PADAUK 應廣科技

LeapDragon（跃龙）

PFC161

Industrial Grade - 7 Touch Keys 8bit MTP MCU

Data Sheet

Version 0.04

February 20, 2024

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.

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

Table of Contents

Revision History

Revision	Date	Description
0.03	2023/02/09	1. Updated “IMPORTANT NOTICE” 2. Add V_{BG} (Section 14.2) 3. Amend Section 9.1.1 and Chapter 12 4. Other known details bug correct.
0.04	2024/02/06	1. Revised DFN8. 2. Modify the value of EOSCR & Scalar. 3. Other known details bug correct.

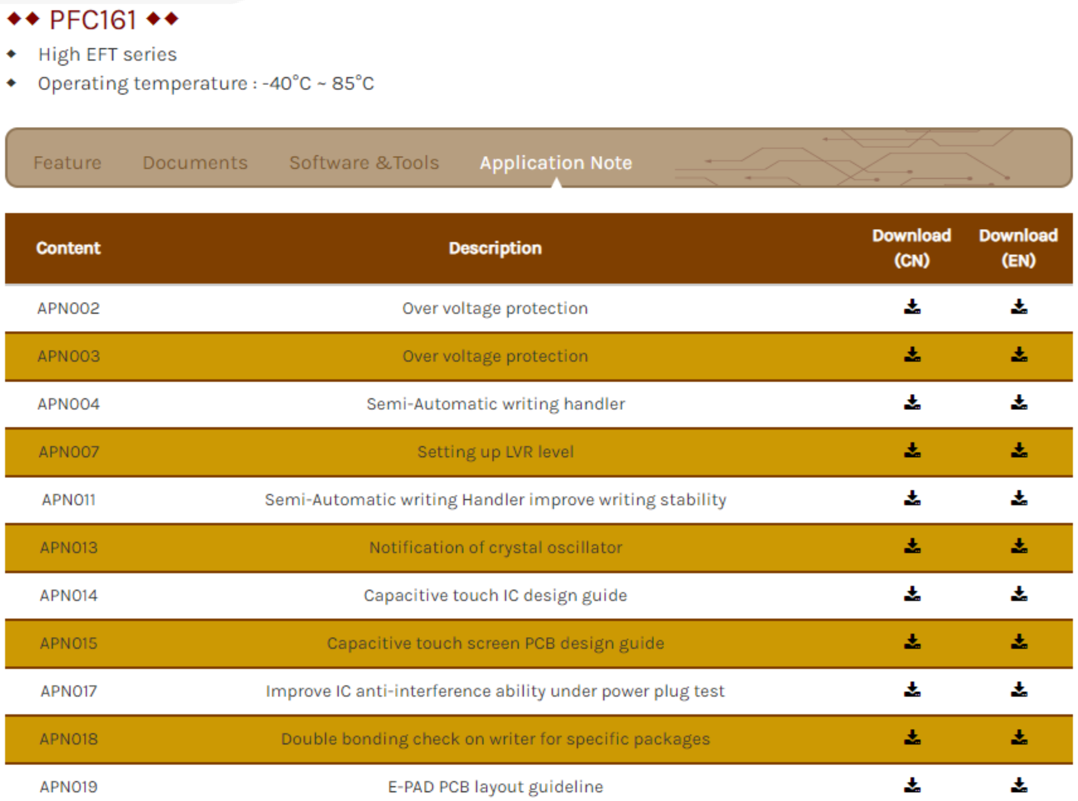
Usage Warning

User must read all application notes of the IC by detail before using it.

Please visit the official website to download and view the latest APN information associated with it.

http://www.padauk.com.tw/en/product/show.aspx?num=117&kw=PFC161

(The following picture are for reference only.)



◆◆ PFC161 ◆◆

- High EFT series
- Operating temperature : -40°C ~ 85°C

Feature Documents Software &Tools Application Note

Content	Description	Download (CN)	Download (EN)
APN002	Over voltage protection		
APN003	Over voltage protection		
APN004	Semi-Automatic writing handler		
APN007	Setting up LVR level		
APN011	Semi-Automatic writing Handler improve writing stability		
APN013	Notification of crystal oscillator		
APN014	Capacitive touch IC design guide		
APN015	Capacitive touch screen PCB design guide		
APN017	Improve IC anti-interference ability under power plug test		
APN018	Double bonding check on writer for specific packages		
APN019	E-PAD PCB layout guideline		

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
- ESD > 8 KV

1.2. System Features

- 2KW MTP program memory (programming cycle at least 1,000 times)
- 128 Bytes data RAM
- Maximum 7 IO pins can be selected as TOUCH PAD individually
- 8 IO pins with optional pull-high / pull-low resistor
- Every IO pin can be configured to enable wake-up function
- One hardware 16-bit timer
- Two hardware 8-bit timer with PWM generators
- One hardware comparator
- Bandgap circuit to provide 1.20V Bandgap voltage
- Clock sources: IHRC, ILRC & EOSC(XTAL mode)
- LVR (Low Voltage Reset) is selectable from 1.8V ~ 4.5V, total 8 levels
- LVD (Low Voltage Detect) is selectable from 1.8V ~ 4.5V, total 16 levels
- Three selectable external interrupt pins

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
- All data memories are available for use as an index pointer
- Register space, memory space and MTP space are independent

1.4. Ordering/ Package Information

- PFC161-U06: SOT23-6 (60mil)
- PFC161-S08A: SOP8A (150mil)
- PFC161-S08B: SOP8B (150mil)
- PFC161-2N08: DFN8 (2*2mm)
- PFC161-EY10: ESSOP10 (150mil)

- Please refer to the official website file for package size information: ”Package information “

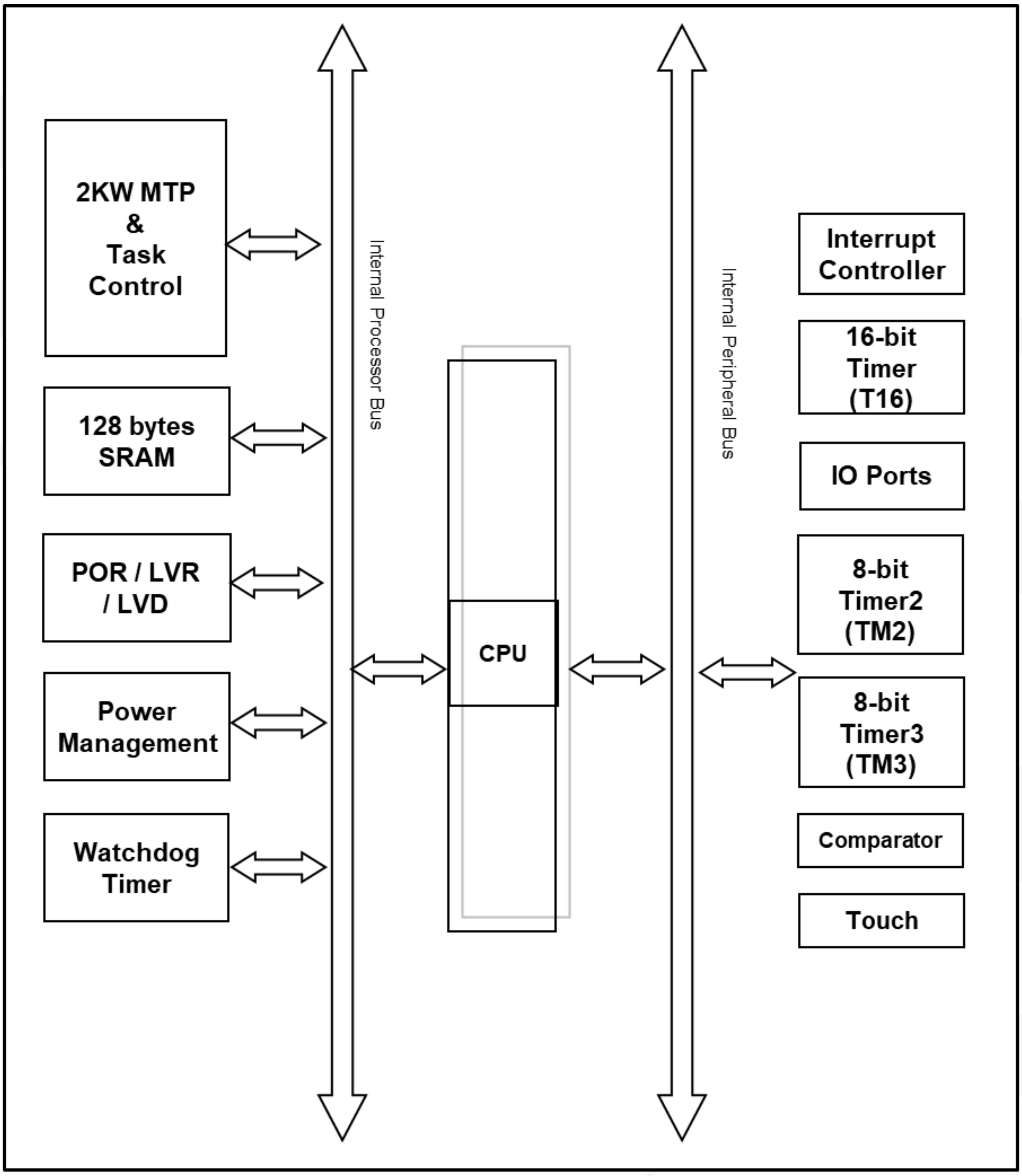
2. General Description and Block Diagram

The PFC161 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.

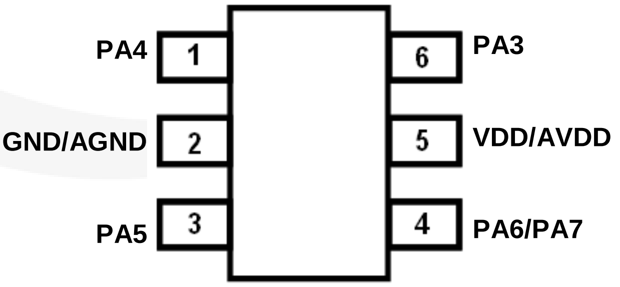
PFC161 has built-in 2KW MTP program memory and 128 byte data storage.

PFC161 has built-in a maximum 7 keys touch controller and a hardware comparator.

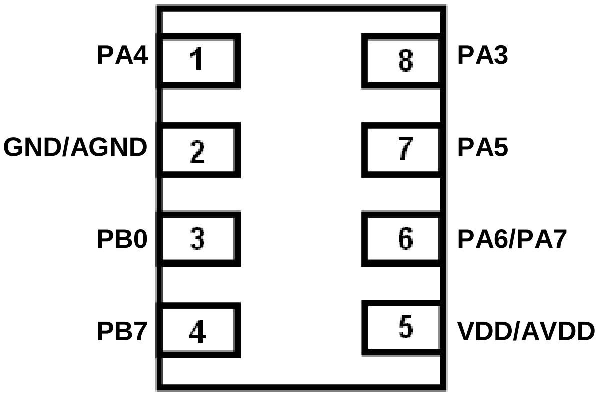
PFC161 provides one hardware 16-bit timer, two hardware 8-bit timers with PWM generation (Timer2, Timer3).



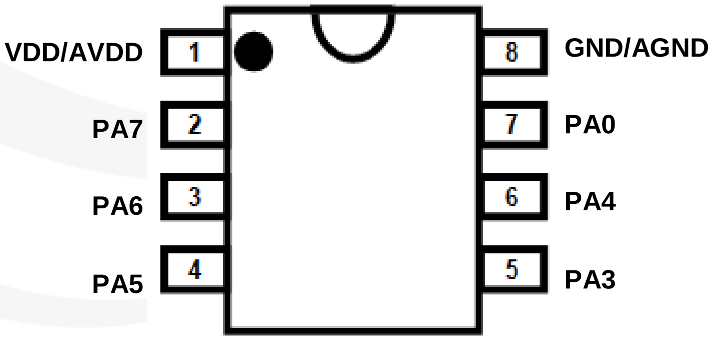
3. Pin Definition and Functional Description



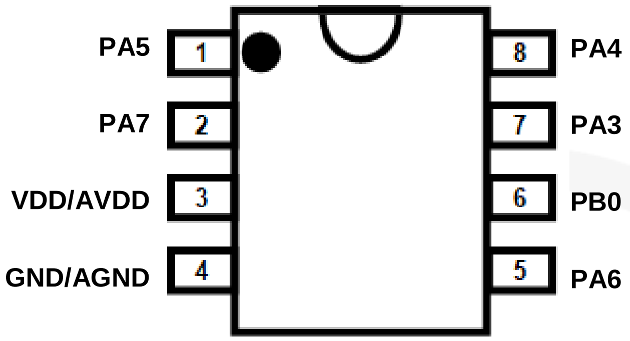
PFC161-U06 (SOT23-6 60mil)



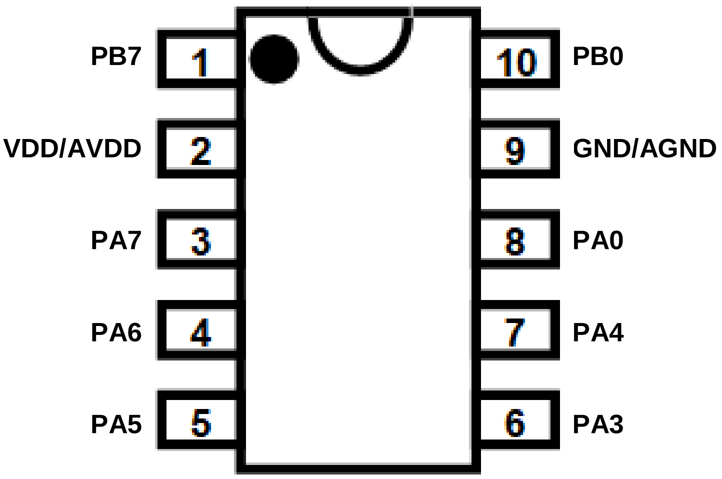
PFC161-2N08 (DFN8-2*2mm)



PFC161-S08A(SOP8A-150mil)



PFC161-S08B(SOP8B-150mil)



PFC161-EY10 (ESSOP10-150mil)

Note: In the PFC161-U06/2N08 chip, the two IO PA6/PA7 share the same pin, so it is not allowed to output 0/1 signal at the same time when using PA6/PA7, otherwise the pin will be short-circuited.

Pin Name	Input / output				Special Functions							
	I / O	Pull High	Pull Low	Wake Up	Crystal	Comparator	PWM	Touch Key	CS Capacitor	External Interrupt	External Reset	Program
PA0	√	√	√	√		CO	TM2PWM	TK7		INT0		
PA3	√	√	√	√		CIN-	TM2PWM	TK5				√
PA4	√	√	√	√		CIN+ CIN-	TM3PWM	TK6				
PA5	√	√	√	√			TM3PWM	TK9		INT0A	√	√
PA6	√	√	√	√	√			TK8				√
PA7	√	√	√	√	√			TK10	√			
PB0	√	√	√	√			TM2PWM	TK11		INT1		
PB7	√	√	√	√		CIN-	TM3PWM	TK4	√			
VDD / AVDD												√
GND / AGND												√
Notes	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. VDD is the IC power supply while AVDD is the Analog positive power supply. AVDD and VDD are double bonding internally and they have the same external pin. 5. GND is the IC ground pin while AGND is the Analog negative ground pin. AGND and GND are double bonding internally and they have the same external pin.											

4. Central Processing Unit (CPU)

4.1. Storage Memory

4.1.1. Program Memory – ROM

The PFC161 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 PFC161 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

PFC161 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 PFC161 is independent of SRAM space and MTP space.

The following is the PFC161 register address and brief description:

	+0	+1	+2	+3	+4	+5	+6	+7
0x00	*FLAG*	-	*SP*	*CLKMD*	*INTEN*	*INTRQ*	*T16M*	-
0x08	*MISC*	-	*EOSCR*	-	*INTEGS*	*PADIER*	*PBDIER*	-
0x10	*PA*	*PAC*	*PAPH*	*PAPL*	*PB*	*PBC*	*PBPH*	*PBPL*
0x18	*GPCC*	*GPCS*	-	-	*TM2C*	*TM2CT*	*TM2S*	*TM2B*
0x20	*TS*	*TCC*	*TKE2*	-	*TKE1*	-	-	-
0x28	*TPS2*	-	-	*TKCH*	*TKCL*	*LVDC*	-	-
0x30	-	-	*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

PAPL: Port A Pull-Low Registers

PADIER: Port A Digital Input Enable Register

PB: Port B Data Registers

PBC: Port B Control Registers

PBPH: Port B Pull-High Registers

PBPL: Port B Pull-Low 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

TS: Touch Selection Register

TCC: Touch Charge Control Register

TKE1 / TKE2: Touch Key Enable Register

TKCH / TKCL:

Touch Key Charge Counter High/Low Register

LVDC: Low Voltage Detect Register

TPS2: Touch Parameter Setting 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 - 6	-	-	Reserved
5	0	WO	Wake-up time 0 / 1 : 3000 ILRC (Slow) / 45 ILRC (Fast)
4 - 3	-	-	Reserved
2	0	WO	Disable LVR function. 0 / 1 : Enable / Disable
1 - 0	00	WO	Watchdog 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 PFC161 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(default)**	MTP content is protected 7/8 words
	Disable	MTP content is not protected so program can be read back
LVR	4.0V	LVR typical range 4.0V
	3.5V	LVR typical range 3.5V
	3.0V	LVR typical range 3.0V
	2.7V	LVR typical range 2.7V
	2.5V(default)	LVR typical range 2.5V
	2.2V	LVR typical range 2.2V
	2.0V	LVR typical range 2.0V
	1.8V	LVR typical range 1.8V
PA3_PA4_Drive	Strong	PA3 & PA4 Drive/Sink current is Strong
	Normal(default)	PA3 & PA4 Drive/Sink current is Normal
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 doesn't 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 doesn't support.)
Comparator Edge	**All Edge(default)**	GPC INT both Rising & Falling edge trigger
	Rising Edge	GPC INT both Rising edge trigger
	Falling Edge	GPC INT both Falling edge trigger
GPC_PWM	**Disable(default)**	Comparator does not control all PWM outputs
	Enable	Comparator controls all PWM outputs (ICE doesn't support.)
Interrupt Src0	**PA.0(default)**	INTEN/INTRQ.Bit0 is from PA.0
	PA.5	INTEN/INTRQ.Bit0 is from PA.5 (ICE doesn't support.)
CS_Sel	**PA7(default)**	Configure pad PA7/CS as CS pad of Touch
	PB7	Configure pad PB7/CS as CS pad of Touch
	Disable	Configure pad PA7/PB7/CS as normal PA7/PB7 IO pad
EMI	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 PFC161: 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 PFC161

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 PFC161 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.

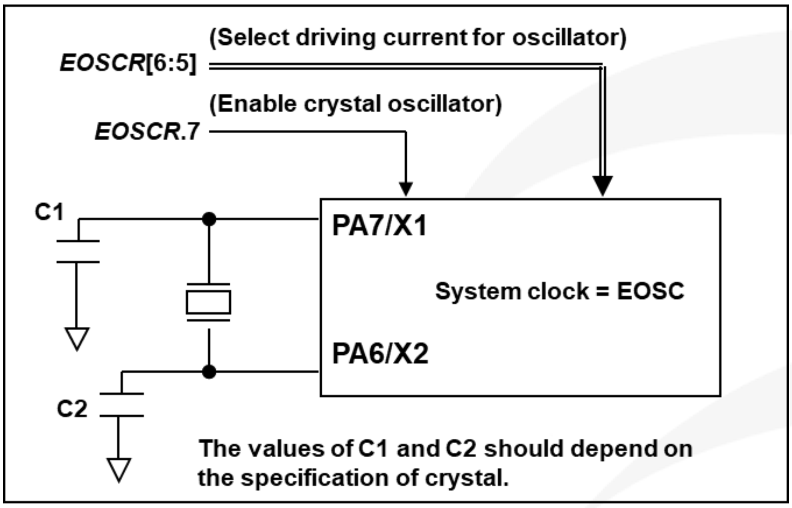


Fig. 1: Connection of crystal oscillator

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 – 0	-	-	Reserved. Please keep 0 for future compatibility.

5.2.2. Usages and Precautions of External Oscillator

Besides crystal, external capacitor and options of PFC161 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 = 0b111_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 PFC161 is shown as Fig. 2.

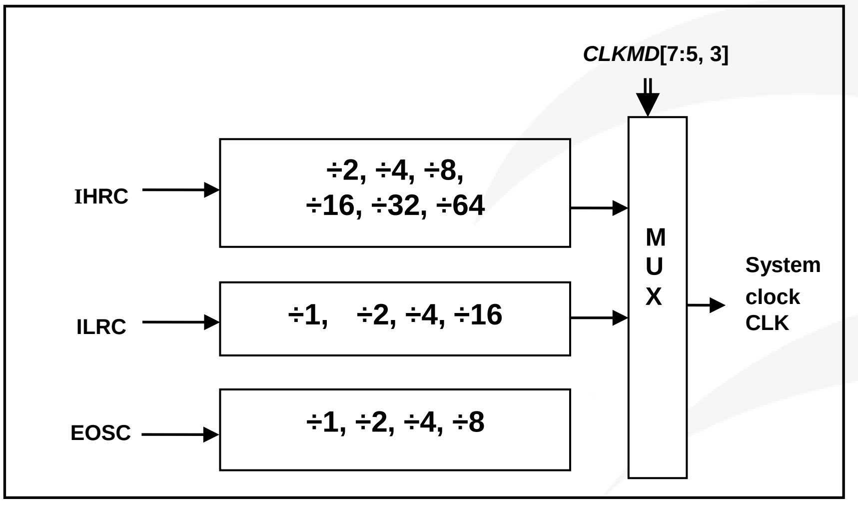


Fig. 2: Options of System Clock

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/8 110: ILRC/2 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; 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 PFC151 under different options:

(1) . ADJUST_IC SYSCLK=IHRC/2, IHRC=16MHz, V_{DD}=5V

After boot, *CLKMD* = 0x34:

a. IHRC frequency is calibrated to 16MHz@V_{DD}=5V and IHRC module is enabled
b. System CLK = IHRC/2 = 8MHz
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 PFC161 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/2

```
…                              //   system clock is ILRC
CLKMD.4     =   1;             //   turn on IHRC first to improve anti-interference ability
CLKMD       =   0x34;          //   switch to IHRC/2, ILRC CAN NOT be disabled here
//  CLKMD.2 =   0;             //   if need, ILRC CAN be disabled at this time
…
```

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

```
…                              //   system clock is IHRC/2
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/2 to IHRC/4

```
…                              //   system clock is IHRC/2, 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       =   0x30;          //   CAN NOT switch clock from ILRC to IHRC/2 and turn off
                               //   ILRC oscillator at the same time
…
```

6. Reset and Power Detect

PFC161 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 PFC161 will be set to the default value.

6.1. Power On Reset - POR

POR (Power-On-Reset) is used to reset PFC161 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 PFC161 data memory is in an uncertain state when the power on reset occurs.

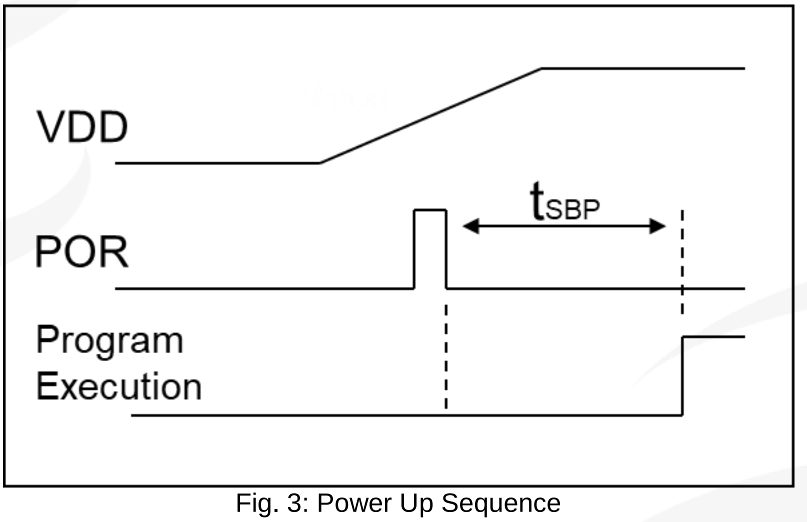


Fig. 3: Power Up Sequence

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.

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

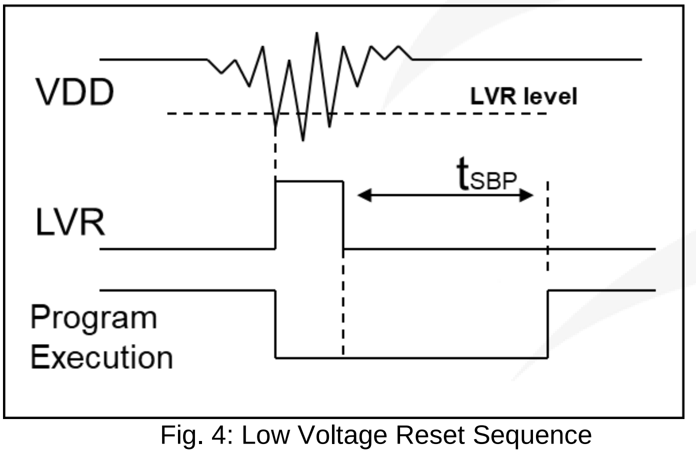


Fig. 4: Low Voltage Reset Sequence

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

(1) The setting of LVR (1.8V ~ 4.0V) will be valid just after successful power-on process.
(2) User can set MISC.2 as “1” to disable LVR. However, V_{DD} must be kept as exceeding the lowest working voltage of chip; Otherwise IC may work abnormally.
(3) The LVR function will be invalid when IC in stopexe or stopsys mode.

MISC Register (*MISC*), address = 0x08			
Bit	**Reset**	**R/W**	**Description**
7 – 6	-	-	Reserved.
5	0	WO	Wake up time.
4 - 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, PFC161 will be reset to restart the program execution. The relative timing diagram of watchdog timer is shown as Fig. 5.

The PFC161 data memory will be reserved when the WDT reset occurs.

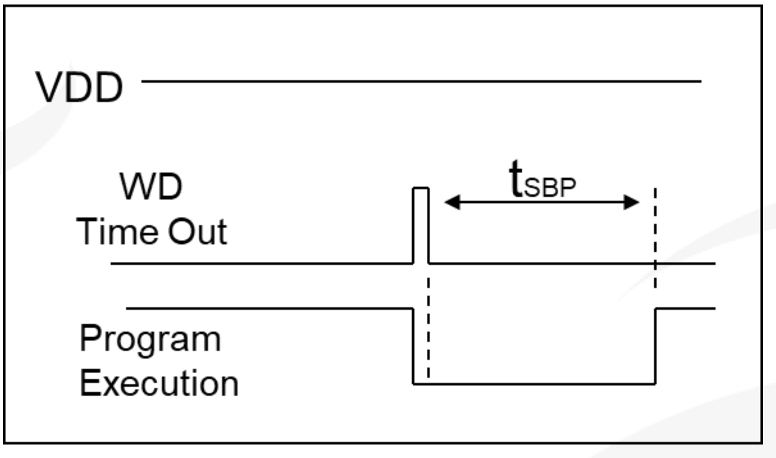


Fig. 5: Sequence of Watch Dog Timeout reset

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 PFC161 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 PFC161 data memory will be reserved when the PRSTB reset occurs.

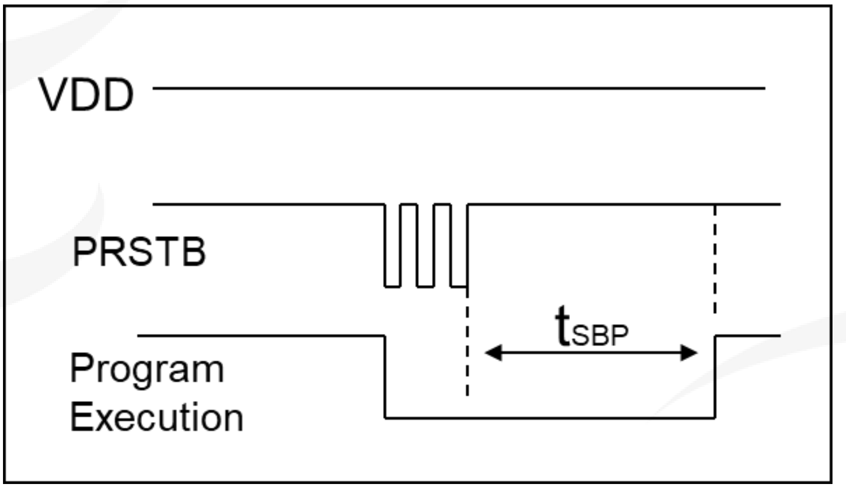


Fig. 6: Sequence of PRSTB reset

6.5. System Power Voltage Detector – LVD

PFC161 is equipped with a VDD power voltage detector to quickly detect the IC working voltage level.

With the *LVDC*[7:4] register, the user can select one of 16 levels from 1.8V to 4.5V to compare with the IC VDD voltage. By reading the compare result from *LVDC*.0, user can monitor the system power to see if it is under the preset level.

6.5.1. Low Voltage Detect Register (*LVDC*), address = 0x2D

Bit	Reset	R/W	Description
7 - 4	0000	WO	LVD voltage level selection. 0000 : 1.8V　　1000 : 2.75V 0001 : 1.9V　　1001 : 3.0V 0010 : 2.0V　　1010 : 3.15V 0011 : 2.1V　　1011 : 3.3V 0100 : 2.2V　　1100 : 3.5V 0101 : 2.3V　　1101 : 3.75V 0110 : 2.4V　　1110 : 4.0V 0111 : 2.5V　　1111 : 4.5V
3 - 1	-	-	Reserved.
0	-	RO	VDD voltage detects result. 0: VDD > LVD 1: VDD < LVD

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

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 recommending 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 PFC161 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 is 1)

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

```
CLKMD   =   0xF4;        //  Change clock from IHRC to ILRC, disable watchdog timer
CLKMD.4 =   0;           //  disable IHRC
…
while (1)
{
        STOPSYS;         //  enter Power-Down mode
        if (…) break;    //  if wake-up happen and check OK, then return to high speed,
                         //  else stay in Power-Down mode again.
}
CLKMD   =   0x34;        //  Change clock from ILRC to IHRC/2
```

7.3. Wake-Up

After entering the Power-Down or Power-Save modes, the PFC161 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	Comparator wake-up
stopsys	Yes	No	No
stopexe	Yes	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 PFC161, registers *PxDIER* should be properly set to enable the wake-up function for every corresponding pin. The time for normal wake-up is about 3000 ILRC clocks counting from wake-up event; fast wake-up can be selected to reduce the wake-up time by *MISC* register, and the time for fast wake-up is about 45 ILRC clocks from IO toggling.

Suspend mode	Wake-up mode	Wake-up time (t_{WUP}) from IO toggle
STOPEXE suspend or STOPSYS suspend	Fast wake-up	45 * T_{ILRC}, Where T_{ILRC} is the time period of ILRC
STOPEXE suspend or STOPSYS suspend	Normal wake-up	3000 * T_{ILRC}, Where T_{ILRC} is the clock period of ILRC

Table 7: Different wake-up time from IO toggle in fast / normal wake-up

8. Interrupt

There are eight interrupt lines for PFC161:

- External interrupt PA0 / PA5
- External interrupt PB0
- Timer16 interrupt
- Timer2 interrupt
- Timer3 interrupt
- GPC interrupt
- Two touch key interrupts (TK_OV and TK_END)

every interrupt request line to CPU has its own corresponding interrupt control bit to enable or disable it. The hardware diagram of interrupt controller is shown as Fig. 7. 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.

Note: the external interrupt source can be switched through Interrupt Src0 or Interrupt Src1 in the Code Option.

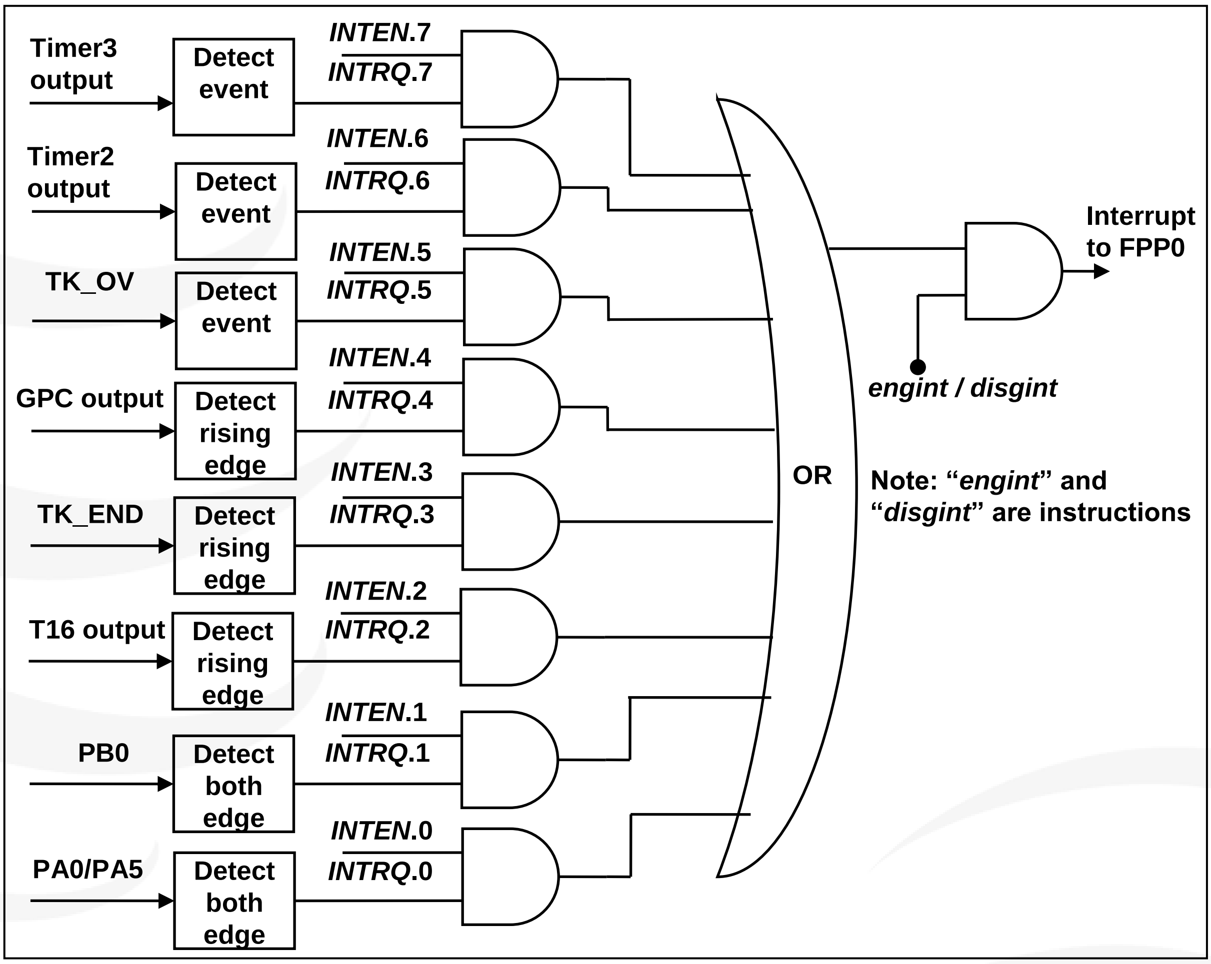


Fig. 7: Hardware diagram of Interrupt controller

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	-	-	Reserved. Please keep 0.
4	-	WO	Timer16 edge selection. 0 : rising edge to trigger interrupt 1 : falling edge to trigger interrupt
3 - 2	-	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	-	WO	PA0 / PA5 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 - 6	11	WO	Enable PA7~PA6 digital input and wake up event. 1 / 0 : enable / disable When using an external crystal oscillator, these bits need set to 0 to prevent power consumption.
5	1	WO	Enable PA5 digital input, wake up event and interrupt request. 1 / 0 : enable / disable
4 - 3	11	WO	Enable PA4~PA3 digital input and wake up event. 1 / 0 : enable / disable
2 - 1	-	-	Reserved. (Please keep 00 for future compatibility)
0	1	WO	Enable PA0 digital input, wake up event and interrupt request. 1 / 0 : enable / disable

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

Bit	Reset	R/W	Description
7	1	WO	Enable PB7 digital input and wake-up event. 1 / 0 : enable / disable. This bit can be set to low to disable wake-up from PB7 toggling.
6 - 1	-	-	Reserved.
0	1	WO	Enable PB0 digital input, wake-up event and interrupt request. 1 / 0 : enable / disable. This bit can be set to low to 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 - 0	0x00	R/W	Port A 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 A Pull-Low Registers (*PAPL*), address = 0x13

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

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

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

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

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

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

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

9.1.10. Port B Pull-Low Registers (*PBPL*), address = 0x17

Bit	Reset	R/W	Description
7 - 0	0x00	R/W	Port B pull-low registers. This register is used to enable the internal pull-low 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 PFC161 have the same structure. The hardware diagram of IO buffer is shown as Fig. 8.

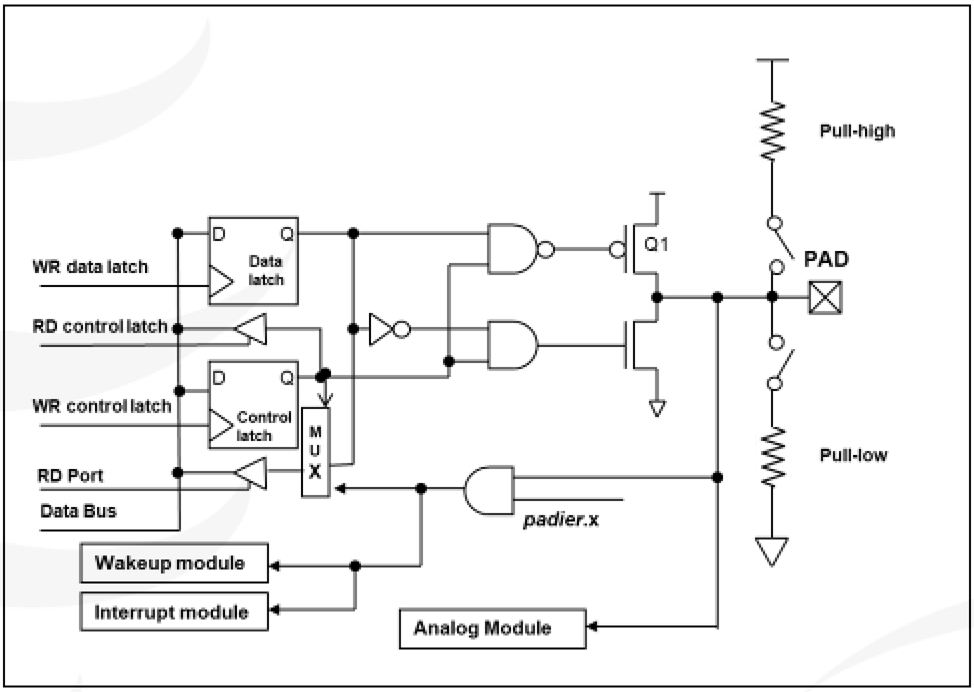


Fig. 8: Hardware diagram of IO buffer

9.2.2. IO Pin Functions

(1) Input / output function:

PFC161 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, pull-low registers (*PAPH/PBPH, PAPL/PBPL).*

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 / pull-low resistor is turned off automatically.

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

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

PA.0	*PAC*.0	*PAPH*.0	*PAPL*.0	Description
X	0	0	0	Input floating, without pull-high / pull-low resistor
X	0	1	0	Input with pull-high resistor
X	0	0	1	Input with pull-low resistor
X	0	1	1	Input with pull-high / pull-low resistor
0	1	X	X	Output low
1	1	X	X	Output high

Table 8: PA0 Configuration Table

(2) Wake-up function:

When PFC161 put in Power-Down or Power-Save mode, every IO pin can be used to wake-up system by toggling its state. Therefore, those pins needed to wake-up system must be set to input mode and set the corresponding bits of registers *PxDIER* to high.

(3) External interrupt function:

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

(4) Drive capability optional:

Parts of the IO can be adjusted their Driving or Sinking current capability to Normal or Strong 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 PRSTB input pin

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

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

- Needs to put a >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

PFC161 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.

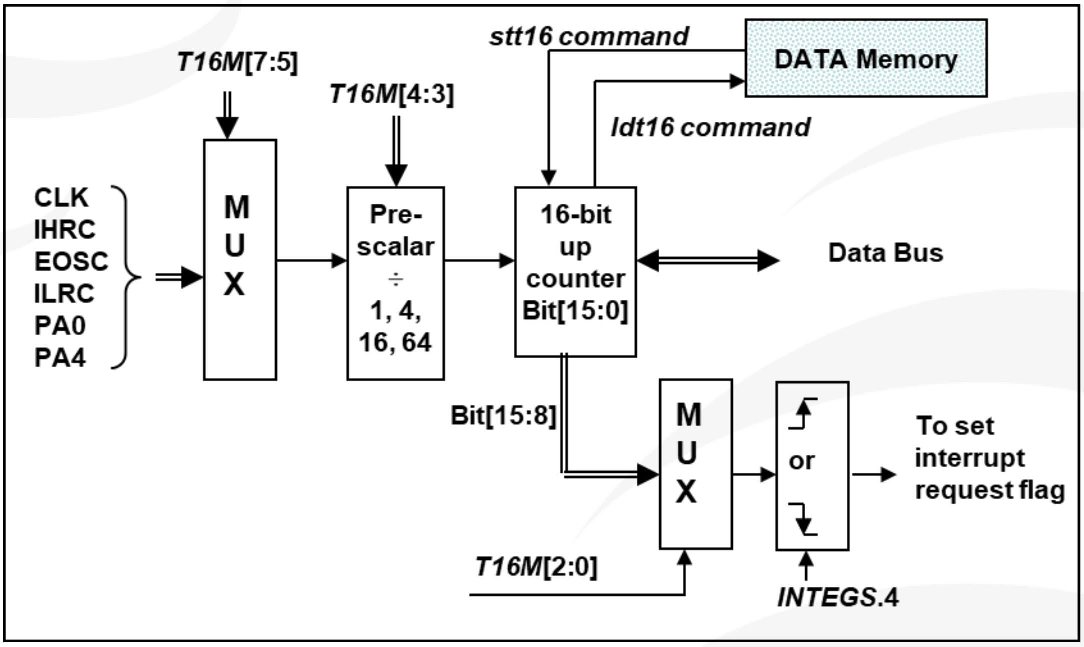


Fig. 9: Hardware diagram of Timer16

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 / 2 = 8 MHz
// SYSCLK/64 = 8 MHz/64 = 8 uS, about every 524 mS 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 PFC161. 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 [6:0] 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.

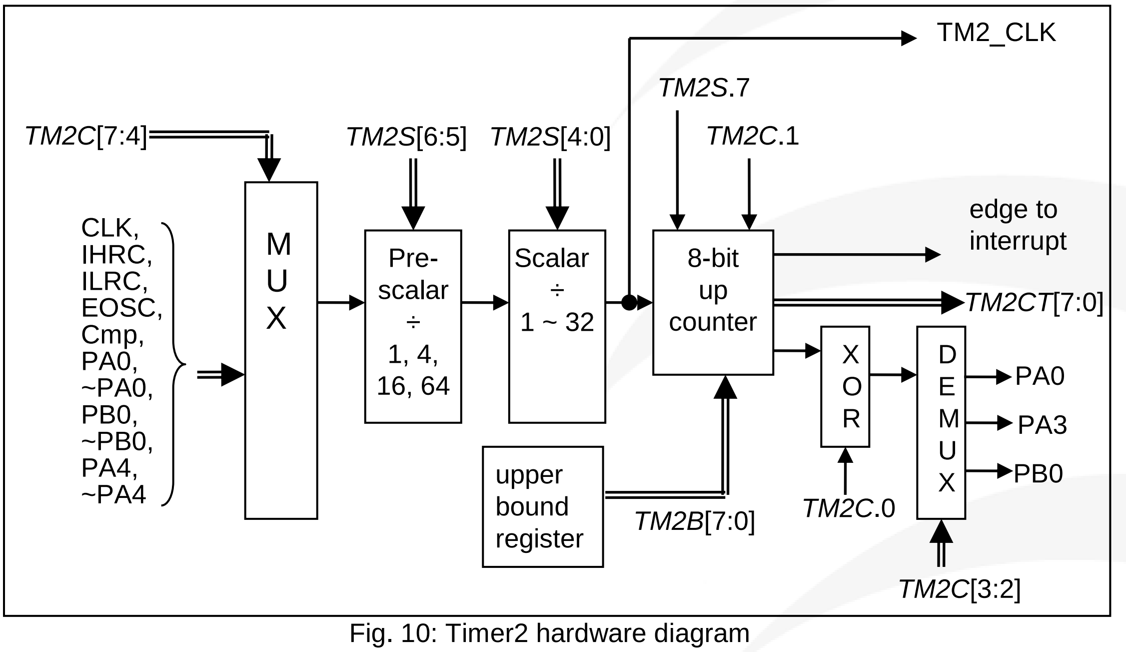


Fig. 10: Timer2 hardware diagram

The output of Timer3 can be sent to pin PA4, PA5 or PB7.

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

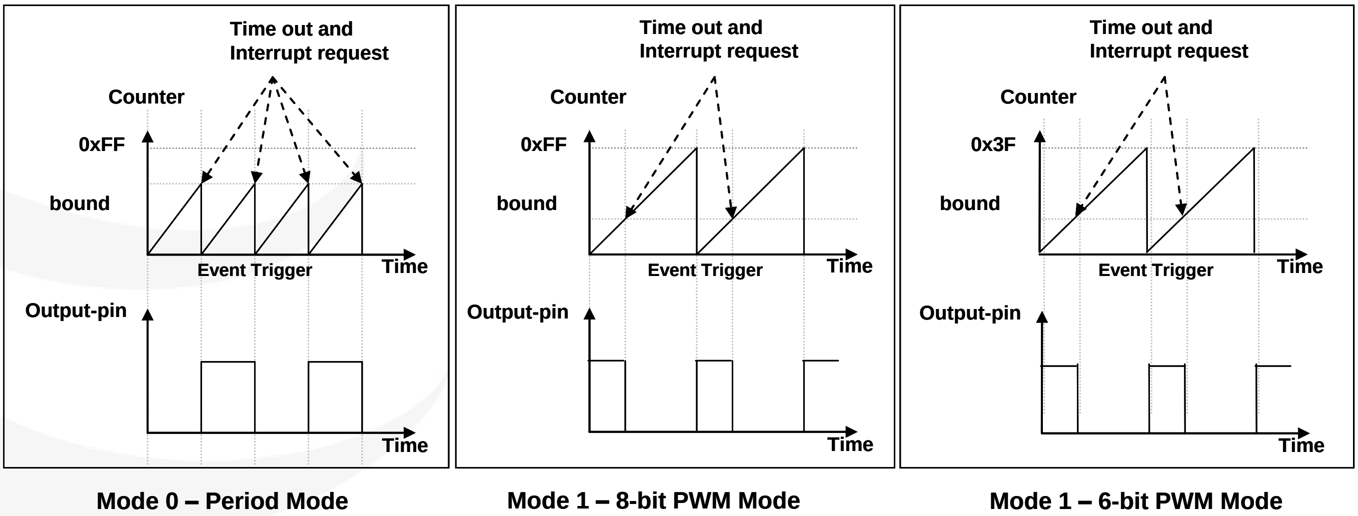


Fig. 11: Timing diagram of Timer2 in period mode and PWM mode (*TM2C*.1=1)

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.

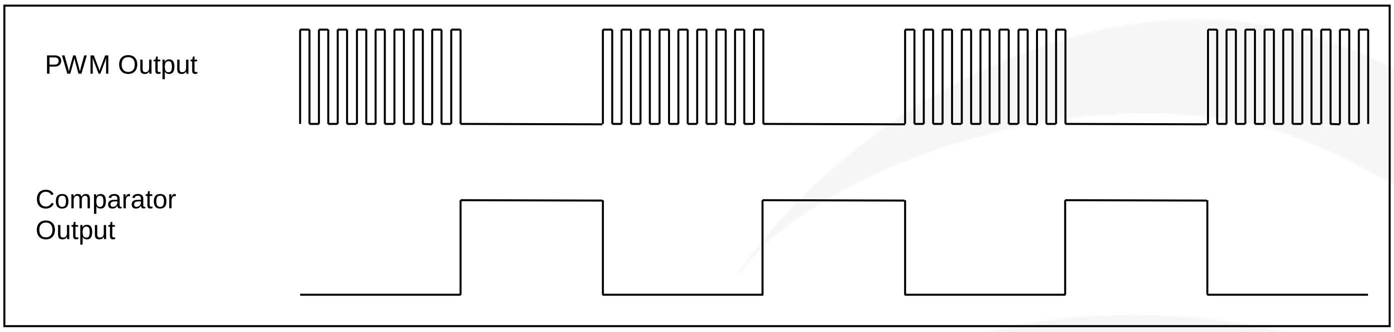


Fig.12: Comparator controls the output of PWM waveform

10.2.1. Timer2, Timer3 Related Registers

10.2.1.1. Timer2 Scalar Register (*TM2S*), address = 0x1E

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 : PA0 10 : PA3 11 : PB0
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 = 0x1F

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 : PA4 10 : PA5 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=8MHz
TM2B = 0b0111_1111, K=127
TM2S = 0b0_00_00000, S1=1, S2=0
frequency of output = 8MHz ÷ [2 × (127+1) × 1 × (0+1)] = 31.25KHz

Example 2:

TM2C = 0b0001_1000, Y=8MHz
TM2B = 0b0000_0001, K=1
TM2S = 0b0_00_00000, S1=1, S2=0
frequency = 8MHz ÷ [2 × (1+1) × 1 × (0+1)]=2MHz

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

```
void    FPPA0 (void)
{
        . ADJUST_IC     SYSCLK=IHRC/2, 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=8MHz
TM2B = 0b0111_1111, K=127
TM2S = 0b0_00_00000, S1=1, S2=0
➔frequency of output = 8MHz ÷ (256 × 1 × (0+1)) = 31.25KHz
➔duty of output = [(127+1) ÷ 256] × 100% = 50%

Example 2:

TM2C = 0b0001_1010, Y=8MHz
TM2B = 0b0000_1001, K = 9
TM2S = 0b0_00_00000, S1=1, S2=0
➔frequency of output = 8MHz ÷ (256 × 1 × (0+1)) = 31.25KHz
➔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/2, 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)

<u>Example 1:</u>

TM2C = 0b0001_1010, Y=8MHz

TM2B = 0b0011_1111, K=63

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

frequency of output = 8MHz ÷ (64 × 1 × (0+1)) = 125KHz

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

<u>Example 2:</u>

TM2C = 0b0001_1010, Y=8MHz

TM2B = 0b0000_0000, K=0

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

frequency of output = 8MHz ÷ (64 × 1 × (0+1)) = 125KHz

duty of output = [(0+1) ÷ 64] × 100% =1.5%

11. Special Functions

11.1. Comparator

One hardware comparator is built inside the PFC161; Fig. 13 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.

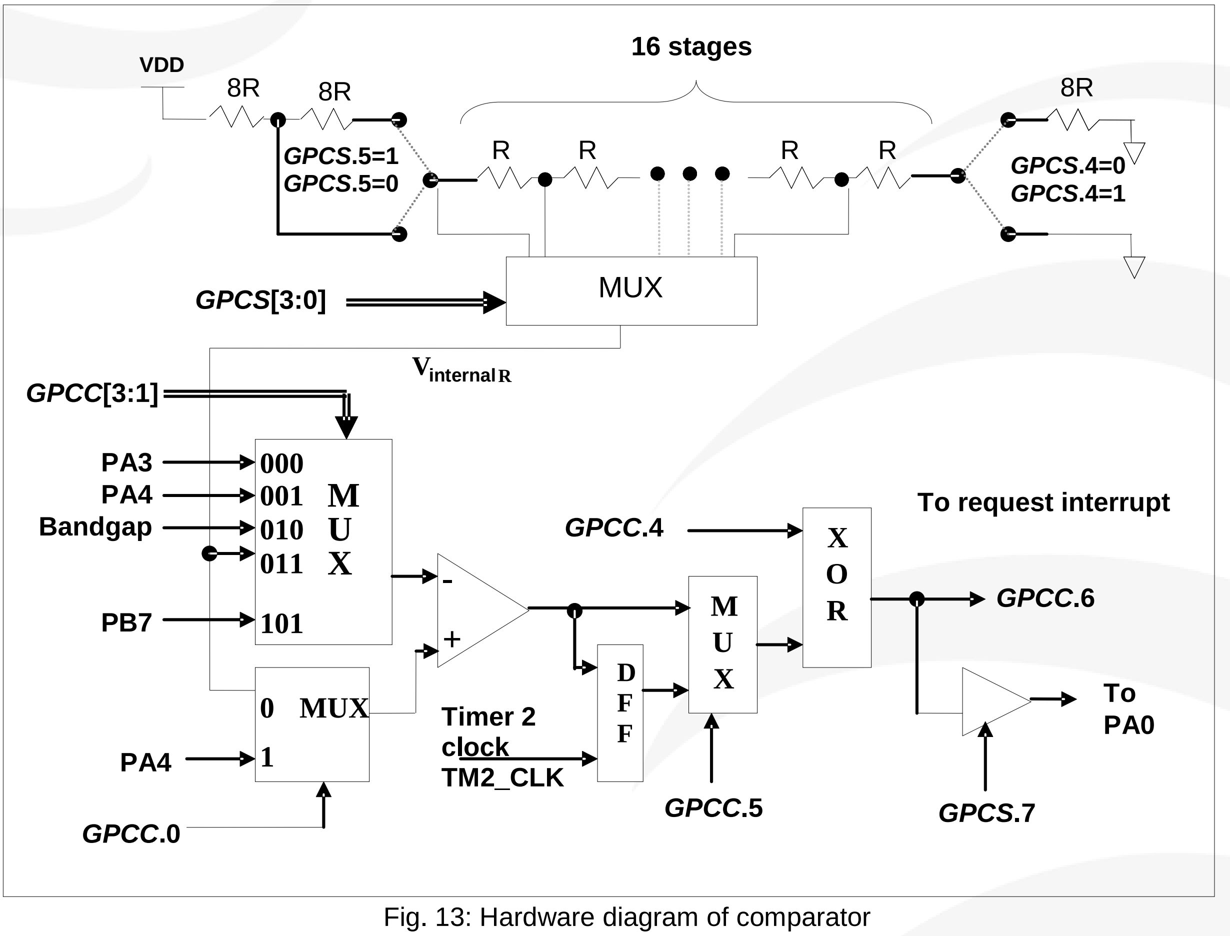


Fig. 13: Hardware diagram of comparator

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 (not suitable for the comparator wake-up function) 011: $V_{internal\ R}$ 100: reserved 101: PB7 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	0	WO	Wakeup by comparator enable. (The comparator wakeup effectively when gpcc.6 electrical level changed) 0 / 1 : disable / enable
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. 14 to Fig. 17 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}.

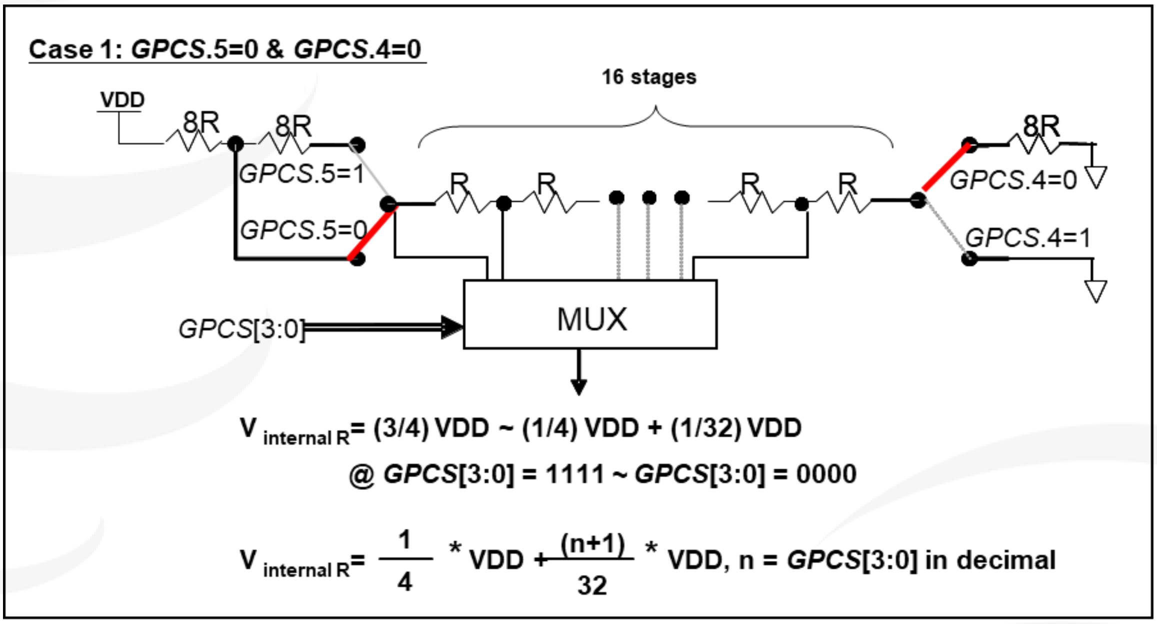


Fig. 14: $V_{internal\ R}$ hardware connection if gpcs.5=0 and gpcs.4=0

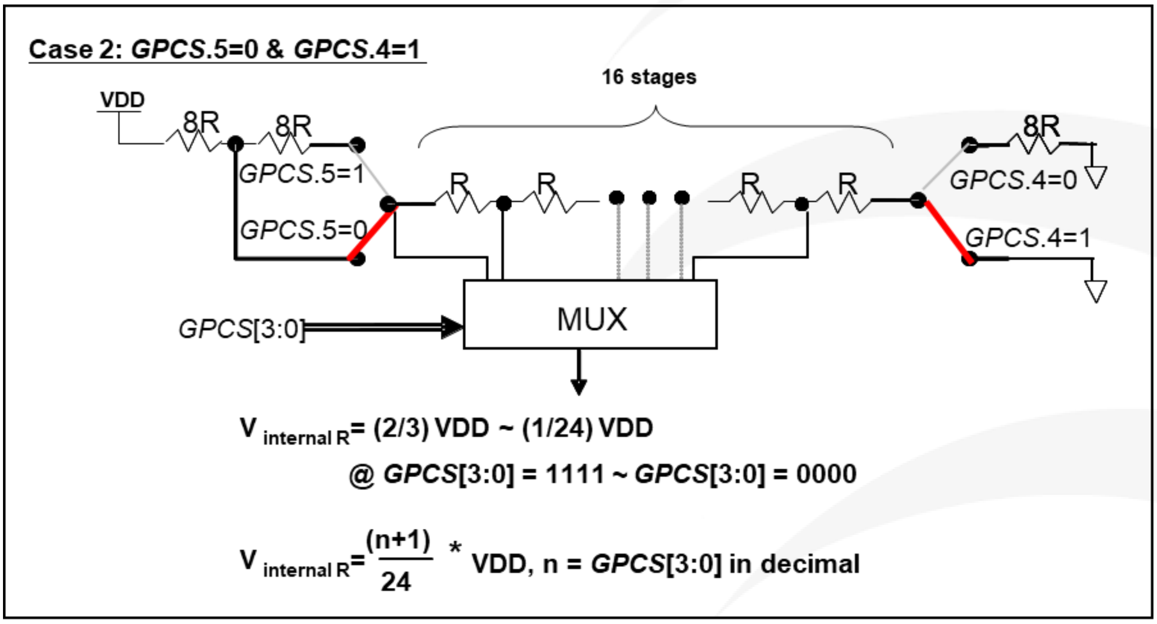


Fig. 15: $V_{internal\ R}$ hardware connection if gpcs.5=0 and gpcs.4=1

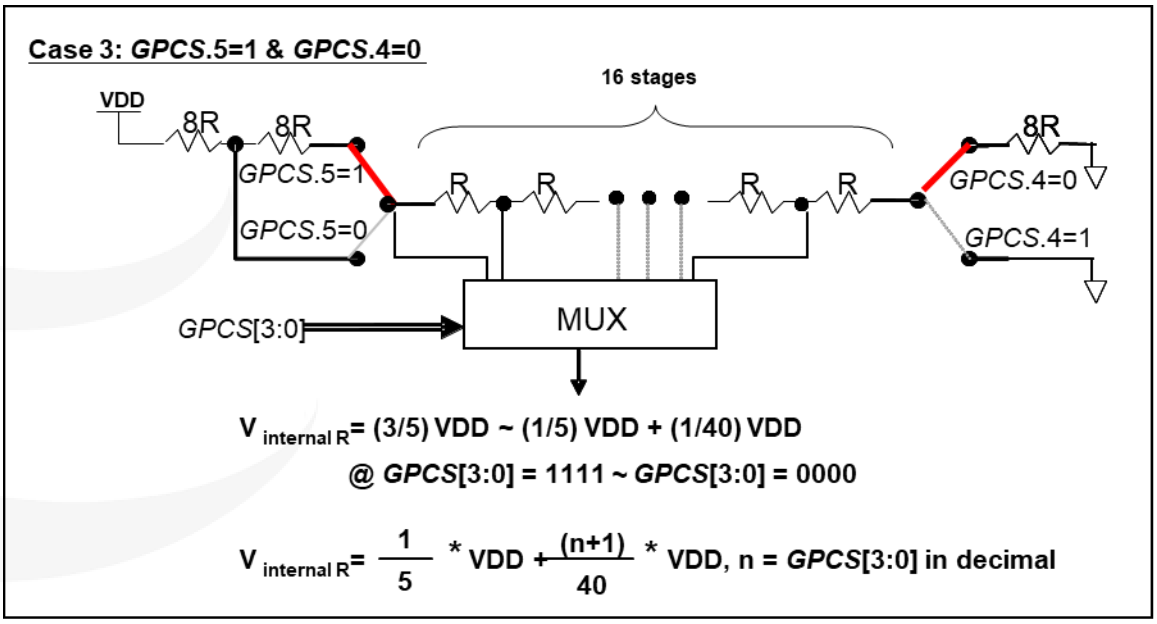


Fig. 16: $V_{internal\ R}$ hardware connection if gpcs.5=1 and gpcs.4=0

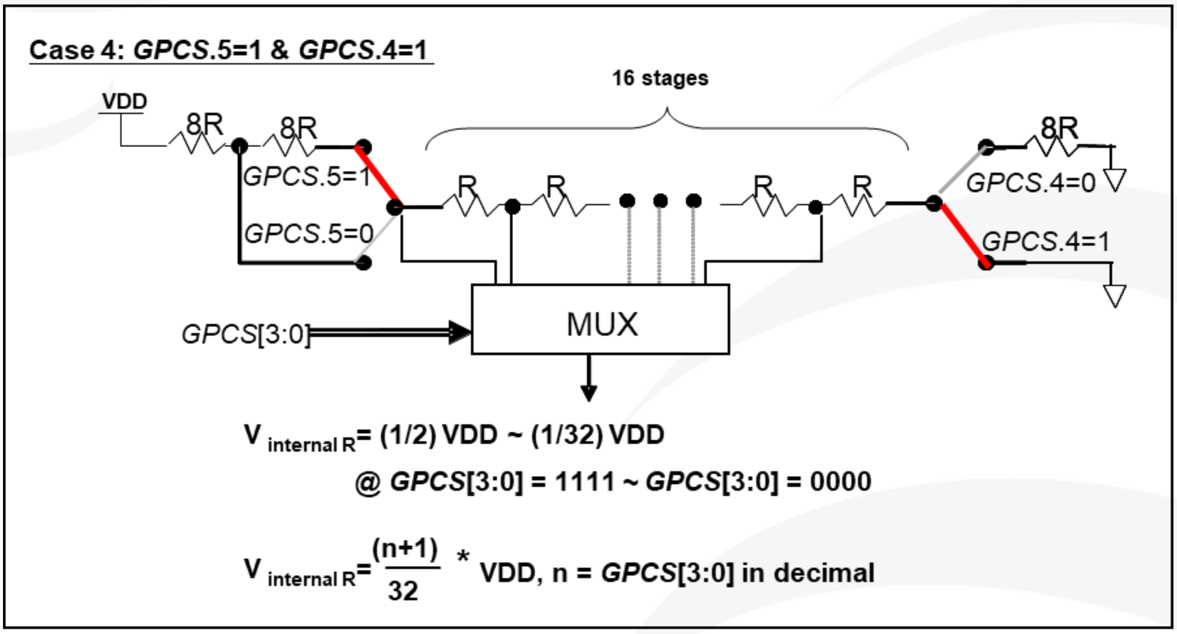


Fig. 17: $V_{internal\ R}$ hardware connection if gpcs.5=1 and gpcs.4=1

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

or

```
$ GPCS    VDD*18/32;
$ GPCC    Enable, N_PA3, P_R;    // - input: N_xx，+ input: P_R(Vinternal 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, Vinternal R = VDD*(22/40)
GPCC    = 0b1_0_0_1_011_1;      // Inverse output, - input: Vinternal R, + input: PA4
PADIER  = 0bxxxx_0_xxx;         // disable PA4 digital input to prevent leakage current
```

or

```
$ GPCS    Output, VDD*22/40;
$ GPCC    Enable, Inverse, N_R, P_PA4;  // - input: N_R(Vinternal 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. Touch Function

A touch detecting circuit is included in PFC161. Its functional block diagram is shown as Fig.18.

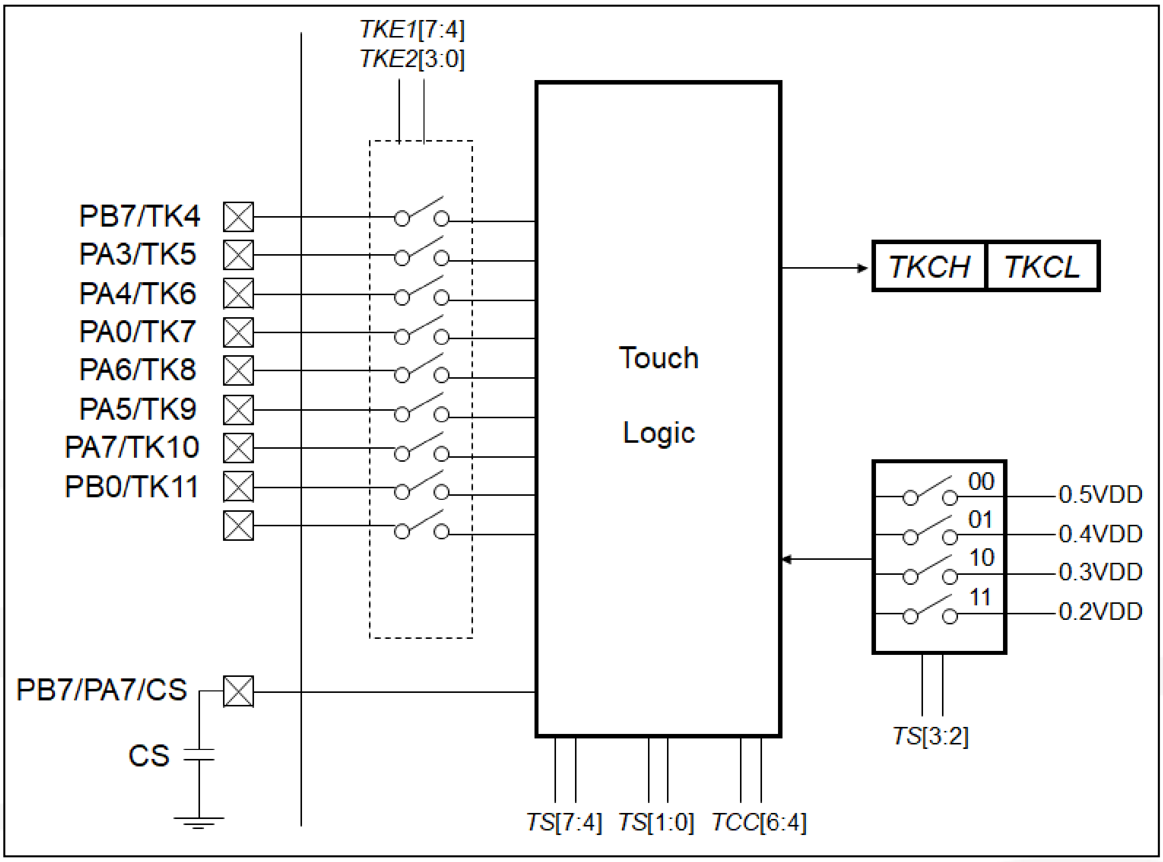


Fig. 18: Functional block diagram of the touch detecting circuit

The Touch detecting circuit in PFC161 applies the method of capacitive sensing, detecting the capacitive virtual ground effect of a finger, or the capacitance between sensors.

An accurate, X7R external capacitor CS is required to connect between one of the PA7/CS pin or PB7/CS pin and GND. User can choose one of the PA7 or PB7 as CS pad by code options PA7_Sel and PB7_Sel. When the CS pad is selected, the other can be selected as Touch key like other IO.

For starting touch detecting process, user should follow the procedures below:

1. Selecting the touch pad to be measure by setting *TKE1* & *TKE2* registers. Only one pad should be selected a time.

2. Issuing a *Touch START* command by writing “0x10” into *TCC* register. The capacitor CS will be automatically discharged to VSS firstly. The discharging time is selectable from 32, 64 and 128 Touch clocks by *TS*[1:0].

3. The larger the CS capacitance value, the longer the discharge time is needed to fully discharge the capacitor to VSS. However, in some cases, 128 Touch clocks may still be not long enough to fully discharge the CS capacitor. At this time, user should do it manually by writing “0x30” into *TCC* register instead of “0x10”. After a certain discharge time decided by the user, user can issue a *Touch START* (0x10) command to continue this touch conversion progress. Or user can also abort the conversion progress by writing “0x00” into TCC register.

4. After discharging, the CS will be charged toward VCC per Touch clock (TK_CLK). The charging speed is determined by the capacitance value of the selected Touch pad.

5. The charging progress will be stopped automatically when its voltage reaches the internal generated threshold voltage (VREF). The program determines whether the charging process is stopped by reading *TCC*[6:4] or *INTRQ*[3].
   The VREF voltage is selectable from 0.2*VCC, 0.3*VCC, 0.4*VCC and 0.5*VCC by *TS*[3:2].

6. By reading the Touch Counter value from *TKCH* & *TKCL* registers, user can monitor the capacitance value change of the Touch pad. The value reads from Touch Counter is related to the ratio of CS and CP, while CP represents the total capacitance that is the combination of PCB, wire and touch pad whose capacitance can be varied by human finger’s touch. Once the CP value is altered, the periods required to charge the CS to VREF shorten. By counting the discrepancy of clock periods, the circuitry can decide if the touch pad is enabled.

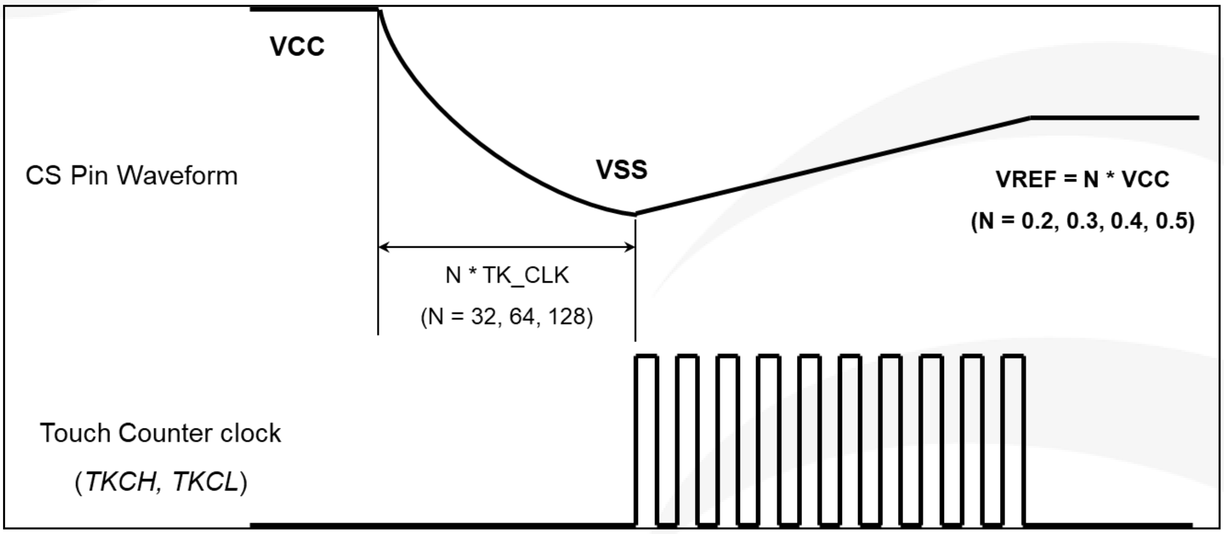


Fig. 19: Timing diagram of Touch converting progress

Note: When the VREF voltage is first set or the reference voltage option is switched midway, please discard the first *TKCH* and *TKCL* data read after that.

11.2.1. Touch Related Registers

11.2.1.1. Touch Selection Register (*TS*), IO address = 0x20

Bit	Reset	R/W	Description
7 - 4	-	R/W	Touch clock selection (TK_CLK) 0010: 4 MHz 0011: 2 MHz 0100: 1 MHz 0101: 500 kHz 0110: 250 kHz 0111: 125 kHz 1000: ILRC Others: reserved
3 - 2	00	R/W	Touch VREF selection 00: 0.5 * VCC 01: 0.4 * VCC 10: 0.3 * VCC 11: 0.2 * VCC
1 - 0	00	R/W	Select the discharge time before starting the touch function (TK_DISCHG) 00: reserved 01: 32 * CLK 10: 64 * CLK 11: 128 * CLK

11.2.1.2. Touch Charge Control Register (*TCC*), IO address = 0x21

<table>
<tr><th>Bit</th><th>Reset</th><th>R/W</th><th colspan="3">Description</th></tr>
<tr><td>7</td><td>-</td><td>-</td><td colspan="3">Reserved</td></tr>
<tr><td rowspan="6">6 - 4</td><td rowspan="6">-</td><td rowspan="6">R/W</td><td colspan="3">Touch control and status</td></tr>
<tr><th>Data</th><th>Command (W)</th><th>Status (R)</th></tr>
<tr><td>000</td><td>TK_STOP
(Touch module power down)</td><td>Ready / End</td></tr>
<tr><td>001</td><td>TK_RUN
Command for Touch Start</td><td>Running</td></tr>
<tr><td>011</td><td>Discharge
(Discharge CS capacitor)</td><td>Discharging</td></tr>
<tr><td>Others</td><td>Reserved</td><td>Reserved</td></tr>
<tr><td>3 - 0</td><td>-</td><td>-</td><td colspan="3">reserved</td></tr>
</table>

11.2.1.3. Touch Key Enable 2 Register (*TKE2*), IO address = 0x22

Bit	Reset	R/W	Description
7 - 4	-	-	Reserved
3	0	R/W	Enable PB0/TK11. 0/1: disable/enable
2	0	R/W	Enable PA7/TK10. 0/1: disable/enable
1	0	R/W	Enable PA5/TK9. 0/1: disable/enable
0	0	R/W	Enable PA6/TK8. 0/1: disable/enable

11.2.1.4. Touch Key Enable 1 Register (*TKE1*), IO address = 0x24

Bit	Reset	R/W	Description
7	0	R/W	Enable PA0/TK7. 0/1: disable/enable
6	0	R/W	Enable PA4/TK6. 0/1: disable/enable
5	0	R/W	Enable PA3/TK5. 0/1: disable/enable
4	0	R/W	Enable PB7/TK4. 0/1: disable/enable
3 - 0	-	-	reserved

11.2.1.5. Touch Key Charge Counter High Register (*TKCH*), IO address = 0x2B

Bit	Reset	R/W	Description
7 - 4	-	-	Reserved
3 - 0	-	RO	tkct[11:8] of touch key charge counter.

11.2.1.6. Touch Key Charge Counter Low Register (*TKCL*), IO address = 0x2C

Bit	Reset	R/W	Description
7 - 0	-	RO	tkct[7:0] of touch key charge counter.

11.2.1.7. Touch Parameter Setting Register 2 (*TPS2*), IO address = 0x28

Bit	Reset	R/W	Description
7 – 6	00	R/W	Touch module type 00: Type A (suggestion: CS connect to VDD on PCB) 01: Type B (suggestion: CS connect to GND on PCB) 1X: Reserved
5 - 3	000	R/W	Reserved, keep 000
2	0	R/W	Reserved, keep 0
1 – 0	00	R/W	Vref non-floating cycles selection 00: Vref always floating after discharge 01: Vref always on after discharge **(recommended)** 10: First 64 cycles after discharge 11: First 128 cycles after discharge

12. Notes for Emulation

It is recommended to use 5S-I-S01/2(B) for emulation of PFC161. The following items should be noted when using 5S-I-S01/2(B) to emulate PFC161:

(1) 5S-I-S01/2(B) doesn't support SYSCLK=ILRC/16 and ILRC/2.

(2) 5S-I-S01/2(B) doesn't support PA5 as the interrupt source.

(3) 5S-I-S01/2(B) doesn't support the code options: GPC_PWM, TMx_source, TMx_bit, CS_Sel, Interrupt_Src0, PA3_PA4_Drive.

(4) 5S-I-S01/2(B) doesn't support *GPCRS* source of *TM2C* and *TM3C*.

(5) 5S-I-S01/2(B) doesn't support *PAPL*, *PABL*.

(6) 5S-I-S01/2(B) doesn't support ALL touch function.

(7) *TM2C* PWM outputs are PA0, PA3 and PB0 on chip; they are PB2, PA3 and PB4 in ICE.

(8) *TM3C* PWM outputs are PA4, PA5 and PB7 on chip; they are PB5, PB6 and PB7 in ICE.

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

(10) 5S-I-S01/2(B) doesn't support Timer2/Timer3 function with comparator as clock source, but 6S-M-001 can emulate.

(11) When simulating PWM waveform, please check the waveform during program running. When the ICE is suspended or single-step running, its waveform may be inconsistent with the reality.

(12) The ILRC frequency of the 5S-I-S01/2(B) simulator is different from the actual IC and is uncalibrated, with a frequency range of about 34K~38KHz.

(13) Fast Wakeup time is different from 5S-I-S01/2(B): 128 SYSCLK, PFC161: 45 ILRC.

(14) Watch dog time out period is different from 5S-I-S01/2(B):

WDT period	5S-I-S01/2(B)	PFC161
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 PDK-5S-P003 to program. 3S-P-002 or older versions do not support programming PFC161.

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 PFC161. Be sure its connected IC and devices will not be damaged by the following voltages, and will not clam the following voltages.

The voltage conditions in normal programming mode:

(1) VDD is 7.5V, and the maximum supply current is up to about 20mA.

(2) PA5 is 6V.

(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 PFC161. 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 “PDK-5S-P003 UM”).

13.3. On-Board Writing

PFC161 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 PDK-5S-P003 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.

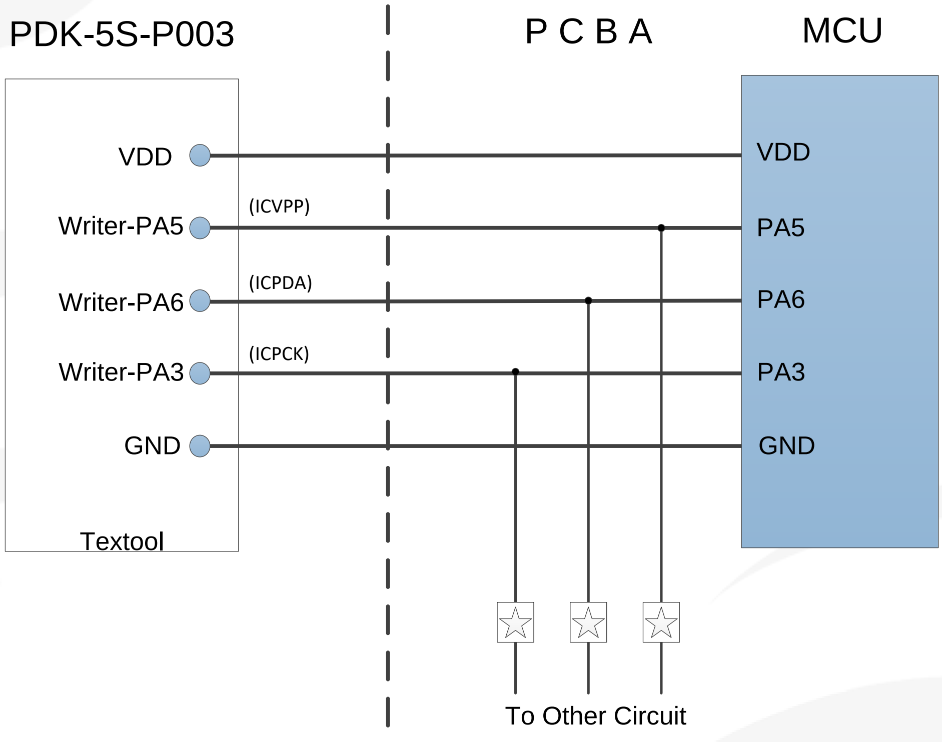


Fig. 20: Schematic Diagram of On-Board Wiring

The symbol ☆ on Fig. 20 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	**If V_{DD} is over the maximum rating, it may lead to a permanent damage of IC.**
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}=3.3V, f_{SYS}=2MHz unless noted.

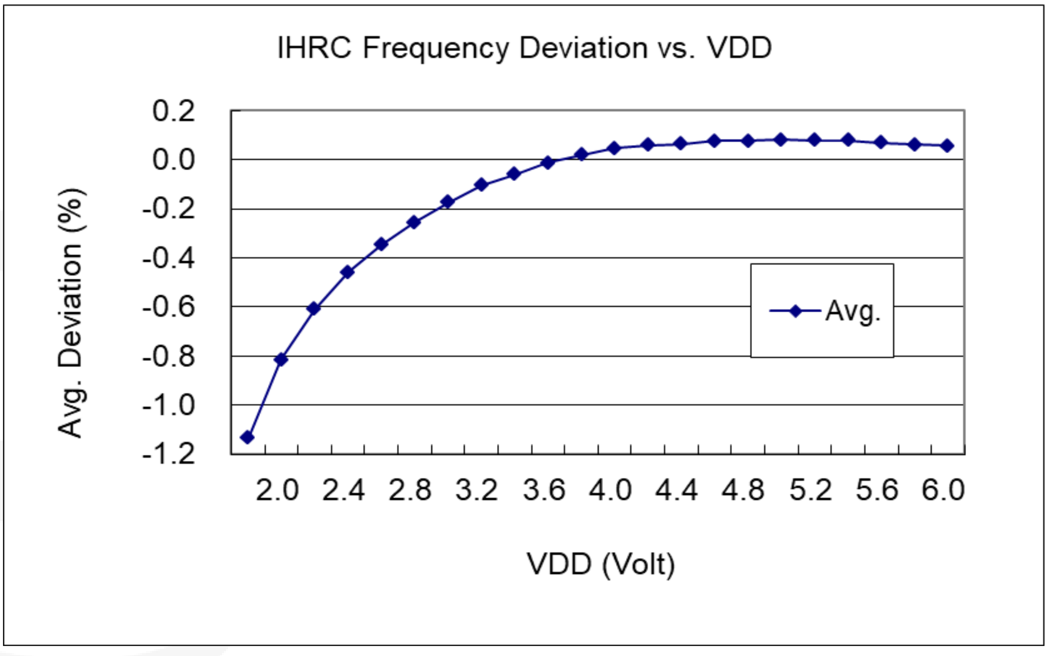
Symbol	Description	Min.	Typ	Max.	Unit	Conditions
V_{DD}	Operating Voltage	2.2#	5	5.5	V	# Subject to LVR tolerance
LVR%	Low Voltage Reset Tolerance	-5		5	%	
f_{SYS}	System clock (CLK)* = IHRC/2 IHRC/4 IHRC/8 ILRC	 0 0 0 	 64K	 8M 4M 2M 	Hz	 V_{DD} ≧ 3.5V V_{DD} ≧ 2.5V V_{DD} ≧ 2.2V V_{DD} =5V
P_{cycle}	Program cycle	1000			cycles	
I_{OP}	Operating Current		0.7 82		mA uA	f_{SYS}=IHRC/16=1MIPS@5V f_{SYS}=ILRC=64KHz@5V
I_{PD}	Power Down Current (by *stopsys* command)		1		uA	V_{DD} =5V
I_{PS}	Power Save Current (by *stopexe* command) *Disable IHRC		3		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}	V	
I_{OL}	IO lines sink current (Strong) PA3/PA4 Others		 40 23		mA	V_{DD}=5.0V, V_{OL}=0.5V
	IO lines sink current (Normal)		23			
I_{OH}	IO lines drive current (Strong) PA3/PA4 Others		 20 10		mA	V_{DD}=5.0V, V_{OH}=4.5V
	IO lines drive current (Normal)		10			
V_{IN}	Input voltage	-0.3		V_{DD}+0.3	V	

Symbol	Description	Min.	Typ	Max.	Unit	Conditions
$I_{INJ (PIN)}$	Injected current on pin		1		uA	V_{DD} +0.3≧V_{IN}≧ -0.3
R_{PH}	Pull-high Resistance PA5 Others		 85 80		KΩ	V_{DD} =5.0V
R_{PL}	Pull-low Resistance PA5 Others		 85 80		KΩ	V_{DD}=5.0V
V_{BG}	Bandgap Reference Voltage	1.145*	1.20*	1.255*	V	V_{DD} = 2.2V ~ 5.5V -40 ºC <Ta<85 ºC *
f_{IHRC}	Frequency of IHRC after calibration *****	15.84*	16*	16.16*	MHz	VDD=5V, Ta=25 ºC
		15.20*	16*	16.80*		V_{DD} =2.2V~5.5V, -40 ºC <Ta<85 ºC *
t_{INT}	Interrupt pulse width	30			ns	V_{DD} = 5.0V
t_{WDT}	Watchdog timeout period		8k		ILRC clock period	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		50		ms	@ V_{DD} =5V
t_{RST}	External reset pulse width	120			us	@ V_{DD} =5V
CPos	Comparator offset*	-	±10	±20	mV	
CPcm	Comparator input common mode*	0		VDD -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

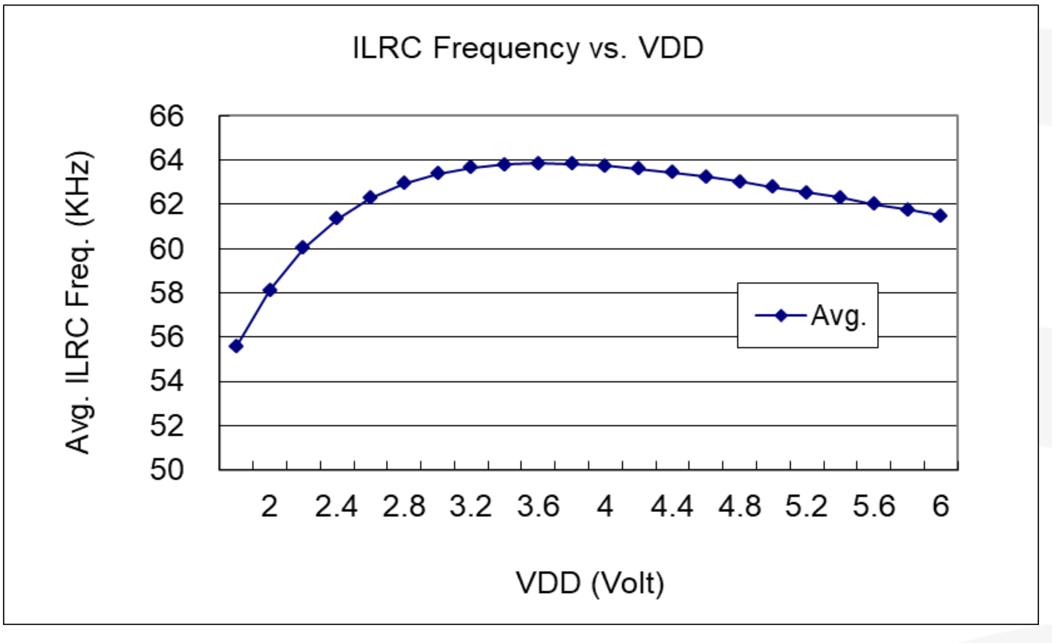
*These parameters are for design reference, not tested for every chip.

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.

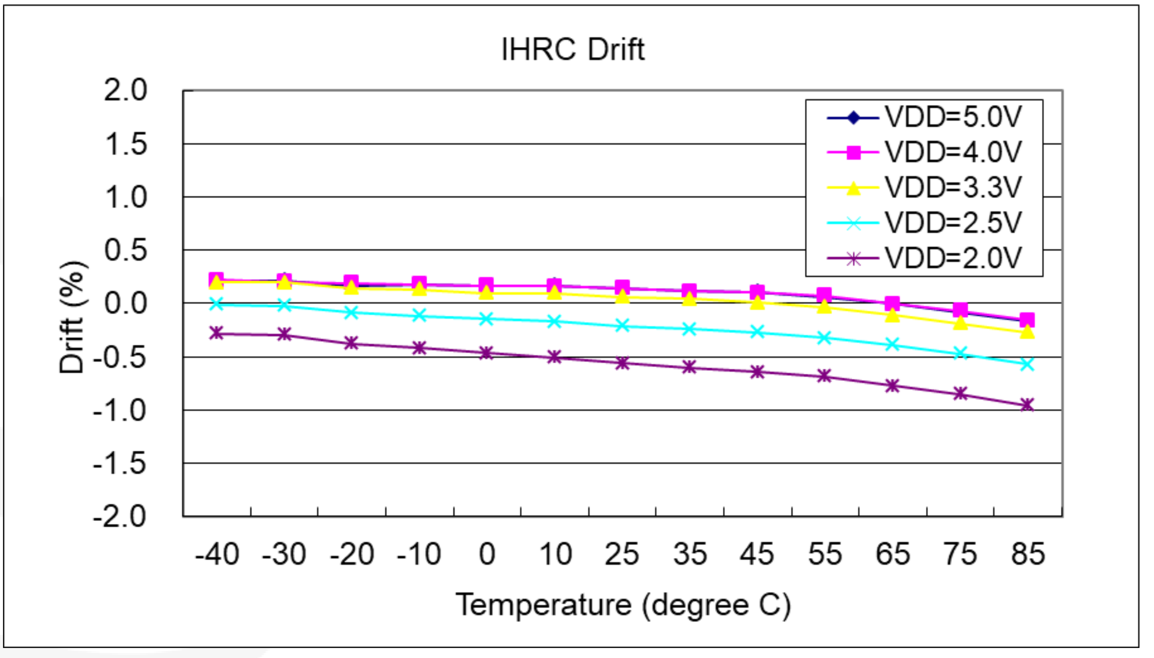
14.3. Typical IHRC Frequency vs. VDD (calibrated to 16MHz)



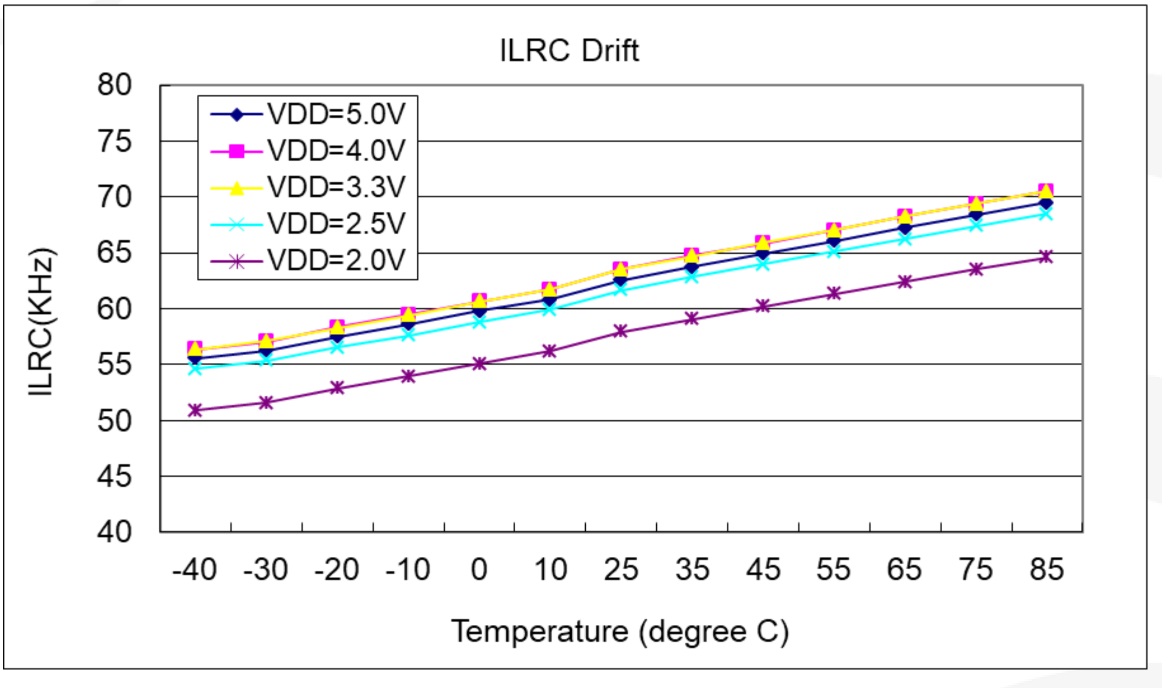
14.4. Typical ILRC Frequency vs. VDD



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



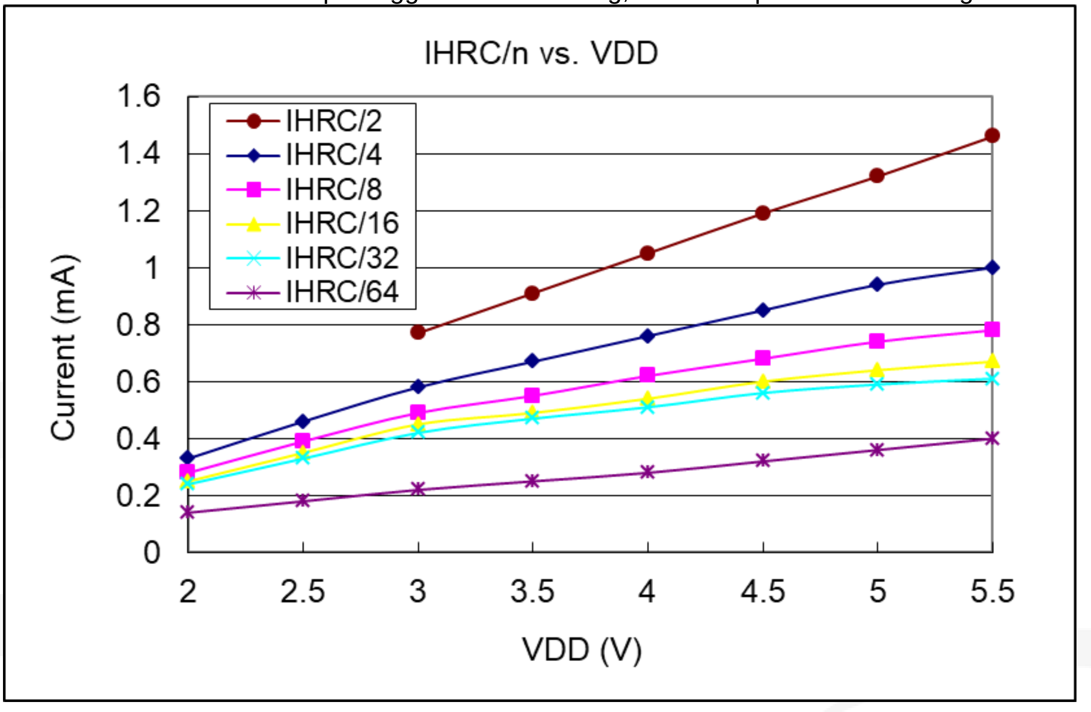
14.6. Typical ILRC Frequency vs. Temperature



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

Conditions: **ON**: EOSC, Bandgap, LVR; **OFF**: T16 modules, IHRC, ILRC modules, Touch module;

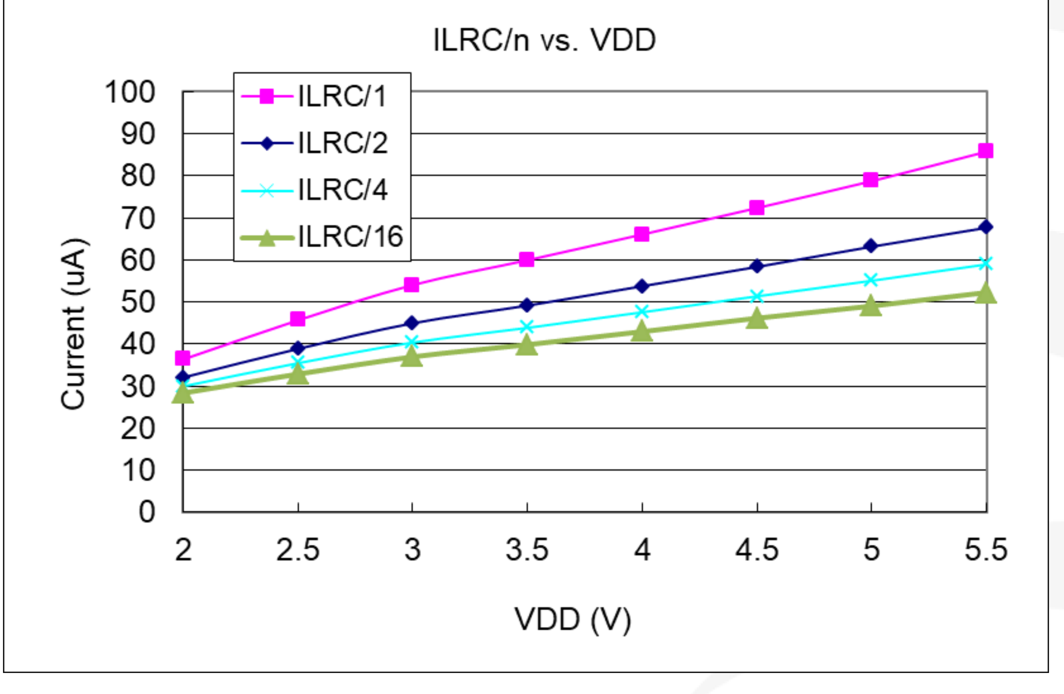
IO: PA0:0.5Hz output toggle and no loading, **others**: input and no floating



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

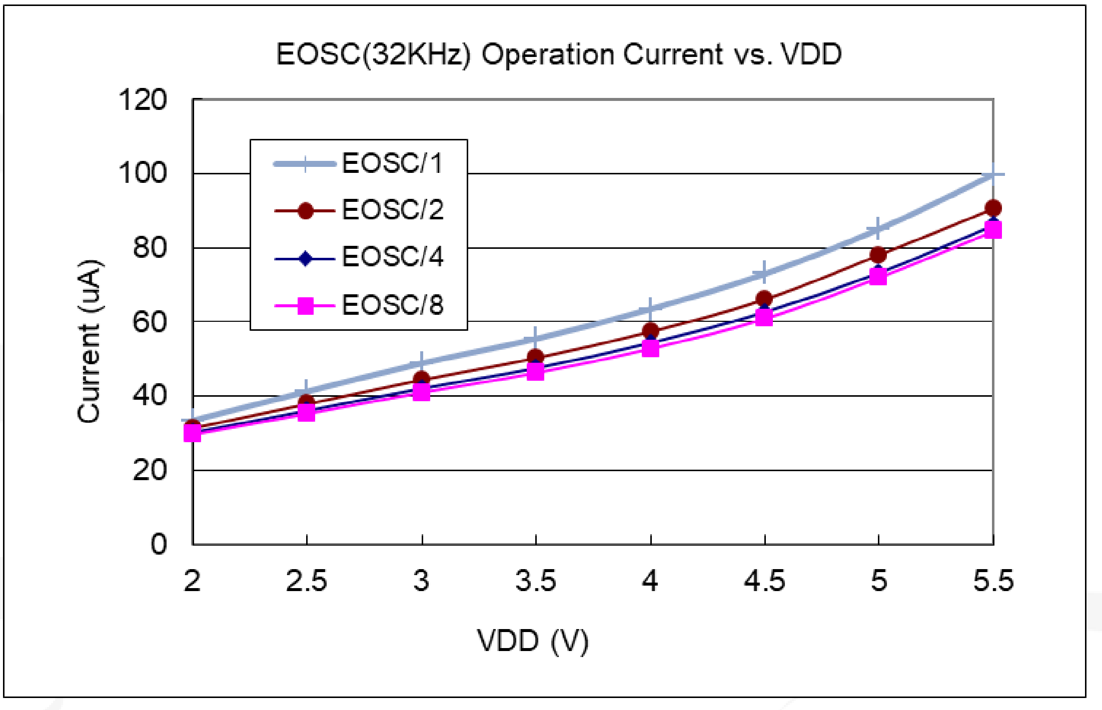
Conditions: **ON**: EOSC, Bandgap, LVR; **OFF**: T16 modules, IHRC, ILRC modules, Touch module;

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



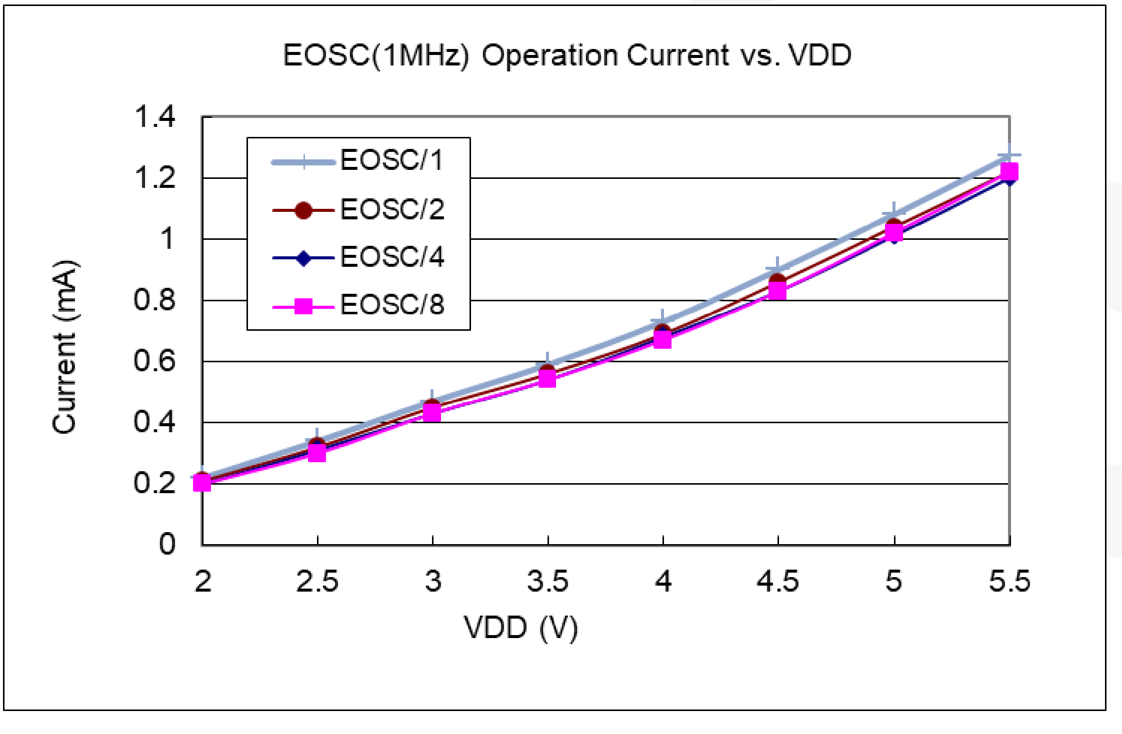
14.9. Typical Operating Current vs. VDD and CLK=32KHz EOSC / n

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



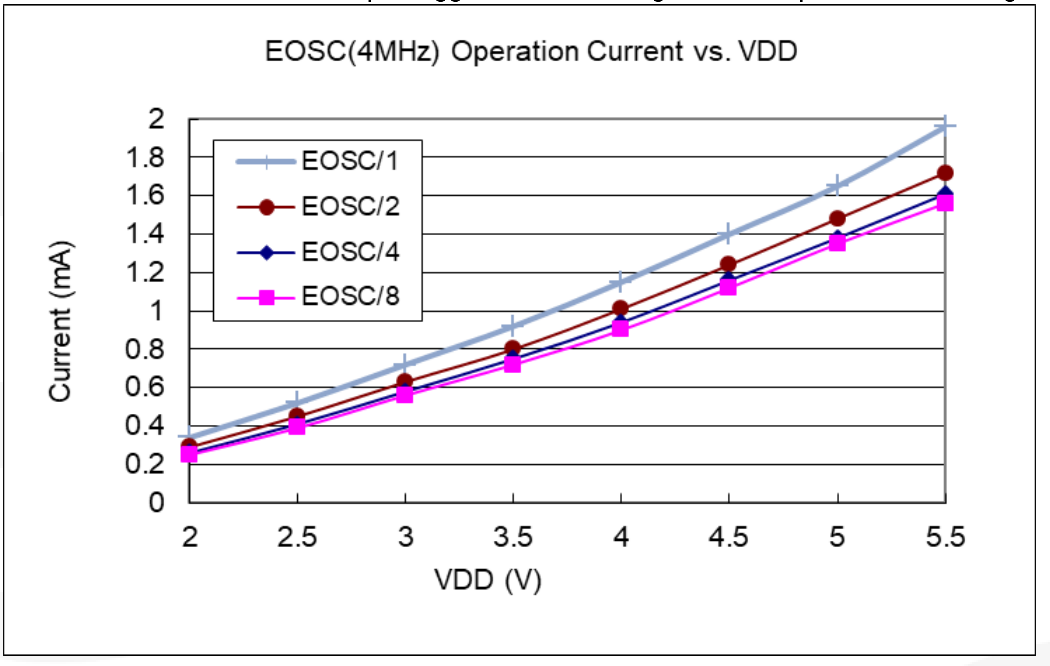
14.10. Typical Operating Current vs. VDD and CLK=1MHz EOSC / n

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

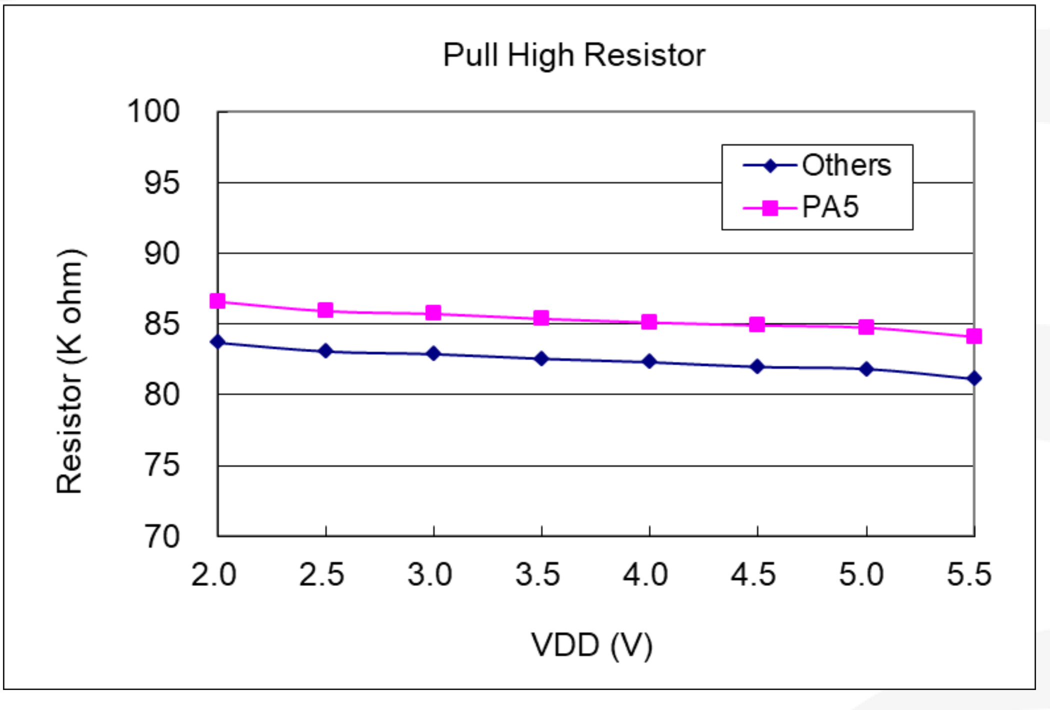


14.11. Typical Operating Current vs. VDD and CLK=4MHz EOSC / n

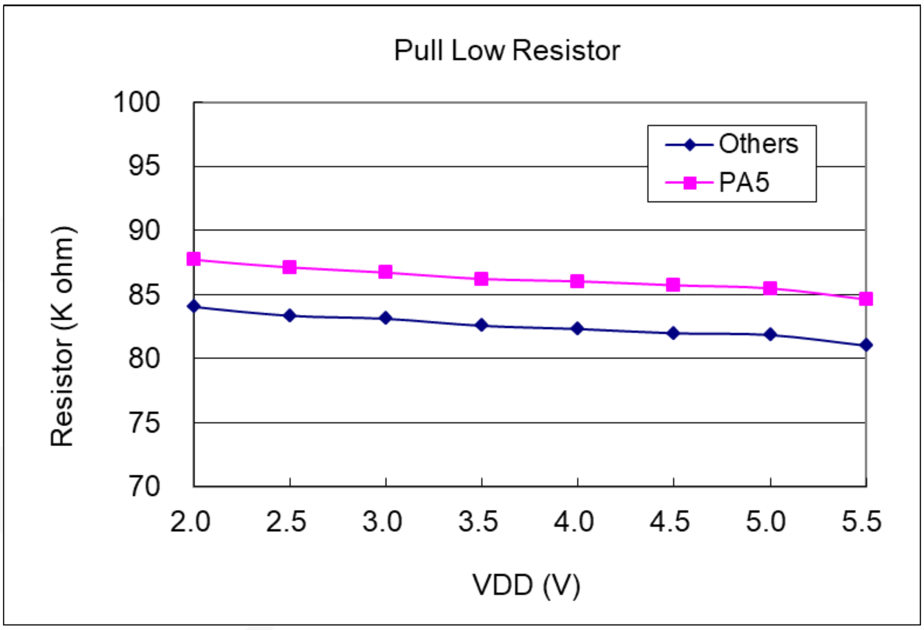
Conditions: **ON**: EOSC, Bandgap, LVR; **OFF**: T16 modules, IHRC, ILRC modules, Touch module;
IO: PA0:0.5Hz output toggle and no loading, **others**: input and no floating



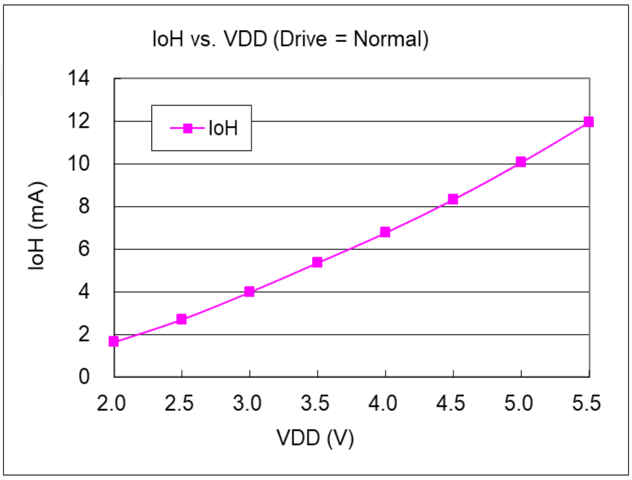
14.12. Typical IO pull high resistance

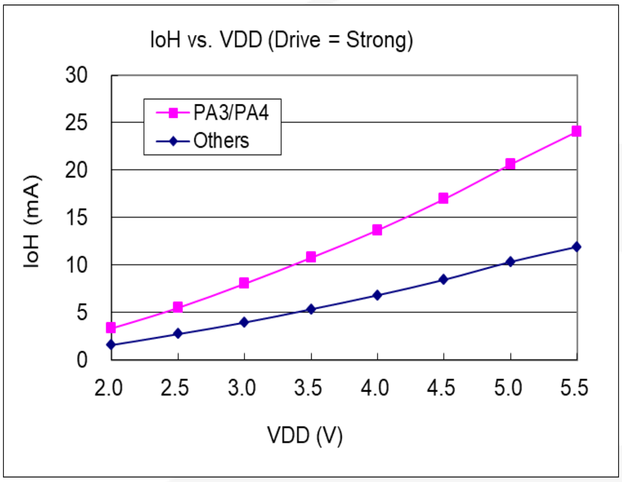


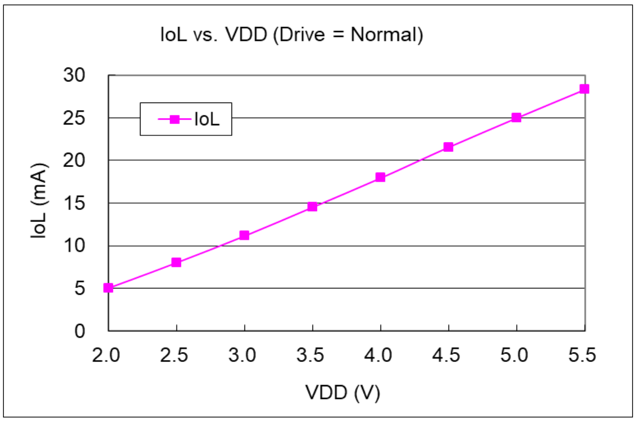
14.13. Typical IO pull low resistance

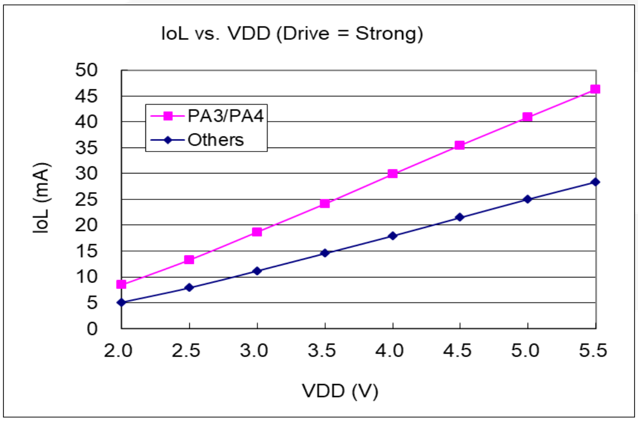


14.14. Typical IO driving current (I_{OH}) and sink current (I_{OL})
(VOH=0.9*VDD, VOL=0.1*VDD)

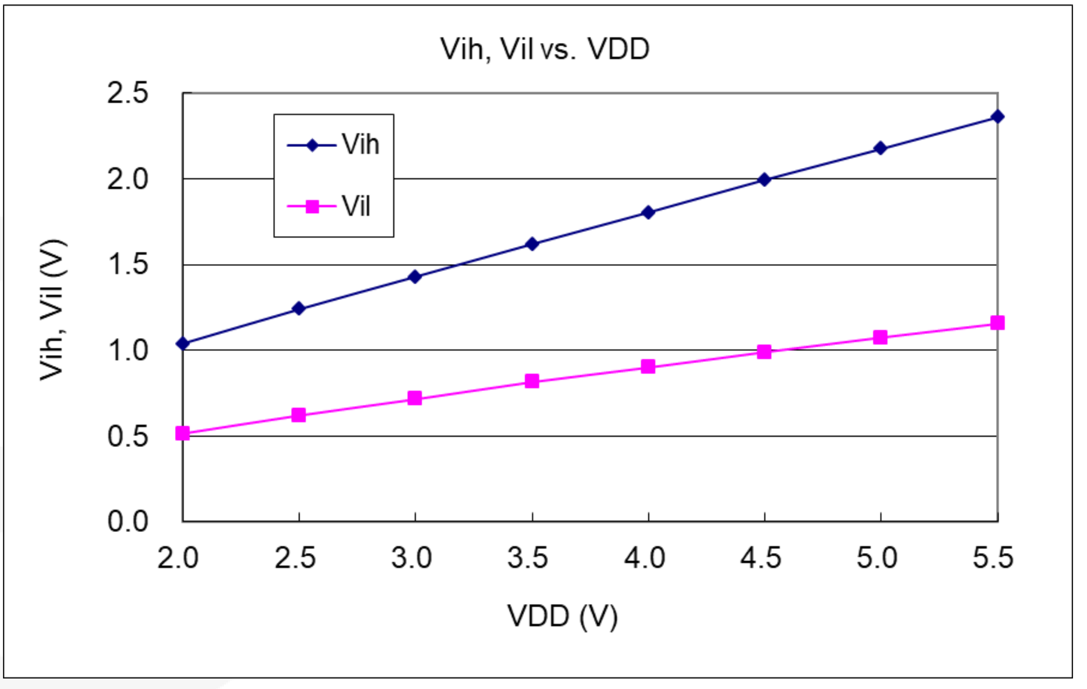




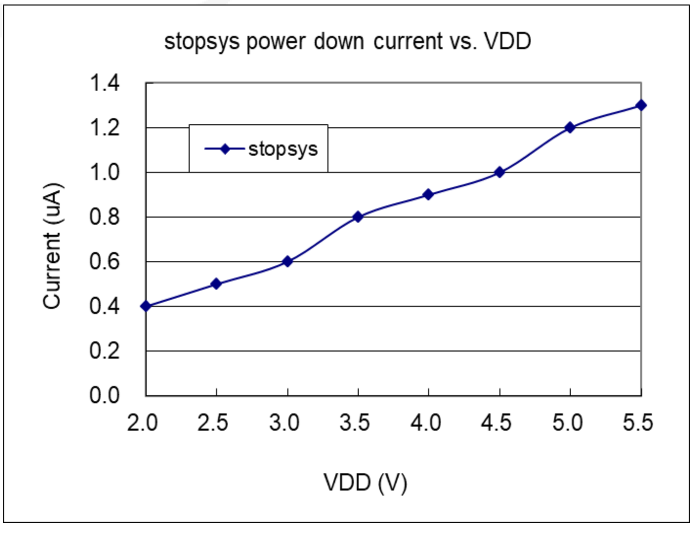


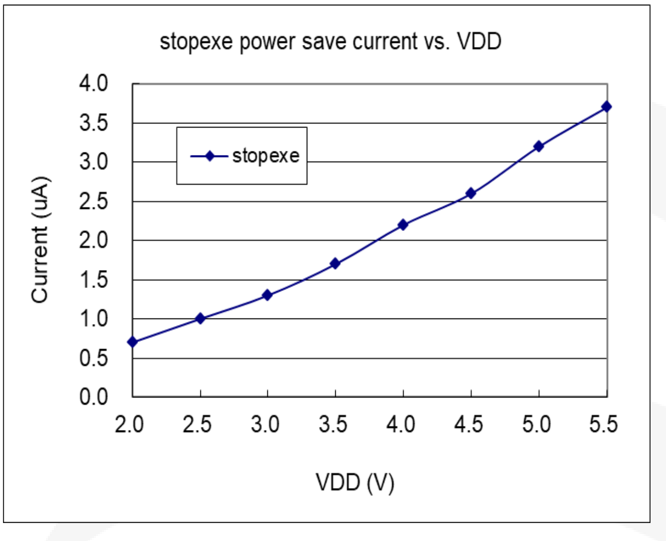


14.15. Typical IO input high/ low threshold voltage (V_{IH}/ V_{IL})



14.16. 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* error; Go to error 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 from 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	-	-	-	-