Doc# TRM-0922-01005, Rev 1.0.0

10555 Scripps Trl, San Diego, CA 92131
858-877-3485 858-408-9550
http://www.pulserain.com

PulseRain M10 – Serial Port

Technical Reference Manual

Sep, 2017

Table of Contents

REFERENCES **1**

1 INTRODUCTION **2**

2 HARDWARE **3**

2.1 ARCHITECTURE 3
2.2 ARDUINO COMPATIBLE PIN MAP 4
2.3 BAUD RATE 5
2.4 PORT LIST 5
2.5 PIN ASSIGNMENT 6
2.6 TIMER 6
2.7 REPOSITORY 7

3 SOFTWARE **8**

3.1 REGISTER DEFINITION 8
3.2 ADDRESS MAP 9
3.3 WORK FLOW 10
3.4 ARDUINO LIBRARY 11
3.4.1 APIs *11*
3.4.2 Examples *12*

References

1. MCS 51 Microcontroller Family User's Manual, Intel Corporation, Feb, 1994
2. The schematic of PulseRain M10 board, Doc# SH-0922-0039, Rev 1.0, 02/2017
3. FT232R USB UART IC Datasheet, Version 2.13, (Document No: FT 000053, Clearance No: FTDI# 38), Future Technology Devices International Limited, 2015
4. Tera Term Open Source Project (https://ttssh2.osdn.jp/index.html.en)

1 Introduction

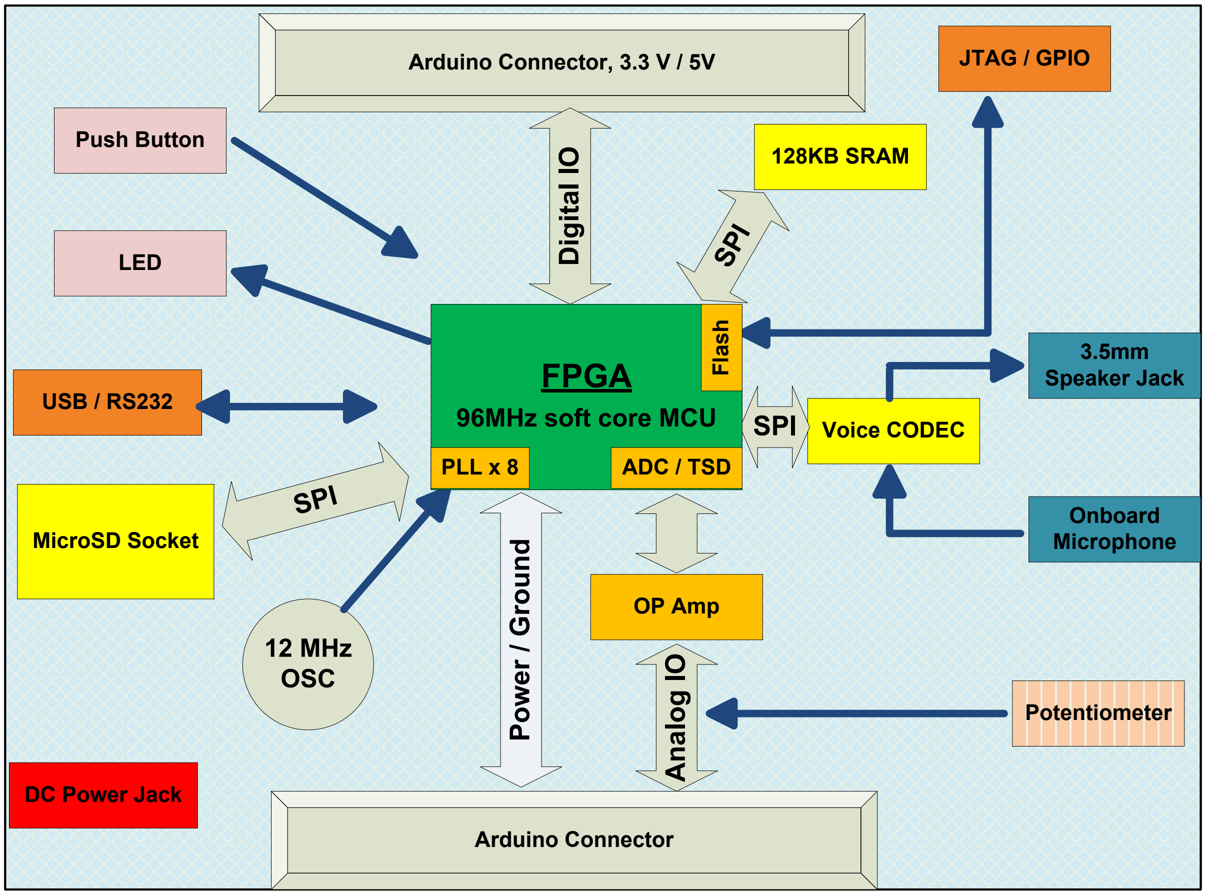

Figure 1-1 The Close View of M10

As shown Figure 1-1, the M10 board takes a distinctive technical approach by embedding an open source soft MCU core (96MHz) into an Intel MAX10 FPGA, while offering an Arduino compatible software interface and form factors. Among all the onboard peripherals, there is a FT232R chip that can bridge between USB and RS232 signals. And accordingly, PulseRain Technology has designed an open source controller for RS232 Serial communication.

In fact, there are two Serial Port Controllers inside the default FPGA image for M10 board. The main one is used to communicate with the host PC through the FT232R USB/RS232 bridge. And there is a second serial port (called SerialAUX, auxiliary serial port) that can be used to communicate with the shield (For example, SerialAUX can be used as a hardware UART for Sparkfun ESP8266 shield). This document serves as a technical reference manual for both of them.

2 Hardware

2.1 Architecture

As illustrated in Figure 2-1, the default FPGA image contains two Serial Controller. The baud rate of each Serial port is determined by the timer associated with it. (The FP51 MCU has two timers available). The receiver in each serial port also has a FIFO to prevent data loss when the processor is engaged in TX activity. (The FIFO size is 4 bytes for the main port, and 8 bytes for the auxiliary port.). The main serial port is connected to the onboard FT232R USB/RS232 bridge, while the auxiliary port is connected to the Arduino Connector.

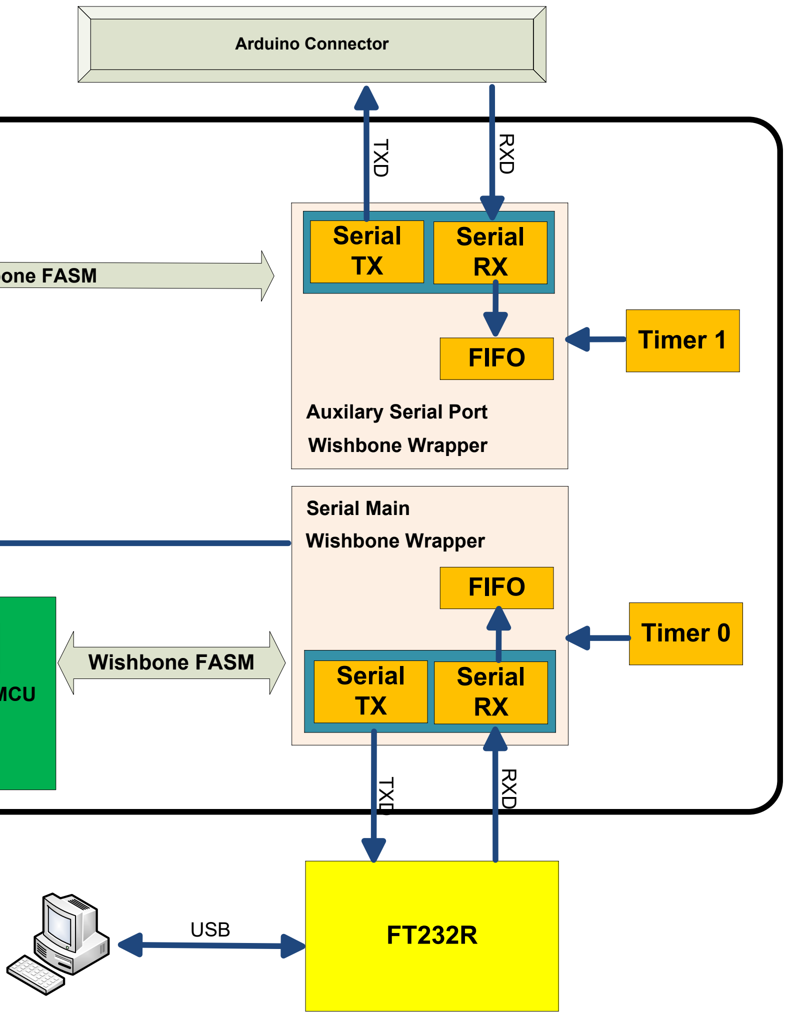

Figure 2-1 The Controller for Serial Port

2.2 Arduino Compatible Pin Map

The M10 board has a connector pin map that is compatible with Arduino UNO Rev 3. As mentioned in Section 2.1, the SerialAUX (auxiliary serial port) is connected to the Arduino Connector, and the pin map for its TXD/RXD is shown in Figure 2-2 (TXD/RXD marked by the red box at the lower right corner)

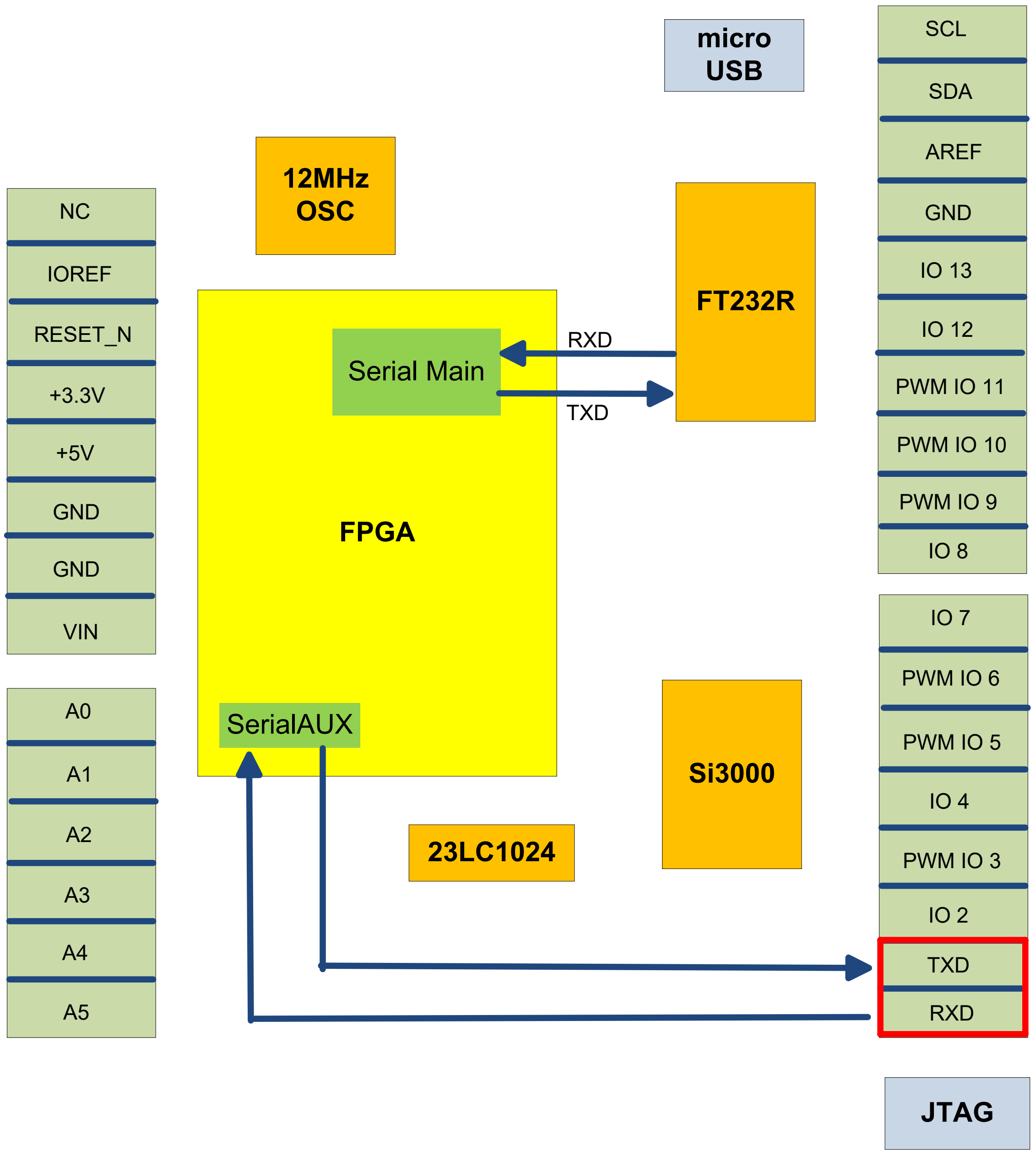

Figure 2-2 The Connector Pin Map for M10 Board

2.3 Baud Rate

As mentioned early, the baud rate of the Serial port is determined by the timer associated with it. And the maximum baud rate supported by the Serial Controller is 921600 bps. In fact, the constraint of 921600 bps does not come from the controller design itself. It is set so because 921600bps is the maximum baud rate supported by the Tera Term test utility (Ref [4]), and test also shows 921600 bps can work reliably with the M10 board. In others words, users are more than welcome to raise the maximum baud rate setting if necessary. (The maximum baud rate is defined by **MAX_UART_BAUD_RATE** in common_pkg.sv)

2.4 Port List

The port list of the Serial Controller is defined in Table 2-1.

Group Name	Signal Name	In/Out	Bit Width	Description
Clock / Reset	clk	Input	1	Clock input, 96MHz
	reset	Input	1	Asynchronous reset, active low
Host Interface	start_TX	Input	1	Set it to 0 if the controller is used for RX. When it is used for TX, this is the enable pulse for byte transmission.
	class_8051_unit_pulse	Input	1	This is supposed to be a baud rate pulse derived from 11.059MHz clock. The only purpose for this signal is to be backward compatible with the classic 8051 modes. Users can simply tie it to high.
	timer_trigger	Input	1	This is the baud rate pulse generated by the timer associated.
	SBUF_in	Input	8	byte data to be sent out
	SM	Input	3	Serial Mode. The valid values are: 3'b000: the baud rate will be based on class_8051_unit_pulse 3'b010: the baud rate will be based on timer_trigger
	REN	Input	1	Receiver Enable. Set it to 0 for TX, and set it to 1 for RX
	SBUF_out	Output	8	byte data received
	TI	Output	1	TX Interrupt. It is cleared when start_TX is active, and it will be set when the byte data is completed transmitted,
	RI	Output	1	RX Interrupt. It is cleared when start_RX is active, and it will be set when a new byte of data is received.
RS232 Interface	TXD	Output	1	RS232, Transmit Data Pin
	RXD	Input	1	RS232, Receive Data Pin

Table 2-1 Port List of Serial Controller

2.5 Pin Assignment

The pin assignment for the Serial Port is as following for the onboard FPGA (10M08SAE144C8G):

Signal Name	FPGA Pin Assignment (10M08SAE144C8G)	Description
UART_RXD	75	RXD for the main Serial Port
UART_TXD	79	TXD for the main Serial Port
UART_AUX_RXD	50	RXD for Serial AUX
UART_AUX_TXD	52	TXD for Serial AUX

Table 2-2 FPGA Pin Assignment

2.6 Timer

As illustrated in Figure 2-1, each Serial Port Controller relies on a timer for baud rate pulse. (Timer 1 for the main Serial Port and Timer 0 for SerialAUX). The timer here is an enhanced version of the classic 8051 timer. Currently it supports two modes: The 16-bit enhanced mode (2'b01) and the 8-bit auto load mode (2'b10). The 8-bit mode is only there for backward compatibility with the classic 8051. And the 16-bit mode is recommended for most scenarios. The port list of the timer core is shown in Table 2-3.

Group Name	Signal Name	In/Out	Bit Width	Description
Clock / Reset	clk	Input	1	Clock input, 96MHz
	reset	Input	1	Asynchronous reset, active low
Control Interface	event_pulse	Input	1	The event pulse for counter event. It is only used for backward compatibility with classic 8051. Tie it to 0 for 16-bit mode
	timer_pulse	Input	1	This signal is only used for classic 8051 timer that derives a pulse from 11.059MHz clock. The only purpose for this signal is to be backward compatible. Users can simply tie it to high for 16-bit mode.
	timer_mode	Input	2	Timer Mode. The supported modes are: 2'b01: 16-bit enhanced mode, recommended for most scenarios 2'b10: 8-bit auto load mode for back compatibility with classic 8051
	INTx	Input	2	External Interrupt and GATE. If GATE is not zero, the correspondent INTx will be used to start/stop timer.
	GATE	Input	2	
	C_T_bit	Input	1	Set it to 1 for counter event pulse. And set it to 0 for 16-bit mode.
	run	Input	1	Timer-enable. It is supposed to be controlled by TCON/TR bit
	TH_in	Input	8	Timer high byte. It is loaded when timer is enabled.
	TL_in	Input	8	Timer low byte. It is loaded when timer is enabled.
Status / Output	TH_out	Output	8	Timer high byte being used
	TL_out	Output	8	Timer low byte being used
	timer_trigger	Output	1	Timer pulse output
	TH_TL_update	Output	1	Set to 1 when timer is enabled

Table 2-3 Port List for Timer Core

2.7 Repository

The RTL code for Serial Port controller and its Wishbone wrapper is part of PulseRain Technology's RTL library, which can be found on GitHub:

https://github.com/PulseRain/PulseRain_rtl_lib

In particular, the UART folder contains all varieties of Serial Port Implementation. And the timer folder contains the design for FP51 timer.

3 Software

3.1 Register Definition

The Wishbone wrapper shown in Figure 2-1 contains all the registers to control the Serial Port. In a nutshell, the registers are defined as following:

- SCON (Serial Control Register)
  The bits for SCON are defined in Table 3-1:

Bits	R/W	Default	Name	Description
0	RO	0	RI	Value of RI (Receive Interrupt)
1	RO	0	TI	Value of TI (Transmit Interrupt)
3:2	N/A	0	RESERVED	RESERVED
4	RW	0	REN	Receiver Enable
7:5	RW	0	SM	Serial Mode

Table 3-1 Bit Map for SCON (Serial Control Register)

- SBUF (8 bit)
  When being read, this register will return the last byte received. And a write to this register will trigger a transmission on the bus.

In addition, the timer associated with the correspondent Serial Port has the following registers for configuration:

- TH (8 bit) and TL (8 bit)
  The timer higher byte and the timer lower byte.

- TMOD (Timer Mode Control)
  The bits for TMOD are defined in Table 3-2.

Bits	R/W	Default	Name	Description
1:0	RW	0	Timer0 Mode	Timer0 Mode. The supported modes are: 2'b01: 16-bit enhanced mode, recommended for most scenarios 2'b10: 8-bit auto load mode for back compatibility with classic 8051
2	RW	0	Timer 0 C/T bit	Set it to 1 for counter event pulse. And set it to 0 for 16-bit mode.
3	RW	0	Timer 0 GATE	If set to 1, INT0 will be used to start/stop timer0
5:4	RW	0	Timer1 Mode	Timer1 Mode. The supported modes are: 2'b01: 16-bit enhanced mode, recommended for most scenarios 2'b10: 8-bit auto load mode for back compatibility with classic 8051
6	RW	0	Timer 1 C/T bit	Set it to 1 for counter event pulse. And set it to 0 for 16-bit mode.
7	RW	0	Timer 1 GATE	If set to 1, INT1 will be used to start/stop timer1

Table 3-2 Bit Map for TMOD (Timer Mode Control Register)

- TCON (Timer Control)

  The bits for TCON are defined in Table 3-3.

Bits	R/W	Default	Name	Description
3:2	N/A	0	RESERVED	RESERVED
4	RW	0	TR0	Enable for Timer 0
5	RW	0	TF0	Timer 0 overflow flag
6	RW	0	TR1	Enable for Timer 1
7	RW	0	TF1	Timer 1 overflow flag

Table 3-3 Bit Map for TCON (Timer Control Register)

3.2 Address Map

The registers defined in Section 3.1 are mapped into MCU's address space, as shown in Table 3-4, for both the main Serial Port and the auxiliary one.

Address	Register Name
0x98	SCON
0x99	SBUF
0x9A	SCON_AUX
0x9B	SBUF_AUX

Table 3-4 Address Definition for Serial Port

Address	Register Name
0x88	TCON
0x89	TMOD
0x8A	TL0
0x8B	TL1
0x8C	TH0
0x8D	TH1

Table 3-5 Address Definition for Timer

And for SCON, its bits can also be accessed individually with bit address, just like the traditional 8051 will do (Ref [1]). These bit addresses are defined in Table 3-6:

Bit Address	Bit Name
0x98	RI
0x99	TI
0x9C	REN
0x9D – 0x9F	SM2 – SM0

Table 3-6 Bit Address Definition for SCON

The similar bit address applies to TCON as well, as shown in Table 3-7:

Bit Address	Bit Name
0x8C	TR0
0x8D	TF0
0x8E	TR1
0x8F	TF1

Table 3-7 Bit Address Definition for TCON

3.3 Work Flow

3.3.1 Setup Timer

Before using the Serial Port Controller, the timer associated with it has to be initialized for the proper baud rate. In the default FPGA image, timer 1 is associated the main Serial Port while timer 0 is associated with the auxiliary Serial Port.

For example, if 115200 bps is the targeted baud rate, and timer can be configured with a 96MHz CPU clock:

96e6 / 115200 = 833; 65536 – 833 = 64703 = 0xFCBF.

So the timer can be set with TL = 0xBF and TH = 0xFC

After that, the correspondent part in TMOD can be set to 16-bit mode, and set TR to 1 to enable the timer.

3.3.2 Transmit a Byte

To transmit a byte, do the following:

1. Write the byte data to SBUF.
2. Wait until TI becomes 1.
3. Set TI to 0.

3.3.3 Receive a Byte

The work flow to receive a byte from the serial port is shown in Figure 3-1.

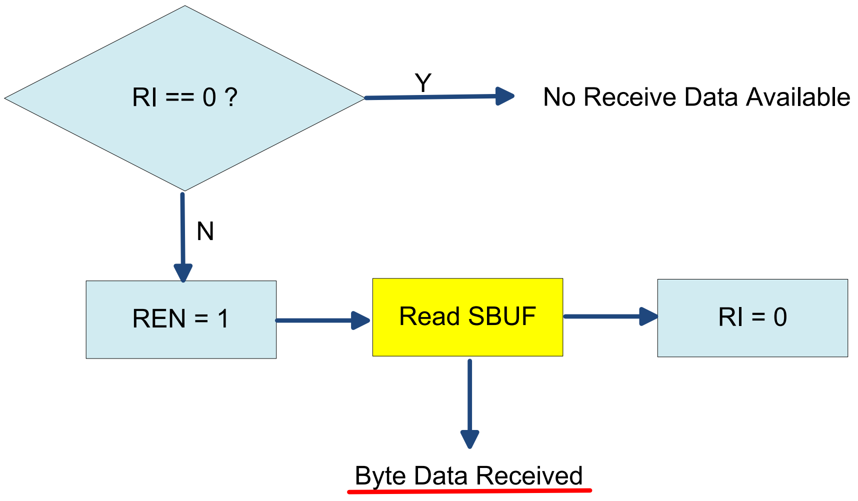

Figure 3-1 The Work Flow to Receive Byte Data

3.4 Arduino Library

For the main Serial Port, its APIs are embedded with the core module. No need to include extra library. However, for auxiliary serial port, the SerialAUX library needs to be included before it can be used.

3.4.1 APIs

- *void begin (uint32_t rate)*

  Call this function to setup the Serial Port with the specified baud rate. (The maximum valid value is 921600).

- *uint8_t available ()*

  Call this function to check if there is any data available in the receiving FIFO

  Return Value:

  1: There is data available
  0: Receiving FIFO is empty

- *void print (int32_t num, uint8_t fmt = DEC)*

  Call this function to print out an integer in specified format. The valid formats are: DEC (Decimal), OCT(octal) and HEX(hexadecimal). And the default format is DEC.

- *void println (int32_t num, uint8_t fmt = DEC)*
  This function does the same as that of *print ()*, plus it will append a carriage return (new line '\n') at the end.

- *uint8_t read ()*
  Call this function to read one byte from the Serial Port

- *uint8_t readBytes (uint8_t* buf, uint16_t length)*
  Call this function to read data with specified length

  Parameters:

buf	: the pointer to the buffer to be filled
length:	the length to be read, in number of bytes

  Return Value:

  0x0: read is successful
  0xff: read time out

- *void write (uint8* buf)*
- *void write (uint8* buf, uint16_t length)*
  Call these two functions to print out a string to the Serial Port. If *length* is not given, the *buf* must be a string that ends with a null (a byte of value zero).

3.4.2 Examples

To further facilitate the software development, the following examples can be referenced:

- *aux_uart*
  This example comes with M10SeiralAUX library. To use this example, connect the AUX serial port (pin 1 - RX, pin 1 - TX, on Arduino UNO Rev 3) to a PC through a FT232RL breakout board (such as the SparkFun USB to Serial Breakout - FT232RL).

  On Windows, open a Tera Term (Ref [4]) and set it to the correspondent **AUX** Serial USB/COM port. And connect Arduino IDE to the M10 board through its main USB/COM port (a different one from the AUX Serial USB/COM port).

  Open the Serial Monitor. Set it to be 115200 bps.
  Set the Tera Term to be 921600 bps.

  Type something in the Tera Term Windows, and the same characters will show up in the Serial Monitor if everything runs ok.