Description

As part of Acroname's MTM product series, the MTM-USBStem is used to implement general purpose control and automation functions in an MTM-based test system. Details on the MTM development platform architecture, BrainStem interface and APIs can be found at www.acroname.com.

The MTM-USBStem implements an on-board BrainStem controller running a RTOS (Real-Time Operating System), which provides a host connection, independent operating capability, the BrainStem interface and the MTM resources identified in this datasheet (GPIO, analog IO, I2C, etc.).

Within the MTM platform architecture, the MTM-USBStem module can operate either independently or as a component in a larger network of MTM modules. Each MTM-USBStem is uniquely addressable and controllable from a host by connecting via the on-board USB connection, the card-edge USB input or through other MTM modules on the local MTM/BrainStem I2C bus.

Acroname's BrainStem™ link is then established over the selected input connection. The BrainStem link allows a connection to the on-board controller and access to the available resources in the MTM-USBStem. The MTM-USBStem can then be controlled via a host running BrainStem APIs or it can operate independently by running locally embedded, user-defined programs based on Acroname's BrainStem Reflex language in the RTOS.

IMPORTANT NOTE:

The MTM-USBStem, like all MTM modules, utilizes a PCIe connector interface but is for use strictly in MTM-based systems – it should never be installed in a PCI slot of a host computer directly. Insertion into a PC or non-MTM system could cause damage to the PC.

Overview

The MTM-USBStem, part of Acroname's MTM (Manufacturing Test Module) product series, is a ruggedized general purpose automation and IO module for use in MTM-based test systems. The MTM-USBStem allows MTM system designers to easily and modularly add digital and analog IO to their designs.

Ideal for reliability and robustness in manufacturing or R&D environments, the analog and digital outputs on the MTM-USBStem are protected and against over-voltage, short-circuit and over-current events. The low profile and small footprint of the MTM-USBStem module makes it ideal for direct integration into test fixtures, thereby eliminating the need for external programmable IO cards or DMMs and their associated cabling.

Built using Acroname's industry-proven and well-adopted BrainStem® technology, resources on the MTM-USBStem are controlled via Acroname's powerful and extensible BrainStem® technology and software APIs.

Typical applications include:

– Manufacturing functional testing
– Validation testing
– Automated test development
– Embedded system development

Features

– 15 overvoltage, short-circuit and over-current protected digital GPIOs
– 3 ADC's (overvoltage and current protected)
– 1 DAC (overvoltage and current protected)
– 1 real-time, user-configurable clock
– 1 user-dedicated I^{2}C 400kHz bus
– 1 BrainStem I^{2}C FM+ (1Mbit/s) bus

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}	6.0	14.0	V
Input Current, I_{supply}	0.0	2.0	A
Voltage to any IO pin	-0.5	V_{supply}+0.5	V
Voltage to any I2C pin	0.0	5.5	V

Table 1: Absolute Maximum Ratings

The MTM system is designed to be used in a system where V_{supply} is the highest voltage connected to all MTM modules. Each module is designed to withstand V_{supply} continuously connected to all IOs, excepting those specified above, including accidental reverse polarity connection between V_{supply} and ground (0V). As with all products, care should be taken to properly match interface voltages and ensure a well-architected current-return path to ground. As with all devices utilizing USB interfaces, care should be taken to avoid ground loops within the USB subsystem. When using the USB interface, ground must be at 0V potential to avoid damaging connected host systems.

Handling Ratings

Parameter	Conditions/Notes	Minimum	Typical	Maximum	Units
Ambient Operating Temperature, T_A	Non-Condensing	0.0	25.0	70.0	°C
Storage Temperature, T_{STG}		-10.0	-	85.0	°C
Electrostatic Discharge, V_{ESD}	IEC 61000-4-2, level 4, contact discharge	0.0	-	±8000	V

Table 2: Handling Ratings

Recommended Operating Ratings

Values presented apply to the full operating temperature range.

Parameter	Conditions/Notes	Minimum	Typical	Maximum	Units
Input Voltage, V_{supply}		6.0	-	12.0	V
Voltage to any IO pin		0	-	3.3	V
Voltage to any I2C pin		0	-	3.3	V

Table 3: Recommended Operating Ratings

Block Diagram

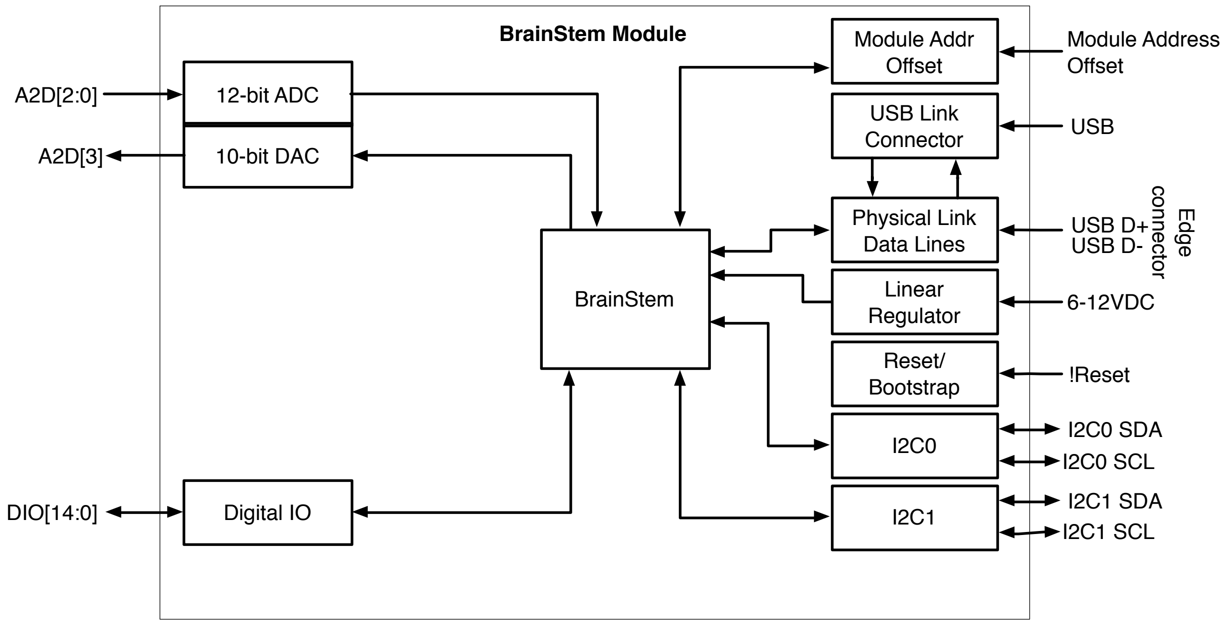


Figure 1: MTM- USBStem Block Diagram

Typical Performance Characteristics

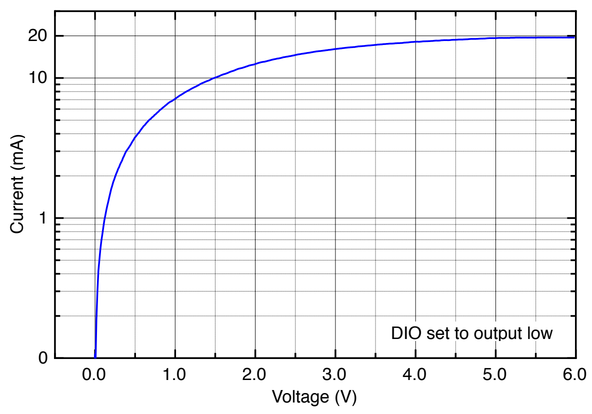
Values presented apply to the full operating temperature range.

Parameter	Conditions/Notes	Minimum	Typical	Maximum	Units
Base Current Consumption, I_{supply}	V_{supply}=6V, Not Enumerated V_{supply}=6V, Enumerated V_{supply}=12V, Not Enumerated V_{supply}=12V, Enumerated	- - - -	82 83 100 102	- - - -	mA
Reset Low Threshold		-	1.2	-	V
Digital Output V_{HI}		-	3.3	-	V
Digital Input Logic High, V_{IH}		2.15	-	-	V
Digital Input Logic Low, V_{IL}		-	-	1.1	V
Digital Output Drive Current	Output high; short to GND Output high into 2.97V	- -	20.0 3.15	30.0 -	mA
Digital Output Sink Current	Output low; short to V_{supply}	-	-20.0	-30.0	mA
Digital Output Short Duration	Output high	-	Infinite	-	hours
Digital Output Overvoltage	V_{supply} on pin	-	Infinite	-	hours
DIO Output Sink Current		-	-	-20.0	mA
DIO Output Source Current	<10% voltage drop (V_{output} >= 2.97V)	-	-	3.15	mA
Digital Sample Rate[1]	Mac OS X Windows 10 Linux – 14.04 LTS Reflex	- - - -	700 1000 850 8200	1000 1000 1000 -	Hz
Analog Output Sink Current		-	-	-20.0	mA
Analog Output Source Current	Set at max. output	-	-	19.0	mA
Analog Output Voltage		0.035	-	3.3	V
Digital Input Resistance	Configuration mode set to both Input and High-Z	-	4.25	4.45	MΩ
Digital Input Leakage Current	Configuration mode set to both Input and High-Z	-	110	-	uA
Analog Input Leakage Current		-	110	-	uA
Analog Input Resistance		-	4.25	4.45	MΩ

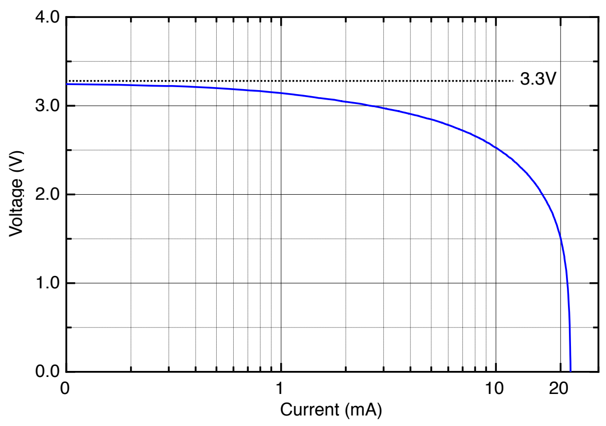
Table 4: Typical Performance Characteristics

[1] Host dependent, test was done as a single instruction, subsequent instructions may affect performance. Measurements taken using BrainStem Library 2.3.2. The Nyquist frequency should be considered when referring to these values.

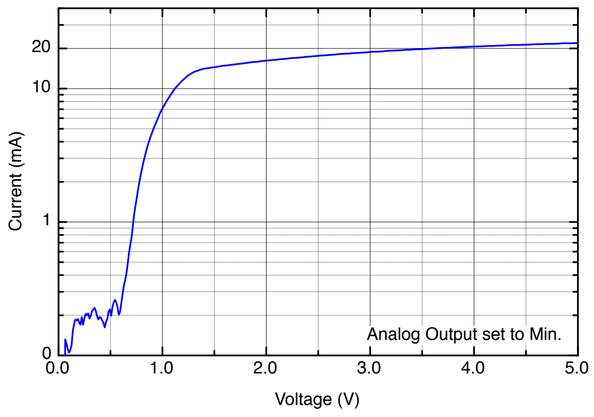
DIO Output Sink Current vs. Applied Voltage



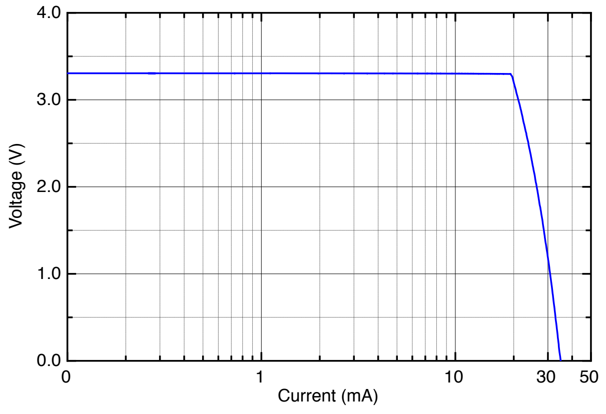
DIO Output High Voltage vs. Current Load



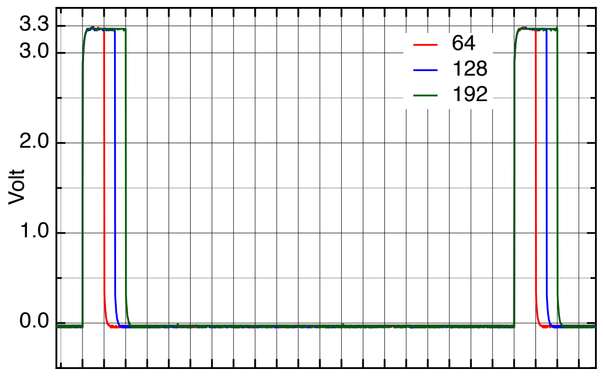
Analog Output Sink Current vs. Applied Voltage



Analog Output (Max.) vs. Current Load

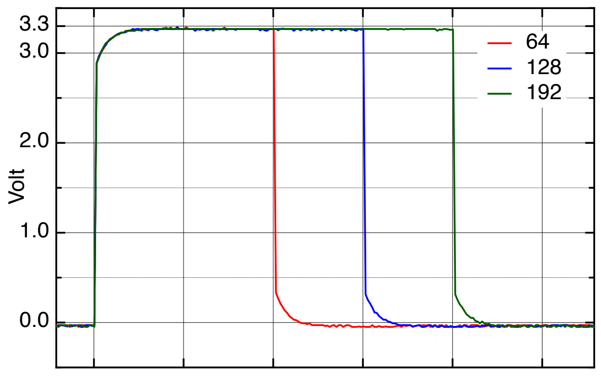


RC Servo Output



1 msec/division

RC Servo Output

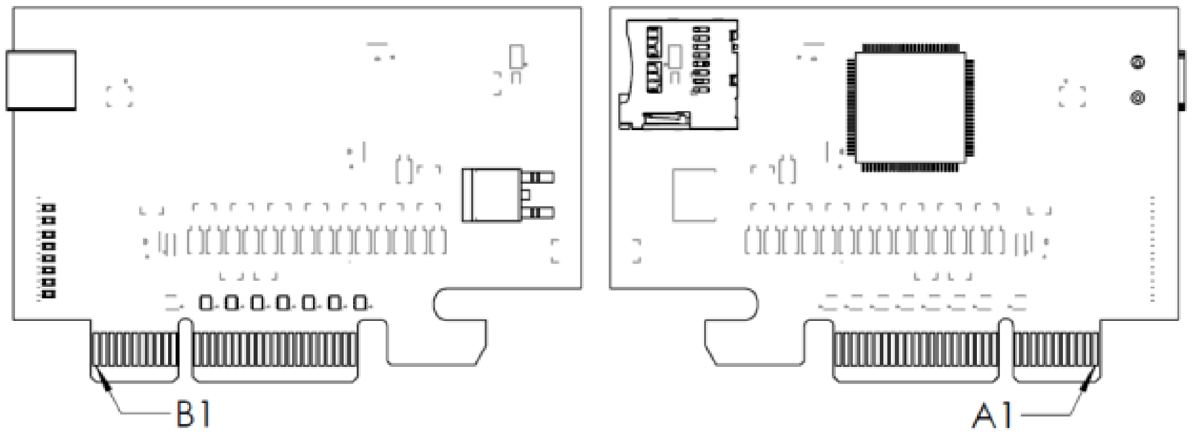


0.5 msec/division

Pinout Descriptions

WARNING: Acroname's MTM line features a PCIe connector that is common in most desktop computers; however, they are NOT intended nor designed to work in these devices. Do NOT insert this product into any PCIe slot that wasn't specifically designed for this product! Failure to follow this warning WILL result in damage to this product and any device you connect it to.

The MTM edge connector pin assignments are shown in the following table. Please refer to Table 3: Recommended Operating Ratings for appropriate signal levels.



Pins Common to all MTM Modules

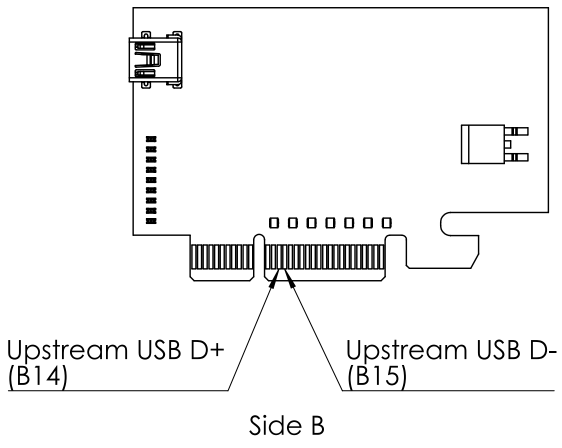
Edge Connector Side A	Edge Connector Side A Description	Edge Connector Side B	Edge Connector Side B Description
1	GND	1	Input Voltage, V_{supply}
2	GND	2	Input Voltage, V_{supply}
3	GND	3	Input Voltage, V_{supply}
4	GND	4	Input Voltage, V_{supply}
5	Reset	5	Input Voltage, V_{supply}
6	GND	6	Reserved, Do Not Connect
7	GND	7	Reserved, Do Not Connect
8	I^2C0 SCL	8	GND
9	I^2C0 SDA	9	GND
10	GND	10	Reserved, Do Not Connect
11	GND	11	Reserved, Do Not Connect
12	Module Address Offset 0	12	Module Address Offset 2
13	Module Address Offset 1	13	Module Address Offset 3

Pins Specific to MTM-USBStem

Edge Connector Side A	Edge Connector Side A Description	Edge Connector Side B	Edge Connector Side B Description
14	Reserved, Do Not Connect	14	USB Upstream Data + (D+)
15	Reserved, Do Not Connect	15	USB Upstream Data - (D-)
16	Reserved, Do Not Connect	16	Reserved, Do Not Connect
17	I²C1 SCL	17	Reserved, Do Not Connect
18	I²C1 SDA	18	Digital IO 0
19	Reserved, Do Not Connect	19	Digital IO 1
20	Reserved, Do Not Connect	20	Digital IO 2
21	Reserved, Do Not Connect	21	Digital IO 3
22	Reserved, Do Not Connect	22	Digital IO 4
23	Reserved, Do Not Connect	23	Digital IO 5
24	Analog 0	24	Digital IO 6
25	Analog 1	25	Digital IO 7
26	Reserved, Do Not Connect	26	Digital IO 8
27	Reserved, Do Not Connect	27	Digital IO 9
28	Reserved, Do Not Connect	28	Digital IO 10
29	Reserved, Do Not Connect	29	Digital IO 11
30	Reserved, Do Not Connect	30	Digital IO 12
31	Analog 2	31	Digital IO 13
32	Analog 3	32	Digital IO 14

Upstream USB Connectivity Options

All MTM modules with upstream USB connections (that's all MTM excluding MTM-EtherStem) have two methods for connection via USB: through the Mini-B connector, or through pins B14 and B15 of the PCIe edge connector (below). The upstream mode defaults to AUTO, which prioritizes based on the presence or absence of VBUS at the Mini-B connector.



Module Hardware and Software Default Values

The MTM-USBStem module utilizes a subset of BrainStem entity implementations that are specific to the hardware's capabilities. Table 5: MTM-USBStem Hardware and Software Default Values details the BrainStem API entities and macros used to interface to the MTM-USBStem module. For C and C++ developers, these macros are defined in `aMTMUSBStem.h` from the BrainStem development package. For Python development, the module `MTMUSBStem` 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 MTM-USBStem module. Table 5: MTM-USBStem 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	6	aMTM_USBSTEM_MODULE _ADDRESS	See aMTMUSBStem.h
Entity Class Definitions:			
`digital` Entity Quantity	15	aMTM_USBSTEM_NUM_DIG	
`analog` Entity Quantity	4	aMTM_USBSTEM_NUM_A2D	
`i2c` Entity Quantity	2	aMTM_USBSTEM_NUM_I2C	
`clock` Entity Quantity	1		
`store` Entity Quantity	2	aMTM_USBSTEM_NUM_STORES	
`system` Entity Quantity	1		
`timer` Entity Quantity	8	aMTM_USBSTEM_NUM_TIMERS	

Table 5: MTM-USBStem Hardware and Software Default Values[2]

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

Capabilities and Interfaces

The MTM-USBStem 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 MTM-USBStem is described in the following sections; refer to Table 7: Supported MTM-USBStem 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[3].

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 MTM-USBStem 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	
Module Software Offset	I2C Rate
Router Address	Boot Slot
Heartbeat Rate	

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 MTM-USBStem 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 MTM-USBStem has three (3) analog inputs (ADC) and one (1) analog 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.

I^{2}C Entities

The MTM-USBStem includes access to two separate I^{2}C 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 I^{2}C 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 I^{2}C 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 I^{2}C bus also includes 330Ω pull-up resistors on the SDA and SCL lines.

•

RC Servo Entities

The MTM-USBStem board is equipped with 4 RC servo inputs and 4 RC servo outputs. The RC Servo entity is an overload of the Digital Entity and thus requires proper configuration before this entity can be enabled. For example, enabling RC servo input mode for a digital pin 0 is done with:

```
stem.digital[0].setConfiguration(digitalCo
nfigurationRCServoInput)
```

With the RC servo entity, digital output pins generate 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. RC servo inputs, measure this high time and return the corresponding position for a servo.

When operating as an RC servo input, enabling the functionality and reading the position is done with:

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

When operating as an RC servo output enabling the functionality and setting the position is done with:

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

Clock Entities

The MTM-USBStem 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)
```

MTM-USBStem Supported Entity Methods Summary

Detailed entity class descriptions can be found in the BrainStem Reference (http://acroname.com/entities/index.html). A summary of MTM-USBStem 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`	
`rcservo[0-7]`	`setEnable`	
	`getEnable`	
	`setPosition`	`Index 4-7 only`
	`getPosition`	
	`setReverse`	`Index 4-7 only`
	`getReverse`	
`i2c[0-1]`	`write`	
	`read`	
`analog[0-2]`	`getValue`	
	`getVoltage`	
	`setBulkCaptureSampleRate`	
	`getBulkCaptureSampleRate`	
	`setBulkCaptureNumberOfSamples`	
	`getBulkCaptureNumberOfSamples`	
	`initiateBulkCapture`	
	`getBulkCaptureState`	
`analog[3]`	`setValue`	
`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	
	setBootSlot	
	getBootSlot	
	getInputVoltage	
	getVersion	
	getModuleBaseAddress	
	getModuleSoftwareOffset	
	setModuleSoftwareOffset	
	getModuleHardwareOffset	
	setHBInterval	
	getHBInterval	
	getRouterAddressSetting	
	getModule	
	getSerialNumber	
	setRouter	
	getRouter	
	getModel	
	routeToMe	
timer[0-8]	getExpiration	
	setExpiration	
	getMode	
	setMode	

Table 7: Supported MTM-USBStem BrainStem Entity API Methods[3]

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

LED Indicators

The MTM-USBStem board has a number of LED indicators to assist with MTM system development, debugging, and monitoring. These LEDs are shown in the diagrams below.

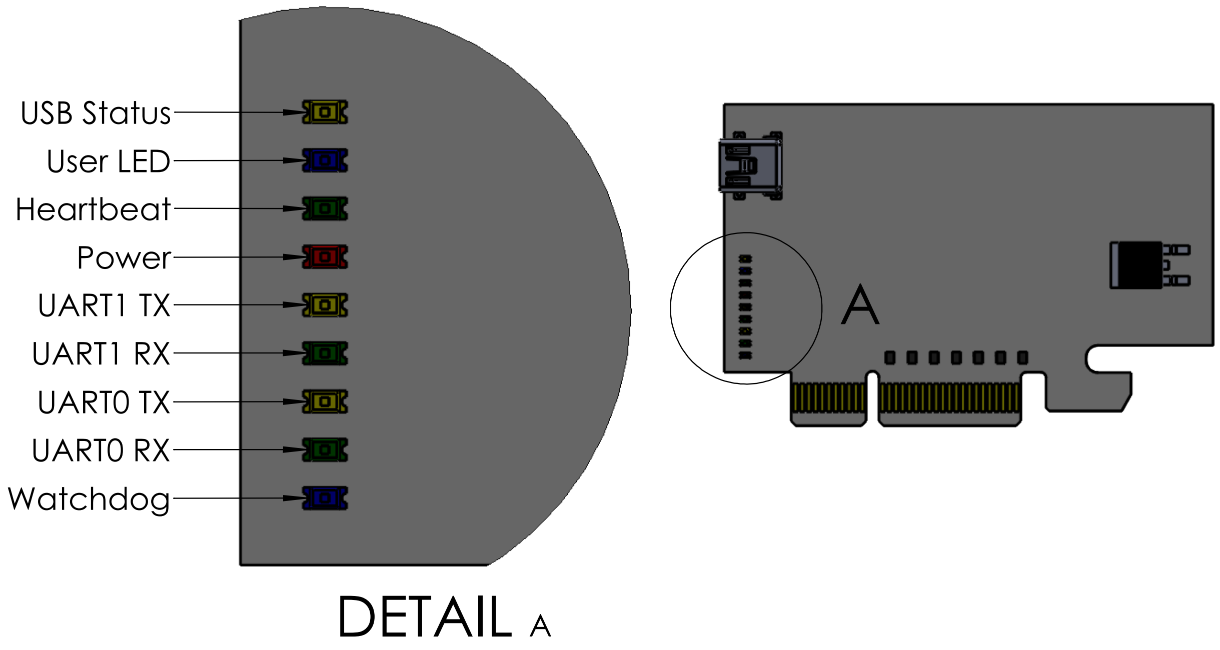


Figure 2: MTM-USBStem LED Indicators

Edge Connector Interface

All MTM products are designed with an edge connector interface that requires a compatible edgeboard connector on the carrier PCB. Acroname recommends the through-hole PCI-Express (PCIe) Vertical Connector. The connectors can be combined with an optional retention clip, as shown below.

MTM Product	Manufacturer	Manufacturer Part Number	Description
MTM-Relay	Amphenol FCI Samtec	10018784-10203TLF PCIE-164-02-F-D-TH	PCI-Express 164-position vertical connector
MTM-IO-Serial	Amphenol FCI Samtec	10018784-10202TLF PCIE-098-02-F-D-TH	PCI-Express 98-position vertical connector
MTM-PM-1	Amphenol FCI Samtec	10018784-10201TLF PCIE-064-02-F-D-TH	PCI-Express 64-position vertical connector
MTM-USBStem	Amphenol FCI Samtec	10018784-10201TLF PCIE-064-02-F-D-TH	PCI-Express 64-position vertical connector
MTM-EtherStem	Amphenol FCI Samtec	10018784-10201TLF PCIE-064-02-F-D-TH	PCI-Express 64-position vertical connector
All Models	Amphenol FCI	10042618-003LF	PCI-Express Retention Clip (optional)

Table 8: PCI-Express Edge Connectors for MTM Products

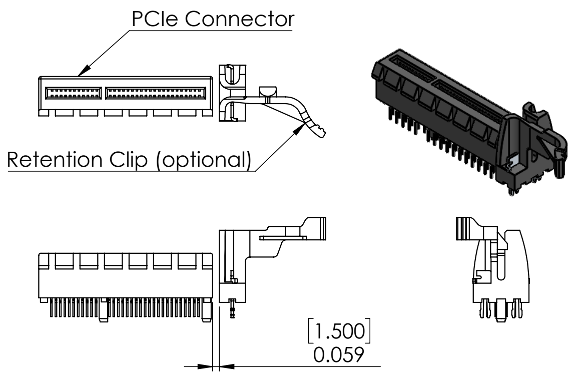


Figure 3: PCIe Vertical Connector with optional Retention Clip

MTM Edge Connector Specifications	Description
Contact Finish	Gold
Card Thickness	0.0625” [1.59mm]
Number of Rows	2
Number of Positions	Variable (see Table 8: PCI-Express Edge Connectors for MTM Products)
Pitch	0.039” (1.00mm)

Table 9: MTM Edge Connector Specifications

Amphenol FCI Drawings and Layout: http://portal.fciconnect.com/Comergent//fci/drawing/10018784.pdf

Amphenol FCI Product Specification: http://portal.fciconnect.com/res/en/pdffiles/Specs/gs-12-233.pdf

Samtec Product Catalog: http://suddendocs.samtec.com/catalog_english/pcie.pdf

Mechanical

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

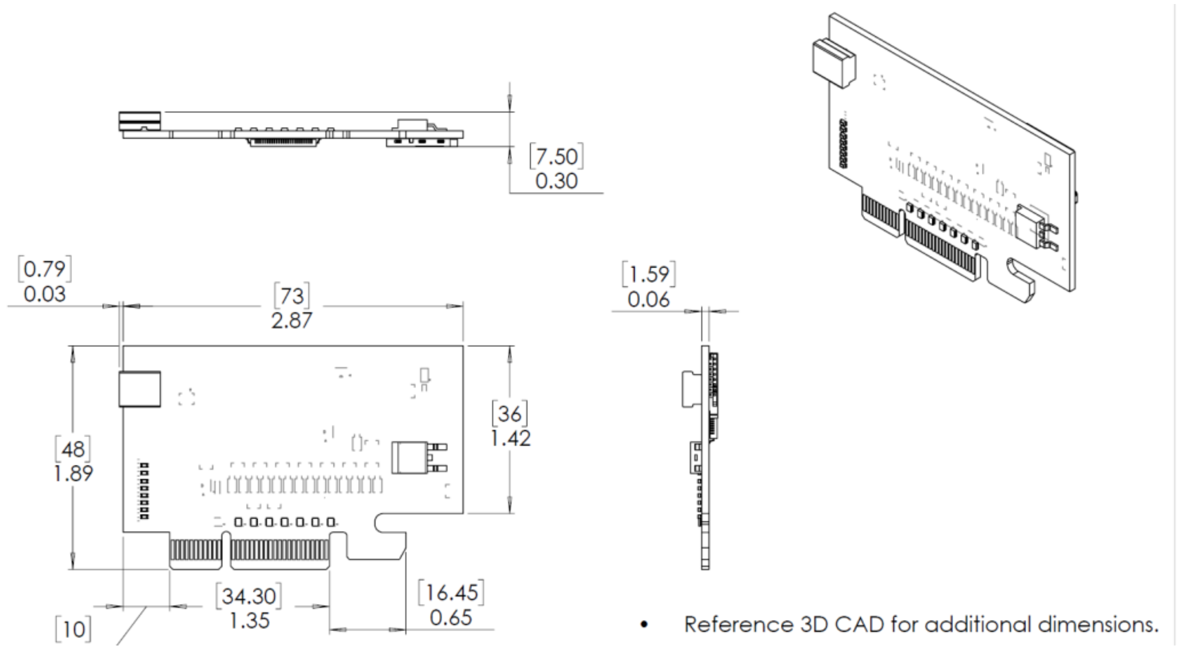


Figure 4: MTM-USBStem Mechanical

Module Address Hardware Offset Configuration

A hardware offset is one of two ways to modify the module's address on the BrainStem network. Using hardware offset pins is useful when more than one of the same type of module is installed on a single BrainStem network. Applying a different hardware offset to each module of the same type in one network allows for all the modules to seamlessly and automatically configure the network for inter-module communication. Further, modules can be simply swapped in and out of the network without needing to pre-configure a module's address before being added to a network. Finally, when a system has more than one of the same type of module in a network, the module address hardware offset can be used to determine the module's physical location and thus its interconnection and intended function. For detailed information on BrainStem networking see the reference guide.

Each hardware offset pin can be left floating or pulled to ground with a 1kΩ resistor or shorted to ground. Pin states are only read when the module boots, either from a power cycle, hardware or software reset. The hardware offset pins are treated as an inverted binary number which is multiplied by 2 and added the to the module's base address. The hardware offset calculation is detailed in the following table.

HW Offset Pin				Address Offset	Module Base Address	Final Module Address
0	1	2	3			
NC	NC	NC	NC	0	4	4
1	NC	NC	NC	2	4	6
NC	1	NC	NC	4	4	8
NC	NC	1	NC	8	4	12
NC	NC	NC	1	16	4	20
1	NC	NC	1	4+16	4	24

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

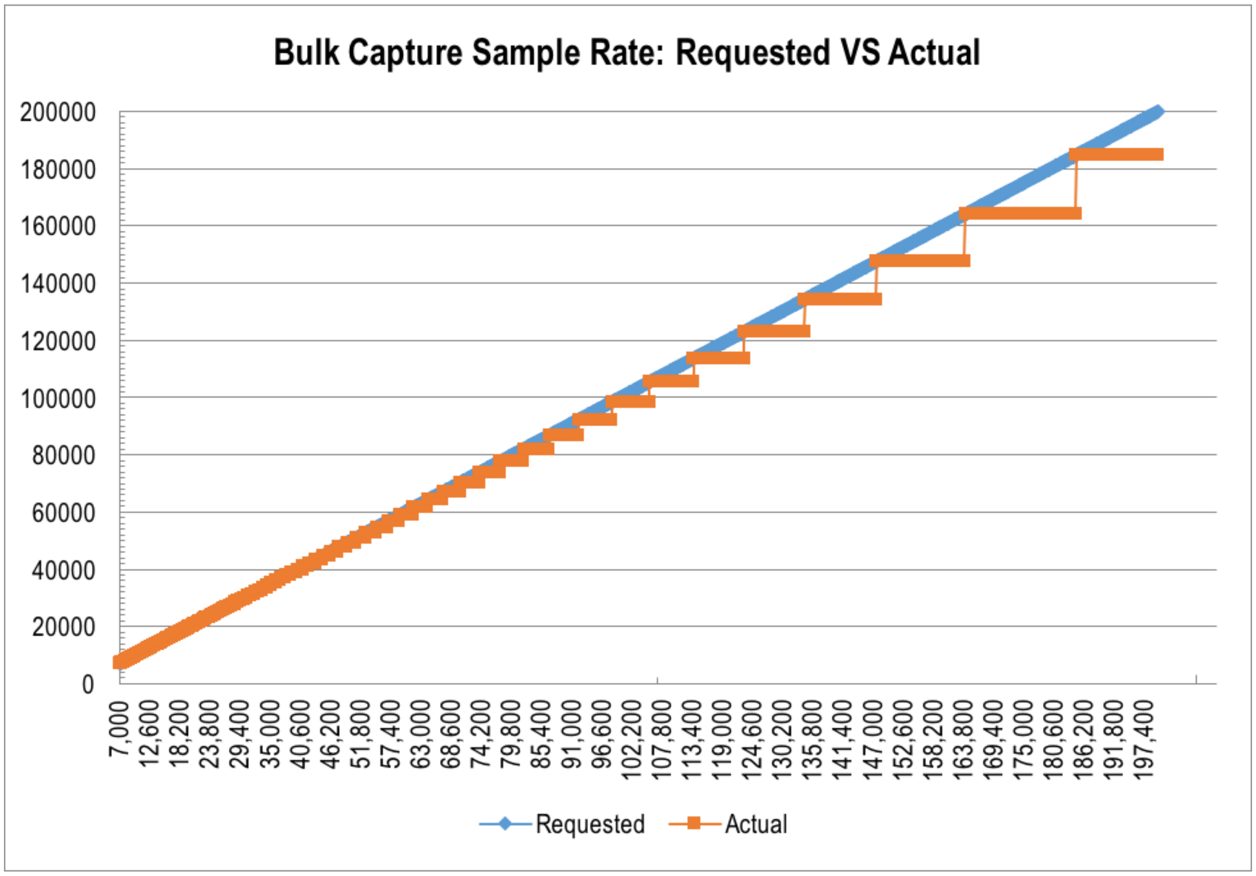
Sr = Sample Rate

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	October 26, 2015	JTD	Initial Revision
1.1	April 12, 206	JTD	Corrected typographical errors
1.2	September 2016	RMN	Formatting, Error checking, updates
1.3	October 2016	LCD	Updated Overview, Features and Description sections
1.4	October 2016	RMN	Added Bulk Capture information
1.5	December 2016	JG	Clarified I2C pull-ups; update supported API calls