**Doc# TRM-0922-01009, Rev 1.0.0**

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

# PulseRain M10 – PWM

## Technical Reference Manual

Sep, 2017

This page is intentionally left blank.

# Table of Contents

## References

1. The schematic of PulseRain M10 board, Doc# SH-0922-0039, Rev 1.0, 02/2017
2. TXS0108E 8-Bit Bi-directional, Level-Shifting, Voltage Translator for Open-Drain and Push-Pull Application (SCES642D), Texas Instruments, Feb 2016
3. L298 Dual Full-Bridge Driver, STMicroelectronics, 2000

# 1 Introduction

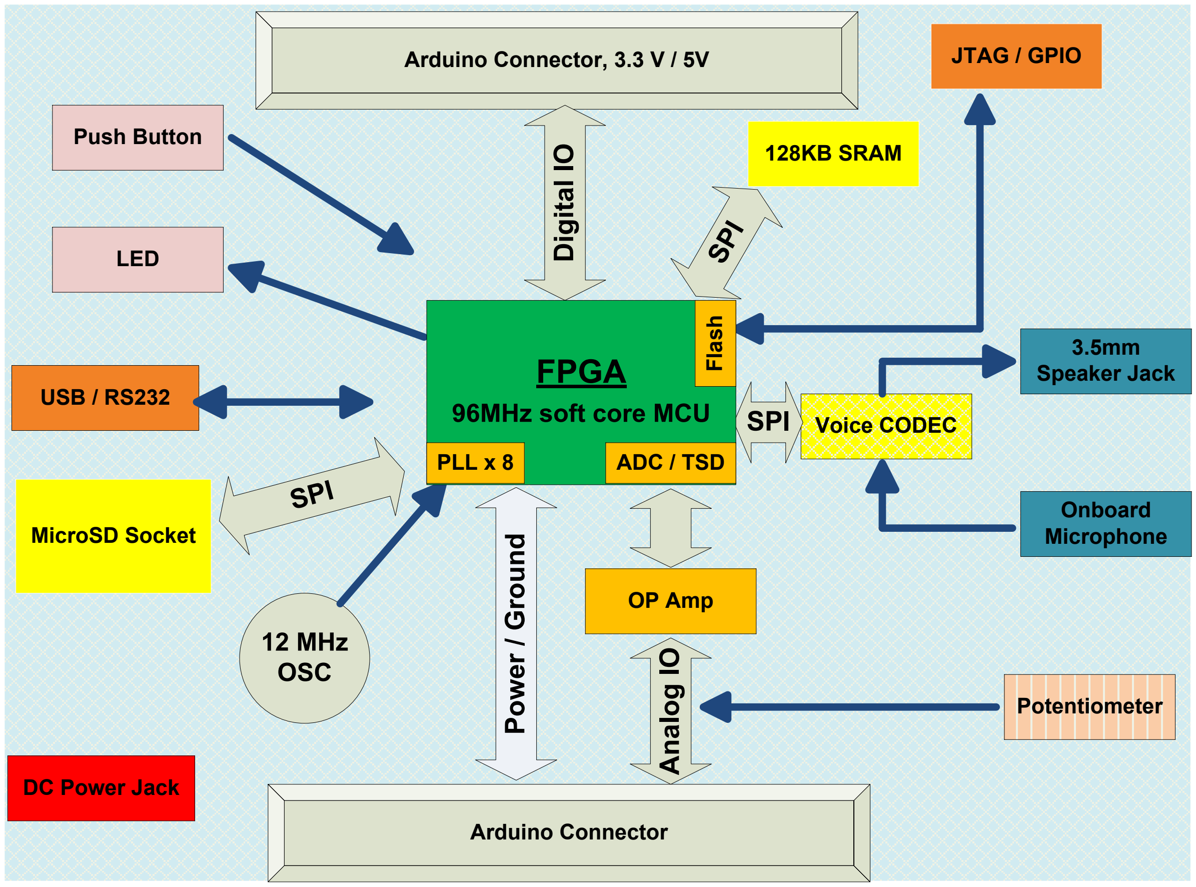

**Figure 1-1 The Whole Picture**

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 IOs of the Arduino Connector, 6 of which are set to function as PWM (Pulse Width Modulation) by default. And the pin map of those 6 PWM IOS are illustrated in Figure 1-2. Accordingly, PulseRain Technology has designed an open source controller and software library to drive those PWM pins. This document serves as the technical reference manual for this matter.

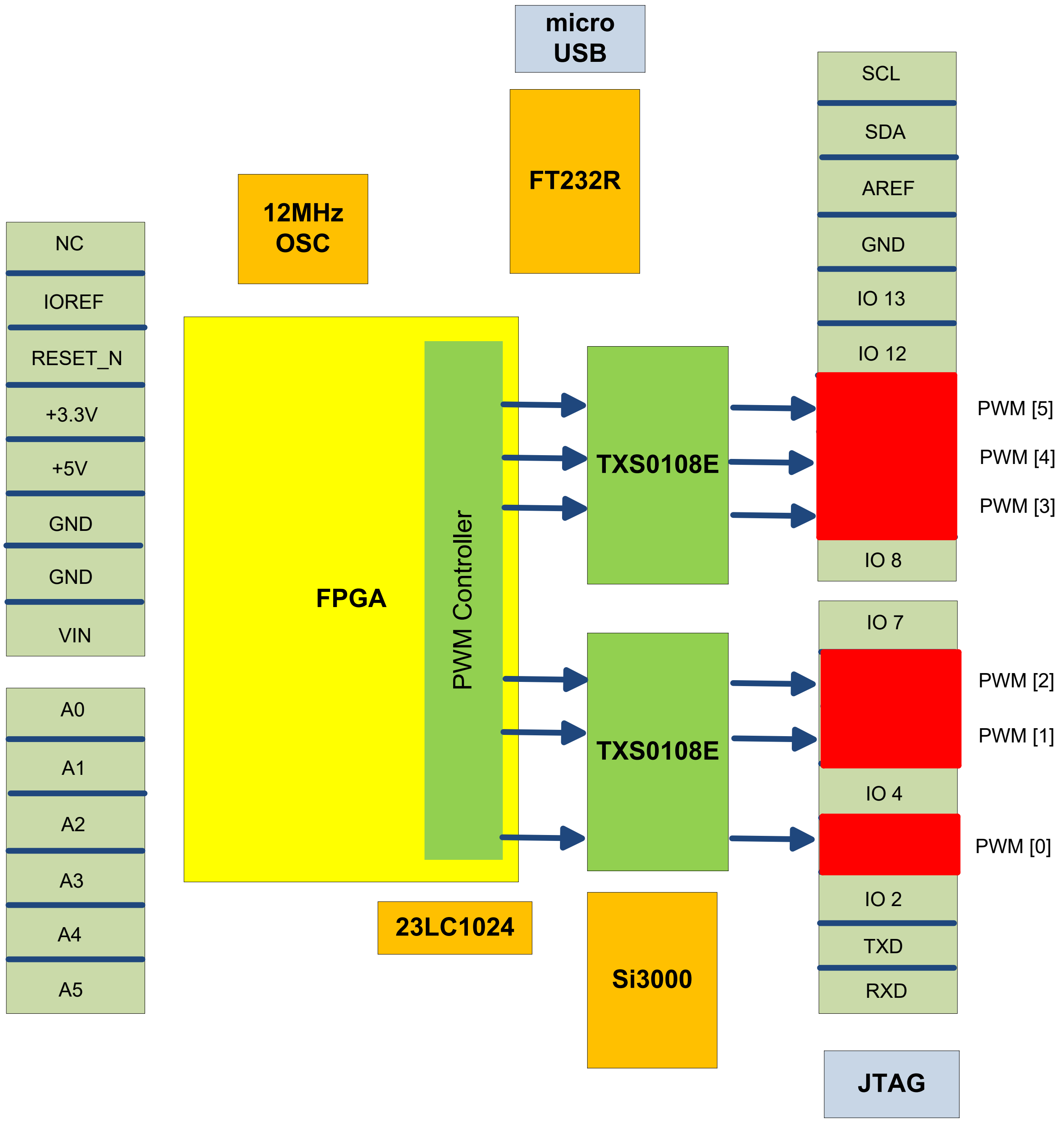

**Figure 1-2 Pin Map for PWM**

# 2 Hardware

## 2.1 Architecture

The PWM controller comprises 6 PWM cores and one Wishbone wrapper, as shown in Figure 2-1. Since the PWM is output only, interrupt is not supported by the PWM controller.

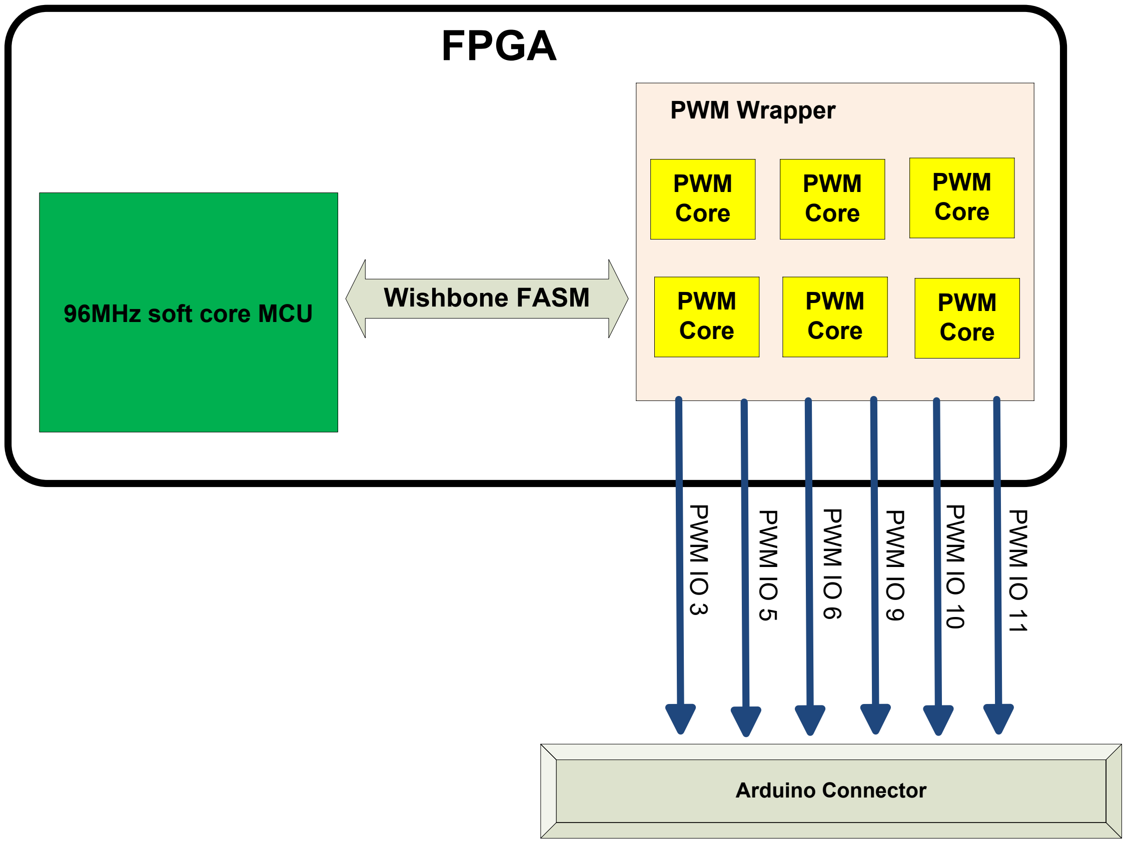

**Figure 2-1 PWM Controller**

## 2.2 Port List

The port list of the PWM core 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 |
| | sync_reset | Input | 1 | Synchronous reset, active high |
| Host Interface | pwm_pulse | Input | 1 | Pulse that gives the base frequency of PWM. On M10 board, the pwm_pulse is actually 96MHz / (X + 1), where X is a 16 bit value of resolution for each PWM channel |
| | pwm_on_reg | Input | 8 | Factor to divide the base frequency for PWM on interval |
| | pwm_off_reg | Input | 8 | Factor to divide the base frequency for PWM off interval |
| PWM Interface | pwm_out | Output | 1 | The output of Pulse Width Modulation |

**Table 2-1 Port List of PWM**

## 2.3 Pin Assignment

The following pins are assigned to PWM for the onboard FPGA (10M08SAE144C8G):

| Signal Name | FPGA Pin Assignment (10M08SAE144C8G) | IO Index of Arduino | Description |
|---|---|---|---|
| PWM_OUT[0] | 55 | 3 | 3.3 V / 5V |
| PWM_OUT[1] | 57 | 5 | |
| PWM_OUT[2] | 58 | 6 | |
| PWM_OUT[3] | 93 | 9 | |
| PWM_OUT[4] | 96 | 10 | |
| PWM_OUT[5] | 98 | 11 | |

**Table 2-2 FPGA Pin Assignment**

## 2.4 IO Drive Voltage / Current

The IO voltage can be set as either 3.3V or 5V through the jumper (JP1). However, all the IOs on the M10 board, including the PWM, are driven by the Voltage Translator TXS0108E (Ref [2]). And the maximum continuous output current for TXS0108E is only 50mA (Ref [2]). If larger current is needed, such as driving a DC motor, the user can adopt a H-bridge driver, such as the one in Ref [3]. (With H bridge, the DC motor can also be driven in both directions.)

## 2.5 Repository

The code for PWM Controller can be found on GitHub, in the "PWM" folder of

https://github.com/PulseRain/PulseRain_rtl_lib

# 3 Software

## 3.1 Register Definition

The PWM controller can be configured by the following registers and sub-registers:

- CSR (Control and Status Register)
  The bits for CSR are defined in Table 3-1:

| Bits | R/W | Default | Description |
|---|---|---|---|
| 2 : 0 | WO | 0 | index to specify the active PWM channel to be configured |
| 3 | WO | 0 | RESERVED |
| 6 : 4 | WO | 0 | **Sub-address**: Address of sub registers. This field specifies the active sub register to be written to. |
| 7 | WO | 0 | Write 1 to this bit will send the data in DATA register to the sub register specified by the sub-address field (bit [6 : 4]). |

**Table 3-1 Bit Map for CSR (Control Status Register)**

- DATA (8 bit, WO)
  The data to be sent to the sub register specified by the CSR[6: 4].

For each PWM channel, it has the following sub-registers:

| Sub-Addr | R/W | Default | Name | Description |
|---|---|---|---|---|
| 3'b000 | WO | 0 | PWM_SUB_ADDR_RESOLUTION_LOW | Lower 8-bit of the resolution pulse, the PWM base frequency is specified by **96MHz / (16_bit_resolution + 1)** |
| 3'b001 | WO | 0 | PWM_SUB_ADDR_RESOLUTION_HIGH | Higher 8-bit of the resolution pulse. |
| 3'b010 | WO | 0 | PWM_SUB_ADDR_REG_ON | 8-bit value to determine the on interval of each PWM pulse. The on-internal is actually **(16_bit_resolution + 1) * reg_on / 96MHz** |
| 3'b011 | WO | 0 | PWM_SUB_ADDR_REG_OFF | 8-bit value to determine the off interval of each PWM pulse. The off-internal is actually **(16_bit_resolution + 1) * reg_on / 96MHz** |
| 3'b100 | WO | 0 | PWM_SUB_ADDR_SYNC_RESET | Write to this register will reset the PWM channel specified by CSR [2:0] |

**Table 3-2 Sub Registers per PWM Channel**

## 3.2 Address Map

The registers defined in Section 3.1 are mapped into MCU's address space, as shown in Table 3-3.

| Address | Register Name |
|---|---|
| 0xD3 | PWM_CSR |
| 0xD4 | PWM_DATA |

**Table 3-3 Address Definition for PWM**

## 3.3 Work Flow

### 3.3.1 Write to Sub Register

To write to certain sub register in Table 3-2, do the following:

1. Write the value for the sub register into PWM_DATA (DATA register at address 0xD4)
2. Write CSR [2:0] to specify the PWM channel
3. Write CSR [6:4] to specify the address of sub register.
4. Write 1 to CSR[7] to copy the data from PWM_DATA to the sub register

### 3.3.2 PWM Generation

To generate PWM signal with certain duty cycles, do the following:

1. Write PWM_SUB_ADDR_RESOLUTION_LOW and PWM_SUB_ADDR_RESOLUTION_HIGH to specify the resolution of the PWM, namely base frequency. The base frequency is then determined by
   **96MHz / (16_bit_resolution + 1),** where 96MHz is the main clock

2. Write PWM_SUB_ADDR_REG_OFF and PWM_SUB_ADDR_REG_ON to setup the duty cycle.
   For each PWM cycle,
   the off-interval is PWM_SUB_ADDR_REG_OFF / base_frequency,
   the on-interval is PWM_SUB_ADDR_REG_ON / base_frequency,
   the duty cycle is PWM_SUB_ADDR_REG_ON / (PWM_SUB_ADDR_REG_ON + PWM_SUB_ADDR_REG_OFF),
   the PWM cycle period is (on-interval + off-interval)

3. Write 1 to PWM_SUB_ADDR_SYNC_RESET to re-initialize the PWM channel.

## 3.4 Arduino Library

PulseRain Technology has provided the M10PWM library to turn the workflow in Section 3.3 into API calls.

### 3.4.1 APIs

- *void (*resolution) (uint8_t pwm_index, uint16_t pwm_resolution)*

  Parameters:

  pwm_index : index for PWM channel, the mapping from index to pins are shown in Table 2-2.
  pwm_resolution: resolution for PWM, which determines the base frequency.

  Return Value:

  None

  Call this function to set the resolution/frequency for PWM channel.

- *void (*dutyCycle) (uint8_t pwm_index, uint8_t pwm_on, uint8_t pwm_off)*

  Parameters:

  pwm_index: index for PWM channel, the mapping from index to pins are shown in Table 2-2.
  pwm_on: The number of base cycles (1 / base_frequency) to be on
  pwm_off: The number of base cycles (1 / base_frequency) to be off

  Return Value:

  None

  Call this function to set the duty cycle of PWM channel.

### 3.4.2 Examples

One example for using PWM to control DC motor can be found in

https://github.com/PulseRain/Lego_Monster_Truck

Please note this example works with the Sparkfun Motor Driver Shield (with L298 chip on it).