- T8:
    - Minimum firmware version: 0.0123
- T7:
    - Minimum firmware version: 0.8800

Expanded Names	Addresses
DIO0_EF_CONFIG_A, DIO1_EF_CONFIG_A, DIO2_EF_CONFIG_A, DIO3_EF_CONFIG_A, DIO4_EF_CONFIG_A, DIO5_EF_CONFIG_A, DIO6_EF_CONFIG_A, DIO7_EF_CONFIG_A, DIO8_EF_CONFIG_A, DIO9_EF_CONFIG_A, DIO10_EF_CONFIG_A, DIO11_EF_CONFIG_A, DIO12_EF_CONFIG_A, DIO13_EF_CONFIG_A, DIO14_EF_CONFIG_A, DIO15_EF_CONFIG_A, DIO16_EF_CONFIG_A, DIO17_EF_CONFIG_A, DIO18_EF_CONFIG_A, DIO19_EF_CONFIG_A, DIO20_EF_CONFIG_A, DIO21_EF_CONFIG_A	44300, 44302, 44304, 44306, 44308, 44310, 44312, 44314, 44316, 44318, 44320, 44322, 44324, 44326, 44328, 44330, 44332, 44334, 44336, 44338, 44340, 44342

DIO#(0:21)_EF_CONFIG_B
- Starting Address: 44400

Function dependent on selected feature index.

- Data type: UINT32 (type index = 1)
- Readable and writable
- T8:
    - Minimum firmware version: 0.0123
- T7:
    - Minimum firmware version: 0.8800

Expanded Names	Addresses
DIO0_EF_CONFIG_B, DIO1_EF_CONFIG_B, DIO2_EF_CONFIG_B, DIO3_EF_CONFIG_B, DIO4_EF_CONFIG_B, DIO5_EF_CONFIG_B, DIO6_EF_CONFIG_B, DIO7_EF_CONFIG_B, DIO8_EF_CONFIG_B, DIO9_EF_CONFIG_B, DIO10_EF_CONFIG_B, DIO11_EF_CONFIG_B, DIO12_EF_CONFIG_B, DIO13_EF_CONFIG_B, DIO14_EF_CONFIG_B, DIO15_EF_CONFIG_B, DIO16_EF_CONFIG_B, DIO17_EF_CONFIG_B, DIO18_EF_CONFIG_B, DIO19_EF_CONFIG_B, DIO20_EF_CONFIG_B, DIO21_EF_CONFIG_B	44400, 44402, 44404, 44406, 44408, 44410, 44412, 44414, 44416, 44418, 44420, 44422, 44424, 44426, 44428, 44430, 44432, 44434, 44436, 44438, 44440, 44442

DIO#(0:21)_EF_CONFIG_C
- Starting Address: 44500

Function dependent on selected feature index.

- Data type: UINT32 (type index = 1)
- Readable and writable
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 0.8800

Expanded Names	Addresses
DIO0_EF_CONFIG_C, DIO1_EF_CONFIG_C, DIO2_EF_CONFIG_C, DIO3_EF_CONFIG_C, DIO4_EF_CONFIG_C, DIO5_EF_CONFIG_C, DIO6_EF_CONFIG_C, DIO7_EF_CONFIG_C, DIO8_EF_CONFIG_C, DIO9_EF_CONFIG_C, DIO10_EF_CONFIG_C, DIO11_EF_CONFIG_C, DIO12_EF_CONFIG_C, DIO13_EF_CONFIG_C, DIO14_EF_CONFIG_C, DIO15_EF_CONFIG_C, DIO16_EF_CONFIG_C, DIO17_EF_CONFIG_C, DIO18_EF_CONFIG_C, DIO19_EF_CONFIG_C, DIO20_EF_CONFIG_C, DIO21_EF_CONFIG_C	44500, 44502, 44504, 44506, 44508, 44510, 44512, 44514, 44516, 44518, 44520, 44522, 44524, 44526, 44528, 44530, 44532, 44534, 44536, 44538, 44540, 44542

DIO#(0:21)_EF_CONFIG_D
- Starting Address: 44600

Function dependent on selected feature index.

- Data type: UINT32 (type index = 1)
- Readable and writable
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 0.8800

Expanded Names	Addresses
DIO0_EF_CONFIG_D, DIO1_EF_CONFIG_D, DIO2_EF_CONFIG_D, DIO3_EF_CONFIG_D, DIO4_EF_CONFIG_D, DIO5_EF_CONFIG_D, DIO6_EF_CONFIG_D, DIO7_EF_CONFIG_D, DIO8_EF_CONFIG_D, DIO9_EF_CONFIG_D, DIO10_EF_CONFIG_D, DIO11_EF_CONFIG_D, DIO12_EF_CONFIG_D, DIO13_EF_CONFIG_D, DIO14_EF_CONFIG_D, DIO15_EF_CONFIG_D, DIO16_EF_CONFIG_D, DIO17_EF_CONFIG_D, DIO18_EF_CONFIG_D, DIO19_EF_CONFIG_D, DIO20_EF_CONFIG_D, DIO21_EF_CONFIG_D	44600, 44602, 44604, 44606, 44608, 44610, 44612, 44614, 44616, 44618, 44620, 44622, 44624, 44626, 44628, 44630, 44632, 44634, 44636, 44638, 44640, 44642

DIO#(0:21)_EF_READ_A
- Starting Address: 3000

Reads an unsigned integer value. The meaning of the integer is dependent on selected feature index.

- Data type: UINT32 (type index = 1)
- Read-only
- This register may be streamed

Expanded Names	Addresses

DIO0_EF_READ_A, DIO1_EF_READ_A, DIO2_EF_READ_A, DIO3_EF_READ_A,	3000, 3002, 3004, 3006,
DIO4_EF_READ_A, DIO5_EF_READ_A, DIO6_EF_READ_A, DIO7_EF_READ_A,	3008, 3010, 3012, 3014,
DIO8_EF_READ_A, DIO9_EF_READ_A, DIO10_EF_READ_A, DIO11_EF_READ_A,	3016, 3018, 3020, 3022,
DIO12_EF_READ_A, DIO13_EF_READ_A, DIO14_EF_READ_A,	3024, 3026, 3028, 3030,
DIO15_EF_READ_A, DIO16_EF_READ_A, DIO17_EF_READ_A,	3032, 3034, 3036, 3038,
DIO18_EF_READ_A, DIO19_EF_READ_A, DIO20_EF_READ_A, DIO21_EF_READ_A	3040, 3042

DIO#(0:21)_EF_READ_A_AND_RESET
- Starting Address: 3100

Reads the same value as DIO#(0:22)_EF_READ_A and forces a reset.

- Data type: UINT32 (type index = 1)
- Read-only
- This register may be streamed

Expanded Names	Addresses
DIO0_EF_READ_A_AND_RESET, DIO1_EF_READ_A_AND_RESET,	3100, 3102,
DIO2_EF_READ_A_AND_RESET, DIO3_EF_READ_A_AND_RESET,	3104, 3106,
DIO4_EF_READ_A_AND_RESET, DIO5_EF_READ_A_AND_RESET,	3108, 3110,
DIO6_EF_READ_A_AND_RESET, DIO7_EF_READ_A_AND_RESET,	3112, 3114,
DIO8_EF_READ_A_AND_RESET, DIO9_EF_READ_A_AND_RESET,	3116, 3118,
DIO10_EF_READ_A_AND_RESET, DIO11_EF_READ_A_AND_RESET,	3120, 3122,
DIO12_EF_READ_A_AND_RESET, DIO13_EF_READ_A_AND_RESET,	3124, 3126,
DIO14_EF_READ_A_AND_RESET, DIO15_EF_READ_A_AND_RESET,	3128, 3130,
DIO16_EF_READ_A_AND_RESET, DIO17_EF_READ_A_AND_RESET,	3132, 3134,
DIO18_EF_READ_A_AND_RESET, DIO19_EF_READ_A_AND_RESET,	3136, 3138,
DIO20_EF_READ_A_AND_RESET, DIO21_EF_READ_A_AND_RESET	3140, 3142

DIO#(0:21)_EF_READ_B
- Starting Address: 3200

Reads an unsigned integer value. The meaning of the integer is dependent on selected feature index.

- Data type: UINT32 (type index = 1)
- Read-only
- This register may be streamed

Expanded Names	Addresses
DIO0_EF_READ_B, DIO1_EF_READ_B, DIO2_EF_READ_B, DIO3_EF_READ_B,	3200, 3202, 3204, 3206,
DIO4_EF_READ_B, DIO5_EF_READ_B, DIO6_EF_READ_B, DIO7_EF_READ_B,	3208, 3210, 3212, 3214,
DIO8_EF_READ_B, DIO9_EF_READ_B, DIO10_EF_READ_B, DIO11_EF_READ_B,	3216, 3218, 3220, 3222,
DIO12_EF_READ_B, DIO13_EF_READ_B, DIO14_EF_READ_B,	3224, 3226, 3228, 3230,
DIO15_EF_READ_B, DIO16_EF_READ_B, DIO17_EF_READ_B,	3232, 3234, 3236, 3238,
DIO18_EF_READ_B, DIO19_EF_READ_B, DIO20_EF_READ_B, DIO21_EF_READ_B	3240, 3242

DIO#(0:21)_EF_READ_A_F
- Starting Address: 3500

Reads a floating point value. The meaning of value is dependent on selected feature index.

- Data type: FLOAT32 (type index = 3)
- Read-only

Expanded Names	Addresses

DIO0_EF_READ_A_F, DIO1_EF_READ_A_F, DIO2_EF_READ_A_F, DIO3_EF_READ_A_F, DIO4_EF_READ_A_F, DIO5_EF_READ_A_F, DIO6_EF_READ_A_F, DIO7_EF_READ_A_F, DIO8_EF_READ_A_F, DIO9_EF_READ_A_F, DIO10_EF_READ_A_F, DIO11_EF_READ_A_F, DIO12_EF_READ_A_F, DIO13_EF_READ_A_F, DIO14_EF_READ_A_F, DIO15_EF_READ_A_F, DIO16_EF_READ_A_F, DIO17_EF_READ_A_F, DIO18_EF_READ_A_F, DIO19_EF_READ_A_F, DIO20_EF_READ_A_F, DIO21_EF_READ_A_F	3500, 3502, 3504, 3506, 3508, 3510, 3512, 3514, 3516, 3518, 3520, 3522, 3524, 3526, 3528, 3530, 3532, 3534, 3536, 3538, 3540, 3542

DIO#(0:21)_EF_READ_A_F_AND_RESET
- Starting Address: 3600

Reads a floating point value and forces a reset. The meaning of value is dependent on selected feature index.

- Data type: FLOAT32 (type index = 3)
- Read-only

Expanded Names	Addresses
DIO0_EF_READ_A_F_AND_RESET, DIO1_EF_READ_A_F_AND_RESET, DIO2_EF_READ_A_F_AND_RESET, DIO3_EF_READ_A_F_AND_RESET, DIO4_EF_READ_A_F_AND_RESET, DIO5_EF_READ_A_F_AND_RESET, DIO6_EF_READ_A_F_AND_RESET, DIO7_EF_READ_A_F_AND_RESET, DIO8_EF_READ_A_F_AND_RESET, DIO9_EF_READ_A_F_AND_RESET, DIO10_EF_READ_A_F_AND_RESET, DIO11_EF_READ_A_F_AND_RESET, DIO12_EF_READ_A_F_AND_RESET, DIO13_EF_READ_A_F_AND_RESET, DIO14_EF_READ_A_F_AND_RESET, DIO15_EF_READ_A_F_AND_RESET, DIO16_EF_READ_A_F_AND_RESET, DIO17_EF_READ_A_F_AND_RESET, DIO18_EF_READ_A_F_AND_RESET, DIO19_EF_READ_A_F_AND_RESET, DIO20_EF_READ_A_F_AND_RESET, DIO21_EF_READ_A_F_AND_RESET	3600, 3602, 3604, 3606, 3608, 3610, 3612, 3614, 3616, 3618, 3620, 3622, 3624, 3626, 3628, 3630, 3632, 3634, 3636, 3638, 3640, 3642

DIO#(0:21)_EF_READ_B_F
- Starting Address: 3700

Reads a floating point value. The meaning of value is dependent on selected feature index.

- Data type: FLOAT32 (type index = 3)
- Read-only

Expanded Names	Addresses
DIO0_EF_READ_B_F, DIO1_EF_READ_B_F, DIO2_EF_READ_B_F, DIO3_EF_READ_B_F, DIO4_EF_READ_B_F, DIO5_EF_READ_B_F, DIO6_EF_READ_B_F, DIO7_EF_READ_B_F, DIO8_EF_READ_B_F, DIO9_EF_READ_B_F, DIO10_EF_READ_B_F, DIO11_EF_READ_B_F, DIO12_EF_READ_B_F, DIO13_EF_READ_B_F, DIO14_EF_READ_B_F, DIO15_EF_READ_B_F, DIO16_EF_READ_B_F, DIO17_EF_READ_B_F, DIO18_EF_READ_B_F, DIO19_EF_READ_B_F, DIO20_EF_READ_B_F, DIO21_EF_READ_B_F	3700, 3702, 3704, 3706, 3708, 3710, 3712, 3714, 3716, 3718, 3720, 3722, 3724, 3726, 3728, 3730, 3732, 3734, 3736, 3738, 3740, 3742

DIO#(0:21)_EF_EASY_FREQUENCY_IN
- Starting Address: 45000

Deprecated. Automatically configures the associated DIO_EF for feature index 11. If already configured for index 11, reads the result and resets the DIO_EF.

- Data type: FLOAT32 (type index = 3)
- Readable and writable
- T8:
  - Minimum firmware version: 0.0123
- T7:

- Minimum firmware version: 1.0105

Expanded Names	Addresses
DIO0_EF_EASY_FREQUENCY_IN, DIO1_EF_EASY_FREQUENCY_IN,	45000, 45002,
DIO2_EF_EASY_FREQUENCY_IN, DIO3_EF_EASY_FREQUENCY_IN,	45004, 45006,
DIO4_EF_EASY_FREQUENCY_IN, DIO5_EF_EASY_FREQUENCY_IN,	45008, 45010,
DIO6_EF_EASY_FREQUENCY_IN, DIO7_EF_EASY_FREQUENCY_IN,	45012, 45014,
DIO8_EF_EASY_FREQUENCY_IN, DIO9_EF_EASY_FREQUENCY_IN,	45016, 45018,
DIO10_EF_EASY_FREQUENCY_IN, DIO11_EF_EASY_FREQUENCY_IN,	45020, 45022,
DIO12_EF_EASY_FREQUENCY_IN, DIO13_EF_EASY_FREQUENCY_IN,	45024, 45026,
DIO14_EF_EASY_FREQUENCY_IN, DIO15_EF_EASY_FREQUENCY_IN,	45028, 45030,
DIO16_EF_EASY_FREQUENCY_IN, DIO17_EF_EASY_FREQUENCY_IN,	45032, 45034,
DIO18_EF_EASY_FREQUENCY_IN, DIO19_EF_EASY_FREQUENCY_IN,	45036, 45038,
DIO20_EF_EASY_FREQUENCY_IN, DIO21_EF_EASY_FREQUENCY_IN	45040, 45042

All CONFIG TAGS:

Name	Start Address	Type	Access
POWER_ETHERNET	48003	UINT16	R/W
POWER_WIFI	48004	UINT16	R/W
POWER_AIN	48005	UINT16	R/W
POWER_LED	48006	UINT16	R/W
POWER_ETHERNET_DEFAULT	48053	UINT16	R/W
POWER_WIFI_DEFAULT	48054	UINT16	R/W
POWER_AIN_DEFAULT	48055	UINT16	R/W
POWER_LED_DEFAULT	48056	UINT16	R/W
IO_CONFIG_CHECK_FOR_FACTORY	49000	UINT32	R
IO_CONFIG_SET_DEFAULT_TO_CURRENT	49002	UINT32	W
IO_CONFIG_SET_DEFAULT_TO_FACTORY	49004	UINT32	W
IO_CONFIG_FACTORY_pREAD	49006	UINT32	R/W
IO_CONFIG_FACTORY_READ	49008	UINT32	R
IO_CONFIG_DEFAULT_pREAD	49010	UINT32	R/W
IO_CONFIG_DEFAULT_READ	49012	UINT32	R
IO_CONFIG_CURRENT_pREAD	49014	UINT32	R/W
IO_CONFIG_CURRENT_READ	49016	UINT32	R
IO_CONFIG_CHECK_FOR_DEFAULT	49018	UINT32	R
IO_CONFIG_CURRENT_CRC32	49020	UINT32	R
CLEANSE	49090	UINT32	W
CORE_TIMER	61520	UINT32	R
SYSTEM_TIMER_20HZ	61522	UINT32	R
WAIT_US_BLOCKING	61590	UINT32	R/W
IO_CONFIG_SET_CURRENT_TO_FACTORY	61990	UINT16	W
IO_CONFIG_SET_CURRENT_TO_DEFAULT	61991	UINT16	W
SYSTEM_REBOOT	61998	UINT32	W
SYSTEM_COUNTER_10KHZ	61502	UINT32	R
TEST	55100	UINT32	R
TEST_UINT16	55110	UINT16	R/W
TEST_UINT32	55120	UINT32	R/W
TEST_INT32	55122	INT32	R/W
TEST_FLOAT32	55124	FLOAT32	R/W
PRODUCT_ID	60000	FLOAT32	R
HARDWARE_VERSION	60002	FLOAT32	R
FIRMWARE_VERSION	60004	FLOAT32	R

BOOTLOADER_VERSION	60006	FLOAT32	R
WIFI_VERSION	60008	FLOAT32	R
HARDWARE_INSTALLED	60010	UINT32	R
LJMCOMMON_VERSION	60018	FLOAT32	R
SERIAL_NUMBER	60028	UINT32	R
INTERNAL_VS_VOLTS	60030	FLOAT32	R
INTERNAL_IS_AMPS	60032	FLOAT32	R
DEVICE_NAME_DEFAULT	60500	STRING	R/W
CURRENT_SOURCE_10UA_CAL_VALUE	1900	FLOAT32	R
CURRENT_SOURCE_200UA_CAL_VALUE	1902	FLOAT32	R
DEVICE_RESET_DBG_REG	60014	UINT32	R

POWER_ETHERNET
- Address: 48003

The current ON/OFF state of the Ethernet module.

- Data type: UINT16 (type index = 0)
- Readable and writable
- T8:
  - Minimum firmware version: 1.0013
- T7:
  - Minimum firmware version: 0.8600
- T4:
  - Minimum firmware version: 0.0100

POWER_WIFI
- Address: 48004

The current ON/OFF state of the WiFi module.

- Data type: UINT16 (type index = 0)
- Readable and writable
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 0.8600

POWER_AIN
- Address: 48005

The current ON/OFF state of the analog input module.

- Data type: UINT16 (type index = 0)
- Readable and writable
- T8:
  - Minimum firmware version: 1.0013
- T7:
  - Minimum firmware version: 0.8600

POWER_LED
- Address: 48006

Sets the LED operation: 0 = Off. Useful for lower power applications. 1 = normal. 2 = Lower power, LEDs will still blink but will normally be off. 3 = Reserved. 4 = Manual, in this mode the LEDs can be user

controlled.

- Data type: UINT16 (type index = 0)
- Readable and writable
- T8:
  - Minimum firmware version: 1.0013
- T7:
  - Minimum firmware version: 0.8600
- T4:
  - Minimum firmware version: 1.0008

POWER_ETHERNET_DEFAULT
- Address: 48053

The ON/OFF state of the Ethernet module after a power-cycle to the device. Provided to optionally reduce power consumption.

- Data type: UINT16 (type index = 0)
- Readable and writable
- T8:
  - Minimum firmware version: 1.0013
- T7:
  - Minimum firmware version: 0.8600
- T4:
  - Minimum firmware version: 0.0100

POWER_WIFI_DEFAULT
- Address: 48054

The ON/OFF state of the WiFi module after a power-cycle to the device. Provided to optionally reduce power consumption.

- Data type: UINT16 (type index = 0)
- Readable and writable
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 0.8600

POWER_AIN_DEFAULT
- Address: 48055

The ON/OFF state of the analog input module after a power-cycle to the device. Provided to optionally reduce power consumption.

- Data type: UINT16 (type index = 0)
- Readable and writable
- T8:
  - Minimum firmware version: 1.0013
- T7:
  - Minimum firmware version: 0.8600

POWER_LED_DEFAULT
- Address: 48056

The ON/OFF state of the LEDs after a power-cycle to the device.

- Data type: UINT16 (type index = 0)
- Readable and writable
- T8:
    - Minimum firmware version: 1.0013
- T7:
    - Minimum firmware version: 0.8600
- T4:
    - Minimum firmware version: 1.0008

IO_CONFIG_CHECK_FOR_FACTORY
- Address: 49000

Returns 1 if default settings are the same as factory settings.

- Data type: UINT32 (type index = 1)
- Read-only
- T8:
    - Minimum firmware version: 1.0012

IO_CONFIG_SET_DEFAULT_TO_CURRENT
- Address: 49002

Write a 1 to cause new default (reboot/power-up) values to be saved to flash. Current values are retrieved and saved as the new defaults. Systems affected: AIN, DIO, DAC, AIN_EF, DIO_EF.

- Data type: UINT32 (type index = 1)
- Write-only
- T8:
    - Minimum firmware version: 1.0012

IO_CONFIG_SET_DEFAULT_TO_FACTORY
- Address: 49004

Write a 1 to cause new default (reboot/power-up) values to be saved to flash. Factory values are retrieved and saved as the new defaults. Systems affected: AIN, DIO, DAC, AIN_EF, DIO_EF.

- Data type: UINT32 (type index = 1)
- Write-only
- T8:
    - Minimum firmware version: 1.0012

IO_CONFIG_FACTORY_pREAD
- Address: 49006

The address in the factory configuration data which reads will start from.

- Data type: UINT32 (type index = 1)
- Readable and writable
- T8:
    - Minimum firmware version: 1.0012

IO_CONFIG_FACTORY_READ
- Address: 49008

Data read from factory configuration

- Data type: UINT32 (type index = 1)
- Read-only

- T8:
    - Minimum firmware version: 1.0012

IO_CONFIG_DEFAULT_pREAD
- Address: 49010

The address in the configuration data which reads will start from.

- Data type: UINT32 (type index = 1)
- Readable and writable
- T8:
    - Minimum firmware version: 1.0012

IO_CONFIG_DEFAULT_READ
- Address: 49012

Data read from the startup configuration

- Data type: UINT32 (type index = 1)
- Read-only
- T8:
    - Minimum firmware version: 1.0012

IO_CONFIG_CURRENT_pREAD
- Address: 49014

The address in the current configuration data which reads will start from.

- Data type: UINT32 (type index = 1)
- Readable and writable
- T8:
    - Minimum firmware version: 1.0012

IO_CONFIG_CURRENT_READ
- Address: 49016

Data read from the current configuration

- Data type: UINT32 (type index = 1)
- Read-only
- T8:
    - Minimum firmware version: 1.0012

IO_CONFIG_CHECK_FOR_DEFAULT
- Address: 49018

Returns 1 if the current configuration matches the default configuration.

- Data type: UINT32 (type index = 1)
- Read-only
- T8:
    - Minimum firmware version: 1.0012

IO_CONFIG_CURRENT_CRC32
- Address: 49020

Collects the current IO configuration and calculates of a CRC32.

- Data type: UINT32 (type index = 1)

- Read-only
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0220

CLEANSE
- Address: 49090

Writing 0x5317052E to this register will trigger a system cleanse.

- Data type: UINT32 (type index = 1)
- Write-only

CORE_TIMER
- Address: 61520

Internal 32-bit system timer running at 1/2 core speed.

- Data type: UINT32 (type index = 1)
- Read-only
- This register may be streamed
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0071

SYSTEM_TIMER_20HZ
- Address: 61522

Internal 32-bit system timer running at 20Hz.

- Data type: UINT32 (type index = 1)
- Read-only
- This register may be streamed
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0071

WAIT_US_BLOCKING
- Address: 61590

Delays for x microseconds. Range is 0-100000. This operation is Blocking. While a blocking function is running no other registers can be read / written.

- Data type: UINT32 (type index = 1)
- Readable and writable
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0026

IO_CONFIG_SET_CURRENT_TO_FACTORY
- Address: 61990

Write a 1 to set current values to factory configuration. The factory values are retrieved from flash and written to the current configuration registers. Systems affected: AIN, DIO, DAC, AIN_EF, DIO_EF.

- Data type: UINT16 (type index = 0)
- Write-only

IO_CONFIG_SET_CURRENT_TO_DEFAULT
- Address: 61991

Write a 1 to set current values to default configuration. The default values are retrieved from flash and written to the current configuration registers, thus this behaves similar to reboot/power-up. Systems affected: AIN, DIO, DAC, AIN_EF, DIO_EF.

- Data type: UINT16 (type index = 0)
- Write-only

SYSTEM_REBOOT
- Address: 61998

Issues a device reboot. Must write 0x4C4Axxxx, where xxxx is number of 50ms ticks to wait before reboot. To reboot immediately write 0x4C4A0000 (d1279918080).

- Data type: UINT32 (type index = 1)
- Write-only
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 0.9419
- T4:
  - Minimum firmware version: 0.2020

SYSTEM_COUNTER_10KHZ
- Address: 61502

A 10 kHz counter synchronized to RTC_TIME_S. This register can be appended to RTC_TIME_S as the decimal portion to get 0.1 ms resolution.

- Data type: UINT32 (type index = 1)
- Read-only
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0229

TEST
- Address: 55100

A read of this test register should always return 0x00112233 or d1122867. If your software has the word swap quirk, you will incorrectly read 0x22330011 or 573767697. If your software has the address-1 quirk, a UINT16 (1-register) read from 55101 will incorrectly return 0x0011 (should read 0x2233). This register is unlike others, in that it allows you can read a single word from 55100 or 55101, or of course 2 words from 55100.

- Data type: UINT32 (type index = 1)
- Read-only
- Default value: 1122867

TEST_UINT16
- Address: 55110

Write a value and read back to test UINT16 operation. Default is 0x0011 or d17.

- Data type: UINT16 (type index = 0)
- Readable and writable
- Default value: 17

TEST_UINT32
- Address: 55120

Write a value and read back to test UINT32 operation. Default is 0x00112233 or d1122867. If your software has the word swap quirk, the default will incorrectly read 0x22330011 or 573767697.

- Data type: UINT32 (type index = 1)
- Readable and writable
- Default value: 1122867

TEST_INT32
- Address: 55122

Write a value and read back to test INT32 operation. Default is 0x8899AABB or d-2003195205. If your software has the word swap quirk, the default will incorrectly read 0xAABB8899 or -1430550375.

- Data type: INT32 (type index = 2)
- Readable and writable
- Default value: -2003195205

TEST_FLOAT32
- Address: 55124

Write a value and read back to test FLOAT32 operation. Default is 0xC61C3C00 or -9999.0. If your software has the word swap quirk, the default will incorrectly read 0x3C00C61C or 0.00786.

- Data type: FLOAT32 (type index = 3)
- Readable and writable
- Default value: -9999

PRODUCT_ID
- Address: 60000

The numeric identifier of the device. Such as 7 for a T7 / T7-Pro.

- Data type: FLOAT32 (type index = 3)
- Read-only

HARDWARE_VERSION
- Address: 60002

The hardware version of the device.

- Data type: FLOAT32 (type index = 3)
- Read-only

FIRMWARE_VERSION
- Address: 60004

The current firmware version installed on the main processor.

- Data type: FLOAT32 (type index = 3)
- Read-only

BOOTLOADER_VERSION
- Address: 60006

The bootloader version installed on the main processor.

- Data type: FLOAT32 (type index = 3)
- Read-only

WIFI_VERSION
- Address: 60008

The current firmware version of the WiFi module, if available.

- Data type: FLOAT32 (type index = 3)
- Read-only

HARDWARE_INSTALLED
- Address: 60010

Bitmask indicating installed hardware options. bit0: High Resolution ADC, bit1: WiFi, bit2: RTC, bit3: microSD.

- Data type: UINT32 (type index = 1)
- Read-only
- Default value: 0

LJMCOMMON_VERSION
- Address: 60018

The MAC address of the wired Ethernet module.

- Data type: FLOAT32 (type index = 3)
- Read-only

SERIAL_NUMBER
- Address: 60028

The serial number of the device.

- Data type: UINT32 (type index = 1)
- Read-only

INTERNAL_VS_VOLTS
- Address: 60030

T8 Only. Returns the power supply voltage.

- Data type: FLOAT32 (type index = 3)
- Read-only

INTERNAL_IS_AMPS
- Address: 60032

T8 Only. Returns the power supply current draw.

- Data type: FLOAT32 (type index = 3)
- Read-only

DEVICE_NAME_DEFAULT
- Address: 60500

Reads return the current device name. Writes update the default and current device name. A reboot is necessary to update the name reported by NBNS. Up to 49 characters, cannot contain periods.

- Data type: STRING (type index = 98)
- Readable and writable

CURRENT_SOURCE_10UA_CAL_VALUE
- Address: 1900

Fixed current source value in Amps for the 10UA terminal. This value is stored during factory calibration, it is not a current reading. Using the equation V=IR, with a known current and voltage, it is possible to calculate resistance of RTDs.

- Data type: FLOAT32 (type index = 3)
- Read-only

CURRENT_SOURCE_200UA_CAL_VALUE
- Address: 1902

Fixed current source value in Amps for the 200UA terminal. This value is stored during factory calibration, it is not a current reading. Using the equation V=IR, with a known current and voltage, it is possible to calculate resistance of RTDs.

- Data type: FLOAT32 (type index = 3)
- Read-only

DEVICE_RESET_DBG_REG
- Address: 60014

Bitmask register that indicates why a device was last re-set. bit0: Power-on reset, bit1: Brown-out, bit2: Wake from idle, bit3: Wake from sleep, bit4: watchdog timer time-out, bit5: reserved, bit6: Software reset, bit7: external reset (MCLR), bit8: Voltage regulator standby, bit9: Configuration mismatch.

- Data type: UINT32 (type index = 1)
- Read-only
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0281
- T4:
  - Minimum firmware version: 1.0023

All ETHERNET TAGS:

Name	Start Address	Type	Access
POWER_ETHERNET	48003	UINT16	R/W
POWER_ETHERNET_DEFAULT	48053	UINT16	R/W
ETHERNET_IP	49100	UINT32	R
ETHERNET_SUBNET	49102	UINT32	R
ETHERNET_GATEWAY	49104	UINT32	R
ETHERNET_DNS	49106	UINT32	R
ETHERNET_ALTDNS	49108	UINT32	R
ETHERNET_DHCP_ENABLE	49110	UINT16	R
ETHERNET_UDP_DISCOVERY_ONLY	49115	UINT16	R
ETHERNET_IP_DEFAULT	49150	UINT32	R/W
ETHERNET_SUBNET_DEFAULT	49152	UINT32	R/W
ETHERNET_GATEWAY_DEFAULT	49154	UINT32	R/W
ETHERNET_DNS_DEFAULT	49156	UINT32	R/W
ETHERNET_ALTDNS_DEFAULT	49158	UINT32	R/W

ETHERNET_DHCP_ENABLE_DEFAULT	49160	UINT16	R/W
ETHERNET_UDP_DISCOVERY_ONLY_DEFAULT	49165	UINT16	R
ETHERNET_APPLY_SETTINGS	49190	UINT32	W
ETHERNET_MAC	60020	UINT64	R

POWER_ETHERNET
- Address: 48003

The current ON/OFF state of the Ethernet module.

- Data type: UINT16 (type index = 0)
- Readable and writable
- T8:
  - Minimum firmware version: 1.0013
- T7:
  - Minimum firmware version: 0.8600
- T4:
  - Minimum firmware version: 0.0100

POWER_ETHERNET_DEFAULT
- Address: 48053

The ON/OFF state of the Ethernet module after a power-cycle to the device. Provided to optionally reduce power consumption.

- Data type: UINT16 (type index = 0)
- Readable and writable
- T8:
  - Minimum firmware version: 1.0013
- T7:
  - Minimum firmware version: 0.8600
- T4:
  - Minimum firmware version: 0.0100

ETHERNET_IP
- Address: 49100

Read the current IP address of wired Ethernet.

- Data type: UINT32 (type index = 1)
- Read-only

ETHERNET_SUBNET
- Address: 49102

Read the current subnet of wired Ethernet.

- Data type: UINT32 (type index = 1)
- Read-only

ETHERNET_GATEWAY
- Address: 49104

Read the current gateway of wired Ethernet.

- Data type: UINT32 (type index = 1)
- Read-only

ETHERNET_DNS
- Address: 49106

Read the current DNS of wired Ethernet.

- Data type: UINT32 (type index = 1)
- Read-only

ETHERNET_ALTDNS
- Address: 49108

Read the current Alt DNS of wired Ethernet.

- Data type: UINT32 (type index = 1)
- Read-only

ETHERNET_DHCP_ENABLE
- Address: 49110

Read the current Enabled/Disabled state of Ethernet DHCP.

- Data type: UINT16 (type index = 0)
- Read-only

ETHERNET_UDP_DISCOVERY_ONLY
- Address: 49115

Restricts which Modbus operations are allowed over UDP. When set to 1, only the registers needed for discovering units can be read and any other operation will throw an error.

- Data type: UINT16 (type index = 0)
- Read-only
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0284

ETHERNET_IP_DEFAULT
- Address: 49150

The IP address of wired Ethernet after a power-cycle to the device.

- Data type: UINT32 (type index = 1)
- Readable and writable

ETHERNET_SUBNET_DEFAULT
- Address: 49152

The subnet of wired Ethernet after a power-cycle to the device.

- Data type: UINT32 (type index = 1)
- Readable and writable

ETHERNET_GATEWAY_DEFAULT
- Address: 49154

The gateway of wired Ethernet after a power-cycle to the device.

- Data type: UINT32 (type index = 1)
- Readable and writable

ETHERNET_DNS_DEFAULT
- Address: 49156

The DNS of wired Ethernet after a power-cycle to the device.

- Data type: UINT32 (type index = 1)
- Readable and writable

ETHERNET_ALTDNS_DEFAULT
- Address: 49158

The Alt DNS of wired Ethernet after a power-cycle to the device.

- Data type: UINT32 (type index = 1)
- Readable and writable

ETHERNET_DHCP_ENABLE_DEFAULT
- Address: 49160

The Enabled/Disabled state of Ethernet DHCP after a power-cycle to the device.

- Data type: UINT16 (type index = 0)
- Readable and writable

ETHERNET_UDP_DISCOVERY_ONLY_DEFAULT
- Address: 49165

The Enabled/Disabled state of ETHERNET_UDP_DISCOVERY_ONLY after a power-cycle to the device.

- Data type: UINT16 (type index = 0)
- Read-only
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0284

ETHERNET_APPLY_SETTINGS
- Address: 49190

Writing 1 to this register power-cycles Ethernet. It tells the device to waits 1s before turning off Ethernet and then 500ms before turning it back on.

- Data type: UINT32 (type index = 1)
- Write-only
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0103
- T4:
  - Minimum firmware version: 0.2000

ETHERNET_MAC
- Address: 60020

The MAC address of the wired Ethernet module.

- Data type: UINT64 (type index = N/A)
- Read-only

All WIFI TAGS:

Name	Start Address	Type	Access
POWER_WIFI	48004	UINT16	R/W
POWER_WIFI_DEFAULT	48054	UINT16	R/W
WIFI_IP	49200	UINT32	R
WIFI_SUBNET	49202	UINT32	R
WIFI_GATEWAY	49204	UINT32	R
WIFI_DHCP_ENABLE	49210	UINT16	R
WIFI_UDP_DISCOVERY_ONLY	49215	UINT16	R
WIFI_SSID	49300	STRING	R
WIFI_IP_DEFAULT	49250	UINT32	R/W
WIFI_SUBNET_DEFAULT	49252	UINT32	R/W
WIFI_GATEWAY_DEFAULT	49254	UINT32	R/W
WIFI_DHCP_ENABLE_DEFAULT	49260	UINT16	R/W
WIFI_UDP_DISCOVERY_ONLY_DEFAULT	49265	UINT16	R
WIFI_SSID_DEFAULT	49325	STRING	R/W
WIFI_PASSWORD_DEFAULT	49350	STRING	W
WIFI_APPLY_SETTINGS	49400	UINT32	W
WIFI_FIRMWARE_UPDATE_TO_VERSIONX	49402	FLOAT32	W
WIFI_STATUS	49450	UINT32	R
WIFI_RSSI	49452	FLOAT32	R
WIFI_FIRMWARE_UPDATE_STATUS	49454	UINT32	R
WIFI_MAC	60024	UINT64	R
WIFI_START_ADHOC	49410	UINT32	W
WIFI_SCAN_START	49406	UINT32	W
WIFI_SCAN_NUM_BYTES	49486	UINT32	R
WIFI_SCAN_DATA	49488	BYTE	R

POWER_WIFI
- Address: 48004

The current ON/OFF state of the WiFi module.

- Data type: UINT16 (type index = 0)
- Readable and writable
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 0.8600

POWER_WIFI_DEFAULT
- Address: 48054

The ON/OFF state of the WiFi module after a power-cycle to the device. Provided to optionally reduce power consumption.

- Data type: UINT16 (type index = 0)
- Readable and writable
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 0.8600

WIFI_IP
- Address: 49200

Read the current IP address of WiFi.

- Data type: UINT32 (type index = 1)
- Read-only

WIFI_SUBNET
- Address: 49202

Read the current subnet of WiFi.

- Data type: UINT32 (type index = 1)
- Read-only

WIFI_GATEWAY
- Address: 49204

Read the current gateway of WiFi.

- Data type: UINT32 (type index = 1)
- Read-only

WIFI_DHCP_ENABLE
- Address: 49210

Read the current Enabled/Disabled state of WiFi DHCP.

- Data type: UINT16 (type index = 0)
- Read-only
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 0.9014

WIFI_UDP_DISCOVERY_ONLY
- Address: 49215

Restricts which Modbus operations are allowed over UDP. When set to 1, only the registers needed for discovering units can be read and any other operation will throw an error.

- Data type: UINT16 (type index = 0)
- Read-only
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0284

WIFI_SSID
- Address: 49300

Read the current SSID (network name) of WiFi

- Data type: STRING (type index = 98)
- Read-only

WIFI_IP_DEFAULT
- Address: 49250

The new IP address of WiFi. Use WIFI_APPLY_SETTINGS.

- Data type: UINT32 (type index = 1)
- Readable and writable

WIFI_SUBNET_DEFAULT
- Address: 49252

The new subnet of WiFi. Use WIFI_APPLY_SETTINGS.

- Data type: UINT32 (type index = 1)
- Readable and writable

WIFI_GATEWAY_DEFAULT
- Address: 49254

The new gateway of WiFi. Use WIFI_APPLY_SETTINGS.

- Data type: UINT32 (type index = 1)
- Readable and writable

WIFI_DHCP_ENABLE_DEFAULT
- Address: 49260

The new Enabled/Disabled state of WiFi DHCP. Use WIFI_APPLY_SETTINGS

- Data type: UINT16 (type index = 0)
- Readable and writable
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 0.9014

WIFI_UDP_DISCOVERY_ONLY_DEFAULT
- Address: 49265

The Enabled/Disabled state of WIFI_UDP_DISCOVERY_ONLY after a power-cycle to the device.

- Data type: UINT16 (type index = 0)
- Read-only
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0284

WIFI_SSID_DEFAULT
- Address: 49325

The new SSID (network name) of WiFi. Use WIFI_APPLY_SETTINGS.

- Data type: STRING (type index = 98)
- Readable and writable

WIFI_PASSWORD_DEFAULT
- Address: 49350

Write the password for the WiFi network, then use WIFI_APPLY_SETTINGS.

- Data type: STRING (type index = 98)
- Write-only

WIFI_APPLY_SETTINGS
- Address: 49400

Apply all new WiFi settings: IP, Subnet, Gateway, DHCP, SSID, Password. 1=Apply

- Data type: UINT32 (type index = 1)
- Write-only

WIFI_FIRMWARE_UPDATE_TO_VERSIONX
- Address: 49402

Start an update by using USB or Ethernet to write the desired version to update to.

- Data type: FLOAT32 (type index = 3)
- Write-only
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 0.9206

WIFI_STATUS
- Address: 49450

Status Codes: ASSOCIATED = 2900, ASSOCIATING = 2901, ASSOCIATION_FAILED = 2902, UNPOWERED = 2903, BOOTING = 2904, START_FAILED = 2905, APPLYING_SETTINGS = 2906, DHCP_STARTED = 2907, OTHER = 2909

- Data type: UINT32 (type index = 1)
- Read-only
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 0.7500

Constant	Value
ASSOCIATED	2900
ASSOCIATING	2901
ASSOCIATION_FAILED	2902
UNPOWERED	2903
BOOTING	2904
START_FAILED	2905
APPLYING_SETTINGS	2906
DHCP_STARTED	2907
OTHER	2909

WIFI_RSSI
- Address: 49452

WiFi RSSI (signal strength). Typical values are -40 for very good, and -75 for very weak. The T7 microcontroller only gets a new RSSI value from the WiFi module when WiFi communication occurs.

- Data type: FLOAT32 (type index = 3)
- Read-only
- T8:

    - Minimum firmware version: 0.0123
- T7:
    - Minimum firmware version: 0.8000

WIFI_FIRMWARE_UPDATE_STATUS
- Address: 49454

CONFIGURING = 2920, IN_PROGRESS = 2921, REBOOTING = 2923, UPDATE_SUCCESS = 2924, UPDATE_FAILED = 2925

- Data type: UINT32 (type index = 1)
- Read-only
- T8:
    - Minimum firmware version: 0.0123
- T7:
    - Minimum firmware version: 0.9206

Constant	Value
CONFIGURING	2920
IN_PROGRESS	2921
REBOOTING	2923
UPDATE_SUCCESS	2924
UPDATE_FAILED	2925

WIFI_MAC
- Address: 60024

The MAC address of the WiFi module.

- Data type: UINT64 (type index = N/A)
- Read-only

WIFI_START_ADHOC
- Address: 49410

Currently unused.

- Data type: UINT32 (type index = 1)
- Write-only
- T8:
    - Minimum firmware version: 0.0123
- T7:
    - Minimum firmware version: 0.9308

WIFI_SCAN_START
- Address: 49406

Currently unused.

- Data type: UINT32 (type index = 1)
- Write-only
- T8:
    - Minimum firmware version: 0.0123
- T7:
    - Minimum firmware version: 0.8700

WIFI_SCAN_NUM_BYTES
- Address: 49486

Currently unused.

- Data type: UINT32 (type index = 1)
- Read-only
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 0.8700

WIFI_SCAN_DATA
- Address: 49488

Currently unused.

- Data type: BYTE (type index = 99)
- Read-only
- This register is a Buffer Register
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 0.8700

All RTC TAGS:

Name	Start Address	Type	Access
SNTP_UPDATE_INTERVAL	49702	UINT32	R/W
RTC_TIME_S	61500	UINT32	R
RTC_SET_TIME_S	61504	UINT32	W
RTC_SET_TIME_SNTP	61506	UINT32	W
RTC_TIME_CALENDAR	61510	UINT16	R

SNTP_UPDATE_INTERVAL
- Address: 49702

Sets the SNTP retry time in seconds. A value of zero will disable SNTP(0=default). Values must be 10 or greater.

- Data type: UINT32 (type index = 1)
- Readable and writable
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0150
- T4:
  - Minimum firmware version: 0.2020

RTC_TIME_S
- Address: 61500

Read the current time in seconds since Jan, 1970, aka Epoch or Unix time. This value is calculated from the 80 MHz crystal, not the RTC 32 kHz crystal. Non-pro devices do not have a real time clock, so the reported time is either seconds since device startup or the time reported by the network time protocol over Ethernet. Pro devices have a real time clock that will be used to initialize this register at startup, the time can then be updated by NTP.

- Data type: UINT32 (type index = 1)
- Read-only
- T8:
    - Minimum firmware version: 0.0123
- T7:
    - Minimum firmware version: 1.0128
- T4:
    - Minimum firmware version: 0.2020

RTC_SET_TIME_S
- Address: 61504

Write a new timestamp to the RTC in seconds since Jan, 1970, aka Epoch or Unix timestamp.

- Data type: UINT32 (type index = 1)
- Write-only
- T8:
    - Minimum firmware version: 0.0123
- T7:
    - Minimum firmware version: 1.0128
- T4:
    - Minimum firmware version: 0.2020

RTC_SET_TIME_SNTP
- Address: 61506

Write any value to instruct the T7 to update its clock from a SNTP server. Requires that SNTP_UPDATE_INTERVAL is non-zero.

- Data type: UINT32 (type index = 1)
- Write-only
- T8:
    - Minimum firmware version: 0.0123
- T7:
    - Minimum firmware version: 1.0128
- T4:
    - Minimum firmware version: 0.2020

RTC_TIME_CALENDAR
- Address: 61510

Read six consecutive addresses of type UINT16, starting with this address. The result will be in year, month, day, hour, minute, second calendar format. i.e. [2014, 10, 21, 18, 55, 35]

- Data type: UINT16 (type index = 0)
- Read-only
- T8:
    - Minimum firmware version: 0.0123
- T7:
    - Minimum firmware version: 1.0128
- T4:
    - Minimum firmware version: 0.2020

Name	Start Address	Type	Access
USER_RAM#(0:38)_F32	46000	FLOAT32	R/W
USER_RAM#(0:8)_I32	46080	INT32	R/W
USER_RAM#(0:38)_U32	46100	UINT32	R/W
USER_RAM#(0:18)_U16	46180	UINT16	R/W
USER_RAM_FIFO#(0:2)_DATA_U16	47000	UINT16	R/W
USER_RAM_FIFO#(0:2)_DATA_U32	47010	UINT32	R/W
USER_RAM_FIFO#(0:2)_DATA_I32	47020	INT32	R/W
USER_RAM_FIFO#(0:2)_DATA_F32	47030	FLOAT32	R/W
USER_RAM_FIFO#(0:2)_ALLOCATE_NUM_BYTES	47900	UINT32	R/W
USER_RAM_FIFO#(0:2)_NUM_BYTES_IN_FIFO	47910	UINT32	R
USER_RAM_FIFO#(0:2)_EMPTY	47930	UINT32	W
BATTERY_RAM#(0:15)	61200	UINT32	R/W

USER_RAM#(0:38)_F32
- Starting Address: 46000

Generic RAM registers. Useful for passing data between a host computer and a Lua script. Will not return an error if alternate data types are used.

- Data type: FLOAT32 (type index = 3)
- Readable and writable
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0023

Expanded Names	Addresses
USER_RAM0_F32, USER_RAM1_F32, USER_RAM2_F32, USER_RAM3_F32, USER_RAM4_F32, USER_RAM5_F32, USER_RAM6_F32, USER_RAM7_F32, USER_RAM8_F32, USER_RAM9_F32, USER_RAM10_F32, USER_RAM11_F32, USER_RAM12_F32, USER_RAM13_F32, USER_RAM14_F32, USER_RAM15_F32, USER_RAM16_F32, USER_RAM17_F32, USER_RAM18_F32, USER_RAM19_F32, USER_RAM20_F32, USER_RAM21_F32, USER_RAM22_F32, USER_RAM23_F32, USER_RAM24_F32, USER_RAM25_F32, USER_RAM26_F32, USER_RAM27_F32, USER_RAM28_F32, USER_RAM29_F32, USER_RAM30_F32, USER_RAM31_F32, USER_RAM32_F32, USER_RAM33_F32, USER_RAM34_F32, USER_RAM35_F32, USER_RAM36_F32, USER_RAM37_F32, USER_RAM38_F32	46000, 46002, 46004, 46006, 46008, 46010, 46012, 46014, 46016, 46018, 46020, 46022, 46024, 46026, 46028, 46030, 46032, 46034, 46036, 46038, 46040, 46042, 46044, 46046, 46048, 46050, 46052, 46054, 46056, 46058, 46060, 46062, 46064, 46066, 46068, 46070, 46072, 46074, 46076

USER_RAM#(0:8)_I32
- Starting Address: 46080

Generic RAM registers. Useful for passing data between a host computer and a Lua script. Will not return an error if alternate data types are used.

- Data type: INT32 (type index = 2)
- Readable and writable
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0162

Expanded Names	Addresses

USER_RAM0_I32, USER_RAM1_I32, USER_RAM2_I32, USER_RAM3_I32, USER_RAM4_I32, USER_RAM5_I32, USER_RAM6_I32, USER_RAM7_I32, USER_RAM8_I32	46080, 46082, 46084, 46086, 46088, 46090, 46092, 46094, 46096

USER_RAM#(0:38)_U32
- Starting Address: 46100

Generic RAM registers. Useful for passing data between a host computer and a Lua script. Will not return an error if alternate data types are used.

- Data type: UINT32 (type index = 1)
- Readable and writable
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0162

Expanded Names	Addresses
USER_RAM0_U32, USER_RAM1_U32, USER_RAM2_U32, USER_RAM3_U32, USER_RAM4_U32, USER_RAM5_U32, USER_RAM6_U32, USER_RAM7_U32, USER_RAM8_U32, USER_RAM9_U32, USER_RAM10_U32, USER_RAM11_U32, USER_RAM12_U32, USER_RAM13_U32, USER_RAM14_U32, USER_RAM15_U32, USER_RAM16_U32, USER_RAM17_U32, USER_RAM18_U32, USER_RAM19_U32, USER_RAM20_U32, USER_RAM21_U32, USER_RAM22_U32, USER_RAM23_U32, USER_RAM24_U32, USER_RAM25_U32, USER_RAM26_U32, USER_RAM27_U32, USER_RAM28_U32, USER_RAM29_U32, USER_RAM30_U32, USER_RAM31_U32, USER_RAM32_U32, USER_RAM33_U32, USER_RAM34_U32, USER_RAM35_U32, USER_RAM36_U32, USER_RAM37_U32, USER_RAM38_U32	46100, 46102, 46104, 46106, 46108, 46110, 46112, 46114, 46116, 46118, 46120, 46122, 46124, 46126, 46128, 46130, 46132, 46134, 46136, 46138, 46140, 46142, 46144, 46146, 46148, 46150, 46152, 46154, 46156, 46158, 46160, 46162, 46164, 46166, 46168, 46170, 46172, 46174, 46176

USER_RAM#(0:18)_U16
- Starting Address: 46180

Generic RAM registers. Useful for passing data between a host computer and a Lua script. Will not return an error if alternate data types are used.

- Data type: UINT16 (type index = 0)
- Readable and writable
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0162

Expanded Names	Addresses
USER_RAM0_U16, USER_RAM1_U16, USER_RAM2_U16, USER_RAM3_U16, USER_RAM4_U16, USER_RAM5_U16, USER_RAM6_U16, USER_RAM7_U16, USER_RAM8_U16, USER_RAM9_U16, USER_RAM10_U16, USER_RAM11_U16, USER_RAM12_U16, USER_RAM13_U16, USER_RAM14_U16, USER_RAM15_U16, USER_RAM16_U16, USER_RAM17_U16, USER_RAM18_U16	46180, 46181, 46182, 46183, 46184, 46185, 46186, 46187, 46188, 46189, 46190, 46191, 46192, 46193, 46194, 46195, 46196, 46197, 46198

USER_RAM_FIFO#(0:2)_DATA_U16
- Starting Address: 47000

Generic FIFO buffer. Useful for passing ORDERED or SEQUENTIAL data between various endpoints, such as between a host and a Lua script. Use up to 4 FIFO buffers simultaneously->1 of each data type, all 4 different data types, or a mixture. e.g. FIFO0_DATA_U16 points to the same memory as other FIFO0 registers, such that there are a total of 4 memory blocks: FIFO0, FIFO1, FIFO2 and FIFO3. It is possible to

write into a FIFO buffer using a different datatype than is being used to read out of it. This register is a buffer. Underrun behavior - throws an error.

- Data type: UINT16 (type index = 0)
- Readable and writable
- Default value: 0
- This register is a Buffer Register
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0163

Expanded Names	Addresses
USER_RAM_FIFO0_DATA_U16, USER_RAM_FIFO1_DATA_U16, USER_RAM_FIFO2_DATA_U16	47000, 47001, 47002

USER_RAM_FIFO#(0:2)_DATA_U32
- Starting Address: 47010

Generic FIFO buffer. Useful for passing ORDERED or SEQUENTIAL data between various endpoints, such as between a host and a Lua script. Use up to 4 FIFO buffers simultaneously->1 of each data type, all 4 different data types, or a mixture. e.g. FIFO0_DATA_U16 points to the same memory as other FIFO0 registers, such that there are a total of 4 memory blocks: FIFO0, FIFO1, FIFO2 and FIFO3. It is possible to write into a FIFO buffer using a different datatype than is being used to read out of it. This register is a buffer. Underrun behavior - throws an error.

- Data type: UINT32 (type index = 1)
- Readable and writable
- Default value: 0
- This register is a Buffer Register
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0163

Expanded Names	Addresses
USER_RAM_FIFO0_DATA_U32, USER_RAM_FIFO1_DATA_U32, USER_RAM_FIFO2_DATA_U32	47010, 47012, 47014

USER_RAM_FIFO#(0:2)_DATA_I32
- Starting Address: 47020

Generic FIFO buffer. Useful for passing ORDERED or SEQUENTIAL data between various endpoints, such as between a host and a Lua script. Use up to 4 FIFO buffers simultaneously->1 of each data type, all 4 different data types, or a mixture. e.g. FIFO0_DATA_U16 points to the same memory as other FIFO0 registers, such that there are a total of 4 memory blocks: FIFO0, FIFO1, FIFO2 and FIFO3. It is possible to write into a FIFO buffer using a different datatype than is being used to read out of it. This register is a buffer. Underrun behavior - throws an error.

- Data type: INT32 (type index = 2)
- Readable and writable
- Default value: 0
- This register is a Buffer Register
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0163

Expanded Names	Addresses
USER_RAM_FIFO0_DATA_I32, USER_RAM_FIFO1_DATA_I32, USER_RAM_FIFO2_DATA_I32	47020, 47022, 47024

USER_RAM_FIFO#(0:2)_DATA_F32
- Starting Address: 47030

Generic FIFO buffer. Useful for passing ORDERED or SEQUENTIAL data between various endpoints, such as between a host and a Lua script. Use up to 4 FIFO buffers simultaneously->1 of each data type, all 4 different data types, or a mixture. e.g. FIFO0_DATA_U16 points to the same memory as other FIFO0 registers, such that there are a total of 4 memory blocks: FIFO0, FIFO1, FIFO2 and FIFO3. It is possible to write into a FIFO buffer using a different datatype than is being used to read out of it. This register is a buffer. Underrun behavior - throws an error.

- Data type: FLOAT32 (type index = 3)
- Readable and writable
- Default value: 0
- This register is a Buffer Register
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0163

Expanded Names	Addresses
USER_RAM_FIFO0_DATA_F32, USER_RAM_FIFO1_DATA_F32, USER_RAM_FIFO2_DATA_F32	47030, 47032, 47034

USER_RAM_FIFO#(0:2)_ALLOCATE_NUM_BYTES
- Starting Address: 47900

Allocate memory for a FIFO buffer. Number of bytes should be sufficient to store users max transfer array size. Note that FLOAT32, INT32, and UINT32 require 4 bytes per value, and UINT16 require 2 bytes per value. Maximum size is limited by available memory. Care should be taken to conserve enough memory for other operations such as AIN_EF, Lua, Stream etc.

- Data type: UINT32 (type index = 1)
- Readable and writable
- Default value: 0
- This register uses system RAM. The maximum RAM is 64KB. For more information, see 4.4 RAM
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0163

Expanded Names	Addresses
USER_RAM_FIFO0_ALLOCATE_NUM_BYTES, USER_RAM_FIFO1_ALLOCATE_NUM_BYTES, USER_RAM_FIFO2_ALLOCATE_NUM_BYTES	47900, 47902, 47904

USER_RAM_FIFO#(0:2)_NUM_BYTES_IN_FIFO
- Starting Address: 47910

Poll this register to see when new data is available/ready. Each read of the FIFO buffer decreases this value, and each write to the FIFO buffer increases this value. At any point in time, the following equation holds: Nbytes = Nwritten - Nread.

- Data type: UINT32 (type index = 1)
- Read-only
- Default value: 0

- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0163

Expanded Names	Addresses
USER_RAM_FIFO0_NUM_BYTES_IN_FIFO, USER_RAM_FIFO1_NUM_BYTES_IN_FIFO, USER_RAM_FIFO2_NUM_BYTES_IN_FIFO	47910, 47912, 47914

USER_RAM_FIFO#(0:2)_EMPTY
- Starting Address: 47930

Write any value to this register to efficiently empty, flush, or otherwise clear data from the FIFO.

- Data type: UINT32 (type index = 1)
- Write-only
- Default value: 0
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0163

Expanded Names	Addresses
USER_RAM_FIFO0_EMPTY, USER_RAM_FIFO1_EMPTY, USER_RAM_FIFO2_EMPTY	47930, 47932, 47934

BATTERY_RAM#(0:15)
- Starting Address: 61200

Saves or recalls data to or from the battery-backed SRAM. Only available on T7-Pros.

- Data type: UINT32 (type index = 1)
- Readable and writable
- Default value: 0
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0000

Expanded Names	Addresses
BATTERY_RAM0, BATTERY_RAM1, BATTERY_RAM2, BATTERY_RAM3, BATTERY_RAM4, BATTERY_RAM5, BATTERY_RAM6, BATTERY_RAM7, BATTERY_RAM8, BATTERY_RAM9, BATTERY_RAM10, BATTERY_RAM11, BATTERY_RAM12, BATTERY_RAM13, BATTERY_RAM14, BATTERY_RAM15	61200, 61202, 61204, 61206, 61208, 61210, 61212, 61214, 61216, 61218, 61220, 61222, 61224, 61226, 61228, 61230

All FILE_IO TAGS:

Name	Start Address	Type	Access
FILE_IO_DIR_CHANGE	60600	UINT16	W
FILE_IO_DIR_CURRENT	60601	UINT16	W
FILE_IO_DIR_MAKE	60602	UINT16	W
FILE_IO_DIR_REMOVE	60603	UINT16	W
FILE_IO_DIR_FIRST	60610	UINT16	W
FILE_IO_DIR_NEXT	60611	UINT16	W
FILE_IO_OPEN	60620	UINT16	W
FILE_IO_CLOSE	60621	UINT16	W

FILE_IO_DELETE	60622	UINT16	W
FILE_IO_ATTRIBUTES	60623	UINT16	R
FILE_IO_SIZE_BYTES	60628	UINT32	R
FILE_IO_DISK_SECTOR_SIZE_BYTES	60630	UINT32	R
FILE_IO_DISK_SECTORS_PER_CLUSTER	60632	UINT32	R
FILE_IO_DISK_TOTAL_CLUSTERS	60634	UINT32	R
FILE_IO_DISK_FREE_CLUSTERS	60636	UINT32	R
FILE_IO_DISK_FORMAT_INDEX	60638	UINT32	R
FILE_IO_PATH_WRITE_LEN_BYTES	60640	UINT32	W
FILE_IO_PATH_READ_LEN_BYTES	60642	UINT32	R
FILE_IO_PATH_WRITE	60650	BYTE	W
FILE_IO_PATH_READ	60652	BYTE	R
FILE_IO_WRITE	60654	BYTE	W
FILE_IO_READ	60656	BYTE	R
FILE_IO_LUA_SWITCH_FILE	60662	UINT32	R/W

FILE_IO_DIR_CHANGE
- Address: 60600

Write any value to this register to change the current working directory (CWD). Must first designate which directory to open by writing to FILE_IO_PATH_WRITE_LEN_BYTES then FILE_IO_PATH_WRITE.

- Data type: UINT16 (type index = 0)
- Write-only
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0156

FILE_IO_DIR_CURRENT
- Address: 60601

Write any value to this register to load the current working directory into FILE_IO_PATH_READ, and its length into FILE_IO_PATH_READ_LEN_BYTES. Can be used to identify current position in a file tree.

- Data type: UINT16 (type index = 0)
- Write-only
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0134

FILE_IO_DIR_MAKE
- Address: 60602

Unimplemented.

- Data type: UINT16 (type index = 0)
- Write-only
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 99.0000

FILE_IO_DIR_REMOVE

- Address: 60603

Unimplemented.

- Data type: UINT16 (type index = 0)
- Write-only
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 99.0000

FILE_IO_DIR_FIRST
- Address: 60610

Write any value to this register to initiate iteration through files and directories in the CWD. Typical sequence: FILE_IO_DIR_FIRST(W), then loop through: [FILE_IO_NAME_READ_LEN(R), FILE_IO_NAME_READ(R), FILE_IO_ATTRIBUTES(R), FILE_IO_SIZE_BYTES(R), FILE_IO_DIR_NEXT(W)] ..until any of the following errors: FILE_IO_END_OF_CWD (2966), FILE_IO_INVALID_OBJECT (2809), or FILE_IO_NOT_FOUND (2960).

- Data type: UINT16 (type index = 0)
- Write-only
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0134

FILE_IO_DIR_NEXT
- Address: 60611

Write any value to this register to continue iteration through files and directories in the CWD.

- Data type: UINT16 (type index = 0)
- Write-only
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0134

FILE_IO_OPEN
- Address: 60620

Write any value to this register to open a file. Must first designate which file to open by writing to FILE_IO_PATH_WRITE_LEN_BYTES then FILE_IO_PATH_WRITE.

- Data type: UINT16 (type index = 0)
- Write-only
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0134

FILE_IO_CLOSE
- Address: 60621

Write any value to this register to close the open file.

- Data type: UINT16 (type index = 0)

- Write-only
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0134

FILE_IO_DELETE
- Address: 60622

Write any value to this register to delete the active file. Must first designate which file to delete by writing to FILE_IO_PATH_WRITE_LEN_BYTES then FILE_IO_PATH_WRITE.

- Data type: UINT16 (type index = 0)
- Write-only
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0134

FILE_IO_ATTRIBUTES
- Address: 60623

Used to differentiate files from directories/folders. Bitmask: Bit0: Reserved, Bit1: Reserved, Bit2: Reserved, Bit3: Reserved, Bit4: 1=Directory, Bit5: 1=File.

- Data type: UINT16 (type index = 0)
- Read-only
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0134

FILE_IO_SIZE_BYTES
- Address: 60628

The size of the file in bytes. Directories have 0 size.

- Data type: UINT32 (type index = 1)
- Read-only
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0134

FILE_IO_DISK_SECTOR_SIZE_BYTES
- Address: 60630

The size of each sector in the SD card in bytes. In Windows this is called the Allocation Size.

- Data type: UINT32 (type index = 1)
- Read-only
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0134

FILE_IO_DISK_SECTORS_PER_CLUSTER
- Address: 60632

The number of sectors in each cluster. Captured on read of FILE_IO_DISK_SECTOR_SIZE_BYTES.

- Data type: UINT32 (type index = 1)
- Read-only
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0134

FILE_IO_DISK_TOTAL_CLUSTERS
- Address: 60634

The total number of clusters in the SD card. Captured on read of FILE_IO_DISK_SECTOR_SIZE_BYTES.

- Data type: UINT32 (type index = 1)
- Read-only
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0134

FILE_IO_DISK_FREE_CLUSTERS
- Address: 60636

Free (available) clusters in the SD card. Used to determine free space. Captured on read of FILE_IO_DISK_SECTOR_SIZE_BYTES.

- Data type: UINT32 (type index = 1)
- Read-only
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0134

FILE_IO_DISK_FORMAT_INDEX
- Address: 60638

Used to determine the format of the SD card. 0=None or Unknown, 1=FAT12, 2=FAT16(Windows FAT), 3=FAT32

- Data type: UINT32 (type index = 1)
- Read-only
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0160

FILE_IO_PATH_WRITE_LEN_BYTES
- Address: 60640

Write the length (in bytes) of the file path or directory to access.

- Data type: UINT32 (type index = 1)
- Write-only
- This register uses system RAM. The maximum RAM is 64KB. For more information, see 4.4 RAM

- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0134

FILE_IO_PATH_READ_LEN_BYTES
- Address: 60642

Read the length (in bytes) of the next file path or directory to access.

- Data type: UINT32 (type index = 1)
- Read-only
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0134

FILE_IO_PATH_WRITE
- Address: 60650

Write the desired file path. Must first write the length of the file path string (in bytes) to FILE_IO_PATH_WRITE_LEN_BYTES. File paths should be null terminated. This register is a buffer.

- Data type: BYTE (type index = 99)
- Write-only
- This register is a Buffer Register
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0134

FILE_IO_PATH_READ
- Address: 60652

Read the next file path in the CWD. Length of the string (in bytes) determined by FILE_IO_PATH_READ_LEN_BYTES. File paths will be null terminated. This register is a buffer. Underrun behavior - fill with zeros.

- Data type: BYTE (type index = 99)
- Read-only
- This register is a Buffer Register
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0134

FILE_IO_WRITE
- Address: 60654

Unimplemented. This register is a buffer.

- Data type: BYTE (type index = 99)
- Write-only
- This register is a Buffer Register
- T8:
  - Minimum firmware version: 0.0123
- T7:

- Minimum firmware version: 99.0000

FILE_IO_READ
- Address: 60656

Read the contents of a file. Must first write to FILE_IO_OPEN. Size of the file (in bytes) determined by FILE_IO_SIZE_BYTES. This register is a buffer. Underrun behavior - throws an error.

- Data type: BYTE (type index = 99)
- Read-only
- This register is a Buffer Register
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0134

FILE_IO_LUA_SWITCH_FILE
- Address: 60662

Write any value to this register to instruct Lua scripts to switch to a new file. Lua script should periodically check LJ.CheckFileFlag() to receive instruction, then call LJ.ClearFileFlag() after file switch is complete. Useful for applications that require continuous logging in a Lua script, and on-demand file access from a host.

- Data type: UINT32 (type index = 1)
- Readable and writable
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0168

All WATCHDOG TAGS:

Name	Start Address	Type	Access
WATCHDOG_ENABLE_DEFAULT	61600	UINT32	R/W
WATCHDOG_ADVANCED_DEFAULT	61602	UINT32	R/W
WATCHDOG_TIMEOUT_S_DEFAULT	61604	UINT32	R/W
WATCHDOG_STARTUP_DELAY_S_DEFAULT	61606	UINT32	R/W
WATCHDOG_STRICT_ENABLE_DEFAULT	61610	UINT32	R/W
WATCHDOG_STRICT_KEY_DEFAULT	61612	UINT32	R/W
WATCHDOG_STRICT_CLEAR	61614	UINT32	W
WATCHDOG_RESET_ENABLE_DEFAULT	61620	UINT32	R/W
WATCHDOG_DIO_ENABLE_DEFAULT	61630	UINT32	R/W
WATCHDOG_DIO_STATE_DEFAULT	61632	UINT32	R/W
WATCHDOG_DIO_DIRECTION_DEFAULT	61634	UINT32	R/W
WATCHDOG_DIO_INHIBIT_DEFAULT	61636	UINT32	R/W
WATCHDOG_DAC0_ENABLE_DEFAULT	61640	UINT32	R/W
WATCHDOG_DAC0_DEFAULT	61642	FLOAT32	R/W
WATCHDOG_DAC1_ENABLE_DEFAULT	61650	UINT32	R/W
WATCHDOG_DAC1_DEFAULT	61652	FLOAT32	R/W

WATCHDOG_ENABLE_DEFAULT
- Address: 61600

Write a 1 to enable the watchdog or a 0 to disable. The watchdog must be disabled before writing any of the other watchdog registers (except for WATCHDOG_STRICT_CLEAR).

- Data type: UINT32 (type index = 1)
- Readable and writable
- Default value: 0
- T8:
    - Minimum firmware version: 1.0012
- T7:
    - Minimum firmware version: 1.0016

WATCHDOG_ADVANCED_DEFAULT
- Address: 61602

A single binary-encoded value where each bit is an advanced option. If bit 0 is set, IO_CONFIG_SET_CURRENT_TO_FACTORY will be done on timeout. If bit 1 is set, IO_CONFIG_SET_CURRENT_TO_DEFAULT will be done on timeout.

- Data type: UINT32 (type index = 1)
- Readable and writable
- Default value: 0
- T8:
    - Minimum firmware version: 1.0012
- T7:
    - Minimum firmware version: 1.0016

WATCHDOG_TIMEOUT_S_DEFAULT
- Address: 61604

When the device receives any communication over USB/Ethernet/WiFi, the watchdog timer is cleared. If the watchdog timer is not cleared within the timeout period, the enabled actions will be done.

- Data type: UINT32 (type index = 1)
- Readable and writable
- Default value: 0
- T8:
    - Minimum firmware version: 1.0012
- T7:
    - Minimum firmware version: 1.0016

WATCHDOG_STARTUP_DELAY_S_DEFAULT
- Address: 61606

This specifies the initial timeout period at device bootup. This is used until the first time the watchdog is cleared or timeout ... after that the normal timeout is used.

- Data type: UINT32 (type index = 1)
- Readable and writable
- Default value: 0
- T8:
    - Minimum firmware version: 1.0012
- T7:
    - Minimum firmware version: 1.0016

WATCHDOG_STRICT_ENABLE_DEFAULT
- Address: 61610

Set to 1 to enable strict mode.

- Data type: UINT32 (type index = 1)

- Readable and writable
- Default value: 0
- T8:
  - Minimum firmware version: 1.0012
- T7:
  - Minimum firmware version: 1.0016

WATCHDOG_STRICT_KEY_DEFAULT
- Address: 61612

When set to strict mode, this is the value that must be written to the clear register.

- Data type: UINT32 (type index = 1)
- Readable and writable
- Default value: 0
- T8:
  - Minimum firmware version: 1.0012
- T7:
  - Minimum firmware version: 1.0016

WATCHDOG_STRICT_CLEAR
- Address: 61614

When running in strict mode, writing the key to this register is the only way to clear the watchdog. Writing to this register while not using strict mode will clear the watchdog.

- Data type: UINT32 (type index = 1)
- Write-only
- Default value: 0
- T8:
  - Minimum firmware version: 1.0012
- T7:
  - Minimum firmware version: 1.0016

WATCHDOG_RESET_ENABLE_DEFAULT
- Address: 61620

Timeout action: Set to 1 to enable device-reset on watchdog timeout.

- Data type: UINT32 (type index = 1)
- Readable and writable
- Default value: 0
- T8:
  - Minimum firmware version: 1.0012
- T7:
  - Minimum firmware version: 1.0016

WATCHDOG_DIO_ENABLE_DEFAULT
- Address: 61630

Timeout action: Set to 1 to enable DIO update on watchdog timeout.

- Data type: UINT32 (type index = 1)
- Readable and writable
- Default value: 0
- T8:
  - Minimum firmware version: 1.0012

- T7:
    - Minimum firmware version: 1.0016

WATCHDOG_DIO_STATE_DEFAULT
- Address: 61632

The state high/low of the digital I/O after a Watchdog timeout. See DIO_STATE

- Data type: UINT32 (type index = 1)
- Readable and writable
- Default value: 0
- T8:
    - Minimum firmware version: 1.0012
- T7:
    - Minimum firmware version: 1.0016

WATCHDOG_DIO_DIRECTION_DEFAULT
- Address: 61634

The direction input/output of the digital I/O after a Watchdog timeout. See DIO_DIRECTION

- Data type: UINT32 (type index = 1)
- Readable and writable
- T8:
    - Minimum firmware version: 1.0012
- T7:
    - Minimum firmware version: 1.0016

WATCHDOG_DIO_INHIBIT_DEFAULT
- Address: 61636

This register can be used to prevent the watchdog from changing specific IO. See DIO_INHIBIT for more information.

- Data type: UINT32 (type index = 1)
- Readable and writable
- Default value: 0
- T8:
    - Minimum firmware version: 1.0012
- T7:
    - Minimum firmware version: 1.0016

WATCHDOG_DAC0_ENABLE_DEFAULT
- Address: 61640

Timeout action: Set to 1 to enable DAC0 update on watchdog timeout.

- Data type: UINT32 (type index = 1)
- Readable and writable
- Default value: 0
- T8:
    - Minimum firmware version: 1.0012
- T7:
    - Minimum firmware version: 1.0016

WATCHDOG_DAC0_DEFAULT
- Address: 61642

The voltage of DAC0 after a Watchdog timeout.

- Data type: FLOAT32 (type index = 3)
- Readable and writable
- Default value: 0
- T8:
  - Minimum firmware version: 1.0012
- T7:
  - Minimum firmware version: 1.0016

WATCHDOG_DAC1_ENABLE_DEFAULT
- Address: 61650

Timeout action: Set to 1 to enable DAC1 update on watchdog timeout.

- Data type: UINT32 (type index = 1)
- Readable and writable
- Default value: 0
- T8:
  - Minimum firmware version: 1.0012
- T7:
  - Minimum firmware version: 1.0016

WATCHDOG_DAC1_DEFAULT
- Address: 61652

The voltage of DAC1 after a Watchdog timeout.

- Data type: FLOAT32 (type index = 3)
- Readable and writable
- Default value: 0
- T8:
  - Minimum firmware version: 1.0012
- T7:
  - Minimum firmware version: 1.0016

All INTFLASH TAGS:

Name	Start Address	Type	Access
INTERNAL_FLASH_KEY	61800	UINT32	R/W
INTERNAL_FLASH_READ_POINTER	61810	UINT32	R/W
INTERNAL_FLASH_READ	61812	UINT32	R
INTERNAL_FLASH_ERASE	61820	UINT32	W
INTERNAL_FLASH_WRITE_POINTER	61830	UINT32	R/W
INTERNAL_FLASH_WRITE	61832	UINT32	W
INTERNAL_FLASH_CALCULATE_CRC	61842	UINT32	R

INTERNAL_FLASH_KEY
- Address: 61800

Sets the region of internal flash to which access is allowed.

- Data type: UINT32 (type index = 1)
- Readable and writable

INTERNAL_FLASH_READ_POINTER

- Address: 61810

The address in internal flash that reads will start from.

- Data type: UINT32 (type index = 1)
- Readable and writable

INTERNAL_FLASH_READ
- Address: 61812

Data read from internal flash. Read size must be an even number of registers.

- Data type: UINT32 (type index = 1)
- Read-only

INTERNAL_FLASH_ERASE
- Address: 61820

Erases a 4k section of internal flash starting at the specified address. This register is a buffer.

- Data type: UINT32 (type index = 1)
- Write-only
- This register is a Buffer Register

INTERNAL_FLASH_WRITE_POINTER
- Address: 61830

Address in internal flash where writes will begin.

- Data type: UINT32 (type index = 1)
- Readable and writable

INTERNAL_FLASH_WRITE
- Address: 61832

Data written here will be written to internal flash. Write size must be an even number of registers. This register is a buffer.

- Data type: UINT32 (type index = 1)
- Write-only
- This register is a Buffer Register

INTERNAL_FLASH_CALCULATE_CRC
- Address: 61842

Calculates the checksum of a 4096 byte flash page.

- Data type: UINT32 (type index = 1)
- Read-only
- This register is a Buffer Register

T4 Only

Name	Start Address	Type	Access
DIO_ANALOG_ENABLE	2880	UINT32	R/W

DIO_ANALOG_ENABLE
- Address: 2880

Read or write the analog configuration of all digital I/O in a single binary-encoded value. 1=Analog mode and 0=Digital mode. When switching from analog to digital, the lines will be set to input. Writes are filtered by the value in DIO_INHIBIT.

- Data type: UINT32 (type index = 1)
- Readable and writable

Constant	Value
Digital mode	0
Analog mode	1

3.2 Stream Mode [T-Series Datasheet]

Stream Mode Overview

Streaming is a fast data input mode. It is more complicated than command-response mode, so it requires more configuration. Using stream is simplified by the LJM stream functions; to stream without them, see Low-Level Streaming below.

For a given stream session, a list of channels/addresses are sampled as input to the device. This list of channels (known as a scan list) is input, as quickly as possible, immediately after a clock pulse. Stream clock pulses are hardware-timed at a constant scan rate. By default, a stream session begins scanning immediately after being started and continuously scans until stopped.

A LabJack device cannot run more than one stream session at any given time.

Command-response can be done while a stream is active, but streaming needs exclusive control of the analog input system, so analog inputs (including the internal temperature sensor) cannot be read via command-response while a stream is active.

Stream can also output data.

Stream sessions can be configured to collect a limited number of scans. (See Burst Stream)

The T7 and T8 support some advanced stream features:

- Stream sessions can be configured to delay scanning until after the device detects a trigger pulse.
- Stream clock pulses can also be read externally at either a constant or a variable rate. *T7 Only

Maximum Stream Speed

T4 / T7

T4 Max Sample Rate: 50 ksamples/second

The T4 max sample rate is 50 ksamples/second. This is achievable for any single-address stream, but for a multi-address stream this is only true when resolution index = 0 or 1.

T7 Max Sample Rate: 100 ksamples/second

The T7 max sample rate is 100 ksamples/second. This is achievable for any single-address stream, but for a multi-address stream this is only true when resolution index = 0 or 1 and when range = ±10V for all analog inputs.

The **max scan rate** depends on how many addresses you are sampling per scan:

- Address => The Modbus address of one channel. (See Streamable Registers, below.)
- Sample => A reading from one address.
- Scan => One reading from all addresses in the scan list.
- SampleRate = NumAddresses * ScanRate

Examples:

- For a T4 streaming 4 channels at resolution index=0, the max scan rate is 10 kscans/second (calculated from 40 ksamples/second divided by 4).
- For a T7 streaming 5 channels at resolution index=0 and all at range=+/-10V, the max scan rate is 20 kscans/second (calculated from 100 ksamples/second divided by 5).

Ethernet provides the best throughput: Ethernet is capable of the fastest stream rates. USB is typically a little slower than Ethernet, and WiFi is much slower. For more information on speeds, see the Data Rates Appendix.

T8

The maximum T8 sample rate is 40 ksamples/second. This rate is achievable for any number of active stream channels, but only guaranteed with a resolution index setting of 1. Higher resolution index settings may result in lower maximum rates.

Unlike the T4 and T7, the maximum T8 scan rate does not depend on how many addresses you sample per scan. The T8 analog inputs are all sampled simultaneously, so acquiring data from multiple channels will not affect the maximum scan rate.

When streaming at 40 kHz with 8 channels of data being read, the T8 will need to transfer 960+ kilobytes per second to avoid a stream buffer overflow. As such, high stream rates are only possible when certain communication requirements are met.

High stream rates cannot be used reliably under any of the following conditions:

- When there is a WiFi hop in the Ethernet network.
- When the connection crosses subnets.
- If there are any other network conditions that can cause long round trip times.

High stream rates can be used reliably under either of the following conditions:

- When using a fully wired Ethernet connection from T8 to a switch and/or a computer.
- When using a USB 3.0 port connected through a powered hub.

Stream-In and/or Stream-Out

There are three input/output combinations of stream mode:

Stream-in: The device collects data and streams it to the host.

Stream-out: The device does not collect data but does outputs data. See Stream-Out below.

Stream-in-out: The device collects data and streams it to the host. It also streams data out.

The stream channels determine which of these modes are used. Streamable channels may be either stream-in or stream-out.

Streamable Registers

The Modbus Map shows which registers can be streamed (by expanding the "details" area). Input registers that can be streamed include:

AIN#
: See 14.0 Analog Inputs.

FIO_STATE
: See 13.0 Digital I/O.

EIO_STATE
: See 13.0 Digital I/O.

CIO_STATE
: See 13.0 Digital I/O.

MIO_STATE
: See 13.0 Digital I/O.

FIO_EIO_STATE
: See 13.0 Digital I/O.

EIO_CIO_STATE
: See 13.0 Digital I/O.

CIO_MIO_STATE
: See 13.0 Digital I/O.

DIO#(0:22)
: See 13.0 Digital I/O.

DIO#(0:22)_EF_READ_A
: See 13.2 DIO Extended Features.

DIO#(0:22)_EF_READ_A_AND_RESET
: See 13.2 DIO Extended Features.

DIO#(0:22)_EF_READ_B
: See 13.2 DIO Extended Features.

CORE_TIMER
: See 4.0 Hardware Overview.

SYSTEM_TIMER_20HZ
: See 4.0 Hardware Overview.

STREAM_DATA_CAPTURE_16
: See below.

Stream Data Framing

Stream data is returned channel interleaved by scan. A scan consists of a sample from every channel in the scan list. For example, if you have a scan list of `[AIN0, AIN1, AIN2]` data will be returned like the following:

```
[AIN0Sample1, AIN1Sample1, AIN2Sample1, AIN0Sample2, AIN1Sample2, AIN2Sample2, ... AIN0SampleN, AIN1SampleN, AIN2SampleN]
```

> Data from every AIN channel will be returned in each scan when using a device that samples its analog inputs simultaneously. See the Simultaneous Sampling section for more information.

Digital Only Stream

On some T-series devices it is possible to stream digital channels without any AIN channels in the scan list. This is referred to as a digital only stream.

T4/T7 - Supported

Digital only stream is supported. When digital only stream is active, the AIN can be read normally via command-response mode communications.

T8 - Unsupported

Digital only stream is not supported. One AIN channel must always be present in the stream scan list. Stream mode takes exclusive control over the analog input system and the AIN cannot be read via command-response mode communications while stream is active.

16-bit or 32-bit Data

Stream data is transferred as 16-bit values, but 32-bit data can be captured by using STREAM_DATA_CAPTURE_16:

Name	Start Address	Type	Access
STREAM_DATA_CAPTURE_16	4899	UINT16	R

STREAM_DATA_CAPTURE_16
- Address: 4899

If a channel produces 32-bits of data the upper 16 will be saved here.

- Data type: UINT16 (type index = 0)
- Read-only
- This register may be streamed
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0071

16-bit

In the normal case of an analog input such as AIN0, the 16-bit binary value is actually is what is transferred. LJM converts it to a float on the host using the calibration constants that LJM reads before starting the stream. To read binary AIN stream data, enable the LJM configuration LJM_STREAM_AIN_BINARY by setting it to 1.

32-bit

Some streamable registers (e.g. DIO4_EF_READ_A) have 32-bit data. When streaming a register that produces 32-bit data, the lower 16 bits (LSW) will be returned and the upper 16 bits (MSW) will be saved in STREAM_DATA_CAPTURE_16. To get the full 32-bit value, add STREAM_DATA_CAPTURE_16 to the stream scan list after any applicable 32-bit register, then combine the two values in software (LSW + 65536*MSW). Note that STREAM_DATA_CAPTURE_16 may be placed in multiple locations in the scan list.

Configuring AIN for Stream

STREAM_SETTLING_US and STREAM_RESOLUTION_INDEX control settling and resolution during stream.

Name	Start Address	Type	Access
STREAM_SETTLING_US	4008	FLOAT32	R/W

STREAM_SETTLING_US
- Address: 4008

Time in microseconds to allow signals to settle after switching the mux. Does not apply to the 1st channel in the scan list, as that settling is controlled by scan rate (the time from the last channel until the start of the next scan). Default=0. When set to less than 1, automatic settling will be used. The automatic settling behavior varies by device.

- Data type: FLOAT32 (type index = 3)
- Readable and writable
- Default value: 0
- T8:
  - T8 description TBD
- T7:
  - The T7 will select the settling time based on resolution and gain settings. When the sample rate is above 60 kHz, settling time will be set to ~5 us. Max is 4400.
- T4:
  - The T4 will default to 10 us. When the scan rate is above 20 kHz, settling time will be set to 5 us.

STREAM_RESOLUTION_INDEX	4010	UINT32	R/W

STREAM_RESOLUTION_INDEX
- Address: 4010

The resolution index for stream readings. A larger resolution index generally results in lower noise and longer sample times.

- Data type: UINT32 (type index = 1)
- Readable and writable
- Default value: 0
- T8:
  - T8 description TBD
- T7:
  - Valid values: 0-8. Default value of 0 corresponds to an index of 1.
- T4:
  - Valid values: 0-5. Default value of 0 corresponds to an index of 1.

The normal AIN configuration registers for range and negative channel still apply to stream.

T7 only: Stream mode is not supported on the hi-res converter. (Resolution indices 9-12 are not supported in stream.)

Stream Timing

When using LJM, there are three ways that stream can be too slow:

1. Sample rate is too high
2. Device buffer overflow
3. LJM buffer overflow

Sample rate is too high: When the sample rate is too high, it causes a STREAM_SCAN_OVERLAP error and stream is terminated.

Scans are triggered by hardware interrupts. If a scan begins and the previous scan has not finished, the device stops streaming and returns a STREAM_SCAN_OVERLAP error (errorcode 2942), which LJM returns immediately upon the next call to LJM_eStreamRead.

Device buffer overflow: When the device buffer overflows, LJM inserts a dummy sample (with the value -9999.0) in place of each skipped sample, or it causes a STREAM_AUTO_RECOVER_END_OVERFLOW error and stream is terminated.

As samples are collected, they are placed in a FIFO ring buffer on the device until retrieved by the host. The size of the buffer is variable and can be set to a maximum of 32768 bytes. Write to STREAM_BUFFER_SIZE_BYTES to allocate memory for the buffer. The stream buffer will be allocated within shared memory. Because we use a ring buffer, two bytes of the allocated space is required to track the number of samples in the buffer, and as a result the usable buffer size will be equal to the value of STREAM_BUFFER_SIZE_BYTES minus 2 bytes.

Name	Start Address	Type	Access
STREAM_BUFFER_SIZE_BYTES	4012	UINT32	R/W

STREAM_BUFFER_SIZE_BYTES
- Address: 4012

Size of the stream data buffer in bytes. A value of 0 equates to the default value. Must be a power of 2. Size in samples is STREAM_BUFFER_SIZE_BYTES/2. Size in scans is (STREAM_BUFFER_SIZE_BYTES/2)/STREAM_NUM_ADDRESSES. Changes while stream is running do not affect the currently running stream.

- Data type: UINT32 (type index = 1)
- Readable and writable
- Default value: 0
- This register uses system RAM. The maximum RAM is 64KB. For more information, see 4.4 RAM
- T8:
  - T8 description TBD
- T7:
  - Max size is 32768. Default size is 4096.
- T4:
  - Max size is 32768. Default size is 8192.

If the device buffer overflows, the device will continue streaming but will discard data until the buffer is emptied, after which data will be stored in the buffer again. The device keeps track of how many scans are discarded and reports that value. Based on the number of scans discarded, the LJM library adds the proper number of dummy samples (with the value -9999.0) such that the correct timing is maintained. This will only work if the first channel in the scan cannot return 0xFFFF.

If the device buffer overflows for too much time, a STREAM_AUTO_RECOVER_END_OVERFLOW error occurs and stream is terminated.

If the device buffer is overflowing, see the LJM stream help page for some mitigation strategies.

LJM buffer overflow: When the LJM buffer overflows, it causes a LJME_LJM_BUFFER_FULL error and stream is terminated.

LJM reads samples from the device buffer and buffers them internally. LJM reads these samples in an internal thread, regardless of what your code does. LJM's buffer can run out of space if it is not read often enough using LJM_eStreamRead, so make sure the LJMScanBacklog parameter does not continually increase.

LJM_eStreamRead blocks until enough data is read from the device, so your code does not need to perform waits.

If the LJM buffer is overflowing, see the LJM stream help page for some mitigation strategies.

Channel-to-Channel (Interchannel) Timing

Channels in a scan list are input or output as quickly as possible after the start of a scan, in the order of the scan list.

For descriptions of typical interchannel delays, see Appendix A-1 Data Rates.

Timing pulses are generated on SPC so that the channel-to-channel timing can be measured. Pulses on SPC are as follows:

- Falling edge at the start of a scan.
- Rising edge at the start of a sample.
- Falling edge at the end of a sample.
- Rising edge at the end of a scan.

Device Clock Scan Time

To calculate the time a scan was collected relative to the device clock, multiply the scan's offset from the first scan with the interval length. The interval length is the inverse of the scan rate:

```
TimeSinceFirstScan = Offset * (1 / ScanRate)
```

For example, with a 500 Hz scan rate, the 1000th scan collected is 2 seconds after the first scan, according to the device clock.

You can use STREAM_START_TIME_STAMP to relate the start of stream with other events:

Name	Start Address	Type	Access
STREAM_START_TIME_STAMP	4026	UINT32	R

STREAM_START_TIME_STAMP
- Address: 4026

This register stores the value of CORE_TIMER at the start of the first stream scan. Note that the first stream scan happens 1 scan period after STREAM_ENABLE is set to 1.

- Data type: UINT32 (type index = 1)
- Read-only
- Default value: 0
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0214

System Clock Scan Time

With writing custom code, you can assign timestamps to each stream scan. These timestamps can be the host computer's system time (calendar time) or the steady clock (absolute) time. These timestamps are based on the computer's real-time clock rather than the device's clock.

Assigning a timestamp to each scan can be beneficial for logging and for multi-device synchronization. It can also help to correct clock drift between your system clock and the device clock.

Using this technique it is realistic that the time accuracy of every scan's timestamp is within 1 ms of your system's clock, unless suboptimal conditions apply, such as asymmetric network routes or network congestion.

Example Technique:

First, start stream. Then read the following registers using normal command-response.

Once stream is started, use STREAM_START_TIME_STAMP to determine when stream started,hen read CORE_TIMER in order to correlate scan times with system host clock times:

Name	Start Address	Type	Access
CORE_TIMER	61520	UINT32	R

CORE_TIMER
- Address: 61520

Internal 32-bit system timer running at 1/2 core speed, thus normally 80M/2 => 40 MHz.

- Data type: UINT32 (type index = 1)
- Read-only
- This register may be streamed
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0071

1. Read the CORE_TIMER and your system time quickly in a loop for e.g. five times. You can assume the CORE_TIMER value is, on average, halfway between when you begin the CORE_TIMER read and when you receive the CORE_TIMER value. Throw out any measurements that take an abnormal amount of time. (If most round-trip reads of CORE_TIMER take 1 ms and one CORE_TIMER read takes 3 ms, it's likely that either the outbound or inbound communication had an unusual delay—but unlikely that both had an equal delay.)
2. Next, calculate the CORE_TIMER value for each scan. Use the actual scan rate to calculate the CORE_TIMER value for each scan starting from STREAM_START_TIME_STAMP. If you're using LJM, LJM_eStreamStart returns the actual scan rate in the ScanRate parameter, which can be slightly different from the specified scan rate. If you're not using LJM, see low-level streaming below.
3. Once you have your current system clock correlated with CORE_TIMER and each scan is assigned a CORE_TIMER value, you can convert each CORE_TIMER value into a system clock time.

Additional considerations:

- Make sure your code handles when the CORE_TIMER rolls over.
- If you're assigning wall-clock timestamps, consider what should happen when the clock skips forward or backward due to an NTP update.
- To prevent clock drift, you must continually re-synchronize the system clock to CORE_TIMER. The T-series clock error at room temperature is 20 ppm. This is 2 ms per 100 seconds, so a re-sync of core-clock to host-clock must be done at least every 50 seconds to maintain 1 ms accuracy.
- Check your own computer's clock specs for a source of additional clock error.

Burst Stream

Burst stream is when stream collects a pre-determined number of scans, then stops. To set the stream burst size, write to STREAM_NUM_SCANS:

Name	Start Address	Type	Access
STREAM_NUM_SCANS	4020	UINT32	R/W

STREAM_NUM_SCANS
- Address: 4020

The number of scans to run before automatically stopping (stream-burst). 0 = run continuously. Limit for STREAM_NUM_SCANS is 2^32-1, but if the host is not reading data as fast as it is acquired you also need to consider STREAM_BUFFER_SIZE_BYTES.

- Data type: UINT32 (type index = 1)
- Readable and writable
- Default value: 0

The LJM library collects burst stream data with the StreamBurst function.

It may be beneficial to set STREAM_BUFFER_SIZE_BYTES to a large value for fast burst stream. See above for details about STREAM_BUFFER_SIZE_BYTES.

T7-Pro only: Burst stream is well-suited for WiFi connections, because WiFi has a lower throughput than other connection types.

Externally Clocked Stream - T7 Only

Externally-clocked stream allows T-series devices to stream from external pulses. It also allows for variable stream scan rates.

Clock Source

The scan rate is generated from the internal crystal oscillator. Alternatively, the scan rate can be a division of an external clock provided on CIO3.

Name	Start Address	Type	Access
STREAM_CLOCK_SOURCE	4014	UINT32	R/W

STREAM_CLOCK_SOURCE
- Address: 4014

Controls which clock source will be used to run the main stream clock.
0 = Internal crystal,
2 = External clock source on CIO3.
Rising edges will increment a counter and trigger a stream scan after the number of edges specified in STREAM_EXTERNAL_CLOCK_DIVISOR. T7 will expect external stream clock on CIO3. All other values reserved.

- Data type: UINT32 (type index = 1)
- Readable and writable
- Default value: 0
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0110

To subdivide the external clock pulses for a slower scan rate, use STREAM_EXTERNAL_CLOCK_DIVISOR.

Name	Start Address	Type	Access
STREAM_EXTERNAL_CLOCK_DIVISOR	4022	UINT32	R/W

STREAM_EXTERNAL_CLOCK_DIVISOR
- Address: 4022

The number of pulses per stream scan when using an external clock.

- Data type: UINT32 (type index = 1)
- Readable and writable
- Default value: 0

To use externally clocked stream with LJM, see the externally clocked stream section of the LJM User's Guide.

Using a DIO_EF as a Clock Source

One option to generate a stream clock is to use the PWM DIO_EF. The clock signal from a single PWM output could be shared between multiple devices. For example, configure a 50 kHz

output with 50% duty cycle on the FIO0 terminal of one LabJack. Jumper FIO0 to CIO3 for use with externally clocked stream. The following pseudocode demonstrates the configuration that is required:

```
STREAM_CLOCK_SOURCE = 2 # Configure externally clocked stream on
CIO3
```

```
DIO_EF_CLOCK0_ENABLE = 0
DIO_EF_CLOCK0_DIVISOR = 1 # No divisor, the base clock is 80 MHz on the
T7
DIO_EF_CLOCK0_ROLL_VALUE = 1600 # 80 MHz / 1600 = 50 kHz output
DIO_EF_CLOCK0_ENABLE = 1
```

```
DIO0_EF_ENABLE = 0
DIO0_EF_INDEX = 0 # PWM output on
FIO0/DIO0
```

```
DIO0_EF_OPTIONS = 0 # Use DIO_EF_CLOCK0 (50kHz output configured above)
DIO0_EF_CONFIG_A = 800 # 800 / 1600 = 50% duty cycle
DIO0_EF_ENABLE = 1
```

Triggered Stream - T7/T8

T7 minimum firmware 1.0186

T8 minimum firmware 1.0125

Stream can be configured to start scanning when a trigger is detected. Trigger sources are DIO_EF modes:

- Frequency In (requires two edges to trigger)
- Pulse Width In (requires two edges in either direction to trigger)
- Conditional Reset (requires one edge to trigger)

Frequency In and Conditional Reset allow you to select rising or falling edges and Pulse Width In will trigger from either edge.

See Appendix A for hysteresis voltage information.

Configuring stream to use a trigger requires setting up a DIO_EF and adding the STREAM_TRIGGER_INDEX register to normal stream configuration.

Name	Start Address	Type	Access
STREAM_TRIGGER_INDEX	4024	UINT32	R/W

STREAM_TRIGGER_INDEX
- Address: 4024

Controls when stream scanning will start.
0 = Start when stream is enabled,
2000 = Start when DIO_EF0 detects an edge,
2001 = Start when DIO_EF1 detects an edge.
See the stream documentation for all supported values.

- Data type: UINT32 (type index = 1)
- Readable and writable
- Default value: 0
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0186

STREAM_TRIGGER_INDEX (address 4024):

- 0 = No trigger. Stream will start when Enabled.
- 2000 = DIO_EF0 will start stream.
- 2001 = DIO_EF1 will start stream.
- 2002 = DIO_EF2 will start stream.
- 2003 = DIO_EF3 will start stream.
- 2006 = DIO_EF6 will start stream.
- 2007 = DIO_EF7 will start stream.

The delay between the trigger and the first collected sample is one scan period.

To use triggered stream with LJM, see the triggered stream section of the LJM User's Guide.

A more complicated stream trigger can be implemented with a Lua script. For example, a Lua script could check for an arbitrary stream trigger condition in conjunction with triggered stream being started as normal. Once the Lua script detects that the stream condition is fulfilled, it writes a pulse to a digital out (such as DIO3) which is then detected by the normal trigger (as specified by STREAM_TRIGGER_INDEX).

Stream Out

Stream-out is a set of streamable registers that move data from a buffer to an output. The output can be digital I/O (DIO) or a DAC. The buffer can be read linearly to generate a irregular waveform or be read in a looping mode to generate a periodic waveform.

A T-series device can output up to 4 waveforms using stream-out.

In terms of timing and data rates, stream-out channels count the same as input channels so see the normal documentation of Streaming Data Rates.

Alternate waveform generation techniques are described in the Waveform Generation App Note.

The stream out waveform only resets when the stream out channel is reset. This allows the stream to be stopped and started again to begin stream out from where the waveform left off. To restart a stream out from the beginning of the waveform after stopping stream, stream out can be reconfigured.

Performing Stream-Out Using LJM

LJM's stream-out functions simplify the use of stream-out. They are available in LJM 1.2100 and later.

- To output a periodic waveform from the device, use LJM_PeriodicStreamOut.
- To output an irregular waveform, use LJM_InitializeAperiodicStreamOut and LJM_WriteAperiodicStreamOut.

Performing Stream-Out Manually (Advanced)

For each waveform being streamed out:

1. Choose which target channel will output the waveform
2. Configure stream-out
3. STREAM_OUT#(0:3)_TARGET
4. STREAM_OUT#(0:3)_BUFFER_ALLOCATE_NUM_BYTES
5. STREAM_OUT#(0:3)_ENABLE
6. Update the stream-out buffer
7. STREAM_OUT#(0:3)_LOOP_NUM_VALUES
8. STREAM_OUT#(0:3)_BUFFER_F32 or STREAM_OUT#(0:3)_BUFFER_U16
9. STREAM_OUT#(0:3)_SET_LOOP
10. Start stream with STREAM_OUT#(0:3) in the scan list
11. Stream loop: read and update buffer as needed
12. Stop stream

Executing stream-out for multiple output waveforms is a matter of performing the above steps in the order above and using corresponding STREAM_OUT#(0:3) addresses in the scan list.

Target Selection

The following target list represents the I/O on the device that can be configured to output a waveform using stream-out. The list includes the analog and digital output lines.

- DAC0
- DAC1
- FIO_STATE
- FIO_DIRECTION
- EIO_STATE
- EIO_DIRECTION
- CIO_STATE
- CIO_DIRECTION

- MIO_STATE
- MIO_DIRECTION

The above listed digital IO registers use the higher byte as an inhibit mask. Bits set in the inhibit mask will prevent the corresponding DIO from being changed. For example, writing a value of 0xFAFF to FIO_STATE will set FIO0 and FIO2 to high. FIO1 and FIO3-7 will remain unchanged.

Digital IO registers (FIO_STATE, EIO_STATE, CIO_STATE, MIO_STATE) do not configure direction. Use the corresponding _DIRECTION register (FIO_DIRECTION, EIO_DIRECTION, CIO_DIRECTION, MIO_DIRECTION) to configure direction. This can be done either before or during stream.

Configure Stream-Out

Configuration will set the buffer size and target. The target specifies which physical I/O to use. Data in the buffer will be output onto the target I/O as a generated waveform.

Stream-Out Configuration

Name	Start Address	Type	Access
STREAM_OUT#(0:3)_TARGET	4040	UINT32	R/W

STREAM_OUT#(0:3)_TARGET
- Starting Address: 4040

Channel that data will be written to. Before writing data to _BUFFER_###, you must write to _TARGET so the device knows how to interpret and store values.

- Data type: UINT32 (type index = 1)
- Readable and writable
- Default value: 0
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0012

Expanded Names	Addresses
STREAM_OUT0_TARGET, STREAM_OUT1_TARGET, STREAM_OUT2_TARGET, STREAM_OUT3_TARGET	4040, 4042, 4044, 4046

STREAM_OUT#(0:3)_BUFFER_ALLOCATE_NUM_BYTES	4050	UINT32	R/W

STREAM_OUT#(0:3)_BUFFER_ALLOCATE_NUM_BYTES
- Starting Address: 4050

Size of the buffer in bytes as a power of 2. Should be at least twice the size of updates that will be written and no less than 32. Before writing data to _BUFFER_###, you must write to _BUFFER_ALLOCATE_NUM_BYTES to allocate RAM for the data. Max is 16384.

- Data type: UINT32 (type index = 1)
- Readable and writable
- Default value: 0
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 0.9300

Expanded Names	Addresses
STREAM_OUT0_BUFFER_ALLOCATE_NUM_BYTES, STREAM_OUT1_BUFFER_ALLOCATE_NUM_BYTES, STREAM_OUT2_BUFFER_ALLOCATE_NUM_BYTES, STREAM_OUT3_BUFFER_ALLOCATE_NUM_BYTES	4050, 4052, 4054, 4056

STREAM_OUT#(0:3)_ENABLE	4090	UINT32	R/W

STREAM_OUT#(0:3)_ENABLE
- Starting Address: 4090

When STREAM_OUT#_ENABLE is enabled, the stream out target is generally updated by one value from the stream out buffer per stream scan. For example, there will generally be one instance of e.g. STREAM_OUT0 in the stream scan list, which will cause one STREAM_OUT0_BUFFER value to be consumed and written to STREAM_OUT0_TARGET for every stream scan. The stream scan list could also contain two instances of STREAM_OUT0, in which case two values from STREAM_OUT0_BUFFER value would be consumed and written for every stream scan.

- Data type: UINT32 (type index = 1)
- Readable and writable
- Default value: 0
- This register uses system RAM. The maximum RAM is 64KB. For more information, see 4.4 RAM
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 0.9300

Constant	Value
DISABLED	0
ENABLED	1

Expanded Names	Addresses
STREAM_OUT0_ENABLE, STREAM_OUT1_ENABLE, STREAM_OUT2_ENABLE, STREAM_OUT3_ENABLE	4090, 4092, 4094, 4096

Configuration can be done before or after stream has started.

Update Buffer

Each stream-out has its own buffer. Data is loaded into the buffer by writing to the appropriate buffer register. Output waveform data points are stored in the buffer as 16-bit values, so values greater than 16-bits will be converted automatically before being stored in the buffer. Use only one buffer per STREAM_OUT channel.

For outputting an analog waveform (DAC output), write an array of floating-point numbers to the STREAM_OUT#(0:3)_BUFFER_F32 register.

For outputting a digital waveform, pass an array of integer 0 or 1 values to the STREAM_OUT#(0:3)_BUFFER_U16 register.

Stream-Out Buffers

Name	Start Address	Type	Access
STREAM_OUT#(0:3)_BUFFER_U16	4420	UINT16	W

STREAM_OUT#(0:3)_BUFFER_U16
- Starting Address: 4420

Data destination when sending 16-bit integer data. Each value uses 2 bytes of the stream-out buffer. This register is a buffer.

- Data type: UINT16 (type index = 0)
- Write-only
- Default value: 0
- This register is a Buffer Register
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 0.9300

Expanded Names	Addresses
STREAM_OUT0_BUFFER_U16, STREAM_OUT1_BUFFER_U16, STREAM_OUT2_BUFFER_U16, STREAM_OUT3_BUFFER_U16	4420, 4421, 4422, 4423

STREAM_OUT#(0:3)_BUFFER_F32	4400	FLOAT32	W

STREAM_OUT#(0:3)_BUFFER_F32
- Starting Address: 4400

Data destination when sending floating point data. Appropriate cal constants are used to convert F32 values to 16-bit binary data, and thus each of these values uses 2 bytes of the stream-out buffer. This register is a buffer.

- Data type: FLOAT32 (type index = 3)
- Write-only
- Default value: 0
- This register is a Buffer Register
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 0.9300

Expanded Names	Addresses
STREAM_OUT0_BUFFER_F32, STREAM_OUT1_BUFFER_F32, STREAM_OUT2_BUFFER_F32, STREAM_OUT3_BUFFER_F32	4400, 4402, 4404, 4406

Once the waveform data points are stored, configure STREAM_OUT#(0:3)_LOOP_NUM_VALUES and STREAM_OUT#(0:3)_SET_LOOP.

Stream-Out Waveform Periodicity

Name	Start Address	Type	Access
STREAM_OUT#(0:3)_LOOP_NUM_VALUES	4060	UINT32	R/W

STREAM_OUT#(0:3)_LOOP_NUM_VALUES
- Starting Address: 4060

The number of values, from the end of the array, that will be repeated after reaching the end of supplied data array.

- Data type: UINT32 (type index = 1)
- Readable and writable
- Default value: 0
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 0.9300

Expanded Names	Addresses
STREAM_OUT0_LOOP_NUM_VALUES, STREAM_OUT1_LOOP_NUM_VALUES, STREAM_OUT2_LOOP_NUM_VALUES, STREAM_OUT3_LOOP_NUM_VALUES	4060, 4062, 4064, 4066

STREAM_OUT#(0:3)_SET_LOOP	4070	UINT32	W

STREAM_OUT#(0:3)_SET_LOOP
- Starting Address: 4070

Controls when new data and loop size are used.
1=Use new data immediately.
2=Wait for synch. New data will not be used until a different stream-out channel is set to Synch.
3=Synch. This stream-out# as well as any stream-outs set to synch will start using new data immediately.

- Data type: UINT32 (type index = 1)
- Write-only
- Default value: 0
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 0.9300

Expanded Names	Addresses
STREAM_OUT0_SET_LOOP, STREAM_OUT1_SET_LOOP, STREAM_OUT2_SET_LOOP, STREAM_OUT3_SET_LOOP	4070, 4072, 4074, 4076

Start Stream

Next, start stream with STREAM_OUT#(0:3) in the scan list.

Name	Start Address	Type	Access
STREAM_OUT#(0:3)	4800	UINT16	R

STREAM_OUT#(0:3)
- Starting Address: 4800

Include one or more of these registers in STREAM_SCANLIST_ADDRESS#(0:127) to trigger stream-out updates. When added to the scan list these do count against the max scan rate just like normal input addresses, but they do not return any data in the stream read.

- Data type: UINT16 (type index = 0)
- Read-only
- Default value: 0
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0012

Expanded Names	Addresses
STREAM_OUT0, STREAM_OUT1, STREAM_OUT2, STREAM_OUT3	4800, 4801, 4802, 4803

The order of STREAM_OUT#(0:3) in the scan list determines when the target updated. For example, if STREAM_OUT3 is before STREAM_OUT0 in the scan list, STREAM_OUT3_TARGET will be updated before STREAM_OUT0_TARGET.

Stream Loop

Read from stream, if there are stream-in channels.

Also, if the output waveform needs to be updated, read STREAM_OUT#(0:3)_BUFFER_STATUS to determine when to write new values to the buffer. When to write values depends on how large the buffer is and how many values need to be written.

Name	Start Address	Type	Access
STREAM_OUT#(0:3)_BUFFER_STATUS	4080	UINT32	R

STREAM_OUT#(0:3)_BUFFER_STATUS
- Starting Address: 4080

The number of values in the buffer that are not currently being used.

- Data type: UINT32 (type index = 1)
- Read-only
- Default value: 0
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 0.9300

Expanded Names	Addresses
STREAM_OUT0_BUFFER_STATUS, STREAM_OUT1_BUFFER_STATUS, STREAM_OUT2_BUFFER_STATUS, STREAM_OUT3_BUFFER_STATUS	4080, 4082, 4084, 4086

For a more thorough description of how a Stream-Out buffer works, see the Stream-Out Description below.

Stop Stream

Stopping a stream that streams out is no different from stopping stream-in.

Example

This example demonstrates how to configure DAC0 to output an analog waveform that resembles a triangle wave, and also quickly measure two analog inputs AIN0 and AIN2 in streaming context.

Configuration steps specific to stream-out

```
STREAM_OUT0_ENABLE = 0                  → Turn off just in case it was already on.
STREAM_OUT0_TARGET = 1000                 → Set the target to DAC0.
STREAM_OUT0_BUFFER_ALLOCATE_NUM_BYTES = 512  → A buffer to hold up to 256
values.
STREAM_OUT0_ENABLE = 1                  → Turn on Stream-Out0.
```

With the LJM library, write these registers with a call to eWriteNames or multiple calls to eWriteName.

General stream configuration

```
STREAM_SCANLIST_ADDRESS0= AIN0         → Add AIN0 to the list of things to stream in.
STREAM_SCANLIST_ADDRESS1= STREAM_OUT0  → Add STREAM_OUT0 (DAC0 is target) to the list of things to
stream-out.
STREAM_SCANLIST_ADDRESS2= AIN2         → Add AIN2 to the list of things to stream in.
STREAM_ENABLE = 1                      → Start streaming. LJM_eStreamStart does this.
```

With the LJM library, this is all done with the call to eStreamStart.

Other settings related to streaming analog inputs have been omitted here but are covered under the section for stream mode.

Load the waveform data points

The following data points have been chosen to produce the triangle waveform: 0.5V, 1V, 1.5V, 1V, so the next step is to write these datum to the appropriate buffer. Because it is a DAC output (floating point number), use the STREAM_OUT0_BUFFER_F32 register.

```
STREAM_OUT0_BUFFER_F32 = [0.5, 1, 1.5, 1]  → Write the four values one at a time or as an
array.
STREAM_OUT0_LOOP_NUM_VALUES = 4            → Loop four values.
STREAM_OUT0_SET_LOOP = 1                   → Begin using new data set immediately.
```

With the LJM library, write the array using eWriteNameArray, and write the other 2 values with a call to eWriteNames or multiple calls to eWriteName.

Observe result with stream mode

Every time the stream is run, AIN0 is read, then DAC0 is updated with a data point from Stream-Out0's buffer, then AIN2 is read. Thus, the streaming speed dictates the frequency of the output waveform.

Sequential Data

Once a sequence of values has been set via the STREAM_OUT#_SET_LOOP register, that sequence of values will loop and only be interrupted at the end of the sequence. Therefore, to have stream-out continuously output a sequence of values that is larger than the size of one stream-out buffer, probably the easiest way to do so is to:

1. Start by dividing the stream-out buffer into 2 halves,

2. Write one half of the buffer with your sequential data,

3. In a loop, every time the STREAM_OUT#_BUFFER_STATUS reads as being half full/empty, write another half buffer-worth of values.

Note that the buffer is a circular array, so you could end up overwriting values if you're not careful.

Here's an example:

Stream-out buffer is 512 bytes, divide that by 2 to get the number of samples the buffer can hold => 256 samples

256 samples divided by 2 to get the "loop" size, AKA the set-of-data-to-be-written-at-a-time size => 128 samples

Write 128 samples:

Write 128 to STREAM_OUT0_LOOP_NUM_VALUES

Write 128 samples to STREAM_OUT0_BUFFER_F32 (This should probably be done by array write, which is much faster than writing values individually.)

Write 1 to STREAM_OUT0_SET_LOOP

Loop while you have more sequential data to write:

Read STREAM_OUT0_BUFFER_STATUS

If STREAM_OUT0_BUFFER_STATUS is 128 or greater, write the next 128 samples, along with STREAM_OUT0_LOOP_NUM_VALUES = 128 and STREAM_OUT0_SET_LOOP = 1

Sleep for something like 1 / scanRate seconds to prevent unnecessary work for the hardware

Maximum Speed Estimations

T4: With 1 channel and a looping waveform, stream-out can perform at 40 kHz.

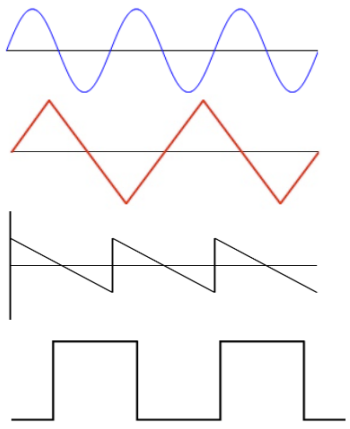
T7: With 1 channel and a looping waveform, stream-out can perform at 100 kHz. When streaming out to DACs at high speeds, you may notice effects of the time constant, depending on the output waveform.

T8: With 1 channel and a looping waveform, stream-out can perform at 20 kHz.

For more information, see maximum stream speeds.

Stream-Out Description

T-Series Stream Out System



The stream out system enables a devices analog output and digital output lines to be controlled without constant interaction with a computer. Additionally, once configured the device can output these data points without suffering from communication delays that occur frequently over USB, Ethernet, and WiFi connections so higher frequency and more precise waveforms can be generated.

Useful applications:

a. Generating arbitrary waveforms with the analog output (DAC) channels ex: sine, triangle, sawtooth, square waves, and more.
b. Generating custom PWM pulse-trains with digital output lines (FIO, EIO, CIO, MIO)

Low-Level Stream

Stream mode is complicated but can easily be executed using the high-level LJM stream functions. LJM is recommend for all users, except users that need to integrate a T-series device into a system that cannot use LJM. The rest of this section is about manually executing stream protocol without LJM.

Executing stream mode involves the following:

- Stream setup
- Stream start
- Stream-in data collection, if any stream includes stream-in channels
- Stream-out buffer updates, if stream includes stream-out channels (See the Stream-Out section)
- Stream stop

Spontaneous Stream vs. Command-Response Stream:

Data can be sent to the host in one of two data collection modes:

- Spontaneous: In spontaneous mode, packets are automatically sent to the host as soon as there is enough data to fill a packet. The packet size is adjustable. See the register definitions below.
- Command-Response (CR): In CR mode, the stream data is stored in the device's buffer and must be read out using a command. CR mode is useful for when the connection is unreliable.

T-series devices connected via either USB and Ethernet are capable of both spontaneous stream and command-response stream.

T7-Pro only: T7-Pro devices connected via WiFi are capable of only command-response stream.

Setup

Manual stream setup requires configuration of the registers that LJM_eStreamStart automatically configures:

Manual Stream Setup

Name	Start Address	Type	Access
STREAM_SCANRATE_HZ	4002	FLOAT32	R/W

STREAM_SCANRATE_HZ
- Address: 4002

Write a value to specify the number of times per second that all channels in the stream scanlist will be read. Max stream speeds are based on Sample Rate which is NumChannels*ScanRate. Has no effect when using an external clock. A read of this register returns the actual scan rate, which can be slightly different due to rounding. For scan rates >152.588, the actual scan interval is multiples of 100 ns. Assuming a core clock of 80 MHz the internal roll value is (80M/(8*DesiredScanRate))-1 and the actual scan rate is then 80M/(8*(RollValue+1). For slower scan rates the scan interval resolution is changed to 1 us, 10 us, 100 us, or 1 ms as needed to achieve the longer intervals.

- Data type: FLOAT32 (type index = 3)
- Readable and writable
- Default value: 0

STREAM_NUM_ADDRESSES	4004	UINT32	R/W

STREAM_NUM_ADDRESSES
- Address: 4004

The number of entries in the scanlist

- Data type: UINT32 (type index = 1)
- Readable and writable
- Default value: 0

STREAM_SAMPLES_PER_PACKET	4006	UINT32	R/W

STREAM_SAMPLES_PER_PACKET
- Address: 4006

Specifies the number of data points to be sent in the data packet. Only applies to spontaneous mode.

- Data type: UINT32 (type index = 1)
- Readable and writable
- Default value: 0

STREAM_AUTO_TARGET	4016	UINT32	R/W

STREAM_AUTO_TARGET
- Address: 4016

Controls where data will be sent. Value is a bitmask.
bit 0: 1 = Send to Ethernet 702 sockets,
bit 1: 1 = Send to USB,
bit 4: 1 = Command-Response mode.
All other bits are reserved.

- Data type: UINT32 (type index = 1)
- Readable and writable
- Default value: 0
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 0.9108

STREAM_SCANLIST_ADDRESS#(0:127)	4100	UINT32	R/W

STREAM_SCANLIST_ADDRESS#(0:127)
- Starting Address: 4100

A list of addresses to read each scan. In the case of Stream-Out enabled, the list may also include something to write each scan.

- Data type: UINT32 (type index = 1)
- Readable and writable
- Default value: 0

Expanded Names	Addresses
STREAM_SCANLIST_ADDRESS0,	4100,
STREAM_SCANLIST_ADDRESS1,	4102,
STREAM_SCANLIST_ADDRESS2,	4104,
STREAM_SCANLIST_ADDRESS3,	4106,
STREAM_SCANLIST_ADDRESS4,	4108,
STREAM_SCANLIST_ADDRESS5,	4110,
STREAM_SCANLIST_ADDRESS6,	4112,
STREAM_SCANLIST_ADDRESS7,	4114,
STREAM_SCANLIST_ADDRESS8,	4116,
STREAM_SCANLIST_ADDRESS9,	4118,
STREAM_SCANLIST_ADDRESS10,	4120,
STREAM_SCANLIST_ADDRESS11,	4122,
STREAM_SCANLIST_ADDRESS12,	4124,
STREAM_SCANLIST_ADDRESS13,	4126,
STREAM_SCANLIST_ADDRESS14,	4128,
STREAM_SCANLIST_ADDRESS15,	4130,
STREAM_SCANLIST_ADDRESS16,	4132,
STREAM_SCANLIST_ADDRESS17,	4134,
STREAM_SCANLIST_ADDRESS18,	4136,
STREAM_SCANLIST_ADDRESS19,	4138,
STREAM_SCANLIST_ADDRESS20,	4140,
STREAM_SCANLIST_ADDRESS21,	4142,
STREAM_SCANLIST_ADDRESS22,	4144,
STREAM_SCANLIST_ADDRESS23,	4146,
STREAM_SCANLIST_ADDRESS24,	4148,
STREAM_SCANLIST_ADDRESS25,	4150,
STREAM_SCANLIST_ADDRESS26,	4152,
STREAM_SCANLIST_ADDRESS27,	4154,

STREAM_SCANLIST_ADDRESS28, 4156,
STREAM_SCANLIST_ADDRESS29, 4158,
STREAM_SCANLIST_ADDRESS30, 4160,
STREAM_SCANLIST_ADDRESS31, 4162,
STREAM_SCANLIST_ADDRESS32, 4164,
STREAM_SCANLIST_ADDRESS33, 4166,
STREAM_SCANLIST_ADDRESS34, 4168,
STREAM_SCANLIST_ADDRESS35, 4170,
STREAM_SCANLIST_ADDRESS36, 4172,
STREAM_SCANLIST_ADDRESS37, 4174,
STREAM_SCANLIST_ADDRESS38, 4176,
STREAM_SCANLIST_ADDRESS39, 4178,
STREAM_SCANLIST_ADDRESS40, 4180,
STREAM_SCANLIST_ADDRESS41, 4182,
STREAM_SCANLIST_ADDRESS42, 4184,
STREAM_SCANLIST_ADDRESS43, 4186,
STREAM_SCANLIST_ADDRESS44, 4188,
STREAM_SCANLIST_ADDRESS45, 4190,
STREAM_SCANLIST_ADDRESS46, 4192,
STREAM_SCANLIST_ADDRESS47, 4194,
STREAM_SCANLIST_ADDRESS48, 4196,
STREAM_SCANLIST_ADDRESS49, 4198,
STREAM_SCANLIST_ADDRESS50, 4200,
STREAM_SCANLIST_ADDRESS51, 4202,
STREAM_SCANLIST_ADDRESS52, 4204,
STREAM_SCANLIST_ADDRESS53, 4206,
STREAM_SCANLIST_ADDRESS54, 4208,
STREAM_SCANLIST_ADDRESS55, 4210,
STREAM_SCANLIST_ADDRESS56, 4212,
STREAM_SCANLIST_ADDRESS57, 4214,
STREAM_SCANLIST_ADDRESS58, 4216,
STREAM_SCANLIST_ADDRESS59, 4218,
STREAM_SCANLIST_ADDRESS60, 4220,
STREAM_SCANLIST_ADDRESS61, 4222,
STREAM_SCANLIST_ADDRESS62, 4224,
STREAM_SCANLIST_ADDRESS63, 4226,
STREAM_SCANLIST_ADDRESS64, 4228,
STREAM_SCANLIST_ADDRESS65, 4230,
STREAM_SCANLIST_ADDRESS66, 4232,
STREAM_SCANLIST_ADDRESS67, 4234,
STREAM_SCANLIST_ADDRESS68, 4236,
STREAM_SCANLIST_ADDRESS69, 4238,
STREAM_SCANLIST_ADDRESS70, 4240,
STREAM_SCANLIST_ADDRESS71, 4242,
STREAM_SCANLIST_ADDRESS72, 4244,
STREAM_SCANLIST_ADDRESS73, 4246,
STREAM_SCANLIST_ADDRESS74, 4248,
STREAM_SCANLIST_ADDRESS75, 4250,
STREAM_SCANLIST_ADDRESS76, 4252,
STREAM_SCANLIST_ADDRESS77, 4254,
STREAM_SCANLIST_ADDRESS78, 4256,
STREAM_SCANLIST_ADDRESS79, 4258,
STREAM_SCANLIST_ADDRESS80, 4260,
STREAM_SCANLIST_ADDRESS81, 4262,
STREAM_SCANLIST_ADDRESS82, 4264,
STREAM_SCANLIST_ADDRESS83, 4266,
STREAM_SCANLIST_ADDRESS84, 4268,
STREAM_SCANLIST_ADDRESS85, 4270,
STREAM_SCANLIST_ADDRESS86, 4272,
STREAM_SCANLIST_ADDRESS87, 4274,
STREAM_SCANLIST_ADDRESS88, 4276,
STREAM_SCANLIST_ADDRESS89, 4278,
STREAM_SCANLIST_ADDRESS90, 4280,

STREAM_SCANLIST_ADDRESS91,	4282,
STREAM_SCANLIST_ADDRESS92,	4284,
STREAM_SCANLIST_ADDRESS93,	4286,
STREAM_SCANLIST_ADDRESS94,	4288,
STREAM_SCANLIST_ADDRESS95,	4290,
STREAM_SCANLIST_ADDRESS96,	4292,
STREAM_SCANLIST_ADDRESS97,	4294,
STREAM_SCANLIST_ADDRESS98,	4296,
STREAM_SCANLIST_ADDRESS99,	4298,
STREAM_SCANLIST_ADDRESS100,	4300,
STREAM_SCANLIST_ADDRESS101,	4302,
STREAM_SCANLIST_ADDRESS102,	4304,
STREAM_SCANLIST_ADDRESS103,	4306,
STREAM_SCANLIST_ADDRESS104,	4308,
STREAM_SCANLIST_ADDRESS105,	4310,
STREAM_SCANLIST_ADDRESS106,	4312,
STREAM_SCANLIST_ADDRESS107,	4314,
STREAM_SCANLIST_ADDRESS108,	4316,
STREAM_SCANLIST_ADDRESS109,	4318,
STREAM_SCANLIST_ADDRESS110,	4320,
STREAM_SCANLIST_ADDRESS111,	4322,
STREAM_SCANLIST_ADDRESS112,	4324,
STREAM_SCANLIST_ADDRESS113,	4326,
STREAM_SCANLIST_ADDRESS114,	4328,
STREAM_SCANLIST_ADDRESS115,	4330,
STREAM_SCANLIST_ADDRESS116,	4332,
STREAM_SCANLIST_ADDRESS117,	4334,
STREAM_SCANLIST_ADDRESS118,	4336,
STREAM_SCANLIST_ADDRESS119,	4338,
STREAM_SCANLIST_ADDRESS120,	4340,
STREAM_SCANLIST_ADDRESS121,	4342,
STREAM_SCANLIST_ADDRESS122,	4344,
STREAM_SCANLIST_ADDRESS123,	4346,
STREAM_SCANLIST_ADDRESS124,	4348,
STREAM_SCANLIST_ADDRESS125,	4350,
STREAM_SCANLIST_ADDRESS126,	4352,
STREAM_SCANLIST_ADDRESS127	4354

STREAM_ENABLE	4990	UINT32	R/W

STREAM_ENABLE
- Address: 4990

Write 1 to start stream. Write 0 to stop stream. Reading this register returns 1 when stream is enabled. When using a triggered stream the stream is considered enabled while waiting for the trigger.

- Data type: UINT32 (type index = 1)
- Readable and writable
- This register uses system RAM. The maximum RAM is 64KB. For more information, see 4.4 RAM
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 0.9421

Additional Configuration Notes

Additionally, address 4018 (STREAM_DATATYPE) must be written with the value 0. Note that address 4018 (STREAM_DATATYPE) is not in ljm_constants.json and is not compatible with LJM_NameToAddress.

STREAM_ENABLE must be written last.

Stream Burst

Writing a non-zero value to the register STREAM_NUM_SCANS will cause the device to automatically stop sending stream data after a certain number of scans. We commonly refer to this noncontinuous stream-mode acquisition as a stream burst.

After STREAM_NUM_SCANS scans have been acquired, the device will no longer scan new data and the status/error STREAM_BURST_COMPLETE will be returned from the LabJack. Upon receiving STREAM_BURST_COMPLETE, stream should be fully disabled by writing 0 to STREAM_ENABLE.

> In most cases, STREAM_NUM_SCANS should not be configured when using LJM. LJM will set STREAM_NUM_SCANS when using StreamBurst.

Name	Start Address	Type	Access
STREAM_NUM_SCANS	4020	UINT32	R/W

STREAM_NUM_SCANS
- Address: 4020

The number of scans to run before automatically stopping (stream-burst). 0 = run continuously. Limit for STREAM_NUM_SCANS is 2^32-1, but if the host is not reading data as fast as it is acquired you also need to consider STREAM_BUFFER_SIZE_BYTES.

- Data type: UINT32 (type index = 1)
- Readable and writable
- Default value: 0

Data Collection

Spontaneous Stream: Once stream has been initiated with STREAM_ENABLE, the device sends data to the target indicated by STREAM_AUTO_TARGET until STREAM_ENABLE is written with the value 0. Stream-out streams that do not contain stream-in channels (see above) do not send data.

Modbus Feedback Spontaneous Packet Protocol:

Bytes 0-1: Transaction ID

Bytes 2-3: Protocol ID

Bytes 4-5: Length, MSB-LSB

Bytes 6: 1 (Unit ID)

Byte 7: 76 (Function #)

Byte 8: 16

Byte 9: Reserved

Bytes 10-11: Backlog Bytes

Bytes 12-13: Status Code

Byte 14-15: Additional status information

Byte 16+: Stream Data (raw sample = 2 bytes MSB-LSB)

Command-Response Stream: When collecting data using command-response stream mode, data must be read from STREAM_DATA_CR (address 4500). Data is automatically discarded as it is read.

Modbus Feedback Command-Response Packet Protocol:

Bytes 0-1: Transaction ID

Bytes 2-3: Protocol ID

Bytes 4-5: Length, MSB-LSB

Bytes 6: 1 (Unit ID)

Byte 7: 76 (Function #)

Bytes 8-9: Number of samples in this read

Bytes 10-11: Backlog Bytes

Bytes 12-13: Status Code

Byte 14-15: Additional status information

Byte 16+: Stream Data (raw sample = 2 bytes MSB-LSB)

Backlog Bytes:

Backlog Bytes is the number bytes contained in the device stream buffer after reading. To convert BacklogBytes to the number of scans still on the device:

BacklogScans = BacklogBytes / (bytesPerSample * samplesPerScan)

Where bytesPerSample is 2 and samplesPerScan is the number of channels.

Status Codes:

- 2940: Auto-recovery Active.
- 2941: Auto-recovery End. Additional Status Information is the number of scans skipped. A scan consisting of all 0xFFFF values indicates the separation between old data and new data.
- 2942: Scan Overlap
- 2943: Auto-recovery End Overflow

- 2944: Stream Burst Complete

Stop

To stop stream, write 0 to STREAM_ENABLE. All stream modes expect to be stopped, except for burst stream (see STREAM_NUM_SCANS for more information on bust stream).

Code Example

A general low-level stream example written in C/C++ can be found here.

Simultaneous Sampling - T8 Only

> The T8 is the only T-series device that samples its analog inputs simultaneously, all other T-series devices sample sequentially.

There are two ways to sample multiple analog inputs: sequentially or simultaneously. When sampling sequentially, each AIN is sampled one after another. The time between sequential samples creates a phase delay in the data. This phase difference can result in errors when comparing two signals such as voltage and current. When sampling simultaneously, the analog inputs will be sampled at the same time and rate, thereby eliminating the phase difference and any associated errors.

Simultaneous AIN Captures when Streaming

`AIN#_CAPTURE` registers are not streamable, however all simultaneously sampled AIN behave similar to `AIN#_CAPTURE` registers when streaming; the stream scan timing determines when new simultaneous AIN readings are captured. New AIN samples always get captured precisely at the start of the stream scan.

Operational Differences Between Simultaneous and Sequential Sampling

Simultaneously sampled stream has a few important operational differences from sequentially sampled stream.

- The first time a request for a new analog input reading (AIN0-7) is encountered, the data for all 8 analog inputs will be loaded into the data-buffer.
- If there is any AIN in the stream scan list, stream data will contain samples from all of the AIN channels. As a result, more than one AIN in the low-level stream scan list will result in an error. However, our high-level LJM library will accept any number of AIN in the aScanList parameter for eStreamStart and process them internally such that the low-level scan list is trimmed to a single AIN. Calling eStreamRead will return trimmed data in the order that AIN registers are placed in the scan list for `eStreamStart` .
- Because the stream scan timing always determines when the AIN readings are captured, you can only acquire one sample from each AIN in a stream scan and the AIN are always

sampled simultaneously. For example, if you have an LJM scan list of `[AIN0, AIN1, AIN2]` LJM will return simultaneously captured readings from AIN0, AIN1, and AIN2 in each scan.
- Reading fewer analog input channels does not result in faster scan rates. Each scan collects data from all analog inputs and places the samples in the data buffer. If some analog inputs do not appear in the scan list passed to LJM, they will be discarded. There is no benefit in terms of device loading or data throughput.

The LJM Library is currently required to stream from devices that perform simultaneous sampling.

4.0 Hardware Overview [T-Series Datasheet]

T4 Hardware Overview

The T4 has 3 different I/O areas:

- **Communication Edge:** The communication edge (top edge in Figure 4.1-1) has a USB Type-B connector and an RJ45 Ethernet connector. Power is always provided through the USB connector, even if USB communication is not used.
- **Screw Terminal Edges:** The screw terminal edges have convenient connections for the 2 analog outputs, 4 high-voltage analog inputs, and 4 flexible I/O (digital I/O, low-voltage analog inputs, or DIO-EF. The screw terminals are arranged in blocks of 4, with each block consisting of VS, GND, and two I/O. Also on the left screw-terminal edge are two LEDs. The Comm LED generally blinks with communication traffic, while the Status LED is used for other indications.
- **DB Edge:** The DB Edge has a 15-pin female D-sub type connector (DB15) which has 12 digital I/O called EIO and CIO. The first 4 EIO lines can be configured as low-voltage analog inputs. The first 2 EIO lines and the 4 CIO lines support some DIO-EF features (timers, counters, etc.).

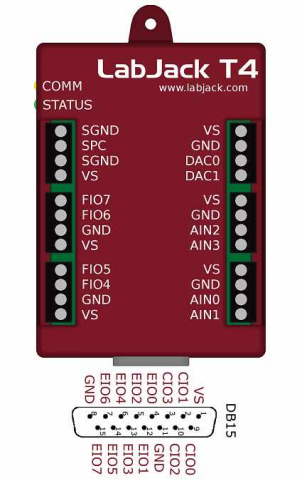


Figure 4.1-1. LabJack T4

USB: Can be used for host communication. Power is always provided through this connector.

Ethernet: 10/100Base-T Ethernet connection can be used for host communication.

LEDs: The Power and Status LEDs convey different information about the device.

VS: All VS terminals are the same. These are outputs that can be used to source about 5 volts.

GND/SGND: All GND terminals are the same. SGND has a self-resetting thermal fuse in series with GND.

FIO#/EIO#/CIO#: These are the 16 digital I/O, and are also referred to as DIO4-DIO19. Besides basic digital I/O operations, some of these terminals can be configured with Extended Features (frequency input, PWM output, etc.), some can be configured as low-voltage analog inputs, and all can be configured for various serial protocols: I2C serial, SPI serial, SBUS serial (EI-1050, SHT sensors), 1-Wire serial, and Asynchronous serial.

AIN#: AIN0-AIN3 are the 4 high-voltage (±10V) analog inputs.

DAC#: DAC0 & DAC1 are the 2 analog outputs. Each DAC can be set to a voltage between about 0.01 and 5 volts with 10-bits of resolution.

For information about reading inputs, start with the Communication section. For information about setting outputs, start with the Waveform Generation Application Note.

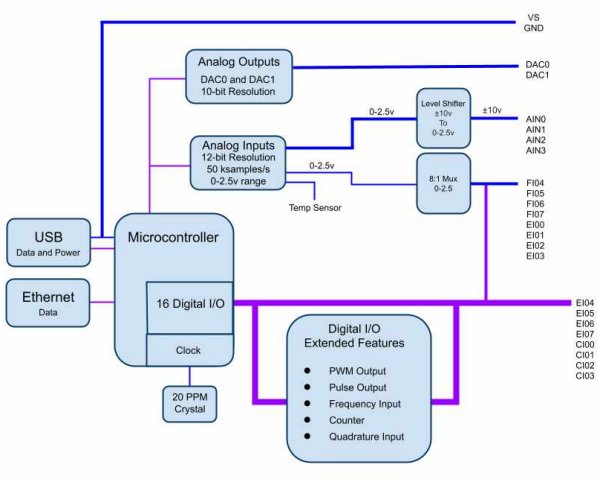


Figure 4.1-2. Block Diagram

T7 Hardware Overview

The T7 has 3 different I/O areas:

- **Communication Edge:** The T7 has a USB Type-B connector and an RJ45 Ethernet connector. The T7-Pro has those and also has an SMA-RP female connector and a WiFi antenna. Power is always provided through the USB connector, even if USB communication is not used.
- **Screw Terminal Edge:** The screw terminal edge has convenient connections for 4 analog inputs, both analog outputs, 4 digital I/O, and both current sources. The screw terminals are arranged in blocks of 4, with each block consisting of VS, GND, and two I/O. Also on this edge are two LEDs. The Comm LED generally blinks with communication traffic, while the Status LED is used for other indications.
- **DB Edge:** The DB Edge has 2 D-sub type connectors: a DB15 and DB37. The DB15 has 12 digital I/O. The DB37 has the same I/O as the screw-terminals, plus additional analog inputs and digital I/O, for a total of 14 analog inputs, 2 analog outputs, 2 fixed current sources, and 11 digital I/O.

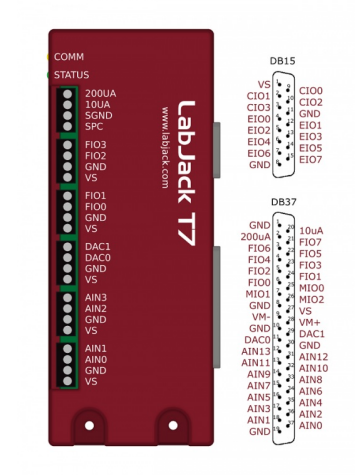


Figure 4.2-1. Enclosure & Connectors

USB: Can be used for host communication. Power is always provided through this connector.

Ethernet: 10/100Base-T Ethernet connection can be used for host communication.

WiFi (T7-Pro only): 2.4 GHz 802.11 b/g WiFi connection can be use for host communication.

LEDs: The Power and Status LEDs convey different information about the device.

VS: All VS terminals are the same. These are outputs that can be used to source about 5 volts.

GND/SGND: All GND terminals are the same. SGND has a self-resetting thermal fuse in series with GND.

10UA/200UA: Fixed current sources providing 10µA/200µA at a max voltage of about 3 volts.

FIO#/EIO#/CIO#/MIO#: These are the 23 digital I/O, and are also referred to as DIO0-DIO22. Besides basic digital I/O operations, some of these terminals can also be configured with Extended Features (frequency input, PWM output, etc.), and all can be configured for various serial protocols: I2C serial, SPI serial, SBUS serial (EI-1050, SHT sensors), 1-Wire serial, and Asynchronous serial.

AIN#: AIN0-AIN13 are the 14 analog inputs.

DAC#: DAC0 & DAC1 are the 2 analog outputs. Each DAC can be set to a voltage between about 0.01 and 5 volts with 12-bits of resolution.

For information about reading inputs, start with the Communication section. For information about setting outputs, start with the Waveform Generation Application Note.

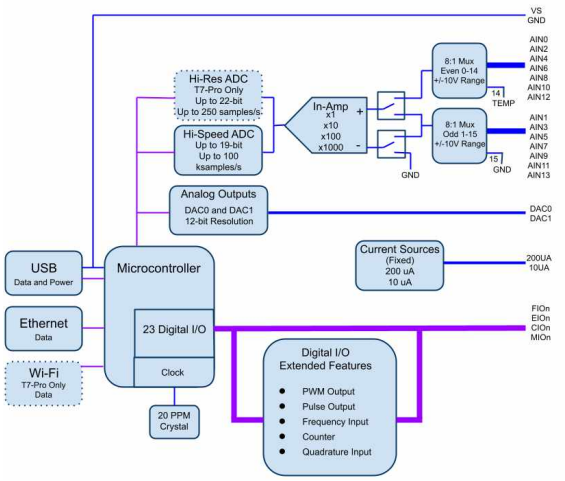


Figure 4.2-2. Block Diagram

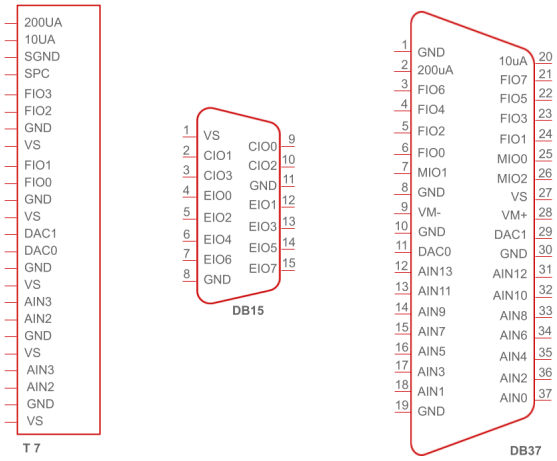


Figure 4.2-3. Wiring Diagram Objects

These wiring diagram objects are intended to be useful for making your own wiring diagrams. Let us know if they are helpful or you would like to see something else. Open this link to make a copy to your own Google Drive.

T8 Hardware Overview

The T8 has 4 different I/O areas:

- **Communication Edge:** The T8 has a USB Type-B connector and a RJ45 Ethernet Jack.
- **Miscellaneous Screw Terminal Edge:** This screw terminal edge has connections for both analog outputs, 8 digital I/O, and the reference voltage. Screw terminals on this edge are arranged in blocks of 4, with each block consisting of VS, GND, and two I/O. Also on this edge are two LEDs. The Comm LED generally blinks with communication traffic, while the Status LED is used for other indications.

- **Analog Input Screw Terminal Edge:** This screw terminal edge has connections for 8 analog inputs.
- **DB Edge:** The DB Edge has a 15-pin female D-sub type connector (DB15) which has 12 digital I/O called EIO and CIO.

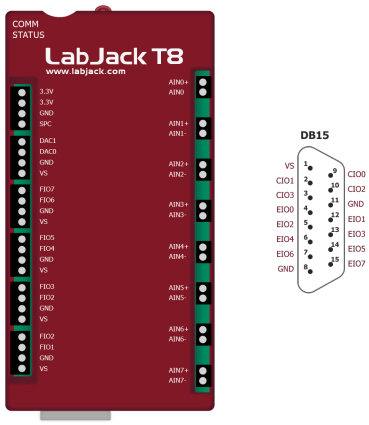


Figure 4.3-1. Enclosure & Connectors

USB: Can be used for host communication. Power can be provided through this connector.

Ethernet: 10/100Base-T Ethernet connection can be used for host communication. Power can be provided through this connector.

LEDs: The Power and Status LEDs convey information about the state of the device, and current operations.

VS: All VS terminals are the same. These are outputs that can be used to source about 5 volts.

GND: All GND terminals are the same.

3.3V: Fixed reference voltage providing 3.3 V at up to 100 mA.

FIO#/EIO#/CIO#: These are the 20 digital I/O. They are also referred to as DIO0-DIO19. All DIO can be configured for output low, high (3.3V) and input. Some DIO can also be configured to run Extended Features (frequency input, PWM output, etc.). All can be configured for various serial protocols: I2C serial, SPI serial, SBUS serial (EI-1050, SHT sensors), 1-Wire serial, and Asynchronous serial.

AIN#: AIN0-AIN7 are the 8 analog inputs.

DAC#: DAC0 & DAC1 are the 2 analog outputs. Each DAC can be set to a voltage between about -0.01 and 10.3 volts with 16-bits of resolution. Each DAC can supply up to 20 mA without significant change to the output voltage.

For information about reading inputs, start in Section 3. For information about setting outputs, start with the Waveform Generation Application Note.

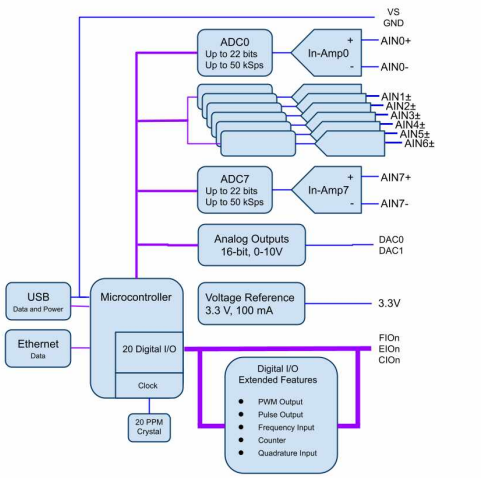


Figure 4.3-2. Block Diagram

Hardware Revisions

T4 Hardware Revisions:

- 1.2: Initial Release of the T4

T7 Hardware Revisions:

- 1.31: Initial release of the T7
- 1.35: Several changes were made to improve testing and manufacturing reliability.
- 1.35a: Switched to a new flash chip. Previous one was discontinued by the manufacturer. This version will not operate properly if a firmware version lower than 1.0218 is installed. Attempting to load firmware below 1.0218 will cause an error to be thrown.

T7-OEM Hardware Revisions:

Same revision history as T7 hardware. HW 1.35 also changed the style of LEDs that are installed. HW 1.31 came with through hole LEDs. HW 1.35 has SMD LEDs installed next to the through hole component locations making them easier to remove. They also draw less power and are lower profile.

General Device Registers

The following list of registers provide general device information

Name	Start Address	Type	Access
PRODUCT_ID	60000	FLOAT32	R

PRODUCT_ID
- Address: 60000

The numeric identifier of the device. Such as 7 for a T7 / T7-Pro.

- Data type: FLOAT32 (type index = 3)
- Read-only

HARDWARE_VERSION	60002	FLOAT32	R

HARDWARE_VERSION
- Address: 60002

The hardware version of the device.

- Data type: FLOAT32 (type index = 3)
- Read-only

FIRMWARE_VERSION	60004	FLOAT32	R

FIRMWARE_VERSION
- Address: 60004

The current firmware version installed on the main processor.

- Data type: FLOAT32 (type index = 3)
- Read-only

BOOTLOADER_VERSION	60006	FLOAT32	R

BOOTLOADER_VERSION
- Address: 60006

The bootloader version installed on the main processor.

- Data type: FLOAT32 (type index = 3)
- Read-only

WIFI_VERSION	60008	FLOAT32	R

WIFI_VERSION
- Address: 60008

The current firmware version of the WiFi module, if available.

- Data type: FLOAT32 (type index = 3)
- Read-only

HARDWARE_INSTALLED	60010	UINT32	R

HARDWARE_INSTALLED
- Address: 60010

Bitmask indicating installed hardware options.
bit0: High Resolution ADC,
bit1: WiFi,
bit2: RTC,
bit3: microSD.

- Data type: UINT32 (type index = 1)
- Read-only
- Default value: 0

ETHERNET_MAC	60020	UINT64	R

ETHERNET_MAC
- Address: 60020

The MAC address of the wired Ethernet module.

- Data type: UINT64 (type index = N/A)
- Read-only

WIFI_MAC	60024	UINT64	R

WIFI_MAC
- Address: 60024

The MAC address of the WiFi module.

- Data type: UINT64 (type index = N/A)
- Read-only

SERIAL_NUMBER	60028	UINT32	R

SERIAL_NUMBER
- Address: 60028

The serial number of the device.

- Data type: UINT32 (type index = 1)
- Read-only

DEVICE_NAME_DEFAULT	60500	STRING	R/W

DEVICE_NAME_DEFAULT
- Address: 60500

Reads return the current device name. Writes update the default and current device name. A reboot is necessary to update the name reported by NBNS. Up to 49 characters, cannot contain periods.

- Data type: STRING (type index = 98)
- Readable and writable

System Reboot

It is possible to reset T-series devices remotely using the following SYSTEM_REBOOT register.

System Reboot Register

Name	Start Address	Type	Access
SYSTEM_REBOOT	61998	UINT32	W

SYSTEM_REBOOT
- Address: 61998

Issues a device reboot. Must write 0x4C4Axxxx, where xxxx is number of 50ms ticks to wait before reboot. To reboot immediately write 0x4C4A0000 (d1279918080).

- Data type: UINT32 (type index = 1)
- Write-only
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 0.9419
- T4:
  - Minimum firmware version: 0.2020

System Timing Registers

T-series devices support several registers that are useful for system timing. The core timer runs at 1/2 the LabJack core clock speed. See Appendix A-5 for device clock specifications.

System Timing Registers

Name	Start Address	Type	Access
CORE_TIMER	61520	UINT32	R

CORE_TIMER
- Address: 61520

Internal 32-bit system timer running at 1/2 core speed.

- Data type: UINT32 (type index = 1)
- Read-only
- This register may be streamed
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0071

SYSTEM_TIMER_20HZ	61522	UINT32	R

SYSTEM_TIMER_20HZ
- Address: 61522

Internal 32-bit system timer running at 20Hz.

- Data type: UINT32 (type index = 1)
- Read-only
- This register may be streamed
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0071

WAIT_US_BLOCKING	61590	UINT32	R/W

WAIT_US_BLOCKING
- Address: 61590

Delays for x microseconds. Range is 0-100000. This operation is Blocking. While a blocking function is running no other registers can be read / written.

- Data type: UINT32 (type index = 1)
- Readable and writable
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0026

Example

You can use the core timer to measure small time intervals between LabJack commands. For example, say you acquire the following two core timer readings within a brief period of time:

`coreTicksRead1 = 12356`

`coreTicksRead2 = 323212`

`diffCoreTimerTicks = coreTicksRead2 - coreTicksRead1 = 310856`

You can convert the difference between these readings to a difference in time. For example, lets say your T-series device has a core clock that runs at 80 MHz (not applicable to all T-series devices). The core timer would run at `80 MHz / 2 = 40 MHz`. We can equivalently call this `40,000,000 clock ticks/second`. From here, we can do unit conversion to find the time between the two core timer readings:

`diffCoreTimerSeconds = diffCoreTimerTicks / coreTicksPerSecond = 310856 / 40000000 ≈ 0.00777 seconds`

The amount of time that passed between the two core timer reads was roughly 0.00777 seconds.

The core timer resolution is defined as `1 / coreTicksPerSecond`. In the 40 MHz core timer example above, the resolution is `1 / 40000000 = 25 ns`.

RAM

T-Series devices use shared memory. The shared memory allows users to allocate resources to optimize the feature set of the device. The following table describes the available RAM by device:

Device	Heap Size
T4	64 kB
T7	64 kB
T8	384 kB

Many features will allocate memory when they are enabled, and hold that memory until disabled. Others will only use memory while they are active. Here is a list of features which require shared memory:

- AIN_EF - Some memory is allocated when enabled, more can be allocated during operation.
- Lua Scripting Engine - Memory usage depends on the size of the script and the memory usage of the script.

- Stream buffers - Allocated when stream is enabled.
- StreamOut buffers - Allocated when a StreamOut channel is enabled.
- USER_RAM FIFOs - Allocated when enabled.
- Asynchronous Serial - Allocated when enabled
- File IO - Allocated to store specified paths.

When there is insufficient memory available, a SYSTEM_MEMORY_BEREFT error will be thrown. Some ideas to free up memory:

- To free allocated stream RAM, stop stream. To use less stream RAM, use a smaller STREAM_BUFFER_SIZE_BYTES and/or smaller STREAM_OUT#(0:3)_BUFFER_ALLOCATE_NUM_BYTES
- To free other allocated RAM:
- Write 0 LUA_RUN
- Write 0 USER_RAM_FIFO#(0:3)_ALLOCATE_NUM_BYTES
- Write 0 AIN#(0:149)_EF_INDEX

RAM-Allocating Registers

The below registers allocate system RAM (and may return SYSTEM_MEMORY_BEREFT):

Stream

Name	Start Address	Type	Access
STREAM_ENABLE	4990	UINT32	R/W

STREAM_ENABLE
- Address: 4990

Write 1 to start stream. Write 0 to stop stream. Reading this register returns 1 when stream is enabled. When using a triggered stream the stream is considered enabled while waiting for the trigger.

- Data type: UINT32 (type index = 1)
- Readable and writable
- This register uses system RAM. The maximum RAM is 64KB. For more information, see 4.4 RAM
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 0.9421

STREAM_OUT#(0:2)_ENABLE	4090	UINT32	R/W

STREAM_OUT#(0:2)_ENABLE
- Starting Address: 4090

When STREAM_OUT#_ENABLE is enabled, the stream out target is generally updated by one value from the stream out buffer per stream scan. For example, there will generally be one instance of e.g. STREAM_OUT0 in the stream scan list, which will cause one STREAM_OUT0_BUFFER value to be consumed and written to STREAM_OUT0_TARGET for every stream scan. The stream scan list could also contain two instances of STREAM_OUT0, in which case two values from STREAM_OUT0_BUFFER value would be consumed and written for every stream scan.

- Data type: UINT32 (type index = 1)
- Readable and writable
- Default value: 0
- This register uses system RAM. The maximum RAM is 64KB. For more information, see 4.4 RAM
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 0.9300

Constant	Value
DISABLED	0
ENABLED	1

Expanded Names	Addresses
STREAM_OUT0_ENABLE, STREAM_OUT1_ENABLE, STREAM_OUT2_ENABLE	4090, 4092, 4094

STREAM_BUFFER_SIZE_BYTES does not allocate RAM, but it does set the amount of RAM allocated by STREAM_ENABLE. LJM_eStreamStart writes to STREAM_ENABLE.

Similarly, STREAM_OUT#(0:3)_BUFFER_ALLOCATE_NUM_BYTES does not allocate RAM, but sets the amount of RAM allocated by STREAM_OUT#(0:3)_ENABLE.

Lua

Name	Start Address	Type	Access
LUA_RUN	6000	UINT32	R/W

LUA_RUN
- Address: 6000

Writing 1 compiles and runs the Lua script that is loaded in RAM.
Writing zero stops the script and begins cleaning up memory.
Users may poll the register after writing a value of 0 to verify that the VM is unloaded, and garbage collection is complete.
0 = VM fully unloaded.
1 = Running/in-progress

- Data type: UINT32 (type index = 1)
- Readable and writable
- This register uses system RAM. The maximum RAM is 64KB. For more information, see 4.4 RAM
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0018

LUA_SOURCE_SIZE	6012	UINT32	R/W

LUA_SOURCE_SIZE
- Address: 6012

Allocates RAM for the source code.

- Data type: UINT32 (type index = 1)
- Readable and writable
- This register uses system RAM. The maximum RAM is 64KB. For more information, see 4.4 RAM
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0018

LUA_DEBUG_ENABLE	6020	UINT32	R/W

LUA_DEBUG_ENABLE
- Address: 6020

Write 1 to this register to enable debugging.

- Data type: UINT32 (type index = 1)
- Readable and writable
- This register uses system RAM. The maximum RAM is 64KB. For more information, see 4.4 RAM
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0018

Lua scripts dynamically allocate RAM.

User

Name	Start Address	Type	Access
USER_RAM_FIFO#(0:2)_ALLOCATE_NUM_BYTES	47900	UINT32	R/W

USER_RAM_FIFO#(0:2)_ALLOCATE_NUM_BYTES
- Starting Address: 47900

Allocate memory for a FIFO buffer. Number of bytes should be sufficient to store users max transfer array size. Note that FLOAT32, INT32, and UINT32 require 4 bytes per value, and UINT16 require 2 bytes per value. Maximum size is limited by available memory. Care should be taken to conserve enough memory for other operations such as AIN_EF, Lua, Stream etc.

- Data type: UINT32 (type index = 1)
- Readable and writable
- Default value: 0
- This register uses system RAM. The maximum RAM is 64KB. For more information, see 4.4 RAM
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0163

Expanded Names	Addresses
USER_RAM_FIFO0_ALLOCATE_NUM_BYTES, USER_RAM_FIFO1_ALLOCATE_NUM_BYTES, USER_RAM_FIFO2_ALLOCATE_NUM_BYTES	47900, 47902, 47904

AIN Extended Features

Name	Start Address	Type	Access
AIN#(0:148)_EF_INDEX	9000	UINT32	R/W

AIN#(0:148)_EF_INDEX
- Starting Address: 9000

Specify the desired extended feature for this analog input with the index value. List of index values:
0=None(disabled);
1=Offset and Slope;
3=Max/Min/Avg;
4=Resistance;
5=Average and Threshold;
10=RMS Flex;
11=FlexRMS;
20=Thermocouple type E;
21=Thermocouple type J;
22=Thermocouple type K;
23=Thermocouple type R;
24=Thermocouple type T;
25=Thermocouple type S;
30=Thermocouple type C;
40=RTD model PT100;
41=RTD model PT500;
42=RTD model PT1000.

- Data type: UINT32 (type index = 1)
- Readable and writable
- Default value: 0
- This register uses system RAM. The maximum RAM is 64KB. For more information, see 4.4 RAM
- T8:
    - Minimum firmware version: 0.0123
- T7:
    - Minimum firmware version: 1.0030
- T4:
    - The T4 does not support thermocouple modes.

Expanded Names	Addresses
AIN0_EF_INDEX, AIN1_EF_INDEX,	9000, 9002,
AIN2_EF_INDEX, AIN3_EF_INDEX,	9004, 9006,
AIN4_EF_INDEX, AIN5_EF_INDEX,	9008, 9010,
AIN6_EF_INDEX, AIN7_EF_INDEX,	9012, 9014,
AIN8_EF_INDEX, AIN9_EF_INDEX,	9016, 9018,
AIN10_EF_INDEX, AIN11_EF_INDEX,	9020, 9022,
AIN12_EF_INDEX, AIN13_EF_INDEX,	9024, 9026,
AIN14_EF_INDEX, AIN15_EF_INDEX,	9028, 9030,
AIN16_EF_INDEX, AIN17_EF_INDEX,	9032, 9034,
AIN18_EF_INDEX, AIN19_EF_INDEX,	9036, 9038,
AIN20_EF_INDEX, AIN21_EF_INDEX,	9040, 9042,
AIN22_EF_INDEX, AIN23_EF_INDEX,	9044, 9046,
AIN24_EF_INDEX, AIN25_EF_INDEX,	9048, 9050,
AIN26_EF_INDEX, AIN27_EF_INDEX,	9052, 9054,
AIN28_EF_INDEX, AIN29_EF_INDEX,	9056, 9058,
AIN30_EF_INDEX, AIN31_EF_INDEX,	9060, 9062,
AIN32_EF_INDEX, AIN33_EF_INDEX,	9064, 9066,
AIN34_EF_INDEX, AIN35_EF_INDEX,	9068, 9070,
AIN36_EF_INDEX, AIN37_EF_INDEX,	9072, 9074,
AIN38_EF_INDEX, AIN39_EF_INDEX,	9076, 9078,
AIN40_EF_INDEX, AIN41_EF_INDEX,	9080, 9082,
AIN42_EF_INDEX, AIN43_EF_INDEX,	9084, 9086,
AIN44_EF_INDEX, AIN45_EF_INDEX,	9088, 9090,
AIN46_EF_INDEX, AIN47_EF_INDEX,	9092, 9094,
AIN48_EF_INDEX, AIN49_EF_INDEX,	9096, 9098,
AIN50_EF_INDEX, AIN51_EF_INDEX,	9100, 9102,
AIN52_EF_INDEX, AIN53_EF_INDEX,	9104, 9106,
AIN54_EF_INDEX, AIN55_EF_INDEX,	9108, 9110,
AIN56_EF_INDEX, AIN57_EF_INDEX,	9112, 9114,
AIN58_EF_INDEX, AIN59_EF_INDEX,	9116, 9118,
AIN60_EF_INDEX, AIN61_EF_INDEX,	9120, 9122,
AIN62_EF_INDEX, AIN63_EF_INDEX,	9124, 9126,
AIN64_EF_INDEX, AIN65_EF_INDEX,	9128, 9130,
AIN66_EF_INDEX, AIN67_EF_INDEX,	9132, 9134,
AIN68_EF_INDEX, AIN69_EF_INDEX,	9136, 9138,
AIN70_EF_INDEX, AIN71_EF_INDEX,	9140, 9142,
AIN72_EF_INDEX, AIN73_EF_INDEX,	9144, 9146,
AIN74_EF_INDEX, AIN75_EF_INDEX,	9148, 9150,
AIN76_EF_INDEX, AIN77_EF_INDEX,	9152, 9154,
AIN78_EF_INDEX, AIN79_EF_INDEX,	9156, 9158,
AIN80_EF_INDEX, AIN81_EF_INDEX,	9160, 9162,
AIN82_EF_INDEX, AIN83_EF_INDEX,	9164, 9166,
AIN84_EF_INDEX, AIN85_EF_INDEX,	9168, 9170,
AIN86_EF_INDEX, AIN87_EF_INDEX,	9172, 9174,
AIN88_EF_INDEX, AIN89_EF_INDEX,	9176, 9178,
AIN90_EF_INDEX, AIN91_EF_INDEX,	9180, 9182,
AIN92_EF_INDEX, AIN93_EF_INDEX,	9184, 9186,

AIN94_EF_INDEX, AIN95_EF_INDEX,	9188, 9190,
AIN96_EF_INDEX, AIN97_EF_INDEX,	9192, 9194,
AIN98_EF_INDEX, AIN99_EF_INDEX,	9196, 9198,
AIN100_EF_INDEX, AIN101_EF_INDEX,	9200, 9202,
AIN102_EF_INDEX, AIN103_EF_INDEX,	9204, 9206,
AIN104_EF_INDEX, AIN105_EF_INDEX,	9208, 9210,
AIN106_EF_INDEX, AIN107_EF_INDEX,	9212, 9214,
AIN108_EF_INDEX, AIN109_EF_INDEX,	9216, 9218,
AIN110_EF_INDEX, AIN111_EF_INDEX,	9220, 9222,
AIN112_EF_INDEX, AIN113_EF_INDEX,	9224, 9226,
AIN114_EF_INDEX, AIN115_EF_INDEX,	9228, 9230,
AIN116_EF_INDEX, AIN117_EF_INDEX,	9232, 9234,
AIN118_EF_INDEX, AIN119_EF_INDEX,	9236, 9238,
AIN120_EF_INDEX, AIN121_EF_INDEX,	9240, 9242,
AIN122_EF_INDEX, AIN123_EF_INDEX,	9244, 9246,
AIN124_EF_INDEX, AIN125_EF_INDEX,	9248, 9250,
AIN126_EF_INDEX, AIN127_EF_INDEX,	9252, 9254,
AIN128_EF_INDEX, AIN129_EF_INDEX,	9256, 9258,
AIN130_EF_INDEX, AIN131_EF_INDEX,	9260, 9262,
AIN132_EF_INDEX, AIN133_EF_INDEX,	9264, 9266,
AIN134_EF_INDEX, AIN135_EF_INDEX,	9268, 9270,
AIN136_EF_INDEX, AIN137_EF_INDEX,	9272, 9274,
AIN138_EF_INDEX, AIN139_EF_INDEX,	9276, 9278,
AIN140_EF_INDEX, AIN141_EF_INDEX,	9280, 9282,
AIN142_EF_INDEX, AIN143_EF_INDEX,	9284, 9286,
AIN144_EF_INDEX, AIN145_EF_INDEX,	9288, 9290,
AIN146_EF_INDEX, AIN147_EF_INDEX,	9292, 9294,
AIN148_EF_INDEX	9296

AIN#(0:148)_EF_READ_A	7000	FLOAT32	R

AIN#(0:148)_EF_READ_A
- Starting Address: 7000

Function dependent on selected feature index.

- Data type: FLOAT32 (type index = 3)
- Read-only
- Default value: 0
- This register uses system RAM. The maximum RAM is 64KB. For more information, see 4.4 RAM
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0030

Expanded Names	Addresses
AIN0_EF_READ_A, AIN1_EF_READ_A,	7000, 7002,
AIN2_EF_READ_A, AIN3_EF_READ_A,	7004, 7006,
AIN4_EF_READ_A, AIN5_EF_READ_A,	7008, 7010,
AIN6_EF_READ_A, AIN7_EF_READ_A,	7012, 7014,
AIN8_EF_READ_A, AIN9_EF_READ_A,	7016, 7018,
AIN10_EF_READ_A, AIN11_EF_READ_A,	7020, 7022,
AIN12_EF_READ_A, AIN13_EF_READ_A,	7024, 7026,
AIN14_EF_READ_A, AIN15_EF_READ_A,	7028, 7030,
AIN16_EF_READ_A, AIN17_EF_READ_A,	7032, 7034,
AIN18_EF_READ_A, AIN19_EF_READ_A,	7036, 7038,
AIN20_EF_READ_A, AIN21_EF_READ_A,	7040, 7042,
AIN22_EF_READ_A, AIN23_EF_READ_A,	7044, 7046,
AIN24_EF_READ_A, AIN25_EF_READ_A,	7048, 7050,

AIN26_EF_READ_A, AIN27_EF_READ_A,	7052, 7054,
AIN28_EF_READ_A, AIN29_EF_READ_A,	7056, 7058,
AIN30_EF_READ_A, AIN31_EF_READ_A,	7060, 7062,
AIN32_EF_READ_A, AIN33_EF_READ_A,	7064, 7066,
AIN34_EF_READ_A, AIN35_EF_READ_A,	7068, 7070,
AIN36_EF_READ_A, AIN37_EF_READ_A,	7072, 7074,
AIN38_EF_READ_A, AIN39_EF_READ_A,	7076, 7078,
AIN40_EF_READ_A, AIN41_EF_READ_A,	7080, 7082,
AIN42_EF_READ_A, AIN43_EF_READ_A,	7084, 7086,
AIN44_EF_READ_A, AIN45_EF_READ_A,	7088, 7090,
AIN46_EF_READ_A, AIN47_EF_READ_A,	7092, 7094,
AIN48_EF_READ_A, AIN49_EF_READ_A,	7096, 7098,
AIN50_EF_READ_A, AIN51_EF_READ_A,	7100, 7102,
AIN52_EF_READ_A, AIN53_EF_READ_A,	7104, 7106,
AIN54_EF_READ_A, AIN55_EF_READ_A,	7108, 7110,
AIN56_EF_READ_A, AIN57_EF_READ_A,	7112, 7114,
AIN58_EF_READ_A, AIN59_EF_READ_A,	7116, 7118,
AIN60_EF_READ_A, AIN61_EF_READ_A,	7120, 7122,
AIN62_EF_READ_A, AIN63_EF_READ_A,	7124, 7126,
AIN64_EF_READ_A, AIN65_EF_READ_A,	7128, 7130,
AIN66_EF_READ_A, AIN67_EF_READ_A,	7132, 7134,
AIN68_EF_READ_A, AIN69_EF_READ_A,	7136, 7138,
AIN70_EF_READ_A, AIN71_EF_READ_A,	7140, 7142,
AIN72_EF_READ_A, AIN73_EF_READ_A,	7144, 7146,
AIN74_EF_READ_A, AIN75_EF_READ_A,	7148, 7150,
AIN76_EF_READ_A, AIN77_EF_READ_A,	7152, 7154,
AIN78_EF_READ_A, AIN79_EF_READ_A,	7156, 7158,
AIN80_EF_READ_A, AIN81_EF_READ_A,	7160, 7162,
AIN82_EF_READ_A, AIN83_EF_READ_A,	7164, 7166,
AIN84_EF_READ_A, AIN85_EF_READ_A,	7168, 7170,
AIN86_EF_READ_A, AIN87_EF_READ_A,	7172, 7174,
AIN88_EF_READ_A, AIN89_EF_READ_A,	7176, 7178,
AIN90_EF_READ_A, AIN91_EF_READ_A,	7180, 7182,
AIN92_EF_READ_A, AIN93_EF_READ_A,	7184, 7186,
AIN94_EF_READ_A, AIN95_EF_READ_A,	7188, 7190,
AIN96_EF_READ_A, AIN97_EF_READ_A,	7192, 7194,
AIN98_EF_READ_A, AIN99_EF_READ_A,	7196, 7198,
AIN100_EF_READ_A, AIN101_EF_READ_A,	7200, 7202,
AIN102_EF_READ_A, AIN103_EF_READ_A,	7204, 7206,
AIN104_EF_READ_A, AIN105_EF_READ_A,	7208, 7210,
AIN106_EF_READ_A, AIN107_EF_READ_A,	7212, 7214,
AIN108_EF_READ_A, AIN109_EF_READ_A,	7216, 7218,
AIN110_EF_READ_A, AIN111_EF_READ_A,	7220, 7222,
AIN112_EF_READ_A, AIN113_EF_READ_A,	7224, 7226,
AIN114_EF_READ_A, AIN115_EF_READ_A,	7228, 7230,
AIN116_EF_READ_A, AIN117_EF_READ_A,	7232, 7234,
AIN118_EF_READ_A, AIN119_EF_READ_A,	7236, 7238,
AIN120_EF_READ_A, AIN121_EF_READ_A,	7240, 7242,
AIN122_EF_READ_A, AIN123_EF_READ_A,	7244, 7246,
AIN124_EF_READ_A, AIN125_EF_READ_A,	7248, 7250,
AIN126_EF_READ_A, AIN127_EF_READ_A,	7252, 7254,
AIN128_EF_READ_A, AIN129_EF_READ_A,	7256, 7258,
AIN130_EF_READ_A, AIN131_EF_READ_A,	7260, 7262,
AIN132_EF_READ_A, AIN133_EF_READ_A,	7264, 7266,
AIN134_EF_READ_A, AIN135_EF_READ_A,	7268, 7270,
AIN136_EF_READ_A, AIN137_EF_READ_A,	7272, 7274,
AIN138_EF_READ_A, AIN139_EF_READ_A,	7276, 7278,
AIN140_EF_READ_A, AIN141_EF_READ_A,	7280, 7282,
AIN142_EF_READ_A, AIN143_EF_READ_A,	7284, 7286,
AIN144_EF_READ_A, AIN145_EF_READ_A,	7288, 7290,
AIN146_EF_READ_A, AIN147_EF_READ_A,	7292, 7294,
AIN148_EF_READ_A	7296

File I/O

Name	Start Address	Type	Access
FILE_IO_PATH_WRITE_LEN_BYTES	60640	UINT32	W

FILE_IO_PATH_WRITE_LEN_BYTES
- Address: 60640

Write the length (in bytes) of the file path or directory to access.

- Data type: UINT32 (type index = 1)
- Write-only
- This register uses system RAM. The maximum RAM is 64KB. For more information, see 4.4 RAM
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0134

SPI

Name	Start Address	Type	Access
SPI_GO	5007	UINT16	W

SPI_GO
- Address: 5007

Write 1 to begin the configured SPI transaction.

- Data type: UINT16 (type index = 0)
- Write-only
- Default value: 0
- This register uses system RAM. The maximum RAM is 64KB. For more information, see 4.4 RAM

Serial

Name	Start Address	Type	Access
ASYNCH_NUM_BYTES_TX	5440	UINT16	R/W

ASYNCH_NUM_BYTES_TX
- Address: 5440

The number of bytes to be transmitted after writing to GO. Max is 256.

- Data type: UINT16 (type index = 0)
- Readable and writable
- Default value: 0
- This register uses system RAM. The maximum RAM is 64KB. For more information, see 4.4 RAM

ASYNCH_ENABLE	5400	UINT16	R/W

ASYNCH_ENABLE
- Address: 5400

1 = Turn on Asynch. Configures timing hardware, DIO lines and allocates the receiving buffer.

- Data type: UINT16 (type index = 0)
- Readable and writable
- Default value: 0
- This register uses system RAM. The maximum RAM is 64KB. For more information, see 4.4 RAM

Pre-Allocated User RAM Registers

User RAM consists of a list of volatile Modbus addresses where data can be sent to, and read from, a Lua script. Lua writes to the Modbus registers, and then a host device can read that information.

There are a total of 200 registers of pre-allocated RAM, which is split into several groups so that users may access it conveniently with different data types.

Use the following USER_RAM registers to store information:

User RAM Registers

Name	Start Address	Type	Access
USER_RAM#(0:39)_F32	46000	FLOAT32	R/W

USER_RAM#(0:39)_F32
- Starting Address: 46000

Generic RAM registers. Useful for passing data between a host computer and a Lua script. Will not return an error if alternate data types are used.

- Data type: FLOAT32 (type index = 3)
- Readable and writable
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0023

Expanded Names	Addresses
USER_RAM0_F32, USER_RAM1_F32,	46000, 46002,
USER_RAM2_F32, USER_RAM3_F32,	46004, 46006,
USER_RAM4_F32, USER_RAM5_F32,	46008, 46010,
USER_RAM6_F32, USER_RAM7_F32,	46012, 46014,
USER_RAM8_F32, USER_RAM9_F32,	46016, 46018,
USER_RAM10_F32, USER_RAM11_F32,	46020, 46022,
USER_RAM12_F32, USER_RAM13_F32,	46024, 46026,
USER_RAM14_F32, USER_RAM15_F32,	46028, 46030,
USER_RAM16_F32, USER_RAM17_F32,	46032, 46034,
USER_RAM18_F32, USER_RAM19_F32,	46036, 46038,
USER_RAM20_F32, USER_RAM21_F32,	46040, 46042,
USER_RAM22_F32, USER_RAM23_F32,	46044, 46046,
USER_RAM24_F32, USER_RAM25_F32,	46048, 46050,
USER_RAM26_F32, USER_RAM27_F32,	46052, 46054,
USER_RAM28_F32, USER_RAM29_F32,	46056, 46058,
USER_RAM30_F32, USER_RAM31_F32,	46060, 46062,
USER_RAM32_F32, USER_RAM33_F32,	46064, 46066,
USER_RAM34_F32, USER_RAM35_F32,	46068, 46070,
USER_RAM36_F32, USER_RAM37_F32,	46072, 46074,
USER_RAM38_F32, USER_RAM39_F32	46076, 46078

USER_RAM#(0:9)_I32	46080	INT32	R/W

USER_RAM#(0:9)_I32
- Starting Address: 46080

Generic RAM registers. Useful for passing data between a host computer and a Lua script. Will not return an error if alternate data types are used.

- Data type: INT32 (type index = 2)
- Readable and writable
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0162

Expanded Names	Addresses
USER_RAM0_I32, USER_RAM1_I32, USER_RAM2_I32, USER_RAM3_I32, USER_RAM4_I32, USER_RAM5_I32, USER_RAM6_I32, USER_RAM7_I32, USER_RAM8_I32, USER_RAM9_I32	46080, 46082, 46084, 46086, 46088, 46090, 46092, 46094, 46096, 46098

USER_RAM#(0:39)_U32	46100	UINT32	R/W

USER_RAM#(0:39)_U32
- Starting Address: 46100

Generic RAM registers. Useful for passing data between a host computer and a Lua script. Will not return an error if alternate data types are used.

- Data type: UINT32 (type index = 1)
- Readable and writable
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0162

Expanded Names	Addresses
USER_RAM0_U32, USER_RAM1_U32,	46100, 46102,
USER_RAM2_U32, USER_RAM3_U32,	46104, 46106,
USER_RAM4_U32, USER_RAM5_U32,	46108, 46110,
USER_RAM6_U32, USER_RAM7_U32,	46112, 46114,
USER_RAM8_U32, USER_RAM9_U32,	46116, 46118,
USER_RAM10_U32, USER_RAM11_U32,	46120, 46122,
USER_RAM12_U32, USER_RAM13_U32,	46124, 46126,
USER_RAM14_U32, USER_RAM15_U32,	46128, 46130,
USER_RAM16_U32, USER_RAM17_U32,	46132, 46134,
USER_RAM18_U32, USER_RAM19_U32,	46136, 46138,
USER_RAM20_U32, USER_RAM21_U32,	46140, 46142,
USER_RAM22_U32, USER_RAM23_U32,	46144, 46146,
USER_RAM24_U32, USER_RAM25_U32,	46148, 46150,
USER_RAM26_U32, USER_RAM27_U32,	46152, 46154,
USER_RAM28_U32, USER_RAM29_U32,	46156, 46158,
USER_RAM30_U32, USER_RAM31_U32,	46160, 46162,
USER_RAM32_U32, USER_RAM33_U32,	46164, 46166,
USER_RAM34_U32, USER_RAM35_U32,	46168, 46170,
USER_RAM36_U32, USER_RAM37_U32,	46172, 46174,
USER_RAM38_U32, USER_RAM39_U32	46176, 46178

USER_RAM#(0:19)_U16	46180	UINT16	R/W

USER_RAM#(0:19)_U16
- Starting Address: 46180

Generic RAM registers. Useful for passing data between a host computer and a Lua script. Will not return an error if alternate data types are used.

- Data type: UINT16 (type index = 0)
- Readable and writable
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0162

Expanded Names	Addresses
USER_RAM0_U16, USER_RAM1_U16, USER_RAM2_U16, USER_RAM3_U16, USER_RAM4_U16, USER_RAM5_U16, USER_RAM6_U16, USER_RAM7_U16, USER_RAM8_U16, USER_RAM9_U16, USER_RAM10_U16, USER_RAM11_U16, USER_RAM12_U16, USER_RAM13_U16, USER_RAM14_U16, USER_RAM15_U16, USER_RAM16_U16, USER_RAM17_U16, USER_RAM18_U16, USER_RAM19_U16	46180, 46181, 46182, 46183, 46184, 46185, 46186, 46187, 46188, 46189, 46190, 46191, 46192, 46193, 46194, 46195, 46196, 46197, 46198, 46199

Power Mode Registers - T4/T7 Only

The following registers are unsupported on the T8. These registers can be used to reduce the device power consumption. We recommend that most users do not change these registers. See the power consumption charts in Appendix-A-5.

Power Registers

Name	Start Address	Type	Access
POWER_MODE	48000	UINT16	R/W

POWER_MODE
- Address: 48000

The current power management state.

- Data type: UINT16 (type index = 0)
- Readable and writable
- T7:
  - Minimum firmware version: 1.0224

POWER_CORE	48001	UINT16	R/W

POWER_CORE
- Address: 48001

The core processor speed. 0=80MHz, 1=20MHz, 2=2MHz, 3=250kHz

- Data type: UINT16 (type index = 0)
- Readable and writable
- T7:
  - Minimum firmware version: 1.0200
- T4:
  - Minimum firmware version: 1.0000

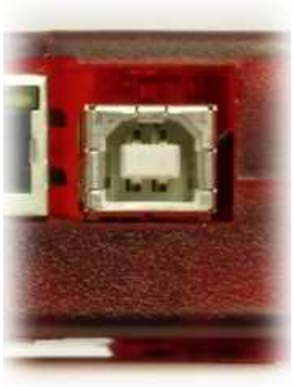
POWER_USB	48002	UINT16	R/W

POWER_USB
- Address: 48002

The current ON/OFF state of the USB module.

- Data type: UINT16 (type index = 0)
- Readable and writable
- T7:
  - Minimum firmware version: 1.0200
- T4:
  - Minimum firmware version: 1.0000

5.0 USB [T-Series Datasheet]



LabJack USB Type-B connector

Connector Type: **USB-B Receptacle**

Compatible: **USB 1.1+**

Max Cable Length: **5 meters**

Max Packet Size: **64 bytes/packet**

The USB connection can supply power, communication, or both.

When USB is used for power, the 5V supply from the USB cable will be used to power the T-Series device. The cable can transfer power from a host computer's USB system or from a 5V wall adapter. Section 9 contains additional information about power supply options.

T-Series devices that are powered through the USB connection can communicate via USB, Ethernet or WiFi (if available).

USB Speeds

USB speeds and other specifications are summarized on Wikipedia, and described in complete detail on the USB website.

- The T4 and T7 are "Full-Speed" USB 2.0 devices and will always establish a "Full-Speed" connection.
- The T8 is a "High-Speed" USB 2.0 device. It will establish a "High-Speed" connection when connected to a USB 2.0 (or higher) system, or "Full-Speed" connection when connected to a USB 1.1 system.

"But I Don't Know How To Speak USB?"

Don't worry ... the LJM Library takes care of the USB details. Just call the Open function to get a handle to a particular device (e.g. HandleA), and then make simple calls such as eReadName(HandleA, "AIN0") which returns the voltage from analog input 0. We have examples for many development environments and even some simple ready-to-run software.

Power Considerations

USB ground is connected to the T-series device's ground (GND), since standard USB is non-isolated. USB ground is generally the same as the ground of the PC chassis and AC mains.

If electrical isolation between the T-series device and host is desired, use Ethernet (or WiFi) instead, or add a USB isolator.

Any host port or self-powered hub port should provide 4.75 - 5.25 volts at any current up to 500 mA, per USB specifications. This is sufficient to power a T4 or T7. When using a T8, a USB hub with a 2.5 to 3 A power supply is recommended. Note that some hubs are not up to USB specification, and at higher currents the supplied voltage drops below 4.75 volts.. Appendix A-5 provides additional information about device power supply requirements.

Designing a Custom Driver

We strongly recommend using our LJM library rather than creating a custom USB driver. LJM provides convenient device discovery, high-level functions, and programming flexibility.

A custom USB host interface will need to implement the basic operations of USB: find, open, write, read, and close. For read and write operations, packets should be formatted as Modbus TCP frames.

The USB interface on T-Series devices consists of the normal bidirectional control endpoint (0 OUT & IN), 3 used bulk endpoints (1 OUT, 2 IN, 3 IN), and 1 dummy endpoint (3 OUT).

- Endpoint 1 - OUT, address 0x01.
- Endpoint 2 - IN, address 0x82
- Endpoint 3 - OUT address = 0x03, IN address = 0x83. Endpoint 3 OUT is not supported, and should never be used.

The size of the endpoints can vary. The T4 and T7 will always use 64-byte endpoints. The T8 will set its endpoints to 512 bytes when operating in high-speed mode, and 64 bytes when operating in low-speed mode.

All commands should be sent on Endpoint 1. The responses to commands will always be returned on Endpoint 2. Endpoint 3 is only used to send stream data from the T-Series device to the host.

The LabJack vendor ID is 0x0CD5.

- T4 product ID is 0x0004.
- T7 product ID is 0x0007.
- T8 product ID is 0x0008;

Troubleshooting

If USB communication is not working, see USB Communication Failure.

6.0 Ethernet [T-Series Datasheet]

LabJack RJ-45/Ethernet Connector

Communication Protocol: **Modbus TCP**

Connector Type: **RJ-45 Socket, Cat 5**

Compatible: **10/100Base-T, Auto-MDIX**

PoE (Power over Ethernet): **T8 only. T4 or T7 See PoE App Note.**

Max Cable Length: **100 meters typical**

Max Packet Size: **1040 bytes/packet (TCP), 64 bytes/packet (UDP)**

The T-series devices each have a 10/100Base-T Ethernet connection. Depending on the device, this connection can be used for communication, power, or both.

- T4, T7 - This connection only provides communication. Power must be provided through the USB connector.
- T8 - This connection can be used for communication or power or both. To supply power over the Ethernet connection, there must be PSE, Power Supplying Equipment, upstream of the T8.

Refer to out WiFi and Ethernet tutorial to get started with Ethernet communication.

Ports

T-series devices will respond to connections or packets on the following ports:

- 502 (TCP) - Responds to Modbus TCP packets received over this port.
- 702 (TCP) - Sends spontaneous data such as stream to devices connected to this port.
- 52362 (UDP) - Responds to Modbus UDP packets received over this port. Usually used to search for T-series devices on a network.

If you need a wireless connection instead of Ethernet, you can buy a wireless bridge. Connect the T-series device to the bridge and the bridge will connect to the wireless network. Find more information in the Convert Ethernet to WiFi App Note.

The following table describes how many simultaneous connections can be maintained per connection type.

Connection Type	Number of Simultaneous Connections
TCP (port 502, Modbus TCP server)	2
TCP (port 702, spontaneous data only)	1
UDP (port 52362)	Unlimited*

* UDP is connectionless and it does not claim any device connection resources.

"But I Don't Know How To Speak Ethernet?"

Don't worry, the LJM Library takes care of the Ethernet details. You just make our Open call to get a handle to a particular device (e.g. HandleA), and then make simple calls such as eReadName(HandleA, "AIN0") which returns the voltage from analog input 0. We have examples for many development environments and even some simple ready-to-run software.

Config (_DEFAULT) Registers

Use the following _DEFAULT registers to configure Ethernet:

Ethernet Config Registers

Name	Start Address	Type	Access
ETHERNET_IP_DEFAULT	49150	UINT32	R/W

ETHERNET_IP_DEFAULT
- Address: 49150

The IP address of wired Ethernet after a power-cycle to the device.

- Data type: UINT32 (type index = 1)
- Readable and writable

ETHERNET_SUBNET_DEFAULT	49152	UINT32	R/W

ETHERNET_SUBNET_DEFAULT
- Address: 49152

The subnet of wired Ethernet after a power-cycle to the device.

- Data type: UINT32 (type index = 1)
- Readable and writable

ETHERNET_GATEWAY_DEFAULT	49154	UINT32	R/W

ETHERNET_GATEWAY_DEFAULT
- Address: 49154

The gateway of wired Ethernet after a power-cycle to the device.

- Data type: UINT32 (type index = 1)
- Readable and writable

ETHERNET_DNS_DEFAULT	49156	UINT32	R/W

ETHERNET_DNS_DEFAULT
- Address: 49156

The DNS of wired Ethernet after a power-cycle to the device.

- Data type: UINT32 (type index = 1)
- Readable and writable

ETHERNET_ALTDNS_DEFAULT	49158	UINT32	R/W

ETHERNET_ALTDNS_DEFAULT
- Address: 49158

The Alt DNS of wired Ethernet after a power-cycle to the device.

- Data type: UINT32 (type index = 1)
- Readable and writable

ETHERNET_DHCP_ENABLE_DEFAULT	49160	UINT16	R/W

ETHERNET_DHCP_ENABLE_DEFAULT
- Address: 49160

The Enabled/Disabled state of Ethernet DHCP after a power-cycle to the device.

- Data type: UINT16 (type index = 0)
- Readable and writable

ETHERNET_APPLY_SETTINGS	49190	UINT32	W

ETHERNET_APPLY_SETTINGS
- Address: 49190

Writing 1 to this register power-cycles Ethernet. It tells the device to waits 1s before turning off Ethernet and then 500ms before turning it back on.

- Data type: UINT32 (type index = 1)
- Write-only
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0103
- T4:
  - Minimum firmware version: 0.2000

These registers can also be read. Configure the Ethernet network configurations in Kipling software.

These _DEFAULT registers are non-volatile. Whatever value you write to a _DEFAULT register will be retained through a reboot or power-cycle. New values written to these _DEFAULT registers are not applied until power-up or until a 1 is written to ETHERNET_APPLY_SETTINGS.

Status Registers

Use the following read-only registers to read the status of Ethernet:

Ethernet Status Registers

Name	Start Address	Type	Access
ETHERNET_IP	49100	UINT32	R

ETHERNET_IP
- Address: 49100

Read the current IP address of wired Ethernet.

- Data type: UINT32 (type index = 1)
- Read-only

ETHERNET_SUBNET	49102	UINT32	R

ETHERNET_SUBNET
- Address: 49102

Read the current subnet of wired Ethernet.

- Data type: UINT32 (type index = 1)
- Read-only

ETHERNET_GATEWAY	49104	UINT32	R

ETHERNET_GATEWAY
- Address: 49104

Read the current gateway of wired Ethernet.

- Data type: UINT32 (type index = 1)
- Read-only

ETHERNET_DNS	49106	UINT32	R

ETHERNET_DNS
- Address: 49106

Read the current DNS of wired Ethernet.

- Data type: UINT32 (type index = 1)
- Read-only

ETHERNET_ALTDNS	49108	UINT32	R

ETHERNET_ALTDNS
- Address: 49108

Read the current Alt DNS of wired Ethernet.

- Data type: UINT32 (type index = 1)
- Read-only

ETHERNET_DHCP_ENABLE	49110	UINT16	R

ETHERNET_DHCP_ENABLE
- Address: 49110

Read the current Enabled/Disabled state of Ethernet DHCP.

- Data type: UINT16 (type index = 0)
- Read-only

Config (_DEFAULT) vs. Status Registers

Example: If you write ETHERNET_IP_DEFAULT="192.168.1.207" (you would actually write/read the 32-bit numeric equivalent—not an IP string), then that value will be retained through reboots and is the default IP address. If DHCP is disabled, this will be the static IP of the device and what you get if you read ETHERNET_IP. If DHCP is enabled, then a read of ETHERNET_IP will return the IP set by the DHCP server.

Power

The following registers configure Ethernet power:

Name	Start Address	Type	Access
POWER_ETHERNET	48003	UINT16	R/W

POWER_ETHERNET
- Address: 48003

The current ON/OFF state of the Ethernet module.

- Data type: UINT16 (type index = 0)
- Readable and writable
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 0.8600
- T4:
  - Minimum firmware version: 0.0100

POWER_ETHERNET_DEFAULT	48053	UINT16	R/W

POWER_ETHERNET_DEFAULT
- Address: 48053

The ON/OFF state of the Ethernet module after a power-cycle to the device. Provided to optionally reduce power consumption.

- Data type: UINT16 (type index = 0)
- Readable and writable
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 0.8600
- T4:
  - Minimum firmware version: 0.0100

Some Examples

Read IP Example: To read the wired IP Address of a device, perform a Modbus read of address 49100. The value will be returned as an unsigned 32-bit number, such as 3232235691. Change this number to an IP address by converting each binary group to an octet, and adding decimal points as necessary. The result in this case would be "192.168.0.171".

Change IP Example: To change the Ethernet IP Address of a device, perform a Modbus write to address 49150. The value must be passed as an unsigned 32-bit number, such as 3232235691. Change this IP address "192.168.0.171" by converting each octet to a binary group, and sticking them together.

Default IP Address

The default is DHCP enabled. If your network does not have a DHCP server, there are a few options to talk to the device so you can change Ethernet settings:

- Use USB.
- Connect to a network with a router (DHCP server).
- Use the SPC-to-AIN3 jumper to temporarily force a static IP.

Isolation

The Ethernet connection on any T-series device has 1500 volts of galvanic isolation. All power supplies shipped by LabJack Corporation with T-series devices have at least 500 volts of isolation.

Note that if you power the T-series device from a USB host/hub, ground from the host/hub is typically connected to upstream USB ground, which often finds its way back to AC mains ground and thus there would be no isolation.

LEDs on the Ethernet jack

Both the green and orange LEDs on the Ethernet jack will illuminate on connection to an active Ethernet cable. The orange LED turns on when an active link is detected and blinks when packets are received/processed. The green LED illuminates when the connection is 100Mbps.

The orange LED is closest to the USB connector.

UDP Discovery-Only Mode - T7 only

Firmware minimum: 1.0284

It is theoretically possible that an unintended UDP packet could be received by a T-series device and processed. However, this is statistically unlikely. By one calculation, the statistical likelihood of this happening is less than 2.7E-10 percent.

However, if your application needs ultimate reliability or your network has Modbus UDP packets broadcast on port 52362, you can enable a discovery-only mode. Discovery-only mode disables all Modbus functionality except a small list of registers that LJM uses to identify devices. The error UDP_DISCOVERY_ONLY_MODE_IS_ENABLED (2317) indicates that an operation attempted to read/write a register outside of this list while discovery-only mode was enabled.

Use ETHERNET_UDP_DISCOVERY_ONLY to enable/disable discovery-only mode until the device is power-cycled:

Name	Start Address	Type	Access
ETHERNET_UDP_DISCOVERY_ONLY	49115	UINT16	R

ETHERNET_UDP_DISCOVERY_ONLY
- Address: 49115

Restricts which Modbus operations are allowed over UDP. When set to 1, only the registers needed for discovering units can be read and any other operation will throw an error.

- Data type: UINT16 (type index = 0)
- Read-only
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0284

Use the _DEFAULT register version to set the startup behavior for discovery-only mode:

Name	Start Address	Type	Access
ETHERNET_UDP_DISCOVERY_ONLY_DEFAULT	49165	UINT16	R

ETHERNET_UDP_DISCOVERY_ONLY_DEFAULT
- Address: 49165

The Enabled/Disabled state of ETHERNET_UDP_DISCOVERY_ONLY after a power-cycle to the device.

- Data type: UINT16 (type index = 0)
- Read-only
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0284

There are separate WiFi-only configurations for discovery-only mode via WiFi.

7.0 WiFi (T7-Pro only) [T-Series Datasheet]

Communication Protocol: **Modbus TCP**

Connector Type: **Female RP-SMA**

Transceiver: **2.4 GHz 802.11 b/g** (Compatible with ac, and n routers)

Security: **None or WPA2-PSK (WPA2-Personal)**

Range: **Similar to laptops and other WiFi devices with stock antenna**

Max Packet Size: **500 bytes/packet**

The T7-Pro has a wireless module. Refer to this WiFi and Ethernet tutorial to get started.

DHCP is enabled from the factory, so to get WiFi going from the factory write the desired SSID string (case-sensitive, alphanumeric only) to WIFI_SSID_DEFAULT and the proper password string (also case-sensitive, alphanumeric only) to WIFI_PASSWORD_DEFAULT. Then write a 1 to WIFI_APPLY_SETTINGS and watch the status codes. If you get back code 2900 the WiFi chip is associated to your network, and you can then read the assigned IP from WIFI_IP. Find more details and troubleshooting tips in the WiFi and Ethernet tutorial.

Data Speed: It's possible to get data faster on a T7-Pro using its Ethernet interface instead of its WiFi interface, both for command response and streaming modes. Wireless bridges and access points do not introduce a speed bottleneck, so a bridge is a good way to get fast wireless data from any T-series device. See the Convert Ethernet to WiFi App Note for setup information.

Network Requirements

As noted above, the T7-Pro's WiFi module connects to a standard WiFi network. It needs to be 2.4 GHz 802.11 b/g with WPA2-PSK security or no security. If your existing WiFi does not match that there are a couple of easy options:

- Add another access point to create a new WiFi network for your T7-Pros.
- Use the Ethernet connection on the LabJack with an Ethernet<->WiFi bridge. See the Convert Ethernet to WiFi App Note. This solution also provides higher performance than

the built-in WiFi.

Port

The T7's WiFi module responds to connections or packets on port 502 for both UDP and TCP.

"But I Don't Know How To Speak WiFi?"

Don't worry, the LJM Library takes care of the WiFi details. You just make our Open call to get a handle to a particular device (e.g. HandleA), and then make simple calls such as eReadName(HandleA, "AIN0") which returns the voltage from analog input 0. We have examples for many development environments and even some simple ready-to-run software.

Config (_DEFAULT) Registers

Use the following _DEFAULT registers to configure WiFi:

WiFi Config Registers

Name	Start Address	Type	Access
WIFI_IP_DEFAULT	49250	UINT32	R/W

WIFI_IP_DEFAULT
- Address: 49250

The new IP address of WiFi. Use WIFI_APPLY_SETTINGS.

- Data type: UINT32 (type index = 1)
- Readable and writable

WIFI_SUBNET_DEFAULT	49252	UINT32	R/W

WIFI_SUBNET_DEFAULT
- Address: 49252

The new subnet of WiFi. Use WIFI_APPLY_SETTINGS.

- Data type: UINT32 (type index = 1)
- Readable and writable

WIFI_GATEWAY_DEFAULT	49254	UINT32	R/W

WIFI_GATEWAY_DEFAULT
- Address: 49254

The new gateway of WiFi. Use WIFI_APPLY_SETTINGS.

- Data type: UINT32 (type index = 1)
- Readable and writable

WIFI_DHCP_ENABLE_DEFAULT	49260	UINT16	R/W

WIFI_DHCP_ENABLE_DEFAULT
- Address: 49260

The new Enabled/Disabled state of WiFi DHCP. Use WIFI_APPLY_SETTINGS

- Data type: UINT16 (type index = 0)
- Readable and writable
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 0.9014

WIFI_SSID_DEFAULT	49325	STRING	R/W

WIFI_SSID_DEFAULT
- Address: 49325

The new SSID (network name) of WiFi. Use WIFI_APPLY_SETTINGS.

- Data type: STRING (type index = 98)
- Readable and writable

WIFI_PASSWORD_DEFAULT	49350	STRING	W

WIFI_PASSWORD_DEFAULT
- Address: 49350

Write the password for the WiFi network, then use WIFI_APPLY_SETTINGS.

- Data type: STRING (type index = 98)
- Write-only

WIFI_APPLY_SETTINGS	49400	UINT32	W

WIFI_APPLY_SETTINGS
- Address: 49400

Apply all new WiFi settings: IP, Subnet, Gateway, DHCP, SSID, Password. 1=Apply

- Data type: UINT32 (type index = 1)
- Write-only

These registers can also be read, except WIFI_PASSWORD_DEFAULT. Configure the WiFi network configurations in Kipling software.

These _DEFAULT registers are non-volatile. Whatever value you write to a _DEFAULT register will be retained through a reboot or power-cycle. New values written to these _DEFAULT registers are not applied until power-up or until a 1 is written to WIFI_APPLY_SETTINGS.

The network name and password must be alphanumeric. Both are also case-sensitive.

Status Registers

Use the following read-only registers to read the status of WiFi:

WiFi Status Registers

Name	Start Address	Type	Access
WIFI_IP	49200	UINT32	R

WIFI_IP
- Address: 49200

Read the current IP address of WiFi.

- Data type: UINT32 (type index = 1)
- Read-only

WIFI_SUBNET	49202	UINT32	R

WIFI_SUBNET
- Address: 49202

Read the current subnet of WiFi.

- Data type: UINT32 (type index = 1)
- Read-only

WIFI_GATEWAY	49204	UINT32	R

WIFI_GATEWAY
- Address: 49204

Read the current gateway of WiFi.

- Data type: UINT32 (type index = 1)
- Read-only

WIFI_DHCP_ENABLE	49210	UINT16	R

WIFI_DHCP_ENABLE
- Address: 49210

Read the current Enabled/Disabled state of WiFi DHCP.

- Data type: UINT16 (type index = 0)
- Read-only
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 0.9014

WIFI_SSID	49300	STRING	R

WIFI_SSID
- Address: 49300

Read the current SSID (network name) of WiFi

- Data type: STRING (type index = 98)
- Read-only

WIFI_STATUS	49450	UINT32	R

WIFI_STATUS
- Address: 49450

Status Codes: ASSOCIATED = 2900, ASSOCIATING = 2901, ASSOCIATION_FAILED = 2902, UNPOWERED = 2903, BOOTING = 2904, START_FAILED = 2905, APPLYING_SETTINGS = 2906, DHCP_STARTED = 2907, OTHER = 2909

- Data type: UINT32 (type index = 1)
- Read-only
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 0.7500

Constant	Value
ASSOCIATED	2900
ASSOCIATING	2901
ASSOCIATION_FAILED	2902
UNPOWERED	2903
BOOTING	2904
START_FAILED	2905
APPLYING_SETTINGS	2906
DHCP_STARTED	2907
OTHER	2909

WIFI_RSSI	49452	FLOAT32	R

WIFI_RSSI
- Address: 49452

WiFi RSSI (signal strength). Typical values are -40 for very good, and -75 for very weak. The T7 microcontroller only gets a new RSSI value from the WiFi module when WiFi communication occurs.

- Data type: FLOAT32 (type index = 3)
- Read-only
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 0.8000

Config (_DEFAULT) vs. Status Registers

Example: If you write WIFI_IP_DEFAULT="192.168.1.208" (you actually write/read the 32-bit numeric equivalent—not an IP string), then that value will be retained through reboots and is the default IP address. If DHCP is disabled, this will be the static IP of the device and what you

get if you read WIFI_IP. If DHCP is enabled, then a read of WIFI_IP will return the IP set by the DHCP server.

Power

The following registers configure WiFi power:

WiFi Power Registers

Name	Start Address	Type	Access
POWER_WIFI	48004	UINT16	R/W

POWER_WIFI
- Address: 48004

The current ON/OFF state of the WiFi module.

- Data type: UINT16 (type index = 0)
- Readable and writable
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 0.8600

POWER_WIFI_DEFAULT	48054	UINT16	R/W

POWER_WIFI_DEFAULT
- Address: 48054

The ON/OFF state of the WiFi module after a power-cycle to the device. Provided to optionally reduce power consumption.

- Data type: UINT16 (type index = 0)
- Readable and writable
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 0.8600

Examples

Read IP Example: To read the wireless IP address of a device, perform a Modbus read of address 49200. The value will be returned as an unsigned 32-bit number, such as 3232235691. Change this number to an IP address by converting each binary group to an octet, and adding decimal points as necessary. The result in this case would be

"192.168.0.171".

Write IP Example: To change the Wireless IP Address of a device, perform a Modbus write to address 49250. The IP address must be passed as an unsigned 32-bit number, such as 3232235691. Change this IP address "192.168.0.171" by converting each octet to a binary group, and sticking them together.

Antenna Details

The T7-Pro ships with a very common RP-SMA 2.4 GHz antenna similar to the W1030 or A24-HASM-450 from Digikey, or search Amazon for "rp-sma" to find plenty of options. To put an antenna further away from the T7-Pro you can use any standard male-female RP-SMA WiFi extension cable.

The connection to the WiFi module on the T7-Pro PCB is made via a snap-on/snap-off ultra miniature coaxial connector called male U.FL (aka AMC, IPEX, IPAX, IPX, MHF, UMC or UMCC). The normal T7-Pro uses a U.FL to bulkhead RP-SMA cable with a length of 140mm, similar to the Taoglas Limited CAB.622, Emerson 415-0100-150, Laird 1300-00041, Amphenol 336306-14-0150, or Amphenol 336306-12-0150.

To search for U.FL to RP-SMA cable options at Digi-Key, go to the "Cable Assemblies => Coaxial Cables (RF)" section, and filter by Style = "RP-SMA to IPX" or "RP-SMA to MHF1" or "RP-SMA to UMC" or "RP-SMA to UMCC". Then look at the picture and make sure it looks correct, as the application of the terms "male" and "female" are not totally standardized.

T7-Pro-OEM ships with a simple 30mm U.FL whip antenna such as the Anaren 66089-2406.

WiFi Range

Base T7 Antenna

The WiFi range on the T7 is typical for a modern WiFi device. In direct line-of-sight with the router, it's possible to get a decent connection at 100m. The table below shows signal strength at varying distances with a stock T7 antenna, and a simple WiFi router. Both the T7 and the router were positioned 3ft off of the ground, with direct line-of-sight.

Distance	RSSI
10m	-44dBm
25m	-45dBm
50m	-55dBm
100m	-59dBm

During testing, it was noted that the T7 had slightly better WiFi range than an HTC One V cell phone. The WiFi signal is spotty at RSSI lower than -75dBm, and the connection will cut off entirely around -80dBm. Note that 90° antenna orientation was used in testing above. That is to say, keep the antenna in the fully bent upright position, don't try to point it at the router, or accidentally leave it at 45° bent. At 45° bent, or directly pointed towards the router, the signal strength is reduced by about 5dBm.

Note that the RSSI value you can read (WIFI_RSSI) is only updated when WiFi communication occurs. That is, if you talk to the T7 over USB to read the RSSI value, you will just get the value from the last WiFi communication that occurred.

OEM Whip Antenna

The OEM whip antenna is a short segment of wire, only 30mm in length. This whip antenna provides an inexpensive solution for adding WiFi to an OEM board, without the need to figure out mounting of a bigger antenna. The signal strength of a 30mm whip antenna is on average 11dB less than that of the stock antenna. The table below demonstrates the 30mm antenna signal strength at various distances.

Distance	RSSI
10m	-49dBm
25m	-58dBm
50m	-70dBm
100m	-73dBm

Improve signal strength

The first step to improve the signal strength at large distances is to ensure direct line-of-sight. Beyond that, the next best thing is to elevate the transmitter and receiver antennas. Even an elevation of 1m off the ground helps quite a bit. Be sure to consider the probability of lightning strikes if the antenna is high relative to the surroundings.

The next step to improve signal strength is to use a directional WiFi antenna. Directional antennas improve range substantially, such that even a homemade solution can increase range to fifteen times that of a non-directional antenna. If you need something to work at 500m, it's possible to buy a simple yagi antenna for $60 USD approx.

Network Schemes

Figure 1 demonstrates a basic network diagram where the T7-Pro is connected to the "Office WiFi" network. The T7-Pro can be controlled by either host option (1 or 2). For much more information on Ethernet and WiFi networking see the Basic Networking & Troubleshooting App Note.

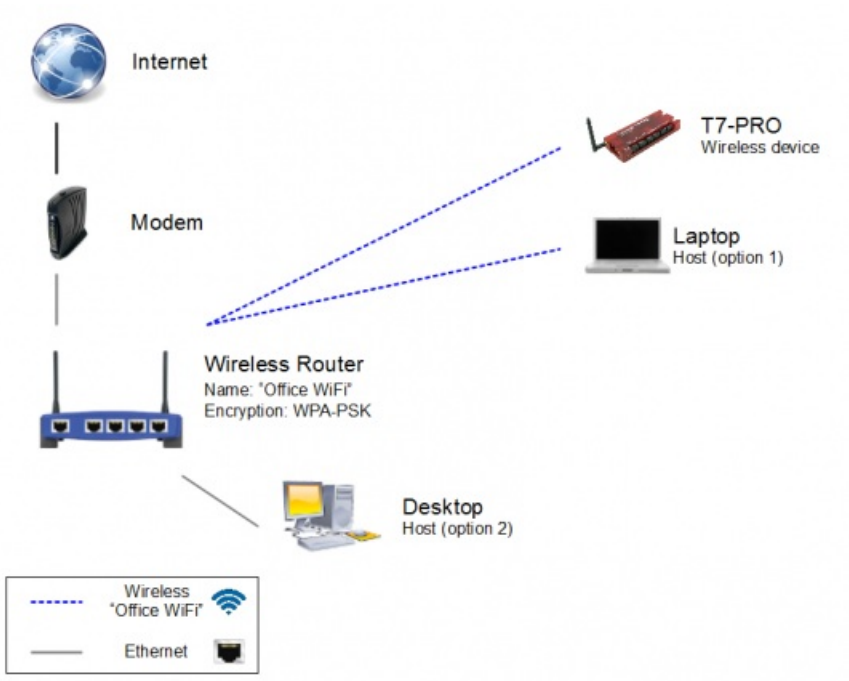


Figure 1. Most common configuration for a home, or small office network.

UDP Discovery-Only Mode

Firmware minimum: 1.0284

If your network has Modbus UDP packets broadcast on port 502, you can enable discovery-only mode. See the Ethernet page for more details.

Use WIFI_UDP_DISCOVERY_ONLY to enable/disable discovery-only mode until the device is power-cycled:

Name	Start Address	Type	Access
WIFI_UDP_DISCOVERY_ONLY	49215	UINT16	R

WIFI_UDP_DISCOVERY_ONLY
- Address: 49215

Restricts which Modbus operations are allowed over UDP. When set to 1, only the registers needed for discovering units can be read and any other operation will throw an error.

- Data type: UINT16 (type index = 0)
- Read-only
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0284

Use the _DEFAULT register version to set the startup behavior for discovery-only mode:

Name	Start Address	Type	Access
WIFI_UDP_DISCOVERY_ONLY_DEFAULT	49265	UINT16	R

WIFI_UDP_DISCOVERY_ONLY_DEFAULT
- Address: 49265

The Enabled/Disabled state of WIFI_UDP_DISCOVERY_ONLY after a power-cycle to the device.

- Data type: UINT16 (type index = 0)
- Read-only
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0284

8.0 LEDs [T-Series Datasheet]

STATUS - The Green LED

The STATUS LED indicates when the T-series device is performing autonomous tasks, such as running an on-board Lua script or streaming. The STATUS LED will blink according to the following rules:

- Continuously while stream is running.
- Once when a Lua script accesses a Modbus register.
- Twice every 2 seconds when the T8 is in recovery mode.
- In conjunction with the COMM LED. See the Combined LED Activity section.

COMM - The Yellow LED

The COMM LED's primary purpose is to indicate packet transfer. The COMM LED will blink according to the following rules:

- Once for each packet transferred. At high packet transfer rates the LED will blink at 10Hz, even if more than 10 packets are being processed per second.
- Continuously when stream mode is running.
- A few times after successfully connecting to a USB host.
- In conjunction with the STATUS LED. See the Combined LED Activity section.

Normal Behavior

Here are 3 examples of normal LED behavior. For troubleshooting it is useful to compare

these examples to what you see on your device.

Power-Up

1. Both LEDs blink rapidly for about 1 second.
2. COMM solid off and STATUS solid on.
3. If USB enumerates, COMM blinks a few times and then stays solid on.

Idle

Both LEDs solid on. No blinking

While Viewing Kipling's Dashboard

Green LED solid on. Orange LED also solid on but with a quick double-blink once per second.

Combined LED Activity

- When the LEDs blink together, the T-series device is computing checksums.
- When the LEDs are alternating, the T-series device is copying a firmware image.

Troubleshooting

If LED behavior is not normal, please see USB Communication Failure.

LED Configuration

The LED mode can be set using the following registers:

Name	Start Address	Type	Access
POWER_LED	48006	UINT16	R/W

POWER_LED
- Address: 48006

Sets the LED operation:
0 = Off. Useful for lower power applications.
1 = normal.
2 = Lower power, LEDs will still blink but will normally be off.
3 = Reserved.
4 = Manual, in this mode the LEDs can be user controlled.

- Data type: UINT16 (type index = 0)
- Readable and writable
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 0.8600
- T4:
  - Minimum firmware version: 1.0008

POWER_LED_DEFAULT	48056	UINT16	R/W

POWER_LED_DEFAULT
- Address: 48056

The ON/OFF state of the LEDs after a power-cycle to the device.

- Data type: UINT16 (type index = 0)
- Readable and writable
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 0.8600
- T4:
  - Minimum firmware version: 1.0008

When POWER_LED is set to manual mode, the LEDs can be turned on and off using the following registers:

Name	Start Address	Type	Access
LED_COMM	2990	UINT16	W

LED_COMM
- Address: 2990

Sets the state of the COMM LED when the LEDs are set to manual, see the POWER_LED register.

- Data type: UINT16 (type index = 0)
- Write-only
- Default value: 0
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0000
- T4:
  - Minimum firmware version: 1.0008

Constant	Value
Off	0
On	1

LED_STATUS	2991	UINT16	W

LED_STATUS
- Address: 2991

Sets the state of the STATUS LED when the LEDs are set to manual, see the POWER_LED register.

- Data type: UINT16 (type index = 0)
- Write-only
- Default value: 0
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0000
- T4:
  - Minimum firmware version: 1.0008

Constant	Value
Off	0
On	1

9.0 VS, Power Supply [T-Series Datasheet]

VS and Power Supply Specifications By Device

T4

Supply Voltage: **4.75 - 5.25 volts (5V ±5% Regulated)**

Device Supply Current: **210 mA Max**

Normal Power Connector: **USB-B Receptacle**

Typical Power Supply: **Any USB-Style Supply**

VS Voltage: **Equal to Supply Voltage**

VS Max Current: **290 mA (500 mA - Device Supply Current)**

T7

Supply Voltage: **4.75 - 5.25 volts (5V ±5% Regulated)**

Device Supply Current: **300 mA Max**

Normal Power Connector: **USB-B Receptacle**

Typical Power Supply: **Any USB-Style Supply**

VS Voltage: **Equal to Supply Voltage**

VS Max Current: **200 mA (500 mA - Device Supply Current)**

T8

Supply Voltage: **4.5 - 5.25 volts (5V ±5% Regulated)**

Device Supply Current: **670 mA Typical**

Normal Power Connector: **USB-B Receptacle**

Typical Power Supply: **High Power (2A+) USB-Style Supply, or PoE**

VS Voltage: **Equal to Supply Voltage**

VS Max Current: **200 mA**

> The T8 requires more power than previous LabJack devices. As such, it is more important to use a high quality power supply and USB cable. Inadequate device power can cause issues with T8 operation. When powering the T8 over USB, we recommend using a wall-powered USB hub.

VS Terminals

The supply voltage (see below) goes through some protection circuitry and then is presented on the VS terminals. The VS terminals are designed as outputs for the supply voltage. The supply voltage is nominally 5 volts and typically provided through the USB connector.

All VS terminals are identical and connect to the same supply.

The VS terminals are outputs, not inputs. Do not connect a power source to VS in normal situations.

The max total current that can be drawn from VS is:

> MaxVScurrent = ratedSupplyCurrent - DeviceSupplyCurrent

Example: If you have a typical USB 2.0 supply that is within spec (up to 500mA supply current), the T4/T7 requires up to 300mA to run, so up to 200mA would be available from the VS terminals of the T4/T7.

The voltage on VS can be noisy and can change unexpectedly. Circuits that are sensitive to changing or noisy supply voltage, such as bridge circuits, should not be supplied from VS. The voltage on VS will also drop as higher amounts of current are drawn by peripherals powered by the T4/T7. This is due to the inline 0.1 ohm resistors that allows for the device's current draw to be measured.

Measuring Current Draw

T4/T7

One way to measure how much current the T4/T7 is drawing is by measuring the voltage across R15. R15 is a 0.1 ohm resistor, so if you measure 0.025 volts, that means the current through the resistor is 250 mA. R15 is a large resistor located on the top of the PCB just behind the green LED. To measure the voltage across R15, connect the positive lead of your meter to the test point "Vhost" and connect the negative lead of your meter to the test point "Vs".

R15 is in series with the 5 volt supply from the USB connector. If powering from J5 (see "Alternate Power Supply" in the OEM section) use R21 instead. The "Vs" test point is also the

negative for R21, but there is no positive test point so you just have to touch the upstream side of R21.

Note that the "Vs" test point is actually the Vsupply bus described under "J5 - Alternate Power Supply" in the OEM section, and technically not exactly the same as the "VS" bus documented in this section.

T8

The T8 can measure it's own supply voltage and current draw. Use the register below to get the internal measurements.

T8 Power Consumption

Name	Start Address	Type	Access
INTERNAL_VS_VOLTS	60030	FLOAT32	R

INTERNAL_VS_VOLTS
- Address: 60030

T8 Only. Returns the power supply voltage.

- Data type: FLOAT32 (type index = 3)
- Read-only

INTERNAL_IS_AMPS	60032	FLOAT32	R

INTERNAL_IS_AMPS
- Address: 60032

T8 Only. Returns the power supply current draw.

- Data type: FLOAT32 (type index = 3)
- Read-only

To measure current manually, measure the voltage across R22 and divide that value by 0.02.

Power Supply

T4/T7

Power supply is typically provided through the USB connector.

- USB host or hub.
- Wall-wart power supply with USB connection (included with normal retail units—not OEM).
- Power-over-Ethernet splitter (e.g. TP-Link TL-POE10R with Tensility 10-00240 with

Tensility 10-00648).
- Car charger (12V supply) with USB ports (e.g. Anker 71AN2452C-WA).
- Rechargeable battery with USB ports (e.g. Anker Astro E5 79AN15K-BA perhaps with Belkin F3U133-06INCH).
- Battery with car charger (e.g. Anker 79AN15K-BA with 71AN2452C-02WA).
- Battery with solar panel (e.g. Anker 79AN15K-BA with 71ANSCP-B145A).
- Pigtail a cable with a USB-B connector to get at the red and black wires, and make some sort of custom cable for your 5V power supply.

The supply range for specified operation is 4.75 to 5.25 volts, which is the same as the USB specification for voltage provided to a device. Nonetheless, we have seen some USB host ports providing a lower voltage. If your USB host port has this problem, add a USB hub with a strong power supply.

Typical power draw for a T7-Pro with everything on is 280 mA. Running at 5 volts that is 1.4 watts. See related data in the General section of Specifications.

USB batteries are typically specified with a mAh rating at the internal battery voltage. A large USB battery such as the Anker PowerCore+ has a battery that provides 26800 mAh at 3.7 volts, which is 99.2 watt-hours. If we guess at 80% efficiency for extracting this energy and converting it to 5 volts, this battery should be able to power aT7-Pro for roughly 0.8* (99.2/1.4) = 57 hours.

See information about Power over Ethernet (PoE) see the PoE App Note.

T8

The T8 has two power supply options, USB and Power over Ethernet (PoE). Both power supplies are fed into a power switch. The switch selects which supply will be used to power the T8. Selection is based on the voltage of the PoE input. When V_{PoE} is below the threshold voltage, the USB supply will be used. When V_{PoE} exceeds the threshold voltage, the PoE supply will be used. See Section A-5 for specific values.

OEM Versions

For board-level connection options see "Alternate Power Supply" in the OEM section. Typical power supply sources include:

Normal retail units (not OEM) include a 5V, 2A wall-wart style power supply:

Compatibility	Make	Mfr. Model No.
North America	VA-PSU-US1	JX-B0520B-1-B
Europe	VA-PSU-EU1	JX-B0520A-1-B
United Kingdom	VA-PSU-UK1	JX-B0520C-1-B
Australia	-	JX-B0520D-1-B
China	-	JX-B0520H-1-B

Note that the JX-B0520 supply is rated for 0 to 40 deg C operation.

Some VS Terminals Not Working?

See the Screw Terminals App Note for assistance.

10.0 SGND and GND [T-Series Datasheet]

SGND (T4 and T7)

SGND

SGND is located near the upper-left of the device. This terminal has a self-resetting thermal fuse in series with GND. This is often a good terminal to use when connecting the ground from another separately powered system that could unknowingly already share a common ground with the T4 or T7.

See the AIN, DAC, and Digital I/O application notes for more information about grounding.

GND

GND

The GND connections available at the screw-terminals and DB connectors provide a common ground for all LabJack functions. All GND terminals are the same and connect to the same ground plane.

GND is also connected to the ground pin on the USB connector, so if there is a connection to a USB port on a hub/host (as opposed to just a power supply connection), then GND is the same as the ground line on the USB connection, which is often the same as ground on the PC chassis, which is often the same as AC mains ground.

For more information about grounding, see the 14.0 AIN, 15.0 DAC, and 13.0 Digital I/O sections.

The max total current that can be sunk into GND is:

```
T4, T7: Max total current = 500mA - DeviceSupplyCurrent
```

```
T8: Max total current = 1100mA - DeviceSupplyCurrent
```

For example, if the T7 needs 250mA to run, the current sunk into GND terminals should be limited to 250mA. Note that sinking substantial current into GND can cause slight voltage differences between different ground terminals, which can cause noticeable errors with single-

ended analog input readings. For information about device supply current, see 9.0 VS, Power Supply.

11.0 SPC [T-Series Datasheet]

The SPC terminal has several uses:

- Outputs diagnostic timing signals while streaming (see Stream Section for details)
- Can be used to force special startup behavior.

Startup Behavior

There are five special startup behaviors:

- **Force Boot to Main Firmware** - Force boot to main firmware (internal) image. Used to boot the internal firmware even if its checksum is bad. Additionally, Lua scripts will not be loaded during boot up.
- **Force Overwrite Main Firmware** - Force copy of backup image to overwrite internal image. Used to load the external firmware even if its checksum is bad.
- **Factory Reset** - Sets the start up configuration to factory settings. Disables Lua script at startup.
- **Boot to Emergency Firmware** - Load emergency image. This option loads a firmware image with minimal functionality (similar to Windows safe-mode). Used to recover from firmware corruption or bugs. The update process is about all that can be done while in this mode.
- **Configure Static Ethernet** - Temporarily sets the Ethernet configuration

To force special startup behavior, securely install a short jumper wire from SPC to one of the corresponding I/O lines described below. The jumper needs to be installed before reset, so make sure the jumper is securely clamped in SPC and the given FIO or AIN terminal, then power up the device.

The jumper must be **securely installed**. Don't try to just hold a wire in a loose screw terminal or touch a wire to the screw head. _DEFAULT Ethernet settings are not changed. The LEDs blink in an alternating pattern 5 times to confirm that the jumper was detected.

This static Ethernet configuration is compatible with a direct Ethernet connection.

T4

FIO4
: Force Boot to Main Firmware

FIO5

Force Overwrite Main Firmware

FIO6

Factory Reset

FIO7

Boot to Emergency Firmware

AIN3

Configure Static Ethernet (Added in firmware 1.0027). Temporarily sets the Ethernet configuration as follows:
DHCP: Off
IP Address: 192.168.1.204
Subnet: 255.255.255.0
Gateway: 192.168.1.1
DNS: 8.8.8.8
Alternate DNS: 8.8.4.4

T7

FIO0

Force Boot to Main Firmware

FIO1

Force Overwrite Main Firmware

FIO2

Factory Reset

FIO3

Boot to Emergency Firmware

AIN3

Configure Static Ethernet (Added in firmware 1.0291). Temporarily sets the Ethernet configuration as follows:
DHCP: Off
IP Address: 192.168.1.207
Subnet: 255.255.255.0
Gateway: 192.168.1.1
DNS: 8.8.8.8
Alternate DNS: 8.8.4.4

T8

FIO0

Force Boot to Main Firmware

FIO1

Force Overwrite Main Firmware

FIO2

Factory Reset

FIO3

Boot to Emergency Firmware

Device Recovery Process

If the device has become unresponsive, we recommend trying the following sequence of our device SPC jumpers, stopping if the device becomes responsive:

1. Factory reset
2. Force copy of backup image to overwrite internal image.
3. Load emergency image. You should update your device firmware using our Kipling software after loading the emergency image.

Firmware Images

T-Series devices have two different firmware images:

- The "primary firmware image", which is synonymous to "main firmware", is the firmware image used when the device is working properly. The primary firmware image (when being used) exists in the microcontroller's internal flash (for execution) as well as on the external flash chip as a backup image.
- The second firmware is known as the "Emergency Image." The Emergency Image implements a minimal feature set, which is enough to update the main firmware.

12.0 200uA and 10uA (T7 Only) [T-Series Datasheet]

Overview - T7 Only

The T7 has 2 fixed current source terminals useful for measuring resistance (thermistors, RTDs, resistors). The 10UA terminal provides approximately 10 µA and the 200UA terminal provides approximately 200 µA, but the actual values should be read from the calibration constants, or better yet measured in real-time using a fixed shunt resistor.

Using the equation V=IR, with a known current and voltage, it is possible to calculate the resistance of the item in question. Figure 12-1 shows a simple setup measuring 1 resistor.

The factory value of each current source is noted during calibration and stored with the calibration constants on the device. These can be viewed using the Device Info tab in Kipling, or read programmatically. Note that these are fixed constants stored during calibration, not some sort of real-time readings.

To read the constants, read from the following registers:

Constant Current Sources

Name	Start Address	Type	Access
CURRENT_SOURCE_200UA_CAL_VALUE	1902	FLOAT32	R

CURRENT_SOURCE_200UA_CAL_VALUE
- Address: 1902

Fixed current source value in Amps for the 200UA terminal. This value is stored during factory calibration, it is not a current reading. Using the equation V=IR, with a known current and voltage, it is possible to calculate resistance of RTDs.

- Data type: FLOAT32 (type index = 3)
- Read-only

CURRENT_SOURCE_10UA_CAL_VALUE	1900	FLOAT32	R

CURRENT_SOURCE_10UA_CAL_VALUE
- Address: 1900

Fixed current source value in Amps for the 10UA terminal. This value is stored during factory calibration, it is not a current reading. Using the equation V=IR, with a known current and voltage, it is possible to calculate resistance of RTDs.

- Data type: FLOAT32 (type index = 3)
- Read-only

Example:

To read the factory value of the 200uA current source, perform a read of Modbus address 1902, and the result would be in the form of a floating point number, e.g. 0.000197456 amps.

Examples Of Measuring Resistance

Multiple resistances can be measured by putting them in series and measuring the voltage across each. Some applications might need to use differential inputs to measure the voltage across each resistor, but for many applications it works just as well to measure the single-ended voltage at the top of each resistor and subtract in software.

Figure 12-1

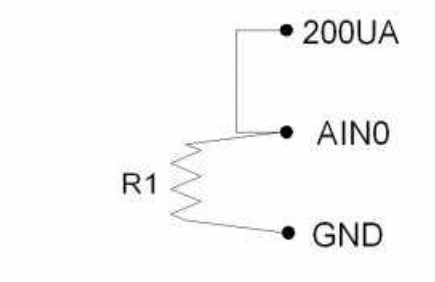


Figure 12-2

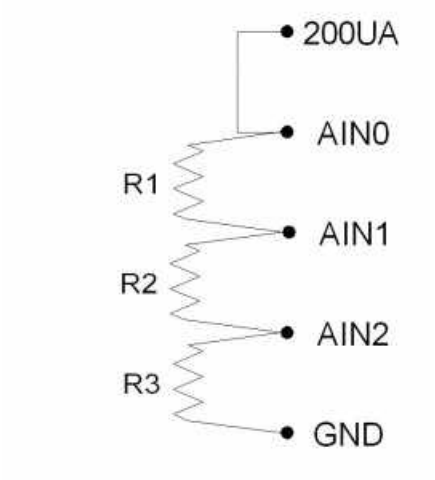


Figure 12-1 shows a simple setup measuring 1 resistor. If R1=3k, the voltage at AIN0 will be 0.6 volts.

Figure 12-2 shows a setup to measure 3 resistors using single-ended analog inputs. If R1=R2=R3=3k, the voltages at AIN0/AIN1/AIN2 will be 1.8/1.2/0.6 volts. That means AIN0 and AIN1 would be measured with the ±10 volt range, while AIN2 could be measured with the ±1 volt range. This points out a potential advantage to differential measurements, as the differential voltage across R1 and R2 could be measured with the ±1 volt range, providing better resolution.

Figure 12-3

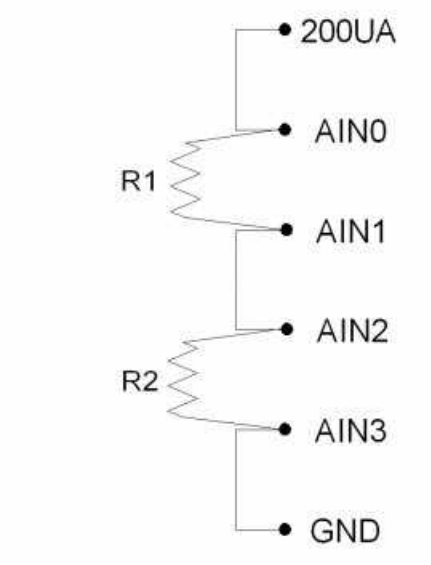


Figure 12-4

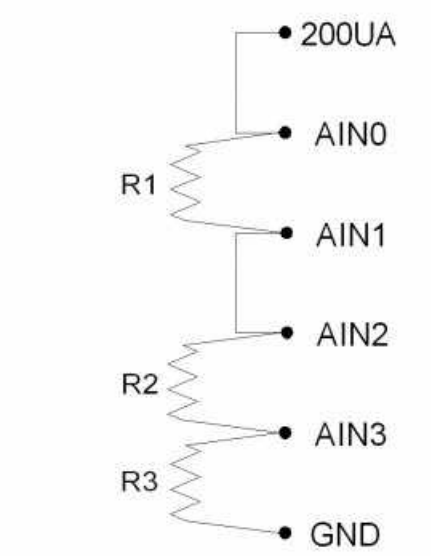


Figure 12-3 shows a setup to measure 2 resistors using differential analog inputs. AIN3 is wasted in this case, as it is connected to ground, so a differential measurement of AIN2-AIN3 is the same as a single-ended measurement of AIN2. That leads to **Figure 12-4**, which shows R1 and R2 measured differentially and R3 measured single-ended.

Remarks

Maximum load resistance: The current sources can drive about 3 volts max, thus limiting the maximum load resistance to about 300 kΩ (10UA) and 15 kΩ (200UA). Keep in mind that high source resistance could cause settling issues for analog inputs.

Using a fixed resistor to calculate actual current: For some applications the accuracy

and temperature coefficient of the current sources is sufficient, but for improvement a fixed resistor can be used as one of the resistors in the figures above. The Y1453-100 and Y1453-1.0K from Digi-Key have excellent accuracy and very low tempco. By measuring the voltage across one of these you can calculate the actual current at any time.

Handling load changes resulting in noise: The current sources are not particularly fast in reacting to load changes. This can show up as noise when rapidly sampling multiple channels using the same current source. Improve behavior by adding a 1 µF ceramic capacitor from the current source to GND and/or increasing settling time.

Temperature coefficients: Figures 12-5 and 12-6 show the typical current source output variation over temperature. Both sources typically have low temperature coefficients at or near 25C. Beyond 25C, the temperature coefficient variation may need to be accounted for, depending on application requirements.

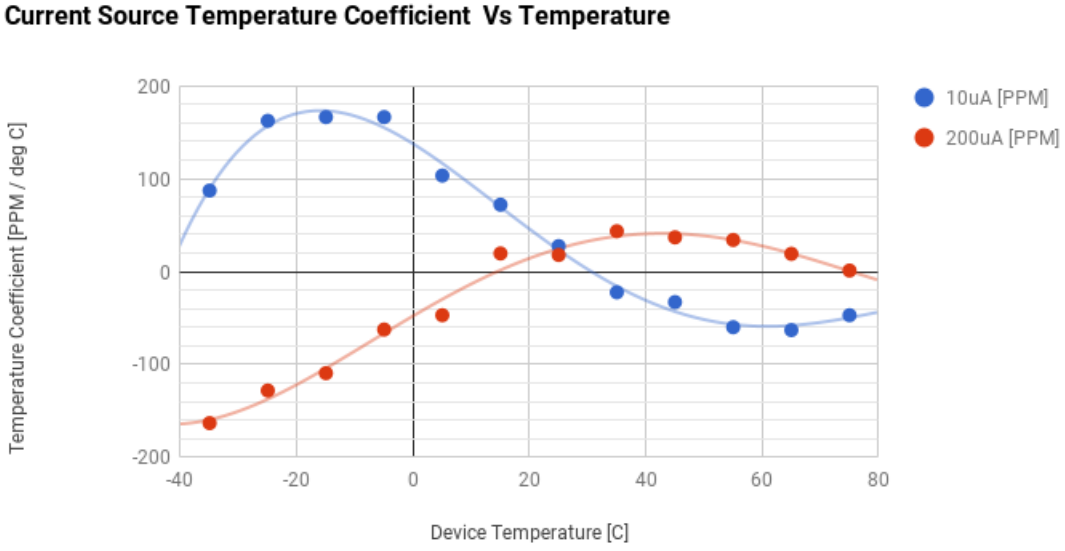


Figure 12-5. Typical temperature coefficient values over operating temperature range

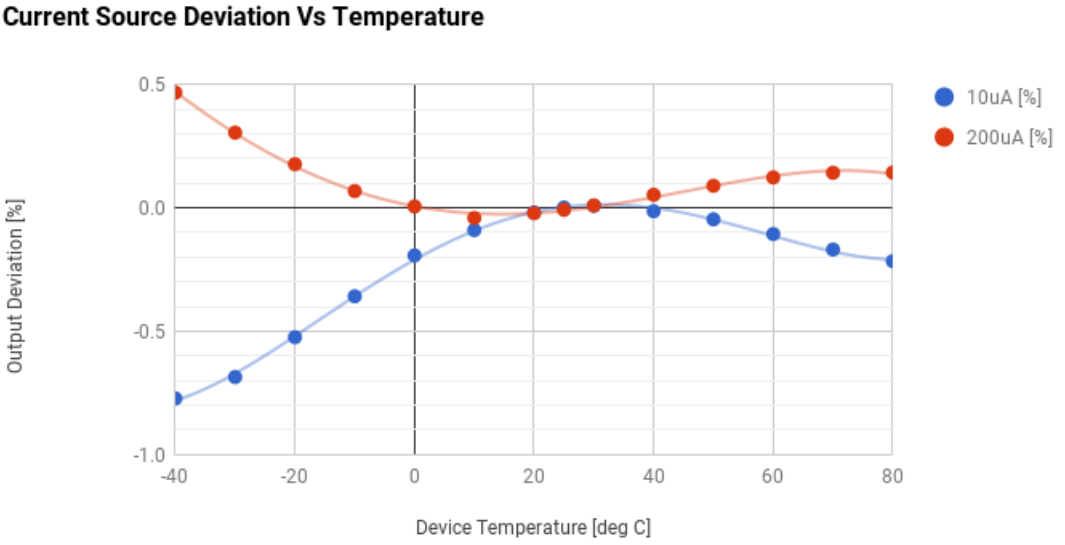


Figure 12-6. Typical current source deviation from 25C output over operating temperature range

Example - PT100 or PT1000 RTD

Assume that R1 in Figure 12-1 is a PT100 RTD. A PT100 RTD is 100 ohms at 0 degC. The response of an RTD is nonlinear, but the linear slope 0.384 ohms/degC works well from about -40 to +150 degC. That leads to the following expression:

```
R = (0.384 * DegC) + 100
```

...which can be rearranged to:

```
DegC = (2.604 * R) - 260.4
```

We are determining R by measuring the voltage that results from a known current passed through R, that is R = V/I, so we can say:

```
DegC = (2.604 * V/I) - 260.4
```

This tells us that the slope is 2.604/I and the offset is -260.4. To determine I, you can just use 0.0002 amps, or use the factory calibration value read from CURRENT_SOURCE_200UA_CAL_VALUE, or use a precision fixed resistor as mentioned above to measure I in real time. Assume we read the factory calibration value as 0.000200 amps, and thus use a constant slope of 2.604/0.0002 = 13020. We can now use the AIN-EF Offset and Slope feature to apply this slope and offset:

```
AIN0_EF_INDEX = 1              // feature index for Offset and
Slope
AIN0_EF_CONFIG_D = 13020.0     // slope
AIN0_EF_CONFIG_E = -260.4      // offset
```

Now reads of AIN0_EF_READ_A will return `(13020.0 * volts) - 260.4` .

Note that you can come up with your own slope and offset for your temperature region of interest. For example, we made this Google Spreadsheet and decided that Slope=2.720 (degC/ohm) and Offset=-277.5 works best for the region of 100 to 300 degC.

Note that a PT1000 simply has 10x the response of a PT100 (~3.84 ohms/degC). The offset still works out to -260.4, but the slope is 0.260.

13.0 Digital I/O [T-Series Datasheet]

Overview

Basics: Individual digital I/O lines can be set to digital input or digital output. DIO is a generic name used for all digital I/O.

Common Uses: For wiring information on open-collector signals, driven signals, controlling relays, and mechanical switches, see the Digital I/O (App Note).

Device Control Basics:

- All T-series device features are controlled by reading and writing Modbus TCP registers via Modbus TCP (either directly or through our LJM library).
- We have register descriptions throughout documentation detailing relevant register names, starting addresses, types, and access permissions (read/write).
- Most DIO registers have a UINT16 type meaning each register requires 16-bits to represent the associated data. Each entry in the device address space holds 16-bits of data, so most DIO registers take up one address space such as 2004 (FIO4 register). See the Modbus Map for related information.
- See Section 3.0 Communication for other detailed communication information.

DIO Extended Features: T-series DIO Extended Features expose more complicated features such as:

- Timers, Counters, PWM, Quadrature Input, and more.

Digital Communication Protocols: T-series DIO lines can also be used to communicate with a large number of sensors that require the use of various digital communication protocols. The T-series devices implement the following protocols:

- I2C - Also reference the I2C Lua Library and Lua I2C Sensor examples.
- SPI
- SBUS
- 1-Wire
- Asynchronous Serial (UART)

Subsections

DIO Summary By Device

T4

Digital I/O: Up to 16 DIO lines (DIO4-DIO19)

- 8 flexible I/O (DIO4-DIO11)
- 8 dedicated I/O (DIO12-DIO19)

Logic Level: 3.3V (Adjustable using a LJTick-LVDigitalIO).

T7

Digital I/O: Up to 23 DIO lines (DIO0-DIO22)

Logic Level: 3.3V (Adjustable using a LJTick-LVDigitalIO).

T8

Digital I/O: Up to 20 DIO lines (DIO0-DIO19)

Logic Level: 3.3V (Adjustable using a LJTick-LVDigitalIO).

Usage

There are two ways to access the simple functionality of the DIO:

1. Read or write individual DIO channels one-at-a-time. Individual DIO channels are automatically configured.
2. Read or write multiple DIO channels at once with the DIO Bitmask Registers, which are manually configured.

Individual DIO Channels

Each T-Series device exposes:

- Some DIOs on the screw terminals.
- Additional DIOs on a connector (either a DB15 or a DB37).

T4

The LabJack T4 has up to 16 built-in digital input/output lines. They can be written/read as registers named DIO4-DIO19.

DIO4-DIO11 are flexible I/O lines. (See 13.1 Flexible I/O.) These are lines that can be configured for analog input or digital input/output:

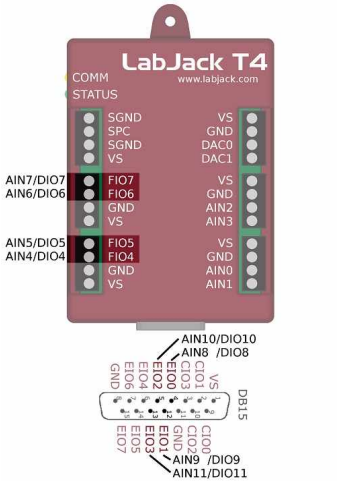


Figure 13.0-1 T4 Flexible I/O

DIO12-DIO19 are dedicated (digital-only) I/O lines:

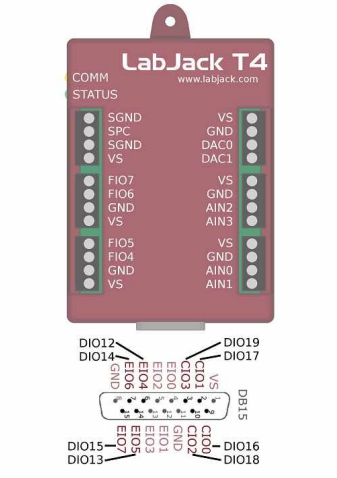


Figure 13.0-2 T4 Dedicated Digital I/O

The registers DIO4-DIO19 can also be accessed using their alternate register names, FIO4-7, EIO0-7, and CIO0-3:

Table 13.0-1. T4 DIO Mapping

	AIN (0-3)				FIO (4-7)				EIO (0-7)								CIO (0-3)			
	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3
					DIO															
DIO Number	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Available Digital I/O					✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
Available Analog I/O	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔								

T4 DIO Channel Registers

Name	Start Address	Type	Access
DIO#(4:19)	2004	UINT16	R/W

DIO#(4:19)
- Starting Address: 2004

Read or set the state of 1 bit of digital I/O. Also configures the direction to input or output. Read 0=Low AND 1=High. Write 0=Low AND 1=High.

- Data type: UINT16 (type index = 0)
- Readable and writable
- This register may be streamed

Expanded Names	Addresses
DIO4, DIO5, DIO6, DIO7, DIO8, DIO9, DIO10, DIO11, DIO12, DIO13, DIO14, DIO15, DIO16, DIO17, DIO18, DIO19	2004, 2005, 2006, 2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019

T7

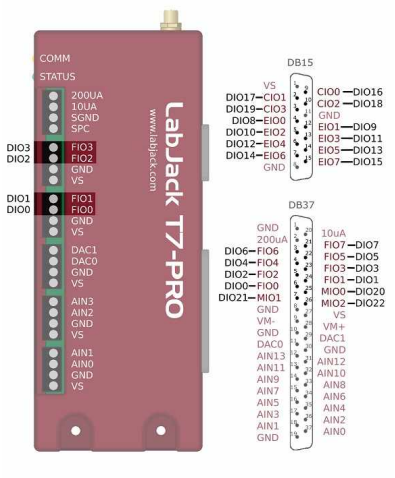


Figure 13.0-3 T7 Digital I/O

The LabJack T7 has 23 built-in digital input/output lines. They can be written/read as registers named DIO0-DIO22. They can also accessed using their alternate register names: FIO0-7, EIO0-7, CIO0-3, and MIO0-2:

Table 13.0-2. T7 DIO Mapping

	FIO (0-7)								EIO (0-7)								CIO (0-3)				MIO (0-2)		
	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	0	1	2
	DIO																						
DIO Number	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
Available Digital I/O	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔

T7 DIO Channel Registers

Name	Start Address	Type	Access
DIO#(0:22)	2000	UINT16	R/W

DIO#(0:22)
- Starting Address: 2000

Read or set the state of 1 bit of digital I/O. Also configures the direction to input or output. Read 0=Low AND 1=High. Write 0=Low AND 1=High.

- Data type: UINT16 (type index = 0)
- Readable and writable
- This register may be streamed

Expanded Names	Addresses
DIO0, DIO1, DIO2, DIO3, DIO4, DIO5, DIO6, DIO7, DIO8, DIO9, DIO10, DIO11, DIO12, DIO13, DIO14, DIO15, DIO16, DIO17, DIO18, DIO19, DIO20, DIO21, DIO22	2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020, 2021, 2022

T8

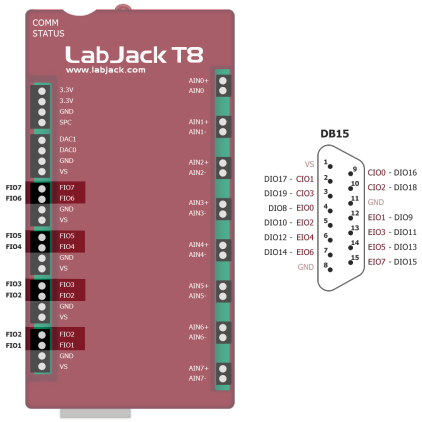


Figure 13.0-4 T8 Digital I/O

The LabJack T8 has 20 built-in digital input/output lines. They can be written/read as registers named DIO0-DIO19. They can also accessed using their alternate register names: FIO0-7, EIO0-7, and CIO0-3.

Table 13.0-3. T8 DIO Mapping

	FIO (0-7)								EIO (0-7)								CIO (0-3)			
	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3
	DIO																			
DIO Number	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Available Digital I/O	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔

T8 DIO Channel Registers

Name	Start Address	Type	Access
DIO#(0:19)	2000	UINT16	R/W

DIO#(0:19)
- Starting Address: 2000

Read or set the state of 1 bit of digital I/O. Also configures the direction to input or output. Read 0=Low AND 1=High. Write 0=Low AND 1=High.

- Data type: UINT16 (type index = 0)
- Readable and writable
- This register may be streamed

Expanded Names	Addresses
DIO0, DIO1, DIO2, DIO3, DIO4, DIO5, DIO6, DIO7, DIO8, DIO9, DIO10, DIO11, DIO12, DIO13, DIO14, DIO15, DIO16, DIO17, DIO18, DIO19	2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019

DIO_PULLDOWN_ENABLE	2870	UINT32	R/W

DIO_PULLDOWN_ENABLE
- Address: 2870

T8 Only. This register will enable pulldowns on DIO lines. This is a binary coded value where bit 0 represent FIO0, and bit 11 represents EIO3, etc. 1 = pulldown enabled, 0 = pulldown disabled. This register only affects flex-lines which can be configured as analog or digital. This register is not affected by the inhibit register.

- Data type: UINT32 (type index = 1)
- Readable and writable

DIO_PULLUP_DISABLE	2890	UINT32	R/W

DIO_PULLUP_DISABLE
- Address: 2890

This register will prevent pullups from being enabled on lines set to digital input. This is a binary coded value where bit 0 represent FIO0 and bit 11 represents EIO3. 1 = pullup disabled, 0 = pullup enabled. This register is not affected by the inhibit register.

- Data type: UINT32 (type index = 1)
- Readable and writable
- T8:
  - Disables the pull-up resistors on DIOs.
- T4:
  - This register only affects flex-lines which can be configured as analog or digital.

Examples

> The individual DIO registers set both the state and direction; reads set the DIO to input and writes to output. To read a DIO without affecting its direction, see the DIO Bitmask section.

- To set FIO4 to an output high (~3.3V) using LJM, write a 1 to address 2004 using a function such as eWriteAddress.
- To set FIO6 to input and read the state using LJM, read address 2006 using a function such as eReadAddress.

DIO Bitmask Registers

The digital I/O bitmask registers allow for the direction (input or output) and state (high or low) of multiple digital I/O lines to be set during a single register write. Each bit in the value written to these registers corresponds to an individual I/O line on the device. The number of valid bits in the bitmask depends on which device is being used. To see which bits are valid for each device, see the above reference tables T4 DIO Mapping and T7 DIO Mapping.

> The lower four bits of these DIO bitmask registers don't apply to the T4.

DIO Bitmask Registers

Name	Start Address	Type	Access

DIO_INHIBIT	2900	UINT32	R/W

DIO_INHIBIT
- Address: 2900

A single binary-encoded value where each bit determines whether _STATE, _DIRECTION or _ANALOG_ENABLE writes affect that bit of digital I/O.
0=Default=Affected,
1=Ignored.

- Data type: UINT32 (type index = 1)
- Readable and writable
- Default value: 0

Constant	Value
Affected	0
Ignored	1

DIO_DIRECTION	2850	UINT32	R/W

DIO_DIRECTION
- Address: 2850

Read or write the direction of all digital I/O in a single binary-encoded value. 0=Input and 1=Output. Writes are filtered by the value in DIO_INHIBIT.

- Data type: UINT32 (type index = 1)
- Readable and writable

Constant	Value
Input	0
Output	1

DIO_STATE	2800	UINT32	R/W

DIO_STATE
- Address: 2800

Read or write the state of all digital I/O in a single binary-encoded value. 0=Low AND 1=High. Does not configure direction. A read of an output returns the current logic level on the terminal, not necessarily the output state written. Writes are filtered by the value in DIO_INHIBIT.

- Data type: UINT32 (type index = 1)
- Readable and writable

Constant	Value
Low	0
High	1

Writing DIO Bitmask Registers

DIO_INHIBIT, DIO_DIRECTION, and DIO_STATE should typically be written together. Each true bit in DIO_INHIBIT prevents a corresponding bit in DIO_DIRECTION and DIO_STATE from being modified. For more details about the DIO_INHIBIT register, see the examples section below, as well as the manual configuration section of the flexible I/O page.

The LJM multiple value functions provide an easy way to write DIO_INHIBIT, DIO_DIRECTION, and DIO_STATE in a single packet.

Write Example

To configure DIO4 and DIO5 as a digital outputs set to high, do the following:

1. Build a bitmask based on the DIO channel numbers being controlled. For this example we are controlling DIO channels 4 and 5. All of the following values are equivalent: (1<<4) | (1<<5) , (2^4 + 2^5) , and 16 + 32 are all equal to 0b00110000, 0x30, and 48.
2. Subtract the bitmask value 0x30 from a value with 23 bits of "1"s (0x7FFFFF)to calculate the value that needs to be written to the DIO_INHIBIT register: 0x7FFFFF - (1<<4)|(1<<5) which equals 0b11111111111111111001111, 0x7FFFCF, or 8388559
3. Write the bitmask value 0x7FFFCF to the DIO_INHIBIT register to inhibit all DIO channels except 4 and 5.
4. Write the bitmask value 0x30 to the DIO_DIRECTION register to set the I/O lines as output.
5. Write the bitmask value 0x30 to the DIO_STATE register to have the two I/O lines output 3.3V.

Reading DIO Bitmask Registers

The typical workflow for reading the DIO Bitmask Registers is to only read DIO_STATE. This is because DIO_INHIBIT and DIO_DIRECTION are typically known.

DIO vs. FIO/EIO/CIO/MIO

DIO is a generic name used for all digital I/O. The DIO are subdivided into different groups called FIO, EIO, CIO, and MIO.

Sometimes these are referred to as different "ports". For example, FIO is an 8-bit port of digital I/O and EIO is a different 8-bit port of digital I/O. The different names (FIO vs. EIO vs. CIO vs. MIO) have little meaning, and generally you can call these all DIO0-DIO22 and consider them all the same. There are a couple details unique to different ports:

- The source impedance of an FIO line is about 550 ohms, whereas the source impedance of EIO/CIO/MIO lines is about 180 ohms. Source impedance might be important when sourcing or sinking substantial currents, such as when controlling relays.
- The MIO lines are automatically controlled when using analog input channel numbers from 16 to 127. This is for controlling external multiplexers or the Mux80 expansion board.

Alternate Digital Channel Names

The following shows the alternate DIO channel registers names:

Table 13.0-4. Alternate DIO Names

	DIO Name	Alternate Name	
FIO	DIO0	FIO0	T7/T8
	DIO1	FIO1	T7/T8
	DIO2	FIO2	T7/T8
	DIO3	FIO3	T7/T8
	DIO4	FIO4	
	DIO5	FIO5	
	DIO6	FIO6	
	DIO7	FIO7	
EIO	DIO8	EIO0	
	DIO9	EIO1	
	DIO10	EIO2	
	DIO11	EIO3	
	DIO12	EIO4	
	DIO13	EIO5	
	DIO14	EIO6	
	DIO15	EIO7	
CIO	DIO16	CIO0	
	DIO17	CIO1	
	DIO18	CIO2	
	DIO19	CIO3	
MIO	DIO20	MIO0	T7 only
	DIO21	MIO1	T7 only
	DIO22	MIO2	T7 only

Digital I/O

Name	Start Address	Type	Access
FIO#(0:7)	2000	UINT16	R/W

FIO#(0:7)
- Starting Address: 2000

Read or set the state of 1 bit of digital I/O. Also configures the direction to input or output. Read 0=Low AND 1=High. Write 0=Low AND 1=High.

- Data type: UINT16 (type index = 0)
- Readable and writable
- This register may be streamed

Expanded Names	Addresses
FIO0, FIO1, FIO2, FIO3, FIO4, FIO5, FIO6, FIO7	2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007

EIO#(0:7)	2008	UINT16	R/W

EIO#(0:7)
- Starting Address: 2008

Read or set the state of 1 bit of digital I/O. Also configures the direction to input or output. Read 0=Low AND 1=High. Write 0=Low AND 1=High.

- Data type: UINT16 (type index = 0)
- Readable and writable
- This register may be streamed

Expanded Names	Addresses
EIO0, EIO1, EIO2, EIO3, EIO4, EIO5, EIO6, EIO7	2008, 2009, 2010, 2011, 2012, 2013, 2014, 2015

CIO#(0:3)	2016	UINT16	R/W

CIO#(0:3)
- Starting Address: 2016

Read or set the state of 1 bit of digital I/O. Also configures the direction to input or output. Read 0=Low AND 1=High. Write 0=Low AND 1=High.

- Data type: UINT16 (type index = 0)
- Readable and writable
- This register may be streamed

Expanded Names	Addresses
CIO0, CIO1, CIO2, CIO3	2016, 2017, 2018, 2019

MIO#(0:2)	2020	UINT16	R/W

MIO#(0:2)
- Starting Address: 2020

Read or set the state of 1 bit of digital I/O. Also configures the direction to input or output. Read 0=Low AND 1=High. Write 0=Low AND 1=High.

- Data type: UINT16 (type index = 0)
- Readable and writable
- This register may be streamed

Expanded Names	Addresses
MIO0, MIO1, MIO2	2020, 2021, 2022

Example:

Writing 0 to FIO4 (address 2004) is the same as writing 0 to DIO4 (address 2004).

FIO/EIO/CIO/MIO Bitmask Registers

The following FIO/EIO/CIO/MIO bitmask registers are similar to the above DIO bitmask registers. However, instead of having a dedicated register designated for the inhibit bits, the inhibit bits are the upper 8-bits of each register.

Lower order bits of the FIO_STATE and FIO_DIRECTION have no affect on the T4. The MIO_STATE and MIO_DIRECTION registers have no affect on the T4 or T8.

FIO/EIO/CIO/MIO State Bitmask Registers

Name	Start Address	Type	Access
FIO_STATE	2500	UINT16	R/W

FIO_STATE
- Address: 2500

Read or write the state of the 8 bits of FIO in a single binary-encoded value. 0=Low AND 1=High. Does not configure direction. Reading lines set to output returns the current logic levels on the terminals, not necessarily the output states written. The upper 8-bits of this value are inhibits.

- Data type: UINT16 (type index = 0)
- Readable and writable
- This register may be streamed
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0000
- T4:
  - Minimum firmware version: 0.2000

EIO_STATE	2501	UINT16	R/W

EIO_STATE
- Address: 2501

Read or write the state of the 8 bits of EIO in a single binary-encoded value. 0=Low AND 1=High. Does not configure direction. Reading lines set to output returns the current logic levels on the terminals, not necessarily the output states written. The upper 8-bits of this value are inhibits.

- Data type: UINT16 (type index = 0)
- Readable and writable
- This register may be streamed
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0000
- T4:
  - Minimum firmware version: 0.2000

CIO_STATE	2502	UINT16	R/W

CIO_STATE
- Address: 2502

Read or write the state of the 4 bits of CIO in a single binary-encoded value. 0=Low AND 1=High. Does not configure direction. Reading lines set to output returns the current logic levels on the terminals, not necessarily the output states written. The upper 8-bits of this value are inhibits.

- Data type: UINT16 (type index = 0)
- Readable and writable
- This register may be streamed
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0000
- T4:
  - Minimum firmware version: 0.2000

MIO_STATE	2503	UINT16	R/W

MIO_STATE
- Address: 2503

Read or write the state of the 3 bits of MIO in a single binary-encoded value. 0=Low AND 1=High. Does not configure direction. Reading lines set to output returns the current logic levels on the terminals, not necessarily the output states written. The upper 8-bits of this value are inhibits.

- Data type: UINT16 (type index = 0)
- Readable and writable
- This register may be streamed
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0000
- T4:
  - Minimum firmware version: 0.2000

Example:

To read the digital state of all FIO lines in a bitmask, read FIO_STATE. If the result is 0b11111011, FIO2 is logic low and all other FIO lines are logic high.

FIO/EIO/CIO/MIO Direction Bitmask Registers

Name	Start Address	Type	Access
FIO_DIRECTION	2600	UINT16	R/W

FIO_DIRECTION
- Address: 2600

Read or write the direction of the 8 bits of FIO in a single binary-encoded value. 0=Input and 1=Output. The upper 8-bits of this value are inhibits.

- Data type: UINT16 (type index = 0)
- Readable and writable
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 0.9402
- T4:
  - Minimum firmware version: 0.2000

EIO_DIRECTION	2601	UINT16	R/W

EIO_DIRECTION
- Address: 2601

Read or write the direction of the 8 bits of EIO in a single binary-encoded value. 0=Input and 1=Output. The upper 8-bits of this value are inhibits.

- Data type: UINT16 (type index = 0)
- Readable and writable
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 0.9402
- T4:
  - Minimum firmware version: 0.2000

CIO_DIRECTION	2602	UINT16	R/W

CIO_DIRECTION
- Address: 2602

Read or write the direction of the 4 bits of CIO in a single binary-encoded value. 0=Input and 1=Output. The upper 8-bits of this value are inhibits.

- Data type: UINT16 (type index = 0)
- Readable and writable
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 0.9402
- T4:
  - Minimum firmware version: 0.2000

MIO_DIRECTION	2603	UINT16	R/W

MIO_DIRECTION
- Address: 2603

Read or write the direction of the 3 bits of MIO in a single binary-encoded value. 0=Input and 1=Output. The upper 8-bits of this value are inhibits.

- Data type: UINT16 (type index = 0)
- Readable and writable
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 0.9402
- T4:
  - Minimum firmware version: 0.2000

Example:

To set FIO1-7 to output, write a value of 0x01FF to FIO_DIRECTION. FIO0 is the least significant bit, so to prevent modification the corresponding inhibit bit is set with 0x01 in the most significant byte. The least significant byte is 0xFF, which is all 8 bits of FIO set to output.

Combination FIO/EIO/CIO/MIO State Registers

These registers are a combination of the FIO/EIO/CIO/MIO State registers.

Combination Direction Bitmask Registers

Name	Start Address	Type	Access
FIO_EIO_STATE	2580	UINT16	R/W

FIO_EIO_STATE
- Address: 2580

Read or write the state of the 16 bits of FIO-EIO in a single binary-encoded value. 0=Low AND 1=High. Does not configure direction. Reading lines set to output returns the current logic levels on the terminals, not necessarily the output states written.

- Data type: UINT16 (type index = 0)
- Readable and writable
- This register may be streamed
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0005
- T4:
  - Minimum firmware version: 0.2000

EIO_CIO_STATE	2581	UINT16	R/W

EIO_CIO_STATE
- Address: 2581

Read or write the state of the 12 bits of EIO-CIO in a single binary-encoded value. 0=Low AND 1=High. Does not configure direction. Reading lines set to output returns the current logic levels on the terminals, not necessarily the output states written. As of firmware 1.0172, MIO states are included in the upper nibble of the CIO byte.

- Data type: UINT16 (type index = 0)
- Readable and writable
- This register may be streamed
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0005
- T4:
  - Minimum firmware version: 0.2000

CIO_MIO_STATE	2582	UINT16	R/W

CIO_MIO_STATE
- Address: 2582

Read or write the state of the 12 bits of CIO-MIO in a single binary-encoded value. 0=Low AND 1=High. Does not configure direction. Reading lines set to output returns the current logic levels on the terminals, not necessarily the output states written.

- Data type: UINT16 (type index = 0)
- Readable and writable
- This register may be streamed
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0172
- T4:
  - Minimum firmware version: 0.2000

Other Considerations

Specifications

See Appendix A-2 for specs including:

- Low Level Input Voltage
- High Level Input Voltage
- Hysteresis Voltage
- Maximum Input Voltage
- Output Low Voltage
- Output High Voltage
- Short Circuit Current
- Output Impedance

Streaming DIO

For details about which DIO registers can be streamed look at section 3.2 Stream Mode. In short, only the FIO/EIO/CIO/MIO State registers can be streamed because stream data is transferred as 16-bit values.

Electrical Overview

All digital I/O on T-series devices are tri-state and thus have 3 possible states: input, output-high, or output-low. Each bit of I/O can be configured individually:

- When configured as an input, a bit has a ~100 kΩ pull-up resistor to 3.3 volts (all digital I/O are at least 5 volt tolerant).
- When configured as output-high, a bit is connected to the internal 3.3 volt supply

(through a series resistor).
- When configured as output-low, a bit is connected to GND (through a series resistor).

T8 Only: In addition to the above settings, the T8 can disable the pull-ups and enable pulldowns:

- By default all DIOs on the T8 have a pull-up enabled. Those pull-ups can be disabled using DIO_PULLUP_DISABLE.
- By default all DIOs on the T8 have a pull-downs disabled. The pull-downs can be enabled using DIO_PULLDOWN_ENABLE.

If a DIO terminal is at about 3.3 volts, and you are not sure if it is set to input or output-high, a couple ways to tell are:

1. Look for a slight change on a terminal with nothing connected except a DMM. For example, a DMM measurement of an input might show 3.30V whereas that same terminal as output-high reads 3.31V.
2. Add a load resistor. If you add a 100k from FIO7 to GND, the terminal should measure about 1.6V for input and 3.3V for output-high.

See Appendix A-2 for more details.

By default, the DIO lines are digital I/O, but they can also be configured as PWM Output, Quadrature Input, Counters, etc. (See 13.2 DIO Extended Feature.)

Power-up Defaults

The default condition of the digital I/O can be configured using Kipling or programmatically. From the factory, all digital I/O are configured as inputs by default. Note that even if the default for a line is changed to output-high or output-low, there could be a small time (milliseconds) during boot-up where all digital I/O are in the factory default condition, or in the case of EIO0 a unique condition. See more details in the IO Config section.

If this brief time in the input state is a problem, consider adding a pull-down resistor. For example, a 10k resistor will form a voltage divider with the internal 100k to 3.3V and result in about 0.3V on the DIO terminal when in the input state. In the output-states, this 10k will be additional load on the output-high or output-low, but often is negligible.

Protection

All the digital I/O include an internal series resistor that provides overvoltage/short-circuit protection. These series resistors also limit the ability of these lines to sink or source current. Refer to Appendix A-2.

The fact that the digital I/O are specified as 5-volt tolerant means that 5 volts can be connected to a digital input without problems (see the actual limits in the specifications in Appendix A).

Increase logic level to 5V

On-board DACs: The DAC0 and DAC1 channels can be set to 5 volts, providing 2 output lines with such capability.

LabJack LJTick-DigitalOut5V: We sell the LJTick-DigitalOut5V that converts our 3.3V outputs to 5V outputs.

Logic Buffer IC: The surefire way to get 5 volts from a digital output is to add a simple logic buffer IC that is powered by 5 volts and recognizes 3.3 volts as a high input. Consider the CD74ACT541E from TI (or the inverting CD74ACT540E). All that is needed is a few wires to bring VS, GND, and the signal from the LabJack to the chip. This chip can level shift up to eight 0/3.3 volt signals to 0/5 volt signals and provides high output drive current (+/-24 mA).

Open-collector: In some cases, an open-collector style output can be used to get a 5V signal. To get a low set the line to output-low, and to get a high set the line to input. When the line is set to input, the voltage on the line is determined by a pull-up resistor. T-series devices have an internal ~100k resistor to 3.3V, but an external resistor can be added to a different voltage. Whether this will work depends on how much current the load is going to draw and what the required logic thresholds are. Say for example a 10k resistor is added from EIO0 to VS. EIO0 has an internal 100k pull-up to 3.3 volts and a series output resistance of about 180 ohms. Assume the load draws just a few microamps or less and thus is negligible. When EIO0 is set to input, there will be 100k to 3.3 volts in parallel with 10k to 5 volts, and thus the line will sit at about 4.85 volts. When the line is set to output-low, there will be 180 ohms in series with the 10k, so the line will be pulled down to about 0.1 volts.

Waveform Generation

DIO registers described on this page may be used for software-timed square/rectangular waveform output.

For other options, see the Waveform Generation App Note.

Reading the State of an Output

Reading an output will exhibit the following behaviors:

- Reading from the DIO registers (address 2000-2022) will set the IO to input. Use DIO_STATE to read the state without affecting direction.
- When reading the state of an output, the actual voltage, as measured behind the protection resistors (see the “Protection” section above), will be used to determine whether the line is logic high or low.
- The voltage is read behind the protection resistors. This creates a voltage divider with the output circuitry. That divider causes the following behaviors:
    - An output-high will read HIGH even when the screw terminal is shorted to GND. Connecting a negative voltage may cause an output-high to read LOW.
    - A CIO or EIO line set to output-low will read HIGH when shorted to VS.
    - A FIO line set to output-low will read LOW when shorted to VS.
- At this time is it not possible to read the latches that control whether the IO is attempting to drive high or low.

Do DIO Change State Automatically?

As described in Electrical Overview above, digital I/O on T-series devices are tri-state and thus have 3 possible states: input, output-high, or output-low. The DIO stay in their current state

until told to go to some different state. There are various ways they could be told to go to a different state:

- Some software sends a command telling a DIO to go to a different state.
- Your Lua script running on the device tells a DIO to go to a different state.
- The watchdog tells DIO to go to a different state.
- The T7 reboots causing the DIO to go to the saved power-up condition.

13.1 Flexible I/O (T4 Only) [T-Series Datasheet]

Flexible I/O Overview - T4 Only

Basics: Flexible I/O are ports, channels, or lines on a LabJack device that may be configured as analog inputs, as digital inputs, or as digital outputs.

Device Control Basics:

- All T-series device features are controlled by reading and writing Modbus TCP registers via Modbus TCP (either directly or through our LJM library).
- We have register descriptions throughout documentation detailing relevant register names, starting addresses, types, and access permissions (read/write).
- See Section 3.0 Communication for other detailed communication information.

Digital I/O: 13.0 Digital I/O

Analog I/O: 14.0 Analog Inputs

Available T4 Flexible I/O Channels

As Figure 13.1-1 shows below, the LabJack T4 has 8 flexible I/O lines:

- Four screw terminals labeled FIO4 through FIO7 (also named as DIO4-DIO7 and as AIN4-AIN7)
- Four DB15 pins labeled EIO0 through EIO3 (also named as DIO8-DIO11 and as AIN8-AIN11)

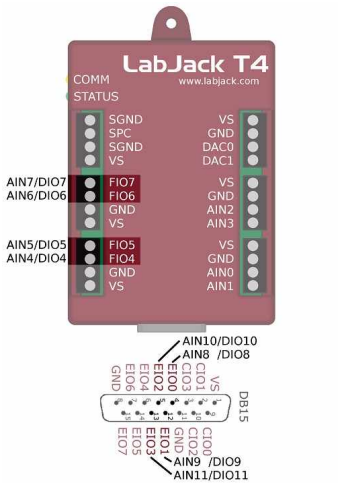


Figure 13.1-1 T4 Flexible I/O

The flexible I/O lines are readable/writable as digital I/O using the register names DIO4 through DIO11:

Name	Start Address	Type	Access
DIO#(4:11)	2004	UINT16	R/W

DIO#(4:11)
- Starting Address: 2004

Read or set the state of 1 bit of digital I/O. Also configures the direction to input or output. Read 0=Low AND 1=High. Write 0=Low AND 1=High.

- Data type: UINT16 (type index = 0)
- Readable and writable
- This register may be streamed

Expanded Names	Addresses
DIO4, DIO5, DIO6, DIO7, DIO8, DIO9, DIO10, DIO11	2004, 2005, 2006, 2007, 2008, 2009, 2010, 2011

The flexible I/O lines are readable as analog inputs using the register names AIN4 through AIN11:

Name	Start Address	Type	Access
AIN#(4:11)	8	FLOAT32	R

AIN#(4:11)
- Starting Address: 8

Returns the voltage of the specified analog input.

- Data type: FLOAT32 (type index = 3)
- Read-only
- This register may be streamed

Expanded Names	Addresses
AIN4, AIN5, AIN6, AIN7, AIN8, AIN9, AIN10, AIN11	8, 10, 12, 14, 16, 18, 20, 22

For examples on how to use these registers, see 13.0 Digital I/O and 14.0 Analog Inputs.

Flexible I/O Auto-Configuration

Flexible I/O lines will be auto-configured in some situations. Flexible I/O can also be manually configured, as described below.

Digital Inputs - Always Auto-Configured

Reading DIO4-DIO11 always auto-configures the given line to be a digital input (before returning the digital state a 1 or 0).

Digital Outputs - Not Always Auto-Configured

Writing DIO4-DIO11 only auto-configures the given line to be a digital output if the line is currently a digital input. If the channel is currently configured as an analog input, the channel will remain configured as an analog input and the write command will be ignored.

To force a flexible I/O line to be a digital output, you can read it as digital, then write to it as digital. Be aware that an analog sensor may be damaged by driving voltage into its output.

Analog Inputs — Always Auto-Configured

Reading from AIN4-AIN11 always auto-configures a channel to be an analog input.

Flexible I/O Manual and Bulk Configuration

The following registers can configure multiple flexible I/O lines at once:

Name	Start Address	Type	Access
DIO_INHIBIT	2900	UINT32	R/W

DIO_INHIBIT
- Address: 2900

A single binary-encoded value where each bit determines whether _STATE, _DIRECTION or _ANALOG_ENABLE writes affect that bit of digital I/O.
0=Default=Affected,
1=Ignored.

- Data type: UINT32 (type index = 1)
- Readable and writable
- Default value: 0

Constant	Value
Affected	0
Ignored	1

DIO_ANALOG_ENABLE	2880	UINT32	R/W

DIO_ANALOG_ENABLE
- Address: 2880

Read or write the analog configuration of all digital I/O in a single binary-encoded value. 1=Analog mode and 0=Digital mode. When switching from analog to digital, the lines will be set to input. Writes are filtered by the value in DIO_INHIBIT.

- Data type: UINT32 (type index = 1)
- Readable and writable

Constant	Value
Digital mode	0
Analog mode	1

DIO_DIRECTION	2850	UINT32	R/W

DIO_DIRECTION
- Address: 2850

Read or write the direction of all digital I/O in a single binary-encoded value. 0=Input and 1=Output. Writes are filtered by the value in DIO_INHIBIT.

- Data type: UINT32 (type index = 1)
- Readable and writable

Constant	Value
Input	0
Output	1

DIO_STATE	2800	UINT32	R/W

DIO_STATE
- Address: 2800

Read or write the state of all digital I/O in a single binary-encoded value. 0=Low AND 1=High. Does not configure direction. A read of an output returns the current logic level on the terminal, not necessarily the output state written. Writes are filtered by the value in DIO_INHIBIT.

- Data type: UINT32 (type index = 1)
- Readable and writable

Constant	Value
Low	0
High	1

To configure multiple flexible I/O lines, set the relevant bits of DIO_INHIBIT, DIO_ANALOG_ENABLE, DIO_DIRECTION, and DIO_STATE—where the relevant bits are the same as the DIO channel numbers. Examples:

- To configure DIO4 (screw terminal FIO4), set bit 4 of DIO_INHIBIT, DIO_ANALOG_ENABLE, etc.
- To configure DIO5 (screw terminal FIO5), set bit 5 of DIO_INHIBIT, DIO_ANALOG_ENABLE, etc.
- To configure DIO8 (DB15 pin EIO0), set bit 8 of DIO_INHIBIT, DIO_ANALOG_ENABLE, etc.

To configure digital input(s):

- Set the relevant bit(s) of DIO_INHIBIT to 0
- Set the relevant bit(s) of DIO_ANALOG_ENABLE to 0
- Set the relevant bit(s) of DIO_DIRECTION to 0
- Read the relevant DIO register(s) or read DIO_STATE

For example, to configure DIO4 (screw terminal FIO4) as a digital input:

- Set bit 4 of DIO_INHIBIT to 0
- Set bit 4 of DIO_ANALOG_ENABLE to 0
- Set bit 4 of DIO_DIRECTION to 0
- Read DIO4 or read bit 4 of DIO_STATE

To configure digital output(s):

- Set the relevant bit(s) of DIO_INHIBIT to 0
- Set the relevant bit(s) of DIO_ANALOG_ENABLE to 0
- Set the relevant bit(s) of DIO_DIRECTION to 1
- Write the relevant DIO register(s) or write to DIO_STATE

For example, to configure DIO4 (screw terminal FIO4) as a digital output:

- Set bit 4 of DIO_INHIBIT to 0
- Set bit 4 of DIO_ANALOG_ENABLE to 0
- Set bit 4 of DIO_DIRECTION to 0
- Write to DIO4 or write bit 4 of DIO_STATE

To configure analog input(s):

- Set the relevant bit(s) of DIO_INHIBIT to 0
- Set the relevant bit(s) of DIO_ANALOG_ENABLE to 1
- Read the relevant AIN register(s)

For example, to configure AIN4 (screw terminal FIO4) as a analog input:

- Set bit 4 of DIO_INHIBIT to 0
- Set bit 4 of DIO_ANALOG_ENABLE to 1
- Read AIN4

Tips for Constructing Bitmasks

The DIO_INHIBIT value for allowing a write command to only affect DIO4 and DIO5 is as follows:

`0x7FFFFF - (1<<4)|(1<<5)` which equals `0b11111111111111111001111` , `0x7FFFCF` , or `8388559` .

After writing `0x7FFFCF` to the DIO_INHIBIT register, the DIO_ANALOG_ENABLE value for configuring DIO4 and DIO5 as analog inputs is as follows:

`(1<<4)|(1<<5)` which equals `0b110000` , `0x30` , or `48` .

Flexible I/O Pull-Up

The T4 can disable the pull-ups on some of its IO lines. An IO line with its pull-up disabled will be floating, any readings will not be meaningful until an external signal is connected. Pull-ups can be disabled on FIO4(DIO4) through EIO3 (DIO11).

Name	Start Address	Type	Access
DIO_PULLUP_DISABLE	2890	UINT32	R/W

DIO_PULLUP_DISABLE
- Address: 2890

This register will prevent pullups from being enabled on lines set to digital input. This is a binary coded value where bit 0 represent FIO0 and bit 11 represents EIO3. 1 = pullup disabled, 0 = pullup enabled. This register is not affected by the inhibit register.

- Data type: UINT32 (type index = 1)
- Readable and writable
- T8:
  - T8 description TBD
- T4:
  - This register only affects flex-lines which can be configured as analog or digital.

13.2 DIO Extended Features [T-Series Datasheet]

Overview

Basics: DIO Extended Features, commonly refered to as "DIO-EF", allow T-Series devices to measure and generate digital waveforms that are more advanced than logic high or logic low. They expose features such as PWM output for servo motor control, Quadrature input for incremental/quadrature encoders, and more.

Device Control Basics:

- All T-series device features are controlled by reading and writing Modbus TCP registers via Modbus TCP (either directly or through our LJM library).
- We have register descriptions throughout documentation detailing relevant register names, starting addresses, types, and access permissions (read/write).
- See Section 3.0 Communication for other detailed communication information.

Register Numbering: DIO-EFs are configured and used through the DIO#(0:22)_EF registers. The numbering of these registers corresponds with the DIO numbers documented in section 13.0 Digital I/O.

Configuration and how to use: The meanings of each of the DIO#_EF_CONFIG registers and DIO#_EF_READ registers changes depending on what DIO-EF index (DIO#_EF_INDEX) is configured, however the general configuration process is the same and is described below. It is helpful to think of DIO-EF features as "sub-systems" that need to be configured before they are started. Once they are started, they can be interacted with by reading the system state and updating system configurations.

DIO-EF System Configurations:

1. Select a feature and determine the number of required DIO lines using the reference tables below.
2. Ensure that the DIO-EF is disabled by writing a 0 to the appropriate DIO#_EF_ENABLE register.
3. If required by the selected DIO-EF feature, configure the DIO-EF clock source.
4. Write the selected feature's index value to the appropriate DIO#_EF_INDEX register.
5. If required by the selected DIO-EF feature, write to the DIO#_EF_OPTIONS register.
6. If required by the selected DIO-EF feature, write to the DIO#_EF_CONFIG registers.
7. Enable the selected DIO-EF feature by writing a 1 to the appropriate DIO#_EF_ENABLE register.

Once a DIO-EF has been started, it can be interacted with using the following registers

- If the selected DIO-EF produces data, read the results from the DIO#_EF_READ registers.
- *(E.g., if DIO6 is configured as an Interrupt Counter, you can read the current count from DIO6_EF_READ_A.)*
- If the selected DIO-EF can be updated on the fly, write to the DIO#_EF_CONFIG registers.
- *(E.g., if DIO0 is configured as a PWM Out, you can update the duty cycle by writing to*

DIO0_EF_CONFIG_A.)

T4

Table 13.2-1.T4 Digital I/O Extended Features

T4 Digital I/O Extended Features		AIN (0-3)				FIO (4-7)				EIO (0-7)								CIO (0-3)			
		DIO																			
Feature	Index#					4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
PWM Out	0							✔	✔												
PWM Out with Phase	1							✔	✔												
Pulse Out	2							✔	✔												
Frequency In	3,4					✔	✔														
Pulse Width In	5					✔	✔														
Line-to-Line In*	6					✔	✔														
High-Speed Counter	7																	✔	✔	✔	✔
Interrupt Counter	8					✔	✔	✔	✔	✔	✔										
Interrupt Counter with Debounce	9					✔	✔	✔	✔	✔	✔										
Quadrature In*	10					✔	✔	✔	✔	✔	✔										
Interrupt Frequency In	11					✔	✔	✔	✔	✔	✔										
Conditional Reset	12					✔	✔	✔	✔	✔	✔										

T7

Table 13.2-2.T7 Digital I/O Extended Features

T7 Digital I/O Extended Features		FIO (0-7)								EIO (0-7)								CIO (0-3)				M
		DIO																				
Feature	Index#	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	2
PWM Out	0	✔		✔	✔	✔	✔															
PWM Out with Phase	1	✔		✔	✔	✔	✔															
Pulse Out	2	✔		✔	✔	✔	✔															
Frequency In	3,4	✔	✔																			
Pulse Width In	5	✔	✔																			
Line-to-Line In*	6	✔	✔																			
High-Speed Counter	7																	✔	✔	✔	✔	
Interrupt Counter	8	✔	✔	✔	✔			✔	✔													
Interrupt Counter with Debounce	9	✔	✔	✔	✔			✔	✔													
Quadrature In*	10	✔	✔	✔	✔			✔	✔													
Interrupt Frequency In	11	✔	✔	✔	✔			✔	✔													
Conditional Reset	12	✔	✔	✔	✔			✔	✔													

T8

Table 13.2-3.T8 Digital I/O Extended Features

T8 Digital I/O Extended Features		FIO (0-7)								EIO (0-7)								CIO (0-3)			
		DIO																			
Feature	Index#	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	
PWM Out	0			✔	✔	✔	✔	✔	✔	✔			✔	✔							
PWM Out with Phase	1			✔	✔	✔	✔	✔	✔	✔			✔	✔							
Pulse Out	2			✔	✔	✔	✔	✔	✔	✔			✔	✔							
Frequency In	3,4	✔	✔	✔	✔	✔	✔			✔			✔		✔						
Pulse Width In	5	✔	✔	✔	✔	✔	✔			✔			✔		✔						
Line-to-Line In*	6	✔	✔	✔	✔	✔	✔			✔			✔		✔						
High-Speed Counter	7							✔	✔	✔		✔			✔	✔	✔				
Interrupt Counter	8	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔				
Interrupt Counter with Debounce	9	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔				
Quadrature In*	10	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔				
Interrupt Frequency In	11	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔				
Conditional Reset	12	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔				

* Line-to-Line In and Quadrature In both require two DIO lines.

* Line-to-Line In and Quadrature In both require two DIO lines.

Kipling Walkthroughs: Kipling's Register Matrix can be used to perform DIO-EF features. Some examples:

- Configuring & Reading a Counter
- Configuring & Reading Frequency
- Configuring a PWM Output

DIO-EF Enable/Disable

This register is used to disable a DIO-EF feature (in order to configure it) and also used to start or enable the DIO-EF subsystem.

A DIO-EF doesn't always need to be disabled for it to be configured, depending on the DIO-EF being enabled.

Name	Start Address	Type	Access
DIO#(0:22)_EF_ENABLE	44000	UINT32	R/W

DIO#(0:22)_EF_ENABLE
- Starting Address: 44000

1 = enabled. 0 = disabled. Must be disabled during configuration. Note that DIO-EF reads work when disabled and do not return an error.

- Data type: UINT32 (type index = 1)
- Readable and writable
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 0.8800

Expanded Names	Addresses
DIO0_EF_ENABLE, DIO1_EF_ENABLE, DIO2_EF_ENABLE, DIO3_EF_ENABLE, DIO4_EF_ENABLE, DIO5_EF_ENABLE, DIO6_EF_ENABLE, DIO7_EF_ENABLE, DIO8_EF_ENABLE, DIO9_EF_ENABLE, DIO10_EF_ENABLE, DIO11_EF_ENABLE, DIO12_EF_ENABLE, DIO13_EF_ENABLE, DIO14_EF_ENABLE, DIO15_EF_ENABLE, DIO16_EF_ENABLE, DIO17_EF_ENABLE, DIO18_EF_ENABLE, DIO19_EF_ENABLE, DIO20_EF_ENABLE, DIO21_EF_ENABLE, DIO22_EF_ENABLE	44000, 44002, 44004, 44006, 44008, 44010, 44012, 44014, 44016, 44018, 44020, 44022, 44024, 44026, 44028, 44030, 44032, 44034, 44036, 44038, 44040, 44042, 44044

DIO-EF Index (Feature Selection)

This register is used to select the extended feature that will get enabled on a given DIO line. The valid DIO lines differ by device. For more specific details look at reference tables 13.2-1 and 13.2-2 as well as the appropriate DIO-EF feature subsection.

Name	Start Address	Type	Access
DIO#(0:22)_EF_INDEX	44100	UINT32	R/W

DIO#(0:22)_EF_INDEX
- Starting Address: 44100

An index to specify the feature you want.

- Data type: UINT32 (type index = 1)
- Readable and writable
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 0.8800

Expanded Names	Addresses
DIO0_EF_INDEX, DIO1_EF_INDEX, DIO2_EF_INDEX, DIO3_EF_INDEX, DIO4_EF_INDEX, DIO5_EF_INDEX, DIO6_EF_INDEX, DIO7_EF_INDEX, DIO8_EF_INDEX, DIO9_EF_INDEX, DIO10_EF_INDEX, DIO11_EF_INDEX, DIO12_EF_INDEX, DIO13_EF_INDEX, DIO14_EF_INDEX, DIO15_EF_INDEX, DIO16_EF_INDEX, DIO17_EF_INDEX, DIO18_EF_INDEX, DIO19_EF_INDEX, DIO20_EF_INDEX, DIO21_EF_INDEX, DIO22_EF_INDEX	44100, 44102, 44104, 44106, 44108, 44110, 44112, 44114, 44116, 44118, 44120, 44122, 44124, 44126, 44128, 44130, 44132, 44134, 44136, 44138, 44140, 44142, 44144

DIO-EF Options and Clock Source Selection

This register isn't used by all DIO-EF features.

If a DIO-EF feature requires the configuration or selection of a clock source (such as PWM Out does), the configuration of this register is required, since it is required for selecting a clock source. See 13.2.1 EF Clock Source for more details about clock source selection.

Name	Start Address	Type	Access
DIO#(0:22)_EF_OPTIONS	44200	UINT32	R/W

DIO#(0:22)_EF_OPTIONS
- Starting Address: 44200

Function dependent on selected feature index.

- Data type: UINT32 (type index = 1)
- Readable and writable
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 0.8800

Expanded Names	Addresses
DIO0_EF_OPTIONS, DIO1_EF_OPTIONS, DIO2_EF_OPTIONS, DIO3_EF_OPTIONS, DIO4_EF_OPTIONS, DIO5_EF_OPTIONS, DIO6_EF_OPTIONS, DIO7_EF_OPTIONS, DIO8_EF_OPTIONS, DIO9_EF_OPTIONS, DIO10_EF_OPTIONS, DIO11_EF_OPTIONS, DIO12_EF_OPTIONS, DIO13_EF_OPTIONS, DIO14_EF_OPTIONS, DIO15_EF_OPTIONS, DIO16_EF_OPTIONS, DIO17_EF_OPTIONS, DIO18_EF_OPTIONS, DIO19_EF_OPTIONS, DIO20_EF_OPTIONS, DIO21_EF_OPTIONS, DIO22_EF_OPTIONS	44200, 44202, 44204, 44206, 44208, 44210, 44212, 44214, 44216, 44218, 44220, 44222, 44224, 44226, 44228, 44230, 44232, 44234, 44236, 44238, 44240, 44242, 44244

DIO-EF Configuration

Configuration registers serve two purposes. They provide a location for settings that need to be configured upon DIO-EF enable and they provide a location for settings that users may need to use to update a DIO-EF feature once it has been enabled.

Initial Configuration: Configuration is the initial setup of the Extended Feature. Configuration requires that any DIO-EF running at the pin in question first be disabled. Options can then be loaded. Then the DIO-EF can be enabled.

Update: Some DIO#_CONFIG registers can be updated while a DIO-EF is running. Updating allows the DIO-EF to change its operation parameters without restarting. Note that the clock source and feature index cannot be changed in an update. Depending on the feature, reads and writes to the update registers have small differences. See the Update portion of each feature for more information.

Name	Start Address	Type	Access
DIO#(0:22)_EF_CONFIG_A	44300	UINT32	R/W

DIO#(0:22)_EF_CONFIG_A
- Starting Address: 44300

Function dependent on selected feature index.

- Data type: UINT32 (type index = 1)
- Readable and writable
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 0.8800

Expanded Names	Addresses
DIO0_EF_CONFIG_A, DIO1_EF_CONFIG_A, DIO2_EF_CONFIG_A, DIO3_EF_CONFIG_A, DIO4_EF_CONFIG_A, DIO5_EF_CONFIG_A, DIO6_EF_CONFIG_A, DIO7_EF_CONFIG_A, DIO8_EF_CONFIG_A, DIO9_EF_CONFIG_A, DIO10_EF_CONFIG_A, DIO11_EF_CONFIG_A, DIO12_EF_CONFIG_A, DIO13_EF_CONFIG_A, DIO14_EF_CONFIG_A, DIO15_EF_CONFIG_A, DIO16_EF_CONFIG_A, DIO17_EF_CONFIG_A, DIO18_EF_CONFIG_A, DIO19_EF_CONFIG_A, DIO20_EF_CONFIG_A, DIO21_EF_CONFIG_A, DIO22_EF_CONFIG_A	44300, 44302, 44304, 44306, 44308, 44310, 44312, 44314, 44316, 44318, 44320, 44322, 44324, 44326, 44328, 44330, 44332, 44334, 44336, 44338, 44340, 44342, 44344

DIO#(0:22)_EF_CONFIG_B	44400	UINT32	R/W

DIO#(0:22)_EF_CONFIG_B
- Starting Address: 44400

Function dependent on selected feature index.

- Data type: UINT32 (type index = 1)
- Readable and writable
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 0.8800

Expanded Names	Addresses
DIO0_EF_CONFIG_B, DIO1_EF_CONFIG_B, DIO2_EF_CONFIG_B, DIO3_EF_CONFIG_B, DIO4_EF_CONFIG_B, DIO5_EF_CONFIG_B, DIO6_EF_CONFIG_B, DIO7_EF_CONFIG_B, DIO8_EF_CONFIG_B, DIO9_EF_CONFIG_B, DIO10_EF_CONFIG_B, DIO11_EF_CONFIG_B, DIO12_EF_CONFIG_B, DIO13_EF_CONFIG_B, DIO14_EF_CONFIG_B, DIO15_EF_CONFIG_B, DIO16_EF_CONFIG_B, DIO17_EF_CONFIG_B, DIO18_EF_CONFIG_B, DIO19_EF_CONFIG_B, DIO20_EF_CONFIG_B, DIO21_EF_CONFIG_B, DIO22_EF_CONFIG_B	44400, 44402, 44404, 44406, 44408, 44410, 44412, 44414, 44416, 44418, 44420, 44422, 44424, 44426, 44428, 44430, 44432, 44434, 44436, 44438, 44440, 44442, 44444

DIO#(0:22)_EF_CONFIG_C	44500	UINT32	R/W

DIO#(0:22)_EF_CONFIG_C
- Starting Address: 44500

Function dependent on selected feature index.

- Data type: UINT32 (type index = 1)
- Readable and writable
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 0.8800

Expanded Names	Addresses
DIO0_EF_CONFIG_C, DIO1_EF_CONFIG_C, DIO2_EF_CONFIG_C, DIO3_EF_CONFIG_C, DIO4_EF_CONFIG_C, DIO5_EF_CONFIG_C, DIO6_EF_CONFIG_C, DIO7_EF_CONFIG_C, DIO8_EF_CONFIG_C, DIO9_EF_CONFIG_C, DIO10_EF_CONFIG_C, DIO11_EF_CONFIG_C, DIO12_EF_CONFIG_C, DIO13_EF_CONFIG_C, DIO14_EF_CONFIG_C, DIO15_EF_CONFIG_C, DIO16_EF_CONFIG_C, DIO17_EF_CONFIG_C, DIO18_EF_CONFIG_C, DIO19_EF_CONFIG_C, DIO20_EF_CONFIG_C, DIO21_EF_CONFIG_C, DIO22_EF_CONFIG_C	44500, 44502, 44504, 44506, 44508, 44510, 44512, 44514, 44516, 44518, 44520, 44522, 44524, 44526, 44528, 44530, 44532, 44534, 44536, 44538, 44540, 44542, 44544

DIO#(0:22)_EF_CONFIG_D	44600	UINT32	R/W

DIO#(0:22)_EF_CONFIG_D
- Starting Address: 44600

Function dependent on selected feature index.

- Data type: UINT32 (type index = 1)
- Readable and writable
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 0.8800

Expanded Names	Addresses
DIO0_EF_CONFIG_D, DIO1_EF_CONFIG_D, DIO2_EF_CONFIG_D, DIO3_EF_CONFIG_D, DIO4_EF_CONFIG_D, DIO5_EF_CONFIG_D, DIO6_EF_CONFIG_D, DIO7_EF_CONFIG_D, DIO8_EF_CONFIG_D, DIO9_EF_CONFIG_D, DIO10_EF_CONFIG_D, DIO11_EF_CONFIG_D, DIO12_EF_CONFIG_D, DIO13_EF_CONFIG_D, DIO14_EF_CONFIG_D, DIO15_EF_CONFIG_D, DIO16_EF_CONFIG_D, DIO17_EF_CONFIG_D, DIO18_EF_CONFIG_D, DIO19_EF_CONFIG_D, DIO20_EF_CONFIG_D, DIO21_EF_CONFIG_D, DIO22_EF_CONFIG_D	44600, 44602, 44604, 44606, 44608, 44610, 44612, 44614, 44616, 44618, 44620, 44622, 44624, 44626, 44628, 44630, 44632, 44634, 44636, 44638, 44640, 44642, 44644

DIO-EF Basic Read Registers

Some DIO-EF produce results or provide status information that can be read. This information is usually a binary integer. When possible, the T-series device will convert the binary integer into a real-world unit such as seconds. When available, converted values can be read from the registers designated with “_F”.

Name	Start Address	Type	Access
DIO#(0:21)_EF_READ_A	3000	UINT32	R

DIO#(0:21)_EF_READ_A
- Starting Address: 3000

Reads an unsigned integer value. The meaning of the integer is dependent on selected feature index.

- Data type: UINT32 (type index = 1)
- Read-only
- This register may be streamed

Expanded Names	Addresses
DIO0_EF_READ_A, DIO1_EF_READ_A, DIO2_EF_READ_A, DIO3_EF_READ_A, DIO4_EF_READ_A, DIO5_EF_READ_A, DIO6_EF_READ_A, DIO7_EF_READ_A, DIO8_EF_READ_A, DIO9_EF_READ_A, DIO10_EF_READ_A, DIO11_EF_READ_A, DIO12_EF_READ_A, DIO13_EF_READ_A, DIO14_EF_READ_A, DIO15_EF_READ_A, DIO16_EF_READ_A, DIO17_EF_READ_A, DIO18_EF_READ_A, DIO19_EF_READ_A, DIO20_EF_READ_A, DIO21_EF_READ_A	3000, 3002, 3004, 3006, 3008, 3010, 3012, 3014, 3016, 3018, 3020, 3022, 3024, 3026, 3028, 3030, 3032, 3034, 3036, 3038, 3040, 3042

DIO#(0:21)_EF_READ_B	3200	UINT32	R

DIO#(0:21)_EF_READ_B
- Starting Address: 3200

Reads an unsigned integer value. The meaning of the integer is dependent on selected feature index.

- Data type: UINT32 (type index = 1)
- Read-only
- This register may be streamed

Expanded Names	Addresses
DIO0_EF_READ_B, DIO1_EF_READ_B,	3200, 3202,
DIO2_EF_READ_B, DIO3_EF_READ_B,	3204, 3206,
DIO4_EF_READ_B, DIO5_EF_READ_B,	3208, 3210,
DIO6_EF_READ_B, DIO7_EF_READ_B,	3212, 3214,
DIO8_EF_READ_B, DIO9_EF_READ_B,	3216, 3218,
DIO10_EF_READ_B, DIO11_EF_READ_B,	3220, 3222,
DIO12_EF_READ_B, DIO13_EF_READ_B,	3224, 3226,
DIO14_EF_READ_B, DIO15_EF_READ_B,	3228, 3230,
DIO16_EF_READ_B, DIO17_EF_READ_B,	3232, 3234,
DIO18_EF_READ_B, DIO19_EF_READ_B,	3236, 3238,
DIO20_EF_READ_B, DIO21_EF_READ_B	3240, 3242

DIO#(0:21)_EF_READ_A_F	3500	FLOAT32	R

DIO#(0:21)_EF_READ_A_F
- Starting Address: 3500

Reads a floating point value. The meaning of value is dependent on selected feature index.

- Data type: FLOAT32 (type index = 3)
- Read-only

Expanded Names	Addresses
DIO0_EF_READ_A_F, DIO1_EF_READ_A_F,	3500, 3502,
DIO2_EF_READ_A_F, DIO3_EF_READ_A_F,	3504, 3506,
DIO4_EF_READ_A_F, DIO5_EF_READ_A_F,	3508, 3510,
DIO6_EF_READ_A_F, DIO7_EF_READ_A_F,	3512, 3514,
DIO8_EF_READ_A_F, DIO9_EF_READ_A_F,	3516, 3518,
DIO10_EF_READ_A_F, DIO11_EF_READ_A_F,	3520, 3522,
DIO12_EF_READ_A_F, DIO13_EF_READ_A_F,	3524, 3526,
DIO14_EF_READ_A_F, DIO15_EF_READ_A_F,	3528, 3530,
DIO16_EF_READ_A_F, DIO17_EF_READ_A_F,	3532, 3534,
DIO18_EF_READ_A_F, DIO19_EF_READ_A_F,	3536, 3538,
DIO20_EF_READ_A_F, DIO21_EF_READ_A_F	3540, 3542

DIO#(0:21)_EF_READ_B_F	3700	FLOAT32	R

DIO#(0:21)_EF_READ_B_F
- Starting Address: 3700

Reads a floating point value. The meaning of value is dependent on selected feature index.

- Data type: FLOAT32 (type index = 3)
- Read-only

Expanded Names	Addresses
DIO0_EF_READ_B_F, DIO1_EF_READ_B_F, DIO2_EF_READ_B_F, DIO3_EF_READ_B_F, DIO4_EF_READ_B_F, DIO5_EF_READ_B_F, DIO6_EF_READ_B_F, DIO7_EF_READ_B_F, DIO8_EF_READ_B_F, DIO9_EF_READ_B_F, DIO10_EF_READ_B_F, DIO11_EF_READ_B_F, DIO12_EF_READ_B_F, DIO13_EF_READ_B_F, DIO14_EF_READ_B_F, DIO15_EF_READ_B_F, DIO16_EF_READ_B_F, DIO17_EF_READ_B_F, DIO18_EF_READ_B_F, DIO19_EF_READ_B_F, DIO20_EF_READ_B_F, DIO21_EF_READ_B_F	3700, 3702, 3704, 3706, 3708, 3710, 3712, 3714, 3716, 3718, 3720, 3722, 3724, 3726, 3728, 3730, 3732, 3734, 3736, 3738, 3740, 3742

DIO-EF Read-and-Reset Registers

Some DIO-EF can be reset while they are running. Resetting can have different results depending on the feature. For instance, counters are reset to zero.

Name	Start Address	Type	Access
DIO#(0:21)_EF_READ_A_AND_RESET	3100	UINT32	R

DIO#(0:21)_EF_READ_A_AND_RESET
- Starting Address: 3100

Reads the same value as DIO#(0:22)_EF_READ_A and forces a reset.

- Data type: UINT32 (type index = 1)
- Read-only
- This register may be streamed

Expanded Names	Addresses
DIO0_EF_READ_A_AND_RESET,	3100,
DIO1_EF_READ_A_AND_RESET,	3102,
DIO2_EF_READ_A_AND_RESET,	3104,
DIO3_EF_READ_A_AND_RESET,	3106,
DIO4_EF_READ_A_AND_RESET,	3108,
DIO5_EF_READ_A_AND_RESET,	3110,
DIO6_EF_READ_A_AND_RESET,	3112,
DIO7_EF_READ_A_AND_RESET,	3114,
DIO8_EF_READ_A_AND_RESET,	3116,
DIO9_EF_READ_A_AND_RESET,	3118,
DIO10_EF_READ_A_AND_RESET,	3120,
DIO11_EF_READ_A_AND_RESET,	3122,
DIO12_EF_READ_A_AND_RESET,	3124,
DIO13_EF_READ_A_AND_RESET,	3126,
DIO14_EF_READ_A_AND_RESET,	3128,
DIO15_EF_READ_A_AND_RESET,	3130,
DIO16_EF_READ_A_AND_RESET,	3132,
DIO17_EF_READ_A_AND_RESET,	3134,
DIO18_EF_READ_A_AND_RESET,	3136,
DIO19_EF_READ_A_AND_RESET,	3138,
DIO20_EF_READ_A_AND_RESET,	3140,
DIO21_EF_READ_A_AND_RESET	3142

DIO#(0:21)_EF_READ_A_F_AND_RESET	3600	FLOAT32	R

DIO#(0:21)_EF_READ_A_F_AND_RESET
- Starting Address: 3600

Reads a floating point value and forces a reset. The meaning of value is dependent on selected feature index.

- Data type: FLOAT32 (type index = 3)
- Read-only

Expanded Names	Addresses
DIO0_EF_READ_A_F_AND_RESET,	3600,
DIO1_EF_READ_A_F_AND_RESET,	3602,
DIO2_EF_READ_A_F_AND_RESET,	3604,
DIO3_EF_READ_A_F_AND_RESET,	3606,
DIO4_EF_READ_A_F_AND_RESET,	3608,
DIO5_EF_READ_A_F_AND_RESET,	3610,
DIO6_EF_READ_A_F_AND_RESET,	3612,
DIO7_EF_READ_A_F_AND_RESET,	3614,
DIO8_EF_READ_A_F_AND_RESET,	3616,
DIO9_EF_READ_A_F_AND_RESET,	3618,
DIO10_EF_READ_A_F_AND_RESET,	3620,
DIO11_EF_READ_A_F_AND_RESET,	3622,
DIO12_EF_READ_A_F_AND_RESET,	3624,
DIO13_EF_READ_A_F_AND_RESET,	3626,
DIO14_EF_READ_A_F_AND_RESET,	3628,
DIO15_EF_READ_A_F_AND_RESET,	3630,
DIO16_EF_READ_A_F_AND_RESET,	3632,
DIO17_EF_READ_A_F_AND_RESET,	3634,
DIO18_EF_READ_A_F_AND_RESET,	3636,
DIO19_EF_READ_A_F_AND_RESET,	3638,
DIO20_EF_READ_A_F_AND_RESET,	3640,
DIO21_EF_READ_A_F_AND_RESET	3642

Streaming DIO-EF Results

Though all operations discussed in this section are supported in command-response mode, some DIO-EF features can be read fast enough to be streamed:

- Frequency In
- Pulse Width In
- High-Speed Counter
- Interrupt Counter
- Interrupt Counter with Debounce
- Quadrature In
- Interrupt Frequency In

In stream mode, you can read from the integer READ registers (A, B, A_AND_RESET), but as mentioned in 3.2 Stream, those reads only return the lower 16 bits so you need to also use STREAM_DATA_CAPTURE_16 in the scan list to get the upper 16 bits.

Other Considerations

Specifications

See Appendix A-2 for specs including:

- Frequency Output
- Counter Input Frequency
- Minimum High & Low Time
- "Interrupt" Total Edge Rate

System Timer

Complications can occur if streaming while enabling a DIO-EF that requires the use of a system timer. Please contact LabJack support if you need to do this.

13.2.1 EF Clock Source [T-Series Datasheet]

Overview

The clock source settings produce the reference frequencies used to generate output waveforms and measure input waveforms. They control output frequency, PWM resolution, maximum measurable period, and measurement resolution. In general, a slower Clock#Frequency will increase the maximum measurable period, and a faster Clock#Frequency will increase measurement resolution.

```
Clock#Frequency = CoreFrequency / DIO_EF_CLOCK#_DIVISOR    // typically 80M/Divisor
```

The CoreFrequency varies by device. See the Clock Speed section for information about your device clock speed settings.

The CoreFrequency is generated by hardware and will have the same specs as the system clock listed in Section A-5.

The valid values for DIO_EF_CLOCK#_DIVISOR are 1, 2, 4, 8, 16, 32, 64, or 256, and a value of 0 (default) equates to a divisor of 1.

There are 3 DIO-EF clock sources available. Each clock source has an associated bit size and several mutual exclusions. Mutual exclusions exist because the clock sources share hardware with other features. A clock source is created from a hardware counter. CLOCK1 uses COUNTER_A (CIO0) and CLOCK2 uses COUNTER_B (CIO1). The 32-bit clock source (CLOCK0) is created by combining the 2 16-bit clock sources (CLOCK1 CLOCK2). The following list provides clock source bit sizes and mutual exclusions:

T4/T7:

- CLOCK0: 32-bit. Mutual Exclusions: CLOCK1, CLOCK2, High-Speed Counter on CIO0, High-Speed Counter on CIO1
- CLOCK1: 16-bit. Mutual Exclusions: CLOCK0, High-Speed Counter on CIO0
- CLOCK2: 16-bit. Mutual Exclusions: CLOCK0, High-Speed Counter on CIO1

T8:

- CLOCK0: 32-bit. Mutual Exclusions: CLOCK1, CLOCK2, High-Speed Counter on EIO6, High-Speed Counter on EIO7
- CLOCK1: 16-bit. Mutual Exclusions: CLOCK0, High-Speed Counter on EIO6
- CLOCK2: 16-bit. Mutual Exclusions: CLOCK0, High-Speed Counter on EIO7

The clock source is not a DIO-EF feature, but the four basic operations of Configure, Read, Update, and Reset still apply.

Configure

There are four registers associated with the configuration of clock sources:

Digital EF Clock Source

Name	Start Address	Type	Access
DIO_EF_CLOCK0_ENABLE	44900	UINT16	R/W

DIO_EF_CLOCK0_ENABLE
- Address: 44900

1 = enabled. 0 = disabled. Must be disabled during configuration.

- Data type: UINT16 (type index = 0)
- Readable and writable
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 0.8700

DIO_EF_CLOCK0_DIVISOR	44901	UINT16	R/W

DIO_EF_CLOCK0_DIVISOR
- Address: 44901

Divides the core clock. Valid options: 1,2,4,8,16,32,64,256.

- Data type: UINT16 (type index = 0)
- Readable and writable
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 0.9320

DIO_EF_CLOCK0_OPTIONS	44902	UINT32	R/W

DIO_EF_CLOCK0_OPTIONS
- Address: 44902

Bitmask: bit0: 1 = use external clock. All other bits reserved.

- Data type: UINT32 (type index = 1)
- Readable and writable
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 0.8700

DIO_EF_CLOCK0_ROLL_VALUE	44904	UINT32	R/W

DIO_EF_CLOCK0_ROLL_VALUE
- Address: 44904

The clock count will increment continuously and then start over at zero as it reaches the roll value. DIO_EF_CLOCK0 is a 32-bit clock, valid values are 0 to (2^32 - 1). 0 results in the max roll value (2^32).

- Data type: UINT32 (type index = 1)
- Readable and writable
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 0.8700

A clock source can be enabled after DIO#_EF_INDEX has been configured. This allows several DIO-EFs to be started at the same time.

DIO_EF_CLOCK0_ROLL_VALUE is a 32-bit value (0-4294967295) if using a 32-bit clock, and a 16-bit value (0-65535) if using a 16-bit clock. 0 results in the max roll value possible (2^{32} for 32-bit clocks, 2^{16} for 16-bit clocks).

Read

To read the clock, read DIO_EF_CLOCK0_COUNT:

Digital EF Clock Source

Name	Start Address	Type	Access
DIO_EF_CLOCK0_COUNT	44908	UINT32	R

DIO_EF_CLOCK0_COUNT
- Address: 44908

Current tick count of this clock. Will read between 0 and ROLL_VALUE-1.

- Data type: UINT32 (type index = 1)
- Read-only
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 0.9430

This can be useful for generating timestamps.

Update

Both the ROLL_VALUE and the DIVISOR can be written while a clock source is running. As long as the clock source's period is greater than 50 µs, the clock will seamlessly switch to the new settings.

Reset

At this time there are no reset operations available for the DIO-EF clock sources.

Example

Configure CLOCK0 so that a PWM output (index=0) will have a frequency of 10 Hz.

T4/T7

```
DIO_EF_CLOCK0_ENABLE = 0
DIO_EF_CLOCK0_DIVISOR = 8 # 80 MHz / 8 = 10
MHz
DIO_EF_CLOCK0_ROLL_VALUE = 1000000
DIO_EF_CLOCK0_ENABLE = 1
```

A frequency input measurement (index=3/4) will be able to count from 0-999999 with each count equal to 0.1 microseconds, and thus a max period of just under 0.1 seconds.

T8

```
DIO_EF_CLOCK0_ENABLE = 0
```

```
DIO_EF_CLOCK0_DIVISOR = 16 # 100 MHz / 16 = 6.25
MHz
```

```
DIO_EF_CLOCK0_ROLL_VALUE = 625000
```

```
DIO_EF_CLOCK0_ENABLE = 1
```

A frequency input measurement (index=3/4) will be able to count from 0-624999 with each count equal to 0.16 microseconds, and thus a max period of just under 0.1 seconds.

13.2.2 PWM Out [T-Series Datasheet]

Overview

T4 Capable DIO: **DIO6, DIO7** (aka FIO6, FIO7)

T7 Capable DIO: **DIO0, DIO2, DIO3, DIO4, DIO5** (aka FIO0, FIO2, FIO3, FIO4, FIO5)

T8 Capable DIO: **DIO2, DIO3, DIO4, DIO5, DIO6, DIO7, DIO8, DIO11, DIO12** (aka FIO2, FIO3, FIO4, FIO5, FIO6, FIO7, EIO0, EIO3, EIO4)

Requires Clock Source: **Yes**

Index: **0**

Streamable: **No**

This PWM Out Extended Feature generates a pulse width modulated wave form.

Operation

PWM output will set the DIO high and low relative to the clock source's count. When the count is zero the DIO line will be set high. When the count matches Config A the line will be set low. Therefore Config A is used to control the duty cycle and the resolution is equal to the roll value.

```
Clock#Frequency = CoreFrequency /
DIO_EF_CLOCK#_DIVISOR
PWMFrequency = Clock#Frequency / DIO_EF_CLOCK#_ROLL_VALUE
DutyCycle% = 100 * DIO#_EF_CONFIG_A /
DIO_EF_CLOCK#_ROLL_VALUE
```

See the DIO-EF Clock Source section for more information about your device core frequency and DIO_EF clock source settings.

DIO_EF_CLOCK#_ROLL_VALUE is a 32-bit value for CLOCK0 and a 16-bit value for CLOCK1 & CLOCK2. A value of 0 corresponds to the max roll value of 2^{32} for the 32-bit clock or 2^{16} for 16-bit clocks.

PWM Out is capable of glitch-free updates in most situations. A glitch-free update means that the PWM will finish the current period consisting of the high time then the low time before loading the new value. The next period will then have the new duty cycle. This is true for all cases except zero. When setting the duty cycle to zero, the line will be set low regardless of the current position. This means that a single high pulse with duration between zero and the previous high time can be output before the line goes low.

The clock roll value can take up to one PWM period to update and this does not block subsequent commands from being processed. It is possible to finish updating DIO_EF_CONFIG_A to a value greater than the non-updated clock roll value (which is invalid) but less than the updated clock roll value (which is valid) and throw the error 2565: EF_VALUE_GREATER_THAN_PERIOD.

Potential fixes:

- disable and re-enable the clock line before updating the clock roll value and duty cycle value.
- Delay for greater than one "non-updated" PWM period between updating the clock roll value and updating the duty cycle value

Configure

DIO#_EF_ENABLE: 0 = Disable, 1 = Enable
DIO#_EF_INDEX: 0
DIO#_EF_OPTIONS: Bits 0-2 specify which clock source to use ... b000 for Clock0, b001 for Clock1, and b010 for Clock2. All other bits are reserved and should be set to 0.
DIO#_EF_CONFIG_A: When the specified clocks source's count matches this value, the line will transition from high to low.
DIO#_EF_CONFIG_B: Not used.
DIO#_EF_CONFIG_C: Not used.
DIO#_EF_CONFIG_D: Not used.

Update

The duty cycle can be updated at any time. To update, write the new value to DIO#_EF_CONFIG_A. The new value will not be used until the clock source rolls to zero. This means that at the end of the current period, the new value will be loaded—resulting in a glitch-free transition.

Read

No information is returned by PWM Out.

Reset

Reset has no effect on this feature.

Example

To generate a 10 kHz PWM starting at 25% DC, first configure the clock source. The higher the roll value, the greater the duty cycle resolution will be. For the highest resolution, you must maximize the roll value, so use the smallest clock divisor that will not result in a roll value greater than the clock source's size (32-bits or 16-bits). For a more detailed walkthrough, see Configuring a PWM Output.

T4/T7 Pseudocode

With a divisor of 1, the roll value will be 8000:

```
80 MHz / (1 * 8000) = 10
kHz
```

Set the clock registers accordingly:

```
DIO_EF_CLOCK0_ENABLE = 0
DIO_EF_CLOCK0_DIVISOR = 1
DIO_EF_CLOCK0_ROLL_VALUE = 8000
DIO_EF_CLOCK0_ENABLE = 1
```

Once the clock source is configured, we can use the roll value to calculate CONFIG_A:

```
DC = 25% = 100 * CONFIG_A /
8000
```

```
CONFIG_A = 25 * 8000 / 100 =
2000
```

Now the PWM can be turned on by writing to the proper registers (use DIO4 on the T4):

```
DIO0_EF_ENABLE = 0
DIO0_EF_INDEX = 0
DIO0_EF_CONFIG_A = 2000
DIO0_EF_ENABLE = 1
```

T8 Pseudocode

With a divisor of 1, the roll value will be 10000:

```
100 MHz / (1 * 10000) = 10
kHz
```

Set the clock registers accordingly:

```
DIO_EF_CLOCK0_ENABLE = 0
DIO_EF_CLOCK0_DIVISOR = 1
DIO_EF_CLOCK0_ROLL_VALUE = 10000
DIO_EF_CLOCK0_ENABLE = 1
```

Once the clock source is configured, we can use the roll value to calculate CONFIG_A:

```
DC = 25% = 100 * CONFIG_A /
10000
```

```
CONFIG_A = 25 * 10000 / 100 =
2500
```

Now the PWM can be turned on by writing to the proper registers:

```
DIO0_EF_ENABLE = 0
DIO0_EF_INDEX = 0
DIO0_EF_CONFIG_A = 2500
DIO0_EF_ENABLE = 1
```

13.2.3 PWM Out with Phase [T-Series Datasheet]

Overview

T4 Capable DIO: **DIO6, DIO7** (aka FIO6, FIO7)

T7 Capable DIO: **DIO0, DIO2, DIO3, DIO4, DIO5** (aka FIO0, FIO2, FIO3, FIO4, FIO5)

T8 Capable DIO: **DIO2, DIO3, DIO4, DIO5, DIO6, DIO7, DIO8, DIO11, DIO12** (aka FIO2, FIO3, FIO4, FIO5, FIO6, FIO7, EIO0, EIO3, EIO4)

Requires Clock Source: **Yes**

Index: **1**

Streamable: **No**

This PWM Out with Phase Extended Feature is similar to the PWM Out DIO-EF, but allows for phase control.

Operation

PWM Output with Phase control generates PWM waveforms with the pulse positioned at different points in the period. This is achieved by setting the DIO line high and low relative to the clock source's count.

```
Clock#Frequency = CoreFrequency /
DIO_EF_CLOCK#_DIVISOR
PWMFrequency = Clock#Frequency / DIO_EF_CLOCK#_ROLL_VALUE
DutyCycle% = 100 * (DIO#_EF_CONFIG_A - DIO#_EF_CONFIG_B) /
DIO_EF_CLOCK#_ROLL_VALUE
PhaseOffset = 360º * DIO#_EF_CONFIG_A / DIO_EF_CLOCK#_ROLL_VALUE
```

When the count matches CONFIG_B, the DIO line will be set high. When the count matches CONFIG_A, the line will be set low. Therefore CONFIG_B minus CONFIG_A controls the duty cycle.

See the DIO-EF Clock Source section for more information about your device core frequency and DIO_EF clock source settings.

DIO_EF_CLOCK#_ROLL_VALUE is a 32-bit value for CLOCK0 and a 16-bit value for CLOCK1 & CLOCK2. A value of 0 corresponds to the max roll value of 2^{32} for the 32-bit clock or 2^{16} for 16-bit clocks.

The clock roll value can take up to one PWM period to update and this does not block subsequent commands from being processed. It is possible to finish updating DIO_EF_CONFIG_A or DIO_CONFIG_B to a value greater than the non-updated clock roll value (which is invalid) but less than the updated clock roll value (which is valid) and throw the error 2565: EF_VALUE_GREATER_THAN_PERIOD.

Potential fixes:

- disable and re-enable the clock line before updating the clock roll value and duty cycle value.
- Delay for greater than one "non-updated" PWM period between updating the clock roll value and updating the duty cycle value

Configure

DIO#_EF_ENABLE: 0 = Disable, 1 = Enable
DIO#_EF_INDEX: 1
DIO#_EF_OPTIONS: Bits 0-2 specify which clock source to use ... 000 for Clock0, 001 for Clock1, and 010 for Clock2. All other bits reserved and should be set to 0.
DIO#_EF_CONFIG_A: When the clock source's count matches this value the line will transition from high to low.
DIO#_EF_CONFIG_B: When the clock source's count matches this value the line will transition from low to high.
DIO#_EF_CONFIG_C: Not used.
DIO#_EF_CONFIG_D: Not used.

Update

The duty cycle can be updated at any time. To update, write the new value to CONFIG_A then CONFIG_B. The value written to CONFIG_A is stored until CONFIG_B is written. After writing CONFIG_B, the new value will be loaded at the start of the next period. Updates are glitch-less unless switching from a very high to very low duty cycle or a very low to very high duty cycle.

DIO#_EF_CONFIG_A: Values written here will set the new falling position. The new value will not take effect until CONFIG_B is written.
DIO#_EF_CONFIG_B: Values written here will set the new rising position. When CONFIG_B is written, the new CONFIG_A is also loaded.

Read

No information is returned by PWM Out with Phase.

Reset

Reset has no affect on this feature.

Example

See the PWM Out documentation for an example.

13.2.4 Pulse Out [T-Series Datasheet]

Overview

T4 Capable DIO: **DIO6, DIO7** (aka FIO6, FIO7)

T7 Capable DIO: **DIO0, DIO2, DIO3, DIO4, DIO5** (aka FIO0, FIO2, FIO3, FIO4, FIO5)

T8 Capable DIO: **DIO2, DIO3, DIO4, DIO5, DIO6, DIO7, DIO8, DIO11, DIO12** (aka FIO2, FIO3, FIO4, FIO5, FIO6, FIO7, EIO0, EIO3, EIO4)

Requires Clock Source: **Yes**

Index: **2**

Streamable: **No**

Pulse output will generate a specified number of pulses and then stop. For continuous pulse output use the PWM Out feature.

Operation

The high time and the low time are specified relative to the clock source the same way as PWM with Phase.

```
Clock#Frequency = CoreFrequency /
DIO_EF_CLOCK#_DIVISOR
PulseOutFrequency = Clock#Frequency /
DIO_EF_CLOCK#_ROLL_VALUE
```

if CONFIG_A > CONFIG_B:

```
DutyCycle% = 100 * (DIO#_EF_CONFIG_A - DIO#_EF_CONFIG_B) /
DIO_EF_CLOCK#_ROLL_VALUE
```

For the common case that CONFIG_B is fixed at 0:

```
DutyCycle% = 100 * DIO#_EF_CONFIG_A /
DIO_EF_CLOCK#_ROLL_VALUE
```

For a 50% duty cycle:

```
DIO#_EF_CONFIG_A = DIO_EF_CLOCK#_ROLL_VALUE /
2
```

See the DIO-EF Clock Source section for more information about your device core frequency and DIO_EF clock source settings.

DIO_EF_CLOCK#_ROLL_VALUE is a 32-bit value for CLOCK0 and a 16-bit value for CLOCK1 & CLOCK2. A value of 0 corresponds to the max roll value of 2^{32} for the 32-bit clock or 2^{16} for 16-bit clocks.

The clock roll value can take up to one pulse out period to update and this does not block subsequent commands from being processed. It is possible to finish updating DIO_EF_CONFIG_A or DIO_EF_CONFIG_B to a value greater than the non-updated clock roll value (which is invalid) but less than the updated clock roll value (which is valid) and throw the error 2565: EF_VALUE_GREATER_THAN_PERIOD.

Potential fixes:

- disable and re-enable the clock line before updating the clock roll value and line transition values.
- Delay for greater than one "non-updated" period between updating the clock roll value and updating the line transition values.

Configure

DIO#: First set the DIO line low (DIO#=0). The line must start low for proper pulse generation.
DIO#_EF_ENABLE: 0 = Disable, 1 = Enable
DIO#_EF_INDEX: 2
DIO#_EF_OPTIONS: Bits 0-2 specify which clock source to use ... 000 for Clock0, 001 for Clock1, and 010 for Clock2. All other bits are reserved and should be set to 0.
DIO#_EF_CONFIG_A: When the specified clock source's count matches this value the line will transition from high to low.
DIO#_EF_CONFIG_B: When the specified clock source's count matches this value the line will transition from low to high.
DIO#_EF_CONFIG_C: The number of pulses to generate.
DIO#_EF_CONFIG_D: Not used.

Update

DIO#_EF_CONFIG_A: Sets a new high to low transition point. Will take effect when writing CONFIG_C.
DIO#_EF_CONFIG_B: Sets a new low to high transition point. Will take effect when writing CONFIG_C.
DIO#_EF_CONFIG_C: Writing to this value will start a new pulse sequence. If a sequence is already in progress it will be aborted. Numbers previously written to CONFIG_A or CONFIG_B will take effect when CONFIG_C is written.

Read

Results are read from the following registers.

DIO#_EF_READ_A: The number of pulses that have been completed.
DIO#_EF_READ_B: The target number of pulses.

Reset

DIO#_EF_READ_A_AND_RESET: Reads number of pulses that have been completed, then restarts the pulse sequence.

Example

Set up a 1 kHz output with 20% duty cycle.

T4/T7 Pseudocode

First, configure the DIO_EF clock source:

```
DIO_EF_CLOCK0_ENABLE = 0
DIO_EF_CLOCK0_DIVISOR = 8
DIO_EF_CLOCK0_ROLL_VALUE = 10000
DIO_EF_CLOCK0_ENABLE = 1
```

```
Clock0Frequency = 80 MHz / 8 = 10
MHz
```

```
PWMFrequency = 10 MHz / 10000 = 1
kHz
```

Once the clock source is configured, you can configure the pulse output.

Disable the feature:

```
DIO0_EF_ENABLE = 0
```

Set DIO0 to output low (use DIO4 on the T4):

```
DIO0 = 0
```

User the pulse out index:

```
DIO0_EF_INDEX = 2
```

Configure high to low counts:

```
DIO0_EF_CONFIG_A = 2000
```

```
duty cycle = 100 * (2000 - 0) / 10000 =
20%
```

Configure low to high counts:

```
DIO0_EF_CONFIG_B = 0
```

Configure the number of pulses:

```
DIO0_EF_CONFIG_C = 5000
```

```
pulseRuntime = 5000 pulses / 1000 pulses/second = 5
seconds
```

Enable the feature:

```
DIO0_EF_ENABLE = 1
```

T8 Pseudocode

First, configure the DIO_EF clock source:

```
DIO_EF_CLOCK0_ENABLE = 0
DIO_EF_CLOCK0_DIVISOR = 16
DIO_EF_CLOCK0_ROLL_VALUE = 6250
DIO_EF_CLOCK0_ENABLE = 1
```

```
Clock0Frequency = 100 MHz / 16 = 6.25
MHz
```

```
PWMFrequency = 6.25 MHz / 6250 = 1
kHz
```

Once the clock source is configured, you can configure the pulse output.

Disable the feature:

```
DIO0_EF_ENABLE = 0
```

Set DIO0 to output low:

```
DIO0 = 0
```

User the pulse out index:

```
DIO0_EF_INDEX = 2
```

Configure high to low counts:

```
DIO0_EF_CONFIG_A = 1250
```

```
duty cycle = 100 * (1250 - 0) / 6250 =
20%
```

Configure low to high counts:

```
DIO0_EF_CONFIG_B = 0
```

Configure the number of pulses:

```
DIO0_EF_CONFIG_C = 5000
```

```
pulseRuntime = 5000 pulses / 1000 pulses/second = 5
seconds
```

Enable the feature:

`DIO0_EF_ENABLE = 1`

13.2.5 Frequency In [T-Series Datasheet]

Overview

T4 Capable DIO: **DIO4, DIO5** (aka FIO4, FIO5)

T7 Capable DIO: **DIO0, DIO1** (aka FIO0, FIO1)

T8 Capable DIO: **DIO0, DIO1, DIO2, DIO3, DIO4, DIO5, DIO8, DIO11, DIO13** (aka FIO0, FIO1, FIO2, FIO3, FIO4, FIO5, EIO0, EIO3, EIO5)

Requires Clock Source: **Yes**

Index: **3 (positive edges) or 4 (negative edges)**

Streamable: **Yes—integer READ registers only.**

Frequency In will measure the period or frequency of a digital input signal by counting the number of clock source ticks between two edges: rising-to-rising (index=3) or falling-to-falling (index=4).

Operation

The number of ticks can be read from DIO#_EF_READ_A.

```
Clock#Frequency = CoreFrequency /
DIO_EF_CLOCK#_DIVISOR
Period (s) = DIO#_EF_READ_A / Clock#Frequency
Frequency (Hz) = Clock#Frequency / DIO#_EF_READ_A
Resolution(s) = 1 / Clock#Frequency
Max Period(s) = DIO_EF_CLOCK#_ROLL_VALUE /
Clock#Frequency
```

Frequency In works best with periodic signals. Due to a hardware constraint, the first edge is often missed. That can cause confusing results when the signal is not periodic. For measuring non-perioidic singles we recommend L2L.

See the DIO-EF Clock Source section for more information about your device core frequency and DIO_EF clock source settings.

Roll value for this feature would typically be left at the default of 0, which is the max value (2^32 for the 32-bit Clock0), but you might be forced to use a lower roll value due to another needed feature such as PWM Out.

A couple of typical scenarios with roll value = 0 and using the 32-bit clock (Clock0):

- Divisor = 1, Resolution = 12.5 nanoseconds, MaxPeriod = 53.7 seconds
- Divisor = 256, Resolution = 3.2 microseconds, MaxPeriod = 229 minutes

Only Clock0 is 32-bit. Clock1 and Clock2 are both 16-bit. Usage of Clock1 or Clock2 would decrease the maximum roll value to 2^{16}.

If you are also using the PWM Out feature, note that the PWM output frequency cannot be measured using the frequency in feature when using the same DIO_EF clock. Consider using the Interrupt Frequency Input feature for measurement instead.

By default, Frequency In operates in one-shot mode where it will measure the frequency once after being enabled and a new measurement only once after each read of a READ_A register. The other option is continuous mode, where the frequency is constantly measured (every edge is processed) and READ registers return the most recent result. Running in continuous mode puts a greater load on the processor.

If you do another read before a new edge has occurred, you will get the same value as before. Some applications will want to use the read-and-reset option so that a value is only returned once and extra reads will return 0. (See Reset below.)

Configure

DIO#_EF_ENABLE: 0 = Disable, 1 = Enable
DIO#_EF_INDEX: 3 or 4
DIO#_EF_OPTIONS: Default = 0. Bits 0-2 specify which clock source to use ... 000 for Clock0, 001 for Clock1, and 010 for Clock2. All other bits reserved and should be set to 0.
DIO#_EF_CONFIG_A: Default = 0. Bit 1: 0 = one-shot, 1 = continuous. All other bits reserved.
DIO#_EF_CONFIG_B: Not used.
DIO#_EF_CONFIG_C: Not used.
DIO#_EF_CONFIG_D: Not used.

Update

No update operations can be performed on Frequency In.

Read

Results are read from the following registers.

DIO#_EF_READ_A: Returns the period in ticks. If a full period has not yet been observed this value will be zero.
DIO#_EF_READ_B: Returns the same value as READ_A.

DIO#_EF_READ_A_F: Returns the period in seconds. If a full period has not yet been observed this value will be zero.
DIO#_EF_READ_B_F: Returns the frequency in Hz. If a full period has not yet been observed this value will be zero.

Note that all "READ_B" registers are capture registers. All "READ_B" registers are only updated when any "READ_A" register is read. Thus it would be unusual to read any B registers without first reading at least one A register.

Stream Read

All operations discussed in this section are supported in command-response mode.
 In stream mode, you can read from the integer READ registers (A, B, A_AND_RESET), but as mentioned in the Stream Section those reads only return the lower 16 bits so you need to also use STREAM_DATA_CAPTURE_16 in the scan list to get the upper 16 bits.

Reset

DIO#_EF_READ_A_AND_RESET: Returns the same data as DIO#_EF_READ_A and then clears the result so that zero is returned by subsequent reads until another full period is measured (2 new edges).

Note that even in continuous mode, with reads happening faster than the signal frequency using a _RESET read will result in measurements of every other cycle not every cycle.

Example

Most applications can use default clock settings, so to configure frequency input on DIO0 you can simply write to 3 registers:

```
DIO_EF_CLOCK0_ENABLE = 1
DIO0_EF_ENABLE = 0
DIO0_EF_INDEX = 3
DIO0_EF_ENABLE = 1
```

Now you can read the period in seconds from a 4th register DIO0_EF_READ_A_F.

Roll Value Considerations

Sometimes, other DIO-EF might interact with this feature. For example, roll value would usually be set to 0 to provide the maximum measurable period, but assume that we have to use 10000 because it is set to that for PWM output on another channel:

T4/T7:

```
DIO_EF_CLOCK0_DIVISOR = 8 // Divisor used for PWM. 80 MHz / 8 = 10 MHz clock
DIO_EF_CLOCK0_ROLL_VALUE = 10000 // Roll value used for PWM
```

T8:

```
DIO_EF_CLOCK0_DIVISOR = 16 // Divisor used for PWM. 100 MHz / 16 = 6.25 MHz clock
```

```
DIO_EF_CLOCK0_ROLL_VALUE = 10000 // Roll value used for PWM
```

This clock configuration results in:

```
Resolution = 1 / 6.25 MHz = 0.16 us
```

```
MaxPeriod = 10000 / 6.25 MHz = 1.6 ms
```

For a more detailed walkthrough, see Configuring & Reading Frequency.

Rate Limits

The maximum measurable frequency varies based on the one-shot setting, concurrent stream rate, and other DIO-EFs set to an input mode.

One-shot or Continuous

When one-shot is enabled, the T-series devices will take a single measurement, then wait for a READ_A register to be read before taking another measurement. This means that with one-shot, only a small fraction of the total periods of a signal are measured. One-shot allows for higher maximum measurable frequency than continuous does.

Stream

The stream process is the highest priority process on T-series devices. When stream needs the processor, all other operations are put on hold. That hold occurs more frequently at higher stream speeds. When a DIO-EF process has to wait, the max frequency that can be measured is reduced.

Multiple DIO-EFs

DIO-EFs on other different lines also require processor time. The amount of processor time required depends on the signal being processed and the DIO-EF's settings. The maximum frequencies in the below table give the total max frequency for all running DIO-EFs—divide the max frequencies below by the number of enabled DIO-EFs to get the max frequency for each individual DIO-EF.

Refer to the following table for maximum measurable frequencies in various combinations of stream and one-shot.

Index	One-shot or Continuous	Stream Rate	Max Frequency
3 or 4	Continuous	Stream not running	200 kHz
3 or 4	One-shot	Stream not running	750 kHz
3 or 4	Continuous	10 kHz	75 kHz
3 or 4	One-shot	10 kHz	750 kHz
3 or 4	Continuous	100 kHz	20 kHz
3 or 4	One-shot	100 kHz	250 kHz
5	Continuous	Stream not running	200 kHz
5	One-shot	Stream not running	750 kHz
5	Continuous	10 kHz	75 kHz
5	One-shot	10 kHz	750 kHz
5	Continuous	100 kHz	20 kHz
5	One-shot	100 kHz	250 kHz

13.2.6 Pulse Width In [T-Series Datasheet]

Overview

T4 Capable DIO: **DIO4, DIO5** (aka FIO4, FIO5)

T7 Capable DIO: **DIO0, DIO1** (aka FIO0, FIO1)

T8 Capable DIO: **DIO0, DIO1, DIO2, DIO3, DIO4, DIO5, DIO8, DIO11, DIO13** (aka FIO0, FIO1, FIO2, FIO3, FIO4, FIO5, EIO0, EIO3, EIO5)

Requires Clock Source: **Yes**

Index: **5**

Streamable: **Yes—integer READ registers only.**

Pulse Width In will measure the high time and low time of a digital input signal, by counting the number of clock source ticks while the signal is high and low. This could also be referred to as duty-cycle input or PWM input.

Operation

The number of high ticks can be read from DIO#_EF_READ_A and the number of low ticks can be read from DIO#_EF_READ_B.

```
Clock#Frequency = CoreFrequency / DIO_EF_CLOCK#_DIVISOR
HighTime(s) = DIO#_EF_READ_A / Clock#Frequency
LowTime(s) = DIO#_EF_READ_B / Clock#Frequency
Resolution(s) = 1 / Clock#Frequency
Max High or Low Time(s) = DIO_EF_CLOCK#_ROLL_VALUE /
Clock#Frequency
```

See the DIO-EF Clock Source section for more information about your device core frequency and DIO_EF clock source settings.

Roll value for this feature would typically be left at the default of 0, which is the max value (2^{32} for the 32-bit Clock0), but you might be using a lower roll value for another feature such as PWM Out.

A couple typical scenarios with roll value = 0 and using the 32-bit clock (Clock0):

- Divisor = 1, Resolution = 12.5 nanoseconds, MaxTime = 53.7 seconds
- Divisor = 256, Resolution = 3.2 microseconds, MaxTime = 229 minutes

Only Clock0 is 32-bit. Clock1 and Clock2 are both 16-bit. Usage of Clock1 or Clock2 would decrease the maximum roll value to 2^{16}.

If you are also using the PWM Out feature, note that the PWM output frequency cannot be measured using the frequency in feature when using the same DIO_EF clock.

Once this feature is enabled, a new measurement happens on every applicable edge and both result registers are updated on every rising edge. If you do another read before a new rising edge has occurred, you will get the same values as before. Many applications will want to use the read-and-reset option so that a value is only read once and extra reads will return 0. (See Reset below.)

Configure

DIO#_EF_ENABLE: 0 = Disable, 1 = Enable
DIO#_EF_INDEX: 5
DIO#_EF_OPTIONS: Bits 0-2 specify which clock source to use ... 000 for Clock0, 001 for Clock1, and 010 for Clock2. All other bits reserved and should be set to 0.
DIO#_EF_CONFIG_A: Bit 1: 1=continuous, 0=one-shot. All other bits reserved.
DIO#_EF_CONFIG_B: Not used.
DIO#_EF_CONFIG_C: Not used.
DIO#_EF_CONFIG_D: Not used.

Update

No update operations can be performed on Pulse Width In.

Read

Results are read from the following registers.

DIO#_EF_READ_A: Returns the measured high time in clock source ticks and saves the low time so that it can be read later. If a full period has not yet been observed this value will be zero.
DIO#_EF_READ_B: Returns the measured low time in clock source ticks. This is a capture register ... it is only updated when one of the READ_A registers is read.
DIO#_EF_READ_A_F: Returns the measured high time in seconds and saves the low time so that it can be read later. If a full period has not yet been observed this value will be zero.
DIO#_EF_READ_B_F: Returns the measured low time in seconds. This is a capture register ... it

is only updated when one of the READ_A registers is read.

Only reading one of the "READ_A" registers will trigger a new measurement. All "READ_B" registers are capture registers, and they are only updated when any "READ_A" register is read.

Stream Read

All operations discussed in this section are supported in command-response mode.
In stream mode, you can read from the integer READ registers (A, B, A_AND_RESET), but as mentioned in the Stream Section those reads only return the lower 16 bits so you need to also use STREAM_DATA_CAPTURE_16 in the scan list to get the upper 16 bits.

Reset

DIO#_EF_READ_A_AND_RESET: Performs the same read as READ_A, but then also clears the register so that zero is returned until another full period is measured.

DIO#_EF_READ_A_F_AND_RESET: Performs the same read as READ_A_F, but then also clears the register so that zero is returned until another full period is measured.

Example

First, configure the clock source. Roll value would usually be set to 0 to provide the maximum measurable period, but assume for this example that we have to use 10000 because of PWM output on another channel.

T4/T7

```
DIO_EF_CLOCK0_DIVISOR = 8 // Divisor used for PWM. 80 MHz / 8 = 10 MHz clock
```

```
DIO_EF_CLOCK0_ROLL_VALUE = 10000 // Roll value used for PWM
```

This clock configuration results in:

```
Resolution = 1 / 10 MHz = 0.1 us
```

```
MaxPeriod = 10000 / 10 MHz = 1 ms
```

Now configure the DIO_EF on DIO0 as pulse width input (use DIO4 on the T4):

```
DIO0_EF_ENABLE = 0
DIO0_EF_INDEX = 5    // Pulse width input feature.
DIO0_EF_OPTIONS = 0  // Set to use clock source
zero.
DIO0_EF_ENABLE = 1   // Enable the DIO_EF
```

After a full period (rising edge, falling edge, rising edge) has occurred, the values are stored in the result registers. This repeats at each rising edge. READ_A and READ_A_F both return the high time and save the low time that can be read from READ_B and READ_B_F. This ensures that both readings are from the same waveform cycle.

T8

```
DIO_EF_CLOCK0_DIVISOR = 16 // Divisor used for PWM. 100 MHz / 16 = 6.25 MHz
clock
```

```
DIO_EF_CLOCK0_ROLL_VALUE = 10000 // Roll value used for
PWM
```

This clock configuration results in:

```
Resolution = 1 / 6.25 MHz = 0.16
us
```

```
MaxPeriod = 10000 / 6.25 MHz = 1.6
ms
```

Now configure the DIO_EF on DIO0 as pulse width input:

```
DIO0_EF_ENABLE = 0
DIO0_EF_INDEX = 5    // Pulse width input feature.
DIO0_EF_OPTIONS = 0  // Set to use clock source
zero.
DIO0_EF_ENABLE = 1   // Enable the DIO_EF
```

After a full period (rising edge, falling edge, rising edge) has occurred, the values are stored in the result registers. This repeats at each rising edge. READ_A and READ_A_F both return the high time and save the low time that can be read from READ_B and READ_B_F. This ensures that both readings are from the same waveform cycle.

Rate Limits

The maximum measurable frequency varies based on the one-shot setting, concurrent stream rate, and other DIO-EFs set to an input mode.

One-shot or Continuous

When one-shot is enabled, the T-series devices will take a single measurement, then wait for a READ_A register to be read before taking another measurement. This means that with one-shot, only a small fraction of the total periods of a signal are measured. One-shot allows for higher maximum measurable frequency than continuous does.

Stream

The stream process is the highest priority process on T-series devices. When stream needs the processor, all other operations are put on hold. That hold occurs more frequently at higher stream speeds. When a DIO-EF process has to wait, the max frequency that can be measured is reduced.

Multiple DIO-EFs

DIO-EFs on other different lines also require processor time. The amount of processor time required depends on the signal being processed and the DIO-EF's settings. The maximum frequencies in the below table give the total max frequency for all running DIO-EFs—divide the max frequencies below by the number of enabled DIO-EFs to get the max frequency for each individual DIO-EF.

Refer to the following table for maximum measurable frequencies in various combinations of stream and one-shot.

Index	One-shot or Continuous	Stream Rate	Max Frequency
3 or 4	Continuous	Stream not running	200 kHz
3 or 4	One-shot	Stream not running	750 kHz
3 or 4	Continuous	10 kHz	75 kHz
3 or 4	One-shot	10 kHz	750 kHz
3 or 4	Continuous	100 kHz	20 kHz
3 or 4	One-shot	100 kHz	250 kHz
5	Continuous	Stream not running	200 kHz
5	One-shot	Stream not running	750 kHz
5	Continuous	10 kHz	75 kHz
5	One-shot	10 kHz	750 kHz
5	Continuous	100 kHz	20 kHz
5	One-shot	100 kHz	250 kHz

13.2.7 Line-to-Line In [T-Series Datasheet]

Overview

T4 Capable DIO: **DIO4, DIO5** (aka FIO4, FIO5)

T7 Capable DIO: **DIO0, DIO1** (aka FIO0, FIO1)

T8 Capable DIO: **DIO0, DIO1, DIO2, DIO3, DIO4, DIO5, DIO8, DIO11, DIO13** (aka FIO0, FIO1, FIO2, FIO3, FIO4, FIO5, EIO0, EIO3, EIO5)

Requires Clock Source: **Yes**

Index: **6**

Streamable: **No**

Line-to-line mode can be used to measure the time between edge transition points on two supported DIO. The measurement occurs between the lower DIO number edge and the upper DIO edge. For example, the time between a falling edge on DIO0 and the next falling edge on DIO1 when using a T7.

Operation

```
Clock#Frequency = CoreFrequency /
DIO_EF_CLOCK#_DIVISOR
Time(s) = DIO#_EF_READ_A / Clock#Frequency
Resolution(s) = 1 / Clock#Frequency
Max Time(s) = DIO_EF_CLOCK#_ROLL_VALUE /
Clock#Frequency
```

See the DIO-EF Clock Source section for more information about your device core frequency and DIO_EF clock source settings.

Roll value for this feature would typically be left at the default of 0, which is the max value (2^32 for the 32-bit Clock0), but you might be using a lower roll value for another feature such as PWM Out.

A couple typical scenarios with roll value = 0 and using the 32-bit clock (Clock0):

- Divisor = 1, Resolution = 12.5 nanoseconds, MaxPeriod = 53.7 seconds
- Divisor = 256, Resolution = 3.2 microseconds, MaxPeriod = 229 minutes

Line-to-Line In operates in a one-shot mode. Once the specified combination of edges is observed, the data is saved and measuring stops. Another measurement can be started by resetting or performing the configuration procedure again.

Configure

Configuring Line-to-Line In requires configuring two digital I/O lines (DIO0 and DIO1 only) as Line-to-Line In feature index 6. The first DIO configured should be the one expecting the first edge. Any extended features on either DIO should be disabled before beginning configuration.

DIO#_EF_ENABLE: 0 = Disable, 1 = Enable
DIO#_EF_INDEX: 6
DIO#_EF_OPTIONS: Bits 0-2 specify which clock source to use ... 000 for Clock0, 001 for Clock1, and 010 for Clock2. All other bits are reserved and should be set to 0.
DIO#_EF_CONFIG_A: 0 = falling edge. 1 = rising edge.
DIO#_EF_CONFIG_B: Not used.
DIO#_EF_CONFIG_C: Not used.
DIO#_EF_CONFIG_D: Not used.

Update

No update operations can be performed on Line-to-Line In.

Read

Results are read from the following registers.

DIO#_EF_READ_A: Returns the one-shot measured time in clock source ticks. If the specified combination of edges has not yet been observed this value will be zero. DIO0 & DIO1 both return the same value.
DIO#_EF_READ_A_F: Returns the time in seconds.

Reset

DIO#_EF_READ_A_AND_RESET: Performs the same operation as DIO#_EF_READ_A, then clears the result and starts another measurement.

Example

First, configure the clock source:

```
DIO_EF_CLOCK0_DIVISOR = 1
DIO_EF_CLOCK0_ROLL_VALUE = 0
DIO_EF_CLOCK0_ENABLE = 1
```

On the T4 and T7 this clock configuration results in:

```
Resolution = 1 / 80 MHz = 12.5
ns
```

```
MaxPeriod = 2^32 / 80 MHz = 53.7 seconds
```

On the T8 this clock configuration results in:

```
Resolution = 1 / 100 MHz = 10
ns
```

```
MaxPeriod = 2^32 / 100 MHz = 42.9 seconds
```

Now configure the DIO-EF on DIO0 and DIO1 as line-to-line (use DIO4 and DIO5 on the T4):

```
DIO0_EF_ENABLE = 0
DIO1_EF_ENABLE = 0

DIO0_EF_INDEX = 6     // Index for line-to-line
feature.
DIO0_EF_OPTIONS = 0   // Select the clock source.
DIO0_EF_CONFIG_A = 0  // Detect falling edge.
DIO0_EF_ENABLE = 1    // Turn on the DIO-EF

DIO1_EF_INDEX = 6     // Index for line-to-line
feature.
DIO1_EF_OPTIONS = 0   // Select the clock source.
DIO1_EF_CONFIG_A = 0  // Detect falling edge.
DIO1_EF_ENABLE = 1    // Turn on the DIO-EF
```

At this point the device is watching DIO0 for a falling edge. Once that happens it watches for a falling edge on DIO1. Once that happens it stores the time between those 2 edges, which you can read from the READ registers described above.

To do another measurement, repeat the DIO0-EF configuration above, or read from DIO0_EF_READ_A_AND_RESET.

Rate Limits

Line-to-line can achieve high resolution while requiring very little processor time. The time between the edges can be as little as 50 ns. Once a measurement has been completed the system will not measure again until reconfigured or reset. Unless you are reading at a high rate, line-to-line will have little impact on other systems.

13.2.8 High-Speed Counter [T-Series Datasheet]

Overview

T4/T7 Capable DIO: **DIO16, DIO17, DIO18, DIO19** (aka CIO0, CIO1, CIO2, CIO3)

T8 Capable DIO: **DIO6, DIO7, DIO8, DIO10, DIO13, DIO14, DIO15** (aka FIO6, FIO7, EIO0, EIO2, EIO5, EIO6, EIO7)

Requires Clock Source: **No**

Index: **7**

Streamable: **Yes—integer READ registers only.**

T4 and T7 devices support up to 4 high-speed rising-edge counters that use hardware to achieve high count rates. These counters are shared with other resources as follows:

- CounterA (DIO16/CIO0): Used by EF Clock0 & Clock1.
- CounterB (DIO17/CIO1): Used by EF Clock0 & Clock2.
- CounterC (DIO18/CIO2): Always available.
- CounterD (DIO19/CIO3): Used by stream mode.

The T8 supports up to 7 high-speed rising-edge counters that use hardware to achieve high count rates. These counters are shared with other resources as follows:

- DIO14/EIO6: Used by EF Clock0 & Clock1.
- DIO15/EIO7: Used by EF Clock0 & Clock2.

Configure

DIO#_EF_ENABLE: 0 = Disable, 1 = Enable
DIO#_EF_INDEX: 7
DIO#_EF_OPTIONS: Not used.
DIO#_EF_CONFIG_A: Not used.
DIO#_EF_CONFIG_B: Not used.
DIO#_EF_CONFIG_B: Not used.
DIO#_EF_CONFIG_B: Not used.

Update

No update operations can be performed with High-Speed Counter.

Read

Results are read from the following register.

DIO#_EF_READ_A: Returns the current count which is incremented on each rising edge.

Stream Read

All operations discussed in this section are supported in command-response mode.
 In stream mode, you can read from the integer READ registers (A, B, A_AND_RESET), but as mentioned in the Stream Section those reads only return the lower 16 bits so you need to also use STREAM_DATA_CAPTURE_16 in the scan list to get the upper 16 bits.

Reset

DIO#_EF_READ_A_AND_RESET: Reads the current count then clears the counter. There is a brief period of time between reading and clearing during which edges can be missed. During normal operation this time period is 10-30 µs. If missed edges at this point are not acceptable, then do not use reset but rather just note the "virtual reset" counter value in software and subtract it from other values.

Frequency Measurement

Counters are often used to measure frequency by taking change in count over change in time:

```
Frequency = (CurrentCount - PreviousCount) / (CurrentTimestamp - PreviousTimestamp)
```

Typically the timestamps are from the host clock (software), but for more accurate timestamps read the CORE_TIMER register in the same Modbus packet as the READ registers. See the System Timing Register section for more information about the CORE_TIMER.

Also note that other digital extended features are available to measure frequency by timing individual pulses rather than counting over time.

Example

Enable a high speed counter on DIO18/CIO2 (use DIO6 for the T8):

```
DIO18_EF_ENABLE = 0
DIO18_EF_INDEX = 7
DIO18_EF_ENABLE = 1
```

Enable a high speed counter on DIO17/CIO1 (use DIO14 for the T8):

```
DIO_EF_CLOCK0_ENABLE = 0    //Make sure Clock0 is
disabled.
DIO_EF_CLOCK2_ENABLE = 0    //Make sure Clock2 is
disabled.
DIO17_EF_ENABLE = 0
DIO17_EF_INDEX = 7
DIO17_EF_ENABLE = 1
```

Results can be read from the READ registers defined above.

Edge Rate Limits

See Appendix A-2.

13.2.9 Interrupt Counter [T-Series Datasheet]

Overview

T4 Capable DIO: **DIO4, DIO5, DIO6, DIO7, DIO8, DIO9** (aka FIO4, FIO5, FIO6, FIO7, EIO0, EIO1)

T7 Capable DIO: **DIO0, DIO1, DIO2, DIO3, DIO6, DIO7** (aka FIO0, FIO1, FIO2, FIO3, FIO6, FIO7)

T8 Capable DIO: **DIO0, DIO1, DIO2, DIO3, DIO4, DIO5, DIO6, DIO7, DIO8, DIO9, DIO10, DIO11, DIO12, DIO13, DIO14, DIO15** (aka FIO0, FIO1, FIO2, FIO3, FIO4, FIO5, FIO6, FIO7, EIO0, EIO1, EIO2, EIO3, EIO4, EIO5, EIO6, EIO7)

Requires Clock Source: **No**

Index: **8**

Streamable: **Yes—integer READ registers only.**

Interrupt Counter counts the rising edge of pulses on the associated IO line. This interrupt-based digital I/O extended feature (DIO-EF) is not purely implemented in hardware, but rather firmware must service each edge. See the discussion of edge rate limits at the bottom of this page.

Configure

DIO#_EF_ENABLE: 0 = Disable, 1 = Enable
DIO#_EF_INDEX: 8
DIO#_EF_OPTIONS: Not used.
DIO#_EF_CONFIG_A: Not used.
DIO#_EF_CONFIG_B: Not used.
DIO#_EF_CONFIG_B: Not used.
DIO#_EF_CONFIG_B: Not used.

There are 3 basic techniques for device configuration:

1. Power-up defaults. Most registers related to I/O configuration are part of the IO Config system where their startup values can be defined by the user. This is easily done with the Power-up Defaults tab in Kipling.

2. Real time software configuration. Software can write any needed configuration values at the beginning of execution, but you might have to consider how to handle if the device later reboots during software execution. An advantage to this method is that a factory device will work out-of-the-box without requiring the user to first configure the device.

3. Startup script. A Lua script can be set to run at startup and can write any values to any registers.

Update

No update operations can be performed on Interrupt Counter.

Read

Results are read from the following register.

DIO#_EF_READ_A: Returns the current Count

Stream Read

All operations discussed in this section are supported in command-response mode.
In stream mode, you can read from the integer READ registers (A, B, A_AND_RESET), but as mentioned in the Stream Section those reads only return the lower 16 bits so you need to also use STREAM_DATA_CAPTURE_16 in the scan list to get the upper 16 bits.

Reset

DIO#_EF_READ_A_AND_RESET: Reads the current count then clears the counter. Note that there is a brief period of time between reading and clearing during which edges can be missed. During normal operation this time period is 10-30 µs. If missed edges at this point can not be tolerated then reset should not be used.

Frequency Measurement

Counters are often used to measure frequency by taking change in count over change in time:

Frequency = (CurrentCount - PreviousCount) / (CurrentTimestamp - PreviousTimestamp)

Typically the timestamps are from the host clock (software), but for more accurate timestamps read the CORE_TIMER register in the same Modbus packet as the READ registers. See the System Timing Register section for more information about the CORE_TIMER.

Also note that other digital extended features are available to measure frequency by timing individual pulses rather than counting over time.

Example

Enable a counter on DIO0 (use DIO4 on the T4):

```
DIO0_EF_ENABLE = 0
DIO0_EF_INDEX = 8
DIO0_EF_ENABLE = 1
```

Use the Register Matrix in Kipling to write those 2 registers above, and also add DIO0_EF_READ_A so you see its value. Now take a wire connected to a GND terminal and tap that wire to the inside-back of the DIO0 screw-terminal (labeled "FIO0"). DIO0_EF_READ_A should increment by 1 or more counts each time you tap the ground wire in DIO0.

As another test you can set DAC1_FREQUENCY_OUT_ENABLE = 1 to enable the 10 Hz test signal (T7 requires firmware 1.0234+). Jumper DAC1 to DIO0 and you should see DIO0_EF_READ_A increment by about 10 counts per second.

For a more detailed walkthrough, see Configuring & Reading a Counter.

Edge Rate Limits

This interrupt-based digital I/O extended feature (DIO-EF) is not purely implemented in hardware, but rather firmware must service each edge. This makes it substantially slower than other DIO-EF that are purely hardware-based. To avoid missed edges, the aggregate limit for edges seen by all interrupt-based DIO-EF is 70k edges/second. If stream mode is active, the limit is reduced to 20k edges/second. Excessive processor loading (e.g. a busy Lua script) can also reduce these limits. Note that interrupt features must process all edges, rising & falling, even if a given feature is configured to only look at one or the other. Additionally, note that intensive interrupt based features may limit the maximum streaming rates due to processor loading.

The more proper way to think of the edge limit, and understand error that could be introduced when using multiple interrupt-based DIO-EF, is to consider that the interrupt that processes an edge can take up to 14 µs to complete. When a particular channel sees an applicable edge, an IF (interrupt flag) is set for that channel that tells the processor it needs to run an ISR (interrupt service routine) for that channel. Once an ISR is started, it runs to completion and no other ISR can run until it is done (except that stream interrupts are higher priority and will preempt other interrupts). When an ISR completes, it clears the IF for that channel. So it is okay to have edges on multiple channels at the same time, as long as there is not another edge on any of those channels before enough time to process all the initial edges.

Say that channel A & B have an edge occur at the same time and an ISR starts to process the edge on channel A. If channel A has another edge during the first 14 µs, that edge will be lost. If channel B has another edge during the first 14 µs, the initial edge will be lost. If channel B has another edge during the second 14 µs (during the ISR for channel B), the new edge will be lost.

13.2.10 Interrupt Counter with Debounce [T-Series Datasheet]

Overview

T4 Capable DIO: **DIO4, DIO5, DIO6, DIO7, DIO8, DIO9** (aka FIO4, FIO5, FIO6, FIO7, EIO0, EIO1)

T7 Capable DIO: **DIO0, DIO1, DIO2, DIO3, DIO6, DIO7** (aka FIO0, FIO1, FIO2, FIO3, FIO6, FIO7)

T8 Capable DIO: **DIO0, DIO1, DIO2, DIO3, DIO4, DIO5, DIO6, DIO7, DIO8, DIO9, DIO10, DIO11, DIO12, DIO13, DIO14, DIO15** (aka FIO0, FIO1, FIO2, FIO3, FIO4, FIO5, FIO6, FIO7, EIO0, EIO1, EIO2, EIO3, EIO4, EIO5, EIO6, EIO7)

Requires Clock Source: **No**

Index: **9**

Streamable: **Yes—integer READ registers only.**

Interrupt Counter with Debounce will increment its count by 1 when it receives a rising edge, a falling edge, or any edge (2x counting). After seeing an applicable edge, any further edges will be ignored during the debounce time.

This interrupt-based digital I/O extended feature (DIO-EF) is not purely implemented in hardware, but rather firmware must service each edge. See the discussion of edge rate limits at the bottom of this page.

Debounce Modes (DIO#_EF_CONFIG_B)

The exact behavior of the counting/debouncing is controlled by an index value written to DIO#_EF_CONFIG_B.

- 0: Count falling, debounce all, self-restarting timeout.
- 1: Count rising, debounce all, self-restarting timeout.
- 2: Count & debounce all, self-restarting timeout.
- 3: Count & debounce falling, fixed timeout.
- 4: Count & debounce rising, fixed timeout.
- 5: Timeout starts on falling edge. During timeout, a rising edge cancels and a falling edge restarts the timeout. After timeout any edge causes a count.
- 6: Timeout starts on rising edge. During timeout, a falling edge cancels and a rising edge restarts the timeout. After timeout any edge causes a count.

Self-restarting timeout means that during timeout any edge will restart the timeout with the value specified with DIO#_EF_CONFIG_A.

Mode 0 is commonly used with a normally-open push-button switch that is actuated by a

person. We only want to count the push (falling edge), but expect bounce on the push & release (falling & rising edges) so need to debounce both.

Mode 4 might be used with some sort of device that outputs a fixed length positive pulse. For example, say a device provides a 1000 µs pulse, and there is always at least 5000 µs between pulses. Set the debounce timeout to 2000 µs so that the timeout period safely covers the entire pulse, but the timeout will for sure be done before another pulse can occur.

Modes 5 & 6 implement a requirement that the state of the line must remain low or high for some amount of time. For example, if you use mode 5 with a push-button switch and set DIO#_EF_CONFIG_A = 50000, that means that someone must push the switch and hold it down solidly for at least 50ms, and then the count will occur when they release the switch. An advantage to these modes is that they will ignore brief transient signals.

Configure

DIO#_EF_ENABLE: 0 = Disable, 1 = Enable
DIO#_EF_INDEX: 9
DIO#_EF_OPTIONS: Not used.
DIO#_EF_CONFIG_A: Debounce timeout in microseconds (µs, 0-1000000).
DIO#_EF_CONFIG_B: Debounce mode index.
DIO#_EF_CONFIG_B: Not used.
DIO#_EF_CONFIG_B: Not used.

Update

No update operations can be performed on Interrupt Counter with Debounce.

Read

Results are read from the following register.

DIO#_EF_READ_A: Returns the current Count

Stream Read

All operations discussed in this section are supported in command-response mode.
 In stream mode, you can read from the integer READ registers (A, B, A_AND_RESET), but as mentioned in the Stream Section those reads only return the lower 16 bits so you need to also use STREAM_DATA_CAPTURE_16 in the scan list to get the upper 16 bits.

Reset

DIO#_EF_READ_A_AND_RESET: Reads the current count then clears the counter. Note that there is a brief period of time between reading and clearing during which edges can be missed. During normal operation this time period is 10-30 µs. If missed edges at this point can not be tolerated then reset should not be used.

Example

Enable a debounce counter on DIO0 (use DIO4 on the T4):

```
DIO0_EF_ENABLE = 0
DIO0_EF_INDEX = 9
DIO0_EF_CONFIG_A = 20000   // 20 ms debounce time
DIO0_EF_CONFIG_B = 0       // count falling, debounce all, self-restarting
timeout
DIO0_EF_ENABLE = 1
```

Results can be read from the READ registers defined above.

Edge Rate Limits

This interrupt-based digital I/O extended feature (DIO-EF) is not purely implemented in hardware, but rather firmware must service each edge. This makes it substantially slower than other DIO-EF that are purely hardware-based. To avoid missed edges, the aggregate limit for edges seen by all interrupt-based DIO-EF is 70k edges/second. If stream mode is active, the limit is reduced to 20k edges/second. Excessive processor loading (e.g. a busy Lua script) can also reduce these limits. Note that interrupt features must process all edges, rising & falling, even if a given feature is configured to only look at one or the other. Additionally, note that intensive interrupt based features may limit the maximum streaming rates due to processor loading.

The more proper way to think of the edge limit, and understand error that could be introduced when using multiple interrupt-based DIO-EF, is to consider that the interrupt that processes an edge can take up to 14 µs to complete. When a particular channel sees an applicable edge, an IF (interrupt flag) is set for that channel that tells the processor it needs to run an ISR (interrupt service routine) for that channel. Once an ISR is started, it runs to completion and no other ISR can run until it is done (except that stream interrupts are higher priority and will preempt other interrupts). When an ISR completes, it clears the IF for that channel. So it is okay to have edges on multiple channels at the same time, as long as there is not another edge on any of those channels before enough time to process all the initial edges.

Say that channel A & B have an edge occur at the same time and an ISR starts to process the edge on channel A. If channel A has another edge during the first 14 µs, that edge will be lost. If channel B has another edge during the first 14 µs, the initial edge will be lost. If channel B has another edge during the second 14 µs (during the ISR for channel B), the new edge will be lost.

13.2.11 Quadrature In [T-Series Datasheet]

Overview

T4 Capable DIO: **DIO4, DIO5, DIO6, DIO7, DIO8, DIO9** (aka FIO4, FIO5, FIO6, FIO7, EIO0, EIO1)

T7 Capable DIO: **DIO0, DIO1, DIO2, DIO3, DIO6, DIO7** (aka FIO0, FIO1, FIO2, FIO3, FIO6, FIO7)

T8 Capable DIO: **DIO0, DIO1, DIO2, DIO3, DIO4, DIO5, DIO6, DIO7, DIO8, DIO9, DIO10, DIO11, DIO12, DIO13, DIO14, DIO15** (aka FIO0, FIO1, FIO2, FIO3, FIO4, FIO5, FIO6, FIO7, EIO0, EIO1, EIO2, EIO3, EIO4, EIO5, EIO6, EIO7)

Requires Clock Source: **No**

Index: **10**

Streamable: **Yes—integer READ registers only.**

Quadrature input uses two DIOs to track a quadrature signal. Quadrature is a directional count often used in various types of **rotary encoders** when you need to keep track of absolute position with forward & reverse movement. If you have movement in only one direction, or another way to know direction, you can simply use a normal counter with one phase of the encoder output.

T-series devices uses 4x quadrature decoding, meaning that every edge observed (rising & falling on both phases) will increment or decrement the count.

This interrupt-based digital I/O extended feature (DIO-EF) is not purely implemented in hardware, but rather firmware must service each edge. See the discussion of edge rate limits at the bottom of this page.

Quadrature is prone to error if the edge rate is exceeded. This is particularly likely during direction change where the time between edges can be very small. Errors can occur when two edges come in too quickly for the device to process, which can result in missed counts or missed change in direction. These errors will be recorded and the quantity encountered can be read. If three edges come in too quickly an undetectable error can occur.

Some quadrature encoders also include a third output channel, called a zero (Z-phase) or index or reference signal, which supplies a single pulse per revolution. This single pulse is used for absolute determination of position. T-series devices support resets according to this reference signal. Z-phase will reset the count when a high state is detected on the specified DIO line at the same time any edge occurs on A or B phases. If the reference pulse is wider than A/B pulses, consider using the Conditional Reset feature instead of this Z-phase support. If the reference pulse is only high in between A/B edges, consider some sort of RC circuit to elongate it or consider using the Conditional Reset feature. If set to one-shot mode, Z-phase will clear the count only once and must be "re-armed" by disabling/re-enabling the feature or a read from DIO#_EF_READ_A_AND_RESET.

Configure

Two DIO must be configured for Quadrature In. The two lines must be an adjacent even/odd pair:

T4: DIO4/5, DIO6/7, and DIO8/9 are valid pairs.

T7: DIO0/1, DIO2/3, and DIO6/7 are valid pairs.

The even IO line will be phase A and the odd will be phase B.

DIO#_EF_ENABLE: 0 = Disable, 1 = Enable
DIO#_EF_INDEX: 10
DIO#_EF_OPTIONS: Not used.
DIO#_EF_CONFIG_A: Z-phase control: 0 = Z-phase off, 1 = Z-phase on, 3 = Z-phase on in one shot mode.
DIO#_EF_CONFIG_B: Z-phase DIO number.
DIO#_EF_CONFIG_C: Not used.
DIO#_EF_CONFIG_D: Not used.

Update

No update operations can be performed with Quadrature In.

Read

The quadrature count and error count are available from the even channel. The odd channel will return zeros.

Results are read from the following registers.

DIO#_EF_READ_A - Returns the current count as a signed 2's complement value.
DIO#_EF_READ_B – Returns the number of detected errors.
DIO#_EF_READ_A_F - Returns the count in a single precision float (float32).

Only reading DIO#_EF_READ_A or DIO#_EF_READ_A_F triggers a new measurement.

Stream Read

All operations discussed in this section are supported in command-response mode.
In stream mode, you can read from the integer READ registers (A, B, A_AND_RESET), but as mentioned in the Stream Section those reads only return the lower 16 bits so you need to also use STREAM_DATA_CAPTURE_16 in the scan list to get the upper 16 bits.

Reset

DIO#_EF_READ_A_AND_RESET – Performs the same operation as DIO#_EF_READ_A, then sets the count to zero.

DIO#_EF_READ_A_F_AND_RESET – Performs the same operation as DIO#_EF_READ_A_F, then sets the count to zero.

Example

See our Kipling quadrature input guide.

Configure DIO6 and DIO7 as quadrature inputs:

```
DIO6_EF_ENABLE = 0    //Make sure DIO-EF is disabled on DIO6
DIO7_EF_ENABLE = 0    //Make sure DIO-EF is disabled on DIO7

DIO6_EF_INDEX = 10    //Set feature index for DIO6 to quadrature.
DIO7_EF_INDEX = 10    //Set feature index for DIO7 to quadrature.

DIO6_EF_ENABLE = 1    //Enable quadrature DIO-EF on DIO6, for A
phase.
DIO7_EF_ENABLE = 1    //Enable quadrature DIO-EF on DIO7, for B
phase.
```

Edges on the two lines will now be decoded and the count will be incremented or decremented according to the edge sequence.

The current count can be read from DIO6_EF_READ_A or DIO6_EF_READ_A_AND_RESET.

To enable z-phase connected to DIO5 you would do:

```
DIO6_EF_CONFIG_A=1
DIO6_EF_CONFIG_B=5
DIO7_EF_CONFIG_A=1
DIO7_EF_CONFIG_B=5
```

Testing

On some LabJack devices you can physically tap a GND wire into the two DIO channels to cause some counts, but that does not work on T-series devices. Testing needs to be done with a proper quadrature signal.

If testing with an actual encoder, start with DIO-EF enabled and simply watch (e.g. in Kipling) the digital inputs as you slowly turn the encoder to see if the inputs change between high and low. That confirms a valid electrical connection.

You can test using 2 digital outputs to create a quadrature sequence. For example, configure quadrature on DIO6/7 as shown above, and connect DIO0 to DIO6 and DIO1 to DIO7. This can

be done with the Register Matrix in Kipling.

```
DIO0 = 1    //Initialize DIO0 to output-high.
DIO1 = 1    //Initialize DIO1 to output-high.

DIO6_EF_ENABLE = 0    //Make sure DIO-EF is disabled on DIO6
DIO7_EF_ENABLE = 0    //Make sure DIO-EF is disabled on DIO7
DIO6_EF_INDEX = 10    //Set feature index for DIO6 to quadrature.
DIO7_EF_INDEX = 10    //Set feature index for DIO7 to quadrature.
DIO6_EF_ENABLE = 1    //Enable quadrature DIO-EF on DIO6, for A
phase.
DIO7_EF_ENABLE = 1    //Enable quadrature DIO-EF on DIO7, for B
phase.
```

Now we can simulate a quadrature signal by toggling DIO0 and DIO1—aka FIO0 and FIO1—in the proper sequence. We will use the FIO_STATE register (address=2500) which operates on all 8 FIO bits at once, but we will set the inhibit bits for FIO2-FIO7 (bits 10-15) so they are not affected. To set bits 10-15 we can simply add 64512 to the desired FIO0/1 state value:

```
Write FIO_STATE = 3 + 64512, then should read DIO6_EF_READ_A =  0
Write FIO_STATE = 1 + 64512, then should read DIO6_EF_READ_A =  1
Write FIO_STATE = 0 + 64512, then should read DIO6_EF_READ_A =  0
Write FIO_STATE = 2 + 64512, then should read DIO6_EF_READ_A = -
1
Write FIO_STATE = 3 + 64512, then should read DIO6_EF_READ_A = -
2
Write FIO_STATE = 2 + 64512, then should read DIO6_EF_READ_A = -
1
Write FIO_STATE = 0 + 64512, then should read DIO6_EF_READ_A =  0
Write FIO_STATE = 1 + 64512, then should read DIO6_EF_READ_A =  1
Write FIO_STATE = 3 + 64512, then should read DIO6_EF_READ_A =  2
Write FIO_STATE = 2 + 64512, then should read DIO6_EF_READ_A =  3
Write FIO_STATE = 0 + 64512, then should read DIO6_EF_READ_A =  4
Write FIO_STATE = 1 + 64512, then should read DIO6_EF_READ_A =  5
Write FIO_STATE = 3 + 64512, then should read DIO6_EF_READ_A =  6
Write FIO_STATE = 2 + 64512, then should read DIO6_EF_READ_A =  7
```

Quadrature Decoding or Simple Counting?

Quadrature decoding is only needed when you need to keep track of absolute position with changes in direction included.

If you want to track absolute position but the direction does not change, or for whatever reason you always know the direction of movement, then you can just count the pulses from one phase (A or B) using a simple counter. The interrupt counters available on 6 of the FIO0 lines have the same 70k edge rate limit discussed below, but they only do 1x counting and only use 1 timer. The high-speed counters on the 4 CIO lines can handle up to 5 MHz per counter.

If you are just trying to measure frequency, you can use a counter as described above and note the change in count over some time interval, or you can use a DIO-EF that measures the time of individual pulses. Either way, just one phase (A or B) is needed.

Edge Rate Limits

Keep in mind that T-series devices do 4x quadrature counting. For example, a 100 pulses/revolution encoder connected to a pair of DIO will generate 400 edges/revolution.

This interrupt-based digital I/O extended feature (DIO-EF) is not purely implemented in hardware, but rather firmware must service each edge. This makes it substantially slower than other DIO-EF that are purely hardware-based. To avoid missed edges, the aggregate limit for edges seen by all interrupt-based DIO-EF is 70k edges/second. If stream mode is active, the limit is reduced to 20k edges/second. Excessive processor loading (e.g. a busy Lua script) can also reduce these limits. Note that interrupt features must process all edges, rising & falling, even if a given feature is configured to only look at one or the other. Additionally, note that intensive interrupt based features may limit the maximum streaming rates due to processor loading.

The more proper way to think of the edge limit, and understand error that could be introduced when using multiple interrupt-based DIO-EF, is to consider that the interrupt that processes an edge can take up to 14 µs to complete. When a particular channel sees an applicable edge, an IF (interrupt flag) is set for that channel that tells the processor it needs to run an ISR (interrupt service routine) for that channel. Once an ISR is started, it runs to completion and no other ISR can run until it is done (except that stream interrupts are higher priority and will preempt other interrupts). When an ISR completes, it clears the IF for that channel. So it is okay to have edges on multiple channels at the same time, as long as there is not another edge on any of those channels before enough time to process all the initial edges.

Say that channel A & B have an edge occur at the same time and an ISR starts to process the edge on channel A. If channel A has another edge during the first 14 µs, that edge will be lost. If channel B has another edge during the first 14 µs, the initial edge will be lost. If channel B has another edge during the second 14 µs (during the ISR for channel B), the new edge will be lost.

For faster quadrature tracking, one option is to use a chip such as the LS7366R-S from US Digital and then use the SPI ability of the T-series device to talk to that chip. In fact, a Lua script can be used to handle the SPI communication with the chip and periodically put the current count in a user-ram register than can be easily read by any host application or Modbus client.

13.2.12 Interrupt Frequency In [T-Series Datasheet]

Overview

T4 Capable DIO: **DIO4, DIO5, DIO6, DIO7, DIO8, DIO9** (aka FIO4, FIO5, FIO6, FIO7, EIO0, EIO1)

T7 Capable DIO: **DIO0, DIO1, DIO2, DIO3, DIO6, DIO7** (aka FIO0, FIO1, FIO2, FIO3, FIO6, FIO7)

T8 Capable DIO: **DIO0, DIO1, DIO2, DIO3, DIO4, DIO5, DIO6, DIO7, DIO8, DIO9, DIO10, DIO11, DIO12, DIO13, DIO14, DIO15** (aka FIO0, FIO1, FIO2, FIO3, FIO4, FIO5, FIO6, FIO7, EIO0, EIO1, EIO2, EIO3, EIO4, EIO5, EIO6, EIO7)

Requires Clock Source: **No. Uses core clock / 2.**

Index: **11**

Streamable: **Yes—integer READ registers only.**

Interrupt Frequency In will measure the frequency of a signal on the associated DIO line.

This interrupt-based digital I/O extended feature (DIO-EF) is not purely implemented in hardware, but rather firmware must service each edge. See the discussion of edge rate limits at the bottom of this page.

To measure the frequency, the T-series device will measure the duration of one or more periods. There are several option available to control the way the LabJack does this. The number of periods to be averaged, the edge direction to trigger on, and whether to measure continuously or in one-shot mode can all be specified.

The clock source for this feature is simply half the core frequency (See the Clock Speed section for information about your device clock speed settings):

```
ClockFrequency = CoreFrequency / 2
Period(s) = DIO#_EF_READ_A / ClockFrequency
Frequency (Hz) = ClockFrequency / DIO#_EF_READ_A
```

The maximum measurable time is 107 s. The number of periods to be averaged multiplied by the maximum expected period must be less than 107 s or the result will overflow:

```
107 > (NumToAverage * MaxPeriod)
```

By default, Interrupt Frequency In operates in one-shot mode where it will measure the frequency once after being enabled and a new measurement only once after each read. The other option is continuous mode, where the frequency is constantly measured (every edge is processed) and READ registers return the most recent result. Running in continuous mode puts a greater load on the processor.

Configure

DIO#_EF_ENABLE: 0 = Disable, 1 = Enable
DIO#_EF_INDEX: 11
DIO#_EF_OPTIONS: Not Used.
DIO#_EF_CONFIG_A: Default = 0. Bit 0: Edge select; 0 = falling, 1 = rising. Bit 1: 0 = one-shot, 1 = continuous.
DIO#_EF_CONFIG_B: Default = 0 which equates to 1. Number of periods to be measured and averaged.
DIO#_EF_CONFIG_C: Not used.
DIO#_EF_CONFIG_D: Not used.

One-Shot
When one-shot mode is enabled, the DIO_EF will complete a measurement then go idle. No more measurements will be made until the DIO_EF has been read or reset.

Continuous
When continuous mode is enabled, the DIO_EF will repeatedly make measurements. If a new reading is completed before the old one has been read the old one will be discarded.

Averaging
When averaging is enabled, the DIO_EF will wait for the specified number of measurements to be completed, then the average of the measurements will be made available in the READ registers.

Update

No update operations can be performed with Interrupt Frequency In.

Read

Results are read from the following registers.

DIO#_EF_READ_A: Returns the average period per cycle in ticks (core clock ticks / 2).
DIO#_EF_READ_B: Returns the total core clock tick count.
DIO#_EF_READ_A_F: Returns the average period per cycle in seconds. Takes into account the number of periods to be averaged and the core clock speed.
DIO#_EF_READ_B_F: Returns the average frequency in Hz. Takes into account the number of periods to be averaged and the core clock speed.

Note that all "READ_B" registers are capture registers. All "READ_B" registers are only updated when any "READ_A" register is read. Thus it would be unusual to read any B registers without first reading at least one A register.

Stream Read

All operations discussed in this section are supported in command-response mode.
In stream mode, you can read from the integer READ registers (A, B, A_AND_RESET), but as mentioned in the Stream Section those reads only return the lower 16 bits so you need to also use STREAM_DATA_CAPTURE_16 in the scan list to get the upper 16 bits.

Reset

DIO#_EF_READ_A_AND_RESET: Returns the same data as DIO#_EF_READ_A and then clears the result so that zero is returned by subsequent reads until another full period is measured (2 new edges).
DIO#_EF_READ_A_AND_RESET_F: Returns the same data as DIO#_EF_READ_A_F and then clears the result so that zero is returned by subsequent reads until another full period is measured (2 new edges).

One-shot, Continuous, Read, Read and Reset

When you configure this feature you can choose one-shot or continuous mode, and when you read from this feature you can do so without or with reset. This leads to 4 different behaviors.

One common scenario is an application that should always display the most recent frequency measurement. You don't want the display to go to 0 if you are reading faster than the pulses are coming in, but rather want it to keep returning the last measurement it did get. This is typically accomplished with one-shot and read without reset, but continuous can also be used.

Another common scenario is where you are reading faster than pulses are coming in, you want 1 and only 1 read to return a measurement when available, and want extra reads to return 0. This is typically accomplished with continuous and read with reset.

One-shot, read without reset:

- When a measurement has been completed, the value(s) are stored and the DIO_EF will be paused.
- When the READ_A register is read, the stored value for READ_A will be returned.
- Reading again, before another measurement has been completed, will return the previous result.
- At most, every other period will be measured. The delay between completing a measurement and the read causes come portion of a period to be missed.
- Use this mode when you need to be able to continuously read the most recent measurement and when you want to reduce the LabJack's processor loading.

One-shot, read with reset:

- When a measurement has been completed, the value is stored, and the DIO_EF will be paused.
- When the READ_A register is read, the stored value for READ_A will be returned and then that stored value will be set to zero. After the read has been completed, a new measurement will be started.

- Reading from READ_A again before another measurement has been completed will return zero. READ_B will continue to return the most recent value.
- At most, every other period will be measured. The delay between completing a measurement and the read causes come portion of a period to be missed.
- Use this mode when you need to get a result once for each new measurement and zero otherwise. This mode is also useful for reducing the LabJack's processor loading.

Continuous, read without reset:

- Each time a measurement is completed, the value is stored. The previously stored value will be overwritten, even if it has not been read.
- When the READ_A register is read, the stored value for READ_A will be returned.
- Measures every period as long as the signal frequency and processor utilization allow.
- Use this mode when you need to measure every period, or you want the value from the most recent measurement.

Continuous, read with reset:

- Each time a measurement has been completed, the value is stored. The previously stored value will be overwritten, even if is has not been read.
- When the READ_A register is read, the stored value for READ_A will be returned and then that stored value will be set to zero.
- Read B, will always return the value from the most recent measurement.
- Measures every period as long as the signal frequency and processor utilization allow.
- Use this mode when you need to measure every period, and you only want a non-zero value when a new measurement has been completed.

Example

To configure Interrupt Frequency In on DIO6 you can simply write to 2 registers:

```
DIO6_EF_ENABLE = 0
DIO6_EF_INDEX = 11
DIO6_EF_ENABLE = 1
```

Now you can read the period in seconds from a 4th register DIO6_EF_READ_A_F.

Edge Rate Limits

This interrupt-based digital I/O extended feature (DIO-EF) is not purely implemented in hardware, but rather firmware must service each edge. This makes it substantially slower than other DIO-EF that are purely hardware-based. To avoid missed edges, the aggregate limit for edges seen by all interrupt-based DIO-EF is 70k edges/second. If stream mode is active, the limit is reduced to 20k edges/second. Excessive processor loading (e.g. a busy Lua script) can

also reduce these limits. Note that interrupt features must process all edges, rising & falling, even if a given feature is configured to only look at one or the other. Additionally, note that intensive interrupt based features may limit the maximum streaming rates due to processor loading.

The more proper way to think of the edge limit, and understand error that could be introduced when using multiple interrupt-based DIO-EF, is to consider that the interrupt that processes an edge can take up to 14 µs to complete. When a particular channel sees an applicable edge, an IF (interrupt flag) is set for that channel that tells the processor it needs to run an ISR (interrupt service routine) for that channel. Once an ISR is started, it runs to completion and no other ISR can run until it is done (except that stream interrupts are higher priority and will preempt other interrupts). When an ISR completes, it clears the IF for that channel. So it is okay to have edges on multiple channels at the same time, as long as there is not another edge on any of those channels before enough time to process all the initial edges.

Say that channel A & B have an edge occur at the same time and an ISR starts to process the edge on channel A. If channel A has another edge during the first 14 µs, that edge will be lost. If channel B has another edge during the first 14 µs, the initial edge will be lost. If channel B has another edge during the second 14 µs (during the ISR for channel B), the new edge will be lost.

13.2.13 Conditional Reset [T-Series Datasheet]

Overview

T4 Capable DIO: **DIO4, DIO5, DIO6, DIO7, DIO8, DIO9** (aka FIO4, FIO5, FIO6, FIO7, EIO0, EIO1)

T7 Capable DIO: **DIO0, DIO1, DIO2, DIO3, DIO6, DIO7** (aka FIO0, FIO1, FIO2, FIO3, FIO6, FIO7)

T8 Capable DIO: **DIO0, DIO1, DIO2, DIO3, DIO4, DIO5, DIO6, DIO7, DIO8, DIO9, DIO10, DIO11, DIO12, DIO13, DIO14, DIO15** (aka FIO0, FIO1, FIO2, FIO3, FIO4, FIO5, FIO6, FIO7, EIO0, EIO1, EIO2, EIO3, EIO4, EIO5, EIO6, EIO7)

Requires Clock Source: **No**

Index: **12**

Streamable: **No**

DIO-EF Conditional Reset will reset a specified DIO-EF after a specified number of edges have been detected.

Configure

To set up a DIO-EF Conditional Reset is simple. Just set the DIO number of the DIO-EF you would like to reset and then set the other options.

DIO#_EF_ENABLE: 0 = Disable, 1 = Enable
DIO#_EF_INDEX: 12
DIO#_EF_OPTIONS: Not used.
DIO#_EF_CONFIG_A: Reset Options bitmask:

- bit 0: Edge select. 1 = rising, 0 = falling
- bit 1: Disable. 1 = disable the selected DIO_EF on reset. 0 = don't disable.
- bit 2: OneShot. 1 = only reset once. 0 = reset every n edges.

DIO#_EF_CONFIG_B: Number of edges per reset.
DIO#_EF_CONFIG_C: IO number of DIO-EF to be reset.
DIO#_EF_CONFIG_D: Not used.

To disable a DIO_EF mode instead of resetting the count, set the CONFIG_A "Disable" bit to 1. This is similar behavior to a OneShot reset, but instead of resetting the count only once, it disables the set DIO_EF mode. There is conflicting behavior between Disable and OneShot so it is recommended to use either Disable or OneShot (or neither), but not both. In the case where both Disable and OneShot are used, Disable has a higher priority so the DIO_EF will be disabled instead of being reset.

Update

No update operations can be performed on Conditional Reset.

Read

Results are read from the following registers.

DIO#_EF_READ_A – Returns the current count.

Example

This example assumes that DIO0 has a running extended feature such as quadrature or a counter. Now we will set up DIO2 as a falling edge trigger that will reset the count of DIO0_EF.

```
DIO2_EF_ENABLE = 0    // Ensure that the DIO-EF is not running so that it can be
configured.
DIO2_EF_INDEX = 12    // Set to Conditional Reset
DIO2_EF_CONFIG_A = 0  // Falling edges
DIO2_EF_CONFIG_B = 1  // Reset every edges
DIO2_EF_CONFIG_C = 0  // Reset events clear the count of DIO0_EF
DIO2_EF_ENABLE = 1    // Turn on the DIO-EF
```

Now falling edges on DIO2 will set the count of DIO0_EF to zero.

For a more detailed walkthrough, see Configuring & Reading a Counter.

Edge Rate Limits

This interrupt-based digital I/O extended feature (DIO-EF) is not purely implemented in hardware, but rather firmware must service each edge. This makes it substantially slower than other DIO-EF that are purely hardware-based. To avoid missed edges, the aggregate limit for edges seen by all interrupt-based DIO-EF is 70k edges/second. If stream mode is active, the limit is reduced to 20k edges/second. Excessive processor loading (e.g. a busy Lua script) can also reduce these limits. Note that interrupt features must process all edges, rising & falling, even if a given feature is configured to only look at one or the other. Additionally, note that intensive interrupt based features may limit the maximum streaming rates due to processor loading.

The more proper way to think of the edge limit, and understand error that could be introduced when using multiple interrupt-based DIO-EF, is to consider that the interrupt that processes an

edge can take up to 14 µs to complete. When a particular channel sees an applicable edge, an IF (interrupt flag) is set for that channel that tells the processor it needs to run an ISR (interrupt service routine) for that channel. Once an ISR is started, it runs to completion and no other ISR can run until it is done (except that stream interrupts are higher priority and will preempt other interrupts). When an ISR completes, it clears the IF for that channel. So it is okay to have edges on multiple channels at the same time, as long as there is not another edge on any of those channels before enough time to process all the initial edges.

Say that channel A & B have an edge occur at the same time and an ISR starts to process the edge on channel A. If channel A has another edge during the first 14 µs, that edge will be lost. If channel B has another edge during the first 14 µs, the initial edge will be lost. If channel B has another edge during the second 14 µs (during the ISR for channel B), the new edge will be lost.

13.3 I2C [T-Series Datasheet]

Overview

I²C or I2C is a two-wire synchronous serial protocol typically used to send data between chips. I2C is considered an advanced topic. A good knowledge of the protocol is recommended. Troubleshooting may require a logic analyzer or oscilloscope. We recommend users inexperienced with I2C purchase our LJTick-DAC and get I2C working with it before attempting configuration with a third-party sensor. LabJack does not typically provide support for specific I2C sensors beyond clarifying the necessary I2C configuration steps. Some general troubleshooting guidance can be found below, including two useful troubleshooting utilities. See the attached "I2C_Test_Utilities.zip" at the bottom of the page (Windows only).

Subsections

T-series I2C Support

T-series devices support master-mode Inter-Integrated Circuit (I²C or I2C) communication. Two digital IO are required to act as clock (SCL) and data (SDA). Any T-series digital I/O line can be SDA or SCL.

The I²C bus generally requires pull-up resistors of perhaps 4.7 kΩ from SDA to VS and SCL to VS.

EIO and CIO digital I/O lines should be used, instead of the FIO lines, due to the extra lower output impedances on those lines. Look at the Digital I/O Specifications page for more details.

Lua scripting is often convenient for serial applications. For example, a script can run the serial communication at a specified interval, and put the result in USER_RAM registers. The host software can read from the USER_RAM registers when convenient. This puts the complications of serial communication in a script running on the T-series device. The I²C Sensor Lua examples contain several serial communication examples.

This I²C bus is not an alternative to the USB connection. Rather, the host application will write/read data to/from the LabJack, and the LabJack will communicate with some other device using I²C.

Data Rates

I²C is done in command-response mode, either from a host application or an on-board Lua script. Throughput can be estimated by looking at the bit rate and number of bits to transfer, and for external host communication adding the packet overhead estimates from the beginning of Appendix A-1. A further consideration is how much you can fit in 1 packet or whether multiple packets will be required.

> Note: I2C is implemented in software rather than with hardware. As a result, the I2C clock timing can vary somewhat depending on the T-series device processor load, and there is not any precise formula to calculate the bus speed based off of I2C_SPEED_THROTTLE.

Clock Stretching

T-Series devices support clock stretching. When a slave device needs more time to complete an operation it can use clock stretching to pause the bus. T-Series devices will allow up to 1000 clock periods of clock stretching. If a slave device needs more time, a slower clock speed should be used.

How-To

Device Control Basics

- All T-series device features are controlled by reading and writing Modbus TCP registers via Modbus TCP (either directly or through our LJM library).
- We have register descriptions throughout documentation detailing relevant register names, starting addresses, types, and access permissions (read/write).
- See Section 3.0 Communication for other detailed communication information.

Process

Running an I2C operation on a T-series device requires:

1. Initial configuration.
2. Transmit load data.
3. Specify the number of bytes to read.
4. Execute the I²C operations.
5. Read the data that was read from the slave device, if any.
6. Debugging (optional): Read the acknowledgement array to determine which bytes were acknowledged by the slave device.

Steps 2-5 can be repeated as long as I²C has not been used to communicate with a different device.

Initial Configuration

Several registers need to be written to configure the T-series device.

Digital IO lines need to be selected to act as SDA and SCL:

Name	Start Address	Type	Access
I2C_SDA_DIONUM	5100	UINT16	R/W

I2C_SDA_DIONUM
- Address: 5100

The number of the DIO line to be used as the I2C data line. Ex: Writing 0 will force FIO0 to become the I2C-SDA line.

- Data type: UINT16 (type index = 0)
- Readable and writable
- Default value: 0

Name	Start Address	Type	Access
I2C_SCL_DIONUM	5101	UINT16	R/W

I2C_SCL_DIONUM
- Address: 5101

The number of the DIO line to be used as the I2C clock line. Ex: Writing 1 will force FIO1 to become the I2C-SCL line.

- Data type: UINT16 (type index = 0)
- Readable and writable
- Default value: 0

Set the clock speed:

Name	Start Address	Type	Access
I2C_SPEED_THROTTLE	5102	UINT16	R/W

I2C_SPEED_THROTTLE
- Address: 5102

This value controls the I2C clock frequency. Pass 0-65535. Default=0 corresponds to 65536 internally which results in ~450 kHz.
1 results in ~40 Hz, 65516 is ~100 kHz.

- Data type: UINT16 (type index = 0)
- Readable and writable
- Default value: 0

The options register controls several compatibility settings:

Name	Start Address	Type	Access
I2C_OPTIONS	5103	UINT16	R/W

I2C_OPTIONS
- Address: 5103

Advanced. Controls details of the I2C protocol to improve device compatibility.
bit 0: 1 = Reset the I2C bus before attempting communication.
bit 1: 0 = Restarts will use a stop and a start, 1 = Restarts will not use a stop.
bit 2: 1 = disable clock stretching.

- Data type: UINT16 (type index = 0)
- Readable and writable
- Default value: 0

For example, setting bit 1 of I2C_OPTIONS to true supports the repeated start condition.

Load the address of the slave device:

Name	Start Address	Type	Access
I2C_SLAVE_ADDRESS	5104	UINT16	R/W

I2C_SLAVE_ADDRESS
- Address: 5104

The 7-bit address of the slave device. Value is shifted left by firmware to allow room for the I2C R/W bit.

- Data type: UINT16 (type index = 0)
- Readable and writable
- Default value: 0

Transmit load data.

Load an array of bytes to send to the slave device. This array does not include the device address.

Name	Start Address	Type	Access
I2C_NUM_BYTES_TX	5108	UINT16	R/W

I2C_NUM_BYTES_TX
- Address: 5108

The number of data bytes to transmit. Zero is valid and will result in a read-only I2C operation.

- Data type: UINT16 (type index = 0)
- Readable and writable
- Default value: 0

I2C_DATA_TX	5120	BYTE	W

I2C_DATA_TX
- Address: 5120

Data that will be written to the I2C bus. This register is a buffer.

- Data type: BYTE (type index = 99)
- Write-only
- Default value: 0
- This register is a Buffer Register

Specify the number of bytes to read.

Write the number of bytes to be read from the slave device. Read operations will always be performed after any write operations.

Name	Start Address	Type	Access
I2C_NUM_BYTES_RX	5109	UINT16	R/W

I2C_NUM_BYTES_RX
- Address: 5109

The number of data bytes to read. Zero is valid and will result in a write-only I2C operation.

- Data type: UINT16 (type index = 0)
- Readable and writable
- Default value: 0

Execute the I²C operations.

Instruct the T-series device to run the write and read operations specified in the last two steps.

Name	Start Address	Type	Access
I2C_GO	5110	UINT16	R/W

I2C_GO
- Address: 5110

Writing to this register will instruct the LabJack to perform an I2C transaction.

- Data type: UINT16 (type index = 0)
- Readable and writable
- Default value: 0
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 0.9504

Read the data that was read from the slave device, if any.

Data received from the slave device will be saved in buffer on the T-series device. Use the I2C_DATA_RX to read the buffer:

Name	Start Address	Type	Access
I2C_DATA_RX	5160	BYTE	R

I2C_DATA_RX
- Address: 5160

Data that has been read from the I2C bus. This register is a buffer. Underrun behavior - fill with zeros.

- Data type: BYTE (type index = 99)
- Read-only
- Default value: 0
- This register is a Buffer Register

Debugging: Read the acknowledgement array to determine which bytes were acknowledged by the slave device.

The below register can help troubleshoot I²C issues. The ACKs register will record all Acknowledgement signals transmitted from the slave device to the master (master is the T-series device). Data is always transmitted over I²C in the following sequence: Slave Address, Write data, Slave Address, Read data. After each byte sent to the slave device an acknowledgement will be stored in bit 0. A 1 indicates that the bytes was acknowledged, a 0 indicates no acknowledgement. Before saving the an ACK or no-ACK the register is shifted left.

For example, if two bytes are transmitted and three are read there will be four ACKs. Bit 3

represents the acknowledgement to the first slave address which starts the write operation, bit 2 is the first data byte, Bit 1 is the seconds data byte, bit 0 is the second slave address which starts the read operation.

Name	Start Address	Type	Access
I2C_ACKS	5114	UINT32	R/W

I2C_ACKS
- Address: 5114

A binary encoded value (array of bits) used to observe ACKs from the slave device.

- Data type: UINT32 (type index = 1)
- Readable and writable
- Default value: 0
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 0.9504

Example

This demonstrates I²C communications with an LJTick-DAC connected to FIO0/FIO1.

First, configure the I²C settings:

```
I2C_SDA_DIONUM = 1      // Set SDA pin number = 1(FIO1)
I2C_SCL_DIONUM = 0      // Set SCL pin number = 0 (FIO0)
I2C_SPEED_THROTTLE = 0  // Set speed throttle = 0 (max)
I2C_OPTIONS = 0         // Set options = 0
I2C_SLAVE_ADDRESS = 80  // Set 7-bit slave address of the I²C chip = 80 (0x50)
```

Read from EEPROM bytes 0-3 in the user memory area. We need a single I²C transmission that writes the chip's memory pointer and then reads the data.

```
I2C_NUM_BYTES_TX = 1        // Set the number of bytes to transmit to 1
I2C_NUM_BYTES_RX = 4        // Set the number of bytes to receive to 4
I2C_DATA_TX = {0}           // Set the TX data. byte 0: Memory pointer =
0.
I2C_GO = 1                  // Do the I²C communications.
I2C_DATA_RX = {?, ?, ?, ?}  // Get the RX data (4 bytes).
```

Write EEPROM bytes 0-3 in the user memory area, using the page write technique. Note that page writes are limited to 16 bytes max, and must be aligned with the 16-byte page intervals. For instance, if you start writing at address 14, you can only write two bytes because byte 16 is the start of a new page.

```
I2C_NUM_BYTES_TX = 5              // Set the number of bytes to transmit to 5
I2C_NUM_BYTES_RX = 0              // Set the number of bytes to receive to 0 (not receiving data)
I2C_DATA_TX = {0, 156, 26, 2, 201}  // Set the TX data. byte 0: Memory pointer = 0, bytes 1-4: EEPROM bytes 0-
3.
I2C_GO = 1                      // Do the I²C communications.
```

If using multiple I²C busses, just include writes to set the DIONUMs each time you want to communicate on a different bus. Other configuration is global also, so if different busses need different configuration include those writes also:

```
I2C_SDA_DIONUM = 2          // Set SDA pin number = 2 (FIO2)
I2C_SCL_DIONUM = 3          // Set SCL pin number = 3 (FIO3)
I2C_SPEED_THROTTLE = 65516  // Set speed throttle for ~100 kbps
I2C_NUM_BYTES_TX = 1        // Set the number of bytes to transmit to 1
I2C_NUM_BYTES_RX = 4        // Set the number of bytes to receive to 4
I2C_DATA_TX = {0}           // Set the TX data. byte 0: Memory pointer =
0.
I2C_GO = 1                // Do the I²C communications.
I2C_DATA_RX = {?, ?, ?, ?}  // Get the RX data (4 bytes).
```

Note: When writing the TX and RX data, LJM functions eWriteNameArray and eReadNameArray functions are recommended for ease of use.

I2C FAQ/Common Questions

Q: Why are no I2C ACK bits being received?

- Double check to make sure pull-up resistors are installed. A general rule for selecting the correct size pull-up resistors is to start with 4.7kΩ and adjust down to 1kΩ as necessary. If necessary, an oscilloscope should be used to ensure proper digital signals are present on the SDA and SCL lines.
- Double check to make sure the correct I/O lines are being used. It is preferred to do I2C communication on EIO/CIO/MIO lines instead of the FIO lines due to the larger series resistance (ESD protection) implemented on the FIO lines.
- Use an oscilloscope to verify the SDA and SCL lines are square waves and not weird arch signals (see "I2C_SPEED_THROTTLE" or use EIO/CIO/MIO lines).
- Use a logic analyzer (some oscilloscopes have this functionality) to verify the correct slave address is being used. See this EEVblog post on budget-friendly options. It is common to not take into account 7-bit vs 8-bit slave addresses or properly understand how LabJack handles the defined slave address and the read/write bits defined by the I2C protocol to perform read and write requests.
- Make sure your sensor is being properly powered. The VS lines of LJ devices are ~5V and the I/O lines are 3.3V. Sometimes this is a problem. Consider buying a LJTick-LVDigitalIO or powering the sensor with an I/O line or DAC channel.

Q: I've tried everything, still no I2C Ack Bits...

- Try slowing down the I2C bus using the "I2C_SPEED_THROTTLE" register/option. Reasons:
- Not all I2C sensors can communicate at the full speed of the LabJack. Check the I2C sensor datasheet.
- The digital signals could be getting corrupted due to the series resistors of the I/O lines on the LabJack.
- Consider finding a way to verify that your sensor is still functioning correctly using an

Arduino and that it isn't broken.
- Try to establish communications with an LJTick-DAC to ensure the DIO are operating properly and that you are configuring I2C properly.

Q: Why is my device not being found by the I2C.search function?

- See information on I2C ACK bits above.

Q: What are I2C Read and Write functions or procedures?

- There are a few really good resources for learning about the general flow of I2C communication.
- TI's Understanding the I2C Bus by Jonathan Valdez and Jared Becker is a high quality resource
- I2C-bus specification by NXP
- Using the I2C Bus by Robot Electronics
- I2C Bus Specification by i2c.info

Q: Why am I getting a I2C_BUS_BUSY (LJM Error code 2720) error?

- See information on I2C ACK Bits above. Try different pull-up resistor sizes.

File Attachment:

I2C_Test_Utilities.zip

13.3.1 I2C Simulation Tool [T-Series Datasheet]

I2C FAQ/Common Questions

Q: Why are no I2C ACK bits being received?

- Double check to make sure pull-up resistors are installed. A general rule for selecting the correct size pull-up resistors is to start with 4.7kΩ and adjust down to 1kΩ as necessary. If necessary, an oscilloscope should be used to ensure proper digital signals are present on the SDA and SCL lines.
- Double check to make sure the correct I/O lines are being used. It is preferred to do I2C communication on EIO/CIO/MIO lines instead of the FIO lines due to the larger series resistance (ESD protection) implemented on the FIO lines.
- Use an oscilloscope to verify the SDA and SCL lines are square waves and not weird arch signals (see "I2C_SPEED_THROTTLE" or use EIO/CIO/MIO lines).
- Use a logic analyzer (some oscilloscopes have this functionality) to verify the correct slave address is being used. See this EEVblog post on budget-friendly options. It is common to not take into account 7-bit vs 8-bit slave addresses or properly understand how LabJack handles the defined slave address and the read/write bits defined by the I2C protocol to perform read and write requests.
- Make sure your sensor is being properly powered. The VS lines of LJ devices are ~5V and the I/O lines are 3.3V. Sometimes this is a problem. Consider buying a LJTick-LVDigitalIO or powering the sensor with an I/O line or DAC channel.

Q: I've tried everything, still no I2C Ack Bits...

- Try slowing down the I2C bus using the "I2C_SPEED_THROTTLE" register/option. Reasons:
- Not all I2C sensors can communicate at the full speed of the LabJack. Check the I2C sensor datasheet.
- The digital signals could be getting corrupted due to the series resistors of the I/O lines on the LabJack.
- Consider finding a way to verify that your sensor is still functioning correctly using an Arduino and that it isn't broken.
- Try to establish communications with an LJTick-DAC to ensure the DIO are operating properly and that you are configuring I2C properly.

Q: Why is my device not being found by the I2C.search function?

- See information on I2C ACK bits above.

Q: What are I2C Read and Write functions or procedures?

- There are a few really good resources for learning about the general flow of I2C communication.
- TI's Understanding the I2C Bus by Jonathan Valdez and Jared Becker is a high quality resource
- I2C-bus specification by NXP

- Using the I2C Bus by Robot Electronics
- I2C Bus Specification by i2c.info

Q: Why am I getting a I2C_BUS_BUSY (LJM Error code 2720) error?

- See information on I2C ACK Bits above. Try different pull-up resistor sizes.

13.4 SPI [T-Series Datasheet]

SPI Overview

SPI (serial peripheral interface) is a four-wire synchronous serial protocol. SPI is considered an advanced topic. A good knowledge of the protocol is recommended. Troubleshooting may require a logic analyzer or oscilloscope.

T-series SPI Support

T-Series devices support Serial Peripheral Interface (SPI) communication as the master only. Four digital IO lines are used: MISO (data line; Master In Slave Out), MOSI (data line; Master Out Slave In), CLK (clock), and CS (chip select).

Lua scripting is often convenient for serial applications. For example, a script can run the serial communication at a specified interval, and put the results in USER_RAM registers. The host software can read from the USER_RAM registers when convenient. This puts the complications of serial communication in a script running on the T-series device. The Lua examples contain several serial communication examples.

SPI is not an alternative to the USB connection. Rather, the host application will write/read data to/from the T-series device, and the T-series device communicates with some other device using the serial protocol.

How-To

Device Control Basics

- All T-series device features are controlled by reading and writing Modbus TCP registers via Modbus TCP (either directly or through our LJM library).
- We have register descriptions throughout documentation detailing relevant register names, starting addresses, types, and access permissions (read/write).
- See Section 3.0 Communication for other detailed communication information.

Process

Running a SPI operation on a T-series device requires:

1. Initial configuration.
2. Send load data to the slave device.
3. Execute the SPI operation.
4. Read the data that was read from the slave device.

Initial configuration.

Several registers need to be written to configure the T-series device.

DIO lines need to be selected to act as MISO, MOSI, CLK, and CS:

Name	Start Address	Type	Access
SPI_CS_DIONUM	5000	UINT16	R/W

SPI_CS_DIONUM
- Address: 5000

The DIO line for Chip-Select.

- Data type: UINT16 (type index = 0)
- Readable and writable
- Default value: 0

Name	Start Address	Type	Access
SPI_CLK_DIONUM	5001	UINT16	R/W

SPI_CLK_DIONUM
- Address: 5001

The DIO line for Clock.

- Data type: UINT16 (type index = 0)
- Readable and writable
- Default value: 0

Name	Start Address	Type	Access
SPI_MISO_DIONUM	5002	UINT16	R/W

SPI_MISO_DIONUM
- Address: 5002

The DIO line for Master-In-Slave-Out.

- Data type: UINT16 (type index = 0)
- Readable and writable
- Default value: 0

Name	Start Address	Type	Access
SPI_MOSI_DIONUM	5003	UINT16	R/W

SPI_MOSI_DIONUM
- Address: 5003

The DIO line for Master-Out-Slave-In.

- Data type: UINT16 (type index = 0)
- Readable and writable
- Default value: 0

The SPI mode controls which edge of clock signals value data, and whether the clock will idle high or low:

Name	Start Address	Type	Access
SPI_MODE	5004	UINT16	R/W

SPI_MODE
- Address: 5004

The SPI mode controls the clock idle state and which edge clocks the data. Bit 1 is CPOL and Bit 0 is CPHA, so CPOL/CPHA for different decimal values:
0 = 0/0 = b00,
1 = 0/1 = b01,
2 = 1/0 = b10,
3 = 1/1 = b11.
For CPOL and CPHA explanations, see Wikipedia article: https://en.wikipedia.org/wiki/Serial_Peripheral_Interface_Bus.

- Data type: UINT16 (type index = 0)
- Readable and writable
- Default value: 0

Set the clock speed with SPI_SPEED_THROTTLE:

Name	Start Address	Type	Access
SPI_SPEED_THROTTLE	5005	UINT16	R/W

SPI_SPEED_THROTTLE
- Address: 5005

This value controls the SPI clock frequency. Pass 0-65535. Default=0 corresponds to 65536 internally which results in ~800 kHz.
65500 = ~100 kHz,
65100 = ~10 kHz,
61100 = ~1 kHz,
21000 = ~100 Hz,
and 1 = ~67 Hz.
Avoid setting too low such that the entire transaction lasts longer than the 250 millisecond timeout of the internal watchdog timer.

- Data type: UINT16 (type index = 0)
- Readable and writable
- Default value: 0

Starting with a low clock speed and increasing the speed after everything is working is usually a good idea.

The following table lists approximate clock rates measured for various values of SPI_SPEED_THROTTLE with T7 firmware 1.0150:

Throttle Value	Clock Speed [kHz]
0	780
65530	380
65500	100
65100	10
61100	1
21000	0.1
1	0.067

Lastly, set SPI_OPTIONS:

Name	Start Address	Type	Access
SPI_OPTIONS	5006	UINT16	R/W

SPI_OPTIONS
- Address: 5006

Bit 0 is Auto-CS-Disable. When bit 0 is 0, CS is enabled. When bit 0 is 1, CS is disabled.
Bit 1: 0 = Set DIO directions before starting the SPI operations, 1 = Do not set DIO directions.
Bit 2: 0 = Transmit data MSB first, 1 = LSB first.
Bits 4-7: This value sets the number of bits that will be transmitted during the last byte of the SPI operation. Default is 8, valid options are 1-8.

- Data type: UINT16 (type index = 0)
- Readable and writable
- Default value: 0

Setting SPI_OPTIONS is normally not needed, but can help with compatibility in the following situations:

- If the hardware setup does not require a CS line.
- If the hardware setup requires special digital IO configuration.
- If data needs to be transmitted LSB first.
- If the number of bits to be transferred is not an even multiple of eight.

Send load data to the slave device.

Name	Start Address	Type	Access
SPI_NUM_BYTES	5009	UINT16	W

SPI_NUM_BYTES
- Address: 5009

The number of bytes to transfer. The maximum transfer size is 100 bytes.

- Data type: UINT16 (type index = 0)
- Write-only
- Default value: 0

Name	Start Address	Type	Access
SPI_DATA_TX	5010	BYTE	W

SPI_DATA_TX
- Address: 5010

Write data here. This register is a buffer.

- Data type: BYTE (type index = 99)
- Write-only
- Default value: 0
- This register is a Buffer Register

SPI is full duplex. That means that data is sent in both directions at the same time. The number of bytes read from the slave must always equal the number of bytes written to the slave and the maximum transfer size is 100 bytes. To read data from a slave device without sending data to it, load dummy data into SPI_DATA_TX.

Execute the SPI operation.

Name	Start Address	Type	Access
SPI_GO	5007	UINT16	W

SPI_GO
- Address: 5007

Write 1 to begin the configured SPI transaction.

- Data type: UINT16 (type index = 0)
- Write-only
- Default value: 0
- This register uses system RAM. The maximum RAM is 64KB. For more information, see 4.4 RAM

Read the data that was read from the slave device.

Name	Start Address	Type	Access
SPI_DATA_RX	5050	BYTE	R

SPI_DATA_RX
- Address: 5050

Read data here. This register is a buffer. Underrun behavior - fill with zeros.

- Data type: BYTE (type index = 99)
- Read-only
- Default value: 0
- This register is a Buffer Register

Common Issues

Timeouts

T-Series devices have an internal watchdog that will timeout, and cause the device to reboot, if a single SPI transaction lasts longer than 250 ms. Stay safely above the following measured minimum values where device did not reboot for different numbers of bytes:

# Bytes	Throttle Value	Clock Rate [Hz]
1	1	67
2	4900	73
3	23900	106
4	33900	140
10	52600	342
16	57400	544
32	61500	1

Noise Immunity

The clock input on the slave device can be vulnerable to noise. The clock input will notice quick edges caused by noise. If you suspect this to be a problem, add a capacitor close to the clock input of the slave. Suggested value is equal to or less than:

```
1 / (2 * pi * f * R)
```

...where f is the SPI frequency and R is the source impedance—which is dominated in this case by the DIO (use 550 ohms for FIO and 180 ohms for EIO).

Example

The following demonstrates a simple loop-back test. Connect a jumper between DIO2 and DIO3. Data will be sent out DIO3 and read from DIO2.

```
SPI_CS_DIONUM = 0          // Use DIO0 as chip select.
SPI_CLK_DIONUM = 1         // Use DIO1 as Clock
SPI_MISO_DIONUM = 2        // Use DIO2 as MISO
SPI_MOSI_DIONUM = 3        // Use DIO3 as MOSI
SPI_MODE = 0               // Select mode zero.
SPI_SPEED_THROTTLE = 65500  // Set clock speed to ~100 kHz.
SPI_OPTIONS = 0            // No special operation
SPI_NUM_BYTES = 1          // Transfer one byte.
SPI_DATA_TX = 0x55         // Load a test data byte. eWriteNameByteArray is the easiest way to write the outgoing
SPI data.
SPI_GO = 1                 // Initiate the transfer
```

At this point the T-series device T7 will run the SPI transaction. If no errors are returned, the received data can be read.

Read a test data byte. eReadNameByteArray is the easiest way to read the received SPI data.

Because of the loop-back, the data read from SPI_DATA_RX will match the data written to SPI_DATA_TX.

13.5 SBUS [T-Series Datasheet]

Overview

SBUS is a serial protocol used with SHT1X and SHT7x sensors from Sensirion. It is similar to I2C, but is not compatible. The EI-1050 uses the SHT11 sensor. Other available sensors are the SHT10, SHT15, SHT71, and SHT75. In the context of our devices, SBUS registers will also be used to interface newer I2C based SHT3x sensors from Sensirion (T4 firmware v1.0029+, T7 v1.0305+, T8 v1.0017+). SBUS protocol sensors used in the EI-1050 are now EOL.

T-Series SBUS Support

T-series devices support SBUS/SHT3x at a higher level compared to other devices using serial communication protocols. Protocols such as SPI and I2C provide direct access to the bus so that arrays of bytes can be sent and received, but T-series devices implement all the functionality necessary to run SBUS/SHT3x and convert the binary data to relative humidity (RH) or temperature.

Lua scripting is often convenient for serial applications. For example, a script can run the serial communication at a specified interval, and put the result in USER_RAM registers. The host software can read from the USER_RAM registers when convenient. This puts the complications of serial communication in a script running on the T-series device. The Lua examples contain several serial communication examples.

Default DIO Assignments

The lines used for data, clock, and power are set at startup according to the below list. The assignments can be changed as desired.

T4

- **DIO4 (FIO4)**: Data Line
- **DIO5 (FIO5)**: Clock Line
- **DIO6 (FIO6)**: Power line. Will be set to output-high. Can be disabled.

T7/T8

- **DIO0 (FIO0)**: Data Line
- **DIO1 (FIO1)**: Clock Line
- **DIO2 (FIO2)**: Power line. Will be set to output-high. Can be disabled.

How-To

SBUS on T-series LabJack devices is set up to work with the EI-1050 probe. SHT1x, SHT7x, and SHT3x sensors can be directly used as well. The most relevant difference is that the EI-1050 adds an enable line.

Device Control Basics

- All T-series device features are controlled by reading and writing Modbus TCP registers via Modbus TCP (either directly or through our LJM library).
- We have register descriptions throughout documentation detailing relevant register names, starting addresses, types, and access permissions (read/write).
- See Section 3.0 Communication for other detailed communication information.

EI-1050

The EI-1050 datasheet should be referenced for hardware connections. The temperature and humidity can be read using the SBUS#_TEMP and SBUS#_RH registers:

Name	Start Address	Type	Access
SBUS#(0:22)_TEMP	30100	FLOAT32	R

SBUS#(0:22)_TEMP
- Starting Address: 30100

Reads temperature in Kelvin from an SBUS sensor (EI-1050/SHT1x/SHT7x). SBUS# is the DIO line for the EI-1050 enable. If SBUS# is the same as the value specified for data or clock line, there will be no control of an enable line.

- Data type: FLOAT32 (type index = 3)
- Read-only
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0056

Expanded Names	Addresses
SBUS0_TEMP, SBUS1_TEMP, SBUS2_TEMP, SBUS3_TEMP, SBUS4_TEMP, SBUS5_TEMP, SBUS6_TEMP, SBUS7_TEMP, SBUS8_TEMP, SBUS9_TEMP, SBUS10_TEMP, SBUS11_TEMP, SBUS12_TEMP, SBUS13_TEMP, SBUS14_TEMP, SBUS15_TEMP, SBUS16_TEMP, SBUS17_TEMP, SBUS18_TEMP, SBUS19_TEMP, SBUS20_TEMP, SBUS21_TEMP, SBUS22_TEMP	30100, 30102, 30104, 30106, 30108, 30110, 30112, 30114, 30116, 30118, 30120, 30122, 30124, 30126, 30128, 30130, 30132, 30134, 30136, 30138, 30140, 30142, 30144

SBUS#(0:22)_RH	30150	FLOAT32	R

SBUS#(0:22)_RH
- Starting Address: 30150

Reads humidity in % from an external SBUS sensor (EI-1050/SHT1x/SHT7x). # is the DIO line for the EI-1050 enable. If # is the same as the value specified for data or clock line, there will be no control of an enable line.

- Data type: FLOAT32 (type index = 3)
- Read-only
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0056

Expanded Names	Addresses
SBUS0_RH, SBUS1_RH, SBUS2_RH, SBUS3_RH, SBUS4_RH, SBUS5_RH, SBUS6_RH, SBUS7_RH, SBUS8_RH, SBUS9_RH, SBUS10_RH, SBUS11_RH, SBUS12_RH, SBUS13_RH, SBUS14_RH, SBUS15_RH, SBUS16_RH, SBUS17_RH, SBUS18_RH, SBUS19_RH, SBUS20_RH, SBUS21_RH, SBUS22_RH	30150, 30152, 30154, 30156, 30158, 30160, 30162, 30164, 30166, 30168, 30170, 30172, 30174, 30176, 30178, 30180, 30182, 30184, 30186, 30188, 30190, 30192, 30194

The register index number determines which DIO port the EI-1050's enable line (brown wire) is connected to. See the Examples section below for examples.

When either SBUS#_TEMP and/or SBUS#_RH are read, the following steps are performed by firmware:

1. The power line is set to output-high.
2. The enable line is set to output-high (enabled).
3. The EI-1050 sensor is instructed to measure temperature or humidity.
4. When the measurement is complete, the result and the checksum are read from the sensor.
5. The enable line is set to output-low (disabled).
6. Firmware verifies the checksum.
7. If the checksum failed, an error will be returned. If the checksum passed, the result value will be converted to a decimal number and returned.

SHT1x or 7x sensor

To use an SHT1x and 7x sensor, connect the data and clock lines to DIO of your choosing. The sensor should be powered from 3.3 V—a digital IO set to output-high will work. The direction settings can be handled by firmware if desired. 5 V should not be used to power the SHT1x and 7x sensors. Doing so will create a logic level mismatch.

The Enable line control and power line control can be disabled.

1. Disable Enable line control. The enable line will be disabled if set to the same line as the clock or data line.
2. Set power control (optional). Set the SBUS_ALL_POWER_DIONUM register to match the hardware configuration. Power control can be disabled by setting the power line to an invalid number, like 9999.
3. Set the data and clock lines to match the hardware configuration.
4. Read from the temperature or humidity register as desired.

SHT3x (requires T4 firmware v1.0029+, T7 v1.0305+, T8 v1.0017+)

To use an SHT3x sensor, connect the data (SDA) and clock (SCL) lines to DIO of your choosing. The sensor should be powered from 3.3 V—a digital IO set to output-high will work. 5V should not be used to power the SHT3x sensor. If your SHT3x sensor has the address (ADDR) line exposed through a wire, we recommend connecting it to GND.

When either SBUS#_TEMP and/or SBUS#_RH are read, the following steps are performed by firmware:

1. If this is the first read, or if the data or clock line registers have been modified, the power line will be set high and a device acknowledge check will be performed. This is expected to take between 1.5ms and 6ms depending on the clock speed.
2. The SHT3x sensor is instructed to measure temperature and humidity.
3. When the measurement is complete, the result and the checksum are read from the sensor.
4. Firmware verifies the checksum.
5. If the checksum failed, an error will be returned. If the checksum passed, the result value will be converted to a decimal number and returned.

Troubleshooting

Checksum or acknowledgement errors can indicate that the clock speed is too fast for the hardware configuration. Use the SBUS_ALL_CLOCK_SPEED register to reduce the clock speed:

Name	Start Address	Type	Access
SBUS_ALL_CLOCK_SPEED	30278	UINT16	R/W

SBUS_ALL_CLOCK_SPEED
- Address: 30278

Sets the clock speed. The clock is software generated so the resulting frequency is not exact. Valid range is 0-65535. Larger values are faster. 0 is the fastest option and is equivalent to 65536. A value of 0 is ~200 kHz. A value of 65000 is ~9.1 kHz.

- Data type: UINT16 (type index = 0)
- Readable and writable
- Default value: 65000
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0142

Register Listing

For configurations besides the default, use the following registers. See the Examples section below for more explanations. Note that SBUS_ALL_CLOCK_DIONUM and SBUS_ALL_DATA_DIONUM should not typically be used with SHT3x sensors.

SBUS Registers

Name	Start Address	Type	Access
SBUS#(0:22)_DATA_DIONUM	30200	UINT16	R/W

SBUS#(0:22)_DATA_DIONUM
- Starting Address: 30200

This is the DIO# that the external sensor's data line is connected to.

- Data type: UINT16 (type index = 0)
- Readable and writable
- Default value: 0
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Default = FIO0
  - Minimum firmware version: 1.0056
- T4:
  - Default = FIO4

Expanded Names	Addresses
SBUS0_DATA_DIONUM, SBUS1_DATA_DIONUM, SBUS2_DATA_DIONUM, SBUS3_DATA_DIONUM, SBUS4_DATA_DIONUM, SBUS5_DATA_DIONUM, SBUS6_DATA_DIONUM, SBUS7_DATA_DIONUM, SBUS8_DATA_DIONUM, SBUS9_DATA_DIONUM, SBUS10_DATA_DIONUM, SBUS11_DATA_DIONUM, SBUS12_DATA_DIONUM, SBUS13_DATA_DIONUM, SBUS14_DATA_DIONUM, SBUS15_DATA_DIONUM, SBUS16_DATA_DIONUM, SBUS17_DATA_DIONUM, SBUS18_DATA_DIONUM, SBUS19_DATA_DIONUM, SBUS20_DATA_DIONUM, SBUS21_DATA_DIONUM, SBUS22_DATA_DIONUM	30200, 30201, 30202, 30203, 30204, 30205, 30206, 30207, 30208, 30209, 30210, 30211, 30212, 30213, 30214, 30215, 30216, 30217, 30218, 30219, 30220, 30221, 30222

SBUS#(0:22)_CLOCK_DIONUM	30225	UINT16	R/W

SBUS#(0:22)_CLOCK_DIONUM
- Starting Address: 30225

This is the DIO# that the external sensor's clock line is connected to.

- Data type: UINT16 (type index = 0)
- Readable and writable
- Default value: 1
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Default = FIO1
  - Minimum firmware version: 1.0056
- T4:
  - Default = FIO5

Expanded Names	Addresses
SBUS0_CLOCK_DIONUM,	30225,
SBUS1_CLOCK_DIONUM,	30226,
SBUS2_CLOCK_DIONUM,	30227,
SBUS3_CLOCK_DIONUM,	30228,
SBUS4_CLOCK_DIONUM,	30229,
SBUS5_CLOCK_DIONUM,	30230,
SBUS6_CLOCK_DIONUM,	30231,
SBUS7_CLOCK_DIONUM,	30232,
SBUS8_CLOCK_DIONUM,	30233,
SBUS9_CLOCK_DIONUM,	30234,
SBUS10_CLOCK_DIONUM,	30235,
SBUS11_CLOCK_DIONUM,	30236,
SBUS12_CLOCK_DIONUM,	30237,
SBUS13_CLOCK_DIONUM,	30238,
SBUS14_CLOCK_DIONUM,	30239,
SBUS15_CLOCK_DIONUM,	30240,
SBUS16_CLOCK_DIONUM,	30241,
SBUS17_CLOCK_DIONUM,	30242,
SBUS18_CLOCK_DIONUM,	30243,
SBUS19_CLOCK_DIONUM,	30244,
SBUS20_CLOCK_DIONUM,	30245,
SBUS21_CLOCK_DIONUM,	30246,
SBUS22_CLOCK_DIONUM	30247

SBUS_ALL_DATA_DIONUM	30275	UINT16	R/W

SBUS_ALL_DATA_DIONUM
- Address: 30275

A write to this global parameter sets all SBUS data line registers to the same value. A read will return the correct setting if all channels are set the same, but otherwise will return 0xFF.

- Data type: UINT16 (type index = 0)
- Readable and writable
- Default value: 0
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0056

SBUS_ALL_CLOCK_DIONUM	30276	UINT16	R/W

SBUS_ALL_CLOCK_DIONUM
- Address: 30276

A write to this global parameter sets all SBUS clock line registers to the same value. A read will return the correct setting if all channels are set the same, but otherwise will return 0xFF.

- Data type: UINT16 (type index = 0)
- Readable and writable
- Default value: 1
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0056

SBUS_ALL_POWER_DIONUM	30277	UINT16	R/W

SBUS_ALL_POWER_DIONUM
- Address: 30277

Sets the power line. This DIO is set to output-high upon any read of SBUS#_TEMP or SBUS#_RH. Default is FIO6 for the T4 and FIO2 for the T7. An FIO line can power up to 4 sensors while an EIO/CIO/MIO line or DAC line can power up to 20 sensors. Set to 9999 to disable. To use multiple power lines, use a DAC line for power, or otherwise control power yourself, set this to 9999 and then control power using writes to normal registers such as FIO5, EIO0, or DAC0.

- Data type: UINT16 (type index = 0)
- Readable and writable
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Default value: 2
  - Minimum firmware version: 1.0056
- T4:
  - Default value: 6

Examples

SHT3x sensors (requires T4 firmware v1.0029+, T7 v1.0305+, T8 v1.0017+):

SHT3x sensors should not typically share data and clock lines, so only one sensor should be connected with default configurations. In this example we connect the wires from a probe to the lines specified by the default configuration:

T7/T8	T4	SHT3x
GND	GND	Ground (black)
DIO0 (FIO0)	DIO4 (FIO4)	Data (SDA)
DIO1 (FIO1)	DIO5 (FIO5)	Clock (SCL)
DIO2 (FIO2)	DIO6 (FIO6)	Power (red)

You can then read from SBUS0_TEMP and SBUS0_RH for the probe without writing any configuration values. In LJLogM, for example, just put the desired register name in any row. A read from SBUS0_TEMP will return the temperature, and a read from SBUS0_RH will return the humidity.

To add additional sensors an SBUS#_DATA_DIONUM and SBUS#_CLOCK_DIONUM register should be configured for each sensor. Each new probe should be connected to their own set of DIO. One exception is that any DIO set to output high or a DAC output set to 3.3V can easily power multiple sensors. For example, say you add two more sensors connected as shown in

the table below:

T4/T7/T8	SHT3x
GND	Ground (black) ProbeB, ProbeC
DAC0	Power (red) ProbeB, ProbeC
DIO8 (EIO0)	Data (SDA) ProbeB
DIO9 (EIO1)	Clock (SCL) ProbeB
DIO10 (EIO2)	Data (SDA) ProbeC
DIO11 (EIO3)	Clock (SCL) ProbeC

You could then set SBUS1_DATA_DIONUM to 8 (DIO8) and SBUS1_CLOCK_DIONUM to 9 (DIO9). Reading SBUS1_TEMP and SBUS1_RH will return the temperature and humidity from ProbeB. You could set SBUS2_DATA_DIONUM to 10 (DIO10) and SBUS2_CLOCK_DIONUM to 11 (DIO11). Reading SBUS2_TEMP and SBUS2_RH will return the temperature and humidity from ProbeC.

EI-1050 probes using default configuration:

The EI-1050 has an enable line that allows multiple probes to use the same pair of data/clock lines. In this example we connect the wires from each probe to the lines specified by the default config:

T7/T8	T4	EI-1050
GND	GND	Ground (black)
DIO0 (FIO0)	DIO4 (FIO4)	Data (green)
DIO1 (FIO1)	DIO5 (FIO5)	Clock (white)
DIO2 (FIO2)	DIO6 (FIO6)	Power (red)

FIO lines can only power 4 EI-1050 probes, so that is the limitation on number of probes using the default config. We can now connect the enable line from each probe to any DIO we want. Let's use:

T7/T8	T4	EI-1050
DIO3 (FIO3)	DIO7 (FIO7)	Enable ProbeA (brown)
DIO4 (FIO4)	DIO8 (EIO0)	Enable ProbeB (brown)
DIO5 (FIO5)	DIO9 (EIO1)	Enable ProbeC (brown)
DIO6 (FIO6)	DIO10 (EIO2)	Enable ProbeD (brown)

You can now read from SBUS#_TEMP and SBUS#_RH for each probe without writing any config values. In LJLogM, for example, just put the desired register name in any row. For the T4, a

read from SBUS8_TEMP will return the temperature from ProbeB, and a read from SBUS9_RH will return the humidity from ProbeC. For the T7/T8, a read from SBUS4_TEMP will return the temperature from ProbeB, and a read from SBUS5_RH will return the humidity from ProbeC.

Note that when using multiple probes this way, you might need to read one value from each probe before they will work. By default, digital I/O are set to input, which has a 100k pull-up, so all 4 probes in this example will be enabled at the same time, which will likely result in a read error. At the end of a read, the enable line is set to output-low, so once you do an initial read from each, they will all be disabled and on further reads only one will be enabled at a time.

EI-1050 probes with custom configuration (shared Data/Clock):

Say you connect 2 probes as follows:

GND	Ground (black)
DIO8 (EIO0)	Data (green)
DIO9 (EIO1)	Clock (white)
DIO10 (EIO2)	Power (red)
DIO11 (EIO3)	Enable ProbeA (brown)
DIO12 (EIO4)	Enable ProbeB (brown)

Write the following registers to configure and disable the probes:

```
SBUS_ALL_DATA_DIONUM = 8
SBUS_ALL_CLOCK_DIONUM = 9
SBUS_ALL_POWER_DIONUM = 10
EIO3 = 0
EIO4 = 0
```

You can now read from SBUS11_TEMP/SBUS11_RH for ProbeA values or SBUS12_TEMP/SBUS12_RH for ProbeB values.

EI-1050 with custom configuration (separate Data/Clock):

This example does not use the enable feature of the EI-1050, and thus applies to the SHT1x and SHT7x also.

In this example each probe has its own data and clock lines. Thus the enable line is not needed and we leave both probes enabled all the time. This technique requires 2 DIO per probe, whereas with the shared data and clock lines you need 1 DIO per probe (enable line) plus 2 DIO (data and clock). A couple downsides to sharing data and clock lines are that it can be difficult to get multiple wires in a single screw terminal and that increased wire capacitance can cause communication problems. Our testing has shown that 6 stock (6 ft cable) EI-1050 probes work fine with a single shared pair of data and clock lines.

Say you connect 2 EI-1050s as follows:

GND	Ground ProbeA (black)

DIO8 (EIO0)	Data ProbeA (green)
DIO9 (EIO1)	Clock ProbeA (white)
DAC0	Power ProbeA (red)
DAC0	Enable ProbeA (brown)
GND	Ground ProbeB (black)
EIO2/DIO10	Data ProbeB (green)
EIO3/DIO11	Clock ProbeB (white)
DAC0	Power ProbeB (red)
DAC0	Enable ProbeB (brown)

Since the EI-1050 enable lines are tied to power they will always be enabled. We can do that because we have assigned both probes dedicated DIO for data and clock.

In this example we use DAC0 to provide power for the sensors. The automatic power control is only supported on digital I/O lines, so we disable automatic power control and manually set DAC0 to 3.3 volts. Alternatively we could use a digital line to power the sensors, with or without automatic power control.

Write the following registers to configure and power the probes:

```
SBUS8_DATA_DIONUM = 8       // Automatic enable control disabled since DATA line is same as SBUS#.
SBUS8_CLOCK_DIONUM = 9
SBUS10_DATA_DIONUM = 10     // Automatic enable control disabled since DATA line is same as
SBUS#.
SBUS10_CLOCK_DIONUM = 11
SBUS_ALL_POWER_DIONUM = 9999 // Disable automatic power control.
DAC0 = 3.3                  // Power for both probes.
```

You can now read from SBUS8_TEMP/SBUS8_RH for ProbeA values or SBUS10_TEMP/SBUS10_RH for ProbeB values.

Note that the "#" in the register names above can be about anything you want. Say for ProbeB you instead did:

```
SBUS7_DATA_DIONUM = 10
SBUS7_CLOCK_DIONUM = 11
```

Now if you read SBUS7_TEMP/SBUS7_RH, the LabJack will use EIO2/3 to talk to the sensor. A possible problem, though, is that the LabJack will also control FIO7 as an enable even though FIO7 has nothing to do with ProbeB. It will set FIO7 to output-high, talk to the sensor, and then set FIO7 to output-low. The way to prevent control of an enable line is to use a "#" that is the same as the data or clock line.

SHT1x or SHT7x sensor using default configuration:

This example was made for the T7/T8. To adapt it for the T4 simply change the line numbers 0-2 to 4-6.

In this example we connect the 4 connections from the raw Sensirion sensor to the lines specified by the default config:

GND	Ground (black)
DIO0 (FIO0)	Data (green)
DIO1 (FIO1)	Clock (white)
DIO2 (FIO2)	Power (red)

Note that the SHT7x datasheet shows an added 10k pull-up resistor from Data to Power. The LabJack has an internal 100k pull-up that usually works, but some applications might need the stronger 10k pull-up (FIO0 to FIO2) and perhaps even a capacitor from Clock to GND. The filter cap should be near the sensor pins, not the LabJack terminals. Suggested value is equal to or less than `1 / (2 * pi * f * R)`, where `f` is the clock frequency and `R` is the source impedance—which is dominated in this case by the DIO (use 550 ohms for FIO and 180 ohms for EIO). 1nF or 10nF should be good for any DIO at the default SBUS clock frequency of 9100 Hz.

You can now read from SBUS0_TEMP & SBUS0_RH without writing any config values. In LJLogM, for example, just put the desired register name in any row. The SHT71 does not have an enable, so we set "#" equal to the data line (0) or clock line (1) which is a signal to the T-series device to not control an enable line.

Since the raw SHT sensors do not have an enable, each sensor must have its own Data and Clock lines.

If an SHT sensor is not working at this point, an oscilloscope or logic analyzer will likely be required to troubleshoot.

13.6 1-Wire [T-Series Datasheet]

Overview

1-Wire is a serial protocol that uses only one data line. Multiple devices can be connected to a single 1-Wire bus and are differentiated using a unique 64-bit number referred to as ROM.

1-Wire is considered an advanced topic. A good knowledge of the protocol is recommended. Troubleshooting may require a logic analyzer or oscilloscope.

The 1-Wire app-note is a good place to start.

1-Wire Support

T-series devices can send and receive data over the 1-Wire bus. The below sections cover the necessary hardware setup considerations and describe how to use the various operations that can be performed over the 1-Wire bus.

Lua scripting is often convenient for serial applications. For example, a script can run the serial communication at a specified interval, and put the results in USER_RAM registers. The host software can read from the USER_RAM registers when convenient. This puts the complications of serial communication in a script running on the T-series device. The Lua examples contain several serial communication examples.

Hardware

Devices on the 1-Wire bus need to be connected to GND, to Vs, and to the data line DQ. DQ also needs a pullup resister of 2.2-4.7 kΩ to Vs.

FIO lines cannot be used for 1-Wire. They have too much impedance, which prevents the signal from reaching the logic thresholds.

T-series devices supports a DPU (dynamic pull up). A dynamic pull up uses an external circuit such as a transistor to provide extra power to the DQ line at proper times. This can be helpful if the line is large or you are using parasitic power.

Device Control Basics

- All T-series device features are controlled by reading and writing Modbus TCP registers via Modbus TCP (either directly or through our LJM library).
- We have register descriptions throughout documentation detailing relevant register names, starting addresses, types, and access permissions (read/write).

- See Section 3.0 Communication for other detailed communication information.

Configuration

ONEWIRE_DQ_DIONUM: This is the DIO line to use for the data line, DQ.
ONEWIRE_DPU_DIONUM: This is the DIO line to use for the dynamic pullup control.
ONEWIRE_OPTIONS: A bit-mask for controlling operation details:

- bit 0: Reserved, write 0.
- bit 1: Reserved, write 0.
- bit 2: DPU Enable. Write 1 to enable the dynamic pullup.
- bit 3: DPU Polarity. Write 1 to set the active state as high, 0 to set the active state as low.

ONEWIRE_FUNCTION: This controls how the ROM address of 1-Wire devices will be used.
ONEWIRE_NUM_BYTES_TX: The number of bytes to transmit to the device. Has no affect when the ROM function is set to Search or Read.
ONEWIRE_NUM_BYTES_RX: The number of bytes to read from the device. Has no affect when the ROM function is set to Search or Read.
ONEWIRE_ROM_MATCH_H: The upper 32-bits of the ROM of the device to attempt to connect to when using the Match ROM function.
ONEWIRE_ROM_MATCH_L: The lower 32-bits of the ROM of the device to attempt to connect to when using the Match ROM function.
ONEWIRE_PATH_H: Upper 32-bits of the search path.
ONEWIRE_PATH_L: Lower 32-bits of the search path.

ROM Functions

0xF0: Search – This function will read the ROM of one device on the bus. The ROM found is placed in ONEWIRE_SEARCH_RESULT and if other devices were detected the branch bits will be set in ONEWIRE_ROM_BRANCHS_FOUND.

0xCC: Skip – This function will skip the ROM addressing step. For this to work properly only one device may be connected to the bus.

0x55: Match – When using this function data will be sent to and read from a device whose ROM matches the ROM loaded into the ONEWIRE_ROM_MATCH registers.

0x33: Read – Reads the ROM of the connected device. For this to work properly only one device may be connected to the bus.

Sending data

When using the Match or Skip Rom functions data can be sent to the device. To do so, set the

number of bytes to send by writing to ONEWIRE_NUM_BYTES_RX and write the data to ONEWIRE_DATA_RX.

Reading data

When using the Match or Skip Rom functions data can be read from the device. To do so, set the number of bytes to send by writing to ONEWIRE_NUM_BYTES_TX and write the data to ONEWIRE_DATA_TX.

Example

Configure the T-series device's 1-Wire interface, and obtain a temperature reading from a DS18B22.

Configuration: Write the common configuration that will not change; the DQ line, DPU, and options. For this example we will use EIO6 (14) as DQ, and the DPU will be left disabled.

```
ONEWIRE_DQ_DIONUM = 14
ONEWIRE_DPU_DIONUM = 0
ONEWIRE_OPTIONS = 0
```

Read ROM: The 64-bit ROM can be read from the device using the Read ROM function if it is the only device on the bus.

```
ONEWIRE_FUNCTION =
0x33
ONEWIRE_GO = 1
```

The T-series device will read the ROM from the connected device and place it in the ONEWIRE_SEARCH_RESULT registers. The ONEWIRE_SEARCH_RESULT_H register will have the upper 32-bits of the ROM read, ONEWIRE_SEARCH_RESULT_L will have the lower 32-bits of the ROM read. This test resulted in a read ROM code of 0x1D000005908D4728; 0x1D000005 was read from ONEWIRE_SEARCH_RESULT_H and 0x908D4728 was read from ONEWIRE_SEARCH_RESULT_L.

Search for ROM: If there is more than one device on the bus the search function can be used to find the ROM of one of the devices. Note that this method does not provide any information about which device has the ROM discovered.

```
ONEWIRE_PATH = 0
ONEWIRE_FUNCTION = 0xF0
ONEWIRE_GO = 1
```

The T-series device will perform the 1-Wire search function. If a ROM is found it will be placed in ONEWIRE_ROM_SEARCH_RESULT and any branches detected will be indicated in ONEWIRE_BRANCHES. The ONEWIRE_PATH field can be used to direct the LabJack to take a different path in subsequent searches.

Results:
ROM - 0x1D000005908D4728

Branches - 0x00000000000002

Now repeat the search with path set to 2.

Results:
ROM - 0xFF00000024AD2C22
Branches - 0x00000000000002

The search can be repeated to find the ROM codes of all devices on the bus.

Write start conversion command to the device:
Do instruct the sensor to start a reading we need to match the device's ROM and send one data byte. The data byte contains the instruction 0x44.

```
ONEWIRE_FUNCTION = 0x55
ONEWIRE_ROM = 0x1D000005908D4728
ONEWIRE_NUM_BYTES_TX = 1
ONEWIRE_DATA_TX = [0x44]
ONEWIRE_GO = 1
```

The sensor will now start a conversion. Depending on the sensor and it's settings up to 500 ms may be needed to complete the conversion.

Read conversion result from the device: After a conversion has been complete we can begin the reading process. This time we need to write the read instruction which is 0x44 and then read 9 bytes of data.

```
ONEWIRE_FUNCTION = 0x55
ONEWIRE_ROM = 0x1D000005908D4728
ONEWIRE_NUM_BYTES_TX = 1
ONEWIRE_NUM_BYTES_RX = 9
ONEWIRE_DATA_TX = [0xBE]
ONEWIRE_GO = 1
```

We can now read the 9 bytes from ONEWIRE_DATA_RX:
0x6A, 0x0A, 0x00, 0x00, 0x24, 0xAD, 0x2C, 0x22, 0x00

The 9 bytes contain the binary reading, a checksum, and some other information about the device. The devices used was set to 12-bit resolution so the conversion is 0.0625ºC/bit. The binary result is data[0] + data[1]*256. The binary temperature reading is 1*256 + 0x6A = 256 + 106 = 362. To convert that to ºC multiply by 0.0625. So the final temperature is 22.6 ºC.

Register Listing

1-Wire Registers

Name	Start Address	Type	Access
ONEWIRE_DQ_DIONUM	5300	UINT16	R/W

ONEWIRE_DQ_DIONUM
- Address: 5300

The data-line DIO number.

- Data type: UINT16 (type index = 0)
- Readable and writable
- Default value: 0
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0056

ONEWIRE_DPU_DIONUM	5301	UINT16	R/W

ONEWIRE_DPU_DIONUM
- Address: 5301

The dynamic pullup control DIO number.

- Data type: UINT16 (type index = 0)
- Readable and writable
- Default value: 0
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0056

ONEWIRE_OPTIONS	5302	UINT16	R/W

ONEWIRE_OPTIONS
- Address: 5302

Controls advanced features. Value is a bitmask.
bit 0: reserved,
bit 1: reserved,
bit 2: 1=DPU Enabled 0=DPU Disabled,
bit 3: DPU Polarity 1=Active state is high, 0=Active state is low
(Dynamic Pull-Up)

- Data type: UINT16 (type index = 0)
- Readable and writable
- Default value: 0
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0056

ONEWIRE_FUNCTION	5307	UINT16	R/W

ONEWIRE_FUNCTION
- Address: 5307

Set the ROM function to use.
0xF0=Search,
0xCC=Skip,
0x55=Match,
0x33=Read.

- Data type: UINT16 (type index = 0)
- Readable and writable
- Default value: 0
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0056

ONEWIRE_NUM_BYTES_TX	5308	UINT16	R/W

ONEWIRE_NUM_BYTES_TX
- Address: 5308

Number of data bytes to be sent.

- Data type: UINT16 (type index = 0)
- Readable and writable
- Default value: 0
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0056

ONEWIRE_NUM_BYTES_RX	5309	UINT16	R/W

ONEWIRE_NUM_BYTES_RX
- Address: 5309

Number of data bytes to be received.

- Data type: UINT16 (type index = 0)
- Readable and writable
- Default value: 0
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0056

ONEWIRE_GO	5310	UINT16	W

ONEWIRE_GO
- Address: 5310

Instructs the T7 to perform the configured 1-wire transaction.

- Data type: UINT16 (type index = 0)
- Write-only
- Default value: 0
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0056

ONEWIRE_ROM_MATCH_H	5320	UINT32	R/W

ONEWIRE_ROM_MATCH_H
- Address: 5320

Upper 32-bits of the ROM to match.

- Data type: UINT32 (type index = 1)
- Readable and writable
- Default value: 0
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0056

ONEWIRE_ROM_MATCH_L	5322	UINT32	R/W

ONEWIRE_ROM_MATCH_L
- Address: 5322

Lower 32-bits of the ROM to match.

- Data type: UINT32 (type index = 1)
- Readable and writable
- Default value: 0
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0056

ONEWIRE_ROM_BRANCHS_FOUND_H	5332	UINT32	R

ONEWIRE_ROM_BRANCHS_FOUND_H
- Address: 5332

Upper 32-bits of the branches detected during a search.

- Data type: UINT32 (type index = 1)
- Read-only
- Default value: 0
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0056

ONEWIRE_ROM_BRANCHS_FOUND_L	5334	UINT32	R

ONEWIRE_ROM_BRANCHS_FOUND_L
- Address: 5334

Lower 32-bits of the branches detected during a search.

- Data type: UINT32 (type index = 1)
- Read-only
- Default value: 0
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0056

ONEWIRE_SEARCH_RESULT_H	5328	UINT32	R

ONEWIRE_SEARCH_RESULT_H
- Address: 5328

Upper 32-bits of the search result.

- Data type: UINT32 (type index = 1)
- Read-only
- Default value: 0
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0056

ONEWIRE_SEARCH_RESULT_L	5330	UINT32	R

ONEWIRE_SEARCH_RESULT_L
- Address: 5330

Lower 32-bites of the search result.

- Data type: UINT32 (type index = 1)
- Read-only
- Default value: 0
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0056

ONEWIRE_PATH_H	5324	UINT32	R/W

ONEWIRE_PATH_H
- Address: 5324

Upper 32-bits of the path to take during a search.

- Data type: UINT32 (type index = 1)
- Readable and writable
- Default value: 0
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0056

ONEWIRE_PATH_L	5326	UINT32	R/W

ONEWIRE_PATH_L
- Address: 5326

Lower 32-bits of the path to take during a search.

- Data type: UINT32 (type index = 1)
- Readable and writable
- Default value: 0
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0056

ONEWIRE_DATA_TX	5340	BYTE	W

ONEWIRE_DATA_TX
- Address: 5340

Data to be transmitted over the 1-wire bus. This register is a buffer.

- Data type: BYTE (type index = 99)
- Write-only
- Default value: 0
- This register is a Buffer Register
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0056

ONEWIRE_DATA_RX	5370	BYTE	R

ONEWIRE_DATA_RX
- Address: 5370

Data received over the 1-wire bus. This register is a buffer. Underrun behavior - buffer is static, old data will fill the extra locations, firmware 1.0225 changes this to read zeros.

- Data type: BYTE (type index = 99)
- Read-only
- Default value: 0
- This register is a Buffer Register
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0056

13.7 Asynchronous Serial [T-Series Datasheet]

Overview

The T-Series devices have universal asynchronous receiver-transmitter (UART) functionality available that supports 3.3V logic level (CMOS/TTL) asynchronous (asynch) serial communication. The TX (transmit) and RX (receive) lines can appear on any digital I/O. Baud rates up to 38400 are supported, but the device's processor is heavily loaded at that rate. The number of data bits, number of stop bits, and parity are all controllable.

Asynchronous (UART) vs. RS-232

The T-series asynchronous support and the RS-232 standard are the same in terms of timing and protocol, but different in terms of electrical specifications. Connection to an RS-232 device will require a converter chip such as the MAX233, which inverts the logic and shifts the voltage levels. On a T-series device, a low is 0 volts (inputs recognize 0.0 to 0.5) and a high (1) is 3.3 volts (inputs recognize 2.64 to 5.8 volts). With RS-232, a low (0) is 3 to 25 volts and a high (1) is -3 to -25 volts; RS-232 has unique voltage levels and is inverted.

Lua Scripting

Lua scripting is often convenient for serial applications. For example, you might write a script that does the serial communication to get a new reading from the serial device once per second, and puts that reading in a USER_RAM register. This puts the complications of serial communication in a script running on the T-series device itself, and then the host software can just do a simple read of the USER_RAM register when convenient. We have many serial examples available for Lua scripting.

A direct connection to a serial device is preferable

This serial link is not an alternative to the USB/Ethernet/WiFi connection. Rather, the host application will write/read data to/from the T-series device over USB/Ethernet/WiFi, and the T-series device communicates with some other device using the serial protocol. Using this serial protocol is considered an advanced topic. A good knowledge of the protocol is recommended, and a logic analyzer or oscilloscope might be needed for troubleshooting.

If it is practical to run a cable directly from the host computer to the serial device, that is usually a better than putting the T-series device inbetween. Use a standard USB<=>RS-232 adapter/converter/dongle (or RS-485 or RS-422).

Multiple asynchronous ports on a single LabJack

The asynchronous feature can only be enabled on one pair of pins at a time, and to be more specific only one RX pin can read data at a time. When the asynchronous feature is enabled on a pair of pins, a buffer is set up and the RX pin reads any data that comes in and stores it in the buffer. This is useful for devices that spontaneously send out data where all that data is wanted all the time. Most serial devices, however, act in a command-response manner where the LabJack sends a command that requests a reading and the device responds with the reading. For these it is easy to do multiple ... just re-do the configuration writes whenever communication is desired on different pins.

How-To

Device Control Basics

- All T-series device features are controlled by reading and writing Modbus TCP registers via Modbus TCP (either directly or through our LJM library).
- We have register descriptions throughout documentation detailing relevant register names, starting addresses, types, and access permissions (read/write).
- See Section 3.0 Communication for other detailed communication information.

Process

1. Initial Configuration
2. Transmit Data
3. Receive Data
4. Debugging data parity errors (if enabled)

Initial Configuration

Several registers need to be written to in order to configure the T-Series device for Asynch communication.

- TX/RX data lines (ASYNCH_TX_DIONUM, ASYNCH_RX_DIONUM)
- Baud rate configuration (ASYNCH_BAUD)
- Configure RX buffer size (ASYNCH_RX_BUFFER_SIZE_BYTES)
- Configure number of bits, number of stop bits, and the parity. (ASYNCH_NUM_DATA_BITS, ASYNCH_NUM_STOP_BITS, ASYNCH_PARITY)

After configuring the various registers, the ASYNCH feature should be enabled by writing a 1 to (ASYNCH_ENABLE).

Asynchronous Serial Configuration Registers

Name	Start Address	Type	Access
ASYNCH_TX_DIONUM	5410	UINT16	R/W

ASYNCH_TX_DIONUM
- Address: 5410

The DIO line that will transmit data. (TX)

- Data type: UINT16 (type index = 0)
- Readable and writable
- Default value: 0

ASYNCH_RX_DIONUM	5405	UINT16	R/W

ASYNCH_RX_DIONUM
- Address: 5405

The DIO line that will receive data. (RX)

- Data type: UINT16 (type index = 0)
- Readable and writable
- Default value: 0

ASYNCH_BAUD	5420	UINT32	R/W

ASYNCH_BAUD
- Address: 5420

The symbol rate that will be used for communication. 9600 is typical. Up to 38400 works, but heavily loads the T7's processor.

- Data type: UINT32 (type index = 1)
- Readable and writable
- Default value: 0

ASYNCH_RX_BUFFER_SIZE_BYTES	5430	UINT16	R/W

ASYNCH_RX_BUFFER_SIZE_BYTES
- Address: 5430

Number of bytes to use for the receiving buffer. Max is 2048. 0 = 200

- Data type: UINT16 (type index = 0)
- Readable and writable
- Default value: 0

ASYNCH_NUM_DATA_BITS	5415	UINT16	R/W

ASYNCH_NUM_DATA_BITS
- Address: 5415

The number of data bits per frame. 0-8, 0=8.

- Data type: UINT16 (type index = 0)
- Readable and writable
- Default value: 0

ASYNCH_NUM_STOP_BITS	5455	UINT16	R/W

ASYNCH_NUM_STOP_BITS
- Address: 5455

The number of stop bits. Values:
0 = zero stop bits,
1 = one stop bit,
2 = two stop bits.

- Data type: UINT16 (type index = 0)
- Readable and writable
- Default value: 0

ASYNCH_PARITY	5460	UINT16	R/W

ASYNCH_PARITY
- Address: 5460

Parity setting:
0=none,
1=odd,
2=even.

- Data type: UINT16 (type index = 0)
- Readable and writable
- Default value: 0

ASYNCH_ENABLE	5400	UINT16	R/W

ASYNCH_ENABLE
- Address: 5400

1 = Turn on Asynch. Configures timing hardware, DIO lines and allocates the receiving buffer.

- Data type: UINT16 (type index = 0)
- Readable and writable
- Default value: 0
- This register uses system RAM. The maximum RAM is 64KB. For more information, see 4.4 RAM

Transmit Data

In order to transmit data a user must do the following:

1. Configure the number of bytes that needs to be sent (ASYNCH_NUM_BYTES_TX)
2. Send data to the T-Series device using the LJM_eWriteNameArray function (ASYNCH_DATA_TX)
3. Write a 1 to the "GO" register (ASYNCH_TX_GO)
   note: The process of writing a 1 to the GO register instructs the T-Series device to transmit the buffered data via the TX line.

Asynchronous Serial Data Transmission Registers

Name	Start Address	Type	Access
ASYNCH_NUM_BYTES_TX	5440	UINT16	R/W

ASYNCH_NUM_BYTES_TX
- Address: 5440

The number of bytes to be transmitted after writing to GO. Max is 256.

- Data type: UINT16 (type index = 0)
- Readable and writable
- Default value: 0
- This register uses system RAM. The maximum RAM is 64KB. For more information, see 4.4 RAM

ASYNCH_DATA_TX	5490	UINT16	W

ASYNCH_DATA_TX
- Address: 5490

Write data to be transmitted here. This register is a buffer.

- Data type: UINT16 (type index = 0)
- Write-only
- Default value: 0
- This register is a Buffer Register

ASYNCH_TX_GO	5450	UINT16	W

ASYNCH_TX_GO
- Address: 5450

Write a 1 to this register to initiate a transmission.

- Data type: UINT16 (type index = 0)
- Write-only
- Default value: 0

Receive Data

T-Series devices buffer received Asynch data up to the size defined in the configuration step when writing to the register "ASYNCH_RX_BUFFER_SIZE_BYTES". The usual method for reading data from the buffer is to do the following:

1. Read how many bytes of information have been received by the device (ASYNCH_NUM_BYTES_RX)
2. Read data from the T-Series device RX buffer using the LJM_eReadNameArray function (ASYNCH_DATA_RX)
   note: When ever possible, it is recommended to read an even number of bytes from the

DATA_RX buffer.

Asynchronous Serial Data Receiving Registers

Name	Start Address	Type	Access
ASYNCH_NUM_BYTES_RX	5435	UINT16	R

ASYNCH_NUM_BYTES_RX
- Address: 5435

The number of data bytes that have been received.

- Data type: UINT16 (type index = 0)
- Read-only
- Default value: 0

ASYNCH_DATA_RX	5495	UINT16	R

ASYNCH_DATA_RX
- Address: 5495

Read received data from here. This register is a buffer. Underrun behavior - fill with zeros.

- Data type: UINT16 (type index = 0)
- Read-only
- Default value: 0
- This register is a Buffer Register

Debugging Data Parity Errors

Asynchronous Serial Data Parity Register

Name	Start Address	Type	Access
ASYNCH_NUM_PARITY_ERRORS	5465	UINT16	R/W

ASYNCH_NUM_PARITY_ERRORS
- Address: 5465

The number of parity errors that have been detected. Cleared when UART is enabled. Can also be cleared by writing 0.

- Data type: UINT16 (type index = 0)
- Readable and writable
- Default value: 0

Examples

For performing asynchronous communication from a computer, see the LJM C examples or the LJM LabVIEW examples.

For performing asynchronous communication on device, see the Lua scripting examples.

14.0 Analog Inputs [T-Series Datasheet]

Overview

Basics: An analog input (commonly referred to as AIN or AI) uses an analog-to-digital converter (ADC) to convert a voltage level into a digital value. LabJack T-series devices have multiple analog inputs.

Common Uses: For information on measuring various analog signals such as typical analog sensors, measuring small voltages, measuring current (4-20mA), and measuring resistance see the Analog Input App Notes. There are also application specific app notes such as for temperature sensors and thermocouples.

Device Control Basics:

- All T-series device features are controlled by reading and writing Modbus TCP registers via Modbus TCP (either directly or through our LJM library).
- We have register descriptions throughout documentation detailing relevant register names, starting addresses, types, and access permissions (read/write).
- Most AIN registers have a FLOAT32 type meaning each register requires 32 bits to represent the associated data. Each entry in the device address space holds 16-bits of data, so most AIN registers take up two address spaces such as 0 and 1 (AIN0 register). See the Modbus Map for related information.
- See Section 3.0 Communication for other detailed communication information.

Configuration: T-series AIN readings can be configured. See below for information

AIN Extended Features: T-series AIN Extended Features simplify operations such as:

- Reading thermocouples and thermistors
- Calculating RMS, RTD, average/max/min, average/threshold, and circuit element resistance

Extended Channels - T7 Only: The T7's Extended Channels range provides extra AIN channels.

Table 14.0-1 Analog Input Channel Overview

	Built-in Analog Inputs																Extended Channels
AIN	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	AIN (48-127)
T4	AIN (0-3) High-Voltage ±10V				AIN (4-11) Low-Voltage 0-2.5V								N/A				N/A
T7	AIN (0-13) High-Voltage ±10V Options: SE/Diff, Gains (1000x, 100x, 10x, 1)														Special Channels		AIN (48-127) Mux80
T8	AIN (0-7) ±11 V to ±0.018 V								N/A								N/A

Subsections

AIN Summary By Device

T4

Analog Inputs:
4 high-voltage (AIN0-AIN3)
8 low-voltage (AIN4-AIN11)
Voltage Ranges:
±10 V or 0-2.5 V (Appendix A-3-1-1 and Appendix A-3-1-3)
Resolution:
12-bit
Max Data Rate:
40000 samples/second in stream mode (Appendix A-1)
Sampling Modes:
Single-ended

T7

Analog Inputs:
14 (AIN0-AIN13)
Voltage Ranges:
±10 V, ±1 V, ±0.1 V, ±0.01 V (Appendix A-3-2-1 and Appendix A-3-2-3)
Resolution:
T7: 16-bit
T7-Pro: 24-bit
Effective Resolution:
T7: 16 to 19 bit*
T7-Pro: 16 to 22 bit*
*At gain 1x. For more details, see Appendix A-3-2-2.
Max Data Rate:
100000 samples/second in stream mode (Appendix A-1)
Sampling Modes:
Configurable as single-ended or differential
Extended Channels:
The number of analog inputs can be extended to 84 with a MUX80 (AIN48-AIN127)

T8

Analog Inputs:
8 (AIN0-AIN7)
Voltage Ranges:
±11 V, ±9.6 V, ±4.8 V, ±2.4 V, ±1.2 V, ±0.6 V, ±0.3 V, ±0.15 V, ±0.075 V, ±0.036 V,

±0.018 V
(Appendix A-3-3-1 and Appendix A-3-3-3)

Resolution:
24-bit

Effective Resolution:
15 to 22 bit
For more details, see Appendix A-3-3-2.

Max Data Rate:
400000 samples/second (in stream mode (Appendix A-1)

Sampling Modes:
Simultaneous and differential

Isolation:
1000V

Available AIN Channels

Each T-series device exposes:

- Some AINs on the screw terminals.
- Additional AINs on a connector (either a DB15 or a DB37).

T4

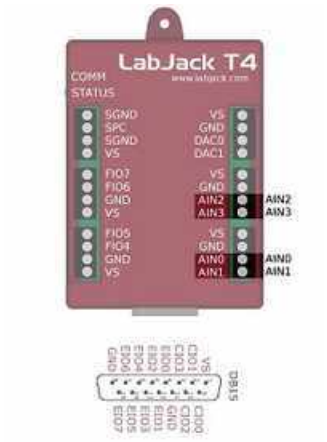


Figure 14.0-1 T4 Dedicated Analog Inputs

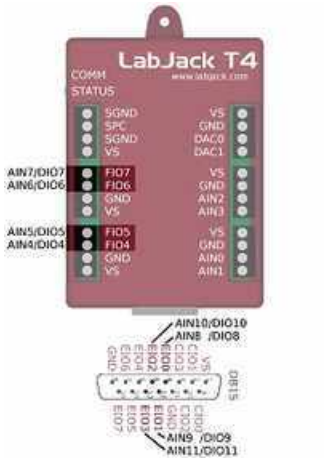


Figure 14.0-2 T4 Flexible I/O

The LabJack T4 has up to 12 built-in analog inputs, readable as AIN0-AIN11:

- AIN0-AIN3 are dedicated ±10 volt inputs:
- Available on screw terminals AIN0-AIN3.
- Always ±10 volt analog inputs.
- AIN4-AIN7 are flexible 0 to +2.5 volt inputs:
- Available on screw terminals FIO4-FIO7.
- Configurable to be digital inputs/outputs. See "Flexible I/O" below.
- AIN8-AIN11 are flexible 0 to +2.5 volt inputs:
- Available on the DB15 connector.
- Configurable to be digital inputs/outputs. See "Flexible I/O" below.

T4 AIN Channel Registers

Name	Start Address	Type	Access
AIN#(0:11)	0	FLOAT32	R

AIN#(0:11)
- Starting Address: 0

Returns the voltage of the specified analog input.

- Data type: FLOAT32 (type index = 3)
- Read-only
- This register may be streamed

Expanded Names	Addresses
AIN0, AIN1, AIN2, AIN3, AIN4, AIN5, AIN6, AIN7, AIN8, AIN9, AIN10, AIN11	0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22

AIN#(0:11)_BINARY	50000	UINT32	R

AIN#(0:11)_BINARY
- Starting Address: 50000

Returns the 24-bit binary representation of the specified analog input. If you prefer 16-bit representation, simply divide this by 256. This is for command-response only. Stream always returns binary and LJM applies cal constants, so the LJM config flag LJM_STREAM_AIN_BINARY is used to get binary values.

- Data type: UINT32 (type index = 1)
- Read-only

Expanded Names	Addresses
AIN0_BINARY, AIN1_BINARY, AIN2_BINARY, AIN3_BINARY, AIN4_BINARY, AIN5_BINARY, AIN6_BINARY, AIN7_BINARY, AIN8_BINARY, AIN9_BINARY, AIN10_BINARY, AIN11_BINARY	50000, 50002, 50004, 50006, 50008, 50010, 50012, 50014, 50016, 50018, 50020, 50022

T7

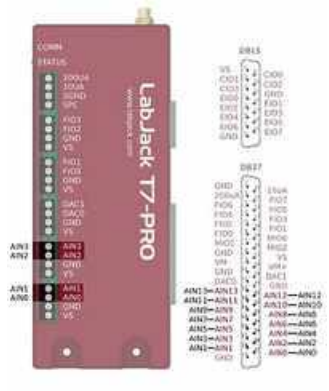


Figure 14.0-3 T7 Analog Inputs

The LabJack T7 has 14 built-in analog inputs, readable as AIN0-13:

- AIN0-AIN3 are available on the screw terminals and on the DB37 connector. See "Duplicated Terminals" below.
- AIN4-AIN13 are available only on the DB37 connector.

T7 AIN Channel Registers

Name	Start Address	Type	Access
AIN#(0:13)	0	FLOAT32	R

AIN#(0:13)
- Starting Address: 0

Returns the voltage of the specified analog input.

- Data type: FLOAT32 (type index = 3)
- Read-only
- This register may be streamed

Expanded Names	Addresses
AIN0, AIN1, AIN2, AIN3, AIN4, AIN5, AIN6, AIN7, AIN8, AIN9, AIN10, AIN11, AIN12, AIN13	0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26

AIN#(0:13)_BINARY	50000	UINT32	R

AIN#(0:13)_BINARY
- Starting Address: 50000

Returns the 24-bit binary representation of the specified analog input. If you prefer 16-bit representation, simply divide this by 256. This is for command-response only. Stream always returns binary and LJM applies cal constants, so the LJM config flag LJM_STREAM_AIN_BINARY is used to get binary values.

- Data type: UINT32 (type index = 1)
- Read-only

Expanded Names	Addresses
AIN0_BINARY, AIN1_BINARY, AIN2_BINARY, AIN3_BINARY, AIN4_BINARY, AIN5_BINARY, AIN6_BINARY, AIN7_BINARY, AIN8_BINARY, AIN9_BINARY, AIN10_BINARY, AIN11_BINARY, AIN12_BINARY, AIN13_BINARY	50000, 50002, 50004, 50006, 50008, 50010, 50012, 50014, 50016, 50018, 50020, 50022, 50024, 50026

In addition to the 14 built-in analog inputs, the T7 has special and extended channels.

- AIN14 is internally connected to an internal temperature sensor.
- AIN15 is internally connected to GND. Useful for measuring noise or looking at offset error.
- AIN16-AIN47 are optional extended channels that can be created with custom analog input muxing circuitry. See Section 14.2 Extended Channels for more information.
- AIN48-AIN127 are extended channels that are available when using a Mux80.

T7 Extended AIN Channel Registers

Name	Start Address	Type	Access
AIN#(48:127)	96	FLOAT32	R

AIN#(48:127)
- Starting Address: 96

Returns the voltage of the specified analog input.

- Data type: FLOAT32 (type index = 3)
- Read-only
- This register may be streamed

Expanded Names	Addresses
AIN48, AIN49, AIN50, AIN51, AIN52, AIN53, AIN54, AIN55, AIN56, AIN57, AIN58, AIN59, AIN60, AIN61, AIN62, AIN63, AIN64, AIN65, AIN66, AIN67, AIN68, AIN69, AIN70, AIN71, AIN72, AIN73, AIN74, AIN75, AIN76, AIN77, AIN78, AIN79, AIN80, AIN81, AIN82, AIN83, AIN84, AIN85, AIN86, AIN87, AIN88, AIN89, AIN90, AIN91, AIN92, AIN93, AIN94, AIN95, AIN96, AIN97, AIN98, AIN99, AIN100, AIN101, AIN102, AIN103, AIN104, AIN105, AIN106, AIN107, AIN108, AIN109, AIN110, AIN111, AIN112, AIN113, AIN114, AIN115, AIN116, AIN117, AIN118, AIN119, AIN120, AIN121, AIN122, AIN123, AIN124, AIN125, AIN126, AIN127	96, 98, 100, 102, 104, 106, 108, 110, 112, 114, 116, 118, 120, 122, 124, 126, 128, 130, 132, 134, 136, 138, 140, 142, 144, 146, 148, 150, 152, 154, 156, 158, 160, 162, 164, 166, 168, 170, 172, 174, 176, 178, 180, 182, 184, 186, 188, 190, 192, 194, 196, 198, 200, 202, 204, 206, 208, 210, 212, 214, 216, 218, 220, 222, 224, 226, 228, 230, 232, 234, 236, 238, 240, 242, 244, 246, 248, 250, 252, 254

AIN#(48:127)_BINARY	50096	UINT32	R

AIN#(48:127)_BINARY
- Starting Address: 50096

Returns the 24-bit binary representation of the specified analog input. If you prefer 16-bit representation, simply divide this by 256. This is for command-response only. Stream always returns binary and LJM applies cal constants, so the LJM config flag LJM_STREAM_AIN_BINARY is used to get binary values.

- Data type: UINT32 (type index = 1)
- Read-only

Expanded Names	Addresses

AIN48_BINARY, AIN49_BINARY,	50096, 50098,
AIN50_BINARY, AIN51_BINARY,	50100, 50102,
AIN52_BINARY, AIN53_BINARY,	50104, 50106,
AIN54_BINARY, AIN55_BINARY,	50108, 50110,
AIN56_BINARY, AIN57_BINARY,	50112, 50114,
AIN58_BINARY, AIN59_BINARY,	50116, 50118,
AIN60_BINARY, AIN61_BINARY,	50120, 50122,
AIN62_BINARY, AIN63_BINARY,	50124, 50126,
AIN64_BINARY, AIN65_BINARY,	50128, 50130,
AIN66_BINARY, AIN67_BINARY,	50132, 50134,
AIN68_BINARY, AIN69_BINARY,	50136, 50138,
AIN70_BINARY, AIN71_BINARY,	50140, 50142,
AIN72_BINARY, AIN73_BINARY,	50144, 50146,
AIN74_BINARY, AIN75_BINARY,	50148, 50150,
AIN76_BINARY, AIN77_BINARY,	50152, 50154,
AIN78_BINARY, AIN79_BINARY,	50156, 50158,
AIN80_BINARY, AIN81_BINARY,	50160, 50162,
AIN82_BINARY, AIN83_BINARY,	50164, 50166,
AIN84_BINARY, AIN85_BINARY,	50168, 50170,
AIN86_BINARY, AIN87_BINARY,	50172, 50174,
AIN88_BINARY, AIN89_BINARY,	50176, 50178,
AIN90_BINARY, AIN91_BINARY,	50180, 50182,
AIN92_BINARY, AIN93_BINARY,	50184, 50186,
AIN94_BINARY, AIN95_BINARY,	50188, 50190,
AIN96_BINARY, AIN97_BINARY,	50192, 50194,
AIN98_BINARY, AIN99_BINARY,	50196, 50198,
AIN100_BINARY, AIN101_BINARY,	50200, 50202,
AIN102_BINARY, AIN103_BINARY,	50204, 50206,
AIN104_BINARY, AIN105_BINARY,	50208, 50210,
AIN106_BINARY, AIN107_BINARY,	50212, 50214,
AIN108_BINARY, AIN109_BINARY,	50216, 50218,
AIN110_BINARY, AIN111_BINARY,	50220, 50222,
AIN112_BINARY, AIN113_BINARY,	50224, 50226,
AIN114_BINARY, AIN115_BINARY,	50228, 50230,
AIN116_BINARY, AIN117_BINARY,	50232, 50234,
AIN118_BINARY, AIN119_BINARY,	50236, 50238,
AIN120_BINARY, AIN121_BINARY,	50240, 50242,
AIN122_BINARY, AIN123_BINARY,	50244, 50246,
AIN124_BINARY, AIN125_BINARY,	50248, 50250,
AIN126_BINARY, AIN127_BINARY	50252, 50254

T8

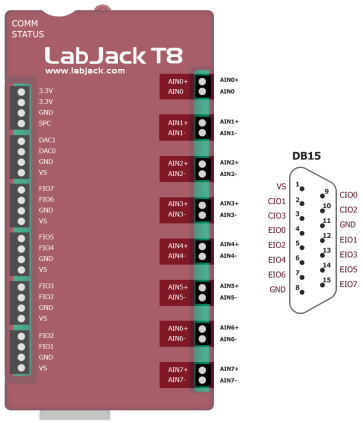


Figure 14.0-4 T8 Analog Inputs

The LabJack T8 has 8 built-in analog inputs. All 8 analog inputs are individually isolated and are sampled simultaneously. The AINs can be read using register names AIN0-7:

- AIN0-AIN7 are available on the screw terminals and on the 2 pin OEM headers on each AIN. See "Duplicated Terminals" below.
- Reading any of AIN0-7 will capture samples for all AIN channels and all TEMPERATURE# channels and vise-versa.
- To get simultaneous readings from multiple AIN and temperature registers, only the first channel should be read as AIN# or TEMPERATURE#. The remaining registers should be read using the appropriate _CAPTURE register. For example, to read the voltages connected to AIN0, AIN1, and AIN2, read the following list of registers:
  [AIN0 (address 0), AIN1_CAPTURE (address 652), AIN2_CAPTURE (address 654)]
  See more information about the TEMPERATURE# registers in the Internal Temp Sensor section.
- If multiple AIN# registers are read in one command rather than using the _CAPTURE registers, each AIN reading will be acquired sequentially rather than simultaneously, and each reading will take 0 to 1/AIN_SAMPLING_RATE_HZ seconds to be acquired. See the AIN Sampling Rate section below for further sampling information.

> Simultaneously sampled analog inputs are handled differently when using stream mode acquisition. See the Simultaneous Sampling stream documentation for further information.

T8 AIN Channel Registers

Name	Start Address	Type	Access
AIN#(0:7)	0	FLOAT32	R

AIN#(0:7)
- Starting Address: 0

Returns the voltage of the specified analog input.

- Data type: FLOAT32 (type index = 3)
- Read-only
- This register may be streamed

Expanded Names	Addresses
AIN0, AIN1, AIN2, AIN3, AIN4, AIN5, AIN6, AIN7	0, 2, 4, 6, 8, 10, 12, 14

AIN#(0:7)_CAPTURE	650	FLOAT32	R

AIN#(0:7)_CAPTURE
- Starting Address: 650

T8 Only. Returns the saved voltage of the specified analog input.

- Data type: FLOAT32 (type index = 3)
- Read-only
- This register may be streamed

Expanded Names	Addresses
AIN0_CAPTURE, AIN1_CAPTURE, AIN2_CAPTURE, AIN3_CAPTURE, AIN4_CAPTURE, AIN5_CAPTURE, AIN6_CAPTURE, AIN7_CAPTURE	650, 652, 654, 656, 658, 660, 662, 664

AIN#(0:7)_BINARY	50000	UINT32	R

AIN#(0:7)_BINARY
- Starting Address: 50000

Returns the 24-bit binary representation of the specified analog input. If you prefer 16-bit representation, simply divide this by 256. This is for command-response only. Stream always returns binary and LJM applies cal constants, so the LJM config flag LJM_STREAM_AIN_BINARY is used to get binary values.

- Data type: UINT32 (type index = 1)
- Read-only

Expanded Names	Addresses
AIN0_BINARY, AIN1_BINARY, AIN2_BINARY, AIN3_BINARY, AIN4_BINARY, AIN5_BINARY, AIN6_BINARY, AIN7_BINARY	50000, 50002, 50004, 50006, 50008, 50010, 50012, 50014

Examples

These examples are primarily for programmatic control. If you intend to use our logging software such as LJLogM or LJStreamM, see our LJLogM Basics Guide.

- To read AIN0 using LJM, read address 0 using a function such as eReadAddress.
- To read AIN1 using LJM, read address 2 using a function such as eReadAddress.

Resolution Index

A higher resolution index results in lower noise and higher effective resolution but increases sample times.

Resolution Index Registers

Name	Start Address	Type	Access
AIN#(0:13)_RESOLUTION_INDEX	41500	UINT16	R/W

AIN#(0:13)_RESOLUTION_INDEX
- Starting Address: 41500

The resolution index for command-response and AIN-EF readings. A larger resolution index generally results in lower noise and longer sample times.

- Data type: UINT16 (type index = 0)
- Readable and writable
- Default value: 0
- T8:
  - Valid values: 0-16. A value of 0 will instruct the T8 to use the best resolution for the rate specified.
- T7:
  - Valid values: 0-8 for T7, 0-12 for T7-Pro. Default value of 0 corresponds to an index of 8 (T7) or 9 (T7-Pro).
- T4:
  - Valid values: 0-5. Default value of 0 corresponds to an index of 5.

Expanded Names	Addresses
AIN0_RESOLUTION_INDEX,	41500,
AIN1_RESOLUTION_INDEX,	41501,
AIN2_RESOLUTION_INDEX,	41502,
AIN3_RESOLUTION_INDEX,	41503,
AIN4_RESOLUTION_INDEX,	41504,
AIN5_RESOLUTION_INDEX,	41505,
AIN6_RESOLUTION_INDEX,	41506,
AIN7_RESOLUTION_INDEX,	41507,
AIN8_RESOLUTION_INDEX,	41508,
AIN9_RESOLUTION_INDEX,	41509,
AIN10_RESOLUTION_INDEX,	41510,
AIN11_RESOLUTION_INDEX,	41511,
AIN12_RESOLUTION_INDEX,	41512,
AIN13_RESOLUTION_INDEX	41513

AIN_ALL_RESOLUTION_INDEX	43903	UINT16	R/W

AIN_ALL_RESOLUTION_INDEX
- Address: 43903

The resolution index for command-response and AIN-EF readings. A larger resolution index generally results in lower noise and longer sample times. A write to this global parameter affects all AIN. A read will return the correct setting if all channels are set the same, but otherwise will return 0xFFFF.

- Data type: UINT16 (type index = 0)
- Readable and writable
- Default value: 0
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Valid values: 0-8 for T7, 0-12 for T7-Pro. Default value of 0 corresponds to an index of 8 (T7) or 9 (T7-Pro).
  - Minimum firmware version: 0.9328
- T4:
  - Valid values: 0-5. Default value of 0 corresponds to an index of 5.

STREAM_RESOLUTION_INDEX	4010	UINT32	R/W

STREAM_RESOLUTION_INDEX
- Address: 4010

The resolution index for stream readings. A larger resolution index generally results in lower noise and longer sample times.

- Data type: UINT32 (type index = 1)
- Readable and writable
- Default value: 0
- T8:
  - T8 description 0-16. A value of 0 will use the best resolution for the specified data rate.
- T7:
  - Valid values: 0-8. Default value of 0 corresponds to an index of 1.
- T4:
  - Valid values: 0-5. Default value of 0 corresponds to an index of 1.

T4/T7

The T4 and T7 allow individual resolution index settings for each channel. The individual channel setting only apply when using command-response. AIN#(0:13)_RESOLUTION_INDEX and AIN_ALL_RESOLUTION_INDEX do not apply to stream mode, though they do apply to all command-response communications, including AIN EF.

Analog inputs read using stream mode use a single resolution setting. Use STREAM_RESOLUTION_INDEX to configure the AIN resolution to be use for streamed channels. See our stream mode documentation for more information.

For typical noise levels and sample times at different combinations of resolution index and gain, see Appendix A-3-1 Noise and Resolution.

T8

The T8 always uses one global resolution setting. Writing the resolution index for any channel will set the global setting. Writing STREAM_RESOLUTION_INDEX will also set the global setting.

Resolution index ranges

- T4 resolution index ranges from 0 to 5
- T7 resolution index ranges from 0 to 8
- T7-Pro resolution index ranges from 0 to 12. Settings 9-12 use the alternate high-resolution converter (24-bit sigma-delta)
- T8 resolution index ranges from 0 to 16

Defaults

Setting resolution index to 0 sets the default resolution index to:

- 5 for command-response mode on a T4
- 8 for command-response mode on a T7
- 9 for command-response mode on a T7-Pro
- auto resolution selection on a T8. This will select the best resolution setting available for the AIN sampling rate. See the AIN Sampling Rate section below.

Example

To read AIN1 with resolution index 4:

1. Set the AIN1 resolution index to 4 by writing 5 to AIN1_RESOLUTION_INDEX (address 41501)
   Read AIN1

Remarks

For general discussion on the meaning of resolution, see the Noise and Resolution App Note.

Flexible I/O - T4 Only

Flexible I/O are I/O ports that may be configured as analog inputs, or as digital inputs or outputs.

Flexible I/O Configuration

Name	Start Address	Type	Access
DIO_INHIBIT	2900	UINT32	R/W

DIO_INHIBIT
- Address: 2900

A single binary-encoded value where each bit determines whether _STATE, _DIRECTION or _ANALOG_ENABLE writes affect that bit of digital I/O. 0=Default=Affected, 1=Ignored.

- Data type: UINT32 (type index = 1)
- Readable and writable
- Default value: 0

Constant	Value
Affected	0
Ignored	1

DIO_ANALOG_ENABLE	2880	UINT32	R/W

DIO_ANALOG_ENABLE
- Address: 2880

Read or write the analog configuration of all digital I/O in a single binary-encoded value. 1=Analog mode and 0=Digital mode. When switching from analog to digital, the lines will be set to input. Writes are filtered by the value in DIO_INHIBIT.

- Data type: UINT32 (type index = 1)
- Readable and writable

Constant	Value
Digital mode	0
Analog mode	1

AIN4-AIN11 are flexible I/O. To configure these channels as analog inputs:

- Set the correct bit of DIO_INHIBIT to 0
- Set the correct bit of DIO_ANALOG_ENABLE to 1

The bit to set of DIO_INHIBIT and of DIO_ANALOG_ENABLE is the same as the channel number. For example, to configure AIN4 (screw terminal FIO4) as an analog input:

- Set bit 4 of DIO_INHIBIT to 0
- Set bit 4 of DIO_ANALOG_ENABLE to 1

AIN0-AIN3 are dedicated analog inputs and cannot be configured as digital I/O.

Note that simply doing an analog read on AIN4-AIN11 automatically configures that line (FIO4-EIO3) as analog, so few people use the 2 registers mentioned above. See "Flexible I/O Auto-Configuration" in Section 13.1 Flexible I/O.

Single-ended or Differential - T7 Only

Single-ended AIN readings are read with ground (GND) as a reference point. Differential readings use a second AIN as a reference point. For more, see the Differential Readings App Note.

Negative Channel Registers

Name	Start Address	Type	Access
AIN#(0:13)_NEGATIVE_CH	41000	UINT16	R/W

AIN#(0:13)_NEGATIVE_CH
- Starting Address: 41000

Specifies the negative channel to be used for each positive channel. 199=Default=> Single-Ended.

- Data type: UINT16 (type index = 0)
- Readable and writable
- Default value: 199
- T8:
  - Ignored. This register has no meaning on the T8.
- T7:
  - For base differential channels, positive must be an even channel from 0-12 and negative must be positive+1. For extended channels 16-127, see Mux80 datasheet.

Expanded Names	Addresses
AIN0_NEGATIVE_CH, AIN1_NEGATIVE_CH, AIN2_NEGATIVE_CH, AIN3_NEGATIVE_CH, AIN4_NEGATIVE_CH, AIN5_NEGATIVE_CH, AIN6_NEGATIVE_CH, AIN7_NEGATIVE_CH, AIN8_NEGATIVE_CH, AIN9_NEGATIVE_CH, AIN10_NEGATIVE_CH, AIN11_NEGATIVE_CH, AIN12_NEGATIVE_CH, AIN13_NEGATIVE_CH	41000, 41001, 41002, 41003, 41004, 41005, 41006, 41007, 41008, 41009, 41010, 41011, 41012, 41013

AIN_ALL_NEGATIVE_CH	43902	UINT16	R/W

AIN_ALL_NEGATIVE_CH
- Address: 43902

A write to this global parameter affects all AIN. Writing 1 will set all AINs to differential. Writing 199 will set all AINs to single-ended. A read will return 1 if all AINs are set to differential and 199 if all AINs are set to single-ended. If AIN configurations are not consistent 0xFFFF will be returned.

- Data type: UINT16 (type index = 0)
- Readable and writable
- Default value: 199
- T8:
  - Ignored. This register has no meaning on the T8.
- T7:
  - Minimum firmware version: 0.9328

The AIN#(0:13)_NEGATIVE_CH and AIN_ALL_NEGATIVE_CH parameters configure whether the AIN performs differential vs. single-ended readings (not to be confused with bipolar and unipolar—the T7 is always bipolar).

When reading differential AINs, the positive AIN's configurations are used and the negative AIN's configurations are ignored. For example: when reading AIN0 with AIN0_NEGATIVE_CH is set to 1, the T7 will measure AIN0-AIN1 using the settings for AIN0 (such as AIN0_RESOLUTION_INDEX).

Example:

To take a differential reading on AIN2, set AIN3 as its negative channel (AIN2-AIN3):

1. Write 3 to AIN2_NEGATIVE_CH (address 41002)
2. Read AIN2

To read AIN2 single-ended again (AIN2-GND):

1. Write 199 to AIN2_NEGATIVE_CH (address 41002)
2. Read AIN2

User software or the Register Matrix in Kipling can be used to configure any analog input as differential, or the Analog Inputs tab in Kipling can be used to quickly set AIN0/2/4/6/8/10/12 to differential.

Testing:

Use 2 jumper wires to securely connect each analog inputs to VS or GND. Using the default +/-10V range the readings should be as follows:

AINpos(VS) and AINneg(GND) => 5 volts
AINpos(GND) and AINneg(VS) => -5 volts
AINpos(VS) and AINneg(VS) => 0 volts

Built-in AIN:

For AIN0 through AIN13, differential channels are adjacent even/odd pairs such that the positive channel is even and the negative channel is greater than the positive channel by 1. Odd channels, such as AIN3_NEGATIVE_CH (address 41003), should not be written to because only an even channel can have an associated negative channel.

Temperature and Ground:

AIN14 is the internal temperature sensor and AIN15 is GND. Neither can be read differentially.

Extended AIN:

For AIN16 and greater, the rules for pairing differential channels is different. See 14.2 Extended Channels or the Mux80 datasheet for more.

Range / Gain - T7/T8

Internal Amplifier: The analog inputs are connected to a high-impedance instrumentation amplifier, as shown in Figure 4.2-2. This in-amp does the following:

T7

- Buffers the signal for the internal ADCs
- Allows for single-ended or differential conversions
- Provides gains of x1, x10, x100, and x1000 (corresponding to ranges of ±10V, ±1V, ±0.1V, and ±0.01V, respectively).

T8

- Buffers the signal for the internal ADCs
- Provides gains corresponding to ranges of ±11V, ±9.6V, ±4.8V, ±2.4V,±1.2V, ±0.6V, ±0.3V, ±0.15V,±0.075V, ±0.036V, and ±0.018V.

Range Registers

Name	Start Address	Type	Access
AIN#(0:13)_RANGE	40000	FLOAT32	R/W

AIN#(0:13)_RANGE
- Starting Address: 40000

The range/span of each analog input. Write the highest expected input voltage.

- Data type: FLOAT32 (type index = 3)
- Readable and writable
- Default value: 0
- T8:
  - Specify the maximum expected voltage. The T8 will select the best gain for the expected voltage.
- T7:
  - Valid values/ranges: 0.0=Default → ±10V. 10.0 → ±10V, 1.0 → ±1V, 0.1 → ±0.1V, and 0.01 → ±0.01V.
- T4:
  - Valid values/ranges: 0.0=Default → 0-2.5 V on LV lines and ±10 V on HV lines.

Expanded Names	Addresses
AIN0_RANGE, AIN1_RANGE, AIN2_RANGE, AIN3_RANGE, AIN4_RANGE, AIN5_RANGE, AIN6_RANGE, AIN7_RANGE, AIN8_RANGE, AIN9_RANGE, AIN10_RANGE, AIN11_RANGE, AIN12_RANGE, AIN13_RANGE	40000, 40002, 40004, 40006, 40008, 40010, 40012, 40014, 40016, 40018, 40020, 40022, 40024, 40026

AIN_ALL_RANGE	43900	FLOAT32	R/W

AIN_ALL_RANGE
- Address: 43900

A write to this global parameter affects all AIN. A read will return the correct setting if all channels are set the same, but otherwise will return -9999.

- Data type: FLOAT32 (type index = 3)
- Readable and writable
- Default value: 0
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 0.9328

Example:

If the voltage source connected to AIN1 has a known range of 0 to 0.7V, the appropriate range for AIN1 is the ±1V range. To read AIN1 with the ±1V range:

1. Write 1.0 to AIN1_RANGE (address 40002)

2. Read AIN1

The range registers (AIN#(0:13)_RANGE and AIN_ALL_RANGE) control the gain of the T7's internal instrumentation amplifier. The in-amp supports gains of x1, x10, x100, and x1000.

• A range of 10	sets gain=x1	so that the analog input range is ±10 volts. This is the default range.
• A range of 1	sets gain=x10	so that the analog input range is ±1 volts.
• A range of 0.1	sets gain=x100	so that the analog input range is ±0.1 volts.
• A range of 0.01	sets gain=x1000	so that the analog input range is ±0.01 volts.

Values written to RANGE are rounded up (except for values greater than 10.0, which are rounded down). For example, writing 0.5 to AIN_ALL_RANGE will set the analog input range to ±1 volts.

The T7 knows what the internal gain is set to and adjusts the return values to give the voltage at the input terminals, so if you connect a 0.8 volt signal to the input terminals, it will be amplified to 8.0 volts before being digitized, but the reading you get back will be 0.8 volts.

Settling

T4/T7:

The settling registers set the time from a step change in the input signal to when the signal is sampled by the ADC, as measured in microseconds. A step change in this case is caused when the internal multiplexers change from one channel to another. In general, more settling time is required as gain and resolution are increased.

The value 0 sets automatic settling, which is recommended for most applications. This "auto" settling ensures that the device meets specifications at any gain and resolution for source impedance up to at least 1000 ohms.

Settling Registers

Name	Start Address	Type	Access
AIN#(0:13)_SETTLING_US	42000	FLOAT32	R/W

AIN#(0:13)_SETTLING_US
- Starting Address: 42000

Settling time for command-response and AIN-EF readings.

- Data type: FLOAT32 (type index = 3)
- Readable and writable
- Default value: 0
- T8:
  - Ignored. This register has no meaning on the T8.
- T7:
  - 0 = Auto. Max is 50000 (microseconds).
  - Minimum firmware version: 0.9328
- T4:
  - 0 = Auto. Max is 10000 (microseconds).

Expanded Names	Addresses
AIN0_SETTLING_US, AIN1_SETTLING_US, AIN2_SETTLING_US, AIN3_SETTLING_US, AIN4_SETTLING_US, AIN5_SETTLING_US, AIN6_SETTLING_US, AIN7_SETTLING_US, AIN8_SETTLING_US, AIN9_SETTLING_US, AIN10_SETTLING_US, AIN11_SETTLING_US, AIN12_SETTLING_US, AIN13_SETTLING_US	42000, 42002, 42004, 42006, 42008, 42010, 42012, 42014, 42016, 42018, 42020, 42022, 42024, 42026

AIN_ALL_SETTLING_US	43904	FLOAT32	R/W

AIN_ALL_SETTLING_US
- Address: 43904

Settling time for command-response and AIN-EF readings. A write to this global parameter affects all AIN. A read will return the correct setting if all channels are set the same, but otherwise will return -9999. Max is 50,000 us.

- Data type: FLOAT32 (type index = 3)
- Readable and writable
- Default value: 0
- T8:
  - Ignored. This register has no meaning on the T8.
- T7:
  - 0 = Auto. Max is 50000 (microseconds).
  - Minimum firmware version: 0.9328
- T4:
  - 0 = Auto. Max is 10000 (microseconds).

STREAM_SETTLING_US	4008	FLOAT32	R/W

STREAM_SETTLING_US
- Address: 4008

Time in microseconds to allow signals to settle after switching the mux. Does not apply to the 1st channel in the scan list, as that settling is controlled by scan rate (the time from the last channel until the start of the next scan). Default=0. When set to less than 1, automatic settling will be used. The automatic settling behavior varies by device.

- Data type: FLOAT32 (type index = 3)
- Readable and writable
- Default value: 0
- T8:
  - Ignored. The T8's analog input chain does not rely on settling time.
- T7:
  - The T7 will select the settling time based on resolution and gain settings. When the sample rate is above 60 kHz, settling time will be set to ~5 us. Max is 4400.
- T4:
  - The T4 will default to 10 us. When the scan rate is above 20 kHz, settling time will be set to 5 us.

Example:

To read AIN3 with a manual settling time of 500 µs:

1. Write 500 to AIN3_SETTLING_US (address 42006)
2. Read AIN3

Remarks:

AIN#(0:13)_SETTLING_US and AIN_ALL_SETTLING_US do not apply to stream mode, though they do apply to all command-response communications, including AIN EF.

For stream mode, use STREAM_SETTLING_US to configure AIN settling. See configuring AIN for stream for more information.

The timings in Appendix A are measured with "auto" settling.

See the Analog Input Settling Time (App Note) for more details.

T8:

The T8 AIN do not require settling time because each AIN has a dedicated analog signal chain.

AIN Sampling Rate (T8 Only)

The T8 temperature sensors and AIN are sampled at one consistent frequency determined by the AIN clock.

- Writing to the register AIN_SAMPLING_RATE_HZ will set the sampling rate to the closest rate available to the AIN clock system.
- The rate written to AIN_SAMPLING_RATE_HZ may not always match the actual sampling rate used. This is due to the AIN clock system having a limited number of valid clock base and divisor options that fit within the specifications of the T8 ADC.
- The actual sampling rate used by the AIN clock system can be read from the register AIN_SAMPLING_TIME_ACTUAL_HZ.
- When doing command response communications, the T8 will wait for the next set of AIN samples before returning the results. This means that it could take up to 1 / samplingRate seconds to acquire new AIN samples after requesting them. For example, if the AIN sampling rate is 100Hz, it can take 0-10 milliseconds to get new samples after requesting them.
- The maximum sampling rate available depends on the resolution index used. We recommend using a resolution index of 0 (auto) to select the best resolution setting for the desired sampling rate. See Appendix A for more detailed resolution index and sampling rate information.

Due to a hardware limitation, the maximum sampling rate of the T8 temperature sensors is 250 Hz. The temperature sensors will return readings of -9999 if the sampling rate is higher than 250 Hz. See Section 18.0 Internal Temp Sensor for more information about the T8 temperature sensors.

T8 Sampling Rate Registers

Name	Start Address	Type	Access
AIN_SAMPLING_RATE_HZ	43710	FLOAT32	R/W

AIN_SAMPLING_RATE_HZ
- Address: 43710

T8 Only. Writing to this register sets the analog sampling rate. Not all rates are possible. Read AIN_SAMPLING_RATE_ACTUAL_HZ to get the actual rate.

- Data type: FLOAT32 (type index = 3)
- Readable and writable
- Default value: 100
- T8:
  - Minimum firmware version: 0.0123

AIN_SAMPLING_RATE_ACTUAL_HZ	43712	FLOAT32	R

AIN_SAMPLING_RATE_ACTUAL_HZ
- Address: 43712

T8 Only. Returns the AIN system's actual sampling rate.

- Data type: FLOAT32 (type index = 3)
- Read-only
- Default value: 255
- T8:
  - Minimum firmware version: 0.0123

Power (T7/T8)

The following registers control the power state of the analog inputs.

T7

T7 Power Registers

Name	Start Address	Type	Access
POWER_AIN	48005	UINT16	R/W

POWER_AIN
- Address: 48005

The current ON/OFF state of the analog input module.

- Data type: UINT16 (type index = 0)
- Readable and writable
- T8:
  - Minimum firmware version: 1.0013
- T7:
  - Minimum firmware version: 0.8600

POWER_AIN_DEFAULT	48055	UINT16	R/W

POWER_AIN_DEFAULT
- Address: 48055

The ON/OFF state of the analog input module after a power-cycle to the device. Provided to optionally reduce power consumption.

- Data type: UINT16 (type index = 0)
- Readable and writable
- T8:
  - Minimum firmware version: 1.0013
- T7:
  - Minimum firmware version: 0.8600

T8

T8 Power Registers

Name	Start Address	Type	Access
AIN_CHANNEL_ENABLE	43700	UINT32	R/W

AIN_CHANNEL_ENABLE
- Address: 43700

T8 Only. This register is a bitmask representing the enable states of the analog inputs.

- Data type: UINT32 (type index = 1)
- Readable and writable
- Default value: 255
- T8:
  - Minimum firmware version: 0.0123

POWER_AIN	48005	UINT16	R/W

POWER_AIN
- Address: 48005

The current ON/OFF state of the analog input module.

- Data type: UINT16 (type index = 0)
- Readable and writable
- T8:
  - Minimum firmware version: 1.0013
- T7:
  - Minimum firmware version: 0.8600

The T8 does not support the POWER_AIN_DEFAULT register, accessing it will return the error POWER_INVALID_SETTING. Instead, the T8 supports individual AIN power settings using the register AIN_CHANNEL_ENABLE. The default value of AIN_CHANNEL_ENABLE can be saved as a part of the device IO Config.

Example:

- To set AIN0 and AIN3 off and all other AIN on: write AIN_CHANNEL_ENABLE = 0b1111 0110 = 246.
- To set AIN4, AIN5, and AIN6 off and all other AIN on: write AIN_CHANNEL_ENABLE = 0b1000 1111 = 143.

Other Considerations

Streaming AIN: See 3.2 Stream Mode for streaming analog inputs. Some stream configurations override the normal AIN configurations:

- Use STREAM_SETTLING_US instead of AIN_SETTLING_US.
- Use STREAM_RESOLUTION_INDEX instead of AIN_RESOLUTION_INDEX.

Command-Response while Streaming: Command-response can be performed while a stream is active. If any of the channels in the scan list are analog inputs, then streaming needs exclusive control of the analog input system. The results is that analog inputs (including the internal temperature sensor) cannot be read via command-response while a stream is active.

Floating Inputs: The analog inputs are not artificially pulled to 0.0 volts, as that would reduce the input impedance, so readings obtained from floating channels will generally not be 0.0 volts. The readings from floating channels depend on adjacent channels and sample rate and have little meaning. See the floating input application note.

Connections: For information regarding typical analog input connections, please see the Analog Input App Note.

Address Step Size:

Addresses of the FLOAT32 type increment in steps of 2 because FLOAT32 uses two sets of 16-bits. FLOAT32 registers include AIN#, AIN#_RANGE, and AIN#_SETTLING_US.

Addresses of the UINT16 type increment in steps of 1 because UNIT16 uses only one set of 16-bits. UINT16 registers include AIN#_NEGATIVE_CH and AIN#_RESOLUTION_INDEX.

T7 Only: Duplicated Terminals (AIN0-AIN3):

AIN0-AIN3 appear on the built-in screw-terminals and also on the DB37 connector. Users should only connect to one or the other, not both at the same time.

To prevent damage due to accidental short circuit, both connection paths have their own series resistor. All AIN lines have a 2.2k series resistor, and in the case of AIN0-AIN3 the duplicated connections each have their own series resistor, so if you measure the resistance between the duplicate terminals you will see about 4.4k.

Calibration:

For command-response communication, analog input calibration is automatically applied by firmware and the AIN# registers return calibrated voltages. The AIN#_BINARY registers will return binary values from the converter.

For stream communication, the AIN# registers return raw binary values and calibration is automatically performed by LJM. For applications not using LJM, see the low-level Modbus streaming example.

14.1 AIN Extended Features [T-Series Datasheet]

Overview

Analog Extended Features (AIN-EF) simplify some common analog input applications. Each AIN-EF feature:

- collects one or more input samples
- performs math on the collected samples

AIN-EF is only supported in command-response mode and not in stream mode.

Kipling Walkthrough: Kipling's Register Matrix can be used to perform AIN-EF features. For example, Configuring & Reading a Thermocouple.

Subsections

Available AIN Extended Features

For any given AIN channel, one AIN-EF feature may be selected. AIN-EF indices:

Index	AIN-EF Name	Supported Devices	Performs Stream Internally?
0:	None (disabled)	All T-series	-
1:	Offset and Slope	All T-series	-

Index	AIN-EF Name	Supported Devices	Performs Stream Internally?
3:	Max, Min, Avg	All T-series	Yes
4:	Resistance	All T-series	-
5:	Average and Threshold	All T-series	Yes
10:	RMS Flex	All T-series	Yes
11:	RMS Auto	All T-series	Yes
20:	Thermocouple type E	T7/T8	-
21:	Thermocouple type J	T7/T8	-
22:	Thermocouple type K	T7/T8	-
23:	Thermocouple type R	T7/T8	-
24:	Thermocouple type T	T7/T8	-
25:	Thermocouple type S	T7/T8	-
27:	Thermocouple type N	T7/T8	-
28:	Thermocouple type B	T7/T8	-
30:	Thermocouple type C	T7/T8	-
40:	RTD PT100	All T-series	-
41:	RTD PT500	All T-series	-
42:	RTD PT1000	All T-series	-
50:	Thermistor using Steinhart-Hart equation	All T-series	-
51:	Thermistor using Beta equation	All T-series	

AIN-EF Usage

To use any AIN-EF:

1. Set the EF_INDEX to select an extended feature
2. Configure the extended feature using the EF_CONFIG registers
3. Configure normal AIN configurations through the normal AIN registers. For AIN-EF modes that perform stream internally, configure AIN for stream.
4. Read from READ_A to perform the extended feature operation
5. Read additional results from B, C, and D

For a quick example of setting up an AIN-EF in thermocouple mode, see Configuring & Reading a Thermocouple.

Set the AIN#_EF_INDEX to select an extended feature

Name

Name	Start Address	Type	Access
AIN#(0:13)_EF_INDEX	9000	UINT32	R/W

AIN#(0:13)_EF_INDEX
- Starting Address: 9000

Specify the desired extended feature for this analog input with the index value. List of index values: 0=None(disabled); 1=Offset and Slope; 3=Max/Min/Avg; 4=Resistance; 5=Average and Threshold; 10=RMS Flex; 11=FlexRMS; 20=Thermocouple type E; 21=Thermocouple type J; 22=Thermocouple type K; 23=Thermocouple type R; 24=Thermocouple type T; 25=Thermocouple type S; 27=Thermocouple type N; 28=Thermocouple type B; 30=Thermocouple type C; 40=RTD model PT100; 41=RTD model PT500; 42=RTD model PT1000; 50=Thermistor Steinhart-Hart; 51=Thermistor Beta.

- Data type: UINT32 (type index = 1)
- Readable and writable
- Default value: 0
- This register uses system RAM. The maximum RAM is 64KB. For more information, see 4.4 RAM
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0030
- T4:
  - The T4 does not support thermocouple modes.

Expanded Names	Addresses
AIN0_EF_INDEX, AIN1_EF_INDEX, AIN2_EF_INDEX, AIN3_EF_INDEX, AIN4_EF_INDEX, AIN5_EF_INDEX, AIN6_EF_INDEX, AIN7_EF_INDEX, AIN8_EF_INDEX, AIN9_EF_INDEX, AIN10_EF_INDEX, AIN11_EF_INDEX, AIN12_EF_INDEX, AIN13_EF_INDEX	9000, 9002, 9004, 9006, 9008, 9010, 9012, 9014, 9016, 9018, 9020, 9022, 9024, 9026

AIN_ALL_EF_INDEX	43906	UINT32	R/W

AIN_ALL_EF_INDEX
- Address: 43906

Write 0 to deactivate AIN_EF on all AINs. No other values may be written to this register. Reads will return the AIN_EF index if all 128 AINs are set to the same value. If values are not the same returns 0xFFFF (65535).

- Data type: UINT32 (type index = 1)
- Readable and writable
- Default value: 0
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0115
- T4:
  - The T4 does not support thermocouple modes.

Write to AIN#(0:14)_EF_INDEX or AIN_ALL_EF_INDEX to select the AIN extended feature.

Configure the extended feature using the AIN#_EF_CONFIG registers

Name

Name	Start Address	Type	Access
AIN#(0:13)_EF_CONFIG_A	9300	UINT32	R/W

AIN#(0:13)_EF_CONFIG_A
- Starting Address: 9300

Function dependent on selected feature index.

- Data type: UINT32 (type index = 1)
- Readable and writable
- Default value: 0
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0030

Expanded Names	Addresses
AIN0_EF_CONFIG_A, AIN1_EF_CONFIG_A, AIN2_EF_CONFIG_A, AIN3_EF_CONFIG_A, AIN4_EF_CONFIG_A, AIN5_EF_CONFIG_A, AIN6_EF_CONFIG_A, AIN7_EF_CONFIG_A, AIN8_EF_CONFIG_A, AIN9_EF_CONFIG_A, AIN10_EF_CONFIG_A, AIN11_EF_CONFIG_A, AIN12_EF_CONFIG_A, AIN13_EF_CONFIG_A	9300, 9302, 9304, 9306, 9308, 9310, 9312, 9314, 9316, 9318, 9320, 9322, 9324, 9326

AIN#(0:13)_EF_CONFIG_B	9600	UINT32	R/W

AIN#(0:13)_EF_CONFIG_B
- Starting Address: 9600

Function dependent on selected feature index.

- Data type: UINT32 (type index = 1)
- Readable and writable
- Default value: 0
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0030

Expanded Names	Addresses
AIN0_EF_CONFIG_B, AIN1_EF_CONFIG_B, AIN2_EF_CONFIG_B, AIN3_EF_CONFIG_B, AIN4_EF_CONFIG_B, AIN5_EF_CONFIG_B, AIN6_EF_CONFIG_B, AIN7_EF_CONFIG_B, AIN8_EF_CONFIG_B, AIN9_EF_CONFIG_B, AIN10_EF_CONFIG_B, AIN11_EF_CONFIG_B, AIN12_EF_CONFIG_B, AIN13_EF_CONFIG_B	9600, 9602, 9604, 9606, 9608, 9610, 9612, 9614, 9616, 9618, 9620, 9622, 9624, 9626

AIN#(0:13)_EF_CONFIG_C	9900	UINT32	R/W

AIN#(0:13)_EF_CONFIG_C
- Starting Address: 9900

Function dependent on selected feature index.

- Data type: UINT32 (type index = 1)
- Readable and writable
- Default value: 0
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0030

Expanded Names	Addresses
AIN0_EF_CONFIG_C, AIN1_EF_CONFIG_C, AIN2_EF_CONFIG_C, AIN3_EF_CONFIG_C, AIN4_EF_CONFIG_C, AIN5_EF_CONFIG_C, AIN6_EF_CONFIG_C, AIN7_EF_CONFIG_C, AIN8_EF_CONFIG_C, AIN9_EF_CONFIG_C, AIN10_EF_CONFIG_C, AIN11_EF_CONFIG_C, AIN12_EF_CONFIG_C, AIN13_EF_CONFIG_C	9900, 9902, 9904, 9906, 9908, 9910, 9912, 9914, 9916, 9918, 9920, 9922, 9924, 9926

AIN#(0:13)_EF_CONFIG_D	10200	FLOAT32	R/W

AIN#(0:13)_EF_CONFIG_D
- Starting Address: 10200

Function dependent on selected feature index.

- Data type: FLOAT32 (type index = 3)
- Readable and writable
- Default value: 0
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0030

Expanded Names	Addresses
AIN0_EF_CONFIG_D, AIN1_EF_CONFIG_D, AIN2_EF_CONFIG_D, AIN3_EF_CONFIG_D, AIN4_EF_CONFIG_D, AIN5_EF_CONFIG_D, AIN6_EF_CONFIG_D, AIN7_EF_CONFIG_D, AIN8_EF_CONFIG_D, AIN9_EF_CONFIG_D, AIN10_EF_CONFIG_D, AIN11_EF_CONFIG_D, AIN12_EF_CONFIG_D, AIN13_EF_CONFIG_D	10200, 10202, 10204, 10206, 10208, 10210, 10212, 10214, 10216, 10218, 10220, 10222, 10224, 10226

AIN#(0:13)_EF_CONFIG_E	10500	FLOAT32	R/W

AIN#(0:13)_EF_CONFIG_E
- Starting Address: 10500

Function dependent on selected feature index.

- Data type: FLOAT32 (type index = 3)
- Readable and writable
- Default value: 0
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0030

Expanded Names	Addresses
AIN0_EF_CONFIG_E, AIN1_EF_CONFIG_E, AIN2_EF_CONFIG_E, AIN3_EF_CONFIG_E, AIN4_EF_CONFIG_E, AIN5_EF_CONFIG_E, AIN6_EF_CONFIG_E, AIN7_EF_CONFIG_E, AIN8_EF_CONFIG_E, AIN9_EF_CONFIG_E, AIN10_EF_CONFIG_E, AIN11_EF_CONFIG_E, AIN12_EF_CONFIG_E, AIN13_EF_CONFIG_E	10500, 10502, 10504, 10506, 10508, 10510, 10512, 10514, 10516, 10518, 10520, 10522, 10524, 10526

AIN#(0:13)_EF_CONFIG_F	10800	FLOAT32	R/W

AIN#(0:13)_EF_CONFIG_F
- Starting Address: 10800

Function dependent on selected feature index.

- Data type: FLOAT32 (type index = 3)
- Readable and writable
- Default value: 0
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0030

Expanded Names	Addresses
AIN0_EF_CONFIG_F, AIN1_EF_CONFIG_F, AIN2_EF_CONFIG_F, AIN3_EF_CONFIG_F, AIN4_EF_CONFIG_F, AIN5_EF_CONFIG_F, AIN6_EF_CONFIG_F, AIN7_EF_CONFIG_F, AIN8_EF_CONFIG_F, AIN9_EF_CONFIG_F, AIN10_EF_CONFIG_F, AIN11_EF_CONFIG_F, AIN12_EF_CONFIG_F, AIN13_EF_CONFIG_F	10800, 10802, 10804, 10806, 10808, 10810, 10812, 10814, 10816, 10818, 10820, 10822, 10824, 10826

AIN#(0:13)_EF_CONFIG_G	11100	FLOAT32	R/W

AIN#(0:13)_EF_CONFIG_G
- Starting Address: 11100

Function dependent on selected feature index.

- Data type: FLOAT32 (type index = 3)
- Readable and writable
- Default value: 0
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0030

Expanded Names	Addresses
AIN0_EF_CONFIG_G, AIN1_EF_CONFIG_G, AIN2_EF_CONFIG_G, AIN3_EF_CONFIG_G, AIN4_EF_CONFIG_G, AIN5_EF_CONFIG_G, AIN6_EF_CONFIG_G, AIN7_EF_CONFIG_G, AIN8_EF_CONFIG_G, AIN9_EF_CONFIG_G, AIN10_EF_CONFIG_G, AIN11_EF_CONFIG_G, AIN12_EF_CONFIG_G, AIN13_EF_CONFIG_G	11100, 11102, 11104, 11106, 11108, 11110, 11112, 11114, 11116, 11118, 11120, 11122, 11124, 11126

Each AIN-EF index requires different configuration parameters, so the meaning of the

AIN#_EF_CONFIG registers depend on which AIN#_EF_INDEX is set.

Configure normal AIN configurations through the normal AIN registers

Analog input range, resolution, settling, and negative channel settings are configured through the normal AIN registers.

AIN-EF modes that perform stream internally may need to configure AIN for stream.

Read from AIN#_EF_READ_A to perform the extended feature operation

Name

Name	Start Address	Type	Access
AIN#(0:13)_EF_READ_A	7000	FLOAT32	R

AIN#(0:13)_EF_READ_A
- Starting Address: 7000

Function dependent on selected feature index.

- Data type: FLOAT32 (type index = 3)
- Read-only
- Default value: 0
- This register uses system RAM. The maximum RAM is 64KB. For more information, see 4.4 RAM
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0030

Expanded Names	Addresses
AIN0_EF_READ_A, AIN1_EF_READ_A, AIN2_EF_READ_A, AIN3_EF_READ_A, AIN4_EF_READ_A, AIN5_EF_READ_A, AIN6_EF_READ_A, AIN7_EF_READ_A, AIN8_EF_READ_A, AIN9_EF_READ_A, AIN10_EF_READ_A, AIN11_EF_READ_A, AIN12_EF_READ_A, AIN13_EF_READ_A	7000, 7002, 7004, 7006, 7008, 7010, 7012, 7014, 7016, 7018, 7020, 7022, 7024, 7026

Only reading AIN#_EF_READ_A will trigger the selected AIN-EF operation. The AIN#_EF_READ_A result is returned. Additional results are saved for later retrieval.

If the AIN-EF index uses stream-burst, reading AIN#_EF_READ_A will block for the length of time it takes to collect the necessary samples.

Read additional results B, C, and D

Name

Name	Start Address	Type	Access
AIN#(0:13)_EF_READ_B	7300	FLOAT32	R/W

AIN#(0:13)_EF_READ_B
- Starting Address: 7300

Function dependent on selected feature index.

- Data type: FLOAT32 (type index = 3)
- Readable and writable
- Default value: 0
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0030

Expanded Names	Addresses
AIN0_EF_READ_B, AIN1_EF_READ_B, AIN2_EF_READ_B, AIN3_EF_READ_B, AIN4_EF_READ_B, AIN5_EF_READ_B, AIN6_EF_READ_B, AIN7_EF_READ_B, AIN8_EF_READ_B, AIN9_EF_READ_B, AIN10_EF_READ_B, AIN11_EF_READ_B, AIN12_EF_READ_B, AIN13_EF_READ_B	7300, 7302, 7304, 7306, 7308, 7310, 7312, 7314, 7316, 7318, 7320, 7322, 7324, 7326

AIN#(0:13)_EF_READ_C	7600	FLOAT32	R/W

AIN#(0:13)_EF_READ_C
- Starting Address: 7600

Function dependent on selected feature index.

- Data type: FLOAT32 (type index = 3)
- Readable and writable
- Default value: 0
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0030

Expanded Names	Addresses
AIN0_EF_READ_C, AIN1_EF_READ_C, AIN2_EF_READ_C, AIN3_EF_READ_C, AIN4_EF_READ_C, AIN5_EF_READ_C, AIN6_EF_READ_C, AIN7_EF_READ_C, AIN8_EF_READ_C, AIN9_EF_READ_C, AIN10_EF_READ_C, AIN11_EF_READ_C, AIN12_EF_READ_C, AIN13_EF_READ_C	7600, 7602, 7604, 7606, 7608, 7610, 7612, 7614, 7616, 7618, 7620, 7622, 7624, 7626

AIN#(0:13)_EF_READ_D	7900	FLOAT32	R

AIN#(0:13)_EF_READ_D
- Starting Address: 7900

Function dependent on selected feature index.

- Data type: FLOAT32 (type index = 3)
- Read-only
- Default value: 0
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0030

Expanded Names	Addresses
AIN0_EF_READ_D, AIN1_EF_READ_D, AIN2_EF_READ_D, AIN3_EF_READ_D, AIN4_EF_READ_D, AIN5_EF_READ_D, AIN6_EF_READ_D, AIN7_EF_READ_D, AIN8_EF_READ_D, AIN9_EF_READ_D, AIN10_EF_READ_D, AIN11_EF_READ_D, AIN12_EF_READ_D, AIN13_EF_READ_D	7900, 7902, 7904, 7906, 7908, 7910, 7912, 7914, 7916, 7918, 7920, 7922, 7924, 7926

Reading from result registers other than AIN#_EF_READ_A will read the saved values and will not initiate a new reading.

14.1.0.1 Excitation Circuits [T-Series Datasheet]

Overview

AIN-EF indices that need to measure resistance (Resistance, RTD, Thermistor) can use different types of excitation circuits. The excitation circuit converts the varying resistance to a voltage signal that can be measured by the LabJack.

Individual AIN-EF indices may only support a subset of the circuits listed here.

For AIN-EF indices that require excitation circuits, the circuit indices below are written to AIN#_EF_CONFIG_B.

The most commonly used circuit is #4, as it is designed for the LJTick-Resistor.

These circuits all use a voltage source or current source for excitation. Note that any noise in the excitation source will result in proportionate noise in the sensor signal. Sources designed for excitation (e.g. voltage reference) are recommended rather than power supplies (e.g. VS).

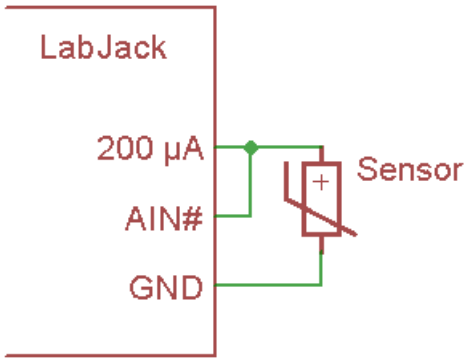
Current Source Excitation Circuits

The first 3 excitation circuits are specific to current source excitation. A current source varies voltage so it can provide the specified fixed current.

Circuit 0 – 200 µA Current Source - T7:

This excitation circuit uses the 200 µA current source available on the T7 to excite a sensor. It calculates resistance based on the measured voltage and the stored factory calibration value for 200UA. This circuit is useful for smaller resistances such as RTDs.

The following figure shows a basic single sensor connection, but if the AIN is configured and connected as differential, multiple sensors can be put in series as shown in figures from the 200UA/10UA documentation.



200µA Current Source

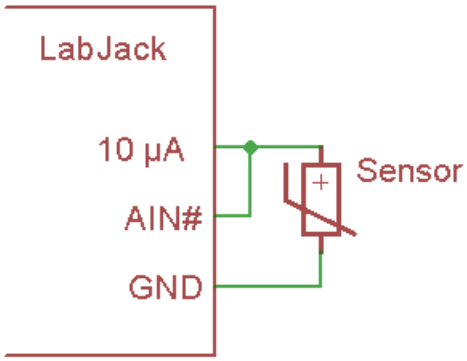
Configuration registers:

- AIN#_EF_CONFIG_C - Ignored
- AIN#_EF_CONFIG_D - Ignored
- AIN#_EF_CONFIG_E - Ignored

Circuit 1 – 10 µA Current Source - T7:

This excitation circuit uses the 10 µA current source available on the T7 to excite a sensor. It calculates resistance based on the measured voltage and the stored factory calibration value for 10UA. This circuit is useful for larger resistances such as thermistors.

The following figure shows a basic single sensor connection, but if the AIN is configured and connected as differential, multiple sensors can be put in series as shown in figures from the 200UA/10UA documentation.



10 µA current Source

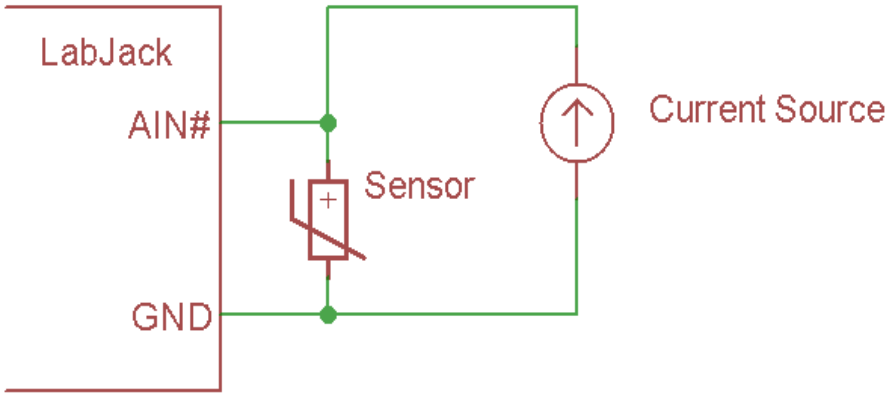
Configuration registers:

- AIN#_EF_CONFIG_C - Ignored
- AIN#_EF_CONFIG_D - Ignored
- AIN#_EF_CONFIG_E - Ignored

Circuit 2 – Custom Current Source:

This excitation circuit uses a current source external to the LabJack. The current provided by the source is specified during configuration of the AIN#_EF.

The following figure shows a basic single sensor connection, but if the AIN is configured and connected as differential, multiple sensors can be put in series.



Configuration registers:

- AIN#_EF_CONFIG_C - Ignored
- AIN#_EF_CONFIG_D - Excitation Amps
- AIN#_EF_CONFIG_E - Ignored

Resistive Divider Excitation Circuits

Divider circuits rely on an excitation source and a fixed resistor in series with the sensor.

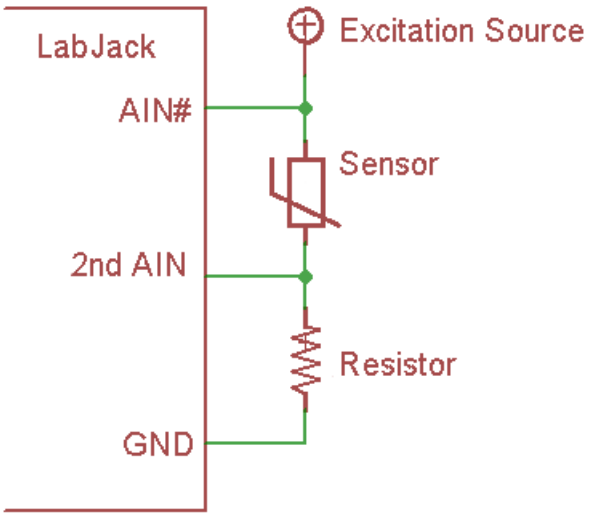
Circuit 3 – Divider with Measured Excitation - Differential:

This circuit has an excitation source in series with a sensor which is then in series with a fixed resistor to ground. The device takes the single-ended reading of 2ndAIN to get Vresistor, and takes the differential reading of AIN# to get Vsensor. Vresistor is divided by the specified fixed resistance to get current, and then Vsensor is divided by that current to get the resistance of the sensor.

The drawing shows the most common way of connecting. AIN# is a positive differential channel (e.g. AIN2) and 2ndAIN is the negative associated with that channel (e.g. AIN3).

Alternatively, any differential pair of analog inputs can be connected across the sensor, and 2ndAIN can be any single-ended analog input. This allows multiple sensors to be connected in series with a single excitation source.

AIN# must be pre-configured by the user as differential or an error will be thrown. Thus this circuit is supported on the T7 but not the T4.

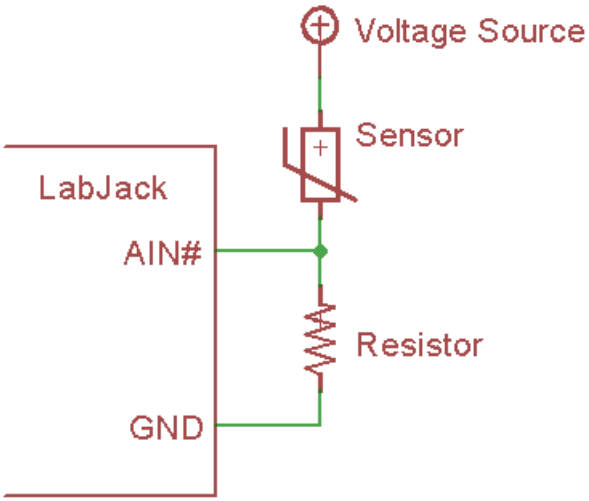


Configuration registers:

- AIN#_EF_CONFIG_C - 2nd AIN: Channel Number to Measure Vresistor
- AIN#_EF_CONFIG_D - Ignored
- AIN#_EF_CONFIG_E - Fixed Resistor Ohms

Circuit 4 – Voltage Source with Specified Value:

This excitation circuit uses a voltage source and a shunt resistor. Values for the output of the voltage source and the resistor must be provided during AIN#_EF configuration. When using this circuit, the LabJack will measure the voltage between the sensor and the resistor, then calculate the resistance of the sensor.



Configuration registers:

- AIN#_EF_CONFIG_C - Ignored
- AIN#_EF_CONFIG_D - Excitation Volts
- AIN#_EF_CONFIG_E - Fixed Resistor Ohms.

Note for the LJTick-Resistance: This excitation circuit #4 is the most common circuit used with the LJTick-Resistance. The "Resistor" shown above is built into the LJTick-Resistance, so to create this circuit simply connect one side of the RTD to LJTR-Vref and the other side of the RTD to LJTR-VINx.

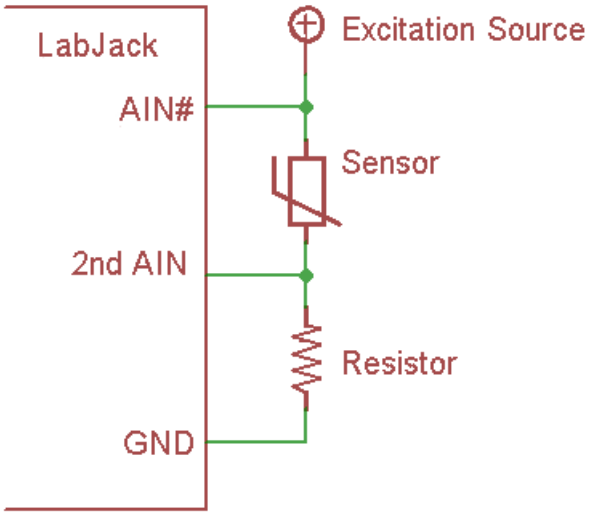
Note for the T8: The T8 has two 3.3 V reference voltage outputs that could be suitable for excitation.

Circuit 5 – Divider with Measured Excitation - Single-Ended:

This circuit has an excitation source in series with a sensor which is then in series with a fixed resistor to ground. The device takes the single-ended reading of 2ndAIN to get Vresistor, and takes the single-ended reading of AIN# minus the reading from 2ndAIN to get Vsensor. Vresistor is divided by the specified resistance to get current, and then Vsensor is divided by current to get the resistance of the sensor.

Must be connected exactly as shown in the drawing. AIN# and 2ndAIN can be any channels.

This excitation circuit looks the same as circuit #3, but the voltage across the sensor is determined by the difference of 2 single-ended readings rather than a single differential reading. That means this circuit is supported on all devices and is limited to a single sensor in series with the excitation source.

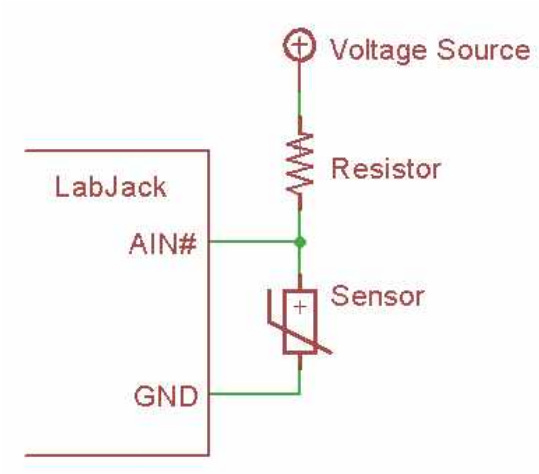


Configuration registers:

- AIN#_EF_CONFIG_C - 2nd AIN: Channel Number to Measure Vresistor
- AIN#_EF_CONFIG_D - Ignored
- AIN#_EF_CONFIG_E - Fixed Resistor Ohms

Circuit 1004 – Voltage Source with Specified Value (swapped fixed resistor position) - T4/T7

This excitation circuit uses a voltage source and a shunt resistor. It is the same as circuit 4 except with the fixed resistor and sensor positions swapped. Values for the output of the voltage source and the resistor must be provided during AIN#_EF configuration. When using this circuit, the LabJack will measure the voltage between the sensor and the resistor, then calculate the resistance of the sensor.



Configuration registers:

- AIN#_EF_CONFIG_C - Ignored
- AIN#_EF_CONFIG_D - Excitation Volts
- AIN#_EF_CONFIG_E - Fixed Resistor Ohms.

Note for the T8: The T8 has two 3.3 V reference voltage outputs that could be suitable for excitation.

14.1.1 Thermocouple (T7/T8) [T-Series Datasheet]

Overview - T7/T8

This feature is only supported on the T7 and T8.

AIN#_EF_INDEX values:

20: Thermocouple type E
21: Thermocouple type J
22: Thermocouple type K
23: Thermocouple type R
24: Thermocouple type T
25: Thermocouple type S
27: Thermocouple type N
28: Thermocouple type B
30: Thermocouple type C

This Thermocouple Extended Feature automatically performs calculations for the thermocouple types listed above.

Thermocouple AIN-EF indices read two analog inputs—one AIN connected to a thermocouple and a second AIN connected to a Cold-Junction Compensation (CJC) sensor.

For more information, see the Thermocouples Application Note and the Thermocouples with the T7 App Note / Thermocouples with the T8 App Note.

Configuration

To configure, write to the following registers.

AIN#_EF_CONFIG_A - Options: Select temperature units for AIN#_EF_READ_A and AIN#_EF_READ_C:

- 0 = K
- 1 = °C
- 2 = °F

AIN#_EF_CONFIG_B - CJC Modbus address: The Modbus address of the second AIN channel that will be read to acquire the CJC reading

The default is the on-board temperature sensor (TEMPERATURE_DEVICE_K, at address 60052). Note: The Modbus address for AIN are always 2*AIN#. For example, AIN2 is at address 4.

AIN#_EF_CONFIG_D - CJC Slope: A slope to be applied to the CJC reading

This value is always in units of K/volt, regardless of AIN#_EF_CONFIG_A, so for the internal sensor it will nominally be 1.00 and for an LM34 it will be 55.56.

Default is 1.0.

AIN#_EF_CONFIG_E - CJC Offset: An offset to be applied to the CJC reading

This value is always in units of K, regardless of AIN#_EF_CONFIG_A, so for the internal sensor it will nominally be 0.0 and for an LM34 it will be 255.37.

Default is 0.0.

Remarks

The normal analog input settings are used for negative channel, resolution index, settling, and range.

Differential thermocouple readings have the advantage of mitigating bad ground loops and ground offset problems. See the Thermocouples App Note for more information.

The ±0.1 volt range is automatically used if the range of the applicable channel (AIN#_RANGE) is set to the default ±10 volts. Otherwise, the specified range will be used.

CJC calculations are always done in kelvin, regardless of whether AIN#_EF_CONFIG_A is used to change the output units. AIN#_EF_CONFIG_D and AIN#_EF_CONFIG_E should be used as needed to convert the CJC sensor reading to kelvin.

Results

Results are read from the following registers. Only reading AIN#_EF_READ_A triggers a new measurement.

AIN#_EF_READ_A: Final calculated temperature of the remote end of the thermocouple. -9999 will be returned when the measured thermocouple voltage is outside of the valid thermocouple voltage range. This commonly occurs when the analog input is floating.
AIN#_EF_READ_B: Measured thermocouple voltage.
AIN#_EF_READ_C: CJC temperature.
AIN#_EF_READ_D: Thermocouple voltage calculated for CJC temperature.

Example

Assume a type K thermocouple is connected to AIN3/GND. To configure:

```
AIN3_EF_INDEX = 22      // feature index for type K thermocouple
AIN3_EF_CONFIG_B = 60052 // Set the CJC source (the address for device
                    // temperature sensor). 60052 is
TEMPERATURE_DEVICE_K
AIN3_EF_CONFIG_D = 1.0   // slope for CJC reading
AIN3_EF_CONFIG_E = 0.0   // offset for CJC reading
```

Read AIN3_EF_READ_A to get the calculated temperature. If the remote end is at room temperature, it will read as approximately 298 kelvin.

For a more detailed walkthrough, see Configuring & Reading a Thermocouple.

14.1.2 Offset and Slope [T-Series Datasheet]

Overview

AIN#_EF_INDEX: **1**

This Offset and Slope Extended Feature automatically adds a slope and an offset to analog readings.

Configuration

To configure, write to the following registers.

AIN#_EF_CONFIG_D - Slope: Custom slope to be applied to the analog voltage reading. Default is 1.00.
AIN#_EF_CONFIG_E - Offset: Custom offset to be applied to the analog voltage reading. Default is 0.00.

Remarks

The normal analog input settings are used for negative channel, resolution index, settling, and range.

Results

For results, read AIN#_EF_READ_A.

AIN#_EF_READ_A - Returns the calculated voltage:

```
measured volts * slope + offset
```

Only reading AIN#_EF_READ_A triggers a new measurement.

Example

To configure Offset and Slope AIN-EF for AIN3:

```
AIN3_EF_INDEX = 1        // feature index for Offset and
Slope
AIN3_EF_CONFIG_D = 2.0    // slope
AIN3_EF_CONFIG_E = -0.5   // offset
```

Now each read of AIN3_EF_READ_A will return `(AIN3 volts * 2.0) - 0.5` .

14.1.3 RTD [T-Series Datasheet]

Overview

AIN#_EF_INDEX values:

40: PT100
41: PT500
42: PT1000

This RTD Extended Feature automatically performs calculations for a Resistance Temperature Detector (RTD). RTD types are listed above.

When AIN#_EF_READ_A is read, the T-series device reads an analog input and calculates the resistance of the RTD. Temperature is then calculated using the rational polynomial technique.

An RTD (aka PT100, PT1000) is a type of temperature sensor. See the Temperature Sensors App Note.

An RTD provides a varying resistance, but the LabJack measures voltage, so some sort of circuit must be used to convert the varying resistance to a varying voltage. This AIN-EF supports various excitation circuits. The best option is usually the LJTick-Resistance, which would be excitation circuit #4.

The resistance to temperature conversion is done using the RTD Rational Polynomial technique. The polynomial coefficients are fixed and assume the most common RTD characteristics, where a PT100 (for example) has a resistance of 100.0 ohms at 0 °C and a coefficient of 0.00385. Math for non-standard RTDs will have to be handled by the user. PT500 is assumed to have 5 times the resistance of a PT100, and a PT1000 is assumed to have 10 times. More information about the polynomial can be found here: http://www.mosaic-industries.com/embedded-systems/microcontroller-projec...

Configuration

To configure, write to the following registers.

AIN#_EF_CONFIG_A - Options: Selects temperature units:

- 0 = K
- 1 = °C
- 2 = °F

AIN#_EF_CONFIG_B - Excitation Circuit Index: The index of the voltage divider excitation circuit to be used.

See 14.1.0.1 Excitation Circuits for circuit indices.

AIN#_EF_CONFIG_C - 2nd AIN: Channel Number to Measure Vresistor: For excitation circuits 3 and 5 this is the extra AIN used to measure the voltage across the fixed resistor. Ignored for other excitation circuits.

AIN#_EF_CONFIG_D - Excitation Volts or Amps: For excitation circuit 2 this is the fixed amps of the current source. For excitation circuit 4 this is the fixed volts of the voltage source. Ignored for other excitation circuits.

AIN#_EF_CONFIG_E - Fixed Resistor Ohms: For excitation circuits 3, 4 and 5, this is the ohms of the fixed resistor.

Remarks

The normal analog input settings are used for negative channel, resolution index, settling, and range.

T7: If the voltage will stay below 1.0V, use the 1.0V range for improved resolution and accuracy.

T8: For improved resolution and accuracy, use the lowest voltage range that is appropriate for your signal's voltage min/max. For example, if the signal's min/max is -0.0064V/0.0549V, use the T8 voltage range of ±0.075V.

Results

Retrieve the results by reading the following registers.

AIN#_EF_READ_A: Calculated temperature.
AIN#_EF_READ_B: Resistance of the RTD.
AIN#_EF_READ_C: Voltage across the RTD.
AIN#_EF_READ_D: Current through the RTD.

Only reading AIN#_EF_READ_A triggers a new measurement, so you must always read A before reading B, C or D.

Example

The LJTick-Resistance-1k is the best and easiest way to measure an RTD, but if you don't have an LJTR the 200UA source on the T7 is a quick way to get readings.

200UA current source, circuit #0:

Connect 200UA to AIN0 and connect a PT100 RTD from AIN0 to GND:

```
AIN0_EF_INDEX = 40      // Set AIN_EF0 to RTD100.
AIN0_EF_CONFIG_A = 0    // Set result units to
kelvin.
AIN0_EF_CONFIG_B = 0    // Set excitation circuit to
0.
```

Now each read of AIN0_EF_READ_A will measure the voltage on AIN0, use that to calculate resistance based on the factory stored value for 200UA, and use that to calculate temperature.

LJTick-Resistance-1k, circuit #4:

Connect a PT100 RTD from Vref to VINA on an LJTick-Resistance-1k that is plugged into the AIN0/AIN1 block.

```
AIN0_EF_INDEX = 40       // Set AIN_EF0 to RTD100.
AIN0_EF_CONFIG_A = 0     // Set result units to kelvin.
AIN0_EF_CONFIG_B = 4     // Set excitation circuit to 4.
AIN0_EF_CONFIG_D = 2.50  // Vref on the LJTR is 2.50 volts.
AIN0_EF_CONFIG_E = 1000  // We are using the 1k version of the LJTR.
```

Now each read of AIN0_EF_READ_A will measure the voltage on AIN0, do the voltage divider math to determine the resistance of the RTD, and use that to calculate temperature.

14.1.4 RMS [T-Series Datasheet]

Overview

AIN#_EF_INDEX values:

10: RMS Flex
11: RMS Auto

This RMS Extended Feature calculates the true root mean square voltage of a signal connected to an analog input. It also calculates peak-to-peak voltage, DC offset (average voltage), and period (1/frequency) of the analog waveform.

RMS Flex: RMS Flex uses registers to specify the number of samples to collect and the frequency at which to collect those samples. The LabJack will then use the entire data set to calculate RMS.

RMS Auto: RMS Auto will collect samples for more than one period and attempt to find a period within the data set. The LabJack will then compute RMS for only the detected period's period.

This feature internally uses stream mode.

Configuration

To configure, write to the following registers.

AIN#_EF_CONFIG_A - Number of Samples: The number of samples to be acquired.

- Default: 200
- Max: 16384

AIN#_EF_CONFIG_B - Hysteresis (RMS Auto only): The smallest step (analog voltage in binary) that will trigger period detection when using RMS Auto. Larger values will better reject noise. Smaller values can increase the accuracy of the period calculation.

- Default: 100 (16-bit counts)
- This value is meaningless for RMS Flex

AIN#_EF_CONFIG_D - Scan Rate: The frequency at which samples will be collected.

- Default: 6000

Sample Time

The maximum possible sample time is 180 ms.

Sample time is the number of samples divided by the sample frequency. For example, the period is 16.7 ms when using a scan rate of 6000 Hz and 100 samples:

100 samples / 6000 samples per second = 16.7 ms

RMS Flex Considerations

RMS Flex requires the sample time to be an even multiple of the period to be measured.

RMS Auto Considerations

RMS Auto requires the sample time to be set so that 1.5 to 4 periods will be observed.

The hysteresis value also needs to be set to control the sensitivity of period detection.

Accuracy

Stream-burst is used to acquire the specified number of samples at hardware timed intervals. The true RMS is then calculated using the following piece-wise math:

```
( (Σ(each sample^2)) / number of samples )
^0.5
```

The accuracy of this calculation depends on the number of samples per cycle. Smooth waveforms will not require many samples per cycle, but waveforms with lots of harmonic content will require more samples per cycle.

Stream Configuration

This extended feature internally uses Stream-Burst to acquire the data set, so stream AIN configurations apply.

- Crucially, set STREAM_RESOLUTION_INDEX between 0 and 8. Resolution index 9 is not supported in stream.

Results

For results, read the following registers.

AIN#_EF_READ_A: RMS Voltage
AIN#_EF_READ_B: Peak-to-Peak Voltage
AIN#_EF_READ_C: DC Offset Voltage (Average)
AIN#_EF_READ_D: Period (Seconds)

Only reading AIN#_EF_READ_A triggers a new measurement. Because multiple measurements are taken, a read of AIN#_EF_READ_A blocks for the length of the sample time.

14.1.5 Thermistor [T-Series Datasheet]

Overview

AIN#_EF_INDEX values:

50: calculate temperature using the Steinhart-Hart equation
51: calculate temperature using the beta equation

This Thermistor Extended Feature automatically performs the necessary calculations for thermistors using the Steinhart-Hart equation or the beta equation.

Steinhart-Hart vs. beta: The beta function works well over a limited range of about 50 ºC. Typical error is ~±0.5 ºC. The Steinhart-Hart is usually more accurate (±0.01 ºC) across a larger range. Note that this is just the accuracy of the math converting resistance to temperature, and there are likely other sources of error in your measurement that are similar or greater (e.g. accuracy of the thermistor itself and accuracy of the resistance to voltage conversion circuit).

Configuration

To configure, write to the following registers.

AIN#_EF_CONFIG_A - Thermistor Options: Selects temperature units:

- 0 = K
- 1 = °C
- 2 = °F

AIN#_EF_CONFIG_B - Excitation Circuit Index: The index of the voltage divider excitation circuit to be used.

See 14.1.0.1 Excitation Circuits for circuit indices.

AIN#_EF_CONFIG_C - 2nd AIN: Channel Number to Measure Vresistor: For excitation circuits 3 and 5 this is the extra AIN used to measure the voltage across the fixed resistor. Ignored for other excitation circuits.

AIN#_EF_CONFIG_D - Excitation Volts or Amps: For excitation circuit 2 this is the fixed amps of the current source. For excitation circuit 4 this is the fixed volts of the voltage source. Ignored for other excitation circuits.

AIN#_EF_CONFIG_E - Fixed Resistor Ohms: For excitation circuits 3, 4 and 5, this is the ohms of the fixed resistor.

AIN#_EF_CONFIG_F - R25 Ohms: The nominal resistance in ohms of the thermistor at 25 °C.

	Steinhart-Hart	Beta
AIN#_EF_CONFIG_G	A coefficient	ß
AIN#_EF_CONFIG_H	B coefficient	ºC at which ß was calculated
AIN#_EF_CONFIG_I	C coefficient	No meaning
AIN#_EF_CONFIG_J	D coefficient	No meaning

The G, H, I, and J config registers have different meaning for Steinhart-Hart and beta. Steinhart-Hart coefficients are normally provided in the thermistor's datasheet or obtained from the manufacturer.

There are 2 forms of the Steinhart-Hart equation:

```
1/T= A + B*ln(R/R25) + C*ln(R/R25)^2 + D*ln(R/R25)^3
1/T= A + B*ln(R) + C*ln(R)^2 + D*ln(R)^3
```

We use the former with "R/R25". If you have coefficients that were generated based on the "R" equation, just set

```
R25 = 1 (AIN#_EF_CONFIG_F = 1).
```

Further, sometimes the ^2 term is dropped and the equation is written `A + B*ln(R) + C*ln(R)^3`. If you have coefficients that were generated based on this form set C = 0 (AIN#_EF_CONFIG_I = 0) and pass the given C value for D (AIN#_EF_CONFIG_J).

The online calculator from daycounter.com uses the "R/R25" form, and thus is useful for testing. LabJack provides a Thermistor Calculator spreadsheet that is also useful for testing and troubleshooting (make a copy if you want an editable version). The online calculator from thinksrs.com can be used to test "R" based coefficients or the beta equation, and can also be used to generate "R" based Steinhart-Hart coefficients from 3 resistance-temperature pairs.

Remarks

The normal analog input registers are used to control negative channel, resolution index, settling, and range.

T7: If the voltage will stay below 1.0V, use the 1.0V range for improved resolution and accuracy.

T8: For improved resolution and accuracy, use the lowest voltage range that is appropriate for your signal's voltage min/max. For example, if the signal's min/max is -0.0064V/0.0549V, use the T8 voltage range of ±0.075V.

Results

For results, read the following registers.

AIN#_EF_READ_A: Calculated thermistor temperature
AIN#_EF_READ_B: Thermistor resistance

AIN#_EF_READ_C: Thermistor voltage

Only reading AIN#_EF_READ_A triggers a new measurement, so you must always read A before reading B or C.

Troubleshooting

Temperature to Voltage

Determine the expected resistance and voltage and compare to what you are seeing. Assume we have a Vishay NTCLE100E3103 10k Thermistor and LJTick-Resistance-10k and are at 22 °C. From the datasheet the expected resistance is 12488 at 20 °C and 10000 ohms at 25 °C, so we interpolate to come up with an expected resistance at 22 °C:

```
R22 = 12488 - ( ((22-20)/(25-20)) * (12488-10000) ) = 11493 ohms
```

Now we use the equation from the LJTick-Resistance Datasheet to calculate the expected voltage:

```
Vout = Vref*Rfixed/(Runknown+Rfixed) = 2.5*10000/(11493+10000) = 1.163 volts
```

Compare the expected resistance and voltage to AIN#_EF_READ_B and AIN#_EF_READ_C.

Resistance to Temperature

Use one of the various online calculators from daycounter.com to check the resistance to temperature conversion. In this case we have the Steinhart-Hart coefficients and the first calculator on that page is applicable. We put in A=0.003354016, B=0.000256985, C=0.000002620, D=0.00000006383, Rt=10000, and R=11493, and we get a result of 21.85 °C. Close enough to 22.0 to tell us things are working right. The main source of error here is the fact that we did a linear interpolation to get expected resistance, but resistance is very non-linear. We know this is the main source of error because if we put the actual table value of 12488 ohms for 20 °C, the calculator gives us 19.998 °C.

Example

This example configures a LabJack to read from a Vishay NTCLE100E3103 10k Thermistor using a LabJack LJTick-Resistance to complete the excitation circuit. The LJTick-Resistance is connected to the AIN0/1 terminal block. The thermistor is connected between the Vref and INA terminals on the LJTick-Resistance.

```
AIN0_EF_INDEX = 50 -- Steinhart-Hart
AIN0_EF_CONFIG_A = 1 -- Output degrees Celsius.
AIN0_EF_CONFIG_B = 4 -- Excitation circuit #4.
AIN0_EF_CONFIG_C = 0 -- Second AIN, not used for excitation circuit #4.
AIN0_EF_CONFIG_D = 2.5 -- 2.5 V provided by the LJTick-Resistance
AIN0_EF_CONFIG_E = 10000 -- 10 kΩ shunt resistor provided by the LJTick-Resistance-10k.
AIN0_EF_CONFIG_F = 10000 -- R25 The nominal resistance of the thermistor at 25 ºC.
AIN0_EF_CONFIG_G = 0.003354016 -- Constants from the thermistor's datasheet.
AIN0_EF_CONFIG_H = 0.000256985
AIN0_EF_CONFIG_I = 0.000002620
AIN0_EF_CONFIG_J = 0.00000006383
```

Results:

```
AIN0_EF_READ_A = 23.19 -- Temperature of the thermistor. (°C)
AIN0_EF_READ_B = 10829.4 -- Calculated resistance. (Ω)
AIN0_EF_READ_C = 1.299774 -- Voltage across the thermistor. (V)
```

14.1.6 Resistance [T-Series Datasheet]

Overview

AIN#_EF_INDEX: **4**

This Resistance Extended Feature will calculate the resistance of a given circuit element.

Configuration

To configure, write to the following registers.

AIN#_EF_CONFIG_B - Excitation Circuit Index: The index of the voltage divider excitation circuit to be used.

See 14.1.0.1 Excitation Circuits for circuit indices.

AIN#_EF_CONFIG_C - 2nd AIN: Channel Number to Measure Vresistor: For excitation circuits 3 and 5 this is the extra AIN used to measure the voltage across the fixed resistor. Ignored for other excitation circuits.

AIN#_EF_CONFIG_D - Excitation Volts or Amps: For excitation circuit 2 this is the fixed amps of the current source. For excitation circuit 4 this is the fixed volts of the voltage source. Ignored for other excitation circuits.

AIN#_EF_CONFIG_E - Fixed Resistor Ohms: For excitation circuits 3, 4 and 5, this is the ohms of the fixed resistor.

Remarks

The normal analog input settings are used for negative channel, resolution index, settling, and range.

T7: If the voltage will stay below 1.0V, use the 1.0V range for improved resolution and accuracy.

T8: For improved resolution and accuracy, use the lowest voltage range that is appropriate for your signal's voltage min/max. For example, if the signal's min/max is -0.0064V/0.0549V, use the T8 voltage range of ±0.075V.

Results

For results, read the following registers.

AIN#_EF_READ_A: Resistance of sensor
AIN#_EF_READ_B: Voltage of sensor
AIN#_EF_READ_C: Current through sensor

Only reading AIN#_EF_READ_A triggers a new measurement, so you must always read A before reading B or C.

Example

200UA current source, circuit #0:

An unknown resistor to be measured (a 10 kΩ resistor is used in this test) is connected in series with the 200UA current source available on the T7.

```
AIN0_EF_INDEX = 4
AIN0_EF_CONFIG_B = 0 -- Excitation circuit 0 (200 µA current
source)
```

Typical reading results:

- AIN0_EF_READ_A returns 10089.7 Ω
- AIN0_EF_READ_B returns 2.012 V
- AIN0_EF_READ_C returns 199 µA

LJTick-Resistance-1k, circuit #4:

Connect a resistor to measure from Vref to VINA on an LJTick-Resistance-1k that is plugged into the AIN0/AIN1 block.

```
AIN0_EF_INDEX = 4       // Set AIN_EF0 to RTD100.
AIN0_EF_CONFIG_B = 4     // Set excitation circuit to 4.
AIN0_EF_CONFIG_D = 2.50  // Vref on the LJTR is 2.50 volts.
AIN0_EF_CONFIG_E = 1000  // We are using the 1k version of the LJTR.
```

Now each read of AIN0_EF_READ_A will measure the voltage on AIN0 and do the voltage divider math to determine the resistance of the resistor.

14.1.7 Average Min Max [T-Series Datasheet]

Overview

AIN#_EF_INDEX: **3**

This Average Mix Max Extended Feature will sample an analog input a specified number of times at a specified rate, then calculate the average, min, and max voltage.

If you are using this feature to reduce noise by oversampling-and-averaging, consider maximizing resolution first, since that is the best and fastest way to reduce noise.

This feature internally uses stream mode.

Configuration

To configure, write to the following registers.

AIN#_EF_CONFIG_A - Number of Samples: The number of samples to be acquired.

- Default: 200
- Max: 16384

AIN#_EF_CONFIG_D - Scan Rate: The frequency at which samples will be collected.

- Default: 6000

Sample Time

The maximum possible sample time is 180 ms.

Sample time is the number of samples divided by the sample frequency. For example, the period is 16.7 ms when using a scan rate of 6000 Hz and 100 samples:

```
100 samples / 6000 samples per second = 16.7
ms
```

If the signal has a known periodic component, then setting the sample time to an even multiple of the period will generally improve results.

Stream Configuration

This extended feature internally uses Stream-Burst to acquire the data set, so stream AIN configurations apply.

- Crucially, set STREAM_RESOLUTION_INDEX between 0 and 8. Resolution index 9 is not

supported in stream.

Results

For results, read the following registers.

AIN#_EF_READ_A: Average volts
AIN#_EF_READ_B: Max volts
AIN#_EF_READ_C: Min volts

Only reading AIN#_EF_READ_A triggers a new measurement. Because multiple measurements are taken, a read of AIN#_EF_READ_A blocks for the length of the sample time.

Example

To measure a 10 Hz sine wave with amplitude 0.1 V and DC offset of 1.2 V, set the number of samples to 100 and the scan rate to 1000. The total acquisition time is 100 ms, which is an even multiple of the signal's period.

```
AIN0_EF_INDEX = 3
AIN0_EF_CONFIG_A = 100 -- Number of samples
AIN0_EF_CONFIG_D = 1000 -- Scan rate
```

Results, with noise levels less than 600 µV:

```
AIN0_EF_READ_A = 1.201 V -- Average volts
AIN0_EF_READ_B = 1.301 V -- Max volts
AIN0_EF_READ_C = 1.101 V -- Min volts
```

We can change the sample time to emphasize the benefit of matching to the period. If we change AIN0_EF_CONFIG_A to 70 (an odd multiple of the signal's period), then the results are still centered on 1.2, 1.3, and 1.1 V—but noise levels are ±5 mV.

14.1.8 Average and Threshold [T-Series Datasheet]

Overview

AIN#_EF_INDEX: **5**

This Average and Threshold Extended Feature will read an input a specified number of times at a specified rate, then average the readings and set a flag if the average is above a specified threshold. The T8 can also return the median and standard deviation of the readings.

Can be used with analog inputs or digital inputs.

This feature internally uses stream mode.

Configuration

To configure, write to the following registers.

AIN#_EF_CONFIG_A - Number of Samples: Number of samples to be acquired.

- Default: 200
- Max: 16384

AIN#_EF_CONFIG_B - Digital Override: Selects whether this AIN-EF takes samples from the default analog line or from a specified digital line.

> *Digital Override is not supported on T8*

- When set to zero, the normal AIN# will be used.
- This is the default.
- For example, if AIN2_EF_CONFIG_B is 0, samples are collected from AIN2 when AIN2_EF_READ_A is read.
- When set to non-zero, sets the Modbus address of a digital IO.
- For example, if AIN2_EF_CONFIG_B is 2001 (the address of FIO1), samples are collected from FIO1 when AIN2_EF_READ_A is read.

AIN#_EF_CONFIG_D - Scan Rate: The frequency at which samples will be collected.

- Default: 6000

AIN#_EF_CONFIG_E - Threshold: If the computed average is above this value, then the threshold flag (AIN#_EF_READ_A) will be set.

Sample time

The maximum possible sample time is 180 ms.

Sample time is the number of samples divided by the sample frequency. For example, the period is 16.7 ms when using a scan rate of 6000 Hz and 100 samples:

```
100 samples / 6000 samples per second = 16.7
ms
```

If the signal has a known periodic component, then setting the sample time to an even multiple of the period will generally improve results.

Stream Configuration

This extended feature internally uses Stream-Burst to acquire the data set, so stream AIN configurations apply.

- Crucially, set STREAM_RESOLUTION_INDEX between 0 and 8. Resolution index 9 is not supported in stream.

Results

For results, read the following registers.

AIN#_EF_READ_A - Threshold Flag: Returns 1.0 if the average is greater than the threshold, 0.0 if not.
AIN#_EF_READ_B - Average: The average of the collected samples.

AIN#_EF_READ_C - Median: The median of the collected samples. *T8 Only

AIN#_EF_READ_D - Standard Deviation: The standard deviation of the collected samples. *T8 Only

Only reading AIN#_EF_READ_A triggers a new measurement. Because multiple measurements are taken, a read of AIN#_EF_READ_A blocks for the length of the sample time.

Example - Analog Input

A 10 Hz sine wave with magnitude 0.1 V and DC offset of 1.2 V is connected to AIN2. A threshold value less than the expected 1.2 V average will produce a positive result.

```
AIN2_EF_INDEX = 5
AIN2_EF_CONFIG_A = 100   -- 100 samples
AIN2_EF_CONFIG_B = 0     -- Use AIN2 to collect
samples
AIN2_EF_CONFIG_D = 1000  -- Scan rate
AIN2_EF_CONFIG_E = 1.15  -- Threshold
```

Reading register AIN2_EF_READ_A returns 1.0, indicating that the average is greater than the threshold. If, instead, the threshold written to AIN2_EF_CONFIG_E was 1.25, then AIN2_EF_READ_A would then return 0.0 to indicate that the threshold was greater than the average.

Example - Digital Input

A 10 Hz sine wave with magnitude 1.2 V and DC offset of 1.6 V is connected to FIO0. The input signal should be interpreted as high two out of three times. In this case, the threshold of 0.65 is a ratio. If the average of the readings is 0.65 or greater, the AIN2_EF_READ_A register should be 1.0.

```
AIN2_EF_INDEX = 5
AIN2_EF_CONFIG_A = 100   -- 100 samples
AIN2_EF_CONFIG_B = 2000  -- Modbus address of FIO0
AIN2_EF_CONFIG_D = 1000  -- Scan rate
AIN2_EF_CONFIG_E = 0.65  -- Threshold
```

Reading register AIN2_EF_READ_A returns 1.0 and the calculated average is 66%.

14.2 Extended Channels (T7 Only) [T-Series Datasheet]

Overview - T7 Only

The T7 has 14 built-in analog inputs (AIN0 through AIN13). For applications that need more, up to 112 total external analog inputs can be utilized by multiplexing the built-in analog inputs. Each built-in analog channel may be multiplexed into 8 new channels, where the starting extended channel number is:

```
startingExtendedChannel = builtinChannel * 8 + 16
```

The extended channel range begins at AIN16.

When a built-in channel is multiplexed, the original channel number becomes unavailable.

For example, if AIN4 is multiplexed, AIN48 through AIN55 are exposed as usable AIN and AIN4 is no longer available.

Differential Channel Pairings

For differential channels in the extended range, a channel may be the positive channel if it is one of the following ranges:

- AIN16 through AIN23
- AIN32 through AIN39
- AIN48 through AIN55
- AIN64 through AIN71
- AIN80 through AIN87
- AIN96 through AIN103
- AIN112 through AIN119

The negative channel is 8 higher than the positive channel. For example, the positive channel AIN48 is paired with AIN56 as the negative channel.

The differential channel pairs for the built-in AIN0 through AIN13 are adjacent even/odd pairs such that the positive channel is even and the negative channel is greater than the positive channel by 1. Since the built-in AIN are multiplexed as described above, an extended negative channel is 8 higher than its corresponding extended positive channel. For example, a valid differential extended channel pair would be a positive channel of AIN70 and AIN78 as the negative channel, since:

- AIN70 maps to AIN6 and
- AIN78 maps to AIN7.

For more information on differential extended channels, see the Mux80 Datasheet.

MIO

The DB37 connector has 3 MIO lines designed to address expansion multiplexer ICs (integrated circuits), allowing for up to 112 total external analog inputs. Whenever a read is done on analog input channel numbers 16 to 127, the T7 will automatically control the MIO lines to read the correct multiplexed AIN.

Mux80

The Mux80 accessory is used to multiplex AIN4 through AIN13, expanding them from 10 inputs to 80 inputs. AIN0 through AIN3 are still available (on the screw terminals of the T7). The extended channels can be read using the following registers. For further details, see the Mux80 Datasheet.

Mux80 Extended Channels

Name	Start Address	Type	Access
AIN#(48:127)	96	FLOAT32	R

AIN#(48:127)
- Starting Address: 96

Returns the voltage of the specified analog input.

- Data type: FLOAT32 (type index = 3)
- Read-only
- This register may be streamed

Expanded Names	Addresses
AIN48, AIN49, AIN50, AIN51, AIN52, AIN53, AIN54, AIN55, AIN56, AIN57, AIN58, AIN59, AIN60, AIN61, AIN62, AIN63, AIN64, AIN65, AIN66, AIN67, AIN68, AIN69, AIN70, AIN71, AIN72, AIN73, AIN74, AIN75, AIN76, AIN77, AIN78, AIN79, AIN80, AIN81, AIN82, AIN83, AIN84, AIN85, AIN86, AIN87, AIN88, AIN89, AIN90, AIN91, AIN92, AIN93, AIN94, AIN95, AIN96, AIN97, AIN98, AIN99, AIN100, AIN101, AIN102, AIN103, AIN104, AIN105, AIN106, AIN107, AIN108, AIN109, AIN110, AIN111, AIN112, AIN113, AIN114, AIN115, AIN116, AIN117, AIN118, AIN119, AIN120, AIN121, AIN122, AIN123, AIN124, AIN125, AIN126, AIN127	96, 98, 100, 102, 104, 106, 108, 110, 112, 114, 116, 118, 120, 122, 124, 126, 128, 130, 132, 134, 136, 138, 140, 142, 144, 146, 148, 150, 152, 154, 156, 158, 160, 162, 164, 166, 168, 170, 172, 174, 176, 178, 180, 182, 184, 186, 188, 190, 192, 194, 196, 198, 200, 202, 204, 206, 208, 210, 212, 214, 216, 218, 220, 222, 224, 226, 228, 230, 232, 234, 236, 238, 240, 242, 244, 246, 248, 250, 252, 254

Note that when using the Mux80 board, the T7's MIO(0-2) lines are consumed for multiplexer signaling.

AIN Configurations on the Mux80

Use the following registers to configure AIN on the Mux80.

Name

Name	Start Address	Type	Access
AIN#(48:127)_NEGATIVE_CH	41048	UINT16	R/W

AIN#(48:127)_NEGATIVE_CH
- Starting Address: 41048

Specifies the negative channel to be used for each positive channel. 199=Default=> Single-Ended.

- Data type: UINT16 (type index = 0)
- Readable and writable
- Default value: 199
- T8:
  - Ignored. This register has no meaning on the T8.
- T7:
  - For base differential channels, positive must be an even channel from 0-12 and negative must be positive+1. For extended channels 16-127, see Mux80 datasheet.

Expanded Names	Addresses

AIN48_NEGATIVE_CH, AIN49_NEGATIVE_CH,	41048, 41049,
AIN50_NEGATIVE_CH, AIN51_NEGATIVE_CH,	41050, 41051,
AIN52_NEGATIVE_CH, AIN53_NEGATIVE_CH,	41052, 41053,
AIN54_NEGATIVE_CH, AIN55_NEGATIVE_CH,	41054, 41055,
AIN56_NEGATIVE_CH, AIN57_NEGATIVE_CH,	41056, 41057,
AIN58_NEGATIVE_CH, AIN59_NEGATIVE_CH,	41058, 41059,
AIN60_NEGATIVE_CH, AIN61_NEGATIVE_CH,	41060, 41061,
AIN62_NEGATIVE_CH, AIN63_NEGATIVE_CH,	41062, 41063,
AIN64_NEGATIVE_CH, AIN65_NEGATIVE_CH,	41064, 41065,
AIN66_NEGATIVE_CH, AIN67_NEGATIVE_CH,	41066, 41067,
AIN68_NEGATIVE_CH, AIN69_NEGATIVE_CH,	41068, 41069,
AIN70_NEGATIVE_CH, AIN71_NEGATIVE_CH,	41070, 41071,
AIN72_NEGATIVE_CH, AIN73_NEGATIVE_CH,	41072, 41073,
AIN74_NEGATIVE_CH, AIN75_NEGATIVE_CH,	41074, 41075,
AIN76_NEGATIVE_CH, AIN77_NEGATIVE_CH,	41076, 41077,
AIN78_NEGATIVE_CH, AIN79_NEGATIVE_CH,	41078, 41079,
AIN80_NEGATIVE_CH, AIN81_NEGATIVE_CH,	41080, 41081,
AIN82_NEGATIVE_CH, AIN83_NEGATIVE_CH,	41082, 41083,
AIN84_NEGATIVE_CH, AIN85_NEGATIVE_CH,	41084, 41085,
AIN86_NEGATIVE_CH, AIN87_NEGATIVE_CH,	41086, 41087,
AIN88_NEGATIVE_CH, AIN89_NEGATIVE_CH,	41088, 41089,
AIN90_NEGATIVE_CH, AIN91_NEGATIVE_CH,	41090, 41091,
AIN92_NEGATIVE_CH, AIN93_NEGATIVE_CH,	41092, 41093,
AIN94_NEGATIVE_CH, AIN95_NEGATIVE_CH,	41094, 41095,
AIN96_NEGATIVE_CH, AIN97_NEGATIVE_CH,	41096, 41097,
AIN98_NEGATIVE_CH, AIN99_NEGATIVE_CH,	41098, 41099,
AIN100_NEGATIVE_CH, AIN101_NEGATIVE_CH,	41100, 41101,
AIN102_NEGATIVE_CH, AIN103_NEGATIVE_CH,	41102, 41103,
AIN104_NEGATIVE_CH, AIN105_NEGATIVE_CH,	41104, 41105,
AIN106_NEGATIVE_CH, AIN107_NEGATIVE_CH,	41106, 41107,
AIN108_NEGATIVE_CH, AIN109_NEGATIVE_CH,	41108, 41109,
AIN110_NEGATIVE_CH, AIN111_NEGATIVE_CH,	41110, 41111,
AIN112_NEGATIVE_CH, AIN113_NEGATIVE_CH,	41112, 41113,
AIN114_NEGATIVE_CH, AIN115_NEGATIVE_CH,	41114, 41115,
AIN116_NEGATIVE_CH, AIN117_NEGATIVE_CH,	41116, 41117,
AIN118_NEGATIVE_CH, AIN119_NEGATIVE_CH,	41118, 41119,
AIN120_NEGATIVE_CH, AIN121_NEGATIVE_CH,	41120, 41121,
AIN122_NEGATIVE_CH, AIN123_NEGATIVE_CH,	41122, 41123,
AIN124_NEGATIVE_CH, AIN125_NEGATIVE_CH,	41124, 41125,
AIN126_NEGATIVE_CH, AIN127_NEGATIVE_CH	41126, 41127

AIN_ALL_NEGATIVE_CH	43902	UINT16	R/W

AIN_ALL_NEGATIVE_CH
- Address: 43902

A write to this global parameter affects all AIN. Writing 1 will set all AINs to differential. Writing 199 will set all AINs to single-ended. A read will return 1 if all AINs are set to differential and 199 if all AINs are set to single-ended. If AIN configurations are not consistent 0xFFFF will be returned.

- Data type: UINT16 (type index = 0)
- Readable and writable
- Default value: 199
- T8:
  - Ignored. This register has no meaning on the T8.
- T7:
  - Minimum firmware version: 0.9328

More details about single-ended vs differential are in the Single-ended or Differential section of the main AIN page.

Resolution Index Registers

Name	Start Address	Type	Access
AIN#(48:127)_RESOLUTION_INDEX	41548	UINT16	R/W

AIN#(48:127)_RESOLUTION_INDEX
- Starting Address: 41548

The resolution index for command-response and AIN-EF readings. A larger resolution index generally results in lower noise and longer sample times.

- Data type: UINT16 (type index = 0)
- Readable and writable
- Default value: 0
- T8:
  - Valid values: 0-16. A value of 0 will instruct the T8 to use the best resolution for the rate specified.
- T7:
  - Valid values: 0-8 for T7, 0-12 for T7-Pro. Default value of 0 corresponds to an index of 8 (T7) or 9 (T7-Pro).
- T4:
  - Valid values: 0-5. Default value of 0 corresponds to an index of 5.

Expanded Names	Addresses
AIN48_RESOLUTION_INDEX, AIN49_RESOLUTION_INDEX, AIN50_RESOLUTION_INDEX, AIN51_RESOLUTION_INDEX, AIN52_RESOLUTION_INDEX, AIN53_RESOLUTION_INDEX,	41548, 41549, 41550, 41551, 41552, 41553,

AIN54_RESOLUTION_INDEX,	41554,
AIN55_RESOLUTION_INDEX,	41555,
AIN56_RESOLUTION_INDEX,	41556,
AIN57_RESOLUTION_INDEX,	41557,
AIN58_RESOLUTION_INDEX,	41558,
AIN59_RESOLUTION_INDEX,	41559,
AIN60_RESOLUTION_INDEX,	41560,
AIN61_RESOLUTION_INDEX,	41561,
AIN62_RESOLUTION_INDEX,	41562,
AIN63_RESOLUTION_INDEX,	41563,
AIN64_RESOLUTION_INDEX,	41564,
AIN65_RESOLUTION_INDEX,	41565,
AIN66_RESOLUTION_INDEX,	41566,
AIN67_RESOLUTION_INDEX,	41567,
AIN68_RESOLUTION_INDEX,	41568,
AIN69_RESOLUTION_INDEX,	41569,
AIN70_RESOLUTION_INDEX,	41570,
AIN71_RESOLUTION_INDEX,	41571,
AIN72_RESOLUTION_INDEX,	41572,
AIN73_RESOLUTION_INDEX,	41573,
AIN74_RESOLUTION_INDEX,	41574,
AIN75_RESOLUTION_INDEX,	41575,
AIN76_RESOLUTION_INDEX,	41576,
AIN77_RESOLUTION_INDEX,	41577,
AIN78_RESOLUTION_INDEX,	41578,
AIN79_RESOLUTION_INDEX,	41579,
AIN80_RESOLUTION_INDEX,	41580,
AIN81_RESOLUTION_INDEX,	41581,
AIN82_RESOLUTION_INDEX,	41582,
AIN83_RESOLUTION_INDEX,	41583,
AIN84_RESOLUTION_INDEX,	41584,
AIN85_RESOLUTION_INDEX,	41585,
AIN86_RESOLUTION_INDEX,	41586,
AIN87_RESOLUTION_INDEX,	41587,
AIN88_RESOLUTION_INDEX,	41588,
AIN89_RESOLUTION_INDEX,	41589,
AIN90_RESOLUTION_INDEX,	41590,
AIN91_RESOLUTION_INDEX,	41591,
AIN92_RESOLUTION_INDEX,	41592,
AIN93_RESOLUTION_INDEX,	41593,
AIN94_RESOLUTION_INDEX,	41594,
AIN95_RESOLUTION_INDEX,	41595,
AIN96_RESOLUTION_INDEX,	41596,
AIN97_RESOLUTION_INDEX,	41597,
AIN98_RESOLUTION_INDEX,	41598,
AIN99_RESOLUTION_INDEX,	41599,
AIN100_RESOLUTION_INDEX,	41600,
AIN101_RESOLUTION_INDEX,	41601,
AIN102_RESOLUTION_INDEX,	41602,
AIN103_RESOLUTION_INDEX,	41603,
AIN104_RESOLUTION_INDEX,	41604,
AIN105_RESOLUTION_INDEX,	41605,
AIN106_RESOLUTION_INDEX,	41606,
AIN107_RESOLUTION_INDEX,	41607,
AIN108_RESOLUTION_INDEX,	41608,
AIN109_RESOLUTION_INDEX,	41609,
AIN110_RESOLUTION_INDEX,	41610,
AIN111_RESOLUTION_INDEX,	41611,
AIN112_RESOLUTION_INDEX,	41612,
AIN113_RESOLUTION_INDEX,	41613,
AIN114_RESOLUTION_INDEX,	41614,
AIN115_RESOLUTION_INDEX,	41615,
AIN116_RESOLUTION_INDEX,	41616,

AIN117_RESOLUTION_INDEX,	41617,
AIN118_RESOLUTION_INDEX,	41618,
AIN119_RESOLUTION_INDEX,	41619,
AIN120_RESOLUTION_INDEX,	41620,
AIN121_RESOLUTION_INDEX,	41621,
AIN122_RESOLUTION_INDEX,	41622,
AIN123_RESOLUTION_INDEX,	41623,
AIN124_RESOLUTION_INDEX,	41624,
AIN125_RESOLUTION_INDEX,	41625,
AIN126_RESOLUTION_INDEX,	41626,
AIN127_RESOLUTION_INDEX	41627

AIN_ALL_RESOLUTION_INDEX	43903	UINT16	R/W

AIN_ALL_RESOLUTION_INDEX
- Address: 43903

The resolution index for command-response and AIN-EF readings. A larger resolution index generally results in lower noise and longer sample times. A write to this global parameter affects all AIN. A read will return the correct setting if all channels are set the same, but otherwise will return 0xFFFF.

- Data type: UINT16 (type index = 0)
- Readable and writable
- Default value: 0
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Valid values: 0-8 for T7, 0-12 for T7-Pro. Default value of 0 corresponds to an index of 8 (T7) or 9 (T7-Pro).
  - Minimum firmware version: 0.9328
- T4:
  - Valid values: 0-5. Default value of 0 corresponds to an index of 5.

STREAM_RESOLUTION_INDEX	4010	UINT32	R/W

STREAM_RESOLUTION_INDEX
- Address: 4010

The resolution index for stream readings. A larger resolution index generally results in lower noise and longer sample times.

- Data type: UINT32 (type index = 1)
- Readable and writable
- Default value: 0
- T8:
  - T8 description 0-16. A value of 0 will use the best resolution for the specified data rate.
- T7:
  - Valid values: 0-8. Default value of 0 corresponds to an index of 1.
- T4:
  - Valid values: 0-5. Default value of 0 corresponds to an index of 1.

More details about resolution are in the Resolution Index section of the main AIN page.

Range Registers

Name	Start Address	Type	Access
AIN#(48:127)_RANGE	40096	FLOAT32	R/W

AIN#(48:127)_RANGE
- Starting Address: 40096

The range/span of each analog input. Write the highest expected input voltage.

- Data type: FLOAT32 (type index = 3)
- Readable and writable
- Default value: 0
- T8:
  - Specify the maximum expected voltage. The T8 will select the best gain for the expected voltage.
- T7:
  - Valid values/ranges: 0.0=Default → ±10V. 10.0 → ±10V, 1.0 → ±1V, 0.1 → ±0.1V, and 0.01 → ±0.01V.
- T4:
  - Valid values/ranges: 0.0=Default → 0-2.5 V on LV lines and ±10 V on HV lines.

Expanded Names	Addresses

AIN48_RANGE, AIN49_RANGE,	40096, 40098,
AIN50_RANGE, AIN51_RANGE,	40100, 40102,
AIN52_RANGE, AIN53_RANGE,	40104, 40106,
AIN54_RANGE, AIN55_RANGE,	40108, 40110,
AIN56_RANGE, AIN57_RANGE,	40112, 40114,
AIN58_RANGE, AIN59_RANGE,	40116, 40118,
AIN60_RANGE, AIN61_RANGE,	40120, 40122,
AIN62_RANGE, AIN63_RANGE,	40124, 40126,
AIN64_RANGE, AIN65_RANGE,	40128, 40130,
AIN66_RANGE, AIN67_RANGE,	40132, 40134,
AIN68_RANGE, AIN69_RANGE,	40136, 40138,
AIN70_RANGE, AIN71_RANGE,	40140, 40142,
AIN72_RANGE, AIN73_RANGE,	40144, 40146,
AIN74_RANGE, AIN75_RANGE,	40148, 40150,
AIN76_RANGE, AIN77_RANGE,	40152, 40154,
AIN78_RANGE, AIN79_RANGE,	40156, 40158,
AIN80_RANGE, AIN81_RANGE,	40160, 40162,
AIN82_RANGE, AIN83_RANGE,	40164, 40166,
AIN84_RANGE, AIN85_RANGE,	40168, 40170,
AIN86_RANGE, AIN87_RANGE,	40172, 40174,
AIN88_RANGE, AIN89_RANGE,	40176, 40178,
AIN90_RANGE, AIN91_RANGE,	40180, 40182,
AIN92_RANGE, AIN93_RANGE,	40184, 40186,
AIN94_RANGE, AIN95_RANGE,	40188, 40190,
AIN96_RANGE, AIN97_RANGE,	40192, 40194,
AIN98_RANGE, AIN99_RANGE,	40196, 40198,
AIN100_RANGE, AIN101_RANGE,	40200, 40202,
AIN102_RANGE, AIN103_RANGE,	40204, 40206,
AIN104_RANGE, AIN105_RANGE,	40208, 40210,
AIN106_RANGE, AIN107_RANGE,	40212, 40214,
AIN108_RANGE, AIN109_RANGE,	40216, 40218,
AIN110_RANGE, AIN111_RANGE,	40220, 40222,
AIN112_RANGE, AIN113_RANGE,	40224, 40226,
AIN114_RANGE, AIN115_RANGE,	40228, 40230,
AIN116_RANGE, AIN117_RANGE,	40232, 40234,
AIN118_RANGE, AIN119_RANGE,	40236, 40238,
AIN120_RANGE, AIN121_RANGE,	40240, 40242,
AIN122_RANGE, AIN123_RANGE,	40244, 40246,
AIN124_RANGE, AIN125_RANGE,	40248, 40250,
AIN126_RANGE, AIN127_RANGE	40252, 40254

AIN_ALL_RANGE	43900	FLOAT32	R/W

AIN_ALL_RANGE
- Address: 43900

A write to this global parameter affects all AIN. A read will return the correct setting if all channels are set the same, but otherwise will return -9999.

- Data type: FLOAT32 (type index = 3)
- Readable and writable
- Default value: 0
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 0.9328

More details about range are in the Range / Gain section of the main AIN page.

Settling Registers

Name	Start Address	Type	Access
AIN#(48:127)_SETTLING_US	42096	FLOAT32	R/W

AIN#(48:127)_SETTLING_US
- Starting Address: 42096

Settling time for command-response and AIN-EF readings.

- Data type: FLOAT32 (type index = 3)
- Readable and writable
- Default value: 0
- T8:
  - Ignored. This register has no meaning on the T8.
- T7:
  - 0 = Auto. Max is 50000 (microseconds).
  - Minimum firmware version: 0.9328
- T4:
  - 0 = Auto. Max is 10000 (microseconds).

Expanded Names	Addresses

AIN48_SETTLING_US, AIN49_SETTLING_US,	42096, 42098,
AIN50_SETTLING_US, AIN51_SETTLING_US,	42100, 42102,
AIN52_SETTLING_US, AIN53_SETTLING_US,	42104, 42106,
AIN54_SETTLING_US, AIN55_SETTLING_US,	42108, 42110,
AIN56_SETTLING_US, AIN57_SETTLING_US,	42112, 42114,
AIN58_SETTLING_US, AIN59_SETTLING_US,	42116, 42118,
AIN60_SETTLING_US, AIN61_SETTLING_US,	42120, 42122,
AIN62_SETTLING_US, AIN63_SETTLING_US,	42124, 42126,
AIN64_SETTLING_US, AIN65_SETTLING_US,	42128, 42130,
AIN66_SETTLING_US, AIN67_SETTLING_US,	42132, 42134,
AIN68_SETTLING_US, AIN69_SETTLING_US,	42136, 42138,
AIN70_SETTLING_US, AIN71_SETTLING_US,	42140, 42142,
AIN72_SETTLING_US, AIN73_SETTLING_US,	42144, 42146,
AIN74_SETTLING_US, AIN75_SETTLING_US,	42148, 42150,
AIN76_SETTLING_US, AIN77_SETTLING_US,	42152, 42154,
AIN78_SETTLING_US, AIN79_SETTLING_US,	42156, 42158,
AIN80_SETTLING_US, AIN81_SETTLING_US,	42160, 42162,
AIN82_SETTLING_US, AIN83_SETTLING_US,	42164, 42166,
AIN84_SETTLING_US, AIN85_SETTLING_US,	42168, 42170,
AIN86_SETTLING_US, AIN87_SETTLING_US,	42172, 42174,
AIN88_SETTLING_US, AIN89_SETTLING_US,	42176, 42178,
AIN90_SETTLING_US, AIN91_SETTLING_US,	42180, 42182,
AIN92_SETTLING_US, AIN93_SETTLING_US,	42184, 42186,
AIN94_SETTLING_US, AIN95_SETTLING_US,	42188, 42190,
AIN96_SETTLING_US, AIN97_SETTLING_US,	42192, 42194,
AIN98_SETTLING_US, AIN99_SETTLING_US,	42196, 42198,
AIN100_SETTLING_US, AIN101_SETTLING_US,	42200, 42202,
AIN102_SETTLING_US, AIN103_SETTLING_US,	42204, 42206,
AIN104_SETTLING_US, AIN105_SETTLING_US,	42208, 42210,
AIN106_SETTLING_US, AIN107_SETTLING_US,	42212, 42214,
AIN108_SETTLING_US, AIN109_SETTLING_US,	42216, 42218,
AIN110_SETTLING_US, AIN111_SETTLING_US,	42220, 42222,
AIN112_SETTLING_US, AIN113_SETTLING_US,	42224, 42226,
AIN114_SETTLING_US, AIN115_SETTLING_US,	42228, 42230,
AIN116_SETTLING_US, AIN117_SETTLING_US,	42232, 42234,
AIN118_SETTLING_US, AIN119_SETTLING_US,	42236, 42238,
AIN120_SETTLING_US, AIN121_SETTLING_US,	42240, 42242,
AIN122_SETTLING_US, AIN123_SETTLING_US,	42244, 42246,
AIN124_SETTLING_US, AIN125_SETTLING_US,	42248, 42250,
AIN126_SETTLING_US, AIN127_SETTLING_US	42252, 42254

AIN_ALL_SETTLING_US	43904	FLOAT32	R/W

AIN_ALL_SETTLING_US
- Address: 43904

Settling time for command-response and AIN-EF readings. A write to this global parameter affects all AIN. A read will return the correct setting if all channels are set the same, but otherwise will return -9999. Max is 50,000 us.

- Data type: FLOAT32 (type index = 3)
- Readable and writable
- Default value: 0
- T8:
  - Ignored. This register has no meaning on the T8.
- T7:
  - 0 = Auto. Max is 50000 (microseconds).
  - Minimum firmware version: 0.9328
- T4:
  - 0 = Auto. Max is 10000 (microseconds).

STREAM_SETTLING_US	4008	FLOAT32	R/W

STREAM_SETTLING_US
- Address: 4008

Time in microseconds to allow signals to settle after switching the mux. Does not apply to the 1st channel in the scan list, as that settling is controlled by scan rate (the time from the last channel until the start of the next scan). Default=0. When set to less than 1, automatic settling will be used. The automatic settling behavior varies by device.

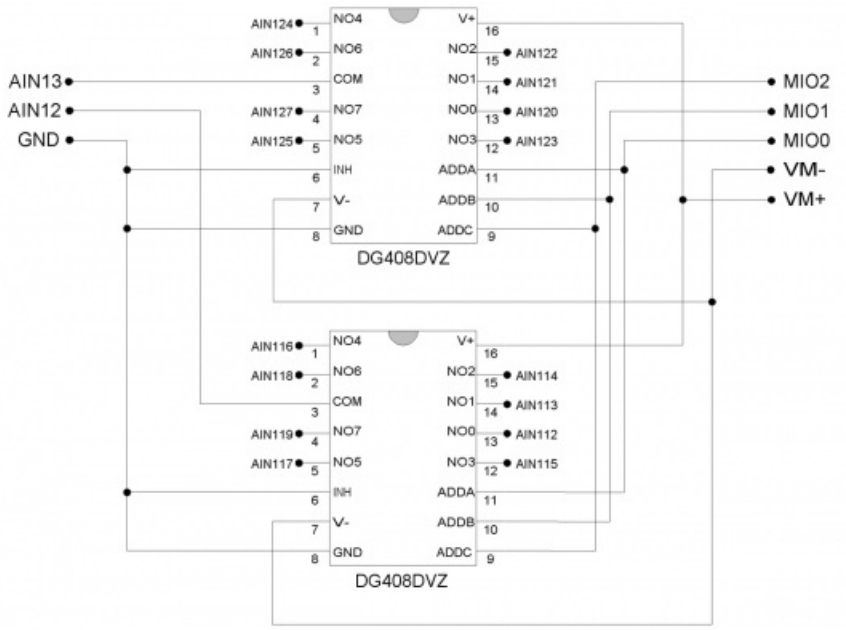
- Data type: FLOAT32 (type index = 3)
- Readable and writable
- Default value: 0
- T8:
  - Ignored. The T8's analog input chain does not rely on settling time.
- T7:
  - The T7 will select the settling time based on resolution and gain settings. When the sample rate is above 60 kHz, settling time will be set to ~5 us. Max is 4400.
- T4:
  - The T4 will default to 10 us. When the scan rate is above 20 kHz, settling time will be set to 5 us.

More details about settling are in the Settling section of the main AIN page.

Custom Multiplexing Applications

Custom multiplexing and signal conditioning boards can be designed and automatically controlled by a T7 using the DG408 multiplexing IC from Intersil. The DG408 multiplexing IC from Intersil is the recommended multiplexer, and a convenient ±12 volt power supply is available on the DB37 so the multiplexers can pass bipolar signals (see Vm+/Vm- documentation in Section 16.0). These are the same ICs that do the device's internal channel multiplexing as well as the ICs integrated into the Mux80. Figure 14.2.1 shows the typical connections for a pair of multiplexers.

Figure 14.2.1 Typical DG408DVZ Multiplexer Connections.



Below is a table indicating the MIO states required for controlling 14 DG408 multiplexing ICs that each route to a single AIN line and the extended analog input channel that must be read for a single-ended signal to be measured correctly.

Output State				**Expanded Channel Multiplexing Mapping**											
MIO0	MIO1	MIO2		AIN0	AIN1	AIN2	AIN3	AIN4	AIN5	AIN6	AIN7	AIN8	AIN9	AIN10	AIN11
0	0	0		16	24	32	40	48	56	64	72	80	88	96	104
1	0	0		17	25	33	41	49	57	65	73	81	89	97	105
0	1	0		18	26	34	42	50	58	66	74	82	90	98	106
1	1	0		19	27	35	43	51	59	67	75	83	91	99	107
0	0	1		20	28	36	44	52	60	68	76	84	92	100	108
1	0	1		21	29	37	45	53	61	69	77	85	93	101	109
0	1	1		22	30	38	46	54	62	70	78	86	94	102	110
1	1	1		23	31	39	47	55	63	71	79	87	95	103	111

More information about adding your own multiplexers can be found in Section 2.6.1 of the U6 Datasheet.

Differential Tables

The tables below show which channels are positive (Pos) or negative (Neg) when paired differentially. Additional information about using differential analog inputs with extended channels can be found in the Mux80 datasheet.

Built-in AIN (not multiplexed):

Pos	Neg
0	1
2	3
4	5
6	7
8	9
10	11
12	13

AIN0 and AIN1 (multiplexed with an external multiplexer):

Pos	Neg
16	24
17	25
18	26
19	27
20	28
21	29
22	30
23	31

AIN2 and AIN3 (multiplexed with an external multiplexer):

Pos	Neg
32	40
33	41
34	42
35	43

Pos	Neg
36	44
37	45
38	46
39	47

AIN4 and AIN5 multiplexed with the Mux80 (or an external multiplexer):

Pos	Neg
48	56
49	57
50	58
51	59
52	60
53	61
54	62
55	63

AIN6 and AIN7 multiplexed with the Mux80 (or an external multiplexer):

Pos	Neg
64	72
65	73
66	74
67	75
68	76
69	77
70	78
71	79

AIN8 and AIN9 multiplexed with the Mux80 (or an external multiplexer):

Pos	Neg
80	88

Pos	Neg
81	89
82	90
83	91
84	92
85	93
86	94
87	95

AIN10 and AIN11 multiplexed with the Mux80 (or an external multiplexer):

Pos	Neg
96	104
97	105
98	106
99	107
100	108
101	109
102	110
103	111

AIN12 and AIN13 multiplexed with the Mux80 (or an external multiplexer):

Pos	Neg
112	120
113	121
114	122
115	123
116	124
117	125
118	126
119	127

15.0 DAC [T-Series Datasheet]

DAC Screw Terminals

Source Impedance: **50 ohms**

Max Output Current: **20mA***

*See Appendix A-4 for details on voltage drop related to current draw.

Device	Resolution (bits)	Resolution (Volts)	Source Impedance (Ω)	Approx. Vout Min	Approx. Vout Max
T7	12	~1.2 mV	50	0.0	5
T4	10	~5 mV	50	0.0	5.0
T8	16	~160 µV	50	-0.1	10.0

Overview

T-Series devices have two DACs (digital-to-analog converters, also known as analog outputs). Each DAC will set its output voltage according to the provided value. The output voltage is limited by the min and max range. The step size or granularity is limited by the resolution. For example: The T7's DACs have a range of 0-5V and resolution of 12-bits, the step size is then (5-0) / 212 ≈1.2 mV with a max of ~5 V and a minimum of ~0 V.

For electrical specifications, See Appendix A-4.

The T7 DACs appear both on the screw terminals and on the DB37 connector. These connections are electrically the same, and the user must exercise caution only to use one connection or the other, and not create a short circuit.

Device Control Basics

- All T-series device features are controlled by reading and writing Modbus TCP registers via Modbus TCP (either directly or through our LJM library).
- We have register descriptions throughout documentation detailing relevant register

names, starting addresses, types, and access permissions (read/write).
- See Section 3.0 Communication for other detailed communication information.

Register Listing

To set DAC output voltage, write to the following registers:

DAC Registers

Name	Start Address	Type	Access
DAC#(0:1)	1000	FLOAT32	R/W

DAC#(0:1)
- Starting Address: 1000

Pass a voltage for the specified analog output.

- Data type: FLOAT32 (type index = 3)
- Readable and writable

Expanded Names	Addresses
DAC0, DAC1	1000, 1002

DAC#(0:1)_BINARY	51000	UINT32	W

DAC#(0:1)_BINARY
- Starting Address: 51000

Writes binary values to the DACs. Binary values are 16-bit. If the DAC's resolution is less than 16 then the lower bits are ignored. 0 = lowest output, 65535 = highest output.

- Data type: UINT32 (type index = 1)
- Write-only

Expanded Names	Addresses
DAC0_BINARY, DAC1_BINARY	51000, 51002

Output Range

T4/T7:

The DAC output amplifiers are designed so that the nominal maximum output is 5.0 volts. The

actual maximum voltage can vary and will always be limited according to the supply voltage. To determine the actual max of each DAC, write an impossibly high value such as 6.0 to the DAC, and then read back the value. This is the expected full-scale output with no load and with a supply voltage greater than the returned value. If Vs is below the returned value, the maximum output will be reduced accordingly. The step size of the DACs will not be affected, instead the output voltage will stop increasing after the expected output reaches VS.

The output range is also limited by the ability of the output amps to drive near the power rails (0 and VS). They can drive quite close, especially at light load, but will never be able to drive all the way to exactly 0.0 or VS.

The output range is further limited under load by the drive ability of the output amp and the 50 ohms of source resistance. The latter is dominant at lower currents, so for example if you set a DAC to 4.000 volts and draw 2 mA from it, the output will be closer to 3.900 volts.

See Appendix A-4 for more information.

T8:

The DACs on the T8 use additional power systems to output voltage levels above and below the supply. This allows the T8 to output approximately -0.1 to 10 V as long as the supply voltage is within the valid operating range.

The DACs on the T8 have good load regulation. Very little voltage drop is expected up to 20mA, usually 1-2 mV.

See Appendix A-4 for more information.

To output voltages from -10 V to 10 V, the LJTick-DAC is a great solution.

Power-up Defaults

The power-up condition of the DACs can be configured by the user. From the factory, the DACs default to be enabled at minimum voltage (~0 V). Note that even if the power-up default for a line is changed to a different voltage or disabled, there is a delay of about 100 ms at power-up where the DACs are in the factory default condition.

Reading

A read of DACx returns the last value that was written to the DAC chip.

Protection

The analog outputs can withstand a continuous short-circuit to ground, even when set at maximum output.

Voltage should never be applied to the analog outputs, as they are voltage sources themselves. In the event that a voltage is accidentally applied to either analog output, they do have protection against transient events such as ESD (electrostatic discharge) and continuous overvoltage (or undervoltage) of a few volts.

10 Hz Square Wave Output

DAC1 can be configured to output a 10 Hz, 3.3 V square wave. Writing a 1 to DAC1_FREQUENCY_OUT_ENABLE will enable the 10 Hz output:

Name

Name	Start Address	Type	Access
DAC1_FREQUENCY_OUT_ENABLE	61532	UINT32	W

DAC1_FREQUENCY_OUT_ENABLE
- Address: 61532

0 = off, 1 = output 10 Hz signal on DAC1. The signal will be a square wave with peaks of 0 and 3.3V. Note that writing to DAC1 or enabling a stream out which is targeting DAC1 will disable this feature.

- Data type: UINT32 (type index = 1)
- Write-only
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0234

The square wave output will be disabled:

- when 0 is written to DAC1_FREQUENCY_OUT_ENABLE,
- when any value is written DAC1,
- or when a stream-out channel targeting DAC1 is enabled.

T7 requires firmware version 1.0234 or later.

Increasing Output to ±10V

There is an accessory available from LabJack called the LJTick-DAC that provides a pair of 14-bit analog outputs with a range of ±10 volts. The LJTick-DAC plugs into any digital I/O block and thus many can be added to a T-series device.

Testing

A good way to test the DACs is to see if they output the expected voltage when loaded. The source impedance of the DACs is about 50 ohms, and that source impedance interacts with load impedance to form a voltage divider. Remove all user connections and connect a 470 ohm resistor from DAC# to GND. A voltage divider is formed such that:

```
Vout = Vset * 470 / (470 + 50) = Vset *
0.9
```

So whatever the DAC is set to, the output at the terminal should be about 10% less. You will see the 10% drop with test voltages up to about 4 volts. Above 4 volts, other effects come into play since this test is drawing substantial current from the DAC line.

16.0 DB37 (T7 Only) [T-Series Datasheet]

Number of Pins: **37**

Screw type: **#4-40**

Contacts: **Gold-coated**

Form factor: **D-Sub**

This high-density connector provides access to the T7 features that are not available on the screw terminal edge of the unit. It brings out analog inputs (AIN), analog outputs (DAC), digital I/O (FIO, MIO), and other signals.

The CB37 is a connector board that provides convenient screw-terminals for the DB37 lines, but the CB37 is not required to access I/O on the DB37. Any method you see fit can be used to access the DB37 lines.

Pinout

Some signals appear on both the DB37 connector and screw terminals, so care must be taken to avoid contention. For such signals, only connect to one location, not both. Signals duplicated on the T7 screw terminals and the DB37 are denoted in **bold**:

Table 16-1. DB37 Connector Pinouts

1	**GND**		20	**10 µA**
2	**200 µA**		21	FIO7
3	FIO6		22	FIO5
4	FIO4		23	**FIO3**
5	**FIO2**		24	**FIO1**
6	**FIO0**		25	MIO0

7	MIO1		26	MIO2
8	**GND**		27	**VS**
9	VM-		28	VM+
10	**GND**		29	**DAC1**
11	**DAC0**		30	**GND**
12	AIN13		31	AIN12
13	AIN11		32	AIN10
14	AIN9		33	AIN8
15	AIN7		34	AIN6
16	AIN5		35	AIN4
17	**AIN3**		36	**AIN2**
18	**AIN1**		37	**AIN0**
19	**GND**			

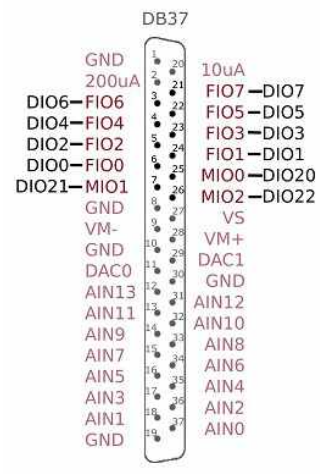


Figure 16-2. Standard DB37 pin numbers looking into the female connector on the T7

Remarks

VS, GND, FIO/MIO, AIN, DAC, 200UA/10UA

Descriptions of these can be found in their related sections of this datasheet.

VM+/VM-

Vm+/Vm- are bipolar power supplies intended to power external multiplexer ICs such as the DG408 from Intersil. The multiplexers can only pass signals within their power supply range, so Vm+/Vm- can be used to pass bipolar signals. Nominal voltage is ±13 volts at no load and ±12 volts at 2.5 mA. Both lines have a 100 ohm source impedance, and are designed to provide 2.5 mA or less. This is the same voltage supply used internally by the T7 to bias the analog input amplifier and multiplexers. If this supply is loaded more than 2.5 mA, the voltage can droop to the point that the maximum analog input range is reduced. If this supply is severely overloaded (e.g. short circuited), then damage could eventually occur. If Vm+/Vm- are used to power multiplexers, series diodes are recommended as shown in Figure 9 of the Intersil DG408 datasheet. Not so much to protect the mux chips, but to prevent current from going back into Vm+/Vm-. Use Schottky diodes to minimize voltage drop.

Duplicated Input Terminals (AIN0-AIN3 and FIO0-FIO3)

AIN0-AIN3 and FIO0-FIO3 appear on the built-in screw-terminals and also on the DB37 connector. You should only connect to one or the other, not both at the same time.

To prevent damage due to accidental short circuit, both connection paths have their own series resistor.

All FIO lines have a 470 ohm series resistor (that is included in the 550 ohm total impedance), and in the case of FIO0-FIO3 the duplicated connections each have their own series resistor, so if you measure the resistance between the duplicate terminals you will see about 940 ohms.

All AIN lines have a 2.2k series resistor, and in the case of AIN0-AIN3 the duplicated connections each have their own series resistor, so if you measure the resistance between the duplicate terminals you will see about 4.4k.

CB37 Terminal Board

The CB37 terminal board from LabJack connects to the DB37 connector and provides convenient screw terminal access to all lines. The CB37 is designed to connect directly to the DB37, but can also connect via a 37-line 1:1 male-female cable.

When using the analog connections on the CB37, the effect of ground currents should be considered, particularly when a cable is used and substantial current is sourced/sunk through the CB37 terminals. When any sizable cable lengths are involved, a good practice is to separate current carrying ground from ADC reference ground. An easy way to do this on the CB37 is to use GND as the current source/sink, and use AGND as the reference ground. This works well for passive sensors (no power supply), such as a thermocouple, where the only ground current is the return of the input bias current of the analog input.

EB37 Experiment Board

The EB37 experiment board connects to the DB37 connector and provides convenient screw terminal access. Also provided is a solderless breadboard and useful power supplies. The EB37 is designed to connect directly to the DB37, but can also connect via a 37-line 1:1 male-female cable.

OEM

The OEM T7 has a separate header location to bring out the same connections as the DB37 connector. This OEM header location is labeled J3. The J3 holes are always present, but are

obstructed when the DB37 connector is installed. Find the pinout, and other OEM information for J3 in OEM Versions.

17.0 DB15 [T-Series Datasheet]

Number of Pins: **15**

Screw type: **#4-40**

Contacts: **Gold-coated**

Form factor: **D-Sub**

The DB15 connector has the potential to be used as an expansion bus, where the 8 EIO are data lines and the 4 CIO are control lines.

The CB15 is a connector board that provides convenient screw-terminals for the DB15 lines, but the CB15 is not required to access I/O on the DB15. Any method you see fit can be used to access the DB15 lines.

These 12 channels include an internal series resistor that provides overvoltage/short-circuit protection. For details, see the "Protection" section of 13.0 Digital I/O.

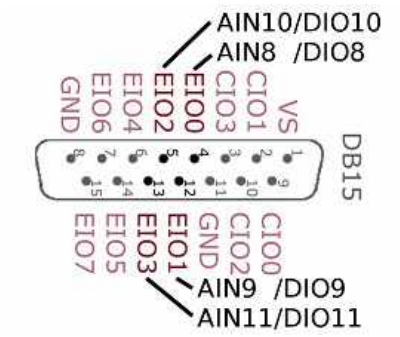
All digital I/O on T-series devices have 3 possible states: input, output-high, or output-low. For details, see the "Electrical Overview" section of 13.0 Digital I/O.

Pinout By Device

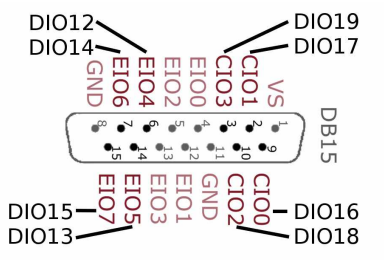
T4

The DB15 connector brings out 4 flexible I/O as well as 8 dedicated digital I/O.

The flexible I/O ports EIO0-EIO3 can be configured to be the analog inputs AIN8-AIN11 or the digital I/O ports DIO8-DIO11:

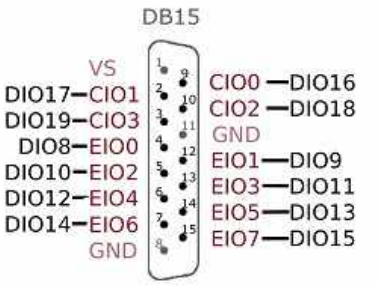


The dedicated digital I/O ports EIO4-CIO3 are DIO12-DIO19:



T7/T8

The DB15 connector brings out 12 additional digital I/O, EIO0-CIO3, which can also be addressed as DIO8-DIO19:



Remarks

CB15

The CB15 terminal board connects to the DB15 connector. It provides convenient screw terminal access to the 12 I/O channels available on the DB15 connector. The CB15 is designed to connect directly to the DB15, or can connect via a standard 15-line 1:1 male-female DB15 cable.

RB12

The RB12 relay board provides a convenient interface for T-series devices to industry standard digital I/O modules, allowing electricians, engineers, and other qualified individuals to interface a LabJack with high voltages/currents. The RB12 relay board connects to the DB15 connector on the LabJack, using the 12 EIO/CIO lines to control up to 12 I/O modules. Output or input types of digital I/O modules can be used. The RB12 is designed to accept G4 series digital I/O modules from Opto22, and compatible modules from other manufacturers such as the G5 series from Grayhill. Output modules are available with voltage ratings up to 200 VDC or 280 VAC, and current ratings up to 3.5 amps.

OEM

OEM T-series devices have a separate header location to bring out the same connections as the DB15 connector. This OEM header location is labeled J2. The J2 holes are always present, but are obstructed when the DB15 connector is installed. Find the pinout, and other OEM information for J2 in OEM Versions.

18.0 Internal Temp Sensor [T-Series Datasheet]

Overview By Device

T4

Sensor Range: **-50°C to 150°C**

T4 Operating Range: **-40°C to 85°C**

Accuracy (20°C to 40°C): **±1.5°C***

Accuracy (-20°C to 50°C): **±2.0°C***

Accuracy (-45°C to 85°C): **±3.5°C***

The T4 has an LM94021 temperature sensor (with GS=10) connected to an internal analog input. The sensor is physically located on the top of the PCB behind the VS screw terminal of the FIO4 and FIO5 screw terminal block.

T7

Sensor Range: **-50°C to 150°C**

T7 Operating Range: **-40°C to 85°C**

Accuracy (20°C to 40°C): **±1.5°C***

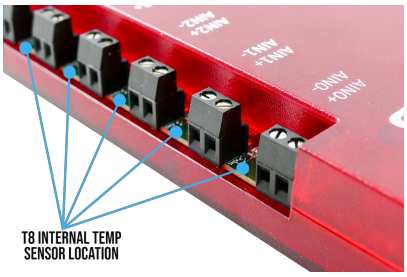
Accuracy (-20°C to 50°C): **±2.0°C***

Accuracy (-45°C to 85°C): **±3.5°C***

The T7 has an LM94021 temperature sensor (with GS=10) connected to internal analog input channel 14 (AIN14). The sensor is physically located on the bottom of the PCB between the AIN0/1 and AIN2/3 screw-terminals.

*Accuracy of measuring device temperature (TEMPERATURE_DEVICE_K). Includes error from LM94021 specifications and error due to linear equation fit.

T8



T8 Internal Temperature Sensor Locations

Sensor Range: **-50°C to 150°C**

T7 Operating Range: **-40°C to 85°C**

Accuracy (20°C to 40°C): **±1.5°C***

Accuracy (-20°C to 50°C): **±2.0°C***

Accuracy (-45°C to 85°C): **±3.5°C***

The T8 has LM94021 temperature sensors (with GS=01) near the screw terminals of each analog input. Multiple sensors are necessary, because the design of the isolation circuity reduces the rate that thermal energy will transfer between the AINs.

The temperature sensors are measured simultaneously along with the analog inputs. The results are stored in capture registers for later reading.

> Due to a hardware limitation, the maximum sampling rate of the T8 temperature sensors is 250 Hz. As of firmware 1.0004, the temperature sensors will return readings of -9999 if the sampling rate is ahigher than 250 Hz.
> The register AIN_SAMPLING_RATE_HZ controls the sampling rate of all T8 temperature sensors and analog inputs. See section 14.0 Analog Inputs for additional information about AIN_SAMPLING_RATE_HZ.

T8 Temperature Registers

Name	Start Address	Type	Access
TEMPERATURE#(0:7)	600	FLOAT32	R

TEMPERATURE#(0:7)
- Starting Address: 600

T8 Only. Returns the temperature of the specified analog input. And saves AIN and temperature inputs for all channels.

- Data type: FLOAT32 (type index = 3)
- Read-only
- This register may be streamed

Expanded Names	Addresses
TEMPERATURE0, TEMPERATURE1, TEMPERATURE2, TEMPERATURE3, TEMPERATURE4, TEMPERATURE5, TEMPERATURE6, TEMPERATURE7	600, 602, 604, 606, 608, 610, 612, 614

TEMPERATURE#(0:7)_CAPTURE	700	FLOAT32	R

TEMPERATURE#(0:7)_CAPTURE
- Starting Address: 700

T8 Only. Returns the saved temperature of the specified analog input.

- Data type: FLOAT32 (type index = 3)
- Read-only
- This register may be streamed

Expanded Names	Addresses
TEMPERATURE0_CAPTURE, TEMPERATURE1_CAPTURE, TEMPERATURE2_CAPTURE, TEMPERATURE3_CAPTURE, TEMPERATURE4_CAPTURE, TEMPERATURE5_CAPTURE, TEMPERATURE6_CAPTURE, TEMPERATURE7_CAPTURE	700, 702, 704, 706, 708, 710, 712, 714

The normal registers, TEMPERATURE_DEVICE_K and TEMPERATURE_AIR_K, will report the value from TEMPERATURE0, the AIN0 temperature sensor.

Device Temperature

Read TEMPERATURE_DEVICE_K to get the device temperature:

Name	Start Address	Type	Access
TEMPERATURE_DEVICE_K	60052	FLOAT32	R

TEMPERATURE_DEVICE_K
- Address: 60052

Takes a reading from the internal temperature sensor using range=+/-10V and resolution=8, and applies the formula Volts*-92.6+467.6 to return kelvins.

- Data type: FLOAT32 (type index = 3)
- Read-only
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - The internal temperature sensor, AIN14, is internally connected to an LM94021 (U24) with GS=10 which is physically located on the bottom of the PCB between the AIN0/1 and AIN2/3 screw-terminals.
  - Minimum firmware version: 1.0105
- T4:
  - The internal temperature sensor is internally connected to an LM94021 (U24) with GS=10 which is physically located on the top of the PCB behind the VS screw terminal of the FIO4 and FIO5 screw terminal block.
  - Minimum firmware version: 0.2020

T7 only:

AIN14 Temperature Sensor Voltage

Alternatively, reading from AIN14 returns the temperature sensor voltage, which can be converted to device temperature using the formula:

`Device temperature K = volts * -92.6 + 467.6`

Stream Mode

You can read the raw temp sensor voltage from AIN14 in stream mode, but cannot read TEMPERATURE_DEVICE_K or TEMPERATURE_AIR_K since the internal math that is required for them is too slow. If streaming, use AIN14 to get volts and use the above formula to get device temperature in K. For an estimate of air temperature, see the following "Air Temperature" section.

Air Temperature

TEMPERATURE_AIR_K is an estimate of the ambient air temperature outside the device:

Name	Start Address	Type	Access
TEMPERATURE_AIR_K	60050	FLOAT32	R

TEMPERATURE_AIR_K
- Address: 60050

Returns the estimated ambient air temperature just outside of the device in its red plastic enclosure. This register is equal to TEMPERATURE_DEVICE_K - 4.3. If Ethernet and/or WiFi is enabled, subtract an extra 0.6 for each.

- Data type: FLOAT32 (type index = 3)
- Read-only
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0105
- T4:
  - Minimum firmware version: 0.2020

It is calculated depending on whether Ethernet and/or WiFi is enabled as follows:

- USB `TEMPERATURE_AIR_K = TEMPERATURE_DEVICE_K - 4.3`
- USB & Ethernet `TEMPERATURE_AIR_K = TEMPERATURE_DEVICE_K - 4.9`
- USB & WiFi `TEMPERATURE_AIR_K = TEMPERATURE_DEVICE_K - 4.9`
- USB & Ethernet & WiFi `TEMPERATURE_AIR_K = TEMPERATURE_DEVICE_K - 5.5`

These offsets were determined from measurements with the enclosure on and in still air. We noted that the time constant was about 12 minutes, meaning that 12 minutes after a step change you are 63% of the way to the new value.

Timing

T7 or T7-Pro: TEMPERATURE_AIR_K and TEMPERATURE_DEVICE_K always use Range = 10 and ResolutionIndex=8. Which means they take about 1.09 ms per reading from Table A.3.1.1.

T4: TEMPERATURE_AIR_K and TEMPERATURE_DEVICE_K always use ResolutionIndex=5. Which means they take about 0.8 ms per reading.

T8: Measurement time will vary base on the current settings of the analog system and which register the reading is being requested from. TEMPERATURE#(0:8) will run a new measurement at the current rate and resolution settings of the analog system. Reading from TEMPERATURE#(0:8)_CAPTURE will return the most recently captured value. Reading the captured value is very fast, on the order of a couple microseconds. The communication overhead and any other registers being read will dominate the command-response time.

Note on thermocouples

The value from TEMPERATURE_DEVICE_K best reflects the temperature of the built-in screw-terminals AIN0-AIN3, so use that for cold junction compensation (CJC) if thermocouples are connected there.

The internal sensor has a specified accuracy of ±2.0 °C across the range of -20 to +50 °C. Allowing for a slight difference between the sensor temperature and the temperature of the screw-terminals, expect the returned value minus 3 °C to reflect the temperature of the built-in screw-terminals with an accuracy of ±2.5 °C.

If thermocouples are connected to the CB37, you want to know the temperature of the screw-terminals on the CB37. The CB37 is typically at the same temperature as ambient air, so use the value from register TEMPERATURE_AIR_K for CJC. Better yet, add a sensor such as the LM34CAZ to an unused analog input on the CB37 to measure the actual temperature of the CB37.

If thermocouples are connected to an LJTick-Amp (typical with the T4), you want to know the temperature of the screw-terminals on the LJTIA, which we would expect to be close to ambient, so use the value from register TEMPERATURE_AIR_K for CJC.

19.0 RTC (T7-Pro Only) [T-Series Datasheet]

Overview - T7-Pro Only

The T7-Pro has a battery-backed RTC (real-time clock) which is useful for assigning timestamps to data that is stored on the microSD card during scripting operations—particularly in situations where the device could experience power failure or other reboots, and does not have a communication connection that can be used to determine real-time after reboot. The system time is stored in seconds since 1970, also known as the Epoch and Unix timestamp.

Note: To use the RTC_TIME_S register please update to a firmware version of 1.0297 or higher on the T7. The start of 2021 exposed a new bug in system time functions on the T7, and this subsequently broke the RTC_TIME_S register functionality on firmware versions 1.0292 and earlier.

Battery Life

Typically, the CR2032 battery can be expected to last 12 years. Several factors can influence the expected life including:

- Temperatures - Cold temperatures will decrease the battery's life.
- Battery age - Batteries naturally discharge as they age. This also has a temperature dependency, but 20% every 10 years is a good rule of thumb.
- External Power - When the T7 is powered up the battery is not used to maintain time. This will extend the battery life.

The RTC requires 925 nA to maintain it's time. A typical CR2032 battery has 210 mAH capacity which works out to a life of 25.9 years. We derate our expectations by ~2 to get the above life expectancy.

Reading Time

Read the system time in seconds with address 61500:

Name	Start Address	Type	Access
RTC_TIME_S	61500	UINT32	R

RTC_TIME_S
- Address: 61500

Read the current time in seconds since Jan, 1970, aka Epoch or Unix time. This value is calculated from the 80 MHz crystal, not the RTC 32 kHz crystal. Non-pro devices do not have a real time clock, so the reported time is either seconds since device startup or the time reported by the network time protocol over Ethernet. Pro devices have a real time clock that will be used to initialize this register at startup, the time can then be updated by NTP.

- Data type: UINT32 (type index = 1)
- Read-only
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0128
- T4:
  - Minimum firmware version: 0.2020

SYSTEM_COUNTER_10KHZ	61502	UINT32	R

SYSTEM_COUNTER_10KHZ
- Address: 61502

A 10 kHz counter synchronized to RTC_TIME_S. This register can be appended to RTC_TIME_S as the decimal portion to get 0.1 ms resolution.

- Data type: UINT32 (type index = 1)
- Read-only
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0229

Get a simple calendar time representation by reading six consecutive addresses, starting with address 61510:

Name	Start Address	Type	Access
RTC_TIME_CALENDAR	61510	UINT16	R

RTC_TIME_CALENDAR
- Address: 61510

Read six consecutive addresses of type UINT16, starting with this address. The result will be in year, month, day, hour, minute, second calendar format. i.e. [2014, 10, 21, 18, 55, 35]

- Data type: UINT16 (type index = 0)
- Read-only
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0128
- T4:
  - Minimum firmware version: 0.2020

To time events faster than 1 second apart, it is possible to read the CORE_TIMER (address 61520) and see how it changes from second to second. To access the core timer value in Lua scripts, use the LJ.Tick() function.

Setting Time

Set the system time by writing a new timestamp (in seconds) to address 61504:

Name	Start Address	Type	Access
RTC_SET_TIME_S	61504	UINT32	W

RTC_SET_TIME_S
- Address: 61504

Write a new timestamp to the RTC in seconds since Jan, 1970, aka Epoch or Unix timestamp.

- Data type: UINT32 (type index = 1)
- Write-only
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0128
- T4:
  - Minimum firmware version: 0.2020

To set the T7-Pro's time from a SNTP server, use RTC_SET_TIME_SNTP and SNTP_UPDATE_INTERVAL:

Name	Start Address	Type	Access
RTC_SET_TIME_SNTP	61506	UINT32	W

RTC_SET_TIME_SNTP
- Address: 61506

Write any value to instruct the T7 to update its clock from a SNTP server. Requires that SNTP_UPDATE_INTERVAL is non-zero.

- Data type: UINT32 (type index = 1)
- Write-only
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0128
- T4:
  - Minimum firmware version: 0.2020

SNTP_UPDATE_INTERVAL	49702	UINT32	R/W

SNTP_UPDATE_INTERVAL
- Address: 49702

Sets the SNTP retry time in seconds. A value of zero will disable SNTP(0=default). Values must be 10 or greater.

- Data type: UINT32 (type index = 1)
- Readable and writable
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0150
- T4:
  - Minimum firmware version: 0.2020

Examples

Read the value of the RTC in a Lua script lua

```
table = {}
table[1] = 0    --year
table[2] = 0    --month
table[3] = 0    --day
table[4] = 0    --hour
table[5] = 0    --minute
table[6] = 0    --second
```

```
table, error = MB.RA(61510, 0, 6)
print(string.format("%04d/%02d/%02d %02d:%02d.%02d", table[1], table[2], table[3], table[4], table[5], table[6]))

>> 2014/10/15 18:55.22
```

Read the value of the RTC in C/C++

C++

```
 int LJMError;
double newValue;
LJMError = LJM_eReadAddress(handle, 61500, 1, &newValue);
printf(newValue);
//returned value of 1413398998 would correspond with Wed, 15 Oct 2014 18:49:58 GMT
```

20.0 Internal Flash [T-Series Datasheet]

Overview

T-series devices have non-volatile flash memory. This guide will often refer the flash memory as "internal flash".

Internal flash memory is divided into two regions:

- The User Area - For storing user-data.
- The Reserved Area - Used to store important device information such as calibration constants.

The size of the internal flash varies by device:

Device	Internal Flash Size	User Area Size	Reserved Area Size
T4	4 MB	2 MB	2 MB
T7	4 MB	2 MB	2 MB
T8	8 MB	4 MB	4 MB

Subsections

Flash Addresses

Internal flash is addressed by byte, but accessed by 32-bit values. That means that the first value is at address zero and the second is at address 4. (These are flash addresses—not to be confused with Modbus addresses.)

Following are some important flash addresses:

T4/T7

Address Name	Address	Length	Key
User Area	0x000000	2 MB	0x6615E336 (1712710454 in decimal)

Address Name	Address	Length	Key
Reserved Area	0x200000	2 MB	N/A
Calibration Constants (in Reserved Area)	0x3C4000	4 kB	0x43A24C42 (1134709826 in decimal)

T8

Address Name	Address	Length	Key
User Area	0x000000	4 MB	0x6615E336 (1712710454 in decimal)
Reserved Area	0x600000	4 MB	N/A
Calibration Constants (in Reserved Area)	0x687000	4 kB	0xA7863777 (2810591095 in decimal)

Reading

To read from flash, write the desired address to INTERNAL_FLASH_READ_POINTER and then read an even number of registers from INTERNAL_FLASH_READ:

Name	Start Address	Type	Access
INTERNAL_FLASH_READ_POINTER	61810	UINT32	R/W

INTERNAL_FLASH_READ_POINTER
- Address: 61810

The address in internal flash that reads will start from.

- Data type: UINT32 (type index = 1)
- Readable and writable

INTERNAL_FLASH_READ	61812	UINT32	R

INTERNAL_FLASH_READ
- Address: 61812

Data read from internal flash. Read size must be an even number of registers.

- Data type: UINT32 (type index = 1)
- Read-only

To read a large number of registers from INTERNAL_FLASH_READ (such as more than ~25 registers), you must split the read into multiple packets while updating INTERNAL_FLASH_READ_POINTER each time. See the LJM explanation for how to perform large reads.

Writing and Erasing

Key

For a region to be written to or to be erased, the key for that region must be written to the INTERNAL_FLASH_KEY register:

Name	Start Address	Type	Access
INTERNAL_FLASH_KEY	61800	UINT32	R/W

INTERNAL_FLASH_KEY
- Address: 61800

Sets the region of internal flash to which access is allowed.

- Data type: UINT32 (type index = 1)
- Readable and writable

The key prevents accidental overwrites. The value in INTERNAL_FLASH_KEY will be cleared when the Modbus packet that wrote it has been fully processed, so INTERNAL_FLASH_KEY must be written in the same packet that does INTERNAL_FLASH_WRITE or INTERNAL_FLASH_ERASE. The LJM Multiple Value functions simplify this. The LJM Multiple Value functions do not correctly split large reads (such as more than ~15 registers), so see the LJM explanation for how to perform large writes.

Writing

To write to flash, the area to be written to must first be erased. Once erased, write the key for the desired region to INTERNAL_FLASH_KEY, write the desired address to INTERNAL_FLASH_WRITE_POINTER, and then write an even number of registers to INTERNAL_FLASH_WRITE:

Name	Start Address	Type	Access
INTERNAL_FLASH_WRITE_POINTER	61830	UINT32	R/W

INTERNAL_FLASH_WRITE_POINTER
- Address: 61830

Address in internal flash where writes will begin.

- Data type: UINT32 (type index = 1)
- Readable and writable

INTERNAL_FLASH_WRITE	61832	UINT32	W

INTERNAL_FLASH_WRITE
- Address: 61832

Data written here will be written to internal flash. Write size must be an even number of registers. This register is a buffer.

- Data type: UINT32 (type index = 1)
- Write-only
- This register is a Buffer Register

Erasing

Flash is erased 4 kB at a time. Erasing sets all bits to 1. To erase a 4 kB region, write the key for the desired region to INTERNAL_FLASH_KEY and the desired address to INTERNAL_FLASH_ERASE:

Name	Start Address	Type	Access
INTERNAL_FLASH_ERASE	61820	UINT32	W

INTERNAL_FLASH_ERASE
- Address: 61820

Erases a 4k section of internal flash starting at the specified address. This register is a buffer.

- Data type: UINT32 (type index = 1)
- Write-only
- This register is a Buffer Register

The address will be rounded down to the nearest 4 kB boundary. Boundaries are easy to identify when the address is displayed in hexadecimal because the lower three digits will be zero. 4 kB is hexadecimal is 0x1000.

Endurance

Flash memory can only be erased so many times before bit errors will start to occur; it is important not to erase or write flash needlessly. Typical life of flash memory is at least 10,000 cycles.

Examples

Internal Flash Examples c#

```
 // Erase address 0
numFrames = 2
aNames[0] = "INTERNAL_FLASH_KEY"
aNames[1] = "INTERNAL_FLASH_ERASE"
aWrites[0] = LJM.CONSTANTS.WRITE
aWrites[1] = LJM.CONSTANTS.WRITE
aNumValues[0] = 1
aNumValues[1] = 1
aValues[0] = 1712710454
aValues[1] = 0
LJM.eNames(handle, numFrames, aNames, aWrites, aNumValues, aValues, errorAddress)


// Write address 0 = 1234,
// write address 4 = 5678
numFrames = 3
aNames[0] = "INTERNAL_FLASH_KEY"
aNames[1] = "INTERNAL_FLASH_WRITE_POINTER"
aNames[2] = "INTERNAL_FLASH_WRITE"
aWrites[0] = LJM.CONSTANTS.WRITE
aWrites[1] = LJM.CONSTANTS.WRITE
aWrites[2] = LJM.CONSTANTS.WRITE
aNumValues[0] = 1
aNumValues[1] = 1
aNumValues[2] = 2
aValues[0] = 1712710454
aValues[1] = 0
aValues[2] = 1234
aValues[3] = 5678
LJM.eNames(handle, numFrames, aNames, aWrites, aNumValues, aValues, errorAddress)


// Read address 0 and address 4
numFrames = 2
aNames[0] = "INTERNAL_FLASH_READ_POINTER"
aNames[1] = "INTERNAL_FLASH_READ"
aWrites[0] = LJM.CONSTANTS.WRITE
aWrites[1] = LJM.CONSTANTS.READ
aNumValues[0] = 1
aNumValues[1] = 2
aValues[0] = 0
aValues[1] = 9999
aValues[2] = 9999
LJM.eNames(handle, numFrames, aNames, aWrites, aNumValues, aValues, errorAddress)


Console.WriteLine(aValues[1], aValues[2])
// Output:
// 1234 5678
```

For an embedded Lua scripting example see write_read_flash.lua.

20.0.0 T4 Calibration Constants [T-Series Datasheet]

The T4 automatically returns calibrated readings, so most people need not concern themselves with this section.

The factory applied calibration constants are stored in Internal Flash and can be accessed at any time through the use of the Modbus registers discussed in the parent to this section (Internal Flash section).

The calibration constants begin at memory address 0x3C4000, or in decimal format d3948544. The structure (location) of each calibration value can be seen in the C code snippet below:

T4 Calibration Constants Structure C

```
typedef struct{
  struct {
     float Slope;
     float Offset;
  } HV[4];
  struct {
     float Slope;
     float Offset;
  } LV;
  struct {
     float Slope;
     float Offset;
  } SpecV;
  struct {
     float Slope;
     float Offset;
  } DAC[2];

  float Temp_Slope;
  float Temp_Offset;

  float I_Bias;
} DeviceCalibrationT4;
```

Nominal Calibration Values

	Slope	Offset
HV[0] (AIN0)	3.235316E-04	-10.532965
HV[1] (AIN1)	3.236028E-04	-10.534480
HV[2] (AIN2)	3.235439E-04	-10.530597
HV[3] (AIN3)	3.236133E-04	-10.530210
LV	3.826692E-05	0.002484
SpecV	-3.839420E-05	2.507430
DAC0	1.310768E+04	54.091066
DAC1	1.310767E+04	54.044314
Temp	-9.260000E+01	467.600000

AIN Bias Current: 0.000000015

20.0.1 T7 Calibration Constants [T-Series Datasheet]

The T7 automatically returns calibrated readings, so most people need not concern themselves with this section.

The factory applied calibration constants are stored in Internal Flash and can be accessed at any time through the use of the Modbus registers discussed in the parent to this section (Internal Flash section).

The calibration constants begin at memory address 0x3C4000, or in decimal format d3948544. The structure (location) of each calibration value can be seen in the C code snippet below.

T7 Calibration Constants Structure C

```
typedef struct{
float PSlope;
float NSlope;
float Center;
float Offset;
}Cal_Set;

typedef struct{
 Cal_Set HS[4];
 Cal_Set HR[4];

 struct{
  float Slope;
  float Offset;
 }DAC[2];

 float Temp_Slope;
 float Temp_Offset;

 float ISource_10u;
 float ISource_200u;

 float I_Bias;
}Device_Calibration;
```

The full size of the calibration section is 164 bytes, or 41 floats.

The reason that there are 'Cal_Set's for each High Speed 'HS' and High Resolution 'HR', is that there are 2 analog converters on a T7-Pro. A standard T7 uses only the High Speed analog converter, so only the HS[4] calibration values will be populated with valid information. A T7-Pro will have calibration information for both high speed, and high resolution converters.

Additionally, there are distinct sets of positive slope (PSlope), negative slope (NSlope), Center, and Offset values for each of the 4 gain settings on the device.

High speed AIN calibration values **HS[4]**:
HS[0] = calibration for gain x1
HS[1] = calibration for gain x10
HS[2] = calibration for gain x100
HS[3] = calibration for gain x1000

High resolution (-Pro only) AIN calibration values **HR[4]**:
HR[0] = calibration for gain x1
HR[1] = calibration for gain x10

HR[2] = calibration for gain x100
HR[3] = calibration for gain x1000

Nominal Calibration Values

±10V Range:

- Positive Slope: 0.000315805780
- Negative Slope: -0.000315805800
- Binary Center: 33523
- Voltage Offset: -10.586956522

±1V Range:

- Positive Slope: 0.000031580578
- Negative Slope: -0.000031580600
- Binary Center: 33523
- Voltage Offset: -1.0586956522

±0.1V Range:

- Positive Slope: 0.000003158058
- Negative Slope: -0.000003158100
- Binary Center: 33523
- Voltage Offset: -0.1058695652

±0.01V Range:

- Positive Slope: 0.000000315806
- Negative Slope: -0.000000315800
- Binary Center: 33523
- Voltage Offset: -0.010586956

DACs:

- Slope: 13200
- Offset: 0

Temperature:

- Slope: -92.6
- Offset: 467.6

Current Sources:

- 10 µA: 0.000010
- 200 µA: 0.000200

AIN Bias Current: 0.000000015

20.0.2 T8 Calibration Constants [T-Series Datasheet]

The T8 uses calibration constants to convert between the binary, used by hardware, and easier to understand units, typically volts.

Generally, users do not need to think about the calibration constants. The T8 automatically applies the calibration before reporting results. A couple exceptions are stream-mode and when the T8 has been instructed to return binary values. When using stream-mode, the calibration constants will be applied by the LJM driver.

The calibration constants are calculated as part of our new-unit test process. Those constants are then stored in the device's Internal Flash memory. The normal flash interface can be used to read the constants: (Internal Flash). Within flash, the calibration constants begin at memory address 0x687000, or in decimal format d6844416. The structure (location) of each calibration value can be seen in the C code snippet below.

T8 Calibration Values C

```
typedef struct{
float PSlope;
float NSlope;
float Center;
float Offset;
}Cal_Set;


typedef struct{
    uint32_t code;
    uint32_t reserved[7];
    uint32_t ain_type[8]; // 32 Contains type information for AINs.
    Cal_Set ain[8][11]; // 64 Calibration data for analog inputs.
    Cal_Set temp[8]; // 1472 Calibration data for temp sensors.
    Cal_Set VS; // 1600 Calibration data for VS measurement.
    Cal_Set IS; // 1616 Calibration data for IS measurement.
    Cal_Set dac[2]; // 1632 Calibration data for DACs
    float SecOSC_Freq; // 1664
}DeviceCalibrationT8;
```

The full size of the calibration section is 1668 bytes.

There are distinct sets of positive slope (PSlope), negative slope (NSlope), Center, and Offset values for each measurement.

For a given analog input x:

- ain[x][0] for range ±11.000
- ain[x][1] for range ±9.768
- ain[x][2] for range ±4.884
- ain[x][3] for range ±2.442
- ain[x][4] for range ±1.221
- ain[x][5] for range ±0.611
- ain[x][6] for range ±0.305
- ain[x][7] for range ±0.153
- ain[x][8] for range ±0.076
- ain[x][9] for range ±0.038
- ain[x][10] for range ±0.019

Nominal Calibration Values

Analog inputs

Each AIN has 11 Cal_Sets, the nominal values for which are below:

{PSlope, NSlope, Center, Offset},

{2.328872681E-006, -2.328872681E-006, 8388608, 19.536},

{1.164436340E-006, -1.164436340E-006, 8388608, -9.768},

{5.822181702E-007, -5.822181702E-007, 8388608, -4.884},

{2.911090851E-007, -2.911090851E-007, 8388608, -2.442},

{1.455545425E-007, -1.455545425E-007, 8388608, -1.221},

{7.277727127E-008, -7.277727127E-008, 8388608, -0.611},

{3.638863564E-008, -3.638863564E-008, 8388608, -0.305},

{1.819431782E-008, -1.819431782E-008, 8388608, -0.153},

{9.097158909E-009, -9.097158909E-009, 8388608, -0.076},

{4.548579454E-009, -4.548579454E-009, 8388608, -0.038},

{2.274289727E-009, -2.274289727E-009, 8388608, -0.019}

Temperature Sensors

{ -91.503268, 0, 0, 192.156863 };

Analog Outputs

{ 6243.64, 6243.64, 0, 800 };

Interval Converter

Vs { 0, 0, 0, 0 };

Is { 0, 0, 0, 0 };

21.0 SD Card (T7 Only) [T-Series Datasheet]

SD Card Location (Case)

SD Card Location (PCB)

The T7-Pro ships with a 2GB (or larger) microSD card installed (SLC technology). It might also be referred to as uSD, µSD, or just SD.

The T7 does not have the microSD card installed, but does have the card holder installed so a compatible microSD card can be installed in the field.

The retainer opens by sliding the metal piece forward, then lifting.

Currently microSDXC is not supported. Generally speaking, anything above 2GB is 'HC' meaning high capacity, and HC cards might need to be reformatted before they work.

The T7 supports FAT and FAT32 file systems, but some makes and sizes behave differently. We recommend the following SD card format:

```
File System: FAT
Allocation unit size: 64
kilobytes
```

FAT32 with an allocation unit size of 16 kB or 32 kB sometimes works, but smaller allocation sizes generally do not. On 2 GB cards it's possible to select FAT format with a 32 kB allocation size, and that sometimes works.

Care must be taken to ensure that power is not lost during file writing or disk corruption could occur. The rated operating temperature of the SD card is -25°C to 85°C. For extremely low temperatures, customers can buy industrial grade SD cards, such as the AF1GUDI-OEM, from

ATP Electronics, Inc.

File and directory names are limited to ASCII characters only.

The maximum number of file handles that can be open at once on a T7 is 4. One file handle is reserved for Modbus communication to allow programs to read files off of the SD card and the remaining 3 are available for Lua Scripts.

Standalone Data Logging

The microSD card is generally only useful for people doing standalone data logging through Lua scripting, since normal T7 operation is with a host connected (so the host can store data). Standalone logging is an advanced topic. We provide Lua examples for logging data to the microSD, but options for retrieving the data are somewhat limited (see next section). Standalone logging is generally limited to command-response data rates.

For information about powering the T7-Pro see the VS, Power Supply Section of this datasheet.

As an alternative to true standalone logging with the T7-Pro, consider that any LabJack combined with a host computer makes an incredibly powerful and flexible data logger. The host computer could be a full-blown desktop machine, but could be a simple SBC (single board computer) such as Raspberry Pi, BeagleBone, or various options from Technologic Systems.

Retrieving Data from the microSD Card

It is pretty easy to write data to the microSD card in a Lua script. At this time there are a few options for retrieving the data from the microSD:

1. Remove (or swap) the microSD from the T7 and put it in a card reader on a computer. On non-OEM versions of the T7/T7-Pro, the enclosure must be opened to access the microSD holder.

2. Windows only: Use the beta SD utility for downloading files from the SD card via USB/Ethernet/WiFi. Note that the microSD and WiFi share a serial bus inside the T7, so sometimes extra thought is required if using both at the same time.

3. Use the registers described below to read data off the microSD.

Testing the microSD Card

Not all SD cards are compatible. If you install a new or different SD card, we recommend the following tests to make sure it works. If you have access to a Windows computer, we recommend downloading and running the T7uSD testing application published on the T-Series Additional Utility Applications page.

A. Check Kipling to make sure that the SD card is properly recognized and that the calibration status of the device is still good.

1. Connect to the device with Kipling
2. On the Device Info tab and make sure there is a green checkmark next to the SD Card Installed hardware option.
3. Check to make sure the device's calibration status is still "Good".

B. Check to make sure the SD card can be read and written.

1. Connect to the device with Kipling.
2. Download and run the Lua example script titled "Log voltage to file".
3. Exit Kipling and open the SD Utility to verify that the file was written properly.

C. Check to make sure the T7's calibration constants can be read from the internal flash chip.

1. If you are on a Windows computer, use LJStreamM to try streaming an analog input register from a T7. If you get stream errors LJME_USING_DEFAULT_CALIBRATION (203) or LJME_INVALID_VALUE (1305), the microSD card being used is likely not compatible with the T7.

Errors Caused by Unsupported microSD Cards

1. A device may get stuck in its recovery firmware version and report upgrade issues because the flash chip is not responding properly.
2. A device may report that it is using default calibration values and it is unable to be used in stream mode. The device's calibration constants are read before streaming is started.
3. A device may have issues when running Lua scripts that try to save data to the microSD card or errors may be reported by the SD Card utility indicating that a file can't be opened.

Accessing the microSD Card While Using WiFi

WiFi shares an internal serial bus with the SD card and Internal Flash, and at the start of joining WiFi needs about 3 seconds of uninterrupted access on this serial bus. If a Lua script does file I/O operations during this time, then WiFi initialization will fail and the WiFi module will immediately try again. If Lua file I/O occurs every 3 seconds or less, it is likely that WiFi will never be able to join. A simple way to make sure WiFi can join, perhaps with a retry or two needed, is to only write every 5 seconds (or longer).

Use the following Lua pseudocode to write to a file once every 5 seconds, and read an analog input once every 500ms. See the "Log voltage to file" Lua examples in Kipling for an actual

script example.

```lua
 LJ.IntervalConfig(0, 500)  --set the DAQ interval to 500ms, should divide evenly into file access interval
LJ.IntervalConfig(1, 5000)  --set the file access interval to 5 seconds

TableSize = 5000/500
data = {}
DAQcount = 0

for i=1, TableSize do
  data[i] = 0
end

while true do
  if LJ.CheckInterval(0) then --if a data point needs to be collected
    data[DAQcount] = MB.R(0, 3)--collect a new reading from AIN0
    DAQcount = DAQcount + 1
  end
  if LJ.CheckInterval(1) then --file access interval complete
    appendToFile(data)          --save the data  to a file
    DAQcount = 0
  end
end
```

File I/O General Info

The T7 uses Unix-style file paths where the separator character is a slash: /

The use of absolute and relative paths is supported, for example to read a file named "test.txt" from the "tmp" folder which is saved in the root directory use the following path:

`/tmp/test.txt`

If the current working directory is already set to "/" then you could also use the path:

`tmp/test.txt`

Common File I/O Operations

See the Register Listing section below for register information.

Get the name of the current working directory (CWD):

1. Write a value of 1 to FILE_IO_DIR_CURRENT. The error returned indicates whether there is a directory loaded as current. No error (0) indicates a valid directory.
2. Read FILE_IO_PATH_READ_LEN_BYTES.
3. Read an array of size FILE_IO_PATH_READ_LEN_BYTES from FILE_IO_PATH_READ.
4. Resultant string will be something like "/" for the root directory, or "/DIR1/DIR2" for a directory.

Get list of items in the CWD:

1. Write a value of 1 to FILE_IO_DIR_FIRST. The error returned indicates whether anything was found. No error (0) indicates that something was found. FILE_IO_NOT_FOUND (2960) indicates that nothing was found.
2. Read FILE_IO_PATH_READ_LEN_BYTES, FILE_IO_ATTRIBUTES, and FILE_IO_SIZE_BYTES. Store the attributes and size associated with each file.
3. Read an array of size FILE_IO_PATH_READ_LEN_BYTES from FILE_IO_PATH_READ. This is the name of the file/folder.
4. Write a value of 1 to FILE_IO_DIR_NEXT. The error returned indicates whether anything was found. No error (0) indicates that there are more items->go back to step 2. Each of the following errors indicate that there are no more items:
5. FILE_IO_END_OF_CWD (2966)
6. FILE_IO_INVALID_OBJECT (2809)
7. FILE_IO_NOT_FOUND (2960)

Change the CWD:

1. Find from the list of items a directory to open, e.g. "/DIR1". Directories can be parsed out of the list of items by analyzing their FILE_IO_ATTRIBUTES bitmask. If bit 4 of the FILE_IO_ATTRIBUTES bitmask is set, then the item is a directory.
2. Write the directory name length in bytes to FILE_IO_PATH_WRITE_LEN_BYTES (ASCII, so each char is 1 byte, also don't forget to add 1 for the null terminator).
3. Write the directory string (converted to an array of bytes, with null terminator) to FILE_IO_PATH_WRITE. (array size = length from step 2)
4. Write a value of 1 to FILE_IO_DIR_CHANGE.
5. Done. Optionally get a list of items in the new CWD.

Get disk size and free space:

1. Read FILE_IO_DISK_SECTOR_SIZE_BYTES, FILE_IO_DISK_SECTORS_PER_CLUSTER, FILE_IO_DISK_TOTAL_CLUSTERS, FILE_IO_DISK_FREE_CLUSTERS. All disk parameters are captured when you read FILE_IO_DISK_SECTOR_SIZE_BYTES.
2. Total size = SECTOR_SIZE * SECTORS_PER_CLUSTER * TOTAL_CLUSTERS.
3. Free size = SECTOR_SIZE * SECTORS_PER_CLUSTER * FREE_CLUSTERS.

Get disk format:

1. Read FILE_IO_DISK_FORMAT_INDEX
2. 2=FAT, 3=FAT32

Read a file:

1. Write the length of the file name (including the null terminator) to FILE_IO_PATH_WRITE_LEN_BYTES
2. Write the name to FILE_IO_PATH_WRITE (with null terminator)
3. Write any value to FILE_IO_OPEN
4. Read file data from FILE_IO_READ using the size from FILE_IO_SIZE_BYTES (FILE_IO_SIZE_BYTES can be read while getting a list of items in the CWD)
5. Write a value of 1 to FILE_IO_CLOSE

Write a file:

Writing directly to the LabJack's installed SD card directly from the host computer is not supported. Most applications only need to create files from onboard Lua scripts.

- For writing configurations to the LabJack, consider writing to the User Area of Internal Flash.
- Other options for writing to an installed SD card include:
- Generating and loading a custom Lua script as needed. Lua scripts can be loaded programmatically to the LabJack.
- Writing a more generic Lua script that reads input from User RAM.

Create a directory:

Unimplemented. Since Lua scripts currently do not have the ability to write files anywhere except the root directory, this feature is not implemented.

Register Listing

File IO Navigation

Name	Start Address	Type	Access
FILE_IO_PATH_WRITE_LEN_BYTES	60640	UINT32	W

FILE_IO_PATH_WRITE_LEN_BYTES
- Address: 60640

Write the length (in bytes) of the file path or directory to access.

- Data type: UINT32 (type index = 1)
- Write-only
- This register uses system RAM. The maximum RAM is 64KB. For more information, see 4.4 RAM
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0134

FILE_IO_PATH_READ_LEN_BYTES	60642	UINT32	R

FILE_IO_PATH_READ_LEN_BYTES
- Address: 60642

Read the length (in bytes) of the next file path or directory to access.

- Data type: UINT32 (type index = 1)
- Read-only
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0134

FILE_IO_PATH_WRITE	60650	BYTE	W

FILE_IO_PATH_WRITE
- Address: 60650

Write the desired file path. Must first write the length of the file path string (in bytes) to FILE_IO_PATH_WRITE_LEN_BYTES. File paths should be null terminated. This register is a buffer.

- Data type: BYTE (type index = 99)
- Write-only
- This register is a Buffer Register
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0134

FILE_IO_PATH_READ	60652	BYTE	R

FILE_IO_PATH_READ
- Address: 60652

Read the next file path in the CWD. Length of the string (in bytes) determined by FILE_IO_PATH_READ_LEN_BYTES. File paths will be null terminated. This register is a buffer. Underrun behavior - fill with zeros.

- Data type: BYTE (type index = 99)
- Read-only
- This register is a Buffer Register
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0134

FILE_IO_DIR_FIRST	60610	UINT16	W

FILE_IO_DIR_FIRST
- Address: 60610

Write any value to this register to initiate iteration through files and directories in the CWD. Typical sequence: FILE_IO_DIR_FIRST(W), then loop through: [FILE_IO_NAME_READ_LEN(R), FILE_IO_NAME_READ(R), FILE_IO_ATTRIBUTES(R), FILE_IO_SIZE_BYTES(R), FILE_IO_DIR_NEXT(W)] ..until any of the following errors: FILE_IO_END_OF_CWD (2966), FILE_IO_INVALID_OBJECT (2809), or FILE_IO_NOT_FOUND (2960).

- Data type: UINT16 (type index = 0)
- Write-only
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0134

FILE_IO_DIR_NEXT	60611	UINT16	W

FILE_IO_DIR_NEXT
- Address: 60611

Write any value to this register to continue iteration through files and directories in the CWD.

- Data type: UINT16 (type index = 0)
- Write-only
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0134

FILE_IO_DIR_CHANGE	60600	UINT16	W

FILE_IO_DIR_CHANGE
- Address: 60600

Write any value to this register to change the current working directory (CWD). Must first designate which directory to open by writing to FILE_IO_PATH_WRITE_LEN_BYTES then FILE_IO_PATH_WRITE.

- Data type: UINT16 (type index = 0)
- Write-only
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0156

FILE_IO_DIR_CURRENT	60601	UINT16	W

FILE_IO_DIR_CURRENT
- Address: 60601

Write any value to this register to load the current working directory into FILE_IO_PATH_READ, and its length into FILE_IO_PATH_READ_LEN_BYTES. Can be used to identify current position in a file tree.

- Data type: UINT16 (type index = 0)
- Write-only
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0134

FILE_IO_DIR_MAKE	60602	UINT16	W

FILE_IO_DIR_MAKE
- Address: 60602

Unimplemented.

- Data type: UINT16 (type index = 0)
- Write-only
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 99.0000

FILE_IO_DIR_REMOVE	60603	UINT16	W

FILE_IO_DIR_REMOVE
- Address: 60603

Unimplemented.

- Data type: UINT16 (type index = 0)
- Write-only
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 99.0000

File IO File Operations

Name	Start Address	Type	Access
FILE_IO_OPEN	60620	UINT16	W

FILE_IO_OPEN
- Address: 60620

Write any value to this register to open a file. Must first designate which file to open by writing to FILE_IO_PATH_WRITE_LEN_BYTES then FILE_IO_PATH_WRITE.

- Data type: UINT16 (type index = 0)
- Write-only
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0134

FILE_IO_CLOSE	60621	UINT16	W

FILE_IO_CLOSE
- Address: 60621

Write any value to this register to close the open file.

- Data type: UINT16 (type index = 0)
- Write-only
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0134

FILE_IO_WRITE	60654	BYTE	W

FILE_IO_WRITE
- Address: 60654

Unimplemented. This register is a buffer.

- Data type: BYTE (type index = 99)
- Write-only
- This register is a Buffer Register
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 99.0000

FILE_IO_READ	60656	BYTE	R

FILE_IO_READ
- Address: 60656

Read the contents of a file. Must first write to FILE_IO_OPEN. Size of the file (in bytes) determined by FILE_IO_SIZE_BYTES. This register is a buffer. Underrun behavior - throws an error.

- Data type: BYTE (type index = 99)
- Read-only
- This register is a Buffer Register
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0134

FILE_IO_DELETE	60622	UINT16	W

FILE_IO_DELETE
- Address: 60622

Write any value to this register to delete the active file. Must first designate which file to delete by writing to FILE_IO_PATH_WRITE_LEN_BYTES then FILE_IO_PATH_WRITE.

- Data type: UINT16 (type index = 0)
- Write-only
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0134

FILE_IO_ATTRIBUTES	60623	UINT16	R

FILE_IO_ATTRIBUTES
- Address: 60623

Used to differentiate files from directories/folders.
Bitmask:
Bit0: Reserved,
Bit1: Reserved,
Bit2: Reserved,
Bit3: Reserved,
Bit4: 1=Directory,
Bit5: 1=File.

- Data type: UINT16 (type index = 0)
- Read-only
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0134

FILE_IO_SIZE_BYTES	60628	UINT32	R

FILE_IO_SIZE_BYTES
- Address: 60628

The size of the file in bytes. Directories have 0 size.

- Data type: UINT32 (type index = 1)
- Read-only
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0134

Read from the disk information registers to get free space and other information.

File IO Disk Information

Name	Start Address	Type	Access
FILE_IO_DISK_SECTOR_SIZE_BYTES	60630	UINT32	R

FILE_IO_DISK_SECTOR_SIZE_BYTES
- Address: 60630

The size of each sector in the SD card in bytes. In Windows this is called the Allocation Size.

- Data type: UINT32 (type index = 1)
- Read-only
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0134

FILE_IO_DISK_SECTORS_PER_CLUSTER	60632	UINT32	R

FILE_IO_DISK_SECTORS_PER_CLUSTER
- Address: 60632

The number of sectors in each cluster. Captured on read of FILE_IO_DISK_SECTOR_SIZE_BYTES.

- Data type: UINT32 (type index = 1)
- Read-only
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0134

FILE_IO_DISK_TOTAL_CLUSTERS	60634	UINT32	R

FILE_IO_DISK_TOTAL_CLUSTERS
- Address: 60634

The total number of clusters in the SD card. Captured on read of FILE_IO_DISK_SECTOR_SIZE_BYTES.

- Data type: UINT32 (type index = 1)
- Read-only
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0134

FILE_IO_DISK_FREE_CLUSTERS	60636	UINT32	R

FILE_IO_DISK_FREE_CLUSTERS
- Address: 60636

Free (available) clusters in the SD card. Used to determine free space. Captured on read of FILE_IO_DISK_SECTOR_SIZE_BYTES.

- Data type: UINT32 (type index = 1)
- Read-only
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0134

FILE_IO_DISK_FORMAT_INDEX	60638	UINT32	R

FILE_IO_DISK_FORMAT_INDEX
- Address: 60638

Used to determine the format of the SD card.
0=None or Unknown,
1=FAT12,
2=FAT16(Windows FAT),
3=FAT32

- Data type: UINT32 (type index = 1)
- Read-only
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0160

Name	Start Address	Type	Access
FILE_IO_LUA_SWITCH_FILE	60662	UINT32	R/W

FILE_IO_LUA_SWITCH_FILE
- Address: 60662

Write any value to this register to instruct Lua scripts to switch to a new file. Lua script should periodically check LJ.CheckFileFlag() to receive instruction, then call LJ.ClearFileFlag() after file switch is complete. Useful for applications that require continuous logging in a Lua script, and on-demand file access from a host.

- Data type: UINT32 (type index = 1)
- Readable and writable
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0168

22.0 OEM Versions [T-Series Datasheet]

Overview

The OEM versions of the T-Series devices are designed for easy customization of the connection ports with pin-headers for integration into larger systems. To reduce price:

- OEM versions of devices are sold without cases, cables, and other accessories.
- OEM versions of devices are manufactured without screw terminals and the DB15 and DB37 connectors.

Ordering

For pricing/ordering, select the appropriate OEM variant:

- T4 Product Page
- T7 Product Page
- T8 Product Page

Customization:

Custom OEM boards carry additional cost,($75 per board standard, $125 per board- RUSH) but they are often necessary for specialized enclosures and seamless integration with other products. LabJack offers a device customization service that allows for OEM devices to be ordered with custom parts installed prior to shipping. For small quantities LabJack can customize boards in house, for larger quantites we will ask our CM to perform the customizations. LabJack also re-calibrates and tests the device after performing the modifications to ensure the devices are still working properly before shipping.

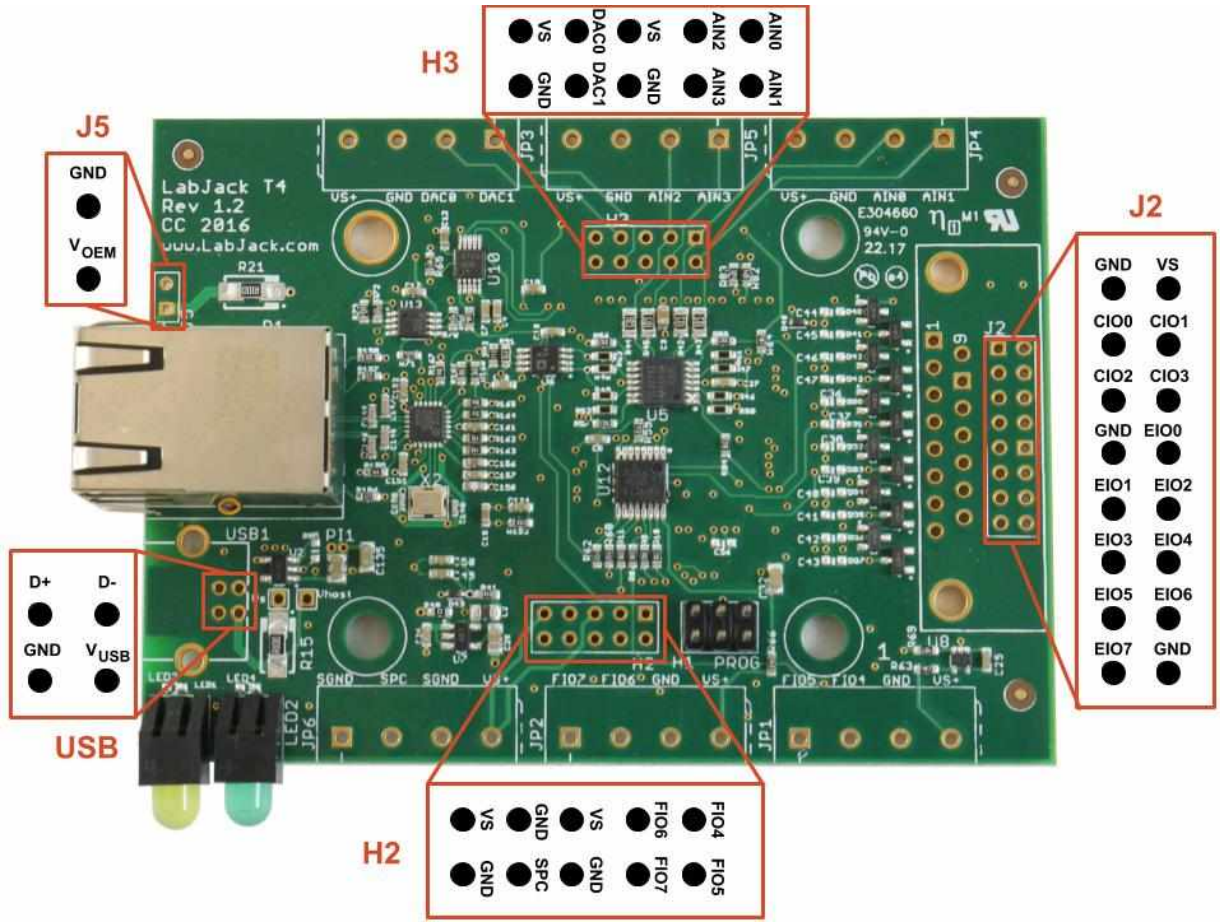
For customization, please send us an email or contact us—we will most likely need to generate a custom quote and order to fulfill and bill against. Please don't hesitate to contact us.

Lead Time: Custom OEM board lead times currently vary from a few weeks to 6+ months depending on complexity, production schedule, part availability, quantity, etc. contact us for an exact quote and lead time based on your specific build and application.

Pinouts

The OEM versions of T-series devices are shown below, with the pinouts of the (T4) H2, H3, J2, J5 and (T7) J2, J3, J5 connectors:

T4-OEM:



The T4-OEM exposes all of its I/O lines through the pin-headers H2, H3, and J2. The device can be externally powered with a regulated 5V supply using the J5 connector. If needed, a USB connector can also be installed—see the USB section below. More details about the PCB dimensions can be found in Appendix B-2 T4 Enclosure and PCB Drawings.

T7-OEM and T7-Pro-OEM:

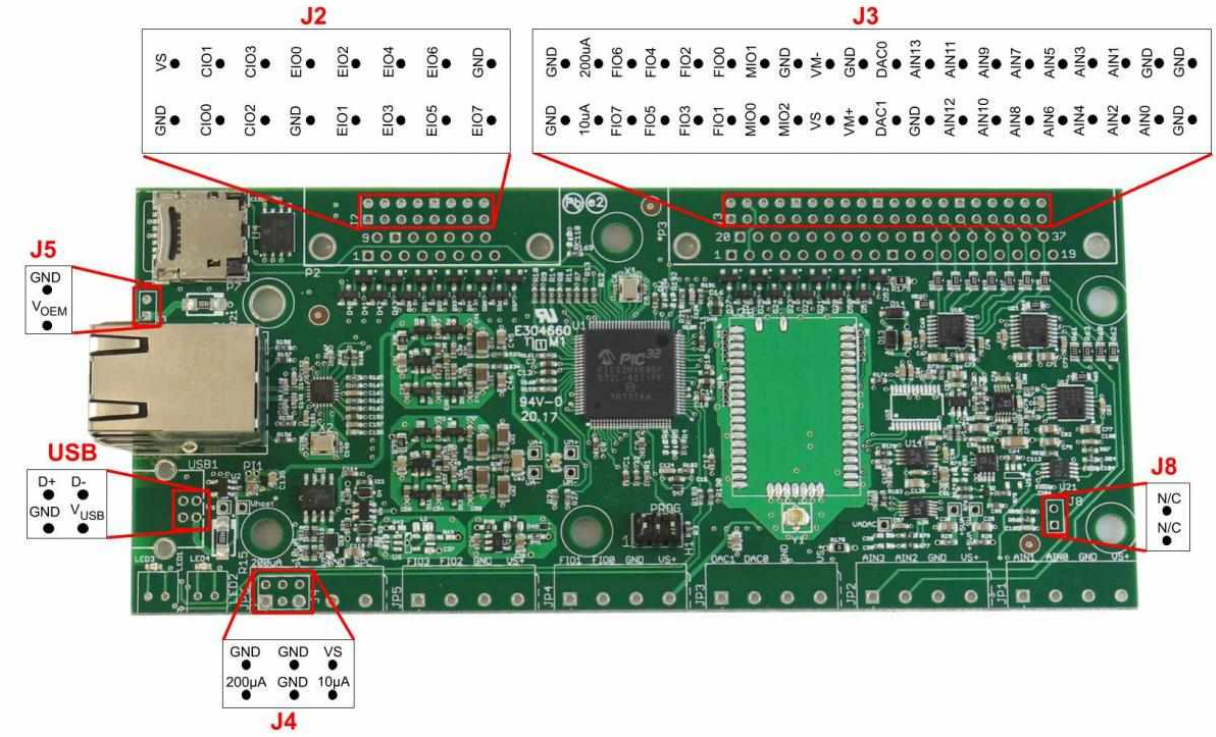


Image 22-2. T7-OEM Pinout

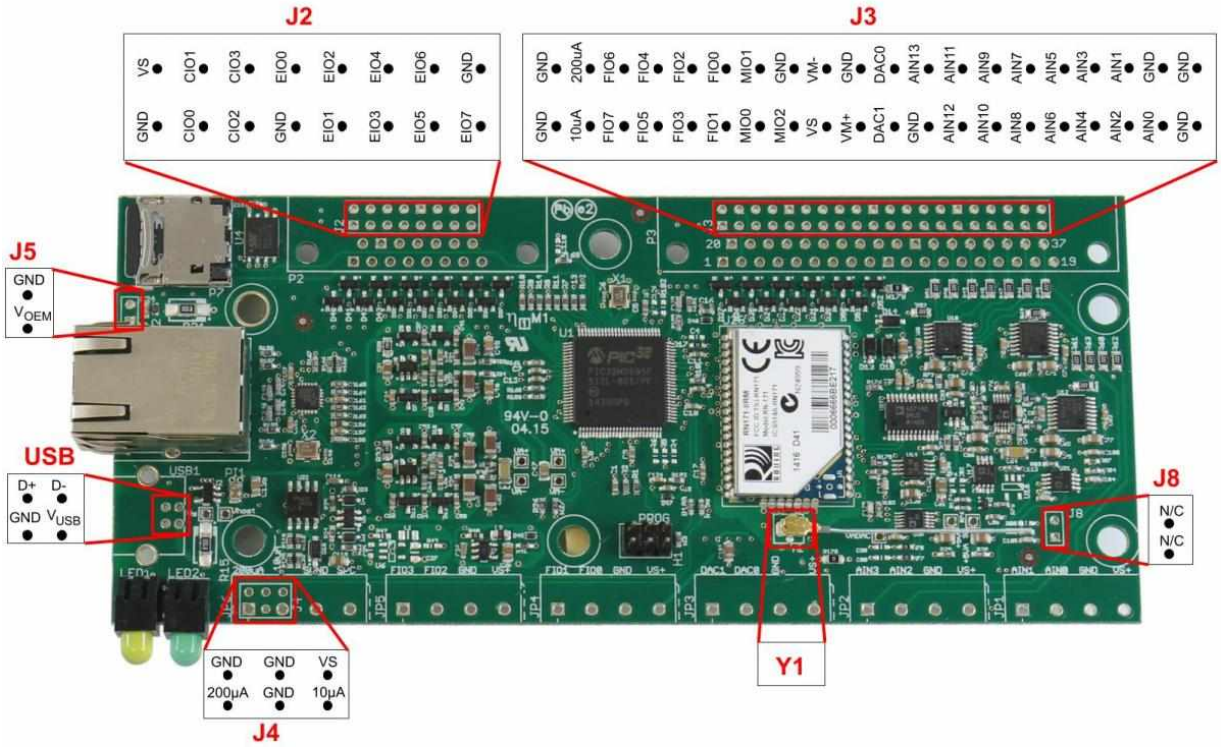


Image 22-3. T7-Pro-OEM Pinout

The T7-OEM and T7-Pro-OEM devices expose all of their I/O lines through the J2 and J3 pin-

header locations. Both devices can be externally powered with a regulated 5V supply using the J5 connector. If needed, a USB connector can also be installed—see the USB section below. More details about the PCB dimensions can be found in Appendix B-2 T7 Enclosure and PCB Drawings.

Suggested Part Orientations: We suggest that customers install pin headers on the top side of our devices to prevent there from being any issues with device re-calibration. During the calibration process, our OEM devices are mounted to the top of our test jigs (pogo pins make contact with the holes on the bottom side). Our device-calibration test jigs use a combination of the screw terminal holes as well as the DB15 and DB37 holes to test each of the I/O lines.

Pin-Headers: The T4 and T7 OEM devices have pin-outs compatible with pin-headers that have a pin pitch of 0.100" (2.54mm). Below is a list of the standard pin-headers that we install into the J2, J3, J5, H2, and H3 locations. Other parts can be installed upon request including shrouded or directional pin-headers.

Table 22-1. Pin-Headers

Name	Pins	Parts
J5	1x2	Jameco 1x2 2.54mm pin-header.
H2 and H3	2x5	Jameco 2x5 2.54mm pin-header.
J2	2x8	Jameco 2x8 2.54mm pin-header.
J3	2x20	Jameco 2x20 2.54mm pin-header.

PCB Dimensions: The PCB dimensions and a variety of mechanical drawings can be found in the Enclosure and PCB Drawings section of the T-Series datasheet.

ESD: Proper ESD precautions should be taken when handling the PCB directly. Many of the parts are ESD-resistant, but depending on the size or location of the shock, the board might be damaged.

Part Categories: Optional parts that can be installed can be broken down into the following categories:

- USB
- Alternate Power Supply (J5)
- DB15 and DB37 equivalent Pin-Header Locations (J2, J3)
- DB15/DB37 (D-Sub) Locations (P2 and P3)
- Screw Terminals (JP1-JP6)
- Ethernet Connector
- WiFi Antenna (T7-Pro Only, Y1)
- Current Sources (T7 Only, J4)
- Mechanical Header (T7 Only, J8)

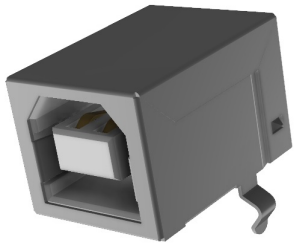
USB

The USB connector is not installed on any of the T-Series OEM devices. The T-Series devices use through-hole Type-B connectors and there are a large number of compatible connectors.

Mounting Location: The USB connector must be installed on the component side of the PCB.

Suggested Parts: We have several suggested USB connectors, however, most USB Type-B through hole connectors are compatible with the T-Series OEM devices. Standard retention USB connectors are installed on all LabJack DAQ devices and are enough for most OEM device applications. High-retention USB connectors are higher in cost, however, they more securely connect a USB cable to a device, which is important in many industrial applications.

Horizontal Mount



Standard Retention	High Retention
On Shore Technology Inc USB-B1HSW6	Samtec USBR-B-S-S-O-TH
FCI 61729-0010BLF	
TE Connectivity 292304-2	

USB Pin-Out: The pictures below show the bottom side of T-series devices and indicate the purpose of the four USB pins. Pin-outs relative to the top side are documented in the pinout pictures above.

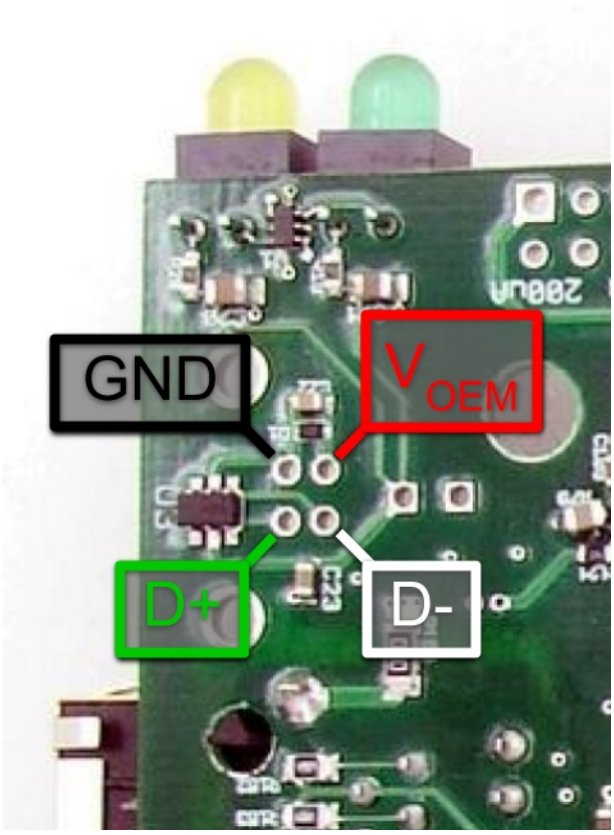


Image 22-4. T7 USB Pinout

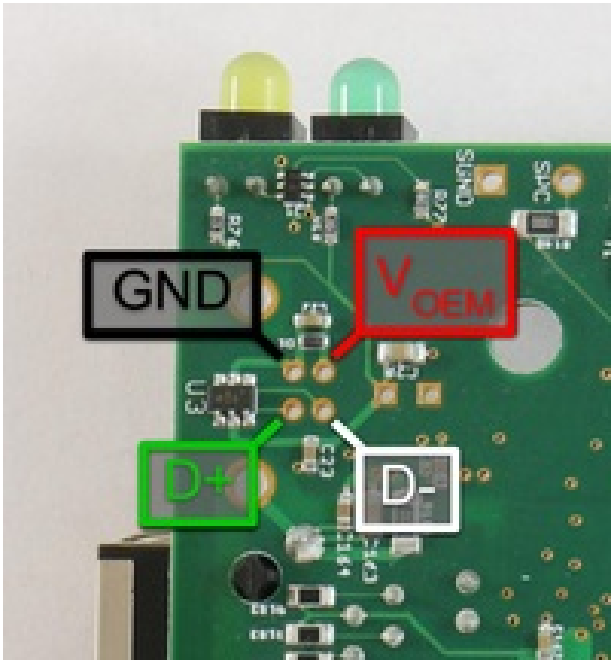


Image 22-5.T4 USB Pinout

USB Cables: A normal USB cable has a shield, and the normal Type-B connector connects the

cable shield to the mounting tabs on the connector which are then soldered into the large USB mounting holes on the PCB. If you are not using a normal USB connector and have a shield in your USB cable, we recommend that the shield be connected to either of the large USB mounting holes on the PCB. Usually the USB shield wires are aluminum—which don't take solder very well—so use a crimp connector like the Molex 02-06-2103, TE 61388-1, TE 350015-2, or the TE 60017-3. Secure the crimp connector to USB shield wires, then squish down the tip of the connector to fit into the large USB mounting holes on the PCB.

To connect to a device without USB, you can use a direct Ethernet connection. Use an SPC-to-AIN3 jumper to temporarily configure a static IP.

Alternate Power Supply (J5)

T-Series OEM devices can be powered through a few different connectors such asthe J5 pin-header holes. The J5 pin-header is useful for applications that only need Ethernet or WiFi device connections. The square shaped pad of the J5 pin-header is V+ and the circular pad is GND. The J5 connector is a 2-pin 0.1" pitch rectangular header. To prevent accidentally switching V+ and GND, use a keyed connector such as TE Connectivity 3-641215-2.

All device power supply options require a 5V input voltage. See the power supply input section of Appendix-A-5 for detailed specs.

T4, T7

The T4 and T7 only support power via USB and power via the J5 power headers (VS-OEM below).

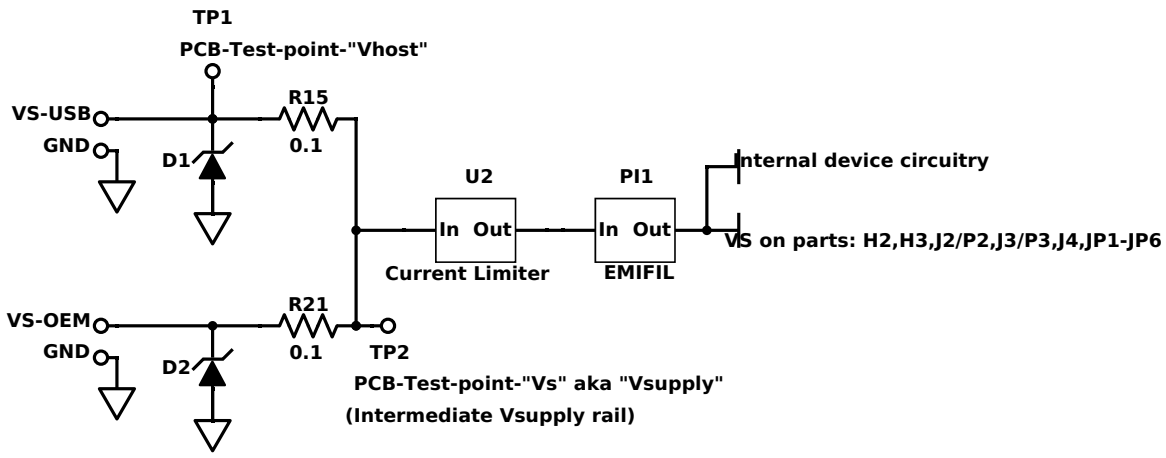


Image 22-6. T4/T7 Power Supply

The 5V supply from J5 (J5-VS) is protected by a TVS (transient voltage suppressor diode D2), then goes through R21 (0.1 ohms), and then connects to the internal Vsupply rail.

The 5V supply from the USB (USB-VS) is exposed through a test point on the PCB labeled Vhost and is protected by the TVS diode D1 and R15 (0.1 ohms) before connecting to the internal Vsupply rail.

The internal Vsupply rail can be measured using the test point on the PCB labeled Vs. After joining, power flows through a current limiting chip (currently the Diodes AP2141WG) and an EMIFIL (EMI suppression filter) before connecting to the device-wide VS bus and out to each of the screw terminals labeled as VS.

Dual Power Supply: R15 and R21 are installed by default and thus the connections for both sources are essentially shorted to each other. Both power supply options should not be connected to a voltage source at the same time. If this happens, one power supply could back-feed the other which may cause damage. If a device is going to be powered using the J5 connector and there is a possibility of power at the USB connection (and USB power is not required) then it is recommended to remove R15 so that only the J5 pin-header is used to power the device. If a dual-power supply method is required, it is recommended to replace the R15 and R21 resistors with Schottky diodes (SMA package). There will still be a small voltage drop that needs to be considered but the device should operate correctly as long as the voltage present on the VS screw terminals is above 4.75V (reference).

T8

Production of the OEM version of the T8 has not been started yet. If you need an OEM version, contact support@labjack.com.

The default configuration of the T8 has two power supply options, USB and PoE (Power over Ethernet). The PoE side is always prioritized; when the PoE exceeds a voltage threshold, that power supply will be used regardless of the voltage on the other supply. The same prioritization system is applied when using J5 (V_{OEM} in image 22-7). If J5 replaces the PoE it will be prioritized. If J5 replaces USB then PoE will still be prioritized.

To configure the T8 for the desired power selection, a few resistors need to be installed or removed. Power supply configurations are:

Configuration	R15	R21	R86	R87
USB and PoE-prioritized	Installed	Not Installed	Not Installed	Installed
USB and J5-prioritized	Installed	Not Installed	Installed	Not Installed
J5 and PoE-prioritized	Not Installed	Installed	Not Installed	Installed

More than Two Power Supplies: If both R15 and R21 or R86 and R87 are installed, then some of the power supplies will be shorted together. Shorted power supplies should not be connected to a voltage source at the same time. If this happens one power supply could back-feed the other, which may cause damage.

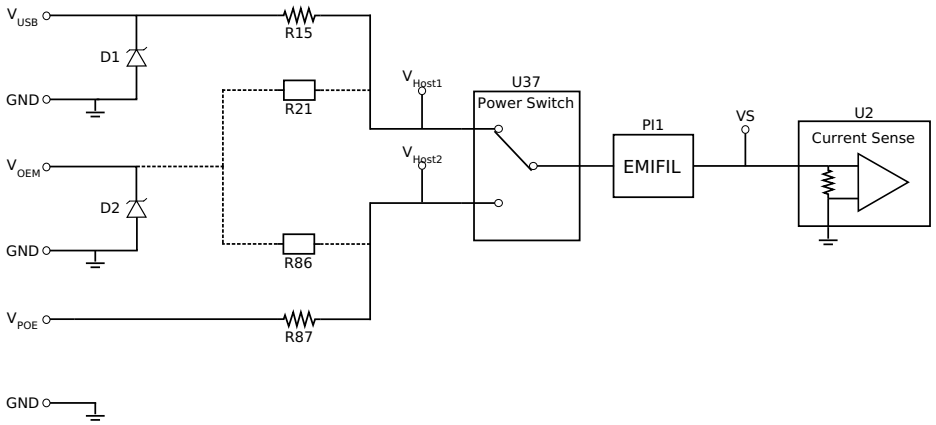


Image 22-7. T8 Power Circuitry

DB15 and DB37 equivalent Pin-Header Locations (J2, J3)

Connectors J2 and J3 provide pin-header alternatives to the DB15 and DB37 connectors. The J2 and J3 holes are always present, but are obstructed when the DB15 and DB37 are installed.

- The T4-OEM only has the J2 header.
- The T7-OEM and T7-Pro-OEM both have the J2 and J3 headers.

J2 and J3 can be seen in the pictures at the top of this page which show the component side of each of the T-Series device PCBs. For both the DBs and pin headers, holes with a square solder pad indicate pin number 1, 10, 20, or 30. For both the DBs and pin headers, pin 1 is at the lower-left. For the DB connectors the pin numbers increment from 1 left to right across the bottom row, and then continue left to right across the top row. For the pin headers, the odd pins increment left to right across the bottom row (1, 3, 5, ...) and the even pins increment left to right across the top row (2, 4, 6, ...).

J2 - 16 position, 2 row, 0.1" pitch, male pin rectangular header

- Unshrouded - Harwin Inc M20-9980846
- Unshrouded 3x Taller - Samtec Inc TSW-108-17-T-D
- Shrouded, Gold Finish - On Shore Technology Inc 302-S161
- Shrouded, Right Angle - TE Connectivity 1-1634689-6

J3 - 40 position, 2 row, 0.1" pitch, male pin rectangular header

- Unshrouded - Harwin Inc M20-9762046
- Unshrouded 3x Taller - Samtec Inc TSW-120-17-T-D
- Shrouded, Gold Finish - On Shore Technology Inc 302-S401

- Shrouded, Right Angle - TE Connectivity 5103310-8
- Shrouded, Gold-Palladium Finish - TE Connectivity 5104338-8

Sometimes customers order tall pin headers that mate directly to a separate custom PCB. Refer to the pinout details below for electrical connections.

Table 22-2. J2 Pinouts

1	GND	2	VS
3	CIO0	4	CIO1
5	CIO2	6	CIO3
7	GND	8	EIO0
9	EIO1	10	EIO2
11	EIO3	12	EIO4
13	EIO5	14	EIO6
15	EIO7	16	GND

Table 22-3. J3 Pinouts

1	GND	2	GND	3	PIN20 (10uA)
4	PIN2 (200uA)	5	FIO7	6	FIO6
7	FIO5	8	FIO4	9	FIO3
10	FIO2	11	FIO1	12	FIO0
13	MIO0	14	MIO1	15	MIO2
16	GND	17	Vs	18	Vm-
19	Vm+	20	GND	21	DAC1
22	DAC0	23	GND	24	AIN13
25	AIN12	26	AIN11	27	AIN10
28	AIN9	29	AIN8	30	AIN7
31	AIN6	32	AIN5	33	AIN4
34	AIN3	35	AIN2	36	AIN1
37	AIN0	38	GND	39	GND
				40	GND

DB15/DB37 (D-Sub) Locations (P2 and P3)

The DB15 and DB37 connectors are not installed on OEM T-Series devices. Customers will typically use the rectangular header locations (J2, J3) instead of the DB connectors. However, if a different DB mating style is required, it is possible to buy an OEM variant and specify custom parts that need to be installed. The DB connectors are standard D-Sub two row receptacles (female sockets), through hole, 15 pin, and 37 pin. The following represent a few valid options:

- FCI 10090099-S154VLF
- FCI D15S33E4GV00LF
- Sullins Connector Solutions SDS101-PRW2-F15-SN13-1
- FCI 10090099-S374VLF
- FCI D37S33E4GV00LF
- Sullins Connector Solutions SDS101-PRW2-F37-SN83-6

Screw Terminals (JP1x)

The screw terminals are not installed on the T-Series OEM device variants. Customers will typically use the rectangular header locations (J2, J3) instead of the screw terminals. However, if a different screw terminal style is required, it is possible to buy an OEM variant and specify custom parts that need to be installed. The screw terminal holes are compatible with almost all 4 position, 1 level, 0.197" (5.00mm) pitch terminal blocks (search results from Digikey). A few possible options are listed below:

- The Phoenix Contact 1729034 works well and accepts 14-24 AWG wire.
- The Wurth Electronics 691137710004 is a good low cost option.
- The previously recommended Weidmuller 9993300000 works quite well, and accepts 14-24 AWG wire. As of 6/26/2019 this part is Obsolete.

Ethernet Connector

The Ethernet connector (XFMRS XFATM9-CTCY1-4M, Wurth 74990112116A) is installed on all T-Series device variations due to the inherent magnetic complexities. However, it is possible to "bring out" a duplicate Ethernet jack to any custom enclosure with one of the following:

- A short Ethernet cable segment and an RJ45 coupler (Plug to Plug). These couplers come in a few varieties: Free hanging (in-line), Chassis Mount, Panel Mount, Bulkhead, Wall Plate, etc. Conec 33TS3101S-88N and Emerson 30-1008KUL are both good options.
- A RJ45 Jack to Plug cable, which is just a standard Ethernet plug on one end, and a Jack (female) on the other end. Again, these come in a wide variety of mounting styles, the simplest of which is the panel mount. TE Connectivity 1546414-4 and Amphenol RJFEZ2203100BTX are both good options.

If selecting your own Ethernet interconnect, ensure that it is RJ45, straight-through, and

without magnetics.

WiFi Antenna (T7-Pro Only, Y1)

The T7-Pro-OEM ships with a simple 30mm U.FL whip antenna such as the Anaren 66089-2406. See "Antenna Details" in the WiFi section.

Current Sources (T7 Only, J4)

Since the screw terminals are not installed on an OEM T7, the J4 header location can be used to gain access to the constant current sources. Any 6 position 0.1" pitch rectangular header will work.

Table 22-4. J4 Pinout

1	200µA
2	GND
3	10µA
4	GND
5	GND
6	VS

Mechanical Header (T7 Only, J8)

The J8 pin header location is purely for mechanical support for that region of the board. There are no electrical connections to either of these pins. It is a 2 position 0.1" pitch rectangular header.

23.0 Watchdog [T-Series Datasheet]

Overview

The Watchdog system can perform various actions if the T-Series device does not receive any communication within a specified timeout period. Actions include:

- Reset/restart the device
- Set DIO
- Set DAC0/DAC1
- Reset IO configs

Example: Reset after no communication for 60 seconds

The most common way to use Watchdog is to write:

```
WATCHDOG_ENABLE_DEFAULT=0       // Disable the Watchdog
WATCHDOG_TIMEOUT_S_DEFAULT=60   // Set the timeout in seconds
WATCHDOG_RESET_ENABLE_DEFAULT=1 // Enable reset upon timeout
WATCHDOG_ENABLE_DEFAULT=1       // Enable the Watchdog
```

With this configuration, the Watchdog will cause the device to reset if it does not receive any communication for 60 seconds. In other words, if nothing is talking to the device, it will reset every 60 seconds.

Use the IO Config system to configure the power-up defaults as desired.

Basic Usage

A typical usage is to use the IO Config system to set the power-up defaults as desired, then configure the Watchdog to reset the device on timeout.

Write 0 to WATCHDOG_ENABLE_DEFAULT to disable the Watchdog while setting Watchdog configurations. Write 1 to it to enable the Watchdog:

Name	Start Address	Type	Access
WATCHDOG_ENABLE_DEFAULT	61600	UINT32	R/W

WATCHDOG_ENABLE_DEFAULT
- Address: 61600

Write a 1 to enable the watchdog or a 0 to disable. The watchdog must be disabled before writing any of the other watchdog registers (except for WATCHDOG_STRICT_CLEAR).

- Data type: UINT32 (type index = 1)
- Readable and writable
- Default value: 0
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0016

Use WATCHDOG_TIMEOUT_S_DEFAULT to set the timeout in seconds:

Name	Start Address	Type	Access
WATCHDOG_TIMEOUT_S_DEFAULT	61604	UINT32	R/W

WATCHDOG_TIMEOUT_S_DEFAULT
- Address: 61604

When the device receives any communication over USB/Ethernet/WiFi, the watchdog timer is cleared. If the watchdog timer is not cleared within the timeout period, the enabled actions will be done.

- Data type: UINT32 (type index = 1)
- Readable and writable
- Default value: 0
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0016

The timeout period is reset when a response to a command-response packet is sent to the host, except when strict mode is enabled (see below).

In addition to enabling the Watchdog and setting a timeout, the Watchdog needs to be configured to take an action when the timeout is complete.

Actions

Most applications will simply need the Watchdog to reset the device. Use WATCHDOG_RESET_ENABLE_DEFAULT to enable device reset:

Name	Start Address	Type	Access
WATCHDOG_RESET_ENABLE_DEFAULT	61620	UINT32	R/W

WATCHDOG_RESET_ENABLE_DEFAULT
- Address: 61620

Timeout action: Set to 1 to enable device-reset on watchdog timeout.

- Data type: UINT32 (type index = 1)
- Readable and writable
- Default value: 0
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0016

To set DIO upon timeout, set the following configurations:

Name	Start Address	Type	Access
WATCHDOG_DIO_ENABLE_DEFAULT	61630	UINT32	R/W

WATCHDOG_DIO_ENABLE_DEFAULT
- Address: 61630

Timeout action: Set to 1 to enable DIO update on watchdog timeout.

- Data type: UINT32 (type index = 1)
- Readable and writable
- Default value: 0
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0016

WATCHDOG_DIO_STATE_DEFAULT	61632	UINT32	R/W

WATCHDOG_DIO_STATE_DEFAULT
- Address: 61632

The state high/low of the digital I/O after a Watchdog timeout. See DIO_STATE

- Data type: UINT32 (type index = 1)
- Readable and writable
- Default value: 0
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0016

WATCHDOG_DIO_DIRECTION_DEFAULT	61634	UINT32	R/W

WATCHDOG_DIO_DIRECTION_DEFAULT
- Address: 61634

The direction input/output of the digital I/O after a Watchdog timeout. See DIO_DIRECTION

- Data type: UINT32 (type index = 1)
- Readable and writable
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0016

WATCHDOG_DIO_INHIBIT_DEFAULT	61636	UINT32	R/W

WATCHDOG_DIO_INHIBIT_DEFAULT
- Address: 61636

This register can be used to prevent the watchdog from changing specific IO. See DIO_INHIBIT for more information.

- Data type: UINT32 (type index = 1)
- Readable and writable
- Default value: 0
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0016

To set DAC0 or DAC1 upon timeout, set the following configurations:

Name	Start Address	Type	Access

WATCHDOG_DAC0_ENABLE_DEFAULT	61640	UINT32	R/W

WATCHDOG_DAC0_ENABLE_DEFAULT
- Address: 61640

Timeout action: Set to 1 to enable DAC0 update on watchdog timeout.

- Data type: UINT32 (type index = 1)
- Readable and writable
- Default value: 0
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0016

WATCHDOG_DAC0_DEFAULT	61642	FLOAT32	R/W

WATCHDOG_DAC0_DEFAULT
- Address: 61642

The voltage of DAC0 after a Watchdog timeout.

- Data type: FLOAT32 (type index = 3)
- Readable and writable
- Default value: 0
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0016

WATCHDOG_DAC1_ENABLE_DEFAULT	61650	UINT32	R/W

WATCHDOG_DAC1_ENABLE_DEFAULT
- Address: 61650

Timeout action: Set to 1 to enable DAC1 update on watchdog timeout.

- Data type: UINT32 (type index = 1)
- Readable and writable
- Default value: 0
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0016

WATCHDOG_DAC1_DEFAULT	61652	FLOAT32	R/W

WATCHDOG_DAC1_DEFAULT
- Address: 61652

The voltage of DAC1 after a Watchdog timeout.

- Data type: FLOAT32 (type index = 3)
- Readable and writable
- Default value: 0
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0016

To reset IO configs, set the following advanced configuration:

Name	Start Address	Type	Access
WATCHDOG_ADVANCED_DEFAULT	61602	UINT32	R/W

WATCHDOG_ADVANCED_DEFAULT
- Address: 61602

A single binary-encoded value where each bit is an advanced option. If bit 0 is set, IO_CONFIG_SET_CURRENT_TO_FACTORY will be done on timeout. If bit 1 is set, IO_CONFIG_SET_CURRENT_TO_DEFAULT will be done on timeout.

- Data type: UINT32 (type index = 1)
- Readable and writable
- Default value: 0
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0016

Short Timeouts

The Watchdog timeout can be set as low as 1 second—but such a low value can prevent the device from communicating if reset is enabled. For example, when a USB device resets it takes a little time for USB to re-enumerate and software to be able to talk to the device again, so you could get in a situation where the device keeps resetting so often that you can't start talking to it again. This might require using the reset-to-factory jumper—see 11.0 SPC for details.

Strict Mode

The default timeout period is reset when a response to a command-response packet is sent to the host. Alternatively, "strict" mode can be enabled using WATCHDOG_STRICT_ENABLE_DEFAULT:

Name	Start Address	Type	Access
WATCHDOG_STRICT_ENABLE_DEFAULT	61610	UINT32	R/W

WATCHDOG_STRICT_ENABLE_DEFAULT
- Address: 61610

Set to 1 to enable strict mode.

- Data type: UINT32 (type index = 1)
- Readable and writable
- Default value: 0
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0016

When strict mode is enabled, the timeout period is reset by writing the key value to WATCHDOG_STRICT_CLEAR:

Name	Start Address	Type	Access
WATCHDOG_STRICT_CLEAR	61614	UINT32	W

WATCHDOG_STRICT_CLEAR
- Address: 61614

When running in strict mode, writing the key to this register is the only way to clear the watchdog. Writing to this register while not using strict mode will clear the watchdog.

- Data type: UINT32 (type index = 1)
- Write-only
- Default value: 0
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0016

To set the key, write to WATCHDOG_STRICT_KEY_DEFAULT:

Name	Start Address	Type	Access
WATCHDOG_STRICT_KEY_DEFAULT	61612	UINT32	R/W

WATCHDOG_STRICT_KEY_DEFAULT
- Address: 61612

When set to strict mode, this is the value that must be written to the clear register.

- Data type: UINT32 (type index = 1)
- Readable and writable
- Default value: 0
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0016

When strict mode is disabled, writing any value to WATCHDOG_STRICT_CLEAR will clear the Watchdog.

Writing to WATCHDOG_STRICT_CLEAR will clear the Watchdog when the write is processed, not when a response packet is sent.

When Using Stream

Normal spontaneous stream data does not reset the Watchdog timeout. Write periodic command-response communication to reset the Watchdog timeout. For the difference between command-response and stream, see the Communication section.

Startup Delay

To set an initial delay, use WATCHDOG_STARTUP_DELAY_S_DEFAULT:

Name	Start Address	Type	Access
WATCHDOG_STARTUP_DELAY_S_DEFAULT	61606	UINT32	R/W

WATCHDOG_STARTUP_DELAY_S_DEFAULT
- Address: 61606

This specifies the initial timeout period at device bootup. This is used until the first time the watchdog is cleared or timeout ... after that the normal timeout is used.

- Data type: UINT32 (type index = 1)
- Readable and writable
- Default value: 0
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0016

24.0 IO Config, _DEFAULT [T-Series Datasheet]

Overview

IO_Config controls the default configuration that will be used when the device boots up, and can set an already running device to a known state. The default configuration sets the values that will be used during boot-up for digital directions, digital states, DACs, AIN_EF, and DIO_EF. IO_Config can also apply either the default or factory configuration to a device that is already running.

Subsections

Configuration

_DEFAULT

Registers ending with "_DEFAULT" will store non-volatile settings. These non-volatile settings will be used to configure the T-Series device during boot-up.

Terms

- DEFAULT – A saved configuration set. These are also the settings that will be used when the device boots up.
- FACTORY – The default settings that are loaded when the device is tested at LabJack.
- CURRENT – The device's current settings.

Register Listing

Use the following registers to configure IO_Config:

Name	Start Address	Type	Access
IO_CONFIG_SET_DEFAULT_TO_CURRENT	49002	UINT32	W

IO_CONFIG_SET_DEFAULT_TO_CURRENT
- Address: 49002

Write a 1 to cause new default (reboot/power-up) values to be saved to flash. Current values are retrieved and saved as the new defaults. Systems affected: AIN, DIO, DAC, AIN_EF, DIO_EF.

- Data type: UINT32 (type index = 1)
- Write-only

IO_CONFIG_SET_DEFAULT_TO_FACTORY	49004	UINT32	W

IO_CONFIG_SET_DEFAULT_TO_FACTORY
- Address: 49004

Write a 1 to cause new default (reboot/power-up) values to be saved to flash. Factory values are retrieved and saved as the new defaults. Systems affected: AIN, DIO, DAC, AIN_EF, DIO_EF.

- Data type: UINT32 (type index = 1)
- Write-only

IO_CONFIG_SET_CURRENT_TO_FACTORY	61990	UINT16	W

IO_CONFIG_SET_CURRENT_TO_FACTORY
- Address: 61990

Write a 1 to set current values to factory configuration. The factory values are retrieved from flash and written to the current configuration registers. Systems affected: AIN, DIO, DAC, AIN_EF, DIO_EF.

- Data type: UINT16 (type index = 0)
- Write-only

IO_CONFIG_SET_CURRENT_TO_DEFAULT	61991	UINT16	W

IO_CONFIG_SET_CURRENT_TO_DEFAULT
- Address: 61991

Write a 1 to set current values to default configuration. The default values are retrieved from flash and written to the current configuration registers, thus this behaves similar to reboot/power-up. Systems affected: AIN, DIO, DAC, AIN_EF, DIO_EF.

- Data type: UINT16 (type index = 0)
- Write-only

Checksum

The IO_CONFIG_CURRENT_CRC32 register returns a CRC32 of the configuration data:

Name	Start Address	Type	Access
IO_CONFIG_CURRENT_CRC32	49020	UINT32	R

IO_CONFIG_CURRENT_CRC32
- Address: 49020

Collects the current IO configuration and calculates of a CRC32.

- Data type: UINT32 (type index = 1)
- Read-only
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0220

The CRC can be used to detect configuration changes. Calculating the checksum takes between 5 and 15 ms. Some configurations are excluded from the CRC because they are expected to change during normal operations. The excluded configurations are:

- Digital States
- Digital Directions
- DAC Voltages

Factory Reset

IO_Config settings can be cleared by a factory reset. See the 11.0 SPC section for more information.

Startup Delay

When a T-Series device starts up it must perform an initialization sequence before the startup defaults can be applied. Before the startup settings are applied, all DIO are in the input state which has a 100k pull-up to 3.3V. EIO0 is a special case, during this pre-config time only, where it behaves more like it has a 10k pull-up to 3.3V. The pull-ups on the T4 default to off. Without the pull-ups the lines are floating until the startup settings are applied. Startup defaults are divided into two groups depending on where they are applied in the startup sequence. The two groups are:

- Fast DIO - Directions and States for DIO lines which are not used for SPC jumpers. Typically applied ~2.5 ms (Tested with T7 1.0288) after power is applied. This is true for all T4 bootloaders, and T7 bootloader 0.96. Prior to T7 bootloader 0.96, all DIO acted like "Slow DIO" described below.

- Slow DIO - Directions and States for DIO lines which are used for SPC jumpers, DIO_EF for all lines, AIN_EF, Ethernet, analog settings, etc. Typically applied ~700 ms (Tested with T7 1.0288) after power is applied. Note that the firmware update process will increase this time to ~12 seconds.

Saving I/O Configurations using Kipling

The Global Configurations and Power-Up Defaults tabs in Kipling can also be used to setup I/O Configurations.

Example

Use normal current configuration registers to write some values, and then save those as defaults so they are in effect at power-up:

```
AIN_ALL_RANGE = 0.1 // Set current range of all AIN to +/-
0.1V
```

```
AIN_ALL_RESOLUTION_INDEX = 12 // Set current resolution index of all AIN to
12.
```

```
IO_CONFIG_SET_DEFAULT_TO_CURRENT = 1 // Set power-up defaults to current
values.
```

24.1 Cleanse (T7 Only) [T-Series Datasheet]

The Cleanse function will reset non-volatile user data and settings to the factory defaults.

To prevent errors, the WiFi module will be disabled.

Requires firmware 1.0225.

Triggering a Cleanse

To trigger a cleanse, write 0x5317052E to the CLEANSE register:

Name	Start Address	Type	Access
CLEANSE	49090	UINT32	W

CLEANSE
- Address: 49090

Writing 0x5317052E to this register will trigger a system cleanse.

- Data type: UINT32 (type index = 1)
- Write-only

The LEDs will blink in an alternating pattern to show that the cleanse is in progress. If the LEDs blink in unison, then an error has occurred.

The CLEANSE function will take a few seconds to complete. This will cause LJM to throw a LJME_RECONNECT_FAILED error (#1239). To prevent this error, LJM's timeout needs to be increased. Alternatively, LJME_RECONNECT_FAILED errors can be ignored.

Once successfully cleared, the items listed below will be cleared.

Items Cleared:

- Device Name
- AIN_EF Settings
- DIO_EF Settings
- DIO States and Directions
- AIN Settings
- Watchdog Timer
- Lua Startup Settings
- Lua Startup Script
- Flash User Space
- Battery Backed RAM (Pro only)
- Ethernet Settings

- Some WiFi Settings

Not Cleared:

- Settings stored in the WiFi module are not erased. Settings in the WiFi module are SSID and Password. To clear these settings, use Kipling to write dummy values.

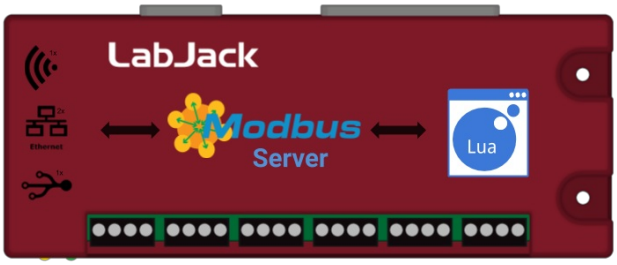
25.0 Lua Scripting [T-Series Datasheet]

Overview

T-Series devices can execute Lua code to allow custom, independent operation. A Lua script can be used to collect data without a host computer or be used to perform complex tasks producing simple results that a host can read.

For maximum speed and benchmarking, see Lua Script Performance.

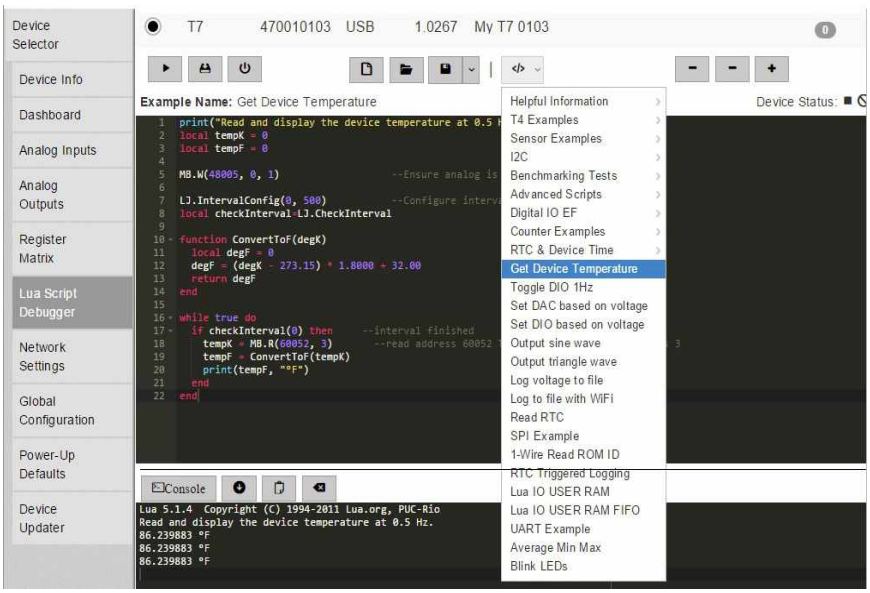
If you have some scripting experience (with a language like Python, for example), the information in this Lua Scripting section combined with the Lua examples are typically sufficient to achieve desired programmatic behavior



Subsections

Getting Started

1. Connect your device to your computer, launch Kipling, and navigate to the Lua Script Debugger tab.



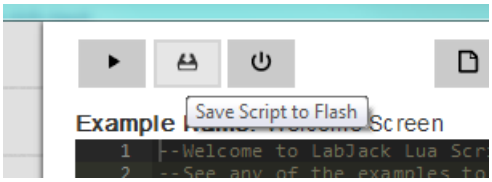
2. Open the "Get Device Temperature" example and click the Run button. The console should show the current device temperature. Now click the Stop button.

3. Try out some other examples.

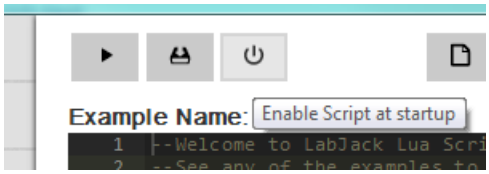
Running a script when the device powers up

A T-Series device can be configured to run a script when it powers on or resets. Typically you should test scripts for a while with the Run/Stop buttons while viewing the debug output in the console. Then once everything is working correctly, enable the script at startup and close Kipling.

To enable the script at startup:



1. Click Save Script to Flash.



2. Click Enable Script at Startup. Now when the device is powered on or reset, it will run your script.

3. After power cycling the device, if it becomes un-usable and the COMM/Status lights are constantly blinking the Lua Script is likely causing the device to enter an invalid state. The device can be fixed by connecting a jumper wire between the SPC terminal and either FIO0 or FIO4, see the SPC section of the datasheet for more details.

A short video tutorial describing how to do this is available on the Screen Casting and Lua Script Tutorials news post.

Learning more about Lua

Learning Lua is very easy. There are good tutorials on Lua.org as well as on several other independent sites. If you are familiar with the basics of programming, such as loops and functions, then you should be able to get going just by looking at the examples.

Lua for T-Series Devices

Try to keep names short. String length directly affects execution speed and code size.

Use local variables instead of global variables (it's faster and less error-prone). For example, use:

```
local a = 10
```

instead of:

```
a = 10
```

You can also assign a function to a local for further optimization:

```
local locW = MB.W
locW(2003, 0, 1) --Write to address
2003
```

Lua supports multi-return. Here, both table and error are returned values:

```
table, error = MB.RA(Address, dataType, nValues)
```

Freeing Lua memory

Since Lua occupies system RAM, it's good to clean up a Lua script when it's no longer needed. When a Lua script ends, by default it does not free memory used by the Lua. The following methods clean up Lua memory:

- Press the stop button in the Lua script tab in Kipling
- Write 0 to LUA_RUN
- Write 0 to LUA_RUN as the last command of your Lua script

For more memory cleaning options, see the system RAM section.

Limitations of Lua on T-Series Devices

Lua on the T-series devices has several limitations:

No Direct Network Communication: Lua scripts cannot directly send or receive data. Lua scripts cannot act as a network client. As an alternative, Lua scripts can output to USER_RAM, which an external service can then poll. More generally, T-series devices act only as servers and cannot act as clients.

Speed: Maximum data rates can only be achieved with a host computer. Lua can not handle as much data, but is not limited by the communication overhead that a host computer is. The lack of overhead means that Lua can respond more quickly. See Lua script performance for more information.

Data Types: Depending on which T-series device you are using, Lua's numeric data type is either IEEE 754 single precision (float) or double precision (double). Refer to the table below to determine which datatype your device uses.

Device	Data Type
T4	single precision (float)
T7	single precision (float)
T8	double precision (double)

When writing a script for a device that uses single precision, there are precision limitations which must be understood and worked with. Here is a good article on floating point numbers and their pitfalls: Floating Point Numbers. The single precision data type means that working with 32-bit values requires extra consideration. See the Working with 32-bit Values section below.

Script size: Lua scripts run within the device's RAM. The Lua virtual machine requires about 25 kBytes. Script code adds memory requirements from there. Both the code itself and RAM allocated by the code require space in RAM. The amount of RAM available depends on which T-Series device is being used as described in our RAM documentation.

When memory starts to get full (when running a script through Kipling, a message "lua: not enough memory" is displayed), there are two places that are likely to throw errors:

1. When a Lua script is loaded and started, but has not yet started running. When a script is started, the source code is transferred to the device and that code is compiled into byte code. Memory for the compilation process can be freed up by reducing comments and the lengths of variables and functions.
2. Memory can be freed up using the collectgarbage function. Collecting prevents garbage

from building up as much. If more memory is still needed, the script needs to be simplified or shrunk. See the information in the Weak Table Lua Documentation.

Notice: From a practical standpoint, Lua Scripts will start becoming too long and throwing out-of-ram errors after around 150 lines (depending on how long each line of code is). Once this limitation is encountered, there are a few tricks that can be used to implement additional features. They all involve making your code less readable but will assist in implementing additional features.

1. Remove any comments from your code as these consume precious RAM.
2. Edit the lua script file using an editor outside of Kipling and upload a minified version of the script instead of the full script file. The suggested tool to use for minifying Lua Scripts is here: https://mothereff.in/lua-minifier. Another Lua minifier is here: https://goonlinetools.com/lua-minifier/

Most Systems Still Involve A Host Computer

Lua scripting allows basic standalone operation, and some applications are totally isolated and totally standalone (also see SD Card section of this datasheet), but most systems use on-board Lua scripting in conjunction with a host computer to enable elegant solutions to complex challenges.

If you are considering Lua scripting to avoid the cost/size/power of a full-blown desktop host computer, consider that a host computer can be a simple SBC (single board computer) such as Raspberry Pi, BeagleBone, or various options from Technologic Systems. Combining a LabJack with a Linux SBC provides a low-cost solution that is very powerful and flexible.

Lua libraries

Most of the Lua 5.1 libraries are available, with the exception of functions that rely on a host operating systems, such as Time and Networking.

There are some LabJack-specific libraries:

- I2C Library: Provides functions which simplify and reduce the memory requirements of scripts that use I2C.
- Bit Library: Provides bitwise functions such as AND, OR, NOR, and XOR.
- LabJack Library: Provides control of script timing and access hardware features of the LabJack device.

Passing data into/out of Lua

User RAM consists of a list of volatile Modbus addresses where data can be sent to, and read from, a Lua script. Lua writes to the Modbus registers, and then a host device can read that information.

There are a total of 200 registers of pre-allocated RAM, which is split into several groups so that users may access it conveniently with different data types.

Use the following USER_RAM registers to store information:

User RAM Registers

Name	Start Address	Type	Access
USER_RAM#(0:39)_F32	46000	FLOAT32	R/W

USER_RAM#(0:39)_F32
- Starting Address: 46000

Generic RAM registers. Useful for passing data between a host computer and a Lua script. Will not return an error if alternate data types are used.

- Data type: FLOAT32 (type index = 3)
- Readable and writable
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0023

Expanded Names	Addresses
USER_RAM0_F32, USER_RAM1_F32, USER_RAM2_F32, USER_RAM3_F32, USER_RAM4_F32, USER_RAM5_F32, USER_RAM6_F32, USER_RAM7_F32, USER_RAM8_F32, USER_RAM9_F32, USER_RAM10_F32, USER_RAM11_F32, USER_RAM12_F32, USER_RAM13_F32, USER_RAM14_F32, USER_RAM15_F32, USER_RAM16_F32, USER_RAM17_F32, USER_RAM18_F32, USER_RAM19_F32, USER_RAM20_F32, USER_RAM21_F32, USER_RAM22_F32, USER_RAM23_F32, USER_RAM24_F32, USER_RAM25_F32, USER_RAM26_F32, USER_RAM27_F32, USER_RAM28_F32, USER_RAM29_F32, USER_RAM30_F32, USER_RAM31_F32, USER_RAM32_F32, USER_RAM33_F32, USER_RAM34_F32, USER_RAM35_F32, USER_RAM36_F32, USER_RAM37_F32, USER_RAM38_F32, USER_RAM39_F32	46000, 46002, 46004, 46006, 46008, 46010, 46012, 46014, 46016, 46018, 46020, 46022, 46024, 46026, 46028, 46030, 46032, 46034, 46036, 46038, 46040, 46042, 46044, 46046, 46048, 46050, 46052, 46054, 46056, 46058, 46060, 46062, 46064, 46066, 46068, 46070, 46072, 46074, 46076, 46078

USER_RAM#(0:9)_I32	46080	INT32	R/W

USER_RAM#(0:9)_I32
- Starting Address: 46080

Generic RAM registers. Useful for passing data between a host computer and a Lua script. Will not return an error if alternate data types are used.

- Data type: INT32 (type index = 2)
- Readable and writable
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0162

Expanded Names	Addresses
USER_RAM0_I32, USER_RAM1_I32, USER_RAM2_I32, USER_RAM3_I32, USER_RAM4_I32, USER_RAM5_I32, USER_RAM6_I32, USER_RAM7_I32, USER_RAM8_I32, USER_RAM9_I32	46080, 46082, 46084, 46086, 46088, 46090, 46092, 46094, 46096, 46098

USER_RAM#(0:39)_U32	46100	UINT32	R/W

USER_RAM#(0:39)_U32
- Starting Address: 46100

Generic RAM registers. Useful for passing data between a host computer and a Lua script. Will not return an error if alternate data types are used.

- Data type: UINT32 (type index = 1)
- Readable and writable
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0162

Expanded Names	Addresses
USER_RAM0_U32, USER_RAM1_U32,	46100, 46102,
USER_RAM2_U32, USER_RAM3_U32,	46104, 46106,
USER_RAM4_U32, USER_RAM5_U32,	46108, 46110,
USER_RAM6_U32, USER_RAM7_U32,	46112, 46114,
USER_RAM8_U32, USER_RAM9_U32,	46116, 46118,
USER_RAM10_U32, USER_RAM11_U32,	46120, 46122,
USER_RAM12_U32, USER_RAM13_U32,	46124, 46126,
USER_RAM14_U32, USER_RAM15_U32,	46128, 46130,
USER_RAM16_U32, USER_RAM17_U32,	46132, 46134,
USER_RAM18_U32, USER_RAM19_U32,	46136, 46138,
USER_RAM20_U32, USER_RAM21_U32,	46140, 46142,
USER_RAM22_U32, USER_RAM23_U32,	46144, 46146,
USER_RAM24_U32, USER_RAM25_U32,	46148, 46150,
USER_RAM26_U32, USER_RAM27_U32,	46152, 46154,
USER_RAM28_U32, USER_RAM29_U32,	46156, 46158,
USER_RAM30_U32, USER_RAM31_U32,	46160, 46162,
USER_RAM32_U32, USER_RAM33_U32,	46164, 46166,
USER_RAM34_U32, USER_RAM35_U32,	46168, 46170,
USER_RAM36_U32, USER_RAM37_U32,	46172, 46174,
USER_RAM38_U32, USER_RAM39_U32	46176, 46178

USER_RAM#(0:19)_U16	46180	UINT16	R/W

USER_RAM#(0:19)_U16
- Starting Address: 46180

Generic RAM registers. Useful for passing data between a host computer and a Lua script. Will not return an error if alternate data types are used.

- Data type: UINT16 (type index = 0)
- Readable and writable
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0162

Expanded Names	Addresses
USER_RAM0_U16, USER_RAM1_U16, USER_RAM2_U16, USER_RAM3_U16, USER_RAM4_U16, USER_RAM5_U16, USER_RAM6_U16, USER_RAM7_U16, USER_RAM8_U16, USER_RAM9_U16, USER_RAM10_U16, USER_RAM11_U16, USER_RAM12_U16, USER_RAM13_U16, USER_RAM14_U16, USER_RAM15_U16, USER_RAM16_U16, USER_RAM17_U16, USER_RAM18_U16, USER_RAM19_U16	46180, 46181, 46182, 46183, 46184, 46185, 46186, 46187, 46188, 46189, 46190, 46191, 46192, 46193, 46194, 46195, 46196, 46197, 46198, 46199

USER_RAM example script lua

```
 LJ.IntervalConfig(0, 1000)
previous = MB.readName("AIN0")
MB.writeName("USER_RAM0_U16", 1)

while true do
 if LJ.CheckInterval(0) then
   enable = MB.readName("USER_RAM0_U16") --Host may disable portion of the script
   if enable >= 1 then
     current = MB.readName("AIN0")
     twoValueAvg = (previous + current) / 2
     print("Rolling average of the last two measurements: ", twoValueAvg)
     MB.writeName("USER_RAM0_F32", twoValueAvg) --Provide a new value to host
     previous = current
   end
 end
end
```

For MB.writeName and MB.readName, the T7 requires firmware 1.0287 or later and the T4 requires firmware 1.0027 or later.

There is also a more advanced system for passing data to/from a Lua script referred to as FIFO buffers. These buffers are useful if you want to send an array of information in sequence to/from a Lua script. Usually 2 buffers are used for each endpoint, one buffer dedicated for each communication direction (read and write). For example, a host may write new data for the Lua script into FIFO0, then once the script reads the data out of that buffer, it responds by writing data into FIFO1, and then the host may read the data out of FIFO1.

Note that the following _DATA registers are buffered registers.

User RAM FIFO Registers - Advanced

Name	Start Address	Type	Access
USER_RAM_FIFO#(0:3)_DATA_U16	47000	UINT16	R/W

USER_RAM_FIFO#(0:3)_DATA_U16
- Starting Address: 47000

Generic FIFO buffer. Useful for passing ORDERED or SEQUENTIAL data between various endpoints, such as between a host and a Lua script. Use up to 4 FIFO buffers simultaneously->1 of each data type, all 4 different data types, or a mixture. e.g. FIFO0_DATA_U16 points to the same memory as other FIFO0 registers, such that there are a total of 4 memory blocks: FIFO0, FIFO1, FIFO2 and FIFO3. It is possible to write into a FIFO buffer using a different datatype than is being used to read out of it. This register is a buffer. Underrun behavior - throws an error.

- Data type: UINT16 (type index = 0)
- Readable and writable
- Default value: 0
- This register is a Buffer Register
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0163

Expanded Names	Addresses
USER_RAM_FIFO0_DATA_U16, USER_RAM_FIFO1_DATA_U16, USER_RAM_FIFO2_DATA_U16, USER_RAM_FIFO3_DATA_U16	47000, 47001, 47002, 47003

USER_RAM_FIFO#(0:3)_DATA_U32	47010	UINT32	R/W

USER_RAM_FIFO#(0:3)_DATA_U32
- Starting Address: 47010

Generic FIFO buffer. Useful for passing ORDERED or SEQUENTIAL data between various endpoints, such as between a host and a Lua script. Use up to 4 FIFO buffers simultaneously->1 of each data type, all 4 different data types, or a mixture. e.g. FIFO0_DATA_U16 points to the same memory as other FIFO0 registers, such that there are a total of 4 memory blocks: FIFO0, FIFO1, FIFO2 and FIFO3. It is possible to write into a FIFO buffer using a different datatype than is being used to read out of it. This register is a buffer. Underrun behavior - throws an error.

- Data type: UINT32 (type index = 1)
- Readable and writable
- Default value: 0
- This register is a Buffer Register
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0163

Expanded Names	Addresses
USER_RAM_FIFO0_DATA_U32, USER_RAM_FIFO1_DATA_U32, USER_RAM_FIFO2_DATA_U32, USER_RAM_FIFO3_DATA_U32	47010, 47012, 47014, 47016

USER_RAM_FIFO#(0:3)_DATA_I32	47020	INT32	R/W

USER_RAM_FIFO#(0:3)_DATA_I32
- Starting Address: 47020

Generic FIFO buffer. Useful for passing ORDERED or SEQUENTIAL data between various endpoints, such as between a host and a Lua script. Use up to 4 FIFO buffers simultaneously->1 of each data type, all 4 different data types, or a mixture. e.g. FIFO0_DATA_U16 points to the same memory as other FIFO0 registers, such that there are a total of 4 memory blocks: FIFO0, FIFO1, FIFO2 and FIFO3. It is possible to write into a FIFO buffer using a different datatype than is being used to read out of it. This register is a buffer. Underrun behavior - throws an error.

- Data type: INT32 (type index = 2)
- Readable and writable
- Default value: 0
- This register is a Buffer Register
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0163

Expanded Names	Addresses
USER_RAM_FIFO0_DATA_I32, USER_RAM_FIFO1_DATA_I32, USER_RAM_FIFO2_DATA_I32, USER_RAM_FIFO3_DATA_I32	47020, 47022, 47024, 47026

USER_RAM_FIFO#(0:3)_DATA_F32	47030	FLOAT32	R/W

USER_RAM_FIFO#(0:3)_DATA_F32
- Starting Address: 47030

Generic FIFO buffer. Useful for passing ORDERED or SEQUENTIAL data between various endpoints, such as between a host and a Lua script. Use up to 4 FIFO buffers simultaneously->1 of each data type, all 4 different data types, or a mixture. e.g. FIFO0_DATA_U16 points to the same memory as other FIFO0 registers, such that there are a total of 4 memory blocks: FIFO0, FIFO1, FIFO2 and FIFO3. It is possible to write into a FIFO buffer using a different datatype than is being used to read out of it. This register is a buffer. Underrun behavior - throws an error.

- Data type: FLOAT32 (type index = 3)
- Readable and writable
- Default value: 0
- This register is a Buffer Register
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0163

Expanded Names	Addresses
USER_RAM_FIFO0_DATA_F32, USER_RAM_FIFO1_DATA_F32, USER_RAM_FIFO2_DATA_F32, USER_RAM_FIFO3_DATA_F32	47030, 47032, 47034, 47036

USER_RAM_FIFO#(0:3)_ALLOCATE_NUM_BYTES	47900	UINT32	R/W

USER_RAM_FIFO#(0:3)_ALLOCATE_NUM_BYTES
- Starting Address: 47900

Allocate memory for a FIFO buffer. Number of bytes should be sufficient to store users max transfer array size. Note that FLOAT32, INT32, and UINT32 require 4 bytes per value, and UINT16 require 2 bytes per value. Maximum size is limited by available memory. Care should be taken to conserve enough memory for other operations such as AIN_EF, Lua, Stream etc.

- Data type: UINT32 (type index = 1)
- Readable and writable
- Default value: 0
- This register uses system RAM. The maximum RAM is 64KB. For more information, see 4.4 RAM
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0163

Expanded Names	Addresses
USER_RAM_FIFO0_ALLOCATE_NUM_BYTES, USER_RAM_FIFO1_ALLOCATE_NUM_BYTES, USER_RAM_FIFO2_ALLOCATE_NUM_BYTES, USER_RAM_FIFO3_ALLOCATE_NUM_BYTES	47900, 47902, 47904, 47906

USER_RAM_FIFO#(0:3)_NUM_BYTES_IN_FIFO	47910	UINT32	R

USER_RAM_FIFO#(0:3)_NUM_BYTES_IN_FIFO
- Starting Address: 47910

Poll this register to see when new data is available/ready. Each read of the FIFO buffer decreases this value, and each write to the FIFO buffer increases this value. At any point in time, the following equation holds: Nbytes = Nwritten - Nread.

- Data type: UINT32 (type index = 1)
- Read-only
- Default value: 0
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0163

Expanded Names	Addresses
USER_RAM_FIFO0_NUM_BYTES_IN_FIFO, USER_RAM_FIFO1_NUM_BYTES_IN_FIFO, USER_RAM_FIFO2_NUM_BYTES_IN_FIFO, USER_RAM_FIFO3_NUM_BYTES_IN_FIFO	47910, 47912, 47914, 47916

USER_RAM_FIFO#(0:3)_EMPTY	47930	UINT32	W

USER_RAM_FIFO#(0:3)_EMPTY
- Starting Address: 47930

Write any value to this register to efficiently empty, flush, or otherwise clear data from the FIFO.

- Data type: UINT32 (type index = 1)
- Write-only
- Default value: 0
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0163

Expanded Names	Addresses
USER_RAM_FIFO0_EMPTY, USER_RAM_FIFO1_EMPTY, USER_RAM_FIFO2_EMPTY, USER_RAM_FIFO3_EMPTY	47930, 47932, 47934, 47936

USER_RAM_FIFO example script lua

```
aF32_Out= {}  --array of 5 values(floats)
aF32_Out[1] = 10.0
aF32_Out[2] = 20.1
aF32_Out[3] = 30.2
aF32_Out[4] = 40.3
aF32_Out[5] = 50.4

aF32_In = {}
```

```
numValuesFIO0 = 5
ValueSizeInBytes = 4
numBytesAllocFIFO0 = numValuesFIO0*ValueSizeInBytes
MB.W(47900, 1, numBytesAllocFIFO0) --allocate USER_RAM_FIFO0_NUM_BYTES_IN_FIFO to 20 bytes

LJ.IntervalConfig(0, 2000)
while true do
 if LJ.CheckInterval(0) then
   --write out to the host with FIFO0
   for i=1, numValuesFIO0 do
     ValOutOfLua = aF32_Out[i]
     numBytesFIFO0 = MB.R(47910, 1)
     if (numBytesFIFO0 < numBytesAllocFIFO0) then
       MB.W(47030, 3, ValOutOfLua)  --provide a new array to host
       print ("Next Value FIFO0: ", ValOutOfLua)
     else
       print ("FIFO0 buffer is full.")
     end
   end
   --read in new data from the host with FIFO1
   --Note that an external computer must have previously written to FIFO1
   numBytesFIFO1 = MB.R(47912, 1) --USER_RAM_FIFO1_NUM_BYTES_IN_FIFO
   if (numBytesFIFO1 == 0) then
     print ("FIFO1 buffer is empty.")
   end
   for i=1, ((numBytesFIFO1+1)/ValueSizeInBytes) do
     ValIntoLua = MB.R(47032, 3)
     aF32_In[i] = ValIntoLua
     print ("Next Value FIFO1: ", ValIntoLua)
   end
 end
end
```

Working with 32-bit Values

On some T-Series devices, Lua's only numeric data type is IEEE 754 single precision (float). This means that working with 32-bit integer registers is difficult (see examples below). If any integer exceeds 24-bits (including sign), the lower bits will be lost. The workaround is to access the Modbus register using two numbers, each 16-bits. Lua can specify the data type for the register being written, so if you are expecting a large number that will not fit in a float (>24 bits), such as a MAC address, then read or write the value as a series of 16-bit integers.

If you expect the value to be counting up or down, use `MB.RA` or `MB.RW` to access the U32 as a contiguous set of 4 bytes.

If the value isn't going to increment (e.g. the MAC address) it is permissible to read it in two separate packets using `MB.R` .

Read a 32-bit register lua

```
 --If value is expected to be changing and is >24 bits: Use MB.RA
aU32[1] = 0x00
aU32[2] = 0x00
aU32, error = MB.RA(3000, 0, 2)   --DIO0_EF_READ_A. Type is 0 instead of 1
DIO0_EF_READ_A_MSW = aU32[1]
DIO0_EF_READ_A_LSW = aU32[2]

--If value is constant and is >16,777,216 (24 bits): Use MB.R twice
--Read ETHERNET_MAC (address 60020)
MAC_MSW = MB.R(60020, 0)  --Read upper 16 bits. Type is 0 instead of 1
MAC_LSW = MB.R(60021, 0)  --Read lower 16 bits.
```

```
--If value is <16,777,216 (24 bits): Use MB.R
--Read AIN0_EF_INDEX (address 9000)
AIN0_index = MB.R(9000, 1)  --Type can be 1, since the value will be smaller than 24 bits.
```

Write a 32-bit register lua

```
 --If value might be changed or incremented by the T7 and is >24 bits: Use MB.WA

aU32[2] = 0xFB5F
error = MB.WA(44300, 0, 2, aU32) --Write DIO0_EF_VALUE_A. Type is 0 instead of 1

--If value is constant and is >24 bits: Use MB.W twice
MB.W(44300, 0, 0xFF2A) --Write upper 16 bits. Type is 0 instead of 1
MB.W(44301, 0, 0xFB5F) --Write lower 16 bits.

--If value is <16,777,216 (24 bits): Use MB.W
--Write DIO0_EF_INDEX (address 44100)
MB.W(44100, 1, 7)  --Type can be 1, since the value(7) is smaller than 24 bits.
```

Is a register 32-bit?

To determine if a register is 32-bit, you can use the Modbus Map tool. If you need to determine programmatically, you can use MB.nameToAddress of the LabJack Library.

Moving a Lua Script from Flash to RAM

Before a Lua script is ran, it must be loaded to RAM. To move a script saved to flash into RAM, write any value to the UINT32 register LUA_LOAD_SAVED.

Anytime a Lua script is stopped in a host application (done by writing a 0 to LUA_RUN) the current Lua script is wiped from RAM. Users who wish to restart their Lua script in a host application must first write a value to LUA_LOAD_SAVED to load a script from flash or manually write a script to RAM using the LUA_SOURCE registers, then write a 1 to LUA_RUN to start the script.

Load Lua Script Manually To Device

While Kipling handles Lua scripting details easily and automatically, the example below shows how to load a Lua script to a T7 manually using the following registers:

Lua Source Registers

Name	Start Address	Type	Access
LUA_SOURCE_SIZE	6012	UINT32	R/W

LUA_SOURCE_SIZE
- Address: 6012

Allocates RAM for the source code.

- Data type: UINT32 (type index = 1)
- Readable and writable
- This register uses system RAM. The maximum RAM is 64KB. For more information, see 4.4 RAM
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0018

LUA_SOURCE_WRITE	6014	BYTE	W

LUA_SOURCE_WRITE
- Address: 6014

Write the source code here. Source will be saved to the RAM allocated by LUA_SOURCE_SIZE. This register is a buffer.

- Data type: BYTE (type index = 99)
- Write-only
- This register is a Buffer Register
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0018

LUA_SAVE_TO_FLASH	6032	UINT32	R/W

LUA_SAVE_TO_FLASH
- Address: 6032

Write 1 to save the loaded source code to flash.

- Data type: UINT32 (type index = 1)
- Readable and writable
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0059

LUA_LOAD_SAVED	6034	UINT32	W

LUA_LOAD_SAVED
- Address: 6034

Writing any value reads the script from Flash and loads it into RAM. The script can now be compiled and run.

- Data type: UINT32 (type index = 1)
- Write-only
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0059

The process is as follows:

1. Define or load a Lua Script and make sure a device has been opened.
2. Make sure there is a null-character at the end of the string.
3. Make sure a Lua Script is not currently running. If one is, stop it and wait for it to be stopped.
4. Write to the LUA_SOURCE_SIZE and LUA_SOURCE_WRITE registers to instruct the T7 to allocate space for a script and to transfer it to the device.
5. (Optional) Enable debugging.
6. Instruct the T-Series device to run the loaded Lua Script.
7. (Optional) write 1 to LUA_SAVE_TO_FLASH to save the script to flash memory. We only recommend doing this after testing the script operation.

The C example below opens a T7, shuts down any Lua script that may be running, loads the script, and runs the script.

C

```
const char * luaScript =
  "LJ.IntervalConfig(0, 500)\n"
  "while true do\n"
  "  if LJ.CheckInterval(0) then\n"
  "    print(LJ.Tick())\n"
  "  end\n"
  "end\n"
  "\0";

// strlen does not include the null-terminating character, so we add 1
// byte to include it.
const unsigned scriptLength = strlen(luaScript) + 1;

int handle = OpenOrDie(LJM_dtT7, LJM_ctANY, "LJM_idANY");

// Disable a running script by writing 0 to LUA_RUN twice
WriteNameOrDie(handle, "LUA_RUN", 0);

// Wait for the Lua VM to shut down (and some T7 firmware versions need
// a longer time to shut down than others):
MillisecondSleep(600);
WriteNameOrDie(handle, "LUA_RUN", 0);

// Write the size and the Lua Script to the device
WriteNameOrDie(handle, "LUA_SOURCE_SIZE", scriptLength);
WriteNameByteArrayOrDie(handle, "LUA_SOURCE_WRITE", scriptLength, luaScript);

// Start the script with debug output enabled
```

```
WriteNameOrDie(handle, "LUA_DEBUG_ENABLE", 1);
WriteNameOrDie(handle, "LUA_DEBUG_ENABLE_DEFAULT", 1);
WriteNameOrDie(handle, "LUA_RUN", 1);
```

The above example is valid C code, where the following functions are error-handling functions that cause the program to exit if an error occurs:

- OpenOrDie wraps LJM_Open
- WriteNameOrDie wraps LJM_eWriteName
- WriteNameByteArrayOrDie wraps LJM_eWriteNameByteArray

The Lua script in the example above is in the form of a C-string, e.g. a string with a null-terminator as the last byte.

To download a version of the above example, see utilities/lua_script_basic.c, which includes a function that reads debug data from the T7.

Reading Debug Data/Print Statements

After a script has started (with debugging enabled) any information printed by a lua script can be read by a user application using the "LUA_DEBUG_NUM_BYTES" and "LUA_DEBUG_DATA" registers. The following pseudocode roughly outlines how Lua debug data can be monitored:

C

```
 double debugBytes;
char debugData[1024];
int err;
int programRunning = 1;
LJM_eWriteName(handle, "LUA_DEBUG_ENABLE", 1); // Enable debugging
while (programRunning) {
    LJM_eReadName(handle, "LUA_DEBUG_NUM_BYTES", &debugBytes); // Check for debug data
    if ((int)debugBytes > 0) {
        LJM_eReadNameByteArray(handle, "LUA_DEBUG_DATA", (int)debugBytes, &debugData, &err);
        printf("%.*s\n", (int)debugBytes, debugData);
    }
}
```

Moving a Lua Script from Flash to RAM

Before a Lua script is ran, it must be loaded to RAM. To move a script saved to flash into RAM, write any value to the UINT32 register LUA_LOAD_SAVED.

Anytime a Lua script is stopped in a host application (done by writing a 0 to LUA_RUN) the current Lua script is wiped from RAM. Users who wish to restart their Lua script in a host application must first write a value to LUA_LOAD_SAVED to load a script from flash or manually write a script to RAM using the LUA_SOURCE registers, then write a 1 to LUA_RUN to start the script.

25.1 I2C Library [T-Series Datasheet]

T7 firmware minimum: 1.0225

The I2C library abstracts most of the Modbus calls needed to run I2C. The abstraction allows users to focus on I2C rather than Modbus, and reduces the memory requirements of scripts. Several I2C examples can be found on the I2C Sensor Examples page.

I2C.config

```
Error = I2C.config(SDA, SCL, Speed, Options, Address)
```

Sets parameters that are not normally changed. Values set by this function will remain unchanged until this function is called again or the equivalent Modbus registers are written to.

Parameters:

- `SDA` - DIO pin # that will be used as the I2C data line
- `SCL` - DIO pin # that will be used as the I2C clock line
- `Speed` - See I2C documentation
- `Options` - See I2C documentation
- `Address` - Left Justified

Returns:

- `Error` - standard LabJack T-Series error codes.

I2C.writeRead

```
RxData, Error = I2C.writeRead(TxData, NumToRead)
```

This function will first write the data in `TxData` to the preset address, then will read `NumToRead` bytes from that same address.

Parameters:

- `TxData` - This is a Lua table containing the values to be transmitted. The size of the table determines the number of bytes that will be transmitted.
- `NumToRead` - The number of data bytes to be read.

Returns:

- `RxData` - A Lua table of the values read
- `Error` - standard LabJack T-Series error codes.

I2C.read

```
RxData, Error = I2C.read(NumToRead)
```

This function will read `NumToRead` bytes from the preset address.

Parameters:

- `NumToRead` - The number of data bytes to be read.

Returns:

- `RxData` - A Lua table of the values read
- `Error` - standard LabJack T-Series error codes.

I2C.write

```
Error = I2C.write(TxData)
```

This function will write the data in `TxData` to the preset address.

Parameters:

- `TxData` - This is a Lua table containing the values to be transmitted. The size of the table determines the number of bytes that will be transmitted.

Returns:

- `Error` - standard LabJack T-Series error codes.

I2C.search

```
AddressList, Error = I2C.search(FirstAddress,
LastAddress)
```

This function will scan the I2C bus addresses are acknowledged. An acknowledged address means that at least one device is set to that address. Addresses are tested sequentially between the first and last address parameters.

Parameters:

- `FirstAddress` - The first address to be tested.
- `LastAddress` - The last address to be tested.

Returns:

- `AddressList` - A Lua table containing all the addresses that responded.
- `Error` - standard LabJack T-Series error codes.

I2C FAQ/Common Questions

Q: Why are no I2C ACK bits being received?

- Double check to make sure pull-up resistors are installed. A general rule for selecting the correct size pull-up resistors is to start with 4.7kΩ and adjust down to 1kΩ as necessary. If necessary, an oscilloscope should be used to ensure proper digital signals are present on the SDA and SCL lines.
- Double check to make sure the correct I/O lines are being used. It is preferred to do I2C communication on EIO/CIO/MIO lines instead of the FIO lines due to the larger series resistance (ESD protection) implemented on the FIO lines.
- Use an oscilloscope to verify the SDA and SCL lines are square waves and not weird arch signals (see "I2C_SPEED_THROTTLE" or use EIO/CIO/MIO lines).
- Use a logic analyzer (some oscilloscopes have this functionality) to verify the correct slave address is being used. See this EEVblog post on budget-friendly options. It is common to not take into account 7-bit vs 8-bit slave addresses or properly understand how LabJack handles the defined slave address and the read/write bits defined by the I2C protocol to perform read and write requests.
- Make sure your sensor is being properly powered. The VS lines of LJ devices are ~5V and the I/O lines are 3.3V. Sometimes this is a problem. Consider buying a LJTick-LVDigitalIO or powering the sensor with an I/O line or DAC channel.

Q: I've tried everything, still no I2C Ack Bits...

- Try slowing down the I2C bus using the "I2C_SPEED_THROTTLE" register/option. Reasons:
- Not all I2C sensors can communicate at the full speed of the LabJack. Check the I2C sensor datasheet.
- The digital signals could be getting corrupted due to the series resistors of the I/O lines on the LabJack.
- Consider finding a way to verify that your sensor is still functioning correctly using an Arduino and that it isn't broken.
- Try to establish communications with an LJTick-DAC to ensure the DIO are operating properly and that you are configuring I2C properly.

Q: Why is my device not being found by the I2C.search function?

- See information on I2C ACK bits above.

Q: What are I2C Read and Write functions or procedures?

- There are a few really good resources for learning about the general flow of I2C communication.
- TI's Understanding the I2C Bus by Jonathan Valdez and Jared Becker is a high quality resource
- I2C-bus specification by NXP
- Using the I2C Bus by Robot Electronics

- I2C Bus Specification by i2c.info

Q: Why am I getting a I2C_BUS_BUSY (LJM Error code 2720) error?

- See information on I2C ACK Bits above. Try different pull-up resistor sizes.

25.2 Bit Library

Lua in T-Series devices is based on Lua 5.1.4 which did not include a bitwise library. A subset of Lua 5.2's Bitwise Library have been added. The name of the library is `bin` or `bit` instead of bit32. To use the AND function use `bin.band` instead of `bit32.band` .

Limitations on T-Series

The T4 and T7 use 32-bit floating point (single precision) numbers. To perform bitwise operations the 32-bit float is converted into an integer. The bitwise operation is performed. Then the integer is converted back into a floating point number. When converting to an integer some information can be lost. Any decimal places will be truncated off and only up to 23-bits will make it to the integer. This is because 8 bits is used for the exponent and 1 bit for sign.

Functions

Operation of the below functions match the Lua 5.2 bitwise library with the exception of the data type limitations.

- `arshift`
- `band`
- `bnot`
- `bor`
- `bxor`
- `lshift`
- `rshift`

25.3 LabJack Library

The MB and LJ libraries allow access to the Modbus map and provide timing control. It is important to note that function calls dealing with 32-bit integers require special attention due to 32-bit floats being the only valid numerical type.

Modbus Name Functions

Required firmware versions:

- T7: 1.0281 and later
- T4: 1.0023 and later

Modbus Name Functions use register names like LJM's eReadName and eWriteName. This makes them easy to understand because registers are referenced by name, like DIO_STATE, instead of addresses, like 2800.

- Name functions are 0.1% to 20% slower than their address equivalents (see Modbus Address Functions below).
- Name functions use more memory than address functions.

MB.readName

```
value, error = MB.readName("register name
string")
```

Reads a single value from a Modbus register. Any errors encountered will be returned in "error." Some examples:

- `firmware_ver = MB.readName("FIRMWARE_VERSION")`
- `serial_num = MB.readName("SERIAL_NUMBER")`

To read multiple 32-bit values, see `MB.readNameArray` below.

MB.writeName

```
error = MB.writeName("register name string",
value)
```

Writes a single value to a Modbus register. Any errors encountered will be returned in error. Some examples:

- `MB.writeName("DIO_EF4_ENABLE", 0)`
- `MB.writeName("IO_CONFIG_SET_CURRENT_TO_FACTORY", 1)`

To write multiple 32-bit values, see `MB.writeNameArray` below.

MB.readNameArray

```
tbl_Values, error = MB.readNameArray("register_name", numValues, dataType_override)
```

Sequentially reads one or more values from Modbus registers. The first register read will be the address associated with the register "name". Subsequent registers will be incremented according to the register's data type.

Parameters:

- `register_name` - Name of the Modbus register to start reading from.
- `numValues` - Integer number of values to be read.
- `dataType_override` (optional) - When this parameter is not provided, the default data-type for the register will be used. When this parameter is provided, the passed data type will be used.

Returns:

- `tbl_Values` - Lua table containing the values read
- `error` (optional) - Returns error codes

Some examples:

- `tbl_Values = MB.readNameArray("AIN2", 3)` - This will read three floating point values (32-bits each) from AIN2, AIN3, and AIN4 (addresses 4, 6, and 8):
- `tbl_Values[1] = AIN2 Voltage`
- `tbl_Values[2] = AIN3 Voltage`
- `tbl_Values[3] = AIN4 Voltage`
- `tbl_Values = MB.readNameArray("ETHERNET_IP", 2, 0)` - This will read the Ethernet IP as two 16-bit unsigned integers. By default, ETHERNET_IP is a 32-bit unsigned integer, which will be truncated if put into in a 32-bit float. This is a limitation with 32-bit data types.
- Given IP = 192.168.1.100 = (0xC0A80164)
- `tbl_Values[1]` = 0xC0A8
- `tbl_Values[2]` = 0x0164
- If we were to read the IP without using the dataType_override the result would be: 0xC0A80100.

MB.writeNameArray

```
error = MB.writeNameArray("name", numValues, tbl_Values,
dataType_override)
```

Sequentially writes one or more values to Modbus registers. The first register written will be the address associated with the register "name". Subsequent registers will be incremented according to the register's data type.

Parameters:

- `register_name` - Name of the Modbus register to start writing to.
- `numValues` - Integer number of values to be written.
- `tbl_Values` - Lua table containing the values to be written.
- `dataType_override` (optional) - When this parameter is not provided, the default data-type for the register will be used. When this parameter is provided, the passed data type will be used.

Returns:

- `error` (optional) - Returns error codes

Some examples:

- `MB.writeNameArray("DAC0", 2, tbl_Values)` - This will write two floating point values (32-bits each) to DAC0 and DAC1 (addresses 1000, and 1002).
- DAC0 will be set to the value in `tbl_Values[1]`
- DAC1 will be set to the value in `tbl_Values[2]`
- `MB.writeNameArray("ETHERNET_IP_DEFAULT", 2, tbl_Values, 0)` - This will write the ETHERNET_IP_DEFAULT register as two 16-bit unsigned integers. By default, ETHERNET_IP_DEFAULT is a 32-bit unsigned integer, which will be truncated if put into in a 32-bit float. This is a limitation with 32-bit data types.
- Given IP = 192.168.1.100
- The upper 16-bits of the IP set to `tbl_Values[1]`
- The lower 16-bits of the IP set to `tbl_Values[2]`

MB.nameToAddress

`address, dataType, error = MB.nameToAddress("Name")`

Searches for a register with "Name" and returns the corresponding address and dataType. This is useful for improving a script's speed while retaining clarity. For example:

`fio5Address, fio5DataType = MB.nameToAddress("FIO5")`

Modbus Address Functions

The following Modbus address functions are similar to eReadAddress, eWriteAddress, eReadAddressArray, and eWriteAddressArray. The address and data type need to be provided. The address controls what you want to read or write. The data type controls how the data should be interpreted. Address and data types can be found in the LabJack Modbus Map and in the Kipling Register Matrix.

Data Types - Below is a list of data type indices. The data types match those used by the LJM Library.

- **0** - unsigned 16-bit integer
- **1** - unsigned 32-bit integer
- **2** - signed 32-bit integer
- **3** - single precision floating point (float)
- **98** - string
- **99** - byte - The "byte" dataType is used to pass arrays of bytes in what Lua calls tables

MB.R

```
value, error = MB.R(Address, dataType)
```

Modbus read. Reads a single value from a Modbus register. The type can be a u16, u32, a float, or a string. Any errors encountered will be returned in `error` .

MB.W

```
error = MB.W(Address, dataType, value)
```

Modbus write. Writes a single value to a Modbus register. The type can be a u16, u32, a float, or a string. Any errors encountered will be returned in `error` .

MB.WA

```
error = MB.WA(Address, dataType, nValues,
table)
```

Modbus write array. Reads integer `nValues` from the supplied table, interprets them according to the `dataType` and writes them as an array to the register specified by `Address` . The table must be indexed with numbers from 1 to `nValues` .

MB.RA

```
table, error = MB.RA(Address, dataType, nValues)
```

Modbus read array. Reads integer `nValues` of type `dataType` from `Address` and returns the results in a Lua table. The table is indexed from 1 to `nValues` .

Shortcut Functions

The following functions are shortcuts used to gain a small speed advantage over the equivalent Modbus functions.

LJ.ledtog

`LJ.ledtog()` --Toggles status LED. Note that reading AINs also toggles the status LED.

LJ.Tick

`Ticks = LJ.Tick()` --Reads the core timer (1/2 core frequency).

LJ.DIO_D_W

`LJ.DIO_D_W(3, 1)` --Quickly change DIO3 (FIO3) direction _D_ to 1 (output).

LJ.DIO_S_W

`LJ.DIO_S_W(3, 0)` --Quickly change the state _S_ of DIO3 (FIO3) to 0 (output low)

LJ.DIO_S_R

`state = LJ.DIO_S_R(3)` --Quickly read the state _S_ of DIO3 (FIO3)

LJ.CheckFileFlag

`flag = LJ.CheckFileFlag()` and `LJ.ClearFileFlag()`

`LJ.CheckFileFlag` and `LJ.ClearFileFlag` work together to provide an easy way to tell a Lua script to switch files. This is useful for applications that require continuous logging in a Lua script and on-demand file access from a host. Since files cannot be opened simultaneously by a Lua script and a host, the Lua script must first close the active file if the host wants to read file contents. The host writes a value of 1 to FILE_IO_LUA_SWITCH_FILE, and the Lua script is setup to poll this parameter using `LJ.CheckFileFlag` . If the file flag is set, Lua code should switch files:

lua

```
fg = LJ.CheckFileFlag() --poll the flag every few seconds
if fg == 1 then
 NumFn = NumFn + 1                --increment filename
 Filename = Filepre..string.format("%02d", NumFn)..Filesuf
 f:close()
 LJ.ClearFileFlag()            --inform host that previous file is available.
 f = io.open(Filename, "w")      --create or replace a new file
 print ("Command issued by host to create new file")
end
```

Timing Functions

LJ.IntervalConfig & LJ.CheckInterval

`LJ.IntervalConfig` and `LJ.CheckInterval` work together to make an easy-to-use timing function. Set the desired interval time with IntervalConfig, then use `LJ.CheckInterval` to watch for timeouts. The interval period will have some jitter but no overall error. Jitter is typically ±30 µs but can be greater depending on processor loading. A small amount of error is induced when the processor's core speed is changed.

Up to 8 different intervals can be active at a time.

`LJ.IntervalConfig(handle, time_ms)`

Sets an interval timer, starting from the current time.

`handle` : Integer value 0-7. Initializes (or reinitializes) an interval identified by this handle.

`time_ms` : 32-bit float which represents the number of milliseconds per interval. The minimum configurable interval time is 10us. The maximum value is 50000 ms.

`timed_out = LJ.CheckInterval(handle)`

`handle` : Integer value 0-7. Identifies an interval.

Returns: nil if no intervals have elapsed. Otherwise, returns the number of elapsed intervals. If this number is greater than one, then the script is too complex for the specified interval.

LJ Interval Function Example lua

```
LJ.IntervalConfig(0, 1000)
while true do
 if LJ.CheckInterval(0) then
   --Code to run once per second here.
 end
end
```

Note: Firmware versions prior to T7 1.0283 and T4 1.0024 had an issue where large quantities of missed intervals could cause a device watchdog timer to be triggered and rebooted.

Lua Performance Functions

LJ.setLuaThrottle

`LJ.setLuaThrottle(newThrottle)`

Set the throttle setting. This controls Lua's processor priority. Value is the integer number of Lua instruction to execute before releasing control to the normal polling loop. After the loop completes Lua will be given processor time again.

LJ.getLuaThrottle

`ThrottleSetting = LJ.getLuaThrottle()`

Reads the current integer throttle setting.

LJ.ResetOpCount

`LJ.ResetOpCount()` --Resets the lua operation counter, which determines when to release control from Lua to the normal device polling loop. This function should be used sparingly; `LJ.setLuaThrottle` should be considered for most situations.

25.4 Lua Script Performance

There are many ways to measure and improve the performance of a Lua Script running on a T-Series device. A set of benchmarking example scripts are available in the Lua Script Examples and in Kipling.

- The analog input benchmark script runs at approximately 12500 loop iterations per second.
- The DIO benchmark script runs at approximately 16000 loop iterations per second.

Test Methodology

The performance tests on this page were obtained by measuring the total time it took to run 100000 iterations of a given chunk of Lua code and then dividing that total time by 100,000 to convert to microseconds (µs per iteration). Test results were obtained using a T4 running firmware version 1.0023, default Lua throttling settings, and default processor speed. A T7-Pro running firmware version 1.0282 produced similar results. Execution times can vary by up to 10% with different firmware versions and device types.

Using `local` to Improve Lookup Time

Define Variables as `local`

When performing operations in a scripted language like Lua, it takes time to look up variables depending on their scope. To speed up a script, define variables as `local` . For example, the average iteration time for the following script was 17.2 µs:

lua

```
x = 0
for i = 1, 100000 do
  x = x + 1
end
```

The same script with variable `x` defined as a local variable took approximately 8.5 µs per iteration:

lua

```
local x = 0
for i = 1, 100000 do
 x = x + 1
end
```

Create `local` References to Functions

It is also faster to use `local` references to functions. The benchmarking example scripts all define `local` references to the functions from the Lua LabJack Library, which provides the

global `MB` and `LJ` objects. For example, the average iteration time for the following script was 23.7 µs:

lua

```
for i = 1, 100000 do
 LJ.DIO_S_W(5,1)
end
```

The same script with a `local` reference to `LJ.DIO_S_W` as `dioStateWrite` took approximately 18.1 µs per iteration.

lua

```
local dioStateWrite = LJ.DIO_S_W
for i = 1, 100000 do
 dioStateWrite(5,1)
end
```

Adjusting `LJ.setLuaThrottle`

After a script is successfully downloaded to a device and no RAM errors are encountered, the T-Series device uses its internal Lua compiler to translate (precompile) the source code into a set of instructions similar to machine code but intended for the Lua interpreter. When it is time for the code to be executed, the T-Series device steps through one or more instructions at a time as defined by the `LJ.setLuaThrottle` function. In practical terms, this setting adjusts the number of Lua machine instruction codes that are executed per service interval. This number can be increased at the expense of potentially preventing other device features from executing as quickly, such as processing packets from a host computer.

The following script took approximately 8.4 µs per iteration when tested:

lua

```
LJ.setLuaThrottle(10)
local x = 1
for i = 1, 100000 do
 x = x + 1
end
```

The same loop took approximately 10.8 µs per iteration when the LabJack was configured to execute 100 instructions per service interval:

lua

```
LJ.setLuaThrottle(100)
local x = 1
for i = 1, 100000 do
 x = x + 1
end
```

Benchmark Comparison

There are three ways to toggle a digital I/O line using the LabJack Lua Library functions:

- `MB.writeName` (a Modbus Name Function)
- `MB.W` (a Modbus Address Function)
- `LJ.DIO_S_W` (a Shortcut Function)

Each of the three methods are shown with benchmark results below. Note that these benchmark tests use local function references.

`MB.writeName` - Modbus Name Function

Approximately 24 µs per iteration.

lua

```
LJ.setLuaThrottle(1000)
local modbus_write_name = MB.writeName
for i = 1, 100000 do
  modbus_write_name("FIO5", 1)
end
```

`MB.W` - Modbus Address Function

Approximately 17 µs per iteration.

lua

```
LJ.setLuaThrottle(1000)
local mbW = MB.W
for i = 1, 100000 do
  mbW(2005, 0, 1)
end
```

`LJ.DIO_S_W` - Shortcut Function

Approximately 11 µs per iteration.

lua

```
LJ.setLuaThrottle(1000)
local dioStateWrite = LJ.DIO_S_W
for i = 1, 100000 do
  dioStateWrite(5,1)
end
```

Comparison with External Communications

Since T-Series devices implement a Modbus Server that can be controlled directly by Lua or by a host computer, the performance of these functions can be useful to compare to round trip command-response times issued by host computers through the USB, Ethernet, or WiFi communication interfaces.

Further Reading

- Lua Performance Tips: chapter 2 by Roberto Lerusalimschy.

25.5 Lua Warnings

Lua will provide warnings to help you avoid common problems. Below is a list of warning and the recommended solutions.

Truncation: Possible truncation error.

A script tried to access a 32-bit integer. Depending on the value transferred, some precision may have been lost. See the floating point section in Section 25.0 for more details. To access a 32-bit value, use an array function of the LabJack Lua library to access it as two 16-bit numbers.

Example

Read the Ethernet IP as a single 32-bit number lua

```
Ethernet_IP = MB.readName("ETHERNET_IP")
print("Read IP as a 32-bit integer: ", string.format("%08X", Ethernet_IP))
```

Read the Ethernet IP as two 16-bit numbers lua

```
Eth_IP = {}
Eth_IP = MB.readNameArray("ETHERNET_IP", 2, 0)
print("Read IP as two 16-bit integers:", string.format("%04X", Eth_IP[1]), string.format("%04X", Eth_IP[2]))
```

Output:

- Read IP as a 32-bit Integer: `0xC0A80100`
- Read IP as two 16-bit Integers: `0xC0A8 0x016A`

When the IP was read as a 32-bit integer the `0x16A` was lost.

Suppressing Possible truncation error.

This warning can be suppressed by writing 1 to LUA_NO_WARN_TRUNCATION:

Name	Start Address	Type	Access
LUA_NO_WARN_TRUNCATION	6050	UINT16	R/W

LUA_NO_WARN_TRUNCATION
- Address: 6050

Writing a 1 to this register will prevent truncation warnings from being displayed. This register will be cleared every time Lua is started.

- Data type: UINT16 (type index = 0)
- Readable and writable
- T8:
  - Minimum firmware version: 0.0123
- T7:
  - Minimum firmware version: 1.0285
- T4:
  - Minimum firmware version: 1.0028

Suppress Truncation Warning Example lua

```
error = MB.writeName("LUA_NO_WARN_TRUNCATION", 1)
```

25.6 Controlling Lua Execution With a Host Application

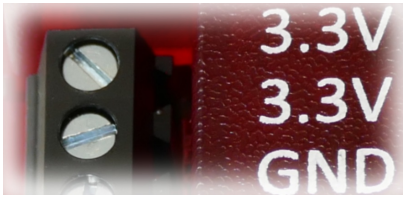
Host applications can easily be used to control multiple lua execution blocks through the use of USER_RAM registers and simple "if, else" statements in lua. Examples in multiple languages can be found below.

C -- see lua_execution_control.c under more\lua in the C/C++ for LJM examples package

Python -- see lua_execution_control.py under Examples\More\Lua in the Python for LJM package

LabVIEW -- see lua_exection_control.vi under Examples\More\Misc in the LabVIEW for LJM package

26.0 3.3V Supply (T8 Only) [T-Series Datasheet]



Two 3.3V terminals on the upper left corner of the LabJack T8

The T8 has 2 fixed 3.3V source terminals that are useful for supplying bridge circuits and digital sensors. For detailed specifications, see Appendix A-5.

Appendix A - Specifications [T-Series Datasheet]

Specifications for describing the T-Series devices can be broken down into several primary sections with a few sub-sections. Navigate the following sections to see specifications.

Subsections

A-1 Data Rates [T-Series Datasheet]

Communication Modes

Communication between the host computer and a T-series device occurs using one of two modes:

1. Command-response

Command-response mode is appropriate for most applications. In command-response mode, the host sends a command data packet, to which the T-series device sends a response data packet.

2. Stream

Stream mode is when the device collects periodic sampling events automatically. Collected data is stored in the device's memory until it retrieved by the host application. The LJM library stream functions simplify data collection from T-series devices. Not all functionality is supported in stream mode.

For more information about command-response and stream, see 3.0 Communication.

Note: These specs are generated using a LabVIEW program that reads data from a device in a simple while(1) loop. Additionally, we used a PC running Windows with a fairly average Intel CPU. We have found the performance of LabVIEW to be very similar to C, C++, and other compiled languages and have therefore chosen LabVIEW to collect these data rates. If an application requires precise timing of CR packets we suggest doing additional research and replicating these results for the system being used.

Figure A1.1depicts the two operating modes.

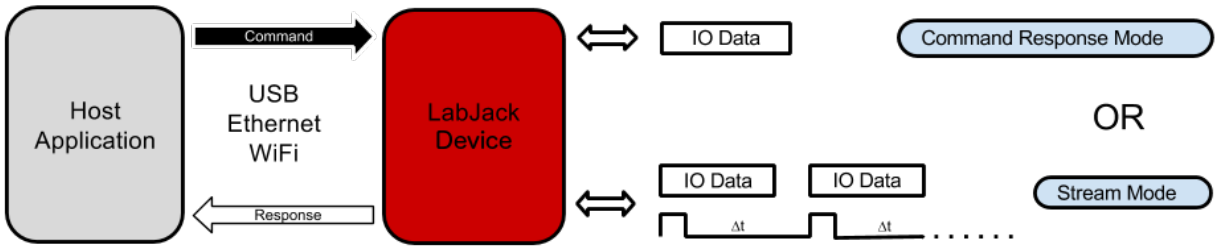


Figure A1.1 Communications Modes

The use of a particular mode will depend on functionality and the hardware response time required by the end application.

Command-Response Data Rates

All communication performed with T-series devices is accomplished using the Modbus TCP protocol, thus allowing direct communication with the device via low-level TCP commands. As

an alternative, the LJM library may be used as a higher level communications layer for added convenience and minimal additional overhead. Tables A.1.1 to A.1.3 list expected communication overhead times associated with Modbus TCP and LJM Library communication options. Tables A.1.1 and A.1.2 times are the same on the T4/T7, and were measured on a T7. Communication times for the T8 can be found in Table A.1.3.

Table A.1.1. Typical communication overhead using direct Modbus TCP with the T4/T7.

	USB (High-High)	USB (Other)	Ethernet	WiFi
	[ms]	[ms]	[ms]	[ms]
No I/O Overhead	0.6	2.1	1.0	6.5
Read All DI	0.7	2.2	1.1	6.6
Write All DO	0.7	2.2	1.1	6.6
Write Both DACs	0.7	2.2	1.1	6.6

Table A.1.2. Typical communication overhead using the LJM Library with the T4/T7.

	USB (High-High)	USB (Other)	Ethernet	WiFi
	[ms]	[ms]	[ms]	[ms]
No I/O Overhead	0.6	2.2	1.1	6.7
Read All DI	0.7	2.3	1.2	6.8
Write All DO	0.7	2.3	1.2	6.8
Write Both DACs	0.7	2.3	1.2	6.8

Table A.1.3. Typical communication overhead using the LJM Library with the T8.

	USB (High-High)	USB (Other)	Ethernet
	[ms]	[ms]	[ms]
Read All DI	0.18	0.18	0.265
Write All DO	0.19	0.19	0.285
Write Both DACs	0.19	0.19	0.295

The times shown in tables A.1.2 and A.1.3 were measured using a LabVIEW program running on Windows where all read and write operations are conducted with a single LJM_eNames() call. The LJM_eNames() functions is used to minimize the number of Modbus packets sent from the host (one packet per command/response set). The test program executes one of the listed tasks within a loop for a specified number of iterations, over a 1-10 second period. The overall execution time is divided by the total number of iterations, providing the average time

per iteration for each task. The execution time includes LabVIEW overhead, LJM library overhead, Windows overhead, communication time (USB/Ethernet/WiFi), and device processing time.

A "USB high-high" configuration means the T-series device is connected to a high-speed USB2 hub which is then connected to a high-speed USB2 host. Even though the Tx is not a high-speed USB device, such a configuration does often provide improved performance. Typical examples of "USB other" would be a Tx connected to an old full-speed hub (hard to find) or more likely the Tx is connected directly to the USB host (your PC) even if the host supports high-speed.

For more information, see this app note: High Speed Command-Response Sample Rates

Preemptive Operating Systems and Thread Priority

It is important to understand that Linux, Mac OS X, and Windows are generally "best-effort" operating systems and not "real-time", meaning that the listed CR speeds can vary based on each individual computer, the hardware inside of it, its currently enabled peripherals, current network traffic, strength of signal, design of the application software, other running software, and many more variables [1].

USB and Ethernet

These times are quite predictable. Software issues mentioned above are important—but, in terms of hardware, the times will be consistent. The device communication does not consume a major portion of total USB or Ethernet bandwidth. Therefore, the overhead times listed are typically maintained even with substantial activity on the bus.

WiFi - T7-Pro

WiFi latency tends to vary more than USB or Ethernet latency. With a solid connection, most WiFi packets have an overhead of 3 to 8 ms, but many will take longer. For example, a test was done in a typical office environment of 1000 iterations that produced an average time of 7.0 ms. The results were:

- 92% of the packets took 3-8 ms,
- 99% took < 30 ms,
- and 3 packets took 300 ms.

All WiFi tests were done with an RSSI between -40 (very strong) and -70 (good). An RSSI less than -75 generally reflects a weak connection, causing a greater number of packets retries. An RSSI greater than -35 reflects a very strong connection, typically within a few feet of the access point. This also results in a greater numbers of retries due to saturation of the RF signal.

ADC Conversions

Analog to digital conversions (ADC) will increase the command-response time depending on the number of channels, the input gain (T7), and the resolution index being used. The tables linked in the sections below list conversion times for various settings when reading a single analog input channel. The total command-response time (CRT) when reading analog inputs is equal to the overhead time from tables A.1.1, A.1.2 and A.1.3 added to the conversion times for the requested channels:

```
CRT (milliseconds) = overhead + (#AINs * AIN Sample
Time)
```

T4 AIN Sample Time

See the tables in Section A-3-1-2: T4 Noise and Resolution.

T7 AIN Sample Time

See the tables in Section A-3-2-2: T7 Noise and Resolution.

T8 AIN Sample Time

See the tables in Section A-3-3-2: T8 Noise and Resolution.

Streaming Data Rates

The fastest data rates on T-series devices occur when operating in stream mode. Much of the command-response overhead is eliminated in stream mode because the device is responsible for initiating IO operations. The device collects scans in its stream buffer, then the host application retrieves multiple scans at once. The end result is a continuous data stream, sampled at regular intervals, collected with a minimum number of communication packets [2.].

There is an important distinction between *scans* and *samples*. Definitions are as follows:

- Address: Also called a *channel*. An address usually returns the value of 1 input connection.
- Sample: A reading from one address.
- Scan: One reading from every address in the scan list.
- Scan list: The list of one or more addresses in a scan.

The scan rate is the rate at which scans are collected. It is a fraction of the sample rate, where the fraction is the inverse of the number of channels being read in a single scan. The scan rate is defined as:

```
ScanRate = SampleRate /
NumAddresses
```

The sample rate and scan rate are equal when the `NumAddresses` is 1.

T4 Stream Rates

The T4 has a **typical maximum sample rate** of 50 ksamples/second. This maximum is reflected in the first row of data in the following table (highlighted). The scan rates reported are the maximum sample rates divided by the number of channels in the scan list (within ~10%).

The scan rates in the following tables are continuous over USB or Ethernet.

The scan rate is defined as (see "Streaming Data Rates" above):

```
ScanRate = SampleRate /
NumAddresses
```

Table A.1.5. T4 Stream: Scan rates for different values of resolution index. Applies to USB and Ethernet. Applies to all streamable addresses including low-voltage and high-voltage analog inputs.

Resolution Index	1 Channel Max Scan Rate [Hz]	2 Channels Max Scan Rate [Hz]	4 Channels Max Scan Rate [Hz]	8 Channels Max Scan Rate [Hz]
1*	50k	25k	12.5k	6.25k
2	15k	7.5k	3.75k	1.875k
3	8k	4k	2k	1k
4	4k	2k	1k	500
5	2k	1k	500	250

* Default stream Resolution index for the T4.

Table A.1.6. T4 Stream: Typical noise and interchannel delay values depending on resolution index.

Resolution Index	Peak-to-Peak Noise [12-bit counts]	Interchannel Delay [µs]
1*	±4	40/13**
2	±3	47
3	±2.5	121
4	±2	230
5	±1.5	446

* Default stream Resolution index for the T4.

** 40 µs for sample rate <= 20k. 13 µs for sample rate > 20k.

T7 Stream Rates

- **Ethernet** can usually maintain just under 120 ksamples/second.
- **USB** generally maxes out right around 100 ksamples/second.
- When using **WiFi**, the device can acquire data at the fastest rates, but transfer of data to the host is limited to about 1 ksamples/second, so the fastest stream rates cannot be maintained continuously. In this case, stream-burst can be used rather than continuous stream, where each stream is limited to a specified number of scans that fits in the device's stream buffer. For high-speed wireless streaming, use the Ethernet connection with an external Ethernet-WiFi bridge.

The T7 has a **typical maximum sample rate** of 100 ksamples/second. This maximum sample rate is achievable when a stream is configured with `RANGE` = ±10V and `RESOLUTION_INDEX` = 0 or 1 [3.]. This maximum is reflected in the first row of data in the following table (highlighted). The scan rates reported are the maximum sample rates divided by the number of channels in the scan list (within ~10%).

The scan rates in the following tables are continuous over USB or Ethernet.

The scan rate is defined as (see "Streaming Data Rates" above):

```
ScanRate = SampleRate /
NumAddresses
```

Table A.1.7. T7 Stream: Scan rates over various gain, resolution index, channel count combinations. Applies to USB and Ethernet.

Resolution Index = 1	Gain : Range	1 Channel Max Scan Rate [Hz]	2 Channels Max Scan Rate [Hz]	4 Channels Max Scan Rate [Hz]	8 Channels Max Scan Rate [Hz]
	1 : ±10V	100k	50k	25k	12.5k
	10 : ±1V	100k	4.1k	1.4k	585
	100 : ±0.1V	100k	850	315	120
	1000 : ±0.01V	100k	N.S.	N.S.	N.S.
Resolution Index = 2	Gain : Range	1 Channel Max Scan Rate [Hz]	2 Channels Max Scan Rate [Hz]	4 Channels Max Scan Rate [Hz]	8 Channels Max Scan Rate [Hz]
	1 : ±10V	48k	19.8k	9.0k	4.0k
	10 : ±1V	48k	3.6k	1.3k	550

	100 : ±0.1V	48k	400	N.S.	N.S.
	1000 : ±0.01V	48k	N.S.	N.S.	N.S.
Resolution Index = 3	Gain : Range	1 Channel Max Scan Rate [Hz]	2 Channels Max Scan Rate [Hz]	4 Channels Max Scan Rate [Hz]	8 Channels Max Scan Rate [Hz]
	1 : ±10V	22k	9.9k	4.5k	2.4k
	10 : ±1V	22k	1.4k	500	225
	100 : ±0.1V	22k	N.S.	N.S.	N.S.
	1000 : ±0.01V	22k	N.S.	N.S.	N.S.
Resolution Index = 4	Gain : Range	1 Channel Max Scan Rate [Hz]	2 Channels Max Scan Rate [Hz]	4 Channels Max Scan Rate [Hz]	8 Channels Max Scan Rate [Hz]
	1 : ±10V	11k	4.9k	2.2k	1.3k
	10 : ±1V	11k	1.3k	45	N.S.
	100 : ±0.1V	11k	N.S.	N.S.	N.S.
	1000 : ±0.01V	11k	N.S.	N.S.	N.S.
Resolution Index = 5	Gain : Range	1 Channel Max Scan Rate [Hz]	2 Channels Max Scan Rate [Hz]	4 Channels Max Scan Rate [Hz]	8 Channels Max Scan Rate [Hz]
	1 : ±10V	5500	2.2k	990	630
	10 : ±1V	5500	630	23	N.S.
	100 : ±0.1V	5500	N.S.	N.S.	N.S.
	1000 : ±0.01V	5500	N.S.	N.S.	N.S.
Resolution Index = 6	Gain : Range	1 Channel Max Scan Rate [Hz]	2 Channels Max Scan Rate [Hz]	4 Channels Max Scan Rate [Hz]	8 Channels Max Scan Rate [Hz]

	1 : ±10V	2500	1.3k	630	315
	10 : ±1V	2500	320	N.S.	N.S.
	100 : ±0.1V	2500	N.S.	N.S.	N.S.
	1000 : ±0.01V	2500	N.S.	N.S.	N.S.
Resolution Index = 7	Gain : Range	1 Channel Max Scan Rate [Hz]	2 Channels Max Scan Rate [Hz]	4 Channels Max Scan Rate [Hz]	8 Channels Max Scan Rate [Hz]
	1 : ±10V	1200	650	315	N.S.
	10 : ±1V	1200	220	N.S.	N.S.
	100 : ±0.1V	1200	N.S.	N.S.	N.S.
	1000 : ±0.01V	1200	N.S.	N.S.	N.S.
Resolution Index = 8	Gain : Range	1 Channel Max Scan Rate [Hz]	2 Channels Max Scan Rate [Hz]	4 Channels Max Scan Rate [Hz]	8 Channels Max Scan Rate [Hz]
	1 : ±10V	600	315	N.S.	N.S.
	10 : ±1V	600	200	N.S.	N.S.
	100 : ±0.1V	600	N.S.	N.S.	N.S.
	1000 : ±0.01V	600	N.S.	N.S.	N.S.

N.S. indicates settings not supported in stream mode.

* When streaming one channel, the maximum sample rate is not affected by range because the signal chain does not need to settle.

** The default stream resolution index for the T7.

The maximum scan rate decreases at higher resolution index and range settings simply because analog conversions take longer to complete. Single-channel stream is a slight exception—its maximum scan rate decreases at higher resolution indices but not according to range settings because it does not need to settle.

Table A.1.8. T7 Stream: Typical noise and interchannel delay values depending on range and resolution index.

Gain/Range: 1/±10V	Resolution Index	Peak-to-Peak Noise [16-bit counts]	Interchannel Delay [µs]
	1*	±3.0	15/8**
	2	±2.0	25
	3	±1.5	45
	4	±1.0	90
	5	±1.0	170
	6	±0.5	335
	7	±0.5	670
	8	±0.5	1,335
Gain/Range: 10/±1V	Resolution Index	Peak-to-Peak Noise [16-bit counts]	Interchannel Delay [µs]
	1*	±4.5	210
	2	±3.0	220
	3	±2.0	545
	4	±1.5	585
	5	±1.0	1,200
	6	±0.5	2,415
	7	±0.5	2,750
	8	±0.5	3,415
Gain/Range: 100/±0.1V	Resolution Index	Peak-to-Peak Noise [16-bit counts]	Interchannel Delay [µs]
	1*	±12.0	1,040
	2	±9.0	2,105
	3	**N.S.**	**N.S.**
	4	**N.S.**	**N.S.**
	5	**N.S.**	**N.S.**
	6	**N.S.**	**N.S.**
	7	**N.S.**	**N.S.**
	8	**N.S.**	**N.S.**

Gain/Range: 1000/±0.01V	Resolution Index	Peak-to-Peak Noise [16-bit counts]	Interchannel Delay [µs]
	1*	**N.S.**	**N.S.**
	2	**N.S.**	**N.S.**
	3	**N.S.**	**N.S.**
	4	**N.S.**	**N.S.**
	5	**N.S.**	**N.S.**
	6	**N.S.**	**N.S.**
	7	**N.S.**	**N.S.**
	8	**N.S.**	**N.S.**

N.S. (Not Supported) indicates settings not supported in stream mode.
* The default stream resolution index for the T7.
** 15 µs for sample rate <= 60k. 8 µs for sample rate > 60k.

T8 Stream Rates

The T8 has a **maximum scan rate** of 40 ksamples/second. Unlike other T-series devices, the scan rate does not depend on how many addresses are sampled per scan since all analog inputs are sampled simultaneously.

The scan rates in the following table are continuous over USB or Ethernet. For reliable streaming, refer to the T8's Maximum Stream Speed section in the Stream Mode documentation.

Table A.1.9. T8 Stream: Scan rates for different values of resolution index. Applies to USB and Ethernet, and all streamable addresses.

Resolution Index	Min Scan Rate [kHz]	Max Scan Rate [kHz]
1	10.0	40.0
2	10.0	40.0
3	10.0	35.5
4	10.0	34.0
5	10.0	32.0
6	10.0	21.3
7	10.0	20.4
8	6.0	16.0
9	5.5	10.6

Resolution Index	Min Scan Rate [kHz]	Max Scan Rate [kHz]
10	5.0	10.2
11	3.5	8.0
12	3.0	5.3
13	2.5	5.1
14	1.5	4.0
15	0.1	2.0
16	0.1	1.0

The maximum scan rate decreases at higher resolution index simply because analog conversions take longer to complete.

Typical noise values depend on range and resolution index. Command-response noise is the same as stream noise. The T8 does not have interchannel delay. Refer to Section A-3-3-2: T8 Noise and Resolution for details.

Interchannel Delay - T4/T7

Interchannel delay is the time between each sample within a scan. For example, say 3 channels are streamed at 1000 scans/second with ResolutionIndex=1 and Range=10. That is a sample rate of 3000 samples/second, so from the table above the interchannel delay is 15 µs. The stream interrupt will fire every 1000 µs, at which time it takes about 5 µs until the 1st channel is sampled, then 15 µs later the 2nd channel is sampled, then 15 µs later the 3rd channel is sampled, and then about 965 µs later the next scan starts.

What if in the above example we wanted the 8 µs delay rather than 15 µs? The sample rate must be greater than 60 ksamples/second for that, so the solution is increase sample rate by scanning more channels (channels can be repeated in the scan list) or scanning faster and discarding the extra data.

The interchannel delay is a fixed time with little jitter, so the known time can be accounted for in user software to adjust phase if those microseconds are important. As an alternative to using the table above, the user can measure interchannel delay on their device by using a scope to look at the SPC timing output described in the Stream Mode section.

Notes

1. Various software issues need consideration when implementing a feedback loop that executes at the desired time interval. Some considerations are: thread priority, logging to file, updating the screen, and other programs running on the machine.

2. The number of packets used to retrieve stream data depends on the number of data points allowed to accumulate in the stream buffer.

3. Setting the resolution index to 0 (default) in stream mode is equivalent to a resolution equal to 1. The default resolution index in stream mode behaves different than command-response mode.

A-2 Digital I/O [T-Series Datasheet]

General Info

Table A2-1. IO Information

Parameter	Conditions	Min	Typical	Max	Units
Low Level Input Voltage	-	-0.3	-	0.5	Volts
High Level Input Voltage	-	2.64	-	5.8	Volts
Hysteresis Voltage [1]	-	-	-	-	-
---Low to High Transition	-	-	-	1.15	Volts
---High to Low Transition	-	0.90	-	-	Volts
Maximum Input Voltage [2]	-	-	-	-	-
---FIO (T4/T7) [5]	-	-10	-	10	Volts
---FIO (T8) [5]	-	-6	-	6	Volts
---EIO/CIO/MIO (T4/T7/T8) [5]	-	-6	-	6	Volts
Output Low Voltage [3][4]	No Load	-	0.01	-	Volts
---FIO (T4/T7) [5]	Sinking 1 mA	-	0.55	-	Volts
---FIO (T8) [5]	Sinking 1 mA	-	0.15	-	Volts
---EIO/CIO/MIO (T4/T7/T8) [5]	Sinking 1 mA	-	0.15	-	Volts
---EIO/CIO/MIO (T4/T7/T8) [5]	Sinking 5 mA	-	0.75	-	Volts
Output High Voltage [3][4]	No Load	-	3.3	-	Volts
---FIO (T4/T7) [5]	Sourcing 1 mA	-	2.75	-	Volts
---FIO (T8) [5]	Sourcing 1 mA	-	3.15	-	Volts
---EIO/CIO/MIO (T4/T7/T8) [5]	Sourcing 1 mA	-	3.15	-	Volts
---EIO/CIO/MIO (T4/T7/T8) [5]	Sourcing 5 mA	-	2.6	-	Volts
Short Circuit Current [3][4]	-	-	-	-	-
---FIO (T4/T7) [5]	-	-	6.3	-	mA

Parameter	Conditions	Min	Typical	Max	Units
---FIO (T8) [5]	-	-	22.9	-	mA
---EIO/CIO/MIO (T4/T7/T8) [5]	-	-	22.9	-	mA
Output Impedance [3][4]	-	-	-	-	-
---FIO (T4/T7) [5]	-	-	550	-	Ω
---FIO (T8) [5]	-	-	180	-	Ω
---EIO/CIO/MIO (T4/T7/T8) [5]	-	-	180	-	Ω

[1] The "Low Level" and "High Level" input voltage specify input voltage ranges guaranteed to produce the correct logic state at any digital input. The "Hysteresis Voltage" represents the voltage where a logic transition will actually occur. Input hysteresis will result in different transition voltages, depending on transition from HIGH to LOW or LOW to HIGH logic states. The overall hysteresis band is ~0.25 volts.

[2] Maximum voltage to avoid damage to the device. Protection works whether the device is powered or not, but continuous voltages over 5.8 volts or less than -0.3 volts are not recommend when the device is unpowered, as the voltage will attempt to supply operating power to the device possibly causing poor start-up behavior.

[3] These specifications provide the answer to the question. "How much current can the digital I/O sink or source?". For instance, if EIO0 is configured as output-high and shorted to ground, the current sourced by EIO0 is configured as output-high and shorted to ground, the current sourced by EIO0 into ground will be about 16 mA (3.3/180). If connected to a load that draws 5 mA, EIO0 can provide that current but the voltage will droop to about 2.4 volts instead of the nominal 3.3 volts. If connected to a 180 ohm load to ground, the resulting voltage and current will be about 1.65 volts @ 9 mA.

[4] It is recommended to use the EIO/CIO digital I/O lines for UART, SPI, I²C, 1-Wire, and other digital communication protocols.

[5] The T4 and T7 have additional protection on their FIO lines. The T8 has the same protection on all digital IO lines.

Extended Features

Table A2-2. DIO extended features information

Extended Features	Conditions	Min	Typical	Max	Units
Frequency Output [1]	-	0.02	-	5 M	Hz

Extended Features	Conditions	Min	Typical	Max	Units
Counter Input Frequency [2]	-	-	-	5	MHz
Minimum High & Low Time [2]	-	-	-	50	ns
"Interrupt" Total Edge Rate [3][4]	T7/T4, No Stream	-	-	70k	edges/s
	T7 Streaming @ 50 kHz	-	-	20k	edges/s
	T4 Streaming @ 20 kHz	-	-	20k	edges/s
	T8, No Stream			100k	edges/s
	T8, Streaming @ 20 kHz			40k	edges/s

[1] Frequencies up to 40MHz are possible, but they are heavily filtered.

[2] Hardware counters. 0 to 3.3 volt square wave.

[3] This is for the "Interrupt" modes. To avoid missing edges, keep the total number of applicable edges on all applicable timers below this limit.

[4] Excessive processor loading could reduce these limits further.

Serial Communication

T-series serial communication abilities information is below. T-series devices use 3.3V logic levels and provide 5V output along the VS screw terminal. Some ICs require the same logic level as provided to the chip's VCC line so extra steps may be required to integrate specific sensors.

Table A2-3. Serial communication information

Serial Communication	Device	Conditions	Min	Max	Units
SPI Characteristics		-	-	-	-
Clock Frequencies	T7/T4		0.08718	870	kHz
	T8	FW 1.0014	1	2100	kHz
I2C Characteristics		-	-	-	-

Serial Communication	Device	Conditions	Min	Max	Units
Clock Frequencies	T7/T4		9.3	472	kHz
	T8	FW 1.0014	10	524	kHz

A-3 Analog Input [T-Series Datasheet]

Please see device-specific subsections below.

Subsections

A-3-1 T4 Analog Input [T-Series Datasheet]

Please see the subsections below.

Subsections

A-3-1-1 T4 AIN General Specs [T-Series Datasheet]

Parameter	Conditions	Min	Typical	Max	Units
Analog Inputs	**-**	**-**	**-**	**-**	**-**
Typical Input Range [1]	low voltage AIN (LV)	0	-	2.5	volts
	high voltage AIN (HV)	-10.0	-	10.0	volts
Max AIN Voltage to GND [2]	No Damage, FIO	-10	-	10	volts
	No Damage, EIO	-6	-	6	volts
	No Damage, HV	-40	-	40	volts
Input Impedance [3]	LV	-	40	-	MΩ
	HV	-	1.3	-	MΩ
Source Impedance [3]	LV	-	-	10	kΩ
	HV	-	-	1	kΩ
Resolution	All Ranges	-	12	-	bits
	LV, 0-2.5	-	0.6	-	mV
	HV, ±10	-	5.0	-	mV
Integral Linearity Error	-	-	±0.05	-	% FS
Differential Linearity Error	-	-	±1	-	counts
Absolute Accuracy [4]	Single-Ended %	-	±0.13	-	% FS
	Single-Ended LV volts	-	±3.3	-	mV
	Single-Ended HV volts	-	±26.8	-	mV
Temperature Drift	-	-	15	-	ppm/°C
Effective Resolution (RMS)	Appendix A-3-1-2	-	>12	-	bits
Command/Response Speed	Appendix A-1	-	-	-	-

Parameter	Conditions	Min	Typical	Max	Units
Stream Performance	Appendix A-1	-	-	50k	-

* LV specs refer to low voltage analog inputs which are available on the T4 analog lines AIN4 - AIN11. HV specs refer to high voltage analog inputs which are available on analog lines AIN0 - AIN3 only.

[1] Note that these are typical input ranges. The actual minimum on the low voltage inputs might not go all the way to 0.0.

[2] Maximum voltage, compared to ground, to avoid damage to the device. Protection level is the same whether the device is powered or not.

[3] The low-voltage analog inputs essentially connect directly to a SAR ADC on the T4, presenting a capacitive load to the signal source. The high-voltage inputs connect first to a resistive level-shifter/divider. The key specification in both cases is the maximum source impedance. As long as the source impedance is not over this value, there will be no substantial errors due to impedance problems.

[4] Absolute accuracy includes INL, DNL, and all other sources of internal error at 25 C and VS=5.0V.

A-3-1-2 T4 Noise and Resolution [T-Series Datasheet]

ADC Noise and Resolution

T-series devices use an internal analog-to-digital converter (ADC) to convert analog voltage into digital representation. The ADC reports an analog voltage in terms of ADC counts, where a single ADC count is the smallest change in voltage that will affect the reported ADC value. A single ADC count is also known as the converter's least significant bit (LSB) voltage. The ADC's resolution defines the number of discrete voltages represented over a given input range. For example, a 16-bit ADC with a ±10 input range can report 65536 discrete voltages (2^16) and has an LSB voltage of 0.305 mV (`20 V ÷ 2^16`).

The stated resolution for an ADC is a theoretical, best-case value assuming no channel noise. In reality, every ADC works in conjunction with external circuitry (amplifiers, filters, etc.) which all possess some level of inherent noise. The noise of supporting hardware, in addition to noise of the ADC itself, all contribute to the channel resolution. In general, the resolution for an ADC and supporting hardware will be less than what is stated for the ADC. The combined resolution for an in-system ADC is termed effective resolution (aka ENOB). Simply put, the effective resolution is the equivalent resolution where analog voltages less than the LSB voltage are no longer differentiable from the inherent hardware noise.

The effective resolution is closely related to the error free code resolution (EFCR) or *flicker-free* code resolution. The EFCR represents the resolution on a channel immune to "bounce" or "flicker" from the inherent system noise. The EFCR is not reported in this appendix. However, it may be closely approximated by the following equation:

```
EFCR = effective resolution - 2.7 bits
```

The T4 and the T7 offer user-selectable effective resolution through the resolution index parameter on any one AIN channel. Internally, the ADC hardware uses modified sampling methods to reduce noise. Valid resolution index values are:

- 0-5 for the T4
- 0-8 for the T7
- 0-12 for the T7-Pro

Increasing the resolution index value will improve the channel resolution, but doing so will usually extend channel sampling times. See section 14.0 AIN for more information on the resolution index parameter and its use.

Noise and Resolution Data

High-Voltage Channels (AIN0-AIN3)

Resolution Index	Effective Resolution [bits]	Effective Resolution [mV]	AIN Sample Time [ms]
1	11.0	10.4	0.07
2	11.4	7.7	0.11
3	12.3	4.2	0.16
4	12.9	2.7	0.29
5*	13.2	2.2	0.5

Low-Voltage Channels (Applicable FIO & EIO)

Resolution Index	Effective Resolution [bits]	Effective Resolution [mV]	AIN Sample Time [ms]
1	10.8	1.2	0.07
2	11.6	0.69	0.11
3	12.0	0.52	0.16
4	12.2	0.43	0.29
5*	12.8	0.29	0.51

* Default command-response Resolution Index for the T4.

A-3-1-3 T4 Signal Range [T-Series Datasheet]

T4 AIN Signal Range

Analog inputs on the T4 are single-ended only. That means the voltage of a given input terminal is acquired versus GND, and thus the signal range is simply the same as the analog input ranges of ±10V or 0-2.5V discussed in various places. See Appendix A-3 for further analog input specs.

A-3-2 T7 Analog Input [T-Series Datasheet]

Please see the subsections below.

Subsections

A-3-2-1 T7 AIN General Specs [T-Series Datasheet]

Parameter	Conditions	Min	Typical	Max	Units
Typical Input Range [1]	Gain=1	-10.6	-	10.1	Volts
	Gain=10	-1.06	-	1.01	Volts
	Gain=100	-0.106	-	0.101	Volts
	Gain=1000	-0.0106	-	0.0101	Volts
Max AIN Voltage to GND [2]	Valid Readings	-11.5	-	11.5	Volts
Max AIN Voltage to GND [3]	No Damage	-20	-	20	Volts
Input Bias Current [4]	**-**	**-**	20	-	nA
Input Impedance [4]	**-**	**-**	1	-	GΩ
Max Source Impedance [4]	**-**	**-**	1	-	kΩ
Integral Linearity Error	Range=10, 1, 0.1	-	-	±0.01	%FS
	Range=0.01	-	-	±0.1	%FS
Absolute Accuracy	Range=10, 1, 0.1	-	-	±0.01	%FS
	Range=10	-	-	±2000	µV
	Range=1	-	-	±200	µV
	Range=0.1	-	-	±20	µV
	Range=0.01	-	-	±0.1	%FS
	Range=0.01	-	-	±20	µV
Temperature Coefficient [5]	**-**	**-**	15	-	ppm/°C
Channel Crosstalk [6]	< 1kHz	-	-100	-	dB
	1kHz - 50kHz	-	20	-	dB/dec

Parameter	Conditions	Min	Typical	Max	Units
High-Speed ADC -3dB Frequency [7]	Gain=1, 10	-	445	-	kHz
	Gain=100	-	337	-	kHz
	Gain=1000	-	63	-	kHz
High-Res ADC -3dB Frequency [7][8]	See Note #7,8	-	-		-
Noise (Peak-To-Peak)	See A-3-2-2	-	-	<1	µV
Effective Resolution (RMS)	See A-3-2-2	-	-	22	bits
Noise-Free Resolution	See A-3-2-2	-	-	20	bits

[1] The range is straightforward for single-ended channels. For differential channels you also need to consider common-mode voltage, so see Appendix A-3-2-3 for more details.

[2] This is the maximum voltage on any AIN pin compared to ground for valid measurements on that channel. For single-ended readings on the channel itself, inputs are limited by the "Typical Input Range" above, and for differential readings consult Appendix A-3-2-3. Further, if a channel has over 13.0 volts compared to ground, readings on other channels could be affected. Because all even channels are on one front-end mux and all odd channels on a second front-end mux, an overvoltage (>13V) on a single channel will generally affect only even or only odd channels.

[3] Maximum voltage, compared to ground, to avoid damage to the device. Protection level is the same whether the device is powered or not. This specification is continuous. For brief transients, such as ESD, the level of protection is much higher.

[4] The key specification here is the maximum source impedance. As long as your source impedance is not over this value, there will be no substantial errors due to impedance problems. For source impedance greater than this value, more settling time might be needed.

[5] Accuracy specs on this page are at room temperature, so Tempco is provided as a typical value reflecting how analog input readings change as temperature changes. For applications attempting better accuracy across varying temperatures there are a couple common strategies. One, on some unused AIN acquire a signal that does not change as the T7 temperature changes, and thus is at a known value so you know how much the T7 readings have changed at any time. Two, evaluate your T7 to determine the relationship of error versus temperature across the temperature range of interest.

[6] Typical crosstalk on a grounded AIN pin, with 20Vpp sine wave on adjacent AIN pin. An adjacent AIN pin refers to multiplexer channel location not channel number, e.g. AIN0-AIN2 or AIN1-AIN3 pairs. An adjacent pin is the worst case. This spec is based on crosstalk seen on a grounded AIN pin, but the same applies if any properly driven signal is connected.

[7] This is the bandwidth of the analog hardware. Any frequencies less than this will go through the analog system to the ADC and be part of the digitized waveform. For DC measurements this is of little concern as ResolutionIndex and averaging can be used to get rid of extra noise. For AC measurements, frequency components below the nyquist point can be removed after digitizing, but frequency components above the nyquist point must be removed before digitizing as they will alias. If unwanted signals with frequencies between the nyquist

point and analog cutoff frequency are expected, and they are expected to have sufficient magnitude to be above the acceptable noise level, then an external hardware filter must be used (often called an anti-alias or anti-aliasing filter).

[8] The fixed -3dB frequencies from note 6 apply to the high-speed ADC (ResolutionIndex = 1-8), but the high-resolution ADC on the T7-Pro (ResolutionIndex = 9-12) has filtering at much lower frequencies. The frequency response at ResolutionIndex=12 is shown in Figure 22 of the AD7190 datasheet. For the response at ResIndex 9/10/11 multiply those x-axis values by 47.9/12.0/2.4. Figure 22 only shows up to 150 Hz, but know that all higher frequencies are also filtered out, except for a narrow passband at 307 kHz. The width of this passband is about 200 Hz at ResIndex=12 increasing to about 10000 Hz at ResIndex=9.

See also: T7 Noise and Resolution

A-3-2-2 T7 Noise and Resolution [T-Series Datasheet]

ADC Noise and Resolution

T-series devices use an internal analog-to-digital converter (ADC) to convert analog voltage into digital representation. The ADC reports an analog voltage in terms of ADC counts, where a single ADC count is the smallest change in voltage that will affect the reported ADC value. A single ADC count is also known as the converter's least significant bit (LSB) voltage. The ADC's resolution defines the number of discrete voltages represented over a given input range. For example, a 16-bit ADC with a ±10 input range can report 65536 discrete voltages (2^16) and has an LSB voltage of 0.305 mV (`20 V ÷ 2^16`).

The stated resolution for an ADC is a theoretical, best-case value assuming no channel noise. In reality, every ADC works in conjunction with external circuitry (amplifiers, filters, etc.) which all possess some level of inherent noise. The noise of supporting hardware, in addition to noise of the ADC itself, all contribute to the channel resolution. In general, the resolution for an ADC and supporting hardware will be less than what is stated for the ADC. The combined resolution for an in-system ADC is termed effective resolution (aka ENOB). Simply put, the effective resolution is the equivalent resolution where analog voltages less than the LSB voltage are no longer differentiable from the inherent hardware noise.

The effective resolution is closely related to the error free code resolution (EFCR) or *flicker-free* code resolution. The EFCR represents the resolution on a channel immune to "bounce" or "flicker" from the inherent system noise. The EFCR is not reported in this appendix. However, it may be closely approximated by the following equation:

`EFCR = effective resolution - 2.7 bits` [1]

The T4 and the T7 offer user-selectable effective resolution through the resolution index parameter on any one AIN channel. Internally, the ADC hardware uses modified sampling methods to reduce noise. Valid resolution index values are:

- 0-5 for the T4
- 0-8 for the T7
- 0-12 for the T7-Pro[2][3]

Increasing the resolution index value will improve the channel resolution, but doing so will usually extend channel sampling times. See section 14.0 AIN for more information on the resolution index parameter and its use.

Notes

[1] The equation used to approximate the EFCR is determined using +/-3.3 standard deviations from the RMS noise measured on an AIN channel.

[2] The default value for RESOLUTION_INDEX is 0, which equates to 8 for T7 command-response reads, 9 for T7-Pro command-response reads, and 1 for T7 & T7-Pro stream reads.

[3] The T7-Pro is equipped with a 24-bit delta-sigma ADC, in addition to the standard 16-bit ADC. Analog conversions occur on the 16-bit ADC when resolution index values 0-8 are used.

Analog conversions occur on the 24-bit ADC when resolution index values 9-12 are used (command response mode only).

[4] The hi-resolution 24-bit ADC is not supported in stream mode.

Noise and Resolution Test procedure

Noise and resolution data was generated by collecting 512 successive voltage readings, using a short jumper between the test channel and ground. To get the effective resolution in volts, we simply take the standard deviation of this array of voltage readings:

Effective Resolution in Volts = StandardDeviation (Data Array in Volts)

To calculate effective resolution in bits, we first convert the voltage readings to 16-bit aligned values. Essentially 16-bit binary values but with decimal places. We then take the standard deviation of those values, and then use the last equation below to calculate effective resolution in bits:

16-bit Aligned Value = 65536.0 * ((Voltage - MinSpanVolts) / (MaxSpanVolts - MinSpanVolts))

RMS Noise = StandardDeviation (16-bit Aligned Values)

Effective Resolution in Bits = 16.0 - log2 (RMS Noise)

See Appendix A-3-2-1 for the min and max span voltages. For example, with Gain = 1 (Range = 10), the min is about -10.6 volts and the max is about 10.1 volts.

Noise and Resolution Data

The data shown below summarizes typical effective resolutions and expected channel sampling times over all resolution index values. Data for the T7 and T7-Pro data are combined and presented together for convenience, where resolution index values 9-12 only apply to the T7-Pro.

The AIN sampling time is the typical amount of time required for the ADC hardware to make a single analog to digital conversion on any channel and is reported in milliseconds per sample. The AIN sampling time does not include command/response and overhead time associated with the host computer/application.

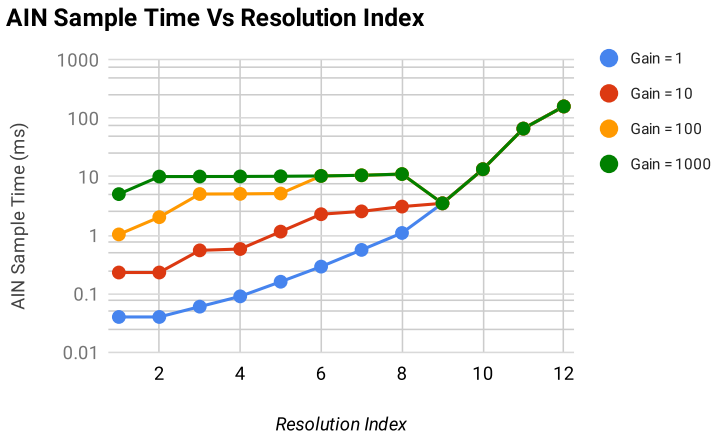


Figure A.3.1.2. T7 analog input effective resolution over various gains and resolution index settings.

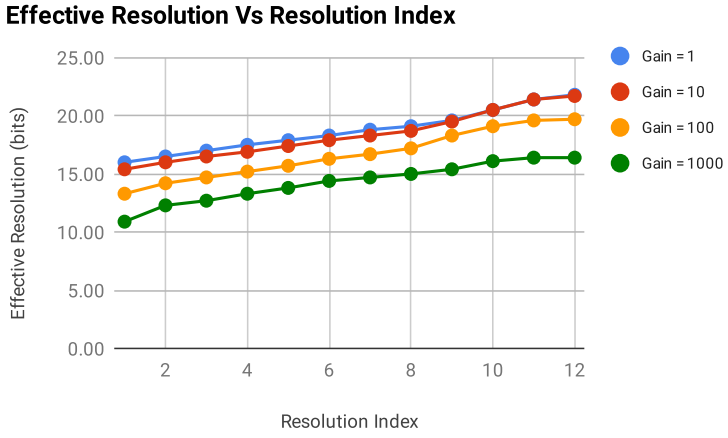


Figure A.3.1.3. T7 analog input LSB voltage over various gains and resolution index settings.

Figure A.3.1.4. T7 AIN sample times for analog inputs over various gains resolution index settings.

Range: ±10 V

Resolution Index	**Effective Resolution** [bits]	**Effective Resolution** [µV]	**AIN Sample** [ms/sample]
1	16.0	316	0.04
2	16.5	223	0.04
3	17.0	158	0.06
4	17.5	112	0.09
5	17.9	85	0.16
6	18.3	64	0.29
7	18.8	45	0.56
8	19.1	37	1.09
9*	19.6	26	3.5
10*	20.5	14	13.4
11*	21.4	7.5	66.2
12*	21.8	5.7	159

* Resolution index 9-12 use the 24-bit ADC that is only available on the T7-Pro.

Range: ±1 V

Resolution Index	**Effective Resolution** [bits]	**Effective Resolution** [µV]	**AIN Sample** [ms/sample]
1	15.4	48	0.23
2	16.0	32	0.23
3	16.5	22	0.55
4	16.9	17	0.58
5	17.4	12	1.15
6	17.9	8.5	2.28
7	18.3	6.4	2.55
8	18.7	4.9	3.08
9*	19.5	2.8	3.5

Resolution Index	**Effective Resolution** [bits]	**Effective Resolution** [µV]	**AIN Sample** [ms/sample]
10*	20.5	1.4	13.4
11*	21.4	0.7	66.2
12*	21.7	0.6	159

* Resolution index 9-12 use the 24-bit ADC that is only available on the T7-Pro.

Gain/Range: ±0.1 V

Resolution Index	**Effective Resolution** [bits]	**Effective Resolution** [µV]	**AIN Sample** [ms/sample]
1	13.3	21	1.03
2	14.2	11	2.03
3	14.7	7.8	5.05
4	15.2	5.5	5.08
5	15.7	3.9	5.15
6	16.3	2.6	10.28
7	16.7	1.9	10.55
8	17.2	1.4	11.08
9*	18.3	0.6	3.5
10*	19.1	0.4	13.4
11*	19.6	0.3	66.2
12*	19.7	0.2	159

* Resolution index 9-12 use the 24-bit ADC that is only available on the T7-Pro.

Gain/Range: ±0.01 V

Resolution Index	**Effective Resolution** [bits]	**Effective Resolution** [µV]	**AIN Sample** [ms/sample]
1	10.9	11	5.03
2	12.3	4.1	10
3	12.7	3.1	10.1

Resolution Index	**Effective Resolution** [bits]	**Effective Resolution** [µV]	**AIN Sample** [ms/sample]
4	13.3	2.1	10.1
5	13.8	1.5	10.2
6	14.4	1.0	10.3
7	14.7	0.8	10.6
8	15.0	0.6	11.1
9*	15.4	0.5	3.5
10*	16.1	0.3	13.4
11*	16.4	0.2	66.2
12*	16.4	0.2	159

* Resolution index 9-12 use the 24-bit ADC that is only available on the T7-Pro.

A-3-2-3 T7 Signal Range [T-Series Datasheet]

T7 AIN Signal Range

This section is only needed for differential measurements where neither channel (positive or negative) is at 0 volts. For single-ended measurements see the simple ranges at the beginning of Table A.3-2.

The instrumentation amplifier in the T7 (see Figure 4.2-2) provides 4 different gains:

- x1 (RANGE is ±10 volts)
- x10 (RANGE is ±1 volts)
- x100 (RANGE is ±0.1 volts)
- x1000 (RANGE is ±0.01 volts)

The figures below show the approximate signal range of the T7 analog inputs at gains of x1 and x1000.

Input Common-Mode Voltage, known as Vcm, is:

```
Vcm = (Vpos + Vneg)/2
```

The voltage of any input compared to GND should be within the VM+ and VM- rails by at least 1.5 volts, so if VM+ and VM- is the typical ±13 volts, the signals should be within ±11.5 volts compared to GND. See Table A5-8 for more information on VM+ and VM-.

Example #1 - invalid because Vcm=10.0 with Vout=10.0 is invalid:

Suppose a differential signal is measured, where:

- Vpos is 10.05 volts compared to GND
- Vneg is 9.95 volts compared to GND
- G=100 (RANGE=±0.1)

That means:

- Vcm=10.0 volts,
- Vdiff=0.1 volts,
- and the expected Vout=10.0 volts.

Figures for G=10 and G=100 are not shown, but Vcm=10.0 volts and Vout=10.0 volts is not valid at G=1 or G=1000, so it is not valid for gains in between.

Example #2 - invalid because Vpos compared to GND is too high:

Suppose a differential signal is measured, where:

- Vpos is 12.0 volts compared to GND

- Vneg is 8.0 volts compared to GND
- G=1 (RANGE=±10)

That means:

- Vcm=10.0 volts,
- Vdiff=4.0 volts,
- and the expected Vout=4.0 volts.

This looks almost okay in the G=1 figure below, but the voltage of Vpos compared to GND is too high so this is not valid.

Example #3 - valid:

Suppose a single-ended signal is measured, where:

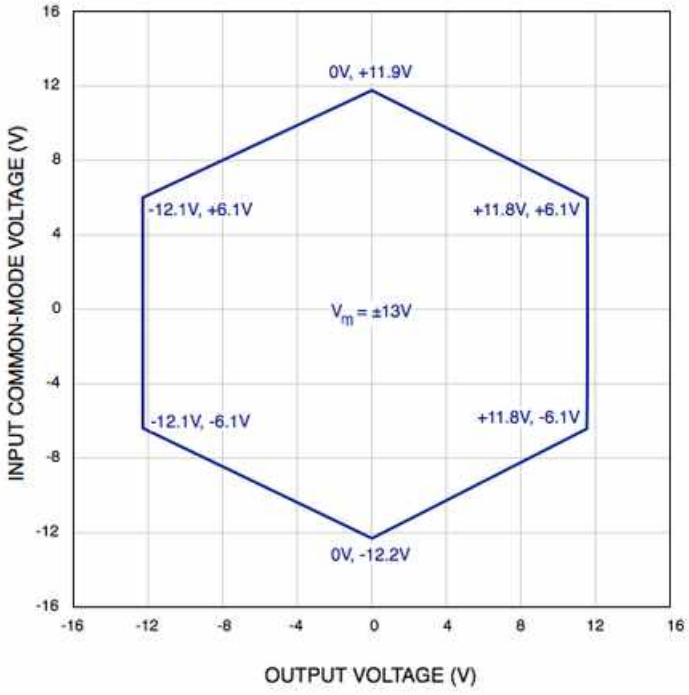
- Vpos is 10.0 volts compared to GND
- G=1 (RANGE=±10)

That means:

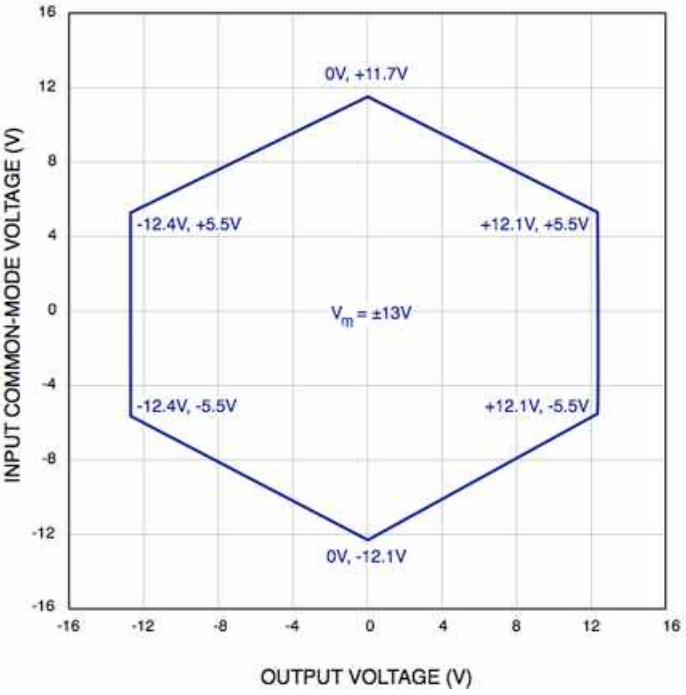
- Vcm=5.0 volts,
- Vdiff=10.0 volts,
- and the expected Vout=10.0 volts.

This is fine according to the figure below.

Input Common-Mode Voltage Range vs. Output Voltage, G = 1



Input Common-Mode Voltage Range vs. Output Voltage, G = 1000



A-3-3 T8 Analog Input [T-Series Datasheet]

Please see the subsections below.

Subsections

A-3-3-1 T8 AIN General Specs [T-Series Datasheet]

Parameter	Conditions	Min	Typical	Max	Units
Typical Input Range	Gain=0.125	-	±11.0	-	Volts
	Gain=0.25	-	±9.6	-	Volts
	Gain=0.5	-	±4.8	-	Volts
	Gain=1	-	±2.4	-	Volts
	Gain=2	-	±1.2	-	Volts
	Gain=4	-	±0.6	-	Volts
	Gain=8	-	±0.3	-	Volts
	Gain=16	-	±0.15	-	Volts
	Gain=32	-	±0.075	-	Volts
	Gain=64	-	±0.036	-	Volts
	Gain=128	-	±0.018	-	Volts
Max AIN Voltage	No Damage	-16	-	16	Volts
Input Bias Current [1]	10kΩ Load	-	4	-	nA
Input Impedance [1]	**-**	**-**	1	-	GΩ
Max Source Impedance [1]	**-**	**-**	10	-	kΩ
Integral Linearity Error	Range=11V [2]	-	-	±0.003	%FS
	Range=9.6V	-	-	±0.005	%FS
	Range=4.8V	-	-	±0.005	%FS
	Range=2.4V	-	-	±0.005	%FS
	Range=1.2V	-	-	±0.005	%FS
	Range=0.6V	-	-	±0.007	%FS
	Range=0.3V	-	-	±0.015	%FS
	Range=0.15V	-	-	±0.015	%FS
	Range=0.075V	-	-	±0.015	%FS

Parameter	Conditions	Min	Typical	Max	Units
	Range=0.036V	-	-	±0.050	%FS
	Range=0.018V	-	-	±0.260	%FS
Absolute Accuracy	Range=11V	-	0.005		%FS
	Range=9.6V	-	0.005		%FS
	Range=4.8V	-	0.006		%FS
	Range=2.4V	-	0.006		%FS
	Range=1.2V	-	0.006		%FS
	Range=0.6V	-	0.008		%FS
	Range=0.3V	-	0.012		%FS
	Range=0.15V	-	0.012		%FS
	Range=0.075V	-	0.024		%FS
	Range=0.036V	-	0.150		%FS
	Range=0.018V	-	0.320		%FS
Temperature Coefficient [3]	**-**	**-**	0.75	-	ppm/°C
Noise (Peak-To-Peak) [4]	See A-3-3-2	-	-	65	µV
Effective Resolution (RMS) [4]	See A-3-3-2	-	-	21.1	bits
Noise-Free Resolution [4]	See A-3-3-2	-	-	18.1	bits

[1] The key specification here is the maximum source impedance. As long as your source impedance is not over this value, there will be no substantial errors due to impedance problems. For source impedance greater than this value, more settling time might be needed.

[2] 11V range characterized from -10 to +10 V.

[3] Tested using the 9.6V range.

[4] Tested using the 9.6V range running at 100 Hz.

See also: T8 Noise and Resolution

A-3-3-2 T8 Noise and Resolution [T-Series Datasheet]

Range: ±11.0 V

Resolution Index	**Rate** [kHz]	**RMS** [bits]	**Bits Noise Free** [bits]
1	10.0	14.8	11.3
1	40.0	14.7	11.3
2	10.0	15.8	12.5
2	40.0	15.6	12.4
3	10.0	16.8	13.5
3	35.5	16.4	13.5
4	10.0	17.3	14.5
4	34.0	16.7	13.9
5	10.0	17.4	14.7
5	32.0	16.7	14.1
6	10.0	17.5	14.7
6	21.3	17.1	14.3
7	10.0	17.5	14.8
7	20.4	17.1	14.4
8	6.0	17.8	15.1
8	16.0	17.3	14.6
9	5.5	18.0	15.3
9	10.6	17.6	14.9
10	5.0	18.0	15.3
10	10.2	17.6	14.9
11	3.5	18.3	15.6
11	8.0	17.8	15.0
12	3.0	18.4	15.7

Resolution Index	Rate [kHz]	RMS [bits]	Bits Noise Free [bits]
12	5.3	18.1	15.4
13	2.5	18.5	15.8
13	5.1	18.1	15.4
14	1.5	18.8	16.1
14	4.0	18.3	15.6
15	0.1	19.6	17.0
15	2.0	15.5	10.5
16	0.1	20.3	17.7
16	1.0	19.3	16.5

Range: ±9.6 V

Resolution Index	Rate [kHz]	RMS [bits]	Bits Noise Free [bits]
1	10.0	15.6	12.1
1	40.0	15.6	12.1
2	10.0	16.6	13.3
2	40.0	16.5	13.3
3	10.0	17.6	14.5
3	35.5	17.2	14.2
4	10.0	18.0	15.3
4	34.0	17.4	14.7
5	10.0	18.2	15.5
5	32.0	17.5	14.8
6	10.0	18.3	15.6
6	21.3	17.9	15.2
7	10.0	18.3	15.6
7	20.4	17.9	15.2

Resolution Index	Rate [kHz]	RMS [bits]	Bits Noise Free [bits]
8	6.0	18.6	15.9
8	16.0	18.1	15.3
9	5.5	18.8	16.1
9	10.6	18.4	15.7
10	5.0	18.8	16.1
10	10.2	18.4	15.6
11	3.5	19.1	16.4
11	8.0	18.6	15.9
12	3.0	19.2	16.5
12	5.3	18.9	16.1
13	2.5	19.3	16.7
13	5.1	18.9	16.1
14	1.5	19.7	17.0
14	4.0	19.1	16.4
15	0.1	20.4	17.7
15	2.0	19.6	16.8
16	0.1	21.1	18.4
16	1.0	20.1	17.3

Range: ±4.8 V

Resolution Index	Rate [kHz]	RMS [bits]	Bits Noise Free [bits]
1	10.0	15.6	12.1
1	40.0	15.5	12.0
2	10.0	16.6	13.4
2	40.0	16.5	13.4
3	10.0	17.5	14.5

Resolution Index	**Rate** [kHz]	**RMS** [bits]	**Bits Noise Free** [bits]
3	35.5	17.2	14.3
4	10.0	18.1	15.3
4	34.0	17.4	14.7
5	10.0	18.2	15.6
5	32.0	17.5	14.8
6	10.0	18.3	15.6
6	21.3	17.9	15.2
7	10.0	18.3	15.6
7	20.4	17.9	15.2
8	6.0	18.6	15.9
8	16.0	18.1	15.4
9	5.5	18.8	16.1
9	10.6	18.4	15.6
10	5.0	18.8	16.2
10	10.2	18.4	15.7
11	3.5	19.1	16.3
11	8.0	18.6	15.8
12	3.0	19.2	16.5
12	5.3	18.9	16.1
13	2.5	19.3	16.6
13	5.1	18.9	16.2
14	1.5	19.6	16.9
14	4.0	19.1	16.3
15	0.1	20.4	17.8
15	2.0	19.6	16.9
16	0.1	21.1	18.5
16	1.0	20.1	17.3

Range: ±2.4 V

Resolution Index	**Rate** [kHz]	**RMS** [bits]	**Bits Noise Free** [bits]
1	10.0	15.5	12.1
1	40.0	15.6	12.1
2	10.0	16.6	13.4
2	40.0	16.5	13.3
3	10.0	17.5	14.4
3	35.5	17.2	14.2
4	10.0	18.0	15.3
4	34.0	17.4	14.8
5	10.0	18.2	15.5
5	32.0	17.5	14.9
6	10.0	18.3	15.5
6	21.3	17.8	15.1
7	10.0	18.3	15.5
7	20.4	17.9	15.2
8	6.0	18.6	15.9
8	16.0	18.0	15.4
9	5.5	18.8	16.1
9	10.6	18.4	15.6
10	5.0	18.8	16.2
10	10.2	18.4	15.7
11	3.5	19.1	16.4
11	8.0	18.5	15.8
12	3.0	19.2	16.4
12	5.3	18.9	16.2
13	2.5	19.3	16.6
13	5.1	18.9	16.2

Resolution Index	Rate [kHz]	RMS [bits]	Bits Noise Free [bits]
14	1.5	19.6	16.9
14	4.0	19.1	16.3
15	0.1	20.4	17.7
15	2.0	19.6	16.9
16	0.1	21.1	18.4
16	1.0	20.1	17.4

Range: ±1.2 V

Resolution Index	Rate [kHz]	RMS [bits]	Bits Noise Free [bits]
1	10.0	15.5	12.1
1	40.0	15.6	12.2
2	10.0	16.6	13.4
2	40.0	16.5	13.3
3	10.0	17.6	14.4
3	35.5	17.2	14.3
4	10.0	18.0	15.4
4	34.0	17.4	14.8
5	10.0	18.2	15.5
5	32.0	17.5	14.8
6	10.0	18.3	15.6
6	21.3	17.8	15.1
7	10.0	18.3	15.5
7	20.4	17.8	15.2
8	6.0	18.6	16.0
8	16.0	18.0	15.3
9	5.5	18.7	16.1

Resolution Index	Rate [kHz]	RMS [bits]	Bits Noise Free [bits]
9	10.6	18.3	15.6
10	5.0	18.8	16.1
10	10.2	18.3	15.6
11	3.5	19.0	16.3
11	8.0	18.5	15.9
12	3.0	19.2	16.5
12	5.3	18.8	16.1
13	2.5	19.3	16.6
13	5.1	18.9	16.2
14	1.5	19.6	16.9
14	4.0	19.0	16.4
15	0.1	20.4	17.8
15	2.0	19.5	16.8
16	0.1	21.0	18.4
16	1.0	20.0	17.3

Range: ±0.6 V

Resolution Index	Rate [kHz]	RMS [bits]	Bits Noise Free [bits]
1	10.0	15.5	12.1
1	40.0	15.5	12.0
2	10.0	16.6	13.3
2	40.0	16.5	13.3
3	10.0	17.5	14.4
3	35.5	17.2	14.2
4	10.0	18.0	15.3
4	34.0	17.3	14.6

Resolution Index	Rate [kHz]	RMS [bits]	Bits Noise Free [bits]
5	10.0	18.1	15.5
5	32.0	17.4	14.8
6	10.0	18.2	15.5
6	21.3	17.8	15.1
7	10.0	18.2	15.5
7	20.4	17.8	15.1
8	6.0	18.6	15.8
8	16.0	18.0	15.3
9	5.5	18.7	16.0
9	10.6	18.3	15.5
10	5.0	18.8	16.1
10	10.2	18.3	15.6
11	3.5	19.0	16.3
11	8.0	18.5	15.8
12	3.0	19.1	16.5
12	5.3	18.8	16.1
13	2.5	19.2	16.5
13	5.1	18.8	16.0
14	1.5	19.5	16.9
14	4.0	19.0	16.2
15	0.1	20.3	17.7
15	2.0	19.5	16.8
16	0.1	21.0	18.3
16	1.0	19.9	17.3

Range: ±0.3 V

Resolution Index	Rate [kHz]	RMS [bits]	Bits Noise Free [bits]

Resolution Index	**Rate** [kHz]	**RMS** [bits]	**Bits Noise Free** [bits]
1	10.0	15.6	12.1
1	40.0	15.5	12.2
2	10.0	16.6	13.3
2	40.0	16.4	13.2
3	10.0	17.4	14.4
3	35.5	17.0	14.2
4	10.0	17.8	15.0
4	34.0	17.1	14.4
5	10.0	17.9	15.2
5	32.0	17.2	14.5
6	10.0	18.0	15.3
6	21.3	17.5	14.9
7	10.0	18.0	15.4
7	20.4	17.6	14.8
8	6.0	18.4	15.7
8	16.0	17.7	15.1
9	5.5	18.5	15.7
9	10.6	18.0	15.3
10	5.0	18.5	15.9
10	10.2	18.1	15.4
11	3.5	18.8	16.0
11	8.0	18.3	15.5
12	3.0	18.9	16.3
12	5.3	18.5	15.9
13	2.5	19.0	16.3
13	5.1	18.6	15.9
14	1.5	19.3	16.7
14	4.0	18.7	16.1

Resolution Index	Rate [kHz]	RMS [bits]	Bits Noise Free [bits]
15	0.1	20.1	17.4
15	2.0	19.2	16.6
16	0.1	20.9	18.3
16	1.0	19.7	17.0

Range: ±0.15 V

Resolution Index	Rate [kHz]	RMS [bits]	Bits Noise Free [bits]
1	10.0	15.5	12.1
1	40.0	15.4	12.0
2	10.0	16.5	13.3
2	40.0	16.1	13.3
3	10.0	17.2	14.3
3	35.5	16.6	13.8
4	10.0	17.4	14.8
4	34.0	16.6	13.9
5	10.0	17.5	14.8
5	32.0	16.7	14.0
6	10.0	17.6	14.8
6	21.3	17.0	14.4
7	10.0	17.5	14.8
7	20.4	17.0	14.3
8	6.0	17.9	15.2
8	16.0	17.2	14.6
9	5.5	18.0	15.3
9	10.6	17.5	14.8
10	5.0	18.1	15.4

Resolution Index	Rate [kHz]	RMS [bits]	Bits Noise Free [bits]
10	10.2	17.5	14.8
11	3.5	18.3	15.6
11	8.0	17.7	15.0
12	3.0	18.4	15.8
12	5.3	18.0	15.3
13	2.5	18.6	15.9
13	5.1	18.1	15.4
14	1.5	18.9	16.2
14	4.0	18.2	15.6
15	0.1	19.6	17.0
15	2.0	18.7	15.9
16	0.1	20.6	17.9
16	1.0	19.2	16.6

Range: ±0.075 V

Resolution Index	Rate [kHz]	RMS [bits]	Bits Noise Free [bits]
1	10.0	15.4	12.0
1	40.0	15.1	11.9
2	10.0	16.2	13.2
2	40.0	15.6	12.8
3	10.0	16.6	13.9
3	35.5	15.8	13.1
4	10.0	16.7	14.0
4	34.0	15.8	13.2
5	10.0	16.7	14.1
5	32.0	15.9	13.2

Resolution Index	Rate [kHz]	RMS [bits]	Bits Noise Free [bits]
6	10.0	16.7	14.0
6	21.3	16.2	13.6
7	10.0	16.7	14.0
7	20.4	16.2	13.5
8	6.0	17.1	14.4
8	16.0	16.4	13.7
9	5.5	17.2	14.5
9	10.6	16.7	14.0
10	5.0	17.3	14.5
10	10.2	16.7	14.0
11	3.5	17.5	14.9
11	8.0	16.9	14.2
12	3.0	17.6	14.9
12	5.3	17.2	14.4
13	2.5	17.8	15.1
13	5.1	17.3	14.6
14	1.5	18.1	15.4
14	4.0	17.4	14.7
15	0.1	19.2	16.5
15	2.0	17.9	15.2
16	0.1	19.8	17.1
16	1.0	18.4	15.7

Range: ±0.036 V

Resolution Index	Rate [kHz]	RMS [bits]	Bits Noise Free [bits]
1	10.0	15.1	11.8

Resolution Index	**Rate** [kHz]	**RMS** [bits]	**Bits Noise Free** [bits]
1	40.0	14.5	11.6
2	10.0	15.5	12.7
2	40.0	14.7	12.0
3	10.0	15.7	13.0
3	35.5	14.9	12.2
4	10.0	15.7	13.0
4	34.0	14.8	12.2
5	10.0	15.8	13.1
5	32.0	14.9	12.2
6	10.0	15.7	13.0
6	21.3	15.2	12.4
7	10.0	15.7	13.0
7	20.4	15.2	12.5
8	6.0	16.1	13.5
8	16.0	15.4	12.7
9	5.5	16.2	13.5
9	10.6	15.7	12.9
10	5.0	16.2	13.6
10	10.2	15.7	13.0
11	3.5	16.5	13.9
11	8.0	15.9	13.3
12	3.0	16.6	14.0
12	5.3	16.2	13.4
13	2.5	16.8	14.0
13	5.1	16.2	13.6
14	1.5	17.1	14.4
14	4.0	16.4	13.6
15	0.1	18.6	15.9
15	2.0	16.9	14.2

Resolution Index	Rate [kHz]	RMS [bits]	Bits Noise Free [bits]
16	0.1	18.7	16.2
16	1.0	17.4	14.7

Range: ±0.018 V

Resolution Index	Rate [kHz]	RMS [bits]	Bits Noise Free [bits]
1	10.0	14.6	11.6
1	40.0	13.7	11.1
2	10.0	14.7	12.0
2	40.0	13.7	11.0
3	10.0	14.7	12.1
3	35.5	13.9	11.3
4	10.0	14.8	12.0
4	34.0	13.9	11.2
5	10.0	14.8	12.1
5	32.0	13.9	11.2
6	10.0	14.8	12.1
6	21.3	14.2	11.4
7	10.0	14.8	12.0
7	20.4	14.2	11.6
8	6.0	15.1	12.5
8	16.0	14.4	11.7
9	5.5	15.2	12.6
9	10.6	14.7	12.0
10	5.0	15.3	12.6
10	10.2	14.7	12.0
11	3.5	15.5	12.9

Resolution Index	**Rate** [kHz]	**RMS** [bits]	**Bits Noise Free** [bits]
11	8.0	14.9	12.2
12	3.0	15.6	12.9
12	5.3	15.2	12.5
13	2.5	15.8	13.1
13	5.1	15.2	12.5
14	1.5	16.1	13.4
14	4.0	15.4	12.7
15	0.1	17.8	15.2
15	2.0	15.9	13.3
16	0.1	17.9	15.3
16	1.0	16.4	13.7

A-3-3-3 T8 Signal Range [T-Series Datasheet]

The T8's analog inputs are isolated. The AIN circuitry will automatically adjust to any common mode within the isolation spec. This ensures that the full differential range is available. For differential ranges, see Appendix A-3-3-1.

A-4 Analog Output [T-Series Datasheet]

Specifications for analog output channels (DAC0 and DAC1) are shown below.

T4

Table A4-1. T4 DAC Information. All specs at room temperature unless otherwise noted.

	Conditions	Min	Typical	Max	Units
Nominal Output Range [1]	No Load	0.01	-	4.98	Volts
	@ ±2.5 mA	0.30	-	4.69	Volts
Resolution	-	-	10	-	Bits
Absolute Accuracy	5% to 95%, No Load	-	±0.15	-	% FS
	-	-	±7.5	-	mV
Integral Linearity Error	-	-	-	±1	counts
Differential Linearity Error	-	-	±0.1	±0.5	counts
Noise [2]	-	-	±1	-	counts
Source Impedance [3]	-	-	110	±10	Ω
Current Limit [4]	Max to GND	-	32	-	mA
Time Constant	-	-	1	-	µs

[1] Maximum and minimum analog output voltage is limited by the supply voltages (VS and GND). The specifications assume VS is 5.0 volts. Also, the ability of the DAC output buffer to driver voltages close to the power rails, decreases with increasing output current.

[2] Specified with no load and set to 0 V. Noise will increase with load and voltage, to a maximum of 3 counts @ 4V with a 100 Ω load. The DAC's ability to reject noise decreases as the output voltage nears the Vs supply.

[3] The source impedance is a combination of fixed resistance and the impedance from the output buffer. The impedance from the buffer will vary depending on the operating conditions. When set to 4 V with a 100 Ω load, Rs is ~ 112 Ω. When set to 4 V with a 10 kΩ load, Rs is ~110 Ω.

[4] The output buffer will limit current to about 30 mA and can maintain this value continuously without damage. Take, for example, a 1.5 ohm resistor from DAC0 to GND, with the internal source impedance of 110 ohms, and DAC0 set to 4.5V. A simple calculation would predict a current of 4.5/(110+1.5) = 40 mA, but the output buffer will limit the current to 32 mA.

T7

Table A4-2. T7 DAC Information. All specs at room temperature unless otherwise noted.

	Conditions	Min	Typical	Max	Units
Nominal Output Range [5]	No Load	0.01	-	4.99	Volts
	@ ±2.5 mA	0.25	-	4.75	Volts
Resolution	-	-	12	-	Bits
Absolute Accuracy	5% to 95% FS	-	±0.06	0	% FS
Integral Linearity Error	-	-	±1.5	±2	counts
Differential Linearity Error	-	-	±0.25	±0.5	counts
Noise [6]	-	-	±100	-	µV
Source Impedance [7]	-	-	50	-	Ω
Current Limit [8]	Max to GND	-	20	-	mA
Time Constant	-	-	4	-	µs

[5] Maximum and minimum analog output voltage is limited by the supply voltages (VS and GND). The specifications assume VS is 5.0 volts. Also, the ability of the DAC output buffer to driver voltages close to the power rails, decreases with increasing output current.

[6] With load, the noise increases if operating too close to VS. With a 1000 ohm load, noise increases noticeably at 4.4V and higher. With a 330 ohm load, noise increases noticeably at 3.7V and higher. With a 100 ohm load, noise increases noticeably at 2.7V and higher.

[7] For currents up to about 8mA, this source impedance dominates the error due to loading. For example, if you load DAC0 with a 1000 ohm resistor from DAC0 to GND, and set DAC0 to 3.0V, the actual voltage at the DAC0 terminal will be about 3.0*1000/(50+1000) = 2.86V. For currents > 8mA, you increasingly get added droop due to the ability of the output buffer to drive substantial current close to the power rails.

[8] The output buffer will limit current to about 20mA and can maintain this value continuously without damage. Take, for example, a 100 ohm resistor from DAC0 to GND, with the internal source impedance of 50 ohms, and DAC0 set to 4.5V. A simple calculation would predict a current of 4.5/(50+150) = 30mA, but the output buffer will limit the current to 20mA. A simple calculation taking into account only the voltage droop due to the internal 50 ohm resistance would predict a voltage at the DAC0 terminal of 4.5*100/(50+100) = 3.0V, but since the current is limited to 20mA the actual voltage at DAC0 would be more like 100*0.02 = 2.0V.

T8

Table A4-3. T8 DAC Information. All specs at room temperature unless otherwise noted.

	Conditions	Min	Typical	Max	Units
Nominal Output Range [1]	No Load	-0.1	-	10.3	Volts
	@ ±20 mA	-0.1	-	10.3	Volts
Resolution	-	-	16	-	Bits
Absolute Accuracy	0V to 10V	-	±0.01	-	% FS
Load Regulation			0.1	0.25	mV / mA
Integral Linearity Error	-	-	±2	±4	counts
Differential Linearity Error	-	-	±1	±2	counts
Noise [2]	-	-	±1	-	counts
Source Impedance [3]	-	-	50	-	Ω
Current Limit	Sourced		20		mA
Slew Rate [4]	-	-	2	-	V/µs

[1] DACs are intended to be 0-10V. Some headroom has been provided to ensure that the full range is usable.

[2] Specified with no load and set to 0 V. Noise will increase with load and voltage, to a maximum of 3 counts.

[3] The T8 will compensate for the voltage drops across its output impendance up to the current limit. Beyond the current limit, the output voltage be reduced to protect the circuitry.

[4] When the current limit is exceeded, the output voltage will be reduced to protect the DAC circuitry.

A-5 General Specs [T-Series Datasheet]

All specs are at room temperature unless otherwise noted.

Power Supply Input

The following table shows the supply voltage that is required. The USB hub or 5V USB adapter in use should fall within the acceptable range.

Table A5-1.

Parameter	Condition	Min	Typical	Max	Units
Supply Voltage	-	4.75	-	5.25	Volts
Supply Current	T4, No connected loads [1]	-	210	-	mA
	T7, No connected loads [1]	-	280	-	mA
	T8, No connected loads, AIN active [1]		670		mA

[1] With no connected loads, the supply current is just the "Power Consumption" from the tables below. The total supply current is this power consumption plus the current provided to any loads (e.g. out of VS terminals). Total supply current should be kept to 500 mA or less on the T4 and T7.

VS Outputs

The following table provides specifications for the VS outputs.

Table A5-2.

Parameter	Condition	Min	Typical	Max	Units
Voltage [1]	-	4.75	-	5.25	Volts
Max Output Current [2]	T4	-	290	-	mA
	T7	-	220	-	mA
	T8, DAC and REFO unused		300		mA

Parameter	Condition	Min	Typical	Max	Units
	T8, Both DACs outputting 20 mA		150		mA
	T8, Both DACs at 20 mA, RefO at 100 mA		75		mA

[1] VS voltage is the same as the power supply voltage.

[2] The max total supply current is 500 mA, so the max current available from VS is 500 mA minus the power consumption of the device from the tables below.

Core Clock

The following table provides specifications for the core clock.

Table A5-3.

Parameter	Condition	Min	Typical	Max	Units
Clock Error	T4, T7 at ~ 25 °C	-	-	±20	ppm
	T4, T7 from -10 to 60 °C	-	-	±50	ppm
	T4, T7 from -40 to 85 °C	-	-	±100	ppm
	T8 from -20 to +75 ºC	-	-	10	ppm
	T8 from -45 to +85 ºC	-	-	30	ppm
Nominal Core Clock Speed	T4	-	80	-	MHz
	T7	-	80	-	MHz
	T8	-	200	-	MHz

Physical

Table A5-4.

Parameter	Condition	Min	Typical	Max	Units

Parameter	Condition	Min	Typical	Max	Units
USB Cable Length	-	-	2	5	meters
Operating Temperature [1]	-	-40	-	85	°C
Operating Humidity [2]	Non-Condensing	-	-	-	-
Screw Terminal Wire Gauge	-	-	26	14	AWG
Mounting Screws	wood screw sizes	#4	#6	#8	-
Enclosure Screws	PH1 pan head	-	#4-20 x 5/8"	-	Phil #1

[1] The device operates across this temperature range, but note that many specs in this appendix are specified at room temperature. In particular, analog input absolute accuracy specs are valid at room temperature, and beyond that there is a tempco that describes how readings might change with temperature.

[2] There is no specific limit on humidity, but the environment must be non-condensing. The T7 must be protected from moisture and particulates.

Power Consumption

At this time USB and Core speed are not intended for user level control, but have been included in the following table to show the capabilities of the device. The values shown are typical.

T4 Power Consumption

Table A5-5. T4 Power Consumption

Core Speed	Ethernet [1]	Ethernet Linked	LEDs	USB [1]	USB Linked	Draw (mA)	Typical Deviation (%)
80M	ON	Yes	ON	ON	Yes	210	±4
80M	ON	Yes	ON	ON	No	195	±4
80M	ON	No	ON	ON	Yes	170	±4
80M	ON	No	OFF	ON	Yes	140	±10

[1] Ethernet and USB require that the core be running at least 20MHz.

T7 Power Consumption

Table A5-6. T7 Power Consumption

Core Speed	Ethernet [1]	Ethernet Linked	AINs	WiFi	WiFi Linked	LEDs	USB [1]	Draw (mA
80M	ON	Yes	ON	ON	Yes	ON	ON	280
80M	ON	Yes	ON	ON	No	ON	ON	280
80M	ON	Yes	ON	OFF	No	ON	ON	250
80M	ON	No	ON	OFF	No	ON	ON	200
80M	OFF	No	ON	OFF	No	ON	ON	170
80M	OFF	No	OFF	OFF	No	ON	ON	140
80M	OFF	No	OFF	OFF	No	OFF	ON	94
80M	OFF	No	OFF	OFF	No	OFF	OFF	86
20M	OFF	No	OFF	OFF	No	OFF	OFF	28
2M	OFF	No	OFF	OFF	No	OFF	OFF	10
250k	OFF	No	OFF	OFF	No	OFF	OFF	6

[1] Ethernet and USB require that the core be running at least 20MHz.

T8 Power Consumption

Table A5-7. T8 Power Consumption

Core Speed (MHz)	Ethernet Linked	# AINs Enabled	DAC0 Sourcing (mA)	DAC1 Sourcing (mA)	REFO Sourcing (mA)	Draw (mA)	Typ Dev (±%
100	No	8	0	0	0	670	4
100	No	0	0	0	0	302	5
100	No	8	20	0	0	765	5
100	No	8	0	20	0	765	5
100	No	8	20	20	0	872	5
100	No	8	0	0	100	766	5
100	Yes	8	0	0	0	695	5

Table A5-8.

Parameter	Condition	Min	Typical	Max	Units
Accuracy vs. Cal Value [1]	~ 25 °C	-	±0.1	±0.2	%
Accuracy vs. Nominal [1]	~ 25 °C	-	-	±5	%
Temperature Coefficient 200UA [2]	~ 25 °C	-	-	-	ppm/°C
Temperature Coefficient 10UA [2]	~ 25 °C	-	-	-	ppm/°C
Maximum Voltage	-	-	2.0V less than VS	-	volts

[1] First spec is the accuracy compared to the value stored during calibration. The second spec is the accuracy compared to the nominal value (e.g. 200.0 µA for the 200µA source).

[2] The temperature coefficient varies strongly with temperature. See the charts in 12.0 200uA and 10uA.

VM+ and VM- (T7)

Table A5-9.

Parameter	Condition	Min	Typical	Max	Units
Typical Voltage	No-load	-	±13	-	volts
	@ 2.5 mA	-	±12	-	volts
Maximum Current	-	-	2.5	-	mA

3.3V RefOut (T8)

The 3.3V Reference output is a load compensated supply which is intended for bridge circuits etc.

Table A5-10.

Parameter	Condition	Min	Typical	Max	Units
Voltage	-	3.293	3.3	3.307	V
Current Regulation	-	8	18	28	µV/mA
Maximum Current	-	-	100	-	mA

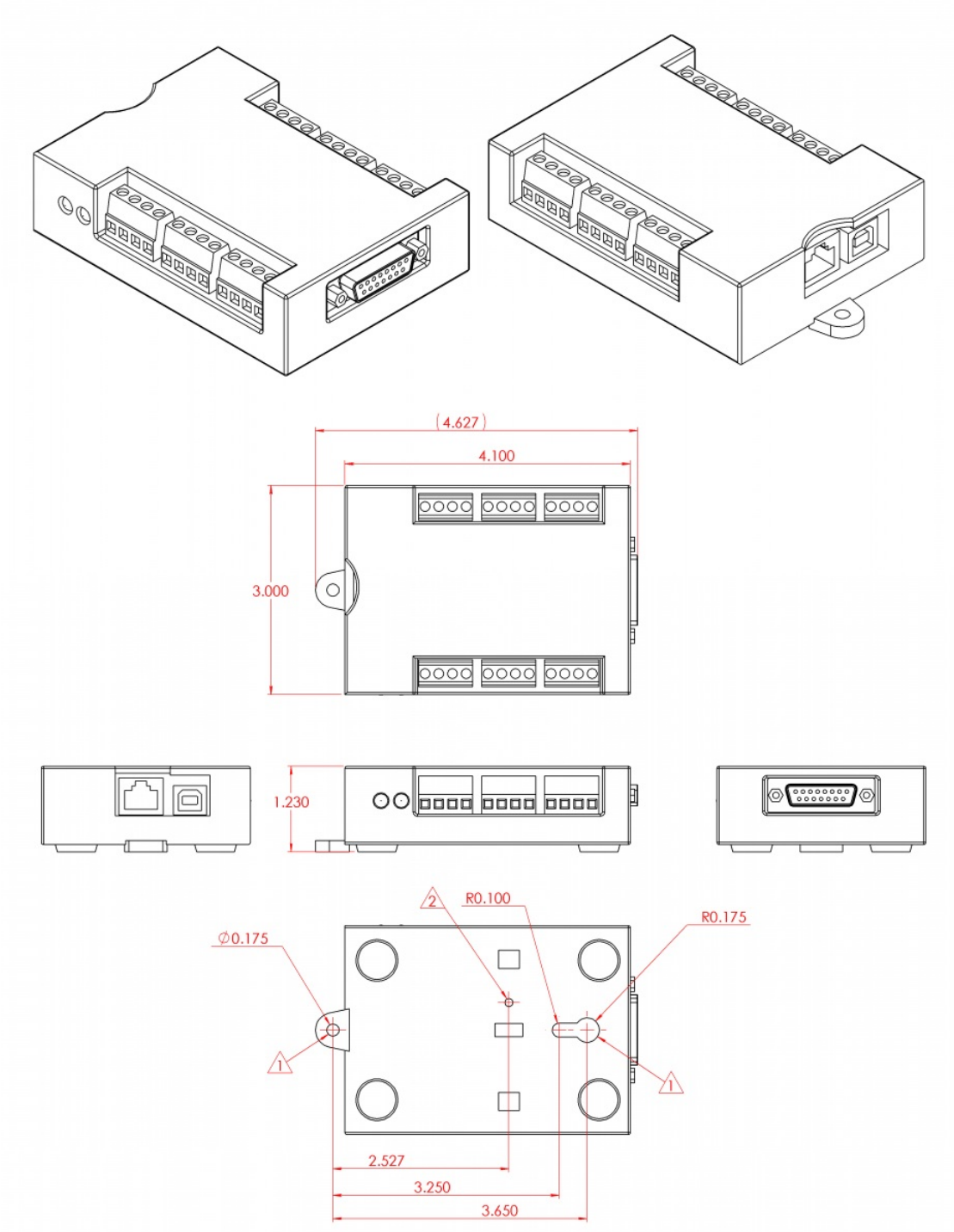
Appendix B - Drawings and CAD Models [T-Series Datasheet]

T-Series enclosure and OEM drawings and models are provided in several neutral file formats. Please see device-specific subsections below.

Subsections

B-1 T4 Drawings and CAD Models [T-Series Datasheet]

CAD drawings are attached to the bottom of this page. The free online Autodesk Viewer can be used to view these and make measurements among other things.



Notes

1. Mounting holes are sized for #8 panhead screws.
2. Receptacle holes for plastic DIN rail clip (TKAD).
3. Dimensions in inches.
4. In addition to the above, there are 4 holes on the bottom of the enclosure with screws to hold the enclosure together; these are #4 panhead screws with 5/8in. length.

Weight

T4 in red enclosure = 144 grams

T4 no enclosure = 74 grams

T4-OEM = 30 grams

More Details

See the OEM page for details on connector pin-headers, holes, power supply information, part options, and more.

Common neutral format CAD models are provided below. Right-click and select the "Save link as..." option to download STEP files.

Files

T4.STEP

T4_PCB-20191202.DXF

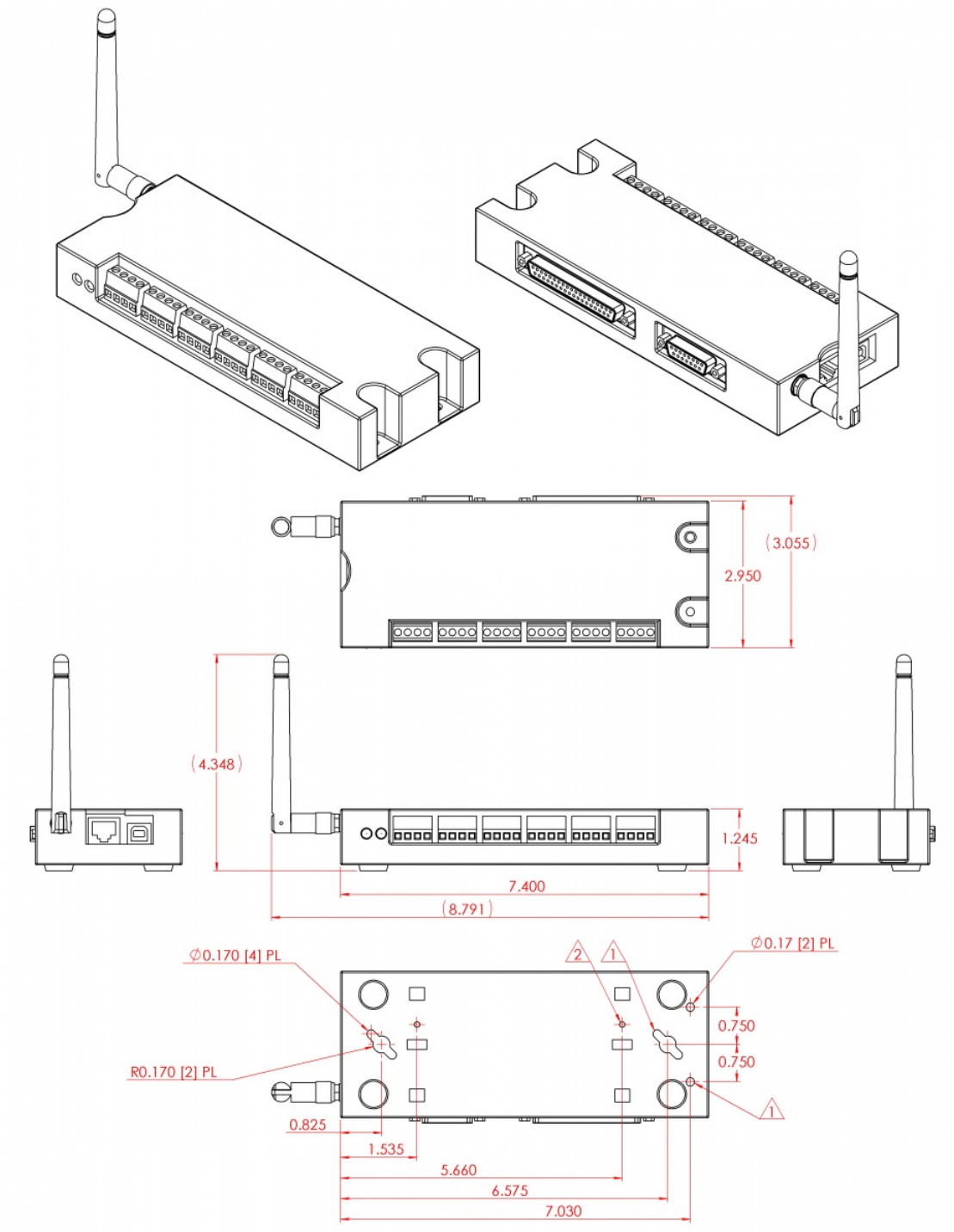
T4_OEM-20191202.STEP

T4_Parasolids-20191017.zip

T4_OEM_Header_Dims_20210728.pdf

B-2 T7 Drawings and CAD Models [T-Series Datasheet]

CAD drawings are attached to the bottom of this page. The free online Autodesk Viewer can be used to view these and make measurements among other things.



Notes

1. Mounting holes are sized for #8 panhead screws.
2. Receptacle holes for plastic DIN rail clip (TKAD).
3. The T7-Pro is depicted above. The standard T7 does not have a wireless antenna.
4. Dimensions in inches.
5. In addition to the above, there are 6 holes on the bottom of the enclosure with screws to hold the enclosure together; these are #4 panhead screws with 5/8in. length.

More Details

See the OEM page for details on connector pin-headers, holes, power supply information, part options, and more.

Common neutral format CAD models are provided below. Right-click and select the "Save link as..." option to download STEP files.

Files

T7_20200925.IGS

T7_20200925.STEP

T7_Pro_20200925.IGS

T7_Pro_20200925.STEP

T7_OEM.STEP

T7_Pro_OEM.STEP

T7_Parasolids.zip

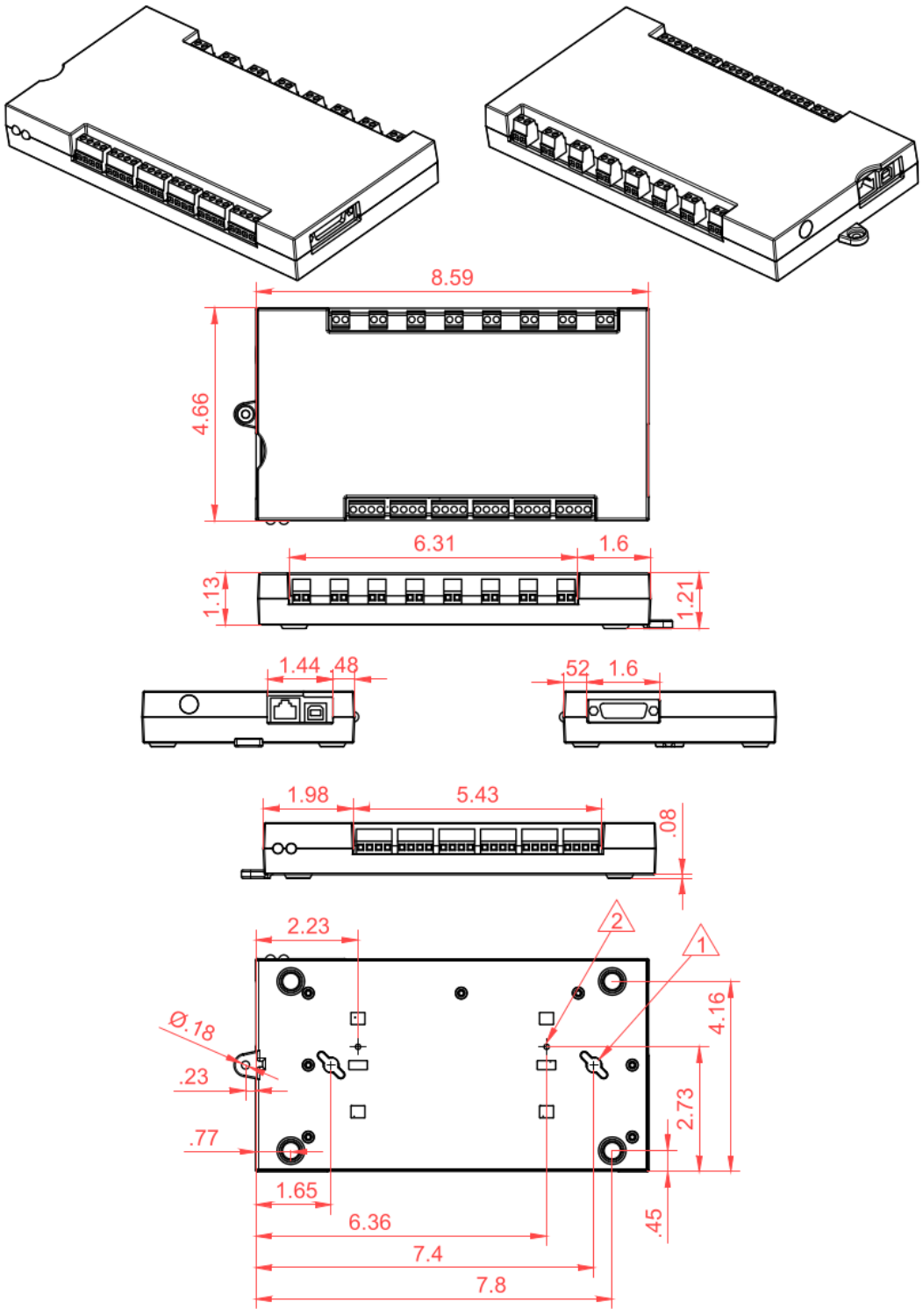
T7_PCB.DXF

T7_OEM_Header_Dims.pdf

T7_PCB_Dims.pdf

B-3 T8 Drawings and CAD Models [T-Series Datasheet]

CAD drawings are attached to the bottom of this page. The free online Autodesk Viewer can be used to view these and make measurements among other things.


8.59
4.66
6.31
1.6
1.13
1.21
1.44
.48
.52
1.6
1.98
5.43
.08
2.23
2
1
Ø.18
.23
.77
4.16
2.73
1.65
6.36
7.4
7.8
.45

Notes

1. Mounting holes are sized for #8 panhead screws.
2. Receptacle holes for plastic DIN rail clip (TKAD).
3. Dimensions in inches.

More Details

See the OEM page for details on connector pin-headers, holes, power supply information, part options, and more. Currently there is not an OEM version of the T8.

Files

Common neutral format CAD models are provided below. Right-click and select the "Save link as..." option to download STEP files.

T8_Case.stp

T8_Model.step

Appendix C - Firmware Revision History [T-Series Datasheet]

Please see your device-specific firmware page:

- T7 Firmware
- T4 Firmware
- T8 Firmware

Appendix D - Packaging Information [T-Series Datasheet]

Please see the following subsections for more information about device packaging.

Subsections

D-1 T4 Packaging Information [T-Series Datasheet]

Package Contents

The normal retail packaged T4 consists of:

- T4 unit itself in red enclosure
- USB cable (6ft / 1.8m)
- Ethernet Cable (6ft / 1.8m)
- USB 5V power supply
- Screwdriver

Other package details

There is no software CD included, so an internet connection is required to download software. Go to the T4 Support Homepage (labjack.com/support/t4) to get started.

Contact support@labjack.com for additional information on shipping.

Package size: 8.5" x 4" x 3"
Package wt: 1.0lb

D-2 T7 Packaging Information [T-Series Datasheet]

Package Contents

The normal retail packaged T7 or T7-Pro consists of:

- T7 (-Pro) unit itself in red enclosure
- USB cable (6ft / 1.8m)
- Ethernet Cable (6ft / 1.8m)
- USB 5V power supply
- Screwdriver
- Antenna (T7-Pro only)

Other package details

There is no software CD included, so an internet connection is required to download software. Go to the T7 Support Homepage (labjack.com/support/t7) to get started.

Contact support@labjack.com for additional information on shipping.

Package size: 10" x 7" x 3"
Package wt: 1.2lb

D-3 T8 Packaging Information [T-Series Datasheet]

Package Contents

The normal retail packaged T8 consists of:

- T8 unit itself in red enclosure
- USB cable (6ft / 1.8m)
- Ethernet Cable (6ft / 1.8m)
- USB 5V power supply
- Screwdriver

LabJack T8 Package Contents

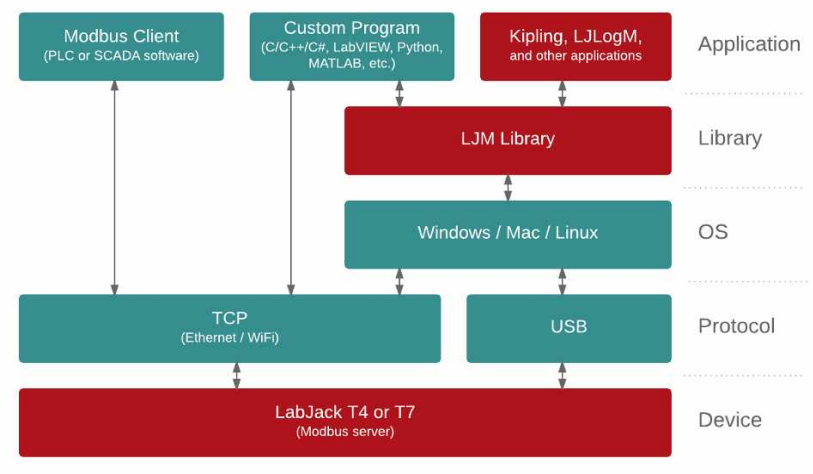
Other package details

There is no software CD included, so an internet connection is required to download software. Go to the T8 Support Homepage (labjack.com/support/t8) to get started.

Contact support@labjack.com for additional information on shipping.

Package size: 10.5" x 7.25" x 3.25"
Package wt: 1.4 lb

Appendix E - Software Options [T-Series Datasheet]



In general, communicating with a T-Series device is quite straightforward. First, look at the T-Series Datasheet or Modbus Map to find the data addresses you want to write or read. Then use one of the software options below to write or read those data addresses.

LabJack software (always free)

- Kipling: A graphical utility for testing, configuration, and troubleshooting. The Register Matrix tab can write and read almost any register.
- Sample applications:
- LJLogM is a simple Windows graphical application to read up to 16 registers by name at a specified interval (typically < 100 scans/second). It displays the values on the screen, can apply scaling, and can log to file.
- LJStreamM is similar to LJLogM but uses stream mode for typical scan rates of > 100 scans/second.
- See the full list
- LJM Library: The high-level cross-platform programming library for simplifying device communication.
- LJM wrappers/examples are available for many languages including Python, LabVIEW, C, Matlab[1], C#[1] and Visual Basic .NET[1], and DAQFactory[1].
- Programming through direct Modbus: You can develop an application that talks directly to the LabJack device over Modbus TCP (Ethernet and WiFi only) using normal TCP sockets and optionally whatever special Modbus support might be available.
- Lua Scripting: Lua code can be loaded on the T-Series' processor itself and executed autonomously. Typically this is done as a complement to software running on a host, but some advanced standalone applications use Lua scripts as the only software.

[1] Windows-only. See LJM library examples page for details.

Third-party software

- DAQFactory: DAQFactory is measurement and automation software from AzeoTech. The free Express version of DAQFactory works with T-Series Devices with limitations on the number of channels. DAQFactory allows non-programmers to make custom applications. It is easy to collect input data, convert to engineering units, display it, and log it to file, without any programming. Scripting is also supported so you can do advanced applications with control and automatic setting of outputs.
- Modbus Client Applications (aka SCADA Software): Most programs that call themselves SCADA are capable of acting as a Modbus client that can directly write/read registers on a Modbus server such as the T7 (Ethernet and WiFi) and the T4 (Ethernet only). PLCs are another common example of Modbus clients and can also write/read directly with a T-Series device.
- 3rd Party Applications: There are other 3rd party applications available for some LabJack devices.