PADAUK 應廣科技

LeapDragon（跃龙）

PFC154

Industrial Grade - 8bit MTP Type IO Controller

Data Sheet

Version 0.01

October 8, 2019

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

TEL: 886-3-572-8688 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.

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

Revision History:

Revision	Date	Description
0.00	2019/08/23	Preliminary version
0.01	2019/10/08	Add SOT23-6, SOP14 and DIP14

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/tw/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 (±4KV) for passing safety regulation tests.
- Operating temperature range: -40°C ~ 85°C

1.2. System Features

- 2KW MTP program memory (programming cycle at least 1,000 times)
- 128 Bytes data RAM
- Clock sources: IHRC, ILRC & EOSC(XTAL mode)
- 14 IO pins with optional pull-high resistor
- Every IO pin can be configured to enable wake-up function
- Two external interrupt pins
- 8 selectable levels of LVR reset from 1.8V to 4.5V
- One hardware 16-bit timer
- Two hardware 8-bit timer with PWM generators
- Three hardware 11-bit PWM generators
- Provide one hardware comparator
- Built-in VDD/2 bias voltage generator to provide maximum 4 x 10 dots LCD display

1.3. CPU Features

- One processing unit operating mode
- 86 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
- Register space, memory space and MTP space are independent

1.4. Package Information

- PFC154-U06: SOT23-6 (60mil)
- PFC154-S08: SOP8 (150mil)
- PFC154-D08: DIP8 (300mil)
- PFC154-S14: SOP14 (150mil)
- PFC154-D14: DIP14 (300mil)
- PFC154-S16: SOP16 (150mil)
- PFC154-D16: DIP16 (300mil)

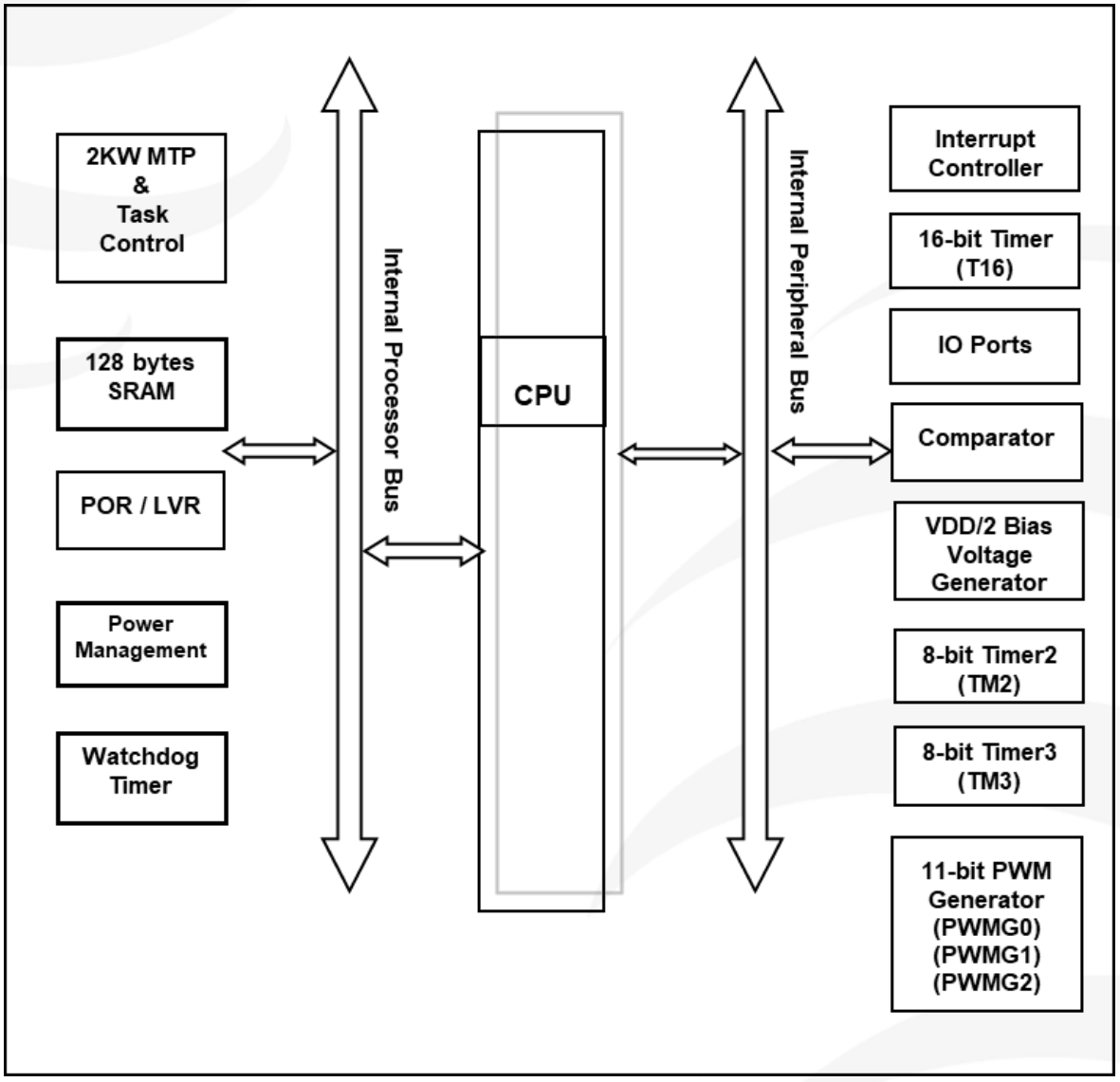
2. General Description and Block Diagram

The PFC154 is an IO-Type, fully static, MTP-based controller; it employs RISC architecture and most the instructions are executed in one cycle except that few instructions are two cycles that handle indirect memory access.

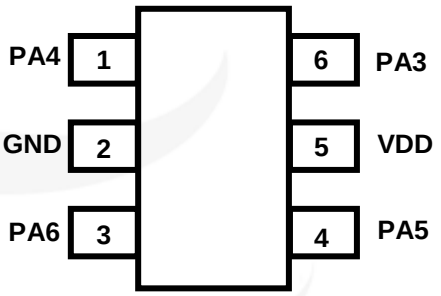
PFC154 has built-in 2KW MTP program memory and 128 byte data storage.

PFC154 provides a hardware 16-bit timer, two hardware 8-bit timers with PWM generation (Timer2, Timer3) and Three hardware 11-bit timers with PWM generation (PWMG0, PWMG1, PWMG2).

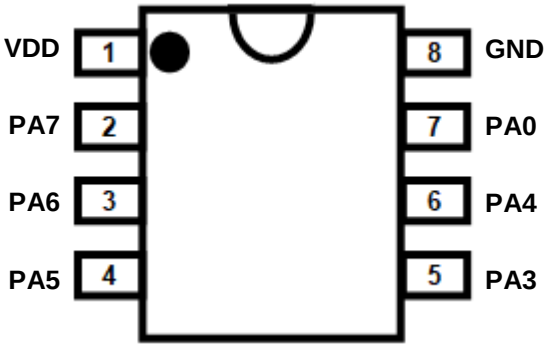
PFC154 also supports one hardware comparator and VDD/2 bias voltage generator for LCD display application.



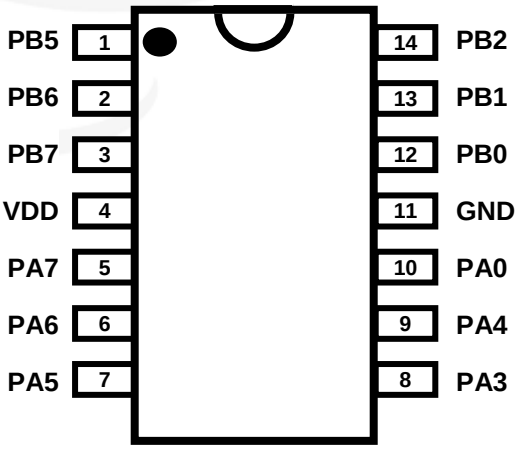
3. Pin Definition and Functional Description



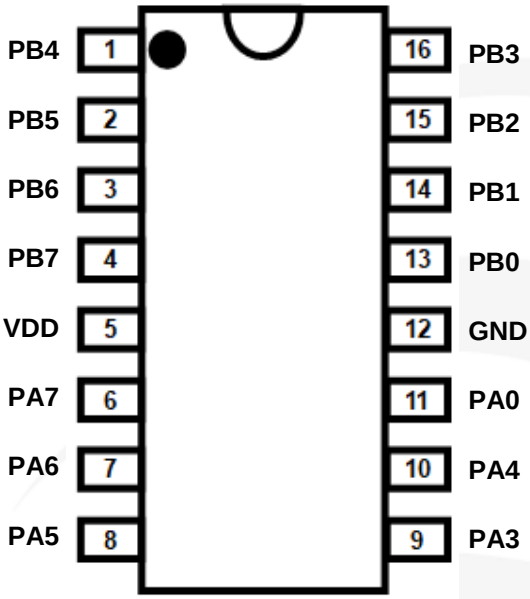
PFC154-U06 (SOT23-6 60mil)



PFC154-S08: SOP8 (150mil)
PFC154-D08: DIP8 (300mil)



PFC154-S14: SOP14 (150mil)
PFC154-D14: DIP14 (300mil)



PFC154-S16: SOP16 (150mil)
PFC154-D16: DIP16 (300mil)

Pin Name	Input / output			Special Functions						
	I / O	Pull High	Wake Up	Crystal	Comparator	PWM	VDD/2	External Interrupt	External Reset	Program
PA0	√	√	√		CO	PG0PWM	COM2	INT0		
PA3	√	√	√		CIN-	TM2PWM PG2PWM	COM4			√
PA4	√	√	√		CIN+ CIN-	PG1PWM	COM3			
PA5	√	√	√			PG2PWM			√	√
PA6	√	√	√	√						√
PA7	√	√	√	√						
PB0	√	√	√				COM1	INT1		
PB1	√	√	√							
PB2	√	√	√			TM2PWM PG2PWM				
PB3	√	√	√			PG2PWM				
PB4	√	√	√			TM2PWM PG0PWM				
PB5	√	√	√			TM3PWM PG0PWM				
PB6	√	√	√		CIN-	TM3PWM PG1PWM				
PB7	√	√	√		CIN-	TM3PWM PG1PWM				
VDD										√
GND										√
Notice	1. All the I/O pins have: Schmitt Trigger input and CMOS voltage level 2. IO function is automatically deactivated when a pin is used as PWM output port. 3. Please put 33Ω resistor in series to have high noise immunity when PA5 is in input mode. 4. ICE doesn't support the function PG2PWM output to PA5.									

4. Central Processing Unit (CPU)

4.1. Storage Memory

4.1.1. Program Memory – ROM

The PFC154 program memory is MTP (Multiple Time Programmable), used to store data (including: data, tables and interrupt entry) and program instructions to be executed. The MTP program memory for PFC154 is 2KW that is partitioned as Table 1.

After reset, the program will start from the initial address 0x000 which is *goto* FPPA0 instruction usually. And the interrupt entry is 0x010 if used.

The MTP memory from address 0x7E0 to 0x7FF are for system using, address space from 0x001 to 0x00F and from 0x011 to 0x7DF are user program spaces.

The last 32 addresses are reserved for system using, like checksum, serial number, etc.

Address	Function
0x000	*goto* FPPA0 instruction
0x001	User program
•	•
0x00F	User program
0x010	Interrupt entry address
0x011	User program
•	•
0x7DF	User program
0x7E0	System Using
•	•
0x7FF	System Using

Table 1: Program Memory Organization

4.1.2. Data Memory – SRAM

PFC154 data memory has a total of 128 bytes. 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.

4.1.3. System Register

The register space of PFC154 is independent of SRAM space and MTP space.

The following is the PFC154 register address and brief description:

	+0	+1	+2	+3	+4	+5	+6	+7
0x00	*FLAG*	-	*SP*	*CLKMD*	*INTEN*	*INTRQ*	*T16M*	-
0x08	*MISC*	*TM2B*	*EOSCR*	*IHRCR*	*INTEGS*	*PADIER*	*PBDIER*	-
0x10	*PA*	*PAC*	*PAPH*	-	*PB*	*PBC*	*PBPH*	*TM2S*
0x18	*GPCC*	*GPCS*	-	-	*TM2C*	*TM2CT*	-	-
0x20	*PWMG0C*	*PWMG0S*	*PWMG0 DTH*	*PWMG0 DTL*	*PWMG0 CUBH*	*PWMG0 CUBL*	*PWMG1C*	*PWMG1S*
0x28	*PWMG1 DTH*	*PWMG1 DTL*	*PWMG1 CUBH*	*PWMG1 CUBL*	*PWMG2C*	*PWMG2S*	*PWMG2 DTH*	*PWMG2 DTL*
0x30	*PWMG2 CUBH*	*PWMG2 CUBL*	*TM3C*	*TM3CT*	*TM3S*	*TM3B*	-	-

FLAG: ACC Status Flag Register
SP: Stack Pointer Register
CLKMD: Clock Mode Register
EOSCR: External Oscillator setting Register
INTEN: Interrupt Enable Register
INTRQ: Interrupt Request Register
INTEGS: Interrupt Edge Select Register
MISC: MISC Register
PA: Port A Data Registers
PAC: Port A Control Registers
PAPH: Port A Pull-High Registers
PADIER: Port A Digital Input Enable Register
PB: Port B Data Registers
PBC: Port B Control Registers
PBPH: Port B Pull-High Registers
PBDIER: Port B Digital Input Enable Register
GPCC: Comparator Control Register
GPCS: Comparator Selection Register

T16M: Timer 16 mode Register
TM2C / *TM3C*: Timer2 / Timer3 Control Register
TM2CT / *TM3CT*: Timer2 / Timer3 Counter Register
TM2S / *TM3S*: Timer2 / Timer3 Scalar Register
TM2B / *TM3B*: Timer2 / Timer3 Bound Register
PWMG0C / *PWMG1C* / *PWMG2C*:
PWMG0 / PWMG1 / PWMG2 control Register
PWMG0S / *PWMG1S* / *PWMG2S*:
PWMG0 / PWMG1 / PWMG2 Scalar Register
PWMG0DTH / *PWMG1DTH* / *PWMG2DTH*:
PWMG0 / PWMG1 / PWMG2 Duty Value High Register
PWMG0DTL / *PWMG1DTL* / *PWMG2DTL*:
PWMG0 / PWMG1 / PWMG2 Duty Value Low Register
PWMG0CUBH / *PWMG1CUBH* / *PWMG2CUBH*:
PWMG0 / PWMG1 / PWMG2 Counter Upper Bound High Register
PWMG0CUBL / *PWMG1CUBL* / *PWMG2CUBL*:
PWMG0 / PWMG1 / PWMG2 Counter Upper Bound Low Register

4.1.3.1. 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.

4.1.3.2. MISC Register (*MISC*), address = 0x08

Bit	Reset	R/W	Description
7 – 5	-	-	Reserved. (keep 0 for future compatibility)
4	0	WO	Enable VDD/2 bias voltage generator 0 / 1 : Disable / Enable (ICE cannot be dynamically switched)
3	-	-	Reserved.
2	0	WO	Disable LVR function. 0 / 1 : Enable / Disable
1 – 0	00	WO	Watch dog time out period 00: 8k ILRC clock period 01: 16k ILRC clock period 10: 64k ILRC clock period 11: 256k ILRC clock period

4.2. Addressing Mode

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 128 bytes data memory of PFC154 can be accessed by indirect access mechanism.

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

4.3. The Stack

The stack memory is defined in the data memory. The stack pointer is defined in the stack pointer register; 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.

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. Please notice that bit 0 should be kept 0 due to program counter is 16 bits.

4.4. Code Options

Option	Selection	Description
Security	Enable	MTP content is protected 7/8 words
	Disable (default)	MTP content is not protected so program can be read back
LVR	4.0V	Select LVR = 4.0V
	3.5V	Select LVR = 3.5V
	3.0V	Select LVR = 3.0V
	2.75V	Select LVR = 2.75V
	2.5V(default)	Select LVR = 2.5V
	2.2V	Select LVR = 2.2V
	2.0V	Select LVR = 2.0V
	1.8V	Select LVR = 1.8V
Drive	Low	IO Low driving and sinking current
	Normal (default)	IO Normal driving and sinking current
Comparator_Edge	**All_Edge (default)**	The comparator will trigger an interrupt on the rising edge or falling edge
	Rising_Edge	The comparator will trigger an interrupt on the rising edge
	Falling_Edge	The comparator will trigger an interrupt on the falling edge

Option	Selection	Description
GPC_PWM	**Disable (default)**	Comparator does not control all PWM outputs
	Enable	Comparator controls all PWM outputs (ICE does NOT Support.)
PWM_Source	**16MHz (default)**	When *PWMG0C*.0= 1, PWMG0 clock source = IHRC = 16MHz When *PWMG1C*.0= 1, PWMG1 clock source = IHRC = 16MHz When *PWMG2C*.0= 1, PWMG2 clock source = IHRC = 16MHz
	32MHz	When *PWMG0C*.0= 1, PWMG0 clock source = IHRC*2 = 32MHz When *PWMG1C*.0= 1, PWMG1 clock source = IHRC*2 = 32MHz When *PWMG2C*.0= 1, PWMG2 clock source = IHRC*2 = 32MHz (ICE does NOT Support.)
TMx_Source	**16MHz (default)**	When *TM2C*[7:4]= 0010, TM2 clock source = IHRC = 16MHz When *TM3C*[7:4]= 0010, TM3 clock source = IHRC = 16MHz
	32MHz	When *TM2C*[7:4]= 0010, TM2 clock source = IHRC*2 = 32MHz When *TM3C*[7:4]= 0010, TM3 clock source = IHRC*2 = 32MHz (ICE does NOT Support.)
TMx_Bit	**6 Bit (default)**	When *TM2S*.7=1, TM2 PWM resolution is 6 Bit When *TM3S*.7=1, TM3 PWM resolution is 6 Bit
	7 Bit	When *TM2S*.7=1, TM2 PWM resolution is 7 Bit When *TM3S*.7=1, TM3 PWM resolution is 7 Bit (ICE does NOT Support.)
EMIEnB	Disable	Disable EMI optimize option
	Enable (default)	The system clock will be slightly vibrated for better EMI performance

5. Oscillator and System Clock

There are three oscillator circuits provided by PFC154: external crystal oscillator (EOSC), internal high RC oscillator (IHRC) and internal low RC oscillator (ILRC).

These three oscillators are enabled or disabled by registers *EOSCR*.7, *CLKMD*.4 and *CLKMD*.2 independently. User can choose one of these three 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
EOSC	*EOSCR*.7
IHRC	*CLKMD*.4
ILRC	*CLKMD*.2

Table2: Three Oscillator Circuits provided by PFC154

5.1. Internal High RC Oscillator and Internal Low RC Oscillator

The frequency of IHRC / ILRC will 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 PFC154 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. External Crystal Oscillator

The range of operating frequency of crystal oscillator can be from 32 KHz to 4MHz, depending on the crystal placed on; higher frequency oscillator than 4MHz is NOT supported. Fig. 1 shows the hardware connection under this application.

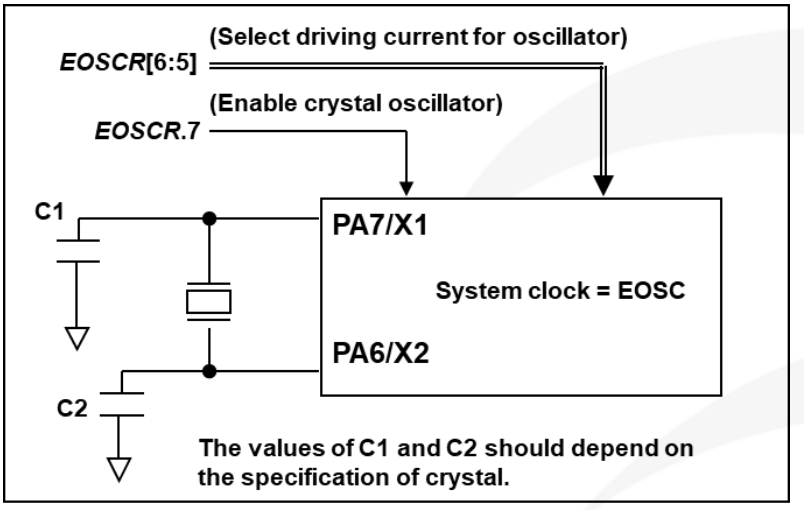


Fig. 1: Connection of crystal oscillator

5.2.1. External Oscillator Setting Register (*EOSCR*), address = 0x0A

Bit	Reset	R/W	Description
7	0	WO	Enable external crystal oscillator. 0 / 1 : Disable / Enable
6 – 5	00	WO	External crystal oscillator selection. 00 : reserved 01 : Low driving capability, for lower frequency, ex: 32KHz crystal oscillator 10 : Middle driving capability, for middle frequency, ex: 1MHz crystal oscillator 11 : High driving capability, for higher frequency, ex: 4MHz crystal oscillator
4 – 1	-	-	Reserved. Please keep 0 for future compatibility.
0	0	WO	Power-Down the Bandgap and LVR hardware modules. 0 / 1: normal / Power-Down.

5.2.2. Usages and Precautions of External Oscillator

Besides crystal, external capacitor and options of PFC154 should be fine tuned in *EOSCR* register to have good sinusoidal waveform. The *EOSCR*.7 is used to enable crystal oscillator module. *EOSCR*.6 and *EOSCR*.5 are used to set the different driving current to meet the requirement of different frequency of crystal oscillator.

Table 3 shows the recommended values of C1 and C2 for different crystal oscillator; the measured start-up time under its corresponding conditions is also shown. Since the crystal or resonator had its own characteristic, the capacitors and start-up time may be slightly different for different type of crystal or resonator, please refer to its specification for proper values of C1 and C2.

Frequency	C1	C2	Measured Start-up time	Conditions
4MHz	4.7pF	4.7pF	6ms	(*EOSCR*[6:5]=11)
1MHz	10pF	10pF	11ms	(*EOSCR*[6:5]=10)
32KHz	22pF	22pF	450ms	(*EOSCR*[6:5]=01)

Table 3: Recommend values of C1 and C2 for crystal and resonator oscillators

Configuration of PA7 and PA6 when using crystal oscillator:
(1) PA7 and PA6 are set as input;
(2) PA7 and PA6 internal pull-high resistors are set to close;
(3) Set PA6 and PA7 as analog inputs with the *PADIER* register to prevent power leakage.

Note: Please read the PMC-APN013 carefully. According to PMC-APN013, the crystal oscillator should be used reasonably. If the following situations happen to cause IC start-up slowly or non-startup, PADAUK Technology is not responsible for this: the quality of the user's crystal oscillator is not good, the usage conditions are unreasonable, the PCB cleaner leakage current, or the PCB layouts are unreasonable.

When using the crystal oscillator, user must pay attention to the stable time of oscillator after enabling it. The stable time of oscillator will depend on frequency, crystal type, external capacitor and supply voltage. Before switching the system to the crystal oscillator, user must make sure the oscillator is stable. The reference program is shown as below:

```
void    FPPA0 (void)
{
    . ADJUST_IC   SYSCLK=IHRC/16, IHRC=16MHz, VDD=5V
    ...
    $ EOSCR  Enable, 4Mhz;          // EOSCR = 0b110_00000;
    $ T16M   EOSC，  /1，  BIT13;   // while T16.Bit13 0 => 1, Intrq.T16 => 1
                                    // suppose crystal eosc. is stable
    WORD     count    =    0;
    stt16    count;
    Intrq.T16  =    0;
    while (! Intrq.T16)  NULL;      // count from 0x0000 to 0x2000, then trigger INTRQ.T16
    CLKMD = 0xB4;                   // switch system clock to EOSC;
    CLKMD.4 = 0;                    // disable IHRC
    ...
}
```

Please notice that the crystal oscillator should be fully turned off before entering the Power-Down mode, in order to avoid unexpected wake-up event.

5.3. System Clock and IHRC Calibration

5.3.1. System Clock

The clock source of system clock comes from IHRC, ILRC or EOSC, the hardware diagram of system clock in the PFC154 is shown as Fig. 2.

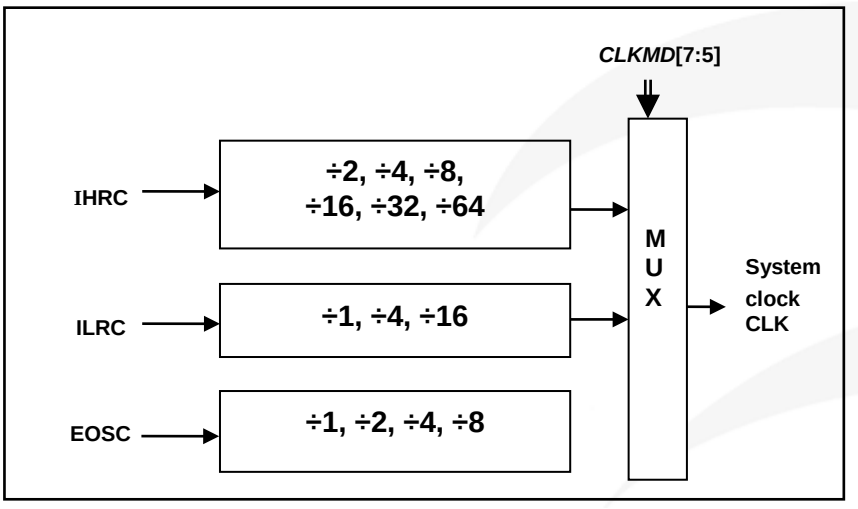


Fig. 2: Options of System Clock

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

Bit	Reset	R/W	Description	
7 – 5	111	R/W	System clock selection	
			Type 0, *CLKMD*[3]=0	Type 1, *CLKMD*[3]=1
			000: IHRC/4 001: IHRC/2 010: reserved 011: EOSC/4 100: EOSC/2 101: EOSC 110: ILRC/4 111: ILRC (default)	000: IHRC/16 001: IHRC/8 010: ILRC/16 (ICE does NOT Support.) 011: IHRC/32 100: IHRC/64 101: EOSC/8 Others: reserved
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	

5.3.2. Frequency Calibration

The IHRC frequency calibration 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_{DD}=(**p3**)V

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=2.2 ~ 5.5; In order to calibrate the chip under different supply voltage.

Usually, .ADJUST_IC will be the first command after boot up, in order to set the target operating frequency whenever stating 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, Bandgap OFF

Table 4: Options for IHRC Frequency Calibration

The following shows the different states of PFC154 under different options:

(1) .ADJUST_IC SYSCLK=IHRC/4, IHRC=16MHz, V_{DD}=5V

After boot, *CLKMD* = 0x14:

a. IHRC frequency is calibrated to 16MHz@V_{DD}=5V and IHRC module is enabled
b. System CLK = IHRC/4 = 4MHz
c. Watchdog timer is disabled, ILRC is enabled, PA5 is in input mode

(2) .ADJUST_IC SYSCLK=IHRC/8, IHRC=16MHz, V_{DD}=2.5V

After boot, *CLKMD* = 0x3C:

a. IHRC frequency is calibrated to 16MHz@V_{DD}=2.5V and IHRC module is enabled
b. System CLK = IHRC/8 = 2MHz
c. Watchdog timer is disabled, ILRC is enabled, PA5 is in input mode

(3) .ADJUST_IC SYSCLK=ILRC, IHRC=16MHz, V_{DD}=5V

After boot, *CLKMD* = 0xE4:

a. IHRC frequency is calibrated to 16MHz@V_{DD}=5V and IHRC module is disabled
b. System CLK = ILRC
c. Watchdog timer is disabled, ILRC is enabled, PA5 is in input mode

(4) .ADJUST_IC DISABLE

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

a. IHRC is not calibrated.
b. System CLK = ILRC or IHRC/64
c. Watchdog timer is enabled, ILRC is enabled, PA5 is in input mode

5.3.2.1. 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.

5.3.3. System Clock Switching

After IHRC calibration, the system clock of PFC154 can be switched among IHRC, ILRC and EOSC 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       =   0x14;           //  switch to IHRC/4, ILRC CAN NOT be disabled here
CLKMD.2     =   0;              //  ILRC CAN be disabled at this time
...
```

Case 2: Switching system clock from IHRC/4 to EOSC

```
...                             //  system clock is IHRC/4
CLKMD       =   0xB0;           //  switch to EOSC, IHRC CAN NOT be disabled here
CLKMD.4     =   0;              //  IHRC CAN be disabled at this time
...
```

Case 3: 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 4: 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

PFC154 reset can be caused by four factors: 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 and all registers of PFC154 will be set to the default value.

6.1. Power On Reset - POR

POR (Power-On-Reset) is used to reset PFC154 when power up. The power up sequence is shown in the Fig. 3. Customer must ensure the stability of supply voltage after power up no matter which option is chosen.

The PFC154 data memory is in an uncertain state when the power on reset occurs.

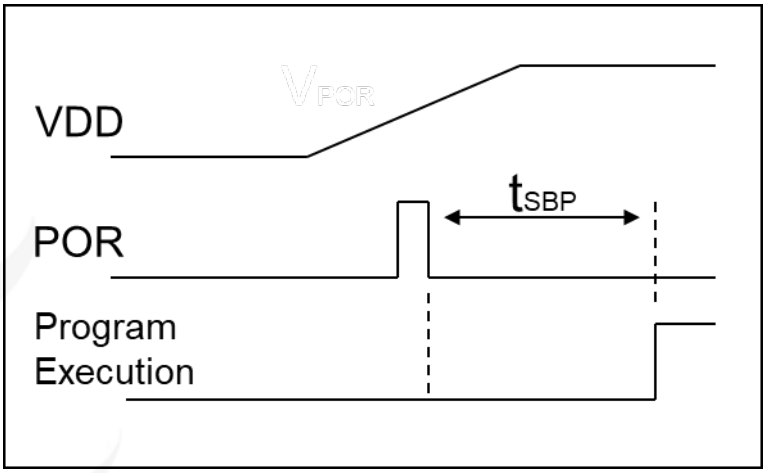


Fig. 3: Power Up Sequence

6.2. Low Voltage Reset - LVR

If VDD drops below the Voltage level of LVR(Low Voltage Reset), LVR Reset will occur in the system. The LVR reset timing diagram is shown in figure 4.

The PFC154 data memory is in an uncertain state when the LVR reset occurs.

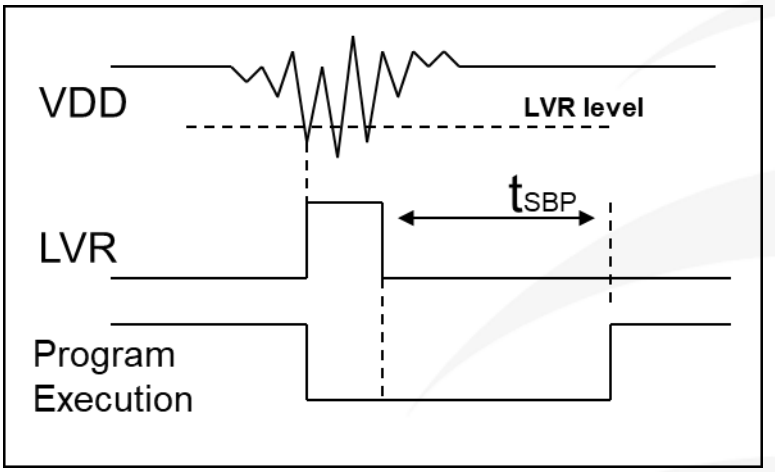


Fig. 4: Low Voltage Reset Sequence

LVR level selection is done at compile time. User must select LVR based on the system working frequency and power supply voltage to make the MCU work stably.

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

SYSCLK	VDD	LVR
8MHz	≧ 3.5V	3.5V
4MHz	≧ 2.5V	2.5V
2MHz	≧ 2.2V	2.2V

Table 5: LVR setting for reference

Users can set *EOSCR*.0 as 1 to power off the LVR module, or *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.

External Oscillator setting Register (*EOSCR*), address = 0x0A			
Bit	**Reset**	**R/W**	**Description**
7	0	WO	Enable external crystal oscillator. 0 / 1 : Disable / Enable
6 – 5	00	WO	External crystal oscillator selection.
4 – 1	-	-	Reserved. Please keep 0 for future compatibility.
0	**0**	**WO**	**Power-Down the Bandgap and LVR hardware modules. 0 / 1: normal / Power-Down.**

MISC Register (*MISC*), address = 0x08			
Bit	**Reset**	**R/W**	**Description**
7 – 5	-	-	Reserved. (keep 0 for future compatibility)
4	0	WO	Enable VDD/2 bias voltage generator 0 / 1 : Disable / Enable
3	-	-	Reserved.
2	**0**	**WO**	**Disable LVR function. 0 / 1 : Enable / Disable**
1 – 0	**00**	**WO**	**Watch dog time out period** **00: 8k ILRC clock period** **01: 16k ILRC clock period** **10: 64k ILRC clock period** **11: 256k ILRC clock period**

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.

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, PFC154 will be reset to restart the program execution. The relative timing diagram of watchdog timer is shown as Fig. 5.

The PFC154 data memory will be reserved when the WDT reset occurs.

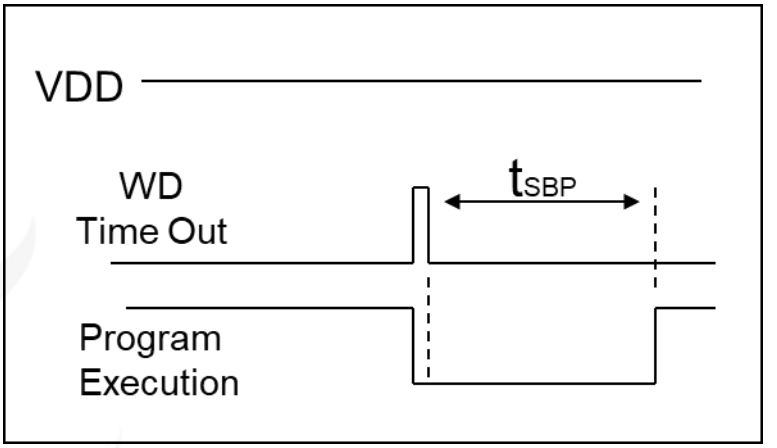


Fig. 5: Sequence of Watch Dog Timeout reset

There are four 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 (*CLKMD*), 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 PFC154 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 figure 6.

The PFC154 data memory will be reserved when the PRSTB reset occurs.

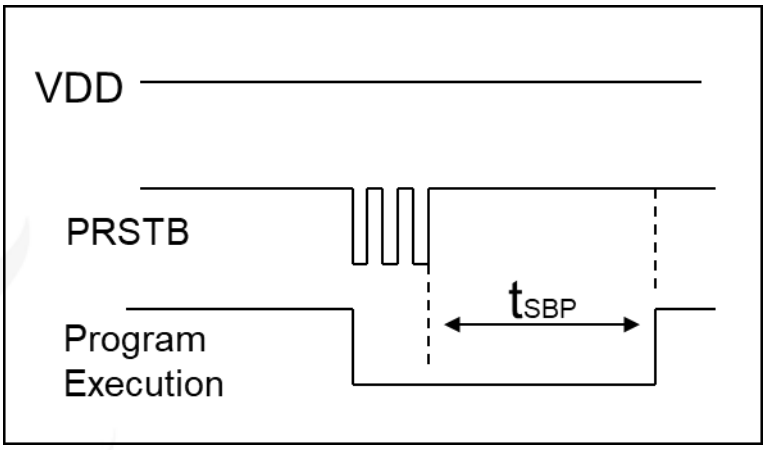


Fig. 6: Sequence of PRSTB reset

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.

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 contains TM16, TM2, TM3, PWMG0, PWMG1, PWMG2.)
(6) Wake-up sources:
    a. IO toggle wake-up: IO toggling in digital input mode (*PxC* bit is 1 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 and *GPCS*.6=1 to enable the comparator wake-up function at the same time.

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. It is recommend to set *GPCC*.7=0 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 PFC154 in detail when *stopsys* command is issued:

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

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

```
CMKMD   =   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 PFC154 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 *stopsys* and *stopexe*		
	IO Toggle	Timer wake-up
stopsys	Yes	No
stopexe	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 PFC154, registers *PxDIER* should be properly set to enable the wake-up function for every corresponding pin.

8. Interrupt

The hardware diagram of interrupt controller is shown as Fig. 7. There are total 7 interrupt sources for PFC154: PA0, PB0, Timer16, Timer2, Timer3, PWMG0, and comparator. Among them, every interrupt request line to CPU has its own corresponding interrupt control bit to enable or disable it. 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.

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, the stack position and level are arranged by the compiler in Mini-C project. When defining the stack level in ASM project, users should arrange their locations carefully to prevent address conflicts.

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

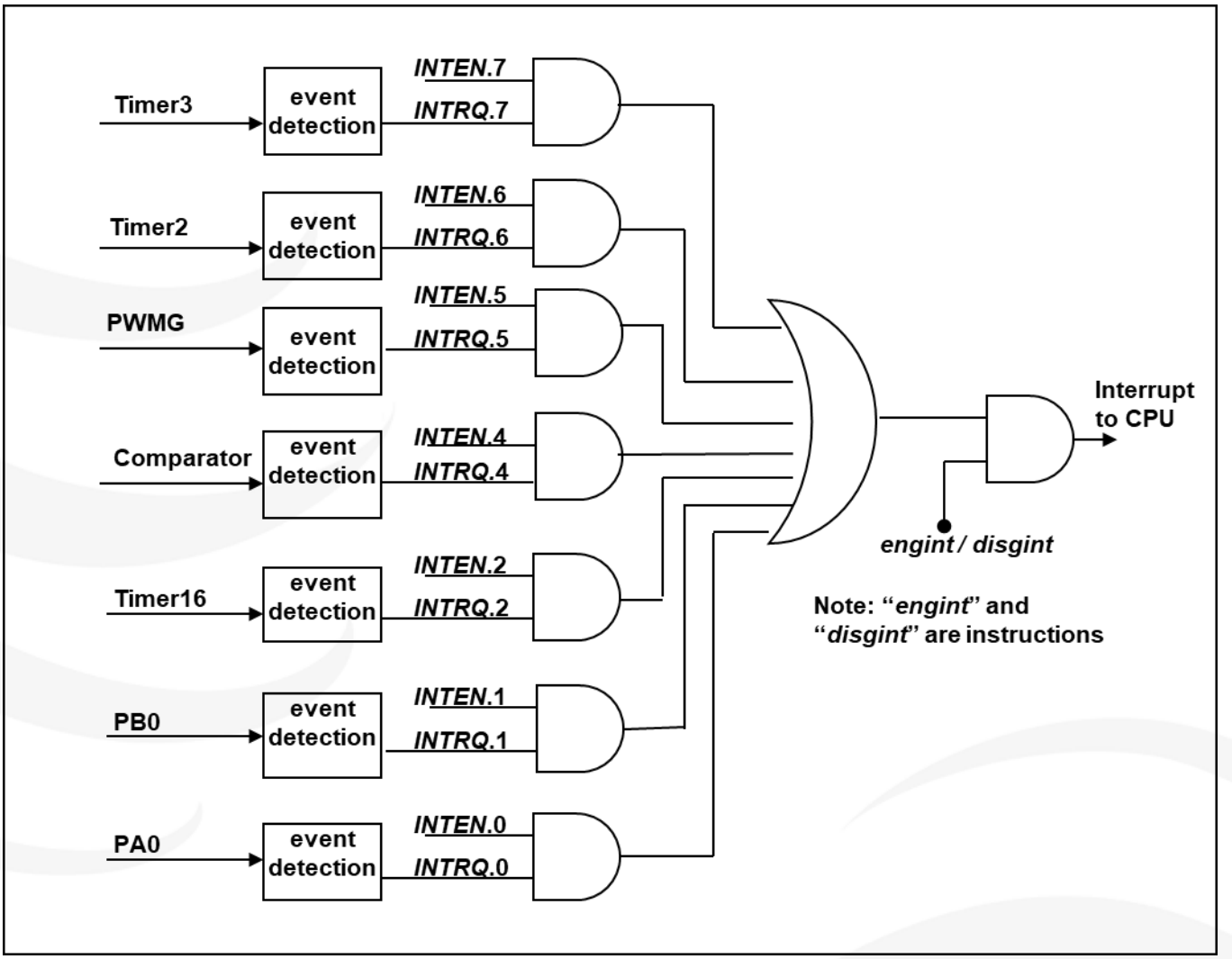


Fig. 7: Hardware diagram of Interrupt controller

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

Bit	Reset	R/W	Description
7	-	R/W	Enable interrupt from Timer3. 0 / 1: disable / enable.
6	-	R/W	Enable interrupt from Timer2. 0 / 1: disable / enable.
5	-	R/W	Enable interrupt from PWMG0. 0 / 1: disable / enable.
4	-	R/W	Enable interrupt from comparator. 0 / 1: disable / enable.
3	-	R/W	Reserved.
2	-	R/W	Enable interrupt from Timer16 overflow. 0 / 1: disable / enable.
1	-	R/W	Enable interrupt from PB0. 0 / 1: disable / enable.
0	-	R/W	Enable interrupt from PA0. 0 / 1: disable / enable.

8.2. 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 PWMG0, this bit is set by hardware and cleared by software. 0 / 1: No request / Request
4	-	R/W	Interrupt Request from comparator, this bit is set by hardware and cleared by software. 0 / 1: No request / Request
3	-	-	Reserved.
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	Interrupt Request from pin PB0, this bit is set by hardware and cleared by software. 0 / 1: No request / Request
0	-	R/W	Interrupt Request from pin PA0, this bit is set by hardware and cleared by software. 0 / 1: No request / Request

Note: *INTEN* and *INTRQ* 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.

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

Bit	Reset	R/W	Description
7 – 5	-	WO	Reserved.
4	0	WO	Timer16 edge selection. 0 : rising edge to trigger interrupt 1 : falling edge to trigger interrupt
3 – 2	00	WO	PB0 edge selection. 00 : both rising edge and falling edge to trigger interrupt 01 : rising edge to trigger interrupt 10 : falling edge to trigger interrupt 11 : reserved.
1 – 0	00	WO	PA0 edge selection. 00 : both rising edge and falling edge to trigger interrupt 01 : rising edge to trigger interrupt 10 : falling edge to trigger interrupt 11 : reserved.

8.4. 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.

8.5. 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.

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

Bit	Reset	R/W	Description
7 – 3	11111	WO	Enable PA7~PA3 digital input and wake up event. 1 / 0: enable / disable. When this bit is “0”, the function is disable to wake up from PA7~PA3 toggling.
2 – 1	-	-	Reserved.
0	1	WO	Enable PA0 digital input and wake up event and interrupt request. 1 / 0 : enable / disable. When this bit is “0”, the function is disable wake up from PA0 toggling and interrupt request from this pin.

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

Bit	Reset	R/W	Description
7 – 1	0xFF	WO	Enable PB7~PB1 digital input and wake up event. 1 / 0: enable / disable. When this bit is “0”, the function is disable wake up from PB7~PB1 toggling.
0		WO	Enable PB0 digital input and wake up event and interrupt request. 1 / 0 : enable / disable. When this bit is “0”, the function is disable wake up from PB0 toggling and interrupt request from this pin.

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

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

9.1.4. 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.

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

Bit	Reset	R/W	Description
7 – 3	00000	R/W	Port PA7 ~ PA3 pull-high registers. This register is used to enable the internal pull-high device on each corresponding pin of port A. 0 / 1 : disable / enable
2 – 1	-	-	Reserved. Please keep 0 for future compatibility.
0	0	R/W	Port PA0 pull-high registers. This register is used to enable the internal pull-high device on each corresponding pin of port A. 0 / 1 : disable / enable

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

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

9.1.7. 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.

9.1.8. 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 PFC154 have the same structure. The hardware diagram of IO buffer is shown as Fig. 8.

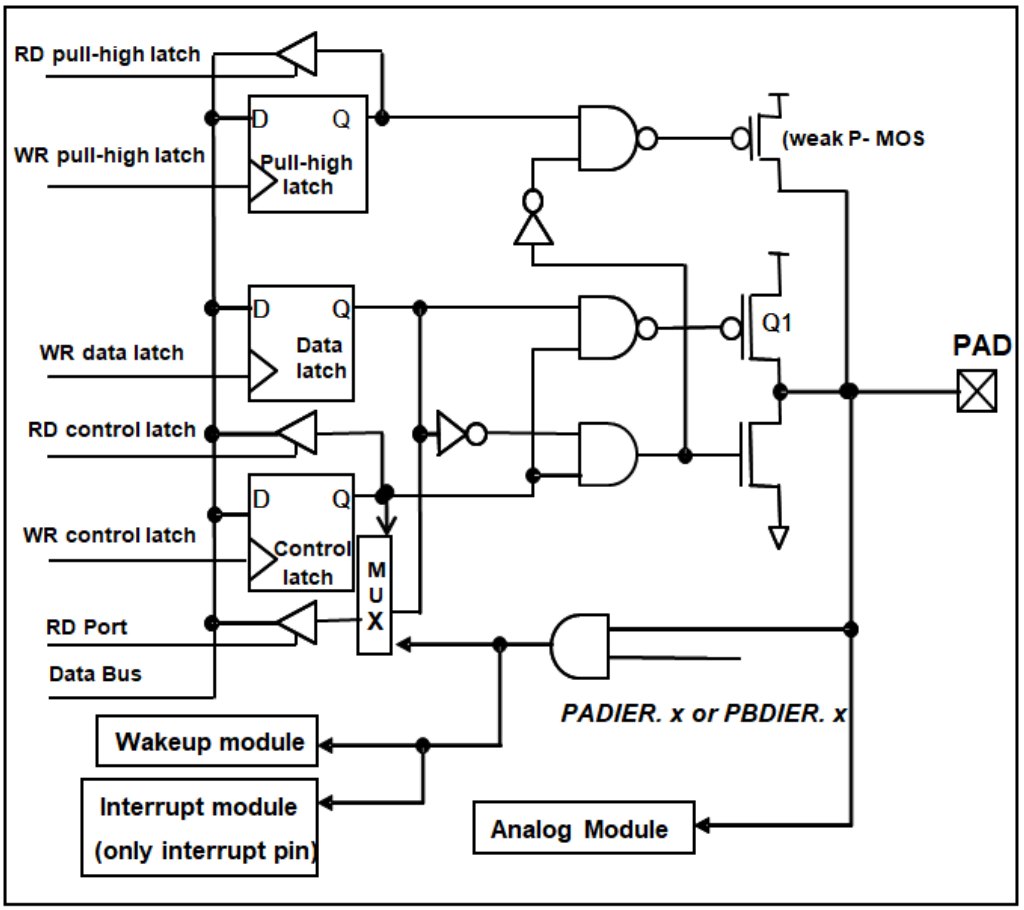


Fig. 8: Hardware diagram of IO buffer

9.2.2. IO Pin Functions

(1) Input / output function:

PFC154 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*) and 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 7 shows the configuration table of bit 0 of port A.

PA.0	*PAC*.0	*PAPH*.0	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 7: PA0 Configuration Table

(2) Wake-up function:

When PFC154 put in Power-Down or Power-Save mode, every 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.

(4) Drive capability optional:

Most IOs can be adjusted their Driving or Sinking current capability to Normal or Low by code option Drive.

9.2.3. IO Pin Usage and Setting

(1) IO pin as digital input

- ◆ When IO is set as digital input, the level of Vih and Vil would changes with the voltage and temperature. Please follow the minimum value of Vih and the maximum value of Vil.
- ◆ 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 output pin

- ◆ PA5 can be set to be Open-Drain output pin only, output high requires adding pull-up resistor.

(4) PA5 is set to be PRSTB input pin.

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

(5) 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 >10Ω 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

PFC154 provide a 16-bit hardware timer (Timer16/T16) and its block diagram is shown in Fig. 9.

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, 4, 16 or 64 is selected by *T16M*[4:3] 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.

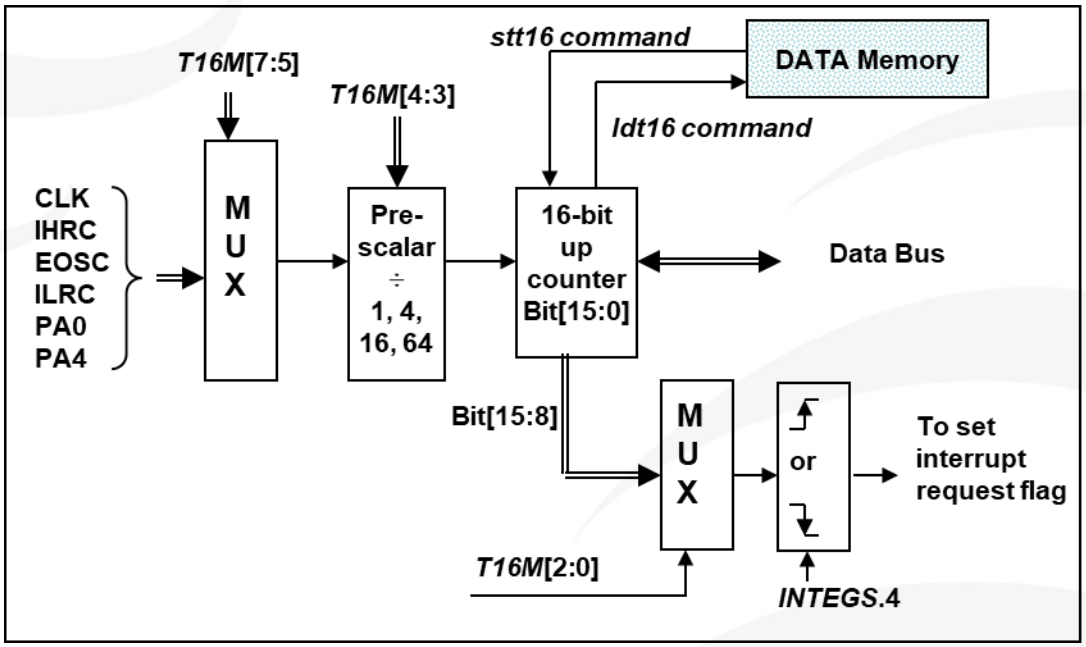


Fig. 9: Hardware diagram of Timer16

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

```
T16M    IO_RW    0x06
$ 7~5:      STOP, SYSCLK, X, PA4_F, IHRC, EOSC, ILRC, PA0_F      // 1st par.
$ 4~3:      /1, /4, /16, /64                                     // 2nd par.
$ 2~0:      BIT8, BIT9, BIT10, BIT11, BIT12, BIT13, BIT14, BIT15 // 3rd 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
```

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

Bit	Reset	R/W	Description
7 – 5	000	R/W	Timer Clock source selection: 000: Timer 16 is disabled 001: CLK (system clock) 010: reserved 011: PA4 falling edge (from external pin) 100: IHRC 101: EOSC 110: ILRC 111: PA0 falling edge (from external pin)
4 – 3	00	R/W	Internal clock divider. 00: /1 01: /4 10: /16 11: /64
2 – 0	000	R/W	Interrupt source selection. Interrupt event happens when selected bit is changed. 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

10.1.3. 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.

10.2. 8-bit Timer with PWM Generation (Timer2, Timer3)

Two 8-bit hardware timers (Timer2/TM2, Timer3/TM3) with PWM generation are implemented in the PFC154. Timer2 is used as the example to describe its function due to these two 8-bit timers are the same. Fig. 10 shows the Timer2 hardware diagram.

Bit[7:4] of register *TM2C* are used to select the clock source of Timer2. And the output of Timer2 is selected by *TM2C*[3:2]. The clock frequency divide module is controlled by bit [6:5] of *TM2S* register. *TM2B* register controls the upper bound of 8-bit counter. It will be clear to zero automatically when the counter values reach for upper bound register. The counter values can be set or read back by *TM2CT* register.

There are two operating modes for Timer2: period mode and PWM mode; period mode is used to generate periodical output waveform or interrupt event; PWM mode is used to generate PWM output waveform with optional 6-bit or 8-bit PWM resolution.

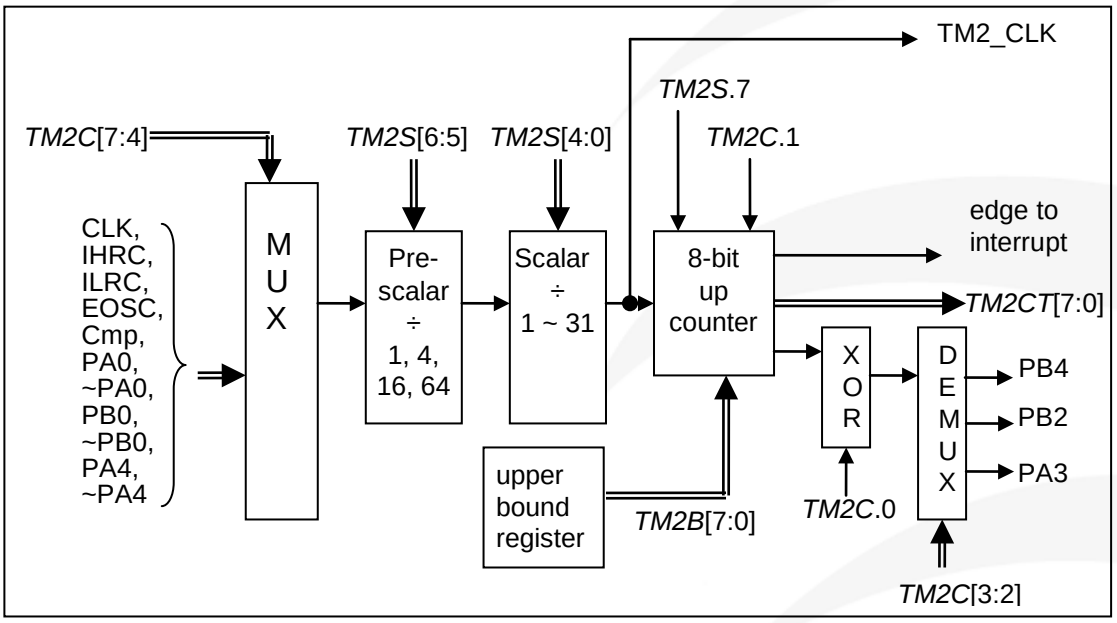
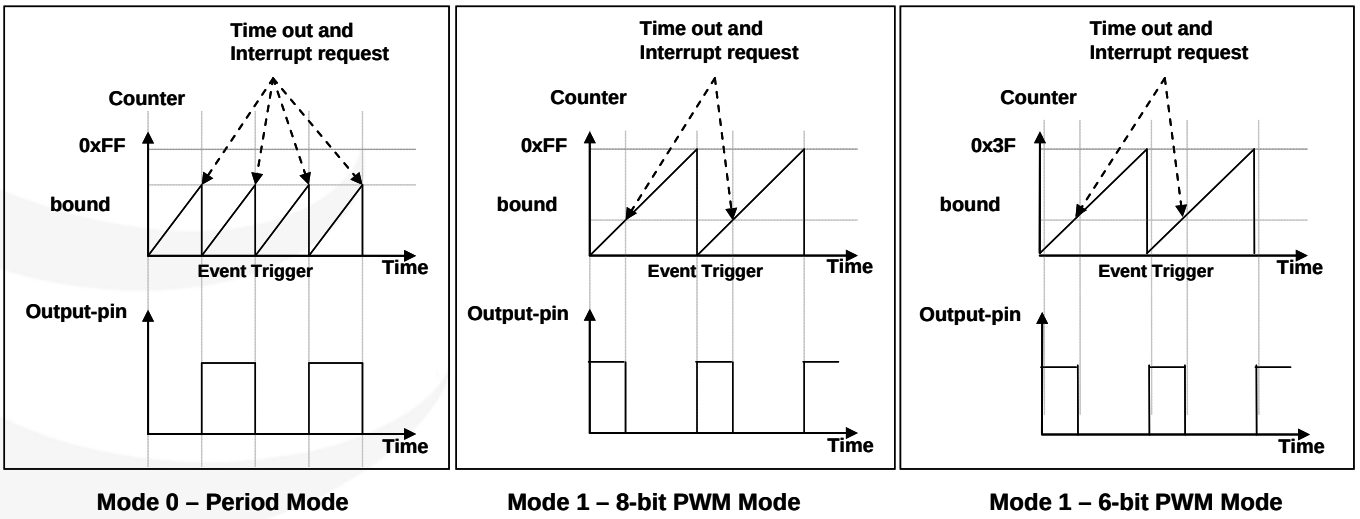


Fig. 10: Timer2 hardware diagram

The output of Timer3 can be sent to pin PB5, PB6 or PB7.

Fig. 11 shows the timing diagram of Timer2 for both period mode and PWM mode.



Mode 0 – Period Mode **Mode 1 – 8-bit PWM Mode** **Mode 1 – 6-bit PWM Mode**

Fig. 11: Timing diagram of Timer2 in period mode and PWM mode (*TM2C*.1=1)

A Code Option GPC_PWM is for the applications which need the generated PWM waveform to be controlled by the comparator result. If the Code Option GPC_PWM is selected, the PWM output stops while the comparator output is 1 and then the PWM output turns on while the comparator output goes back to 0, as shown in Fig. 12.

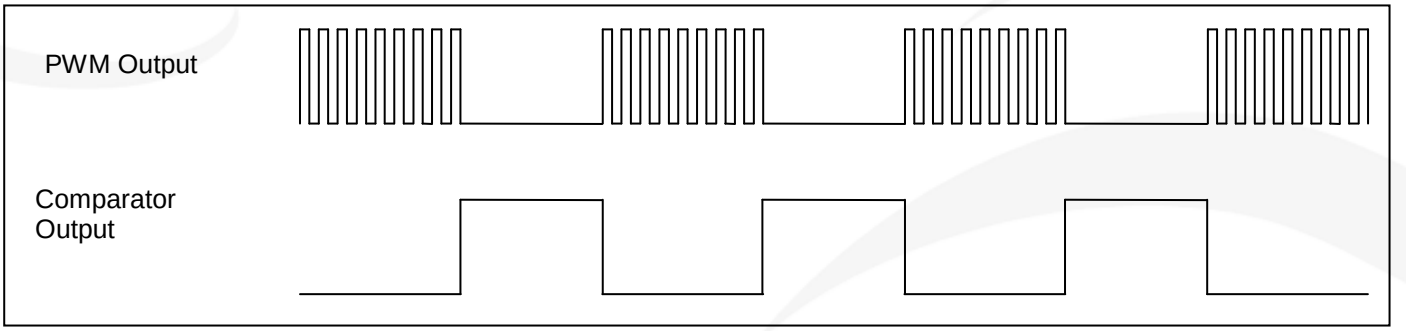


Fig.12: Comparator controls the output of PWM waveform

10.2.1. Timer2, Timer3 Related Registers

10.2.1.1. Timer2 Scalar Register (*TM2S*), address = 0x17

Bit	Reset	R/W	Description
7	0	WO	PWM resolution selection. 0 : 8-bit 1 : 6-bit or 7-bit (by code option TMx_bit)
6 – 5	00	WO	Timer2 clock pre-scalar. 00 : ÷ 1 01 : ÷ 4 10 : ÷ 16 11 : ÷ 64
4 – 0	00000	WO	Timer2 clock scalar.

10.2.1.2. Timer2 Control Register (*TM2C*), address = 0x1C

Bit	Reset	R/W	Description
7 – 4	0000	R/W	Timer2 clock selection. 0000 : disable 0001 : CLK 0010 : IHRC or IHRC *2 (by code option TMx_source) 0011 : EOSC 0100 : ILRC 0101 : comparator output 1000 : PA0 (rising edge) 1001 : ~PA0 (falling edge) 1010 : PB0 (rising edge) 1011 : ~PB0 (falling edge) 1100 : PA4 (rising edge) 1101 : ~PA4 (falling edge) Others: reserved **Notice:** In ICE mode and IHRC is selected for Timer2 clock, the clock sent to Timer2 does NOT be stopped, Timer2 will keep counting when ICE is in halt state.
3 – 2	00	R/W	Timer2 output selection. 00 : disable 01 : PB2 10 : PA3 11 : PB4
1	0	R/W	Timer2 mode selection. 0 / 1 : period mode / PWM mode
0	0	R/W	Enable to inverse the polarity of Timer2 output. 0 / 1: disable / enable

10.2.1.3. Timer2 Counter Register (*TM2CT*), address = 0x1D

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

10.2.1.4. Timer2 Bound Register (*TM2B*), address = 0x09

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

10.2.1.5. Timer3 Counter Register (*TM3CT*), address = 0x33

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

10.2.1.6. Timer3 Scalar Register (*TM3S*), address = 0x34

Bit	Reset	R/W	Description
7	0	WO	PWM resolution selection. 0 : 8-bit 1 : 6-bit or 7-bit (by code option TMx_bit)
6 – 5	00	WO	Timer3 clock pre-scalar. 00 : ÷ 1 01 : ÷ 4 10 : ÷ 16 11 : ÷ 64
4 – 0	00000	WO	Timer3 clock scalar.

10.2.1.7. Timer3 Bound Register (*TM3B*), address = 0x35

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

10.2.1.8. Timer3 Control Register (*TM3C*), address = 0x32

Bit	Reset	R/W	Description
7 – 4	0000	R/W	Timer3 clock selection. 0000 : disable 0001 : CLK 0010 : IHRC or IHRC *2 (by code option TMx_source) 0011 : EOSC 0100 : ILRC 0101 : comparator output 1000 : PA0 (rising edge) 1001 : ~PA0 (falling edge) 1010 : PB0 (rising edge) 1011 : ~PB0 (falling edge) 1100 : PA4 (rising edge) 1101 : ~PA4 (falling edge) Others: reserved **Notice:** In ICE mode and IHRC is selected for Timer3 clock, the clock sent to Timer3 does NOT be stopped, Timer3 will keep counting when ICE is in halt state.
3 – 2	00	R/W	Timer3 output selection. 00 : disable 01 : PB5 10 : PB6 11 : PB7
1	0	R/W	Timer3 mode selection. 0 / 1 : period mode / PWM mode
0	0	R/W	Enable to inverse the polarity of Timer3 output. 0 / 1: disable / enable

10.2.2. Using the Timer2 to Generate Periodical Waveform

If periodical mode is selected, the duty cycle of output is always 50%. Its frequency can be summarized as below:

Frequency of Output = Y ÷ [2 × (K+1) × S1 × (S2+1)]

Where,

Y = *TM2C*[7:4] : frequency of selected clock source
K = *TM2B*[7:0] : bound register in decimal
S1 = *TM2S*[6:5] : pre-scalar (S1 = 1, 4, 16, 64)
S2 = *TM2S*[4:0] : scalar register in decimal (S2 = 0 ~ 31)

Example 1:

TM2C = 0b0001_1000, Y=4MHz
TM2B = 0b0111_1111, K=127
TM2S = 0b0_00_00000, S1=1, S2=0
frequency of output = 4MHz ÷ [2 × (127＋1) × 1 × (0＋1)] = 15.625KHz

Example 2:

TM2C = 0b0001_1000, Y=4MHz
TM2B = 0b0000_0001, K=1
TM2S = 0b0_00_00000, S1=1, S2=0
frequency = 4MHz ÷ [2 × (1＋1) × 1 × (0＋1)]=1MHz

The sample program for using the Timer2 to generate periodical waveform to PA3 is shown as below:

```
void    FPPA0 (void)
{
    . ADJUST_IC     SYSCLK=IHRC/4, IHRC=16MHz, VDD=5V
    ...
    TM2CT = 0x00;
    TM2B = 0x7f;
    TM2S = 0b0_00_00001;            //  8-bit PWM, pre-scalar = 1, scalar = 2
    TM2C = 0b0001_10_0_0;           //  system clock, output=PA3, period mode
    while(1)
    {
        nop;
    }
}
```

10.2.3. Using the Timer2 to Generate 8-bit PWM Waveform

If 8-bit PWM mode is selected, it should set *TM2C*[1]=1 and *TM2S*[7]=0, the frequency and duty cycle of output waveform can be summarized as below:

Frequency of Output = Y ÷ [256 × S1 × (S2+1)]

Duty of Output =[(K＋1) ÷ 256] × 100%

Where,

Y = *TM2C*[7:4] : frequency of selected clock source
K = *TM2B*[7:0] : bound register in decimal
S1= *TM2S*[6:5] : pre-scalar (S1 = 1, 4, 16, 64)
S2 = *TM2S*[4:0] : scalar register in decimal (S2 = 0 ~ 31)

Example 1:

TM2C = 0b0001_1010, Y=4MHz
TM2B = 0b0111_1111, K=127
TM2S = 0b0_00_00000, S1=1, S2=0

frequency of output = 4MHz ÷ (256 × 1 × (0+1)) = 15.625KHz
duty of output = [(127+1) ÷ 256] × 100% = 50%

Example 2:

TM2C = 0b0001_1010, Y=4MHz
TM2B = 0b0000_1001, K = 9
TM2S = 0b0_00_00000, S1=1, S2=0

frequency of output = 4MHz ÷ (256 × 1 × (0+1)) = 15.625KHz
duty of output = [(9+1) ÷ 256] × 100% = 3.9%

The sample program for using the Timer2 to generate PWM waveform from PA3 is shown as below:

```
void    FPPA0 (void)
{
    .ADJUST_IC    SYSCLK=IHRC/4, IHRC=16MHz, VDD=5V
    wdreset;
    TM2CT = 0x00;
    TM2B = 0x7f;
    TM2S = 0b0_00_00001;        //    8-bit PWM, pre-scalar = 1, scalar = 2
    TM2C = 0b0001_10_1_0;       //    system clock, output=PA3, PWM mode
    while(1)
    {
        nop;
    }
}
```

10.2.4. Using the Timer2 to Generate 6-bit PWM Waveform

If 6-bit PWM mode is selected, it should set *TM2C*[1]=1 and *TM2S*[7]=1, the frequency and duty cycle of output waveform can be summarized as below:

Frequency of Output = Y ÷ [64 × S1 × (S2+1)]

Duty of Output = [(K＋1) ÷ 64] × 100%

Where,

TM2C[7:4] = Y : frequency of selected clock source

TM2B[7:0] = K : bound register in decimal

TM2S[6:5] = S1 : pre-scalar (S1 = 1, 4, 16, 64)

TM2S[4:0] = S2 : scalar register in decimal (S2 = 0 ~ 31)

Example 1:

TM2C = 0b0001_1010, Y=4MHz

TM2B = 0b0011_1111, K=63

TM2S = 0b1_00_00000, S1=1, S2=0

frequency of output = 4MHz ÷ (64 × 1 × (0+1)) = 62.5KHz

duty of output = [(63+1) ÷ 64] × 100% = 100%

Example 2:

TM2C = 0b0001_1010, Y=4MHz

TM2B = 0b0000_0000, K=0

TM2S = 0b1_00_00000, S1=1, S2=0

Frequency = 4MHz ÷ (64 × 1 × (0+1)) = 62.5KHz

Duty = [(0+1) ÷ 64] × 100% =1.5%

10.3. 11-bit PWM Generation

Three 11-bit hardware PWM generators (PWMG0, PWMG1 & PWMG2) are implemented in the PFC154. PWMG0 is used as the example to describe its functions due to all of them are almost the same. Their individual outputs are listed as below:

(1) PWMG0 – PA0, PB4, PB5
(2) PWMG1 – PA4, PB6, PB7
(3) PWMG2 – PA3, PB2, PB3, PA5 (ICE does NOT support PA5)

10.3.1. PWM Waveform

A PWM output waveform (Fig. 13) has a time-base (T_{Period} = Time of Period) and a time with output high level (Duty Cycle). The frequency of the PWM output is the inverse of the period ($f_{PWM} = 1/T_{Period}$), the resolution of the PWM is the clock count numbers for one period (N bits resolution, $2^N \times T_{clock} = T_{Period}$).

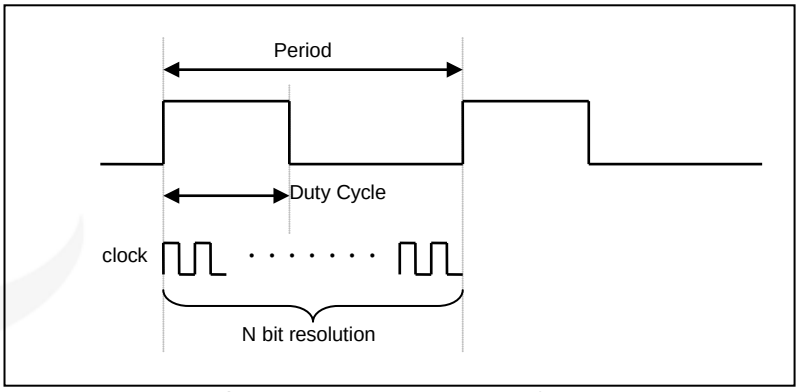


Fig. 13: PWM Output Waveform

10.3.2. Hardware and Timing Diagram

Fig. 14 shows the hardware diagram of 11-bit Timer. The clock source can be IHRC or system clock. The PWM output pin is selected by register *PWMG0C*. The period of PWM waveform is defined in the PWM upper bond high and low registers (*PWMG0CUBH* and *PWMG0CUBL*), the duty cycle of PWM waveform is defined in the PWM duty high and low registers (*PWMG0DTH* and *PWMG0DTL*).

Selecting code option GPC_PWM can also control the generated PWM waveform by the comparator result. Please refer the section of Timer2 for further information.

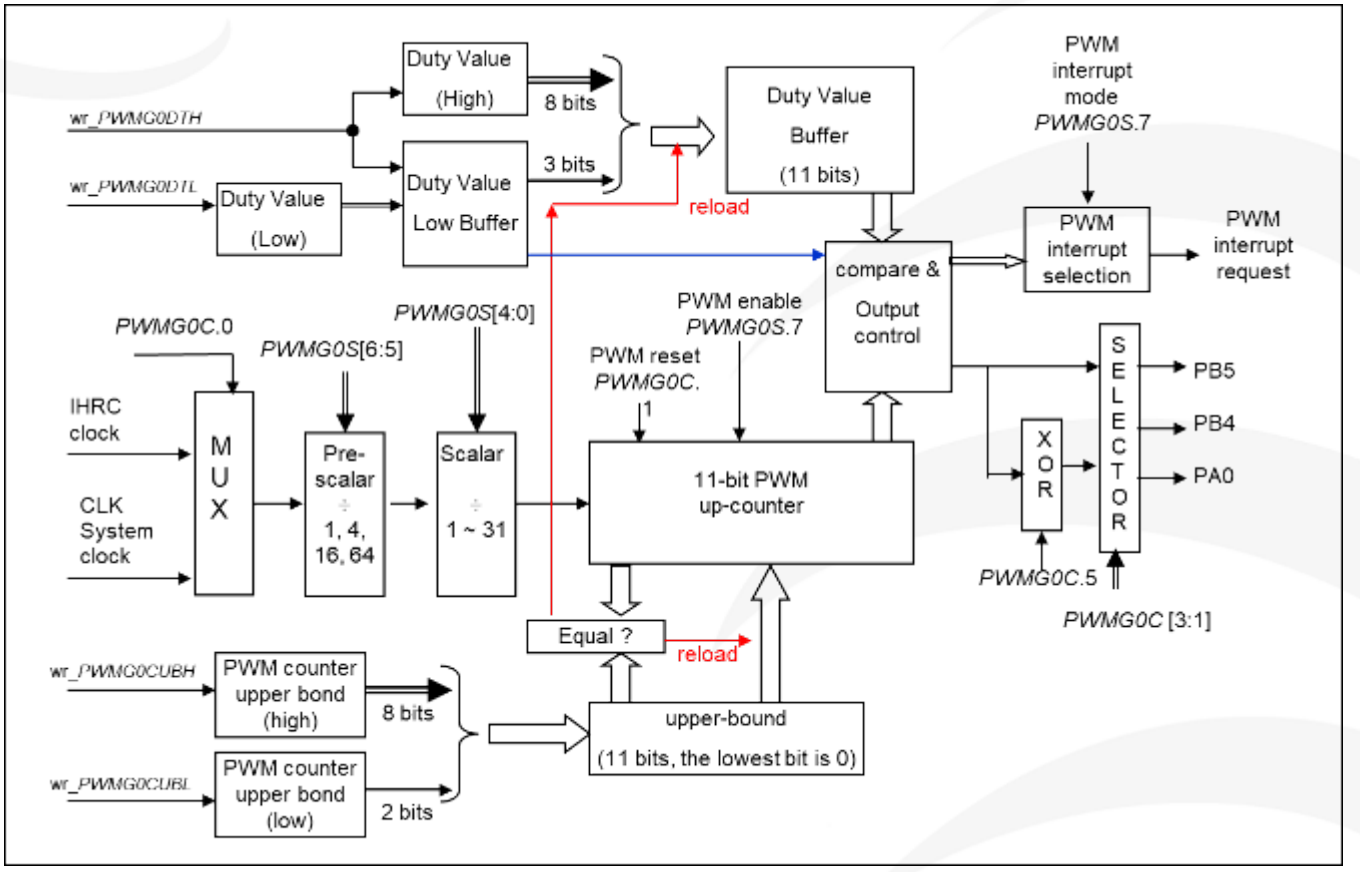


Fig. 14: Hardware Diagram of 11-bit PWM Generator 0 (PWMG0)

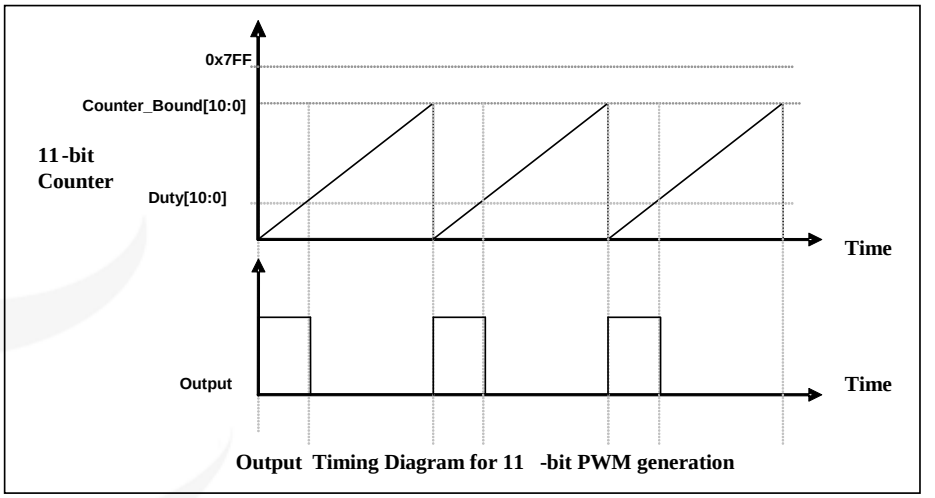


Fig. 15: Output Timing Diagram of 11-bit PWM Generator

10.3.3. Equations for 11-bit PWM Generator

The frequency and duty cycle of 11bit PWM can be obtained from the following formula:

PWM Frequency F_{PWM} = $F_{clock\ source}$ ÷ [P × (K + 1) × (CB10_1 + 1)]

PWM Duty(in time) = (1 / F_{PWM}) × (DB10_1 + DB0 × 0.5 + 0.5) ÷ (CB10_1 + 1)

PWM Duty(in percentage) = (DB10_1 + DB0 × 0.5 + 0.5) ÷ (CB10_1 + 1) × 100%

Where,

P = *PWMGxS* [6:5] : pre-scalar (**P** = 1, 4, 16, 64)

K = *PWMGxS* [4:0] : scalar in decimal (**K** =0 ~ 31)

DB10_1 = Duty_Bound[10:1] = {*PWMGxDTH[7:0], PWMGxDTL[7:6]*}, duty bound

DB0 = Duty_Bound[0] = *PWMGxDTL[5]*

CB10_1 = Counter_Bound[10:1] = {*PWMGxCUBH[7:0], PWMGxCUBL[7:6]*}, counter bound

10.3.4. 11bit PWM Related Registers

10.3.4.1. PWMG0 control Register (*PWMG0C*), address = 0x20

Bit	Reset	R/W	Description
7	0	R/W	Enable PWMG0 generator. 0 / 1: disable / enable
6	-	RO	Output status of PWMG0 generator.
5	0	R/W	Enable to inverse the polarity of PWMG0 generator output. 0 / 1: disable / enable.
4	0	R/W	PWMG0 counter reset. Writing “1” to clear PWMG0 counter and this bit will be self clear to 0 after counter reset.
3 – 1	0	R/W	Select PWM output pin for PWMG0. 000: none 001: PB5 011: PA0 100: PB4 Others: reserved
0	0	R/W	Clock source of PWMG0 generator. 0: CLK, 1: IHRC or IHRC*2 (by code option PWM_source)

10.3.4.2. PWMG0 Scalar Register (*PWMG0S*), address = 0x21

Bit	Reset	R/W	Description
7	0	WO	PWMG0 interrupt mode 0: Generate interrupt when counter matches the duty value 1: Generate interrupt when counter is 0
6 – 5	0	WO	PWMG0 clock pre-scalar 00 : ÷1 01 : ÷4 10 : ÷16 11 : ÷64
4 – 0	0	WO	PWMG0 clock divider

10.3.4.3. PWMG0 Counter Upper Bound High Register (*PWMG0CUBH*), address = 0x24

Bit	Reset	R/W	Description
7 – 0	-	WO	Bit[10:3] of PWMG0 counter upper bound.

10.3.4.4. PWMG0 Counter Upper Bound Low Register (*PWMG0CUBL*), address = 0x25

Bit	Reset	R/W	Description
7 – 6	-	WO	Bit[2:1] of PWMG0 counter upper bound.
5	-	WO	Bit[0] of PWMG0 counter upper bound.
4 – 0	-	-	Reserved

10.3.4.5. PWMG0 Duty Value High Register (*PWMG0DTH*), address = 0x22

Bit	Reset	R/W	Description
7 – 0	-	WO	Duty values bit[10:3] of PWMG0.

10.3.4.6. PWMG0 Duty Value Low Register (*PWMG0DTL*), address = 0x23

Bit	Reset	R/W	Description
7 – 5	-	WO	Duty values bit [2:0] of PWMG0.
4 – 0	-	-	Reserved

Note: It's necessary to write *PWMG0DTL* Register before writing *PWMG0DTH* Register.

10.3.4.7. PWMG1 Control Register (*PWMG1C*), address = 0x26

Bit	Reset	R/W	Description
7	0	R/W	Enable PWMG1. 0 / 1: disable / enable
6	-	RO	Output of PWMG1.
5	0	R/W	Enable to inverse the polarity of PWMG1 output. 0 / 1 : disable / enable.
4	0	R/W	PWMG1 counter reset. Writing "1" to clear PWMG1 counter and this bit will be self clear to 0 after counter reset.
3 – 1	0	R/W	Select PWMG1 output pin. 000: none 001: PB6 011: PA4 100: PB7 Others: reserved
0	0	R/W	Clock source of PWMG1. 0: CLK, 1: IHRC or IHRC*2 (by code option PWM_source)

10.3.4.8. PWMG1 Scalar Register (*PWMG1S*), address = 0x27

Bit	Reset	R/W	Description
7	0	WO	PWMG1 interrupt mode. 0: Generate interrupt when counter matches the duty value 1: Generate interrupt when counter is 0
6 – 5	0	WO	PWMG1 clock pre-scalar. 00 : ÷1 01 : ÷4 10 : ÷16 11 : ÷64
4 – 0	0	WO	PWMG1 clock divider.

10.3.4.9. PWMG1 Counter Upper Bound High Register (*PWMG1CUBH*), address = 0x2A

Bit	Reset	R/W	Description
7 – 0	0x00	WO	Bit[10:3] of PWMG1 counter upper bound.

10.3.4.10. PWMG1 Counter Upper Bound Low Register (*PWMG1CUBL*), address = 0x2B

Bit	Reset	R/W	Description
7 – 6	00	WO	Bit[2:1] of PWMG1 counter upper bound.
5	0	WO	Bit[0] of PWMG1 counter upper bound.
4 – 0	-	-	Reserved

10.3.4.11. PWMG1 Duty Value High Register (*PWMG1DTH*), address = 0x28

Bit	Reset	R/W	Description
7 – 0	0x00	WO	Duty values bit[10:3] of PWMG1.

10.3.4.12. PWMG1 Duty Value Low Register (*PWMG1DTL*), address = 0x29

Bit	Reset	R/W	Description
7 – 5	000	WO	Duty values bit[2:0] of PWMG1.
4 – 0	-	-	Reserved

Note: It's necessary to write *PWMG1DTL* Register before writing *PWMG1DTH* Register.

10.3.4.13. PWMG2 Control Register (*PWMG2C*), address = 0x2C

Bit	Reset	R/W	Description
7	0	R/W	Enable PWMG2. 0 / 1: disable / enable.
6	-	RO	Output of PWMG2.
5	0	R/W	Enable to inverse the polarity of PWMG2 output. 0 / 1: disable / enable.
4	0	R/W	PWMG2 counter reset. Writing "1" to clear PWMG2 counter and this bit will be self clear to 0 after counter reset.
3 – 1	0	R/W	Select PWMG2 output pin. 000: disable 001: PB3 011: PA3 100: PB2 101: PA5 (ICE does NOT Support.) Others: reserved
0	0	R/W	Clock source of PWMG2. 0: CLK, 1: IHRC or IHRC*2 (by code option PWM_source)

10.3.4.14. PWMG2 Scalar Register (*PWMG2S*), address = 0x2D

Bit	Reset	R/W	Description
7	0	RO	PWMG2 interrupt mode. 0: Generate interrupt when counter matches the duty value 1: Generate interrupt when counter is 0.
6 – 5	0	RO	PWMG2 clock pre-scalar. 00 : ÷1 01 : ÷4 10 : ÷16 11 : ÷64
4 – 0	0	RO	PWMG2 clock divider.

10.3.4.15. PWMG2 Counter Upper Bound High Register (*PWMG2CUBH*), address = 0x30

Bit	Reset	R/W	Description
7 – 0	0x00	WO	Bit[10:3] of PWMG2 counter upper bound.

10.3.4.16. PWMG2 Counter Upper Bound Low Register (*PWMG2CUBL*), address = 0x31

Bit	Reset	R/W	Description
7 – 6	00	WO	Bit[2:1] of PWMG2 counter upper bound.
5	0	WO	Bit[0] of PWMG2 counter upper bound.
4 – 0	-	-	Reserved

10.3.4.17. PWMG2 Duty Value High Register (*PWMG2DTH*), address = 0x2E

Bit	Reset	R/W	Description
7 – 0	0x00	WO	Duty values bit[10:3] of PWMG2.

10.3.4.18. PWMG2 Duty Value Low Register (*PWMG2DTL*), address = 0x2F

Bit	Reset	R/W	Description
7 – 5	000	WO	Duty values bit[2:0] of PWMG2.
4 – 0	-	-	Reserved

Note: It's necessary to write *PWMG2DTL* Register before writing *PWMG2DTH* Register.

10.3.5. Examples of PWM Waveforms with Complementary Dead Zones

Users can use two 11bit PWM generators to output two complementary PWM waveforms with dead zones. Take PWMG0 output PWM0, PWMG1 output PWM1 as an example, the program reference is as follows.

In addition, Timer2 and Timer3 can also output 8-bit PWM waveforms with complementary dead zones of two bands. The principle is similar to this, and it will not be described in detail.

```
#define  dead_zone_R  2          //    Control dead-time before rising edge of PWM1.
#define  dead_zone_F  3          //    Control dead-time after falling edge of PWM1.

void     FPPA0 (void)
{
  .ADJUST_IC     SYSCLK=IHRC/16, IHRC=16MHz, VDD=5V;
  //.....
  Byte   duty     =    60;          //    Represents the duty cycle of PWM0
  Byte   _duty    =    100 - duty;  //    Represents the duty cycle of PWM1

  //************** Set the counter upper bound and duty cycle **************
  PWMG0DTL     =    0x00;
  PWMG0DTH     =    _duty;
  PWMG0CUBL    =    0x00;
  PWMG0CUBH    =    100;

  PWMG1DTL     =    0x00;
  PWMG1DTH     =    _duty - dead_zone_F;
  // Use duty cycle to adjust the dead-time after the falling edge of PWM1
  PWMG1CUBL    =    0x00;
  PWMG1CUBH    =    100;       // The above values are assigned before enable PWM output

  //******************** PWM out control **********************
  $ PWMG0C Enable,Inverse,PA0,SYSCLK;       //    PWMG0 output the PWM0 waveform to PA0
  $ PWMG0S INTR_AT_DUTY,/1,/1;

  .delay dead_zone_R;        //    Use delay to adjust the dead-time before the rising edge of PWM1

  $ PWMG1C Enable, PA4, SYSCLK;             //    PWMG1 output the PWM1 waveform to PA4
  $ PWMG1S INTR_AT_DUTY, /1, /1;

  //***** Note: for the output control part of the program, the code sequence can not be moved ******//

 While(1)
      {  nop;      }
}
```

The PWM0 / PWM1 waveform obtained by the above program is shown in Fig. 16.

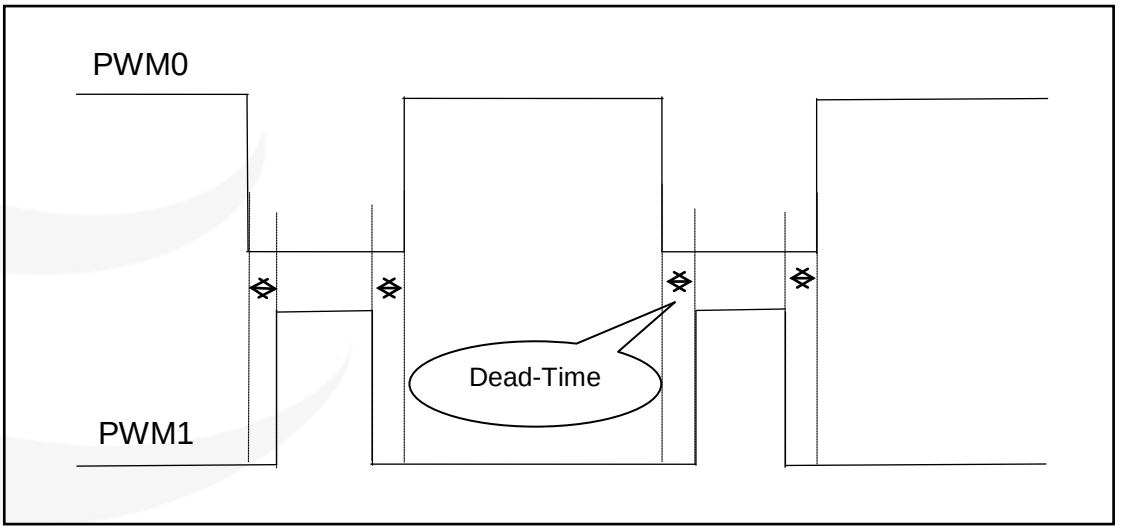


Fig. 16: Two complementary PWM waveforms with dead zones

Users can modify the **dead_zone_R** and **dead_zone_F** values in the program to adjust the dead-time. Table 8 provides data corresponding to different dead-time for users' reference. Where, if dead-time = 4us, then there are dead zones of 4us before and after PWM1 high level.

dead-time (us)	dead_zone_R	dead_zone_F
4 (minimum)	0	2
6	2	3
8	4	4
10	6	5
12	8	6
14	10	7

Table 8: The value of dead-time for reference

Dead_zone_R and **dead_zone_F** need to work together to get an ideal dead-time. If user wants to adjust other dead-time, please note that **dead_zone_R** and **dead_zone_F** need to meet the following criteria:

dead_zone_R	dead_zone_F
1 / 2 / 3	> 1
4 / 5 / 6 / 7	> 2
8 / 9	> 3
...	...

11. Special Functions

11.1. Comparator

One hardware comparator is built inside the PFC154; Fig. 17 shows its hardware diagram. It can compare signals between two input pins. The two signals to be compared, one is the plus input and the other one is the minus input. The plus input pin is selected by register *GPCC*.0, and the minus input pin is selected by *GPCC*[3:1].

The output result can be:
(1) read back by *GPCC*.6;
(2) inversed the polarity by *GPCC*.4;
(3) sampled by Time2 clock (TM2_CLK) which comes from *GPCC*.5;
(4) enabled to output to PA0 directly by *GPCS*.7;
(5) used to request interrupt service.

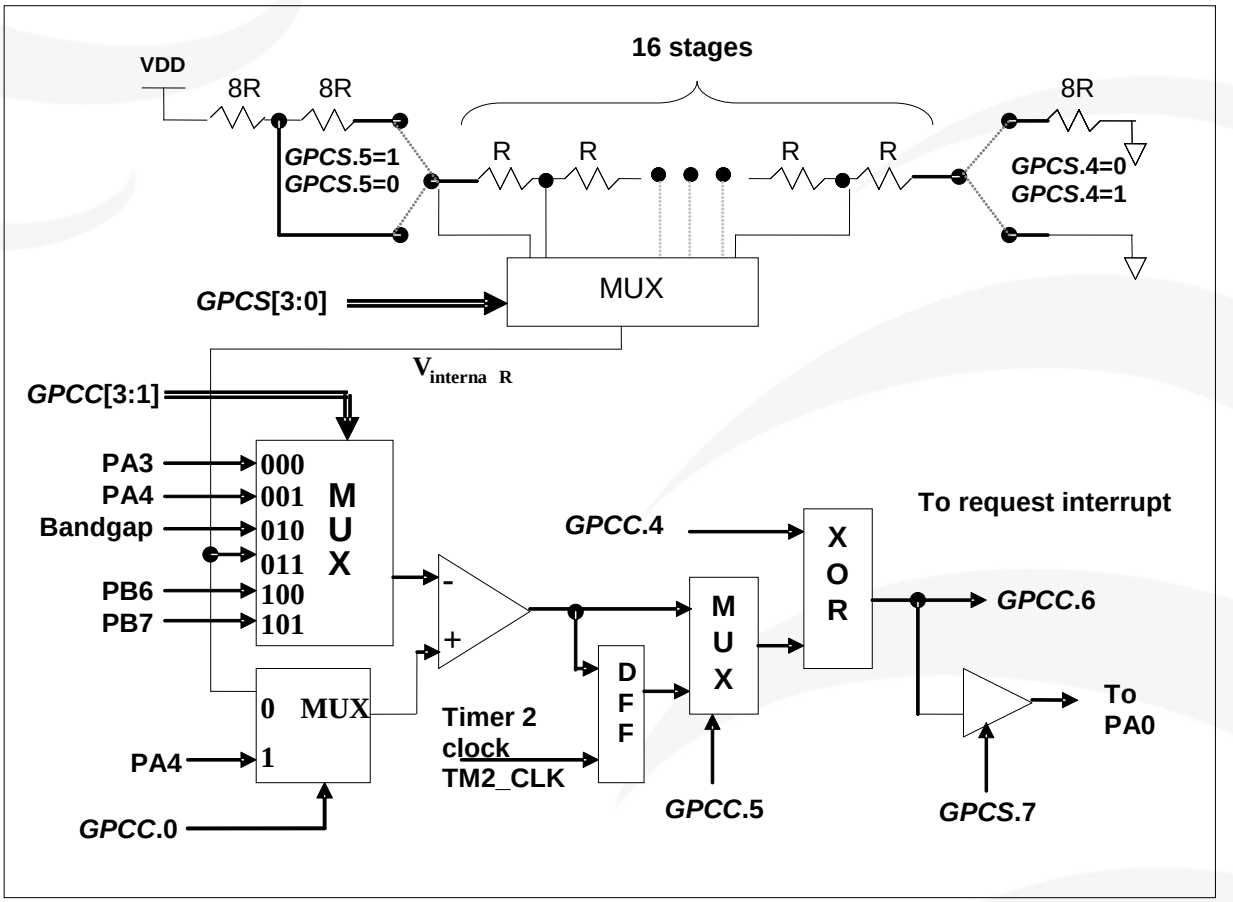


Fig. 17: Hardware diagram of comparator

11.1.1. Comparator Control Register (*GPCC*), address = 0x18

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	Select whether the comparator result output will be sampled by TM2_CLK? 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 comparator. 000: PA3 001: PA4 010: Internal 1.20 volt Bandgap reference voltage 011: $V_{internal\ R}$ 100: PB6 (not for EV5) 101: PB7 (not for EV5) 11X: reserved
0	0	R/W	Selection the plus input (+) of comparator. 0 : $V_{internal\ R}$ 1 : PA4

11.1.2. Comparator Selection Register (*GPCS*), address = 0x19

Bit	Reset	R/W	Description
7	0	WO	Comparator output enable (to PA0). 0 / 1 : disable / enable (Please avoid this situation: GPCS will affect the PA3 output function when selecting output to PA0 output in ICE.)
6	-	-	Reserved.
5	0	WO	Selection of high range of comparator.
4	0	WO	Selection of low range of comparator.
3 – 0	0000	WO	Selection the voltage level of comparator. 0000 (lowest) ~ 1111 (highest)

11.1.3. 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 ***gpcs*** register are used to select the maximum and minimum values of $V_{internal\ R}$ and bit [3:0] of ***gpcs*** register are used to select one of the voltage level which is deivided-by-16 from the defined maximum level to minimum level. Fig. 18 to Fig. 21 shows four conditions to have different reference voltage $V_{internal\ R}$. By setting the ***gpcs*** register, the internal reference voltage $V_{internal\ R}$ can be ranged from (1/32)*V_{DD} to (3/4)*V_{DD}.

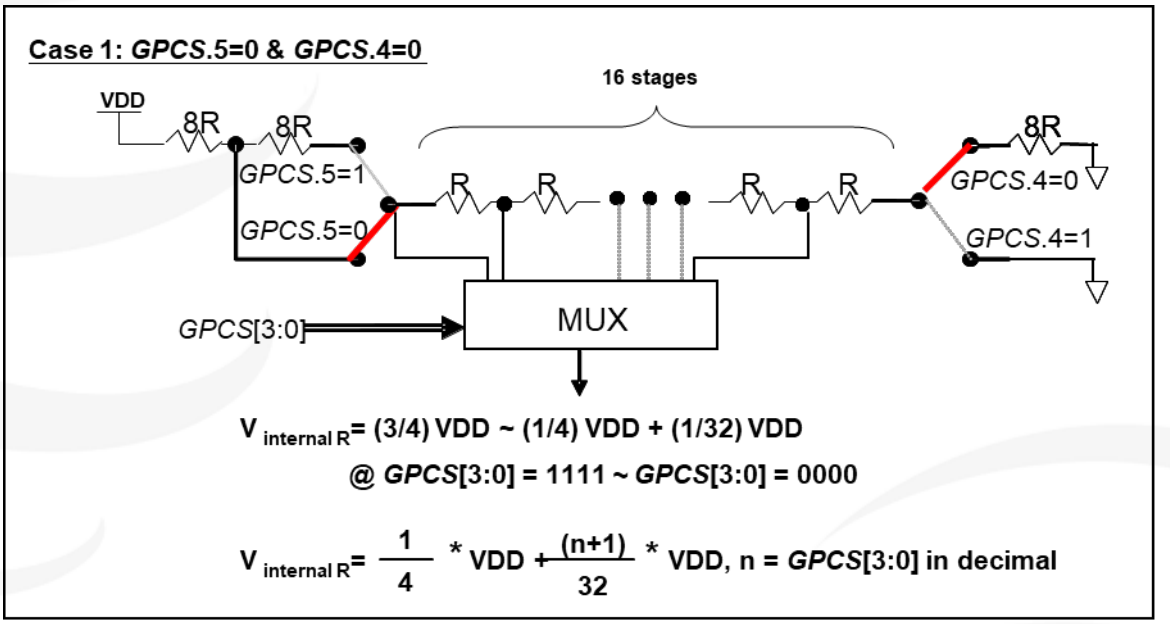


Fig. 18: $V_{internal\ R}$ hardware connection if gpcs.5=0 and gpcs.4=0

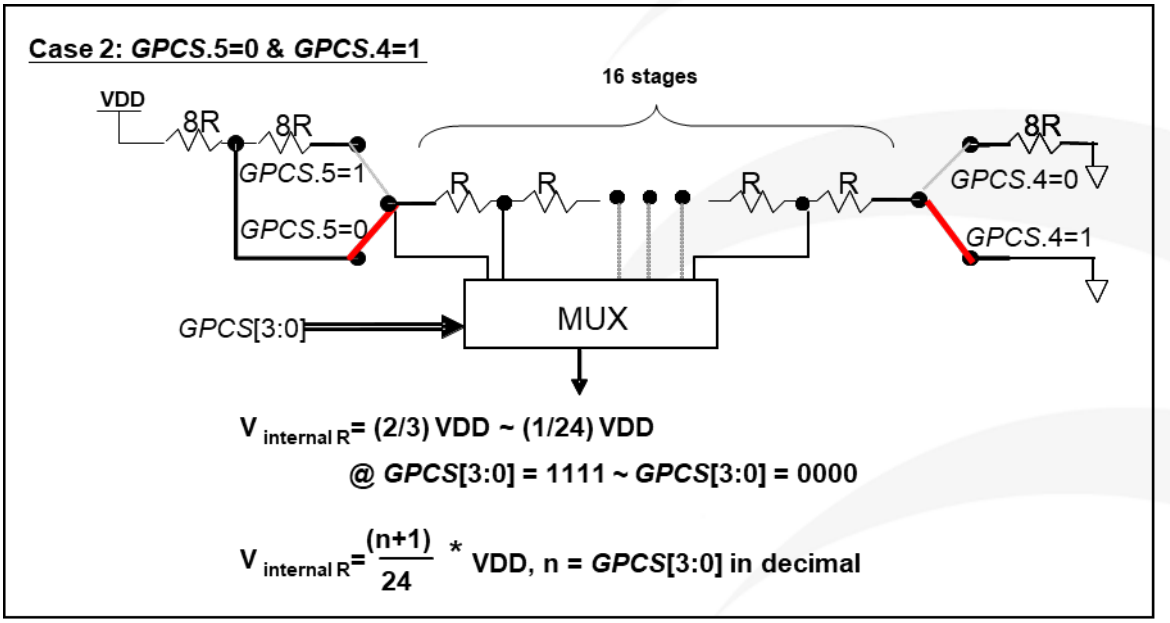


Fig. 19: $V_{internal\ R}$ hardware connection if gpcs.5=0 and gpcs.4=1

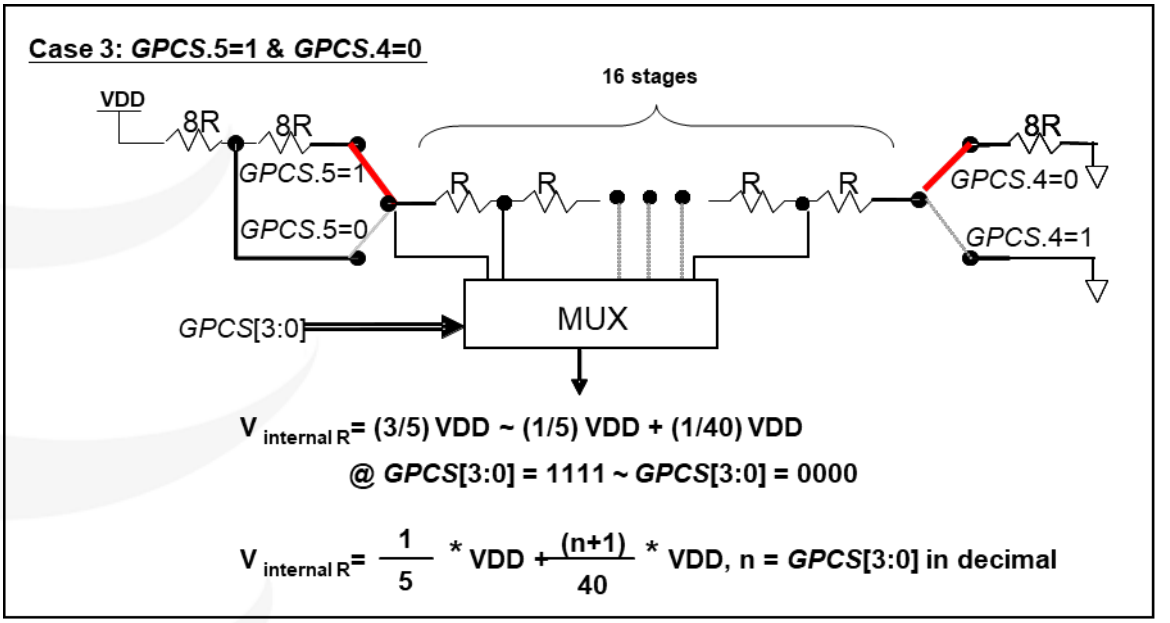


Fig. 20: $V_{internal\ R}$ hardware connection if gpcs.5=1 and gpcs.4=0

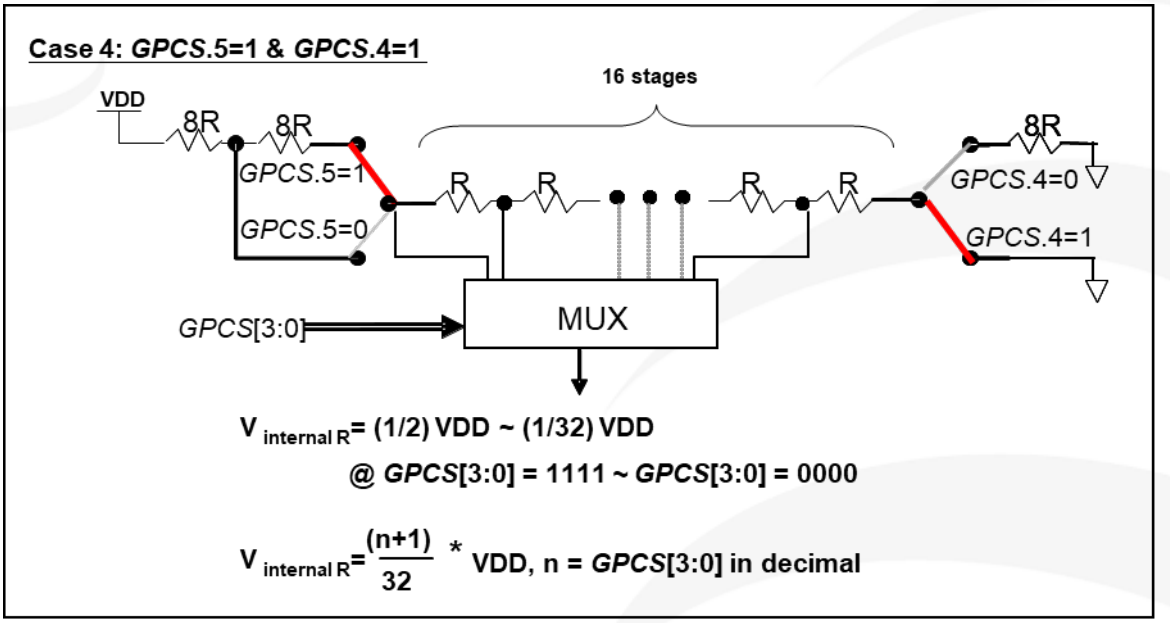


Fig. 21: $V_{internal\ R}$ hardware connection if gpcs.5=1 and gpcs.4=1

11.1.4. Using the Comparator

Case 1:

Choosing PA3 as minus input and $V_{internal\ R}$ with (18/32)*V_{DD} voltage level as plus input. $V_{internal\ R}$ is configured as the above Figure "*GPCS*[5:4] = 2b'00" and *GPCS*[3:0] = 4b'1001 (n=9) to have $V_{internal\ R}$ = (1/4)*V_{DD} + [(9+1)/32]*V_{DD} = [(9+9)/32]*V_{DD} = (18/32)*V_{DD}.

```
GPCS    = 0b0_0_00_1001;        // V_internal R = V_DD*(18/32)
GPCC    = 0b1_0_0_0_000_0;      // enable comp, - input: PA3, + input: V_internal R
PADIER  = 0bxxxx_0_xxx;         // disable PA3 digital input to prevent leakage current
```

or

```
$ GPCS    V_DD*18/32;
$ GPCC    Enable, N_PA3, P_R;   // - input: N_xx，+ input: P_R(V_internal R)
PADIER = 0bxxxx_0_xxx;
```

Case 2:

Choosing $V_{internal\ R}$ as minus input with (22/40)*V_{DD} voltage level and PA4 as plus input, the comparator result will be inversed and then output to PA0. $V_{internal\ R}$ is configured as the above Figure "*GPCS*[5:4] = 2b'10" and *GPCS*[3:0] = 4b'1101 (n=13) to have $V_{internal\ R}$ = (1/5)*V_{DD} + [(13+1)/40]*V_{DD} = [(13+9)/40]*V_{DD} = (22/40)*V_{DD}.

```
GPCS    = 0b1_0_10_1101;        // output to PA0, V_internal R = V_DD*(22/40)
GPCC    = 0b1_0_0_1_011_1;      // Inverse output, - input: V_internal R, + input: PA4
PADIER  = 0bxxxx_0_xxx;         // disable PA4 digital input to prevent leakage current
```

or

```
$ GPCS    Output, V_DD*22/40;
$ GPCC    Enable, Inverse, N_R, P_PA4;  // - input: N_R(V_internal R)，+ input: P_xx
PADIER = 0bxxx_0_xxxx;
```

Note: When selecting output to PA0 output, *GPCS* will affect the PA3 output function in ICE. Though the IC is fine, be careful to avoid this error during emulation.

11.1.5. Using the Comparator and Bandgap 1.20V

The internal Bandgap module provides a stable 1.20V output, and it can be used to measure the external supply voltage level. The Bandgap 1.20V is selected as minus input of comparator and $V_{internal\ R}$ is selected as plus input, the supply voltage of $V_{internal\ R}$ is VDD, the VDD voltage level can be detected by adjusting the voltage level of $V_{internal\ R}$ to compare with Bandgap.

If N (*GPCS*[3:0] in decimal) is the number to let $V_{internal\ R}$ closest to Bandgap 1.20 volt, the supply voltage VDD can be calculated by using the following equations:

For using Case 1: V_{DD} = [32 / (N+9)] * 1.20 volt ;
For using Case 2: V_{DD} = [24 / (N+1)] * 1.20 volt ;
For using Case 3: V_{DD} = [40 / (N+9)] * 1.20 volt ;
For using Case 4: V_{DD} = [32 / (N+1)] * 1.20 volt ;

Case 1:

```
$ GPCS   VDD*12/40;                 //  4.0V * 12/40 = 1.2V
$ GPCC   Enable, BANDGAP, P_R;      //   - input: BANDGAP, + input: P_R(Vinternal R)
....
if   (GPC_Out)                      //  or GPCC.6
{                                   //  when VDD > 4V
}
else
{                                   //  when VDD < 4V
}
```

11.2. VDD/2 Bias Voltage Generator

The four pins of PFC154: PA4, PA3, PA0 and PB0, can generate VDD/2 as the COM function when driving the LCD display. This function can be enabled by setting the register *MISC*.4 as 1.

MISC Register (*MISC*), address = 0x08			
Bit	**Reset**	**R/W**	**Description**
7 – 5	-	-	Reserved. (keep 0 for future compatibility)
4	**0**	**WO**	**Enable VDD/2 bias voltage generator** **0 / 1 : Disable / Enable (ICE cannot be dynamically switched)**
3	-	-	Reserved.
2	0	WO	Disable LVR function. 0 / 1 : Enable / Disable
1 – 0	00	WO	Watch dog time out period

The COM port can generate VDD/2 by switching it to input mode (*PAC*.x / *PBC*.x=0). However, keep in mind to turn off the pull-high resistor (*PAPH.x* / *PBPH.x*=0) and digital input from *PADIER*.x / *PBDIER*.x to register prevent the output voltage level from disturbing. Fig.22 shows how to use this function.

The output function of COM port is the same as other normal IO.

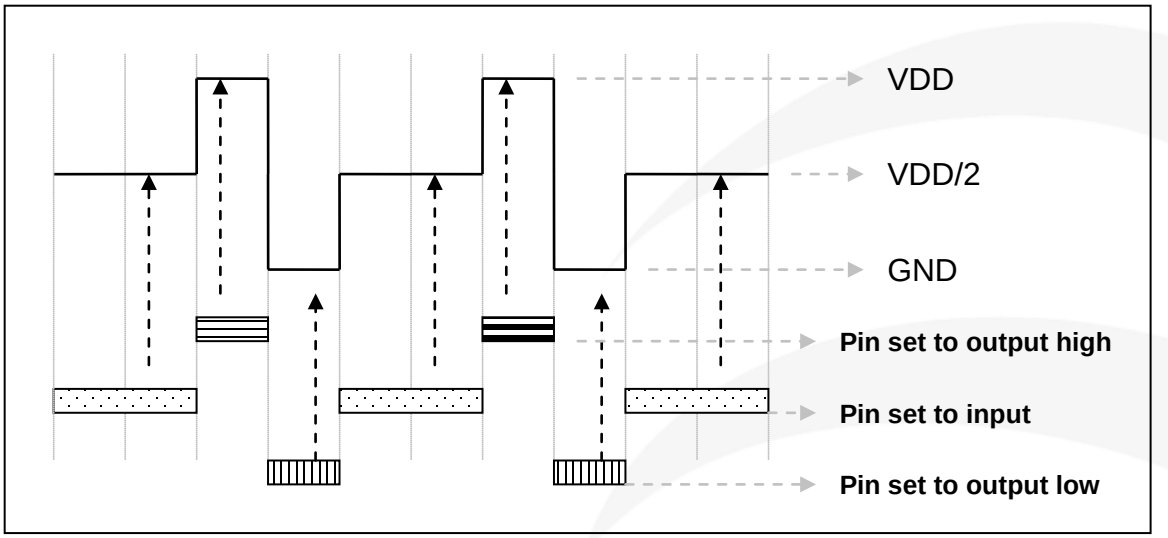


Fig. 22: Using VDD/2 bias voltage generator

12. Notes for Emulation

It is recommended to use PDK5S-I-S01/2(B) for emulation of PFC154. The following items should be noted when using PDK5S-I-S01/2(B) to emulate PFC154:

(1) PDK5S-I-S01/2(B) doesn't support the instruction *nadd, comp*.

(2) PDK5S-I-S01/2(B) doesn't support SYSCLK=ILRC/16.

(3) PDK5S-I-S01/2(B) doesn't support the dynamic setting of function *MISC.4* (Only fix to 0 or 1).

(4) PDK5S-I-S01/2(B) doesn't support the function *TM2C.GPCRS* and *TM3C.GPCRS.*

(5) PDK5S-I-S01/2(B) doesn't support the function *PWMG2C*.PA5.

(6) The PA3 output function will be affected when *GPCS* selects output to PA0.

(7) Watch dog time out period is different from PDK5S-I-S01/2(B):

WDT period	PDK5S-I-S01/2(B)	PFC154
MISC[1:0]=00	2048 * T_{ILRC}	8192 * T_{ILRC}
MISC[1:0]=01	4096 * T_{ILRC}	16384 * T_{ILRC}
MISC[1:0]=10	16384 * T_{ILRC}	65536 * T_{ILRC}
MISC[1:0]=11	256 * T_{ILRC}	262144 * T_{ILRC}

13. Program Writing

Please use PDK5S-P-003 to program. PDK3S-P-002 or older versions do not support programming PFC154.
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.

13.1. 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 PFC154. 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 7.8V, 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 0.01uF capacitor between VDD and GND at the handler port to the IC is always good for suppressing disturbance. But please DO NOT connect with > 0.01uF capacitor, otherwise, programming mode may be fail.**

13.2. 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 PFC154. 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 “PDK5S-P-003 UM”).

13.3. On-Board Writing

PFC154 can support On-board writing. On-Board Writing is known as the situation that the IC have to be programmed when the IC itself and other peripheral circuits and devices have already been mounted on the PCB. Five wires of PDK5S-P-003 are used for On-Board Writing: ICPCK, ICPDA, VDD, GND and ICVPP. They are used to connect PA3, PA6, VDD, GND and PA5 of the IC correspondingly.

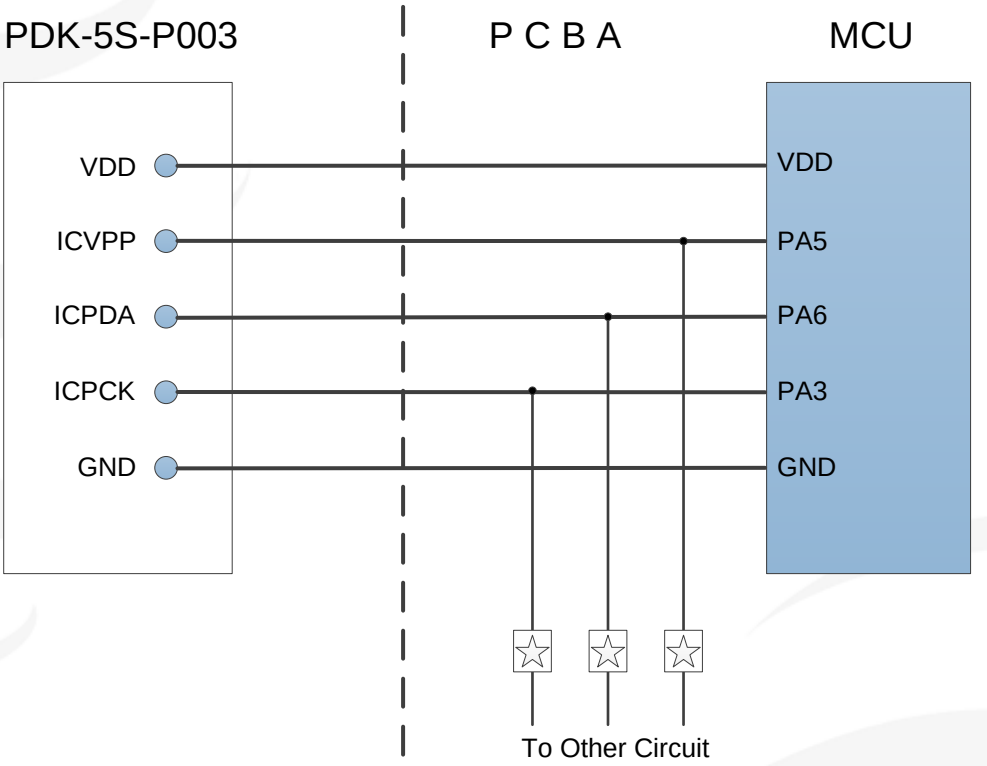


Fig. 23: Schematic Diagram of On-Board Wiring

The symbol ☆ on Fig. 23 can be either resistors or capacitors. They are used to isolate the programming signal wires from the peripheral circuit. it should be≧10KΩ for resistance while ≦220pF for capacitance.

Notice:

- In general, the limited-voltage programming mode is used in On-board Writing, Please refers to the 13.2 for more detail about limited-voltage programming mode.
- Any zener diode ≦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 ≧500uF SHOULD NOT be connected between VDD and GND of the PCB.
- In general, the writing signal pins PA3, PA5 and PA6 SHOULD NOT be considered as strong output pins.

14. Device Characteristics

14.1. Absolute Maximum Ratings

Name	Min	Typ.	Max	Unit	Notes
Supply Voltage (VDD)	2.2		5.5	V	**Exceed the maximum rating may cause permanent damage !!**
Input Voltage	-0.3		V_{DD} + 0.3	V	
Operating Temperature	-40		85	°C	
Storage Temperature	-50		125	°C	
Junction Temperature		150		°C	

14.2. DC/AC Characteristics

All data are acquired under the conditions of V_{DD}=5V, f_{SYS}=2MHz unless noted.

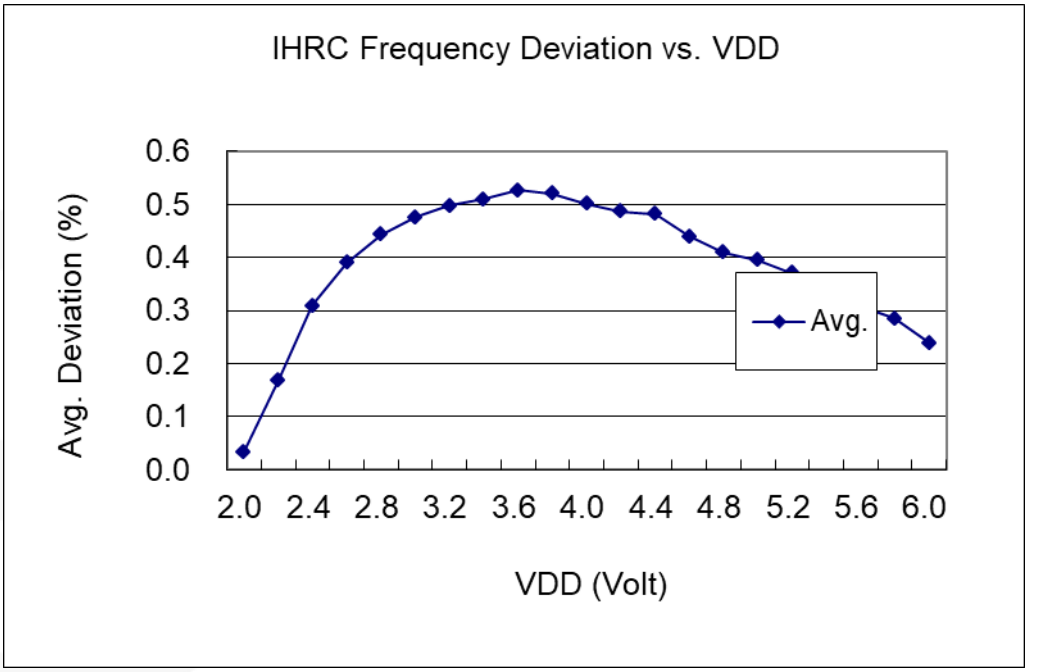
Symbol	Description	Min	Typ.	Max	Unit	Conditions(Ta=25°C)
V_{DD}	Operating Voltage	2.2*		5.5	V	* Subject to LVR tolerance
LVR%	Low Voltage Reset tolerance	-5		5	%	
f_{SYS}	System clock (CLK)* = IHRC/2 IHRC/4 IHRC/8 ILRC	 0 0 0	 66K	 8M 4M 2M	Hz	 V_{DD}≧3.5V V_{DD} ≧ 2.5V V_{DD} ≧ 2.2V V_{DD} = 5V
V_{POR}	Power On Reset Voltage	2.1	2.2	2.3	V	
I_{OP}	Operating Current		0.55 83 92		mA uA uA	f_{SYS}=IHRC/16=1MIPS@5V f_{SYS}=ILRC=66KHz@5V f_{SYS}=EOSC=32KHz@5V
I_{PD}	Power Down Current (by *stopsys* command)		1		uA	f_{SYS}= 0Hz, V_{DD} =5V
I_{PS}	Power Save Current (by *stopexe* command) *Disable IHRC		5		uA	V_{DD} =5V
V_{IL}	Input low voltage for IO lines	0		0.1 V_{DD}	V	
V_{IH}	Input high voltage for IO lines	0.7 V_{DD}		V_{DD}		
I_{OL}	IO lines sink current (normal)					
	*PA0,PA3,PA4,PB2,PB5,PB6		26		mA	V_{DD}=5V, V_{OL}=0.5V
	*PA6,PA7,PA5,PB0,PB1,PB3,PB4,PB7		20			
	IO lines sink current (low)					
	All IO		7		mA	V_{DD}=5V, V_{OL}=0.5V

Symbol	Description	Min	Typ.	Max	Unit	Conditions(Ta=25°C)
I_{OH}	IO lines drive current (normal)		15		mA	V_{DD}=5V, V_{OH}=4.5V
	IO lines drive current (low)		5			
V_{IN}	Input voltage	-0.3		V_{DD}+0.3	V	
$I_{INJ\ (PIN)}$	Injected current on pin			1	mA	V_{DD} +0.3≧V_{IN}≧ -0.3
R_{PH}	Pull-high Resistance		105		KΩ	V_{DD}=5.0V
f_{IHRC}	Frequency of IHRC after calibration *	15.84*	16*	16.16*	MHz	V_{DD}=5V, Ta=25°C
		15.20*		16.80*		V_{DD} =2.2V~5.5V, -40°C <Ta<85°C*
t_{INT}	Interrupt pulse width	30			ns	V_{DD} = 3.3V
V_{DR}	RAM data retention voltage*	1.5			V	In Power-Down mode
t_{WDT}	Watchdog timeout period		8k		T_{ILRC}	misc[1:0]=00 (default)
			16k			misc[1:0]=01
			64k			misc[1:0]=10
			256k			misc[1:0]=11
t_{SBP}	System boot-up period from power-on		47		ms	@ V_{DD} =5V
t_{WUP}	Wake-up time period		3000		T_{ILRC}	Where T_{ILRC} is the time period of ILRC
t_{RST}	External reset pulse width	120			us	
CPos	Comparator offset*	-	±10	±20	mV	
CPcm	Comparator input common mode*	0		V_{DD} -1.5	V	
CPspt	Comparator response time**		100	500	ns	Both rising and falling
CPmc	Stable time to change comparator mode		2.5	7.5	us	
CPcs	Comparator current consumption		20		uA	V_{DD} = 3.3V

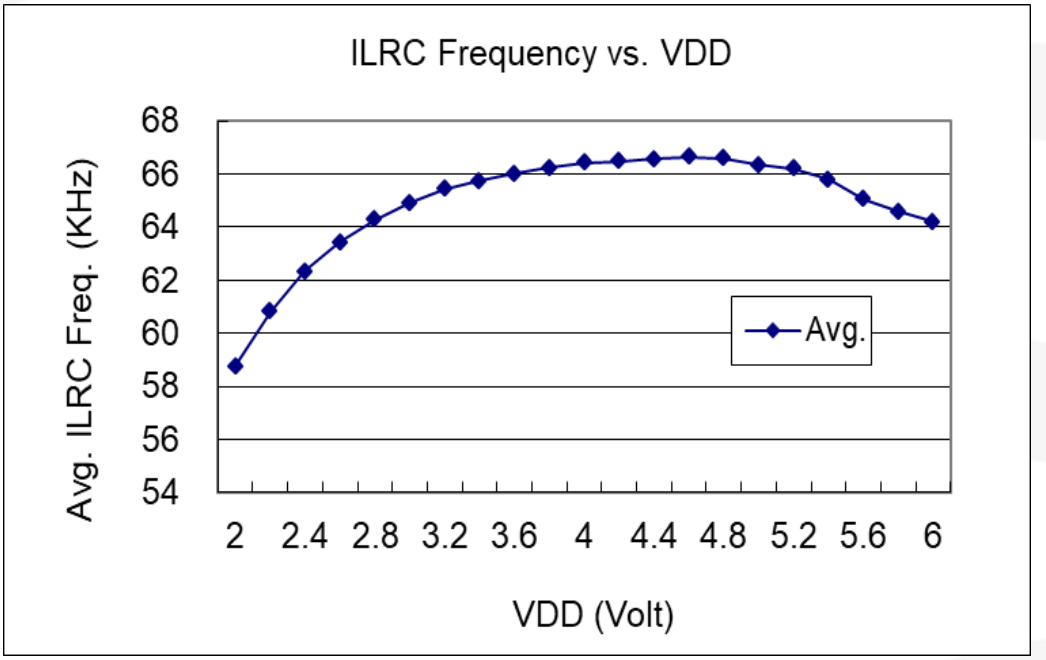
*These parameters are for design reference, not tested for every chip.

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.

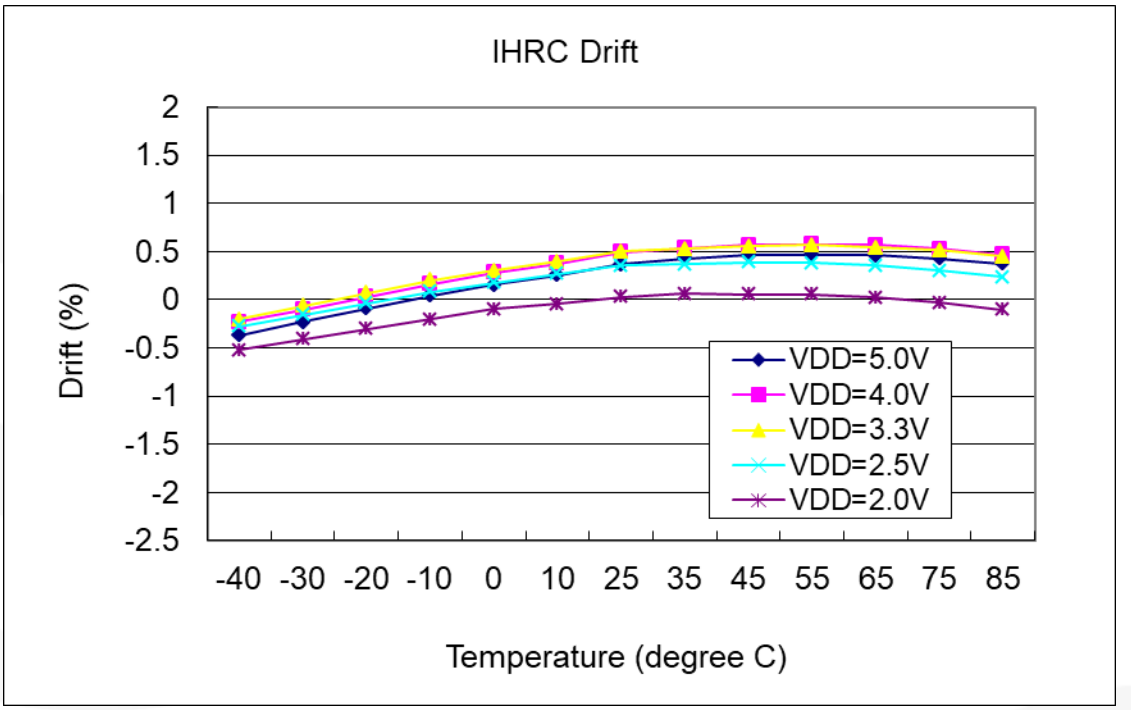
14.3. Typical IHRC Frequency vs. VDD (calibrated to 16MHz)



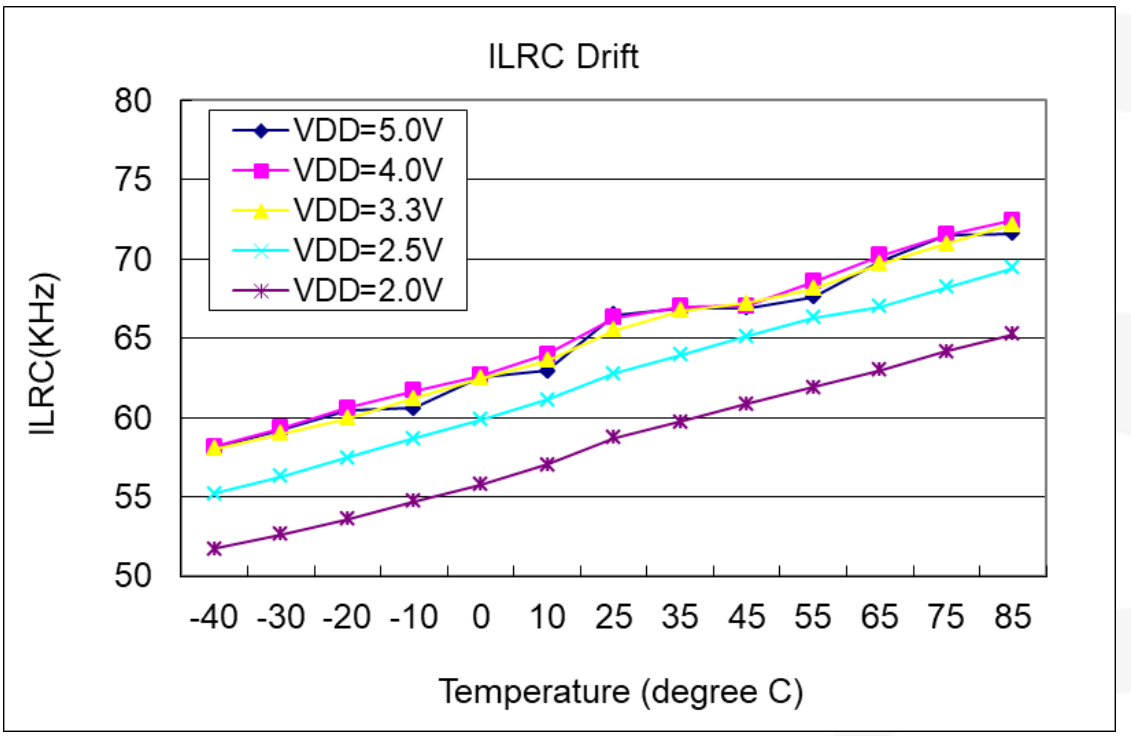
14.4. Typical ILRC Frequency vs. VDD



14.5. Typical IHRC Frequency vs. Temperature (calibrated to 16MHz)



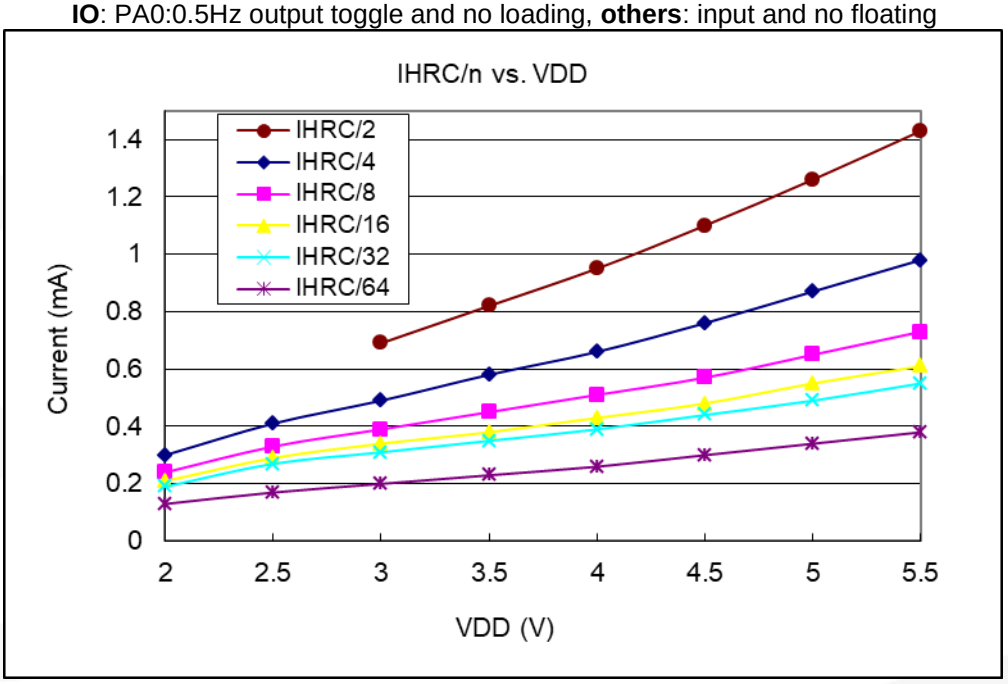
14.6. Typical ILRC Frequency vs. Temperature



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

Conditions: **ON**: IHRC, Band-gap; **OFF**: LVR, T16 modules, ILRC modules;

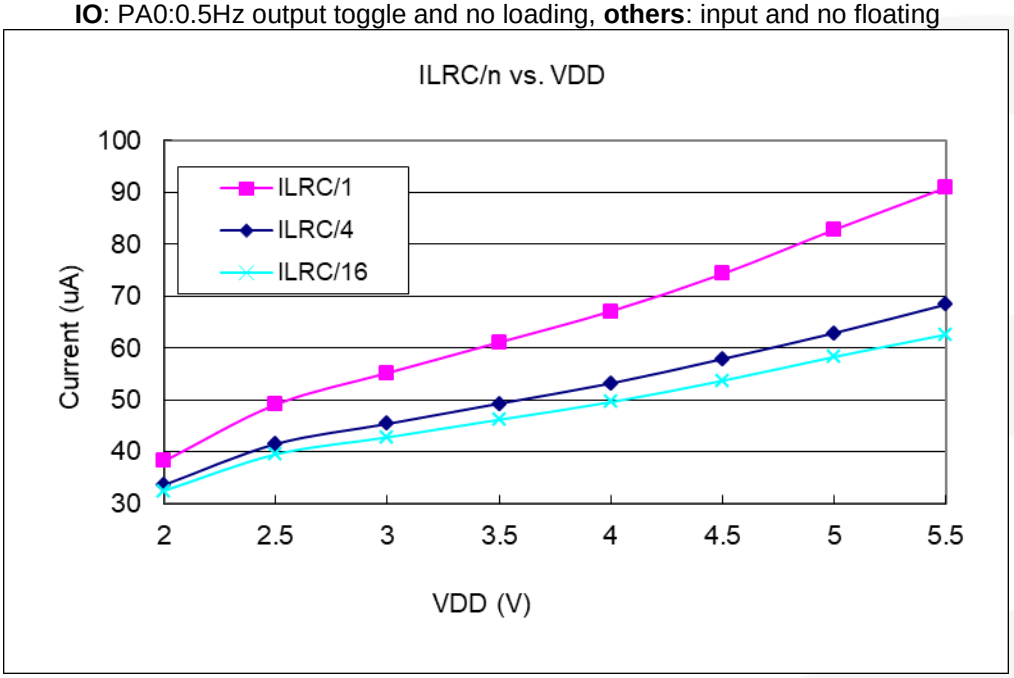
IO: PA0:0.5Hz output toggle and no loading, **others**: input and no floating



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

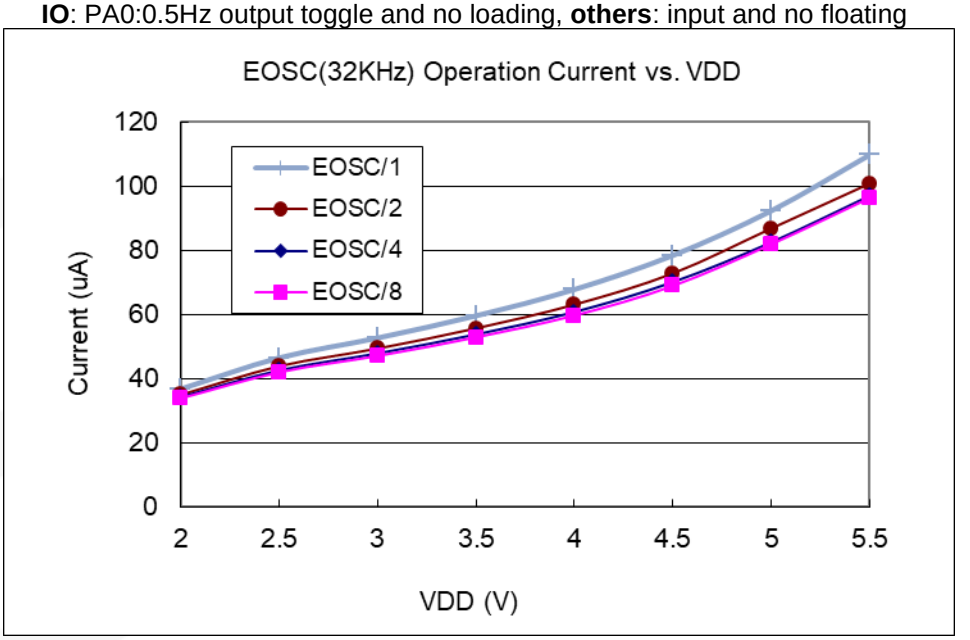
Conditions: **ON**: ILRC, Band-gap; **OFF**: LVR, T16 modules, IHRC modules;

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



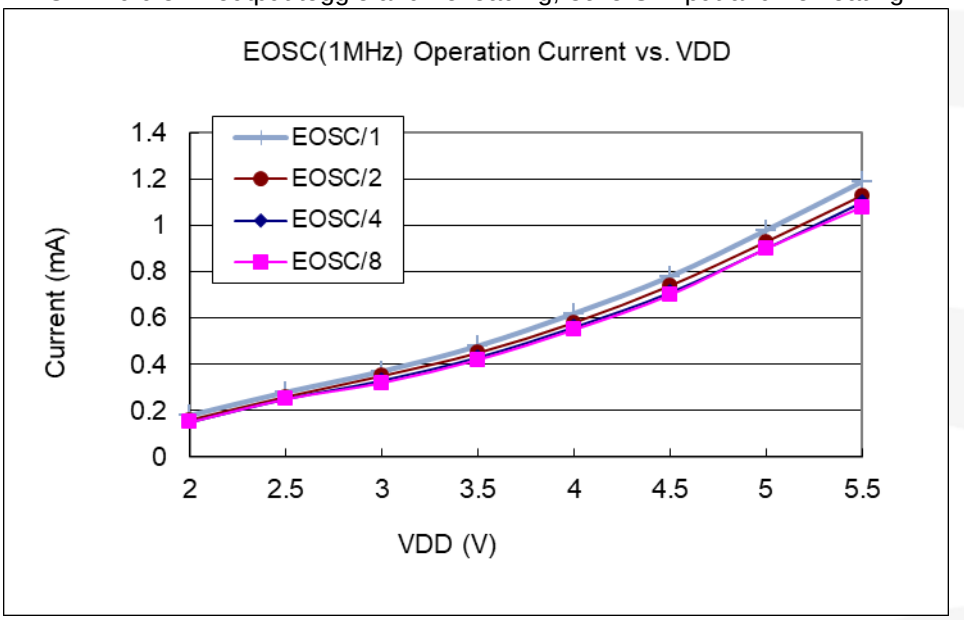
14.9. Typical Operating Current vs. VDD and CLK=32KHz EOSC / n

Conditions: **ON**: EOSC, Band-gap; **OFF**: LVR, T16 modules, IHRC, ILRC modules;
IO: PA0:0.5Hz output toggle and no loading, **others**: input and no floating



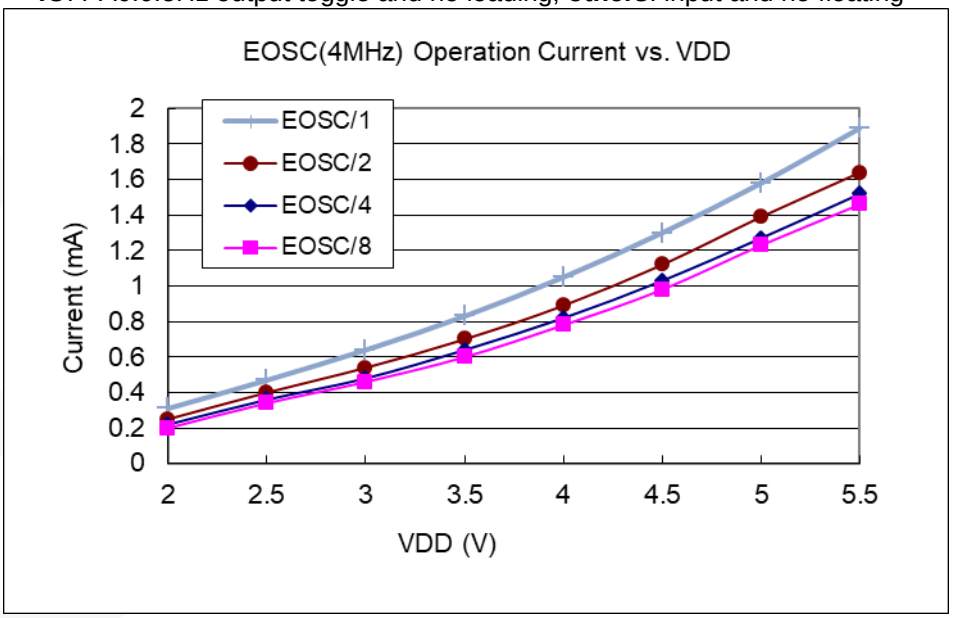
14.10. Typical Operating Current vs. VDD and CLK=1MHz EOSC / n

Conditions: **ON**: EOSC, Band-gap; **OFF**: LVR, T16 modules, IHRC, ILRC modules;
IO: PA0:0.5Hz output toggle and no loading, **others**: input and no floating

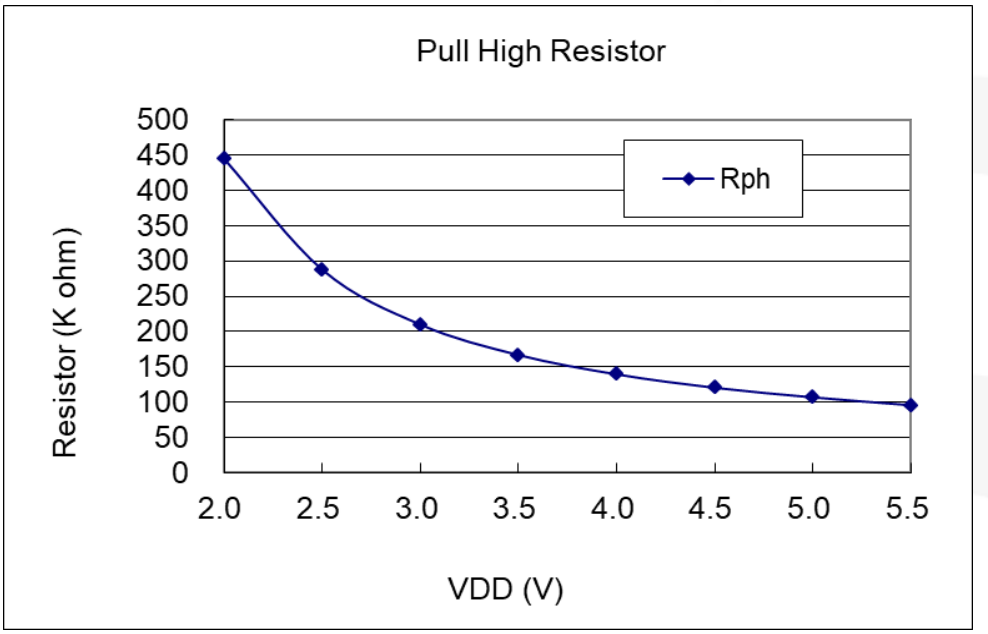


14.11. Typical Operating Current vs. VDD and CLK=4MHz EOSC / n

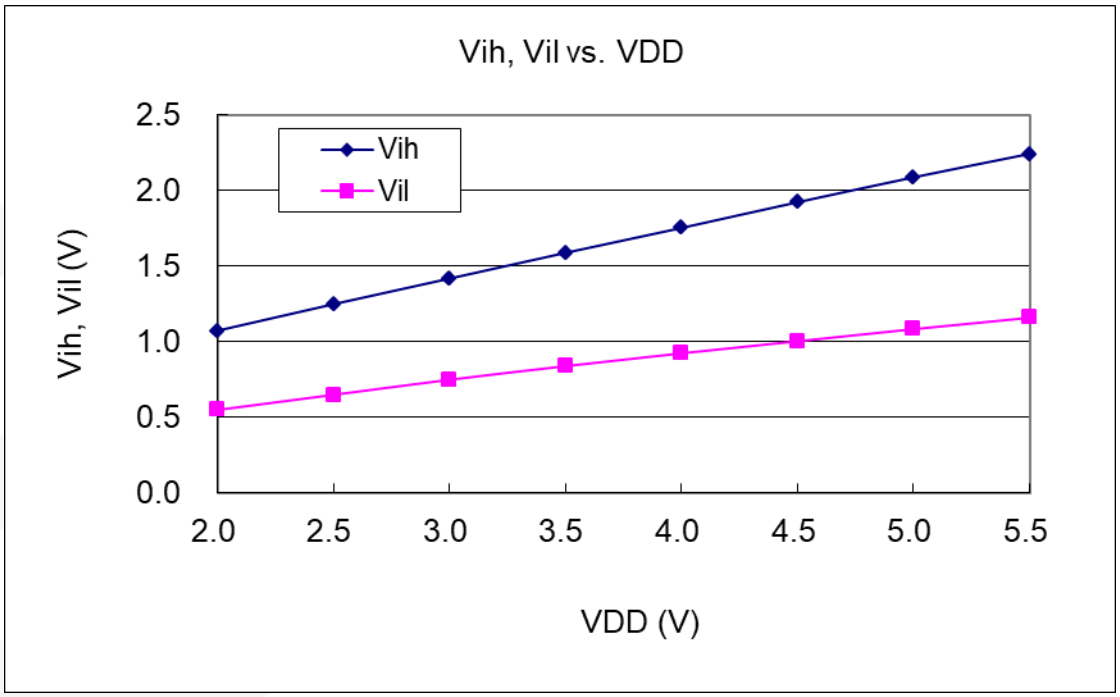
Conditions: **ON**: EOSC, Band-gap; **OFF**: LVR, T16 modules, IHRC, ILRC modules;
IO: PA0:0.5Hz output toggle and no loading, **others**: input and no floating



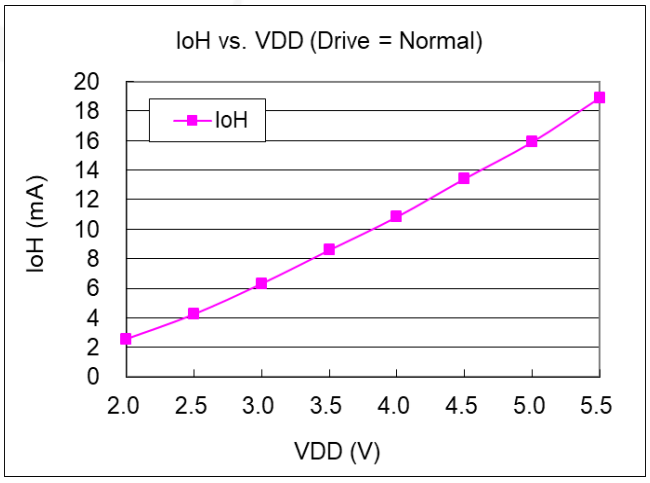
14.12. Typical IO pull high resistance

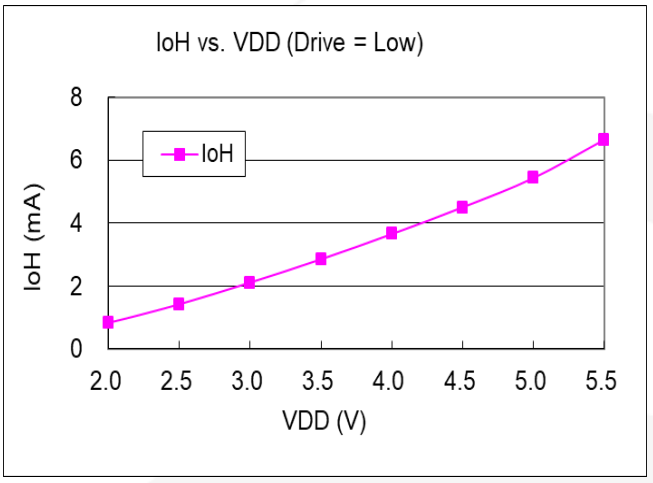


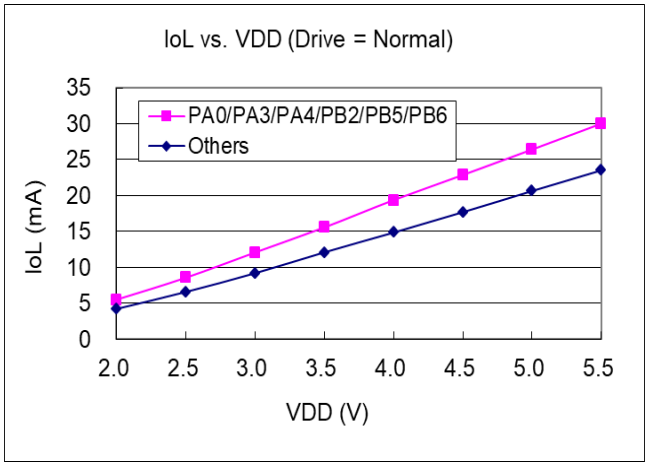
14.13. Typical IO input high/low threshold voltage (V_{IH}/V_{IL})

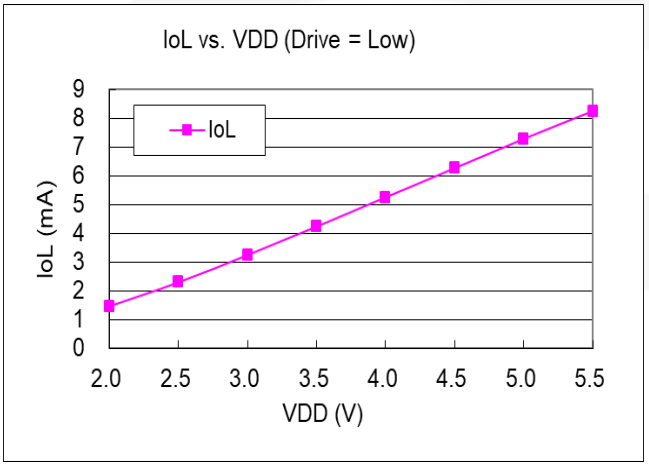


14.14. Typical IO driving current (I_{OH}) and sink current (I_{OL})

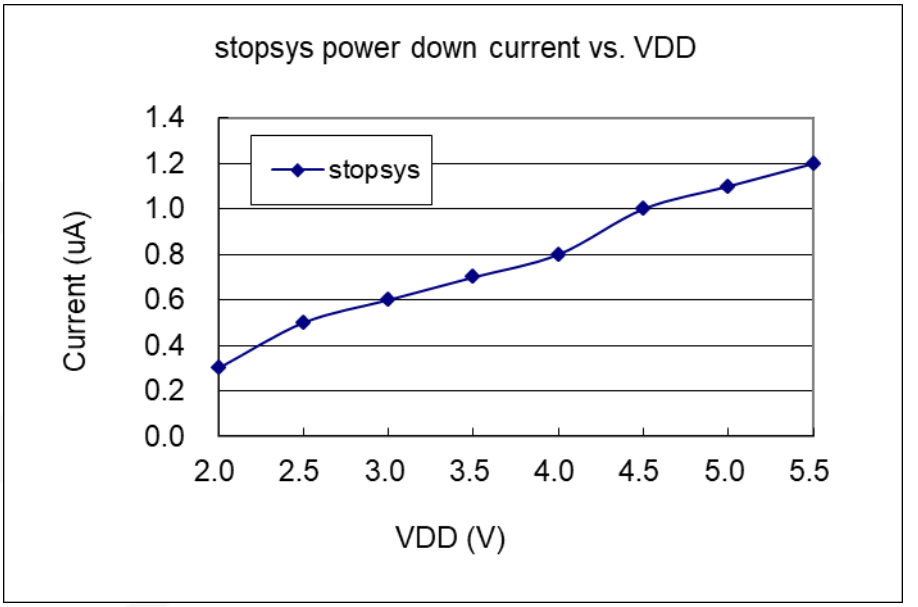


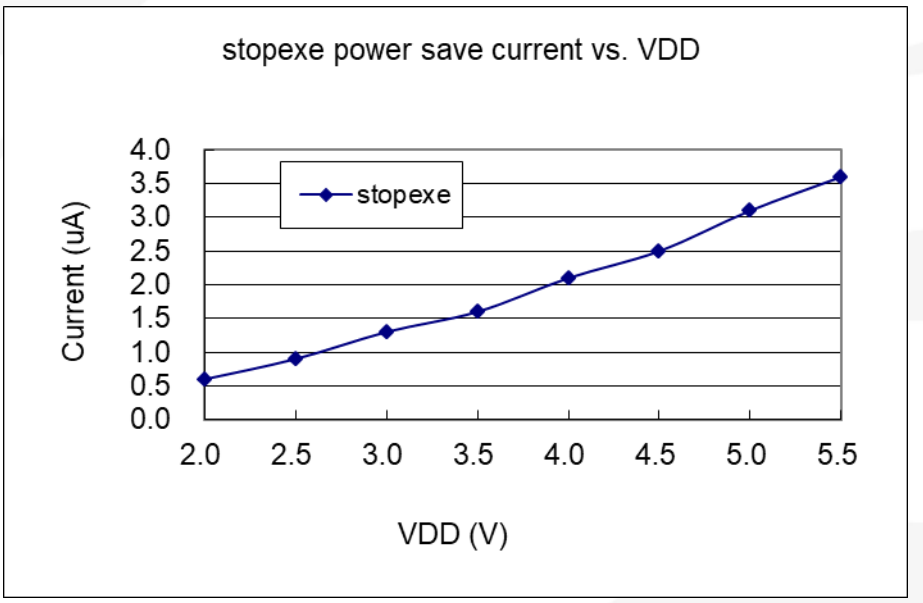






14.15. Typical power down current (I_{PD}) and power save current (I_{PS})





15. Instructions

Symbol	Description
ACC	Accumulator (Abbreviation of accumulator)
a	Accumulator (Symbol of accumulator in program)
sp	Stack pointer
flag	ACC status flag register
I	Immediate data
&	Logical AND
|	Logical OR
←	Movement
^	Exclusive logic OR
+	Add
−	Subtraction
∽	NOT (logical complement, 1's complement)
∓	NEG (2's complement)
OV	Overflow (The operational result is out of range in signed 2's complement number system)
Z	Zero (If the result of *ALU* operation is zero, this bit is set to 1)
C	Carry (The operational result is to have carry out for addition or to borrow carry for subtraction in unsigned number system)
AC	Auxiliary Carry (If there is a carry out from low nibble after the result of *ALU* operation, this bit is set to 1)
IO.n	The bit of register
M.n,	Only addressed in 0~0x3F (0~63) is allowed

15.1. Instruction Table

Instructions	Function	Cycles	Z	C	AC	OV
Data Transfer Instructions						
mov a, I	*mov* a, 0x0f; a ← 0fh;	1	-	-	-	-
mov M, a	*mov* MEM, a; MEM ← a	1	-	-	-	-
mov a, M	*mov* a, MEM ; a ← MEM; Flag Z is set when MEM is zero.	1	Y	-	-	-
mov a, IO	*mov* a, pa ; a ← pa; Flag Z is set when pa is zero.	1	Y	-	-	-
mov IO, a	*mov* pb, a; pb ← a ;	1	-	-	-	-
ldt16 word	*ldt16* word; word ← 16-bit timer	1	-	-	-	-
stt16 word	*stt16* word; 16-bit timer ← word	1	-	-	-	-
idxm a, index	*idxm* a, index; a ← [index], where index is declared by word.	**2**	-	-	-	-
idxm index, a	*idxm* index, a; [index] ← a; where index is declared by word.	**2**	-	-	-	-
xch M	*xch* MEM ; MEM ← a , a ← MEM	1	-	-	-	-
pushaf	*pushaf*; [sp] ← {flag, ACC}; sp ← sp + 2 ;	1	-	-	-	-
POPAF	*popaf;* sp ← sp - 2 ; {Flag, ACC} ← [sp] ;	1	Y	Y	Y	Y
Arithmetic Operation Instructions						
add a, I	*add* a, 0x0f ; a ← a + 0fh	1	Y	Y	Y	Y
add a, M	*add* a, MEM ; a ← a + MEM	1	Y	Y	Y	Y
add M, a	*add* MEM, a; MEM ← a + MEM	1	Y	Y	Y	Y
addc a, M	*addc* a, MEM ; a ← a + MEM + C	1	Y	Y	Y	Y
addc M, a	*addc* MEM, a ; MEM ← a + MEM + C	1	Y	Y	Y	Y
addc a	*addc* a ; a ← a + C	1	Y	Y	Y	Y
addc M	*addc* MEM ; MEM ← MEM + C	1	Y	Y	Y	Y
nadd a, M	*nadd* a, MEM ; a ← ∓a + MEM	1	Y	Y	Y	Y
nadd M, a	*nadd* MEM, a; MEM ← ∓MEM + a	1	Y	Y	Y	Y
sub a, I	*sub* a, 0x0f; a ← a - 0fh (a + [2's complement of 0fh])	1	Y	Y	Y	Y
sub a, M	*sub* a, MEM ; a ← a - MEM (a + [2's complement of M])	1	Y	Y	Y	Y
sub M, a	*sub* MEM, a; MEM ← MEM - a (MEM + [2's complement of a])	1	Y	Y	Y	Y
subc a, M	*subc* MEM, a ; a ← a – MEM - C	1	Y	Y	Y	Y
subc M, a	*subc* MEM, a ; MEM ← MEM – a - C	1	Y	Y	Y	Y
subc a	*subc* a; a ← a - C	1	Y	Y	Y	Y
subc M	*subc* MEM; MEM ← MEM - C	1	Y	Y	Y	Y
inc M	*inc* MEM ; MEM ← MEM + 1	1	Y	Y	Y	Y
dec M	*dec* MEM; MEM ← MEM - 1	1	Y	Y	Y	Y
clear M	*clear* MEM ; MEM ← 0	1	-	-	-	-

Instructions	Function	Cycle	Z	C	AC	OV
Shift Operation Instructions						
sr a	*sr* a ; a (0,b7,b6,b5,b4,b3,b2,b1) ← a (b7,b6,b5,b4,b3,b2,b1,b0), C ← a(b0)	1	-	Y	-	-
src a	*src* a ; a (c,b7,b6,b5,b4,b3,b2,b1) ← a (b7,b6,b5,b4,b3,b2,b1,b0), C ← a(b0)	1	-	Y	-	-
sr M	*sr* MEM ; MEM(0,b7,b6,b5,b4,b3,b2,b1) ← MEM(b7,b6,b5,b4,b3,b2,b1,b0), C ← MEM(b0)	1	-	Y	-	-
src M	*src* MEM ; MEM(c,b7,b6,b5,b4,b3,b2,b1) ← MEM (b7,b6,b5,b4,b3,b2,b1,b0), C ← MEM(b0)	1	-	Y	-	-
sl a	*sl* a ; a (b6,b5,b4,b3,b2,b1,b0,0) ← a (b7,b6,b5,b4,b3,b2,b1,b0), C ← a (b7)	1	-	Y	-	-
slc a	*slc* a ; a (b6,b5,b4,b3,b2,b1,b0,c) ← a (b7,b6,b5,b4,b3,b2,b1,b0), C ← a(b7)	1	-	Y	-	-
sl M	*sl* MEM ; MEM (b6,b5,b4,b3,b2,b1,b0,0) ← MEM (b7,b6,b5,b4,b3,b2,b1,b0), C ← MEM(b7)	1	-	Y	-	-
slc M	*slc* MEM ; MEM (b6,b5,b4,b3,b2,b1,b0,C) ← MEM (b7,b6,b5,b4,b3,b2,b1,b0), C ← MEM (b7)	1	-	Y	-	-
swap a	*swap* a ; a (b3,b2,b1,b0,b7,b6,b5,b4) ← a (b7,b6,b5,b4,b3,b2,b1,b0)	1	-	-	-	-
Logic Operation Instructions						
and a, I	*and* a, 0x0f ; a ← a & 0fh	1	Y	-	-	-
and a, M	*and* a, RAM10 ; a ← a & RAM10	1	Y	-	-	-
and M, a	*and* MEM, a ; MEM ← a & MEM	1	Y	-	-	-
or a, I	*or* a, 0x0f ; a ← a | 0fh	1	Y	-	-	-
or a, M	*or* a, MEM ; a ← a | MEM	1	Y	-	-	-
or M, a	*or* MEM, a ; MEM ← a | MEM	1	Y	-	-	-
xor a, I	*xor* a, 0x0f ; a ← a ^ 0fh	1	Y	-	-	-
xor IO, a	*xor* *pa, a ;* pa ← a ^ pa ;	1	-	-	-	-
xor a, M	*xor* a, MEM ; a ← a ^ RAM10	1	Y	-	-	-
xor M, a	*xor* MEM, a ; MEM ← a ^ MEM	1	Y	-	-	-
not a	*not* a ; a ← ∽a	1	Y	-	-	-
not M	*not* MEM ; MEM ← ∽MEM	1	Y	-	-	-
neg a	*neg* a; a ← ∓a	1	Y	-	-	-
neg M	*neg* MEM; MEM ← ∓MEM	1	Y	-	-	-
comp a, M	*comp* a,MEM; Flag will be changed by regarding as (a - MEM)	1	Y	Y	Y	Y
comp M, a	*comp* MEM,a; Flag will be changed by regarding as (MEM - a)	1	Y	Y	Y	Y

Instructions	Function	Cycles	Z	C	AC	OV
Bit Operation Instructions						
set0 IO.n	*set0* pa.5 ; PA5=0	1	-	-	-	-
set1 IO.n	*set1* pb.5 ; PB5=1	1	-	-	-	-
set0 M.n	*set0* MEM.5 ; set bit 5 of MEM to low	1	-	-	-	-
set1 M.n	*set1* MEM.5 ; set bit 5 of MEM to high	1	-	-	-	-
swapc IO.n	*swapc* IO.0; C ← IO.0 , IO.0 ← 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	-	-
Conditional Operation Instructions						
ceqsn a, I	*ceqsn* a, 0x55 ; *inc* MEM ; *goto* error ; If a=0x55, then “goto error”; otherwise, “inc MEM”.	**1 / 2**	Y	Y	Y	Y
ceqsn a, M	*ceqsn* a, MEM; If a=MEM, skip next instruction	**1 / 2**	Y	Y	Y	Y
cneqsn a, M	*cneqsn* a, MEM; If a≠MEM, skip next instruction	**1 / 2**	Y	Y	Y	Y
cneqsn a, I	*cneqsn* a, 0x55 ; *inc* MEM ; *goto* error ; If a≠0x55, then “goto error”; Otherwise, “inc MEM”	**1 / 2**	Y	Y	Y	Y
t0sn IO.n	*t0sn* pa.5; If bit 5 of port A is low, skip next instruction	**1 / 2**	-	-	-	-
t1sn IO.n	*t1sn* pa.5; If bit 5 of port A is high, skip next instruction	**1 / 2**	-	-	-	-
t0sn M.n	*t0sn* MEM.5 ; If bit 5 of MEM is low, then skip next instruction	**1 / 2**	-	-	-	-
t1sn M.n	*t1sn* MEM.5 ; If bit 5 of MEM is high, then skip next instruction	**1 / 2**	-	-	-	-
izsn a	*izsn* a; a ← a + 1,skip next instruction if a = 0	**1 / 2**	Y	Y	Y	Y
dzsn a	*dzsn* a; a ← a – 1，skip next instruction if a = 0	**1 / 2**	Y	Y	Y	Y
izsn M	*izsn* MEM; MEM ← MEM + 1，skip next instruction if MEM= 0	**1 / 2**	Y	Y	Y	Y
dzsn M	*dzsn* MEM; MEM ← MEM - 1，skip next instruction if MEM= 0	**1 / 2**	Y	Y	Y	Y
System Control Instructions						
call label	*call* function1; [sp] ← pc + 1, pc ← function1, sp ← sp + 2	**2**	-	-	-	-
goto label	*goto* routine1; Go to routine1 and execute program.	**2**	-	-	-	-
ret I	*ret* 0x55; A ← 55h ret ;	**2**	-	-	-	-
ret	*ret;* sp ← sp - 2 pc ← [sp]	**2**	-	-	-	-
reti	*reti*; Return to program that is interrupt service routine. After this command is executed, global interrupt is enabled automatically.	**2**	-	-	-	-
nop	*nop*; Nothing changed.	1	-	-	-	-
pcadd a	*pcadd a*; pc ← pc + a	**2**	-	-	-	-
engint	*engint*; Interrupt request can be sent to FPP0	1	-	-	-	-
disgint	*disgint* ; Interrupt request is blocked from FPP0	1	-	-	-	-
stopsys	*stopsys*; Stop the system clocks and halt the system	1	-	-	-	-
stopexe	*stopexe*; Stop the system clocks and keep oscillator modules active.	1	-	-	-	-
reset	*reset*; Reset the whole chip.	1	-	-	-	-
wdreset	*wdreset* ; Reset Watchdog timer.	1	-	-	-	-