



PADAUK

應廣科技

***LeapDragon*** ( 跃龙 )

**PFC886**

Industrial Grade - 8bit MTP MCU (FPPA™)

Data Sheet

*Version 0.05*

*Oct. 31, 2023*

Copyright © 2023 by PADAUK Technology Co., Ltd., all rights reserved.

6F-6, No.1, Sec. 3, Gongdao 5<sup>th</sup> Rd., Hsinchu City 30069, Taiwan, R.O.C.

TEL: 886-3-572-8688  [www.padauk.com.tw](http://www.padauk.com.tw)

## **IMPORTANT NOTICE**

**PADAUK Technology reserves the right to make changes to its products or to terminate production of its products at any time without notice. Customers are strongly recommended to contact PADAUK Technology for the latest information and verify whether the information is correct and complete before placing orders.**

**PADAUK Technology products are not warranted to be suitable for use in life-support applications or other critical applications. PADAUK Technology assumes no liability for such applications. Critical applications include, but are not limited to, those that may involve potential risks of death, personal injury, fire or severe property damage.**

**Any programming software provided by PADAUK Technology to customers is of a service and reference nature and does not have any responsibility for software vulnerabilities. PADAUK Technology assumes no responsibility for any issue caused by a customer's product design. Customers should design and verify their products within the ranges guaranteed by PADAUK Technology. In order to minimize the risks in customers' products, customers should design a product with adequate operating safeguards.**

## Table of Contents

<b>Revision History.....</b>	<b>9</b>
<b>Usage Warning.....</b>	<b>9</b>
<b>1. Features.....</b>	<b>10</b>
1.1. Special Features.....	10
1.2. System Features .....	10
1.3. High Performance RISC CPU Array .....	11
1.4. Ordering/ Package Information.....	11
<b>2. General Description and Block Diagram .....</b>	<b>12</b>
<b>3. Pin Definition and Functional Description.....</b>	<b>13</b>
<b>4. Central Processing Unit (CPU) .....</b>	<b>15</b>
4.1. Functional Description .....	15
4.1.1. Processing Units .....	15
4.1.2. Program Counter.....	17
4.1.3. Program Structure.....	18
4.1.4. Arithmetic and Logic Unit .....	18
4.2. Storage Memory .....	19
4.2.1. Program Memory – ROM .....	19
4.2.2. Data Memory – SRAM .....	21
4.2.3. System Register.....	22
4.2.3.1. ACC Status Flag Register (FLAG), address = 0x00.....	23
4.2.3.2. FPPA unit Enable Register (FPPEN), address = 0x01.....	23
4.2.3.3. Hopping Setting Register (HOP), address = 0x23 .....	23
4.2.3.4. Option 2 Register (OPR2), address = 0x3A .....	24
4.2.3.5. MISC Register (MISC), address = 0x3B .....	24
4.3. The Stack .....	24
4.3.1. Stack Pointer Register (SP), address = 0x02 .....	25
4.4. Code Options.....	25
<b>5. Oscillator and System Clock .....</b>	<b>26</b>
5.1. Internal High RC Oscillator and Internal Low RC Oscillator .....	26
5.2. System Clock and IHRC Calibration .....	26
5.2.1. System Clock .....	26
5.2.1.1. Clock Mode Register (CLKMD), address = 0x03 .....	27
5.2.2. Frequency Calibration .....	28

5.2.2.1. Special Statement.....	29
5.2.3. System Clock Switching.....	29
<b>6. Reset.....</b>	<b>30</b>
6.1. Power On Reset - POR.....	30
6.1.1. POR for DC Fan Application .....	31
6.2. Low Voltage Reset - LVR.....	32
6.3. Watch Dog Timeout Reset.....	34
6.4. External Reset Pin - PRSTB .....	35
<b>7. System Operating Mode.....</b>	<b>35</b>
7.1. Power-Save Mode (“stopexe”) .....	36
7.2. Power-Down Mode (“stopsys”).....	37
7.3. Wake-Up .....	38
<b>8. Interrupt.....</b>	<b>39</b>
8.1. Interrupt Enable Register ( <i>INTEN</i> ), address = 0x04 .....	41
8.2. Interrupt Enable 2 Register ( <i>INTEN2</i> ), address = 0x08 .....	41
8.3. Interrupt Request Register ( <i>INTRQ</i> ), address = 0x05 .....	41
8.4. Interrupt Request 2 Register ( <i>INTRQ2</i> ), address = 0x09 .....	42
8.5. Interrupt Edge Select Register ( <i>INTEGS</i> ), address = 0x0C.....	42
8.6. Interrupt Work Flow .....	43
8.7. General Steps to Interrupt.....	43
8.8. Example for Using Interrupt .....	44
<b>9. I/O Port.....</b>	<b>45</b>
9.1. IO Related Registers .....	45
9.1.1. General Data register for IO ( <i>GDIO</i> ), address = 0x07 .....	45
9.1.2. Port A Digital Input Enable Register ( <i>PADIER</i> ), address = 0x0D .....	45
9.1.3. Port B Digital Input Enable Register ( <i>PBDIER</i> ), address = 0x0E.....	45
9.1.4. Port A Data Registers ( <i>PA</i> ), address = 0x10.....	46
9.1.5. Port A Control Registers ( <i>PAC</i> ), address = 0x11 .....	46
9.1.6. Port A Pull-High Registers ( <i>PAPH</i> ), address = 0x12 .....	46
9.1.7. Port B Data Registers ( <i>PB</i> ), address = 0x14.....	46
9.1.8. Port B Control Registers ( <i>PBC</i> ), address = 0x15 .....	46
9.1.9. Port B Pull-High Registers ( <i>PBPH</i> ), address = 0x16 .....	46
9.2. IO Structure and Functions .....	47
9.2.1. IO Pin Structure .....	47
9.2.2. IO Pin Functions .....	47
9.2.3. IO Pin Usage and Setting .....	48

<b>10. Timer / PWM Counter.....</b>	<b>49</b>
10.1. 16-bit Timer (Timer16).....	49
10.1.1. Timer16 Introduction.....	49
10.1.2. Timer16 Time Out.....	50
10.1.3. Timer16 Mode Register ( <i>T16M</i> ), address = 0x06.....	51
10.2. 16-bit Timer (Timer16N) .....	52
10.2.1. Timer16N Introduction .....	52
10.2.2. Timer16N mode Register ( <i>T16NM</i> ), address = 0x19.....	53
10.2.3. Timer16N Bound High Byte Register ( <i>T16NBH</i> ), address = 0x73 .....	53
10.2.4. Timer16N Bound Low Byte Register ( <i>T16NBL</i> ), address = 0x74 .....	53
10.3. 8-bit Timer(Timer2, Timer3).....	54
10.3.1. Timer2 Control Register ( <i>TM2C</i> ), address = 0x24.....	55
10.3.2. Timer2 Counter Register ( <i>TM2CT</i> ), address = 0x25 .....	55
10.3.3. Timer2 Scalar Register ( <i>TM2S</i> ), address = 0x26 .....	56
10.3.4. Timer2 Bound Register ( <i>TM2B</i> ), address = 0x27 .....	56
10.3.5. Timer3 Control Register ( <i>TM3C</i> ), address = 0x28.....	56
10.3.6. Timer3 Counter Register ( <i>TM3CT</i> ), address = 0x29 .....	56
10.3.7. Timer3 Scalar Register ( <i>TM3S</i> ), address = 0x2A.....	57
10.3.8. Timer3 Bound Register ( <i>TM3B</i> ), address = 0x2B.....	57
10.4. 12-bit PWM Generation.....	57
10.4.1. PWM Waveform .....	57
10.4.2. Hardware PWM without dead zone and Timing Diagram .....	57
10.4.3. Equations for 12-bit PWM Generator .....	58
10.4.4. Hardware PWM with dead zone.....	59
10.4.5. 12-bit PWM Related Registers.....	60
10.4.5.1. PWM Generator control Register (PWMGC), address = 0x30 .....	60
10.4.5.2. PWM Generator Scalar Register (PWMGM), address = 0x31 .....	60
10.4.5.3. PWM Generator control 1 Register (PWMGC1), address = 0x37 .....	61
10.4.5.4. PWM Generator control 2 Register (PWMGC2), address = 0x38 .....	61
10.4.5.5. PWM Counter Upper Bound High Register (PWMCUBH), address = 0x50 ..	61
10.4.5.6. PWM Counter Upper Bound Low Register (PWMCUBL), address = 0x51 ...	62
10.4.5.7. PWM0 Duty Value High Register (PWM0DTH), address = 0x52.....	62
10.4.5.8. PWM0 Duty Value Low Register (PWM0DTL), address = 0x53 .....	62
10.4.5.9. PWM1 Duty Value High Register (PWM1DTH), address = 0x54.....	62
10.4.5.10. PWM1 Duty Value Low Register (PWM1DTL), address = 0x55 .....	62
10.4.5.11. PWM2 Duty Value High Register (PWM2DTH), address = 0x56.....	62
10.4.5.12. PWM2 Duty Value Low Register (PWM2DTL), address = 0x57 .....	62
10.4.5.13. PWM Dead-zone Register (PWMDZ), address = 0x58 .....	62

<b>11. Special Functions .....</b>	<b>63</b>
11.1. General Purpose Comparator .....	63
11.1.1. General Purpose Comparator Hardware Diagram.....	63
11.1.2. General Purpose Comparator 1 Control Register ( <i>GPC1C</i> ), address = 0x34 .....	66
11.1.3. General Purpose Comparator 1 Selection Register ( <i>GPC1S</i> ), address = 0x35 .....	66
11.1.4. General Purpose Comparator 2 Control Register ( <i>GPC2C</i> ), address = 0x1E.....	67
11.1.5. General Purpose Comparator 2 Selection Register ( <i>GPC2S</i> ), address = 0x1F .....	67
11.1.6. Analog Inputs .....	68
11.1.7. Internal Reference Voltage ( $V_{\text{internal R}}$ ) .....	68
11.1.8. Synchronizing General Purpose Comparator 1 Output to Timer2.....	70
11.1.9. Synchronizing General Purpose Comparator 2 Output to Timer3.....	71
11.1.10. Using the Comparator1 .....	71
11.2. 8X Operational Amplifier (OPAmP) module .....	72
11.2.1. Configure the analog pins .....	73
11.2.2. OPA Control Register ( <i>AMPC</i> ), address = 0x18 .....	73
11.3. Analog-to-Digital Conversion (ADC) module .....	74
11.3.1. The input requirement for AD conversion .....	75
11.3.2. ADC clock selection .....	75
11.3.3. AD conversion .....	75
11.3.4. Configure the analog pins .....	76
11.3.5. Using the ADC .....	77
11.3.6. ADC Related Registers.....	78
11.3.6.1. ADC Control Register ( <i>ADCC</i> ), address = 0x20 .....	78
11.3.6.2. ADC Mode Register ( <i>ADCM</i> ), address = 0x21 .....	78
11.3.6.3. ADC Result High Register ( <i>ADCRH</i> ), address = 0x4A .....	78
11.3.6.4. ADC Result Low Register ( <i>ADCRL</i> ), address = 0x4B.....	79
11.4. PWM generator Trigger AD Conversion .....	79
11.4.1. Hopping Setting Register ( <i>HOP</i> ), address = 0x23.....	80
11.4.2. PWM Trigger ADC - PWM Counter High Register ( <i>PWMADCH</i> ), address = 0x5E ..	80
11.4.3. PWM Trigger ADC - PWM Counter Low Register ( <i>PWMADCL</i> ), address = 0x5F ...	80
11.5. 1 or 3-phase Brushless DC Motor .....	81
11.5.1. Single-Phase PWM Protection.....	81
11.5.2. Three-Phase PWM Protection .....	82
11.6. Input Pulse Capture .....	82
11.6.1. Pulse Capture Control Register ( <i>PLSCC</i> ), address = 0x32 .....	83
11.6.2. Pulse Capture Scalar Register ( <i>PLSCS</i> ), address = 0x33.....	84
11.6.3. Pulse Capture Pulse Width High Register ( <i>PLSPWH</i> ), address = 0x4C.....	84
11.6.4. Pulse Capture Pulse Width Low Register ( <i>PLSPWL</i> ), address = 0x4D .....	84

11.6.5. Pulse Capture Pulse High High Register ( <i>PLSPHH</i> ), address = 0x4E .....	84
11.6.6. Pulse Capture Pulse High Low Register ( <i>PLSPHL</i> ), address = 0x4F .....	84
11.7. Multiplier and Divider .....	85
11.7.1. 16x8 multiplier .....	85
11.7.2. 16x16 multiplier .....	85
11.7.3. 16/16 Divider .....	86
11.7.4. Arithmetic Operation Register ( <i>EARITH</i> ), address = 0x3C .....	86
11.7.5. 16X8 Multiplier Operand 1 High Byte Register ( <i>M8OP1H</i> ), address = 0x44 .....	86
11.7.6. 16X8 Multiplier Operand 1 Low Byte Register ( <i>M8OP1L</i> ), address = 0x45 .....	86
11.7.7. 16X8 Multiplier Operand 2 Register ( <i>M8OP2</i> ), address = 0x46 .....	86
11.7.8. 16X8 Multiplier Result2 Register ( <i>M8RS2</i> ), address = 0x47 .....	87
11.7.9. 16X8 Multiplier Result1 Register ( <i>M8RS1</i> ), address = 0x48 .....	87
11.7.10. 16X8 Multiplier Result0 Register ( <i>M8RS0</i> ), address = 0x49 .....	87
11.7.11. 16X16 Multiplier Operand 1 High Byte Register ( <i>M16OP1H</i> ), address = 0x60 .....	87
11.7.12. 16X16 Multiplier Operand 1 Low Byte Register ( <i>M16OP1L</i> ), address = 0x61 .....	87
11.7.13. 16X16 Multiplier Operand 2 High Byte Register ( <i>M16OP2H</i> ), address = 0x62 .....	87
11.7.14. 16X16 Multiplier Operand 2 Low Byte Register ( <i>M16OP2L</i> ), address = 0x63 .....	87
11.7.15. 16X16 Multiplier Result3 Register ( <i>M16RS2H</i> ), address = 0x64 .....	87
11.7.16. 16X16 Multiplier Result2 Register ( <i>M16RS2L</i> ), address = 0x65 .....	87
11.7.17. 16X16 Multiplier Result1 Register ( <i>M16RS1H</i> ), address = 0x66 .....	87
11.7.18. 16X16 Multiplier Result0 Register ( <i>M16RS1L</i> ), address = 0x67 .....	88
11.7.19. 16/16 Divider Dividend High Byte Register ( <i>D16DEH</i> ), address = 0x68 .....	88
11.7.20. 16/16 Divider Dividend Low Byte Register ( <i>D16DEL</i> ), address = 0x69 .....	88
11.7.21. 16/16 Divider Divisor High Byte Register ( <i>D16DSH</i> ), address = 0x6A .....	88
11.7.22. 16/16 Divider Divisor Low Byte Register ( <i>D16DSL</i> ), address = 0x6B .....	88
11.7.23. 16/16 Divider Quotient High Byte Register ( <i>D16QUH</i> ), address = 0x6C .....	88
11.7.24. 16/16 Divider Quotient Low Byte Register ( <i>D16QUL</i> ), address = 0x6D .....	88
11.7.25. 16/16 Divider Remainder High Byte Register ( <i>D16REH</i> ), address = 0x6E .....	88
11.7.26. 16/16 Divider Remainder Low Byte Register ( <i>D16REL</i> ), address = 0x6F .....	88
<b>12. Program Writing .....</b>	<b>89</b>
12.1. Normal Programming Mode .....	89
12.2. Limited-Voltage Programming Mode .....	89
12.3. On-Board Writing .....	90
<b>13. Device Characteristics .....</b>	<b>91</b>
13.1. Absolute Maximum Ratings .....	91
13.2. DC/AC Characteristics .....	91
13.3. Typical ILRC frequency vs. VDD and temperature .....	93

13.4.	Typical IHRC frequency deviation vs. VDD and temperature.....	94
13.5.	Typical Operating Current vs. VDD and CLK=IHRC/n .....	94
13.6.	Typical Operating Current vs. VDD and CLK=ILRC/n.....	95
13.7.	Typical IO pull high resistance.....	95
13.8.	Typical IO driving current ( $I_{OH}$ ) and sink current ( $I_{OL}$ ) .....	96
13.9.	Typical IO input high/low threshold voltage ( $V_{IH}/V_{IL}$ ) .....	97
<b>14.</b>	<b>Instructions .....</b>	<b>98</b>
14.1.	Instruction Table.....	99



## Revision History

Revision	Date	Description
0.04	2023/03/10	<ol style="list-style-type: none"><li>1. Updated "IMPORTANT NOTICE"</li><li>2. Amend Operating Min. Voltage</li><li>3. Amend Section 1.2, 1.4, 4.4, 5.2.2, 10.3.1, 10.3.5, 10.3.7, 11.6.2, 11.7.15~11.7.18</li><li>4. Amend Fig. 5, Fig. 18, Fig. 20, Fig. 21, Fig. 35 and Chapter 3</li><li>5. Other known details bug correct.</li></ol>
0.05	2023/10/31	<ol style="list-style-type: none"><li>1. Add PFC886-1J16A (QFN3*3-16P-0.5pitch)</li><li>2. Amend Watch Dog Timeout Reset, Reset, Fig. 20&amp;21</li><li>3. Amend 4MIPS</li></ol>

## Usage Warning

User must read all application notes of the IC by detail before using it. Please download the related application notes from the following link:

<http://www.padauk.com.tw/en/technical/index.aspx>

## 1. Features

### 1.1. Special Features

- ◆ High EFT series

Especially fit for the products that are AC powered with even using RC step-down circuit, or require strong noise immunity, or required high EFT capability ( $\pm 4\text{KV}$ ) for passing safety regulation tests.

- ◆ Operating temperature range:  $-40^{\circ}\text{C} \sim 85^{\circ}\text{C}$

### 1.2. System Features

- ◆ 4KW MTP program memory for 8 FPPA units (programming cycle at least 1,000 times)
- ◆ 512 Bytes data RAM for all FPPA units
- ◆ Support In System Programming for MTP
- ◆ Two hardware 16-bit timer
- ◆ Two hardware 8-bit timer
- ◆ One 12-bit hardware PWM generator with programmable period and duty
- ◆ Support PWM generator trigger AD conversion
- ◆ Up to 11-channel 11-bit resolution ADC with 1-channel for internal Band-gap reference voltage
- ◆ One 8-time voltage gain Operational Amplifier for current detection
- ◆ Hardware Pulse Capture
- ◆ One 16X8 hardware multiplier
- ◆ One 16X16 hardware multiplier
- ◆ One 16/16 hardware divider
- ◆ Two general purpose comparators
- ◆ Support 1 or 3-Phase brushless DC motor
- ◆ 16 IO pins with option 14 mA or 7mA capability and optional pull-high resistor
- ◆ 11 levels of VDD voltage reset: 4.5V, 4.0V, 3.75V, 3.5V, 3.3V, 3.15V, 3.0V, 2.7V, 2.5V, 2.4V, 2.3V
- ◆ 11 levels of VDD voltage detection: 4.5V, 4.0V, 3.75V, 3.5V, 3.3V, 3.15V, 3.0V, 2.7V, 2.5V, 2.4V, 2.3V
- ◆ Selectable four external interrupt pins: PA0 or PA5, PB0 or PB7
- ◆ Every IO pin can be configured to enable wake-up function
- ◆ Operating voltage range: 2.3V ~ 6V

## 1.3. High Performance RISC CPU Array

- ◆ Patented Field Programmable Processor Array (FPPA™) Technology
- ◆ Operating modes: 8 processing units FPPA™ mode
- ◆ 106 powerful instructions
- ◆ Most instructions are 1T execution cycle
- ◆ Programmable stack pointer and adjustable stack level
- ◆ Direct and indirect addressing modes for data access. Data memories are available for use as an index pointer of Indirect addressing mode
- ◆ Support security function to protect MTP data
- ◆ Register space, memory space and MTP space are independent

## 1.4. Ordering/ Package Information

- ◆ PFC886-Y20: SSOP20 (150mil);
  - ◆ PFC886-Y16: SSOP16 (150mil);
  - ◆ PFC886-S16: SOP16(150mil);
  - ◆ PFC886-M10: MSOP10(118mil)
  - ◆ PFC886-1J16A (QFN3\*3-16P-0.5pitch)
- 
- Please refer to the official website file for package size information: "Package information "

## 2. General Description and Block Diagram

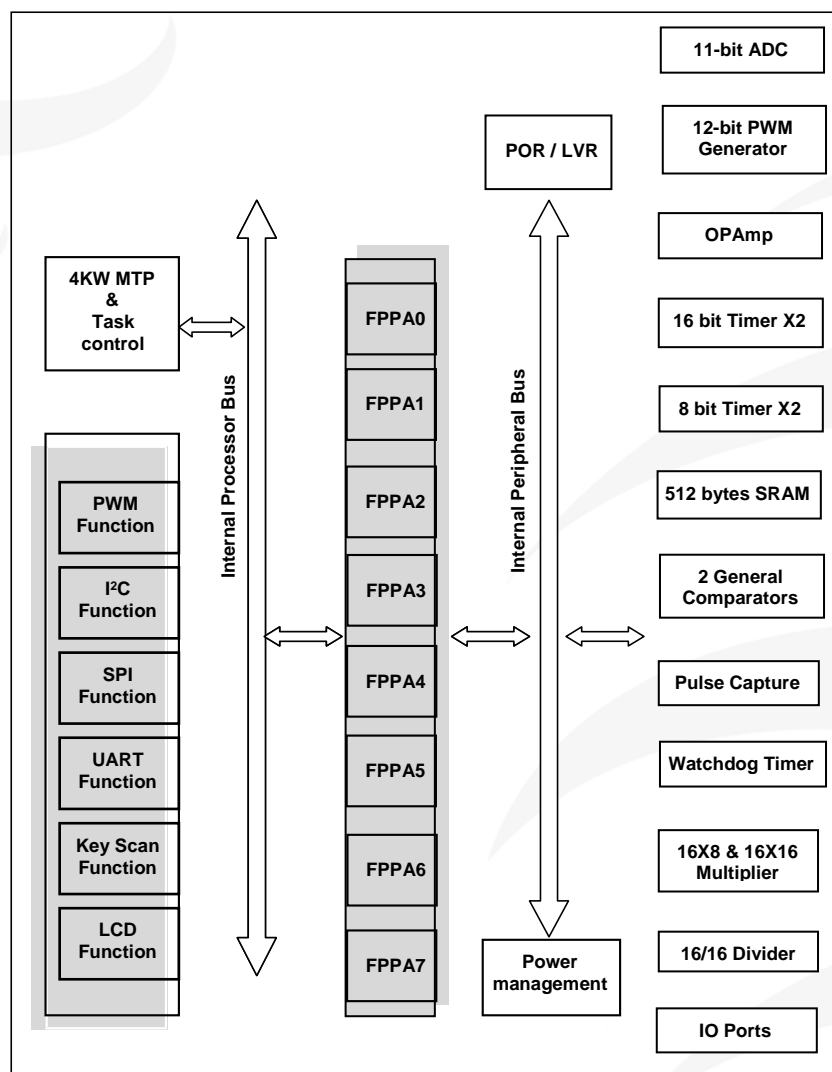
The PFC886 is an ADC-Type of PADAUK's parallel processing, fully static, MTP-based CMOS 8x8 bit processor array that can execute eight peripheral functions in parallel. It employs RISC architecture based on patent pending FPPA™ (Field Programmable Processor Array) technology and all the instructions are executed in one cycle except that some instructions are two cycles that handle indirect memory access.

4KW MTP program memory and 512 bytes data SRAM are inside for 8 FPPA units using.

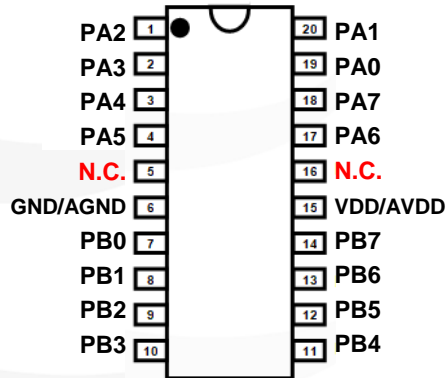
One up to 11 channels 11-bit ADC is built inside the chip with one channel for internal Bandgap reference voltage or  $0.25 \times VDD$ .

PFC886 provide one 8-time voltage gain OPamp, two general purpose comparators and four hardware timers: two are 16-bit timers and two are 8-bit timers.

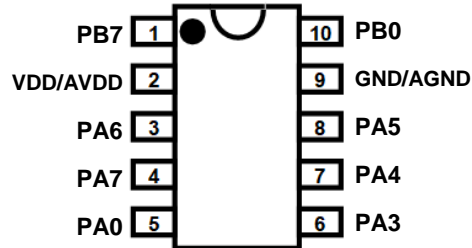
One hardware Pulse Capture, 12-bit hardware PWM generator and PWM protection modules are also built inside the PFC886 in order to provide the best solution for BLDC controller.



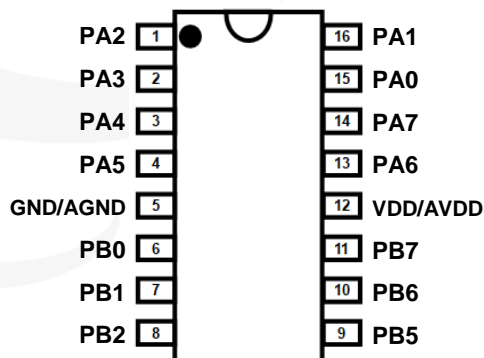
## 3. Pin Definition and Functional Description



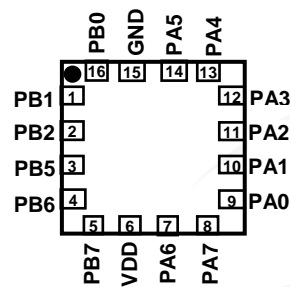
PFC886-Y20: SSOP20 (150 mil)



PFC886-M10: MSOP10 (118 mil)



PFC886-Y16: SSOP16 (150 mil)  
PFC886-S16: SOP16 (150 mil)



PFC886-1J16A (QFN3\*3-16P-0.5pitch)

### Pin description:

Pin Name	Input / output			Special Functions							
	I / O	Pull High	Wake Up	GPC	PWM	Pulse Capture	ADC	OPA	External Interrupt	External Reset	Program
PA0	√	√	√		√				INT0		
PA1	√	√	√		√						
PA2	√	√	√	CMP20	√						
PA3	√	√	√	CMP1-	√		AD9				√
PA4	√	√	√		√		AD10				√
PA5	√	√	√	CMP2+		PCIN			INT2	√	√
PA6	√	√	√					OPO			√
PA7	√	√	√		√						

Pin Name	Input / output			Special Functions							
	I / O	Pull High	Wake Up	GPC	PWM	Pulse Capture	ADC	OPA	External Interrupt	External Reset	Program
PB0	√	√	√	CMP1- CMP2- CMP1O		PCIN	AD0	OPIN-	INT1		
PB1	√	√	√	CMP1- CMP2-		PCIN	AD1				
PB2	√	√	√	CMP1- CMP2-			AD2	OPIN-			
PB3	√	√	√	CMP1-			AD3				
PB4	√	√	√	CMP1- CMP2-			AD4				
PB5	√	√	√	CMP1- CMP2-			AD5	OPIN-			
PB6	√	√	√	CMP1- CMP2-		PCIN	AD6				
PB7	√	√	√	CMP2+		PCIN	AD7		INT3		
VDD AVDD											√
GND AGND											√
<b>Notice</b>	<ol style="list-style-type: none"> <li>1. All the I/O pins have: Schmitt Trigger input and CMOS voltage level.</li> <li>2. IO function is automatically deactivated when a pin is used as PWM output port.</li> <li>3. Please put 33Ω resistor in series to have high noise immunity when PA5 is in input mode.</li> <li>4. VDD is the IC power supply while AVDD is the Analog positive power supply. AVDD and VDD are double bonding internally and they have the same external pin.</li> <li>5. GND is the IC ground pin while AGND is the Analog negative ground pin. AGND and GND are double bonding internally and they have the same external pin.</li> <li>6. <b>Suggest a 1uF and a 0.1uF capacitor in parallel between VDD/AVDD and GND/AGND.</b></li> </ol>										

## 4. Central Processing Unit (CPU)

### 4.1. Functional Description

There are eight processing units (FPPA unit) inside the chip of PFC886. In each processing unit, it includes:

- its own Program Counter to control the program execution sequence
- its own Stack Pointer to store or restore the program counter for program execution
- its own accumulator
- status Flag to record the status of program execution.

Each FPPA unit has its own program counter and accumulator for program execution, flag register to record the status, and stack pointer for jump operation. Based on such architecture, FPPA unit can execute its own program independently, thus parallel processing can be expected.

#### 4.1.1. Processing Units

These eight FPPA units share the same 4Kx16 bits MTP user program memory, 512 bytes data SRAM and all the IO ports, these eight FPPA units are operated at mutual exclusive clock cycles to avoid interference. One task switch is built inside the chip to decide which FPPA unit should be active for the corresponding cycle. The hardware diagram of processing units is illustrated in Fig. 1.

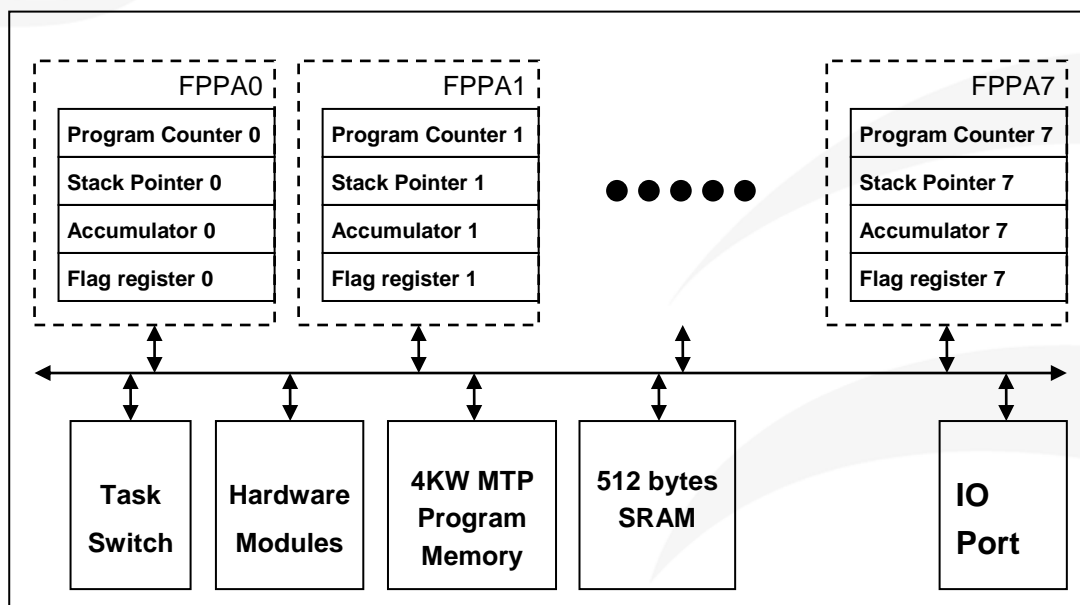


Fig. 1: Hardware Diagram of FPPA units

These eight FPPA units are operated at mutual exclusive clock cycles and can be enabled independently. The system performance is shared to the assigned FPPA units via pmode command; please refer to the description of pmode instruction. The bandwidth assignment is nothing to do with FPPA enable, means that the bandwidth is also allocated to the assigned FPPA unit even though it is disabled.

## Two FPPA units

Fig. 2 shows the timing sequence of FPPA units for  $pmode=0$  which will assign the bandwidth to two FPPA units only. FPPA0 and FPPA1 each have half computing power of whole system; for  $pmode=0$ , FPPA0 and FPPA1 will be operated at 4MHz if system clock is 8MHz.

For FPPA0 unit, its program will be executed in sequence every other system clock, shown as (M-1)<sup>th</sup>, M<sup>th</sup>, .... (M+4)<sup>th</sup> instructions. For FPPA1 unit, its program will be also executed in sequence every other system clock, shown as (N-1)<sup>th</sup>, N<sup>th</sup>, .... (N+3)<sup>th</sup> instructions.

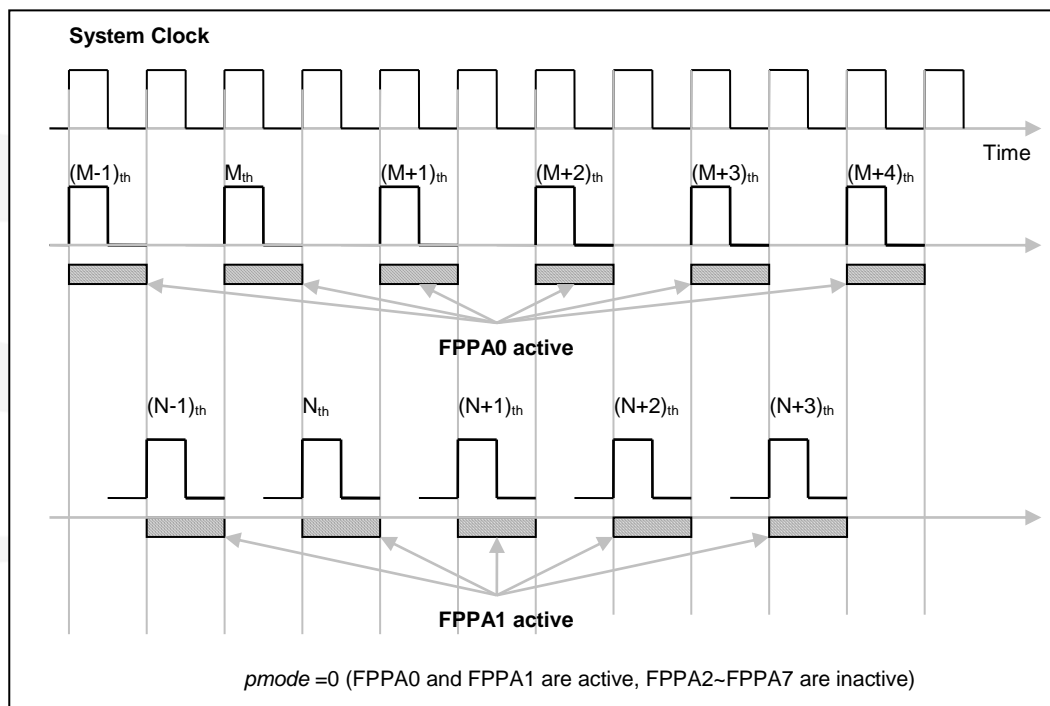


Fig. 2: Timing Sequence of Processing units for  $pmode=0$

## Four FPPA units

Fig. 3 shows the timing sequence of FPPA units for  $pmode=6$  which will assign the bandwidth to four FPPA units (FPPA0, FPPA1, FPPA2, FPPA3); for  $pmode=6$ , FPPA0, FPPA1, FPPA2 and FPPA3 will be operated at 2MHz if system clock is 8MHz, means that each FPPA unit has quarter computing power of whole system, however, FPPA4, FPPA5, FPPA6 and FPPA7 are inactive.

For FPPA0 unit, its program will be executed once in sequence every four system clock, shown as (M-1)<sup>th</sup>, M<sup>th</sup>, .... (M+4)<sup>th</sup> instructions. For FPPA1 unit, its program will be also executed once in sequence every four system clock, shown as (N-1)<sup>th</sup>, N<sup>th</sup>, .... (N+3)<sup>th</sup> instructions. For FPPA2 unit, its program will be also executed once in sequence every four system clock, shown as (O-1)<sup>th</sup>, O<sup>th</sup>, .... (O+3)<sup>th</sup> instructions. For FPPA3 unit, its program will be also executed once in sequence every four system clock, shown as (P-1)<sup>th</sup>, P<sup>th</sup>, .... (P+3)<sup>th</sup> instructions.



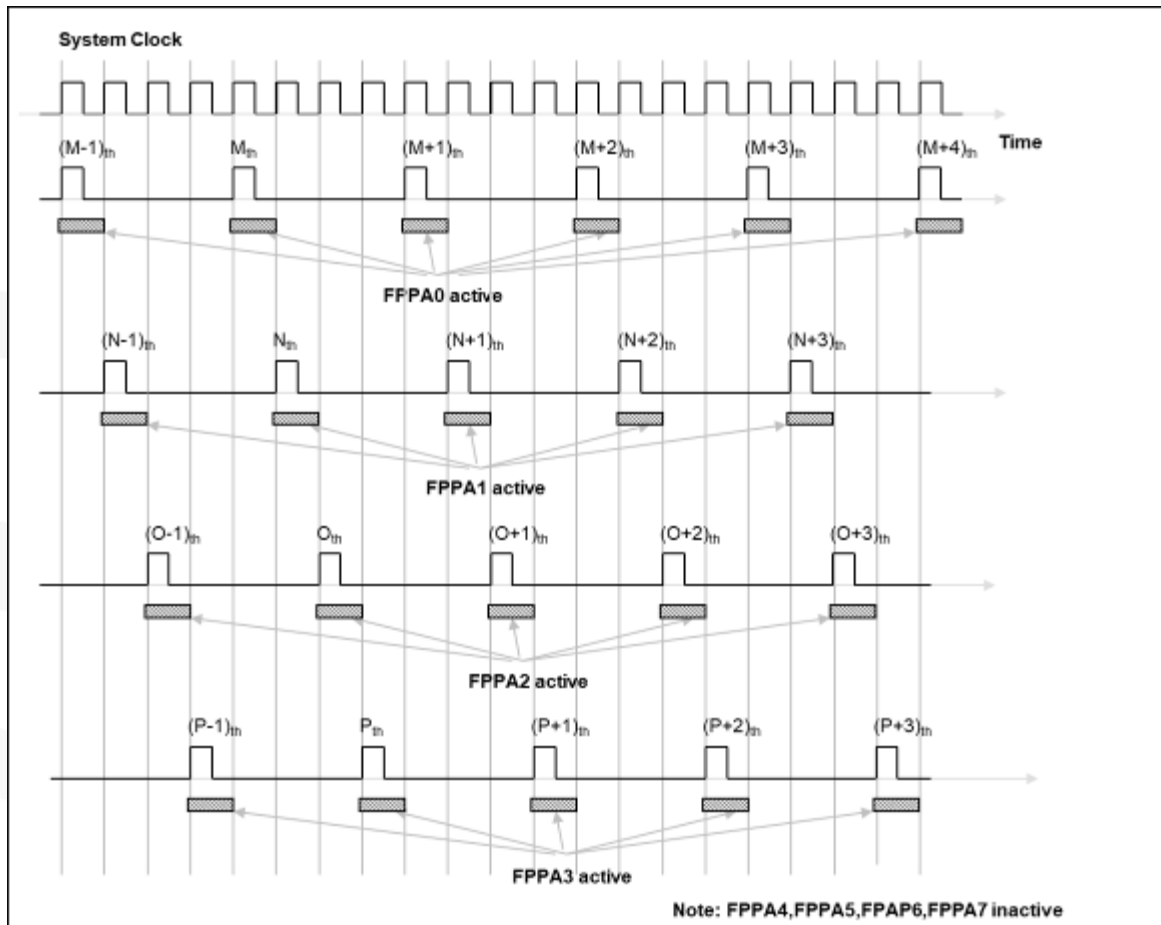


Fig. 3: Timing Sequence of Processing units for  $pmode=6$

The FPPA unit can be enabled or disabled by programming the FPPA unit Enable Register, only FPPA0 is enabled after power-on reset. The system initialization will be started from FPPA0 and other units can be enabled by user's program if necessary. All the FPPA units can be enabled or disabled by using any one FPPA unit, including itself.

## 4.1.2. Program Counter

Program Counter (PC) is the unit that contains the address of an instruction to be executed next. The program counter is automatically incremented at each instruction cycle so that instructions are retrieved sequentially from the program memory. Certain instructions, such as branches and subroutine calls, interrupt the sequence by placing a new value in the program counter.

The bit length of the program counter is 12 for PFC886. The program counter of FPPA0 is 0 after hardware reset, 1 for FPPA1, 2 for FPPA2, 3 for FPPA3, 4 for FPPA4, 5 for FPPA5, 6 for FPPA6 and 7 for FPPA7. Whenever interrupt event happens, only FPPA0 will be informed and its program counter will jump to 0x10 for interrupt service routine. All the FPPA units have its own program counter to control the program execution sequence.

## 4.1.3. Program Structure

After power-up, the program starting address of FPPA0 is 0x000, 0x001 for FPPA1, 0x002 for FPPA2, 0x003 for FPPA3, 0x004 for FPPA4, 0x005 for FPPA5, 0x006 for FPPA6 and 0x007 for FPPA7. The 0x010 is the entry address of interrupt service routine, which belongs to FPPA0 only. The basic firmware structure for PFC886 is shown as Fig. 4, it shows that there are four FPPA units are used; the program codes of four FPPA units are placed in one whole program space. Except for the initial addresses of processing units and entry address of interrupt, the memory location is not specially specified; the program codes of processing unit can be resided at any location no matter what the processing unit is. After power-up, the FPPA0Boot will be executed first, which will include the system initialization and other FPPA units enabled.

<pre>// Page 1 .romadr 0x00 // Program Begin goto FPPA0Boot; goto FPPA1Boot; goto FPPA2Boot; goto FPPA3Boot; //-----Interrupt service Routine----- .romadr 0x010 pushaf ; t0sn intrq.0; //PA.0 ISR goto ISR_PA0; t0sn intrq.1; //PB.0 ISR //-----End of ISR----- //----- Begin of FPPA0 ----- FPPA0Boot : //--- Initialize FPPA0 SP and so on... ... FPPA0Loop:</pre>	<pre>// Page 2 //----- Begin of FPPA1 ----- FPPA1Boot : //--- Initialize FPPA1 SP and so on... FPPA1Loop: ... goto FPPA1Loop: //----- End of FPPA1 ----- //----- Begin of FPPA2 ----- FPPA2Boot : //--- Initialize FPPA2 SP and so on... FPPA2Loop: ... goto FPPA2Loop: //----- End of FPPA2 ----- //----- Begin of FPPA3 ----- FPPA3Boot : //--- Initialize FPPA3 SP and so on... FPPA3Loop:</pre>
---	---

Fig. 4: Program Structure

## 4.1.4. Arithmetic and Logic Unit

Arithmetic and Logic Unit (ALU) is the computation element to operate integer arithmetic, logic, shift and other specialized operations. The operation data can be from instruction, accumulator or SRAM data memory. Computation result could be written into accumulator or SRAM. All the FPPA units share ALU for its corresponding operation.

## 4.2. Storage Memory

### 4.2.1. Program Memory – ROM

The PFC886 program memory is MTP (Multiple Time Programmable), used to store data (including: data, tables and interrupt entry) and program instructions to be executed. All the user program codes for all FPPA units are stored in this 4KW MTP memory which is partitioned as Table 1.

After reset, the initial address for FPPA0 is 0x000, 0x001 for FPPA1, 0x002 for FPPA2, 0x003 for FPPA3, 0x004 for FPPA4, 0x005 for FPPA5, 0x006 for FPPA6, 0x007 for FPPA7. The interrupt entry is 0x010 if used and interrupt function is for FPPA0 only.

The MTP memory from address 0xFF8 to 0xFF are for system using, address space from 0x008 to 0x00F and from 0x011 to 0xFF7 are user program spaces. And the address 0x000 to 0x007 is the FPPA units initial address.

Address	Function
0x000	FPPA0 reset – <i>goto</i> instruction
0x001	FPPA1 reset – <i>goto</i> instruction
0x002	FPPA2 reset – <i>goto</i> instruction
0x003	FPPA3 reset – <i>goto</i> instruction
0x004	FPPA4 reset – <i>goto</i> instruction
0x005	FPPA5 reset – <i>goto</i> instruction
0x006	FPPA6 reset – <i>goto</i> instruction
0x007	FPPA7 reset – <i>goto</i> instruction
0x008	User program memory
•	•
0x00F	User program memory
0x010	Interrupt entry address
0x011	User program memory
•	•
•	•
0xFF7	User program memory
0xFF8	System using
•	•
0xFFFF	System using

Table 1: Program Memory Organization

## Example of Using Program Memory for Two FPPA mode

In order to have maximum flexibility for user program using, the user program memory is shared for all FPPA units, and the program space allocation is done by program compiler automatically, user does not need to specify the address if not necessary. Table 2 shows one example of program memory using which two FPPA units are used.

Address	Function
0x000	FPPA0 reset – goto instruction ( <i>goto</i> 0x020)
0x001	FPPA1 reset – goto instruction ( <i>goto</i> 0x7A1)
0x002	Reserved
•	•
0x007	Reserved
0x008	Not used
•	•
0x00F	Not used
0x010	Interrupt entry address(FPPA0 only)
•	•
0x01F	End of ISR (depend on user code size)
0x020	Begin of FPPA0 user program
•	•
•	•
0x7A0	End of FPPA0 user program
0x7A1	Begin of FPPA1 program
•	•
•	•
0xF37	End of FPPA1 program
0xF38	Not used
•	•
•	•
0xFF7	Not used
0xFF8	System Using
•	•
0xFFFF	System Using

Table 2: Example of Program Memory Using

## 4.2.2. Data Memory – SRAM

Fig. 5 shows the SRAM data memory organization of PFC886, all the SRAM data memory could be accessed by every FPPA unit directly with 1T clock cycle. The access of data memory can be byte or bit operation. Besides data storage, the SRAM data memory is also served as data pointer of indirect access method and the stack memory for all FPPA units.

The stack memory for each processing unit should be independent from each other, and defined in the data memory. The stack pointer is defined in the stack pointer register of each processing unit; the depth of stack memory of each processing unit is defined by the user. The arrangement of stack memory fully flexible and can be dynamically adjusted by the user.

For indirect memory access mechanism, the data memory is used as the data pointer to address the data byte. All the data memory could be the data pointer; it's quite flexible and useful to do the indirect memory access. All the 512 bytes SRAM data memory of PFC886 can be accessed by indirect access mechanism.

Bit defined: Only addressed at 0x00 ~ 0x3F.

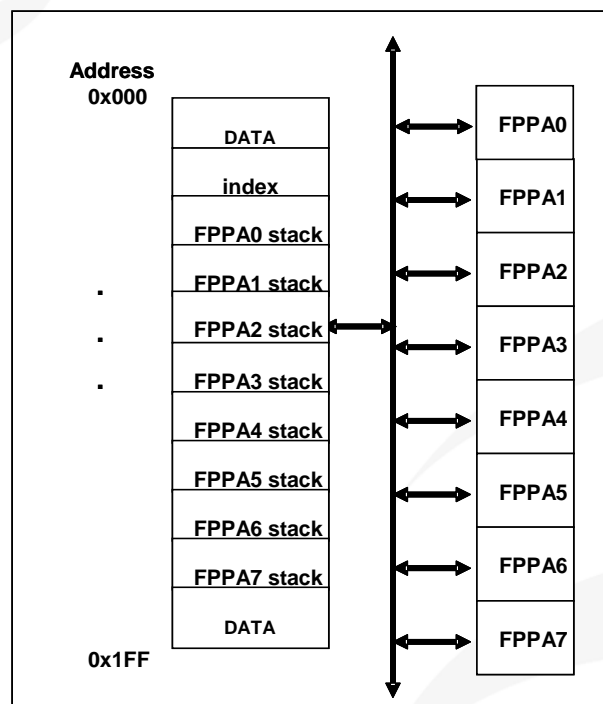


Fig. 5: SRAM Data Memory Organization

## 4.2.3. System Register

The register space of PFC886 is independent of SRAM space and MTP space.

The following is the PFC886 register address and brief description:

	+0	+1	+2	+3	+4	+5	+6	+7
0x00	FLAG	FPPEN	SP	CLKMD	INTEN	INTRQ	T16M	GDIO
0x08	INTEN2	INTRQ2	-	-	INTEGS	PADIER	PBDIER	-
0x10	PA	PAC	PAPH	-	PB	PBC	PBPH	-
0x18	AMPC	T16NM	-	-	-	-	GPC2C	GPC2S
0x20	ADCC	ADCM	-	HOP	TM2C	TM2CT	TM2S	TM2B
0x28	TM3C	TM3CT	TM3S	TM3B	-	-	-	-
0x30	PWMGC	PWMGM	PLSCC	PLSCS	GPC1C	GPC1S	TMOTP	PWMGC1
0x38	PWMGC2	-	OPR2	MISC	EARITH			-
0x40	-	-	-	-	M8OP1H	M8OP1L	M8OP2	M8RS2
0x48	M8RS1	M8RS0	ADCRH	ADCRL	PLSPWH	PLSPWL	PLSPHH	PLSPHL
0x50	PWMCUBH	PWMCUBL	PWM0DTH	PWM0DTL	PWM1DTH	PWM1DTL	PWM2DTH	PWM2DTL
0x58	PWMDZ	-	-	-	-	-	PWMADCH	PWMADCL
0x60	M16OP1H	M16OP1L	M16OP2H	M16OP2L	M16RS3	M16RS2	M16RS1	M16RS0
0x68	D16DEH	D16DEL	D16DSH	D16DSL	D16QUH	D16QUL	D16REH	D16REL
0x70				T16NBH	T16NBL	-		-
0x78		-	-	-	-	-	-	-

## 4.2.3.1.ACC Status Flag Register (*FLAG*), address = 0x00

Bit	Reset	R/W	Description
7 - 4	-	-	Reserved. These four bits are "1" when read.
3	-	R/W	OV (Overflow). This bit is set whenever the sign operation is overflow.
2	-	R/W	AC (Auxiliary Carry). There are two conditions to set this bit, the first one is carry out of low nibble in addition operation, and the other one is borrow from the high nibble into low nibble in subtraction operation.
1	-	R/W	C (Carry). There are two conditions to set this bit, the first one is carry out in addition operation, and the other one is borrow in subtraction operation. Carry is also affected by shift with carry instruction.
0	-	R/W	Z (Zero). This bit will be set when the result of arithmetic or logic operation is zero; Otherwise, it is cleared.

## 4.2.3.2.FPPA unit Enable Register (*FPPEN*), address = 0x01

Bit	Reset	R/W	Description
7	0	R/W	FPPA7 enable. This bit is used to enable FPPA7. 0 / 1: disable / enable
6	0	R/W	FPPA6 enable. This bit is used to enable FPPA6. 0 / 1: disable / enable
5	0	R/W	FPPA5 enable. This bit is used to enable FPPA5. 0 / 1: disable / enable
4	0	R/W	FPPA4 enable. This bit is used to enable FPPA4. 0 / 1: disable / enable
3	0	R/W	FPPA3 enable. This bit is used to enable FPPA3. 0 / 1: disable / enable
2	0	R/W	FPPA2 enable. This bit is used to enable FPPA2. 0 / 1: disable / enable
1	0	R/W	FPPA1 enable. This bit is used to enable FPPA1. 0 / 1: disable / enable
0	1	R/W	FPPA0 enable. This bit is used to enable FPPA0. 0 / 1: disable / enable

## 4.2.3.3.Hopping Setting Register (*HOP*), address = 0x23

Bit	Reset	R/W	Description
7	0	R/W	PWM interrupt mode 0: Generate interrupt request when counter matches the boundary value 1: Generate interrupt request when counter is 0
6	0	R/W	PWM counter trigger ADC mode 0: up count matched PWMADC register 1: down count matched PWMADC register
5	0	R/W	Hall interrupt pins selection 0: PB0/PB1/PB2 1: PB5/PB6/PB7
4	0	R/W	ADC can trigger by PWM (Hardware clear by ADC done) 0: Disable 1: Enable
3 - 0	-	-	Reserved

## 4.2.3.4.Option 2 Register (*OPR2*), address = 0x3A

Bit	Reset	R/W	Description
7 - 4	-	-	Reserved. Please keep 0
3	0	WO	Option for Timer16N clock pre-divider selection. Please see the <i>T16NM</i> register
2	0	WO	Option for Timer16 clock pre-divider selection. Please see the <i>T16M</i> register
1	0	WO	Option for external interrupt 1 pin selection
0	0	WO	Option for external interrupt 0 pin selection

## 4.2.3.5.MISC Register (*MISC*), address = 0x3B

Bit	Reset	R/W	Description
2	0	WO	Disable LVR function. 0 / 1 : Enable / Disable
1 - 0	00	WO	Watch dog time out period 00: Reserved 01: 4096 ILRC clock period 10: 16384 ILRC clock period 11: Reserved

## 4.3. The Stack

The stack pointer in each processing unit is used to point the top of the stack area where the local variables and parameters to subroutines are stored; the stack pointer register (*SP*) is located in register address 0x02. The bit number of stack pointer is 8 bit; the stack memory cannot be accessed over 512 bytes and should be defined within 512 bytes from 0x00 address. The stack memory of PFC886 for each FPPA unit can be assigned by user via stack pointer register, means that the depth of stack pointer for each FPPA unit is adjustable in order to optimize system performance. The following example shows how to define the stack in the ASM (assembly language) project:

```

.ROMADR0
GOTO      FPPA0
GOTO      FPPA1
...
.RAMADR0                                     // Address must be less than 0x100
WORD      Stack0 [1]                       // one WORD
WORD      Stack1 [2]                       // two WORD
...
FPPA0:
SP  =      Stack0;                         // assign Stack0 for FPPA0,
                                           // one level call because of Stack0[1]

...
call      function1
...
FPPA1:
SP  =      Stack1;                         // assign Stack1 for FPPA1,
                                           // two level call because of Stack1[2]

...
call      function2
...

```



In Mini-C project, the stack calculation is done by system software, user will not have effort on it, and the example is shown as below:

```
void      FPPA0 (void)
{
    ...
}
```

User can check the stack assignment in the window of program disassembling, Fig. 6 shows that the status of stack before FPPA0 execution, system has calculated the required stack space and has reserved for the program.

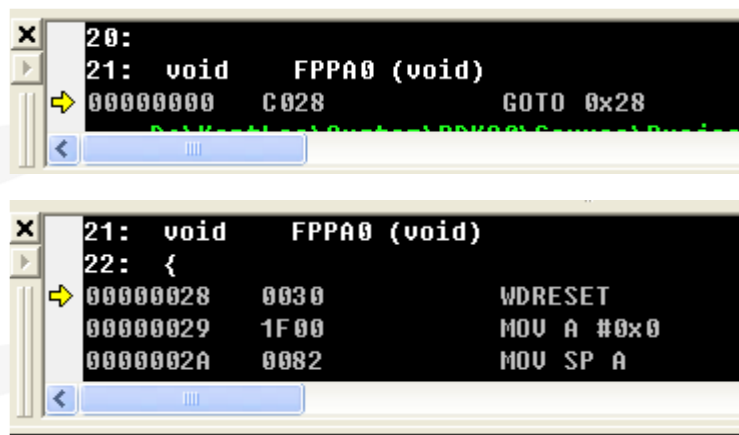


Fig. 6: Stack Assignment in Mini-C project

#### 4.3.1. Stack Pointer Register (SP), address = 0x02

Bit	Reset	R/W	Description
7 - 0	-	R/W	Stack Pointer Register. Read out the current stack pointer, or write to change the stack pointer.

#### 4.4. Code Options

Option	Selection	Description
Security	Enable	MTP content is protected and program cannot be read back
	<b>Disable(default)</b>	MTP content is not protected so program can be read back
LVR	4.5V	Select LVR = 4.5V
	4.0V	Select LVR = 4.0V
	3.75V	Select LVR = 3.75V
	3.5V	Select LVR = 3.5V
	3.3V	Select LVR = 3.3V
	3.15V	Select LVR = 3.15V
Drive	Low	IO Drive / Sink Current is low
	<b>Normal(default)</b>	IO Drive / Sink Current is Normal

## 5. Oscillator and System Clock

There are two oscillator circuits provided by PFC886: internal high RC oscillator (IHRC) and internal low RC oscillator (ILRC).

These two oscillators are enabled or disabled by registers *CLKMD.4* and *CLKMD.2* independently. User can choose one of these two oscillators as system clock source and use *CLKMD* register to target the desired frequency as system clock to meet different applications.

Oscillator Module	Enable / Disable	Default after boot-up
IHRC	<i>CLKMD.4</i>	Enabled
ILRC	<i>CLKMD.2</i>	Enabled

Table 3: Two Oscillator Circuits provided by PFC886

### 5.1. Internal High RC Oscillator and Internal Low RC Oscillator

After boot-up, the IHRC and ILRC oscillators are enabled. The frequency of IHRC can be calibrated to eliminate process variation by *IHRCR* register; normally it is calibrated to 16MHz. The frequency deviation can be within 2% normally after calibration and it still drifts slightly with supply voltage and operating temperature. The frequency of ILRC will also vary by process, supply voltage and temperature. Please refer to the measurement chart for IHRC / ILRC frequency verse  $V_{DD}$  and IHRC / ILRC frequency verse temperature.

The PFC886 writer tool provides IHRC frequency calibration (usually up to 16MHz) to eliminate frequency drift caused by factory production. ILRC has no calibration operation. For applications that require accurate timing, please do not use the ILRC clock as a reference time.

### 5.2. System Clock and IHRC Calibration

#### 5.2.1. System Clock

The clock source of system clock comes from IHRC or ILRC, the hardware diagram of system clock in the PFC886 is shown as Fig. 7.

After power up, the running system clock will be ILRC/1, user can change the system clock any time by setting *CLKMD* register and the new system clock will be changed into the new one immediately after writing the *CLKMD* register.

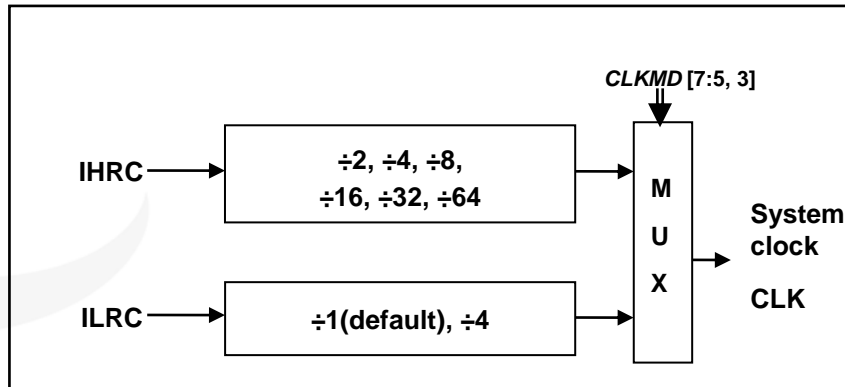


Fig. 7: Options of System Clock

## 5.2.1.1. Clock Mode Register (*CLKMD*), address = 0x03

Bit	Reset	R/W	Description	
7 - 5	111	R/W	System clock selection	
			Type 0, <i>CLKMD</i> [3]=0	Type 1, <i>CLKMD</i> [3]=1
			000: IHRC/4	000: IHRC/16
			001: IHRC/2	001: IHRC/8
			010: reserved	010: reserved
			011: reserved	011: IHRC/32
			100: reserved	100: IHRC/64
			101: reserved	101: reserved
			110: ILRC/4	11x: reserved
111: ILRC (default)				
4	1	R/W	IHRC oscillator Enable. 0 / 1: disable / enable	
3	0	R/W	Clock Type Select. This bit is used to select the clock type in bit [7:5]. 0 / 1: Type 0 / Type 1	
2	1	R/W	ILRC Enable. 0 / 1: disable / enable If ILRC is disabled, watchdog timer is also disabled.	
1	1	R/W	Watch Dog Enable. 0 / 1: disable / enable	
0	0	R/W	Pin PA5/PRSTB function. 0 / 1: PA5 / PRSTB	

## 5.2.2. Frequency Calibration

The IHRC frequency and Bandgap reference voltage may be different chip by chip due to manufacturing variation, PFC886 provide both the IHRC frequency calibration and Band-gap calibration to eliminate this variation, and this function can be selected when compiling user's program and the command will be inserted into user's program automatically.

The calibration command is shown as below:

**.ADJUST\_IC SYSClk=IHRC/(p1), IHRC=(p2)MHz, V<sub>DD</sub>=(p3)V, Band-gap=(p4);**

Where,

**p1**= 2, 4, 8, 16, 32, 64; In order to provide different system clock.

**p2**= 16 ~ 18; In order to calibrate the chip to different frequency, 16MHz is the usually one.

**p3**= 3.15 ~ 5.5; In order to calibrate the chip under different supply voltage.

**p4**= On or Off; Band-gap calibration is On or Off.

Usually, .ADJUST\_IC will be the first command after boot up, in order to set the target operating frequency whenever starting the system. The program code for IHRC frequency calibration is executed only one time that occurs in writing the codes into MTP memory; after then, it will not be executed again.

If the different option for IHRC calibration is chosen, the system status is also different after boot. As shown in table 4:

SYSClk	CLKMD	IHRCR	Description
○ Set IHRC / 2	= 34h (IHRC / 2)	Calibrated	IHRC calibrated to 16MHz, CLK=8MHz (IHRC/2)
○ Set IHRC / 4	= 14h (IHRC / 4)	Calibrated	IHRC calibrated to 16MHz, CLK=4MHz (IHRC/4)
○ Set IHRC / 8	= 3Ch (IHRC / 8)	Calibrated	IHRC calibrated to 16MHz, CLK=2MHz (IHRC/8)
○ Set IHRC / 16	= 1Ch (IHRC / 16)	Calibrated	IHRC calibrated to 16MHz, CLK=1MHz (IHRC/16)
○ Set IHRC / 32	= 7Ch (IHRC / 32)	Calibrated	IHRC calibrated to 16MHz, CLK=0.5MHz (IHRC/32)
○ Set ILRC	= E4h (ILRC / 1)	Calibrated	IHRC calibrated to 16MHz, CLK=ILRC
○ Disable	No change	No Change	IHRC not calibrated, CLK not changed

Table 4: Options for IHRC Frequency Calibration

The following shows the different states of PFC886 under different options:

### **(1) .ADJUST\_IC SYSClk=IHRC/4, IHRC=16MHz, V<sub>DD</sub>=5V**

After boot, CLKMD = 0x14:

- IHRC frequency is calibrated to 16MHz@V<sub>DD</sub>=5V and IHRC module is enabled
- System CLK = IHRC/4 = 4MHz
- Watchdog timer is disabled, ILRC is enabled, PA5 is in input mode

### **(2) .ADJUST\_IC SYSClk=ILRC, IHRC=16MHz, V<sub>DD</sub>=5V**

After boot, CLKMD = 0xE4:

- IHRC frequency is calibrated to 16MHz@V<sub>DD</sub>=5V and IHRC module is disabled
- System CLK = ILRC
- Watchdog timer is disabled, ILRC is enabled, PA5 is in input mode

### (3) .ADJUST\_IC DISABLE

After boot, *CLKMD* is not changed (Do nothing):

- a. IHRC is not calibrated and IHRC module is disabled
- b. System CLK = ILRC
- c. Watchdog timer is enabled, ILRC is enabled, PA5 is in input mode

#### 5.2.2.1.Special Statement

- (1) The IHRC frequency calibration is performed when IC is programmed by the writer.
- (2) Because the characteristic of the Epoxy Molding Compound (EMC) would some degrees affects the IHRC frequency (either for package or COB), if the calibration is done before molding process, the actual IHRC frequency after molding may be deviated or becomes out of spec. Normally, the frequency is getting slower a bit.
- (3) It usually happens in COB package or Quick Turnover Programming (QTP). And PADAUK would not take any responsibility for this situation.
- (4) Users can make some compensatory adjustments according to their own experiences. For example, users can set IHRC frequency to be 0.5% ~ 1% higher and aim to get better re-targeting after molding.

#### 5.2.3. System Clock Switching

After IHRC calibration, the system clock of PFC886 can be switched between IHRC and ILRC by setting the *CLKMD* register at any time, **but please notice that the original clock module can NOT be turned off at the same time as writing command to *CLKMD* register.** For example, when switching from A clock source to B clock source, you should first switch the system clock source to B and then close the A clock source. The examples are shown as below and more information about clock switching, please refer to the "Help" -> "Application Note" -> "IC Introduction" -> "Register Introduction" -> *CLKMD*.

##### Case 1: Switching system clock from ILRC to IHRC/4

```
...                               // system clock is ILRC
CLKMD.4      = 1;                // turn on IHRC first to improve anti-interference ability
CLKMD        = 0x14;            // switch to IHRC/4, ILRC CAN NOT be disabled here
// CLKMD.2    = 0;                // if need, ILRC CAN be disabled at this time
...
```

##### Case 2: Switching system clock from IHRC/8 to IHRC/4

```
...                               // system clock is IHRC/8, ILRC is enabled here
CLKMD        = 0x14;            // switch to IHRC/4
...
```

##### Case 3: System may hang if it is to switch clock and turn off original oscillator at the same time

```
...                               // system clock is ILRC
CLKMD        = 0x10;            // CAN NOT switch clock from ILRC to IHRC/4 and turn off
                                // ILRC oscillator at the same time
...
```

## 6. Reset

There are many causes to reset the PFC886, the main four factors are: power-on reset, LVR reset, watchdog timeout overflow reset, and PRSTB pin reset. After the reset, the system will restart. The program counter will jump to address 0x000.

After a power-on reset or LVR reset occurs, if VDD is greater than VDR (data storage voltage), the value of the data memory will be retained, but if the SRAM is cleared after re-power, the data cannot be retained; if VDD is less than VDR, the data The value of the memory will be turned into an unknown state that is in an indeterminate state.

If a reset occurs, and there is an instruction or syntax to clear SRAM in the program, the previous data will be cleared during program initialization and cannot be retained.

The content will be kept when reset comes from PRSTB pin or WDT timeout.

### 6.1. Power On Reset - POR

POR (Power-On-Reset) is used to reset PFC886 when power up, however, the supply voltage may be not stable. To ensure the stability of supply voltage after power up, it will wait 1024 ILRC clock cycles before first instruction being executed, which is  $T_{SBP}$  and shown in the Fig. 8. After boot up procedure, the default system clock is ILRC. If user wants to switch the system clock source from ILRC to IHRC, user must enable the corresponding oscillator module and make sure clock is already stable.

The PFC886 data memory is in an uncertain state when the power on reset occurs.

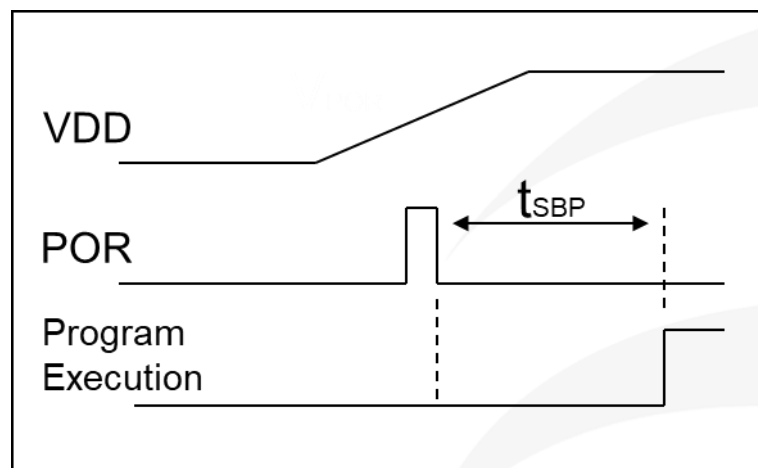


Fig. 8: Power-On Sequence

Fig. 9 shows the typical program flow after boot up, it shows all the FPPA units are used. Please notice that the FPPA1~FPPA7 is disabled after reset and recommend NOT enabling FPPA1~FPPA7 before system and FPPA0 initialization.

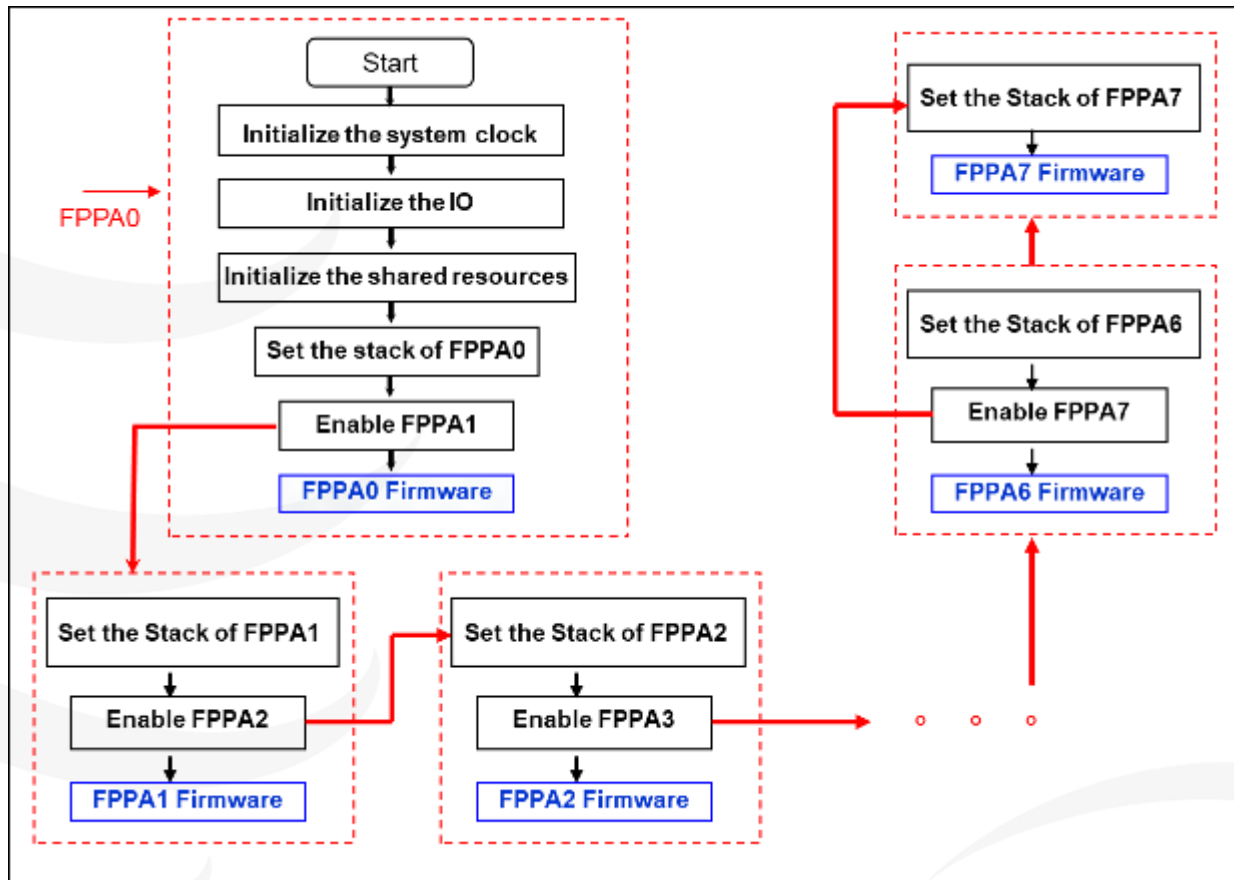


Fig. 9: Boot Procedure

## 6.1.1. POR for DC Fan Application

As shown in Fig 10, PFC886 generates a good power on reset (POR) signal when VDD rises from 0V to 5V in TPOR (max. 50ms) and rises through the voltage range of 0.7V to 1.6V in TFSV (max. 10ms).

If there are a lot of abnormal power noises in VDD power-on time period and TPOR and TFSV do not meet the specifications, PFC886 is not able to guarantee circuit initialization and may cause a malfunction.

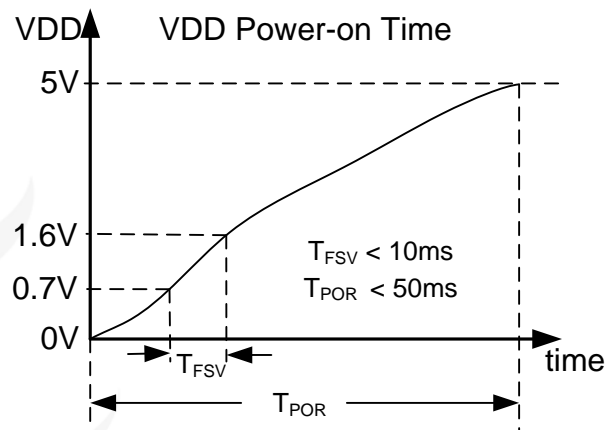


Fig. 10

During power-off as shown in Fig 11 and Fig 12, VDD has to be discharged to  $V_{PDRV}$  (max. is 0.7V) for the next power-on. In case VDD is more than  $V_{PDRV}$ , it is not recognized to do the next power-on.

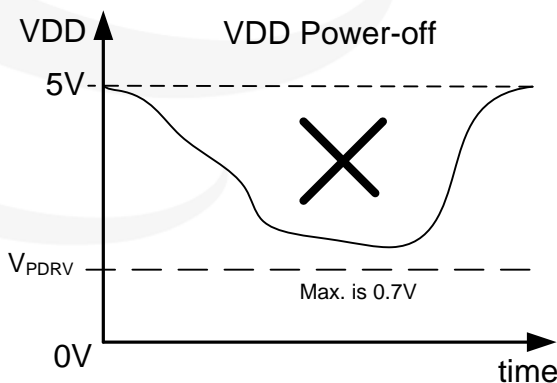


Fig. 11

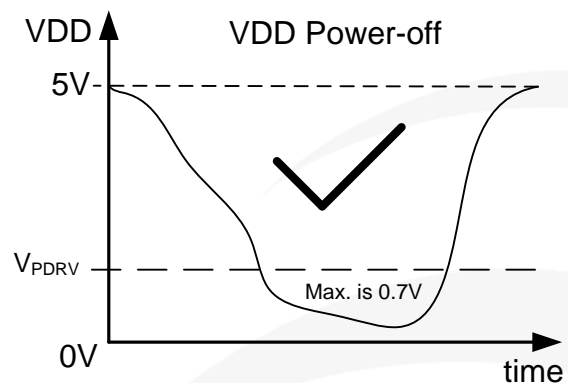


Fig. 12

## NOTICE:

It may cause PFC886 malfunction, fail or crash when power-on and power-off do not meet the specifications. Another proper POR is needed in order to get rid of such status.

## 6.2. Low Voltage Reset - LVR

User can choose different operating system clock depends on its requirement; the selected operating system clock should be in conjunction with supply voltage and LVR level to ensure system stable. If VDD drops below the LVR level, then the LVR Reset will occur in the system, and its timing diagram is shown in Fig. 13.



After LVR reset, the SRAM data will be kept when  $VDD > VDR$  (SRAM data retention voltage). However, if SRAM is cleared after power-on again, the data cannot be kept, the data memory is in an uncertain state when  $VDD < VDR$ .

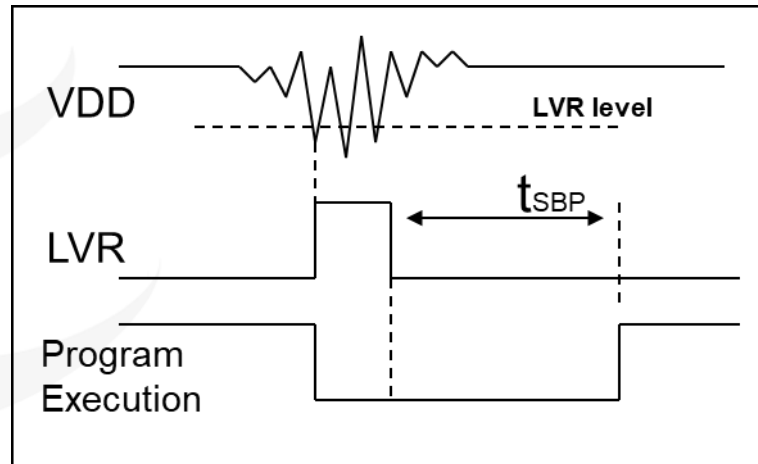


Fig. 13: Low Voltage Reset Sequence

User can choose different operating system clock depends on its requirement; the selected operating system clock should be combined with supply voltage and LVR level to make system stable. The LVR level will be selected during compilation, and the lowest LVR levels can be chosen for different operating frequencies. Please refer to Section 13.1.

The following are suggestions for setting operating frequency, power supply voltage and LVR level:

SYSCLK	VDD	LVR
8MHz	$\geq 3.5V$	3.5V
4MHz	$\geq 2.3V$	2.3V

Table 5: LVR setting for reference

Users can set *MISC.2* as 1 to disable the LVR function. At this time, it should be ensured that the VDD is above the minimum working voltage of the chip, otherwise the IC may not work properly.

MISC Register ( <i>MISC</i> ), address = 0x08			
Bit	Reset	R/W	Description
2	0	WO	Disable LVR function 0 / 1 : Enable / Disable
1 - 0	00	WO	Watch dog time out period 00: Reserved 01: 4096 ILRC clock period 10: 16384 ILRC clock period 11: Reserved

## 6.3. Watch Dog Timeout Reset

The watchdog timer (WDT) is a counter with clock coming from ILRC, so it will be invalid if ILRC is off. The frequency of ILRC may drift a lot due to the variation of manufacture, supply voltage and temperature. User should reserve guard band for safe operation.

Besides, watchdog is open by default, but when the program executes `ADJUST_IC`, the watchdog will be closed. To use the watchdog, you need to reconfigure the open. Watchdog will be inactive once ILRC is disabled.. It is suggested to clear WDT by `wdreset` command after these events to ensure enough clock periods before WDT timeout.

To ensure the watchdog is cleared before the timeout overflow, the instruction `wdreset` can be used to clear the WDT within a safe time. WDT can be cleared by power-on-reset or by command `wdreset` at any time.

When WDT is timeout, PFC886 will be reset to restart the program execution. The relative timing diagram of watchdog timer is shown as Fig. 14.

The PFC886 data memory will be reserved and the *GDIO* register (address 0x07) will keep the same value when the WDT reset occurs.

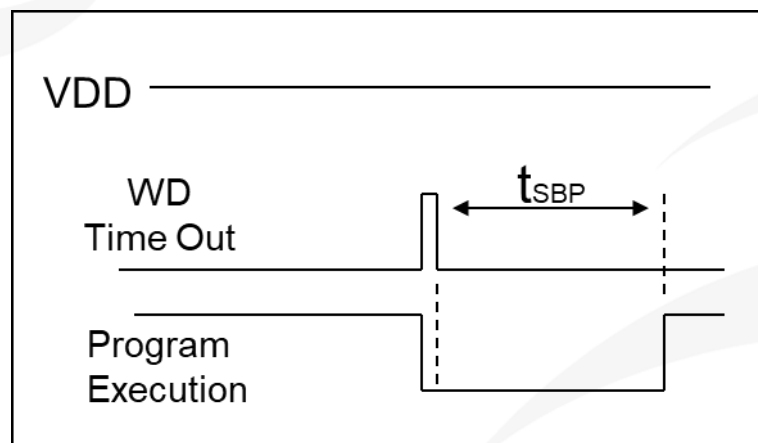


Fig. 14: Sequence of Watch Dog Timeout reset

There are two different timeout periods of watchdog timer can be chosen by setting the *MISC*[1:0]. And watchdog timer can be disabled by *CLKMD*.1.

Clock Mode Register ( <i>CLKMD</i> ), address = 0x03			
Bit	Reset	R/W	Description
7 – 5	111	R/W	System clock selection
4	1	R/W	IHRC oscillator Enable. 0 / 1: disable / enable
3	0	R/W	Clock Type Select.
2	1	R/W	ILRC Enable. 0 / 1: disable / enable If ILRC is disabled, watchdog timer is also disabled.
1	1	R/W	Watch Dog Enable. 0 / 1: disable / enable
0	0	R/W	Pin PA5/PRSTB function. 0 / 1: PA5 / PRSTB

## 6.4. External Reset Pin - PRSTB

The PFC886 supports external reset and its external reset pin shares the same IO port with PA5. Using external reset function requires:

- (1) Set PA5 as input;
- (2) Set  $CLKMD.0 = 1$  to make PA5 as the external PRSTB input pin.

When the PRSTB pin is high, the system is in normal working state. Once the reset pin detects a low level, the system will be reset. The timing diagram of PRSTB reset is shown in Fig. 15.

The PFC886 data memory will be reserved when the PRSTB reset occurs.

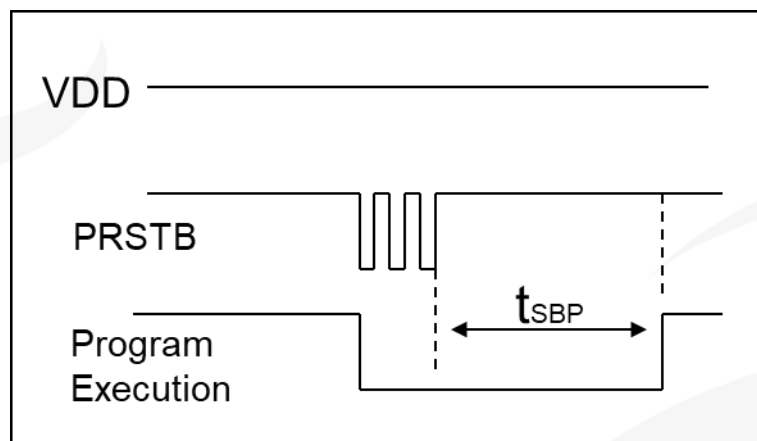


Fig. 15: Sequence of PRSTB reset

## 7. System Operating Mode

There are three operational modes defined by hardware:

- (1) ON mode
- (2) Power-Save mode
- (3) Power-Down mode

ON mode is the state of normal operation with all functions ON.

Power-Save mode (*stopexe*) is the state to reduce operating current and CPU keeps ready to continue.

Power-Down mode (*stopsys*) is used to save power deeply.

Therefore, Power-Save mode is used in the system which needs low operating power with wake-up periodically and Power-Down mode is used in the system which needs power down deeply with seldom wake-up.

## 7.1. Power-Save Mode (“stopexe”)

Using *stopexe* instruction to enter the Power-Save mode, only system clock is disabled, remaining all the oscillator modules be active. So only the CPU stops executing instructions. Wake-up from input pins can be considered as a continuation of normal execution, and *nop* command is recommended to follow the *stopexe* command.

The detail information for Power-Save mode shown below:

- (1) IHRC and oscillator modules: No change, keep active if it was enabled
- (2) ILRC oscillator modules: must remain enabled, need to start with ILRC when waking up
- (3) System clock: Disable, therefore, CPU stops execution
- (4) MTP memory is turned off.
- (5) Timer counter: Stop counting if its clock source is system clock or the corresponding oscillator module is disabled; otherwise, it keeps counting. (The Timer only contains Timer16. Please notice that Timer16N TM2, TM3 and PWMG have no wake up function.)
- (6) Wake-up sources:
  - a. IO toggle wake-up: IO toggling in digital input mode (*PxC* bit is 0 and *PxDIER* bit is 1)
  - b. Timer wake-up: If the clock source of Timer is not the SYSCLK, the system will be awakened when the Timer counter reaches the set value.
  - c. Comparator wake-up: It need setting *GPCC.7=1* (*GPC1C/GPC2C*) and *GPCS.6=1* (*GPC1S/GPC2S*) to enable the comparator wake-up function at the same time.

Please note: the internal 1.20V bandgap reference voltage is not suitable for the comparator wake-up function.

The watchdog timer must be disabled before issuing the *stopexe* command, the example is shown as below:

```
CLKMD.En_WatchDog = 0;      // disable watchdog timer
stopexe;
nop;
....                          // power saving
Wdreset;
CLKMD.En_WatchDog = 1;      // enable watchdog timer
```

An example shows how to use Timer16 to wake-up from “*stopexe*”:

```
$ T16M  ILRC, /1, BIT8      // Timer16 setting
...
WORD   count   = 0;
STT16  count;
stopexe;
...
```

The initial counting value of Timer16 is zero and the system will be woken up after the Timer16 counts 256 ILRC clocks.

## 7.2. Power-Down Mode (“*stopsys*”)

Power-Down mode is the state of deeply power-saving with turning off all the oscillator modules. By using the *stopsys* instruction, this chip will be put on Power-Down mode directly. The internal low frequency RC oscillator must be enabled before entering the Power-Down mode, means that bit 2 of register *CLKMD* must be set to high before issuing *stopsys* command in order to resume the system when wakeup. It is recommend to set *GPCC.7=0*(*GPC1C/GPC2C*) to disable the comparator before the command *stopsys*.

Wake-up from input pins can be considered as a continuation of normal execution. To minimize power consumption, all the I/O pins should be carefully manipulated before entering Power-Down mode.

The following shows the internal status of PFC886 in detail when *stopsys* command is issued:

- (1) All the oscillator modules are turned off
- (2) Enable internal low RC oscillator (set bit 2 of register *CLKMD*)
- (3) MTP memory is turned off
- (4) The contents of SRAM and registers remain unchanged
- (5) Wake-up sources: IO toggle in digital mode (*PxDIER* bit is 1)

The reference sample program for power down mode is shown as below:

```

CLKMD  =  0xF4;           //  Change clock from IHRC to ILRC, disable watchdog timer
CLKMD.4 =  0;             //  disable IHRC
...
while (1)
{
    STOPSYS;                //  enter Power-Down mode
    if (...) break;          //  if wake-up happen and check OK, then return to high speed,
                                //  else stay in Power-Down mode again.
}
CLKMD  =  0x14;           //  Change clock from ILRC to IHRC/4

```

## 7.3. Wake-Up

After entering the Power-Down or Power-Save modes, the PFC886 can be resumed to normal operation by toggling IO pins. Wake-up from timer are available for Power-Save mode ONLY. Table 6 shows the differences in wake-up sources between *stopsys* and *stopexe*.

Differences in wake-up sources between <i>stopsys</i> and <i>stopexe</i>		
	IO Toggle	Timer wake-up
<i>stopsys</i>	Yes	No
<i>stopexe</i>	Yes	Yes

Table 6: Differences in wake-up sources between Power-Save mode and Power-Down mode

When using the IO pins to wake-up the PFC886, registers *PxDIER* should be properly set to enable the wake-up function for every corresponding pin. The wake-up time for normal wake-up is about 1024 ILRC clocks counting from wake-up event.

Suspend mode	system clock source	Wake-up time ( $t_{WUP}$ ) from IO toggle
<i>Stopexe</i> suspend	Any one	$1024 * T_{ILRC}$ , Where $T_{ILRC}$ is the time period of ILRC
<i>Stopsys</i> suspend	Any one	$1024 * T_{ILRC}$ , Where $T_{ILRC}$ is the clock period of ILRC

Table 7: wake up time

## 8. Interrupt

There are 12 interrupt lines for PFC886:

- (1) four external interrupt lines (PA0 or PA5, PB0 or PB7, the edge type is specified by *INTEGS* register)
- (2) LVD interrupt
- (3) ADC interrupt
- (4) GPC interrupt
- (5) GPC2 interrupt
- (6) Timer16 interrupt
- (7) Timer16N interrupt
- (8) Timer2 interrupt
- (9) Timer3 interrupt
- (10) PWM generator interrupt
- (11) Hall interrupt

Every interrupt request line to CPU has its own corresponding interrupt control bit to enable or disable it. The hardware diagram of interrupt controller is shown as Fig. 16. All the interrupt request flags are set by hardware and cleared by writing *INTRQ* register. When the request flags are set, it can be rising edge, falling edge or both, depending on the setting of register *INTEGS*. All the interrupt request lines are also controlled by *engint* instruction (enable global interrupt) to enable interrupt operation and *disgint* instruction (disable global interrupt) to disable it.

Only FPPA0 can accept the interrupt request, other FPPA unit will not be interfered by interrupt. The stack memory for interrupt is shared with data memory and its address is specified by stack register *SP*. Since the program counter is 16 bits width, the bit 0 of stack register *SP* should be kept 0. Moreover, user can use *pushaf* / *popaf* instructions to store or restore the values of *ACC* and *flag* register to / from stack memory.

Since the stack memory is shared with data memory, user should manipulate the memory using carefully. By adjusting the memory location of stack pointer, the depth of stack pointer for every FPPA unit could be fully specified by user to achieve maximum flexibility of system.

During the interrupt service routine, the interrupt source can be determined by reading the *INTRQ* register.

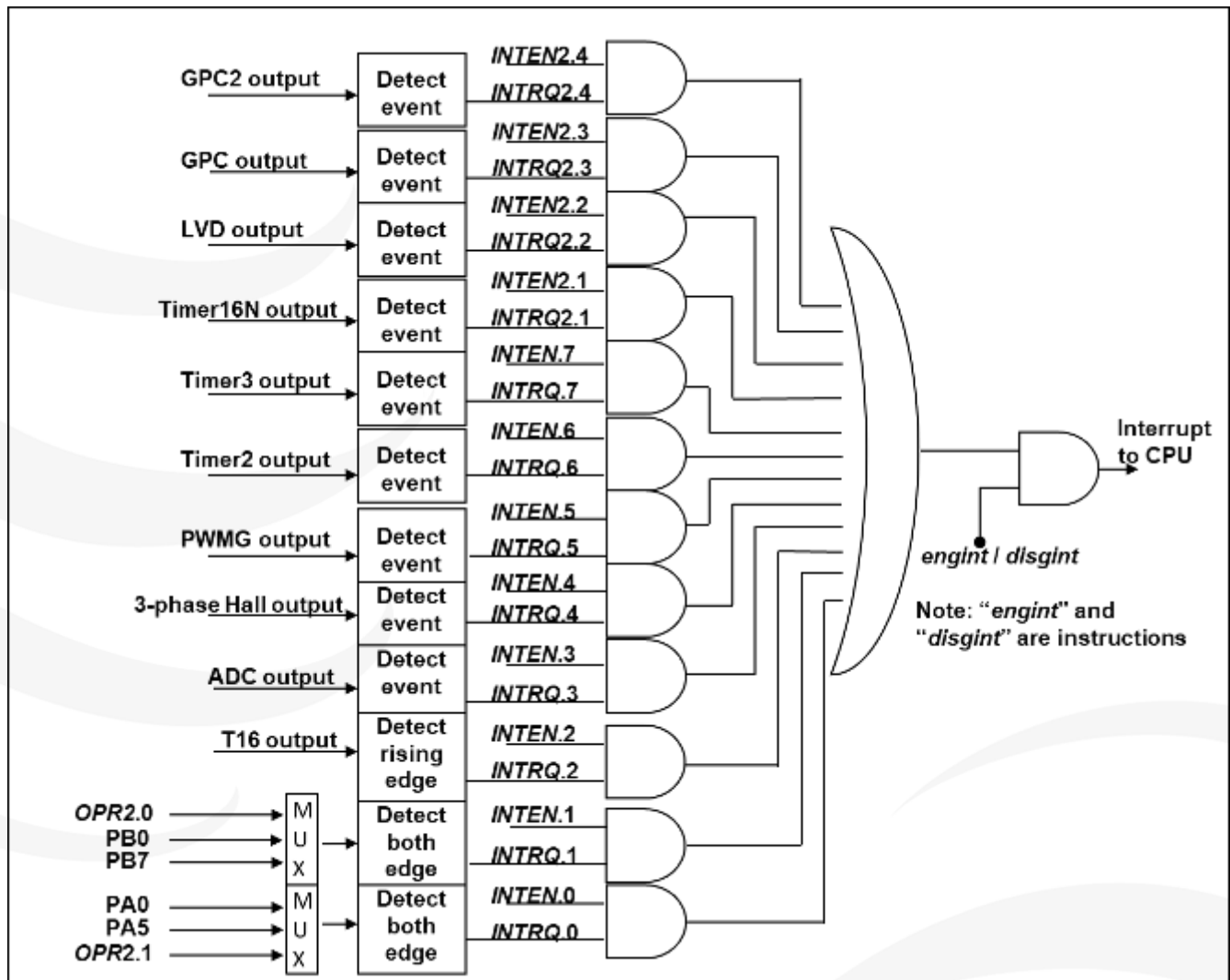


Fig. 16: Hardware diagram of Interrupt controller

Option 2 Register (OPR2), address = 0x3A			
Bit	Reset	R/W	Description
7 – 4	-	-	Reserved. Please keep 0.
3	0	WO	Option for Timer16N clock pre-divider selection. Please see the <i>T16NM</i> register.
2	0	WO	Option for Timer16 clock pre-divider selection. Please see the <i>T16M</i> register.
1	0	WO	Option for external interrupt 1 pin selection.
0	0	WO	Option for external interrupt 0 pin selection.



## 8.1. Interrupt Enable Register (*INTEN*), address = 0x04

Bit	Reset	R/W	Description
7	0	R/W	Enable interrupt from Timer3. 0 / 1: disable / enable
6	0	R/W	Enable interrupt from Timer2. 0 / 1: disable / enable
5	0	R/W	Enable interrupt from PWM generator. 0 / 1: disable / enable
4	0	R/W	Enable interrupt from 3-phase Hall. 0 / 1: disable / enable
3	0	R/W	Enable interrupt from ADC. 0 / 1: disable / enable
2	0	R/W	Enable interrupt from Timer16 overflow. 0 / 1: disable / enable
1	0	R/W	<i>OPR2.0</i> =0 Enable interrupt from PB0 pin. 0 / 1: disable / enable
			<i>OPR2.0</i> =1 Enable interrupt from PB7 pin. 0 / 1: disable / enable
0	0	R/W	<i>OPR2.1</i> =0 Enable interrupt from PA0 pin. 0 / 1: disable / enable
			<i>OPR2.1</i> =1 Enable interrupt from PA5 pin. 0 / 1: disable / enable

Where, *OPR2* is an optional register with address 0x3A

## 8.2. Interrupt Enable 2 Register (*INTEN2*), address = 0x08

Bit	Reset	R/W	Description
7 - 5	-	-	Reserved
4	0	R/W	Enable interrupt from GPC2. 0 / 1: disable / enable
3	0	R/W	Enable interrupt from GPC1. 0 / 1: disable / enable
2	0	R/W	Enable interrupt from LVD detector. 0 / 1: disable / enable
1	0	R/W	Enable interrupt from Timer16N. 0 / 1: disable / enable
0	-	-	Reserved

## 8.3. Interrupt Request Register (*INTRQ*), address = 0x05

Bit	Reset	R/W	Description
7	-	R/W	Interrupt Request from Timer3, this bit is set by hardware and cleared by software 0 / 1: No request / Request
6	-	R/W	Interrupt Request from Timer2, this bit is set by hardware and cleared by software 0 / 1: No request / Request
5	-	R/W	Interrupt Request from PWM generator, this bit is set by hardware and cleared by software 0 / 1: No request / Request
4	-	R/W	Interrupt Request from 3-phase Hall, this bit is set by hardware and cleared by software 0 / 1: No request / Request
3	-	R/W	Interrupt Request from ADC, this bit is set by hardware and cleared by software 0 / 1: No request / Request
2	-	R/W	Interrupt Request from Timer16, this bit is set by hardware and cleared by software 0 / 1: No request / Request
1	-	R/W	<i>OPR2.0</i> =0 Interrupt Request from PB0 pin, this bit is set by hardware and cleared by software. 0 / 1: No request / Request
			<i>OPR2.0</i> =1 Interrupt Request from PB7 pin, this bit is set by hardware and cleared by software. 0 / 1: No request / Request
0	-	R/W	<i>OPR2.1</i> =0 Interrupt Request from PA0 pin, this bit is set by hardware and cleared by software. 0 / 1: No Request / request
			<i>OPR2.1</i> =1 Interrupt Request from PA5 pin, this bit is set by hardware and cleared by software. 0 / 1: No Request / request

Where, *OPR2* is an optional register with address 0x3A

## 8.4. Interrupt Request 2 Register (*INTRQ2*), address = 0x09

Bit	Reset	R/W	Description
7 - 5	-	-	Reserved
4	-	R/W	Interrupt Request from GPC2, this bit is set by hardware and cleared by software. 0 / 1: No request / Request
3	-	R/W	Interrupt Request from GPC1, this bit is set by hardware and cleared by software. 0 / 1: No request / Request
2	-	R/W	Interrupt Request from LVD detector, this bit is set by hardware and cleared by software. 0 / 1: No request / Request
1	-	R/W	Interrupt Request from Timer16N, this bit is set by hardware and cleared by software. 0 / 1: No request / Request
0	-	-	Reserved

## 8.5. Interrupt Edge Select Register (*INTEGS*), address = 0x0C

Bit	Reset	R/W	Description
7 - 5	-	-	Reserved.
4	0	WO	Timer16 edge selection. 0 : rising edge of the selected bit to trigger interrupt 1 : falling edge of the selected bit to trigger interrupt
3 - 2	00	WO	PB0 or PB7 interrupt edge selection. 00 : both rising edge and falling edge of the selected bit to trigger interrupt 01 : rising edge of the selected bit to trigger interrupt 10 : falling edge of the selected bit to trigger interrupt 11 : reserved. Note: If <i>OPR2.0</i> =0, PB0 is selected; If <i>OPR2.0</i> =1, PB7 is selected.
1 - 0	00	WO	PA0 or PA5 interrupt edge selection. 00 : both rising edge and falling edge of the selected bit to trigger interrupt 01 : rising edge of the selected bit to trigger interrupt 10 : falling edge of the selected bit to trigger interrupt 11 : reserved. Note: If <i>OPR2.1</i> =0, PA0 is selected; If <i>OPR2.1</i> =1, PA5 is selected.

**Note:**

*INTEN*/*INTEN2* and *INTRQ*/*INTRQ2* have no initial values. Please set required value before enabling interrupt function. Even if *INTEN*=0, *INTRQ* will be still triggered by the interrupt source, so are the *INTEN2* and *INTRQ2*.

## 8.6. Interrupt Work Flow

Once the interrupt occurs, its operation will be:

- (1) The program counter will be stored automatically to the stack memory specified by register *SP*.
- (2) New *SP* will be updated to *SP*+2.
- (3) Global interrupt will be disabled automatically.
- (4) The next instruction will be fetched from address 0x010.

After finishing the interrupt service routine and issuing the *reti* instruction to return back, its operation will be:

- (1) The program counter will be restored automatically from the stack memory specified by register *SP*.
- (2) New *SP* will be updated to *SP*-2.
- (3) Global interrupt will be enabled automatically.
- (4) The next instruction will be the original one before interrupt.

## 8.7. General Steps to Interrupt

When using the interrupt function, the procedure should be:

- Step1: Set *INTEN* register, enable the interrupt control bit.
- Step2: Clear *INTRQ* register.
- Step3: In the main program, using *engint* to enable CPU interrupt function.
- Step4: Wait for interrupt. When interrupt occurs, enter to Interrupt Service Routine.
- Step5: After the Interrupt Service Routine being executed, return to the main program.

When interrupt service routine starts, use *pushaf* instruction to save *ALU* and *FLAG* register. *Popaf* instruction is to restore *ALU* and *FLAG* register before *reti* as below:

```
void Interrupt (void)    // Once the interrupt occurs, jump to interrupt service routine
{                        // enter disgint status automatically, no more interrupt is accepted
    PUSHAF;
    ...
    POPAF;
}    // reti will be added automatically. After reti being executed, engint status will be restored
```

\* Use *disgint* in the main program can disable all interrupts.

## 8.8. Example for Using Interrupt

User must reserve enough stack memory for interrupt, two bytes stack memory for one level interrupt and four bytes for two levels interrupt.

For interrupt operation, the following sample program shows how to handle the interrupt, noticing that it needs four bytes stack memory to handle interrupt and *pushaf*.

```

void      FPPA0  (void)
{
    ...
    $ INTEN PA0;      // INTEN =1; interrupt request when PA0 level changed
    INTRQ = 0;         // clear INTRQ
    ENGINT              // global interrupt enable
    ...
    DISGINT            // global interrupt disable
    ...
}

void Interrupt (void)    // interrupt service routine
{
    PUSHAF              // store ALU and FLAG register

    // If INTEN.PA0 will be opened and closed dynamically,
    // user can judge whether INTEN.PA0 =1 or not.
    // Example: If (INTEN.PA0 && INTRQ.PA0) {...}

    // If INTEN.PA0 is always enable,
    // user can omit the INTEN.PA0 judgement to speed up interrupt service routine.

    If (INTRQ.PA0)
    {
        // Here for PA0 interrupt service routine
        INTRQ.PA0 = 0;    // Delete corresponding bit (take PA0 for example)
        ...
    }
    ...
    // (X:) INTRQ = 0;      // It is not recommended to use INTRQ = 0 to clear all at the end of the
                           // interrupt service routine.
                           // It may accidentally clear out the interrupts that have just occurred
                           // and are not yet processed.

    POPAF              // restore ALU and FLAG register
}

```

## 9. I/O Port

### 9.1. IO Related Registers

#### 9.1.1. General Data register for IO (*GDIO*), address = 0x07

Bit	Reset	R/W	Description
7 – 0	00	R/W	General data for IO. This port is the general data buffer in IO space and cleared when POR or LVR, and <u>it will KEEP the old values when reset from watch-dog timeout.</u> It can perform the IO operation, like <i>wait0</i> <i>gdio.x</i> , <i>wait1</i> <i>gdio.x</i> and <i>tog</i> <i>gdio.x</i> to replace of operations which instructions are supported in memory space (ex: <i>wait1</i> mem; <i>wait0</i> mem; <i>tog</i> mem).

#### 9.1.2. Port A Digital Input Enable Register (*PADIER*), address = 0x0D

Bit	Reset	R/W	Description
7 - 6	11	WO	Enable PA7 / PA6 digital input and wake-up event 1 / 0 : enable / disable
5	1	WO	Enable PA5 digital input, wake-up event and interrupt request 1 / 0 : enable / disable
4 - 1	1111	WO	Enable PA4 ~ PA1 digital input and wake-up event 1 / 0 : enable / disable
0	1	WO	Enable PA0 digital input, wake-up event and interrupt request 1 / 0: enable / disable

#### 9.1.3. Port B Digital Input Enable Register (*PBDIER*), address = 0x0E

Bit	Reset	R/W	Description
7	0xFF	WO	Enable PB7 digital input, wake up event and interrupt request 1 / 0: enable / disable
6 - 1		WO	Enable PB6~PB1 digital input and wake-up event 1 / 0: enable / disable
0		WO	Enable PB0 digital input, wake-up event and interrupt request 1 / 0: enable / disable

## 9.1.4. Port A Data Registers (*PA*), address = 0x10

Bit	Reset	R/W	Description
7 - 0	0x00	R/W	Data registers for Port A.

## 9.1.5. Port A Control Registers (*PAC*), address = 0x11

Bit	Reset	R/W	Description
7 - 0	0x00	R/W	Port A control registers. This register is used to define input mode or output mode for each corresponding pin of port A. 0 / 1: input / output.

## 9.1.6. Port A Pull-High Registers (*PAPH*), address = 0x12

Bit	Reset	R/W	Description
7 - 0	0x00	R/W	Port A pull-high register. This register is used to enable the internal pull-high device on each corresponding pin of port A and this pull high function is active only for input mode. 0 / 1: disable / enable

## 9.1.7. Port B Data Registers (*PB*), address = 0x14

Bit	Reset	R/W	Description
7 - 0	0x00	R/W	Data registers for Port B.

## 9.1.8. Port B Control Registers (*PBC*), address = 0x15

Bit	Reset	R/W	Description
7 - 0	0x00	R/W	Port B control registers. This register is used to define input mode or output mode for each corresponding pin of port B. 0 / 1: input / output.

## 9.1.9. Port B Pull-High Registers (*PBPH*), address = 0x16

Bit	Reset	R/W	Description
7 - 0	0x00	R/W	Port B pull-high registers. This register is used to enable the internal pull-high device on each corresponding pin of port B. 0 / 1 : disable / enable

## 9.2. IO Structure and Functions

### 9.2.1. IO Pin Structure

All the IO pins of PFC886 have the same structure. The hardware diagram of IO buffer is shown as Fig. 17.

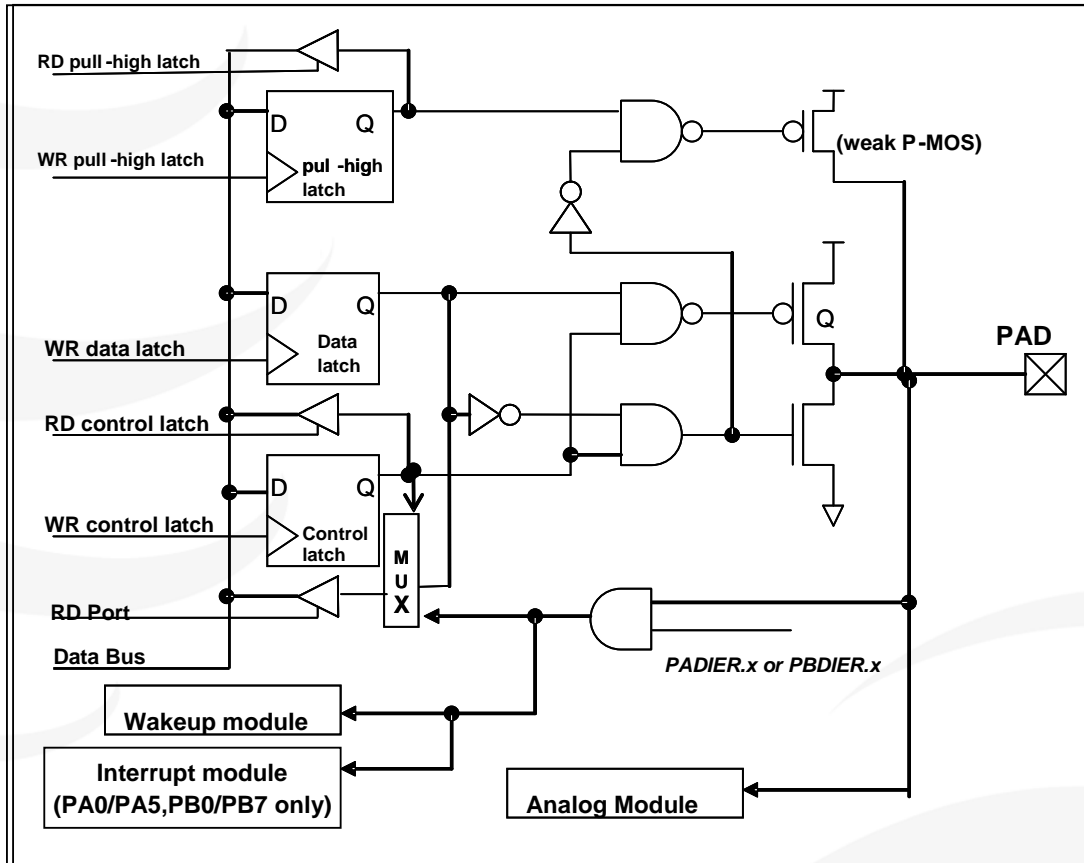


Fig. 17: Hardware diagram of IO buffer

### 9.2.2. IO Pin Functions

#### (1) Input / output function:

PFC886 all the pins can be independently set into digital input, analog input, output low, output high.

Each IO pin can be independently configured for different state by configuring the data registers (*PA/PB*), control registers (*PAC/PBC*), pull-high registers (*PAPH/PBPH*).

The corresponding bits in registers *PxDIER* should be set to low to prevent leakage current for those pins are selected to be analog function. When it is set to output low, the pull-high resistor is turned off automatically.

If user wants to read the pin state, please notice that it should be set to input mode before reading the data port. If user reads the data port when it is set to output mode, the reading data comes from data register, NOT from IO pad.

As an example, Table 8 shows the configuration table of bit 0 of port A.

<i>PA.0</i>	<i>PAC.0</i>	<i>PAPH.0</i>	Description
X	0	0	Input without pull-high resistor
X	0	1	Input with pull-high resistor
0	1	X	Output low without pull-high resistor
1	1	0	Output high without pull-high resistor
1	1	1	Output high with pull-high resistor

Table 8: PA0 Configuration Table

## (2) Wake-up function:

When PFC886 put in Power-Down or Power-Save mode, every IO pin can be used to wake-up system by toggling its state. Therefore, those pins needed to wake-up system must be set to input mode and set the corresponding bits of registers *PxDIER* to high.

## (3) External interrupt function:

When the IO acts as an external interrupt pin, the corresponding bit of *PxDIER* should be set to high. For example, *PADIER.0* should be set to high when PA0 is used as external interrupt pin.

## 9.2.3. IO Pin Usage and Setting

### (1) IO pin as digital input

- ◆ When IO is set as digital input, the level of  $V_{ih}$  and  $V_{il}$  would changes with the voltage and temperature. Please follow the minimum value of  $V_{ih}$  and the maximum value of  $V_{il}$ .
- ◆ The value of internal pull high resistor would also changes with the voltage, temperature and pin voltage. It is not the fixed value.

### (2) If IO pin is set to be digital input and enable wake-up function

- ◆ Configure IO pin as input mode by *PxC* register.
- ◆ Set corresponding bit to "1" in *PxDIER*.
- ◆ For those IO pins of PA that are not used, *PADIER*[1:2] should be set low in order to prevent them from leakage.

### (3) PA5 is set to be PRSTB input pin.

- ◆ Configure PA5 as input.
- ◆ Set *CLKMD.0*=1 to enable PA5 as PRSTB input pin.

### (4) PA5 is set to be input pin and to connect with a push button or a switch by a long wire

- ◆ Needs to put a  $>10\Omega$  resistor in between PA5 and the long wire.
- ◆ Avoid using PA5 as input in such application.



## 10. Timer / PWM Counter

### 10.1. 16-bit Timer (Timer16)

#### 10.1.1. Timer16 Introduction

PFC886 provide a 16-bit hardware timer (Timer16) and its block diagram is shown in Fig. 18.

The clock source of Timer16 is selected by register  $T16M[7:5]$ . Before sending clock to the 16-bit counter (counter16), a pre-scaling logic with divided-by-1/2/4/8/16/32/64 or 128 is selected by  $T16M[4:3]$  and  $OPR2[2]$  for wide range counting.

$T16M[2:0]$  is used to select the interrupt request. The interrupt request from Timer16 will be triggered by the selected bit which comes from bit[15:8] of this 16-bit counter. Rising edge or falling edge can be optional chosen by register  $INTEGS.4$ .

The 16-bit counter performs up-counting operation only, the counter initial values can be stored from data memory by issuing the *stt16* instruction and the counting values can be loaded to data memory by issuing the *ldt16* instruction when  $T16NM[0]$  is 0.

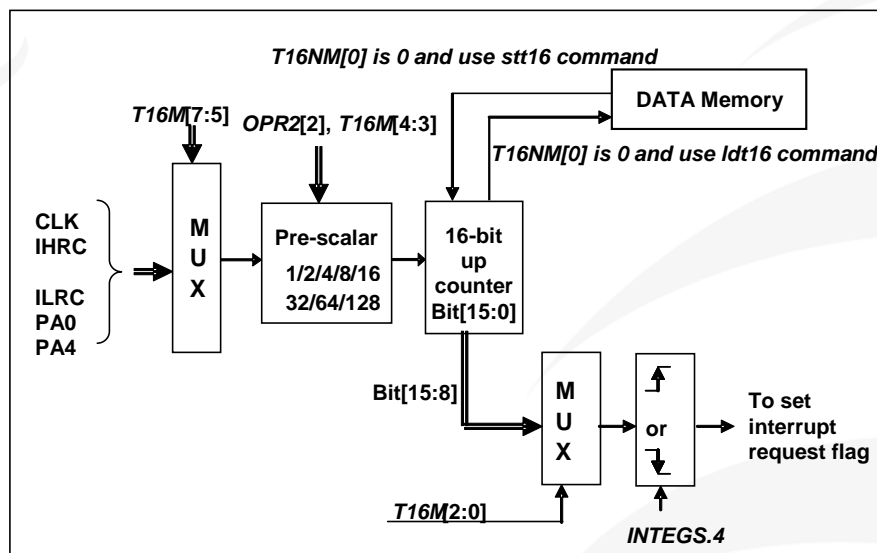


Fig. 18: Hardware diagram of Timer16

There are three parameters to define the Timer16 using; 1<sup>st</sup> parameter is used to define the clock source of Timer16, 2<sup>nd</sup> parameter is used to define the pre-scalar and the 3<sup>rd</sup> one is to define the interrupt source.

<b><i>T16M</i></b>	<b><i>IO_RW</i></b>	<b><i>0x06</i></b>	
<b>\$ 7~5:</b>	<b><i>STOP, SYSCLK, X, PA4_F, IHRC, X, ILRC, PA0_F</i></b>		// 1 <sup>st</sup> par.
<b>\$ 4~3:</b>	<b><i>/1, /4, /16, /64(OPR2[2]=0), /2, /8, /32, /128(OPR2[2]=1)</i></b>		// 2 <sup>nd</sup> par.
<b>\$ 2~0:</b>	<b><i>BIT8, BIT9, BIT10, BIT11, BIT12, BIT13, BIT14, BIT15</i></b>		// 3 <sup>rd</sup> par.

User can choose the proper parameters of *T16M* to meet system requirement, examples as below:

```

$ T16M    SYSCLK, /64, BIT15;
// choose (SYSCLK/64) as clock source, every 2^16 clock to set INTRQ.2=1
// if system clock SYSCLK = IHRC / 4 = 4 MHz
// SYSCLK/64 = 4 MHz/64 = 16 uS, about every 1 S to generate INTRQ.2=1

$ T16M    PA0, /1, BIT8;
// choose PA0 as clock source, every 2^9 to generate INTRQ.2=1
// receiving every 512 times PA0 to generate INTRQ.2=1

$ T16M    STOP;
// stop Timer16 counting

```

If Timer16 is operated at free running, the frequency of interrupt can be described as below:

$$F_{INTRQ\_T16M} = F_{\text{clock source}} \div P \div 2^{n+1}$$

Where, F is the frequency of selected clock source to Timer16;

P is the selection of *OPR2[2]* and *T16M[4:3]*; (1, 4, 16, 32, 64, 128)

N is the n<sup>th</sup> bit selected to request interrupt service, for example: n=10 if bit 10 is selected.

## 10.1.2. Timer16 Time Out

When select \$ *INTEGS BIT\_R* (default value) and *T16M* counter BIT8 to generate interrupt, if *T16M* counts from 0, the first interrupt will occur when the counter reaches to 0x100 (BIT8 from 0 to 1) and the second interrupt will occur when the counter reaches 0x300 (BIT8 from 0 to 1). Therefore, selecting BIT8 as 1 to generate interrupt means that the interrupt occurs every 512 counts. Please notice that if *T16M* counter is restarted, the next interrupt will occur once Bit8 turns from 0 to 1.

If select \$ *INTEGS BIT\_F* (BIT triggers from 1 to 0) and *T16M* counter BIT8 to generate interrupt, the *T16M* counter changes to an interrupt every 0x200/0x400/0x600/. Please pay attention to two differences with setting *INTEGS* methods.

## 10.1.3. Timer16 Mode Register (*T16M*), address = 0x06

Bit	Reset	R/W	Description
7 - 5	000	R/W	Timer Clock source selection. 000: Timer16 is disabled. 001: CLK (system clock) 010: reserved 011: PA4 100: IHRC 101: reserved 110: ILRC 111: PA0
4 - 3	00	R/W	<div> <div><i>OPR2.2</i>=0</div> <div>Timer16 clock pre-divider. 00: /1 01: /4 10: /16 11: /64</div> </div> <div> <div><i>OPR2.2</i>=1</div> <div>Timer16 clock pre-divider. 00: /2 01: /8 10: /32 11: /128</div> </div>
2 - 0	000	R/W	Interrupt source selection. Interrupt event happens when selected bit goes high. 0 : bit 8 of Timer16 1 : bit 9 of Timer16 2 : bit 10 of Timer16 3 : bit 11 of Timer16 4 : bit 12 of Timer16 5 : bit 13 of Timer16 6 : bit 14 of Timer16 7 : bit 15 of Timer16

Where, *OPR2* is an optional register with address 0x3A.

## Option 2 Register (*OPR2*), address = 0x3A

Bit	Reset	R/W	Description
7 - 4	-	-	Reserved. Please keep 0.
3	0	WO	Option for Timer16N clock pre-divider selection. Please see the T16NM register.
2	0	WO	<b>Option for Timer16 clock pre-divider selection. Please see the T16M register.</b>
1	0	WO	Option for external interrupt 1 pin selection.
0	0	WO	Option for external interrupt 0 pin selection.

## 10.2. 16-bit Timer (Timer16N)

### 10.2.1. Timer16N Introduction

PFC886 provide an another 16-bit hardware timer (Timer16N) and its block diagram is shown in Fig. 19.

The clock source of Timer16N is selected by register *T16NM*[7:5]. Before sending clock to the 16-bit counter (counter16), a pre-scaling logic with divided-by-1/2/4/8/16/32/64 or 128 is selected by *T16NM*[4:3] and *OPR2*[3] for wide range counting.

The 16-bit counter will be clear to zero and generate interrupt request automatically whenever its values reach to that of bound register *T16NBH* and *T16NBL*. The bound register is used to define the period of Timer16N and need to write *T16NBH* first.

The 16-bit counter performs up-counting operation only, the counter initial values can be stored from data memory by issuing the *stt16* instruction and the counting values can be loaded to data memory by issuing the *ldt16* instruction when *T16NM*[0] is 1.

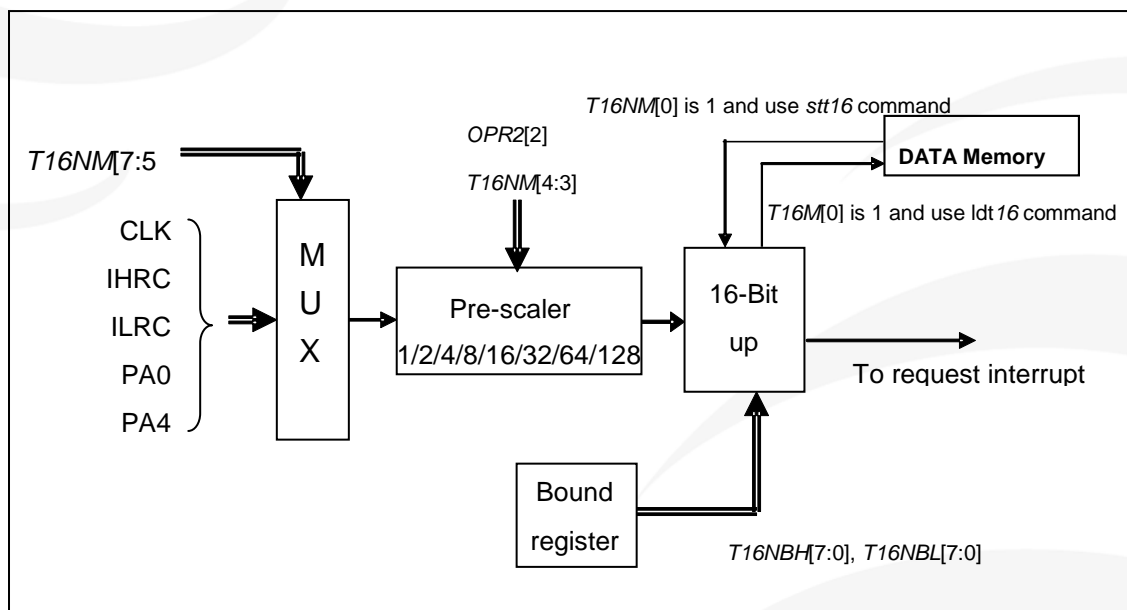


Fig. 19 Timer16N hardware diagram

Option 2 Register ( <i>OPR2</i> ), address = 0x3A			
Bit	Reset	R/W	Description
7 – 4	-	-	Reserved. Please keep 0.
3	0	WO	Option for Timer16N clock pre-divider selection. Please see the T16NM register.
2	0	WO	Option for Timer16 clock pre-divider selection. Please see the T16M register.
1	0	WO	Option for external interrupt 1 pin selection.
0	0	WO	Option for external interrupt 0 pin selection.

## 10.2.2. Timer16N mode Register (*T16NM*), address = 0x19

Bit	Reset	R/W	Description
7 - 5	000	R/W	Timer Clock source selection. 000: Timer16N is disabled. 001: CLK (system clock) 010: reserved 011: PA4 100: IHRC 101: reserved 110: ILRC 111: PA0
4 - 3	00	R/W	<i>OPR2.3=0</i> Timer16N clock pre-divider. 00: /1 01: /4 10: /16 11: /64 <i>OPR2.3=1</i> Timer16N clock pre-divider. 00: /2 01: /8 10: /32 11: /128
2 - 1	00	RO	Read as 0
0	0	R/W	Idt16/stt16 Instruction support 1: T16N 0: T16

Where, *OPR2* is an optional register with address 0x3A.

## 10.2.3. Timer16N Bound High Byte Register (*T16NBH*), address = 0x73

Bit	Reset	R/W	Description
7 - 0	0x00	WO	Timer16N bound high byte register.

## 10.2.4. Timer16N Bound Low Byte Register (*T16NBL*), address = 0x74

Bit	Reset	R/W	Description
7 - 0	0x00	WO	Timer16N bound low byte register.

## 10.3. 8-bit Timer (Timer2, Timer3)

Two 8-bit hardware timers (Timer2/TM2, Timer3/TM3) are implemented in the PFC886. Timer2 is used as the example to describe its function due to these two 8-bit timers are the same. Fig. 20 shows the Timer2 hardware diagram.

Bit [7:4] of register *TM2C* are used to select the clock source of Timer2. The clock frequency divide module is controlled by bit [6:5] of *TM2S* register. And *TM2S*[4:0] provide one scaling module with divided-by-1~31. In conjunction of pre-scaling function and scaling function, the frequency of Timer2 clock (*TM2\_CLK*) can be wide range and flexible. The counter values can be set or read back by *TM2CT* register.

The Timer2 counter performs 8-bit up-counting operation only. The 8-bit counter will be clear to zero and generate interrupt request automatically whenever its values reach to that of bound register *TM2B*, the bound register is used to define the period of Timer2.

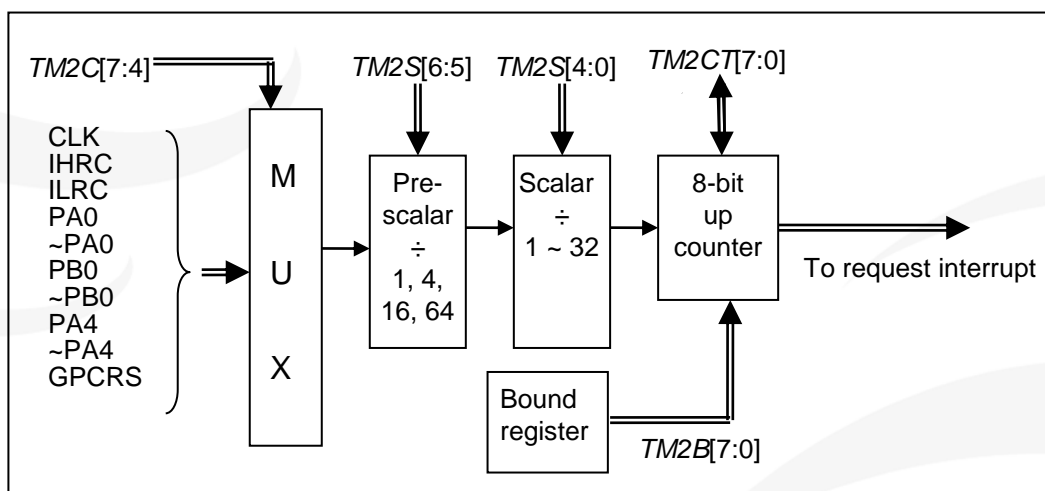


Fig. 20: Timer2 hardware diagram

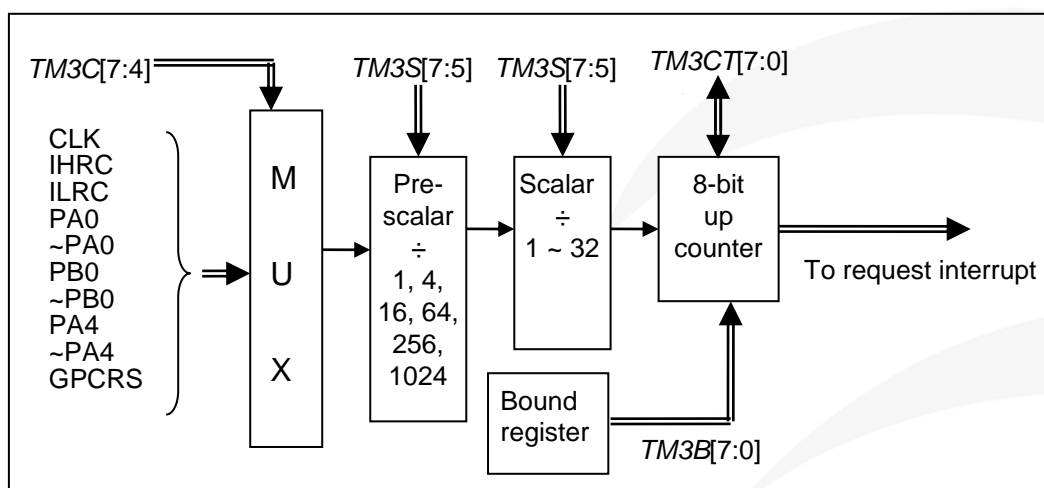


Fig. 21: Timer3 hardware diagram

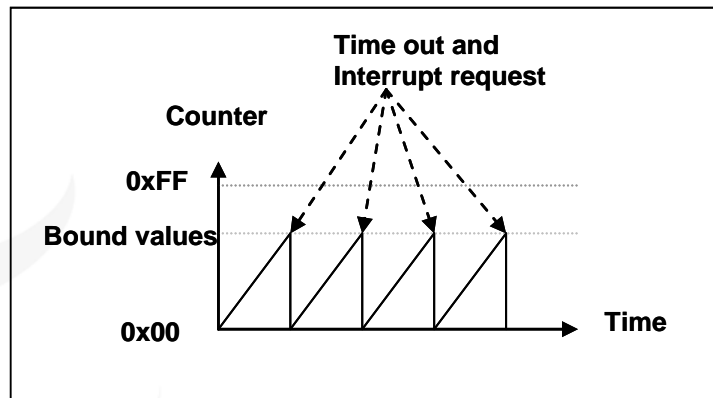


Fig. 22: Timing diagram of Timer2/Timer3

### 10.3.1. Timer2 Control Register (*TM2C*), address = 0x24

Bit	Reset	R/W	Description
7 - 4	0000	R/W	Timer2 clock selection. 0000 : disable 0001 : system clock 0010 : IHRC 0100 : ILRC 0101 : GPCRS 011x : reserved 1000 : PA0 (rising edge) 1001 : ~PA0 (falling edge) 1010 : PB0 (rising edge) 1011 : ~PB0 (falling edge) 1100 : PA4 (rising edge) 1101 : ~PA4 (falling edge)
3 - 0	-	-	Reserved, please keep 0.

### 10.3.2. Timer2 Counter Register (*TM2CT*), address = 0x25

Bit	Reset	R/W	Description
7 - 0	0x00	R/W	Bit [7:0] of Timer2 counter register

## 10.3.3. Timer2 Scalar Register (*TM2S*), address = 0x26

Bit	Reset	R/W	Description
7	-	-	Reserved.
6 - 5	00	WO	Timer2 clock pre-scalar. 00 : ÷ 1 01 : ÷ 4 10 : ÷ 16 11 : ÷ 64
4 - 0	00000	WO	Timer2 clock scalar.

## 10.3.4. Timer2 Bound Register (*TM2B*), address = 0x27

Bit	Reset	R/W	Description
7 - 0	0x00	WO	Timer2 bound register.

## 10.3.5. Timer3 Control Register (*TM3C*), address = 0x28

Bit	Reset	R/W	Description
7 - 4	0000	R/W	Timer3 clock selection. 0000 : disable 0001 : system clock 0010 : IHRC 0100 : ILRC 0101 : GPCRS 011x : reserved 1000 : PA0 (rising edge) 1001 : ~PA0 (falling edge) 1010 : PB0 (rising edge) 1011 : ~PB0 (falling edge) 1100 : PA4 (rising edge) 1101 : ~PA4 (falling edge)
3 - 0	-	-	Reserved, please keep 0.

## 10.3.6. Timer3 Counter Register (*TM3CT*), address = 0x29

Bit	Reset	R/W	Description
7 - 0	0x00	R/W	Bit [7:0] of Timer3 counter register.



## 10.3.7. Timer3 Scalar Register (*TM3S*), address = 0x2A

Bit	Reset	R/W	Description
7 - 5	000	WO	Timer3 clock pre-scalar. 000 : ÷ 1 001 : ÷ 4 010 : ÷ 16 011 : ÷ 64 100 : ÷ 256 101 : ÷ 1024
4 - 0	00000	WO	Timer3 clock scalar.

## 10.3.8. Timer3 Bound Register (*TM3B*), address = 0x2B

Bit	Reset	R/W	Description
7 - 0	0x00	WO	Timer3 bound register.

## 10.4. 12-bit PWM Generation

One 12-bit hardware PWM generators with programmable period and duty is built inside the PFC886. The PWM can be configured as normal mode or complementary mode. The clock source can be IHRC or IHRCX2 divided by scalar 1, 4, 16 or 64.

### 10.4.1. PWM Waveform

A PWM output waveform (Fig. 23) has a time-base ( $T_{\text{Period}} = \text{Time of Period}$ ) and a time with output high level (Duty Cycle). The frequency of the PWM output is the inverse of the period ( $f_{\text{PWM}} = 1/T_{\text{Period}}$ ), the resolution of the PWM is the clock count numbers for one period ( $N \text{ bits resolution}, 2^N \times T_{\text{clock}} = T_{\text{Period}}$ ).

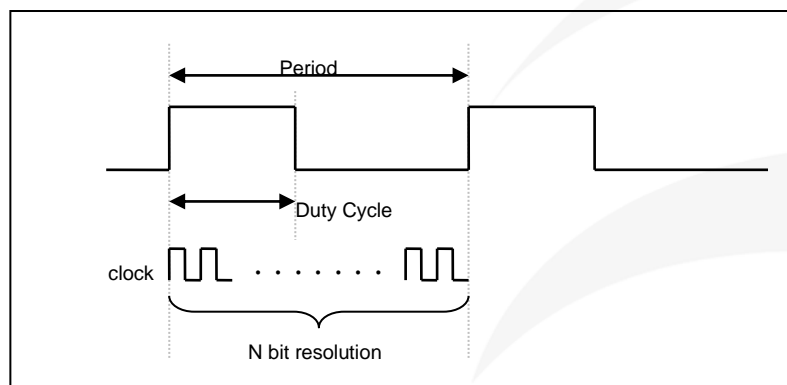


Fig. 23: PWM Output Waveform

### 10.4.2. Hardware PWM without dead zone and Timing Diagram

In normal mode, the output pins can be PA2, PA3, PA4 or PA7 via *PWMGC* register selection. The period of PWM waveform is defined in the PWM upper bond high and low registers, the duty cycle of PWM waveform is defined in the PWM duty high and low registers. Fig. 24 shows its hardware diagram.

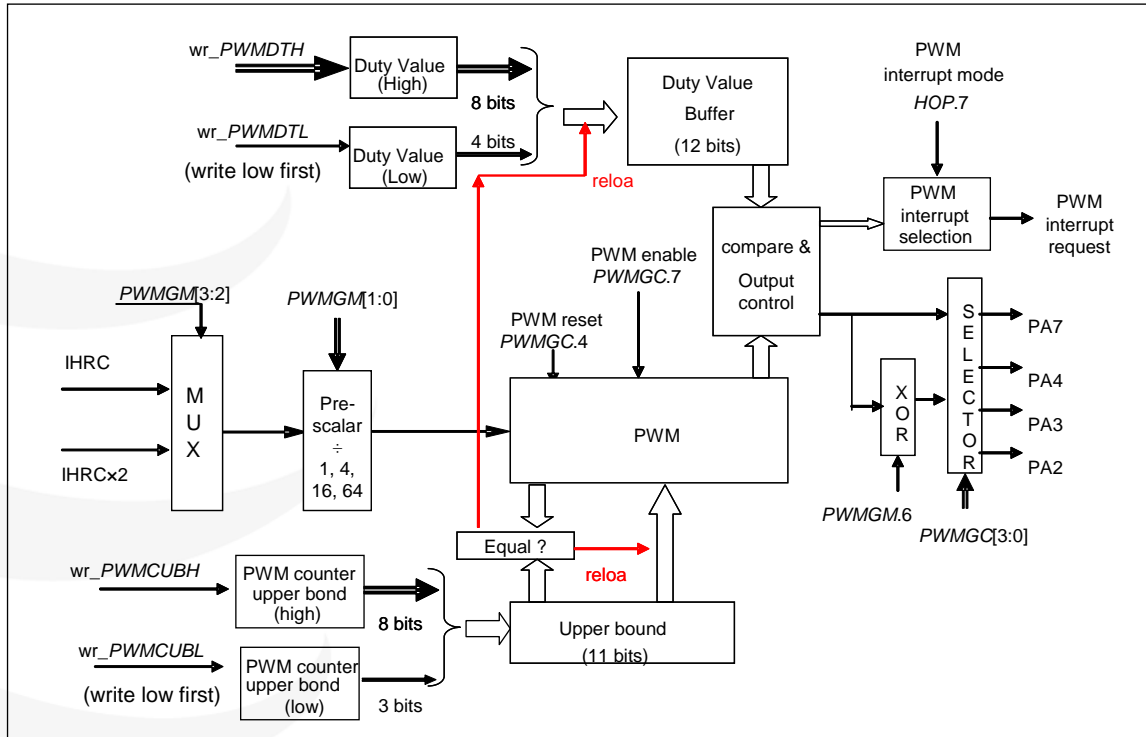


Fig. 24: Normal mode of 12-bit PWM Generator

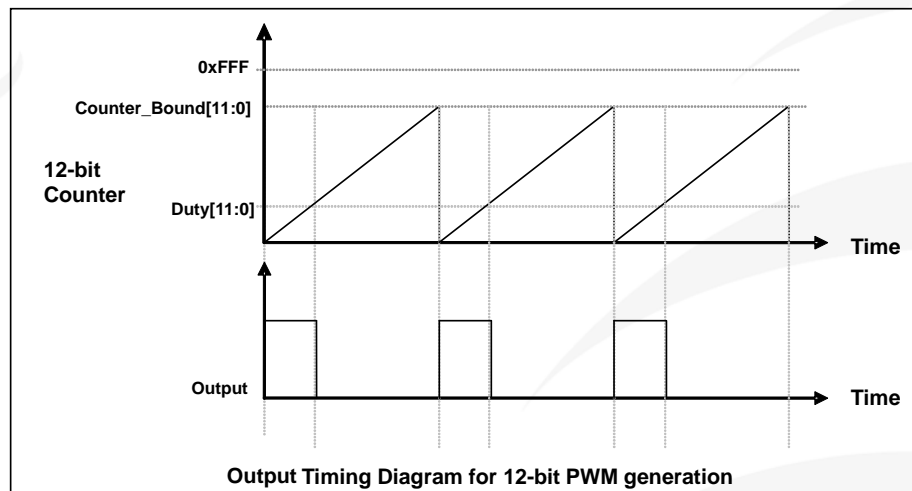


Fig. 25: Output Timing Diagram of 12-bit PWM Generator

### 10.4.3. Equations for 12-bit PWM Generator

If  $F_{IHRC}$  is the frequency of IHRC oscillator and IHRC is the chosen clock source for 12-bit PWM generator, the PWM frequency and duty cycle in time will be:

$$\text{Frequency of PWM Output} = F_{IHRC} \div [P \times CB]$$

$$\text{Duty Cycle of PWM Output (in time)} = (1/F_{IHRC}) * [DB \div CB]$$

Where,  $PWMGM[1:0] = P$  ; pre-scaler

12-bit Duty\_Bound[11:0] = {pwmPTH[7:0],pwmDTL[7:4]} = **DB**; duty bound

11-bit Counter\_Bound[11:1] X2 = {pwmCUBH[7:0], pwmCUBL[7:5]} X2 = **CB**; counter bound

## 10.4.4. Hardware PWM with dead zone

The PWM generator with dead zone control is built inside the PFC886 for complementary mode; Fig. 26 shows its hardware diagram. The complementary group output pins can be PA7/PA4, PA0/PA3, PA1/PA2. The period of PWM waveform is defined in the PWM upper bond high and low registers, and the duty cycle of the PWM waveform is defined in the PWM0/PWM1/PWM2 duty high and low registers, respectively.

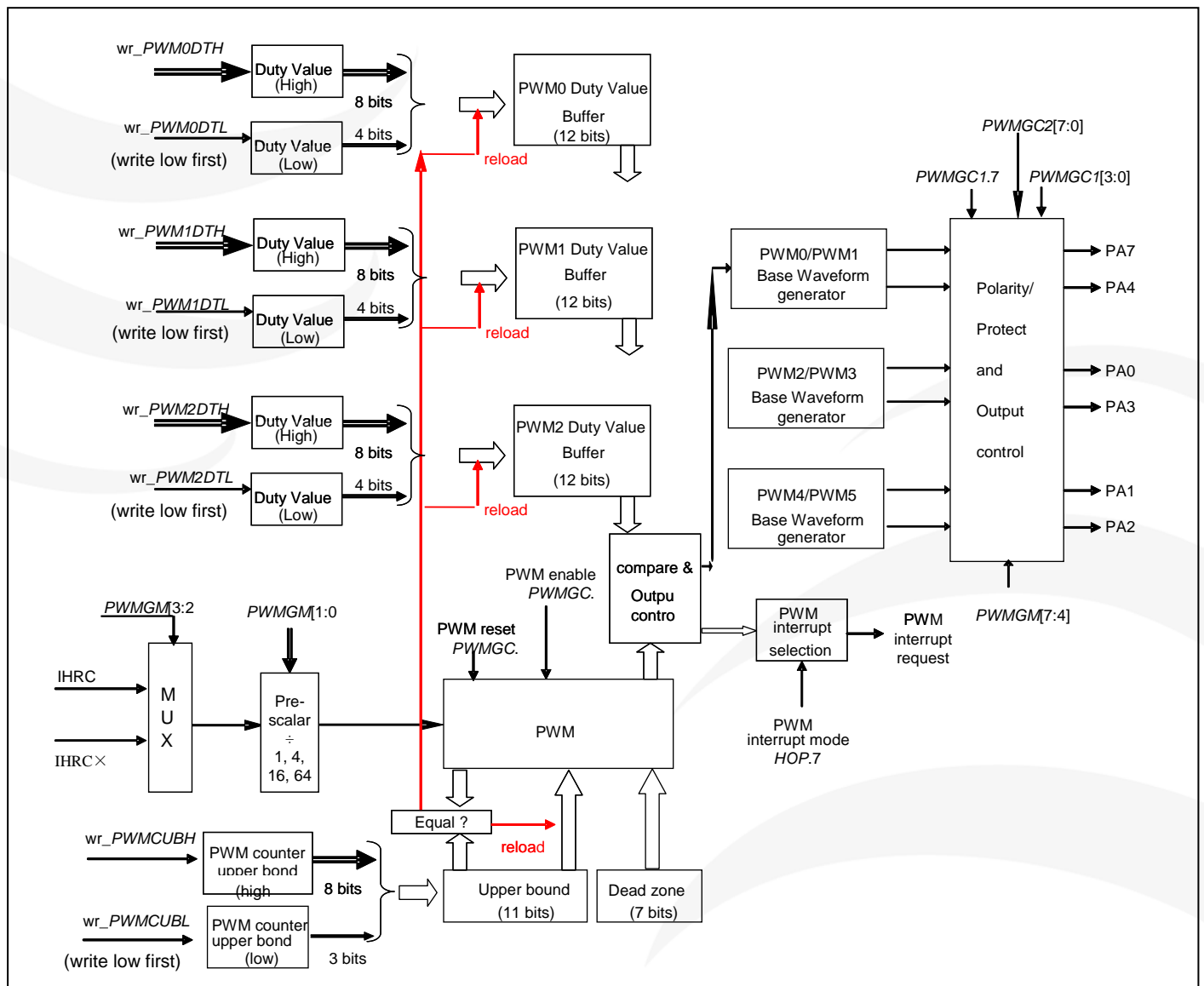


Fig. 26: Dead zone mode of 12-bit PWM Generator

## 10.4.5. 12-bit PWM Related Registers

### 10.4.5.1. PWM Generator control Register (*PWMGC*), address = 0x30

Bit	Reset	R/W	Description
7	0	R/W	Enable PWM generator. 0 / 1 : disable / enable.
6	-	RO	Output of PWM0 generator.
5	0	R/W	PWM alignment mode. 0 : edge alignment mode. 1: center alignment mode.
4	0	R/W	PWM counter reset. Writing "1" to clear PWM counter and this bit will be self clear to 0 after counter reset.
3	0	R/W	PWM0 output to PA7. 1: Enable 0: Disable
2	0	R/W	PWM0 output to PA4. 1: Enable 0: Disable
1	0	R/W	PWM0 output to PA3. 1: Enable 0: Disable
0	0	R/W	PWM0 output to PA2. 1: Enable 0: Disable

### 10.4.5.2. PWM Generator Scalar Register (*PWMGM*), address = 0x31

Bit	Reset	R/W	Description
7	0	WO	Enable to inverse the polarity of high-side PWM output. 0 / 1: disable / enable.
6	0	WO	Enable to inverse the polarity of low-side PWM output. 0 / 1: disable / enable.
5	0	R/W	Side Protection Enable. 0 / 1: disable / enable.
4	0	R/W	Side Protection mode. 0 : enable protection when High-side is in high level 1 : enable protection when High-side is in low level
3 - 2	00	R/W	PWM generator clock source. 0x : Disable 10 : IHRC 11 : IHRC*2
1 - 0	00	R/W	PWM generator clock pre-scalar. 00 : /1 01 : /4 10 : /16 11 : /64

## 10.4.5.3. PWM Generator control 1 Register (*PWMGC1*), address = 0x37

Bit	Reset	R/W	Description
7	0	R/W	Complementary PWM0~PWM5 generator output enable. 0 / 1: disable / enable.
6 - 5	-	WO	10: PWM generator support for Single-phase BLDC. 01: Reserved 00/11 : PWM generator support for Three-phase BLDC
4	0	RO	Read as 0.
3 - 2	00	R/W	PWM5 output type. 00 : force low 01 : PWM+ 10 : PWM- 11 : force high
1 - 0	00	R/W	PWM4 output type. 00 : force low 01 : PWM+ 10 : PWM- 11 : force high

## 10.4.5.4. PWM Generator control 2 Register (*PWMGC2*), address = 0x38

Bit	Reset	R/W	Description
7 - 6	00	R/W	PWM3 output type. 00 : force low 01 : PWM+ 10 : PWM- 11 : force high
5 - 4	00	R/W	PWM2 output type. 00 : force low 01 : PWM+ 10 : PWM- 11 : force high
3 - 2	00	R/W	PWM1 output type. 00 : force low 01 : PWM+ 10 : PWM- 11 : force high
1 - 0	00	R/W	PWM0 output type. 00 : force low 01 : PWM+ 10 : PWM- 11 : force high

## 10.4.5.5. PWM Counter Upper Bound High Register (*PWMCUBH*), address = 0x50

Bit	Reset	R/W	Description
7 - 0	8'h00	WO	Bit[11:4] of PWM counter upper bound.

## 10.4.5.6. PWM Counter Upper Bound Low Register (*PWMCUBL*), address = 0x51

Bit	Reset	R/W	Description
7 - 5	3'h0	WO	Bit[3:1] of PWM counter upper bound.
4 - 0	-	-	Reserved.

## 10.4.5.7. PWM0 Duty Value High Register (*PWM0DTH*), address = 0x52

Bit	Reset	R/W	Description
7 - 0	8'h00	WO	Duty values bit[11:4] of PWM0 generator.

## 10.4.5.8. PWM0 Duty Value Low Register (*PWM0DTL*), address = 0x53

Bit	Reset	R/W	Description
7 - 4	4'h0	WO	Duty values bit[3:0] of PWM0 generator.
3 - 0	-	-	Reserved.

## 10.4.5.9. PWM1 Duty Value High Register (*PWM1DTH*), address = 0x54

Bit	Reset	R/W	Description
7 - 0	8'h00	WO	Duty values bit[11:4] of PWM1 generator.

## 10.4.5.10. PWM1 Duty Value Low Register (*PWM1DTL*), address = 0x55

Bit	Reset	R/W	Description
7 - 4	4'h0	WO	Duty values bit[3:0] of PWM1 generator.
3 - 0	-	-	Reserved.

## 10.4.5.11. PWM2 Duty Value High Register (*PWM2DTH*), address = 0x56

Bit	Reset	R/W	Description
7 - 0	8'h00	WO	Duty values bit[11:4] of PWM2 generator.

## 10.4.5.12. PWM2 Duty Value Low Register (*PWM2DTL*), address = 0x57

Bit	Reset	R/W	Description
7 - 4	4'h0	WO	Duty values bit[3:0] of PWM2 generator.
3 - 0	-	-	Reserved.

## 10.4.5.13. PWM Dead-zone Register (*PWMDZ*), address = 0x58

Bit	Reset	R/W	Description
7 - 1	-	WO	Dead-zone values bit[7:1] of PWM generator.
0	-	-	Reserved.

## 11. Special Functions

### 11.1. General Purpose Comparator

#### 11.1.1. General Purpose Comparator Hardware Diagram

Two general purpose comparators are built inside the PFC886. It can compare signals between two pins or with either internal reference voltage  $V_{\text{internal R}}$  or internal Band-gap reference voltage. The two signals to be compared, one will be the plus input of comparator and the other one is the minus input of comparator. For general purpose comparator 1, the plus input pin is selected by register *GPC1C.0*, and the minus input pin is selected by *GPC1C[3:1]*. For comparator 2, it will be *GPC1C.0* and *GPC1C[3:1]*.

The comparator result can be:

- (1) read back by *GPC1C.6* or *GPC2C.6*;
- (2) inversed the polarity by *GPC1C.4* or *GPC2C.4*;
- (3) sampled by rising edge of Time2 clock (TM2\_CLK) or Time3 clock (TM3\_CLK);
- (4) enabled to output to PB0 / PA2 directly by *GPC1S.7* / *GPC2S.7*;
- (5) used to request interrupt service.

The comparator is disabled after power-on reset and can be enabled by setting *GPC1C.7=1* or *GPC2C.7=1*. The comparator module can be put into power-down mode only when issuing *stopsys* command which will put PFC886 into power-down mode.

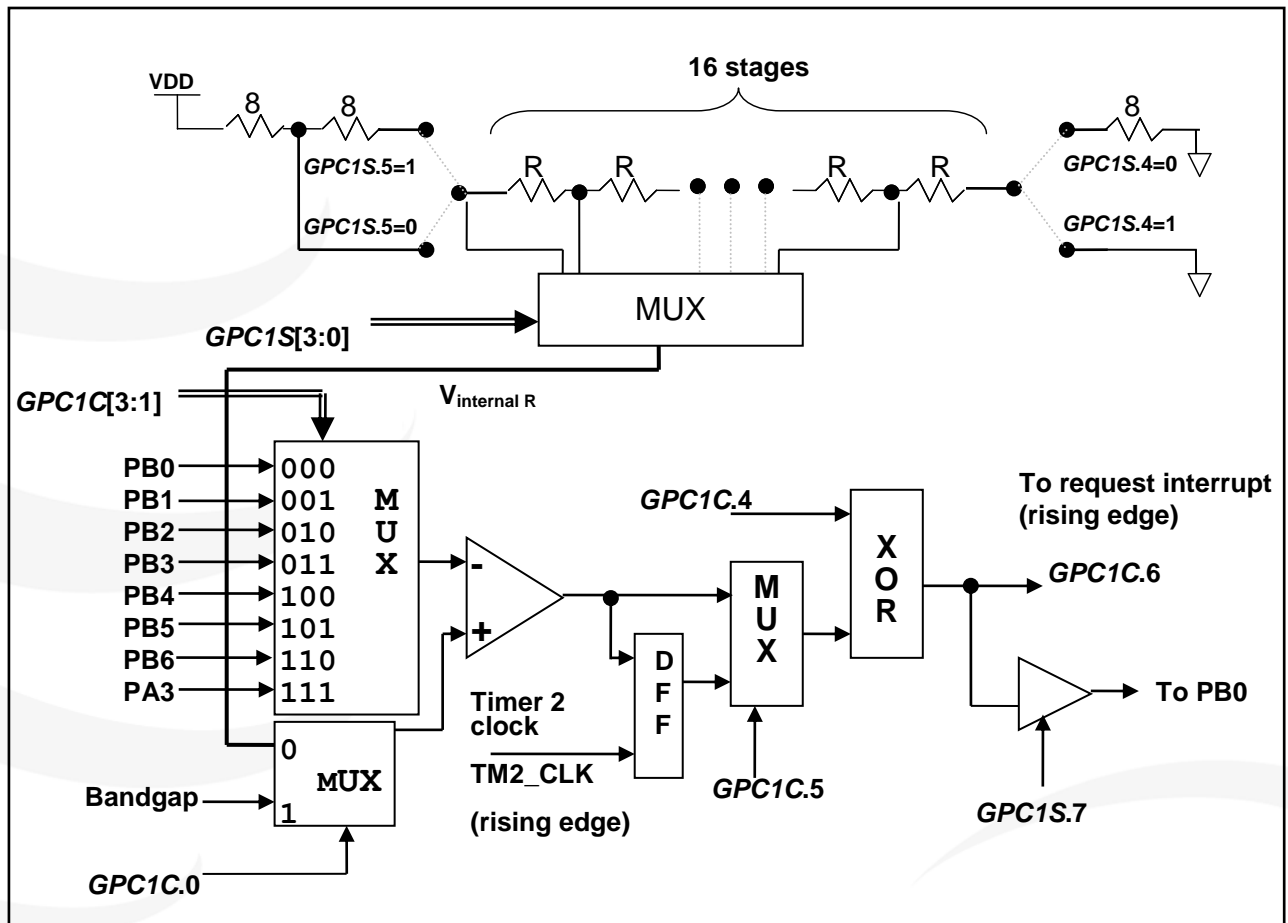


Fig. 27: Hardware diagram of comparator 1



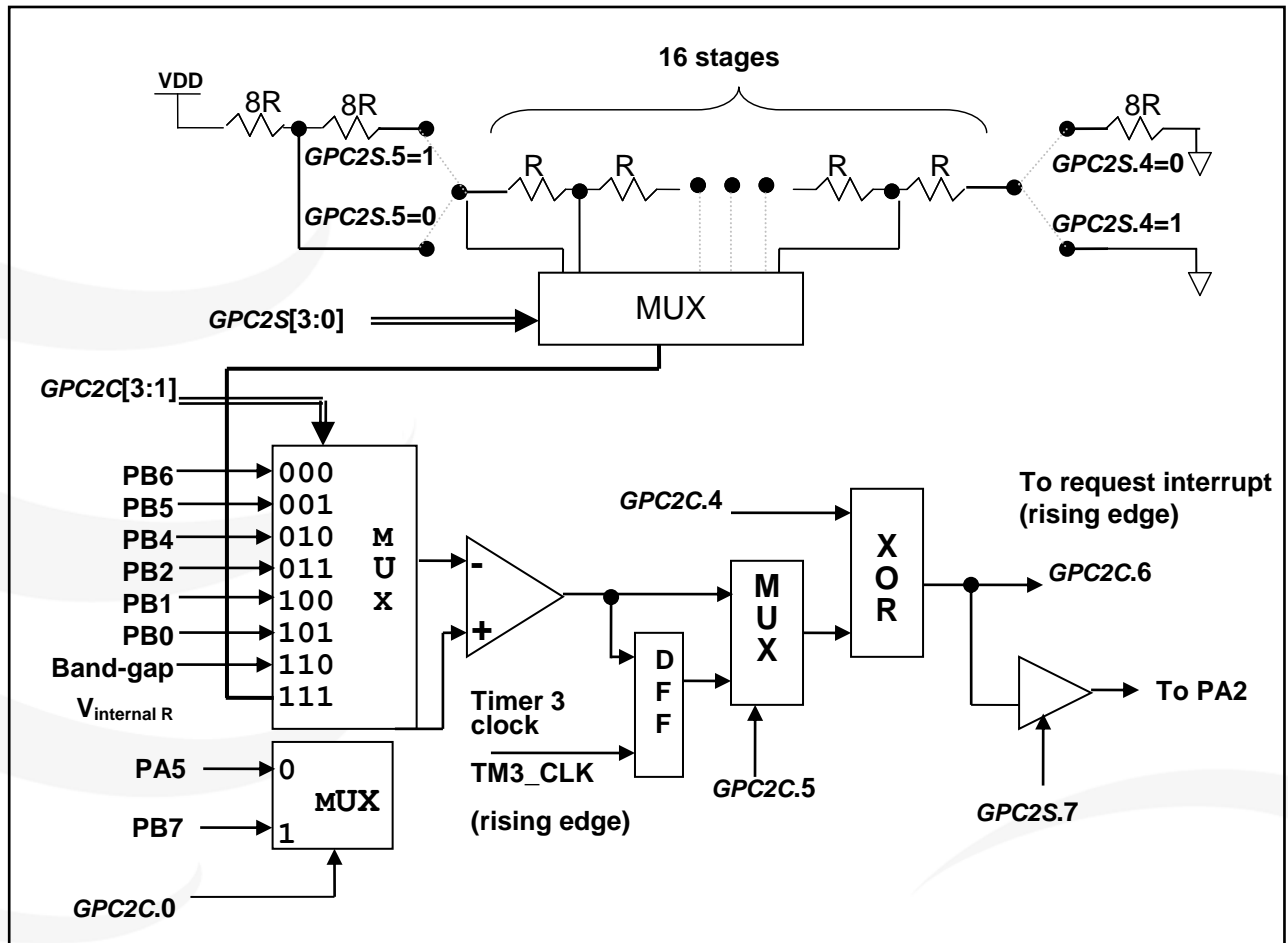


Fig. 28: Hardware diagram of comparator 2

## 11.1.2. General Purpose Comparator 1 Control Register (GPC1C), address = 0x34

Bit	Reset	R/W	Description
7	0	R/W	Enable comparator. 0 / 1 : disable / enable When this bit is set to enable, please also set the corresponding input pins to be digital disable to prevent IO leakage.
6	-	RO	Comparator result. 0: plus input < minus input 1: plus input > minus input
5	0	R/W	Selection the sampled source of comparator result. 0: result output NOT sampled by TM2_CLK 1: result output sampled by TM2_CLK
4	0	R/W	Inverse the polarity of result output of comparator. 0: polarity is NOT inversed. 1: polarity is inversed.
3 - 1	000	R/W	Selection the minus input (-) of general purpose comparator. 000 : PB0 001 : PB1 010 : PB2 011 : PB3 100 : PB4 101 : PB5 110 : PB6 111 : PA3
0	0	R/W	Selection the plus input (+) of comparator. 0 : V <sub>internal R</sub> 1 : Band-gap output

## 11.1.3. General Purpose Comparator 1 Selection Register (GPC1S), address = 0x35

Bit	Reset	R/W	Description
7	0	WO	Comparator output enable (to PB0). 0 / 1 : disable / enable
6	0	WO	General purpose comparator enables to wake up system. 0 / 1 : disable / enable The system will be wake up from power down mode if the result goes high.
5	0	WO	Selection of high range of general purpose comparator.
4	0	WO	Selection of low range of general purpose comparator.
3 - 0	0000	WO	Selection the voltage level of general purpose comparator. 0000 (lowest) ~ 1111 (highest)

## 11.1.4. General Purpose Comparator 2 Control Register (GPC2C), address = 0x1E

Bit	Reset	R/W	Description
7	0	R/W	Enable general purpose comparator 2. 0 / 1 : disable / enable When this bit is set to enable, please also set the corresponding analog input pins to be digital disable to prevent IO leakage.
6	-	RO	Comparator result of general purpose comparator 2. 0: plus input < minus input 1: plus input > minus input
5	0	R/W	Selection the sampled source of comparator 2 result 0: result output NOT sampled by TM3_CLK 1: result output sampled by TM3_CLK
4	0	R/W	Inverse the polarity of the comparator 2 result output 0: polarity is NOT inversed 1: polarity is inversed
3 - 1	000	R/W	Selection the Minus source of general purpose comparator 2. 000 : PB6 001 : PB5 010 : PB4 011 : PB2 100 : PB1 101 : PB0 110 : Band-gap output 111 : internal R
0	0	R/W	Selection the Plus source of general purpose comparator 2. 0 : PA5 1 : PB7

## 11.1.5. General Purpose Comparator 2 Selection Register (GPC2S), address = 0x1F

Bit	Reset	R/W	Description
7	0	WO	General purpose comparator 2 output enable (to PA2) 0 / 1 : disable / enable
6	0	WO	General purpose comparator 2 enables to wake up system. 0 / 1 : disable / enable The system will be wake up from power down mode if the result goes high.
5	0	WO	Selection of high range of general purpose comparator 2.
4	0	WO	Selection of low range of general purpose comparator 2.
3 - 0	0000	WO	Selection the voltage level of general purpose comparator 2. 0000 (lowest) ~ 1111 (highest)

## 11.1.6. Analog Inputs

A simplified circuit for the analog inputs is shown in the Fig. 29. All the analog input pins for general purpose comparator are shared function with a digital input which has reverse biased ESD protection diodes to VDD and GND, therefore, the analog input signal must be between VDD and GND.

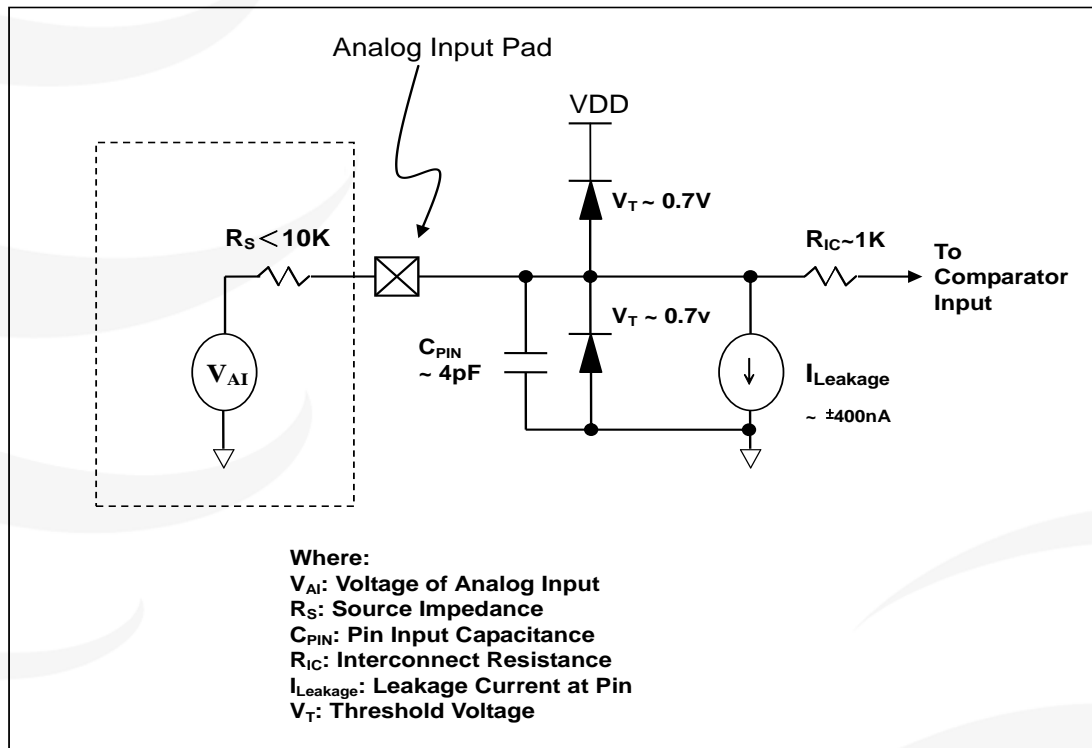


Fig. 29: Analog Input Model of General Purpose Comparator

## 11.1.7. Internal Reference Voltage ( $V_{internal R}$ )

The internal reference voltage  $V_{internal R}$  is built by series resistance to provide different level of reference voltage, bit 4 and bit 5 of *GPC1S/GPC2S* (as *GPCS* thereafter) register are used to select the maximum and minimum values of  $V_{internal R}$  and *GPCS*[3:0] are used to select one of the voltage level which is divided-by-16 from the defined maximum level to minimum level. By setting the *GPCS* register, the internal reference voltage  $V_{internal R}$  can be ranged from  $(1/32) \cdot V_{DD}$  to  $(3/4) \cdot V_{DD}$ .

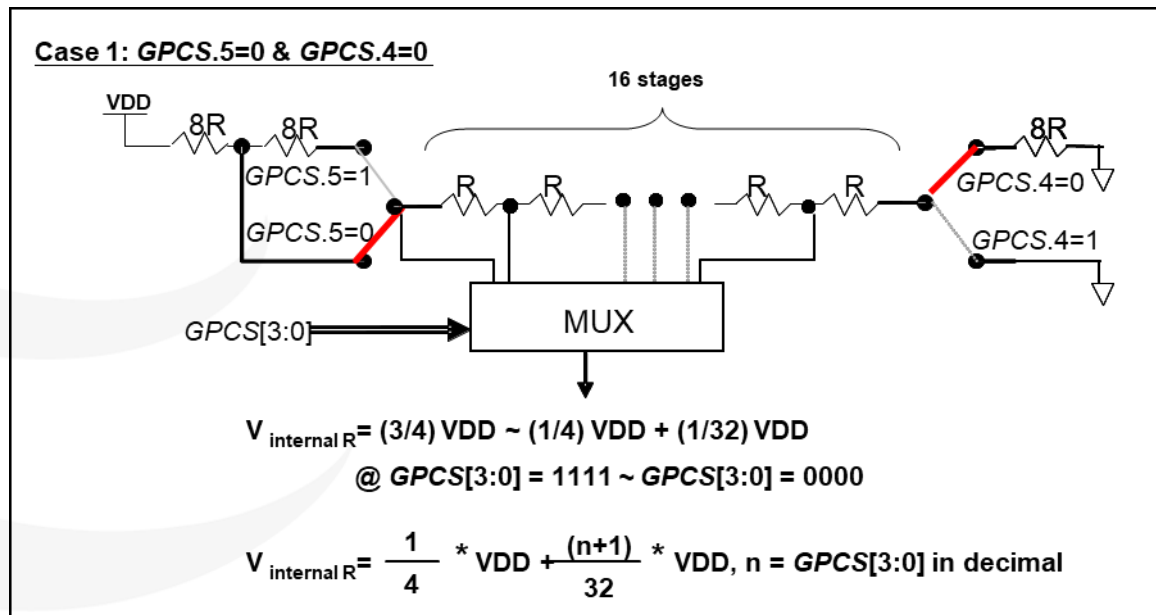


Fig. 30:  $V_{internal R}$  hardware connection if  $GPCS.5=0$  and  $GPCS.4=0$

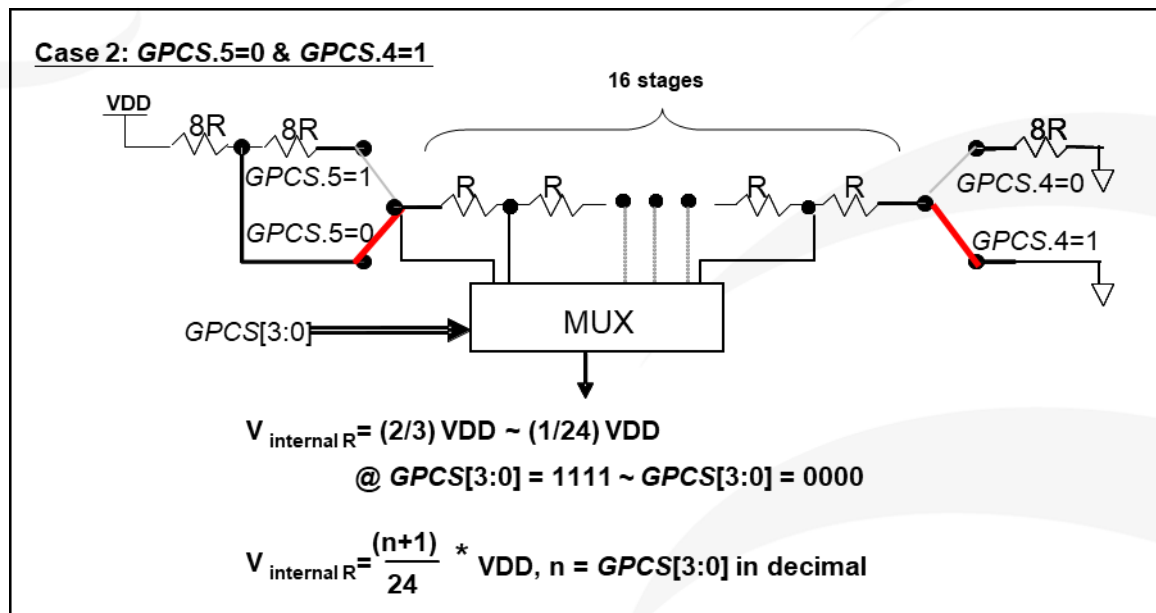


Fig. 31:  $V_{internal R}$  hardware connection if  $GPCS.5=0$  and  $GPCS.4=1$

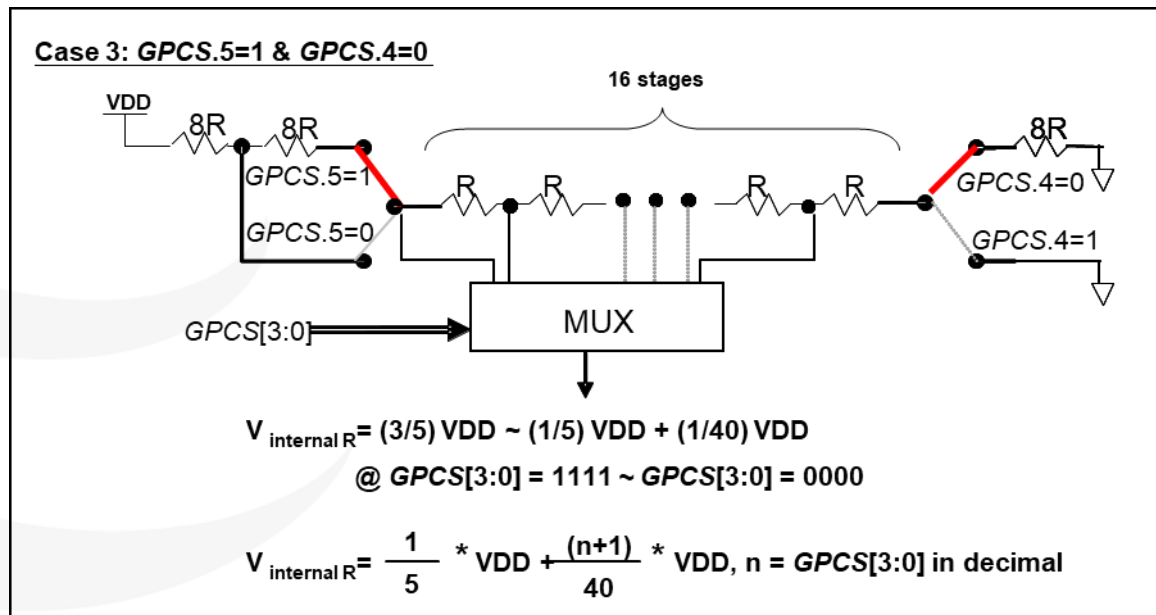


Fig. 32:  $V_{internal R}$  hardware connection if  $GPCS.5=1$  and  $GPCS.4=0$

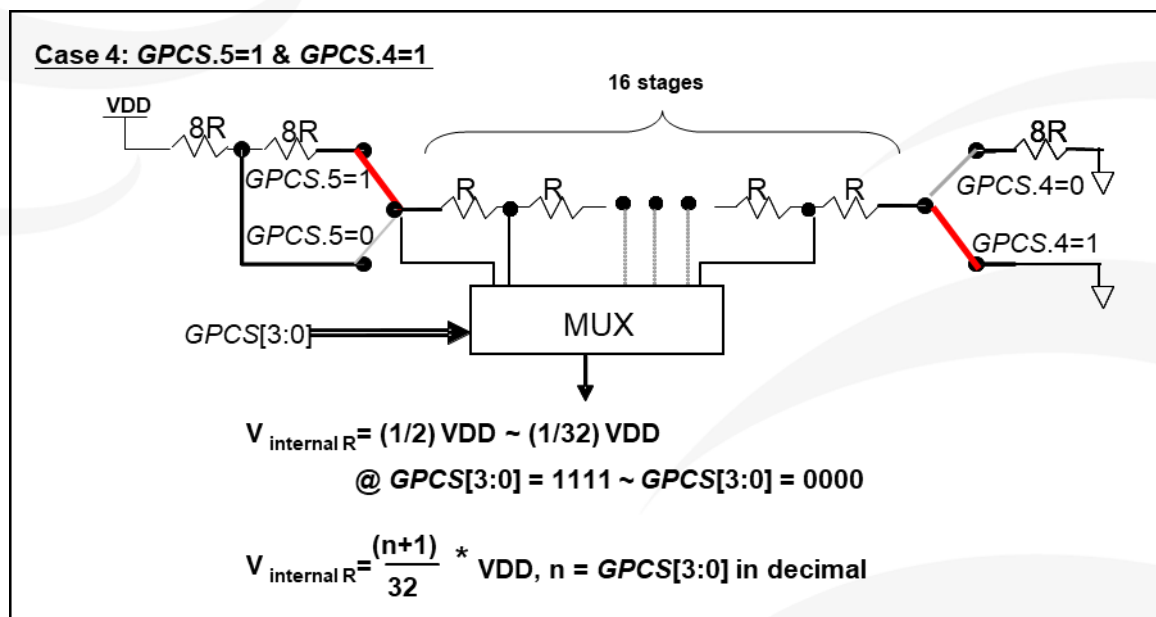


Fig. 33:  $V_{internal R}$  hardware connection if  $GPCS.5=1$  and  $GPCS.4=1$

## 11.1.8. Synchronizing General Purpose Comparator 1 Output to Timer2

The general purpose comparator output can be synchronized with Timer2 by setting  $GPC1C.5=1$ . When enabled, the comparator output is sampled by the rising edge of Timer2 clock source (TM2\_CLK). Please refer to Timer2 hardware diagram and general purpose comparator hardware diagram, if the pre-scalar function is used with Timer2, the comparator output is sampled after the pre-scaling and scaling functions, and will be sent to Timer2 counter for counting and comparator for sampling clock.

## 11.1.9. Synchronizing General Purpose Comparator 2 Output to Timer3

The general purpose comparator 2 output can be synchronized with Timer3 by setting  $GPC2C.5=1$ . When enabled, the comparator output is sampled by the rising edge of Timer3 clock source (TM3\_CLK). Please refer to Timer3 hardware diagram and general purpose comparator 2 hardware diagram, if the pre-scalar function is used with Timer3, the comparator output is sampled after the pre-scaling and scaling functions, and will be sent to Timer3 counter for counting and comparator for sampling clock.

## 11.1.10. Using the Comparator1

### Case 1:

Choosing PB6 as minus input and  $V_{\text{internal R}}$  with  $(18/32)*V_{DD}$  voltage level as plus input.  $V_{\text{internal R}}$  is configured as the above Figure “ $GPC1S[5:4] = 2b'00$ ” and  $GPC1S[3:0] = 4b'1001$  ( $n=9$ ) to have  $V_{\text{internal R}} = (1/4)*V_{DD} + [(9+1)/32]*V_{DD} = [(9+9)/32]*V_{DD} = (18/32)*V_{DD}$ .

```
GPC1S = 0b1_0_00_1001;    // output to PB0,  $V_{\text{internal R}} = V_{DD}*(18/32)$ 
GPC1C = 0b1_0_0_0_110_0;   // enable comp, - input: PB6, + input:  $V_{\text{internal R}}$ 
PBDIER = 0bx_0_xx_xxxx;    // disable PB6 digital input to prevent leakage current
```

### Case 2:

Choosing  $V_{\text{internal R}}$  as plus input with  $(22/40)*V_{DD}$  voltage level and PB0 as minus input, the comparator result will be inversed and without output to PB0.  $V_{\text{internal R}}$  is configured as the above Figure “ $GPC1S[5:4] = 2b'10$ ” and  $GPC1S[3:0] = 4b'1101$  ( $n=13$ ) to have  $V_{\text{internal R}} = (1/5)*V_{DD} + [(13+1)/40]*V_{DD} = [(13+9)/40]*V_{DD} = (22/40)*V_{DD}$ .

```
GPC1S = 0b0_0_10_1101;    //  $V_{\text{internal R}} = V_{DD}*(22/40)$ 
GPC1C = 0b1_0_0_1_000_0;   // Inverse output, + input:  $V_{\text{internal R}}$ , - input: PB0
PBDIER = 0bxxxx_xxx_0;    // disable PB0 digital input to prevent leakage current
```

## 11.2. 8X Operational Amplifier (OPamp) module

An Operational Amplifier (OPA) is built-in in the PFC886, the basic configuration is shown in the following diagram.

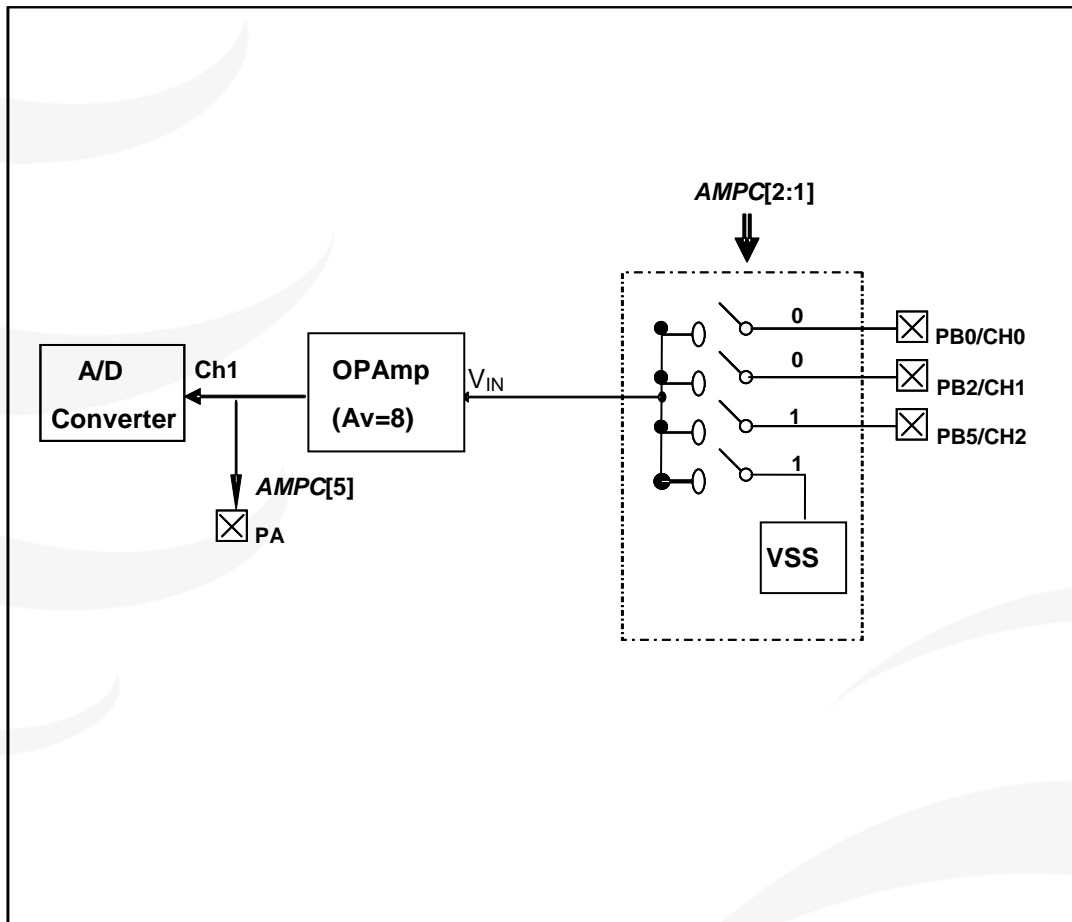


Fig. 34: OPamp Block Diagram

The PFC886 provides one single ended input, single ended output OPamp module with three external channels and one internal channel for internal VSS reference voltage; it processes the sensor small signal of the single ended input and amplified 8 times to ADC module or PA6. Before using OPamp, one need to use the VSS channel and record the offset voltage, then switch to another channel to generate an amplified signal.

**It is recommended to place a capacitance higher than 1uF and 0.1uF between VDD and GND for better OPamp characteristics.**



## 11.2.1. Configure the analog pins

The 3 external analog input signals for OPAm shared with PB0, PB2 and PB5. In order to avoid leakage current at the digital circuit portion, those pins which are defined for analog inputs should disable the digital input function (set the corresponding bit of *PBDIER* register to be 0). Because the measurement signals of OPAm belong to small signal; it should avoid the measured signal to be interfered during the measurement period.

The selected pin should:

- (1) be set to input mode
- (2) turn off weak pull-high resistor
- (3) set the corresponding pin to analog input by port B digital input disable register (*PBDIER*)

**It is recommended to place a 100 ohm resistor and 1nF capacitor at analog input channel for better OPAm characteristics.**

## 11.2.2. OPA Control Register (*AMPC*), address = 0x18

Bit	Reset	R/W	Description
7	0	WO	Power down both Band-gap and LVR hardware modules. 0 / 1 : Normal / Power-down
6	1	R/W	ADC internal reference high voltage selection. 0: 1.2V 1: VDD
5	0	R/W	OPAm output to PA6. 0: Disable 1: Enable
4	0	R/W	Read as 0.
3	0	R/W	Reserved.
2 - 1	00	R/W	OPAm Channel selector. These two bits are used to select input signal for OPAm. 00: PB0, 01: PB2, 10: PB5, 11: VSS
0	0	R/W	OPAm function. 0: Disable 1: Enable

## 11.3. Analog-to-Digital Conversion (ADC) module

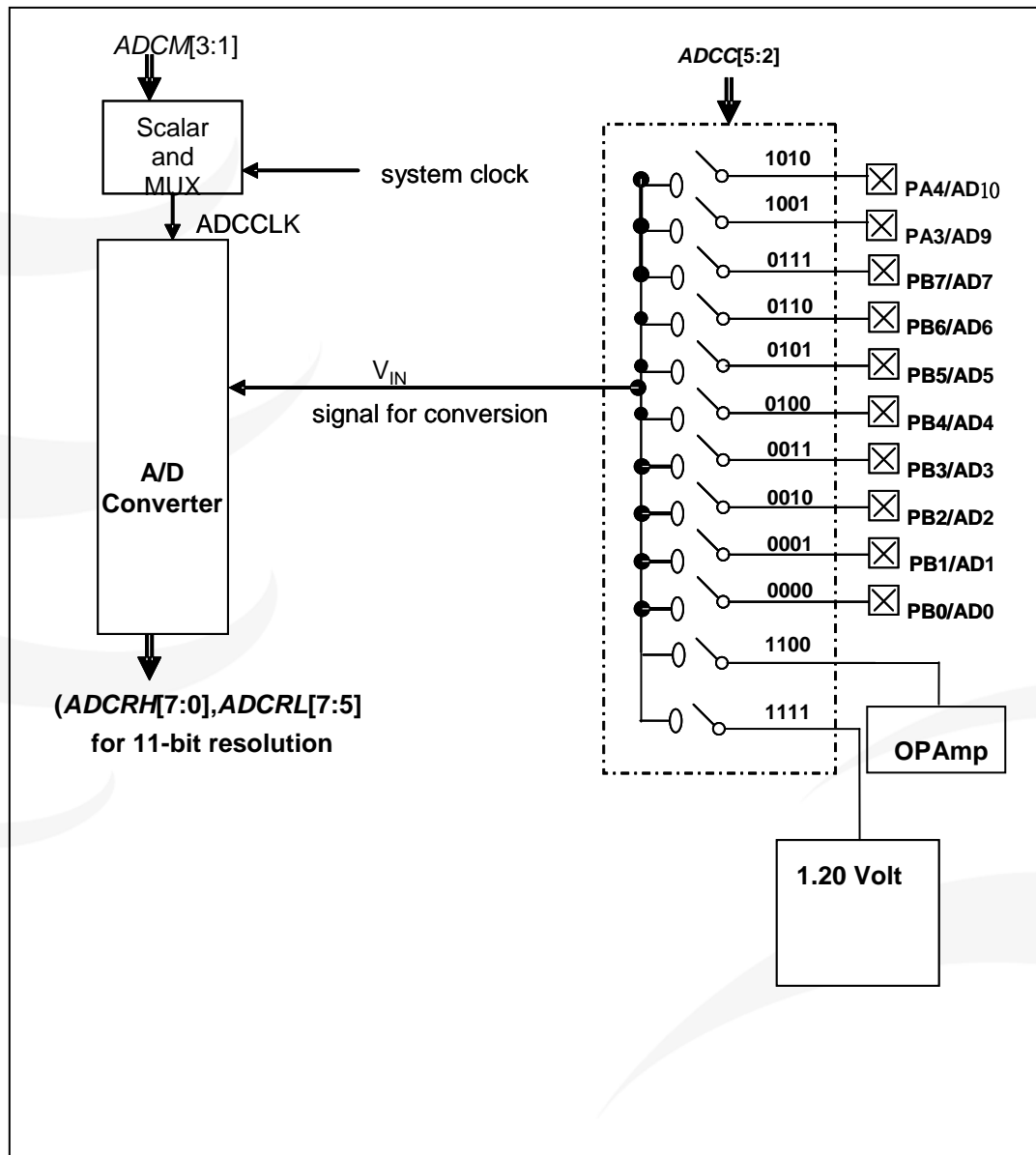


Fig.35: ADC Block Diagram

The PFC886 provides one 11-bit resolution analog-to-digital conversion module with 10 external channels and 2 internal channels respectively for 1.20 volt Band-gap reference voltage and amplified signal from OPamp.

For the conversion process, analog signal will be sampled and held first, then sending into the converter to generate the result via successive approximation. The analog reference high voltage of ADC is the positive supply voltage (VDD) and the reference low is always the GND.

**Higher than 1uF capacitor is recommended to be placed between VDD and GND to have better AD conversion result.**

## 11.3.1. The input requirement for AD conversion

For the AD conversion to meet its specified accuracy, the charge holding capacitor ( $C_{HOLD}$ ) must be allowed to fully charge to the voltage reference high level ( $V_{DD}$ ) and discharge to the voltage reference low level ( $GND$ ). The analog input model is shown as Fig.36, the signal driving source impedance ( $R_s$ ) and the internal sampling switch impedance ( $R_{ss}$ ) will affect the required time to charge the capacitor  $C_{HOLD}$  directly. The internal sampling switch impedance may vary with ADC supply voltage; the signal driving source impedance will affect accuracy of analog input signal. User must ensure the measured signal is stable before sampling; therefore, the maximum signal driving source impedance is highly dependent on the frequency of signal to be measured. The recommended maximum impedance for analog driving source is about 10K $\Omega$  under 500KHz input frequency.

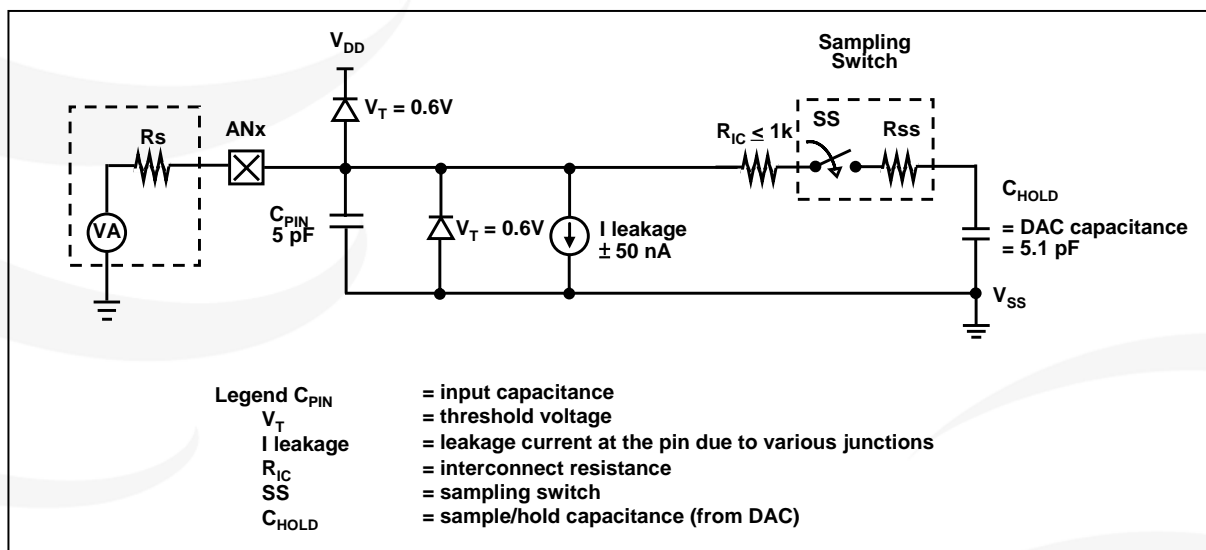


Fig.36: Analog Input Model

Before starting the AD conversion, the minimum signal acquisition time should be met for the selected analog input signal. The signal acquisition time ( $T_{ACQ}$ ) of ADC in PFC886 series is fixed to one clock period of ADCLK, the selection of ADCLK must be met the minimum signal acquisition time.

## 11.3.2. ADC clock selection

The clock of ADC module (ADCLK) can be selected by *ADCM* register; there are 8 possible options for ADCLK from  $CLK \div 1$  to  $CLK \div 128$  ( $CLK$  is the system clock). Due to the signal acquisition time  $T_{ACQ}$  is one clock period of ADCLK, the ADCLK must meet that requirement. The recommended ADC clock is to operate at 2 $\mu$ s.

## 11.3.3. AD conversion

The process of AD conversion starts from setting START/DONE (ADCC.6) to high, the START/DONE flag for read will be cleared automatically, then converting analog signal bit by bit and finally setting START/DONE high to indicate the completion of AD conversion. If ADCLK is selected,  $T_{ADCLK}$  is the period of ADCLK and the AD conversion time can be calculated as follows:

- ◆ 11-bit resolution: AD conversion time = 14  $T_{ADCLK}$

## 11.3.4. Configure the analog pins

There are 12 analog signals can be selected for AD conversion, 10 analog input signals come from external pins and one is from internal 1.20 Volt Bandgap reference voltage and the another one is from OPamp. For external pins, the analog signals are shared with PA5, PA6 and PB[7:0]. To avoid leakage current at the digital circuit, those pins defined for analog input should disable the digital input function (set the corresponding bit of *PADIER* or *PBDIER* register to be 0).

The measurement signals of ADC belong to small signal; it should avoid the measured signal to be interfered during the measurement period. The selected pin should:

- (1) be set to input mode
- (2) turn off weak pull-high resistor
- (3) set the corresponding pin to analog input by port A/B digital input disable register (*PADIER* / *PBDIER*).

The following steps are recommended to do the AD conversion procedure:

- (1) Configure the ADC module:
  - ◆ Configure the voltage reference high by *ADCC* register
  - ◆ Select the ADC input channel by *ADCC* register
  - ◆ Configure the AD conversion clock by *ADCM* register
  - ◆ Configure the pin as analog input by *PADIER* / *PBDIER* register
  - ◆ Enable the ADC module by *ADCC* register
- (2) Configure interrupt for ADC: (if desired)
  - ◆ Clear the ADC interrupt request flag in bit 3 of *INTRQ* register
  - ◆ Enable the ADC interrupt request in bit 3 of *INTEN* register
  - ◆ Enable global interrupt by issuing *engint* command
- (3) Start AD conversion:
  - ◆ Set ADC process control bit in the *ADCC* register to start the conversion (set1 *ADCC.6*).
- (4) Wait for the completion flag of AD conversion, by either:
  - ◆ Waiting for the completion flag by using command “*wait1 ADCC.6*”; or
  - ◆ Waiting for the ADC interrupt.
- (5) Read the ADC result registers:
  - ◆ Read *ADCRH* and *ADCRL* the result registers
- (6) For next conversion, go to step 1 or step 2 as required.

## 11.3.5. Using the ADC

The following example shows how to use ADC with PB0~PB3.

First, defining the selected pins:

```
PBC      = 0B_XXXX_0000;      // PB0 ~ PB3 as Input
PBPH    = 0B_XXXX_0000;      // PB0 ~ PB3 without pull-high
PBDIER  = 0B_XXXX_0000;      // PB0 ~ PB3 digital input is disabled
```

Next, setting ADCC register, example as below:

```
$ ADCC Enable, PB3;           // set PB3 as ADC input
$ ADCC Enable, PB2;           // set PB2 as ADC input
$ ADCC Enable, PB0;           // set PB0 as ADC input
```

Next, setting ADCM and ADCRGC register, example as below:

```
$ ADCM /16;                   // recommend /16 @System Clock=8MHz
$ ADCM /8;                    // recommend /8 @System Clock=4MHz
```

Then, start the ADC conversion:

```
AD_START = 1;                 // start ADC conversion
while (! AD_DONE) NULL;       // wait ADC conversion result
```

Finally, it can read ADC result when AD\_DONE is high:

```
WORD      Data;              // two bytes result: ADCRH and ADCRL
Data$1    = ADCRH
Data$0    = ADCRL;
Data      = Data >> 4;        // or Data = (ADCRH << 8) | ADCRL;
```

The ADC can be disabled by using the following method:

```
$ ADCC Disable;
```

or

```
ADCC      = 0;
```

## 11.3.6. ADC Related Registers

### 11.3.6.1. ADC Control Register (*ADCC*), address = 0x20

Bit	Reset	R/W	Description
7	0	R/W	Enable ADC function. 0/1: Disable/Enable.
6	-	R/W	ADC process control bit. Write "1" to start AD conversion, and the flag is cleared automatically when starting the AD conversion ; Read "1" to indicate the completion of AD conversion and "0" is in progressing.
5 - 2	0000	R/W	Channel selector. These four bits are used to select input signal for AD conversion. 0000: PB0/AD0, 0001: PB1/AD1, 0010: PB2/AD2, 0011: PB3/AD3, 0100: PB4/AD4, 0101: PB5/AD5, 0110: PB6/AD6, 0111: PB7/AD7 1001: PA3/AD9 1010: PA4/AD10 1100: OPamp 1111: Bandgap 1.2 volt reference voltage Others: reserved
1 - 0	-	-	Reserved. Please keep 0.

### 11.3.6.2. ADC Mode Register (*ADCM*), address = 0x21

Bit	Reset	R/W	Description
7 - 4	-	-	Reserved (keep 0 for future compatibility).
3 - 1	000	WO	ADC clock source selection. 000: CLK (system clock) ÷ 1, 001: CLK (system clock) ÷ 2, 010: CLK (system clock) ÷ 4, 011: CLK (system clock) ÷ 8, 100: CLK (system clock) ÷ 16, 101: CLK (system clock) ÷ 32, 110: CLK (system clock) ÷ 64, 111: CLK (system clock) ÷ 128,
0	-	-	Reserved

### 11.3.6.3. ADC Result High Register (*ADCRH*), address = 0x4A

Bit	Reset	R/W	Description
7 - 0	-	RO	These eight read-only bits will be the bit [11:4] of AD conversion result. The bit 7 of this register is the MSB of ADC result for any resolution.

## 11.3.6.4. ADC Result Low Register (*ADCRL*), address = 0x4B

Bit	Reset	R/W	Description
7 - 4	-	RO	These four bits will be the bit [3:0] of AD conversion result.
3 - 0	-	-	Reserved

## 11.4. PWM generator Trigger AD Conversion

The PWM generator also supports triggering AD conversion. It can trigger AD conversion by setting the trigger enable bit of Hop register (*HOP.4*) and ADC enable bit of ADCC register(*ADCC.7*). When the PWM counter count matches the set value of the *PWMADCH* and *PWMADCL* registers, it will issue a control signal to trigger the ADC start, as shown in Fig. 37. The trigger enable bit will be cleared automatically after the AD conversion is completed.

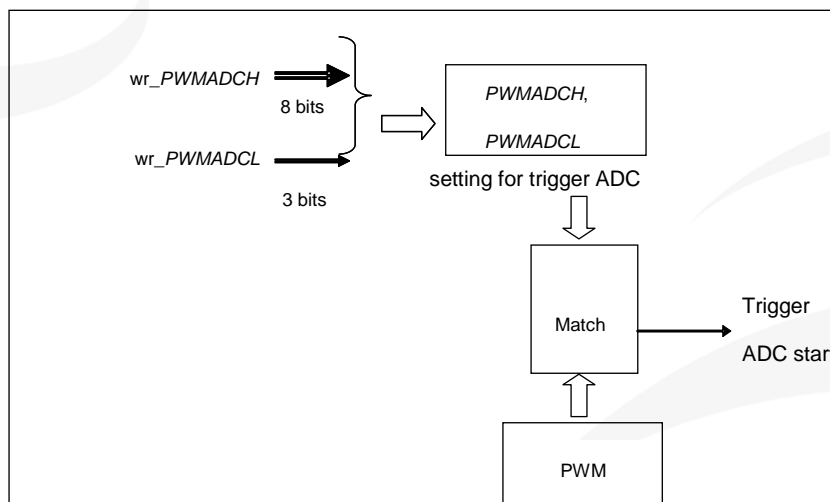


Fig. 37: Block Diagram of PWM generator trigger AD conversion

## 11.4.1. Hopping Setting Register (*HOP*), address = 0x23

Bit	Reset	R/W	Description
7	0	R/W	PWM interrupt mode. 0: Generate interrupt request when counter matches the boundary value 1: Generate interrupt request when counter is 0.
6	0	R/W	PWM counter trigger ADC mode. 0: up count matched PWMADC register. 1: down count matched PWMADC register.
5	0	R/W	Hall interrupt pins selection. 0: PB0/PB1/PB2. 1: PB5/PB6/PB7.
4	0	R/W	ADC can trigger by PWM (Hardware clear by ADC done) 0: Disable 1: Enable
3 - 0	-	-	Reserved.

## 11.4.2. PWM Trigger ADC - PWM Counter High Register (*PWMADCH*), address = 0x5E

Bit	Reset	R/W	Description
7 - 0	8'h00	WO	Trigger ADC - PWM counter values bit[11:4] setting.

## 11.4.3. PWM Trigger ADC - PWM Counter Low Register (*PWMADCL*), address = 0x5F

Bit	Reset	R/W	Description
7 - 5	3'h0	WO	Trigger ADC - PWM counter values bit[3:1] setting.
4 - 0	-	-	Reserved.



## 11.5. 1 or 3-phase Brushless DC Motor

### 11.5.1. Single-Phase PWM Protection

For single-phase motor application, it's quite important to avoid ON state for both high side and low side simultaneously. PFC886 provides the hardware PWM protection circuit, shown as Fig. 38; the hardware PWM protection module controlled by *PWMGC* and *PWMGM* registers to meet single-phase BLDC protection application. If the setting of *PWMGM.4* mismatches with the state of high side and the protection is enable, the PWM generator will be disabled to force output in inactive state.

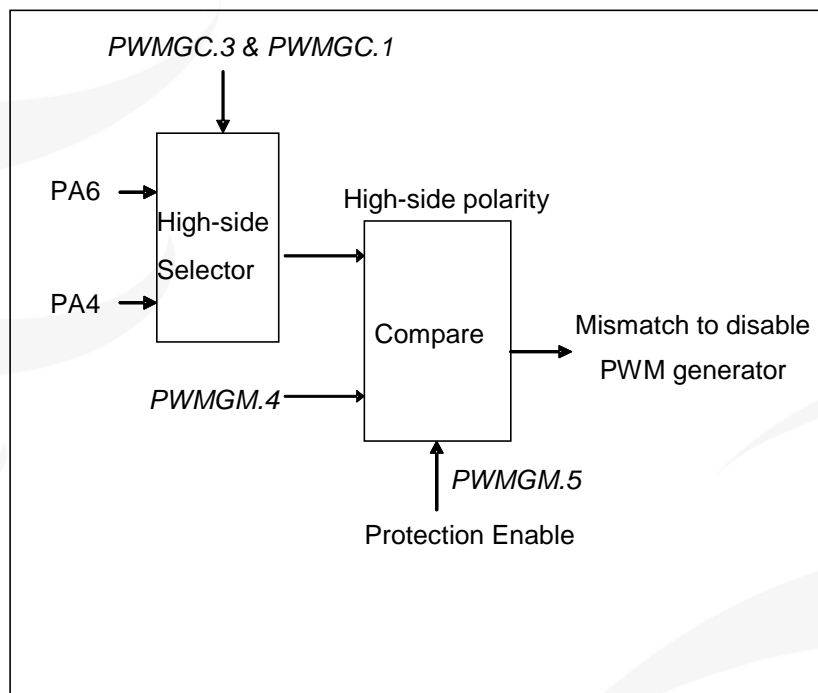


Fig. 38: Block Diagram of Single-Phase PWM protection

PWM Generator Scalar Register ( <i>PWMGM</i> ), address = 0x31			
Bit	Reset	R/W	Description
7	0	W	Enable to inverse the polarity of high-side PWM output. 0 / 1 : disable / enable.
6	0	W	Enable to inverse the polarity of low-side PWM output. 0 / 1 : disable / enable.
5	0	R/W	<b>Side Protection Enable.</b> 0 / 1: disable / enable.
4	0	R/W	<b>Side Protection mode.</b> 0 : enable protection when High-side is in high level 1 : enable protection when High-side is in low level
3 – 2	00	R/W	PWM generator clock source.
1 – 0	00	R/W	PWM generator clock pre-scalar.

## 11.5.2. Three-Phase PWM Protection

For three-phase motor application, it's quite important to avoid ON state for both high side and low side simultaneously. PFC886 provides the hardware Three-Phase PWM protection circuit, shown as Fig. 39; the hardware PWM protection module controlled by *PWMGM* registers to meet three-phase BLDC protection application. If the setting of *PWMGM.4* mismatches with the polarity state of high side and the protection is enable, the PWM generator will be disabled to force output in inactive state.

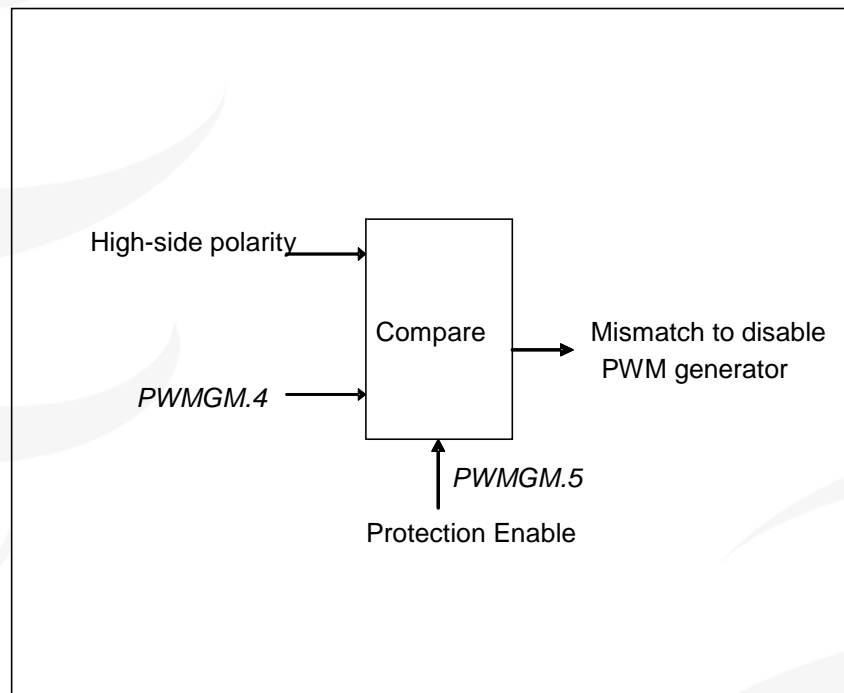


Fig. 39: Block Diagram of PWM protection

## 11.6. Input Pulse Capture

The feature of input Pulse Capture is useful in applications which requiring frequency and pulse measurement. Fig. 40 shows the hardware diagram of input Pulse Capture in PFC886, the time base of Pulse Capture module can be IHRC and IHRCX2, the input signals for measurement can be comparator 2 output, PB1, PA5, PB6, PB0 or PB7.

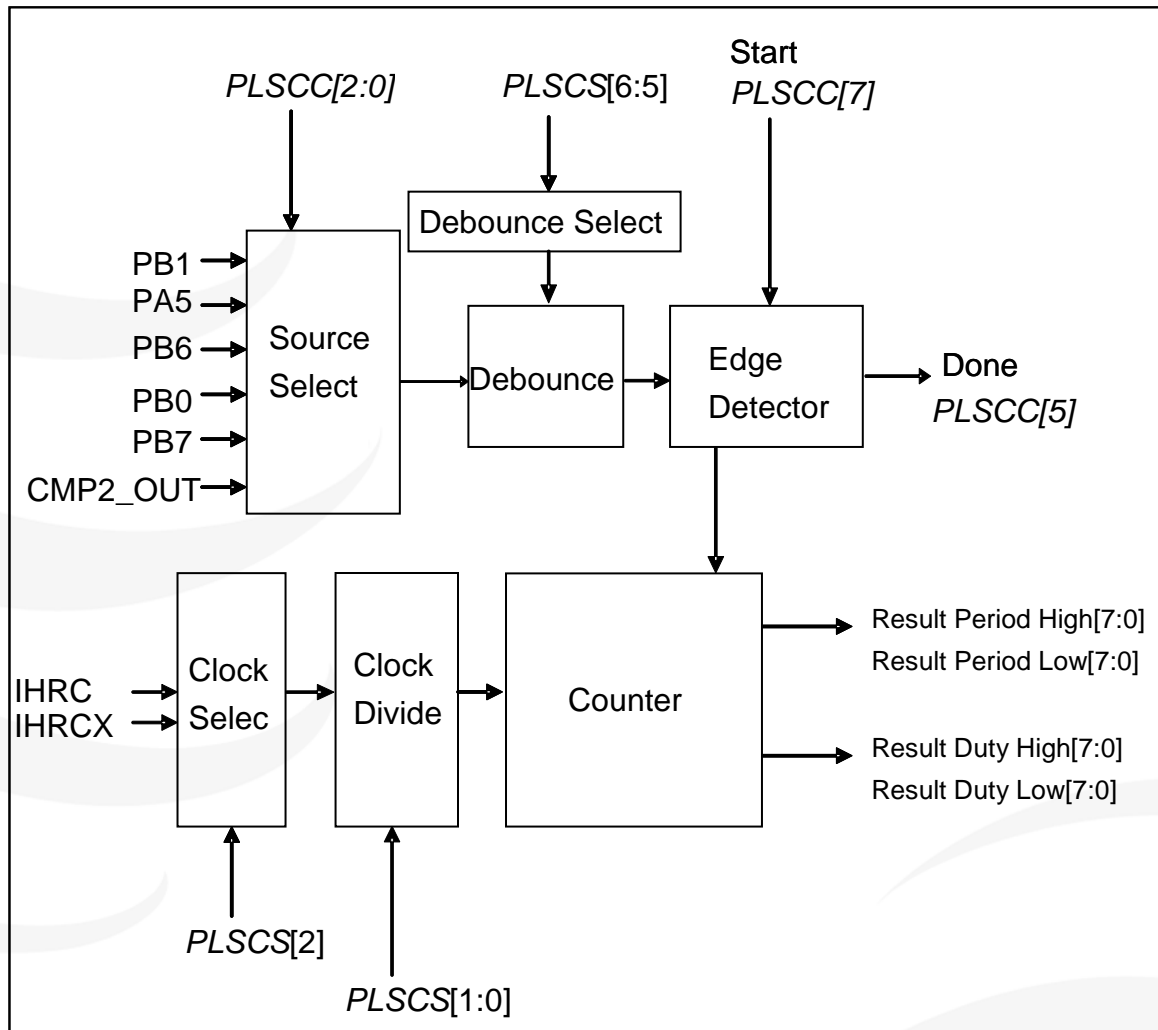


Fig. 40: Hardware Diagram of Input Pulse Capture

## 11.6.1. Pulse Capture Control Register (*PLSCC*), address = 0x32

Bit	Reset	R/W	Description
7	0	R/W	W: Start to do Pulse Capture and clear Pulse Capture Overflow or Done Flag. R: Pulse Capture overflow or Done Flag.
6	0	R	Pulse Capture overflow Flag. clears this bit automatically after finishing the Pulse Capture operation
5	0	R/W	W: Clear Pulse Capture Done Flag. R: Pulse Capture Done Flag.
2 - 0	000	R/W	Sources for Pulse Capture. 000: PB1 001: PA5 010: PB6 011: PB0 100: PB7 Others: comparator 2 output

Write Register only support by MOV instruction

## 11.6.2. Pulse Capture Scalar Register (*PLSCS*), address = 0x33

Bit	Reset	R/W	Description
7	-	-	Reserved, please keep 0.
6 - 5	00	R/W	Debounce of Pulse Capture input source. 00 : None 01 : 1 Pulse Capture clock 10 : 2 Pulse Capture clocks 11 : 3 Pulse Capture clocks
4 - 3	-	-	Reserved, please keep 0.
2	00	R/W	Pulse Capture clock source. 0: IHRC 1: IHRC*2
1 - 0	00	R/W	Clock divider of Pulse Capture. 00 : /1 01 : /4 10 : /16 11 : /64

## 11.6.3. Pulse Capture Pulse Width High Register (*PLSPWH*), address = 0x4C

Bit	Reset	R/W	Description
7 - 0	-	RO	High byte of Pulse Capture Pulse Width.

## 11.6.4. Pulse Capture Pulse Width Low Register (*PLSPWL*), address = 0x4D

Bit	Reset	R/W	Description
7 - 0	-	RO	Low byte of Pulse Capture Pulse Width.

## 11.6.5. Pulse Capture Pulse High High Register (*PLSPHH*), address = 0x4E

Bit	Reset	R/W	Description
7 - 0	-	RO	High byte of Pulse Capture Pulse High.

## 11.6.6. Pulse Capture Pulse High Low Register (*PLSPHL*), address = 0x4F

Bit	Reset	R/W	Description
7 - 0	-	RO	Low byte of Pulse Capture Pulse High.

## 11.7. Multiplier and Divider

### 11.7.1. 16x8 multiplier

There is a 16x8 multiplier on the PFC886 that enhances the hardware capabilities of the arithmetic function. The multiplication is a 16x8 unsigned operation and can be completed in 9 system clock cycles. Before setting the start bit (*EARITH.6*), the multiplicand and multiplier must be placed in registers *M8OP1H/M8OP1L* and *M8OP2*; after 16x8 is completed, the done bit will be set (*EARITH.1*) and the result will be placed in registers *M8RS2/M8RS1/M8RS0*. The hardware diagram of this multiplier is shown as Fig. 41.

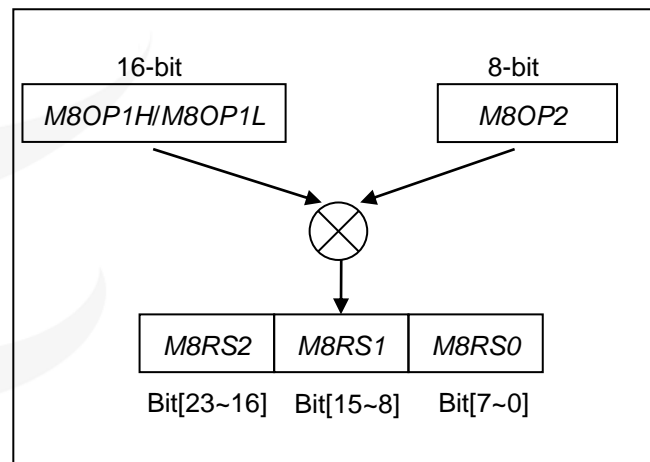


Fig. 41: Block diagram of hardware multiplier

### 11.7.2. 16x16 multiplier

There is a 16x16 multiplier on the PFC886 that enhances the hardware capabilities of the arithmetic function. The multiplication is a 16x16 unsigned operation and can be completed in 9 system clock cycles. Before setting the start bit (*EARITH.7*), the multiplicand and multiplier must be placed in registers *M16OP1H/M16OP1L* and *M16OP2H/M16OP2L*; after 16x16 is completed, the done bit will be set (*EARITH.2*) and the result will be placed in registers *M16RS3/M16RS2/M16RS1/M16RS0*. The hardware diagram of this multiplier is shown as Fig. 42.

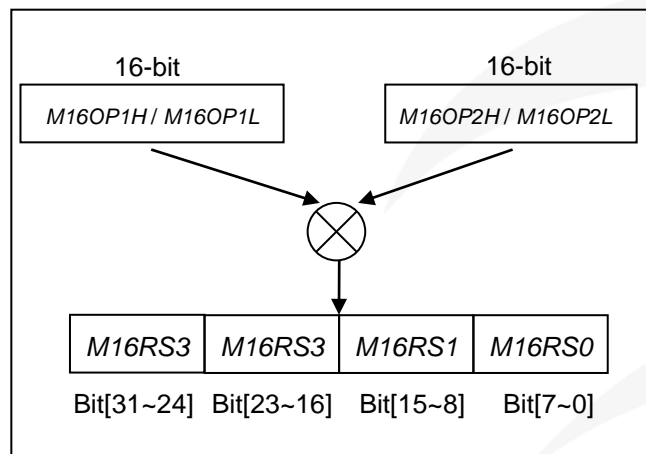


Fig. 42: Block diagram of hardware multiplier

## 11.7.3. 16/16 Divider

There is a 16/16 divider on the PFC886 that enhances the hardware capabilities of the arithmetic function. The division is a 16/16 unsigned operation and can be completed in 14 system clock cycles. Before setting the start bit (*EARITH.5*), the dividend and divisor must be placed in registers *D16DEH/D16DEL* and *D16DSH/D16DSL*; after 16/16 completed, the done bit will be set (*EARITH.0*), the quotient result will be placed in registers *D16QUH/D16QUL* and the remainder result will be placed in registers *D16REH/D16REL*. The hardware diagram of this divider is shown as Fig. 43.

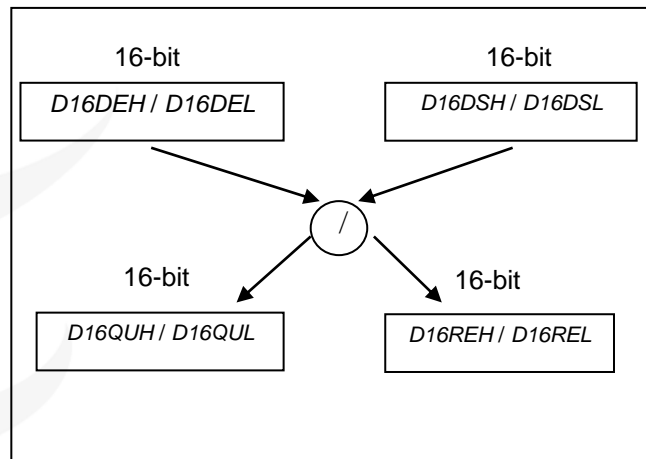


Fig.43: Block diagram of hardware divider

## 11.7.4. Arithmetic Operation Register (*EARITH*), address = 0x3C

Bit	Reset	R/W	Description
7	-	WO	16X16 Multiplier start bit and clear by hardware
6	-	WO	16X8 Multiplier start bit and clear by hardware
5	-	WO	16/16 Divider start bit and clear by hardware
4 - 3	-	WO	Reserved, please keep 0.
2	-	RO	16X16 Multiplier done flag and clear by start bit
1	-	RO	16X8 Multiplier done flag and clear by start bit
0	-	RO	16/16 Divider done flag and clear by start bit

## 11.7.5. 16X8 Multiplier Operand 1 High Byte Register (*M8OP1H*), address = 0x44

Bit	Reset	R/W	Description
7 - 0	-	WO	Operand 1 High Byte for hardware multiplication operation.

## 11.7.6. 16X8 Multiplier Operand 1 Low Byte Register (*M8OP1L*), address = 0x45

Bit	Reset	R/W	Description
7 - 0	-	WO	Operand 1 Low Byte for hardware multiplication operation.

## 11.7.7. 16X8 Multiplier Operand 2 Register (*M8OP2*), address = 0x46

Bit	Reset	R/W	Description
7 - 0	-	WO	Operand 2 for hardware multiplication operation.

## 11.7.8. 16X8 Multiplier Result2 Register (*M8RS2*), address = 0x47

Bit	Reset	R/W	Description
7 - 0	-	RO	Result bits 23~16 of multiplication operation.

## 11.7.9. 16X8 Multiplier Result1 Register (*M8RS1*), address = 0x48

Bit	Reset	R/W	Description
7 - 0	-	RO	Result bits 15~8 of multiplication operation.

## 11.7.10. 16X8 Multiplier Result0 Register (*M8RS0*), address = 0x49

Bit	Reset	R/W	Description
7 - 0	-	RO	Result bits 7~0 of multiplication operation.

## 11.7.11. 16X16 Multiplier Operand 1 High Byte Register (*M16OP1H*), address = 0x60

Bit	Reset	R/W	Description
7 - 0	-	WO	Operand 1 High Byte for hardware multiplication operation.

## 11.7.12. 16X16 Multiplier Operand 1 Low Byte Register (*M16OP1L*), address = 0x61

Bit	Reset	R/W	Description
7 - 0	-	WO	Operand 1 Low Byte for hardware multiplication operation.

## 11.7.13. 16X16 Multiplier Operand 2 High Byte Register (*M16OP2H*), address = 0x62

Bit	Reset	R/W	Description
7 - 0	-	WO	Operand 2 High Byte for hardware multiplication operation.

## 11.7.14. 16X16 Multiplier Operand 2 Low Byte Register (*M16OP2L*), address = 0x63

Bit	Reset	R/W	Description
7 - 0	-	WO	Operand 2 Low Byte for hardware multiplication operation.

## 11.7.15. 16X16 Multiplier Result3 Register (*M16RS2H*), address = 0x64

Bit	Reset	R/W	Description
7 - 0	-	RO	Result bits 31~24 of multiplication operation (read only).

## 11.7.16. 16X16 Multiplier Result2 Register (*M16RS2L*), address = 0x65

Bit	Reset	R/W	Description
7 - 0	-	RO	Result bits 23~16 of multiplication operation (read only).

## 11.7.17. 16X16 Multiplier Result1 Register (*M16RS1H*), address = 0x66

Bit	Reset	R/W	Description
7 - 0	-	RO	Result bits 15~8 of multiplication operation (read only).

## 11.7.18. 16X16 Multiplier Result0 Register (*M16RS1L*), address = 0x67

Bit	Reset	R/W	Description
7 - 0	-	RO	Result bits 7~0 of multiplication operation (read only).

## 11.7.19. 16/16 Divider Dividend High Byte Register (*D16DEH*), address = 0x68

Bit	Reset	R/W	Description
7 - 0	-	WO	Dividend High Byte for hardware division operation.

## 11.7.20. 16/16 Divider Dividend Low Byte Register (*D16DEL*), address = 0x69

Bit	Reset	R/W	Description
7 - 0	-	WO	Dividend Low Byte for hardware division operation.

## 11.7.21. 16/16 Divider Divisor High Byte Register (*D16DSH*), address = 0x6A

Bit	Reset	R/W	Description
7 - 0	-	WO	Divisor High Byte for hardware division operation.

## 11.7.22. 16/16 Divider Divisor Low Byte Register (*D16DSL*), address = 0x6B

Bit	Reset	R/W	Description
7 - 0	-	WO	Divisor Low Byte for hardware division operation.

## 11.7.23. 16/16 Divider Quotient High Byte Register (*D16QUH*), address = 0x6C

Bit	Reset	R/W	Description
7 - 0	-	RO	Quotient High Byte for hardware division operation.

## 11.7.24. 16/16 Divider Quotient Low Byte Register (*D16QUL*), address = 0x6D

Bit	Reset	R/W	Description
7 - 0	-	RO	Quotient Low Byte for hardware division operation

## 11.7.25. 16/16 Divider Remainder High Byte Register (*D16REH*), address = 0x6E

Bit	Reset	R/W	Description
7 - 0	-	RO	Remainder High Byte for hardware division operation

## 11.7.26. 16/16 Divider Remainder Low Byte Register (*D16REL*), address = 0x6F

Bit	Reset	R/W	Description
7 - 0	-	RO	Remainder Low Byte for hardware division operation.



## 12. Program Writing

Please use 5S-P-003 to program. 3S-P-002 or older versions do not support programming PFC886.

Jumper connection: Please follow the instruction inside the writer software to connect the jumper.

Please select the following program mode according to the actual situation.

### 12.1. Normal Programming Mode

Range of application:

- Single-Chip-Package IC with programming at the writer IC socket or on the handler.
- Multi-Chip-Package (MCP) with PFC886. Be sure its connected IC and devices will not be damaged by the following voltages, and will not clam the following voltages.

**The voltage conditions in normal programming mode:**

- (1) VDD is 7.5V, and the maximum supply current is up to about 20mA.
- (2) PA5 is 5.5V.
- (3) The voltages of other program pins (except GND) are the same as VDD.

**Important Cautions:**

- You **MUST** follow the instructions on APN004 and APN011 for programming IC on the handler.
- Connecting a 0.1uF capacitor between VDD and GND at the handler port to the IC is always good for suppressing disturbance. But please **DO NOT** connect with  $\geq 0.22\mu\text{F}$  capacitor, otherwise, programming mode may be fail.

### 12.2. Limited-Voltage Programming Mode

Range of application:

- On-Board writing. Its peripheral circuits and devices will not be damaged by the following voltages, and will not clam the following voltages. Please refer to Chapter 13.3 for more details about On-Board Writing.
- Multi-Chip-Package (MCP) with PFC886. Please be sure that its connected IC and devices will not be damaged by the following voltages, and will not clam the following voltages.

**The voltage conditions in Limited-Voltage programming mode:**

- (1) VDD is 5.0V, and the maximum supply current is up to about 20mA.
- (2) PA5 is 5.0V.
- (3) The voltage of other program pins (except GND) is the same as VDD.

Please select "MTP On-Board VDD Limitation" or "On-Board Program" on the writer screen to enable the limited-voltage programming mode. (Please refer to the file of Writer "5S-P-003 UM").

## 12.3. On-Board Writing

PFC886 can support On-board writing. On-Board Writing is known as the situation that the IC has to be programmed when the IC itself and other peripheral circuits and devices have already been mounted on the PCB. Six wires of 5S-P-003 are used for On-Board Writing: ICPCCK, ICPDA, ICPDA2, VDD, GND and ICVPP. They are used to connect PA3, PA6, PA4, VDD, GND and PA5 of the IC correspondingly.

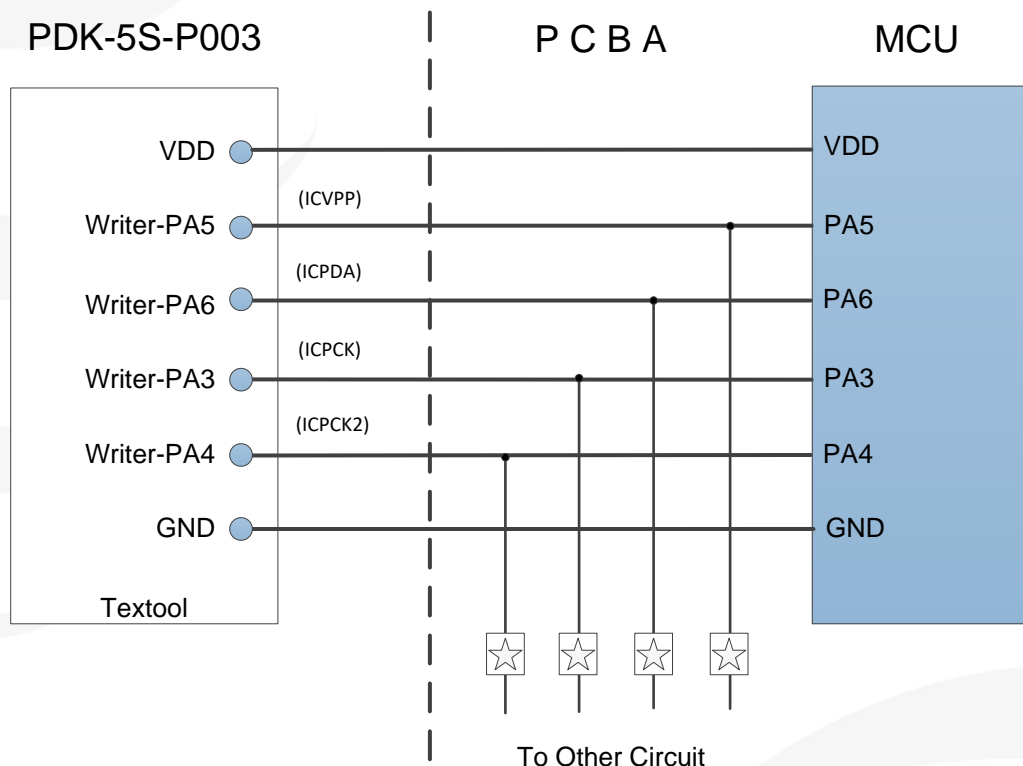


Fig. 44: Schematic Diagram of On-Board Wiring

The symbol ☆ on Fig. 44 can be either resistors or capacitors. They are used to isolate the programming signal wires from the peripheral circuit. It should be  $\geq 10K\Omega$  for resistance while  $\leq 220pF$  for capacitance.

### Notice:

- In general, the limited-voltage programming mode is used in On-board Writing; Please refers to the 12.2 for more detail about limited-voltage programming mode.
- Any zener diode  $\leq 5.0V$ , or any circuitry which clam the 5.0V to be created SHOULD NOT be connected between VDD and GND of the PCB.
- Any capacitor  $\geq 500\mu F$  SHOULD NOT be connected between VDD and GND of the PCB.
- In general, the writing signal pins SHOULD NOT be considered as strong output pins.

## 13. Device Characteristics

### 13.1. Absolute Maximum Ratings

Name	Min	Typ.	Max	Unit	Notes
Supply Voltage (VDD)	2.3		6	V	Exceed the maximum rating may cause permanent damage!!
Input Voltage	-0.3		$V_{DD} + 0.2$	V	
Operating Temperature	-40		85	°C	
Storage Temperature	-50		125	°C	
Junction Temperature		150		°C	

### 13.2. DC/AC Characteristics

Symbol	Description	Min	Typ	Max	Unit	Conditions (Ta=25°C)
V <sub>DD</sub>	Operating Voltage	2.3 <sup>#</sup>	5.0	6	V	<sup>#</sup> Subject to V <sub>BRD</sub> tolerance
V <sub>FSV</sub>	Forbidden V <sub>DD</sub> startup voltage range	0.7		1.6	V	
V <sub>PDRV</sub>	V <sub>DD</sub> power down release voltage			0.7	V	
T <sub>POR</sub>	V <sub>DD</sub> power on time (V <sub>DD</sub> from 0V to 5V)			50	ms	
T <sub>FSV</sub>	V <sub>DD</sub> power on time during V <sub>FSV</sub> range			10	ms	
f <sub>SYS</sub>	System clock IHRC IHRC Internal low RC oscillator	0 0	33.8K	8M 4M	Hz	V <sub>DD</sub> = 3.3V V <sub>DD</sub> = 2.3V V <sub>DD</sub> = 5.0V
P <sub>cycle</sub>	Program cycle	1000			cycles	
I <sub>OP</sub>	Operating Current		1.8 3.5 150 8		mA mA uA uA	f <sub>SYS</sub> =1MIPS@5.0V f <sub>SYS</sub> =8MIPS@5.0V f <sub>SYS</sub> =ILRC ~ 32KHz@5.0V f <sub>SYS</sub> =ILRC ~ 12KHz@3.3V
I <sub>PD</sub>	Power Down Current (by <b>stopsys</b> command)		3 1		uA uA	V <sub>DD</sub> =5.0V V <sub>DD</sub> =3.3V
I <sub>PS</sub>	Power Save Current (by <b>stopexe</b> command)		0.4		mA	V <sub>DD</sub> =5.0V; Band-gap, LVR, IHRC, ILRC, Timer16 modules are ON.
V <sub>IL</sub>	Input low voltage for IO lines	0		0.2V <sub>DD</sub>	V	
V <sub>IH</sub>	Input high voltage for IO lines	0.8 V <sub>DD</sub>		V <sub>DD</sub>	V	
I <sub>OL</sub>	IO lines sink current	11	14	17	mA	V <sub>DD</sub> =5.0V, V <sub>OL</sub> =0.5V
I <sub>OH</sub>	IO lines drive current	-8	-10	-12	mA	V <sub>DD</sub> =5.0V, V <sub>OH</sub> =4.5V
R <sub>PH</sub>	Pull-high Resistance		90		KΩ	V <sub>DD</sub> =5.0V

Symbol	Description	Min	Typ	Max	Unit	Conditions (Ta=25°C)
			170			V <sub>DD</sub> =3.3V
V <sub>BRD</sub>	Low Voltage Detect Voltage * (Brown-out voltage)		4.5* 4* 3.75* 3.5* 3.3* 3.15* 3.0* 2.7* 2.5* 2.4* 2.3*		V	
V <sub>BG</sub>	Band-gap Reference Voltage (before calibration)	1.12	1.20	1.28	V	V <sub>DD</sub> =5V, 25°C
	Band-gap Reference Voltage * (after calibration)	1.17*	1.20*	1.23*		V <sub>DD</sub> =3.15V ~ 5.5V, -40°C <Ta<85°C*
f <sub>IHRC</sub>	Frequency of IHRC after calibration *	15.52*	16*	16.48*	MHz	25°C, V <sub>DD</sub> =3.15V~5.5V
		14*	16*	17.28*		V <sub>DD</sub> =3.15V~5.5V, -40°C <Ta<85°C*
f <sub>ILRC</sub>	Frequency of ILRC *	31.5*	33.8*	35*	KHz	V <sub>DD</sub> =5.0V, Ta=25°C
		29*	33.8*	38.4*		V <sub>DD</sub> =5.0V, -40°C <Ta<85°C*
		32*	34*	35.5*		V <sub>DD</sub> =3.3V, Ta=25°C
		29*	34*	40*		V <sub>DD</sub> =3.3V, -40°C <Ta<85°C*
V <sub>ADC</sub>	Workable ADC operating Voltage	3.15		5.0	V	
V <sub>AD</sub>	AD Input Voltage	0		V <sub>DD</sub>	V	
ADrs	ADC resolution			11	bit	
ADclk	ADC clock period		2		us	3.15V ~ 5.5V
t <sub>ADCONV</sub>	ADC conversion time (T <sub>ADCLK</sub> is the period of the selected AD conversion clock)		14		T <sub>ADCLK</sub>	
AD DNL	ADC Differential NonLinearity		±3*		LSB	
AD INL	ADC Integral NonLinearity		±3*		LSB	
ADos	ADC offset*		3		LSB	-40°C <Ta<85°C*
t <sub>INT</sub>	Interrupt pulse width	30			ns	V <sub>DD</sub> = 5.0V
V <sub>DR</sub>	RAM data retention voltage*	1.5			V	In power-down mode.
t <sub>WDT</sub>	Watchdog timeout period (T <sub>ILRC</sub> is the clock period of ILRC)		4096			misc[1:0]=01
			16384			misc[1:0]=10
t <sub>SBP</sub>	System boot-up period from power-on		1024		T <sub>ILRC</sub>	Where T <sub>ILRC</sub> is the clock period of ILRC

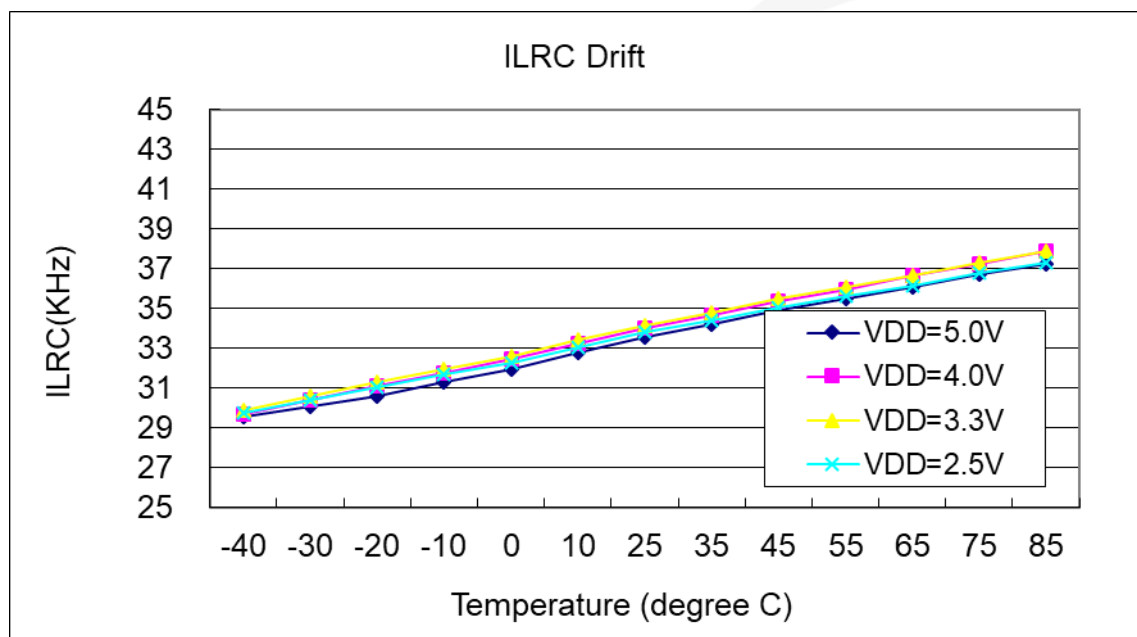
Symbol	Description	Min	Typ	Max	Unit	Conditions (Ta=25°C)
twup	System wake-up period					
	Fast wake-up by IO toggle from STOPEXE suspend		128		T <sub>sys</sub>	Where T <sub>sys</sub> is the time period of system clock
	Fast wake-up by IO toggle from STOPSYS suspend, IHRC is the system clock		128 T <sub>sys</sub> + T <sub>SIHRC</sub>			Where T <sub>SIHRC</sub> is the stable time of IHRC from power-on.
	Fast wake-up by IO toggle from STOPSYS suspend, ILRC is the system clock		128 T <sub>sys</sub> + T <sub>SILRC</sub>			Where T <sub>SILRC</sub> is the stable time of ILRC from power-on.
	Normal wake-up from STOPEXE or STOPSYS suspend		1024		T <sub>ILRC</sub>	Where T <sub>ILRC</sub> is the clock period of ILRC
HCPos	Comparator offset*	-	±10	±20	mV	
HCPcm	Comparator input common mode*	0		V <sub>DD</sub> -1.5	V	
HCPspt	Comparator response time**		100	500	ns	Both Rising and Falling
HCPmc	Stable time to change comparator mode		2.5	7.5	us	

\*These parameters are for design reference, not tested for every chip.

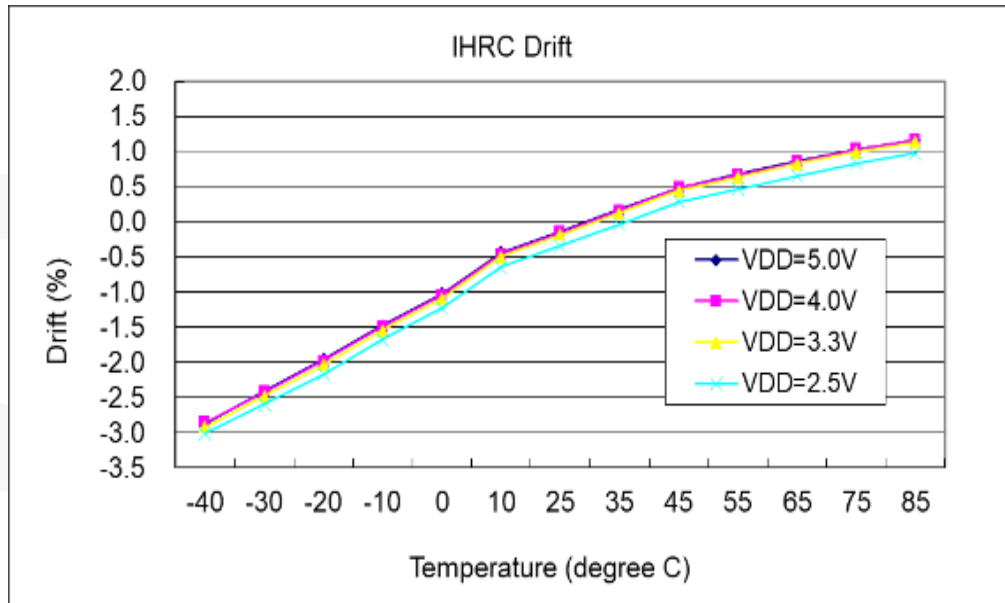
\*\* Response time is measured with comparator input at (V<sub>DD</sub>-1.5)/2 -100mV, and (V<sub>DD</sub>-1.5)/2+100mV

The characteristic diagrams are the actual measured values. Considering the influence of production drift and other factors, the data in the table are within the safety range of the actual measured values.

### 13.3. Typical ILRC frequency vs. VDD and temperature



## 13.4. Typical IHRC frequency deviation vs. VDD and temperature



## 13.5. Typical Operating Current vs. VDD and CLK=IHRC/n

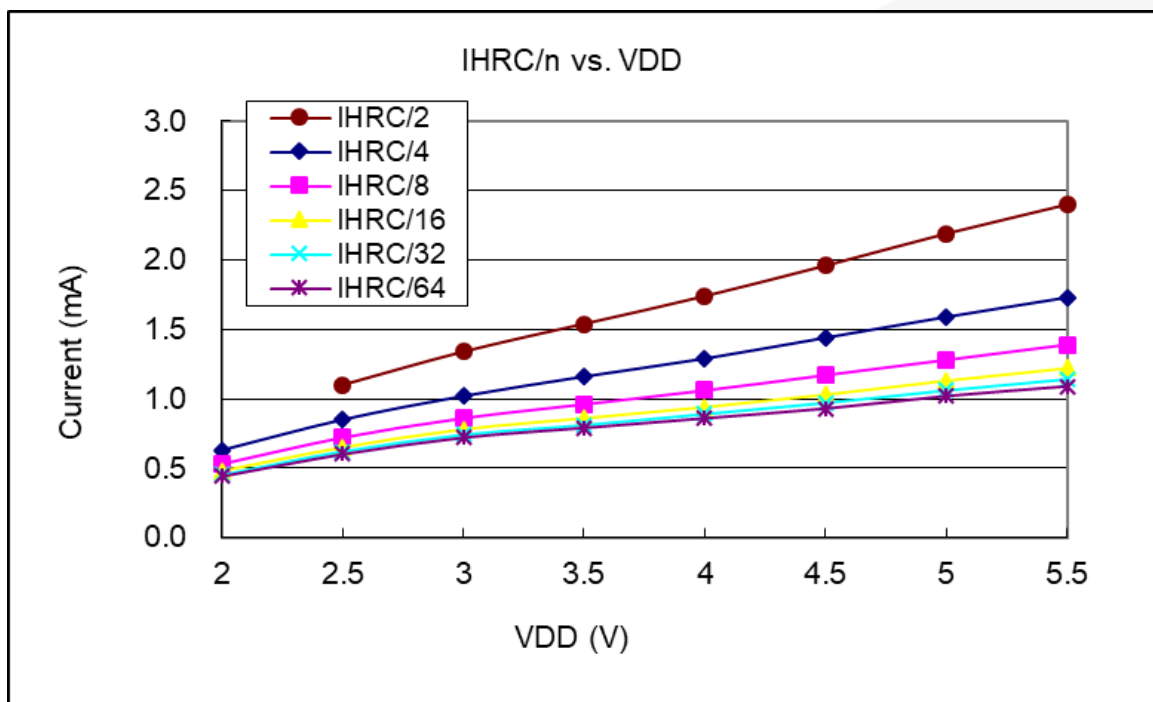
Conditions:

1-FPPA (code option)

**ON:** Band-gap, LVR, IHRC;

**OFF:** ILRC, T16, TM2, ADC, PWM;

**IO:** PA0:0.5Hz output toggle and no loading, others: input and no floating



## 13.6. Typical Operating Current vs. VDD and CLK=ILRC/n

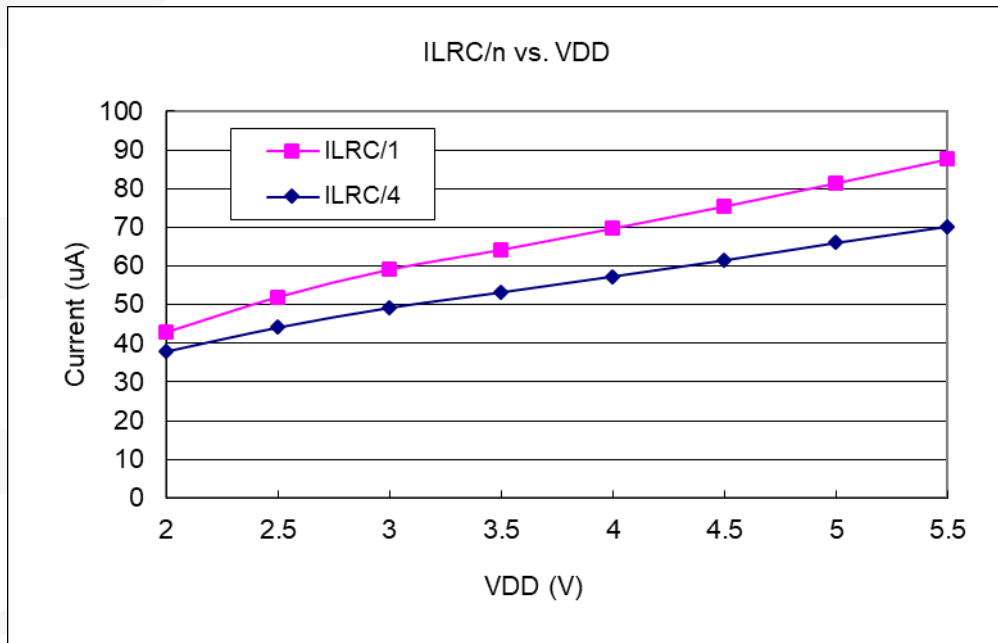
Conditions:

1-FPPA (code option)

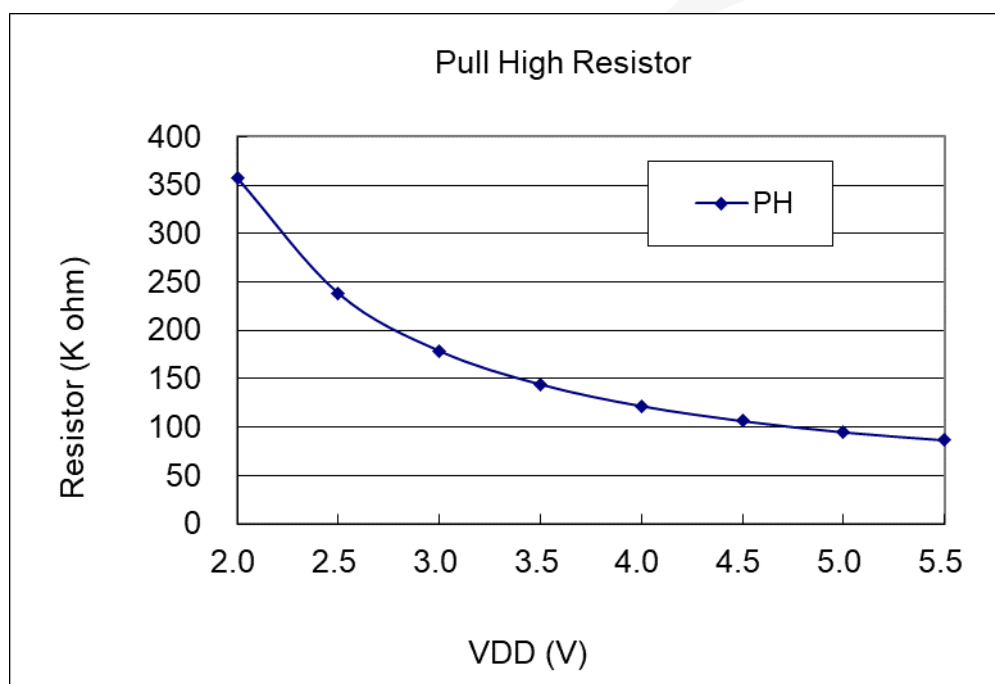
**ON:** ILRC;

**OFF:** Band-gap, LVR, IHRC, T16, TM2, ADC, PWM;

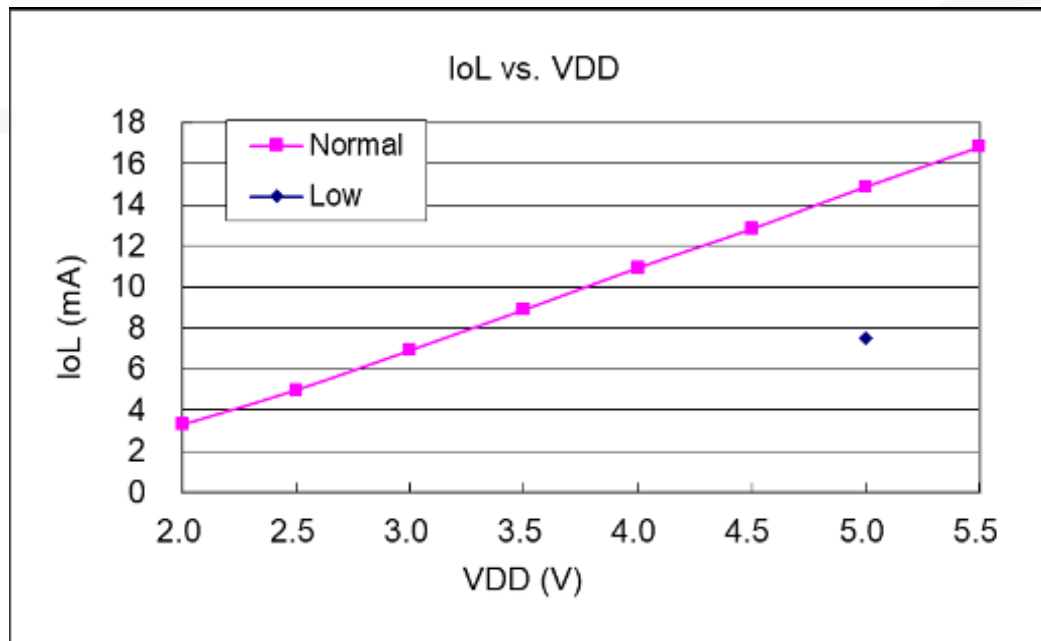
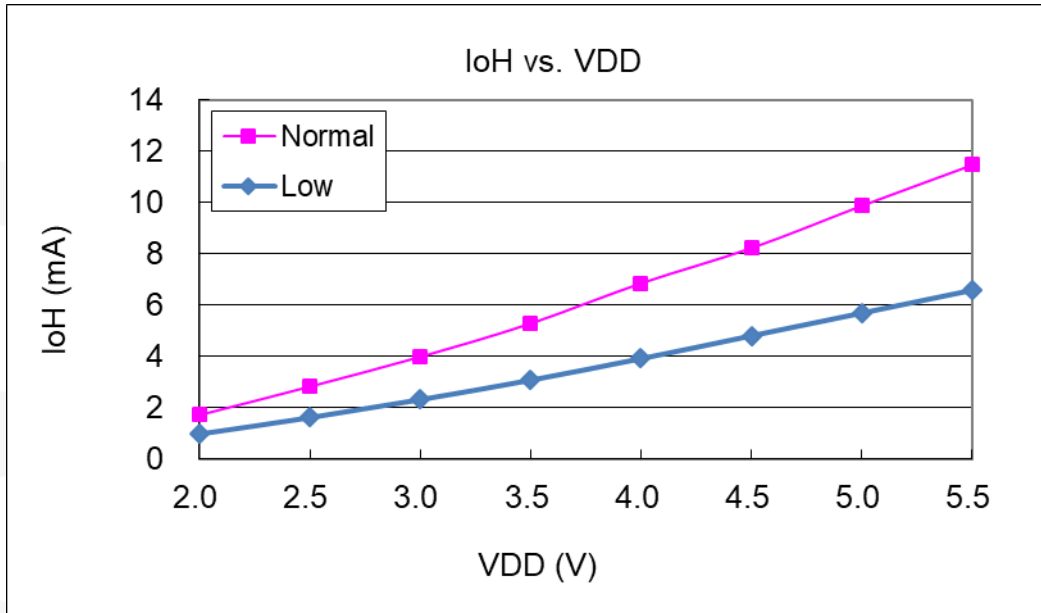
**IO:** PA0:0.5Hz output toggle and no loading, others: input and no floating



## 13.7. Typical IO pull high resistance

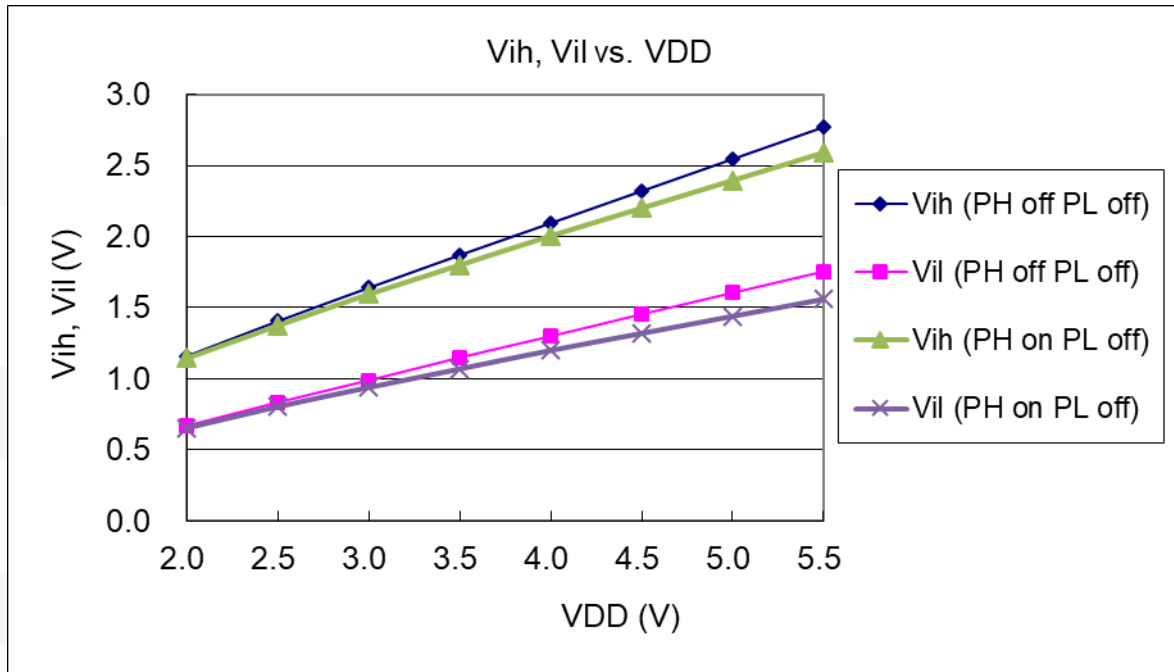


## 13.8. Typical IO driving current ( $I_{OH}$ ) and sink current ( $I_{OL}$ ) ( $V_{OH}=0.9 \cdot V_{DD}$ , $V_{OL}=0.1 \cdot V_{DD}$ )





## 13.9. Typical IO input high/low threshold voltage ( $V_{IH}/V_{IL}$ )



## 14. Instructions

Symbol	Description
<b>ACC</b>	Accumulator
<b>a</b>	Accumulator
<b>sp</b>	Stack pointer
<b>flag</b>	ACC status flag register
<b>I</b>	Immediate data
<b>&amp;</b>	Logical AND
<b> </b>	Logical OR
<b>←</b>	Movement
<b>^</b>	Exclusive logic OR
<b>+</b>	Add
<b>-</b>	Subtraction
<b>~</b>	NOT (logical complement, 1's complement)
<b>⌋</b>	NEG (2's complement)
<b>OV</b>	Overflow (The operational result is out of range in signed 2's complement number system)
<b>Z</b>	Zero (If the result of ALU operation is zero, this bit is set to 1)
<b>C</b>	Carry (The operational result is to have carry out for addition or to borrow carry for subtraction in unsigned number system)
<b>AC</b>	Auxiliary Carry (If there is a carry out from low nibble after the result of ALU operation, this bit is set to 1)
<b>pc0</b>	Program counter for FPPA0
<b>pc1</b>	Program counter for FPPA1
<b>pc2</b>	Program counter for FPPA2
<b>pc3</b>	Program counter for FPPA3
<b>pc4</b>	Program counter for FPPA4
<b>pc5</b>	Program counter for FPPA5
<b>pc6</b>	Program counter for FPPA6
<b>pc7</b>	Program counter for FPPA7

## 14.1. Instruction Table

Instructions	Function	Cycles	Z	C	AC	OV
<b>Data Transfer Instructions</b>						
<i>mov a, l</i>	<i>mov a, 0x0f; a</i> ← 0fh;	1	-	-	-	-
<i>mov M, a</i>	<i>mov MEM, a; MEM</i> ← a	1	-	-	-	-
<i>mov a, M</i>	<i>mov a, MEM; a</i> ← MEM; Flag Z is set when MEM is zero.	1	Y	-	-	-
<i>mov a, IO</i>	<i>mov a, pa; a</i> ← pa; Flag Z is set when pa is zero.	1	Y	-	-	-
<i>mov IO, a</i>	<i>mov pb, a; pb</i> ← a;	1	-	-	-	-
<i>nmov M, a</i>	<i>nmov MEM, a; MEM</i> ← $\overline{a}$	1	-	-	-	-
<i>nmov a, M</i>	<i>mov a, MEM; a</i> ← $\overline{MEM}$ ; Flag Z is set when $\overline{MEM}$ is zero.	1	-	-	-	-
<i>ldtabh index</i>	<i>ldtabh index; a</i> ← {bit 15~8 of MTP [index]};	2	-	-	-	-
<i>ldtabl index</i>	<i>ldtabl index; a</i> ← {bit7~0 of MTP [index]};	2	-	-	-	-
<i>ldt16 word</i>	<i>ldt16 word; word</i> ← 16-bit timer	1	-	-	-	-
<i>stt16 word</i>	<i>stt16 word; 16-bit timer</i> ← word	1	-	-	-	-
<i>idxm a, index</i>	<i>idxm a, index; a</i> ← [index], where index is declared by word.	2	-	-	-	-
<i>idxm index, a</i>	<i>idxm index, a; [index]</i> ← a; where index is declared by word.	2	-	-	-	-
<i>xch M</i>	<i>xch MEM; MEM</i> ← a, a ← MEM	1	-	-	-	-
<i>pushaf</i>	<i>pushaf; [sp]</i> ← {flag, ACC}; sp ← sp + 2;	1	-	-	-	-
<i>popaf</i>	<i>popaf; sp</i> ← sp - 2; {Flag, ACC} ← [sp];	1	Y	Y	Y	Y
<i>pushw word</i>	<i>pushw word; [sp]</i> ← word; sp ← sp + 2	2	-	-	-	-
<i>popw word</i>	<i>popw word; sp</i> ← sp - 2; word ← [sp];	2	-	-	-	-

Instructions	Function	Cycles	Z	C	AC	OV
<b>Arithmetic Operation Instructions</b>						
<i>add</i> a, I	<i>add</i> a, 0x0f ; $a \leftarrow a + 0fh$	1	Y	Y	Y	Y
<i>add</i> a, M	<i>add</i> a, MEM ; $a \leftarrow a + MEM$	1	Y	Y	Y	Y
<i>add</i> M, a	<i>add</i> MEM, a; $MEM \leftarrow a + MEM$	1	Y	Y	Y	Y
<i>addc</i> a, M	<i>addc</i> a, MEM ; $a \leftarrow a + MEM + C$	1	Y	Y	Y	Y
<i>addc</i> M, a	<i>addc</i> MEM, a ; $MEM \leftarrow a + MEM + C$	1	Y	Y	Y	Y
<i>addc</i> a	<i>addc</i> a ; $a \leftarrow a + C$	1	Y	Y	Y	Y
<i>addc</i> M	<i>addc</i> MEM ; $MEM \leftarrow MEM + C$	1	Y	Y	Y	Y
<i>nadd</i> a, M	<i>nadd</i> a, MEM ; $a \leftarrow \neg a + MEM$	1	Y	Y	Y	Y
<i>nadd</i> M, a	<i>nadd</i> MEM, a; $MEM \leftarrow \neg MEM + a$	1	Y	Y	Y	Y
<i>sub</i> a, I	<i>sub</i> a, 0x0f; $a \leftarrow a - 0fh$ ( $a + [2's \text{ complement of } 0fh]$ )	1	Y	Y	Y	Y
<i>sub</i> a, M	<i>sub</i> a, MEM ; $a \leftarrow a - MEM$ ( $a + [2's \text{ complement of } M]$ )	1	Y	Y	Y	Y
<i>sub</i> M, a	<i>sub</i> MEM, a; $MEM \leftarrow MEM - a$ ( $MEM + [2's \text{ complement of } a]$ )	1	Y	Y	Y	Y
<i>subc</i> a, M	<i>subc</i> MEM, a ; $a \leftarrow a - MEM - C$	1	Y	Y	Y	Y
<i>subc</i> M, a	<i>subc</i> MEM, a ; $MEM \leftarrow MEM - a - C$	1	Y	Y	Y	Y
<i>subc</i> a	<i>subc</i> a; $a \leftarrow a - C$	1	Y	Y	Y	Y
<i>subc</i> M	<i>subc</i> MEM; $MEM \leftarrow MEM - C$	1	Y	Y	Y	Y
<i>inc</i> M	<i>inc</i> MEM ; $MEM \leftarrow MEM + 1$	1	Y	Y	Y	Y
<i>dec</i> M	<i>dec</i> MEM; $MEM \leftarrow MEM - 1$	1	Y	Y	Y	Y
<i>clear</i> M	<i>clear</i> MEM ; $MEM \leftarrow 0$	1	-	-	-	-

Instructions	Function	Cycle	Z	C	AC	OV
<b>Shift Operation Instructions</b>						
<i>sr a</i>	<i>sr a</i> ; $a(0, b7, b6, b5, b4, b3, b2, b1) \leftarrow a(b7, b6, b5, b4, b3, b2, b1, b0)$ , $C \leftarrow a(b0)$	1	-	Y	-	-
<i>src a</i>	<i>src a</i> ; $a(c, b7, b6, b5, b4, b3, b2, b1) \leftarrow a(b7, b6, b5, b4, b3, b2, b1, b0)$ , $C \leftarrow a(b0)$	1	-	Y	-	-
<i>sr M</i>	<i>sr MEM</i> ; $MEM(0, b7, b6, b5, b4, b3, b2, b1) \leftarrow MEM(b7, b6, b5, b4, b3, b2, b1, b0)$ , $C \leftarrow MEM(b0)$	1	-	Y	-	-
<i>src M</i>	<i>src MEM</i> ; $MEM(c, b7, b6, b5, b4, b3, b2, b1) \leftarrow MEM(b7, b6, b5, b4, b3, b2, b1, b0)$ , $C \leftarrow MEM(b0)$	1	-	Y	-	-
<i>sl a</i>	<i>sl a</i> ; $a(b6, b5, b4, b3, b2, b1, b0, 0) \leftarrow a(b7, b6, b5, b4, b3, b2, b1, b0)$ , $C \leftarrow a(b7)$	1	-	Y	-	-
<i>slc a</i>	<i>slc a</i> ; $a(b6, b5, b4, b3, b2, b1, b0, c) \leftarrow a(b7, b6, b5, b4, b3, b2, b1, b0)$ , $C \leftarrow a(b7)$	1	-	Y	-	-
<i>sl M</i>	<i>sl MEM</i> ; $MEM(b6, b5, b4, b3, b2, b1, b0, 0) \leftarrow MEM(b7, b6, b5, b4, b3, b2, b1, b0)$ , $C \leftarrow MEM(b7)$	1	-	Y	-	-
<i>slc M</i>	<i>slc MEM</i> ; $MEM(b6, b5, b4, b3, b2, b1, b0, C) \leftarrow MEM(b7, b6, b5, b4, b3, b2, b1, b0)$ , $C \leftarrow MEM(b7)$	1	-	Y	-	-
<i>swap a</i>	<i>swap a</i> ; $a(b3, b2, b1, b0, b7, b6, b5, b4) \leftarrow a(b7, b6, b5, b4, b3, b2, b1, b0)$	1	-	-	-	-
<i>swap M</i>	<i>swap MEM</i> ; $MEM(b3, b2, b1, b0, b7, b6, b5, b4) \leftarrow MEM(b7, b6, b5, b4, b3, b2, b1, b0)$	1	-	-	-	-
<b>Logic Operation Instructions</b>						
<i>and a, l</i>	<i>and a, 0x0f</i> ; $a \leftarrow a \& 0fh$	1	Y	-	-	-
<i>and a, M</i>	<i>and a, RAM10</i> ; $a \leftarrow a \& RAM10$	1	Y	-	-	-
<i>and M, a</i>	<i>and MEM, a</i> ; $MEM \leftarrow a \& MEM$	1	Y	-	-	-
<i>or a, l</i>	<i>or a, 0x0f</i> ; $a \leftarrow a   0fh$	1	Y	-	-	-
<i>or a, M</i>	<i>or a, MEM</i> ; $a \leftarrow a   MEM$	1	Y	-	-	-
<i>or M, a</i>	<i>or MEM, a</i> ; $MEM \leftarrow a   MEM$	1	Y	-	-	-
<i>xor a, l</i>	<i>xor a, 0x0f</i> ; $a \leftarrow a \wedge 0fh$	1	Y	-	-	-
<i>xor IO, a</i>	<i>xor pa, a</i> ; $pa \leftarrow a \wedge pa$ ;	1	-	-	-	-
<i>xor a, M</i>	<i>xor a, MEM</i> ; $a \leftarrow a \wedge RAM10$	1	Y	-	-	-
<i>xor M, a</i>	<i>xor MEM, a</i> ; $MEM \leftarrow a \wedge MEM$	1	Y	-	-	-
<i>not a</i>	<i>not a</i> ; $a \leftarrow \sim a$	1	Y	-	-	-
<i>not M</i>	<i>not MEM</i> ; $MEM \leftarrow \sim MEM$	1	Y	-	-	-
<i>neg a</i>	<i>neg a</i> ; $a \leftarrow \overline{a}$	1	Y	-	-	-
<i>neg M</i>	<i>neg MEM</i> ; $MEM \leftarrow \overline{MEM}$	1	Y	-	-	-
<i>comp a, l</i>	<i>comp a, 0x55</i> ; Flag will be changed by regarding as ( $a - 0x55$ )	1	Y	Y	Y	Y
<i>comp a, M</i>	<i>comp a, MEM</i> ; Flag will be changed by regarding as ( $a - MEM$ )	1	Y	Y	Y	Y
<i>comp M, a</i>	<i>comp MEM, a</i> ; Flag will be changed by regarding as ( $MEM - a$ )	1	Y	Y	Y	Y

Instructions	Function	Cycles	Z	C	AC	OV
<b>Bit Operation Instructions</b>						
<i>set0</i> IO.n	<i>set0</i> pa.5 ; PA5=0	1	-	-	-	-
<i>set1</i> IO.n	<i>set1</i> pb.5 ; PB5=1	1	-	-	-	-
<i>set0</i> M.n	<i>set0</i> MEM.5 ; set bit 5 of MEM to low	1	-	-	-	-
<i>set1</i> M.n	<i>set1</i> MEM.5 ; set bit 5 of MEM to high	1	-	-	-	-
<i>swapc</i> IO.n	<i>swapc</i> IO.0; C $\leftarrow$ IO.0 , IO.0 $\leftarrow$ C When IO.0 is a port to output pin, carry C will be sent to IO.0; When IO.0 is a port from input pin, IO.0 will be sent to carry C;	1	-	Y	-	-
<i>tog</i> IO.n	<i>tog</i> pa.5 ; toggle bit 5 of port A	1	-	-	-	-
<b>Conditional Operation Instructions</b>						
<i>ceqsn</i> a, l	<i>ceqsn</i> a, 0x55 ; <i>inc</i> MEM ; <i>goto</i> error ; If a=0x55, then "goto error"; otherwise, "inc MEM".	1 / 2	Y	Y	Y	Y
<i>ceqsn</i> a, M	<i>ceqsn</i> a, MEM; If a=MEM, skip next instruction	1 / 2	Y	Y	Y	Y
<i>ceqsn</i> M, a	<i>ceqsn</i> MEM, a; If a=MEM, skip next instruction	1 / 2	Y	Y	Y	Y
<i>cneqsn</i> a, M	<i>cneqsn</i> a, MEM; If a $\neq$ MEM, skip next instruction	1 / 2	Y	Y	Y	Y
<i>cneqsn</i> M, a	<i>cneqsn</i> MEM,a; If a $\neq$ MEM, skip next instruction	1 / 2	Y	Y	Y	Y
<i>cneqsn</i> a, l	<i>cneqsn</i> a, 0x55 ; <i>inc</i> MEM ; <i>goto</i> error ; If a $\neq$ 0x55, then "goto error"; Otherwise, "inc MEM"	1 / 2	Y	Y	Y	Y
<i>t0sn</i> IO.n	<i>t0sn</i> pa.5; If bit 5 of port A is low, skip next instruction	1 / 2	-	-	-	-
<i>t1sn</i> IO.n	<i>t1sn</i> pa.5; If bit 5 of port A is high, skip next instruction	1 / 2	-	-	-	-
<i>t0sn</i> M.n	<i>t0sn</i> MEM.5 ; If bit 5 of MEM is low, then skip next instruction	1 / 2	-	-	-	-
<i>t1sn</i> M.n	<i>t1sn</i> MEM.5 ; If bit 5 of MEM is high, then skip next instruction	1 / 2	-	-	-	-
<i>izsn</i> a	<i>izsn</i> a; a $\leftarrow$ a + 1, skip next instruction if a = 0	1 / 2	Y	Y	Y	Y
<i>dzsn</i> a	<i>dzsn</i> a; a $\leftarrow$ a - 1, skip next instruction if a = 0	1 / 2	Y	Y	Y	Y
<i>izsn</i> M	<i>izsn</i> MEM; MEM $\leftarrow$ MEM + 1, skip next instruction if MEM= 0	1 / 2	Y	Y	Y	Y
<i>dzsn</i> M	<i>dzsn</i> MEM; MEM $\leftarrow$ MEM - 1, skip next instruction if MEM= 0	1 / 2	Y	Y	Y	Y
<i>wait0</i> IO.n	<i>wait0</i> pa.5; Wait here until bit n of IO port is low.	1	-	-	-	-
<i>wait1</i> IO.n	<i>wait1</i> pa.5; Wait here until bit n of IO port is high.	1	-	-	-	-

Instructions	Function	Cycles	Z	C	AC	OV
<b>System Control Instructions</b>						
<i>call</i> label	<i>call</i> function1; [sp] ← pc + 1, pc ← function1, sp ← sp + 2	2	-	-	-	-
<i>goto</i> label	<i>goto</i> routine1; Go to routine1 and execute program.	2	-	-	-	-
<i>delay</i> l	<i>delay</i> 0x05; Delay 6 cycles here	1	-	-	-	-
<i>delay</i> a	<i>delay</i> a; Delay 16 cycles here if ACC=0fh	1	-	-	-	-
<i>delay</i> M	<i>delay</i> M; Delay 256 cycles here if M=ffh	1	-	-	-	-
<b>Notes for <i>delay</i> instruction:</b> (1) Because ACC is the temporarily buffer for counting, please make sure that it will not be interrupted when executing <i>delay</i> instruction. Otherwise, it may be not the expected delay time. (2) <u>This instruction is not supported in single FPPA mode.</u>						
<i>ret</i> l	<i>ret</i> 0x55; A ← 55h <i>ret</i> ;	2	-	-	-	-
<i>ret</i>	<i>ret</i> ; sp ← sp - 2 pc ← [sp]	2	-	-	-	-
<i>reti</i>	<i>reti</i> ; Return to program from interrupt service routine. After this command is executed, global interrupt is enabled automatically.	2	-	-	-	-
<i>nop</i>	<i>nop</i> ; Nothing changed.	1	-	-	-	-
<i>pcadd</i> a	<i>pcadd</i> a; pc ← pc + a	2	-	-	-	-
<i>engint</i>	<i>engint</i> ; Interrupt request can be sent to FPPA0	1	-	-	-	-
<i>disgint</i>	<i>disgint</i> ; Interrupt request is blocked from FPPA0	1	-	-	-	-
<i>stopsys</i>	<i>stopsys</i> ; Stop the system clocks and halt the system	1	-	-	-	-
<i>stopexe</i>	<i>stopexe</i> ; Stop the system clocks and keep oscillator modules active.	1	-	-	-	-
<i>reset</i>	<i>reset</i> ; Reset the whole chip.	1	-	-	-	-
<i>wdreset</i>	<i>wdreset</i> ; Reset Watchdog timer.	1	-	-	-	-
<i>pmode</i> n	Operational mode selection for each FPPA unit <i>pmode</i> 0; FPPA units bandwidth sharing is set to mode 0	1	-	-	-	-

Instructions	Function	Cycles	Z	C	AC	OV
<b>System Control Instructions</b>						
<i>pmode</i> n	Operational mode selection for each FPPA unit <i>pmode</i> 0; FPPA units bandwidth sharing is set to mode 0 Mode FPPA0 ~ FPPA7 bandwidth sharing 0: /2, /2 1: /2, /4, /4 2: /4, /2, /4 3: /2, /4, /8, /8 4: /4, /2, /8, /8 5: /8, /2, /4, /8 6: /4, /4, /4, /4 7: /8, /4, /4, /4, /8 8: /2, /8, /8, /8, /8 9: /4, /4, /4, /8, /8 10: /8, /2, /8, /8, /8 11: /2, /8, /8, /8, /16, /16 12: /16, /2, /8, /8, /8, /16 13: /4, /4, /8, /8, /8, /8 14: /8, /4, /4, /8, /8, /8 15: /4, /4, /4, /8, /16, /16 16: /8, /4, /4, /4, /16, /16 17: /16, /4, /4, /4, /8, /16 18: /2, /8, /8, /16, /16, /16, /16 19: /8, /2, /8, /16, /16, /16, /16 20: /16, /2, /8, /8, /16, /16, /16 21: /4, /4, /4, /16, /16, /16, /16 22: /16, /4, /4, /4, /16, /16, /16 23: /4, /8, /8, /8, /8, /8, /8 24: /8, /2, /16, /16, /16, /16, /16, /16 25: /4, /8, /4, /8, /16, /16, /16, /16 26: /8, /4, /4, /8, /16, /16, /16, /16 27: /2, /8, /16, /16, /16, /16, /16, /16 28: /4, /4, /8, /8, /16, /16, /16, /16 29: /16, /2, /8, /16, /16, /16, /16, /16 30: /8, /4, /4, /8, /16, /16, /16, /16 31: /8, /8, /8, /8, /8, /8, /8, /8	1	-	-	-	-