

PWM Micro-Architecture Specification

**Revision**: v0.4

**Date**: 06/09/2021

Copyright © 2019 SOPHON TECHNOLOGIES LIMITED. All rights reserved.

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Revision** | **Issue Date** | **Author** | **Comments** |
| 0.1 | 07/24/2019 | Yunqi Liu | Initial Release. |
| 0.2 | 07/24/2019 | Yunqi Liu | 1. Added requirements for clock and reset in Section 2.2 Integration environment. 2. Modified the fonts in some of the pictures. 3. The waveform results after the configuration registers are added in Chapter 6. 4. Modify the module name from BM1684\_PWM to pwmx4. 5. Completing the number of some pictures. |
| 0.3 | 07/24/2020 | Jun Wang | 1. Re-org spec sections. 2. Fix typos. |
| 0.4 | 06/09/2021 | Ran Wang | 1. Fix PWM generation & example 2. Fix register address |

**Table of Contents**

[Revision History 3](#_Toc74148063)

[1 Overview 6](#_Toc74148064)

[1.1 Features and Parameters 6](#_Toc74148065)

[1.2 Terms and Abbreviations 6](#_Toc74148066)

[2 Top Interface 7](#_Toc74148067)

[2.1 Interface Signal Description 7](#_Toc74148068)

[2.2 Integration Requirement 8](#_Toc74148069)

[2.2.1 Synchronization of clock and reset 8](#_Toc74148070)

[3 Function Description 9](#_Toc74148071)

[3.1 PWM generation 9](#_Toc74148072)

[3.1.1 Period of PWM 9](#_Toc74148073)

[3.1.2 Duty Cycle of PWM 9](#_Toc74148074)

[3.2 Pulse detection 10](#_Toc74148075)

[3.2.1 Time Window 10](#_Toc74148076)

[3.2.2 Glitch Filter 10](#_Toc74148077)

[4 Internal Blocks 12](#_Toc74148078)

[4.1 Partition Overview 12](#_Toc74148079)

[4.2 PWM Generator Module 12](#_Toc74148080)

[4.3 Pulse Detector Module 13](#_Toc74148081)

[5 Register Definition 14](#_Toc74148082)

[5.1 Memory Mapped Registers Summary 14](#_Toc74148083)

[5.2 Register Description 14](#_Toc74148084)

[5.2.1 HLPERIOD0(0x0000) 14](#_Toc74148085)

[5.2.2 PERIOD0(0x0004) 14](#_Toc74148086)

[5.2.3 HLPERIOD1(0x0008) 14](#_Toc74148087)

[5.2.4 PERIOD1(0x000C) 15](#_Toc74148088)

[5.2.5 HLPERIOD2(0x0010) 15](#_Toc74148089)

[5.2.6 PERIOD2(0x0014) 15](#_Toc74148090)

[5.2.7 HLPERIOD3(0x0018) 15](#_Toc74148091)

[5.2.8 PERIOD3(0x001C) 15](#_Toc74148092)

[5.2.9 FREQ0NUM(0x0020) 15](#_Toc74148093)

[5.2.10 FEQ0DATA(0x0024) 15](#_Toc74148094)

[5.2.11 FREQ1NUM(0x0028) 16](#_Toc74148095)

[5.2.12 FEQ1DATA(0x002C) 16](#_Toc74148096)

[5.2.13 FREQ2NUM(0x0030) 16](#_Toc74148097)

[5.2.14 FEQ2DATA(0x0034) 16](#_Toc74148098)

[5.2.15 FREQ3NUM(0x0038) 16](#_Toc74148099)

[5.2.16 FEQ3DATA(0x003C) 16](#_Toc74148100)

[6 Software Program Guide 17](#_Toc74148101)

[6.1 PWM generation 17](#_Toc74148102)

[6.1.1 Reset State 17](#_Toc74148103)

[6.1.2 Start PWM 17](#_Toc74148104)

[6.1.3 Pull High 17](#_Toc74148105)

[6.1.4 Pull Low 17](#_Toc74148106)

[6.1.5 Example of starting PWM 17](#_Toc74148107)

[6.2 Pulse detection 18](#_Toc74148108)

[6.2.1 Reset State 18](#_Toc74148109)

[6.2.2 Start pulse detection 18](#_Toc74148110)

[6.2.3 An example of starting pulse detection 18](#_Toc74148111)

[7 Known Issues and Future Work 20](#_Toc74148112)

# Overview

The pwmx4 contains two functions:

* Generate PWM waveforms.
* Detect the number of pulses over a period of time.

Figure 1-1 provides the overview of pwmx4. All control of the pwmx4 is performed via the APB interface.



Figure -1 pwmx4 Overview

## Features and Parameters

The pwmx4 provides the following features:

* 4 PWM wave output channel.
* 4 Pulse detect channel.
* Ability to filter glitch that may be included in the pulse.
* All configurations and results are read and written via the APB interface.

## Terms and Abbreviations

PWM Pulse-width modulation

# Top Interface

## Interface Signal Description

Figure 2-1 illustrates the top connection of the module.



Figure -1 Connection Overview

The detailed signals and their description are shown in Table 1.

Table Top Interface

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **I/O** | **Connection** | **Description** |
| APB Signal | | | |
| p\_clk | I | APB bridge | Clock from APB bridge. The only clock source of this module. |
| p\_resetn | I | APB bridge | Async reset (active low) from APB bridge. The only reset source of this module. |
| p\_enable | I | APB bridge | APB enable signal. |
| p\_write | I | APB bridge | APB write signal. |
| p\_sel | I | APB bridge | APB sel signal. |
| p\_addr[5:0] | I | APB bridge | APB address signal. Connected to the lower 6 bits of the APB address bus. The lowest 2 bits are not used. |
| p\_wdata[31:0] | I | APB bridge | APB write data signal. |
| p\_rdata[31:0] | O | APB bridge | APB read data signal. |
| Output Signal | | | |
| pwm\_o0 | O | PWM receiver | PWM channel 0 output. |
| pwm\_o1 | O | PWM receiver | PWM channel 1 output. |
| pwm\_o2 | O | PWM receiver | PWM channel 2 output. |
| pwm\_o3 | O | PWM receiver | PWM channel 3 output. |
| Input Signal | | | |
| freq\_i0 | I | Pulse source | Pulse detect channel 0 input. |
| freq\_i1 | I | Pulse source | Pulse detect channel 1 input. |
| freq\_i2 | I | Pulse source | Pulse detect channel 2 input. |
| freq\_i3 | I | Pulse source | Pulse detect channel 3 input. |

## Integration Requirement

### Synchronization of clock and reset

p\_clk is the only clock signal for this module.

p\_resetn is the only reset signal for this module. There is no synchronization of reset\_n inside the module. Therefore, p\_resetn needs to be synchronized outside the module with p\_clk.

# Function Description

The pwmx4 contains two functions:

* Generate PWM waveforms.
* Detect the number of pulses over a period of time.

## PWM generation

The generation of PWM waves is controlled by two registers, PERIODx and HLPERIODx. Where x can be 0, 1, 2, 3, corresponding to 4 PWM channels. Figure 3-1 shows how registers PERIODx and HLPERIODx affect the PWM waveform.



Figure -1 The effect of registers on PWM waves

### Period of PWM

The period of the output PWM wave is

The range of PERIODx is 3 to 4294967295.

**Attention:** The register PERIODx can be written to all values in the range of 0 to 4294967295, but only the above range is legal. Exceeding this range may result in no output waveform or output waveform frequency error.

### Duty Cycle of PWM

The duty cycle of the PWM wave is

The range of HLPERIODx is 2 to 4294967295.

**Attention:** The register HLPERIODx can be written to all values in the range of 0 to 4294967295 and must be smaller than the corresponding PERIODx, but only the above range is legal. Exceeding this range may result in no output waveform or output waveform frequency error.

## Pulse detection

The time of each pulse detection is controlled by the register FRExNUM, and the result is written to the FRExDATA register after each detection count is completed. The pulse detection circuit will always work cyclically.Figure 3-2 shows how FRExNUM affects the pulse detection time and how FRExDATA records the detection count results. Where x can be 0, 1, 2, 3, corresponding to 4 pulse detection channels.

The signal is passed through Glitch Filter to eliminate glitch before it is detected and counted.



Figure -2 The effect of registers on Pulse Detection

### Time Window

The time window width of one pulse detection is

The range of FRExNUM is 1 to 4294967295.

**Attention:** The register FRExNUM cannot be set to 0, otherwise the FRExDATA value will always be 0. But this does not mean that the circuit stops working, the circuit will continue to work and the power consumption will not decrease.

### Glitch Filter

Pulse Detection Filters the input signal using the Glitch filter before counting the pulses. Figure 3-3 shows the role of Glitch Filter.



Figure -3 The role of Glitch Filter

Any pulse narrower than GlitchWidth is defined as a glitch.Where

The Glitch filter considers all pulses with a width greater than or equal to Glitchwidth to be valid, and the rest are glitch. And the glitch in the input signal will be removed and output.

Figure 3-4 shows how the glitch filter works according to GlitchWidth.



Figure -4 The operation of the glitch filter on the rising and falling edge of the pulse

**Attention 1:** The width of the pulse must be greater than GlitchWidth to pass the glitch filter and be detected and counted.

**Attention 2:** A glitch with a width greater than GlitchWidth cannot be removed.

# Internal Blocks

## Partition Overview

The pwmx4 contains three modules internally, of which PWM Generator and Pulse Detector implement PWM waveform generation and pulse detection respectively. APB slave is used to read and write the internal registers of the other two modules. The implementation of the APB slave will be omitted below.

Figure 4-1 shows the internal parathion of pwmx4:



Figure 4-1 The microarchitecture of pwmx4

## PWM Generator Module

The PWM Generator provides 4 channels of PWM waveform output with frequency and duty cycle controlled by registers PERIODx and HLPERIODx. Where x can be 0, 1, 2, 3, corresponding to 4 channels. Figure 4-2 shows the microarchitecture of the PWM generator.



Figure 4-2 The microarchitecture of the PWM generator

## Pulse Detector Module

The Pulse Detector provides 4 channels of pulse detection input. The time window length of one test is configured by FREQxNUM, and the count result is automatically loaded into FREQxDATA. Where x can be 0, 1, 2, 3, corresponding to 4 channels.

The signal of each input pulse detection channel will enter the glitch filter first. The behavior of the glitch filter is detailed in the function description. This filter is based on a 16-bit shift register.

The signal processed by the glitch filter is converted to a single p\_clk clock cycle pulse using a single-cycle pulse generator. This single-cycle pulse signal is used to enable the counter to complete the counting of the pulses.

When a frequently enabled counter is used for pulse detection, the detection result will be written to FREQxDATA when it reaches the time window length.

Figure 4-3 shows the microarchitecture of the pulse detector.



Figure 4-3 The microarchitecture of the pulse detector

# Register Definition

## Memory Mapped Registers Summary

|  |  |  |  |
| --- | --- | --- | --- |
| **Offset** | **Register Name** | **Default** | **Attribute** |
| 0x0000 | HLPERIOD0 | 32’h00000000 | RW |
| 0x0004 | PERIOD0 | 32’h00000000 | RW |
| 0x0008 | HLPERIOD1 | 32’h00000000 | RW |
| 0x000C | PERIOD1 | 32’h00000000 | RW |
| 0x0010 | HLPERIOD2 | 32’h00000000 | RW |
| 0x0014 | PERIOD2 | 32’h00000000 | RW |
| 0x0018 | HLPERIOD3 | 32’h00000000 | RW |
| 0x001C | PERIOD3 | 32’h00000000 | RW |
| 0x0020 | FREQ0NUM | 32’h00000000 | RW |
| 0x0024 | FERQ0DATA | 32’h00000000 | RO |
| 0x0028 | FREQ1NUM | 32’h00000000 | RW |
| 0x002C | FERQ1DATA | 32’h00000000 | RO |
| 0x0030 | FREQ2NUM | 32’h00000000 | RW |
| 0x0034 | FERQ2DATA | 32’h00000000 | RO |
| 0x0038 | FREQ3NUM | 32’h00000000 | RW |
| 0x003C | FERQ3DATA | 32’h00000000 | RO |

## Register Description

### HLPERIOD0(0x0000)

|  |  |  |  |
| --- | --- | --- | --- |
| **Bit** | **Attribute** | **Default** | **Description** |
| 31:0 | RW | 32’h00000000 | The time that the PWM wave inchannel 0 remains high for one cycle. The actual duration is p\_clk clock period multiplied by this register value. |

### PERIOD0(0x0004)

|  |  |  |  |
| --- | --- | --- | --- |
| **Bit** | **Attribute** | **Default** | **Description** |
| 31:0 | RW | 32’h00000000 | The PWM wave period of channel 0 based on p\_clk. The actual period is the p\_clk period multiplied by this register value. |

### HLPERIOD1(0x0008)

|  |  |  |  |
| --- | --- | --- | --- |
| **Bit** | **Attribute** | **Default** | **Description** |
| 31:0 | RW | 32’h00000000 | The time that the PWM wave in channel 1 remains high for one cycle. The actual duration is p\_clk clock period multiplied by this register value. |

### PERIOD1(0x000C)

|  |  |  |  |
| --- | --- | --- | --- |
| **Bit** | **Attribute** | **Default** | **Description** |
| 31:0 | RW | 32’h00000000 | The PWM wave period of channel 1 based on p\_clk. The actual period is the p\_clk period multiplied by this register value. |

### HLPERIOD2(0x0010)

|  |  |  |  |
| --- | --- | --- | --- |
| **Bit** | **Attribute** | **Default** | **Description** |
| 31:0 | RW | 32’h00000000 | The time that the PWM wave in channel 2 remains high for one cycle. The actual duration is p\_clk clock period multiplied by this register value. |

### PERIOD2(0x0014)

|  |  |  |  |
| --- | --- | --- | --- |
| **Bit** | **Attribute** | **Default** | **Description** |
| 31:0 | RW | 32’h00000000 | The PWM wave period of channel 2 based on p\_clk. The actual period is the p\_clk period multiplied by this register value. |

### HLPERIOD3(0x0018)

|  |  |  |  |
| --- | --- | --- | --- |
| **Bit** | **Attribute** | **Default** | **Description** |
| 31:0 | RW | 32’h00000000 | The time that the PWM wave in channel 3 remains high for one cycle. The actual duration is p\_clk clock period multiplied by this register value. |

### PERIOD3(0x001C)

|  |  |  |  |
| --- | --- | --- | --- |
| **Bit** | **Attribute** | **Default** | **Description** |
| 31:0 | RW | 32’h00000000 | The PWM wave period of channel 3 based on p\_clk. The actual period is the p\_clk period multiplied by this register value. |

### FREQ0NUM(0x0020)

|  |  |  |  |
| --- | --- | --- | --- |
| **Bit** | **Attribute** | **Default** | **Description** |
| 31:0 | RW | 32’h00000000 | The length of time used for the pulse detection channel 0. The actual length of time is the value of this register multiplied by the p\_clk period. |

### FEQ0DATA(0x0024)

|  |  |  |  |
| --- | --- | --- | --- |
| **Bit** | **Attribute** | **Default** | **Description** |
| 31:0 | RO | 32’h00000000 | The number of pulses detected by pulse detection channel 0 within the length of time defined by register FRE0NUM. |

### FREQ1NUM(0x0028)

|  |  |  |  |
| --- | --- | --- | --- |
| **Bit** | **Attribute** | **Default** | **Description** |
| 31:0 | RW | 32’h00000000 | The length of time used for the pulse detection channel 1. The actual length of time is the value of this register multiplied by the p\_clk period. |

### FEQ1DATA(0x002C)

|  |  |  |  |
| --- | --- | --- | --- |
| **Bit** | **Attribute** | **Default** | **Description** |
| 31:0 | RO | 32’h00000000 | The number of pulses detected by pulse detection channel 1 within the length of time defined by register FRE1NUM. |

### FREQ2NUM(0x0030)

|  |  |  |  |
| --- | --- | --- | --- |
| **Bit** | **Attribute** | **Default** | **Description** |
| 31:0 | RW | 32’h00000000 | The length of time used for the pulse detection channel 2. The actual length of time is the value of this register multiplied by the p\_clk period. |

### FEQ2DATA(0x0034)

|  |  |  |  |
| --- | --- | --- | --- |
| **Bit** | **Attribute** | **Default** | **Description** |
| 31:0 | RO | 32’h00000000 | The number of pulses detected by pulse detection channel 2 within the length of time defined by register FRE2NUM. |

### FREQ3NUM(0x0038)

|  |  |  |  |
| --- | --- | --- | --- |
| **Bit** | **Attribute** | **Default** | **Description** |
| 31:0 | RW | 32’h00000000 | The length of time used for the pulse detection channel 3. The actual length of time is the value of this register multiplied by the p\_clk period. |

### FEQ3DATA(0x003C)

|  |  |  |  |
| --- | --- | --- | --- |
| **Bit** | **Attribute** | **Default** | **Description** |
| 31:0 | RO | 32’h00000000 | The number of pulses detected by pulse detection channel 3 within the length of time defined by register FRE3NUM. |

# Software Program Guide

The PWM generation and pulse detection functions are independent of each other. The following two sections will explain how to use these two functions.

## PWM generation

### Reset State

After reset, the output of the PWM channel is always **high**.

And the value of PERIODx is 0x00000000.

The value of HLPERIODx is 0x00000000.

### Start PWM

The steps to start the PWM waveform output on one channel are as follows:

1. Configure PERIODx. The calculation formula for this value is given in Function Description.
2. Configure HLPERIODx. The calculation formula for this value is given in Function Description.

The PWM wave output does not stop until if is no other operation.



Figure 6-1 The effect of registers on PWM waves

### Pull High

The steps to stop the PWM wave output and pull the output high are as follows:

1. Configure PERIODx to 0x00000000.
2. Configure HLPERIODx to 0x00000000.

### Pull Low

The steps to stop the PWM wave output and pull the output low are as follows:

1. Configure HLPERIODx to 0x00000001.
2. Configure PERIODx to 0x00000000.

### Example of starting PWM

Assume that the current p\_clk frequency is 100M, that is, its period is 10 ns.

The PWM wave frequency that needs to be output is 10M, that is, its period is 100ns.

The duty cycle required for the PWM wave is 70 percent.

The channel that needs to output the PWM wave is channel 0.

According to the formula given by the function description, you can get a PERIODx value of 10, and a HLPERIODx value of 7.

The steps to start the PWM wave on channel 0 are as follows:

1. Configure PERIODx to 0x0000000a.
2. Configure HLPERIODx to 0x00000007.



Figure 6-2 An example of generating a PWM wave

## Pulse detection

### Reset State

After reset, the value of FERQxNUM is 0x00000000.

The value of FREQxDATA is 0x00000000.

### Start pulse detection

The steps to start a pulse detection are as follows:

1. Configure FREQxNUM. The calculation formula for this value is given in Function Description.
2. Wait for a while until the test is completed at least once. This time must be long enough to ensure that at least one time window is run after the registers in the module are configured.
3. Reads the value in FREQxDATA, which is the number of pulses detected in a time window.

Pulse detection cannot be turned off. If you need to clear the value in FREQxDATA, you can set FREQxNUM to 0.

### An example of starting pulse detection

Assume that the current p\_clk frequency is 100M, that is, its period is 10 ns.

The time window width required for pulse detection is 1 ms.

Use channel 0 for pulse detection.

According to the formula given by the function description, the value of FREQ0NUM is 100000.

The steps to start a pulse detection are as follows:

1. Configure FREQ0NUM to 0x000186A0.
2. Wait for a while to ensure that the module runs for more than 1 ms after the register is written. For example, you can wait for 1.5ms under normal conditions.
3. Reads the value in FREQ0DATA.

Figure 6-3 shows the process of this example.



Figure 6-3 An example of pulse detection

# Known Issues and Future Work

TBD