
# PFB190

## 8bit MTP type MCU with Charger

## *Datasheet*

***Version 0.02 – November 26, 2025***



**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 which 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 content

# Revision History

| Revision | Date | Description |
|---|---|---|
| 0.00 | 2025/08/08 | Preliminary version |
| 0.01 | 2025/10/02 | 1. Add PFB190-1J16A:QFN3*3-16L(0.75pitch)<br>2. Section 4.2: Specification Update — Inclusion of Value Range / Maximum Ratings and Remarks for Power Supply Voltage VCC. |
| 0.02 | 2025/11/26 | 1. Section 4.1: fSYS: IHRC/4, 4M, VDD ≧ 2.2V adjusted to 2.3V.<br>2. Section 4.1: Add RMS Current Source. |

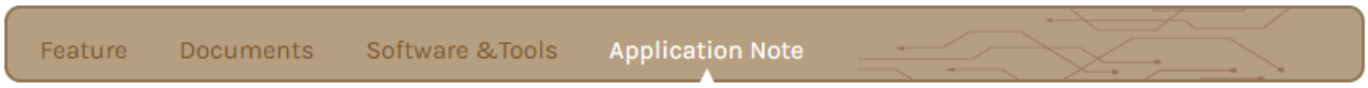
# Usage Warning

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.

https://www.padauk.com.tw/en/product/search_list.aspx?kw=PFB

(The following picture are for reference only.)




| Content | Description | Download (CN) | Download (EN) |
|---|---|---|---|
| APN001 | Output impedance of ADC analog signal source | | |
| APN002 | Over voltage protection | | |
| APN003 | Over voltage protection | | |
| APN004 | Semi-Automatic writing handler | | |
| APN005 | Effects of over voltage input to ADC | | |
| APN007 | Setting up LVR level | | |
| APN011 | Semi-Automatic writing Handler improve writing stability | | |
| APN013 | Notification of crystal oscillator | | |
| APN019 | E-PAD PCB layout guideline | | |

# 1. Features

## 1.1 Special Features

- General purpose series
- In applications with AC RC step-down power supply or high EFT requirements, the system circuit needs to be modified if necessary to improve the anti-interference capability
- Operating temperature range: -40°C ~ 85°C

## 1.2 System Features

| Series | MTP program memory | RAM (byte) | Max IO no. | Max ADC Channel no. |
|---|---|---|---|---|
| PFB190 | 3KW | 128 | 18 | 14 |

- One hardware 16-bit timer
- Two hardware 8-bit timers with PWM generation
- Three hardware 11-bit PWM generators (PWMG0, PWMG1 & PWMG2)
- Band-gap circuit to provide 1.20V reference voltage
- Up to 13-channel 12-bit resolution ADC with one channel comes from internal band-gap reference voltage or 0.375*$V_{DD}$
- ADC reference high voltage: external input, internal VDD, Band-gap 1.20V, 2.2V, 2.4V, 2.68V
- Max. 18 IO pins with optional pull-high & pull-low resistor
- 18 IO pins Driving capability, Sink current = 30 / 60mA (Strong) and Drive current = 20mA
- Every IO pin can be configured to enable wake-up function
- VDD/2 LCD bias, voltage generation for up to 4×9 LCD display
- Clock sources: IHRC, ILRC
- One low-power clock (NILRC) wake-up stopsys regularly.
- For every wake-up enabled IO, two optional wake-up speed are supported: normal and fast
- LVR range: 1.8V ~ 4.5V
- External interrupt pins by code option
- VCC input range: 4.3V ~ 20V
- Programmable Charge Current Up to 500mA
- Provide CC/CV operation with Thermal Regulation to Maximize Charge Rate Without Risk of Overheating
- Sense has two main functions: Microphone Capacitor Sense (MCS) and Heating Resistor Sense (HRS)
  1. MCS supports capacitance range from 10pF to 24pF, and
     a. Typical detection noise = 21 LSB when $C_m$ = 10pF
     b. Typical detection noise = 22 LSB when $C_m$ = 24pF
  2. MCS supports MEMS-capacitor, and
     a. Typical detection noise = 20 LSB when $C_{MEMS\text{-}capacitor}$ = 1.4pF
  3. HRS supports resistor range from 0.5Ω to 1.5Ω, and 0.1Ω resistor variation detection

## 1.3 CPU Features

- 8-bit high performance RISC CPU
- 93 powerful instructions
- Most instructions are 1T execution cycle
- Programmable stack pointer to provide adjustable stack level
- Direct and indirect addressing modes for data access.
  Data memories are available for use as an index pointer of Indirect addressing mode
- IO space and memory space are independent

## 1.4 Ordering/ Package Information

- PFB190-1J16A:QFN3*3-16L(0.75pitch)
- PFB190-2J24A:QFN4*4-24L(0.5pitch)

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

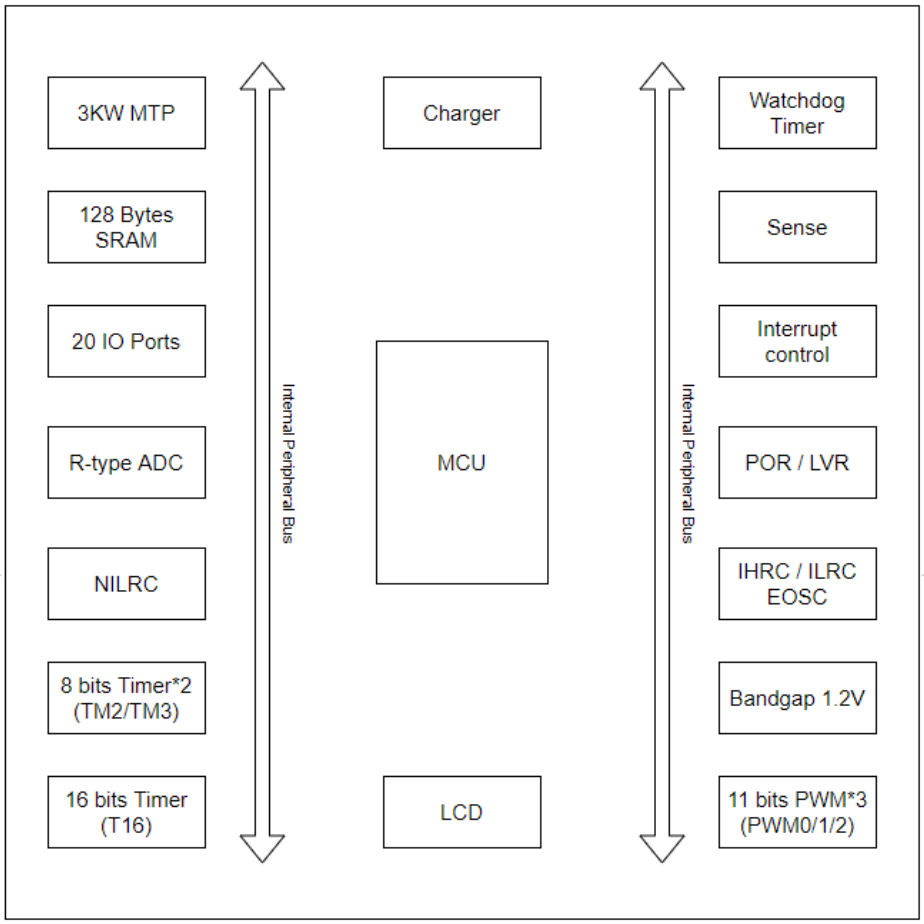
## 2. General Description and Block Diagram

The PFB190 family is an MTP-based CMOS 8-bit microcontroller with Charger, Sense and 12bit ADC. It employs RISC architecture and all the instructions are executed in one cycle except that some instructions are two cycles that handle indirect memory access.

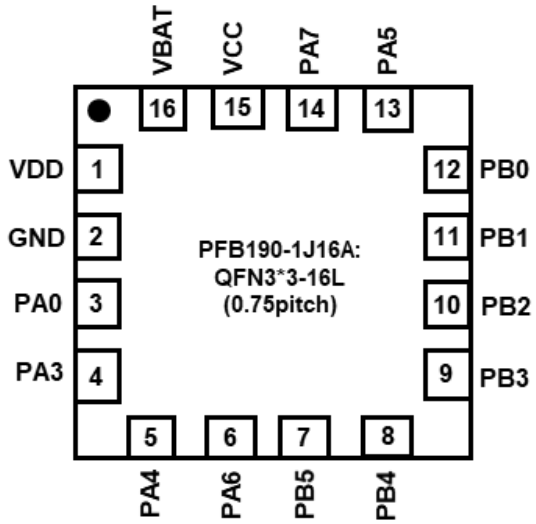
3KW MTP program memory and 128 bytes data SRAM are inside. One up to 13 channels 12-bit ADC is built inside the chip with multiple reference voltage sources selectable. PFB190 also provides six hardware timers: one is 16-bit timer, two are 8-bit timers with PWM generation, and three hardware 11-bit timers with PWM generation are also included. PFB190 also supports VDD/2 LCD bias voltage generator for LCD display application.

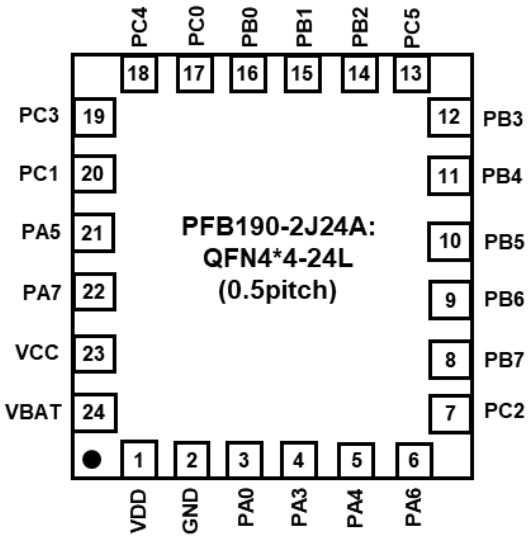
The charger in PFB190 is a constant current/constant voltage charger for single cell Li-Ion batteries and is designed to work within USB power specifications. An internal block regulates the current when the junction temperature increases, in order to protect the device when it operates in high power or high ambient temperature. The charge voltage range is 3.6V ~ 4.28V, and the charge current limitation can be programmed by the registers without the external resistor and the current up to 500mA.

The Sense in PFB190, it has two main functions: Microphone Capacitor Sense (MCS) and Heating Resistor Sense (HRS). In MCS mode, SENSE circuit supports capacitor range from 10pF to 24pF, and typical peak to peak detection noise is 21 ~ 22 LSB. In MEMS-capacitor detection, SENSE circuit supports capacitor of 1.4pF and typical peak to peak detection noise is 20 LSB. In HRS mode, SENSE circuit supports resistor range from 0.5Ω to 1.5Ω, and 0.1Ω resistor variation detection.



## 3. Pin Assignment and Description




| Pin Name | Pin Type & Buffer Type | Description |
|---|---|---|
| PA7 / INT0C / | IO ST / CMOS | The functions of this pin can be:<br>(1) Bit 7 of port A. It can be configured as digital input or two-state output, with pull-high / low resistor.<br>(2) External interrupt line 0. It can be used as an external interrupt line 0. Both rising edge and falling edge are accepted to request interrupt service and configurable by register setting.<br>This pin can be used to wake-up system during sleep mode; however, wake-up function is also disabled if bit 7 of ***padier*** register is "0". |
| PA6 / M2 | CMOS | The functions of this pin can be:<br>M2 is a dedicated pin for MIC detection. |
| PA5 / PRSTB / LCD2 | IO (OD) ST / CMOS | The functions of this pin can be:<br>(1) Bit 5 of port A. It can be configured as digital input or two-state output, with pull-high / low resistor.<br>(2) Hardware reset.<br>(3) LCD2 will provide (1/2 $V_{DD}$) for LCD display for group 2.<br>When this pin is configured as analog input, please use bit 5 of register ***padier*** to disable the digital input to prevent current leakage. The bit 5 of ***padier*** register can be set to "0" to disable digital input; wake-up function by toggling this pin is also disabled. |
| PA4 / M1 | CMOS | The functions of this pin can be:<br>M1 is a dedicated pin for MIC detection. |

| Pin Name | Pin Type & Buffer Type | Description |
|---|---|---|
| PA3 / AD8 / LCD2 / INT1B / TM2PWM / PG2PWM / P2 | IO ST / CMOS / Analog | The functions of this pin can be:<br>(1) Bit 3 of port A. It can be configured as digital input or two-state output, with pull-high / low resistor independently by software.<br>(2) Channel 8 of ADC analog input.<br>(3) LCD2 to provide (1/2 $V_{DD}$) for LCD display for group 2.<br>(4) External interrupt line 1. It can be used as an external interrupt line 1. Both rising edge and falling edge are accepted to request interrupt service and configurable by register setting.<br>(5) PWM output from Timer2.<br>(6) Output of 11-bit PWM generator PWMG2.<br>(7) P2 is constant current detect pin.<br>When this pin is configured as analog input, please use bit 3 of register ***padier*** to disable the digital input to prevent current leakage. The bit 3 of ***padier*** register can be set to "0" to disable digital input; wake-up function by toggling this pin is also disabled. |
| PA0 / AD10 / LCD2 / INT0 / PG0PWM P1 | IO ST / CMOS / Analog | The functions of this pin can be:<br>(1) Bit 0 of port A. It can be configured as digital input or two-state output, with pull-high / low resistor independently by software.<br>(2) Channel 10 of ADC analog input.<br>(3) LCD2 to provide (1/2 $V_{DD}$) for LCD display for group 2.<br>(4) External interrupt line 0. It can be used as an external interrupt line 0. Both rising edge and falling edge are accepted to request interrupt service and configurable by register setting.<br>(5) Output of 11-bit PWM generator PWMG0.<br>(6) P1 is constant current detect pin.<br>When this pin is configured as analog input, please use bit 0 of register ***padier*** to disable the digital input to prevent current leakage. The bit 0 of ***padier*** register can be set to "0" to disable digital input; wake-up function by toggling this pin is also disabled. |

| Pin Name | Pin Type & Buffer Type | Description |
|---|---|---|
| PB7 / AD7 / TM3PWM / PG1PWM LCD0 | IO ST / CMOS / Analog | The functions of this pin can be:<br>(1) Bit 7 of port B. It can be configured as digital input or two-state output, with pull-high / low resistor independently by software<br>(2) Channel 7 of ADC analog input<br>(3) PWM output from Timer3<br>(4) Output of 11-bit PWM generator PWMG1<br>(5) LCD0 to provide (1/2 $V_{DD}$) for LCD display for group 0.<br>When this pin is configured as analog input, please use bit 7 of register ***pbdier*** to disable the digital input to prevent current leakage. The bit 7 of ***pbdier*** register can be set to "0" to disable digital input; wake-up function by toggling this pin is also disabled. |
| PB6 / AD6 / LCD1 / INT1C / TM3PWM / PG1PWM | IO ST / CMOS / Analog | The functions of this pin can be:<br>(1) Bit 6 of port B. It can be configured as digital input or two-state output, with pull-high / low resistor independently by software.<br>(2) Channel 6 of ADC analog input.<br>(3) External interrupt line 1C. It can be used as an external interrupt line 1. Both rising edge and falling edge are accepted to request interrupt service and configurable by register setting.<br>(4) PWM output from Timer3.<br>(5) Output of 11-bit PWM generator PWMG1.<br>When this pin is configured as analog input, please use bit 6 of register ***pbdier*** to disable the digital input to prevent current leakage. The bit 6 of ***pbdier*** register can be set to "0" to disable digital input; wake-up function by toggling this pin is also disabled. |
| PB5 / AD5 / LCD1 / INT0A / TM3PWM / PG0PWM | IO ST / CMOS / Analog | The functions of this pin can be:<br>(1) Bit 5 of port B. It can be configured as digital input or two-state output, with pull-high / low resistor independently by software.<br>(2) Channel 5 of ADC analog input.<br>(3) LCD1 to provide (1/2 $V_{DD}$) for LCD display for group 1.<br>(4) External interrupt line 0A. It can be used as an external interrupt line 0. Both rising edge and falling edge are accepted to request interrupt service and configurable by register setting.<br>(5) PWM output from Timer3.<br>(6) Output of 11-bit PWM generator PWMG0.<br>When this pin is configured as analog input, please use bit 5 of register ***pbdier*** to disable the digital input to prevent current leakage. The bit 5 of ***pbdier*** register can be set to "0" to disable digital input; wake-up function by toggling this pin is also disabled. |

| Pin Name | Pin Type & Buffer Type | Description |
|---|---|---|
| PB4 / AD4 / TM2PWM / PG0PWM | IO ST / CMOS / Analog | The functions of this pin can be:<br>(1) Bit 4 of port B. It can be configured as digital input or two-state output, with pull-high / low resistor independently by software.<br>(2) Channel 4 of ADC analog input.<br>(3) PWM output from Timer2.<br>(4) Output of 11-bit PWM generator PWMG0.<br>When this pin is configured as analog input, please use bit 4 of register ***pbdier*** to disable the digital input to prevent current leakage. The bit 4 of ***pbdier*** register can be set to "0" to disable digital input; wake-up function by toggling this pin is also disabled. |
| PB3 / AD3 / PG2PWM | IO ST / CMOS / Analog | The functions of this pin can be:<br>(1) Bit 3 of port B. It can be configured as digital input or two-state output, with pull-high / low resistor independently by software.<br>(2) Channel 3 of ADC analog input.<br>(3) Output of 11-bit PWM generator PWMG2.<br>When this pin is configured as analog input, please use bit 3 of register ***pbdier*** to disable the digital input to prevent current leakage. The bit 3 of ***pbdier*** register can be set to "0" to disable digital input; wake-up function by toggling this pin is also disabled. |
| PB2 / AD2 / LCD1 / TM2PWM / PG2PWM | IO ST / CMOS / Analog | The functions of this pin can be:<br>(1) Bit 2 of port B. It can be configured as digital input or two-state output, with pull-high / low resistor independently by software.<br>(2) Channel 2 of ADC analog input.<br>(3) LCD1 to provide (1/2 $V_{DD}$) for LCD display for group 1.<br>(4) PWM output from Timer2.<br>(5) Output of 11-bit PWM generator PWMG2.<br>When this pin is configured as analog input, please use ***pbdier.2*** to disable the digital input to prevent current leakage. The ***pbdier.2*** can be set to "0" to disable digital input; wake-up functionby toggling this pin is also disabled. |
| PB1 / AD1 / LCD1 | IO ST / CMOS / Analog | The functions of this pin can be:<br>(1) Bit 1 of port B. It can be configured as digital input or two-state output, with pull-high / low resistor independently by software.<br>(2) Channel 1 of ADC analog input.<br>(3) LCD1 to provide (1/2 $V_{DD}$) for LCD display for group 1.<br>When this pin is configured as analog input, please use bit 1 of register ***pbdier*** to disable the digital input to prevent current leakage. The bit 1 of ***pbdier*** register can be set to "0" to disable digital input; wake-up function by toggling this pin is also disabled. |

| Pin Name | Pin Type & Buffer Type | Description |
|---|---|---|
| PB0 / AD0 / LCD2 / INT1 | IO ST / CMOS / Analog | The functions of this pin can be:<br>(1) Bit 0 of port B. It can be configured as digital input or two-state output, with pull-high / low resistor independently by software.<br>(2) Channel 0 of ADC analog input.<br>(3) LCD2 to provide (1/2 $V_{DD}$) for LCD display for group 2.<br>(4) External interrupt line 1. It can be used as an external interrupt line 1. Both rising edge and falling edge are accepted to request interrupt service and configurable by register setting.<br>When this pin is configured as analog input, please use bit 0 of register ***pbdier*** to disable the digital input to prevent current leakage. The bit 0 of ***pbdier*** register can be set to "0" to disable digital input; wake-up function by toggling this pin is also disabled. |
| PC6 / LCD0 | IO ST / CMOS | The function of this pin can be:<br>(1) Bit 6 of port C. It can be configured as digital input, two-state output with pull-high / low resistor independently by software.<br>(2) LCD0 to provide (1/2 $V_{DD}$) for LCD display for group.<br>The bit 6 of ***pcdier*** register can be set to "0" to disable digital input; wake-up from power-down by toggling this pin is also disabled. |
| PC5 / LCD0 | IO ST / CMOS | The function of this pin can be:<br>(1) Bit 5 of port C. It can be configured as digital input, two-state output with pull-high / low resistor independently by software.<br>(2) LCD0 to provide (1/2 $V_{DD}$) for LCD display for group.<br>The bit 5 of ***pcdier*** register can be set to "0" to disable digital input; wake-up from power-down by toggling this pin is also disabled. |
| PC4 / LCD0 | IO ST / CMOS | The function of this pin can be:<br>(1) Bit 4 of port C. It can be configured as digital input, two-state output with pull-high / low resistor independently by software.<br>(2) LCD0 to provide (1/2 $V_{DD}$) for LCD display for group.<br>The bit 4 of ***pcdier*** register can be set to "0" to disable digital input; wake-up from power-down by toggling this pin is also disabled. |
| PC3 / PG1PWM / LCD0 | IO ST / CMOS | The functions of this pin can be:<br>(1) Bit 3 of port B. It can be configured as digital input or two-state output, with pull-high / low resistor independently by software.<br>(2) Output of 11-bit PWM generator PWMG1.<br>(3) LCD0 to provide (1/2 $V_{DD}$) for LCD display for group.<br>The bit 3 of ***pcdier*** register can be set to "0" to disable digital input; wake-up function by toggling this pin is also disabled. |

| Pin Name | Pin Type & Buffer Type | Description |
|---|---|---|
| PC2 / AD12 / PG0PWM / LCD0 | IO ST / CMOS / Analog | The functions of this pin can be:<br>(1) Bit 2 of port C. It can be configured as digital input or two-state output, with pull-high / low resistor independently by software.<br>(2) Channel 12 of ADC analog input.<br>(3) Output of 11-bit PWM generator PWMG0.<br>(4) LCD0 to provide (1/2 $V_{DD}$) for LCD display for group.<br>When this pin is configured as analog input, please use bit 2 of register ***pcdier*** to disable the digital input to prevent current leakage. The bit 2 of ***pcdier*** register can be set to "0" to disable digital input; wake-up function by toggling this pin is also disabled. |
| PC1 / AD11 / LCD0 | IO ST / CMOS / Analog | The functions of this pin can be:<br>(1) Bit 1 of port C. It can be configured as digital input or two-state output, with pull-high / low resistor independently by software.<br>(2) Channel 11 of ADC analog input.<br>(3) LCD0 to provide (1/2 $V_{DD}$) for LCD display for group.<br>When this pin is configured as analog input, please use bit 2 of register ***pcdier*** to disable the digital input to prevent current leakage. The bit 2 of ***pcdier*** register can be set to "0" to disable digital input; wake-up function by toggling this pin is also disabled. |
| PC0 / PG2PWM LCD0 | IO ST / CMOS | The functions of this pin can be:<br>(1) Bit 0 of port C. It can be configured as digital input or two-state output, with pull-high / low resistor independently by software.<br>(2) Output of 11-bit PWM generator PWMG2.<br>(3) LCD0 to provide (1/2 $V_{DD}$) for LCD display for group.<br>The bit 0 of ***pcdier*** register can be set to "0" to disable digital input; wake-up function by toggling this pin is also disabled. |
| VDD/ VBAT/ VCC | VDD/ VBAT | VDD: Digital positive power.<br>VBAT: Charger Battery positive power.<br>VCC: Charger positive power. |
| GND | GND | GND: Digital negative power |
| Notes: **IO**: Input/Output; **ST**: Schmitt Trigger input; **OD**: Open Drain; **Analog**: Analog input pin<br>**CMOS**: CMOS voltage level | | |

# 4. Device Characteristics

## 4.1. AC/DC Device Characteristics

All data are acquired under the conditions of Ta= -40 °C ~ 85 °C, $V_{BAT}$ =5.0V, $f_{SYS}$ =2MHz unless noted.

| Symbol | Description | Min | Typ | Max | Unit | Conditions (Ta=25°C) |
|---|---|---|---|---|---|---|
| $V_{BAT}$ / $V_{DD}$ | Operating Voltage | 1.8# | 5.0 | 5.5 | V | # Subject to LVR tolerance |
| VCC | Charger Input Supply Voltage | 4.3 | 5 | 6.5 | V | |
| LVR% | Low Voltage Reset Tolerance | -5 | | 5 | % | |
| $f_{SYS}$ | System clock (CLK)* =<br>IHRC/2<br>IHRC/4<br>IHRC/8<br>ILRC | | <br>8M<br>4M<br>2M<br>90K | | Hz | <br>$V_{DD}$ ≧ 3.5V<br>$V_{DD}$ ≧ 2.3V<br>$V_{DD}$ ≧ 1.8V<br>$V_{DD}$ = 5.0V |
| $V_{POR}$ | Power On Reset Voltage | | 2.0* | | V | * Subject to LVR tolerance |
| $I_{OP}$ | Operating Current | | 0.7<br>426 | | mA<br>uA | $f_{SYS}$=IHRC/16=1MIPS@5.0V<br>$f_{SYS}$=ILRC=90KHz@5.0V |
| $I_{PD}$ | Power Down Current<br>(by ***stopsys*** command) | | 0.7<br>0.3 | | uA<br>uA | $f_{SYS}$= 0Hz, $V_{BAT}$ =5.0V<br>$f_{SYS}$= 0Hz, $V_{BAT}$ =3.0V |
| $I_{PS}$ | Power Save Current<br>(by ***stopexe*** command) | | 2.9 | | uA | $V_{BAT}$ =5.0V; $f_{SYS}$= ILRC<br>Only ILRC module is enabled. |
| $V_{IL}$ | Input low voltage for IO lines | 0 | | 0.2 $V_{BAT}$ | V | |
| $V_{IH}$ | Input high voltage for IO lines | 0.7 $V_{BAT}$ | | $V_{BAT}$ | V | |
| $I_{OL}$ | IO lines sink current | | | | | |
| | PA0, PA3, PA5, PA7,<br>Port B & Port C<br>Strong<br>Normal | | <br><br>62<br>32 | | mA | $V_{BAT}$ =5.0V, $V_{OL}$=0.5V |
| $I_{OH}$ | IO lines drive current | | | | | |
| | All IO | | -20 | | mA | $V_{BAT}$ =5.0V, $V_{OH}$=4.5V |
| $V_{IN}$ | Input voltage | -0.3 | | $V_{BAT}$ +0.3 | V | |
| $I_{INJ\ (PIN)}$ | Injected current on pin | | | 1 | mA | $V_{BAT}$ +0.3≧$V_{IN}$≧ -0.3 |
| $R_{PH}$ | Pull-high Resistance | | 72 | | KΩ | $V_{BAT}$ =5.0V |
| $R_{PL}$ | Pull-low Resistance | | 72 | | KΩ | $V_{BAT}$ =5.0V |
| $V_{BG}$ | Bandgap Reference Voltage | 1.145* | 1.20* | 1.255* | V | $V_{BAT}$ =2.2V ~ 5.5V<br>-40°C <Ta< 85°C* |
| $f_{IHRC}$ | Frequency of IHRC after calibration * | 15.76* | 16* | 16.24* | MHz | 25°C, $V_{BAT}$ =2.2V~5.5V |
| | | 15.20* | 16* | 16.80* | | $V_{BAT}$ =2.2V~5.5V,<br>-40°C <Ta< 85°C* |
| | | 13.60* | 16* | 18.40* | | $V_{BAT}$ =1.8V~5.5V,<br>-40°C <Ta< 85°C |
| $f_{ILRC}$ | Frequency of ILRC * | | 90 | | KHz | $V_{BAT}$ = 5.0V |

| Symbol | Description | Min | Typ | Max | Unit | Conditions (Ta=25°C) |
|---|---|---|---|---|---|---|
| $f_{NILRC}$ | Frequency of NILRC * | | 15 | | KHz | $V_{BAT}$ = 5.0V |
| $t_{INT}$ | Interrupt pulse width | 30 | | | ns | $V_{BAT}$ = 5.0V |
| $V_{AD}$ | AD Input Voltage | 0 | | $V_{DD}$ | V | |
| ADrs | ADC resolution | | | 12<br>10 | bit | 0°C <Ta<50°C*<br>-40°C <Ta<85°C* |
| ADcs | ADC current consumption | | 0.9<br>0.8 | | mA | @5.0V<br>@3.0V |
| ADclk | ADC clock period | | 2 | | us | 1.8V ~ 5.5V |
| $t_{ADCONV}$ | ADC conversion time ($t_{ADCLK}$ is the period of the selected AD conversion clock) | | 16 | | $t_{ADCLK}$ | 12-bit resolution |
| AD DNL | ADC Differential Non-Linearity | | ±4* | | LSB | 12-bit resolution LSB |
| AD INL | ADC Integral Non-Linearity | | ±8* | | LSB | 12-bit resolution LSB |
| ADos | ADC offset | | 5* | | mV | @ $V_{DD}$ =3.0V |
| $N_{MCS}$ | MCS detection noise | | 21 | | LSB | $C_m$ = 10 pF,<br>12-bit resolution LSB |
| | | | 22 | | LSB | $C_m$ = 24 pF,<br>12-bit resolution LSB |
| | | | 20 | | LSB | $C_{MEMS-capacitor}$ = 1.4 pF,<br>12-bit resolution LSB |
| $V_{DR}$ | RAM data retention voltage* | 1.5 | | | V | in stop mode |
| $t_{WDT}$ | Watchdog timeout period | | 8K | | $T_{ILRC}$ | misc[1:0]=00 (default) |
| | | | 16K | | | misc[1:0]=01 |
| | | | 64K | | | misc[1:0]=10 |
| | | | 256K | | | misc[1:0]=11 |
| $t_{WUP}$ | Wake-up time period for fast wake-up | | 45 | | $T_{ILRC}$ | Where $T_{ILRC}$ is the time period of ILRC |
| | Wake-up time period for slow wake-up | | 3000 | | | |
| $t_{SBP}$ | System boot-up period from power-on boot-up | | 32 | | ms | $V_{BAT}$ = 5.0V |
| $t_{RST}$ | External reset pulse width | 120 | | | us | @ $V_{BAT}$ = 5.0V |
| $I_{VCC}$ | Charger Input Supply Current | | 200 | 500 | μA | Charging Mode |
| | | | 57 | | | Standby Mode |
| | | | 38 | | | Shutdown mode |
| | | | 0 | | | Sleep mode |
| $I_{CCM}$ | Constant Current Mode Charge Current | -15% | 145 | +15% | mA | @VCC=5V |
| | | | 275 | | | |
| | | | 375 | | | |
| | | | 500 | | | |

| Symbol | Description | Min | Typ | Max | Unit | Conditions (Ta=25°C) |
|---|---|---|---|---|---|---|
| $V_{ASD}$ | VCC-$V_{BAT}$ lockout threshold voltage | | 100 | | mV | VCC rising |
| | | | 30 | | mV | VCC falling |
| $t_{RECHA}$ | Recharge Comparator Filter Time | | 2 | | mS | $V_{BAT}$ High to Low |
| $t_{TERM}$ | Termination Comparator Filter Time | | 1 | | mS | $I_{CCM}$ is less than 1/10 |
| $I_{TERM}$ | C/10 termination current threshold | | 0.1 | | | |
| △VRECHA | Recharge Battery Threshold Voltage. | | 150 | | mV | |
| $T_{LIM}$ | Junction Temperature in Constant Temperature Mode | | 90 | | ℃ | |
| $V_{OVP}$ | Overvoltage Protection Threshold Voltage | 6.5 | 6.9 | 7.3 | V | |
| $V_{OVPHYS}$ | | | 0.8 | | V | |
| Irms | RMS Current Source | -5% | 50 | +5% | mA | |

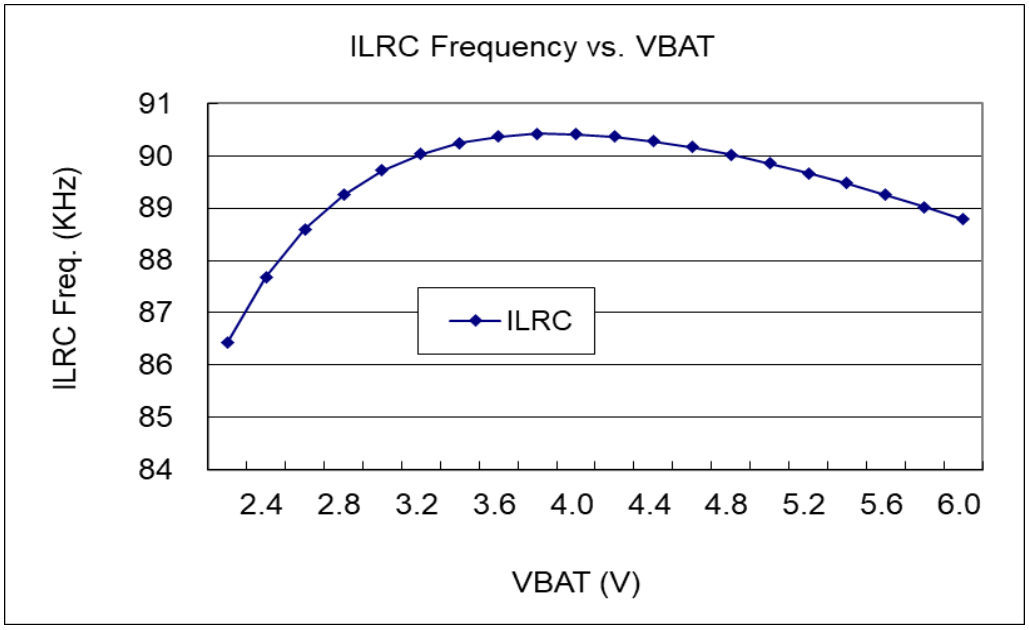
* These parameters are for design reference, not tested for each chip.

**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.

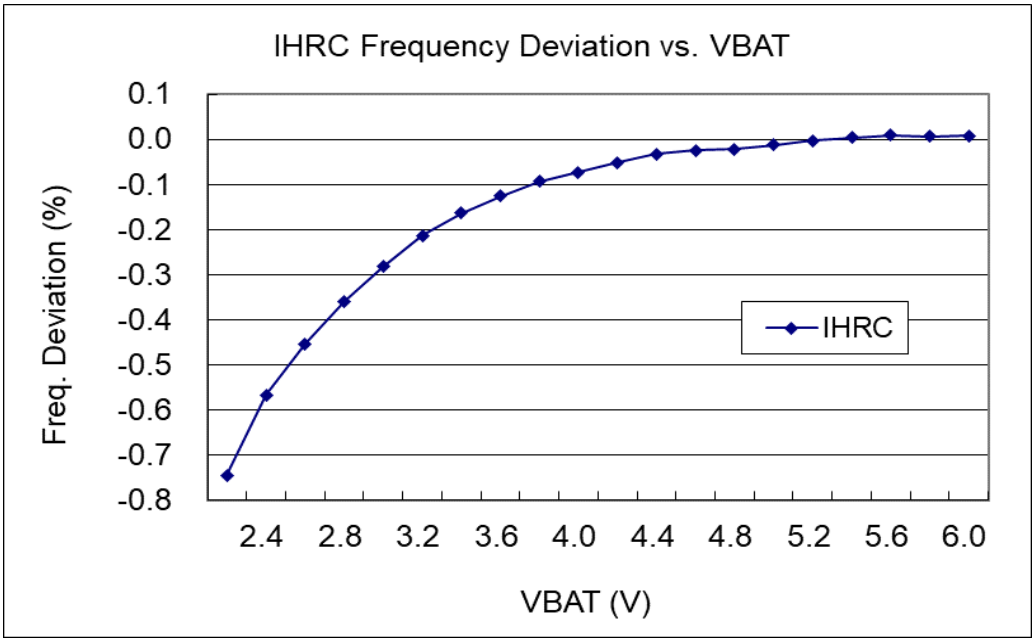
## 4.2. Absolute Maximum Ratings

| Item | Value Range / Maximum | Note |
|---|---|---|
| Supply Voltage (VBAT, VDD) | 1.8V ~ 5.5V (Maximum Rating: 5.5V) | If $V_{BAT}$ / $V_{DD}$ exceeds max rating, the IC may be damaged. |
| Supply Voltage VCC | Maximum Rating 24V | 1. When VCC is higher than $V_{OVP}$, it stops charge.<br>2. When VCC is short to power source higher than 24V for more than 20ms, chip will be damaged. |
| Input Voltage | -0.3V ~ $V_{BAT}$ + 0.3V | |
| Operating Temperature | -40°C ~ 85°C | |
| Storage Temperature | -50°C ~ 125°C | |
| Junction Temperature | 150°C | |

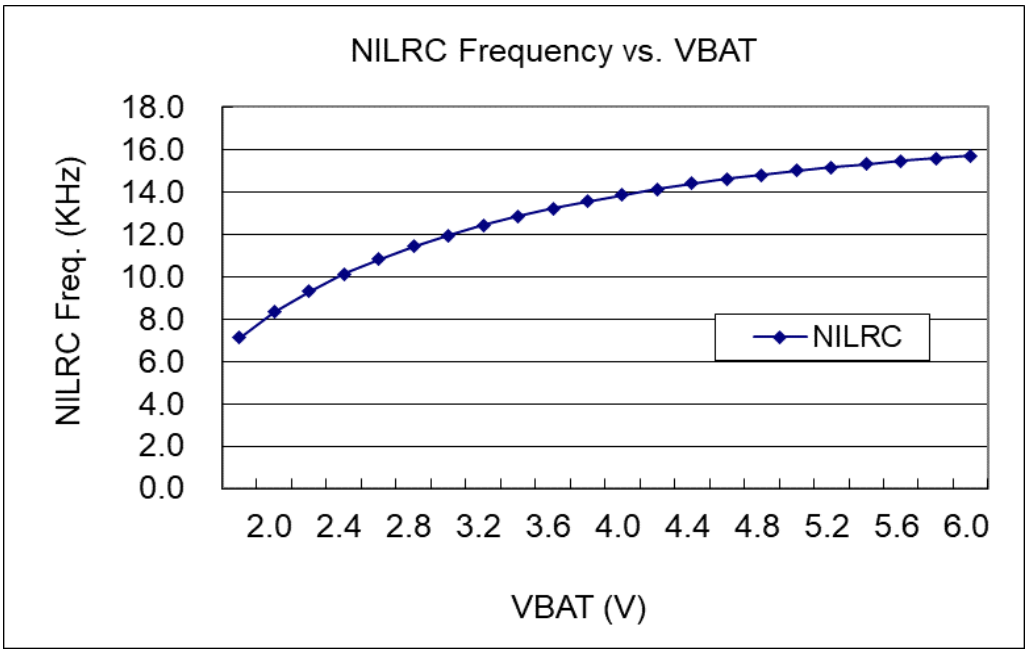
## 4.3. Typical ILRC frequency vs. $V_{BAT}$ ($V_{BAT}$ = $V_{DD}$)



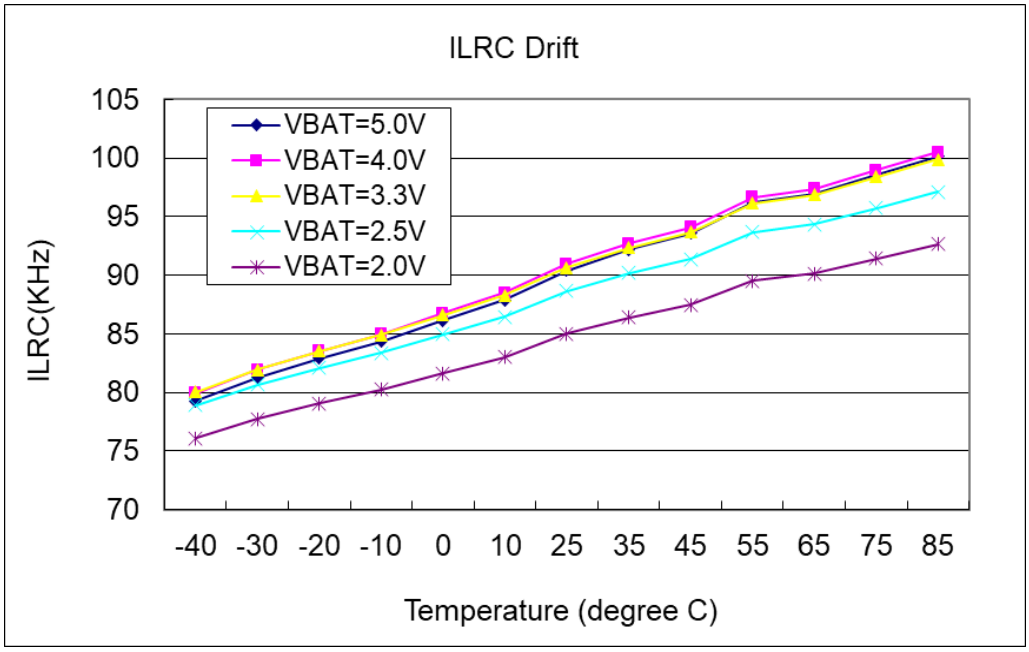
### 4.4. Typical IHRC frequency deviation vs. $V_{BAT}$ (calibrated to 16MHz, $V_{BAT} = V_{DD}$)



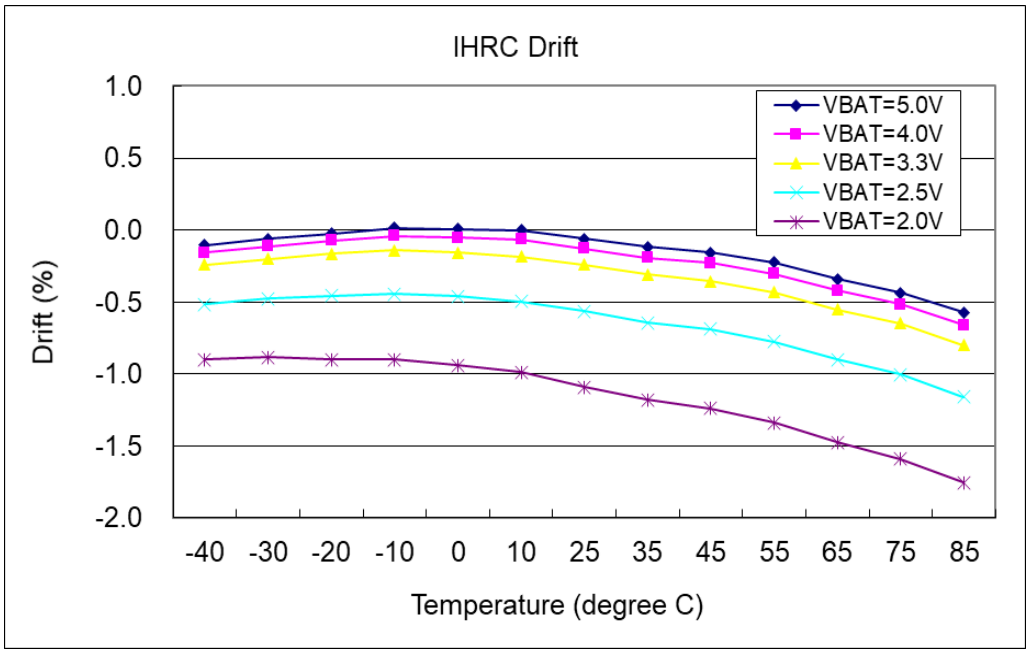

### 4.5. Typical NILRC Frequency vs. $V_{BAT}$ ($V_{BAT} = V_{DD}$)



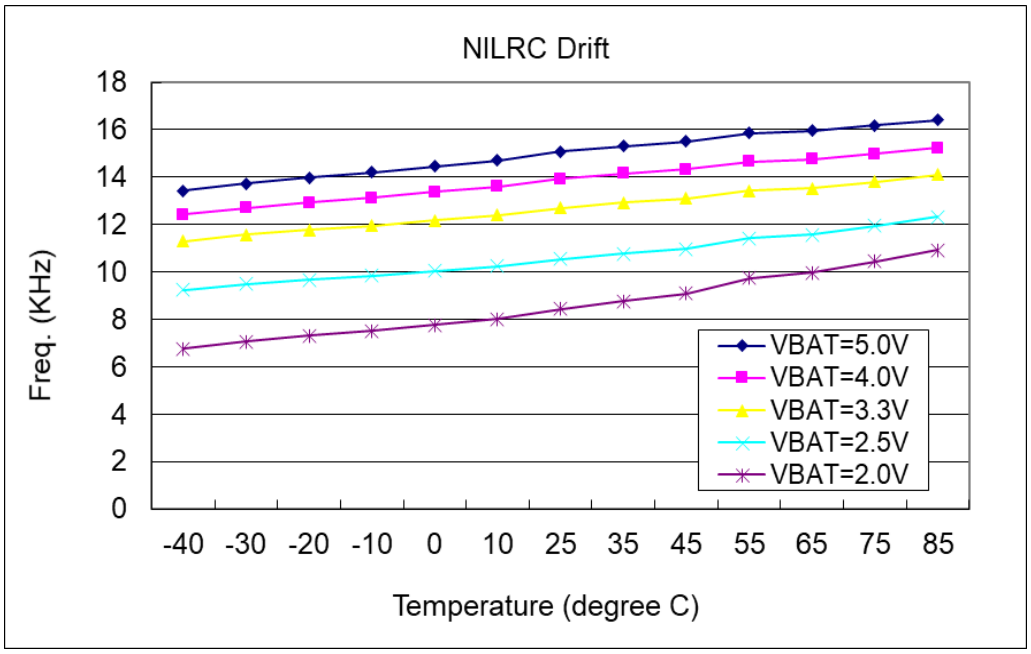
## 4.6. Typical ILRC Frequency vs. Temperature ($V_{BAT}$ = $V_{DD}$)




## 4.7. Typical IHRC Frequency vs. Temperature (calibrated to 16MHz, $V_{BAT}$ = $V_{DD}$)

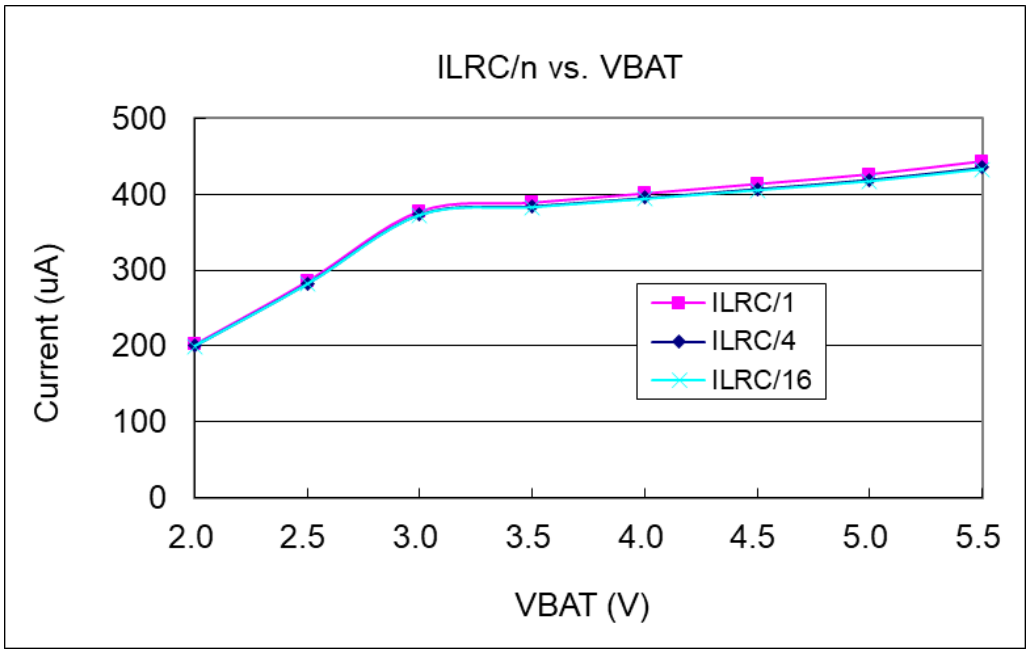


## 4.8. Typical NILRC Frequency vs. Temperature ($V_{BAT}$ = $V_{DD}$)



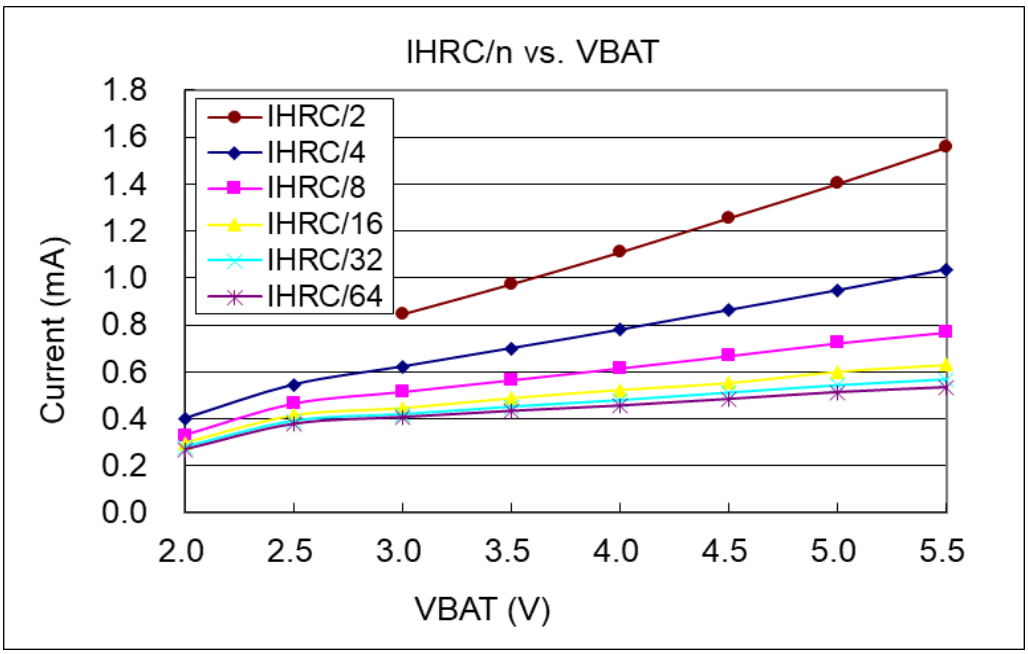

## 4.9. Typical operating current vs. $V_{BAT}$ @ system clock = ILRC/n ($V_{BAT}$ = $V_{DD}$)

- Conditions: **ON**: Bandgap, LVR, ILRC; **OFF**: IHRC, T16, TM2, LPWM, GPC;
  **IO**: PA0:0.5Hz output toggle and no loading, **others**: input and no floating



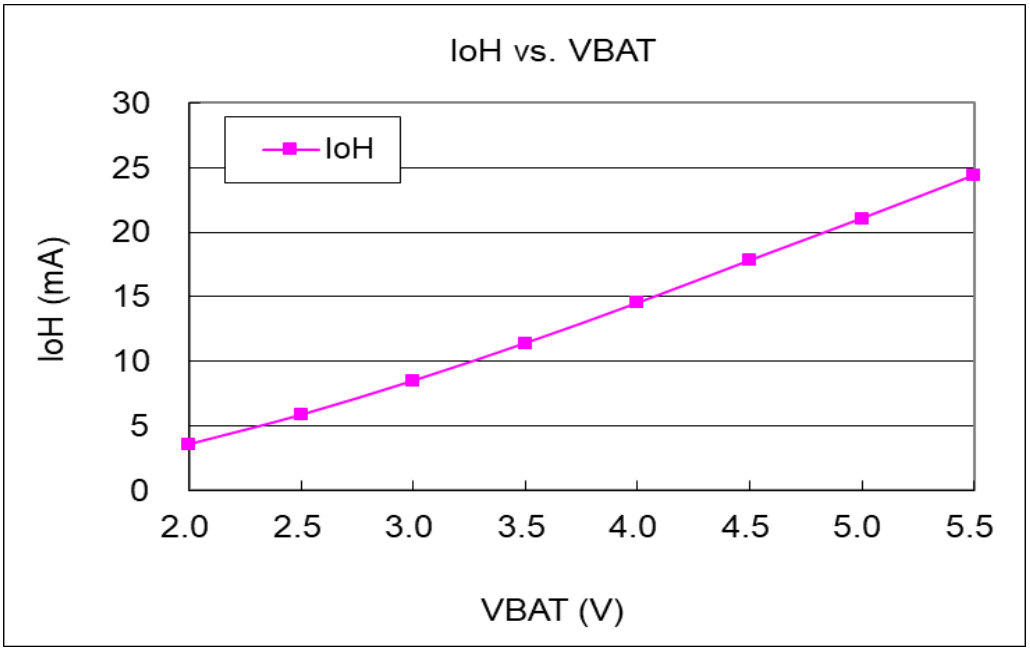
## 4.10. Typical operating current vs. $V_{BAT}$ @ system clock = IHRC/n ($V_{BAT}$ = $V_{DD}$)

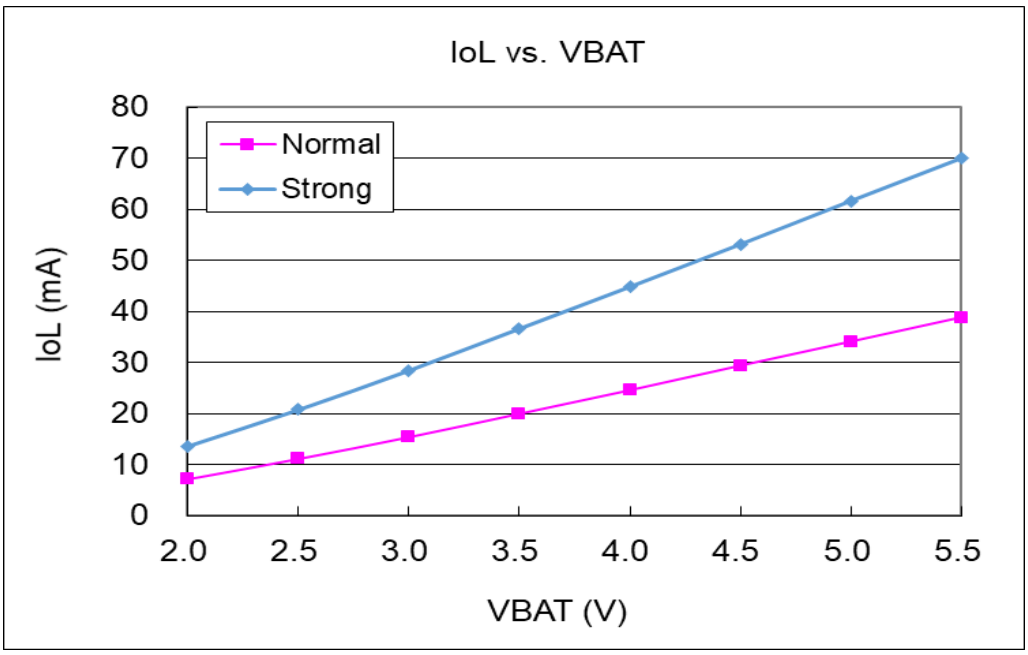
- Conditions: **ON**: Bandgap, LVR, IHRC; **OFF**: ILRC, T16, TM2, LPWM, GPC;
  **IO**: PA0:0.5Hz output toggle and no loading, **others**: input and no floating.



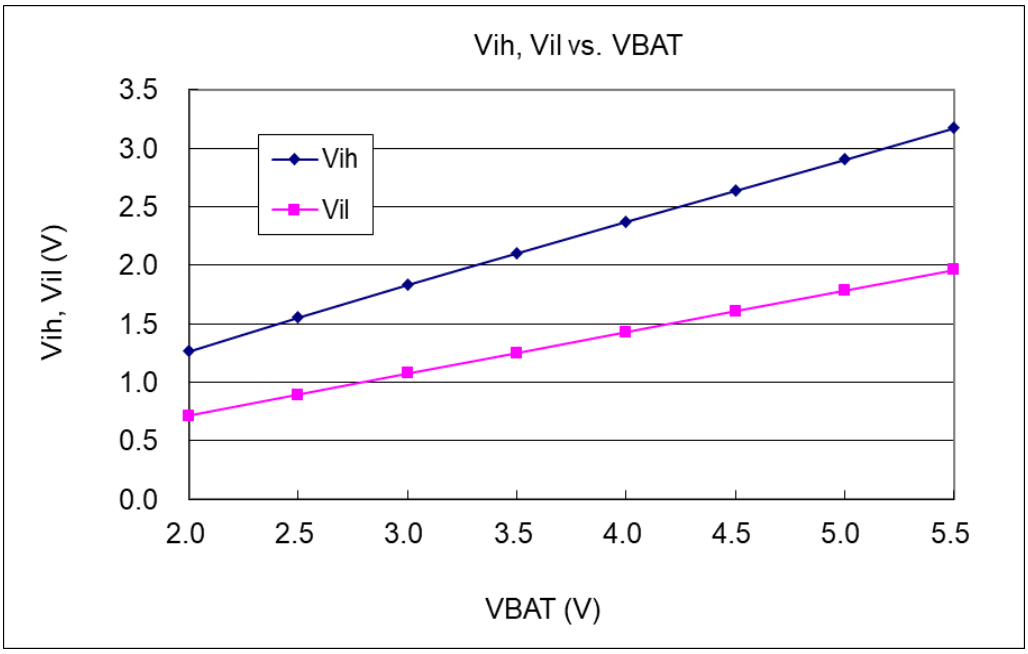

## 4.11. IO driving current ($I_{OH}$) and sink current ($I_{OL}$) Curve($V_{BAT}$ = $V_{DD}$)

**(VOH=0.9* $V_{BAT}$, VOL=0.1* $V_{BAT}$)**

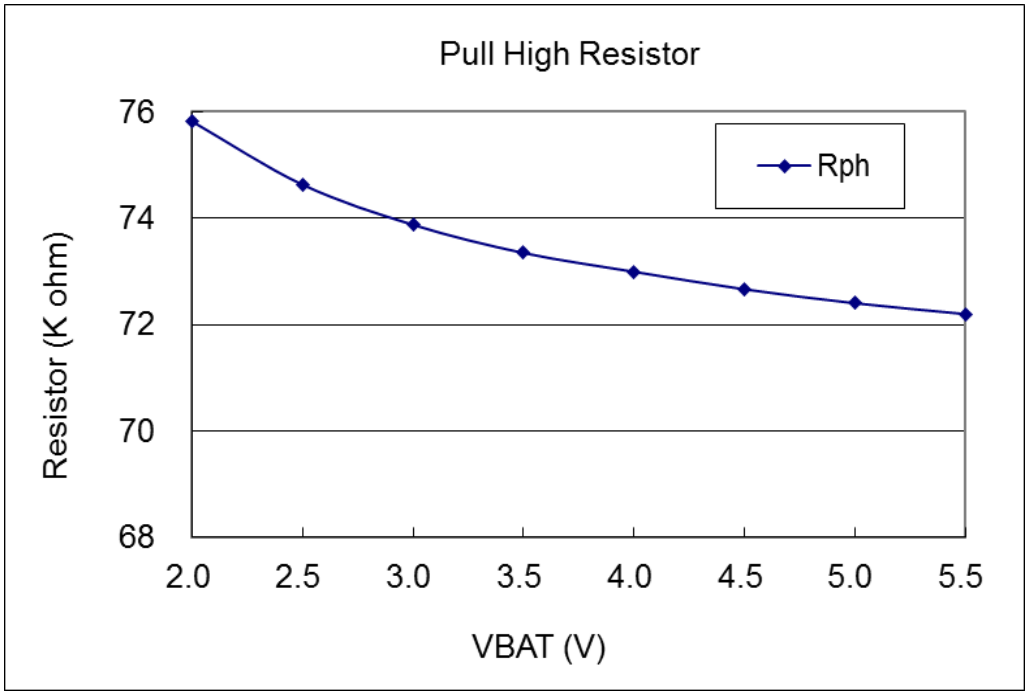


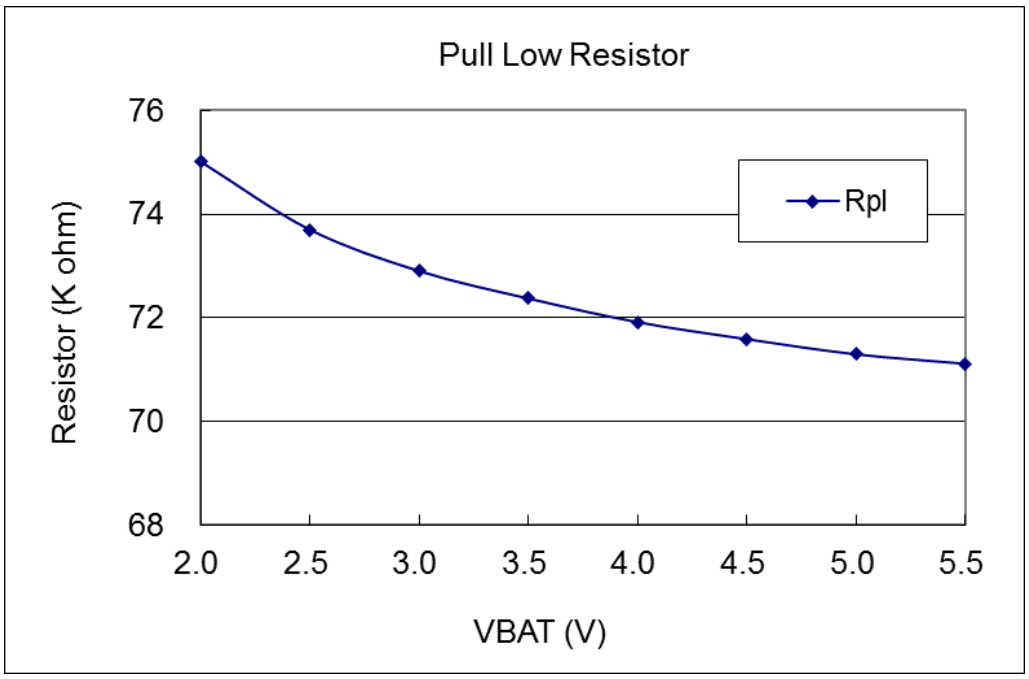



## 4.12. IO Pin Input High/Low Threshold Voltage Curve ($V_{IH}/V_{IL}$) ($V_{BAT} = V_{DD}$)

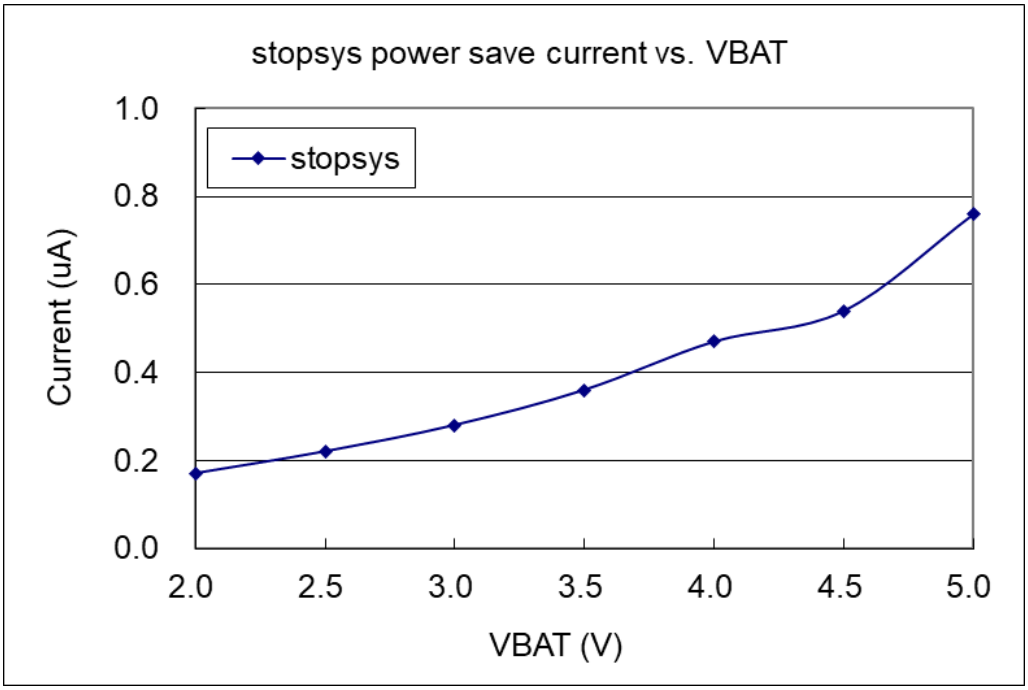


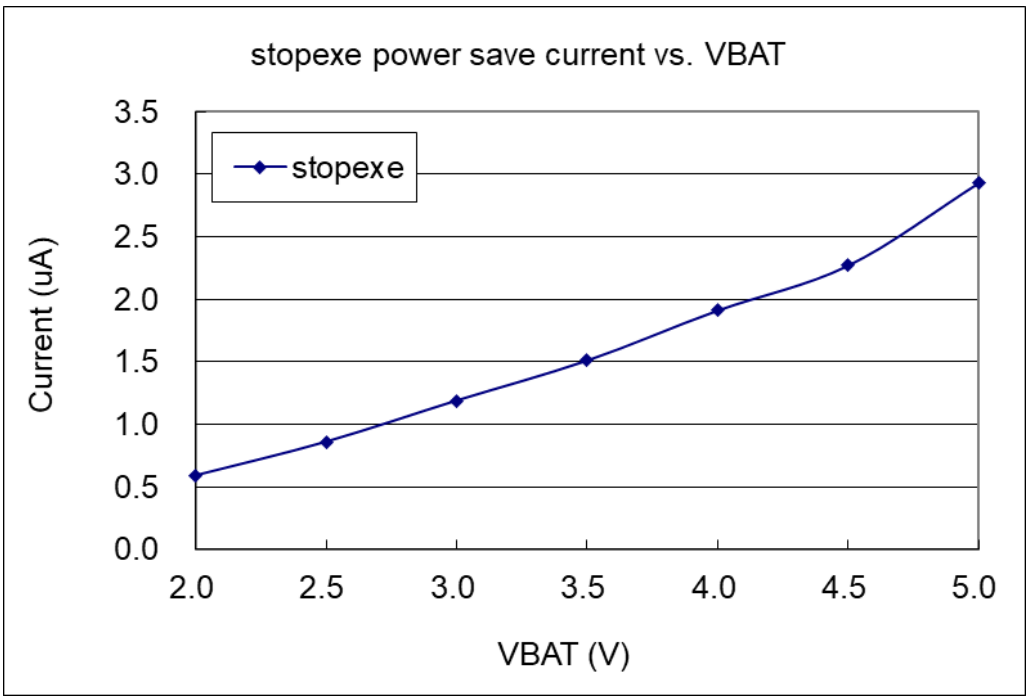
### 4.13. IO Pin Pull-up/Pull-down Impedance Curve ($V_{BAT}$ = $V_{DD}$)



## 4.14. Curve of Power-Down Current Consumption (IPD) vs. Power-Saving Current Consumption (IPS) (VBAT = VDD)



# 5. Functional Description

## 5.1 Program Memory - MTP

The MTP (Multiple Time Programmable) program memory is used to store the program instructions to be executed. The MTP program memory may contains the data, tables and interrupt entry. After reset, the program will start from the initial address 0x000 which is GOTO FPPA0 instruction usually. The interrupt entry is 0x10 if used, the last 32 addresses are reserved for system using, like checksum, serial number, etc. The MTP program memory for PFB190 is 3KW that is partitioned as Table 1. The MTP memory from address 0XBE0 to 0xBFF is for system using, address space from 0x001 to 0x00F and from 0x011 to 0XBDF are user program spaces.

| Address | Function |
|---|---|
| 0x000 | GOTO FPPA0 instruction |
| 0x001 | User program |
| . | . |
| 0x00F | User program |
| 0x010 | Interrupt entry address |
| 0x011 | User program |
| . | . |
| 0xBDF | User program |
| 0XBE0 | System Using |
| . | . |
| 0xBFF | System Using |

Table 1: Program Memory Organization

## 5.2 Boot Procedure

POR (Power-On-Reset) is used to reset PFB190 when power up. The boot up time can be optional fast or normal. Customer must ensure the stability of supply voltage after power up no matter which option is chosen, the power up sequence is shown in the Fig. 1 and $t_{SBP}$ is the boot up time.

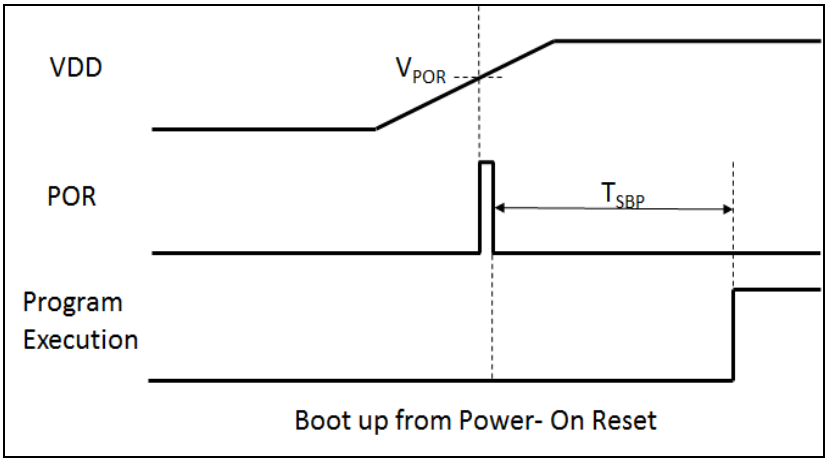
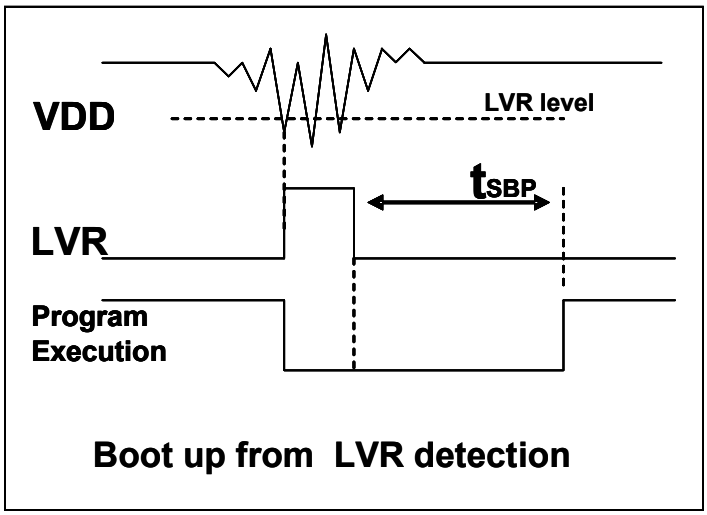
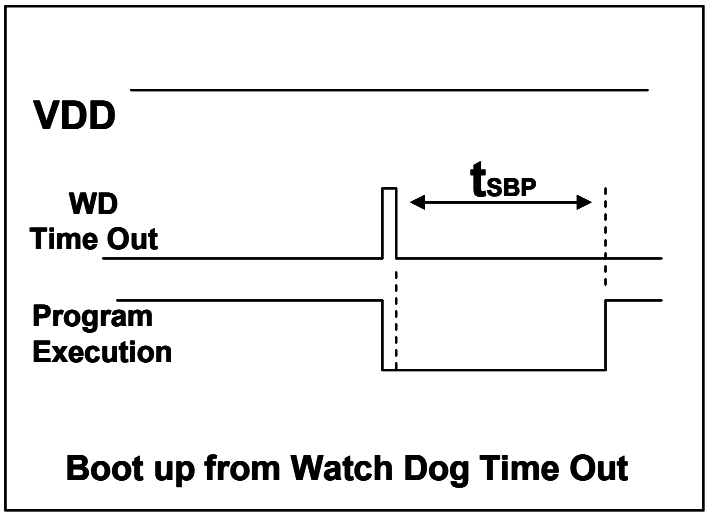
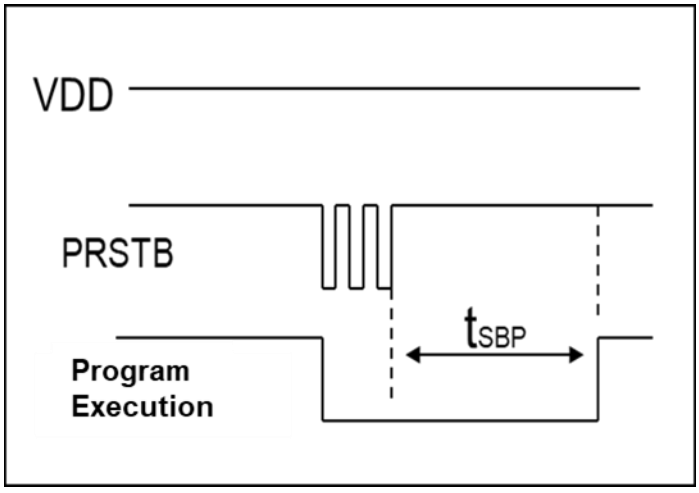



Fig.1: Power-Up Sequence

### 5.2.1 Timing charts for reset conditions



## 5.3 Data Memory - SRAM

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.

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.

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. Since the data width is 8-bit, all the 128 bytes data memory of PFB190 can be accessed by indirect access mechanism.

## 5.4 Oscillator and clock

There are three oscillator circuits provided by PFB190: internal high RC oscillator (IHRC), internal low RC oscillator (ILRC) and low power oscillator (NILRC). IHRC and ILRC are enabled or disabled by registers clkmd.4 and clkmd.2 independently. User can choose one of these two oscillators as system clock source and use ***clkmd*** register to target the desired frequency as system clock to meet different applications. NILRC is always enabled and is used to wake up the system from stopsys.

| Oscillator Module | Enable/Disable |
|---|---|
| IHRC | clkmd.4 |
| ILRC | clkmd.2 |
| NILRC | Always Enable |

Table 2: Three oscillation circuits

### 5.4.1 Internal High RC oscillator and Internal Low RC oscillator

After boot-up, the IHRC and ILRC oscillators are enabled. The frequency of IHRC can be calibrated to eliminate process variation by ***ihrcr*** register; normally it is calibrated to 16MHz. Please refer to the measurement chart for IHRC frequency verse VDD and IHRC frequency verse temperature.

The frequency of ILRC will vary by process, supply voltage and temperature, please refer to DC specification and do not use for accurate timing application.

### 5.4.2 Chip calibration

The IHRC frequency and bandgap reference voltage may be different chip by chip due to manufacturing variation, PFB190 provide the IHRC frequency calibration to eliminate this variation, and this function can be selected when compiling user's program and the command will be inserted into user's program automatically. The calibration command is shown as below:

.ADJUST_IC SYSCLK=IHRC/(**p1**), IHRC=(**p2**)MHz, $V_{DD}$=(**p3**)V;

Where, **p1**=2, 4, 8, 16, 32; In order to provide different system clock.
**p2**=14 ~ 18; In order to calibrate the chip to different frequency, 16MHz is the usually one.
**p3**=2.5 ~ 5.5; In order to calibrate the chip under different supply voltage.

### 5.4.3 IHRC Frequency Calibration and System Clock

During compiling the user program, the options for IHRC calibration and system clock are shown as Table 3:

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

Table 3: Options for IHRC Frequency Calibration

Usually, .ADJUST_IC will be the first command after boot up, in order to set the target operating frequency whenever starting the system. The program code for IHRC frequency calibration is executed only one time that occurs in writing the codes into MTP memory; after then, it will not be executed again. If the different option for IHRC calibration is chosen, the system status is also different after boot. The following shows the status of PFB190 for different option:

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

After boot up, CLKMD = 0x34：

- IHRC frequency is calibrated to 16MHz@$V_{DD}$=5V and IHRC module is enabled
- System CLK = IHRC/2 = 8MHz
- Watchdog timer is disabled, ILRC is enabled, PA5 is in input mode

**(2)** .ADJUST_IC SYSCLK=IHRC/4, IHRC=16MHz, $V_{DD}$=3.3V

After boot up, CLKMD = 0x14：

- IHRC frequency is calibrated to 16MHz@$V_{DD}$=3.3V and IHRC module is enabled
- System CLK = IHRC/4 = 4MHz
- Watchdog timer is disabled, ILRC is enabled, PA5 is in input mode

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

After boot up, CLKMD = 0x3C：

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

**(4)** .ADJUST_IC SYSCLK=IHRC/16, IHRC=16MHz, $V_{DD}$=2.5V

After boot up, CLKMD = 0x1C：

- IHRC frequency is calibrated to 16MHz@$V_{DD}$=2.5V and IHRC module is enabled
- System CLK = IHRC/16 = 1MHz
- Watchdog timer is disabled, ILRC is enabled, PA5 is in input mode

**(5)** .ADJUST_IC SYSCLK=IHRC/32, IHRC=16MHz, $V_{DD}$=5V

After boot up, CLKMD = 0x7C：

- IHRC frequency is calibrated to 16MHz@$V_{DD}$=5V and IHRC module is enabled
- System CLK = IHRC/32 = 500KHz
- Watchdog timer is disabled, ILRC is enabled, PA5 is in input mode

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

After boot up, CLKMD = 0XE4：

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

**(7)** .ADJUST_IC DISABLE

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

- IHRC is not calibrated and IHRC module is disabled by Boot-up Time
- System CLK = ILRC or IHRC/64 (by Boot-upTime)
- Watchdog timer is enabled, ILRC is enabled, PA5 is in input mode

### 5.4.4 System Clock and LVR level

The clock source of system clock comes from IHRC and ILRC, the hardware diagram of system clock in the PFB190 is shown as Fig.2.

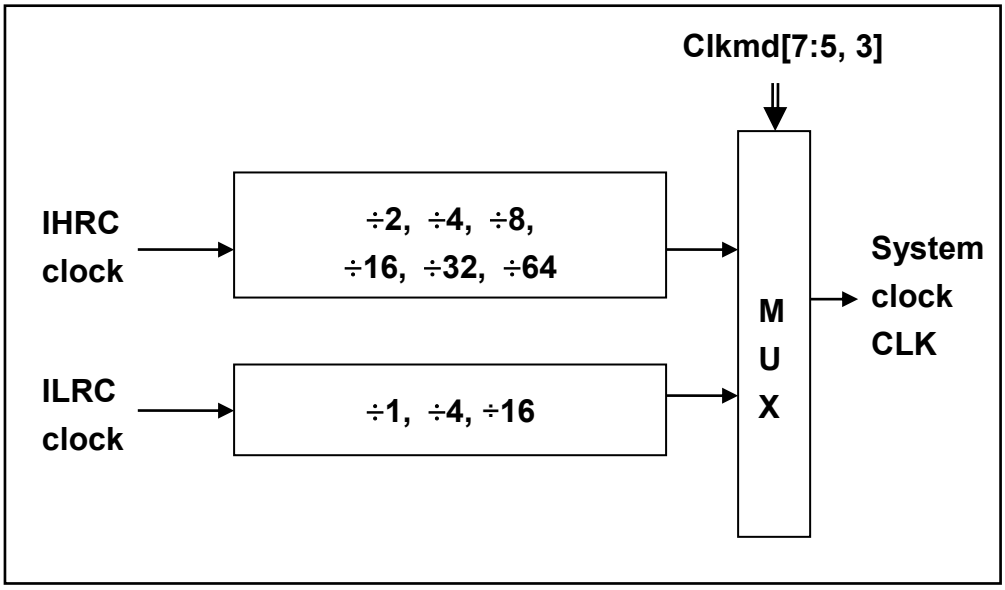



Fig.2: Options of System Clock

User can choose different operating system clock depends on its requirement; the selected operating system clock should be combined with supply voltage and LVR level to make system stable. The LVR level will be selected during compilation. Please refer to Section 4.1.

### 5.4.5 System Clock Switching

After IHRC calibration, user may want to switch system clock to a new frequency or may switch system clock at any time to optimize the system performance and power consumption. Basically, the system clock of PFB190 can be switched among IHRC and ILRC by setting the ***clkmd*** register at any time; system clock will be the new one after writing to ***clkmd*** register immediately. **Please notice that the original clock module can NOT be turned off at the same time as writing command to *clkmd* register.** 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

```
...
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 ILRC

```
...
CLKMD       =   0xF4 ;  // switch to ILRC, 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

```
...
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

```
...
CLKMD       =   0x30 ;  // CAN NOT switch clock from ILRC to IHRC/2 and
                        // turn off ILRC oscillator at the same time
```

## 5.5 VDD/2 LCD Bias Voltage Generator

This function can be enabled by misc.4 and code option LCD2. There are three sets of pins can be selected as the LCD COM ports. By selecting PB0_PA035 for LCD2, PB0, PA0, PA3 and PA5 are defined to output VDD/2 voltage during input mode, and be used as COM function for LCD applications. By selecting PB1256 of LCD2, PB1, PB2, PB5 and PB6 are defined as COM ports. By selecting PB7_PC0~6 of LCD2, PB7 and PC0 to PC6 are defined as COM ports.

If user wants to output VDD, VDD/2, GND three levels voltage, enabling VDD/2 bias voltage (by set misc.4=1), then set to output-high for VDD, with input mode for VDD/2, and output-low for GND correspondingly, Fig.3 shows how to use this function.

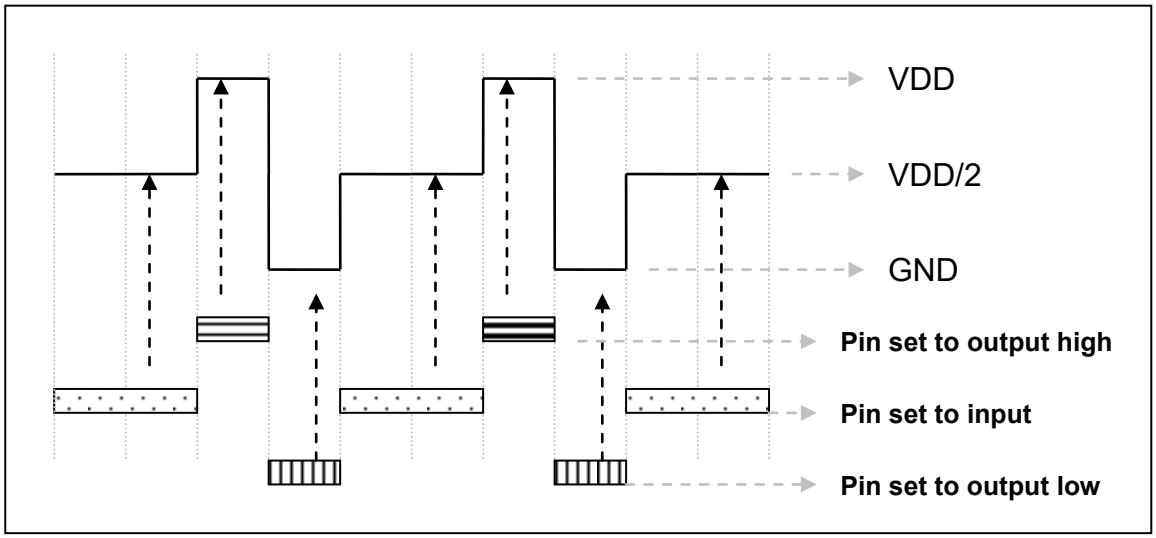



Fig. 3: Using VDD/2 LCD bias voltage generator

## 5.6 16-bit Timer(Timer16)

A 16-bit hardware timer (Timer16) is implemented in the PFB190, the clock sources of Timer16 may come from system clock (CLK), internal high RC oscillator (IHRC), internal low RC oscillator (ILRC) and PA0, a multiplex is used to select clock output for the clock source. Before sending clock to the counter16, a pre-scaling logic with divided-by-1, 4, 16, and 64 is used for wide range counting.

The 16-bit counter performs up-counting operation only, the counter initial values can be stored from memory by stt16 instruction and the counting values can be loaded to memory by ldt16 instruction. A selector is used to select the interrupt condition of Timer16, whenever overflow occurs, the Timer16 interrupt can be triggered. The hardware diagram of Timer16 is shown as Fig.4. The interrupt source of Timer16 comes from one of bit 8 to 15 of 16-bit counter, and the interrupt type can be rising edge trigger or falling edge trigger which is specified in the bit 4 of integs register (IO address 0x0C).

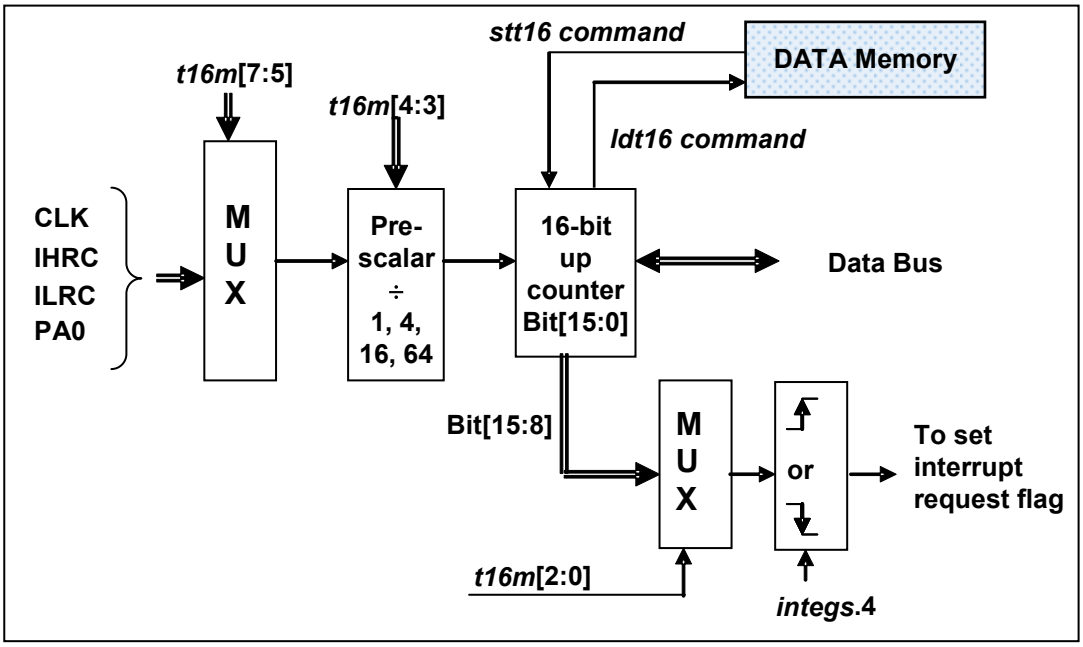



Fig.4: Hardware diagram of Timer16

When using the Timer16, the syntax for Timer16 has been defined in the .INC file. There are three parameters to define the Timer16; 1st parameter is used to define the clock source of Timer16, 2nd parameter is used to define the pre-scalar and the last one is to define the interrupt source. The detail description is shown as below:

```
T16M        IO_RW           0x06
        $ 7~5: STOP, SYSCLK, X, X , IHRC, 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 define the parameters of T16M based on system requirement, some examples are shown below and more examples please refer to “Help → Application Note → IC Introduction → Register Introduction → T16M” in IDE utility.

```
$ T16M SYSCLK, /64, BIT15;
    // choose (SYSCLK/64) as clock source, every 2^16 clock to set INTRQ.2=1
    // if using System Clock = IHRC / 2 = 8 MHz
    // SYSCLK/64 = 8 MHz/64 = 125KHz, about every 524 mS to generate INTRQ.2=1

$ T16M PA0_F, /1, BIT8;
    // choose PA0 as clock source, every 2^9 to generate INTRQ.2=1
    // receiving every 512 times PA0 to generate INTRQ.2=1

$ T16M STOP;
    // stop Timer16 counting
```

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

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

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

P is the selection of t16m [4:3]; (1, 4, 16, 64)

N is the nth bit selected to request interrupt service, for example: n=10 if bit 10 is selected.

## 5.7 8-bit Timer (Timer2/Timer3) with PWM generation

Two 8-bit hardware timers (Timer2 and Timer3) with PWM generation are implemented in the PFB190. The following descriptions thereinafter are for Timer2 only. It is because Timer3 have same structure with Timer2. Please refer to Fig.5 shown the hardware diagram of Timer2, the clock sources of Timer2 may come from system clock, internal high RC oscillator (IHRC), internal Low-Speed RC Oscillator (ILRC), PA0 Pin, PB0 Pin, and Comparator. Bit [7:4] of register tm2c are used to select the clock of Timer2. If IHRC is selected for Timer2 clock source, the clock sent to Timer2 will keep running when using ICE in halt state. The output of Timer2 can be sent to pin PB2, PA3 or PB4, depending on bit [3:2] of tm2c register. A clock pre-scaling module is provided with divided-by- 1, 4, 16, and 64 options, controlled by bit [6:5] of tm2s register; one scaling module with divided-by-1~32 is also provided and controlled by bit [4:0] of tm2s register. In conjunction of pre-scaling function and scaling function, the frequency of Timer2 clock (TM2_CLK) can be wide range and flexible.

The Timer2 counter performs 8-bit up-counting operation only; the counter values can be set or read back by tm2ct register. The 8-bit counter will be clear to zero automatically when its values reach for upper bound register, the upper bound register is used to define the period of timer or duty of PWM. 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 to 8-bit PWM resolution, Fig.6 shows the timing diagram of Timer2 for both period mode and PWM mode.

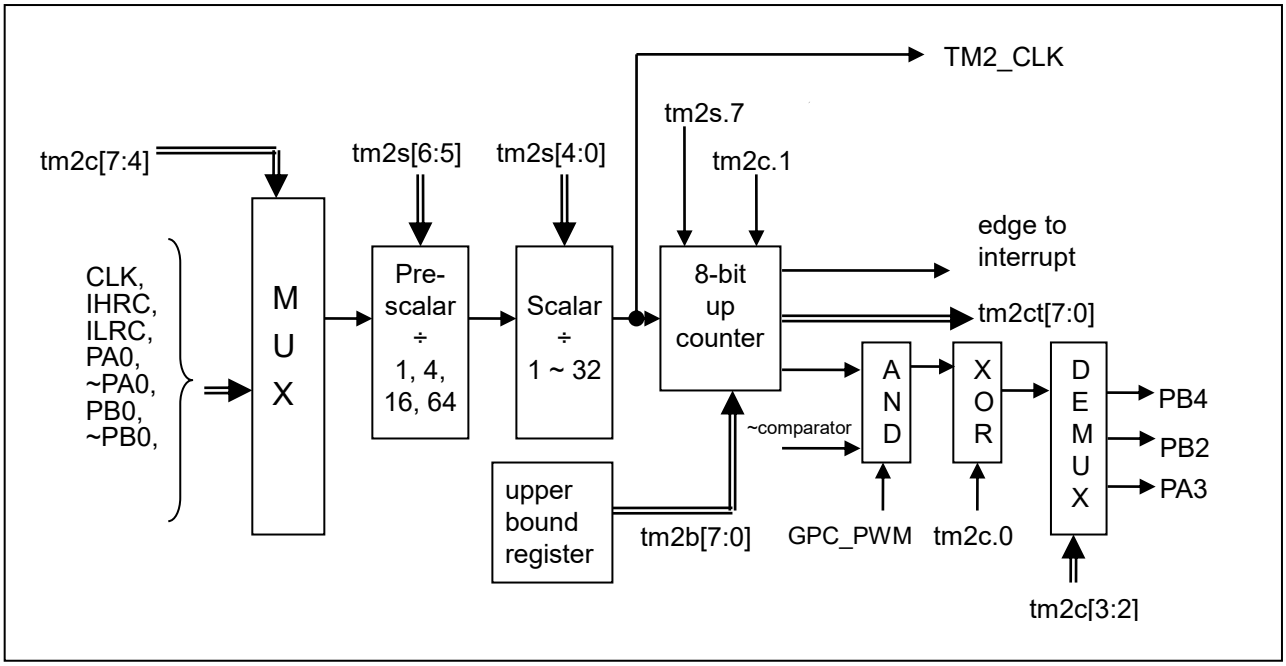



Fig.5: Timer2 hardware diagram

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

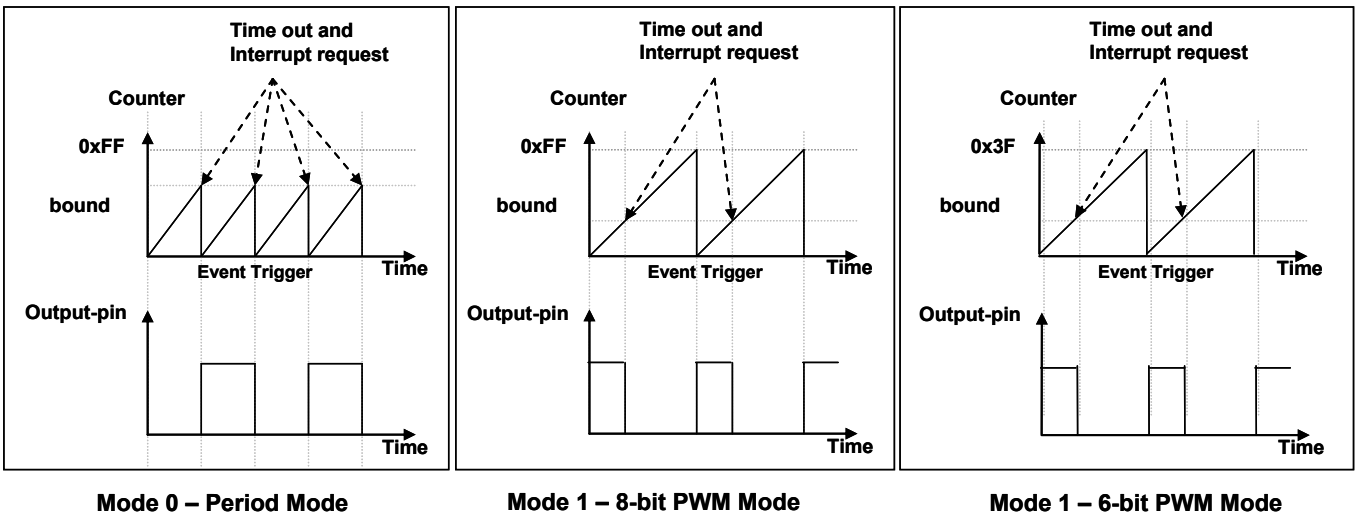



Fig.6: Timing diagram of Timer2 in period mode and PWM mode (tm2c.1=1)

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. 7.

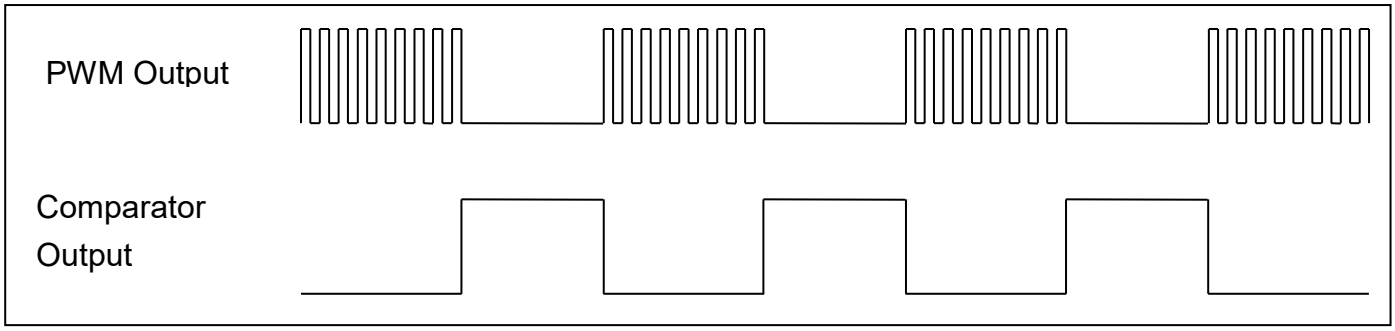



Fig.7：Comparator controls the output of PWM waveform

### 5.7.1 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 = 0b0000_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 = 0b0111_1111, K=127

tm2s[7:0] = 0b0111_11111, S1=64 , S2 = 31

➔ Frequency = 8MHz ÷ ( 2 × (127+1) × 64 × (31+1) ) =15.25Hz

Example 3:

tm2c = 0b0001_1000, Y=8MHz

tm2b = 0b0000_1111, K=15

tm2s = 0b0000_00000, S1=1, S2=0

➔ Frequency = 8MHz ÷ ( 2 × (15+1) × 1 × (0+1) ) = 250KHz

Example 4:

tm2c = 0b0001_1000, Y=8MHz

tm2b = 0b0000_0001, K=1

tm2s = 0b0000_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 from 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;
    }
}
```

### 5.7.2 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 = 0b0000_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 = 0b0111_1111, K=127
tm2s = 0b0111_11111, S1=64, S2=31
➔ frequency of output = 8MHz ÷ ( 256 × 64 × (31+1) ) = 15.25Hz
➔ duty of output = [(127+1) ÷ 256] × 100% = 50%

Example 3:

tm2c = 0b0001_1010, Y=8MHz
tm2b = 0b1111_1111, K=255
tm2s = 0b0000_00000, S1=1, S2=0
➔ PWM output keep high
➔ duty of output = [(255+1) ÷ 256] × 100% = 100%

Example 4:

tm2c = 0b0001_1010, Y=8MHz
tm2b = 0b0000_1001, K = 9
tm2s = 0b0000_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, VBAT =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;
    }
}
```

### 5.7.3 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 (1, 4, 16, 64)

tm2s[4:0] = S2 : scalar register in decimal (1 ~ 31)

Users can set Timer2 to be 7-bit PWM mode instead of 6-bit mode by using ***TMx_Bit*** code option. At that time, the calculation factors of the above equations become 128 instead of 64.

Example 1:

tm2c = 0b0001_1010, Y=8MHz

tm2b = 0b0001_1111, K=31

tm2s = 0b1000_00000, S1=1, S2=0

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

➔ duty = [(31+1) ÷ 64] × 100% = 50%

Example 2:

tm2c = 0b0001_1010, Y=8MHz

tm2b = 0b0001_1111, K=31

tm2s = 0b1111_11111, S1=64, S2=31

➔ frequency of output = 8MHz ÷ ( 64 × 64 × (31+1) ) = 61.03 Hz

➔ duty of output = [(31+1) ÷ 64] × 100% = 50%

Example 3:

tm2c = 0b0001_1010, Y=8MHz

tm2b = 0b0011_1111, K=63

tm2s = 0b1000_00000, S1=1, S2=0

➔ PWM output keep high

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

Example 4:

tm2c = 0b0001_1010, Y=8MHz

tm2b = 0b0000_0000, K=0

tm2s = 0b1000_00000, S1=1, S2=0

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

➔ duty = [(0+1) ÷ 64] × 100% =1.5%

## 5.8 11-bit PWM Generators

One set of triple 11-bit SuLED (Super LED) hardware PWM generator is implemented in the PFB190. It consists of three PWM generators (LPWMG0, LPWMG1 & LPWMG2). Their individual outputs are listed as below:

- LPWMG0 – PA0, PB4, PB5
- LPWMG1 – PB6, PB7
- LPWMG2 – PA3, PB2, PB3, PA5

### 5.8.1 PWM Waveform

A PWM output waveform (Fig.8) has a time-base ($T_{Period}$ = Time of Period) and a time with output high level (Duty Cycle). The frequency of the PWM output is the inverse of the period ($f_{PWM} = 1/T_{Period}$).

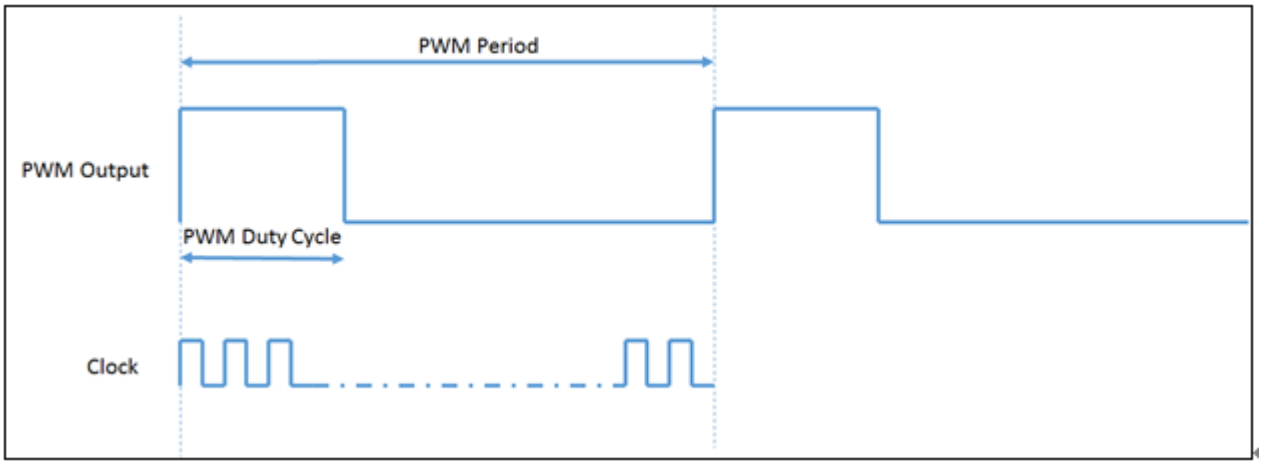



Fig.8: PWM Output Waveform

### 5.8.2 Hardware Diagram

Fig.9 shows the hardware diagram of the whole set of SuLED 11-bit hardware PWM generators. Those three PWM generators use a common Up-Counter and clock source selector to create the time base, and so the start points (the rising edge) of the PWM cycle are synchronized. The clock source can be IHRC or system clock. The PWM signal output pins that can be selected via ***lpwmgxc*** register selection. The period of PWM waveform is defined by the common PWM upper bound high and low registers, and the duty cycle of individual PWM waveform is defined by the individual set in the PWM duty high and low registers.

The additional OR and XOR logic of LPWMG0 channel is used to create the complementary switching waveforms with dead zone control. Selecting code option GPC_LPWM can also control the generated PWM waveform by the comparator result.

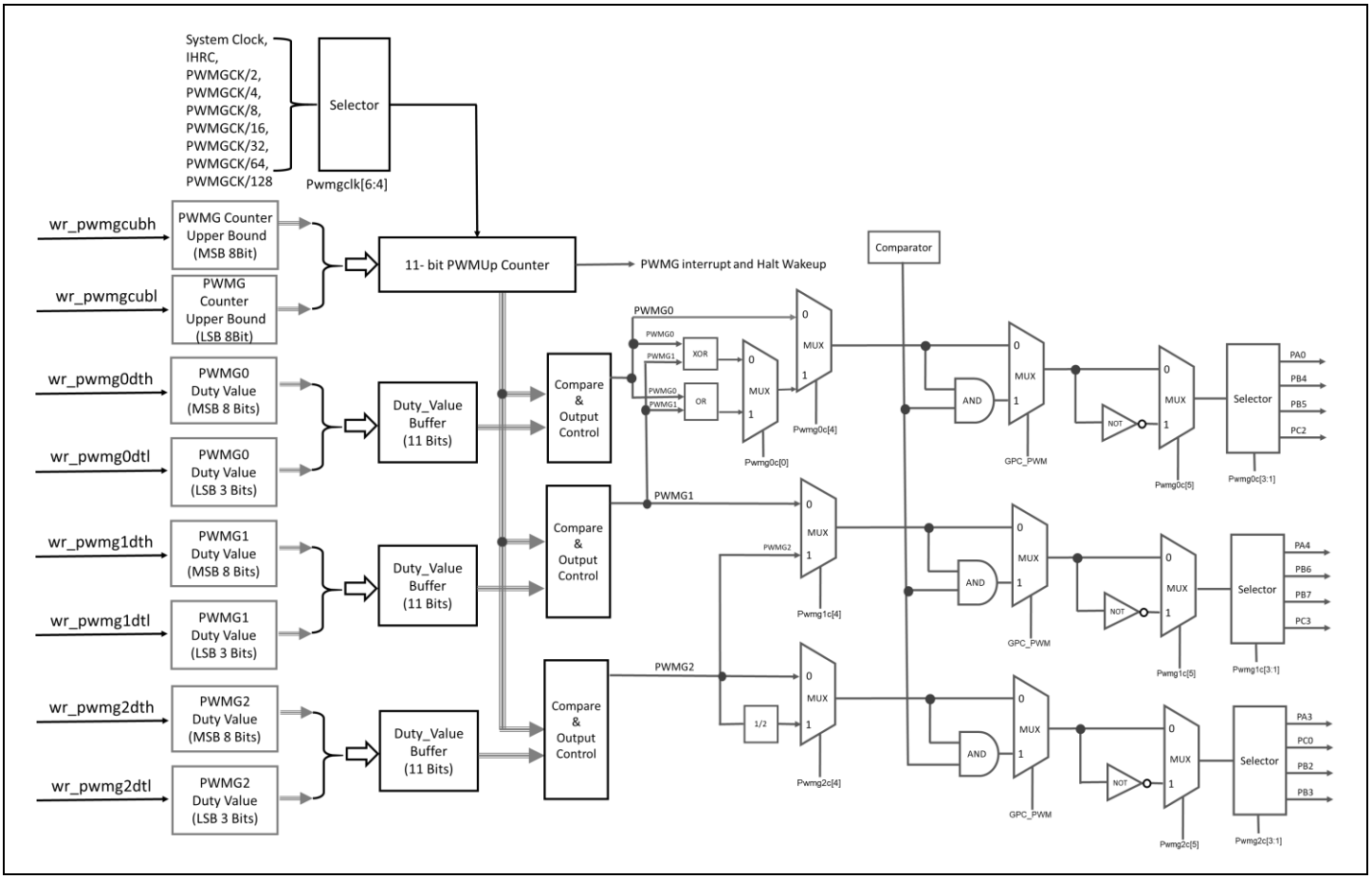



Fig.9: Hardware diagram of whole set of triple SuLED 11-bit PWM generators

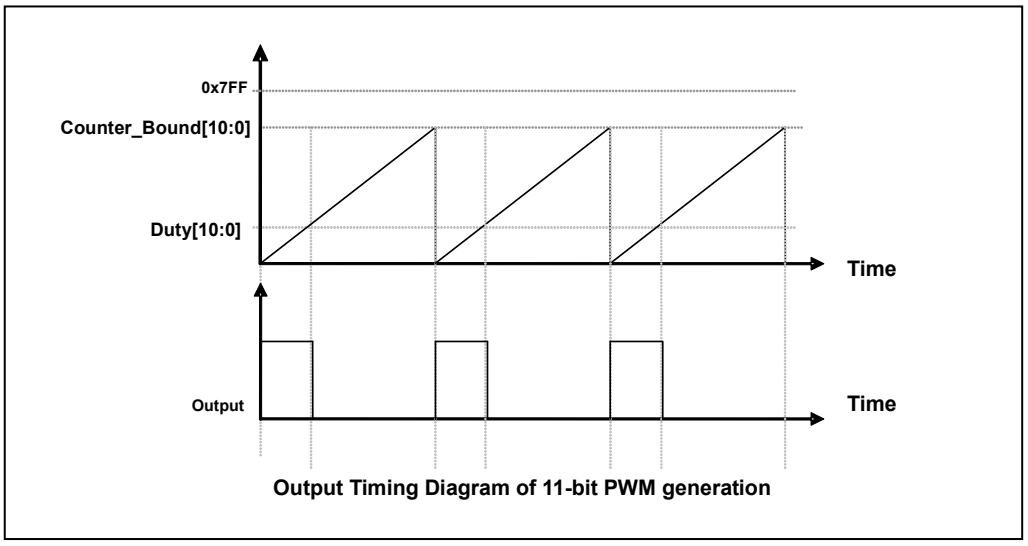



Fig.10: Output Timing Diagram of 11-bit PWM Generator

### 5.8.3 Equations for 11-bit PWM Generator

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

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

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

Where, **P = LPWMGCLK [6:4]; pre-scalar P=1,2,4,8,16,32,64,128**

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

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

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

### 5.8.4 PWM Waveforms with Complementary Dead Zones

Based on the specific 11-bit PWM architecture of PFB190, here we employ PWM2 output and PWM0 inverse output after PWM0 xor PWM1 to generate two PWM waveforms with complementary dead zones.

Example program is as follows:

```
#define   dead_zone       10      //   dead time = 10% * (1/PWM_Frequency) us
#define   PWM_Pulse       50      //   set 50% as PWM duty cycle

#define   PWM_Pulse_1     35      //   set 35% as PWM duty cycle
#define   PWM_Pulse_2     60      //   set 60% as PWM duty cycle
#define   switch_time     400*2   //   adjusting switch time
// Note: To avoid noise, switch_time must be a multiple of PWM period. In this example PWM period = 400us,
// so switch_time = 400*2 us.

void FPPA0 (void)
{
.ADJUST_IC      SYSCLK=IHRC/16, IHRC=16MHz, VBAT =5V;
//******* Generating fixed duty cycle waveform ************************
//------  Set the counter upper bound and duty cycle -----------------
LPWMG0DTL   =   0x00;
LPWMG0DTH   =   PWM_Pulse + dead_zone;

LPWMG1DTL   =   0x00;
LPWMG1DTH   =   dead_zone;      // After LPWMG0 xor LPWMG, PWM duty cycle=PWM_Pulse%

LPWMG2DTL   =   0x00;
LPWMG2DTH   =   PWM_Pulse + dead_zone*2;

LPWMGCUBL   =   0x00;
LPWMGCUBH   =   100;
```

```
//---- Configure clock and pre-scalar ------------------
$ LPWMGCLK    Enable, /1, sysclk;
//------- Output control -----------------------------
$ LPWMG0C     Enable,Inverse,LPWM_Gen,PA0,gen_xor;   // After LPWMG0 xor LPWMG,
                                                     //   output the inversed waveform through PA0
$ LPWMG1C     Enable, LPWMG1,disable;                //   disable LPWMG1 output
$ LPWMG2C     Enable, PA3;                           //   output LPWMG2 waveform through PA3

while(1)
{
    //******** Switching duty cycle *********************************
    // To avoid the possible instant disappearance of dead zone, user should comply with the following
    // instruction sequence.
    // When increase the duty cycle:   50%/60%   →   35%
    LPWMG0DTL       =       0x00;
    LPWMG0DTH       =       PWM_Pulse_1 + dead_zone;
    LPWMG2DTL       =       0x00;
    LPWMG2DTH       =       PWM_Pulse_1 + dead_zone*2;
    .delay      switch_time

    // When decrease the duty cycle:   35% → 60%
    LPWMG2DTL       =       0x00;
    LPWMG2DTH       =       PWM_Pulse_2 + dead_zone*2;
    LPWMG0DTL       =       0x00;
    LPWMG0DTH       =       PWM_Pulse_2 + dead_zone;
    .delay      switch_time
}
}
```

The following figures show the waveforms at different condition.

1. The PWM waveform in a fixed-duty cycle:

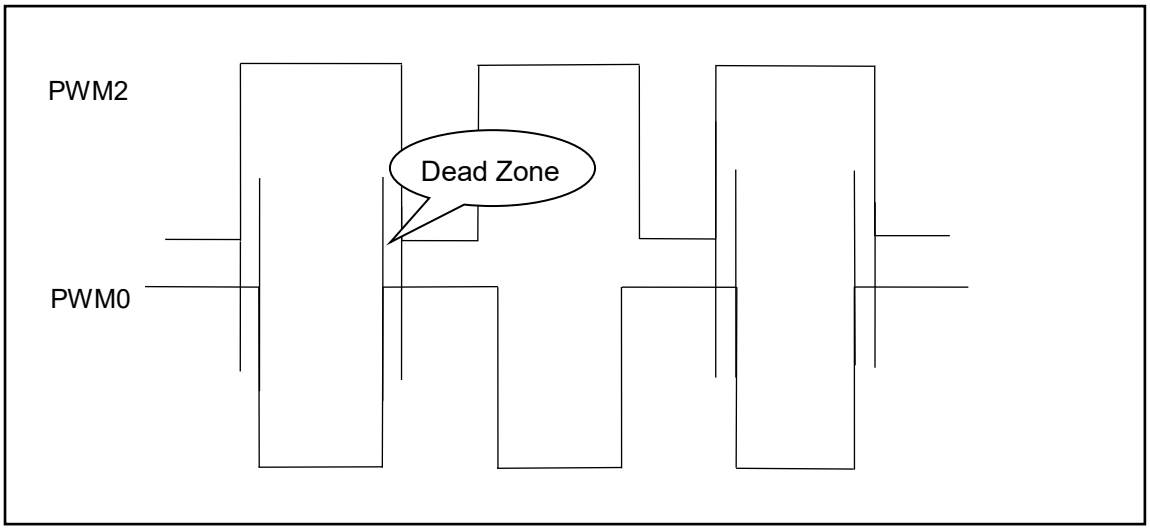



Fig.11: Complementary PWM waveform with dead zones

2. PWM waveform when switching two duty cycles:

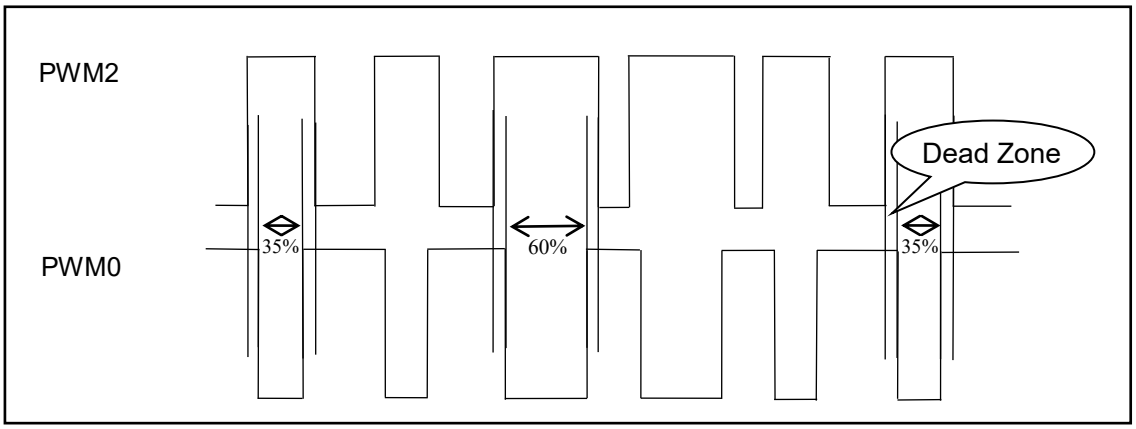



Fig.12: Complementary PWM waveform with dead zones

User can find that above example only provides dead zone where PWM are both in high. If need dead zone where PWM are both in low, you can realize it by resetting each control register's Inverse like:

***$ LPWMG0C  Enable,PWM_Gen,PA0,gen_xor;***
***$ LPWMG2C Enable, Inverse, PA3;***

## 5.9 WatchDog Timer

The watchdog timer (WDT) is a counter with clock coming from ILRC. WDT can be cleared by power-on-reset or by command ***wdreset*** at any time. There are four different timeout periods of watchdog timer to be chosen by setting the ***misc*** register, it is:

- ◆ 8k ILRC clocks period if register misc[1:0]=00 (default)
- ◆ 16k ILRC clocks period if register misc[1:0]=01
- ◆ 64k ILRC clocks period if register misc[1:0]=10
- ◆ 256k ILRC clocks period if register misc[1:0]=11

The frequency of ILRC may drift a lot due to the variation of manufacture, supply voltage and temperature; user should reserve guard band for save operation. Besides, the watchdog period will also be shorter than expected after Reset or Wakeup events. It is suggested to clear WDT by wdreset command after these events to ensure enough clock periods before WDT timeout.

When WDT is timeout, PFB190 will be reset to restart the program execution. The relative timing diagram of watchdog timer is shown as Fig.13.

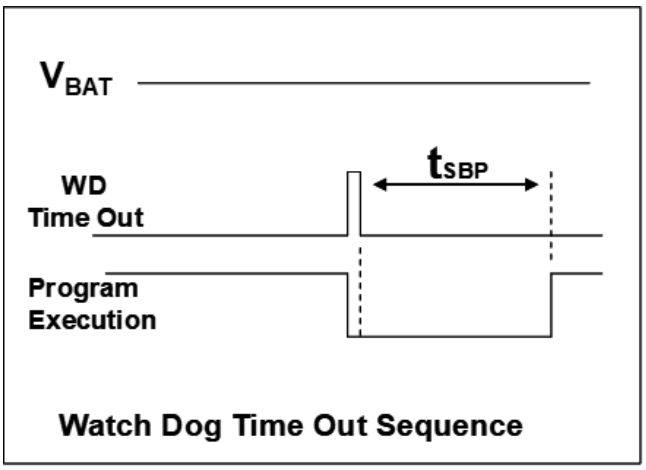



Fig.13: Sequence of Watch Dog Time Out

## 5.10 Interrupt

There are 7 interrupt lines for PFB190:

- External interrupt PA0/PA7/PB5
- External interrupt PB0/PB6/PA3
- ADC interrupt
- Timer16 interrupt
- LPWMG interrupt
- Timer2 interrupt
- Timer3 interrupt

Every interrupt request line has its own corresponding interrupt control bit to enable or disable it; the hardware diagram of interrupt function is shown as Fig.14. 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.

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

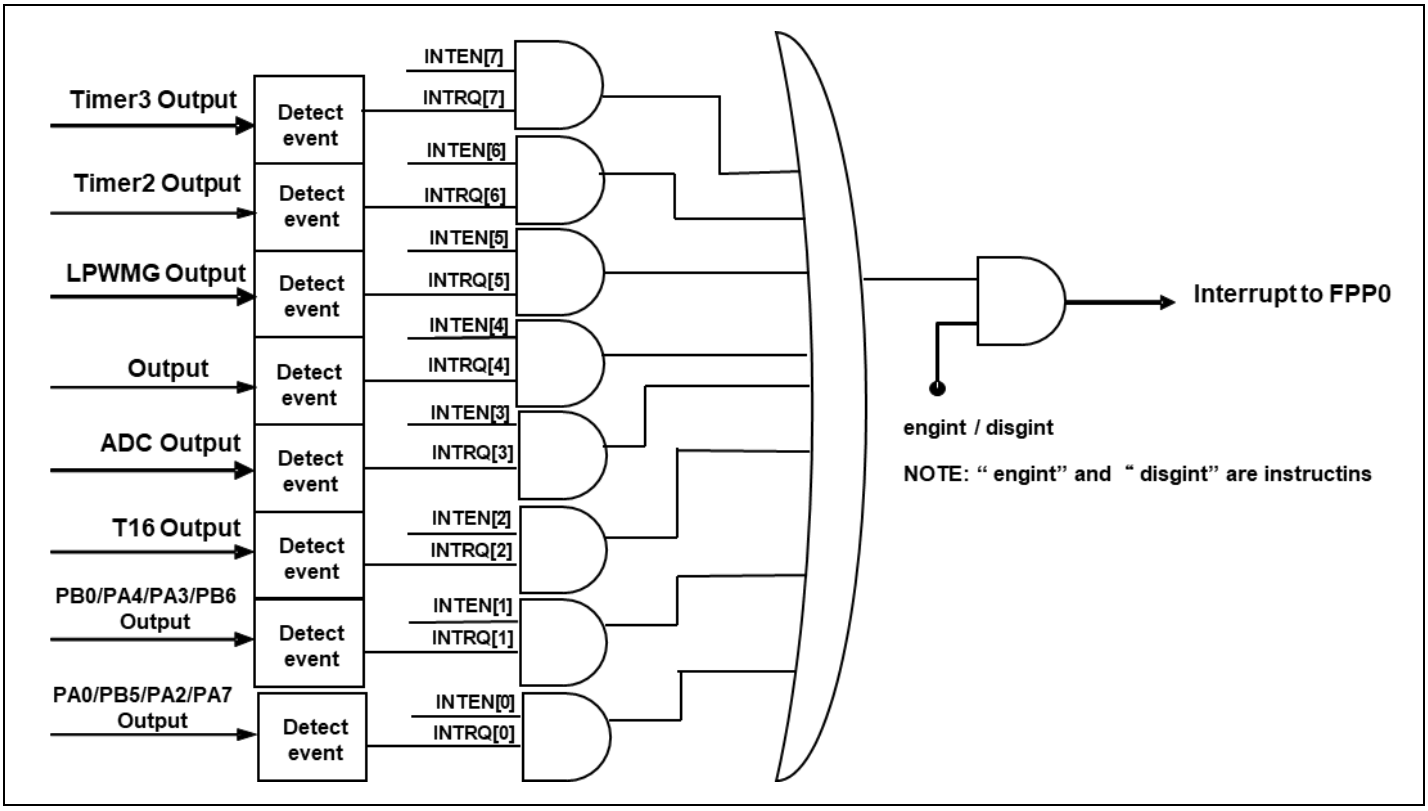



Fig.14: Hardware diagram of interrupt controller

Once the interrupt occurs, its operation will be:

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

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

**Note:** Even if INTEN=0, INTRQ will be still triggered by the interrupt source.
After finishing the interrupt service routine and issuing the ***reti*** instruction to return back, its operation will be:

- The program counter will be restored automatically from the stack memory specified by register ***sp***.
- New ***sp*** will be updated to ***sp-2***.
- Global interrupt will be enabled automatically.

The next instruction will be the original one before 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
}
```

## 5.11 Power-Save and Power-Down

There are three operational modes defined by hardware: ON mode, Power-Save mode and Power-Down modes. 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 occasionally and Power-Down mode is used in the system which needs power down deeply with seldom wake-up. Table 4 shows the differences in oscillator modules between Power-Save mode ("***stopexe***") and Power-Down mode ("***stopsys***").

| Differences in oscillator modules between STOPSYS and STOPEXE | | | |
|---|---|---|---|
| | IHRC | ILRC | NILRC |
| STOPSYS | Stop | Stop | No Change |
| STOPEXE | No Change | No Change | No Change |

Table 4: Differences in oscillator modules between STOPSYS and STOPEXE

### 5.11.1 Power-Save mode ("*stopexe*")

Using "***stopexe***" instruction to enter the Power-Save mode, only system clock is disabled, remaining all the oscillator modules active. For CPU, it stops executing; however, for Timer16, counter keep counting if its clock source is not the system clock. The wake-up sources for "stopexe" can be IO-toggle or Timer16 counts to set values when the clock source of Timer16 is IHRC or ILRC modules, NILRC wake-up of tm2c/tm3c Wake-up from input pins can be considered as a continuation of normal execution, the detail information for Power-Save mode shows below:

- IHRC oscillator modules: No change, keep active if it was enabled.
- ILRC oscillator modules: must remain enabled, need to start with ILRC when be wakening up.
- System clock: Disable, therefore, CPU stops execution.
- MTP memory is turned off.
- 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, LPWMG0, LPWMG1, LPWMG2.)
- Wake-up sources:
  a. IO toggle wake-up: IO toggling in digital input mode (*PAC* bit is 1 and *PADIER* 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. TM2C wake up with NILRC clock source:

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.

### 5.11.2 Power-Down mode ("stopsys")

Power-Down mode is the state of deeply power-saving with turning off all the oscillator modules. By using the "*stopsys*" instruction, this chip will be put on Power-Down mode directly. The following shows the internal status of PFB190 detail when "***stopsys***" command is issued:

- IHRC and ILRC modules are turned off.
- NILRC is turned on.
- MTP memory is turned off.
- The contents of SRAM and registers remain unchanged.
- Wake-up sources: IO toggle in digital mode. (PxDIER bit is 1)

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 reference sample program for power down is shown as below:

```
CLKMD   =   0xF4;       //  Change clock from IHRC to ILRC
CLKMD.4 =   0;          //  disable IHRC
...
while (1)
{
            STOPSYS;    //  enter power-down
            if (...) break; //  if wakeup 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
```

### 5.11.3 Wake-up

After entering the Power-Down or Power-Save modes, the PFB190 can be resumed to normal operation by toggling IO pins or NILRC wake-up with Timer2 and Timer3, Timer interrupt is 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 Interrupt | NILRC wake-up with Timer2 and Timer3 |
| STOPSYS | Yes | No | Yes |
| STOPEXE | Yes | Yes | Yes |

Table 5: Differences in wake-up sources between Power-Save mode and Power-Down mode

When using the IO pins to wake-up the PFB190, registers ***padier*** 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<br>or<br>STOPSYS suspend | Fast wake-up | 45 * $T_{ILRC}$,<br>Where $T_{ILRC}$ is the time period of ILRC |
| STOPEXE suspend<br>or<br>STOPSYS suspend | Normal wake-up | 3000 * $T_{ILRC}$,<br>Where $T_{ILRC}$ is the clock period of ILRC |

Table 6: Differences wake-up Speed

Please notice that when Fast boot-up is selected, no matter which wake-up mode is selected in ***misc***.5, the wake-up mode will be forced to be FAST. If Normal boot-up is selected, the wake-up mode is determined by ***misc***.5.

## 5.12 IO Pins

All the pins can be independently set into two states output or input by configuring the data registers (***pa, pb, pc***), control registers (***pac, pbc, pcc***) and pull-high/pull-low resistor (***paph/papl, pbph/pbpl, pcph/pcpl***). All these pins have Schmitt-trigger input buffer and output driver with CMOS level. When it is set to output low, the pull- high resistor is turned off automatically. If user wants to read the pin state, please notice that it should be set to input mode before reading the data port; if user reads the data port when it is set to output mode, the reading data comes from data register, NOT from IO pad. As an example, Table 7 shows the configuration table of bit 0 of port A. The hardware diagram of IO buffer is also shown as Fig.15.

| ***pa.0*** | ***pac.0*** | ***papl.0*** | ***paph.0*** | Description |
|---|---|---|---|---|
| X | 0 | 0 | 0 | Input without pull- high/pull-low resistor |
| X | 0 | 0 | 1 | Input with pull- high resistor, without pull-low resistor |
| X | 0 | 1 | 0 | Input with pull-low resistor, without pull- high resistor |
| 0 | 1 | 0 | X | Output low without pull-low resistor |
| 0 | 1 | 1 | X | Output low with pull-low resistor |
| 1 | 1 | X | 0 | Output high without pull- high resistor |
| 1 | 1 | X | 1 | Output high with pull- high resistor |

Table 7: PA0 Configuration Table

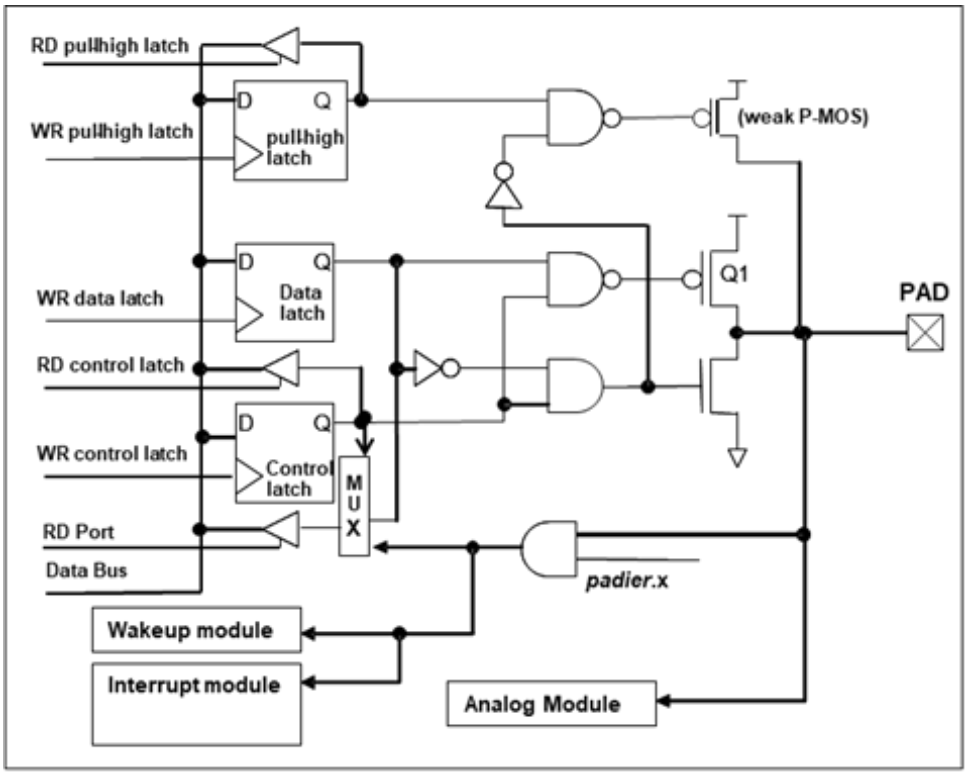



Fig.15: Hardware diagram of IO buffer

All the IO pins has the same structure. The corresponding bits in registers ***padier, pbdier and pcdier*** should be set to low to prevent leakage current for those pins are selected to be analog function. When PFB190 is put in power-down or power-save mode, every pin can be used to wake-up system by toggling its state. Therefore, those pins needed to wake-up system must be set to input mode and set the corresponding bits of registers ***padier, pbdier and pcdier*** to high**.** The same reason, ***padier***.0 should be set high when PA0 is used as external interrupt pin.

## 5.13 Reset and LVR

### 5.13.1 Reset

There are many causes to reset the PFB190, once reset is asserted, most of all the registers in PFB190 will be set to default values, system should be restarted once abnormal cases happen, or by jumping program counter to address 0x00.

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

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

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

### 5.13.2 LVR reset

By code option, there are many different levels of LVR for reset. Usually, user selects LVR reset level to be in conjunction with operating frequency and supply voltage.

## 5.14 Charger

The PFB190 has a built-in hardware charger. This charger is fully constant current/constant voltage linear charging for single cell Li-ion battery charge management. Due to the internal MOSFET structure, there is no need for external sense resistors or blocking diodes, and the maximum charging current is up to 500 mA. The charger works automatically when power is supplied, and does not require the MCU program to make the startup settings, which means that it can manage the battery charging in the factory default state.

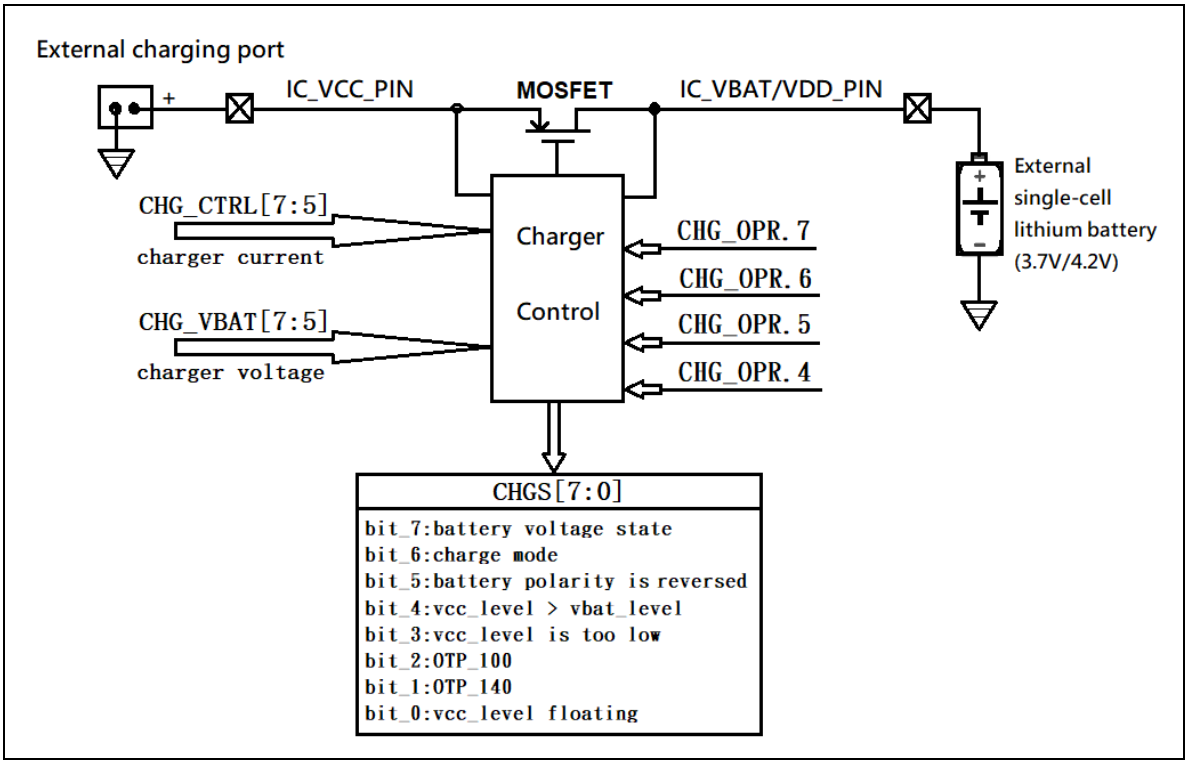
Users can set or adjust the charging current through the three Bits of CHG_CTRL[7:5]. There are 4 groups of charging current can be set, maximum 500mA, minimum 125mA.

MCU program can read register CHGS[6] to judge the charger working status. The MCU program can read register CHGS[4] to determine if the charging Vcc voltage is greater than Vbat. Read register CHGS[3] to determine if the charging Vcc voltage is too low to make the charger shut down automatically. When CHGS[4:3] = 0b00 it can be judged that the charging interface is connected and the charging voltage is normal.

The charger also integrates a charging over-temperature protection circuit, and the charging over-temperature protection has two options: 100°C and 140°C. The charging over-temperature protection can be controlled by writing CHGS[2:1]. Charging over-temperature protection can be controlled by writing CHG_OPR[6:5], and reading CHGS [2:1] can determine whether charging over-temperature is triggered or not.

The charging voltage and current of the PFB190 charger have been calibrated at the factory and the calibration values are written into the system parameter protection area. When the MCU is successfully powered on and executes the .Adjust_IC macro instruction, the voltage/current factory calibration value of the charger will be filled into the CHG_TRIM / CHG_CUR registers, and then the charging voltage and current of the charger will be measured as the factory calibration value, the charging voltage and current of the charger will be uncalibrated when the MTP of the MCU is blank and no program or the power-on failure. CHG_Trim / CHG_CUR registers are not recommended to be rewritten by users. Adjust_IC Disable or write assembly code can be discussed with FAE. If you want to fine tune the charging voltage and current of the charger, you can use ReLoad_Vbat BGTRIM and ReLoad_ChargerCURTRIM macros.

The PFB190 detects the battery voltage Vbat using ADC. Set BG2V4 or BG2V68 as the reference voltage of ADC. The ADC channel selects channel F and 3/8 VDD as the voltage to be tested, and the voltage value of Vbat can be obtained.




The default charging voltage of the charger is 4.2V at startup, which can be adjusted by the program through the CHG_VBAT[7:5] register. The charging current can be set by register CHG_CTRL[7:5]. When the final floating voltage is reached, the charging current will automatically drop to 1/10 of the programmed value set in registers CHG_CTRL[7:5], and the charger will automatically stop the current charging cycle.

After removing the input power (wall outlet or USB power), the PFB190 charger IP automatically enters a low current state that reduces the battery leakage current to less than 2uA.
The charger is also designed with other features such as under voltage lockout, automatic charging and trickle charge mode.

### 5.14.1 Thermal Limiting

The charger has thermal feedback to regulate the charging current to limit it to high power operation or high ambient temperatures. When the internal temperature of the PFB190 IC rises above a preset value of approximately 90°C, the internal thermal feedback loop will reduce the programmed charge current. This feature protects the PFB190 from excessive temperatures and allows the user to push the limits of the power handling capability of a given board without damaging the PFB190. The charging current can be set based on typical (non-worst case) ambient temperatures, ensuring that the charger automatically reduces the current in worst case scenarios.

### 5.14.2 Power Dissipation

The conditions that cause the PFB190 to reduce charge current through thermal feedback can be approximated by considering the power dissipated in the IC. Nearly all of this power dissipation is generated by the internal MOSFE This is calculated to be approximately:

**$P_D = (V_{VCC} – V_{VBAT}) • I_{VBAT}$**

Were $P_D$ is the power dissipated, $V_{VCC}$ is the input supply voltage, $V_{VBAT}$ is the battery voltage and $I_{VBAT}$ is the charge current. The approximate ambient temperature at which the thermal feedback begins to protect the IC is:

**$T_A = 90°C – P_D\theta_{JA}$**
**$T_A = 90°C – (V_{VCC} – V_{VBAT}) • I_{VBAT} • \theta_{JA}$**

Example: The PFB190 operating from a 5V USB supply is programmed to supply 400mA full-scale current to a discharged Li-Ion battery with a voltage of 3.75V. Assuming $\theta_{JA}$ is 150°C/W (see Board Layout Considerations), the ambient temperature at which the PFB190 will begin to reduce the charge current is approximately:

**$T_A = 90°C – (5V – 3.75V) • (400mA) • 150°C/W$**
**$T_A = 90°C – 0.5W • 150°C/W = 90°C – 75°C$**
**$T_A = 15°C$**

The PFB190 can be used above 15°C ambient, but the charge current will be reduced from 400mA. The approximate current at a given ambient temperature can be approximated by:

$$I_{VBAT} = \frac{90°C - T_A}{(V_{VCC} - V_{VBAT}) \cdot \theta_{JA}}$$

Using the previous example with an ambient temperature of 60°C, the charge current will be reduced to approximately:

$$I_{VBAT} = \frac{90°C - 60°C}{(5V - 3.75V) \cdot 150\,°C/W} = \frac{30°C}{187.5\,°C/A}$$

$$I_{VBAT} = 160mA$$

It is important to remember that PFB190 applications do not need to be designed for worst-case thermal conditions since the IC will automatically reduce power dissipation when the junction temperature reaches approximately 90°C.

### 5.14.3 Thermal Considerations

Because of the small size package, it is very important to use a good thermal PC board layout to maximize the available charge current. The thermal path for the heat generated by the IC is from the die to the copper lead frame, through the package leads, (especially the ground lead) to the PC board copper. The PC board copper is the heat sink. The footprint copper pads should be as wide as possible and expand out to larger copper areas to spread and dissipate the heat to the surrounding ambient. The feed through vias to inner or backside copper layers are also useful in improving the overall thermal performance of the charger. Other heat sources on the board, not related to the charger, must also be considered when designing a PC board layout because they will affect overall temperature rise and the maximum charge current.

### 5.14.4 EPAD

The PCB layout alignment should ensure that the package pins GND and EPAD have enough thermal paths to make the thermal paths effective.

## 5.15 Analog-to-Digital Conversion (ADC) module

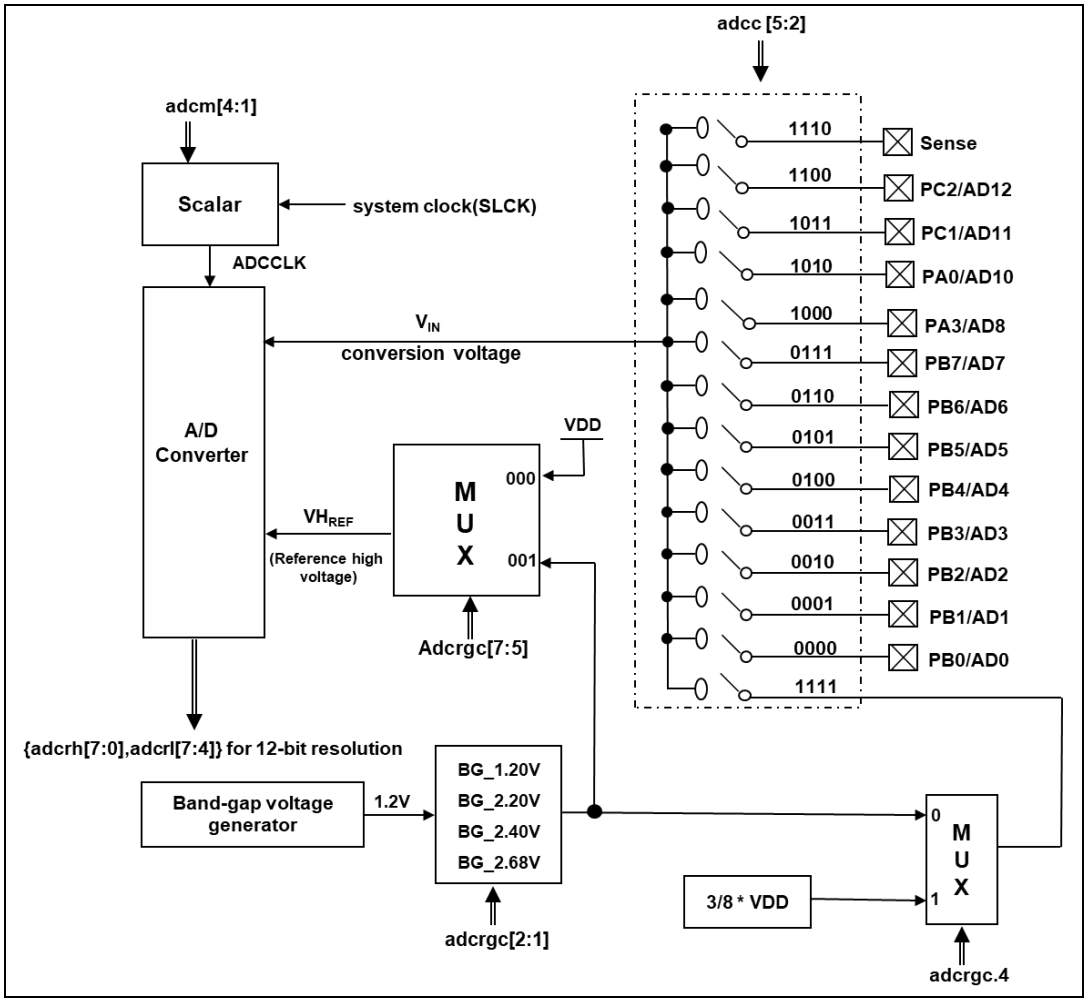



Fig.16: ADC Block Diagram

There are seven registers when using the ADC module, which are:

- ADC Control Register (***adcc***)
- ADC Regulator Control Register (***adcrgc***)
- ADC Mode Register (***adcm***)
- ADC Result High/Low Register (***adcrh***, ***adcrl***)
- Port A/B/C Digital Input Enable Register (***padier***, ***pbdier***, ***pcdier***)

The following steps are required to do the ADC conversion procedure:

(1) Configure the voltage reference high by ***adcrgc*** register

(2) Configure the AD conversion clock by ***adcm*** register

(3) Configure the pin as analog input by ***padier***, ***pbdier*** register

(4) Select the ADC input channel by ***adcc*** register

(5) Enable the ADC module by ***adcc*** register

(6) Delay a certain amount of time after enabling the ADC module

**Condition1:** Using bandgap 1.2V or 2.2V/2.4V/2.68V related circuit, either it is used as an internal reference high voltage or an AD Input channel, it must delay more than 1ms. Or it must delay 200 AD clocks when the time of 200 AD clocks is larger than 1ms. When internal BG1.2V or 2.2V/2.4V/2.68V is enabled as reference high voltage, IHRC must be opened.

**Condition 2:** Without using any bandgap 1.2V or 2.2V/2.4V/2.68V related circuit, it needs to delay 200 AD clocks only.

**Note:** The 200 AD clocks in the above two conditions, which refer to the ADC conversion clock rather than the system clock (SYSCLK) after configured by the ADCM register.

(7) Execute the AD conversion and check if ADC data is ready

Set '1' to ***addc.***6 to start the conversion and check whether ***addc***.6 is '1'

(8) Read the ADC result registers:

First read the ***adcrh*** register and then read the ***adcrl*** register.

If user power down the ADC and enable the ADC again, or switch ADC reference voltage and input channel, be sure to go to step 6 to confirm the ADC becomes ready before the conversion.

### 5.15.1 The input requirement for AD conversion

For the AD conversion to meet its specified accuracy, the charge holding capacitor ($C_{HOLD}$) must be allowed to fully charge to the voltage reference high level and discharge to the voltage reference low level. The analog input model is shown as Fig.17, the signal driving source impedance (Rs) and the internal sampling switch impedance (Rss) will affect the required time to charge the capacitor $C_{HOLD}$ directly. The internal sampling switch impedance may vary with ADC supply voltage; the signal driving source impedance will affect accuracy of analog input signal. User must ensure the measured signal is stable before sampling; therefore, the maximum signal driving source impedance is highly dependent on the frequency of signal to be measured. The recommended maximum impedance for analog driving source is about 10KΩ under 500KHz input frequency.

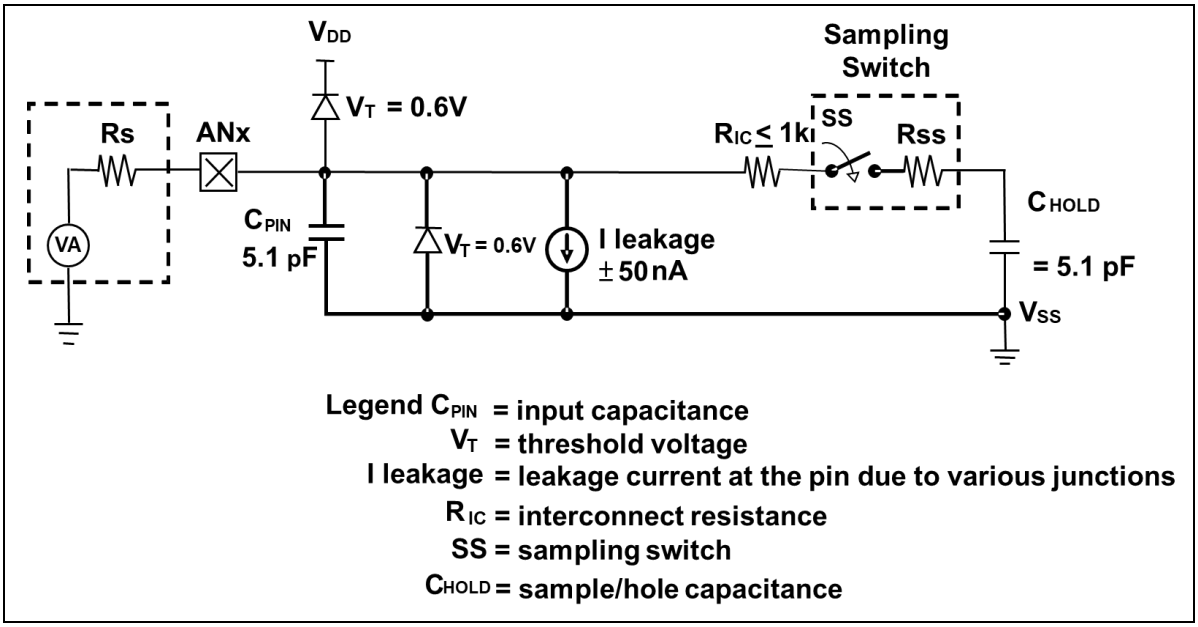



Fig.17: Analog Input Model

Before starting the AD conversion, the minimum signal acquisition time should be met for the selected analog input signal, the selection of ADCLK must be met the minimum signal acquisition time.

### 5.15.2 Select the reference high voltage

The ADC reference voltage channel can be selected via bit[7:5] of register ***adcrgc*** and its option can be $V_{DD}$, or bandgap reference voltage. The bandgap reference voltage is determined by the ADCRG bit [2:1] setting.

### 5.15.3 ADC clock selection

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

### 5.15.4 Configure the analog pins

There are 15 analog signals can be selected for AD conversion, 13 analog input signals come from external pins, one is from Sense and one is from internal bandgap reference voltage or 3/8*$V_{DD}$. There are 4 voltage levels selectable for the internal bandgap reference, they are 1.2V, 2.2V, 2.4V and 2.68V. For external pins. To avoid leakage current at the digital circuit, those pins defined for analog input should disable the digital input function (set the corresponding bit of ***padier / pbdier / pcder*** register to be 0).

The measurement signals of ADC belong to small signal; it should avoid the measured signal to be interfered during the measurement period, the selected pin should (1) be set to input mode (2) turn off weak pull-high resistor (3) set the corresponding pin to analog input by port A/B/C digital input disable register (***padier / pbdier / pcdier***).

### 5.15.5 Using the ADC

The following example shows how to use ADC with PB0~PB3.
First, defining the selected pins:

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

Next, setting ***ADCC*** register, example as below:

```
$ ADCC Enable, PB3;           // set PB3 as ADC input
$ ADCC Enable, PB2;           // set PB2 as ADC input
$ ADCC Enable, PB0;           // set PB0 as ADC input
// Note: Only one input channel can be selected for each AD conversion
```

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

```
$ ADCM 12BIT, /16;            // recommend /16 @System Clock=8MHz
$ ADCM 12BIT, /8;             // recommend /8 @System Clock=4MHz
$ ADCRGC VDD;
```

Next, delay 400us(ADCLK=500KHz, 200*ADCLK=400us), example as below:

```
.Delay 8*400;                 // System Clock=8MHz
.Delay 4*400;                 // System Clock=4MHz
```

Note: If using internal reference high voltage such as bandgap 1.2V, the delay time must be more than 1ms.

```
$ ASDCRGC BG, BG_2V;;         // AD reference voltage is 2.0V
.Delay 4*1010;                // if the system clock=4MHz
                              // the delay time must be more than 1ms
```

Please Note: If using bandgap 1.2V as ADC input channel, the delay time must be more than 1ms.

```
$ ADCC ADC
$ ADCRGC    VDD   ADC_BG   BG_2V    // reference voltage is VDD
                                    // input channel is BG_2V
.Delay 4*1010;                      // if the system clock=4MHz
                                    // the delay time must be more than 1ms
```

Then, start the ADC conversion:

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

Finally, it can read ADC result when AD_DONE is high:

```
WORD        Data;                   // two bytes result: ADCRH and ADCRL
Data$1  =   ADCRH
Data$0  =   ADCRL;
Data    =   Data >> 4;
```

The ADC can be disabled by using the following method:

```
$ ADCC  Disable;
```

or

```
ADCC    =   0;
```

### 5.15.6 How to calculate Vbat voltage

For PFB190, VDD and bandgap voltage can be selected as VREF for ADC. When the VDD voltage is supplied by the battery, if you want to measure the battery voltage, you can set the ADC reference channel to BG2V4 or BG2V68, select channel F as the ADC input channel, and set it to 3/8VDD.

## 5.16 Sense Function

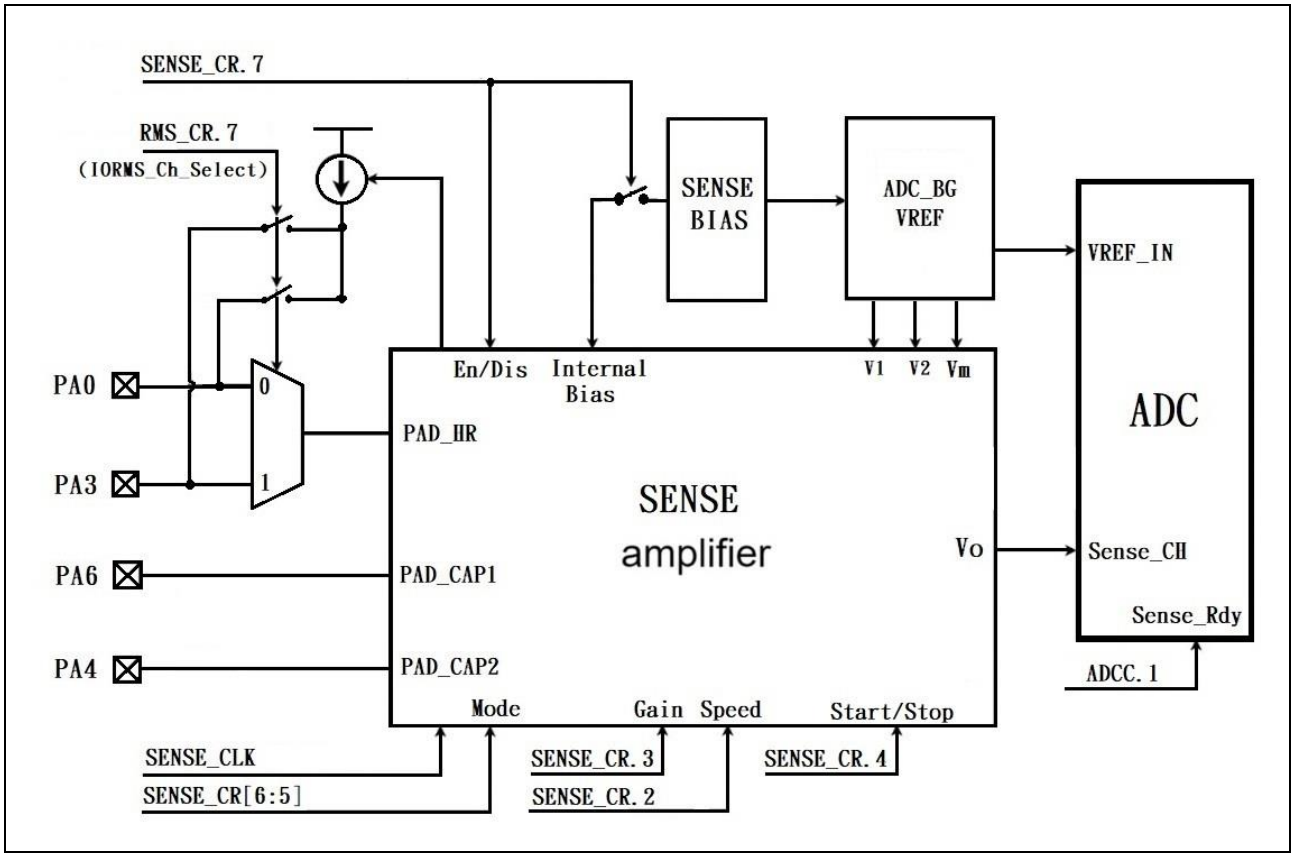



Fig.18 : Sense Block

The Sense in PFB190, it has two main functions: Microphone Capacitor Sense (MCS) and Heating Resistor Sense (HRS). The function of MCS is to obtain external passive capacitance, which can be used to measure capacitive MIC. The function of HRS is to obtain external resistance, which can be used for resistance detection of heating wire. The sense module converts the external capacitance or resistance into voltage, which is then converted into digital through the internal ADC module.

When using the Sense module to detect capacitance and resistance, there are several registers that need to be configured. They are:

- ADC Control Register (adcc)
- ADC Adjustment Control Register (***adcrgc***)
- ADC Mode Register (***adcm***)
- ADC result register (***adcrh, adcrl***)
- Port A digital input enable register (padier)
- SENSE Control Register (***sense_cr***)
- SENSE bias voltage register (***sense_bias***)
- HRS Control Register (***rms_cr***)

The following are the steps of the SENSE with ADC conversion process:

(1) Configure the reference high voltage through register adcrgc.
(2) Configure the AD conversion clock signal through the adcm register.
(3) Configure the analog input pins through the padier register.
(4) Set Sense enable and mode through the sense_cr register.
(5) Enable the ADC module, select the Sense channel and sense mode through the adcc register.
(6) Delay for at least 5u Second.
(7) Set ADCC.1 to 1 (ADC switches to Sense mode).
(8) Enable the Sense Start through the sense_cr register.
(9) Perform AD conversion and check whether the ADC conversion data has been completed.
Set addc.6 to 1 to start AD conversion and check if addc.6 is '1'.
(10) Read the conversion result from the ADC register.
(11) Set Sense to disable through the sense_cr register.

### 5.16.1 Capacitance Sense：(Microphone Capacitor Sense, MCS)

The PFB190 capacitance sense (Microphone Capacitor Sense, MCS) supports general passive capacitive MIC and passive MEMS MIC. In MCS detection, it is suitable for load capacitance of 10pF ~ 24pF. The two pins of the load capacitor are connected to the PA4 and PA6 pins respectively. For MEMS MIC, MEMS with capacitance of 1.4pF is verified. The two pins of MEMS MIC are connected to the PA4 and PA6 pins respectively.

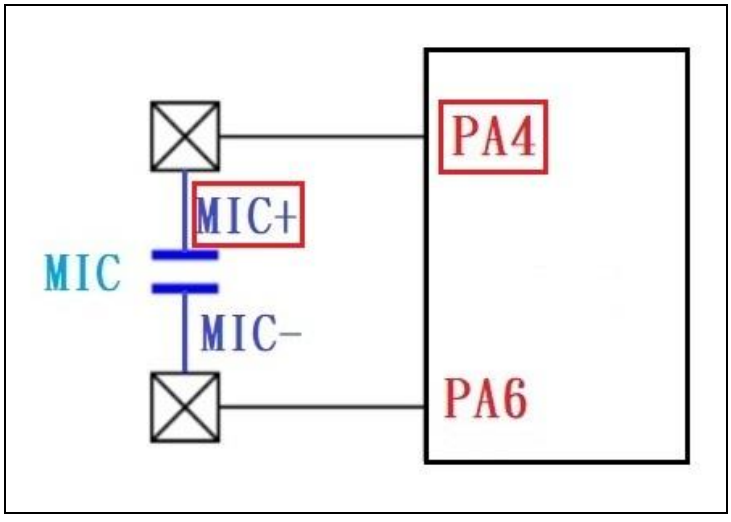



Fig.19: Connection diagram of mic in MCS

### 5.16.2 Capacitance Sense Bias Calibration:

The MIC component's capacitance value should be adjusted accordingly for the following three operating states:

1. **Capacitance value when unused (The Sense_ADC is about half scale of ADC resolution)**
2. **Capacitance value during inhalation (The Sense_ADC will change with the capacitance)**
3. **Capacitance value when blowing (The Sense_ADC will change with the capacitance)**

The capacitance sense needs to be calibrated with the MIC idling without external airflow pressure. The purpose of the Sense bias calibration is to convert the MIC capacitance to a suitable ADC value, which is usually set at half the scale of the ADC resolution. Adjust the calibration value of the sense_bias register so that the voltage output by the sense module can be close to the half-scale voltage of ADC_Vref.

The sense bias calibration can be performed under the following two conditions: (calibration is performed when the MIC is idling and there is no external airflow pressure)

**1.**After the chip is powered on and the program is initialized, the sense bias calibration procedure is performed
**2.**After the program is set up with specific operations, it enters the re-sense bias calibration procedure.

The steps of the sense bias calibration process are as follows:

**Step_1:** Set the sense_bias register to 0xFF, read the MIC_Sense value, and execute Step_2
**Step_2:** Determine whether the MIC_Sense value is close to the half-scale of ADC_Vref (1024 for 11-Bits ADC).
If the judgment result is yes, execute Step_4. If the judgment result is no, execute Step.3
**Step_3:** Decrement the sense_bias register setting value, read the MIC_Sense value, and execute Step_2
**Step_4:** The sense bias calibration is completed

The sense bias calibration routine code is as follows:

```
// MIC Bias voltage calibration
// SENSE_BIAS Initialize and calibrate Vo to 0.5*ADC_Vref
void       SENSE_BIAS_Trim(void)
{
  byte all_tune = 0xFF;
  byte SENSE_VREF_Buf = all_tune;              // Sense BIAS register buffer
  $ ADCRGC BG, ADC_BG, BG_2V68;                // ADC Reference Voltage = BG 2.68V
  $ ADCM     /(2000000/500000);                // ADC clock = 500K, System clock = 2M
@@.Begin:
  SENSE_BIAS = all_tune;
  $ SENSE_CR Enable, SENSE_MIC_MODE;           // Set Sense to MIC mode
  $ ADCC    Enable, Sense, Sense_Rdy;          // ADC Channel Selection Sense Channel
   .delay 10;                                  // must delay 5us, system clock = 2M
  AD_START = 1;                                // Start ADC
  SENSE_CR.Start = 1;                          // Start Sense ADC
  while(!AD_DONE) {}                           // Wait for ADC measurement to complete
  Sense_Value$1 = ADCRH;
```

```
    Sense_Value$0 = ADCRL;
    Sense_Value >>= 5;                          // The ADC value is calculated with 11-bit accuracy.
    SENSE_CR.Stop = 1;                          // Stop Sense ADC

    // Compare the Sense ADC data with 0.5*ADC_Vref (11BIT ADC data, so 1024),
    // and find the combination closest to 0.5*ADC_Vref     static word min_Detal = 0xFFFF;
    static word Diff;


    Diff = 1024 - Sense_Value;
    T1SN FLAG.1;                                // CF: Operation carry borrow sign
    goto @F;
    Diff = -Diff;
@@:
    if(Diff <= min_Detal)                       // Get the configuration closest to 1024
    {
            min_Detal = Diff;      // Update minimum value data and register configuration
            SENSE_VREF_Buf = all_tune;
    }
    if (all_tune != 0)
    {
       all_tune--;                              // Reconfigure the SENSE_VREF register
     goto @B.Begin;
    }
@@.End:
// Configure the best SENSE_BIAS so that the Sense_ADC is closest to 1024
    SENSE_BIAS = SENSE_VREF_Buf;
}
```

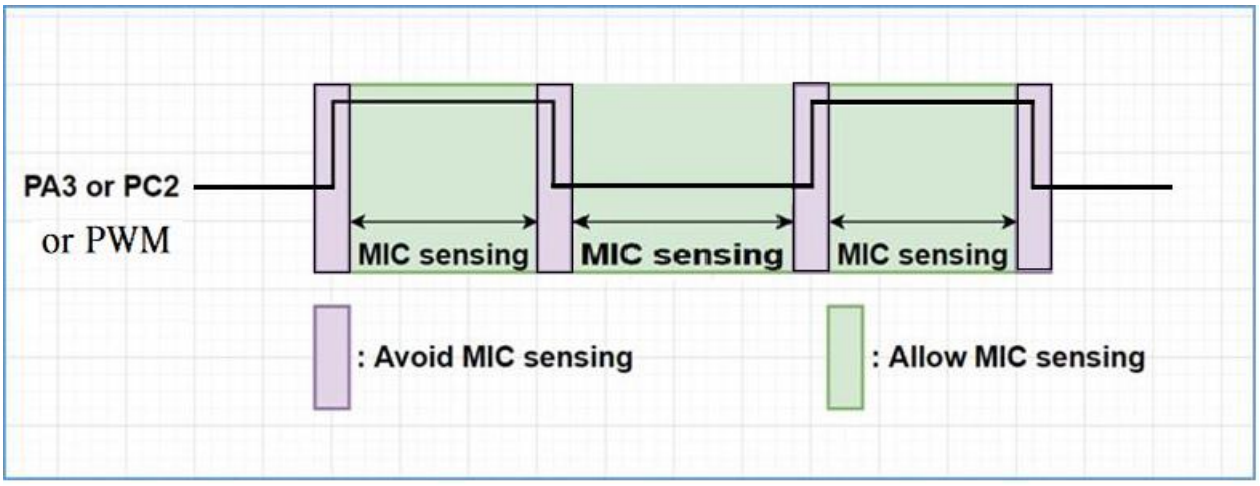
### 5.16.3 Capacitance Sense note:

MIC sensing is a high-precision conversion and is easily interfered by external and other digital signal waveforms. Therefore, the MIC sensing should avoid sampling at the positive and negative edges of the PWM waveform. It is recommended that MIC sensing should be synchronized with PWM and fixed at the high level or low level of the PWM wave for sampling.

(1) Avoid MIC sensing during PA3/PC2 toggle edge.

(2) Aviod MIC sensing during edges of the PWM.

(3) As illustrated below.




### 5.16.4 Resistance Sense：(Heating Resistor Sense, HRS)

The PFB190 resistance sense (Heating Resistor Sense, HRS) supports general heating wire resistance value detection (0.5Ω to 1.5Ω). The working principle of the resistance detector (HRS) is that the Sense module will output a constant current of 50mA from the PA0 or PA3 pin to the external load resistor (usually a heating wire). A small voltage is generated on the external heating wire resistor, which is amplified by the internal sense amplifier and subtracted from Vm (1.2V) to obtain the sense output voltage Vo. Vo will be output to the Sense channel of the ADC module and converted into digital by the ADC module. The RMS measurement channel can be set through CodeOption or by the user's program in bit 7 of RMS_CR.

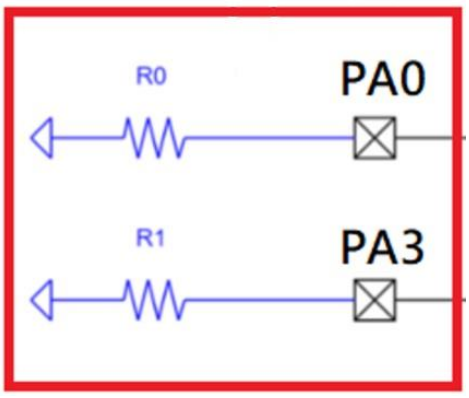



Fig.20: Connection diagram of external heating wire

| HRS calculation formula | |
|---|---|
| $V_o = V_m - Gain * I_{rms} * R$ | |
| $\frac{V_o}{ADC\_Vref} = \frac{ADC\_Value}{2^n}$ | |
| **Formula parameter introduction** | |
| **Vo** | is the Sense Block output voltage |
| **Vm** | fixed at 1.2V. ADC_Vref is controlled by register ADCRGC[2:1] bits<br>ADCRGC[2:1] = (BG_1V2,BG_2V2,BG_2V4,BG_2V68) |
| **Gain** | is fixed at x4 |
| **Irms** | Internal adjustable current source of the chip, controlled by register RMS_CR[5:0],<br>it is not recommended for customers to manually adjust. |
| **R** | is the resistance of the heating wire |
| **2n** | is the resolution of ADC, the maximum resolution of ADC is 12BIT |
| **ADC_Value** | is the value collected by the 12-Bits ADC, which is composed of ADCRH[7:0] and ADCRL[7:4] |
| ADC_Result[11:0] = ADRCRH[7:0] \| ADCRL[7:5] | |

**Example:**

- In this case, the external resistance load R is 1 Ω
- Internal constant current, calibrated to 50 mA at the factory
- sense_gain set to 4
- ADC VREF at 2.40V
- $V_o = 1.2V - 4 \cdot 50mA \cdot 1\Omega = 1.00V$
- $SO \ldots 12\text{bits}\ ADC_{cout} = \frac{1.00V}{1LSB} = \frac{1.00V}{\frac{2.40V}{4096}} \approx 1706$

The Sense_ADC conversion codes for MIC and HRMS are as follows:

```
// Get the value of Sense
void Get_Sense_Data(void)
{
// Before collecting MIC data, the PWM OUT-PIN needs to be turned off to prevent the MIC sampling data from
// being interfered with.
//---- The PWM setting can be modified by the user according to actual use. ----
// byte LPWMG0C_BUF = LPWMG0C;
// byte LPWMG1C_BUF = LPWMG1C;
// $ LPWMG0C;
// $ LPWMG1C;

   Byte    tmp = SENSE_CR & 0b_0_11_0_0000;
   if (tmp == 0b_0_10_0_0000)
           $ ADCRGC BG, BG_2V4;                    // HRMS Mode
   else
           $ ADCRGC BG, BG_2V68;                   // MIC Mode
   $ ADCC    Enable, Sense, Sense_Rdy;   // ADC channel selection Sense channel, Sense mode and enable
   .delay 10;                              // Delay 5us, system clock = 2M
   AD_START = 1;                           // Start ADC
   SENSE_CR.Start = 1;                     // Start Sense ADC
   while(!AD_DONE) {}                      // Wait for ADC measurement to complete
    //---- The PWM setting can be modified by the user according to actual use. ----
    // LPWMG0C = LPWMG0C_BUF;
    // LPWMG1C = LPWMG1C_BUF;

   Sense_Value$1 = ADCRH;
   Sense_Value$0 = ADCRL;
   Sense_Value >>= 5;                      // The ADC value has 11-bit accuracy.
   SENSE_CR.Stop = 1;                      // Stop Sense ADC
}
```

# 6. IO Registers

## 6.1 ACC Status Flag Register (*flag*), IO address = 0x00

| Bit | Reset | R/W | Description |
|---|---|---|---|
| 7 - 4 | - | - | Reserved. Please do not use. |
| 3 | 0 | R/W | OV (Overflow Flag). This bit is set to be 1 whenever the sign operation is overflow. |
| 2 | 0 | R/W | AC (Auxiliary Carry Flag). 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 | 0 | R/W | C (Carry Flag). 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 | 0 | R/W | Z (Zero Flag). This bit will be set when the result of arithmetic or logic operation is zero; Otherwise, it is cleared. |

## 6.2 Stack Pointer Register (*sp*), IO 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. |

## 6.3 Clock Mode Register (clkmd), IO address = 0x03

| Bit | Reset | R/W | Description | |
|---|---|---|---|---|
| 7 - 5 | 111 | R/W | System clock (CLK) selection: | |
| | | | Type 0, clkmd[3]=0 | Type 1, clkmd[3]=1 |
| | | | 000: IHRC÷4<br>001: IHRC÷2<br>01X: reserved<br>10X: reserved<br>110: ILRC÷4<br>111: ILRC (default) | 000: IHRC÷16<br>001: IHRC÷8<br>010: ILRC÷16<br>011: IHRC÷32<br>100: IHRC÷64<br>101: reserved<br>11x: reserved |
| 4 | 1 | R/W | Internal High RC 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].<br>0 / 1: Type 0 / Type 1. | |
| 2 | 1 | R/W | Internal Low RC Enable. 0 / 1: disable / enable<br>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 Interrupt Enable Register (inten), IO address = 0x04

| Bit | Reset | R/W | Description |
|---|---|---|---|
| 7 | 0 | R/W | Enable interrupt from Timer3. 0 / 1: Disable / Enable. |
| 6 | 0 | R/W | Enable interrupt from Timer2. 0 / 1: Disable / Enable. |
| 5 | 0 | R/W | Enable interrupt from PWMG0. 0 / 1: Disable / Enable. |
| 4 | 0 | R/W | Reserved. |
| 3 | 0 | R/W | Enable interrupt from ADC. 0 / 1: Disable / Enable. |
| 2 | 0 | R/W | Enable interrupt from Timer16 overflow. 0 / 1: Disable / Enable. |
| 1 | 0 | R/W | Enable interrupt from PB0/PA3/PB6. 0 / 1: Disable / Enable. |
| 0 | 0 | R/W | Enable interrupt from PA0/PB5/PA7. 0 / 1: Disable / Enable. |

### 6.5 Interrupt Request Register (intrq), IO address = 0x05

| Bit | Reset | R/W | Description |
|---|---|---|---|
| 7 | - | R/W | Interrupt Request from Timer3, this bit is set by hardware and cleared by software.<br>0 / 1: No request / Request |
| 6 | - | R/W | Interrupt Request from Timer2, this bit is set by hardware and cleared by software.<br>0 / 1: No request / Request |
| 5 | - | R/W | Interrupt Request from PWMG0, this bit is set by hardware and cleared by software.<br>0 / 1: No request / Request |
| 4 | - | R/W | Reserved |
| 3 | - | R/W | Interrupt Request from ADC, this bit is set by hardware and cleared by software.<br>0 / 1: No request / Request |
| 2 | - | R/W | Interrupt Request from Timer16, this bit is set by hardware and cleared by software.<br>0 / 1: No request / Request |
| 1 | - | R/W | Interrupt Request from pin PB0/PA3/PB6, this bit is set by hardware and cleared by software.<br>0 / 1: No request / Request |
| 0 | - | R/W | Interrupt Request from pin PA0/PB5/PA7, this bit is set by hardware and cleared by software.<br>0 / 1: No Request / request |

### 6.6 Timer16 mode Register (t16m), IO address = 0x06

| Bit | Reset | R/W | Description |
|---|---|---|---|
| 7 - 5 | 000 | R/W | Timer16 Clock source selection.<br>000: Disable<br>001: CLK (system clock)<br>010: Reserved<br>011: Reserved<br>100: IHRCI<br>101: Reserved<br>110: ILRC<br>111: PA0 falling edge (from external pin) |
| 4 - 3 | 00 | R/W | Timer16 clock pre-divider.<br>00: ÷1<br>01: ÷4<br>10: ÷16<br>11: ÷64 |
| 2 - 0 | 000 | R/W | Interrupt source selection. Interrupt event happens when the selected bit status is changed.<br>0 : bit 8 of Timer16<br>1 : bit 9 of Timer16<br>2 : bit 10 of Timer16<br>3 : bit 11 of Timer16<br>4 : bit 12 of Timer16<br>5 : bit 13 of Timer16<br>6 : bit 14 of Timer16<br>7 : bit 15 of Timer16 |

### 6.7 Port A Pull-Low Register (papl), IO address = 0x08

| Bit | Reset | R/W | Description |
|---|---|---|---|
| 7 - 0 | 0x00 | R/W | Port A pull-low register. This register is used to enable the internal pull-low device on each corresponding pin of port A and this pull high function is active only for input mode.<br>0 / 1 : Disable / Enable<br>PAPL.4 and PAPL.6: Reserved |

### 6.8 Port B Pull-Low Register (pbpl), IO address = 0x09

| Bit | Reset | R/W | Description |
|---|---|---|---|
| 7 – 0 | 0x00- | R/W | Port B pull-low register. This register is used to enable the internal pull-low device on each corresponding pin of port B and this pull high function is active only for input mode.<br>0 / 1 : Disable / Enable |

### 6.9 Port C Pull-Low Register (pcpl), IO address = 0x0A

| Bit | Reset | R/W | Description |
|---|---|---|---|
| 7 | - | - | Reserved |
| 6 - 0 | 0x00 | R/W | Port C pull-low register. This register is used to enable the internal pull-low device on each corresponding pin of port C and this pull high function is active only for input mode.<br>0 / 1 : Disable / Enable |

### 6.10 Interrupt Edge Select Register (integs), IO address = 0x0c

| Bit | Reset | R/W | Description |
|---|---|---|---|
| 7 - 5 | - | - | Reserved |
| 4 | 0 | WO | Timer16 edge selection.<br>0 : rising edge of the selected bit to trigger interrupt<br>1 : falling edge of the selected bit to trigger interrupt |
| 3 - 2 | 00 | WO | PB0/PA3/PB6 edge selection.<br>00: both rising edge and falling edge of the selected bit to trigger interrupt<br>01: rising edge of the selected bit to trigger interrupt<br>10: falling edge of the selected bit to trigger interrupt<br>11: reserved |
| 1 - 0 | 00 | WO | PA0/PB5/PA7 edge selection.<br>00 : both rising edge and falling edge of the selected bit to trigger interrupt<br>01 : rising edge of the selected bit to trigger interrupt<br>10 : falling edge of the selected bit to trigger interrupt<br>11 : reserved |

### 6.11 Port A Digital Input Enable Register (padier), IO address = 0x0d

| Bit | Reset | R/W | Description |
|---|---|---|---|
| 7 - 0 | 0x00 | WO | Enable Port A digital input to prevent leakage when the pin is assigned for AD input. When disable is selected, the wakeup function from this pin is also disabled.<br>0 / 1 : Disable / Enable<br>PADIER.4 and PADIER.6 are not used. |

### 6.12 Port B Digital Input Enable Register (pbdier), IO address = 0x0e

| Bit | Reset | R/W | Description |
|---|---|---|---|
| 7 - 0 | 0x00 | WO | Enable Port B digital input to prevent leakage when the pin is assigned for AD input. When disable is selected, the wakeup function from this pin is also disabled.<br>0 / 1 : Disable / Enable |

### 6.13 Port C Digital Input Enable Register (pcdier), IO address = 0x0f

| Bit | Reset | R/W | Description |
|---|---|---|---|
| 7 | - | - | Reserved |
| 6 - 0 | 0x00 | WO | Enable Port C digital input to prevent leakage when the pin is assigned for AD input. When disable is selected, the wakeup function from this pin is also disabled.<br>0 / 1 : Disable / Enable |

**Note:** Detail settings please refer to Section 9.2.

### 6.14 Port A Data Register (pa), IO address = 0x10

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

### 6.15 Port A Control Register (pac), IO address = 0x11

| Bit | Reset | R/W | Description |
|---|---|---|---|
| 7 - 0 | 0x00 | R/W | Port A Control Registers. These registers are used to define the input mode or output mode of each corresponding pin of Port A.<br>0 / 1: Input / Output<br>PAC.4 and PAC.6 are not used |

### 6.16 Port A Pull-High Register (paph), IO address = 0x12

| Bit | Reset | R/W | Description |
|---|---|---|---|
| 7 - 0 | 0x00 | R/W | Port A pull-high register. This register is used to enable the internal pull-high device on each corresponding pin of port A and this pull high function is active only for input mode.<br>0 / 1: Disable / Enable<br>PAPH.4 and PAPH.6: Reserved |

### 6.17 Port B Data Register (pb), IO address = 0x13

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

### 6.18 Port B Control Register (pbc), IO address = 0x14

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

### 6.19 Port B Pull-High Register (pbph), IO address = 0x15

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

### 6.20 Port C Data Register (pc), IO address = 0x16

| Bit | Reset | R/W | Description |
|---|---|---|---|
| 7 | - | - | Reserved |
| 6 - 0 | 0x00 | R/W | Data register for Port C. |

### 6.21 Port C Control Register (pcc), IO address = 0x17

| Bit | Reset | R/W | Description |
|---|---|---|---|
| 7 | - | - | Reserved |
| 6 - 0 | 0x00 | R/W | Port C control register. This register is used to define input mode or output mode for each corresponding pin of port C.<br>0 / 1: Input / Output |

### 6.22 Port C Pull-High Register (pcph), IO address = 0x18

| Bit | Reset | R/W | Description |
|---|---|---|---|
| 7 | - | - | Reserved |
| 6 - 0 | 0x00 | R/W | Port C pull-high register. This register is used to enable the internal pull-high device on each corresponding pin of port C and this pull high function is active only for input mode.<br>0 / 1: Disable / Enable |

### 6.23 ADC Control Register (adcc), IO address = 0x20

| Bit | Reset | R/W | Description |
|---|---|---|---|
| 7 | 0 | R/W | Enable ADC function. 0/1: Disable/Enable. |
| 6 | 0 | R/W | ADC process control bit.<br>Write “1” to start conversion<br>Read “1” to indicate the ADC is ready or end of conversion. |
| 5 - 2 | 0000 | R/W | Channel selector. These four bits are used to select input signal for AD conversion.<br>0000: PB0/AD0,<br>0001: PB1/AD1,<br>0010: PB2/AD2,<br>0011: PB3/AD3,<br>0100: PB4/AD4,<br>0101: PB5/AD5,<br>0110: PB6/AD6,<br>0111: PB7/AD7,<br>1000: PA3/AD8, |

| Bit | Reset | R/W | Description |
|---|---|---|---|
| | | | 1001: Reserved,<br>1010: PA0/AD10,<br>1011: PC1/AD11,<br>1100: PC2/AD12,<br>1110: Sense<br>1111: Bandgap voltage or 3/8*$V_{DD}$<br>Others: Reserved |
| 1 | 0 | R/W | 0 : ADC Control from ADC (Normal ADC Mode)<br>1 : Sense_Rdy (Sense_ADC Mode) |
| 0 | - | - | Reserved |

### 6.24 ADC Mode Register (adcm), IO address = 0x21

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

### 6.25 ADC Regulator Control Register (adcrgc), IO address = 0x24

| Bit | Reset | R/W | Description |
|---|---|---|---|
| 7 - 5 | 000 | WO | These three bits are used to select input signal for ADC reference high voltage channel<br>000: $V_{DD}$,<br>001: ADC Bandgap<br>Others: reserved |
| 4 | 0 | WO | ADC channel F selector:<br>0: Bandgap reference voltage<br>1: 3/8*$V_{DD}$. |
| 3 | - | - | Reserved |
| 2 - 1 | 00 | WO | ADC Vref from Bandgap<br>00: 1.2V<br>01: 2.2V<br>10: 2.4V<br>11: 2.68V |
| 0 | - | - | Reserved. Please keep 0. |

### 6.26 ADC Result High Register (adcrh), IO address = 0x22

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

### 6.27 ADC Result Low Register (adcrl), IO address = 0x23

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

### 6.28 MISC Register (misc), IO address = 0x26

| Bit | Reset | R/W | Description |
|---|---|---|---|
| 7 - 6 | - | - | Reserved. (keep 0 for future compatibility) |
| 5 | 0 | WO | Enable fast Wake up.<br>0: Normal wake up. The wake-up time is 3000 ILRC clocks (Not for fast boot-up)<br>1: Fast wake up. The wake-up time is 45 ILRC clocks. |
| 4 | 0 | WO | Enable VDD/2 LCD bias voltage generator.<br>0 / 1 : Disable / Enable (ICE cannot be dynamically switched)<br>If Code Option selects LCD output, but MISC.4 does not set to 1, then the VDD/2 bias cannot be output on the IC. However, the emulator is always OK. Two above phenomena are different. |
| 3 | - | - | Reserved. |
| 2 | 0 | WO | Disable LVR function. 0 / 1: Enable / Disable |

| Bit | Reset | R/W | Description |
|---|---|---|---|
| 1 - 0 | 00 | WO | Watch dog time out period<br>00: 8k ILRC clock period<br>01: 16k ILRC clock period<br>10: 64k ILRC clock period<br>11: 256k ILRC clock period |

### 6.29 Timer2 Control Register (tm2c), IO address = 0x28

| Bit | Reset | R/W | Description |
|---|---|---|---|
| 7 - 4 | 0000 | R/W | Timer2 clock selection.<br>0000 : disable<br>0001 : CLK<br>0010 : IHRC or IHRC *2 (by code option TMx_source)<br>0011 : Reserved<br>0100 : ILRC<br>0101 : NILRC<br>1000 : PA0 (rising edge)<br>1001 : ~PA0 (falling edge)<br>1010 : PB0 (rising edge)<br>1011 : ~PB0 (falling edge)<br>110X : Reserved<br>Others : Reserved<br>**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.<br>00 : disable<br>01 : PB2<br>10 : PA3<br>11 : PB4 |
| 1 | 0 | R/W | Timer 2 mode selection.<br>0 : Period Mode<br>1 : PWM Mode |
| 0 | 0 | R/W | Enable to inverse the polarity of Timer2 output.<br>0 : Disable<br>1 : Enable |

### 6.30 Timer2 Counter Register (tm2ct), IO address = 0x29

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

### 6.31 Timer2 Scalar Register (tm2s), IO address = 0x2A

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

### 6.32 Timer2 Bound Register (tm2b), IO address = 0x2B

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

### 6.33 Timer3 Control Register (tm3c), IO address = 0x2C

| Bit | Reset | R/W | Description |
|---|---|---|---|
| 7 - 4 | 0000 | R/W | Timer3 clock selection.<br>0000 : Disable<br>0001 : CLK<br>0010 : IHRC or IHRC *2 (by code option TMx_source)<br>0011 : Reserved<br>0100 : ILRC<br>0101 : NILRC<br>1000 : PA0 (rising edge)<br>1001 : ~PA0 (falling edge)<br>1010 : PB0 (rising edge)<br>1011 : ~PB0 (falling edge)<br>110X : Reserved<br>Others : Reserved |
| 3 - 2 | 00 | R/W | Timer3 output selection.<br>00 : disable<br>01 : PB5<br>10 : PB6<br>11 : PB7 |

| Bit | Reset | R/W | Description |
|---|---|---|---|
| 1 | 0 | R/W | Timer3 mode selection.<br>0 : Period Mode<br>1 : PWM Mode |
| 0 | 0 | R/W | Enable to inverse the polarity of Timer3 output.<br>0 : Disable<br>1 : Enable |

### 6.34 Timer3 Counter Register (tm3ct), IO address = 0x2D

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

### 6.35 Timer3 Scalar Register (tm3s), IO address = 0x2E

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

### 6.36 Timer3 Bound Register (tm3b), IO address = 0x2F

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

### 6.37 Charger Current Control Register (chg_ctrl ), IO address = 0x32

| Bit | Reset | R/W | Description |
|---|---|---|---|
| 7 | - | - | Reserved |
| 6 - 5 | 00 | WO | Charge current select<br>00: 145mA<br>00: 250mA<br>00: 375mA<br>00: 500mA |
| 4 - 0 | - | - | Reserved |

### 6.38 Charger Voltage Control Register (chg_vbat), IO address = 0x33

| Bit | Reset | R/W | Description |
|---|---|---|---|
| 7 - 5 | 0x000 | WO | Charge Voltage select<br>000: 4.2V<br>001: 4.1V<br>010: 4.35V<br>011: 3.6V<br>100: 3.7V<br>101: 3.8V<br>110: 4.24V<br>111: 4.28V |
| 4 - 0 | - | - | Reserved |

### 6.39 Charger Output Signal (chgs), IO address = 0x34

| Bit | Reset | R/W | Description |
|---|---|---|---|
| 7 | - | RO | Battery Voltage state<br>0 / 1: Battery is not full / full. |
| 6 | - | RO | Charger Mode.<br>0 / 1: Normal mode / Charge mode. |
| 5 | - | - | Reserved |
| 4 | - | RO | The status indicator bit of Vcc voltage source.<br>It can be used to judge the status of charging Vcc<br>1: Vcc < $V_{BAT}$<br>0: Vcc > $V_{BAT}$ |
| 3 | - | RO | Charging action indicator bit.<br>1: Charging in progress. Vcc is too low and charger shutdown.<br>0: Vcc is normal. |
| 2 | - | RO | Over-temperature charging protection bit (Temperature less than 90°C).<br>Read "0" to trigger the over-temperature protection, and read "1" to be normal. |
| 1 | - | RO | Over temperature charging protection bit (Temperature less than 140°C).<br>Read "0" to trigger the over-temperature protection, and read "1" to be normal. |
| 0 | - | RO | 0 / 1: VCC is floating / connecting. |

### 6.40 Charger Option control (chg_opr), IO address = 0x35

| Bit | Reset | R/W | Description |
|---|---|---|---|
| 7 | 1 | WO | Enable the charger function module. 0/1: Disable/Enable. |
| 6 | 1 | - | Reserved |
| 5 | 1 | - | Reserved |
| 4 | 1 | WO | OVP function control<br>0 / 1: off / on |
| 3 | 0 | WO | Reserved, only 0 can be written. |
| 2 - 0 | - | - | Reserved |

### 6.41 Sense control Register (sense_cr), IO address = 0x37

| Bit | Reset | R/W | Description |
|---|---|---|---|
| 7 | 0 | R/W | 0 / 1: Sense function Disable / Enable. |
| 6 - 5 | 01 | R/W | Sense mode select<br>00: Microphone Capacitor Sense (MCS)<br>01: Reserved<br>10: Heating Resistor Sense (HRS)<br>11: Reserved |
| 4 | 0 | RW | 0 / 1: Sense function stop / start. |
| 3 | 0 | R/W | Reserved |
| 2 | 0 | R/W | Sense Operation Time Normal / Double(X2)<br>0: x1<br>1: x2 |
| 1 - 0 | - | - | Reserved |

### 6.42 Sense bias Register (sense_bias), IO address = 0x38

| Bit | Reset | R/W | Description |
|---|---|---|---|
| 7-6 | 01 | R/W | V2, Sense bias voltage for MCS. (coarse-tune)<br>00 : min<br>11 : max |
| 5 - 0 | 0x20 | R/W | V1, Sense bias voltage for MCS. (fine-tune)<br>00_0000 : min<br>11_1111 : max |

### 6.43 RMS control Register (rms_cr), IO address = 0x39

| Bit | Reset | R/W | Description |
|---|---|---|---|
| 7 | - | R/W | 0 / 1: RMS Channel Select PA0 / PA3 |
| 6 - 0 | - | - | Reserved |

### 6.44 PWMG0 control Register (pwmg0c), IO address = 0x40

| Bit | Reset | R/W | Description |
|---|---|---|---|
| 7 | - | - | Reserved |
| 6 | - | RO | Output status of PWMG0 generator. |
| 5 | 0 | R/W | Pwmg0 output. 0 / 1: Buffered / Inverted |
| 4 | 0 | R/W | PWMG0 output selection.<br>0: PWMG0 Output<br>1: (PWMG0 ^ PWMG1) \| (PWMG0 \| PWMG1) (by pwmg0c.0) |
| 3 - 1 | 000 | R/W | PWMG0 Output Port Selection<br>000: PWMG0 Output Disable<br>001: PWMG0 Output to PB5<br>010: PWMG0 Output to PC2<br>011: PWMG0 Output to PA0<br>100: PWMG0 Output to PB4<br>Other: Reserved |
| 0 | 0 | R/W | PWMG0 output pre- selection.<br>0: PWMG0 ^ PWMG1<br>1: PWMG0 \| PWMG1 |

### 6.45 PWMG Clock Register (pwmgclk), IO address = 0x41

| Bit | Reset | R/W | Description |
|---|---|---|---|
| 7 | 0 | WO | PWMG Disable/ Enable<br>0: PWMG Disable<br>1: PWMG Enable |
| 6 - 4 | 000 | WO | LPWMG clock pre-scalar.<br>000: ÷1<br>001: ÷2<br>010: ÷4<br>011: ÷8<br>100: ÷16<br>101: ÷32<br>110: ÷64<br>111: ÷128 |
| 3 - 1 | - | - | Reserved |
| 0 | 0 | WO | PWMG clock source selection<br>0: System Clock<br>1: IHRC or IHRC*2 (by code option PWM_Source). |

### 6.46 PWMG0 Duty Value High Register (pwmg0dth), IO address = 0x42

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

### 6.47 PWMG0 Duty Value Low Register (pwmg0dtl), IO address = 0x43

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

**Note:** It's necessary to write PWMG0 Duty_Value Low Register before writing PWMG0 Duty_Value High Register.

### 6.48 PWMG Counter Upper Bound High Register (pwmgcubh ), IO address = 0x44

| Bit | Reset | R/W | Description |
|---|---|---|---|
| 7 - 0 | - | WO | Bit [10:3] of LPWMG Counter Bound. |

### 6.49 PWMG Counter Upper Bound Low Register (pwmgcubl ), IO address = 0x45

| Bit | Reset | R/W | Description |
|---|---|---|---|
| 7 - 6 | - | WO | Bit [2:1] of LPWMG Counter Bound. |
| 5 - 0 | - | - | Reserved |

### 6.50 PWMG1 control Register (pwmg1c), IO address = 0x46

| Bit | Reset | R/W | Description |
|---|---|---|---|
| 7 | - | - | Reserved |
| 6 | - | RO | Output status of PWMG1 generator. |
| 5 | 0 | R/W | Pwmg1 output. 0 / 1: Buffered / Inverted |
| 4 | 0 | R/W | PWMG1 output selection.<br>0: PWMG1<br>1: PWMG2 |
| 3 - 1 | 000 | R/W | PWMG1 Output Port Selection<br>000: PWMG1 Output Disable<br>001: PWMG1 Output to PB6<br>010: PWMG1 Output to PC3<br>011: Reserved<br>100: PWMG1 Output to PB7<br>Other: Reserved |
| 0 | - | - | Reserved |

### 6.51 PWMG1 Duty Value High Register (pwmg1dth), IO address = 0x47

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

### 6.52 PWMG1 Duty Value Low Register (pwmg1dtl), IO address = 0x48

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

**Note:** It's necessary to write PWMG1 Duty_Value Low Register before writing PWMG1 Duty_Value High Register.

### 6.53 PWMG2 control Register (pwmg2c), IO address = 0x49

| Bit | Reset | R/W | Description |
|---|---|---|---|
| 7 | - | - | Reserved |
| 6 | - | RO | Output status of PWMG2 generator. |
| 5 | 0 | R/W | Pwmg2 output. 0 / 1: Buffered / Inverted |
| 4 | 0 | R/W | PWMG2 output selection.<br>0: PWMG2<br>1: PWMG2/2 |
| 3 - 1 | 000 | R/W | PWMG2 Output Port Selection<br>000: PWMG2 Output Disable<br>001: PWMG2 Output to PB3<br>010: PWMG2 Output to PC0<br>011: PWMG2 Output to PA3<br>100: PWMG2 Output to PB2<br>Other: Reserved |
| 0 | - | - | Reserved |

### 6.54 PWMG2 Duty Value High Register (pwmg2dth), IO address = 0x4E

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

### 6.55 PWMG2 Duty Value Low Register (pwmg2dtl), IO address = 0x4F

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

**Note:** It's necessary to write PWMG2 Duty_Value Low Register before writing PWMG2 Duty_Value High Register.

# 7. 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<br>(The operational result is to have carry out for addition or to borrow carry for subtraction in unsigned number system) |
| **AC** | Auxiliary Carry<br>(If there is a carry out from low nibble after the result of ALU operation, this bit is set to 1) |
| **M.n** | Only addressed in 0x00~0x7F (0~127) is allowed |

## 7.1 Data Transfer Instructions

| | |
|---|---|
| *mov* a, I | Move immediate data into ACC.<br>Example: *mov* a, 0x0f;<br>Result: a ← 0fh;<br>Affected flags: 『N』Z 『N』C 『N』AC 『N』OV |
| *mov* M, a | Move data from ACC into memory<br>Example: *mov* MEM, a;<br>Result: MEM ← a<br>Affected flags: 『N』Z 『N』C 『N』AC 『N』OV |
| *mov* a, M | Move data from memory into ACC<br>Example: *mov* a, MEM ;<br>Result: a ← MEM; Flag Z is set when MEM is zero.<br>Affected flags: 『Y』Z 『N』C 『N』AC 『N』OV |
| *mov* a, IO | Move data from IO into ACC<br>Example: *mov* a, pa ;<br>Result: a ← pa; Flag Z is set when pa is zero.<br>Affected flags: 『Y』Z 『N』C 『N』AC 『N』OV |
| *mov* IO, a | Move data from ACC into IO<br>Example: *mov* pb, a;<br>Result: pb ← a<br>Affected flags: 『N』Z 『N』C 『N』AC 『N』OV |
| *nmov* M, a | Take the negative logic (2's complement) of ACC to put on memory<br>Example: *mov* MEM, a;<br>Result: MEM ← 〒a<br>Affected flags: 『N』Z 『N』C 『N』AC 『N』OV<br>Application Example:<br>---<br>*mov* a, 0xf5 ; // ACC is 0xf5<br>*nmov* ram9, a; // ram9 is 0x0b, ACC is 0xf5<br>--- |
| *nmov* a, M | Take the negative logic (2's complement) of memory to put on ACC<br>Example: *mov* a, MEM ;<br>Result: a ← 〒MEM; Flag Z is set when 〒MEM is zero.<br>Affected flags: 『Y』Z 『N』C 『N』AC 『N』OV<br>Application Example:<br>---<br>*mov* a, 0xf5 ;<br>*mov* ram9, a ; // ram9 is 0xf5<br>*nmov* a, ram9 ; // ram9 is 0xf5, ACC is 0x0b<br>--- |
| *ldtabh* index | Load high byte data in MTP program memory to ACC by using index as MTP address. It needs 2T to execute this instruction.<br>Example: *ldtabh index;*<br>Result: a ← {bit 15~8 of MTP [index]};<br>Affected flags: 『N』Z 『N』C 『N』AC 『N』OV<br>Application Example:<br>---<br>*word ROMptr ; // declare a pointer of ROM in RAM* |

```
    ...
    mov         a, la@TableA ;   // assign pointer to ROM TableA (LSB)
    mov         lb@ROMptr, a ;   // save pointer to RAM (LSB)
    mov         a, ha@TableA ;   // assign pointer to ROM TableA (MSB)
    mov         hb@ROMptr, a ;   // save pointer to RAM (MSB)
    ...
    ldtabh      ROMptr ;             // load TableA MSB to ACC (ACC=0X02)
    ...
TableA :    dc      0x0234, 0x0042, 0x0024, 0x0018 ;
```

---

*ldtabl* index

Load low byte data in MTP to ACC by using index as MTP address. It needs 2T to execute this instruction.

Example: *ldtabl index;*

Result: a ← {bit7~0 of MTP [index]};

Affected flags: 『N』Z 『N』C 『N』AC 『N』OV

Application Example:

```
    word        ROMptr ;        // declare a pointer of ROM in RAM
    ...
    mov         a, la@TableA ; // assign pointer to ROM TableA (LSB)
    mov         lb@ROMptr, a ; // save pointer to RAM (LSB)
    mov         a, ha@TableA ; // assign pointer to ROM TableA (MSB)
    mov         hb@ROMptr, a ; // save pointer to RAM (MSB)
    ...
    ldtabl      ROMptr ;            // load TableA LSB to ACC (ACC=0x34)
    ...
TableA :    dc      0x0234, 0x0042, 0x0024, 0x0018 ;
```

---

*ldt16* word

Move 16-bit counting values in Timer16 to memory in word.

Example: *ldt16* word;

Result: word ← 16-bit timer

Affected flags: 『N』Z 『N』C 『N』AC 『N』OV

Application Example:

```
    word        T16val ;            // declare a RAM word
    …
    clear       lb@ T16val ;        // clear T16val (LSB)
    clear       hb@ T16val ;        // clear T16val (MSB)
    stt16       T16val ;            // initial T16 with 0
    …
    set1        t16m.5 ;            // enable Timer16
    …
    set0        t16m.5 ;            // disable Timer 16
    ldt16       T16val ;            // save the T16 counting value to T16val
    ….
```

---

| | |
|---|---|
| *stt16* word | Store 16-bit data from memory in word to Timer16.<br>Example: *stt16* word;<br>Result: 16-bit timer ←word<br>Affected flags: 『N』Z 『N』C 『N』AC 『N』OV<br>Application Example:<br>---<br>word T16val ; // declare a RAM word<br>…<br>*mov* a, 0x34 ;<br>*mov* lb@ T16val , a ; // move 0x34 to T16val (LSB)<br>*mov* a, 0x12 ;<br>*mov* hb@ T16val , a ; // move 0x12 to T16val (MSB)<br>*stt16* T16val ; // initial T16 with 0x1234<br>--- |
| *xch M* | Exchange data between ACC and memory<br>Example: xch MEM ;<br>Result: MEM ← a , a ← MEM<br>Affected flags: 『N』Z 『N』C 『N』AC 『N』OV |
| *popaf* | Restore ACC and flag from the memory which address is specified in the stack pointer.<br>Example: popaf;<br>Result: sp ← sp - 2 ;<br>{Flag, ACC} ← [sp] ;<br>Affected flags: 『Y』Z 『Y』C 『Y』AC 『Y』OV |
| *Idxm index, a* | Move data from ACC to specified memory by indirect method. It needs 2T to execute this instruction.<br>Example: idxm index, a;<br>Result: [index] ← a; where index is declared by word.<br>Affected flags: 『N』Z 『N』C 『N』AC 『N』OV<br>Application Example:<br>---<br>word RAMIndex ; // declare a RAM pointer<br>…<br>mov a, 0x5B ; // assign pointer to an address (LSB)<br>mov lb@RAMIndex, a ; // save pointer to RAM (LSB)<br>mov a, 0x00 ; // assign 0x00 to an address (MSB), should be 0<br>mov hb@RAMIndex, a ; // save pointer to RAM (MSB)<br>…<br>mov a, 0xA5 ;<br>idxm RAMIndex, a ; // mov 0xA5 to memory in address 0x5B<br>--- |
| *idxm a, index* | Move data from specified memory to ACC by indirect method. It needs 2T to execute this instruction.<br>Example: idxm a, index;<br>Result: a ← [index], where index is declared by word.<br>Affected flags: 『N』Z 『N』C 『N』AC 『N』OV<br>Application Example:<br>--- |

```
word        RAMIndex ;              // declare a RAM pointer
…
mov         a, 0x5B ;               // assign pointer to an address (LSB)
mov         lb@RAMIndex, a ;        // save pointer to RAM (LSB)
mov         a, 0x00 ;               // assign 0x00 to an address (MSB), should be 0
mov         hb@RAMIndex, a ;        // save pointer to RAM (MSB)
…
idxm        a, RAMIndex ;           // move memory data in address 0x5B to ACC
```

*pushaf*

Move the ACC and flag register to memory that address specified in the stack pointer.
Example: pushaf;
Result: [sp] ← {flag, ACC};
sp ← sp + 2 ;
Affected flags: 『N』Z 『N』C 『N』AC 『N』OV
Application Example:

```
.romadr 0x10 ;                  // ISR entry address
        pushaf ;                // put ACC and flag into stack memory
        …                       // ISR program
        …                       // ISR program
        popaf ;                 // restore ACC and flag from stack memory
        reti ;
```

## 7.2 Arithmetic Operation Instructions

*add* a, I

Add immediate data with ACC, then put result into ACC
Example: *add* a, 0x0f ;
Result: a ← a + 0fh
Affected flags: 『Y』Z 『Y』C 『Y』AC 『Y』OV

*add* a, M

Add data in memory with ACC, then put result into ACC
Example: *add* a, MEM ;
Result: a ← a + MEM
Affected flags: 『Y』Z 『Y』C 『Y』AC 『Y』OV

*add* M, a

Add data in memory with ACC, then put result into memory
Example: *add* MEM, a;
Result: MEM ← a + MEM
Affected flags: 『Y』Z 『Y』C 『Y』AC 『Y』OV

*addc* a, M

Add data in memory with ACC and carry bit, then put result into ACC
Example: *addc* a, MEM ;
Result: a ← a + MEM + C
Affected flags: 『Y』Z 『Y』C 『Y』AC 『Y』OV

*addc* M, a

Add data in memory with ACC and carry bit, then put result into memory
Example: *addc* MEM, a ;
Result: MEM ← a + MEM + C
Affected flags: 『Y』Z 『Y』C 『Y』AC 『Y』OV

*addc* a

Add carry with ACC, then put result into ACC
Example: *addc* a ;

| | |
|---|---|
| | Result: a ← a + C<br>Affected flags: 『Y』Z 『Y』C 『Y』AC 『Y』OV |
| *addc* M | Add carry with memory, then put result into memory<br>Example: *addc* MEM ;<br>Result: MEM ← MEM + C<br>Affected flags: 『Y』Z 『Y』C 『Y』AC 『Y』OV |
| *nadd* a, M | Add negative logic (2's complement) of ACC with memory<br>Example: *nadd a, MEM ;*<br>Result: a ← ∓a + MEM<br>Affected flags: 『Y』Z 『Y』C 『Y』AC 『Y』OV |
| *nadd* M, a | Add negative logic (2's complement) of memory with ACC<br>Example: *nadd MEM, a ;*<br>Result: MEM ← ∓MEM + a<br>Affected flags: 『Y』Z 『Y』C 『Y』AC 『Y』OV |
| *sub* a, I | Subtraction immediate data from ACC, then put result into ACC.<br>Example: *sub* a, 0x0f;<br>Result: a ← a - 0fh ( a + [2's complement of 0fh] )<br>Affected flags: 『Y』Z 『Y』C 『Y』AC 『Y』OV |
| *sub* a, M | Subtraction data in memory from ACC, then put result into ACC<br>Example: *sub* a, MEM ;<br>Result: a ← a - MEM ( a + [2's complement of M] )<br>Affected flags: 『Y』Z 『Y』C 『Y』AC 『Y』OV |
| *sub* M, a | Subtraction data in ACC from memory, then put result into memory<br>Example: *sub* MEM, a;<br>Result: MEM ← MEM - a ( MEM + [2's complement of a] )<br>Affected flags: 『Y』Z 『Y』C 『Y』AC 『Y』OV |
| *subc* a, M | Subtraction data in memory and carry from ACC, then put result into ACC<br>Example: *subc* a, MEM;<br>Result: a ← a – MEM - C<br>Affected flags: 『Y』Z 『Y』C 『Y』AC 『Y』OV |
| *subc* M, a | Subtraction ACC and carry bit from memory, then put result into memory<br>Example: *subc* MEM, a ;<br>Result: MEM ← MEM – a - C<br>Affected flags: 『Y』Z 『Y』C 『Y』AC 『Y』OV |
| *subc* a | Subtraction carry from ACC, then put result into ACC<br>Example: *subc* a;<br>Result: a ← a - C<br>Affected flags: 『Y』Z 『Y』C 『Y』AC 『Y』OV |
| *subc* M | Subtraction carry from the content of memory, then put result into memory<br>Example: *subc* MEM;<br>Result: MEM ← MEM - C<br>Affected flags: 『Y』Z 『Y』C 『Y』AC 『Y』OV |
| *inc* M | Increment the content of memory<br>Example: *inc* MEM ; |

| | |
|---|---|
| | Result: MEM ← MEM + 1<br>Affected flags: 『Y』Z 『Y』C 『Y』AC 『Y』OV |
| *dec* M | Decrement the content of memory<br>Example: *dec* MEM;<br>Result: MEM ← MEM - 1<br>Affected flags: 『Y』Z 『Y』C 『Y』AC 『Y』OV |
| *clear* M | Clear the content of memory<br>Example: *clear* MEM ;<br>Result: MEM ← 0<br>Affected flags: 『N』Z 『N』C 『N』AC 『N』OV |
| *mul* | Multiplication operation, 8x8 unsigned multiplications will be executed.<br>Example: *mul* ;<br>Result: {MulRH,ACC} ← ACC * MulOp<br>Affected flags: 『N』Z 『N』C 『N』AC 『N』OV<br>Application Example :<br>---<br>…<br>mov a, 0x5a ;<br>mov mulop, a ;<br>mov a, 0xa5 ;<br>mul // 0x5A * 0xA5 = 3A02 (mulrh + ACC)<br>mov ram0, a ; // LSB, ram0=0x02<br>mov a, mulrh ; // MSB, ACC=0X3A<br>…<br>--- |

## 7.3 Shift Operation Instructions

| | |
|---|---|
| *sr* a | Shift right of ACC, shift 0 to bit 7<br>Example: *sr* a ;<br>Result: a (0,b7,b6,b5,b4,b3,b2,b1) ← a (b7,b6,b5,b4,b3,b2,b1,b0), C ← a(b0)<br>Affected flags: 『N』Z 『Y』C 『N』AC 『N』OV |
| *src* a | Shift right of ACC with carry bit 7 to flag<br>Example: *src* a ;<br>Result: a (c,b7,b6,b5,b4,b3,b2,b1) ← a (b7,b6,b5,b4,b3,b2,b1,b0), C ← a(b0)<br>Affected flags: 『N』Z 『Y』C 『N』AC 『N』OV |
| *sr* M | Shift right the content of memory, shift 0 to bit 7<br>Example: *sr* MEM ;<br>Result: MEM(0,b7,b6,b5,b4,b3,b2,b1) ← MEM(b7,b6,b5,b4,b3,b2,b1,b0), C ← MEM(b0)<br>Affected flags: 『N』Z 『Y』C 『N』AC 『N』OV |
| *src* M | Shift right of memory with carry bit 7 to flag<br>Example: *src* MEM ;<br>Result: MEM(c,b7,b6,b5,b4,b3,b2,b1) ← MEM (b7,b6,b5,b4,b3,b2,b1,b0), C ← MEM(b0)<br>Affected flags: 『N』Z 『Y』C 『N』AC 『N』OV |
| *sl* a | Shift left of ACC shift 0 to bit 0<br>Example: *sl* a ;<br>Result: a (b6,b5,b4,b3,b2,b1,b0,0) ← a (b7,b6,b5,b4,b3,b2,b1,b0), C ← a (b7)<br>Affected flags: 『N』Z 『Y』C 『N』AC 『N』OV |

| | |
|---|---|
| *slc* a | Shift left of ACC with carry bit 0 to flag<br>Example: *slc* a ;<br>Result: a (b6,b5,b4,b3,b2,b1,b0,c) ← a (b7,b6,b5,b4,b3,b2,b1,b0), C ← a(b7)<br>Affected flags: 『N』Z 『Y』C 『N』AC 『N』OV |
| *sl* M | Shift left of memory, shift 0 to bit 0<br>Example: *sl* MEM ;<br>Result: MEM (b6,b5,b4,b3,b2,b1,b0,0) ← MEM (b7,b6,b5,b4,b3,b2,b1,b0), C ← MEM(b7)<br>Affected flags: 『N』Z 『Y』C 『N』AC 『N』OV |
| *slc* M | Shift left of memory with carry bit 0 to flag<br>Example: *slc* MEM ;<br>Result: MEM (b6,b5,b4,b3,b2,b1,b0,C) ← MEM (b7,b6,b5,b4,b3,b2,b1,b0), C ← MEM (b7)<br>Affected flags: 『N』Z 『Y』C 『N』AC 『N』OV |
| *swap* a | Swap the high nibble and low nibble of ACC<br>Example: *swap* a ;<br>Result: a (b3,b2,b1,b0,b7,b6,b5,b4) ← a (b7,b6,b5,b4,b3,b2,b1,b0)<br>Affected flags: 『N』Z 『N』C 『N』AC 『N』OV |
| *swap* M | Swap the high nibble and low nibble of memory<br>Example: *swap* MEM ;<br>Result: MEM (b3,b2,b1,b0,b7,b6,b5,b4) ← MEM (b7,b6,b5,b4,b3,b2,b1,b0)<br>Affected flags: 『N』Z 『N』C 『N』AC 『N』OV |

## 7.4 Logic Operation Instructions

| | |
|---|---|
| *and* a, I | Perform logic AND on ACC and immediate data, then put result into ACC<br>Example: *and* a, 0x0f ;<br>Result: a ← a & 0fh<br>Affected flags: 『Y』Z 『N』C 『N』AC 『N』OV |
| *and* a, M | Perform logic AND on ACC and memory, then put result into ACC<br>Example: *and* a, RAM10 ;<br>Result: a ← a & RAM10<br>Affected flags: 『Y』Z 『N』C 『N』AC 『N』OV |
| *and* M, a | Perform logic AND on ACC and memory, then put result into memory<br>Example: *and* MEM, a ;<br>Result: MEM ← a & MEM<br>Affected flags: 『Y』Z 『N』C 『N』AC 『N』OV |
| *or* a, I | Perform logic OR on ACC and immediate data, then put result into ACC<br>Example: *or* a, 0x0f ;<br>Result: a ← a \| 0fh<br>Affected flags: 『Y』Z 『N』C 『N』AC 『N』OV |
| *or* a, M | Perform logic OR on ACC and memory, then put result into ACC<br>Example: *or* a, MEM ;<br>Result: a ← a \| MEM<br>Affected flags: 『Y』Z 『N』C 『N』AC 『N』OV |
| *or* M, a | Perform logic OR on ACC and memory, then put result into memory<br>Example: *or* MEM, a ;<br>Result: MEM ← a \| MEM<br>Affected flags: 『Y』Z 『N』C 『N』AC 『N』OV |
| *xor* a, I | Perform logic XOR on ACC and immediate data, then put result into ACC |

| | |
|---|---|
| | Example: *xor* a, 0x0f ;<br>Result: a ← a ^ 0fh<br>Affected flags: 『Y』 Z 『N』 C 『N』 AC 『N』 OV |
| *xor* a, IO | Perform logic XOR on ACC and IO register, then put result into ACC<br>Example: *xor a, pa ;*<br>Result: a ← a ^ pa ; // pa is the data register of port A<br>Affected flags: 『Y』 Z 『N』 C 『N』 AC 『N』 OV |
| *xor* IO, a | Perform logic XOR on ACC and IO register, then put result into IO register<br>Example: *xor pa, a ;*<br>Result: pa ← a ^ pa ; // pa is the data register of port A<br>Affected flags: 『N』 Z 『N』 C 『N』 AC 『N』 OV |
| *xor* a, M | Perform logic XOR on ACC and memory, then put result into ACC<br>Example: *xor* a, MEM ;<br>Result: a ← a ^ RAM10<br>Affected flags: 『Y』 Z 『N』 C 『N』 AC 『N』 OV |
| *xor* M, a | Perform logic XOR on ACC and memory, then put result into memory<br>Example: *xor* MEM, a ;<br>Result: MEM ← a ^ MEM<br>Affected flags: 『Y』 Z 『N』 C 『N』 AC 『N』 OV |
| *not* a | Perform 1's complement (logical complement) of ACC<br>Example: *not* a ;<br>Result: a ← ~a<br>Affected flags: 『Y』 Z 『N』 C 『N』 AC 『N』 OV<br>Application Example:<br>---<br>*mov* a, 0x38 ; // ACC=0X38<br>*not* a ; // ACC=0XC7<br>--- |
| *not* M | Perform 1's complement (logical complement) of memory<br>Example: *not* MEM ;<br>Result: MEM ← ~MEM<br>Affected flags: 『Y』 Z 『N』 C 『N』 AC 『N』 OV<br><br>Application Example:<br>---<br>*mov* a, 0x38 ;<br>*mov* mem, a ; // mem = 0x38<br>*not* mem ; // mem = 0xC7<br>--- |
| *neg* a | Perform 2's complement of ACC<br>Example: *neg* a;<br>Result: a ← ∓a<br>Affected flags: 『Y』 Z 『N』 C 『N』 AC 『N』 OV<br><br>Application Example:<br>--- |

| | |
|---|---|
| | *mov* a, 0x38 ; // ACC=0X38<br>*neg* a ; // ACC=0XC8<br>---------- |
| *neg M* | Perform 2' s complement of memory<br>Example: neg MEM;<br>Result: MEM ← 〒MEM<br>Affected flags: 『Y』Z 『N』C 『N』AC 『N』OV<br><br>Application Example:<br>----------<br>mov a, 0x38 ;<br>mov mem, a ; // mem = 0x38<br>not mem ; // mem = 0xC8<br>---------- |
| *comp* a, M | Compare ACC with the content of memory<br>Example: *comp a, MEM;*<br>Result: Flag will be changed by regarding as ( a - MEM )<br>Affected flags: 『Y』Z 『Y』C 『Y』AC 『Y』OV<br>Application Example:<br>----------<br>*mov a, 0x38 ;*<br>*mov mem, a ;*<br>*comp a, mem ; // Z flag is set as 1*<br>*mov a, 0x42 ;*<br>*mov mem, a ;*<br>*mov a, 0x38 ;*<br>*comp a, mem ; // C flag is set as 1*<br>---------- |
| *comp* M, a | Compare ACC with the content of memory<br>Example: *comp MEM, a;*<br>Result: Flag will be changed by regarding as ( MEM - a )<br>Affected flags: 『Y』Z 『Y』C 『Y』AC 『Y』OV |

## 7.5 Bit Operation Instructions

| | |
|---|---|
| *set0* IO.n | Set bit n of IO port to low<br>Example: *set0* pa.5 ;<br>Result: set bit 5 of port A to low<br>Affected flags: 『N』Z 『N』C 『N』AC 『N』OV |
| *set1* IO.n | Set bit n of IO port to high<br>Example: *set1* pb.5 ;<br>Result: set bit 5 of port B to high<br>Affected flags: 『N』Z 『N』C 『N』AC 『N』OV |
| *swapc* IO.n | Swap the nth bit of IO port with carry bit<br>Example: swapc IO.0; |

| | |
|---|---|
| | Result: C ← IO.0 , IO.0 ← C<br>When IO.0 is a port to output pin, carry C will be sent to IO.0;<br>When IO.0 is a port from input pin, IO.0 will be sent to carry C;<br>Affected flags: 『N』Z 『Y』C 『N』AC 『N』OV<br>Application Example1 (serial output) :<br>---<br>...<br>set1 pac.0 ; // set PA.0 as output<br>...<br>set0 flag.1 ; // C=0<br>swapc pa.0 ; // move C to PA.0 (bit operation), PA.0=0<br>set1 flag.1 ; // C=1<br>swapc pa.0 ; // move C to PA.0 (bit operation), PA.0=1<br>---<br>Application Example2 (serial input) :<br>---<br>...<br>set0 pac.0 ; // set PA.0 as input<br>...<br>swapc pa.0 ; // read PA.0 to C (bit operation)<br>src a ; // shift C to bit 7 of ACC<br>swapc pa.0 ; // read PA.0 to C (bit operation)<br>src a ; // shift new C to bit 7, old C<br>--- |
| *set0* M.n | Set bit n of memory to low<br>Example: *set0* MEM.5 ;<br>Result: set bit 5 of MEM to low<br>Affected flags: 『N』Z 『N』C 『N』AC 『N』OV |
| *set1* M.n | Set bit n of memory to high<br>Example: *set1* MEM.5 ;<br>Result: set bit 5 of MEM to high<br>Affected flags: 『N』Z 『N』C 『N』AC 『N』OV |

## 7.6 Conditional Operation Instructions

| | |
|---|---|
| *ceqsn* a, I | Compare ACC with immediate data and skip next instruction if both are equal.<br>Flag will be changed like as (a ← a – I)<br>Example: *ceqsn* a, 0x55 ;<br>*inc* MEM ;<br>*goto* error ;<br>Result: If a=0x55, then “goto error” ; otherwise, “inc MEM” .<br>Affected flags: 『Y』Z 『Y』C 『Y』AC 『Y』OV |
| *ceqsn* a, M | Compare ACC with memory and skip next instruction if both are equal.<br>Flag will be changed like as (a ← a - M) |

| | |
|---|---|
| | Example: ceqsn a, MEM;<br>Result: If a=MEM, skip next instruction<br>Affected flags: 『Y』 Z 『Y』 C 『Y』 AC 『Y』 OV |
| *cneqsn* a, M | Compare ACC with memory and skip next instruction if both are not equal.<br>Flag will be changed like as (a ← a - M)<br>Example: cneqsn a, MEM;<br>Result: If a≠MEM, skip next instruction<br>Affected flags: 『Y』 Z 『Y』 C 『Y』 AC 『Y』 OV |
| *cneqsn* a, I | Compare ACC with immediate data and skip next instruction if both are no equal.<br>Flag will be changed like as (a ← a - I)<br>Example: *cneqsn a,0x55 ;*<br>*inc MEM ;*<br>*goto error ;*<br>Result: If a≠0x55, then "goto error" ; Otherwise, "inc MEM" .<br>Affected flags: 『Y』 Z 『Y』 C 『Y』 AC 『Y』 OV |
| *t0sn* IO.n | Check IO bit and skip next instruction if it' s low<br>Example: *t0sn* pa.5;<br>Result: If bit 5 of port A is low, skip next instruction<br>Affected flags: 『N』 Z 『N』 C 『N』 AC 『N』 OV |
| *t1sn* IO.n | Check IO bit and skip next instruction if it' s high<br>Example: *t1sn* pa.5 ;<br>Result: If bit 5 of port A is high, skip next instruction<br>Affected flags: 『N』 Z 『N』 C 『N』 AC 『N』 OV |
| *t0sn* M.n | Check memory bit and skip next instruction if it' s low<br>Example: *t0sn* MEM.5 ;<br>Result: If bit 5 of MEM is low, then skip next instruction<br>Affected flags: 『N』 Z 『N』 C 『N』 AC 『N』 OV |
| *t1sn* M.n | Check memory bit and skip next instruction if it' s high<br>EX: *t1sn* MEM.5 ;<br>Result: If bit 5 of MEM is high, then skip next instruction<br>Affected flags: 『N』 Z 『N』 C 『N』 AC 『N』 OV |
| *izsn* a | Increment ACC and skip next instruction if ACC is zero<br>Example: *izsn* a;<br>Result: a ← a + 1,skip next instruction if a = 0<br>Affected flags: 『Y』 Z 『Y』 C 『Y』 AC 『Y』 OV |
| *dzsn* a | Decrement ACC and skip next instruction if ACC is zero<br>Example: *dzsn* a;<br>Result: A ← A - 1,skip next instruction if a = 0<br>Affected flags: 『Y』 Z 『Y』 C 『Y』 AC 『Y』 OV |
| *izsn* M | Increment memory and skip next instruction if memory is zero<br>Example: *izsn* MEM;<br>Result: MEM ← MEM + 1, skip next instruction if MEM= 0<br>Affected flags: 『Y』 Z 『Y』 C 『Y』 AC 『Y』 OV |
| *dzsn* M | Decrement memory and skip next instruction if memory is zero<br>Example: *dzsn* MEM; |

| | |
|---|---|
| | Result: MEM ← MEM - 1, skip next instruction if MEM = 0<br>Affected flags: 『Y』Z 『Y』C 『Y』AC 『Y』OV |

## 7.7 System control Instructions

| | |
|---|---|
| *call* label | Function call, address can be full range address space<br>Example: *call* function1;<br>Result: [sp] ← pc + 1<br>pc ← function1<br>sp ← sp + 2<br>Affected flags: 『N』Z 『N』C 『N』AC 『N』OV |
| *goto* label | Go to specific address which can be full range address space<br>Example: *goto* error;<br>Result: Go to error and execute program.<br>Affected flags: 『N』Z 『N』C 『N』AC 『N』OV |
| *ret* I | Place immediate data to ACC, then return<br>Example: *ret* 0x55;<br>Result: A ← 55h<br>ret ;<br>Affected flags: 『N』Z 『N』C 『N』AC 『N』OV |
| *ret* | Return to program which had function call<br>Example: *ret;*<br>Result: sp ← sp - 2<br>pc ← [sp]<br>Affected flags: 『N』Z 『N』C 『N』AC 『N』OV |
| *reti* | Return to program from interrupt service routine. After this command is executed, global interrupt is enabled automatically.<br>Example: *reti*;<br>Affected flags: 『N』Z 『N』C 『N』AC 『N』OV |
| *nop* | No operation<br>Example: *nop*;<br>Result: nothing changed<br>Affected flags: 『N』Z 『N』C 『N』AC 『N』OV |
| *pcadd* a | Next program counter is current program counter plus ACC.<br>Example: *pcadd a*;<br>Result: pc ← pc + a<br>Affected flags: 『N』Z 『N』C 『N』AC 『N』OV |
| | Application Example:<br>---<br>…<br>mov a, 0x02 ;<br>pcadd a ; // PC <- PC+2<br>goto err1 ;<br>goto correct ; // jump here<br>goto err2 ; |

| | |
|---|---|
| | goto err3 ;<br>…<br>correct: // jump here<br>…<br>---------- |
| *engint* | Enable global interrupt enable<br>Example: *engint*;<br>Result: Interrupt request can be sent to CPU<br>Affected flags: 『N』Z 『N』C 『N』AC 『N』OV |
| *disgint* | Disable global interrupt enable<br>Example: *disgint* ;<br>Result: Interrupt request is blocked from CPU<br>Affected flags: 『N』Z 『N』C 『N』AC 『N』OV |
| *stopsys* | System halt.<br>Example: *stopsys*;<br>Result: Stop the system clocks and halt the system<br>Affected flags: 『N』Z 『N』C 『N』AC 『N』OV |
| *stopexe* | CPU halt. The oscillator module is still active to output clock, however, system clock is disabled to save power.<br>Example: *stopexe*;<br>Result: Stop the system clocks and keep oscillator modules active.<br>Affected flags: 『N』Z 『N』C 『N』AC 『N』OV |
| *reset* | Reset the whole chip, its operation will be same as hardware reset.<br>Example: *reset*;<br>Result: Reset the whole chip.<br>Affected flags: 『N』Z 『N』C 『N』AC 『N』OV |
| *wdreset* | Reset Watchdog timer.<br>Example: *wdreset* ;<br>Result: Reset Watchdog timer.<br>Affected flags: 『N』Z 『N』C 『N』AC 『N』OV |

## 7.8 Summary of Instructions Execution Cycle

| | | |
|---|---|---|
| 2T | | *goto, call, idxm, pcadd, ret, reti, ldtabl , ldtabh* |
| 2T | Condition is fulfilled | *ceqsn,* cneqsn,*t0sn, t1sn, dzsn, izsn* |
| 1T | Condition is not fulfilled | |
| 1T | | Others |

## 7.9 Summary of affected flags by Instructions

| Instruction | Z | C | AC | OV | Instruction | Z | C | AC | OV | Instruction | Z | C | AC | OV |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *mov* a, I | - | - | - | - | *mov* M, a | - | - | - | - | *mov* a, M | Y | - | - | - |
| *mov* a, IO | Y | - | - | - | *mov* IO, a | - | - | - | - | *ldt16* word | - | - | - | - |
| *stt16* word | - | - | - | - | *idxm* a, index | - | - | - | - | *idxm* index, a | - | - | - | - |
| *xch* M | - | - | - | - | *pushaf* | - | - | - | - | *popaf* | Y | Y | Y | Y |
| *add* a, I | Y | Y | Y | Y | *add* a, M | Y | Y | Y | Y | *add* M, a | Y | Y | Y | Y |
| *addc* a, M | Y | Y | Y | Y | *addc* M, a | Y | Y | Y | Y | *addc* a | Y | Y | Y | Y |
| *addc* M | Y | Y | Y | Y | *nadd a, M* | Y | Y | Y | Y | *nadd M, a* | Y | Y | Y | Y |
| *sub* a, I | Y | Y | Y | Y | *sub* a, M | Y | Y | Y | Y | *sub* M, a | Y | Y | Y | Y |
| *subc* a, M | Y | Y | Y | Y | *subc* M, a | Y | Y | Y | Y | *subc* a | Y | Y | Y | Y |
| *subc* M | Y | Y | Y | Y | *inc* M | Y | Y | Y | Y | *dec* M | Y | Y | Y | Y |
| *clear* M | - | - | - | - | *mul* | - | - | - | - | *sr* a | - | Y | - | - |
| *src* a | - | Y | - | - | *sr* M | - | Y | - | - | *src* M | - | Y | - | - |
| *sl* a | - | Y | - | - | *slc* a | - | Y | - | - | *sl* M | - | Y | - | - |
| *slc* M | - | Y | - | - | *swap* a | - | - | - | - | *and* a, I | Y | - | - | - |
| *and* a, M | Y | - | - | - | *and* M, a | Y | - | - | - | *or* a, I | Y | - | - | - |
| *or* a, M | Y | - | - | - | *or* M, a | Y | - | - | - | *xor* a, I | Y | - | - | - |
| *xor* IO, a | - | - | - | - | *xor* a, M | Y | - | - | - | *xor* M, a | Y | - | - | - |
| *not* a | Y | - | - | - | *not* M | Y | - | - | - | *neg* a | Y | - | - | - |
| *neg* M | Y | - | - | - | *comp a, M* | Y | Y | Y | Y | *comp M, a* | Y | Y | Y | Y |
| *set0* IO.n | - | - | - | - | *set1* IO.n | - | - | - | - | *set0* M.n | - | - | - | - |
| *set1* M.n | - | - | - | - | *swapc IO.n* | - | Y | - | - | *ceqsn* a, I | Y | Y | Y | Y |
| *ceqsn* a, M | Y | Y | Y | Y | *cneqsn a,M* | Y | Y | Y | Y | *cneqsn a, I* | Y | Y | Y | Y |
| *t0sn* IO.n | - | - | - | - | *T1sn* IO.n | - | - | - | - | *t0sn* M.n | - | - | - | - |
| *t1sn* M.n | - | - | - | - | *izsn* a | Y | Y | Y | Y | *dzsn* a | Y | Y | Y | Y |
| *izsn* M | Y | Y | Y | Y | *dzsn* M | Y | Y | Y | Y | *call* label | - | - | - | - |
| *goto* label | - | - | - | - | *ret* I | - | - | - | - | *ret* | - | - | - | - |
| *reti* | - | - | - | - | *nop* | - | - | - | - | *pcadd* a | - | - | - | - |
| *engint* | - | - | - | - | *disgint* | - | - | - | - | *stopsys* | - | - | - | - |
| *stopexe* | - | - | - | - | *reset* | - | - | - | - | *wdreset* | - | - | - | - |
| *ldtabl* index | - | - | - | - | *ldtabh* index | - | - | - | - | *xor* a, IO | Y | - | - | - |
| *swap* M | - | - | - | - | *nmov* M, a | - | - | - | - | *nmov* a, M | Y | - | - | - |

## 7.10 BIT definition

Bit access of RAM is only available for address from 0x00 to 0x7F.

# 8. Code Options

| Option | Selection | Description |
|---|---|---|
| Security | **Enable** | **Security Enable** |
| | Disable | Security Disable |
| LVR | 4.0V | Select LVR = 4.0V |
| | 3.5V | Select LVR = 3.5V |
| | 3.0V | Select LVR = 3.0V |
| | 2.7V | Select LVR = 2.7V |
| | 2.5V | Select LVR = 2.5V |
| | 2.2V | Select LVR = 2.2V |
| | 2.0V | Select LVR = 2.0V |
| | 1.8V | Select LVR = 1.8V |
| LCD2 (please refer to MISC.4) | **Disable** | **VDD/2 LCD bias voltage generator disabled. All are normal IO pins** |
| | PB0_A035 | VDD/2 LCD bias voltage generator enabled, PB0 and PA[0,3,5] are VDD/2 if input mode |
| | PB7_C0~6 | VDD/2 LCD bias voltage generator enabled, PB7 and PC[0~6] are VDD/2 if input mode |
| | PB1256 | VDD/2 LCD bias voltage generator enabled, PB[1,2,5,6] are VDD/2 if input mode |
| Interrupt Src0 | **PA.0** | **INTEN/ INTRQ.Bit0 is from PA.0** |
| | PB.5 | INTEN/ INTRQ.Bit0 is from PB.5 |
| | PA.7 | INTEN/ INTRQ.Bit0 is from PA.7 |
| Interrupt Src1 | **PB.0** | **INTEN/ INTRQ.Bit1 is from PB.0** |
| | PA.3 | INTEN/ INTRQ.Bit1 is from PA.3 |
| | PB.6 | INTEN/ INTRQ.Bit1 is from PB.6 |

| Option | Selection | Description |
|---|---|---|
| IO_Drive | Normal | Group1: PA0/PA3/PB5/PB6/PB7/PC2<br>Group2: PA5/PA7/PC0/PC1/PC3/PC4<br>Group3: PB0/PB1/PB2/PB3/PB4/PC5/PC6 |
| | **Strong** | Group1: PA0/PA3/PB5/PB6/PB7/PC2<br>Group2: PA5/PA7/PC0/PC1/PC3/PC4<br>Group3: PB0/PB1/PB2/PB3/PB4/PC5/PC6 |
| PWM_Source | **16MHZ** | When pwmgclk.0= 1, LPWMG clock source = IHRC = 16MHZ |
| | 32MHZ | When pwmgclk.0= 1, LPWMG clock source = IHRC*2 = 32MHZ |
| TMx_Source | **16MHZ** | **When tm2c[7:4]= 0010, TM2 clock source = IHRC = 16MHZ**<br>**When tm3c[7:4]= 0010, TM3 clock source = IHRC = 16MHZ** |
| | 32MHZ | When tm2c[7:4]= 0010, TM2 clock source = IHRC*2 = 32MHZ<br>When tm3c[7:4]= 0010, TM3 clock source = IHRC*2 = 32MHZ<br>(ICE does NOT Support.) |
| TMx_Bit | **6 Bit** | **When tm2s.7=1, TM2 PWM resolution is 6 Bit**<br>**When tm3s.7=1, TM3 PWM resolution is 6 Bit** |
| | 7 Bit | When tm2s.7=1, TM2 PWM resolution is 7 Bit<br>When tm3s.7=1, TM3 PWM resolution is 7 Bit<br>(ICE does NOT Support.) |
| RMS_Channel | **PA.0** | **RMS Channel is PA.0** |
| | PA.3 | RMS Channel is PA.3 |

Note: The **Bolded** options are the default options.

# 9. Special Notes

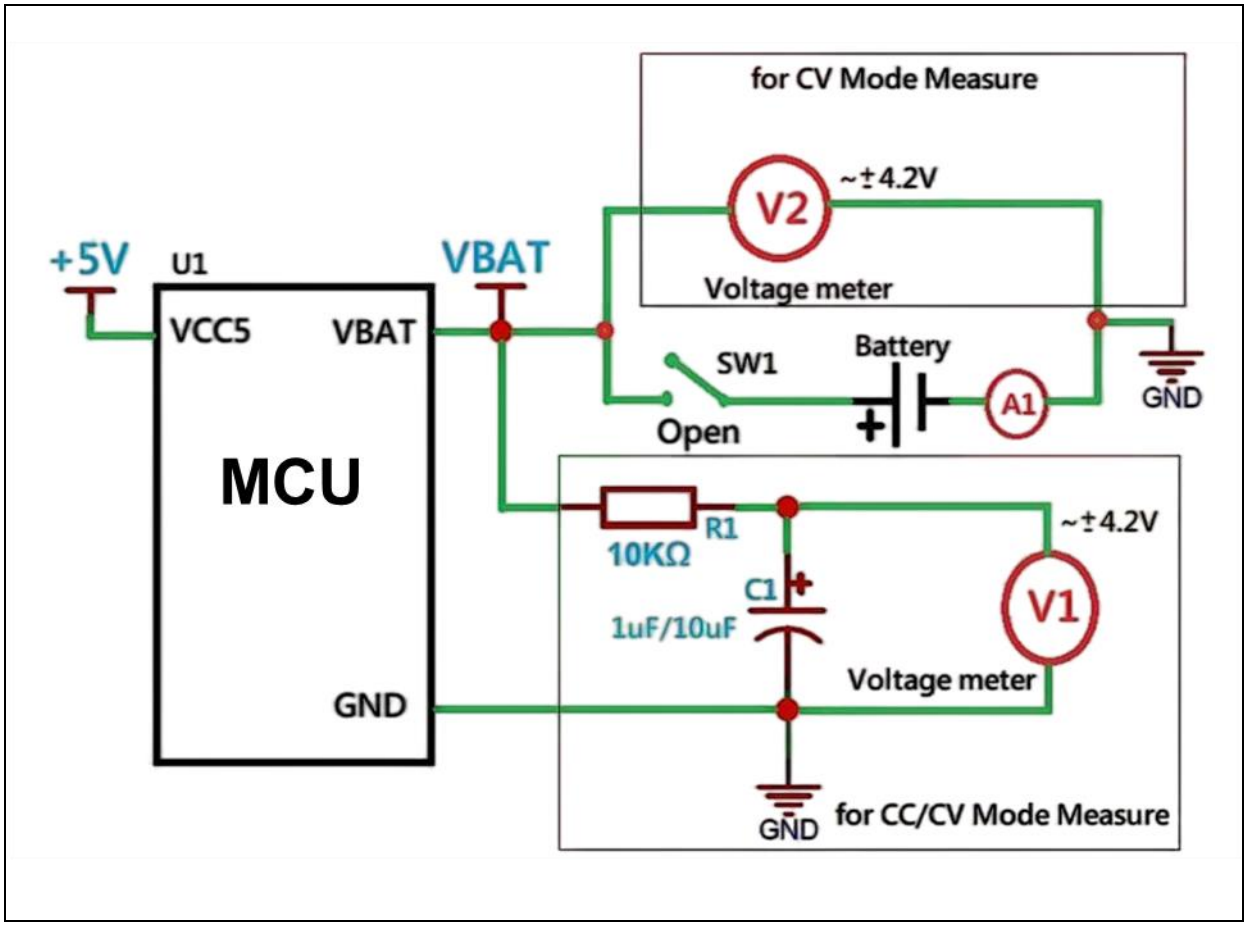
This chapter is to remind user who use PFB190 series IC to avoid frequent errors upon operation.

## 9.1. Using IC

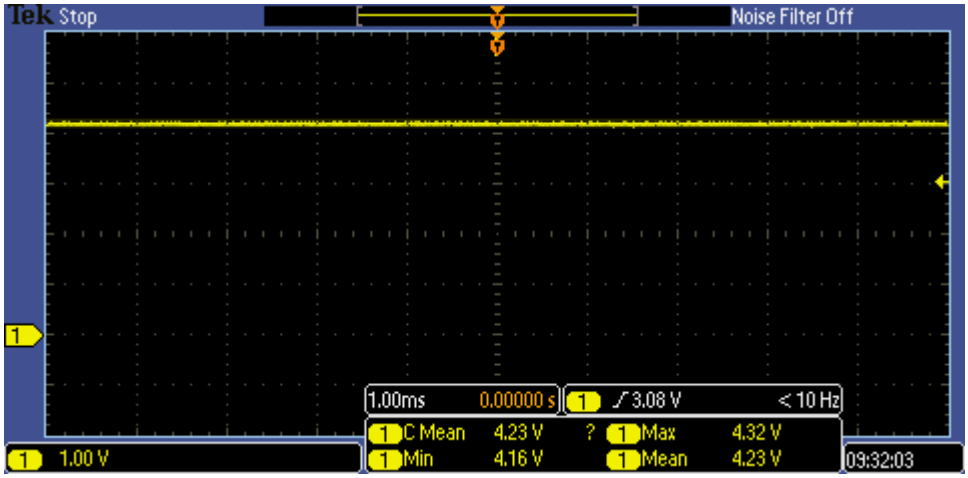
### 9.1.1. Charger Use and Setting

(1) Charging voltage and current measurement

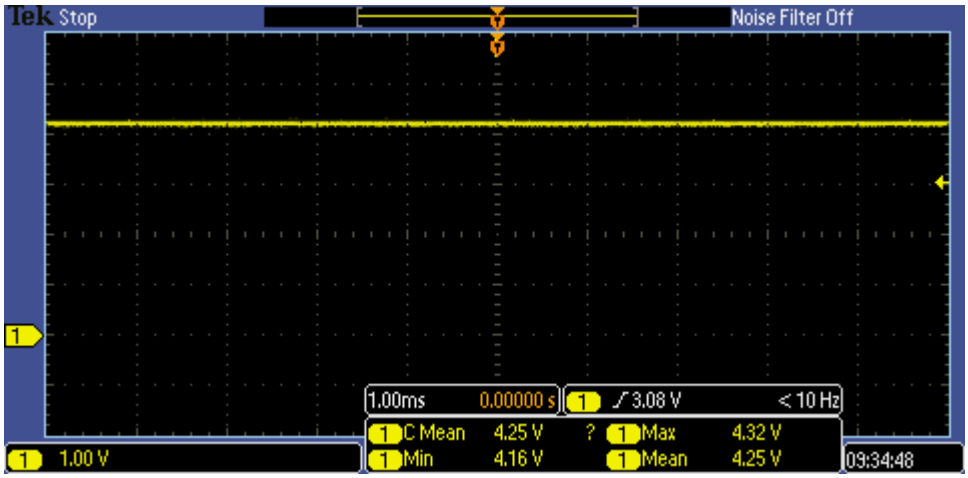
- The PFB190 charger and the MCU execution program work independently of each other, and the MCU reset does not affect the charging action.
- Charging voltage and current measurements on an empty PFB190 charger are the voltage/current values when the charger is not loaded with calibration parameters.
- The PFB190 will write calibration parameters for the charger after the program has started working properly, at which time the PFB190 charger can be electrically measured.
- The wiring diagram for measuring the full voltage of the PFB190 charger is shown as follows: in CC Mode, you need to connect an RC circuit in series at the VBAT pin (to simulate the equivalent circuit of the battery's internal resistance), and the voltmeter V1 is connected in parallel to the capacitor C1, while in CV Mode, you can connect the voltmeter V2 in parallel to the VBAT pin.



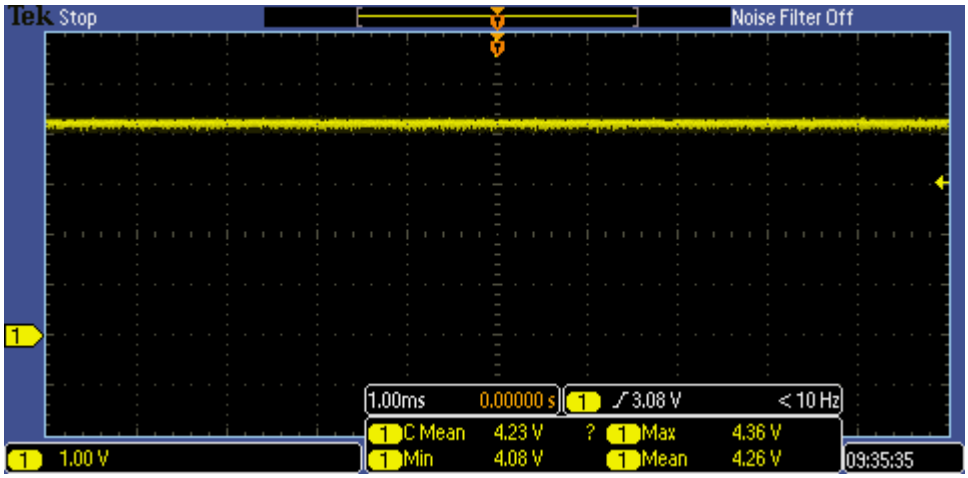
V1 Voltage waveform: (CC Mode, R1 = 10Kohm, C1 = 1uF)




V1 voltage waveform: (CV Mode, R1 = 10Kohm, C1 = 1uF)

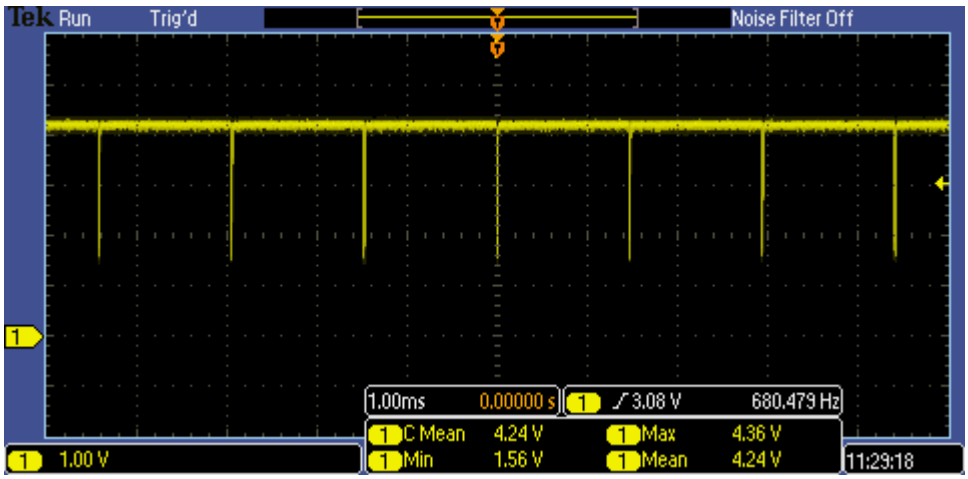



V2 voltage waveform: (CV Mode, No R1 and C1)

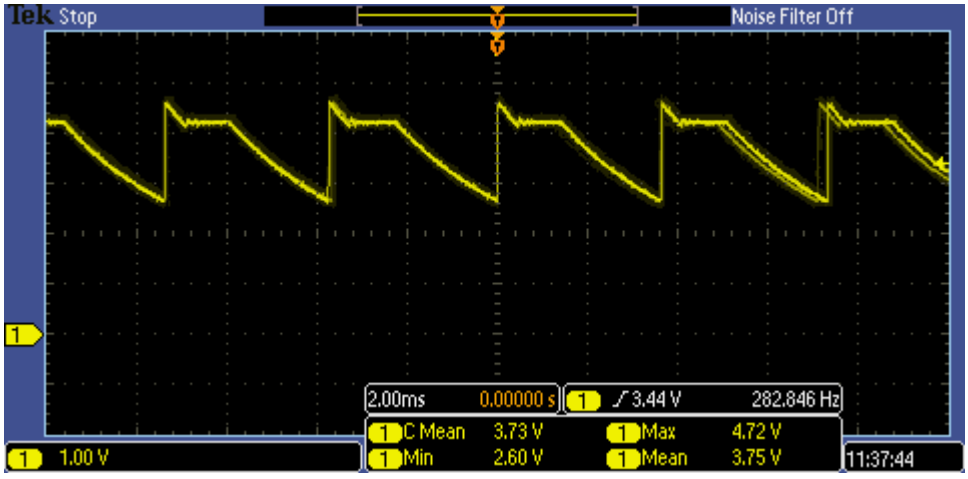


The average voltage measured when the VBAT pin is not connected to the battery or only connected to the filter/electrolytic capacitor in CC mode may be low due to the charging/discharging effect of the charger on the external capacitor. Charge re-cycle time is about 1~3ms when charging to 4.2V, if VBAT pin is not connected to battery or only connected to capacitor, the average voltage measured by voltmeter will be lower due to the discharge time. For example, if a voltmeter is used to measure the VBAT voltage directly, adding a 1uF capacitor to the VBAT pin will result in a measurement error (much lower than the target value) because the discharge time will be longer, resulting in a lower average value measured by the voltmeter. The waveform is shown below:

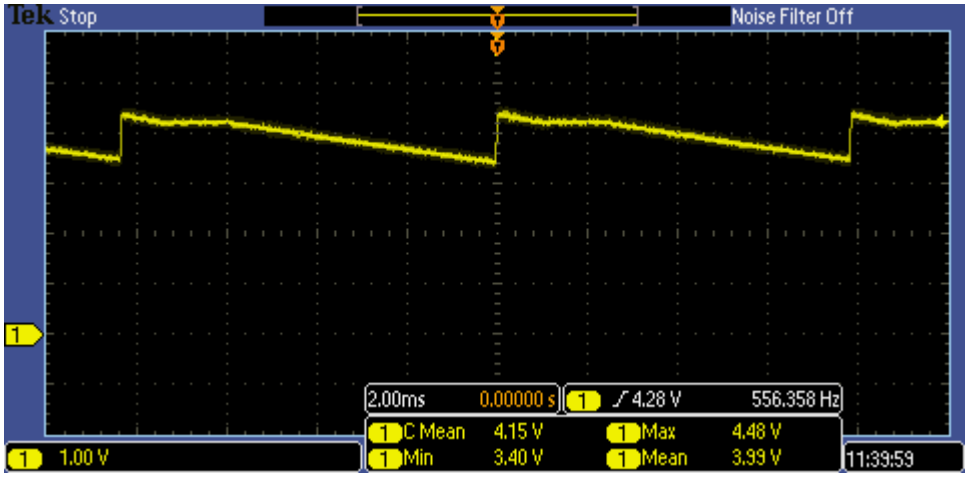
V2 voltage waveform: (CC Mode, No R1 and C1)



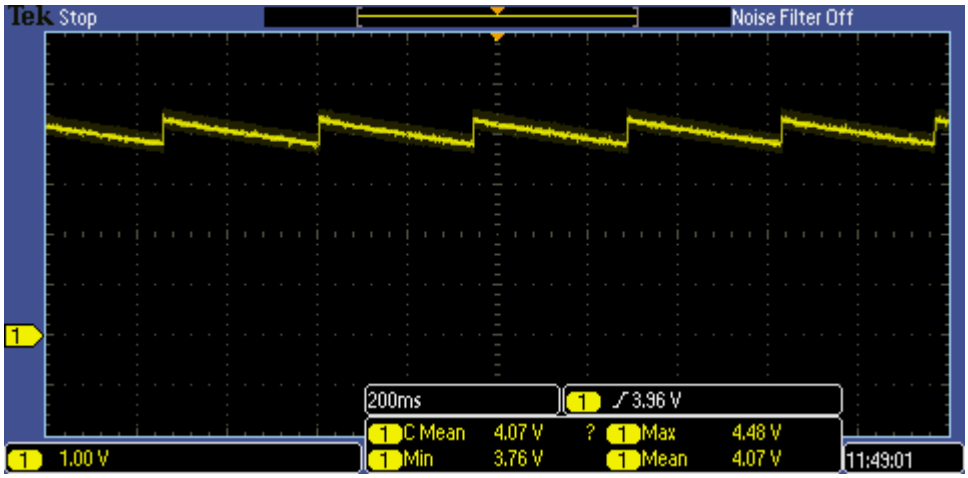

V2 voltage waveform: (CC Mode, R1 = 0R, C1 = 1uF)



V2 voltage waveform: (CC Mode, R1 = 0R, C1 = 4.7uF)




V2 voltage waveform: (CC Mode, R1 = 0R, C1 = 470uF)



### 9.1.2. IO pin usage and setting

(1) IO pin is set to be 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 change with the voltage, temperature and pin voltage. It is not the fixed value.

(2) IO pin as digital input and enable wakeup function

- Configure IO pin as input
- Set corresponding bit to “1” in PXDIER
- For those IO pins of PA that are not used, PADIER[1:2] should be set low 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 >33Ω resistor in between PA5 and the long wire
- Avoid using PA5 as input in such application.

### 9.1.3. Interrupt

(1) 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

*Use DISGINT in the main program to disable all interrupts

*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
```

(2) INTEN and INTRQ have no initial values. Please set required value before enabling interrupt function.

### 9.1.4. System clock switching

System clock can be switched by CLKMD register. **Please notice that, NEVER switch the system clock and turn off the original clock source at the same time.** For example: When switching from clock A to clock B, please switch to clock B first; and after that turn off the clock A oscillator through CLKMD.

◆ Example: Switch system clock from ILRC to IHRC/2

```
CLKMD   =   0x36;        // switch to IHRC, ILRC can not be disabled here
CLKMD.2 =   0;           // ILRC can be disabled at this time
```

◆ **ERROR:** Switch ILRC to IHRC and turn off ILRC simultaneously

```
CLKMD   =   0x50;        // MCU will hang
```

### 9.1.5. Watchdog

Watchdog is open by default, but the program executes ADJUST_IC ten, and the watchdog will be closed. To use the watchdog, you need to reconfigure the open. Watchdog will be inactive once ILRC is disabled.

### 9.1.6. TIMER 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.

### 9.1.7. IHRC

(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.

(3) Normally, the frequency is getting slower a bit. 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.

### 9.1.8. LVR

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 | $V_{BAT}$ | LVR |
|---|---|---|
| 2MHz | ≧ 1.8V | ≧ 2.0V / 1.8V |
| 4MHz | ≧ 2.2V | ≧ 2.5V / 2.2V |
| 8MHz | ≧ 3.5V | ≧ 4.0V/ 3.5V |

Table 8: 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 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.

### 9.1.9. Programming Writing

There are 4 pins for using the writer to program: PA5, PA7, $V_{DD}$ and GND.

Please use 5S-P-003Bx / 5S-P-003C or 5S-P-C01 to program PFB190 real chip.

- Special notes about voltage and current while Multi-Chip-Package (MCP) or On-Board Programming

(1) $V_{BAT}$ may be higher than 9.5V, and its maximum current may reach about 20mA.

(2) All other signal pins level (except GND) are the same as $V_{BAT}$.

User should confirm when using this product in MCP or On-Board Programming, the peripheral components or circuit will not be damaged by the above voltages, and will not clam the above voltages.

**Important Cautions：**

- **You MUST follow the instructions on APN004 and APN011 for programming IC on the handler.**
- **Connecting a 0.01uF capacitor between $V_{BAT}$ 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 may be fail.**

#### 9.1.9.1. Using 5S-P-003Bx/ 5S-P-003C to program PFB190

For 5S-P-003Bx / 5S-P-003C to write PFB190, Use jumper7 to adapt program signal connection. The connection of signal depends on the IC package. Please refer to. Chapter 5 of the Writer user manual to find example and make the jumper-7 adaptive board for target IC package. User can get the user manual from the following linker web page.

http://www.padauk.com.tw/en/technical/index.aspx?kind=27

**Single chip normal programming：**

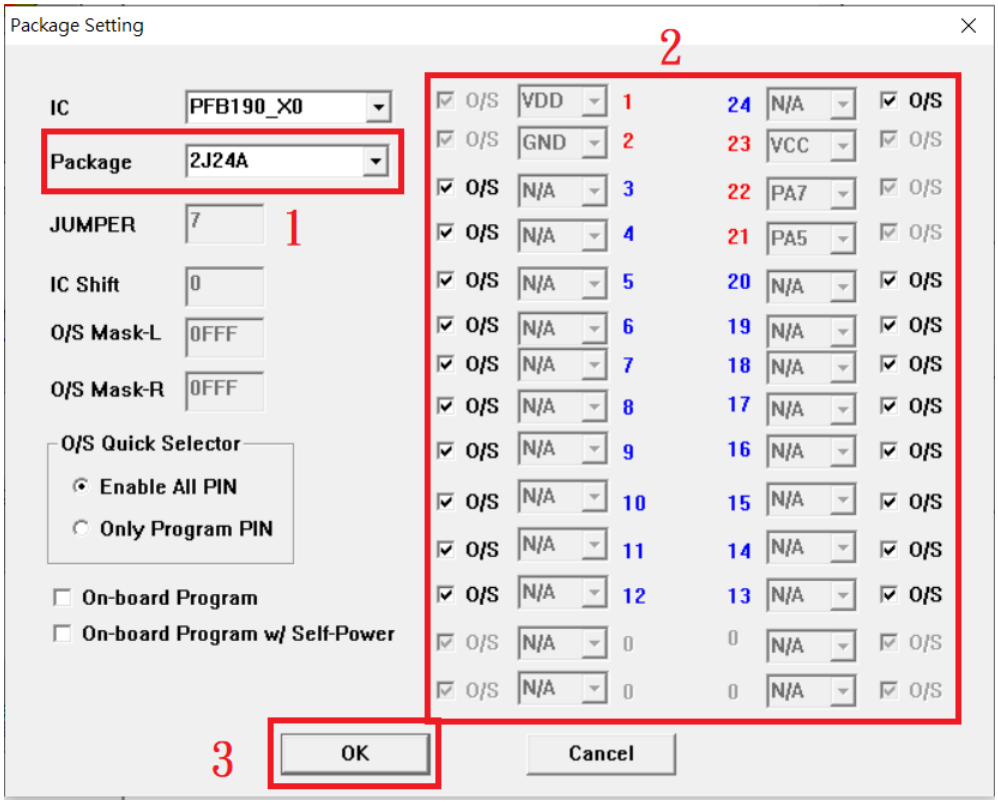
After downloading the PDK file, click <Convert> ➔ <To Package> ➔"Select the PDK to download" ➔"Select Package"➔ After saving the file ➔ click Auto Program

**5-wire on-board programming:**

(1) P-003B2/P-003C writer does not support “on-board self-powered 4-wire programming

(2) The system board is not powered (no lithium battery power)

After downloading the PDK file, click <Convert> ➔ <To Package> ➔ "Select the PDK to download" ➔ "Select Package"➔ **"Check On-Board Program"** ➔ After saving the file ➔ click Auto Program




Load PDK from GUI, insert JP7 and then input IC on the socket without shift. After LCDM displays IC ready, it can be written.

Note: O/S Test for VCC pins is not supported on Writer

In addition, the information of Convert PDK can also be directly defined in the program, as follows

```
//              1          2    3   4   5  6   7   8   9   10    11     12     13     14        15
// S20      pin_cnt       vbat pa0 pa3 pa4 pa5 pa6 pa7 gnd vcc5  agnd   mask1  mask2  shift  option
.writer package 20,        2,   0,  0,  12,  0,  9,  0,  20,  1,  0,     0x102, 0x101,  0,     0x10

// option
//    bit2 - onboard
//    bit4 - vbat/vpp swap
//  others - reserved
```

1. As shown in figure 21, it is the Jumper7 connection method. (using QFN24 as example)

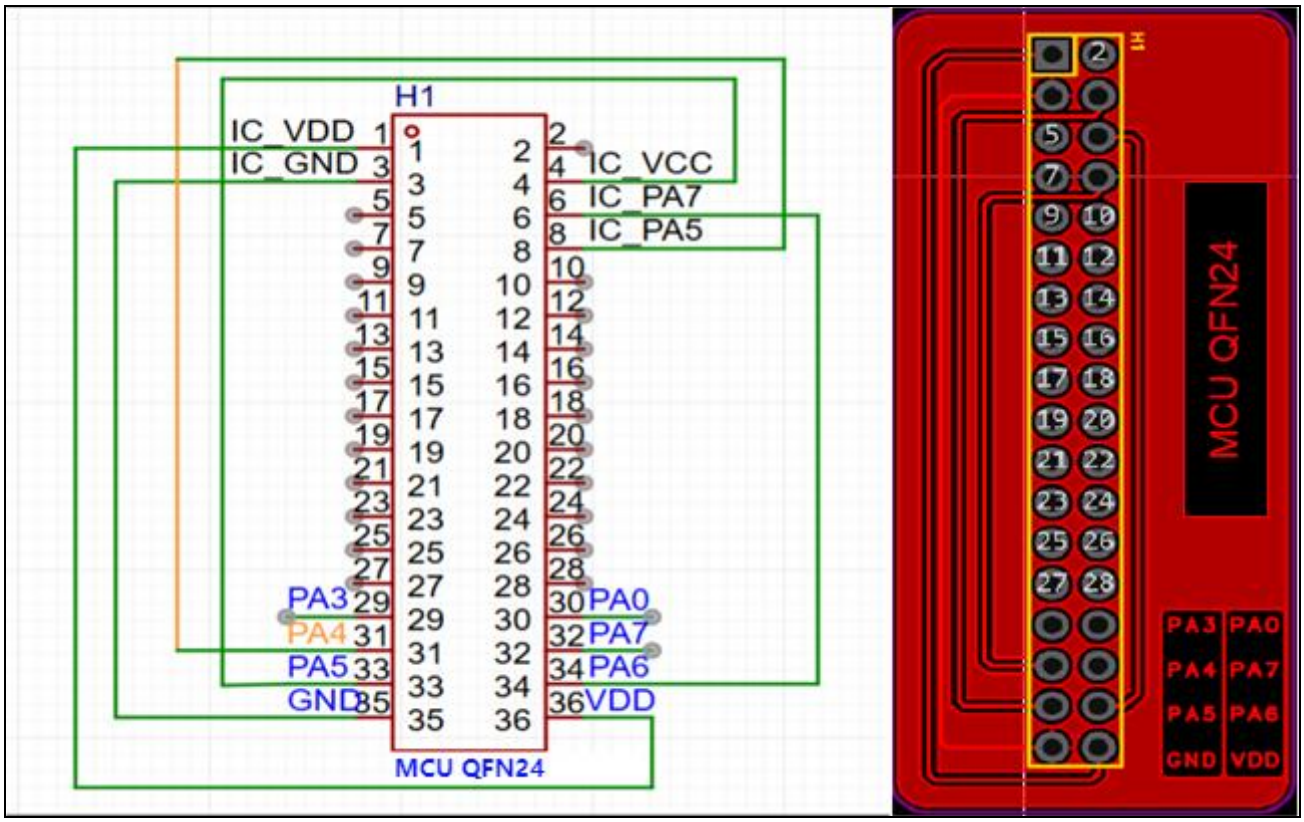
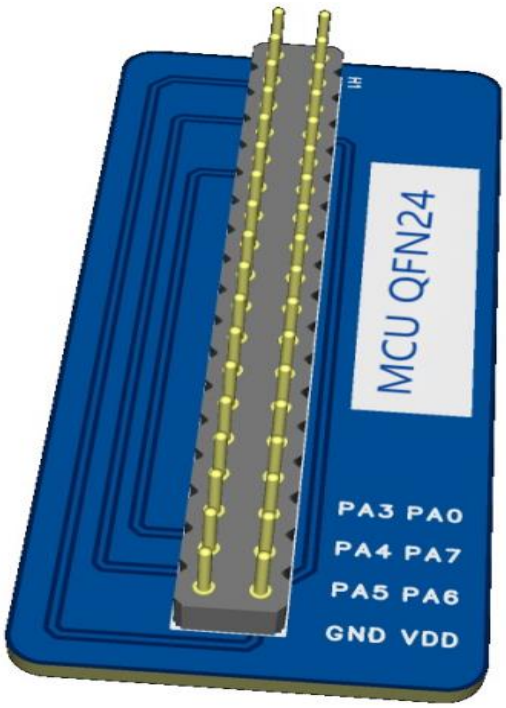



Fig.21: schematic diagram of Jumper7 for P-003B2 / P-003C




**JP7 jumper:**

**WRT_VDD ←→ IC_VDD**

**WRT_GND ←→ IC_GND**

**WRT_PA5 ←→ IC_VCC**

**WRT_PA6 ←→ IC_PA7**

**WRT_PA4 ←→ IC_PA5**

2. Insert JP7 and input IC on the socket without shift. After LCDM displays IC ready, it can be written.

#### 9.1.9.2. Using 5S-P-C01 to program PFB190

(1) Only supports on-board self-powered 4-wire programming, PCBA must be powered by a lithium Battery.

(2) The 5S-P-C01 programmer only supports PDK file downloads with OBP settings checked.

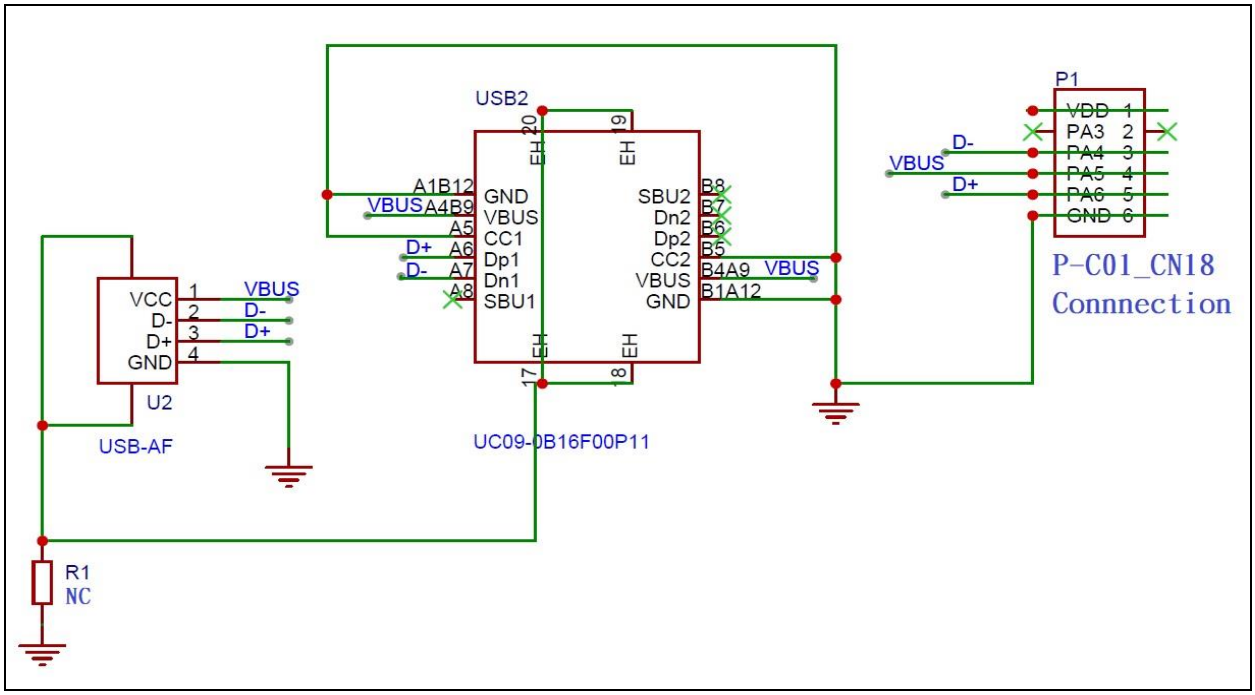
(3) 5S-P-C01 programmer supports stand-alone offline on-board programming of PFB190.

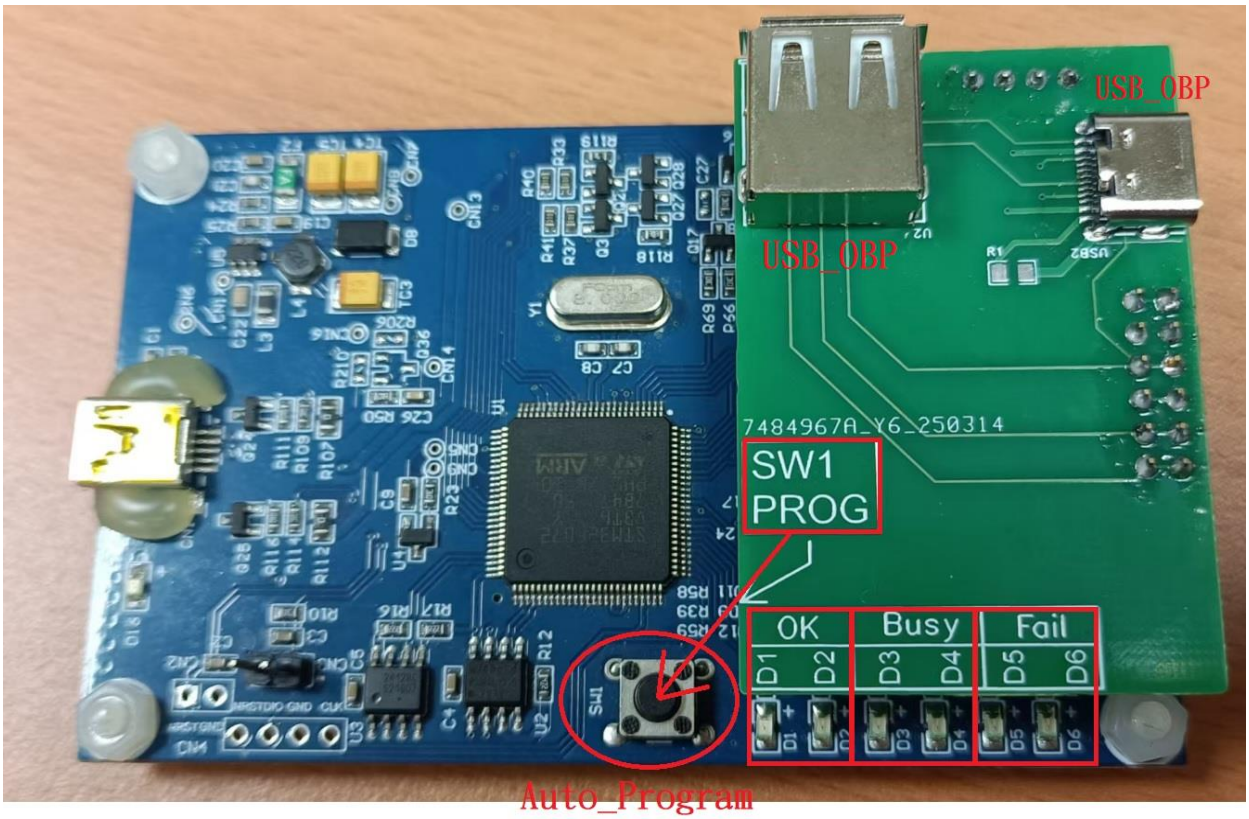
(4) When 5S-P-C01 is used for stand-alone offline programming, SW1 is the Auto Program button

| | D1 / D2 | D3 / D4 | D5 / D6 |
|---|---|---|---|
| **IC_Remove** | **On** | **On** | **On** |
| **IC_Ready** | **Off** | **Off** | **Off** |
| **Program OK** | **On** | **Off** | **Off** |
| **Program Fail** | **Off** | **Off** | **On** |
| **Programming Busy** | **Off** | **On ⇔ Off Flash** | **Off** |

Table 9: 5S-P-C01 LED Status Table

After downloading the PDK file, click <Convert> ➔ <To Package> ➔ "Select the PDK to download" ➔"Select Package"➔ **"Check 'On-Board Program w/ Self-Power"** ➔ "After saving the file ➔ click Auto Program "


USB2
A1B12
VBUS A4B9
A5
D+ A6
D- A7
A8
GND
VBUS
CC1
Dp1
Dn1
SBU1
SBU2
Dn2
Dp2
CC2
VBUS
GND
B8
B7
B6
B5
B4A9 VBUS
B1A12
UC09-0B16F00P11
VCC 1
D- 2
D+ 3
GND 4
U2
USB-AF
VBUS
D-
D+
R1
NC
P1
VDD 1
PA3 2
PA4 3
PA5 4
PA6 5
GND 6
VBUS
D-
D+
P-C01_CN18
Connnection


USB_OBP
USB_OBP
7484967A_Y6_250314
SW1
PROG
OK
Busy
Fail
D1
D2
D3
D4
D5
D6
Auto_Program

Program wiring:

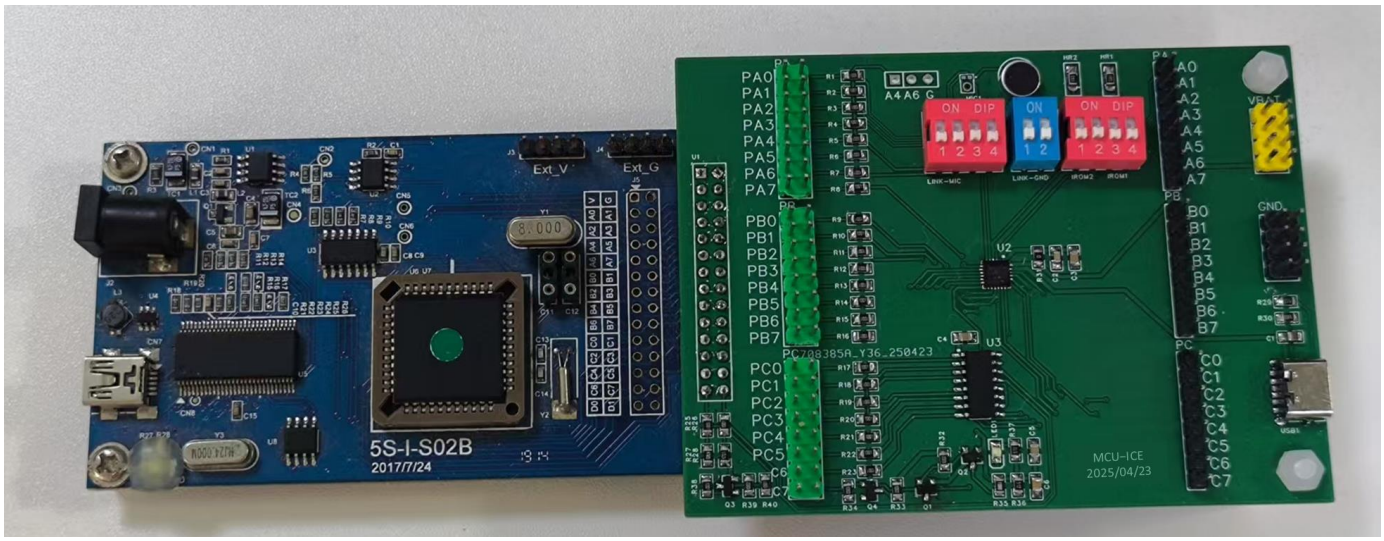
| P-C01 | | PFB190 | | Battery |
|---|---|---|---|---|
| | | IC_VDD | ←→ | lithium Battery (+) |
| WRT_GND | ←→ | IC_GND | ←→ | lithium Battery (-) |
| WRT_PA5 | ←→ | IC_VCC | | |
| WRT_PA6 | ←→ | IC_PA7 | | |
| WRT_PA4 | ←→ | IC_PA5 | | |

#### 9.1.10. Application Manual

For more precautions on the use of the charger, refer to the APN-021 documentation instructions

## 9.2. Using ICE

5S-I-CEB001 is a simulation tool launched by PADAUK Technology. It needs to be used with PADAUK Technology's IDE software for online simulation. When using 5S-I-CEB001, it needs to be used with 5S-I-S01/2 (B). Refer to the following pictures:




5S-I-CEB001Simulation Notes:

1. 5S-I-CEB001 must be matched with 5S-I-S01/2 (B) series emulators to simulate functions such as PXPL/ LPWM/ LVDC/ TIMER2/ GPC/ CHARGE
2. 5S-I-CEB001 power supply by PDK5S-I-S01/2 (B) In order to stabilize the power supply, please connect 5S-I-S01/S02 (B) to the DC9V power adapter.
3. IDE version 1.04D3 starts supporting emulation of 5S-I-CEB001.
4. When emulating PxPL/ LPWM/ LVDC/ TIMER2/ GPC/ CHARGE related functions, 5S-I-S01/2(B) will communicate with the emulation board 5S-I-CEB001, so the emulation timing is slow with the actual IC, and

the main affected registers are intrq/ PxPL/ LVDC/ CHGC/ CHGS/ LPWMGCLK/ LPWMGCUBH/ LPWMGCUBL/ LPWMGxC/ LPWMGxDTH/ LPWMGxDTL/ GPCC/ GPCS/ LPWMGxDTL/ GPCC/ GPCS LPWMGCLK/ LPWMGCUBH/ LPWMGCUBL/ LPWMGxC/ LPWMGxDTH/ LPWMGxDTL/ GPCC/ GPCS/ TM2CT/ TM2B/ TM2S/ TM2C.

5. During simulation, the communication between ICE and 5S-I-CEB001 will delay the trigger of the interrupt, and the delay time is about 0~ 335us.
6. Using T16 interrupt timing, interrupt timing error, 5S-I-S01/2(B) and 5S-I-CEB001 communication will switch the system clock, and will make the interrupt delay trigger, this phenomenon often occurs in the timer clock source selection SYSCLK or when the timing time is shorter. Therefore, it is recommended to simulate a single timing interrupt period ≥ 10ms.
7. Emulation does not support the comparator wakeup
8. Emulation does not support GPC control PWM signal.
9. When emulating GPC/Timer2/LPWM, the interrupt does not support hardware trigger, and the interrupt trigger signal can only be obtained by polling.
10. Since the IC program on 5S-I-CEB001 is 8M, when simulating Timer2 or LPWM, the system clock is selected as SYSCLK, and the module is running at 8M, i.e., the system clock is forced to 8M, but the actual IC is the normal system clock.
11. Timer2 or LPWM emulation does not support 32M clock, but the actual IC is OK.
12. Timer2 emulation does not support 7BIT resolution.
13. Emulation of PFB190, the operation of the PC IO port should be noted, PC7 in the battery to ensure that the output is high, not connected to the battery to ensure that the output is low
14. When not connected to the battery, 5S-I-S01/2(B) than 5S-I-CEB001 voltage to be higher than 0.3V-0.4V (5S-I-CEB001 diode on the cause)

| WDT period | 5S-I-S01/2(B) | PFB190 |
|---|---|---|
| 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 * $T_{ILRC}$ |

## 9.3. Typical Application

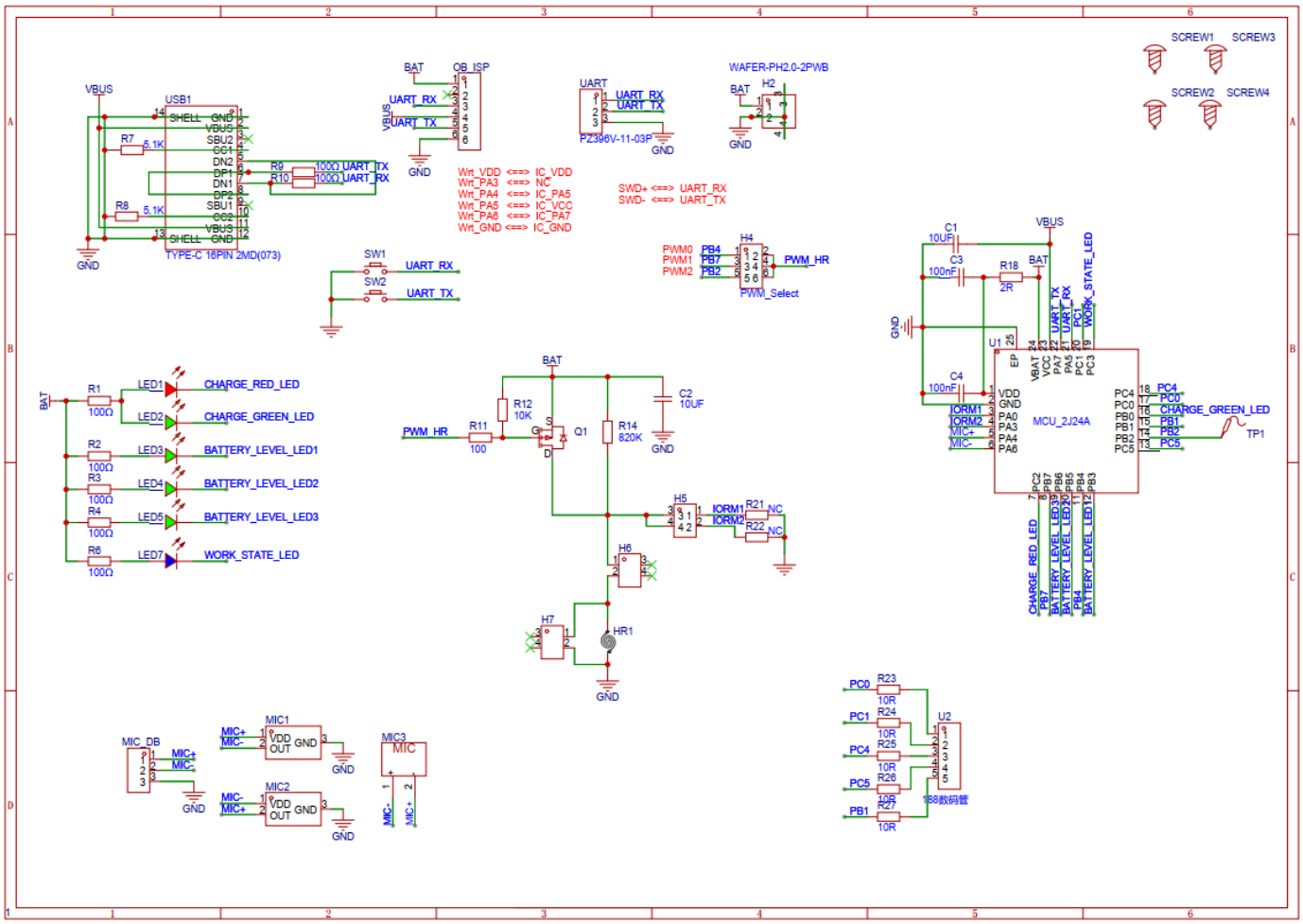



Fig.22: DemoBoard Schematic Diagram

MIC (+) pin connected to **PA4** and MIC (-) pin connected to **PA6**

## 9.4. Program Countermeasures for Lithium Ion Battery Power-Up and Jitter Interference

During the manufacturing process of PFB190 application boards, there is often a need to connect the Li-ion battery electrode pads with the application boards through spot welding process. This will cause the power supply on the application board to be unstable, resulting in jittery interference in the PFB190 power-up waveform. This process may last for tens to hundreds of milliseconds, which is a harsh challenge for the chip to power on and off, and may cause the chip to crash, the program to run away, abnormal functions, and system parameters to be messed up...and so on. To address this situation, the following settings can be used in the program to enhance the stability of the PFB190 IC during power-on and power jitter.

**(1) CodeOption setting:**

- CodeOption Boot speed selection Slow. (If the chip has support for this function)

**(2) The power-up program prioritizes the IO output low and delay settings before .Adjust_IC:**

- The power-on startup program executes IO output low with ILRC priority (before .Adjust_IC).
- After IO output low, execute delay with ILRC system frequency (before .Adjust_IC).
- Add x_first to the parameters of .Adjust_IC.
- It is recommended to power on with a delay of 100ms ~ 200ms before entering Stopexe Mode.

**(3) System frequency:**

- It is recommended to keep the system frequency for ILRC operation after the IC is powered on and turned on until it enters the Stopexe / Stopsys mode.
- It is recommended to switch the system frequency to high frequency operation only after Stopexe / Stopsys wake up.
- It is recommended to set the system clock frequency below 1MHz (inclusive) to improve stability.
- It is recommended that the ILRC can always be enabled, so that it can be switched directly when switching from high frequency to low frequency.
- When switching the system main frequency from ILRC to IHRC, it is necessary to enable IHRC first and wait for the running time of two or more instructions before switching. Avoid doing Enable IHRC and switching frequency in one line of instruction.

```
//-------------------------------------------------------
// Macro Name: PowerOn_Delay
// Parameter: none
//
//-------------------------------------------------------
byte pndt;
PowerOn_Delay  macro          // delay 2048 ILRC period = 20.48ms
    PA = 0x00;
    PAC = 0xFF;               // change to Output Mode
    pndt = 0xFF;
    do
     {      nop; nop; nop; nop; nop;   }
    while(pndt--);
    PAC = 0x00;               // change to Output Mode
```

```
    endm

 void    FPPA0 (void)
{
 #if     _SYS(AT_EV)
   $ MISC WDT_64K;                    // 64K*ILRC = 640ms
 #endif
   PowerOn_Delay   //delay 20ms
   .ADJUST_IC      SYSCLK=ILRC (IHRC/16), IHRC=16MHz, VDD=4.2V, O_WDRST, X_FIRST;
   .wdreset;
   .delay 7000                        // delay 70ms for VDD = 5V
 //---- ReLoad_All_Param
   ReLoad_IHRC                        //Reload IHRCR Parameter
   ReLoad_ChargerCURTRIM              //Reload Charger Current Trim Bits
   ReLoad_VbatBGTRIM                  //Reload Charger Vbat Trim Bits
  $ MISC WDT_64K, Fast_Wake_Up;                  // 64K*ILRC = 640ms
   .wdreset;
   $ CLKMD IHRC/16, En_IHRC, En_ILRC, En_WatchDog;

   while(1)
   {
     .delay 10000;
     $ PA.6 Out, High;
     .delay 10000;
     $ PA.6 Out, Low;
   }
}
```

(4) **Watchdog**:

- Keeps the watchdog as Enable and does not turn off the watchdog. .Adjust_IC's parameter plus **O_WDTRST**.
- The watchdog does not need to be turned off when entering Sleep Stopexe Mode, the chip hardware will automatically turn off the watchdog and re-enable the watchdog after wakeup, and the watchdog counter will be cleared as well.
- It is recommended that the execution cycle of the watchdog clear instruction is not too intensive. It is recommended to execute it once in the main program.
- If interrupts are used, it is not recommended to add the watchdog clear instruction in the interrupt program.
- When switching the system frequency, it is necessary to pay attention to keep the watchdog on to avoid shutting down the watchdog by mistake.
- It is recommended to set the watchdog time to 64K ILRC, which is about 640ms. considering the frequency drift error of ILRC, it is recommended to clear the watchdog period with a time error of 320ms ~ 1280ms.

**(5) Stopexe/Stopsys Wakeup**

- It is recommended that the **ReLoad_IHRC / ReLoad_ChargerCURTRIM / ReLoad_VbatBGTRIM** macros can be executed after stopexe/Stopsys wake up to rewrite the system calibration parameter registers once again.