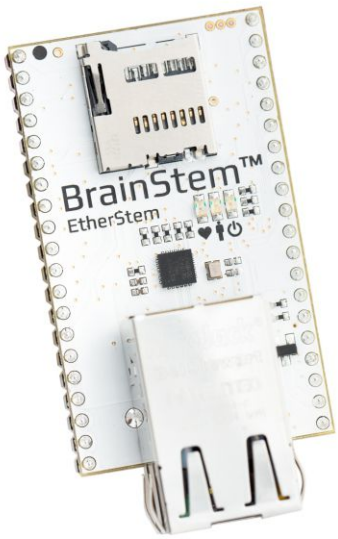


Overview

The EtherStem module is one of a family of 40-pin BrainStem modules which share a common IO inter- face meant for docking to breakout boards, prototyping boards and OEM solutions on a 0.1” inch spaced header. The only difference in functionality and form- factor is the link transport which is through a Ethernet connection.

Features

- 3 Analog (ADC) Inputs at 12-bit resolution
- ADC sampling up to 200kS/s
- ADC bulk capture to RAM up to 3kS
- 15 GPIO Digital Input/Outputs (3.3 V)
- 1 DAC, 10-bit resolution
- I2C Fast Mode Plus (Fm+, 1Mbit) intended for BrainStem bus
- I2C Fast Mode (Fm, 400kbit) user peripheral applications
- User, Link status, Power, Heartbeat LEDs
- 48kB persistent internal storage (4kB slots)
- 1 Internal 16kB RAM Storage Slot
- Small size footprint (50.8 mm x 30.48 mm x 14.03
- mm)

Applications

The BrainStem EtherStem Module is a modular power supply subsystem designed for supplying software controllable voltage power rail to a device under test (DUT) in automated testing environments and research and development. Accurate voltage, temperature and cur- rent rail measurements can be read through the Brain- Stem API.

Description

The BrainStem EtherStem Module is a core BrainStem microcontroller module intended for general purpose applications. The BrainStem EtherStem Module can be easily integrated into any system in order to provide a collection of IO functions for automation systems, em- bedded control systems and remote data collection. It supports GPIO, I2C, A2D, and DAC. The module pro- vides a flexible embedded virtual machine runtime with simple programming interface to enable use with a wide range of input and output devices. There is also a powerful host-side C/C++ API which enables host-PC inter- actions with the module and subordinate hardware de- vices. All Acroname software and APIs are full featured on all host platforms (Mac, Windows, Linux).

Absolute Maximum Ratings

Stresses beyond those listed under ABSOLUTE MAXIMUM RATINGS can cause permanent damage to the device. These are stress ratings only and functional operation of the device at these or any other conditions beyond those indicated under RECOMMENDED OPERATING CONDITIONS is not implied. Exposure to absolute-maximum rated conditions for extended periods affects device reliability and may permanently damage the device.

Parameter	Minimum	Maximum	Units
Input Voltage, V_{supply}	-0.5	4.6	V
V_{supply} current (per pin)	0.0	100mA	A
Voltage to any IO pin	-0.5	4.6	V
Voltage to any I2C pin	0.0	5.5	V

Table 1: Absolute Maximum Ratings

Recommended Operating Ratings

The values presented apply over the full operating temperature, otherwise specifications are at Ta = 25C

Parameter	Conditions/Notes	Minimum	Typical	Maximum	Units
Input Voltage (VCC)		2.4	3.3	3.6	V
RTC Supply Voltage (VRTC)		2.1	3.3	3.6	V
Nominal Supply current	Operating at 25C with 3.3Vcc		72.2		mA
Supply current (Deep Sleep)	Operating at 25C with 3.3Vcc		180.0		uA
ADC Maximum Input Voltage		-0.5		4.6	V
ADC Usable Input Voltage Range		0.0		3.3	V
DAC Voltage Output		0.0		3.3	V
Digital Input Voltage		0.0		5.5	V
GPIO Current, per pin		18.0	20.0	22.0	mA
GPIO Input Logic Low Threshold				1.0	V
GPIO Input Logic High Threshold		2.3			V
GPIO Output Voltage		0.0		3.3	V
Operating Temperature		0.0		80.0	C
Unregulated Voltage Input Measurement		0.0		32.76	V

Table 2: Recommended Operating Ratings

Block Diagram

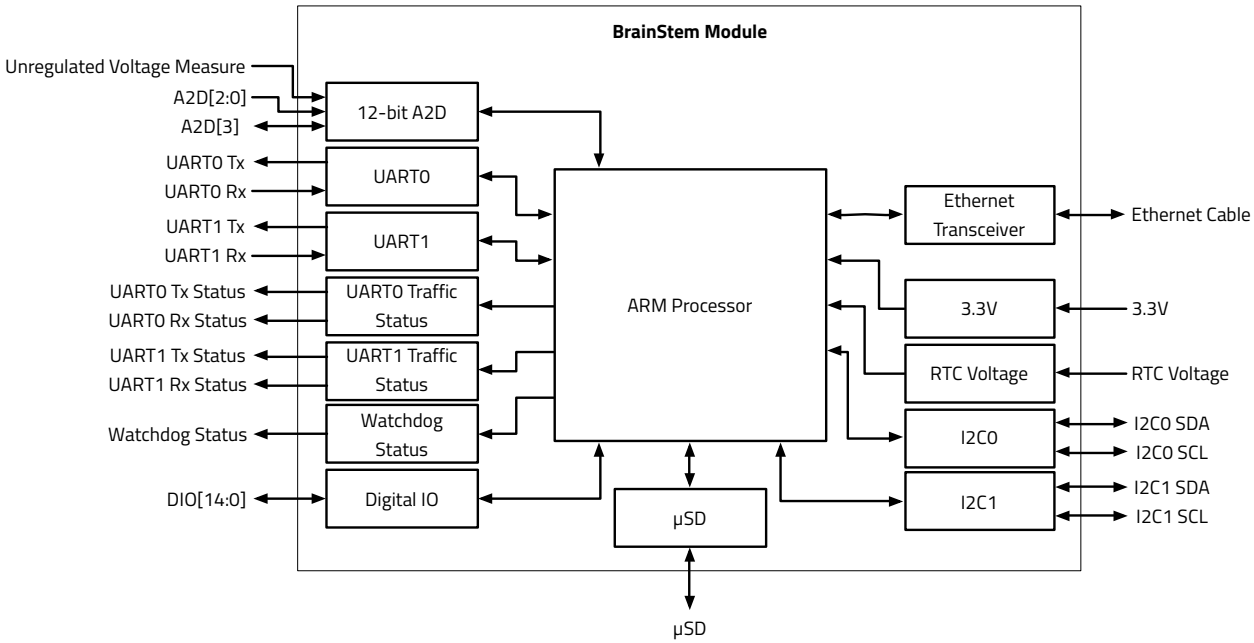


Table 3: System Block Diagram

Pin Functionality

All pin mapping functionality is described in the following table.

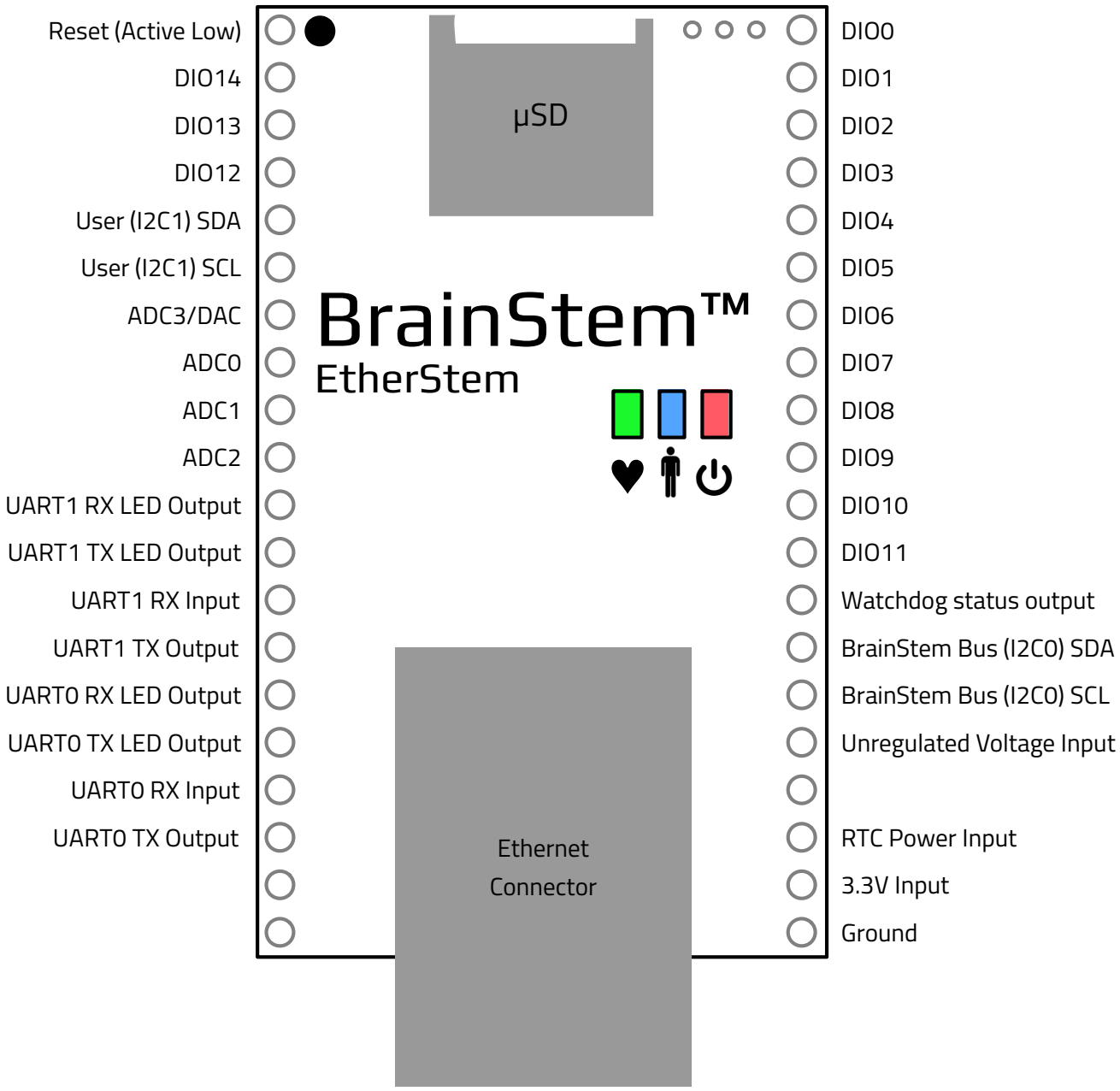


Table 4: BrainStem EtherStem Module pinout drawing.

Pin	Description	Notes
1	Reset	Logic low asserts reset
2	DIO 14	
3	DIO 13	
4	DIO 12	
5	I2C1 SDA	Hardware pull up is included
6	I2C1 SCL	Hardware pull up is included
7	ADC3/DAC	Analog input and output capable
8	ADC0	Analog input only
9	ADC1	Analog input only
10	ADC2	Analog input only
11	UART1 Rx LED status indication	
12	UART1 Tx LED status indication	
13	UART1 Rx	
14	UART1 Tx	
15	UART0 Rx LED status indication	
16	UART0 Tx LED status indication	
17	UART0 Rx	
18	UART0 Tx	
19		
20		
21	Ground	
22	VCC	Recommended 3.3V. See electrical characteristics
23	RTC Voltage input	Recommended 3.3V. See electrical characteristics
24		
25	Voltage Unregulated Measurement	
26	I2C0 SCL	Hardware pull up is included. This is primarily used as BrainStem network bus
27	I2C0 SDA	Hardware pull up is included. This is primarily used as BrainStem network bus
28	Watchdog status pin	
29	DIO 11	
30	DIO 10	
31	DIO 9	
32	DIO 8	

33	DIO 7	
34	DIO 6	
35	DIO 5	
36	DIO 4	
37	DIO 3	
38	DIO 2	
39	DIO 1	
40	DIO 0	

Module Hardware and Software Default Values

The EtherStem module utilizes a subset of BrainStem entity implementations that are specific to the hardware's capabilities. Table 5: EtherStem Hardware and Software Default Values details the BrainStem API entities and macros used to interface to the EtherStem module. For C and C++ developers, these macros are defined in `aEtherStem.h` from the BrainStem development package. For Python development, the module EtherStem class defines the extent of each entity array.

While the BrainStem API entities define the full potential functionality of a given interface, not all features are supported by the EtherStem module. Table 5: EtherStem Hardware and Software Default Values defines each of the options implemented with each entity, which varies by entity index. Calling an unsupported entity option will return an appropriate error (e.g.: `aErrInvalidEntity`, `aErrInvalidOption`, `aErrMode`, or `aErrUnimplemented`) as defined in `aError.h` for C and C++ and the `Result` class in Python.

Parameter	Index	Macro Name or Implemented Options	Notes
Module Definitions:			
Module Base Address	2	aETHERSTEM_MODULE_ADDRESS	See aEtherStem.h
Entity Class Definitions:			
`digital` Entity Quantity	15	aETHERSTEM_NUM_DIG	
`analog` Entity Quantity	4	aETHERSTEM_NUM_A2D	
`i2c` Entity Quantity	2	aETHERSTEM_NUM_I2C	
`clock` Entity Quantity	1		
`Store` Entity Quantity	3	aETHERSTEM_NUM_STORES	
`system` Entity Quantity	1		
`timer` Entity Quantity	8	aETHERSTEM_NUM_TIMERS	

Table 5: EtherStem Hardware and Software Default Values[1]

[1] Refer to `aEtherStem.h` within the BrainStem Development Kit download for actual file

Device Drivers

No drivers are required to use Acroname's EtherStem module.

Capabilities and Interfaces

The EtherStem module software is built on Acroname's BrainStem technology. The module adheres to the BrainStem protocol on I²C and uses BrainStem software APIs. For the most part, functionality that is unique to the EtherStem is described in the following sections; refer to Table 7: Supported EtherStem BrainStem Entity API Methods for a complete list of all available API functionality. All shortened code snippets are loosely based on the C++ method calls – Python and Reflex are virtually the same. Please consult the BrainStem Reference for implementation details[2].

System Entities

Every BrainStem module includes a single System Entity. The System Entity allows access to configuration settings such as the module address, input voltage, control over the user LED and many more.

Saving Entity Settings

Some entities can be configured and saved to non-volatile memory. This allows a user to modify the startup and operational behavior for the EtherStem away from the factory default settings. Saving system settings preserves the settings to become the new default. Most changes to system settings require a save and reboot before taking effect. Use the following command to save changes to system settings before reboot:

```
stem.system.save()
```

Saved Configurations	
Software Offset	I2C Rate
Router Address	I2C Pullup State
Heartbeat Rate	Static IP Address
Boot Slot	IP Configuration

Store Entities

Every BrainStem module includes several Store entities and on-board memory slots to load Reflex files (for details on Reflex, see BrainStem Reference online http://acroname.com/entities/index.html). One Reflex file can be stored per slot. Store[0] refers to the internal memory, with 12 available slots, and store[1] refers to RAM, with 1 available slot.

Digital Entities

The EtherStem has fifteen (15) digital input/outputs (DIO) controlled by the digital entity. Each DIO is controllable via software and is independently current limited for both source and sink currents.

All DIO are input and output capable.

```
stem.digital[0].setConfiguration(mode)
stem.digital[0].getConfiguration(mode)
```

The *mode* parameter is an integer that correlates to the following:

- 0 (digitalConfigurationInput)
- 1 (digitalConfigurationOutput)
- 2 (digitalConfigurationRCServoInput)
- 3 (digitalConfigurationRCServoOutput)
- 4 (digitalConfigurationHiZ)

If a digital pin is configured as an output, setting the digital logic level:

```
stem.digital[0].setState(level)
```

If a digital pin is configured as an input, reading the digital logic level:

```
stem.digital[0].getState(level)
```

If a digital pin is configured in HighZ mode its internal circuitry has been disconnected to create a high impedance. There are no functions that can act on this configuration.

Configuring a digital pin as an RCServo input or output requires the use of the RCServo Entity.

Digital	Input	Output	HighZ	RCServo
DIO0	Yes	Yes	Yes	Input
DIO 1	Yes	Yes	Yes	Input
DIO 2	Yes	Yes	Yes	Input
DIO 3	Yes	Yes	Yes	Input
DIO 4	Yes	Yes	Yes	Output
DIO 5	Yes	Yes	Yes	Output
DIO 6	Yes	Yes	Yes	Output
DIO 7	Yes	Yes	Yes	Output

DIO 8	Yes	Yes	Yes	None
DIO 9	Yes	Yes	Yes	None
DIO 10	Yes	Yes	Yes	None
DIO 11	Yes	Yes	Yes	None
DIO 12	Yes	Yes	Yes	None
DIO 13	Yes	Yes	Yes	None
DIO 14	Yes	Yes	Yes	None

Table 6: Digital IO pin configurations

Analog Entities

The EtherStem has three (3) dedicated analog inputs (ADC) and one (1) analog input that can also be configured as an output (DAC). All controlled by the analog entity. Each analog is controllable via software and is independently current limited for both source and sink currents.

The analog inputs are connected to a 12-bit ADC, and return a value between 0 and 65535, corresponding to a range of 0-3.3V. The analog output is connected to a 10-bit DAC and takes a set value between 0 and 65535, corresponding to a voltage range of 0-3.3V.

For the analog output (analog[3]), setting the DAC value:

```
stem.analog[0].setValue(value)
```

For the analog inputs (analog[0-3]), reading the ADC value:

```
stem.analog[0].getValue(value)
```

The MTM-USBSTEM's ADC's also have the ability of being captured in bulk based on a user defined sample rate. See "Calculating the actual Bulk Capture Sample Rate" for additional information.

I²C Entities

The EtherStem includes access to two separate I²C busses: one operating at a set 1Mbit/s rate, and the other at 400kbits/s.

NOTE: The 1Mbit/s bus, while user-accessible, is also used for primary BrainStem communication so there may be other, non-user-initiated traffic as well, particularly with linked BrainStem units.

The maximum data size for individual `read` and `write` operations on an I²C bus through the BrainStem API is 20 bytes. Sending more than 20 bytes of information has to be done as an iterated sequence. For example, sending 2 bytes (0xBEEF) through the I²C bus to a device with an address 0x42 would be written:

```
stem.i2c.write(0x42, 2, 0xBEEF)
```

Reading 2 bytes of data from a device with an address 0x42 would be written:

```
stem.i2c.read(0x42, 2, buffer)
```

Where *buffer* would be a char array in C++.

Each I²C bus also includes, as a convenience, software-controllable 330Ω pull-up resistors on the SDA and SCL lines, disabled by default. When using the EtherStem in a linked system (communicating over the 1Mbit/s bus), only a single set of pull-ups along the bus should be enabled in order for the I²C bus to work properly (if more than one set is enabled, the lines cannot be pulled low for communication). Similarly, when using a single 40 Pin devices device to communicate with an external device over the I²C bus, either the internal pull-ups can be enabled, or external hardware pull-ups added.

```
stem.i2c.setPullUp(bEnable)
```

The *bEnable* parameter is an integer that correlates to the following:

- 0 (I²C pull-ups off)
- 1 (i2cSetPullup)

RCServo Entities

The EtherStem board is equipped with 4x RCServo inputs and 4x RCServo outputs. The RCServo entity is an overload of the Digital Entity and thus requires proper configuration before this entity can be enabled.

This entity provides a pulsed signal based on the RC Servo standard. Consisting of a period lasting 20ms and high pulse time between 1-2ms. The high time corresponds to a specific position determined by the specific servo being used.

When operating as an RSServo Input you can enable the functionality and then get the position as needed.

```
stem.RCServo[0].setEnable(bool)
stem.RCServo[0].getPosition(position)
```

When operating as an RCServo Output you can enable the functionality and then set the position as needed.

```
stem.RCServo[4].setEnable(bool)
stem.RCServo[4].setPosition(position)
```

Clock Entities

The EtherStem includes a real-time, user-configurable clock entity tracking a time object consisting of year,

month, day-of-the-month, hour, minute and second. These values can be set independently:

```
stem.clock.setYear(year)
stem.clock.setSecond(second)
```

They can also be read independently:

```
stem.clock.getYear(year)
stem.clock.getSecond(second)
```

EtherStem Supported Entity Methods Summary

Detailed entity class descriptions can be found in the BrainStem Reference (http://acroname.com/entities/index.html). A summary of EtherStem class options are shown below. Note that when using Entity classes with a single index (aka, 0), the index parameter can be dropped. For example:

```
stem.system[0].setLED(1) → stem.system.setLED(1)
```

Entity Class	Entity Option	Variable(s) Notes
digital[0-14]	setConfiguration	
	getConfiguration	
	setState	
	getState	
i2c[0-1]	write	
	read	
	setPullup	Disabled by default. I2C communication requires a single set of pull-ups enabled across the bus.
analog[0-3]	getValue	
	getVoltage	
	setBulkCaptureSampleRate	
	getBulkCaptureSampleRate	
	setBulkCaptureNumberOfSamples	
	getBulkCaptureNumberOfSamples	
	initiateBulkCapture	
	getBulkCaptureState	
	getConfiguration	
analog[3]	setValue	
	setConfiguration	
clock[0]	setYear	
	getYear	
	setMonth	
	getMonth	
	setDay	
	getDay	
	setHour	

	getHour	
	setMinute	
	getMinute	
	setSecond	
	getSecond	
store[0-2]	getSlotState	
	loadSlot	
	unloadSlot	
	slotEnable	
	slotDisable	
	slotCapacity	
	slotSize	
system[0]	save	
	reset	
	setLED	
	getLED	
	setSleep	
	setBootSlot	
	getBootSlot	
	getInputVoltage	
	getVersion	
	getModuleBaseAddress	
	getModuleSoftwareOffset	
	setModuleSoftwareOffset	
	setHBInterval	
	getHBInterval	
	getRouterAddressSetting	
	getModule	
	getSerialNumber	
	setRouter	
	getRouter	
	getModel	
timer[0-8]	getExpiration	
	setExpiration	
	getMode	
	setMode	

Table 7: Supported EtherStem BrainStem Entity API Methods[2]

Mechanical

Dimensions are shown in inches [mm]. 3D CAD models are available through the EtherStem product page's Downloads section.

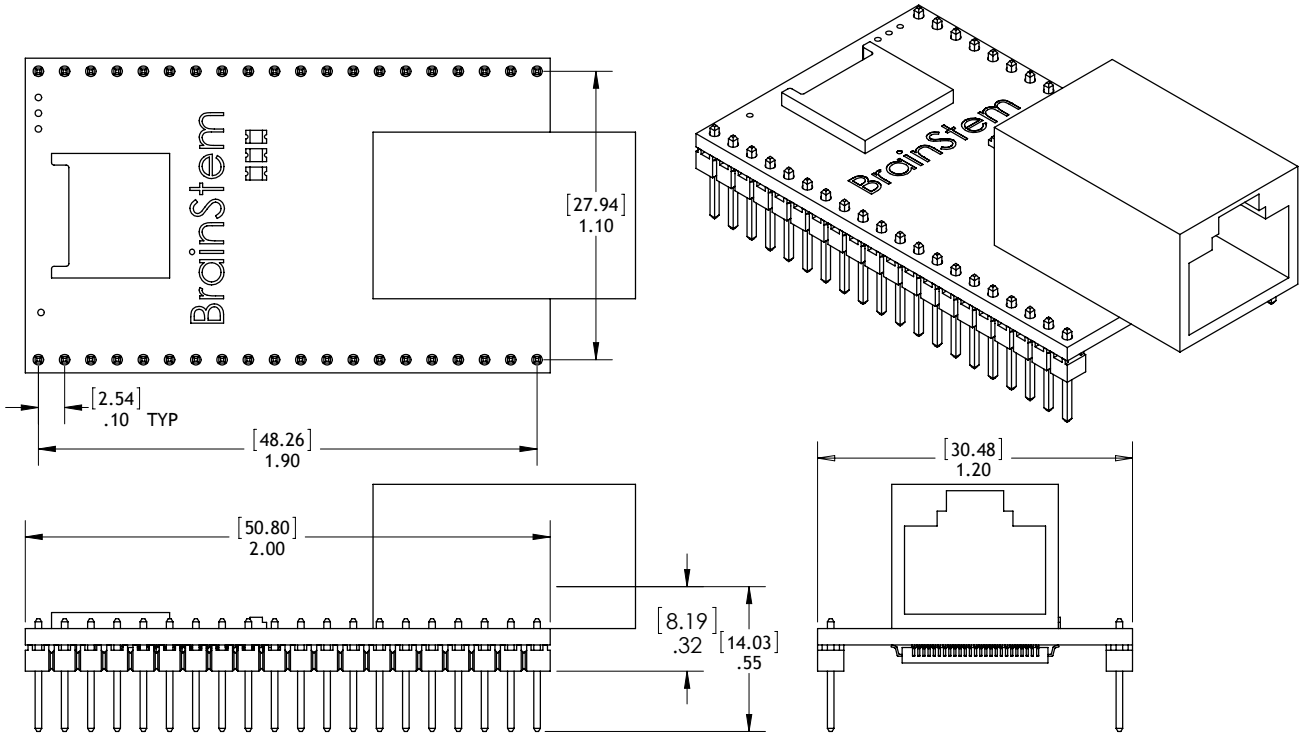
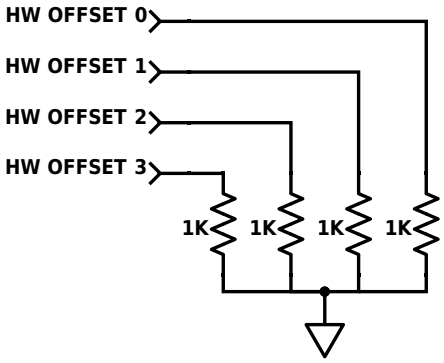


Figure 1: EtherStem Mechanical

[2] See BrainStem software API reference at https://acroname.com/reference/ for further details about all BrainStem API methods and information.

Module Address Hardware Offset Configuration

A hardware offset is one of two ways to modify the devices Module/I2C address. For detailed information on BrainStem networking see the reference guide.



Calculating the actual Bulk Capture Sample Rate

Step 1: Calculate Clock Divisor

Cd = Clock Divisor (This value must be rounded up to the nearest whole number

Cf = Clock Frequency = 96,000,000 Hz

n = Number of cycles required for Analog conversion = 65.

Rf = Requested Frequency in Hz

Cd = Cf / (n * Rf)

Step 2: Calculate Actual Bulk Capture Sample Rate

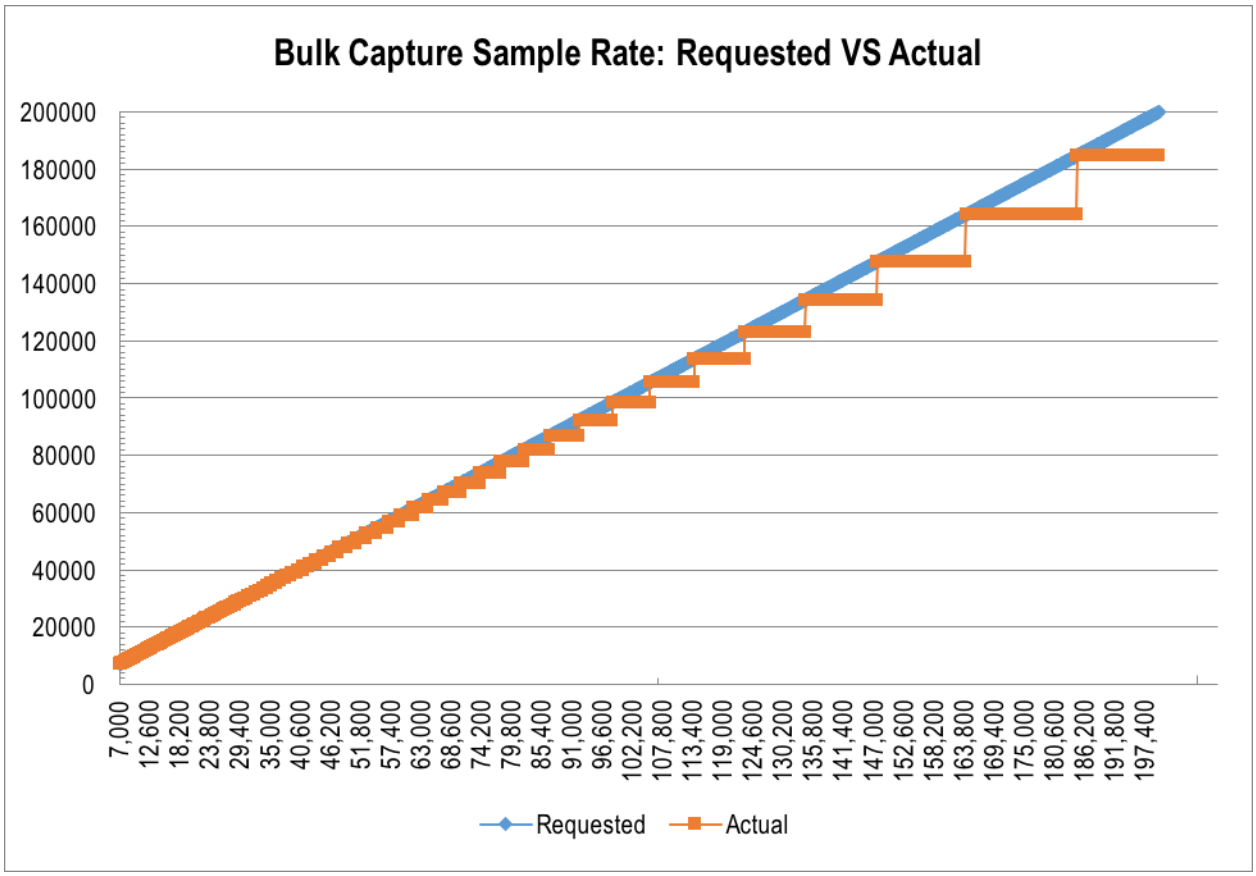
Sr = Sample Rate

Cf = Clock Frequency = 96,000,000 Hz

n = Number of cycles required for Analog conversion = 65.

Cd = Clock Divisor (Calculated in Step 1)

Sr = Cf / (n * Cd)



Document Revision History

All major documentation changes will be marked with a dated revision code

Revision	Date	Engineer	Description
1.0	July 7, 2014	MJK	Initial Revision
1.1	August 27, 2014	ECM	Updated pin mapping, absolute and recommended ratings table
2.0	September 14, 2016	RMN	Converted .tex to .docx
2.1	October 18, 2016	RMN	Added Bulk Capture information