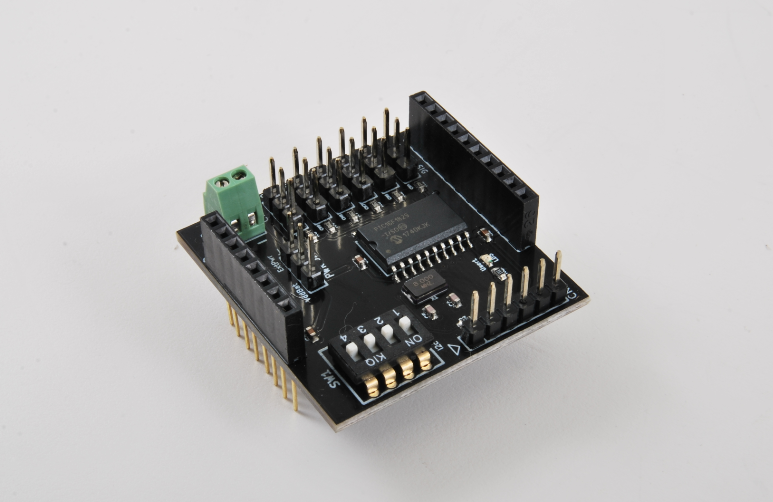
Mercury System

SB130



Servo Board - Product Datasheet

|  |  |
| --- | --- |
| Author | Francesco Ficili |
| Date | 21/10/2018 |
| Status | Released |

|  |  |  |  |
| --- | --- | --- | --- |
| Revision History | | | |
| Version | Date | Author | Changes |
| 1.0 | 21/10/2018 | Francesco Ficili | Initial Release. |
| 1.1 | 24/03/2019 | Francesco Ficili | Added command 0x71, 0x72 and 0x73. |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**SUMMARY**

[1. Introduction 4](#_Toc510204595)

[2. Block Diagram 6](#_Toc510204596)

[3. Hardware 8](#_Toc510204597)

[4. Pinouts 9](#_Toc510204598)

[Mercury Connector 9](#_Toc510204599)

[Programmer Connector 10](#_Toc510204600)

[Relay Connectors 10](#_Toc510204601)

[5. Command Set 12](#_Toc510204602)

[Specific Command Set 12](#_Toc510204603)

[Examples 13](#_Toc510204604)

[6. Technical Specifications 14](#_Toc510204605)

# Introduction

The Mercury System (MS in short) is a modular system for the development of connectivity and IoT applications. The system uses various type of electronic boards (logic unit, modems, slave board equipped with sensors and actuators, power boards...) and a complete SW framework to allow the realization of complex applications. Scalability, ease of use and modularity are key factors and are granted by the use of a heterogeneous set of components that allow to assemble the system like a construction made with LEGO© bricks.

The board set which composes the system is made up by the following “families”:

* **Base Board (BB):** It’s the “brain” of the system and contains the main logic unit as well as different communication buses and connector to interfaces the slaves. It also contains a simple power supply system and a recharge unit for a single LiPo cell (it can satisfy the power requirements of simpler systems). It can exist in different variants, depending on the employed microcontroller unit.
* **Modem Board (MB):** this one is the board that allow network connectivity. It can exist in different variant, depending on the network interface (GSM/GPRS, Wi-Fi, BT, Radio…). It’s interfaced to the Base Board with a dedicated serial line.
* **Power Board (PB):** it’s the board that allow to satisfy the particular power requirement of the system, when it’s necessary. They can be vary depending on the particular power requirement to satisfy (high power, solar harvesting, piezo harvesting, etc.).
* **Slave Board (SB):** these are the system’s peripherals, and they vary depending on the specific mounted sensor or actuator. Typical examples are SB with relay, temperature sensors, RGB LED controller, servo controller, accelerometer, etc. They communicate with the BB with I2C or UART and a dedicated command set.
* **Expansion Board (EB):** these are the board that allow planar connection of Mercury boards. There are variants which can contains Displays, battery socket, etc.
* **Brain-Less Board (BL):** these are the controller-less boards. They in general contain really simple sensor or actuators that don’t need the bus interface. There are meant as an alternative to slave boards for cost-sensitive applications.

Slave Boards and Modem Board are provided pre-programmed with a FW which implements a dedicated command set for a high-level management of the boards, while the Base Boards are provided with a SW framework which provides all the low-level services (operative system, device drivers, system services, etc.), leaving to the user only the development of application level logic. Moreover, the Base Board comes with an USB bootloader, so it can be programmed without the need of a flashing device.

Figure 1 shows a typical system connection:

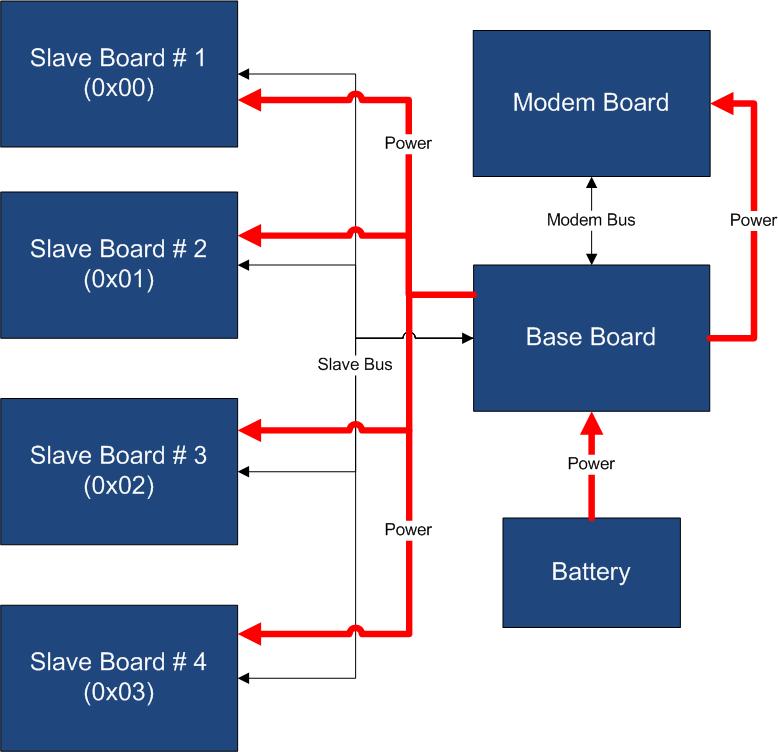


Figure 1 – Typical System Connection

Examples of application fields of MS are:

* Home automation System,
* IoT applications,
* Connectivity Applications,
* Monitoring and control Systems,
* Remote Control,
* Industrial Process control,
* Robotics applications,
* Test benches,
* Etc…

# Block Diagram

The SB130 is a 6-channel Servo board, able to generate and maintain an R/C servo control signal for each channel. Figure 2 shows the SB130 block diagram. The heart of the system is a PIC16F1829 8-bit RISC microcontroller, produced by Microchip Technology Inc.

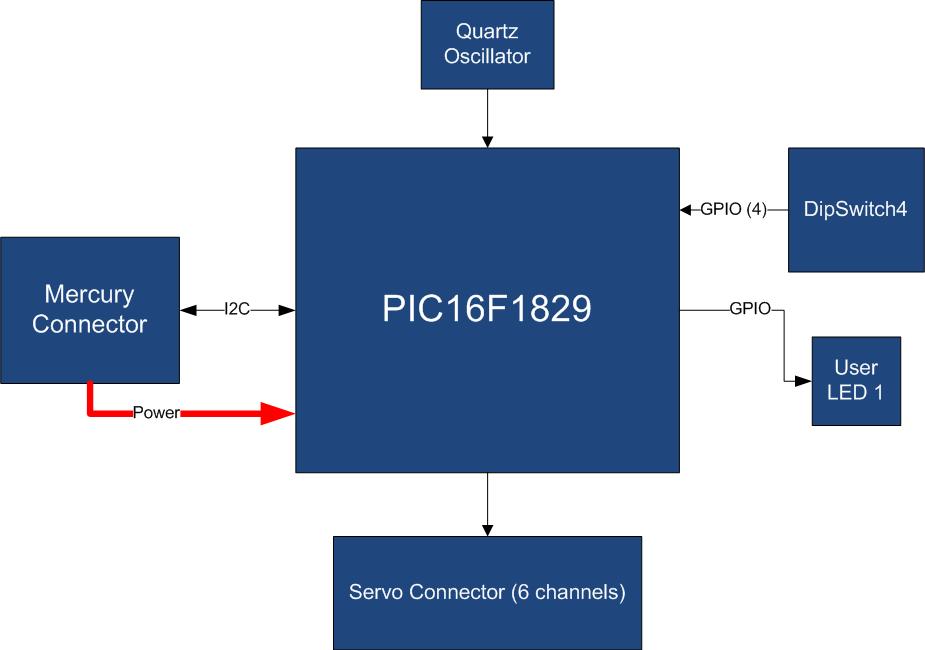


Figure 2 – Block Diagram

The main characteristics of the employed MCU are resumed in Table 1:

Table 1 – MCU characteristics

|  |  |
| --- | --- |
| Parameter Name | Description |
| Program Memory Type | Flash |
| Program Memory (KB) | 14 |
| CPU Speed (MIPS) | 8 |
| RAM Bytes | 1,024 |
| Data EEPROM (bytes) | 256 |
| Digital Communication Peripherals | 1-UART, 1-A/E/USART, 1-SPI, 1-I2C1-MSSP(SPI/I2C) |
| Capture/Compare/PWM Peripherals | 2 CCP, 2 ECCP |
| Timers | 4 x 8-bit, 1 x 16-bit |
| ADC | 12 ch, 10-bit |
| Comparators | 2 |
| Temperature Range I | -40 to 125 |
| Operating Voltage Range (V) | 1.8 to 5.5 |
| Pin Count | 20 |
| XLP | Yes |

The SB130 is connected to the BB by means of I2C bus. The address of the board could be dynamically set by means of a 4 positions dip switch, allowing up to 15 address values (address 0x00 is reserved for I2C general call broadcast addressing scheme).

Table 2 resumes the SB130 board main characteristics:

Table 2 – Board Characteristics

|  |  |  |
| --- | --- | --- |
| Parameter | Description | Notes |
| Board Type | Slave Board (SB) |  |
| Supported Bus | I2C |  |
| Addressing | Dip Switch 4 |  |
| Peripheral Description | 6 Servo Channels |  |

# Hardware

This section goes deeper in the HW details of SB130. Figure 3 depicts the most important components of the board:

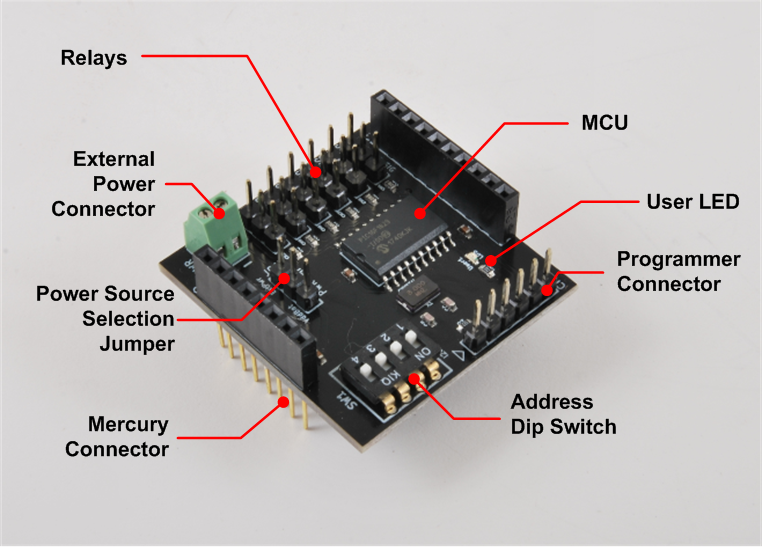


Figure 3 – Hardware Highlight

Table 3 provides a description of board’s main components:

Table 3 – Hardware characteristics

|  |  |
| --- | --- |
| Name | Description |
| User LED | Board User LED, by default it’s configured as heartbeat LED (periodic pulses). |
| Servo Outputs | Output Servo connectors. |
| Mercury Connector | Mercury connector used to interface the board with the others MS boards. |
| Address Dip Switch | Dip Switch to set the address of the board within the Mercury System. |
| MCU | PIC16F1829 main controller board. |
| Programmer Connector | PicKit 3 Microchip Programmer/debugger connector. It is directly connected to the MCU debug port, in order to allow advanced debugging and programming features, if needed. |
| Power Source Sel. Jumper | Jumper to select between internal (VddBat) and externally provided power source for Servo Channels. |
| External Power Connector | Screw-terminal connector for external power source. |

# Pinouts

This section highlights the pinouts of SB130 connectors.

## Mercury Connector

The Mercury Connector is the connector which interfaces the SB130 with the rest of Mercury System. The connector’s pinout is depicted in Figure 4 and Table 4 explains the meaning of each single pin (NC stands for “Not Connected”).

Table 4 – Mercury Connector Pinout

|  |  |  |
| --- | --- | --- |
| Pin Name | Pin Number | Description |
| VddBat | CN1 – 1  CN2 – 2 | This pin is connected to the main power source. |
| VddMcu | CN1 – 2 | This pin is connected to MCU regulated positive voltage reference (3,3V). |
| GND | CN1 – 3  CN2 – 1 | This pin is connected to the board reference voltage. |
| SDA | CN2 – 7 | This pin is connected to I2C SDA line (Data Line). |
| SCL | CN2 – 8 | This pin is connected to I2C SCL line (Clock Line). |

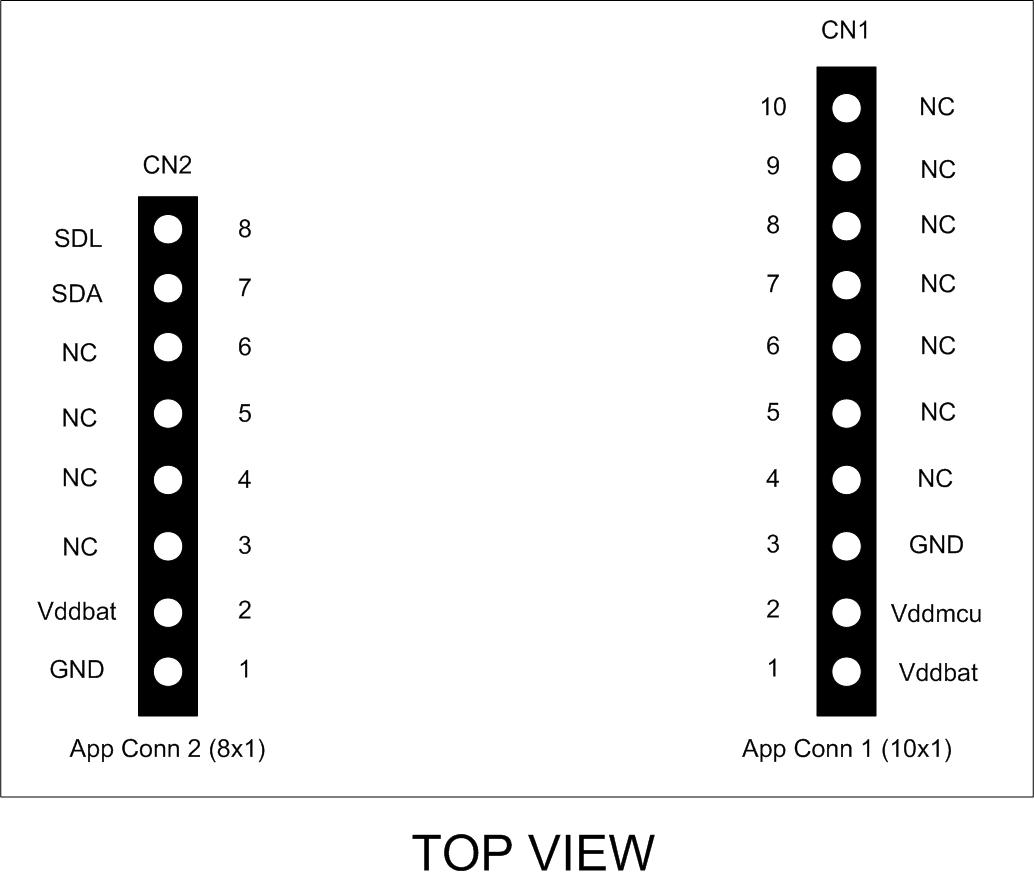


Figure 4 – Mercury Connector Pinout

## Programmer Connector

The Programmer Connector is the connector which allows to re-program the SB130 using Microchip Technology ICSP (In-Circuit Serial Programming) interface. The connector’s pinout is depicted in Figure 5 and Table 5 explains the meaning of each single pin (NC stands for “Not Connected”).

Table 5 – Programmer Connector Pinout

|  |  |  |
| --- | --- | --- |
| Pin Name | Pin Number | Description |
| MCLR | CN3 – 1 | Microcontroller Master Clear (RESET) pin. |
| Vdd | CN3 – 2 | Positive power supply reference. |
| GND | CN3 – 3 | Negative power supply reference. |
| PGD | CN3 – 4 | Program Data pin. |
| PGC | CN3 – 5 | Program Clock pin. |

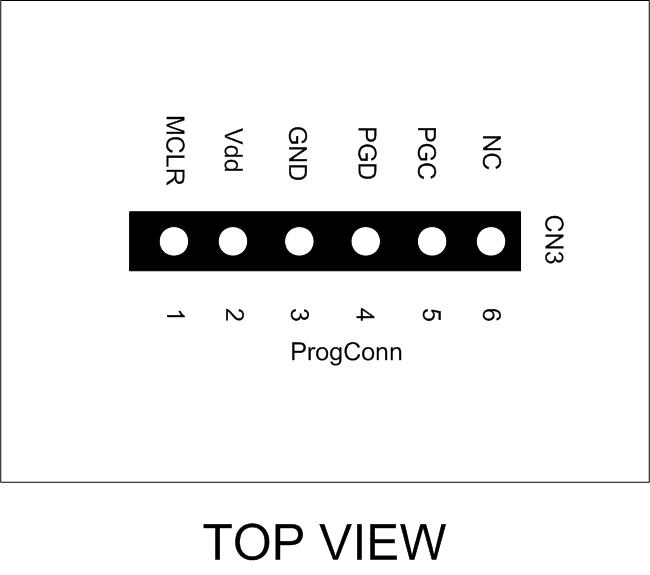


Figure 5 – Programmer Connector Pinout

## Servo Connectors

The Servo Connectors interface the SB130 servo channels. The connector’s pinout is depicted in Figure 6 and Table 6 explains the meaning of each single pin.

Table 6 – Servo Connectors pinout

|  |  |  |
| --- | --- | --- |
| Pin Name | Pin Number | Description |
| Sig | S6,5,4,3,2,1 – 1 | Servo signal. |
| Vdd | S6,5,4,3,2,1 – 2 | Positive power supply voltage. |
| GND | S6,5,4,3,2,1 – 3 | Power ground. |

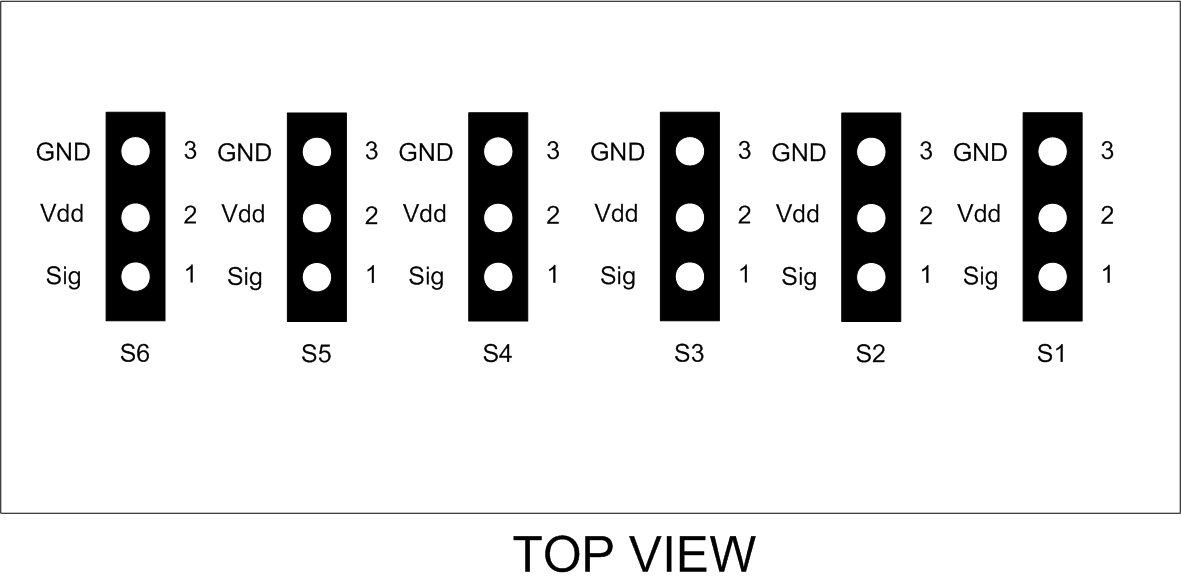


Figure 6 – Servo Connectors pinout

# Command Set

## Specific Command Set

The SB130 board supports both the MS Generic Command Set (see document MS\_Generic\_Command\_Set) and a set of specific commands (also called Specific Command Set).

Table 7 lists the SB130 Specific Command Set:

Table 7 – Command Set

|  |  |  |  |
| --- | --- | --- | --- |
| Code | Cmd Name | Parameters | Description |
| 0x51 | Set Servo 1 DC | ServoDC (1 byte) | Set the DC of Servo 1 as indicated by the parameter (0-100%). |
| 0x52 | Set Servo 2 DC | ServoDC (1 byte) | Set the DC of Servo 2 as indicated by the parameter (0-100%). |
| 0x53 | Set Servo 3 DC | ServoDC (1 byte) | Set the DC of Servo 3 as indicated by the parameter (0-100%). |
| 0x54 | Set Servo 4 DC | ServoDC (1 byte) | Set the DC of Servo 4 as indicated by the parameter (0-100%). |
| 0x55 | Set Servo 5 DC | ServoDC (1 byte) | Set the DC of Servo 5 as indicated by the parameter (0-100%). |
| 0x56 | Set Servo 6 DC | ServoDC (1 byte) | Set the DC of Servo 6 as indicated by the parameter (0-100%). |
| 0x61 | Request Servo 1 DC | None | Request the DC of Servo 1 (This command prepares 1 byte which represents the DC of servo in percentage). |
| 0x62 | Request Servo 1 DC | None | Request the DC of Servo 2 (This command prepares 1 byte which represents the DC of servo in percentage). |
| 0x63 | Request Servo 1 DC | None | Request the DC of Servo 3 (This command prepares 1 byte which represents the DC of servo in percentage). |
| 0x64 | Request Servo 1 DC | None | Request the DC of Servo 4 (This command prepares 1 byte which represents the DC of servo in percentage). |
| 0x65 | Request Servo 1 DC | None | Request the DC of Servo 5 (This command prepares 1 byte which represents the DC of servo in percentage). |
| 0x66 | Request Servo 1 DC | None | Request the DC of Servo 6 (This command prepares 1 byte which represents the DC of servo in percentage). |
| 0x71 | Set all Servos DC | Servos DC (6 byte) | Set the DC of all Servos as indicated by the parameter (0-100%). The DS are provided in sequence (1 byte per Servo from Servo 1 to Servo 6). |
| 0x72 | Set Servo DC bitmask | Bitmask (1 byte) Servos Pos. (Variable, depending on bitmask, from 1 to 6 bytes) | This command uses a bitmask to drive the position of a group of servos only. The command is composed as:  <Cmd>, <bitmask>, <ser. pos. 1> … <ser. pos. n>  The bitmask is a bitfield that select the servos to drive with the subsequent group of bytes. So, if for example, the user wants to drive servo 1 to 20% and 4 to 60%, the resulting command will be:  0x72 0x09 0x14 0x3C |
| 0x73 | Set Servos DC Bitmask | Bitmask (1 byte)  Position (1 byte) | This command is similar to the previous one, but it sets the same position for all the addressed servos. Thus the payload is fixed and composed by the bitmask itself and the global position. |

## Examples

Some examples of Specific Command Set usage are listed below:

1. Set Servo 1 DC to 50%: **[0x61] [0x32]**
2. Set Servo 3 DC to 75%: **[0x63] [0x4B]**
3. Request Servo 1 DC: **[0x61] + Read Operation on I2C bus**
4. Set servo 1 to 50% and 5 to 70%: **[0x72] [0x11] [0x32] [0x46]**
5. Set servo 1, 3 and 6 to 50%: **[0x73] [0x25] [0x32]**

# Technical Specifications

Table 8 resumes the board technical specifications:

Table 8 - Board Technical Specifications

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Parameter | Max | Typ | Min | Unit | Notes |
| Supply Voltage | 3.6 | 3.3 | 2.0 | V |  |
| Current Cons. (Normal) |  | 10 |  | uA |  |
| Current Cons. (Peak) |  | 1 |  | mA |  |
| Current Cons. (Low Power) |  | 100 |  | nA |  |
| Startup Time |  | 100 |  | mS |  |