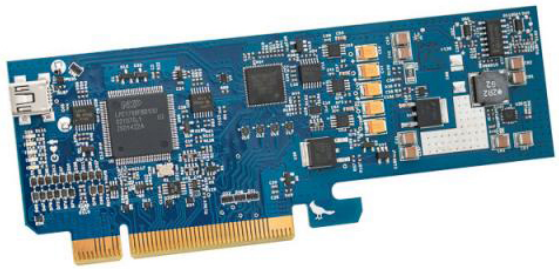


Overview

The Acroname MTM-IO-Serial module, part of Acroname's MTM (Manufacturing Test Module) product series, is a software-controlled USB 2.0 hub, designed for MTM-based manufacturing or R&D test systems. The MTM-IO-Serial allows MTM system designers to easily and modularly add USB 2.0 connectivity as well as serial UART and GPIO functions at two variable IO levels.

Built using Acroname's industry-proven and well-adopted BrainStem® technology, resources on the MTM-IO-Serial 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
– Firmware loading
– Serial communications
– USB switching and control

Features

– 4 downstream USB 2.0 ports, software controlled Hi-Z disable
– 2 independent 1.8-5.0V adjustable high-current voltage rails, current limited to 100mA
– 4 adjustable-voltage serial UART ports (2 per adjustable rail) software controlled Hi-Z disable
– 4 adjustable-voltage digital GPIO (2 per adjustable rail)
– All GPIO overvoltage and overcurrent protected
– 1 fixed high-current 5.0V output, current limited to 100mA
– 1 downstream USB 2.0 port on edge connector, always on for daisy-chaining multiple modules over USB
– 1 downstream USB 2.0 port type-A connector, always on for daisy-chaining multiple modules over USB
– 1 BrainStem I²C FM+ (1Mbit/s) bus

Description

As part of Acroname's MTM series, the MTM-IO-Serial module is a key component for manufacturing test for electronic devices using a USB 2.0 interface, serial UART and/or one or more IO interface voltages. For more information on the MTM platform architecture, please see www.acroname.com.

The MTM-IO-Serial implements an on-board BrainStem controller running a RTOS (Real-Time Operating System), which provides a USB host connection, independent operating capability and the BrainStem interface, for control of the MTM resources identified in this datasheet (Rail0, Rail1, GPIO, UART, USB CHx, etc.).

The MTM-IO-Serial provides three key functions via the BrainStem API interface: (1) a 4-port programmable USB 2.0 hub with ability to separate enable/disable capability for data lines and Vbus connections; (2) two adjustable IO rails; (3) two banks of UART interfaces and GPIO pins, both of which are associated with an adjustable IO rail.

The serial UARTs are placed into two groups of independent, software-adjustable voltage rails with associated digital input/output (DIO) pins with logic levels based on rail voltage. This configuration allows the MTM-IO-Serial module UARTs and GPIOs to be used to interface to DUTs with two different voltage planes.

Within the MTM platform architecture, the MTM-IO-Serial module can operate either independently or as a component in a larger network of MTM modules. Each MTM-IO-Serial 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 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-IO-Serial. The MTM-IO-Serial 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-IO-Serial, 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.

Absolute Maximum Ratings

Stresses beyond those listed under ABSOLUTE MAXIMUM RATINGS 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	4.0	A
Voltage to any IO pin	0.0	V_{supply}	V
Voltage to any I2C pin	0.0	6.0	V
Voltage to VBUS and USB D+/-	-0.5	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 use a well architected current-return path to ground for the targeted application.

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

Table 3: Recommended Operating Ratings

Block Diagram

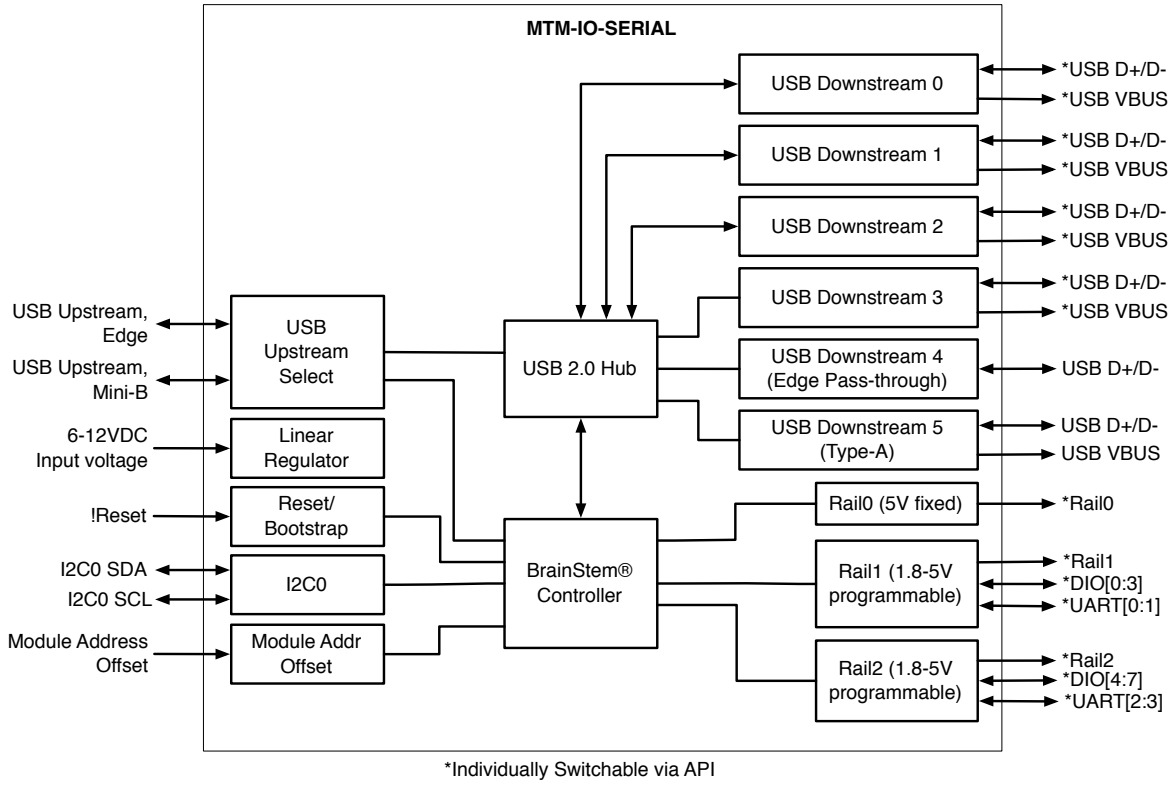


Figure 1: MTM-IO-Serial Block Diagram

Typical Performance Characteristics

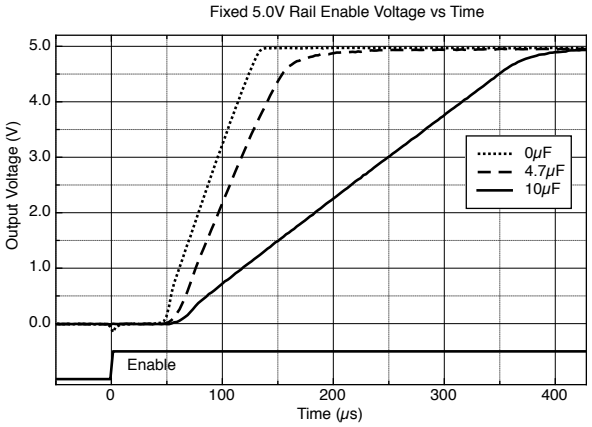
Values presented apply to the full operating temperature range.

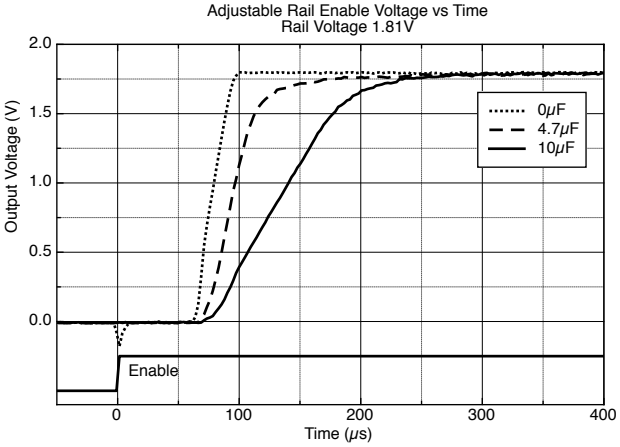
Parameter	Conditions/Notes	Minimum	Typical	Maximum	Units
Base Current Consumption, I_{supply}	V_{supply}=6V, Not Enumerated V_{supply}=6V, Enumerated V_{supply}=12V, Not Enumerated V_{supply}=12V, Enumerated	- - - -	135 343 127 232	- - - -	mA
Reset Low Threshold		-	1.2	-	V
I2C SDA, SCL Pins		-	3.3	-	V
UART Tx/Rx Logic High, V_{IH}		0.65 × V_{rail}	-	-	V
UART Tx/Rx Logic Low, V_{IL}		-	-	0.35 × V_{rail}	V
Digital Input Logic High, V_{IH}		0.65 × V_{rail}	-	-	V
Digital Input Logic Low, V_{IL}		-	-	0.35 × V_{rail}	V
Rail 0 Output Voltage, V_{rail0}	±2%	4.9	5.0	5.1	V
Rail 0 Switched Output Current	Current limited	100	150	200	mA
Rail 1-2 Output Voltage, V_{rail1}, V_{rail2}	Software controlled, ±2%	1.764	-	5.1V	V
Rail 1-2 Switch Output Current	Current Limited	100	200	250	mA
Rail 1-2 Voltage Error	$V_{rail} \geq$ 2.5V V_{rail} < 2.5V	- -	- -	1 2	%
Rail 1-2 Voltage	V_{rail} = 1.8V; ±2% V_{rail} = 3.3V; ±1% V_{rail} = 4.0V; ±1%	1.764 3.267 4.950	1.800 3.300 5.000	1.836 3.333 5.050	V
Digital Output Drive Current[1]	Output high; short to GND	-	20.0	30.0	mA
Digital Output Sink Current	Output low; short to V_{rail}	-	-10.0	-30.0	mA
Digital Output Short Duration	Output high	-	Infinite	-	hours
Digital Output Overvoltage	V_{supply} on pin	-	Infinite	-	hours

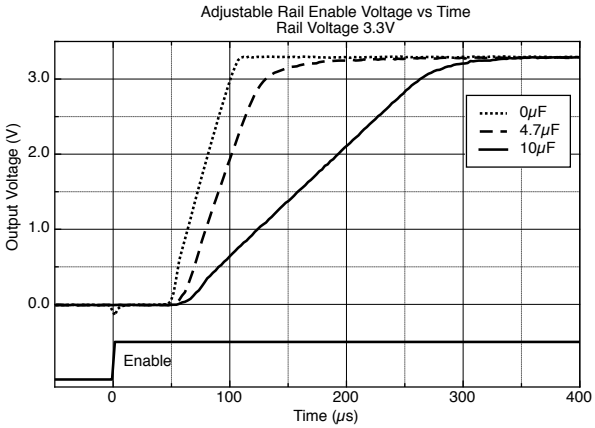
[1] It is not recommended to continuously apply more than V_{rail} to any DIO or UART pin

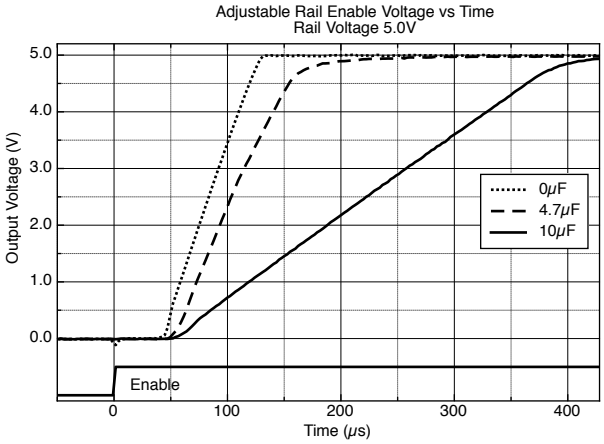
Digital Sample Rate[2]	Mac – OSX (C++) Mac – OSX (Python) Windows 10 (C++) Windows 10 (Python) Linux – 14.04 LTS (C++) Linux – 14.04 LTS (Python) Reflex	- - - - - - -	700 640 1000 1000 850 790 11000	1000 1000 1000 1000 1000 1000 -	Hz Hz Hz Hz Hz Hz Hz
Digital Output Jitter	Using Reflex only Reflex w/ BrainStem load	- -	- -	25 100	uS uS
USB V_{bus} current limit		500.0	500.0	800.0	mA

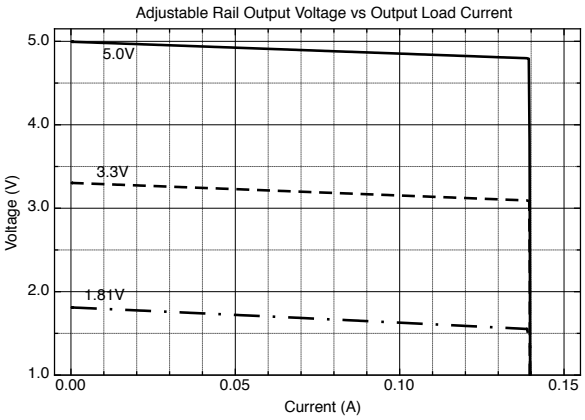
[2] Host dependent, test was done as a single instruction, subsequent instructions may affect performance. Measurements taken using BrainStem Library 2.3.2. The Nyquist frequency should be considered when referring to these values.

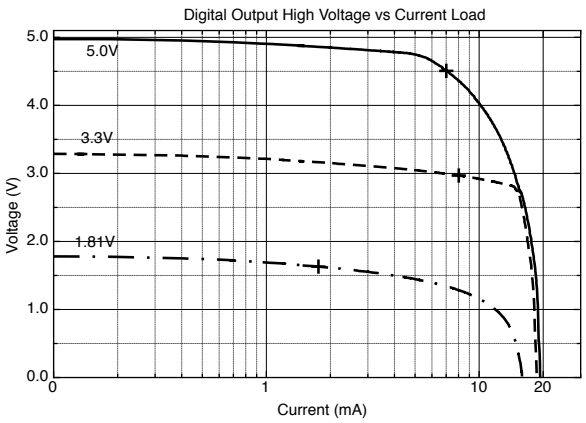

Fixed 5.0V Rail Enable Voltage vs Time
Output Voltage (V)
Time (µs)
0µF
4.7µF
10µF
Enable

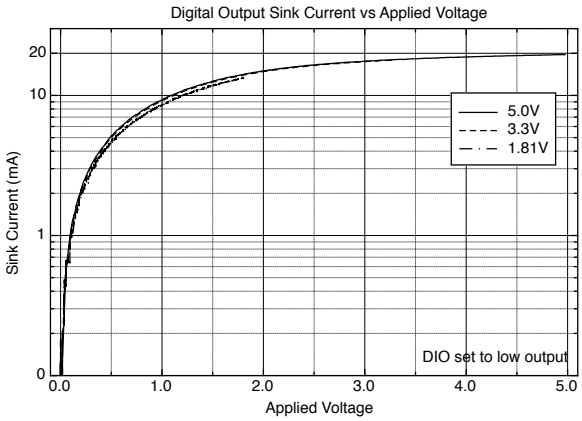

Adjustable Rail Enable Voltage vs Time
Rail Voltage 1.81V
Output Voltage (V)
Time (µs)
0µF
4.7µF
10µF
Enable

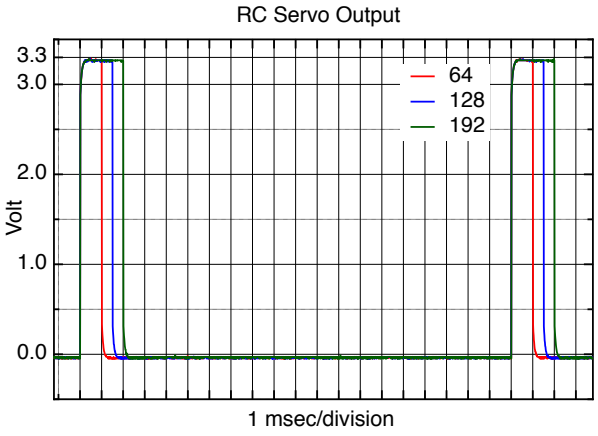

Adjustable Rail Enable Voltage vs Time
Rail Voltage 3.3V
Output Voltage (V)
Time (µs)
0µF
4.7µF
10µF
Enable

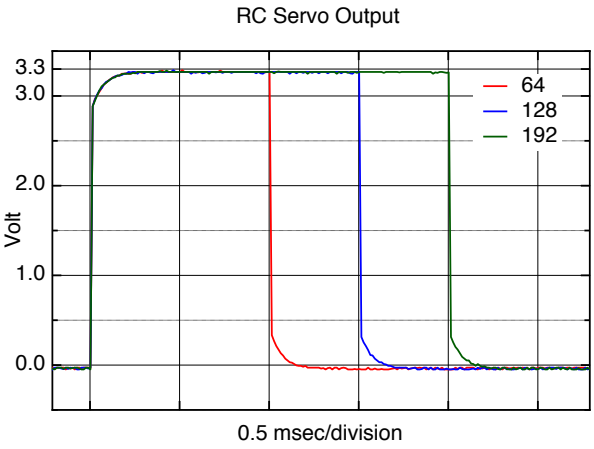

Adjustable Rail Enable Voltage vs Time
Rail Voltage 5.0V
Output Voltage (V)
Time (µs)
0µF
4.7µF
10µF
Enable


Adjustable Rail Output Voltage vs Output Load Current
Voltage (V)
Current (A)
5.0V
3.3V
1.81V


Digital Output High Voltage vs Current Load
Voltage (V)
Current (mA)
5.0V
3.3V
1.81V


Digital Output Sink Current vs Applied Voltage
Sink Current (mA)
20
10
1
0
0.0
1.0
2.0
3.0
4.0
5.0
Applied Voltage
5.0V
3.3V
1.81V
DIO set to low output

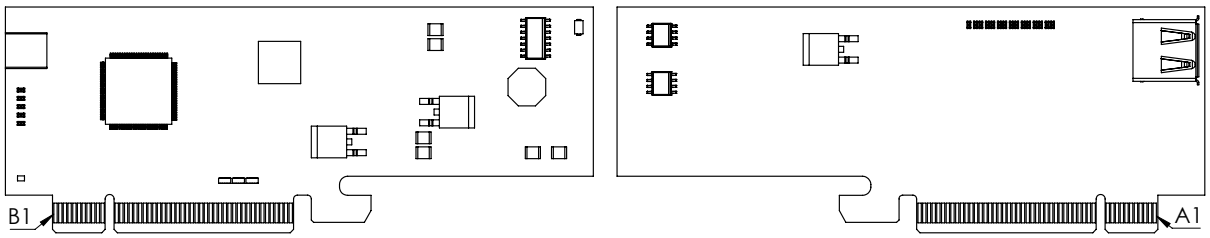

RC Servo Output
Volt
3.3
3.0
2.0
1.0
0.0
64
128
192
1 msec/division


RC Servo Output
Volt
3.3
3.0
2.0
1.0
0.0
64
128
192
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	UART0 Transmit
11	GND	11	UART0 Receive
12	Module Address Offset 0	12	Module Address Offset 2
13	Module Address Offset 1	13	Module Address Offset 3

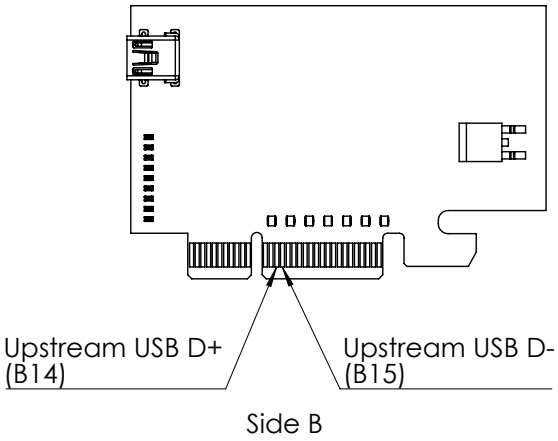
Pins Specific to MTM-IO-Serial

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 +
15	UART2 Tx	15	USB Upstream Data -
16	UART2 Rx	16	UART1 Tx
17	UART3 Tx	17	UART1 Rx
18	UART3 Rx	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	Reserved, Do Not Connect	24	Digital IO 6
25	Reserved, Do Not Connect	25	Digital IO 7
26	Reserved, Do Not Connect	26	Reserved, Do Not Connect
27	Reserved, Do Not Connect	27	Reserved, Do Not Connect
28	Reserved, Do Not Connect	28	Reserved, Do Not Connect
29	Reserved, Do Not Connect	29	Reserved, Do Not Connect
30	Reserved, Do Not Connect	30	Reserved, Do Not Connect
31	Rail 1 Output	31	Rail 2 Output
32	Reserved, Do Not Connect	32	Rail 0 Output
33	GND	33	USB2 V_{bus}
34	USB0 D+	34	USB2 D-
35	USB0 D-	35	USB2 D+
36	USB0 V_{bus}	36	GND
37	GND	37	USB3 V_{bus}
38	USB1 D+	38	USB3 D-
39	USB1 D-	39	USB3 D+
40	USB0 V_{bus}	40	GND
41	Reserved, Do Not Connect	41	Reserved, Do Not Connect
42	Reserved, Do Not Connect	42	USB Edge D-
43	Reserved, Do Not Connect	43	USB Edge D+

44	Reserved, Do Not Connect	44	GND
45	V_{supply}	45	GND
46	V_{supply}	46	GND
47	V_{supply}	47	GND
48	V_{supply}	48	GND
49	V_{supply}	49	GND

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-IO-Serial module utilizes a subset of BrainStem entity implementations that are specific to the hard- ware's capabilities. The table below details the BrainStem API entities and macros used to interface to the MTM-IO-Serial module. As a convenience for C and C++ developers, these macros are defined in aMTMIOSerial.h from the BrainStem development package. For Python development, the module MTM-IO-Serial class properly 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-IO-Serial module. The table below defines each 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`, `aErrUnimplemented`, or `aErrMode`) 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	8	aMTMIOSERIAL_MODULE _ADDRESS	See aMTMIOSerial.h
Entity Class Definitions:			
`digital` Entity Quantity	8	aMTMIOSERIAL_NUM_DIG	
`i2c` Entity Quantity	1		
`rail` Entity Quantity	3	aMTMIOSERIAL_NUM_RAILS	
5.0V Rail `(RAIL0)`	0	aMTMIOSERIAL_5VRAIL	
Adjustable Rail `(RAIL1)`	1	aMTMIOSERIAL_ADJRAIL1	
Adjustable Rail `(RAIL2)`	2	aMTMIOSERIAL_ADJRAIL2	
`UART` Entity Quantity	4	aMTMIOSERIAL_NUM_UART	
`USB` Entity Quantity	1	aMTMIOSERIAL_USB_NUM_CHANNELS	
`Store` Entity Quantity	2	aMTMBRAINSTEM _STORES	
	0	aSTORE_INTERNAL	
	1	aSTORE_RAM	
`system` Entity Quantity	1		
`timer` Entity Quantity	8	aMTMBRAINSTEM_NUM_TIMERS	

Table 4: MTM-IO-Serial Hardware and Software Default Values[3]

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

Capabilities and Interfaces

The MTM-IO-Serial 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-IO-Serial is described in the following sections; refer to Table 9: Supported MTM-IO-Serial 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[7].

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-IO-Serial 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	Boot Slot

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-IO-Serial has eight (8) digital input/outputs (DIO) controlled by the digital entity.

Digital inputs and outputs on the MTM-IO-Serial module are powerful as each digital input/output (DIO) is voltage adjustable via software and current limited. Each DIO pin has a current limiting circuit up to 20mA to be sourced or sinked. When all DIO in on one rail are configured as outputs, the whole rail (`DIO0-3` or `DIO4-7`) is limited in total to 20mA sourcing, while each pin can independently sink 20mA. A plot of expected output voltage as a function of current sourcing as well as a plot of current sinking ability as versus applied voltage is shown in Performance Characteristics. Since the current sourcing ability is tied to each pin bank's adjustable rail, the voltage output of the whole bank may be affected by one or more pin sourcing or sinking high levels of current. Note: it is not recommended to apply more than the configured rail voltage to any DIO pin.

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 output mode, setting the digital logic level high:

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

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

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

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

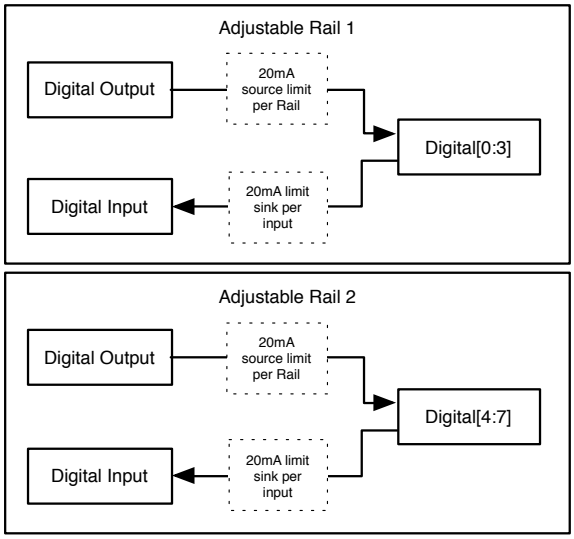


Figure 2: DIO and voltage rail grouping

The supported digital operating modes, and how they correspond to voltage rails, are shown in *Table 5: Digital operational modes*. DIO capabilities may change with future firmware revisions. Please contact Acroname to request specific pin capabilities.

Digital	Input	Output	Rail	HighZ	RCServo
DIO0	Yes	Yes	1	No	Input
DIO1	Yes	Yes	1	No	Input
DIO2	Yes	Yes	1	No	Input
DIO3	Yes	Yes	1	No	Input
DIO4	Yes	Yes	2	No	Output
DIO5	Yes	Yes	2	No	Output
DIO6	Yes	Yes	2	No	Output
DIO7	Yes	Yes	2	No	Output

Table 5: Digital operational modes

The DIOs are controlled with an array of the BrainStem digital class. The MTM-IO-Serial module implements a subset of the digital class for each DIO index. The implemented entity options for digital class entity index are summarized below.

I²C Entities

The MTM-IO-Serial includes access to a single I²C bus operating at a set 1Mbit/s rate.

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 330Ω pull-up resistors on the SDA and SCL lines which should allow for reliable bus communication upto 1Mbps (FastMode+).

RC Servo Entities

The MTM-IO-Serial 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)
```

Rail Entities

Rails allow other devices and peripherals to consume power from the MTM-IO-Serial module in a controlled fashion. Three (3) different rails are available for use in a variety of application: a fixed 5.0V rail (`RAIL0`) and 2

adjustable voltage rails (RAIL1, RAIL2). These rails are accessed through an array of BrainStem rail class entities. The MTM-IO-Serial module implements a subset of the BrainStem rail class for each of these rails. The implemented rail entity options for each entity index are summarized below.

All three rails can be switched on or off through using the API.

```
stem.rail[1].setEnableExternal(state)
```

Each rail is current limited in hardware to 150mA and an over-current condition will disable the power. Once the over-current condition is removed, the rail power can be turned back on by disabling and then re- enabling the rail.

RAIL0 can be configured to use two different regulation stages: linear (LDO) or switch-mode power supply (SMPS).

```
stem.rail[0].setOperationalMode(mode)
stem.rail[0].getOperationalMode(mode)
```

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

- 0 (operationalModeAuto) default
- 1 (operationalModeLinear)
- 2 (operationalModeSwitcher)

Auto configuration chooses the most appropriate mode based on input voltage and safe operating temperature ranges. The API can be used to read the actual operational mode state.

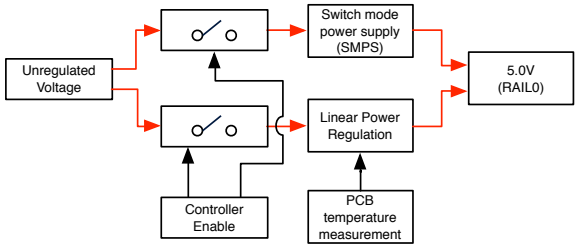
stem.rail[0].getOperationalState(mode)

RAIL1 and RAIL2 always use a linear regulator to generate their adjustable voltage.

Overvoltage conditions occur when a voltage above the rail's setpoint will also disable the rail's output. After the overvoltage condition is safely removed, the voltage rail will resume desired operation without any software intervention.

The 5.0V fixed regulation stage is unique as it can be configured to operate in either a switch mode power supply or linear regulation mode. For applications such as RF system testing, one might want to operate only in linear regulation mode to eliminate any potential EMI sources. While operating in linear mode, one must be aware of power dissipation through the linear regulation stage. A higher input voltage will result in higher power dissipation. When linear mode is desired and high current operation is desired it is recommended to run the input voltage close to the MTM-IO-Serial module's minimum input voltage. Switch mode power supply operation will allow a broader range of input voltages while maintaining high current demand limits. Default behavior is to auto-switch to switch mode power supply if an input voltage greater than 7.25V is applied.

A simplified block diagram for the 5.0V regulation stage shows the two different power paths.



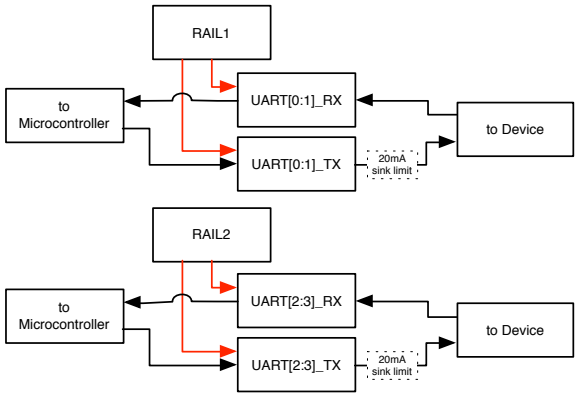
Printed circuit board (PCB) temperature can be monitored at the 5.0V rail (RAIL0) linear regulation stage. Reading this value is possible through the API. Temperature monitoring is also used internally to prevent the power regulation stage from overheating and self-pre- serving the power stage. If an over-temperature condition occurs, then the MTM-IO-Serial module will disable the linear regulator until safe operating temperatures are reached.

UART Entities

UART entities provide a mechanism to enable and disable UART data lines.

All of the UARTs that are passed down from the MTM- IO-SERIAL module can be turned on/off through software control. If a voltage is applied that is higher than the current RAIL voltage set point, each UART transmit line is current limited to 20mA sinking. Therefore, only a small amount of current will flow into the device, preventing any damage to the MTM-IO-Serial module's hardware.

UARTs are grouped in two with a corresponding volt- age rail. UART0 and UART1 share RAIL1's voltage reference. UART2 and UART3 share RAIL2's voltage reference.



When a UART is disabled by means of the command `UART`, all exposed UART data lines will be discharged by being pulled to Ground through a 10kΩ resistor.

USB Entities

The `usb` entity manages the software-controllable downstream USB 2.0 channels of the MTM-IO-Serial (there are also two non-software-controllable USB channels on the module, one through the edge connector and the other through the on-board type-A connector, which are always on), as well as the upstream USB connection mode. All downstream USB ports are configured as SDP (Standard Data Port).

USB Downstream

Each of the four software-controllable USB channels (ports 0-3) can be individually manipulated using the `usb` entity. The API individually controls port power, data, or both together.

```
stem.usb.setPowerEnable(port)          (just VBUS)
stem.usb.setPowerDisable(port)

stem.usb.setDataEnable(port)           (just D+/D-)
stem.usb.setDataDisable(port)

stem.usb.setPortEnable(port)           (both)
stem.usb.setPortDisable(port)
```

The *port* parameter is an integer that correlates to the software-controllable downstream channel (0-3).

USB Upstream

The MTM-IO-Serial has two upstream USB connection options – through the edge connector, or via the mini-B connector on the board itself. The upstream mode can be set or read using the `usb` entity.

```
stem.usb.setUpstreamMode(mode)
stem.usb.getUpstreamMode(mode)
```

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

- 0 (edge connector)
- 1 (mini-B connector)
- 2 (auto configuration) *default*

Auto configuration chooses the upstream connection based on the presence or absence of VBUS power at the mini-B connector; if VBUS is present, the the mini-B connector is used, otherwise the edge connector is used. The actual upstream connection in use can be read using the `usb` entity.

```
stem.usb.getUpstreamState(mode)
```

USB Hub Operational Mode

In addition to targeting individual downstream USB ports, a bit-mapped hub state interface is also available. This interface allows the reading or setting of all USB downstream ports in one functional call.

```
stem.usb.getHubMode(state)
stem.usb.setHubMode(state)
```

The value *state* must be a 32-bit word, defined as the following:

Bit	Hub Operational Mode Result Bitwise Description
0	USB Channel 0 USB Hi Speed Data Enabled
1	USB Channel 0 USB VBUS Enabled
2	USB Channel 1 USB Hi Speed Data Enabled
3	USB Channel 1 USB VBUS Enabled
4	USB Channel 2 USB Hi Speed Data Enabled
5	USB Channel 2 USB VBUS Enabled
6	USB Channel 3 USB Hi Speed Data Enabled
7	USB Channel 3 USB VBUS Enabled
8:31	Reserved

Table 6: Hub Operational Mode Result Bitwise Description

USB Port State

Each downstream port reports information regarding its operating state represented in bit-packed results from:

```
stem.usb.getPortState(channel,state)
```

where *channel* can be [0-3], and the value *status* is 32-bit word, defined as the following:

Bit	Hub State Result Bitwise Description
0	USB VBUS Enabled
1	USB2 Data Enabled
2:18	Reserved
19	USB Error Flag
20	USB2 Boost Enabled
21:31	Reserved

Table 7: Port State: Result Bitwise Description

USB Hub Error Status Mapping

It is possible to retrieve current error states for all downstream ports in a single 32-bit word. As only 4

downstream USB ports are available, the *bank* parameter should be set to 0 for the USBHub2x4.

```
stem.usb.getHubErrorStatus(bank=0,status)
```

Errors can be cleared on each individual channel (0, 1, 2 or 3) by calling the following method:

```
stem.usb.clearPortErrorStatus(channel)
```

Details about the hub error status 32-bit word are as follows:

Bit	Hub Error Status Result Bitwise Description
0	USB CH0 overcurrent limit exceeded[4]
1	USB CH0 VBUS back drive[5]
2	USB CH0 hub power system failure
3	USB CH0 VBUS discharge[6]
4:7	Reserved
8	USB CH1 overcurrent limit exceeded[4]
9	USB CH1 VBUS back drive[5]
10	USB CH1 hub power system failure
11	USB CH1 VBUS discharge[6]
12:15	Reserved
16	USB CH2 overcurrent limit exceeded[4]
17	USB CH2 VBUS back drive[5]
18	USB CH2 hub power system failure
19	USB CH2 VBUS discharge[6]
20:23	Reserved
24	USB CH3 overcurrent limit exceeded
25	USB CH3 VBUS back drive[5]
26	USB CH3 hub power system failure
27	USB CH3 VBUS discharge[6]
28:31	Reserved

Table 8: Hub Error Status Result Bitwise Description

MTM-IO-Serial 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-IO-Serial class options are shown below. Note that when using Entity classes with a single index (e.g., 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[0-7]	setConfiguration	
	getConfiguration	
	setState	
	getState	
rcservo[0-7]	setEnable	
	getEnable	
	setPosition	Index 4-7 only
	getPosition	
	setReverse	Index 4-7 only

[4] Current limit value is defined by API settings section on USB Downstream Channels

[5] VBUS exceeds 5.150V for longer than 5ms

[6] VBUS discharge circuitry is activated. At the end of the 200ms the hub will confirm that VBUS was discharged if the VBUS voltage is not below 0.750V

	getReverse	
i2c[0]	write	
	read	
	setPullup	Disabled by default. I2C communication requires a single set of pull-ups enabled across the bus.
usb[0]	setPortEnable	
	setPortDisable	
	setDataEnable	
	setDataDisable	
	setHiSpeedDataEnable	
	setHiSpeedDataDisable	
	setPowerEnable	
	setPowerDisable	
	getHubErrorStatus	
	clearPortErrorStatus	
	getSystemTemperature	In microcelsius
	setUpstreamMode	
	getUpstreamMode	
	getUpstreamState	
	getDownstreamDataSpeed	
	getHubMode	
	setHubMode	
	getPortState	
UART[0-3]	setEnable	
	setDisable	
rail[0]	setEnable	
	getEnable	
	getTemperature	
	setOperationalMode	
	getOperationalMode	
	getOperationalState	
	getVoltage	
rail[1-2]	setEnable	
	getEnable	

	setVoltage	In microvolts
	getVoltage	In microvolts
store[0-2]	getSlotState	
	loadSlot	
	unloadSlot	
	slotEnable	
	slotDisable	
	slotCapacity	
	slotSize	
system[0]	save	
	reset	
	setLED	
	getLED	
	setSleep	
	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 9: Supported MTM-IO-Serial BrainStem Entity API Methods[7]

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

LED Indicators

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

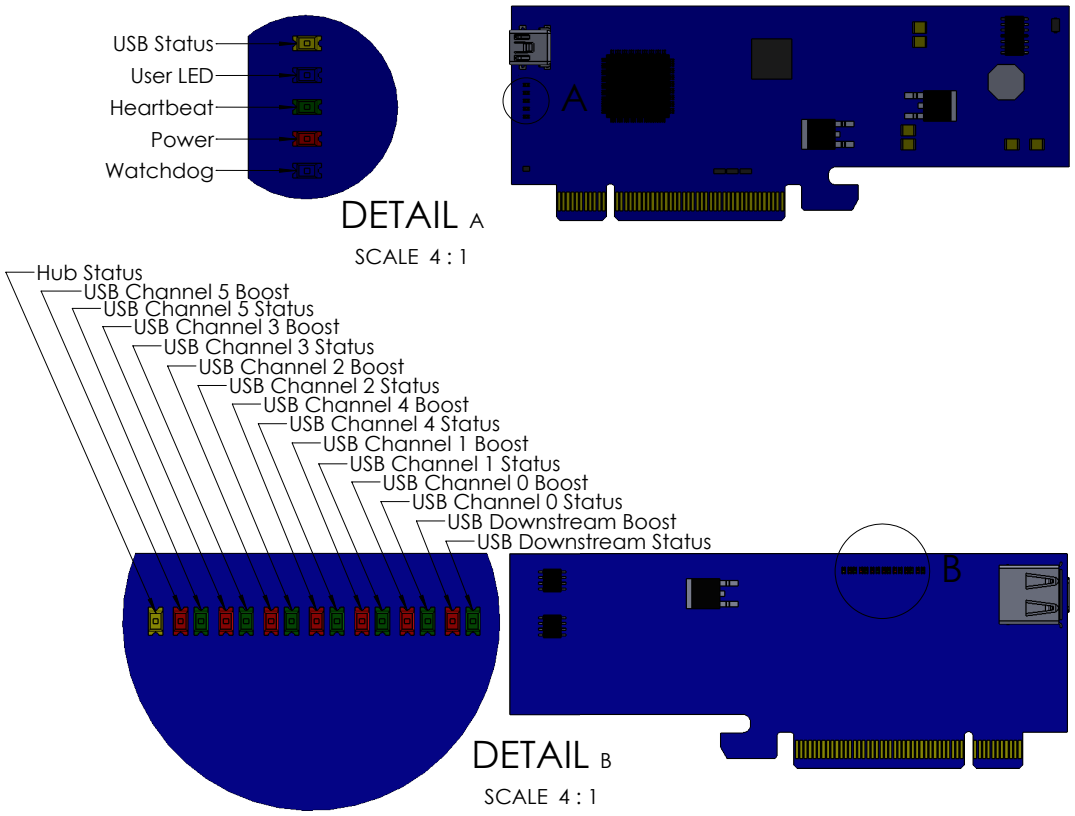


Figure 3: MTM-IO-Serial LED Indicators

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 10: PCI-Express Edge Connectors for MTM Products

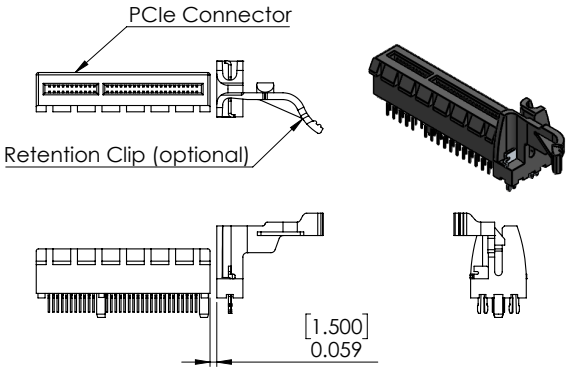


Figure 4: PCIe Vertical Connector with optional Retention Clip

MTM Edge Connector Specifications	Description
Contact Finish	Gold
Card Thickness	0.0625" [1.59mm]
Number of Rows	2
Number of Positions	Variable (see Table 10: PCI-Express Edge Connectors for MTM Products)
Pitch	0.039" (1.00mm)

Table 11: 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-IO-Serial product page's Downloads section.

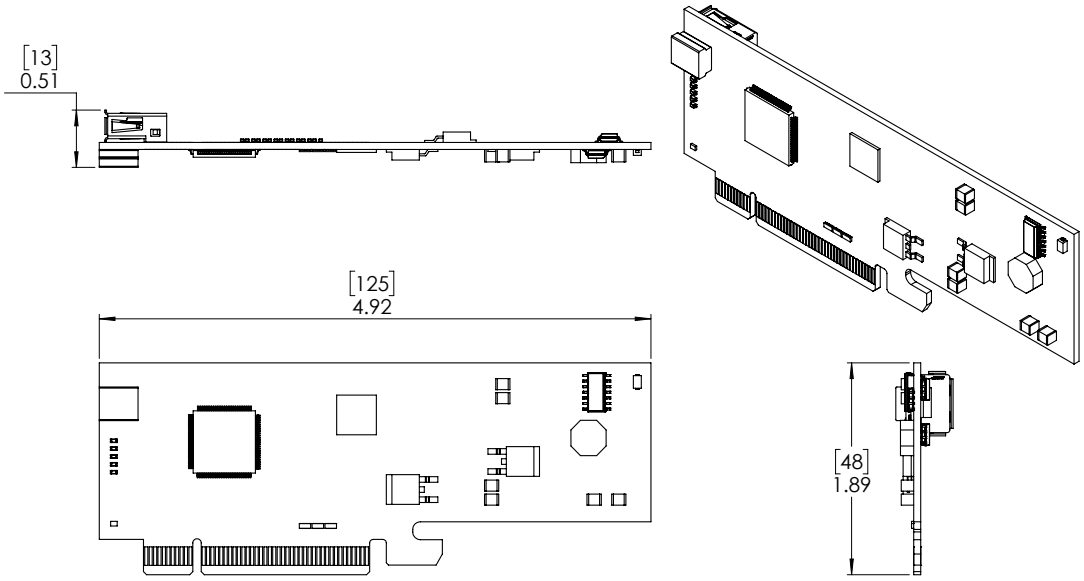


Figure 5: MTM-IO-Serial Mechanical

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

Document Revision History

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

Revision	Date	Engineer	Description
1.0	July 2015	MJK	Initial Revision
1.1	January 2016	JTD	Updated to BrainStem 2.2 firmware
1.2	September 2016	RMN	Formatting, Error checking, updates
1.3	October 2016	LCD	Updated Overview, Features, Description sections, added DO jitter
1.4	December 2016	JG	Clarified I2C pull-ups; update supported API calls
1.5	March 2017	JTD	Updated block diagram, USB maximum ratings
1.6	May 2017	RMN	Replaced references of MUX with UART.
1.7	April 2018	RMN	Swapped hubState for portState