# Mercury System

# SB810


## Proto Board - Product Datasheet

| | |
|---|---|
| Author | Francesco Ficili |
| Date | 28/10/2018 |
| Status | Released |

| Revision History | | | |
|---|---|---|---|
| Version | Date | Author | Changes |
| 1.0 | 28/10/2018 | Francesco Ficili | Initial Release. |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

# SUMMARY

# 1. 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:

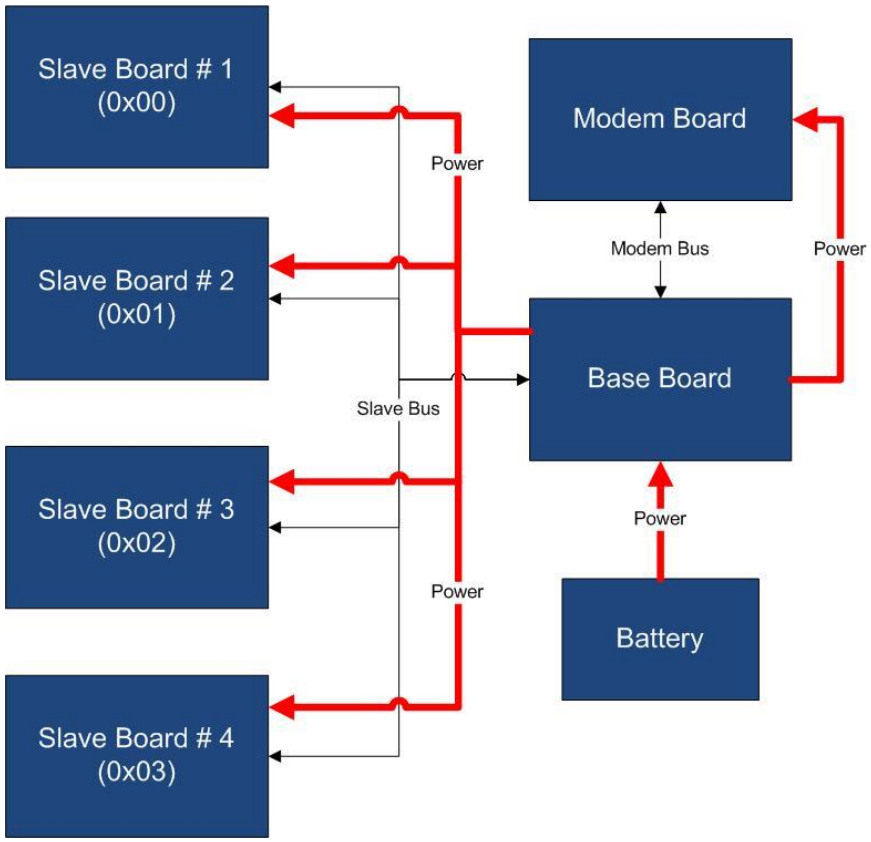



*Figure 1 - Typical System Connection*

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…

## 2. Block Diagram

The SB810 is a Mercury System Slave Proto Board. This board allow the user to make a custom slave device by soldering sensor/actuators on its prototyping area. The sensors and actuators mounted on the proto area can be then interfaced by means of 4 digital I/O and 4 analog channels available from the MCU. Figure 2 shows the SB110 block diagram. The heart of the system is a PIC16F1829 8-bit RISC microcontroller, produced by Microchip Technology Inc.

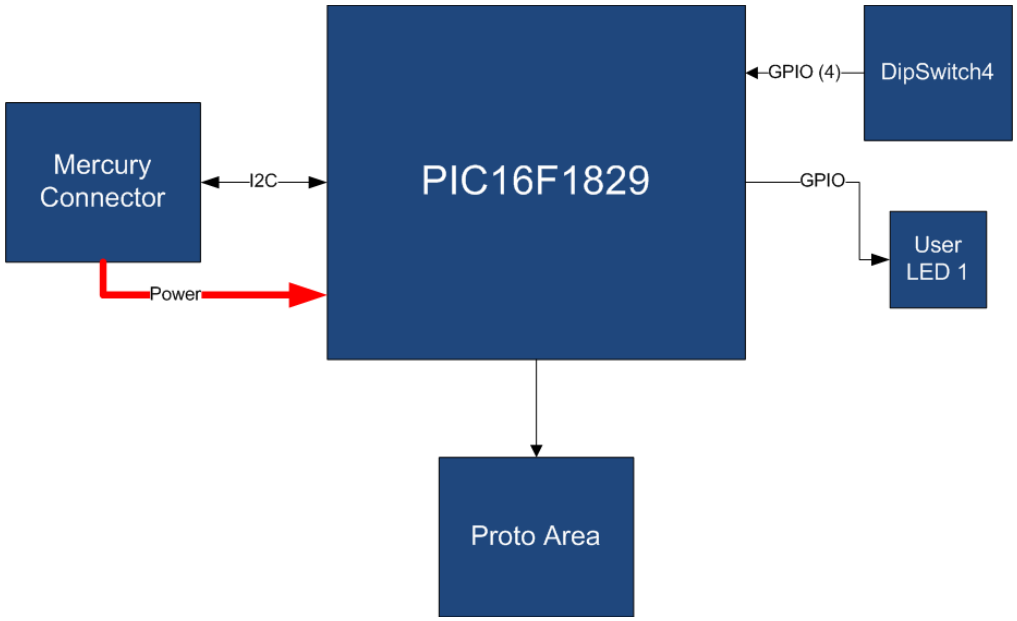



*Figure 2 - Block Diagram*

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 (C) | -40 to 125 |
| Operating Voltage Range (V) | 1.8 to 5.5 |

| | |
|---|---|
| Pin Count | 20 |
| XLP | Yes |

The SB810 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 SB810 board main characteristics:

*Table 2 – Board Characteristics*

| Parameter | Description | Notes |
|---|---|---|
| Board Type | Slave Board (SB) | |
| Supported Bus | I2C | |
| Addressing | Dip Switch 4 | |
| Peripheral Description | 4 Analog and 4 GPIO Channels + Prototyping Area | |

# 3. Hardware

This section goes deeper in the HW details of SB810. Figure 3 depicts the most important components of the board:

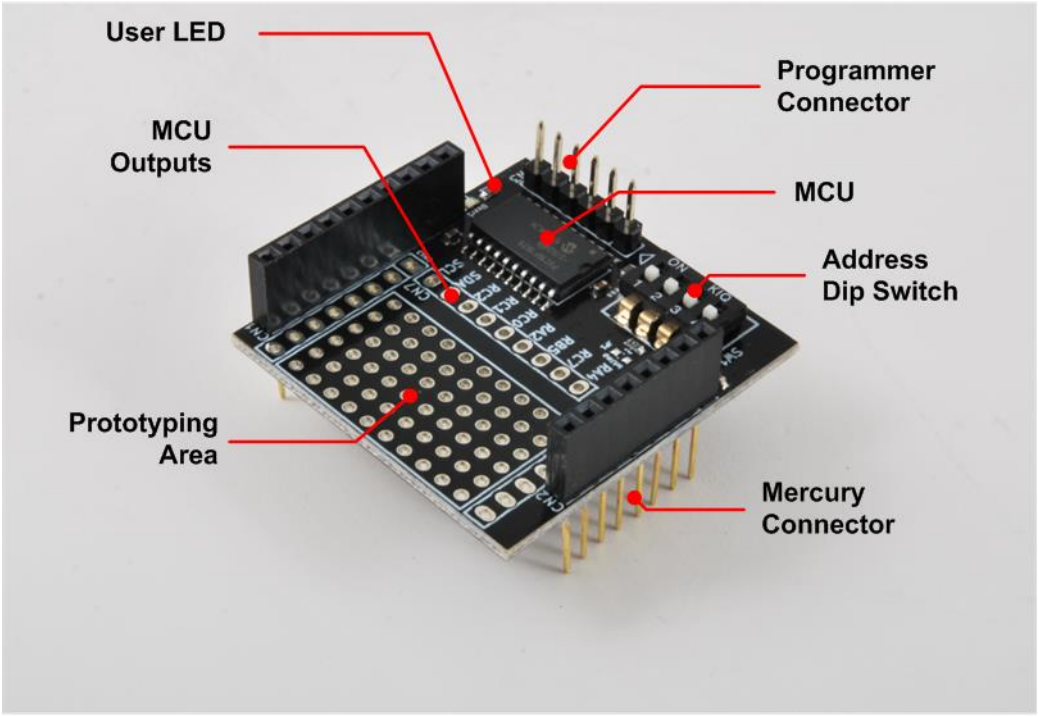



*Figure 3 - Hardware Highlight*

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). |
| MCU Outputs | 4 Digital I/O and 4 Analog channels. |
| Prototyping Area | Proto area available for sensor/actuator mounting. |
| 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. |

# 4. Pinouts

This section highlights the pinouts of SB110 connectors.

## Mercury Connector

The Mercury Connector is the connector which interfaces the SB110 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<br>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<br>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). |

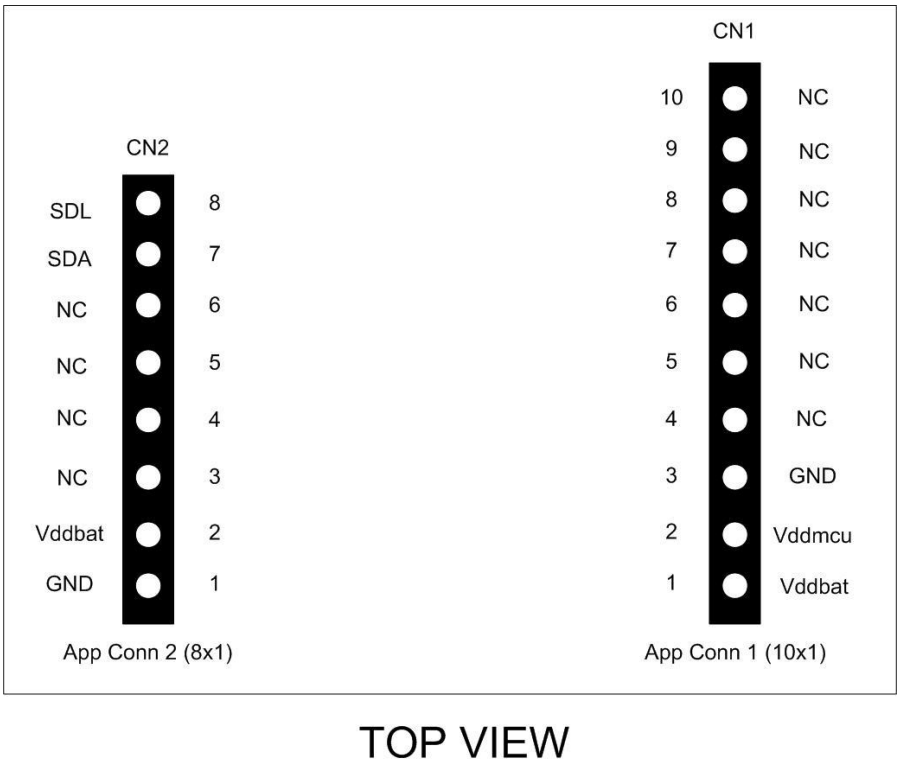


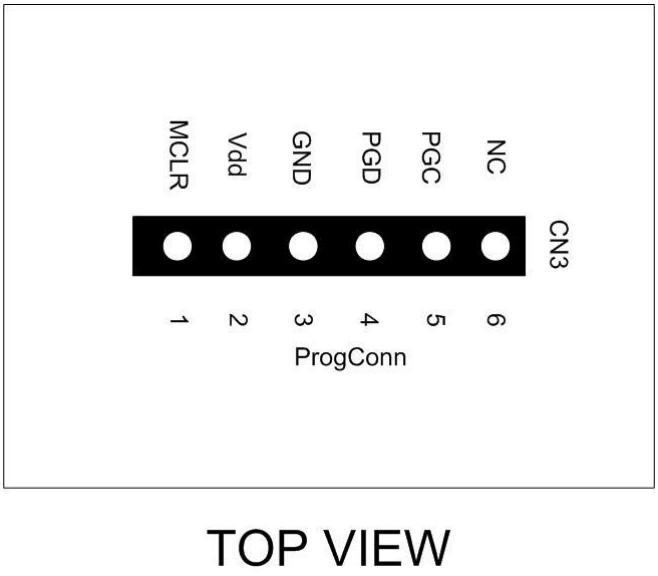

*Figure 4 - SB110 Mercury Connector Pinout*

## Programmer Connector

The Programmer Connector is the connector which allows to re-program the SB110 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. |




TOP VIEW

*Figure 5 - SB110 Programmer Connector Pinout*

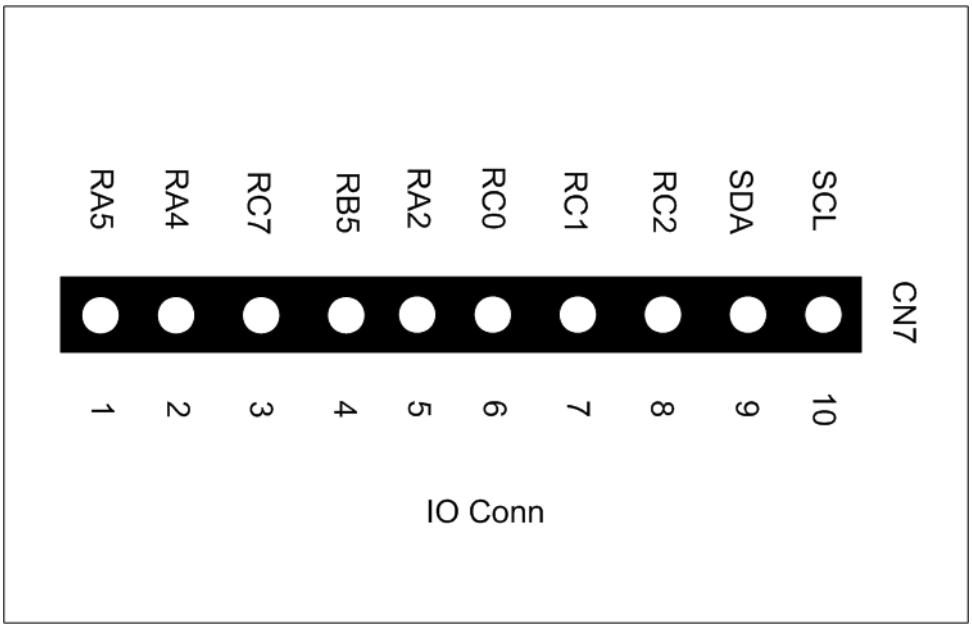
## MCU Output Connector

The MCU Output connector provides 4 digital I/O and 4 analog channels. The connector's pinout is depicted in Figure 6 and Table 6 explains the meaning of each single pin.

*Table 6 - MCU Output Connector pinout*

| Pin Name | Pin Number | Description |
|---|---|---|
| RA5 | CN7 – 1 | General Purpose Input Output Ch 3. |
| RA4 | CN7 – 2 | General Purpose Input Output Ch 2. |
| RC7 | CN7 – 3 | Analog Input Ch 3 |
| RB5 | CN7 – 4 | Analog Input Ch 2 |
| RA2 | CN7 – 5 | Analog Input Ch 1 |

| | | |
|---|---|---|
| RC0 | CN7 – 6 | Analog Input Ch 0 |
| RC1 | CN7 – 7 | General Purpose Input Output Ch 1. |
| RC2 | CN7 – 8 | General Purpose Input Output Ch 0. |
| SDA | CN7 – 9 | I2C data line. |
| SCL | CN7 – 10 | I2C clock line. |




TOP VIEW

*Figure 6 – MCU Output Connector pinout*

# 5. Command Set

## Specific Command Set

The SB810 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 SB810 Specific Command Set:

*Table 7 - Command Set*

| Code | Cmd Name | Parameters | Description |
|---|---|---|---|
| 0x50 | Set Dig Channel Dir | Ch (1byte)<br>ChDir (1byte) | Set the direction of digital channels (input or output). The first parameter of the command identifies the channel (0-3) and the second identifies the channel direction (0 = output, 1 = input). |
| 0x51 | Set Dig Channel Output status | Ch (1byte)<br>ChSts (1byte) | Set the direction of an output digital channels (ON or OFF). The first parameter of the command identifies the channel (0-3) and the second identifies the channel status (0 = OFF, 1 = ON). |
| 0x60 | Request Dig Channel 0 Status Raw | None | Request the current status of digital channel 0. This command prepares 1 byte which represents the status of the channel (raw), an I2C read request must be issued to read the prepared value. |
| 0x61 | Request Dig Channel 1 Status Raw | None | Request the current status of digital channel 1. This command prepares 1 byte which represents the status of the channel (raw), an I2C read request must be issued to read the prepared value. |
| 0x62 | Request Dig Channel 2 Status Raw | None | Request the current status of digital channel 2. This command prepares 1 byte which represents the status of the channel (raw), an I2C read request must be issued to read the prepared value. |
| 0x63 | Request Dig Channel 3 Status Raw | None | Request the current status of digital channel 3. This command prepares 1 byte which represents the status of the channel (raw), an I2C read request must be issued to read the prepared value. |
| 0x70 | Request Analog Channel 0 Status Raw | None | Request the current status of analog channel 0. This command prepares 2 bytes that represent the status of the channel (raw), an I2C read request must be issued to read the prepared value. |

| | | | |
|---|---|---|---|
| 0x71 | Request Analog Channel 1 Status Raw | None | Request the current status of analog channel 0. This command prepares 2 bytes that represent the status of the channel (raw), an I2C read request must be issued to read the prepared value. |
| 0x72 | Request Analog Channel 2 Status Raw | None | Request the current status of analog channel 0. This command prepares 2 bytes that represent the status of the channel (raw), an I2C read request must be issued to read the prepared value. |
| 0x73 | Request Analog Channel 3 Status Raw | None | Request the current status of analog channel 0. This command prepares 2 bytes that represent the status of the channel (raw), an I2C read request must be issued to read the prepared value. |
| 0x80 | Request Analog Channel 0 Status Ascii | None | Request the current status of analog channel 0 Ascii codified. This command prepares 4 bytes that represent the status of the channel (ascii), an I2C read request must be issued to read the prepared value. |
| 0x81 | Request Analog Channel 1 Status Ascii | None | Request the current status of analog channel 1 Ascii codified. This command prepares 4 bytes that represent the status of the channel (ascii), an I2C read request must be issued to read the prepared value. |
| 0x82 | Request Analog Channel 2 Status Ascii | None | Request the current status of analog channel 2 Ascii codified. This command prepares 4 bytes that represent the status of the channel (ascii), an I2C read request must be issued to read the prepared value. |
| 0x83 | Request Analog Channel 3 Status Ascii | None | Request the current status of analog channel 3 Ascii codified. This command prepares 4 bytes that represent the status of the channel (ascii), an I2C read request must be issued to read the prepared value. |

## Examples

Some examples of Specific Command Set usage are listed below:

1) Set of GPIO1 as output: **[0x50] [0x01] [0x00]**
2) Set of GPIO1 output On: **[0x51] [0x01] [0x01]**
3) Set of GPIO2 as input: **[0x50] [0x02] [0x01]**
4) Request GPIO2 read raw: **[0x61] + Read Operation on I2C bus**
5) Request AN1 read raw: **[0x70] + Read Operation on I2C bus**

6) Request AN3 read ascii: **[0x82] + Read Operation on I2C bus**

# 6. 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 | |