# Panchip Microelectronics Co., Ltd.

**PAN2416AV**

**Datasheet**

**2.4GHz Wireless Transceiver**

Version:        1.3

Release date:    May. 2025

## Shanghai Panchip Microelectronics Co., Ltd.

Address: Room 302 of Building D, No. 666 Shengxia Road

Zhangjiang Hi-Tech Park, Shanghai

People's Republic of China

Tel:        021-50802371

Website: http://www.panchip.com

## USING THIS DOCUMENT

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

## TRADEMARKS

Other names mentioned in this document are trademarks/registered trademarks of their respective owners.

## DISCLAIMER

All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

## REVISION HISTORY

| Version | Date | Description |
|---------|------|-------------|
| V1.0 | Sep 2017 | Initial |
| V1.1 | Dec 2022 | Add PAN2416CL |
| V1.2 | Mar. 2023 | 1) Update Table 4-2 Current Consumption in Sleep Mode Maximum<br>2) Update typical and maximum Quiescent Current values in the 22.1 MCU DC Characteristics |
| V1.3 | May. 2025 | Update the Ordering information. |

# Table of Contents

# List of Figures

# List of Tables

## Abbreviation

| | |
|---|---|
| ADC | Analog-to-Digital Converter |
| CPU | Central Processing Unit |
| GFSK | Gauss Frequency Shift Keying |
| ISM | Industrial Scientific Medical |
| LVR | Low-Voltage Release |
| MCU | Microcontroller Unit |
| OTP | One Time Programmable |
| PWM | Pulse Width Modulation |
| RAM | Random Access Memory |
| RF | Radio Frequency |
| RTC | Real-Time Clock |
| SP | Stack Pointer |
| SPI | Serial Peripheral Interface |
| WDT | Watchdog Timer |

# Ordering information

| Part number | Chip type | Package | Pin count | IO | OTP | RAM | Temperature | Packing |
|---|---|---|---|---|---|---|---|---|
| **PAN2416AV** | 2.4G | SOP | 16 | 10 | 4K×16Bit | 176×8Bit | -40~85℃ | Tube |

# 1 General Description

The PAN2416AV is a monolithic wireless transceiver chip working in the 2.400 ~ 2.483GHz world's universal ISM band. The chip integrates RF transceiver, frequency generator, crystal oscillator, modem, low-power MCU and other functional modules. Moreover, it supports one-to-multi networking and communication mode with ACK.

The user sends an instruction to the chip through the I/O port of the MCU, and the chip automatically completes the sending and receiving configuration to communicate, and automatically determines whether the data sending / receiving is successful according to the response information so as to perform resending, packet loss, resume sending and waiting, which simplifies the user program. Transmit output power, working channel and communication data rate can be configured.

The PAN2416AV requires a small number of peripheral devices to support single/double layer PCB designs.

## 1.1 Key Features

Featrues of the PAN2416AV include:

- MCU
    - 8-bit microcontroller core
    - OTP: 4K×16Bit
    - Universal RAM: 176×8Bit
- Peripherals
    - GPIO
    - WDT
    - PWM
    - 12-bit ADC
    - Timer
    - Including a crystal and a small amount of capacitance
    - Support double or single layer PCB design, you can use the PCB microstrip antenna or wire antenna
    - The chip comes with a part of the link layer communication protocol, configuration of a small amount of parameter registers, easy to be used
- RF
    - Radio
        - Frequency band: 2.400GHz ~ 2.483GHz
        - Data rate: 2Mbps, 1Mbps, 250Kbps

- GFSK modulation
- Sleep current 2uA

- Receiver
  - -83dBm@2Mbps
  - -87dBm@1Mbps
  - -91dBm@250Kbps
- Transmitter
  - Output power: 8dBm
  - 19mA@2dBm
- RF Synthesizer
  - Fully integrated synthesizer
  - Accept low cost ±60ppm 16MHz crystal for the data rate of 1Mbps and 2Mbps
  - Accept low cost ±20ppm 16MHz crystal for the data rate of 250Kbps
- Protocol Engine
  - Support 1 to 32 or 64 byte payload length
  - Support automatic reply and automatic retransmission
  - 6 data pipes receiver for 1:6 star networks

- Power Manage
  - Integrated voltage regulator
  - 2.2V to 3.3V supply range
- Package
  - SOP16 package
- Operating Condition
  - Operating temperature range: -40~85℃

## 1.2 Typical Applications

- wireless mouse
- Wireless gamepad
- Common remote control

- TV and set-top box remote control
- Remote control toys
- Smart home

# 2 Block Diagram
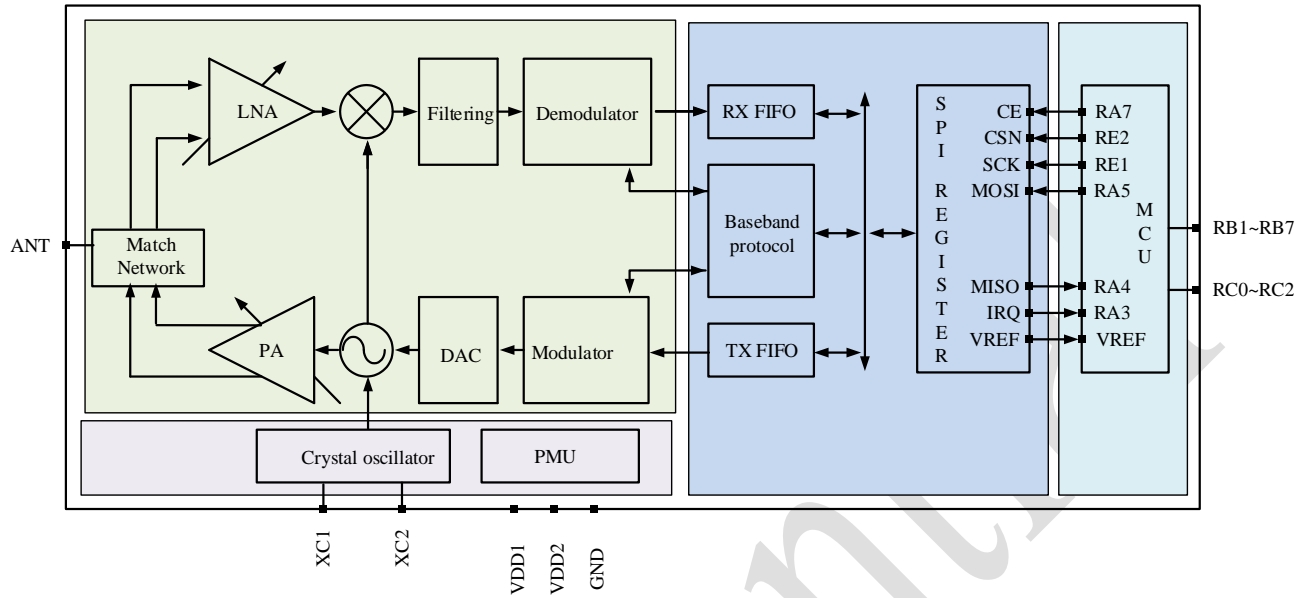


Figure 2-1 Block Diagram

# 3 PIN information

## 3.1 Pin Assignment
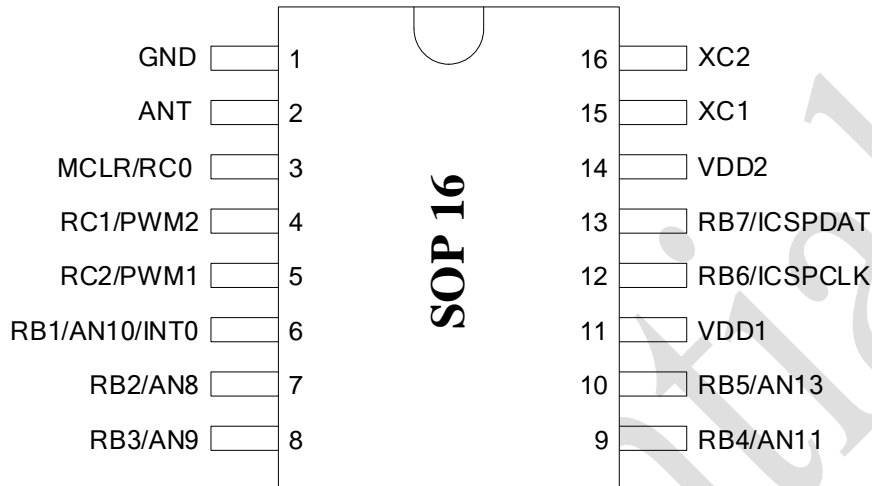


Figure 3-1 PIN Diagram

## 3.2 Pin Descriptions

Table 3-1 Pin descriptions

| SOP16 | Name | IO | Function |
|---|---|---|---|
| 1 | GND | P | Ground (GND) |
| 2 | ANT | I/O | Antenna interface |
| 3 | RC0 | I/O | Programmable as input pin, open drain output pin |
| | MCLR | I | Write high-voltage input pin |
| 4 | RC1 | I/O | Programmable as input pin, with pull-up resistor input pin, push-pull output pin |
| | PWM2 | O | PWM output |
| 5 | RC2 | I/O | Programmable as input pin, with pull-up resistor input pin, push-pull output pin |
| | PWM1 | O | PWM output |
| 6 | RB1 | I/O | Programmable as input pin, with pull-up resistor input pin, push-pull output pin |
| | AN10 | I | 12-bit ADC input |
| | INT0 | O | External interrupt input |
| 7 | RB2 | I/O | Programmable as input pin, with pull-up resistor input pin, push-pull output pin |
| | AN8 | I | 12-bit ADC input |
| 8 | RB3 | I/O | Programmable as input pin, with pull-up resistor input pin, push-pull output pin |
| | AN9 | I | 12-bit ADC input |
| 9 | RB4 | I/O | Programmable as input pin, with pull-up resistor input pin, push-pull output pin |

| | | | |
|---|---|---|---|
| | AN11 | I | 12-bit ADC input |
| 10 | RB5 | I/O | Programmable as input pin, with pull-up resistor input pin, push-pull output pin |
| | AN13 | I | 12-bit ADC input |
| 11 | VDD1 | P | Power supply voltage input pin |
| 12 | RB6 | I/O | Programmable as input pin, with pull-up resistor input pin, push-pull output pin |
| | ICSPCLK | I | Programming clock pin |
| 13 | RB7 | I/O | Programmable as input pin, with pull-up resistor input pin, push-pull output pin |
| | ICSPDAT | I/O | Programming data pin |
| 14 | VDD2 | P | Power supply voltage input pin |
| 15 | XC1 | I | Crystal input |
| 16 | XC2 | O | Crystal output |

## 3.3 Internal Connection

Table 3-2 RF and MCU Connection Pin Description

| Interface function | RF interface name of the SPI | Direction | SPI interface name of MCU |
|---|---|---|---|
| Mode chip select signal | CE | → | RA7 |
| SPI chip select signal | CSN | → | RE2 |
| SPI clock signal | SCK | → | RE1 |
| SPI Data Host out slave in | MOSI | → | RA5 |
| SPI Data Host in slave out | MISO | ← | RA4 |
| Interrupt signal | IRQ | ← | RA3 |
| The reference voltage | VREF | → | VREF |

# 4 Electrical Specification

## 4.1 Absolute Maximum Ratings

Table 4-1 Absolute Maximum Ratings

| Symbol | Parameter Condition (VCC = 3V±5%, TA=25℃) | Parameter value | | | Units |
|---|---|---|---|---|---|
| | | Min | Type | Max | |
| $V_{DD}$ | Supply voltage | -0.3 | - | 3.6 | V |
| $V_I$ | Input voltage | -0.3 | - | 3.6 | V |
| $V_O$ | Output voltage | VSS | - | VDD | - |
| $Pd$ | Total Power Dissipation (TA=-40℃~85℃) | - | - | 300 | mW |
| $T_{OP}$ | Operating Temperature | -40 | - | 85 | ℃ |
| $T_{STG}$ | Storage Temperature | -40 | - | 125 | ℃ |

Note1: More than one or more limit maximum rating in use may cause permanent damage to the device.

Note2: Electrostatic sensitive devices follow the rules of protection when operating.

## 4.2 Current Consumption

Table 4-2 Current Consumption

| Symbol | Parameter Condition (VCC=3V±5%, TA=25℃) | Parameter value | | | Unit |
|---|---|---|---|---|---|
| | | Min | Type | Max | |
| $ICC$ | Sleep | - | 2 | 5 | uA |
| | Standby I | - | 30 | - | uA |
| | Standby III | - | 650 | - | uA |
| | Standby II | - | 780 | - | uA |
| | TX at -35dBm output power | - | 9 | - | mA |
| | TX at -20dBm output power | - | 9.5 | - | mA |
| | TX at 0dBm output power | - | 16 | - | mA |
| | TX at 2dBm output power | - | 19 | - | mA |
| | TX at 8dBm output power | - | 30 | - | mA |
| | TX at 13dBm output power | - | 66 | - | mA |
| | RX at 2Mbps | - | 16.5 | - | mA |

| | RX at 1Mbps | - | 15.5 | - | mA |
|---|---|---|---|---|---|
| | RX at 250kbps | - | 15 | - | mA |

## 4.3 General RF Conditions

Table 4-3 General RF Conditions

| Symbol | Parameter Condition (VCC = 3V±5%, TA=25℃) | Parameter value | | | Unit |
|---|---|---|---|---|---|
| | | Min | Type | Max | |
| $f_{OP}$ | Operating frequency | 2400 | - | 2483 | MHz |
| $PLL_{res}$ | PLL Programming resolution | - | 1 | - | MHz |
| $f_{XTAL}$ | Crystal frequency | - | 16 | - | MHz |
| DR | Data rate | 0.25 | - | 2 | Mbps |
| $\Delta f_{250K}$ | Frequency deviation at 250kbps | - | 125 | 150 | KHz |
| $\Delta f_{1M}$ | Frequency deviation at 1Mbps | - | 160 | 300 | KHz |
| $\Delta f_{2M}$ | Frequency deviation at 2Mbps | - | 320 | 550 | KHz |
| $FCH_{250K}$ | Channel spacing at 250Kbps | - | 1 | - | MHz |
| $FCH_{1M}$ | Channel spacing at 1Mbps | - | 1 | - | MHz |
| $FCH_{2M}$ | Channel spacing at 2Mbps | - | 2 | - | MHz |

## 4.4 Transmitter Operation

Table 4-4 Transmitter Operation

| Symbol | Parameter Condition (VCC = 3V±5%, TA=25℃) | Parameter value | | | Unit |
|---|---|---|---|---|---|
| | | Min | Type | Max | |
| $PRF$ | Typical output power | 2 | 8 | 8 | dBm |
| $PRFC$ | Output Power Range | -35 | - | 8 | dBm |
| $PBW1$ | 20dB Bandwidth for Modulated Carrier at 250Kbps | - | 500 | - | KHz |
| $PBW2$ | 20dB Bandwidth for Modulated Carrier at 1Mbps | - | 1 | - | MHz |
| $PBW3$ | 20dB Bandwidth for Modulated Carrier at 2Mbps | - | 2 | - | MHz |

## 4.5 Receiver Operation

Table 4-5 Receiver Operation

| Symbol | Parameter Condition (VCC = 3V±5%, TA=25℃) | Parameter value | | | Unit |
|---|---|---|---|---|---|
| | | Min | Type | Max | |
| $RX_{max}$ | Maximum received signal at <0.1% BER | - | 0 | - | dBm |
| $RXSENS3$ | Sensitivity (0.1%BER) @250Kbps | - | -91 | - | dBm |
| $RXSENS2$ | Sensitivity (0.1%BER) @1Mbps | - | -87 | - | dBm |
| $RXSENS1$ | Sensitivity (0.1%BER) @2Mbps | - | -83 | - | dBm |
| $C/I_{CO}$ | C/I Co-channel (@250Kbps) | - | 2 | - | dBc |
| $C/I_{1ST}$ | 1st Adjacent Channel Selectivity C/I | - | -8 | - | dBc |
| $C/I_{2ND}$ | 2nd Adjacent Channel Selectivity C/I | - | -18 | - | dBc |
| $C/I_{3RD}$ | 3rd Adjacent Channel Selectivity C/I | - | -24 | - | dBc |
| $C/I_{4TH}$ | 4th Adjacent Channel Selectivity C/I | - | -28 | - | dBc |
| $C/I_{5TH}$ | 5th Adjacent Channel Selectivity C/I | - | -32 | - | dBc |
| $C/I_{6TH}$ | 6th Adjacent Channel Selectivity C/I | - | -35 | - | dBc |
| $C/I_{CO}$ | C/I Co-channel (@1Mbps) | - | 10 | - | dBc |
| $C/I_{1ST}$ | 1st Adjacent Channel Selectivity C/I | - | 1 | - | dBc |
| $C/I_{2ND}$ | 2nd Adjacent Channel Selectivity C/I | - | -18 | - | dBc |
| $C/I_{3RD}$ | 3rd Adjacent Channel Selectivity C/I | - | -23 | - | dBc |
| $C/I_{4TH}$ | 4th Adjacent Channel Selectivity C/I | - | -28 | - | dBc |
| $C/I_{5TH}$ | 5th Adjacent Channel Selectivity C/I | - | -32 | - | dBc |
| $C/I_{6TH}$ | 6th Adjacent Channel Selectivity C/I | - | -35 | - | dBc |
| $C/I_{CO}$ | C/I Co-channel (@2Mbps) | - | 10 | - | dBc |

| $C/I_{1ST}$ | 1st Adjacent Channel Selectivity C/I | - | -6 | - | dBc |
| $C/I_{2ND}$ | 2nd Adjacent Channel Selectivity C/I | - | -10 | - | dBc |
| $C/I_{3RD}$ | 3rd Adjacent Channel Selectivity C/I | - | -22 | - | dBc |
| $C/I_{4TH}$ | 4th Adjacent Channel Selectivity C/I | - | -28 | - | dBc |
| $C/I_{5TH}$ | 5th Adjacent Channel Selectivity C/I | - | -34 | - | dBc |

Note1: In the crystal 16MHz integer multiples (such as 2416, 2432MHz, etc) channels and the adjacent positive and negative 1MHz channels, receiving sensitivity degenerate 2dB.

Note2: The maximum length of data sent in 250kbps mode is 16 bytes.

## 4.6 DC Characteristics

Table 4-6 DC Characteristics

| Symbol | Parameter Condition (VCC = 3V±5%, TA=25°C) | Parameter value | | | Unit |
| --- | --- | --- | --- | --- | --- |
| | | Min | Type | Max | |
| $VDD$ | Supply voltage | 2.2 | 3 | 3.3 | V |
| $VSS$ | Ground | - | 0 | - | V |
| $V_{OH}$ | Output high level voltage | VDD-0.3 | - | VDD | V |
| $V_{OL}$ | Output low level voltage | VSS | - | VSS+0.3 | V |
| $V_{IH}$ | Input high level voltage | VDD-0.3 | - | VDD | V |
| $V_{IL}$ | Input low level voltage | VSS | - | VSS+0.3 | V |

# 5 Operational Modes

This chapter describes the various operating modes of the PAN2416AV and the methods for controlling the chip to get into each operating mode. PAN2416AV chip comes with the state machine controlled by the chip's internal register configuration values and external pin signals.

Figure 5-1 is PAN2416AV's working state diagram, showing the transition between five operating modes. PAN2416AV begins to work properly when VDD is greater than 2.2V. Even entering sleep mode, the MCU can still send configuration commands through the SPI and the CE pin to make the chip get into the other five states.
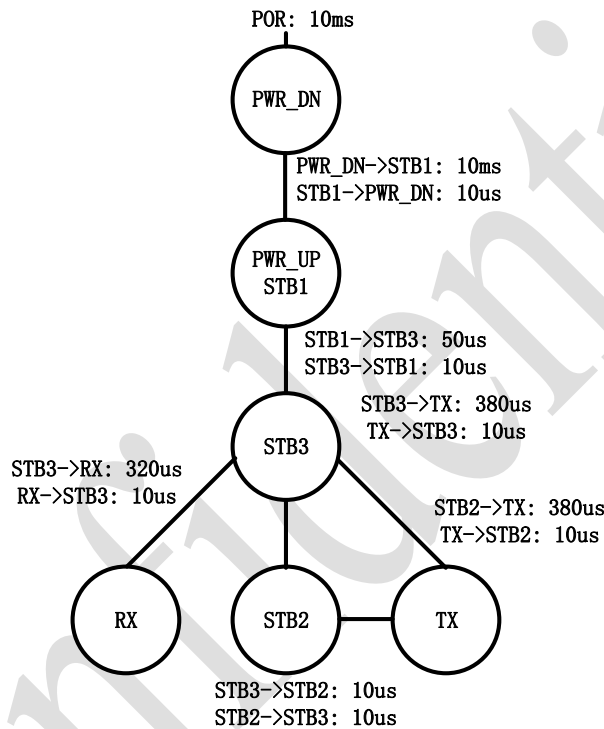


Figure 5-1 State diagram

The Table 5-1 describes how to configure the operational modes and the functions.

Table 5-1 Operational modes

| MODE | | PWR_DN | STB1 | STB3 | STB2 | RX | TX |
|---|---|---|---|---|---|---|---|
| CONTROL BIT | PWR_UP | 0 | 1 | 1 | 1 | 1 | 1 |
| | EN_PM | 0 | 0 | 1 | 1 | 1 | 1 |
| | CE | 0 | 0 | 0 | 1 | 1 | 1 |
| | PRIM_RX | X | X | X | 0 | 1 | 0 |
| | FIFO state | X | X | X | TX FIFO empty | X | Data in TX FIFOs |
| FUNCTION DESCRIPTION | SPI operation | √ | √ | √ | √ | √ | √ |
| | Keep register value | √ | √ | √ | √ | √ | √ |
| | Crystal oscillator work | X | √ | √ | √ | √ | √ |
| | Crystal oscillator output | X | X | X | √ | √ | √ |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | Main power management work | X | X | √ | √ | √ | √ |
| | Transmit module enabled | X | X | X | X | X | √ |
| | Receive module enabled | X | X | X | X | √ | X |

## 5.1 Power down Mode

In sleep mode, all functions of the PAN2416AV are turned off, keeping the current consumption to a minimum. After entering sleep mode, PAN2416AV stops working, but the register's contents remain unchanged. Sleep mode is controlled by the PWR_UP bit in the register.

## 5.2 Standby-I Mode (STB1)

In standby mode-I, the chip maintains crystal oscillation but doesn't output to other modules, the remaining functional modules are off, the current consumption is small. In sleep mode, the chip enters standby mode -I by setting the register PWR_UP to 1. In transmit or receive mode, the chip returns to standby mode -I by setting the CE and EN_PM control signals to 0.

## 5.3 Standby-III Mode (STB3)

In standby mode -I, the chip enters standby mode -III when the EN_PM control signal is set to 1. The main purpose of the standby mode -III is to make the power management module of the chip prior to the output of the oscillator.

## 5.4 Standby-II Mode (STB2)

Transmitting terminal TX FIFO register is empty and the CE pin is set to 0, entering standby mode-II (standby mode-II can usually be understood as the preparatory transmission mode). At this point, the crystal oscillator has a strong output drive capability and chip's power management module is turned on.In standby mode-II, if there are data packets into the TX FIFO, the chip's internal phase-locked loop immediately start to work and the transmitter will launch the packet after a period of phase-locked loop.

## 5.5 RX Mode

When PWR_UP, PRIM-RX, EN_PM, CE are set to 1, the chip enters receiving mode.

In the RX mode, the RF part receives the signal from the antenna, and then amplifies, down-converts, filters and demodulates the received signal. According to the address, check code, data length, etc., to determine whether the package is effective. If it is, then the effective package will be uploaded to RX FIFO, and the RF part will report interruption. If the RX FIFO is full, the received data packets will be discarded.

## 5.6 TX Mode

When PWR_UP, EN_PM is set, PRIM-RX is set to 0, CE is set and valid data exists in the TX FIFO, the chip enters to transmit mode. The PAN2416AV is kept in the transmit mode before the data packet is sent. After sending, the chip returns to standby mode. The PAN2416AV uses PLL open loop transmit mode, and the data packet is sent by single packet.

# 6 Communication Modes

The PAN2416AV completes the communication function together with the MCU. The link layer, such as data framing, parity check, address judgment, scrambling data blanking, data retransmission and ACK response, is done internally by the chip without MCU involvement.

The PAN2416AV chip can be configured as two different RX FIFO registers (32 bytes) or one RX FIFO register (64 bytes, six receive channels are shared), two different TX FIFO registers (32 bytes), or one TX FIFO register (64 bytes). The MCU can access the FIFO registers in Sleep and Standby modes.

PAN2416AV chip mainly has two kinds of data communication modes:

1) Communication mode without automatic retransmission without ACK (hereinafter referred to as normal mode). The transmitter can use the command W_TX_PAYLOAD, REUSE_TX_PL and so on.

2) Communication mode with automatic retransmission with ACK (latter referred to as enhanced mode). The transmitter can use the command W_TX_PAYLOAD, W_TX_PAYLOAD_NO-ACK, REUSE_TX_PL, etc. The receiver can use the command W_ACK_PAYLOAD, etc.

Table 6-1 Normal Mode

| Name of the communication | Normal mode | |
|---|---|---|
| Communication party | PTX | PRX |
| Characteristic | One-way transmission | One-way receiving |
| Frame mode of sending data | I | - |
| Command for openning the REUSE_TX_PL | Repeatedly send the previous packet data | - |

Table 6-2 Enhanced Mode

| Name of the communication | Enhanced mode | |
|---|---|---|
| Communication party | PTX | PRX |
| Characteristic | After sending data, wait to receive ACK | After receiving the data, send the ACK back |
| Frame mode of sending data | Send data framing mode II | Send ACK framing mode III back |
| PTX uses the REUSE_TX_PL command | Repeatedly send the previous packet data | Send ACK back every time a packet is received |
| PTX uses the W_TX_PAYLOAD command<br>PRX uses the W_ACK_PAYLOAD command | After sending data, wait to receive ACK PAYLOAD | After receiving the data, send ACK PAYLOAD back, framing mode II |
| PTX uses the W_TX_PAYLOAD_NO ACK command | Send a data, not waiting for ACK, framing mode II | Receive data, not sending ACK back |

## 6.1 Normal Mode

In normal mode, the transmitter fetches data from the TX FIFO register and sends it. After the transmission is complete, the interrupt is reported (the interrupt needs to be cleared) and the TX FIFO register clears the data (TX FIFO needs to be cleared). When the receiver receives a valid address and data, it reports the interrupt to the MCU. The MCU can then read the data from the RX FIFO register (the TX FIFO and RX FIFO need to be cleared and the interrupt needs to be cleared).

Normal mode, (0X01) EN_AA register is set to 0X00, (0X04) SETUP_RETR register is set to 0X00, (0X1C) DYNPD register is set to 0X00, the low 3 bit of the (0X1D) FEATURE register is set to 000.

## 6.2 Enhanced Mode

In enhanced mode, the party initiating the communication is referred to as the PTX (primary origi-nator), and the party receiving the data and responding there to is referred to as PRX (primary ter-minator). After the PTX sends the data, the chip waits for the response signal. After the PRX re-ceives the valid data, it replies with the response signal. If PTX did not receive the response signal within the specified time, it will automatically resend the data. Automatic retransmission and auto-answer function come with the PAN2416AV chip, without MCU's involvement.

PTX automatically switches to receive mode and waits for a reply after sending data. If the correct answer signal is not received within the specified time, the PTX will resend the same data packet until it receives a reply signal or if the number of transfers exceeds the value of ARC (SETUP_RETR register), a MAX_RT interrupt is generated. The PTX receives the response signal, which means the data has been transmitted successfully (PRX received valid data), cleared the data in TX FIFO and generated TX_DS interrupt (TX FIFO and RX FIFO need to be cleared and inter-rupt needs to be cleared).

PRX will send an ACK signal back every time a packet of valid data is received. If the data is new (the PID value is different from the previous packet data), it will be saved to RX FIFO, or it will be discarded.

Enhanced mode, we need to ensure that the PTX TX address (TX_ADDR), Channel 0 RX address (such as RX_ADDR_P0), and PRX RX address (such as RX_ADDR_P5) keep the same. For ex-ample: In Figure 5, PTX5 corresponds to PRX data channel 5, the address is set as follows:

PTX5: TX_ADDR=0xC2C3C4C5C1

PTX5: RX_ADDR_P0=0xC2C3C4C5C1

RX: RX_ADDR_P5=0xC2C3C4C5C1

The enhanced mode has the following characteristics:

1) Reduce MCU control and simplify software operation.

2) Strong anti-interference ability, reduce packet loss due to instantaneous co-channel interference in wireless transmission and easy to develop frequency hopping algorithm.

3) During the re-transmission, reduce the operation time of the data written and being sent every

time the MCU passes through the SPI interface.

## 6.3 Enhanced Send Mode

1) When CE is set to 0, the PRIM_RX bit in the CONFIG register is set to 0.

2) When the data is sent, the sending address (TX_ADDR) and the valid data (TX_PLD) are written to the address register and the TX FIFO by the SPI interface by the byte. When CSN pin is low, the data is written. When the CSN pin is high again, the data completes being written.

3) When CE changes from 0 to 1, the transmission is started (CE at least continues to set 1 above 30us, the operation takes effect).

4) In auto answer mode (SETUP_RETR register is not set to 0, ENAA_P0 = 1), PTX will automatically switch channel 0 to receive mode and wait for response signal immediately after sending data. If an ACK signal is received within the valid response time, the data transmission is considered as successful and the TX_DS bit in the status register is set and the data in the TX FIFO is automatically cleared. If no response signal is received within the set time range, the data is automatically retransmitted.

5) If the automatic transfer counter (ARC_CNT) overflows (exceeds the set value), the MAX_RT bit in the status register is set to 1 and data in the TX FIFO is not cleared. When MAX_RT or TX_DS is 1, the IRQ pin generates a low-level interrupt (interrupts must be enabled). Interrupts can be reset by writing to the status register.

6) The packet loss counter (PLOS_CNT) is incremented by one each time a MAX_RT interrupt is generated. Automatic transmission counter ARC_CNT count the times of the retransmission data packets. Packet Loss Counter PLOS_CNT counts the number of packets that still fail to be transmitted when the maximum number of transmission is allowed.

7) After the MAX_RT or TX_DS interrupt is generated, the system enters standby mode.

## 6.4 Enhanced Reception Mode

1) When CE is set to 0, the PRIM_RX bit in the CONFIG register is set to 1. The channel ready to receive data must be enabled (EN_RXADDR register) and all auto-acknowledge functions for the data channel operating in enhanced communication mode are enabled by the EN_AA register, and the valid data width is set by the RX_PW_PX register.

2) The receive mode is started by setting CE to 1.

3) After the preset waiting time, PRX starts to detect the wireless signal.

4) After receiving a valid data packet, the data is stored in RX_FIFO and the RX_DR bit is set to 1, an interrupt is generated. The RX_P_NO bit in the status register shows which channel the data was received from.

5) Send an ACK response signal automatically.

6) If CE remains 1, continue to receive mode. If CE is set to 0, enter standby mode-III.

7) The MCU reads the data through the SPI port at the proper rate.

## 6.5 Packet Identification in Enhanced Mode
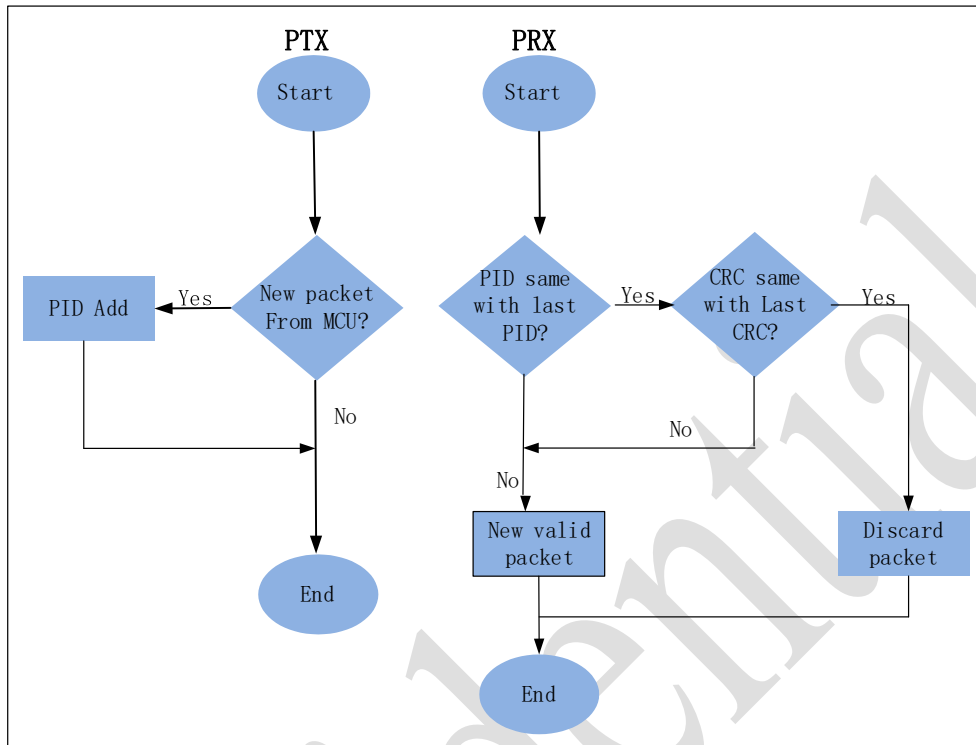


Figure 6-1 PID generation and detection

Each packet of data includes a two-bit PID (packet flag), to help the receiver to identify the data is new or retransmitted to prevent the multiple data packets into the same store, PID generation and detection is shown in Figure 6-1. The PID value adds one once the transmitting terminal gets a new packet from the MCU.

## 6.6 The PTX and PRX Timing of Enhanced BURST
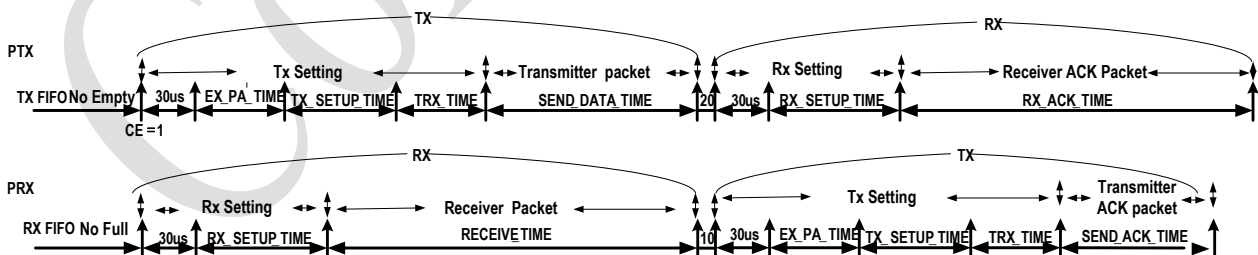


Figure 6-2 Timing diagram for PTX and PRX in enhanced mode (Sent successfully)

Figure 6-2 shows a chip internal timing diagram of a PTX and PRX communication, to make the communication success must meet the following two conditions:

1) Condition 1: The three period sum of phase-locked loop stability that PTX (or PRX) transmitted, amplifier enabled and phase-locked loop open loop must be 20us greater than the time of the PRX (or PTX) received phase-locked loop stabilization. This ensures that the time period during

which the PTX (or PRX) transmits data is within the time period during which the PRX (or PTX) receives the data, that is to say:

EX_PA_TIME + TX_SETUP_TIME + TRX_TIME > RX_SETUP_TIME +20us

2) Condition 2: The four period sum of phase-locked loop stability that PRX transmitted ACK, amplifier enabled, phase-locked loop open loop and ACK transmitted must be 80us less than the two period sum of phase-locked loop stabilization that the PTX received and waiting for the ACK. This ensures that the time period during which the PRX replies ACK is within the time period during which the PTX waits for the ACK, that is to say:

EX_PA_TIME + TX_SETUP_TIME + TRX_TIME + SEND_ACK_TIME <

RX_SETUP_TIME + RX_ACK_TIME - 80us

## 6.7 One-To-Multi Communication at the Receiving End in Enhanced BURST

As a transmitter, the PAN2416AV chip can communicate with multiple receivers with different addresses for one-to-multi communications.

As a receiver, the PAN2416AV chip can receive the data from 6 channels of different addresses and the same frequency. Each data channel has its own address.

Those data channels which enabled are set by the register EN_RXADDR. The address of each data channel is configured by register RX_ADDR_PX. In general, different data channels are not allowed to set the exactly same addresses. As shown below, an example of the configuration of multiple receive channel addresses is given in Figure 6-3.

| | Byte 4 | Byte 3 | Byte 2 | Byte 1 | Byte 0 |
|---|---|---|---|---|---|
| Data pipe 0(RX_ADDR_P0) | 0xF1 | 0xD2 | 0xE6 | 0xA2 | 0x33 |
| Data pipe 1(RX_ADDR_P1) | 0xD3 | 0xD3 | 0xD3 | 0xD3 | 0xD3 |
| Data pipe 2(RX_ADDR_P2) | 0xD3 | 0xD3 | 0xD3 | 0xD3 | 0xD4 |
| Data pipe 3(RX_ADDR_P3) | 0xD3 | 0xD3 | 0xD3 | 0xD3 | 0xD5 |
| Data pipe 4(RX_ADDR_P4) | 0xD3 | 0xD3 | 0xD3 | 0xD3 | 0xD6 |
| Data pipe 5(RX_ADDR_P5) | 0xD3 | 0xD3 | 0xD3 | 0xD3 | 0xD7 |
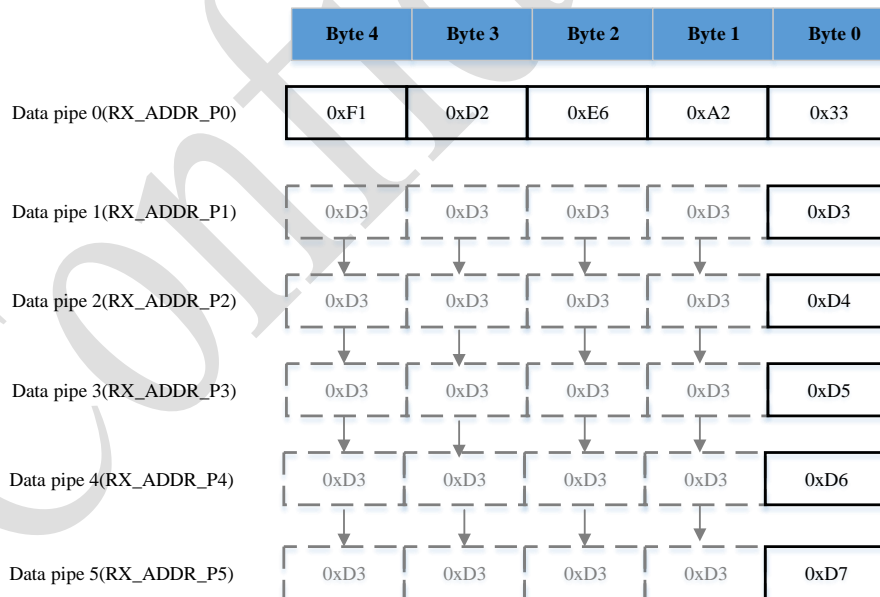
Figure 6-3 Multi-channel address setting

It can be seen from Figure 6-3 that a total of 40 bits of 5 bytes of data channel 0 are configurable. The addresses of the data channel 1~5 are configured as 32 bit shared addresses (shared with data channel 1) and 8 bit respective addresses (minimum bytes).

The PAN2416AV chip can communicate with up to 6 different channels in receive mode, as shown

in Figure 6-4. Each data channel uses a different address and shares the same channel. All transmitters and receivers are set to enhanced mode.

After receiving valid data, PRX records the TX address of PTX and sends a response signal with this address as the destination address. When the PTX data channel 0 is used to receive a response signal, the RX address of the data channel 0 is equal to the TX address to ensure that the correct response signal is received. Figure 6-4 shows an example of how the PTX and PRX addresses are configured.
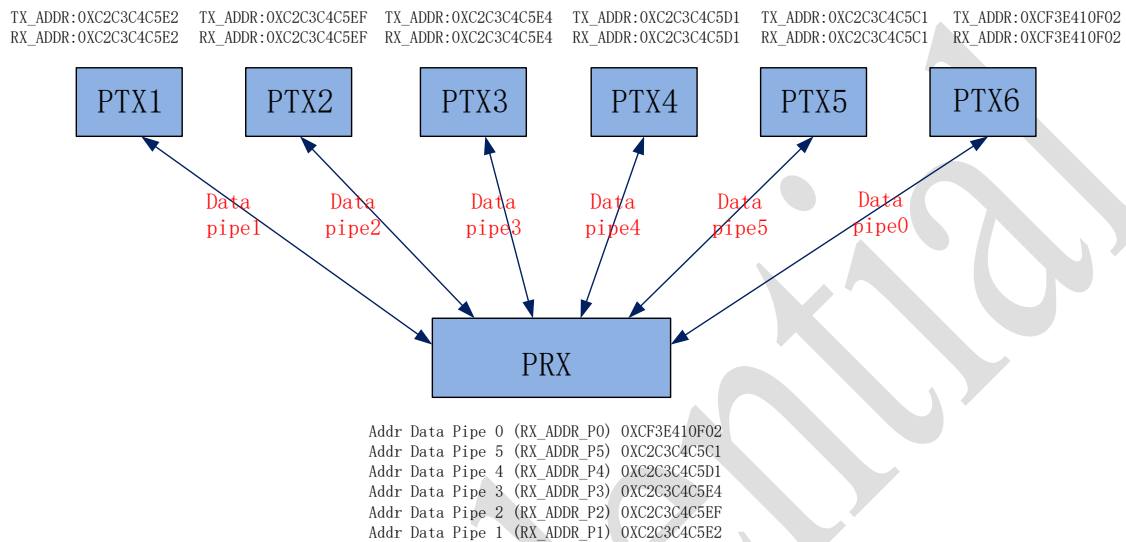


Figure 6-4 Example of Multichannel Data Transfer Response Address

## 6.8 DATA FIFO
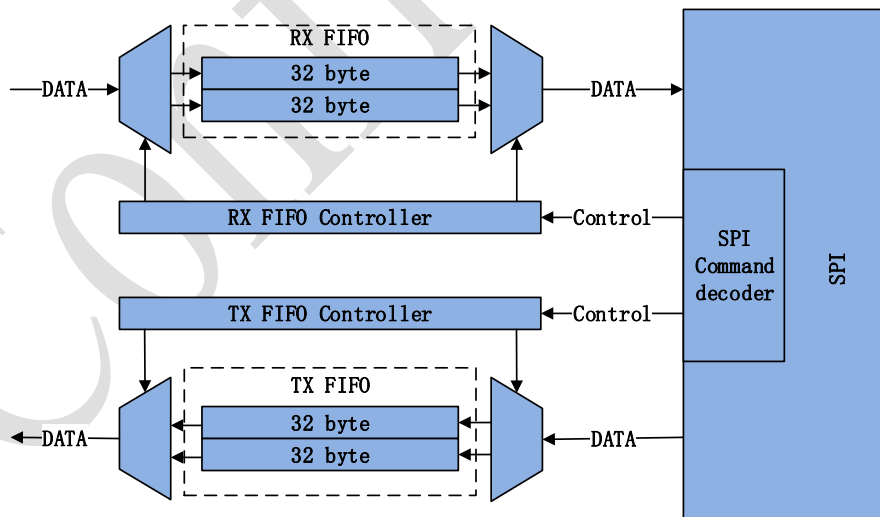


Figure 6-5 FIFO Block Diagram

PAN2416AV contains TX_FIFO and RX_FIFO. FIFO can be read and written through the SPI command. In transmit mode, TX_FIFO is written with the W_TX_PAYLOAD and W_TX_PAYLOAD_NO_ACK instructions. If a MAX_RT interrupt is generated, the data in TX_FIFO will not be cleared. In receive mode, the payload of RX_FIFO is read with R_RX_PAYLOAD instruction. R_RX_PL_WID instruction read the length of payload. The FIFO_STATUS register indicates the

status of the FIFO.

## 6.9 Interrupt Pin

The interrupt pin (IRQ) of PAN2416AV is triggered by a low level, the initial state of IRQ pin is high. When the TX_DS, RX_DR or MAX_RT bit in the status register is set to 1, and the corresponding interrupt enable bit is set to 0, the IRQ pin interrupt is triggered. When the MCU writes '1' to the corresponding interrupt source, the interrupt is cleared. The interrupt trigger of IRQ pin can be masked or enabled by setting interrupt enable bit to 1 which IRQ pin interrupt is disabled.

# 7 SPI CONTROL INTERFACE

The PAN2416AV is read and written to each register through the SPI control interface. The PAN2416AV is as a slave, and the data rate of the SPI interface generally depends on the interface speed of the MCU, its maximum data transmission rate is 4Mbps. In order to save power, the maximum transmission rate of SPI is 1Mbps in the sleep mode and standby mode-I.

The SPI interface is the standard SPI interface to see Table 7-1, which can be used to simulate the SPI interface with the universal I/O port of MCU. When the CSN pin is 0, the SPI interface waits for the execution instruction. A change from 1 to 0 of the CSN pin executes an instruction. After the CSN pin changes from 1 to 0, the content of the state register can be read through the MISO.

Table 7-1 SPI Interface

| PIN | I/O direction | Function description |
|------|---------------|----------------------|
| CSN | input | Chip select enable, low enabled |
| SCK | input | Clock |
| MOSI | input | Serial input |
| MISO | output | Serial output |

Table 7-2 The Pin Description of the RF and MCU Interface

| Interface Function | Interface Name (RF) | Interface state (RF) | Interface Name (MCU) | Interface state (MCU) |
|--------------------|---------------------|----------------------|----------------------|-----------------------|
| Mode chip select signal | CE | input | RA7 | output |
| SPI chip select signal | CSN | input | RE2 | output |
| SPI clock signal | SCK | input | RE1 | output |
| SPI data master out slave in | MOSI | input | RA5 | output |
| SPI data master in slave out | MISO | output | RA4 | input |
| Interrupt Signal | IRQ | output | RA3 | input |

Note: The interface status of the RF part in Table 7-2 are the default status and can not be changed. The interface status of MCU part need to be configured by the software to the table above, the chip can work properly.

## 7.1 SPI Instruction Format

<Command word: MS Bit to LS Bit (one byte)>

<Data bytes: LS Byte to MS Byte, MS Bit in each byte first>

Table 7-3 SPI commands

| Command | Command word (binary) | Data bytes | Operation |
|---------|------------------------|------------|-----------|
| R_REGISTER | 000A AAAA | 1 to 5<br>LS Byte first | Read status register<br>AAAAA = 5bit register address |
| W_REGISTER | 001A AAAA | 1 to 5 | Write status register |

| | | LS Byte first | AAAAA = 5bit register address<br>Only work in power down and standby mode-I |
|---|---|---|---|
| R_RX_PAYLOAD | 0110 0001 | 1 to 32/64<br>LS Byte first | Read the receiving data, read usually starts from 0 bytes, and the data will be deleted from RX FIFO after reading, work in receive mode. |
| W_TX_PAYLOAD | 1010 0000 | 1 to 32/64<br>LS Byte first | Write transmit mode, the write operation is usually started from 0 bytes. |
| FLUSH_TX | 1110 0001 | 0 | Clear TX FIFO |
| FLUSH_RX | 1110 0010 | 0 | Clear RX FIFO |
| REUSE_TX_PL | 1110 0011 | 0 | Used at the PTX side, the data sent in the last frame is used again and sent. This command is available immediately after sending data and executing the FLUSH_TX. This command can not be used during data transmission. |
| ACTIVATE<br><br><br><br><br><br><br>DEACTIVATE | 0101 0000 | 1 | With this command followed by data 0x73, the following functions are activated:<br>• R_RX_PL_WID<br>• W_TX_PAYLOAD_NOACK<br>• W_ACK_PAYLOAD<br>This command is only executed in sleep and standby modes.<br>With this command followed by data 0x8C, the above function will be disabled. |
| R_RX_PL_WID | 0110 0000 | 0 | Read the top RX-payload data width of the RX FIFO. |
| W_ACK_PAY-LOAD | 1010 1PPP | 1 to 32/64<br>LS Byte first | Rx mode is available.<br>Write Payload to be transmitted together with ACK packet on PIPE PPP. (PPP valid in the range from 000 to 101). Up to 2 ACK packets can be set. The data of the same PIPE will be sent following the principle of first-in first-out. Write usually starts from 0 bytes |
| W_TX_PAY-LOAD_NOACK | 1011 0000 | 1 to 32/64<br>LS Byte first | Write transmit data, usually start from 0 bytes. Execution in TX mode, using this command to send data does not treat as automatic response. |
| CE_FSPI_ON | 1111 1101 | 1 | The SPI command sets the inside logic of the CE to 1, followed by data 0x00 after this command. |
| CE_FSPI_OFF | 1111 1100 | 1 | The SPI command sets the inside logic of the CE to 0, followed by data 0x00 after this command. |
| RST_FSPI_HOLD<br><br><br>RST_FSPI_RELS | 0101 0011 | 1 | Use this command followed by data 0x5A, enter the reset state and hold.<br>Use this command followed by data 0xA5, release the reset state and start to work properly. |

| NOP | 1111 1111 | 0 | No operation. |
|-----|-----------|---|---------------|

The R_REGISTER and W_REGISTER registers may operate on single-byte or multi-byte registers. The first byte to be read / written when accessing a multi-byte register is the high bit of the lowest byte. Multi-byte registers can write part of the byte only, the high byte without written will keep the original content unchanged. For example: The lowest byte of the RX_ADDR_P0 register can be changed by writing a byte to the register RX_ADDR_P0.

## 7.2 Timing Sequence
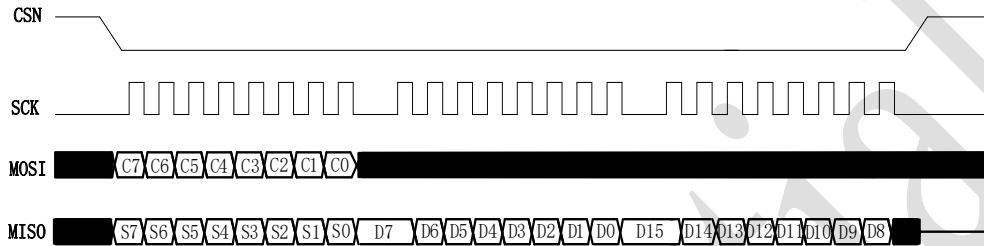


Figure 7-1 SPI Read Operation
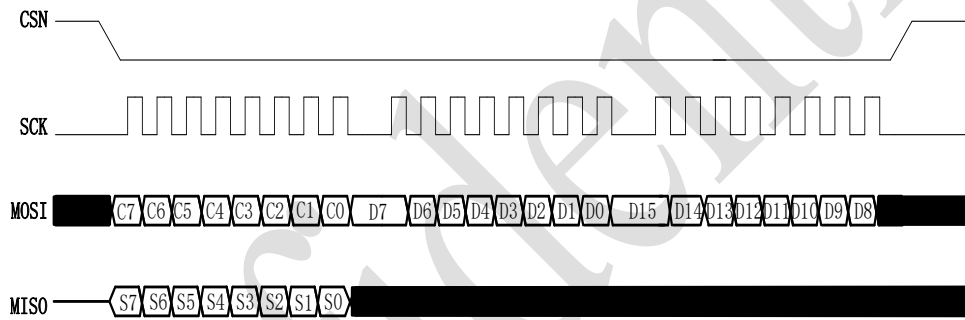


Figure 7-2 SPI Write Operation
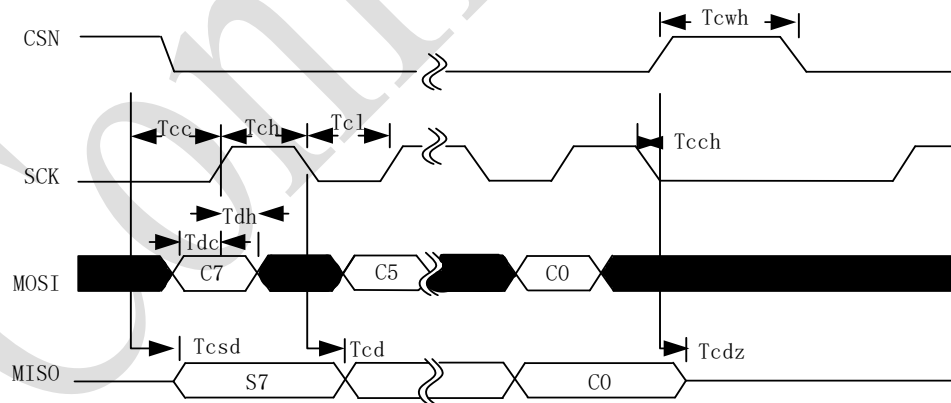


Figure 7-3 SPI, NOP Operation Timing Diagram

Table 7-4 SPI Operation Reference time

| Symbol | Parameters | Min | Max | Units |
|--------|-----------|-----|-----|-------|
| Tdc | Data set-up time | 15 | - | ns |
| Tdh | Data hold-up time | 2 | - | ns |
| Tcsd | CSN signal valid time | - | 40 | ns |

| Tcd | SCK signal valid time | - | 51 | ns |
|---|---|---|---|---|
| Tcl | SCK signal low level time | 38 | - | ns |
| Tch | SCK signal high level time | 38 | - | ns |
| Fsck | SCK signal frequency | - | 8 | MHz |
| Tr,Tf | SCK signal rise and fall time | - | 110 | ns |
| Tcc | CSN signal setup time | 2 | - | ns |
| Tcch | CSN signal holdup time | 2 | - | ns |
| Tcwh | CSN invalid time | 49 | - | ns |
| Tcdz | CSN signal high impedance | - | 40 | ns |

Note: The parameters of Table 7-4 can be adjusted according to the selected MCU.

Figure 7-1 through Figure 7-3 and Table 7-4 show the SPI operation and timing. The following symbols are used in the figure:

Ci-SPI instruction bit

Si-Status register bit

Di-Data bit (Remarks: From low byte to high byte, high in the front at each byte )

Among: i=1, 2, 3……n.

# 8 CONTROL REGISTER

You can configure and control the PAN2416AV by SPI reading and writing the registers in Table 8-1. Registers whose address with symbol * need to be modified when being used.

Table 8-1 Control Register

| Address (hex ) | Register | Bit | Reset value | type | Description |
|---|---|---|---|---|---|
| 00* | CONFIG | - | - | - | Working register |
| | EN_PM | 7 | 0 | R/W | Enter STB3 mode (Precondition PWR_UP=1) 1: enter STB3 0: enter STB1 (In the STB3 mode, it is necessary to wait for more than 50uS to jump to other working states) |
| | MASK_RX_DR | 6 | 0 | R/W | Interrupt enable bit of receiving data successful 1: Interrupts are not reflected on the IRQ pin 0: The RX_DR interrupt is reflected on the IRQ pin |
| | MASK_TX_DS | 5 | 0 | R/W | Interrupt enable bit of transmitting Data Successful 1: Interrupts are not reflected on the IRQ pin 0: The TX_DR interrupt is reflected on the IRQ pin |
| | MASK_MAX_RT | 4 | 0 | R/W | Interrupt enable bit of sending fail and reaching maximum transmit times 1: Interrupts are not reflected on the IRQ pin 0: The MAX_RT interrupt is reflected on the IRQ pin |
| | EN_CRC | 3 | 1 | R/W | CRC enable bit 1: CRC enabled, 2 byte 0: CRC not enabled, and No CRC verification |
| | N/A | 2 | 0 | R/W | Saved, need to be set to 1 |
| | PWR_UP | 1 | 0 | R/W | Chip enable bit 1: POWER_UP 0: POWER_DOWN |
| | PRIM_RX | 0 | 0 | R/W | RX/TX control bit 1: PRX 0: PTX |
| 01 | EN_AA Enhanced Burst | - | - | - | The automatic response enable of the receiving channel(When EN_AA of receiver is not 0X00, it is enhanced mode) |

| | | | | | |
|---|---|---|---|---|---|
| | Reserved | 7:6 | 00 | R/W | Only 00 allowed |
| | ENAA_P5 | 5 | 0 | R/W | Enable auto acknowledgement data pipe 5 |
| | ENAA_P4 | 4 | 0 | R/W | Enable auto acknowledgement data pipe 4 |
| | ENAA_P3 | 3 | 0 | R/W | Enable auto acknowledgement data pipe 3 |
| | ENAA_P2 | 2 | 0 | R/W | Enable auto acknowledgement data pipe 2 |
| | ENAA_P1 | 1 | 0 | R/W | Enable auto acknowledgement data pipe 1 |
| | ENAA_P0 | 0 | 1 | R/W | Enable auto acknowledgement data pipe 0 |
| 02 | EN_RXADDR | - | - | - | Enable of the receiving channel |
| | Reserved | 7:6 | 00 | R/W | Only 00 allowed |
| | ERX_P5 | 5 | 0 | R/W | Data pipe5 enabled |
| | ERX_P4 | 4 | 0 | R/W | Data pipe4 enabled |
| | ERX_P3 | 3 | 0 | R/W | Data pipe3 enabled |
| | ERX_P2 | 2 | 0 | R/W | Data pipe2 enabled |
| | ERX_P1 | 1 | 0 | R/W | Data pipe1 enabled |
| | ERX_P0 | 0 | 1 | R/W | Data pipe0 enabled |
| 03 | SETUP_AW | - | - | - | Address width setting |
| | Reserved | 7:2 | 000000 | R/W | Only 000000 allowed |
| | AW | 1:0 | 11 | R/W | RX/TX address width<br>00: invalid<br>01: 3 bytes<br>10: 4 bytes<br>11: 5 bytes<br>If the address width is set below 5 bytes, the address uses the LS byte |
| 04 | SETUP_RETR | - | - | - | Automatic transmission setting |
| | ARD | 7:4 | 0000 | R/W | Automatic transmission delay<br>0000: 250μs<br>0001: 500μs<br>0010: 750μs<br>……<br>1111: 4000μs |
| | ARC | 3:0 | 0011 | R/W | Automatic setting for the number of transmission<br>0000: normal mode<br>0001~1111: enhanced mode<br>0001: enhanced 1 transmission<br>0002: enhanced 2 transmission<br>......<br>1111: enhanced 15 transmission |
| 05 | RF_CH | - | - | - | Communication channel setting |
| | Reserved | 7 | 0 | R/W | Only 0 allowed |
| | RF_CH | 6:0 | 1001110 | R/W | The use channel is set as<br>Channel=RF_CH + 2400 |
| 06* | RF_SETUP | - | - | - | Communication parameters configuration |

| | | | | | |
|---|---|---|---|---|---|
| | RF_DR | 7:6 | 00 | R/W | Data rate<br>01: 2Mbps<br>00: 1Mbps<br>11: 250kbps<br>10: saved |
| | PA_GC | 5:3 | 111 | R/W | The output amplitude of PA's driver level, the transmitted power can be adjusted.<br>111: large amplitude<br>......<br>000: small amplitude |
| | PA_PWR | 2:0 | 111 | R/W | The power selection of PA's output level, the transmitted power can be adjusted.<br>111: large output power<br>......<br>000: small output power |
| 07 | STATUS | - | - | - | State register |
| | Reserved | 7 | 0 | R/W | Only 0 allowed |
| | RX_DR | 6 | 0 | R/W | The receiving data interrupt bit of RX FIFO, an interrupt is generated when new data is received and reaches the RX FIFO.<br>Write 1 to clear bit. |
| | TX_DS | 5 | 0 | R/W | The sending data interrupt bit of TX FIFO.<br>Without automatic retransmission mode, an interrupt is generated after data transmission completes.<br>With automatic retransmission mode, this bit is set high only after the sender has received the ACK signal.<br>Write 1 to clear bit. |
| | MAX_RT | 4 | 0 | R/W | Maximum number of TX retransmits interrupt write 1 to clear bit. If MAX_RT is asserted it must be cleared to enable further communication. |
| | RX_P_NO | 3:1 | 111 | R | Read pipe's number from RX_FIFO<br>000-101: pipe number<br>110: Not used<br>111: RX_FIFO null |
| | TX_FULL | 0 | 0 | R | TX FIFO full mark<br>1: TX FIFO is full<br>0: TX FIFO is not full and is available |
| 08 | OBSERVE_TX | - | - | - | Transmission status register |
| | PLOS_CNT | 7:4 | 0 | R | Packet loss counter<br>The counter will stop counting when it reaches the maximum value of 15. |

| | | | | | |
|---|---|---|---|---|---|
| | | | | | This counter is reset when RF_CH is written. When this value is not reset, communication can continue. |
| | ARC_CNT | 3:0 | 0 | R | The transmission times counter of automatic re-transmission. Transmit once more, then ARC_CNT adds one. When the ARC_CNT reaches the ARC limit, it is regarded as a packet loss and a PLOS_CNT plus one. The counter is reset when the new data is written to TX FIFO. |
| 09* | DATAOUT | - | - | - | Data read register (The premise is DATAOUT_SEL = 0) |
| | ANADATA7 | 7 | 0 | R | The third bits (the highest bits) of the real time RSSI value of the receiver (for test) |
| | ANADATA6 | 6 | 0 | R | The second bits of the real time RSSI value of the receiver (for test) |
| | ANADATA5 | 5 | 0 | R | The first bits of the real time RSSI value of the receiver (for test) |
| | ANADATA4 | 4 | 0 | R | The zeroth bits of the real time RSSI value of the receiver (for test) |
| | ANADATA3 | 3 | 0 | R | The third bits (the highest bits) of the RSSI value of the receiver successfully receiving the package |
| | ANADATA2 | 2 | 0 | R | The second bits of the RSSI value of the receiver successfully receiving the package |
| | ANADATA1 | 1 | 0 | R | The first bits of the RSSI value of the receiver successfully receiving the package |
| | ANADATA0 | 0 | 0 | R | The zeroth bits of the RSSI value of the receiver successfully receiving the package |
| 0A | RX_ADDR_P0 | 39:0 | 0xE7E7E 7E7E7 | R/W | The receiving address of data pipe 0, up to 5 bytes. (Write from LS byte. The address length is defined by SETUP_AW) |
| 0B | RX_ADDR_P1 | 39:0 | 0xC2C2C 2C2C2 | R/W | The receiving address of data pipe 1, up to 5 bytes. (Write from low byte. The address length is defined by SETUP_AW) |
| 0C | RX_ADDR_P2 | 7:0 | 0xC3 | R/W | The receiving address of data pipe 2, only the lowest bit. The high bits are equal to RX_ADDR_P1[39:8] |
| 0D | RX_ADDR_P3 | 7:0 | 0xC4 | R/W | The receiving address of data pipe 3, only the lowest bit. The high bits are equal to RX_ADDR_P1[39:8] |
| 0E | RX_ADDR_P4 | 7:0 | 0xC5 | R/W | The receiving address of data pipe 4, only the lowest bit. The high bits are equal to RX_ADDR_P1[39:8] |

| | | | | | |
|---|---|---|---|---|---|
| 0F | RX_ADDR_P5 | 7:0 | 0xC6 | R/W | The receiving address of data pipe 5, only the lowest bit. The high bits are equal to RX_ADDR_P1[39:8] |
| 10 | TX_ADDR | 39:0 | 0xE7E7E 7E7E7 | R/W | The transmitter address (write from LS byte) can only be used in a chip configured as a PTX mode. RX_ADDR_P0 needs to be set equal to this address in order to receive an ACK auto-response. |
| 11 | RX_PW_P0 | - | - | - | The data length of the RX payload in data pipe 0 |
| | Reserved | 7 | 0 | R/W | Only 0 allowed |
| | RX_PW_P0 | 6:0 | 0000000 | R/W | The data length of the RX payload in data pipe 0 (from 1 to 32/64 bytes) <br> 0: the pipe not used <br> 1=1 byte <br> ... <br> 32/64= 32/64 bytes |
| 12 | RX_PW_P1 | - | - | - | The data length of the RX payload in data pipe 1 |
| | Reserved | 7 | 0 | R/W | Only 0 allowed |
| | RX_PW_P1 | 6:0 | 0000000 | R/W | The data length of the RX payload in data pipe 1 (from 1 to 32/64 bytes) <br> 0: the pipe not used <br> 1=1 byte <br> ... <br> 32/64= 32/64 bytes |
| 13 | RX_PW_P2 | - | - | - | The data length of the RX payload in data pipe 2 |
| | Reserved | 7 | 0 | R/W | Only 0 allowed |
| | RX_PW_P2 | 6:0 | 0000000 | R/W | The data length of the RX payload in data pipe 2(from 1 to 32/64 bytes) <br> 0: the pipe not used <br> 1=1 byte <br> ... <br> 32/64= 32/64 bytes |
| 14 | RX_PW_P3 | - | - | - | The data length of the RX payload in data pipe 3 |
| | Reserved | 7 | 0 | R/W | Only 0 allowed |
| | RX_PW_P3 | 6:0 | 0000000 | R/W | The data length of the RX payload in data pipe 3 (from 1 to 32/64 bytes) <br> 0: the pipe not used <br> 1=1 byte <br> ... <br> 32/64= 32/64 bytes |
| 15 | RX_PW_P4 | - | - | - | The data length of the RX payload in data pipe 4 |
| | Reserved | 7 | 0 | R/W | Only 0 allowed |
| | RX_PW_P4 | 6:0 | 0000000 | R/W | The data length of the RX payload in data pipe 4 (from 1 to 32/64 bytes) |

| | | | | | |
|---|---|---|---|---|---|
| | | | | | 0: the pipe not used<br>1=1 byte<br>...<br>32/64= 32/64 bytes |
| 16 | RX_PW_P5 | - | - | - | The data length of the RX payload in data pipe 5 |
| | Reserved | 7 | 0 | R/W | Only 0 allowed |
| | RX_PW_P5 | 6:0 | 0000000 | R/W | The data length of the RX payload in data pipe 5 (from 1 to 32/64 bytes)<br>0: the pipe not used<br>1=1 byte<br>...<br>32/64= 32/64 bytes |
| 17* | FIFO_STATUS | - | - | - | FIFO state register |
| | N/A | 7 | 0 | R | Saved |
| | TX_REUSE | 6 | 0 | R | Call the indicator bit for the previous frame of the data.<br>After using the REUSE_TX_PL command, the bit is 1, and the last frame data of the transmission is retransmitted. The bit can be reset by command W_TX_PAYLOAD, W_TX_PAYLOAD_NO-ACK, DEACTIVATE, and FLUSH TX. |
| | TX_FULL | 5 | 0 | R | TX FIFO full mark bits<br>1: TX FIFO is full<br>0: TX FIFO is available |
| | TX_EMPTY | 4 | 1 | R | TX FIFO null mark bits<br>1: TX FIFO is null<br>0: TX FIFO has data |
| | N/A | 3 | 0 | R | Saved |
| | N/A | 2 | 0 | R | Saved |
| | RX_FULL | 1 | 0 | R | RX FIFO full mark bits<br>1: RX FIFO is full<br>0: RX FIFO is available |
| | RX_EMPTY | 0 | 1 | R | RX FIFO null mark bits<br>1: RX FIFO is null<br>0: RX FIFO has data |
| N/A | TX_PLD | 255:0 | X | W | TX sending data<br>TX data is written by SPI command, and the data is stored in the 2 level 32 bytes or 1 level 64 bytes of FIFO |
| N/A | RX_PLD | 255:0 | X | R | RX receiving data<br>RX data is read by SPI command, and the data is stored in the 2 level 32 bytes or 1 level 64 bytes of FIFO. |

| | | | | | All the RX PIPE share the same FIFO. |
|---|---|---|---|---|---|
| 19* | DEMOD_CAL | 7:0 | - | - | Modulation and demodulation parameter register (Can be configured by the needs of the program) |
| | CHIP | 7 | 0 | R/W | Set the chip whether into the test mode<br>1: get into test mode<br>0: drop out test mode |
| | CARR | 6:5 | 00 | R/W | Set the chip whether into carrier test mode<br>11: get into single carrier test mode, and CHIP is set to 1<br>00: drop out single carrier test mode |
| | GAUS_CAL | 4:1 | 0111 | R/W | The signal amplitude of the output of the Gauss filter to the DAC is adjusted. The size of the output signal is one of the determinants of the transmit modulation frequency offset.<br>1111: small amplitude<br>...<br>1000: medium amplitude<br>...<br>0000: large amplitude |
| | Scramble_en | 0 | 1 | R/W | Scrambling code function whether enabled. Turning on the scramble function can whitelize the data to be sent, thereby reducing the length of 1 long 0 data. To enable the scrambling function needs to send and receive both terminals of the same configuration.<br>1: enable scrambling<br>0: close scrambling |
| 1A* | RF_CAL2 | 47:0 | - | - | Supplementary RF register (Generally use the default value) |
| | IVCO_SEL<1:0> | 47:46 | 01 | R/W | Bias current selection of VCO<br>00: IBG 0uA + IPTAT 55uA = 55uA<br>01: IBG 9uA + IPTAT 44uA = 53uA<br>10: IBG 18uA + IPTAT 33uA = 51uA<br>11: IBG 27uA + IPTAT 22uA = 49uA |
| | BW_500K | 45 | 0 | R/W | Filter bandwidth selection<br>0: Narrow bandwidth<br>1: Wide bandwidth |
| | GC_500K | 44 | 1 | R/W | Filter gain selection<br>0: low gain<br>1: high gain |
| | NC | 43:39 | 00000 | R/W | Reserved |
| | PA_ramp_sel | 38:37 | 01 | R/W | Select the way of PA ramp up<br>00: No ramp up |

| | | | | |
|---|---|---|---|---|
| | | | | 01: 4us ramp each step<br>10: Start ramp from half-current<br>11: 2us ramp each step |
| TX_VCO_BIAS<2:0> | 36:34 | 110 | R/W | VCO current setting<br>000: 900uA<br>001: 1050uA<br>010: 1200uA<br>011: 1350uA<br>100: 1500uA<br>101: 1650uA<br>110: 1800uA<br>111: 1950uA |
| NC | 33 | 1 | R/W | Reserved |
| BPF_CTRL_BW | 32 | 0 | R/W | 1dB bandwidth selection for receiving intermediate frequency filters<br>1: ×1<br>0: ×0.85 |
| BPF_CTRL_GAIN | 31 | 1 | R/W | Gain control for receiving intermediate frequency filters<br>1: 5dB<br>0: 19dB |
| VCOBUF_IC | 30:29 | 01 | R/W | The diver current selection of the MIXH driven by VCO<br>00: 600uA<br>01: 800uA<br>10: 1mA<br>11: 1.2mA |
| VCO_CT | 28:27 | 01 | R/W | Selection of VCO load added capacitance<br>00: few capacitances, high VCO frequency<br>11: many capacitances, low VCO frequency |
| CAL_VREF_SEL | 26 | 1 | R/W | The reference voltage selection of VCO auto correction<br>1: 1.15V<br>0: 1.25V |
| SPI_CAL_EN | 25 | 0 | R/W | VCO single trigger automatic correction process VCO automatic correction process will be triggered each time the bit is set from 0 to 1. In addition, the VCO automatic correction process will also be triggered by changing the work channel and entering the receiving and transmitting state from the standby state. |
| PREAMP_CTM | 24:22 | 011 | R/W | Load capacitance selection at the driver level of PA |

| | | | | |
|---|---|---|---|---|
| | | | | 000: 399fF |
| | | | | 100: 171fF |
| | | | | 111: 0fF |
| DA_LPF_BW | 21 | 1 | R/W | Filter bandwidth selection of DAC<br>1: wide bandwidth<br>0: narrow bandwidth |
| RX_CTM | 20:19 | 01 | R/W | Selection of resonance frequency (load capacitance) of LNA<br>00: 2.45GHz<br>01: 2.52GHz<br>10: 2.59GHz<br>11: 2.66GHz |
| RCCAL_EN | 18 | 1 | R/W | Automatic correction enable of receiving band-pass filters<br>1: enabled<br>0: not enabled |
| EN_VCO_CAL | 17 | 1 | R/W | VCO automatic correction enabled position<br>1: enabled<br>0: not enabled |
| PRE_BC | 16:14 | 100 | R/W | DC current selection of prefrequency divider<br>000: ×1<br>001&010: ×1.5<br>100&011: ×2<br>101&110: ×2.5<br>111: ×3 |
| VCO_CODE_IN | 13:10 | 1000 | R/W | VCO band selection bit, only valid when EN_VCO_CAL is set to 0<br>1111: high frequency period<br>0000: low frequency period |
| RCCAL_IN | 9:4 | 010100 | R/W | Intermedium frequency correction bit setting for receiving band pass filter, only valid when RCCAL_EN is set to 0.<br>111111: Low medium center frequency<br>000000: High medium center frequency |
| CPSEL | 3:2 | 01 | R/W | Current setting of PLL charge pump<br>RX           TX<br>00: 26uA      26uA<br>01: 26uA      52uA<br>10: 52uA      78uA<br>11: 78uA      104uA |
| DATAOUT_SEL | 1 | 0 | R/W | Data read select bit, set 0 |
| RSSI_SEL | 0 | 1 | R/W | Sampling point selection of RSSI signal<br>1: Sampling signal passes through filter |

| | | | | | |
|---|---|---|---|---|---|
| | | | | | 0: Sampling signal does not pass through filter (for test) |
| 1B | DEM_CAL2 | 23:0 | - | - | Supplementary parameter register(Generally use the default value) |
| | PIN | 23:21 | 000 | R/W | Set the output PIN after the chip enters the test mode(MISO pin/IRQ pin) <br> 000(and CHIP=0) is at working mode, as data output and interrupt output. <br> 000(and CHIP=1) is at testing sensitivity mode, as demodulating data and clock output. <br> 110(and CHIP=1) is at test receiving mode, as limit I and Q two road output. |
| | EN_RX | 20 | 0 | R/W | Whether the receiving channel and the PLL are open at the same time. <br> 1: open at the same time <br> 0: open at different time |
| | DELAY1 | 19 | 0 | R/W | Whether the open-loop PLL is enabled, the open loop state of the PLL enable can be used as a carrier drift test. <br> 1: PLL enable open loop <br> 0: whether the PLL is open loop is controlled by the state machine. |
| | DELAY0 | 18 | 0 | R/W | Whether the demodulator overlay the reported initial offset, the demodulator not overlay the initial offset can act as a receiver sensitivity test. <br> 1: No overlay initial offset <br> 0: overlay initial frequency offset, the error code caused by the central frequency offset can be neutralized in receiving mode. |
| | TH1 | 17 | 1 | R/W | In standby mode-II, whether LDO (except DVDD's LDO) is enabled. <br> In test mode, this bit is set to 1 when testing the transmit single carrier and receive sensitivity. <br> 1: enabled <br> 0: not enabled |
| | PTH | 16:13 | 0110 | R/W | Set the Correlation threshold of the lead code of the receiver digital demodulator. <br> Correlation threshold of 24 bit lead code=PTH+16 <br> 1000: 24 bits <br> 0110: 22 bits <br> 0000: 16 bits |
| | SYNC_SEL | 12 | 1 | R/W | 4 times sampling of receiver digital demodulator, take a few corrections to calculate the bit whether |

| | | | | | |
|---|---|---|---|---|---|
| | | | | | the data is correct. 1: 3bit 0: 2bit |
| | DECOD_INV | 11 | 1 | R/W | Whether the lead code reversed by bit, set to 1 generally so that the function needs to be sent and received at both ends. 1: not reversed by bit 0: reversed by bit |
| | GAIN1 | 10:7 | 1110 | R/W | The data center value of the demodulator adjusts the amplitude of the baseline waveform of the loop, and sets 1110. |
| | GAIN2 | 6:1 | 000101 | R/W | The data center value of the demodulator adjusts the speed of the baseline waveform of the loop, and sets 000101. |
| | AGGRESSIVE | 0 | 1 | R/W | The speed selection of the code rate synchronization unit of the demodulator 1: Large step length adjustment, fast speed 0: Small step length adjustment, slow speed |
| 1C | DYNPD | - | - | - | Dynamic PAYLOAD length enable |
| | Reserved | 7:6 | 00 | R/W | Only 00 allowed |
| | DPL_P5 | 5 | 0 | R/W | Dynamic PAYLOAD length of the Enable PIPE 5 (need EN_DPL and ENAA_P5) |
| | DPL_P4 | 4 | 0 | R/W | Dynamic PAYLOAD length of the Enable PIPE 4 (need EN_DPL and ENAA_P4) |
| | DPL_P3 | 3 | 0 | R/W | Dynamic PAYLOAD length of the Enable PIPE 3 (need EN_DPL and ENAA_P3) |
| | DPL_P2 | 2 | 0 | R/W | Dynamic PAYLOAD length of the Enable PIPE 2 (need EN_DPL and ENAA_P2) |
| | DPL_P1 | 1 | 0 | R/W | Dynamic PAYLOAD length of the Enable PIPE 1 (need EN_DPL and ENAA_P1) |
| | DPL_P0 | 0 | 0 | R/W | Dynamic PAYLOAD length of the Enable PIPE 0 (need EN_DPL and ENAA_P0) |
| 1D* | FEATURE | 7:0 | | R/W | Feature register |
| | Reserved | 7 | 0 | R/W | Only 00 allowed |
| | MUX_PA_IRQ | 6 | 0 | R/W | Select IRQ signal output or EN_PA signal to PIN 0: IRQ signal output to PIN 1: EN_PA signal output to PIN |

| | | | | | |
|---|---|---|---|---|---|
| | CE_SEL | 5 | 0 | R/W | Enable the CE to be opened by command<br>0: CE controlled by CE's pin<br>1: CE controlled by command |
| | DATA_LEN_SEL | 4:3 | 00 | R/W | Data length selection<br>11: 64byte (512bit) mode<br>00: 32byte (256bit) mode |
| | EN_DPL | 2 | 0 | R/W | Enable dynamic PAYLOAD length |
| | EN_ACK_PAY | 1 | 0 | R/W | Enable ACK with PAYLOAD |
| | EN_NOACK | 0 | 0 | R/W | Enable W_TX_PAYLOAD_NOACK command |
| 1E* | RF_CAL | 23:0 | - | R/W | RF reference register (Can be configured by the needs of the program) |
| | OSC_IC | 23 | 0 | R/W | Excitation current selection of the OSC<br>1: ×1<br>0: ×0.75 |
| | DA_VREF_MB | 22:20 | 101 | R/W | The reference voltage of the DAC's comparator circuit at positive terminal. The greater the positive reference voltage is, the greater DAC output amplitude is.<br>111: large positive reference voltage<br>000: small positive reference voltage |
| | DA_VREF_LB | 19:17 | 110 | R/W | The reference voltage of the DAC's comparator circuit at negative terminal. The greater the negative reference voltage is, the greater DAC output amplitude is.<br>111: small negative reference voltage<br>000: large negative reference voltage |
| | DA_LPF_CTRL | 16 | 1 | R/W | The control bit of DAC's output amplitude<br>1: output amplitude×0.8<br>0: output amplitude×0.5 |
| | RSSI_EN | 15 | 0 | R/W | RSSI enable bit<br>1: RSSI enabled<br>0: RSSI not enabled |
| | RSSI_Gain_CTR | 14:13 | 01 | R/W | The selection bit of RSSI's signal gain attenuation<br>00: Unattenuated<br>01: -6dB<br>10: -12dB<br>11: -18dB |
| | MIXL_GC | 12 | 1 | R/W | The gain selection of receiving MIXL<br>1: 14dB<br>0: 8dB |
| | POWPA_CTM<1:0> | 11:10 | 11 | R/W | The selection of resonance capacitance at PA output stage |

| | | | | | |
|---|---|---|---|---|---|
| | | | | | 00: big capacitance, low resonance frequency point<br>01: ...<br>10: ...<br>11: small capacitance, high resonance frequency point |
| | LNA_GC | 9:8 | 11 | R/W | The LNA gain selection<br>11: 17dB<br>10: 11dB<br>01: 5.4dB<br>00: -0.4dB |
| | RX_VCO_BIAS<2:0> | 7:5 | 111 | R/W | The VCO current setting of RX<br>000: 900uA<br>001: 1050uA<br>010: 1200uA<br>011: 1350uA<br>100: 1500uA<br>101: 1650uA<br>110: 1800uA<br>111: 1950uA |
| | RES_SEL | 4:3 | 10 | - | The load selection of chip's reference current<br>00: 26kR<br>01: 24kR<br>10: 22kR<br>11: 20kR |
| | LNA_HCURR | 2 | 1 | R/W | Set LNA high current enable<br>1: high current<br>0: low current |
| | MIXL_BC | 1 | 1 | R/W | Receiving MIXL current selection<br>1: ×1<br>0: ×0.5 |
| | IB_BPF_TRIM | 0 | 0 | R/W | Receiving bandpass filter's current selection<br>1: ×1<br>0: ×0.5 |
| 1F* | BB_CAL | 7:0<br>15:8<br>23:16<br>31:24<br>39:32 | - | R/W | Digital baseband parameter register (Generally use the default value) |
| | Reserved | 39:32 | 01000110 | R/W | Only 0X01000110 allowed |
| | INVERTER | 31 | 1 | R/W | Whether to reverse the data of RX path before entering RX_block<br>1: reverse |

| | | | | |
|---|---|---|---|---|
| | | | | 0: keep the same |
| DAC_MODE | 30 | 0 | R/W | Whether to reverse the output of dac_out [5:0], dac_out [5:0] are DAC's data input terminal. 1: dac_out[5:0]<= [0:5] 0: dac_out[5:0]<= [5:0] |
| DAC_BASAL | 29:24 | 011100 | R/W | Initial value of DAC input data in pre-sending phase |
| TRX_TIME | 23:21 | 011 | R/W | The time interval between the open loop of a PLL and the beginning of the launch of the data, time length calculation: TRX_TIME×8+7.5, the unit is us. |
| EX_PA_TIME | 20:16 | 00111 | R/W | The time interval between transmitting the data and the open loop of a PLL , time length calculation: EX_PA_TIME×16, the unit is us. |
| TX_SETUP_TIME | 15:11 | 01101 | R/W | The time interval between transmitting PA enable and the open loop of a PLL , time length calculation: TX_SETUP_TIME×16, the unit is us. |
| RX_SETUP_TIME | 10:6 | 10100 | R/W | The stabilization time of RX RF channel PLL, time length calculation: RX_SETUP_TIME × 16, the unit is us. |
| RX_ACK_TIME | 5:0 | 001010 | R/W | The maximum time that the PTX waits for an ACK after it has switched to receive mode, and the transmission fails if over that time. Time length calculation in 2Mbps mode: RX_ACK_TIME×16, the unit is us. Time length calculation in 1Mbps mode: RX_ACK_TIME×32, the unit is us. Time length calculation in 250kbps mode: RX_ACK_TIME×128, the unit is us. |

Note1: The 0X1B and 0X1F registers configured to their default value can work. 0X19, 0X1A and 0X1E registers need to be configured.

Note2: When accessing multibyte registers/addresses/data, the read/write sequence is low byte first, then followed by the high byte. High bit is in the front and Low bit in the back inside a single byte.

# 9 Packet Format Description

## 9.1 Packet Format for Normal BURST

Table 9-1 Packet format for Normal BURST

| Preamble (3 bytes) | Address (3~5 bytes) | Payload (1~32/64 bytes) | CRC (0/2 bytes) |
|---|---|---|---|

## 9.2 Packet Format for Enhanced BURST

Table 9-2 Packet format for Enhanced BURST

| Preamble (3 byte) | Address (3 byte) | Package control field (10bit) | | | Payload (0~32/64 byte) | CRC (0/2 byte) |
|---|---|---|---|---|---|---|
| | | Payload length (7bit) | PID (2bit) | NO_ACK (1bit) | | |

## 9.3 Packet Format of ACK for Enhanced BURST

Table 9-3 Packet format of ACK for Enhanced BURST

| Preamble (3 byte) | Address (3~5 byte) | Package control field (10bit) | | | CRC (0/2 byte) |
|---|---|---|---|---|---|
| | | Payload length (7bit) | PID (2bit) | NO_ACK (1bit) | |

# 10 MCU Register

## 10.1 General Description

- Memory
  - Universal RAM: 176×8Bit
  - OTP: 4K×16Bit
- 8-level stack buffer
- Simple and practical instruction (68 instructions)
- High-precision 12-bit ADC
- Internal low-voltage detection circuit
- Interrupt source
  - 3 timer interrupts
  - RB port level change interrupt
  - Other peripheral interrupt
- PWM module
- Operating voltage range:
  - 2.2V ~ 3.3V
- Operating temperature range: -40℃ ~ 85℃
- Internal 8MHz RC oscillation
- Instruction cycle (single instruction or double instruction)
- Built-in WDT timer
- Lookup function
- Timer:
  - 8-bit timer: TIMER0, TIMER2
  - 16-bit timer: TIMER1

## 10.2 System Structure Diagram
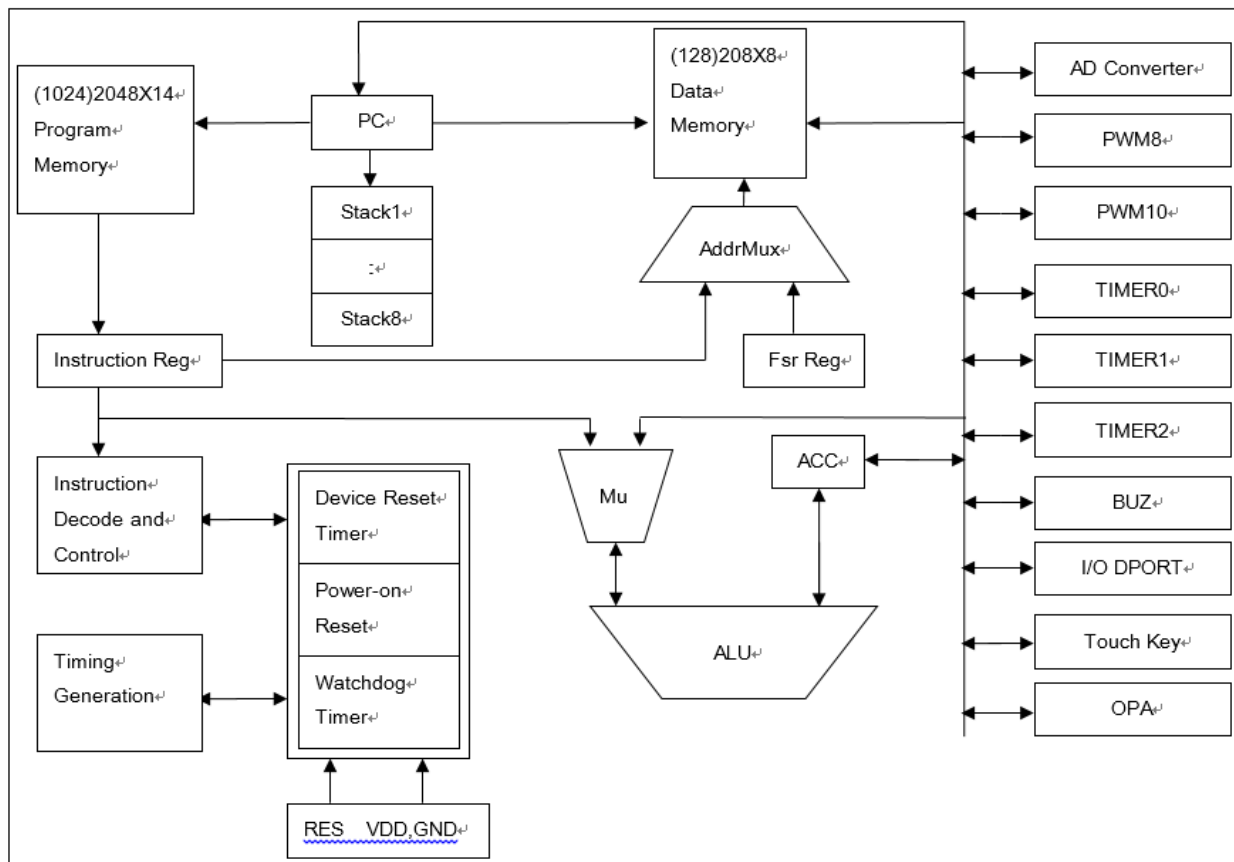


Figure 10-1 System Block Diagram

## 10.3 System Configuration Register

The system configuration register is the OTP option for the initial MCU condition. It can only be burned writer, users can not access and operation.

It contains the following:

1) OSC

  – INTRC: Internal RC oscillation

2) WDT

  – ENABLE

  – DISABLE

3) PROTECT

  – ENABLE

- DISABLE

4) LVR_SEL

   - 1.8V

   - 2.5V

## 10.4  In-Circuit Serial Programming

PAN2416AV chip is programmable in the designated application circuit over the following 5 lines:

- Power line

- Ground line

- Data line

- Clock line

- Program voltage line

It allows user to assemble circuit boards by using unprogrammed MCU and program MCU before delivery. Thus the latest version of program can be written in MCU.
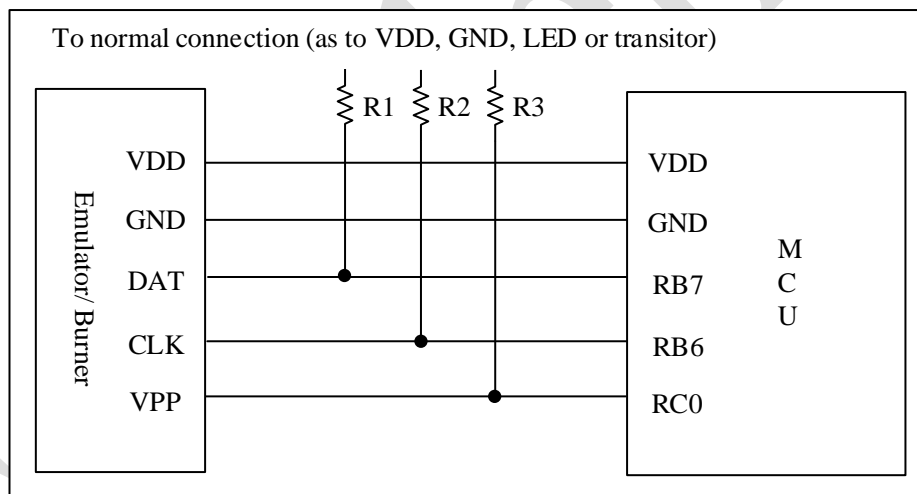


Figure 10-2 Typical In-circuit Serial Programming Method

In the above figure, R1, R2, and R3 are electric isolation devices. Usually they are resistors and the value is as below: R1≥4.7K, R2≥4.7K, R≥30K.

# 11   CPU

## 11.1   Internal Memory

### 11.1.1 Program Internal Memory

OTP:4K
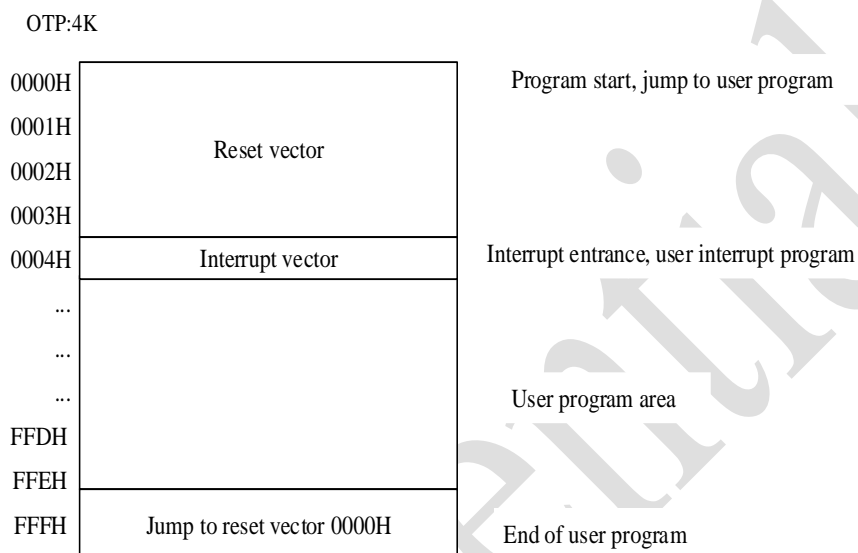


Figure 11-1 Chip program memory space

#### 11.1.1.1 Reset Vector (0000H)

The microcontroller has a word length system reset vector (0000H). Has the following three reset methods:

- Power on reset
- Watch dog reset
- Low voltage reset (LVR)

After any of the above resets occur, the program restart from 0000H and the system registers will all be reset to their default values. According to STATUS register PD and TO flag can determine the system reset mode. The following program demonstrates how to define the reset vector in ROM.

Example: Define the reset vector

```
        ORG        0000H              ;System reset vector
        JP         START
        ORG        0010H              ;User program start
START:
        …                             ;User program
        …
        END                           ;The procedure is over
```

### 11.1.1.2 Interrupt Vector

Address of interrupt vector is 0004H. Once interrupt is responded, current value of program counter (PC) will be stored into stack buffer and jump to 0004H, then start to execute interrupt service routine. All interrupts start from interrupt vector 0004H and which interrupt will be executed is determined by the value of interrupt request flag register (INT_FLAG). Example below shows how to write interrupt service routine.

Example: Define the interrupt vector, interrupt program on the user program

```
            ORG         0000H               ;System reset vector
            JP          START
            ORG         0004H               ; User program start
INT_START:
            CALL        PUSH                ; Save ACC and STATUS
            …                               ; User interrupt program
            …
INT_BACK:
            CALL        POP                 ; Return ACC and STATUS
            RETI                            ; Interrupt return
START:
            …                               ; User program
            …
            END                             ; The procedure is over
```

Notes: Since the microcontroller does not provide special stack instructions, the user needs to protect the site.

Example: Interrupt access protection site

```
PUSH:
            LD          ACC_BAK,A           ;Save ACC to register ACC_BAK
            SWAPA       STATUS              ;Status register high and low High and low half - byte inter-
                                            change
            LD          STATUS_BAK,A        ;Save to custom register STATUS_BAK
            RET                             ;Return
```

Example: Interrupt export recovery site

```
POP:
            SWAPA       STATUS_BAK          ;The High and low half-byte interchange data of ACC_BAK as-
                                            sign to ACC
            LD          STATUS,A            ;Assign ACC value to the status register STATUS
            SWAPR       ACC_BAK             ; The data of ACC_BAK High and low half - byte interchange
            SWAPA       ACC_BAK             ; The High and low half - byte interchange data of ACC_BAK
                                            assign to ACC
            RET                             ;Return
```

**11.1.1.3 Look-up Table**

The chip has the function of lookup table, and any address of ROM space can be used as a lookup table.

Related instructions:

−    TABLE [R]: Send the low byte of the table contents to register R, and the high byte to register TABLE_DATAH.

−    TABLEA: Send the low byte of the table contents to the accumulator ACC, and the high byte to the register TABLE_DATAH.

Related registers:

−    The TABLE_SPH read-write register is used to indicate the form of high 5-bit address.

−    The TABLE_SPL read-write registers is used to indicate the form of low 8-bit address.

−    The TABLE_DATAH Read-only register is store the contents of the high byte of the table.

Notes: The table address should be written to TABLE_SPH and TABLE_SPL before checking the table. If both the main program and the interrupt service program use the lookup instruction, the TABLE_SPH value in the main program may change due to the lookup instructions executed in the interrupt, resulting in an error. In other words, you should avoid using table lookup instructions in both the main program and the interrupt service routine. However, if this is necessary, we can stop the interruption before checking the table, and then open the interrupt after the table, so as to avoid mistakes.

The following example shows how to call a table in a program.

| | | |
|---|---|---|
| … | | ;Connect the user program |
| LDIA | 02H | ;Form low address |
| LD | TABLE_SPL,A | |
| LDIA | 06H | ;Form high address |
| LD | TABLE_SPH,A | |
| TABLE | R01 | ; Table instruction, the table low 8-bit (56H) of the table assign to the register R01 |
| LD | A,TABLE_DATAH | ;High 8-bit (34H) of the table to the ACC accumulator |
| LD | R02,A | ;The ACC value (34H) assign to the register R02 |
| … | | ;User program |
| | | |
| ORG | 0600H | ;Form start address |
| DW | 1234H | ;0600H address table content |
| DW | 2345H | ;0601H address table content |
| DW | 3456H | ;0602H address table contents |
| DW | 0000H | ;0603H address table content |

**11.1.1.4 Jump Table**

The jump table can perform multi-address jump function. Add PCL and ACC value can get new PCL value, so we can add PCL to different ACC to achieve multi-adress jump. If ACC value is N,

PCL+ACC means current address add N, the PCL value will be added 1 automatically after executing current instruction, as shown in the following example. If overflow occurs when PCL adds to W, PC will not carry automatically. So pay attention to it when programming. Thus user can easily to achieve multi-address jump by setting the value of W.

PCLATH is a PC high buffer register. When PCL operation is performed, PCLATH must be assigned first.

Example: correct multi-address jump program

```
OTP Address
              LDIA      01H
              LD        PCLATH,A      ; PCLATH must be assigned
              …
    0110H:    ADDR      PCL           ;ACC+PCL
    0111H:    JP        LOOP1         ;ACC=0, jump to LOOP1
    0112H:    JP        LOOP2         ;ACC=1, jump to LOOP2
    0113H:    JP        LOOP3         ;ACC=2, jump to LOOP3
    0114H:    JP        LOOP4         ;ACC=3, jump to LOOP4
    0115H:    JP        LOOP5         ;ACC=4, jump to LOOP5
    0116H:    JP        LOOP6         ;ACC=5, jump to LOOP6
```

Example: wrong multi-address jump program

```
OTP Address
              CLR       PCLATH
              …
    00FCH:    ADDR      PCL           ;ACC+PCL
    00FDH:    JP        LOOP1         ;ACC=0, Jump to LOOP1
    00FEH:    JP        LOOP2         ;ACC=1, Jump to LOOP2
    00FFH:    JP        LOOP3         ;ACC=2, Jump to LOOP3
    0100H:    JP        LOOP4         ;ACC=3, Jump to 0000H address
    0101H:    JP        LOOP5         ;ACC=4, Jump to 0001H address
    0102H:    JP        LOOP6         ;ACC=5, Jump to 0002H address
```

Notes: Because PCL overflow will not carry to high order automatically, do not put this program at page break in FLASH memery when use PCL to do multi-address jump.

## 11.1.2 Data Register

Table 11-1 Chip Data Memory List

| | Address | | Address | | Address | | Address |
|---|---|---|---|---|---|---|---|
| INDF | 00H | INDF | 80H | INDF | 100H | INDF | 180H |
| TMR0 | 01H | OPTION_REG | 81H | TMR0 | 101H | OPTION_REG | 181H |
| PCL | 02H | PCL | 82H | PCL | 102H | PCL | 182H |
| STATUS | 03H | STATUS | 83H | STATUS | 103H | STATUS | 183H |
| FSR | 04H | FSR | 84H | FSR | 104H | FSR | 184H |
| PORTA | 05H | TRISA | 85H | WDTCON | 105H | - | 185H |

| BANK0 | | BANK1 | | BANK2 | | BANK3 | |
|---|---|---|---|---|---|---|---|
| PORTB | 06H | TRISB | 86H | PORTB | 106H | TRISB | 186H |
| PORTC | 07H | TRISC | 87H | - | 107H | - | 187H |
| - | 08H | - | 88H | - | 108H | - | 188H |
| PORTE | 09H | TRISE | 89H | - | 109H | - | 189H |
| PCLATH | 0AH | PCLATH | 8AH | PCLATH | 10AH | PCLATH | 18AH |
| INTCON | 0BH | INTCON | 8BH | INTCON | 10BH | INTCON | 18BH |
| PIR1 | 0CH | PIE1 | 8CH | - | 10CH | - | 18CH |
| - | 0DH | - | 8DH | - | 10DH | - | 18DH |
| TMR1L | 0EH | - | 8EH | - | 10EH | WPUA | 18EH |
| TMR1H | 0FH | OSCCON | 8FH | - | 10FH | WPUC | 18FH |
| T1CON | 10H | OSCTUNE | 90H | TABLE_SPH | 110H | - | 190H |
| TMR2 | 11H | - | 91H | TABLE_SPL | 111H | - | 191H |
| T2CON | 12H | PR2 | 92H | TABLE_DATAH | 112H | - | 192H |
| - | 13H | - | 93H | - | 113H | - | 193H |
| - | 14H | - | 94H | - | 114H | - | 194H |
| CCPR1L | 15H | WPUB | 95H | - | 115H | - | 195H |
| CCPR1H | 16H | IOCB | 96H | - | 116H | - | 196H |
| CCP1CON | 17H | - | 97H | - | 117H | - | 197H |
| - | 18H | - | 98H | - | 118H | | 198H |
| - | 19H | - | 99H | - | 119H | | 199H |
| - | 1AH | - | 9AH | WPUE | 11AH | | 19AH |
| CCPR2L | 1BH | - | 9BH | - | 11BH | - | 19BH |
| CCPR2H | 1CH | - | 9CH | - | 11CH | | 19CH |
| CCP2CON | 1DH | - | 9DH | - | 11DH | | 19DH |
| ADRESH | 1EH | ADRESL | 9EH | - | 11EH | | 19EH |
| ADCON0 | 1FH | ADCON1 | 9FH | - | 11FH | | 19FH |
| | 20H | | A0H | | 120H | | 1A0H |
| Universal register 96Bytes | | Universal register 80Bytes | - | - | | - | |
| | 6FH | | EFH | | 16FH | | 1EFH |
| | 70H | Quick storage area70H-7FH | F0H | Quick storage area70H-7FH | 170H | Quick storage area 70H-7FH | 1F0H |
| | -- | | -- | | -- | | -- |
| | 7FH | | FFH | | 17FH | | 1FFH |

The data storage is composed of 512 x 8 bits and is divided into two functional intervals: special function registers and general data storage. Most of the data storage units are readable/written, but some are read-only. The special function register address is from 00H-1FH, 80-9FH, 100-11FH, 180-197H.

Table 11-2 MCU Special Function Register Summary Bank0

| Ad-dress | Name | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 | Reset |
|---|---|---|---|---|---|---|---|---|---|---|
| 00H | INDF | Addressing this unit uses the contents of FSR to address data memory (not a physical register) | | | | | | | | | xxxxxxxx |
| 01H | TMR0 | TIMER0 data register | | | | | | | | | xxxxxxxx |
| 02H | PCL | Program counter low byte | | | | | | | | | 00000000 |
| 03H | STATUS | IRP | RP1 | RP0 | TO | PD | Z | DC | C | 00011xxx |
| 04H | FSR | Indirect Data Memory Address Pointer | | | | | | | | | xxxxxxxx |
| 05H | PORTA | RA7 | - | RA5 | RA4 | RA3 | - | - | - | x-xxx--- |
| 06H | PORTB | RB7 | RB6 | RB5 | RB4 | RB3 | RB2 | RB1 | - | xxxxxxx- |
| 07H | PORTC | - | - | - | - | - | RC2 | RC1 | RC0 | -----xxx |
| 09H | PORTE | - | - | - | - | - | RE2 | RE1 | - | -----xx- |
| 0AH | PCLATH | - | - | - | - | Program counter high 4-bit write buffer | | | | ----0000 |
| 0BH | INTCON | GIE | PEIE | T0IE | INTE | RBIE | T0IF | INTF | RBIF | 00000000 |
| 0CH | PIR1 | - | ADIF | - | - | - | - | TMR2IF | TMR1IF | -0---000 |
| 0EH | TMR1L | 16-bit TIMER1 register low byte data register | | | | | | | | | xxxxxxxx |
| 0FH | TMR1H | 16-bit TIMER1 register high byte data register | | | | | | | | | xxxxxxxx |
| 10H | T1CON | - | - | T1CKPS1 | T1CKPS0 | - | - | - | TMR1ON | --00---0 |
| 11H | TMR2 | TIMER2 module register | | | | | | | | | 00000000 |
| 12H | T2CON | - | TOUTPS3 | TOUTPS2 | TOUTPS1 | TOUTPS0 | TMR2ON | T2CKPS1 | T2CKPS0 | -0000000 |
| 15H | CCPR1L | Low byte of PWM register 1 | | | | | | | | | xxxxxxxx |
| 16H | CCPR1H | High byte of PWM register 1 | | | | | | | | | xxxxxxxx |
| 17H | CCP1CON | - | - | DC1B1 | DC1B0 | CCP1M3 | CCP1M2 | CCP1M1 | CCP1M0 | --000000 |
| 1BH | CCPR2L | Low byte of PWM register 2. | | | | | | | | | xxxxxxxx |
| 1CH | CCPR2H | High byte of PWM register 2. | | | | | | | | | xxxxxxxx |
| 1DH | CCP2CON | - | - | DC2B1 | DC2B0 | CCP2M3 | CCP2M2 | CCP2M1 | CCP2M0 | --000000 |
| 1EH | ADRESH | High bit of the A / D result register | | | | | | | | | xxxxxxxx |
| 1FH | ADCON0 | ADCS1 | ADCS0 | CHS3 | CHS2 | CHS1 | CHS0 | GO/DONE | ADON | 00000000 |

Table 11-3 MCU Special Function Register Summary Bank1

| Address | Name | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 | Reset |
|---|---|---|---|---|---|---|---|---|---|---|
| 80H | INDF | Addressing this location uses the contents of the FSR to address data memory (not a physical register) | | | | | | | | xxxxxxxx |
| 81H | OP-TION_REG | RBPU | INTEDG | T0CS | T0SE | PSA | PS2 | PS1 | PS0 | 11111111 |
| 82H | PCL | Low byte of program counter (PC) | | | | | | | | 00000000 |
| 83H | STA-TUS | IRP | RP1 | RP0 | TO | PD | Z | DC | C | 00011xxx |
| 84H | FSR | Indirect Data Memory Address Pointer | | | | | | | | xxxxxxxx |
| 85H | TRISA | - | - | - | TRISA4 | TRISA3 | TRISA2 | - | - | ---111-- |
| 86H | TRISB | TRISB7 | TRISB6 | TRISB5 | TRISB4 | TRISB3 | TRISB2 | TRISB1 | - | 1111111- |
| 87H | TRISC | - | - | - | - | - | TRISC2 | TRISC1 | - | -----11- |
| 89H | TRISE | - | - | - | - | - | TRISE2 | TRISE1 | - | -----11- |
| 8AH | PCLATH | - | - | - | - | Program counter high 4-bit write buffer | | | | ----0000 |
| 8BH | INTCON | GIE | PEIE | T01E | INTE | RBIE | T0IF | INTF | RBIF | 00000000 |
| 8CH | PIE1 | - | ADIE | - | - | - | - | TMR2IE | TMR1IE | -0----00 |
| 8FH | OS-CCON | - | IRCF2 | IRCF1 | IRCF0 | - | - | - | SCS | -110---0 |
| 90H | OS-CTUNE | - | - | - | TUN4 | TUN3 | TUN2 | TUN1 | TUN0 | ---00000 |
| 92H | PR2 | TIMER2 period register | | | | | | | | ---11111 |
| 95H | WPUB | WPUB7 | WPUB6 | WPUB5 | WPUB4 | WPUB3 | WPUB2 | WPUB1 | - | 0000000- |
| 96H | IOCB | IOCB7 | IOCB6 | IOCB5 | IOCB4 | IOCB3 | IOCB2 | IOCB1 | - | 0000000- |
| 9EH | ADRESL | Low bit of the A/D result register | | | | | | | | xxxxxxxx |
| 9FH | AD-CON1 | ADFM | - | - | - | - | - | - | - | 0------- |

Table 11-4 MCU Special Function Register Summary Bank2

| Address | Name | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 | Reset |
|---------|------|------|------|------|------|------|------|------|------|-------|
| 100H | INDF | Addressing this location uses the contents of the FSR to address data memory (not a physical register) | | | | | | | | xxxxxxxx |
| 101H | TMR0 | TIMER0 module register | | | | | | | | xxxxxxxx |
| 102H | PCL | Low byte of program counter (PC) | | | | | | | | 00000000 |
| 103H | STATUS | IRP | RP1 | RP0 | TO | PD | Z | DC | C | 00011xxx |
| 104H | FSR | Indirect Data Memory Address Pointer | | | | | | | | xxxxxxxx |
| 105H | WDTCON | - | - | - | - | - | - | - | SWDTEN | -------0 |
| 106H | PORTB | RB7 | RB6 | RB5 | RB4 | RB3 | RB2 | RB1 | - | xxxxxxx- |
| 10AH | PCLATH | - | - | - | Program counter high 5-bit write buffer | | | | | ---00000 |
| 10BH | INTCON | GIE | PEIE | T0IE | INTE | RBIE | T0IF | INTF | RBIF | 00000000 |
| 110H | TABLE_SPH | Table high pointer | | | | | | | | ---xxxxx |
| 111H | TABLE_SPL | Table low pointer | | | | | | | | xxxxxxxx |
| 112H | TA-BLE_DATAH | Table high data | | | | | | | | xxxxxxxx |

Table 11-5 MCU Special Function Register Summary Bank3

| Address | Name | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 | Reset |
|---------|------|------|------|------|------|------|------|------|------|-------|
| 180H | INDF | Addressing this location uses the contents of the FSR to address data memory (not a physical register) | | | | | | | | xxxxxxxx |
| 181H | OP-TION_REG | RBPU | INTEDG | T0CS | T0SE | PSA | PS2 | PS1 | PS0 | 11111111 |
| 182H | PCL | Low byte of program counter (PC) | | | | | | | | 00000000 |
| 183H | STATUS | IRP | RP1 | RP0 | TO | PD | Z | DC | C | 00011xxx |
| 184H | FSR | Indirect Data Memory Address Pointer | | | | | | | | xxxxxxxx |
| 186H | TRISB | TRISB7 | TRISB6 | TRISB5 | TRISB4 | TRISB73 | TRISB2 | TRISB1 | - | 1111111- |
| 18AH | PCLATH | - | - | - | - | Program counter high 4-bit write buffer | | | | ----0000 |
| 18BH | INTCON | GIE | PEIE | T01E | INTE | RBIE | T01F | INTF | RBIF | 00000000 |

## 11.2 Addressing Method

### 11.2.1 Direct Addressing

Operate on RAM through the work register (ACC).

Example: The value of ACC is assign to the 30H register

| | |
|---|---|
| LD | 30H,A |

Example: The value of the 30H register is assign to ACC

| | |
|---|---|
| LD | A,30H |

## 11.2.2 Immediately Addressing

Assign immediate data to working register (ACC).

Example: Immediately assign to the ACC 12H

| | |
|---|---|
| LDIA | 12H |

## 11.2.3 Indirect Addressing

Data storage can be addressed directly or indirectly. It can be addressed indirectly through the INDF register, INDF is not a physical register. When accessing the INDF, it will be on the basis of FSR register values low (8) and the STATUS register of the IRP (9) as the address, and point to the address of the register, so in setting up the FSR and STATUS registers of IRP, can be as INDF registers of the destination register to access. Reading INDF indirectly (FSR=0) will generate 00H. Indirect writing to the INDF register will result in an empty operation. The following example shows the use of indirect addressing in the program.

Example: FSR and INDF applications

```
        LDIA    30H
        LD      FSR,A         ; Indirect addressing pointer points to 30H
        CLRB    STATUS,IRP    ; Bit 9 of the pointer is cleared
                              ; Clear INDF is actually clear the RAM at the 30H address
        CLR     INDF
                              pointed to by the FSR
```

Example: Indirect Addressing Clear RAM (20H-7FH)

```
        LDIA    1FH
        LD      FSR,A         ; Indirect addressing pointer points to 1FH
        CLRB    STATUS,IRP
LOOP:
        INCR    FSR           ; Address plus 1, the initial address of 30H
        CLR     INDF          ; Clear the address pointed to by the FSR
        LDIA    7FH
        SUBA    FSR
        SNZB    STATUS,C      ; Always cleared to FSR address 7FH
        JP      LOOP
```

## 11.3 Stack

The stack cache of the chip is 8 layers. The stack buffer is neither part of the data memory, nor is part of the program memory, and can neither be read nor written. The operation of it is implemented by the stack pointer (SP), and the stack pointer (SP) cannot be read or written. When the system is reset, the stack pointer points to the top of the stack. The program counter (PC) value at the time of a subroutine call and interrupt is pushed into the stack buffer, when returning from interrupt or subroutine will return to give the program counter (PC). The following figure shows how it works.
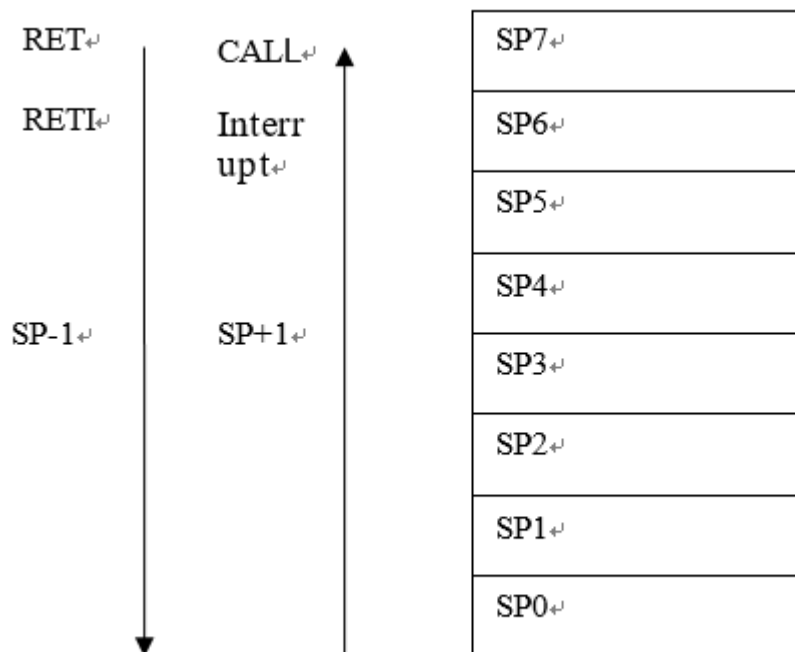
Figure 11-2 Stack working principle

Notes: The stack register has only eight layers. If the stack is full and unmasked interrupt occurs, only the interrupt flag is recorded and the interrupt response is suppressed until the stack pointer decrements, the interrupt is response. This function can prevent the interrupt stack overflow. as if the stack is full, and subroutine calls, the stack overflow, then first into the content of the stack will be lost, only the last eight return address is retained, so the user should pay attention to this point when writing programs, lest produce program broke down.

## 11.4 Working Register (ACC)

### 11.4.1 Overview

ALU is 8Bit wide arithmetic logic unit, all the mathematics and logic operations of MCU are completed by it. It can add, subtract, shift and logical operation of data; the ALU also controls the status bit (STATUS register) to indicate the status of the result.

The ACC register is an 8Bit register. The result of the ALU operation can be stored here. It is not part of the data memory but is located in the CPU for the ALU to use in its operation, therefore cannot be addressed. It can only be accessed by the supplied instruction to use.

### 11.4.2 ACC Application

Example: Data transfer with ACC

|   |   |   |
|---|---|---|
| LD | A,R01 | ; Assign the value of register R01 to ACC |
| LD | R02,A | ; Assign the value of ACC to register R02 |

Example: Use ACC for immediate addressing target operands.

| | | |
|---|---|---|
| LDIA | 30H | ; Assign 30H to ACC |
| | | ; The value of the current ACC with the immediate number |
| ANDIA | 30H | 30H "and" operation, |
| | | ; Results into the ACC |
| | | ; The value of the current ACC with the immediate number |
| XORIA | 30H | 30H XOR operation, |
| | | ; Results into the ACC |

Example: Use ACC to do the first operand of a double operand instruction

| | | |
|---|---|---|
| HSUBA | R01 | ;ACC-R01, result into the ACC |
| HSUBR | R01 | ;ACC-R01, result into the R01 |

Example: Use ACC to do the second operand of a double operand instruction

| | | |
|---|---|---|
| SUBA | R01 | ;R01-ACC, result into the ACC |
| SUBR | R01 | ;R01-ACC, result into the R01 |

## 11.5   Program Status Register (STATUS)

The STATUS register is shown in the following table, including:

− The arithmetic state of the ALU

− Reset status

− Memory Area Select bits of data memory (GPR and SFR)

Like other registers, the STATUS register can be the target of any instruction. If an instruction that affects Z, DC, or C bits is used as the target register with the STATUS register, the three STATUS bits cannot be written. These bits are set or cleared according to the chip logic. And you cannot write TO and PD bit. Therefore, the instruction that uses STATUS as the target register may not get the expected result.

For example, the CLR STATUS will be cleared high 3 bits and set the Z bit to 1. The value of STATUS will be 000uu1uu (where u=unchanged). Therefore, it is recommended that only the CLRB, SETB, SWAPA, SWAPR instructions be used to change the STATUS registers, because these instructions do not affect any STATUS bits.

Program Status Register STATUS (03H)

| 03H | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|
| STATUS | IRP | RP1 | RP0 | TO | PD | Z | DC | C |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 1 | 1 | X | X | X |

Bit7    IRP: Register storage area selection (for indirect addressing)

1=Bank2 and Bank3 (100h-1FFh)

0=Bank0 and Bank1 (00h-FFh)

Bit6-5    RP [1:0]: Storage location selection

0 0 Select Bank 0

0 1 Select Bank 1

1 0 Select Bank 2

1 1 Select Bank 3

Bit4    TO: Timeout

1= Power or execute the CLRWDT instruction or STOP instruction

0= The WDT timeout occurred.

Bit3    PD: Power down bit

1= Power up or execute the CLRWDT instruction

0= execute the STOP instruction

Bit2    Z: The result is zero

1= The result of an arithmetic or logical operation is zero

0= The result of an arithmetic or logical operation is not zero

Bit1    DC: Semi-carry / borrow

1= The 4th lowest result of the carry is higher

0= The 4th low of the result did not carry to the top

Bit0    C: Carry / borrow

1= The most significant bit of the result is a carry

0= The most significant bit of the result did not occur

TO and PD marks can reflect the reason of chip reset, and the following list of events affecting TO, PD and the status of TO and PD after various reset.

Table 11-6 Impact PD, TO's event table

| Event | TO | PD |
|-------|----|----|
| Power up | 1 | 1 |
| WDT overflow | 0 | X |
| STOP instruction | 1 | 0 |
| CLRWDT instruction | 1 | 1 |
| Sleep | 1 | 0 |

Table 11-7 TO / PD status after reset

| TO | PD | Reset Reason |
|----|----|--------------|
| 0 | 0 | WDT Time-out Wake-up MCU |
| 0 | 1 | WDT overflow non-sleeping state |
| - | - | - |
| 1 | 1 | Power up |

## 11.6  Prescaler (OPTION_REG)

The OPTION_REG register is a read-write register, including various control bits for configuration:

－   TIMER0/WDT Prescaler

－   TIMER0

– PORTB Pull-up resistor control

Prescaler control register OPTION_REG (81H)

| 81H | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|
| OPTION_REG | RBPU | INTEDG | T0CS | T0SE | PSA | PS2 | PS1 | PS0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

Bit7　　　RBPU: PORTB Pull-up enable bit

　　　　　1= PORTB pull up is disabled

　　　　　0= PORTB pull up are enabled by the latch values for the port

Bit6　　　INTEDG: The edge of the trigger interrupt

　　　　　1= The rising edge of the INT pin triggers an interrupt

　　　　　0= The falling edge of the INT pin triggers an interrupt

Bit5　　　T0CS: TIMER0 clock source select bit

　　　　　1= T0CKI pin input

　　　　　0= Internal Instruction Cycle Clock (FOSC / 4)

Bit4　　　T0SE: TIMER0 Clock source edge select bit

　　　　　1= Increment on T0CKI pin transition from high to low

　　　　　0= Increment on T0CKI pin transition from low to high

Bit3　　　PSA: Prescaler allocation bit.

　　　　　1= Prescaler assigned to WDT

　　　　　0= Prescaler assigned to the TIMER0 module

Bit2-0　　PS2～PS0: Prescaler parameter configuration bit.

| PS2-PS1-PS0 | TIMER0 Divide ratio | WDT Divide ratio |
|---|---|---|
| 000 | 1: 2 | 1: 1 |
| 001 | 1: 4 | 1: 2 |
| 010 | 1: 8 | 1: 4 |
| 011 | 1: 16 | 1: 8 |
| 100 | 1: 32 | 1: 16 |
| 101 | 1: 64 | 1: 32 |
| 110 | 1: 128 | 1: 64 |
| 111 | 1: 256 | 1: 128 |

The prescaler register is actually an 8bit counter that is used to monitor the WDT register as a post-scaler. When used as a timer/counter, it is commonly referred to as a prescaler. There is only one physical divider in the film, which can only be used for both WDT/TIMER0 and cannot be used at the same time. In other words, if it is used for TIMER0, WDT cannot use the prescaler, and vice versa.

When used in WDT, the CLRWDT command will reset both the prescaler and WDT timer.

When used for TIMER0, all instructions for writing TIMER0 (e.g. CLR TMR0, SETB TMR0, 1, etc.) will clear the prescaler.

The prescaler is used by TIMER0 or WDT, which is completely controlled by software and can be dynamically changed. In order to avoid the possibility of a chip reset, the following instructions

should be executed when changing from TIMER0 to WDT.

| | | |
|---|---|---|
| CLR | TMR0 | ;TMR0 Clear |
| CLRWDT | | ;WDT Clear |
| LDIA | B'00xx1111' | |
| LD | OPTION_REG,A | |
| LDIA | B'00xx1xxx' | ; |
| LD | OPTION_REG,A | |

To switch the predivider from the assigned WDT to the TIMER0 module, the following instructions should be executed.

| | | |
|---|---|---|
| CLRWDT | | ;WDT Clear |
| LDIA | B'00xx0xxx' | ; |
| LD | OPTION_REG,A | |

Notes: To get a 1:1 prescaler configuration for TIMER0, assign the prescaler to the WDT by setting the PSA bit in the option register.

## 11.7   Program Counter (PC)

The program counter (PC) controls the order of instruction execution in OTP. It can address the entire range of OTP. After obtaining the instruction code, the PC automatically increments the address of the next instruction code. But if perform jumps to the photo, conditional jumps, assignment, subroutine calls, initialize the reset, interrupt, interrupt return, returned to operations such as subroutine, the PC will load instruction related address instead of the next instruction.

When subjected to a conditional jump instruction and conform to the jump condition, read the instruction execution process of the next instruction will be discarded, and insert an empty instruction operation cycle, then the right orders can be achieved. Otherwise, the next instruction is executed in sequence.

Program counter (PC) is 12-Bit width, and the lower 8 bits can be accessed by the PCL (02H) register, but the high 4 users cannot be accessed. Can hold 4Kx16 bit program address. Assigning a PCL will result in a short jump and the jump range is 256 addresses of the current page.

Notes: When the user is using PCL for short jump, it first needs to assign PC high buffer register PCLATH.

The following are the PC values for several special cases.

| Reset | PC=0000 |
|---|---|
| Interrupt | PC=0004 (The original PC+1 will be automatically pushed onto the stack) |
| CALL | PC= Program specified address (the original PC + 1 will be automatically pushed onto the stack) |
| RET, RETI, RETi | PC= Stack out of the value |
| Operate PCL | PC[11:8] constant, PC[7:0]= User-specified value |
| JP | PC= Program specified value |
| Other instructions | PC=PC+1 |

## 11.8  Watchdog Counter (WDT)

The Watch Dog Timer is a self-oscillating RC oscillator Timer that doesn't require any peripheral components, and even if the chip's main clock stops working, the WDT can keep the clock ticking. The WDT timer overflow will result in a reset.

### 11.8.1  WDT Cycle

The WDT and TIMER0 share an 8bit prescaler. After all resets, the WDT overflow cycle is 18ms, and if you need to change the WDT cycle, you can set the OPTION_REG register. The overflow cycle of WDT will be affected by environmental temperature, power supply voltage and other parameters.

The "CLRWDT" and "STOP" instructions will clear the WDT timer and the count in the predivider (when the predivider is assigned to WDT). WDT is generally used to prevent system out of control, or can be said to be used to prevent the microcontroller program out of control. Under normal conditions, the WDT should be cleared by the "CLRWDT" instruction before it overflows to prevent a reset. If the program is out of control due to some kind of interference, the "CLRWDT" instruction cannot be executed before the WDT overflow, which will cause the WDT overflow to generate a reset. To restart the system without losing control. If the WDT overflow generates a reset, the "TO" position of the STATUS register is cleared, and the user can determine if the reset is caused by the WDT overflow.

Notes:

1)  If you use the WDT function, you must place the "CLRWDT" command somewhere in the program to ensure that it is cleared before the WDT overflow. Otherwise, the chip will continue to reset, resulting in the system does not work properly.

2)  The WDT cannot be cleared in the interrupt program, otherwise the main program "running" cannot be detected.

3)  In the program, there should be a clear WDT operation in the main program, and try not to clear WDT in multiple branches, which can maximize the protection function of the watchdog counter.

4)  There is a certain difference between the overflow time of the watchdog counter for different chips, so when the WDT time is set, the overflow time of WDT should have a large redundancy, so as to avoid unnecessary WDT reset.

### 11.8.2  Watchdog Timer Control Register WDTCON

Watchdog timer control register WDTCON (105H)

| 105H | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|------|------|------|------|------|------|------|------|------|
| WDTCON | - | - | - | - | - | - | - | SWDTEN |
| R/W | - | - | - | - | - | - | - | R/W |
| Reset | - | - | - | - | - | - | - | 0 |

Bit7-1          Reserve, Read as 0

Bit0              SWDTEN: Software enable or disable watchdog timer bit

1=Enable

0=Disable

Notes: If the WDT configuration bit =1 in CONFIG, the WDT is always enabled, regardless of the state of the control bit of SWDTEN. If the WDT configuration bit =0 in CONFIG, you can use SWDTEN control to enable or disable WDT.

# 12 System Clock

## 12.1 Overview

The clock signal is generated by internal oscillations, and four non-overlapping orthogonal clock signals are produced in the chip, which are called Q1, Q2, Q3 and Q4 respectively. In IC, each Q1 makes the program counter (PC) incremented by one. Q4 takes the instruction from the program storage unit and locks it in the instruction register. Decoding and executing the extracted instructions between the next Q1 and Q4, which means that four clock cycles will execute one instruction. The following figure shows the timing diagram of the clock and instruction cycle.

An instruction cycle contains 4 Q cycle, instruction execution and access is to use an assembly line structure, refers to occupy one instruction cycle, the decoding and execution to take up another instruction cycle, but because of pipeline structure, from a macro point of view, effective execution time of each instruction is an instruction cycle. If an instruction causes the program counter address changes (for example, JP) then prefetch instruction operation code is invalid, two instruction cycle is required to complete this order, this is for PC operating instructions are to take up the cause of the two clock cycles.
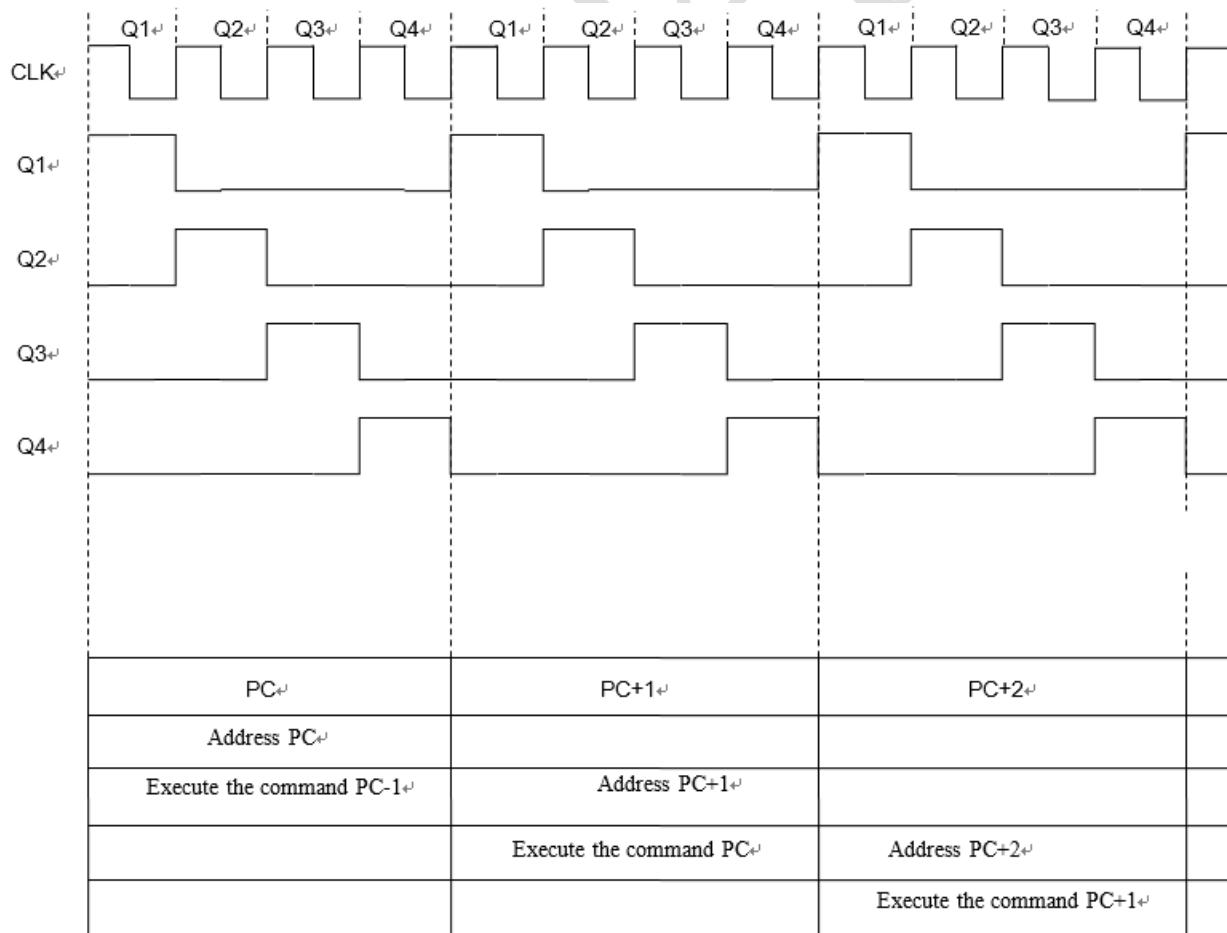


Figure 12-1 Clock and instruction cycle timing diagram

The following lists the relationship between the frequency of oscillations and the speed of instruction.

| Frequency | Double instruction cycle | Single instruction cycle |
|-----------|--------------------------|--------------------------|
| 1MHz | 8μs | 4μs |
| 2MHz | 4μs | 2μs |
| 4MHz | 2μs | 1μs |
| 8MHz | 1μs | 500ns |

## 12.2   System Oscillator

The chip has one kinds of oscillation mode, internal RC oscillation.

The default oscillation mode of the chip is internal RC oscillation whose oscillation frequency is 8MHz. The operating frequency of the chip can be set by the OSCCON register. The oscillation frequency is factory calibrated with an error of within ± 3%.

## 12.3   Start-up Time

Reset Time refers to the time from chip reset to chip oscillation stability, and its design value is 18ms.

Notes: Whether the chip is powered on reset, or other causes reset, there will be this start-up time.

## 12.4   Oscillator Control Register

Oscillator Control (OSCCON) Register Control System Clock and Frequency Selection, Oscillator Tuning Register OSCTUNE can be used to adjust the internal oscillation frequency in software.

Oscillator control register OSCCON (8FH)

| 8FH | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|-----|------|------|------|------|------|------|------|------|
| OSCCON | - | IRCF2 | IRCF1 | IRCF0 | - | - | - | SCS |
| R/W | - | R/W | R/W | R/W | - | - | - | R/W |
| Reset | - | 1 | 1 | 0 | - | - | - | 0 |

    Bit7          Reserve, read as 0

    Bit6-4         IRCF<2:0>: Internal oscillator frequency select bit

                      111=8MHz

                      110=4MHz (Default)

                      101=2MHz

                      100=1MHz

                      011=500kHz

                      010=250kHz

                      001=125kHz

                      000=32kHz (LFINTOSC)

    Bit3-Bit1      Reserve

Bit0          SCS: System clock selection bit

1= The internal oscillator is used as the system clock

0= The clock source is defined by CONFIG

Oscillator adjustment register OSCTUNE (90H)

| 90H | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|
| OSCTUNE | - | - | - | TUN4 | TUN3 | TUN2 | TUN1 | TUN0 |
| R/W | - | - | - | R/W | R/W | R/W | R/W | R/W |
| Reset | - | - | - | 0 | 0 | 0 | 0 | 0 |

Bit7-5          Reserve

Bit4-0          TUN<4:0>: Frequency adjustment bit

01111= The highest frequency

01110=

.

.

.

00001=

00000= The oscillator module runs after the manufacturer's calibration

11111=

.

.

.

10000= Lowest frequency

# 13 Reset

The chip has the following three kinds of reset methods:

- Power up reset

- LVR reset

- Watch dog overflow reset during normal operation

When any of the above reset occurs, all the system registers will reset to the default state, the program will stop running, at the same time, the program counter will be cleared. After reset, the program will run from reset vector 0000H. The TO and PD flags of STATUS provide information on the system reset status (see STATUS for details). The user can control the program run path according to the status of PD and TO.

Any kind of reset situation needs a certain response time, and the system provides a complete reset process to ensure the smooth reset action.

## 13.1 Power up Reset

Power up reset is closely related to LVR operation. System power-up process is a gradual curve, and will take some time to reach the normal level. The following shows the normal power up reset timing:

- Power on: the system detects the voltage rise and waits for its stability

- System Initialization: All system registers are set to their initial values

- Oscillator starts working: Oscillator starts to supply system clock

- Executive procedures: power up, the program began to run

## 13.2 Power off Reset

### 13.2.1 Power off Reset Overview

The power off reset is for system voltage dropping condition caused by external factors (e.g. disturbing or changing of external loading). When external rest is used, the power off reset might cause system abnormal or program running error. The voltage dropping may cause system falling into dead band which means power supply cannot satisfied the minimum working voltage.
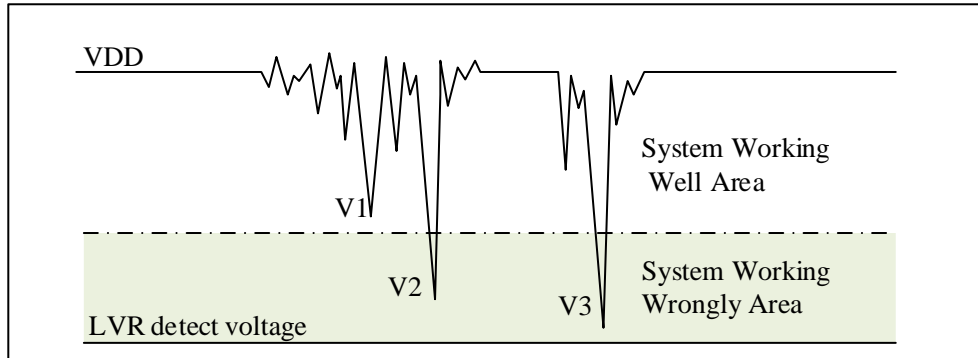
Figure 13-1 Power off Reset Diagram

The diagram above is a typical power off reset diagram. In this diagram, VDD is a seriously interferred, and voltage drops a lot. The system works well in the area above the dash line, and system comes into unknown working status in the area below the dash line, this area is called dead band. When VDD drops to V1, system is in normal working condition still. However, once VDD drops to V2 and V3, system comes into dead band and system error easily occurs.

In following conditions, the system may come to dead band:

1) DC application:

The power source of DC application is usually battery. When battery voltage is too low or MCU drives loading, system voltage drops and keeps in dead band. At this point, system voltage won't drop to LVD, system remains in dead band.

2) AC application:

In AC power application, DC voltage is affected by AC noise. When external loading is heavy, like driving motor, the loading action will affect DC power. When VDD drops under the minimum voltage by disturbing, the system may become unstable.

The power on duration and power off duration are relatively long in AC application. The power sequence protection ensures system working properly, but the power off situation is like DC application. When AC power is turned off, VDD voltage drops slowly and it is easy to get into dead band.

Like showing in Figure 13-1, the working voltage is higher than system reset voltage. And reset voltage is determined by low voltage detect level (LVR). When system execution speed rises, the system lowest working voltage will rise accordingly. As the system reset voltage is fixed, there is a voltage gap between system lowest working voltage and system reset voltage. In this voltage gap, system can't work nor reset. This area is dead band.

## 13.2.2 Improving Methods of Power off Reset

There are some advises on how to improve the performance of system power off reset:

- Choosing a higher LVR voltage helps reset more reliably
- Enable watchdog timer
- Reduce the system's operating frequency
- Increase the voltage drop slope
- Watchdog timer

The watchdog timer is used to ensure the normal operation of the program. When the system enters the working dead zone or the program running is in error, the watchdog timer will overflow and the system reset.

1) Reduce the system's work speed

The faster the system working frequency, the higher the minimum operating voltage of the sys tem. Thereby increasing the scope of the work dead zone, reducing system operating speed can reduce the minimum operating voltage, thus effectively reducing the probability of the system working in the dead zone voltage.

2) Increase the voltage drop slope

This method can be used for the system to work in the AC power supply environment, the general AC power supply system, the system voltage drops slowly during power-down, which will cause the chip to work for a long time in the dead zone voltage, at this time if the system power up, the working status of the chip may be wrong. It is suggested to add a discharging resistor between the power supply of the chip and the ground so that the MCU can quickly pass the dead zone and enter the reset area to avoid the possibility error of the chip power on.

## 13.3 Watchdog Reset

Watchdog reset is a system protection setting. In normal condition, watchdog timer is cleared by program. If error happens, system is in unknown status. Watchdog timer will overflow and system will reset. After watchdog resets, system restarts and returns to normal status.

Watchdog reset time sequence is as following:

1) Watchdog timer status: The system detects whether the watchdog timer overflow, if overflow, the system reset;

2) System initialization: All system registers are set to default status;

3) Oscillator start up: Oscillator starts to provide system clock;

4) Program executing: Reset is finished and program starts to run.

# 14  Sleep Mode

## 14.1  Enter Sleep Mode

Execute STOP instruction to enter sleep mode, if the WDT is enable, then:

1) WDT will be cleared and continue.

2) The PD bit of STATUS register will be cleared.

3) TO bit will be set.

4) Turn off the oscillator driver.

5) The I/O port maintains the state before the STOP command (driven to high level, low level or high resistance).

In sleep mode, all I/O pins should be held at VDD or GND in order to minimize current consumption. In order to avoid the input pin floating to introduce switch current, the high impedance I/O pin should be pulled high or low externally. In order to minimize current consumption, you should also consider the impact of the chip pull up resistor.

## 14.2  Wakeup from Sleep Mode

The device can be woken up from sleep mode by any of the following events:

1) Watch dog timer wake up (WDT enable).

2) PORTB level change interrupt and peripheral interrupt.

Both of these events are considered as the continuations of program execution, and the TO and PD bits in the STATUS register are used to determine the cause of device reset. The PD bit is set on power up and cleared on execution of the STOP instruction. The TO bit is cleared when WDT wake up.

When stop instruction is executed, the next instruction is fetched in advance. If the device is to be awakened by an interrupt event, the corresponding interrupt must be allowed in position 1 (allowed). Wake up have nothing to do with the status of the GIE bit. If the GIE bit is cleared, the device will continue executing the instruction following the STOP instruction. If the GIE bit is set the device executes the instruction after the STOP instruction, and then jumps to the interrupt address (0004h) to execute the code. If you do not want to execute the instruction following the STOP instruction, you should place a NOP instruction after the STOP instruction. When the device waked up from sleep mode, the WDT will be cleared regardless of the cause of wake up.

## 14.3  Wake Up with Interrupt

When a global interrupt is disabled (GIE is cleared), any interrupt source has its interrupt enable bit and interrupt flag set, one of the following events will occur:

1) If an interrupt is produced before the STOP instruction is executed, the STOP instruction is executed as an NOP instruction. Therefore, WDT and its prescaler and the postscaler (if enabled) will not be cleared, and the TO bit will not be set TO 1, and PD will not be cleared.

2) If an interrupt is generated during or after the STOP instruction, the device will be immediately wake up from sleep mode. The STOP instruction will be executed before waking up. Therefore, The WDT, its prescaler and postscaler will be cleared, and the TO bit will be set and the PD will be cleared. Even before executing the STOP command, check the flag bit for 0, and it may be set to 1 before the STOP instruction is executed. To determine whether the STOP command is executed, you can test the PD bit. If the PD bit is set, the STOP instruction is executed as a NOP instruction. Before executing the STOP instruction, a CLRWDT instruction must be executed to ensure that the WDT is cleared.

## 14.4 Example of Sleep Mode Application

Before system turns into SLEEP mode, if small sleep current is required, user should check all I/O status firstly. If there are unused I/O ports in user's application, every unused port need to be setted as output port to ensure every input port has a fixed state. This way can prevent sleep current increase caused by uncertain state of port level when I/O ports are input ones. Turning off AD module and comparator module can help to reduce sleep current. According to different requirements of actual application, WDT function can be disabled to further decrease sleep current.

Example: Program of getting into SLEEP mode

```
SLEEP_MO
    DE:
        CLR         INTCON          ;Disable interrupt;
        LDIA        B'00000000'
        LD          TRISA,A
        LD          TRISB,A         ;All I/O configured as output;
        LD          TRISC,A
        LD          TRISE,A
        …                           ;Turn off the other function；
        LDIA        0A5H
        LD          SP_FLAG,A       ;Set sleep mode memory register (user define);
        CLRWDT                      ;Clear WDT；
        STOP                        ;Executed STOP instruction.
```

## 14.5 Sleep Mode Wakeup Time

When the MCU wakes up from sleep mode, wait for an oscillation stabilization time (Reset Time), which is nominally 18ms.

# 15   I/O Port

Chip has four types of I/O ports: PORT0, PORT1 and PORT2. The read/write port data register can access these ports directly.

Table 15-1 Overall port configuration

| Port | Bit | Pin description | Input/ Output |
|------|-----|-----------------|---------------|
| PORTB | 1 | Schmitt trigger input, push-pull output, AN10 input | I/O |
| | 2 | Schmitt trigger input, push-pull output, AN8 input | I/O |
| | 3 | Schmitt trigger input, push-pull output, AN9 input | I/O |
| | 4 | Schmitt trigger input, push-pull output, AN11 input | I/O |
| | 5 | Schmitt trigger input, push-pull output, AN13 input | I/O |
| | 6 | Schmitt trigger input, push-pull output, Programming clock input | I/O |
| | 7 | Schmitt trigger input, push-pull output, Programming data input/output | I/O |
| PORTC | 0 | Schmitt trigger input, NMOS open drain output | I/O |
| | 1 | Schmitt trigger input, push-pull output, CCP2 | I/O |
| | 2 | Schmitt trigger input, push-pull output, CCP1 | I/O |
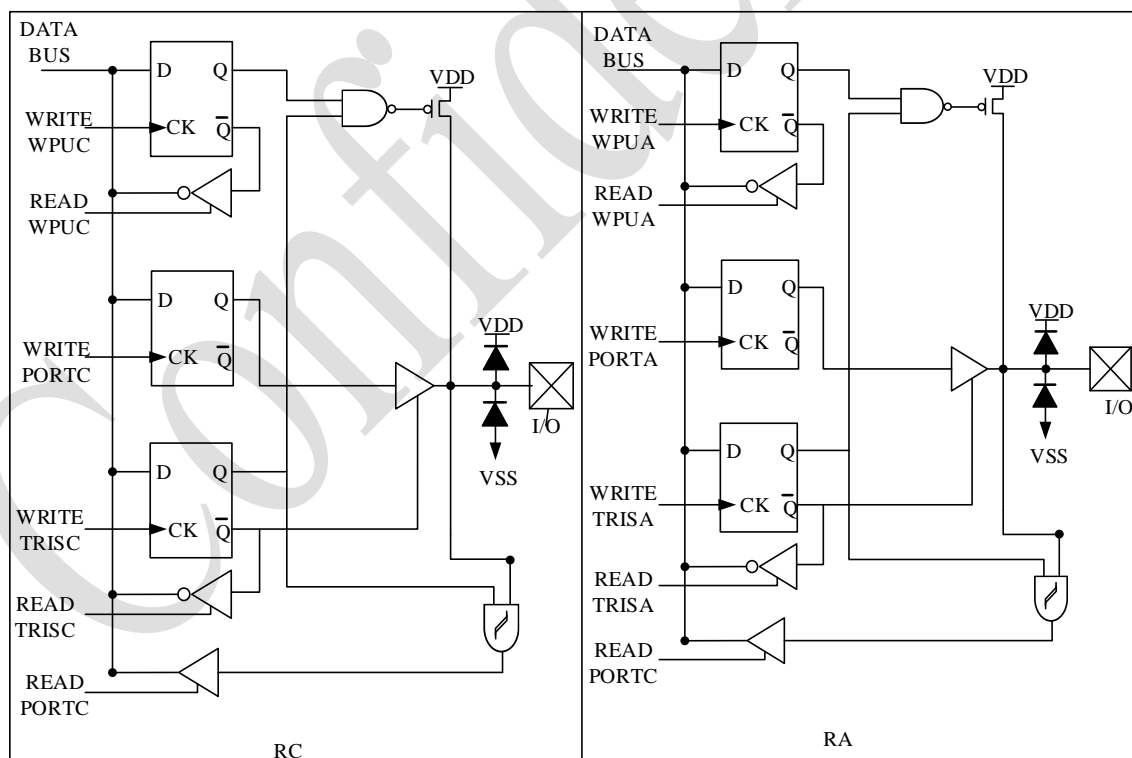


Figure 15-1 RA/RC Block diagram
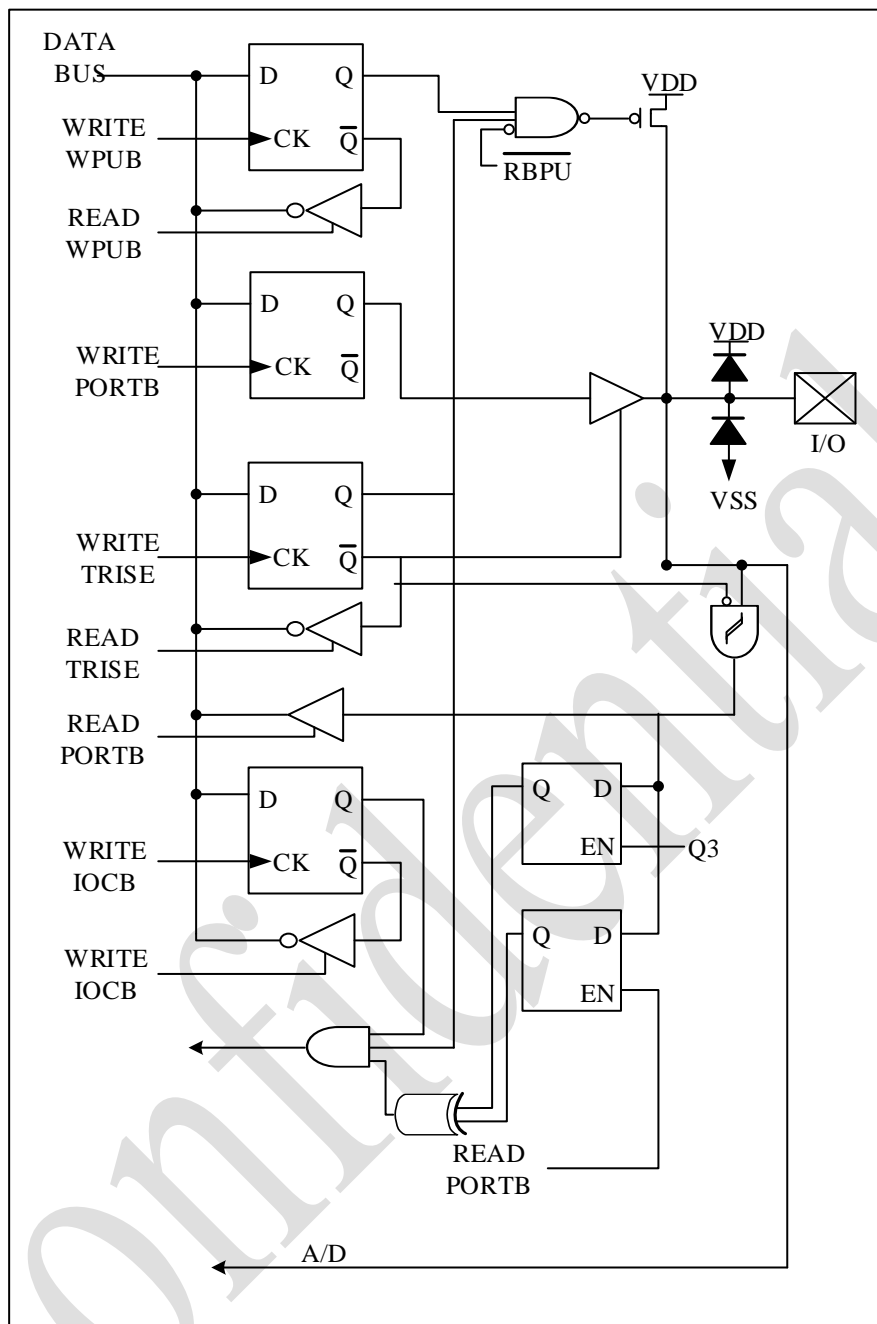
Figure 15-2 RB Port Block Diagram

## 15.1  PORTA

PORTA is a 4 bit I/O port. The corresponding data direction register is TRISA. Setting a bit of TRISA configures the corresponding pin as input. Clearing a bit of TRISA configures the corresponding pin as output.

Reading the PORTA register is read the state of the pin and writing to this register will write to the port latch. All write operation is read-modify-write operation. Therefore, writing a port means reading the port's pin level, modifying the read value, and then writing the changed value to the port data latch. The RA7 and RA5 ports must be set as input.

The register relate to PORTA is PORTA and TRISA.

PORTA data register PORTA (05H)

| 05H | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|-----|------|------|------|------|------|------|------|------|
| PORTA | RA7 | - | RA5 | RA4 | RA3 | - | - | - |
| R/W | R/W | - | R/W | R/W | R/W | - | - | - |
| Reset | x | - | x | x | x | - | - | - |

    Bit7          PORTA<7>: PORTA I/O Pin location
                    1= level>VIH
                    0= level<VIL
    Bit5-3       PORTA<5:3>: PORTA I/O Pin location
                    1= level>VIH
                    0= level<VIL

PORTA Direction control register TRISA (85H)

| 85H | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|-----|------|------|------|------|------|------|------|------|
| TRISA | TRISA7 | - | TRISA5 | TRISA4 | TRISA3 | - | - | - |
| R/W | R/W | - | R/W | R/W | R/W | - | - | - |
| Reset | 1 | - | 1 | 1 | 1 | - | - | - |

    Bit7          TRISA<7>: PORTA Tri state control bit
                    1= Input
                    0= Output
    Bit5-3       TRISA<5:3>: PORTA Tri state control bit
                    1= Input
                    0= Output

Example: PORTA hander

```
LDIA     B'11110000'    ;Set PORTA<3:0> as output and PORTA<7:4> as input
LD       TRISA,A
LDIA     03H            ;PORTA<1:0> output is high, PORTA<3:2 > output is low
LD       PORTA,A        ;Since PORTA<7:4> is input port, neither 0 nor 1 is af-
                        fected
```

## 15.2  PORTB

### 15.2.1  PORTB Data and Control

PORTB is a 7-bit wide bidirectional port. The corresponding data direction register is PORTB. Setting a bit in TRISB make the corresponding PORTB pin as input. Clearing a bit in TRISB will make the corresponding PORTB pin as output.

Reading the PORTA register is read the state of the pin and writing to this register will write to the

port latch. All write operation is read-modify-write operation. Therefore, writing a port means reading the port's pin level, modifying the read value, and then writing the changed value to the port data latch.

Registers relate to PORTB are PORTB, TRISB, WPUB, IOCB and so on.

PORTB data register PORTB (06H)

| 06H | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|
| PORTB | RB7 | RB6 | RB5 | RB4 | RB3 | RB2 | RB1 | - |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | - |
| Reset | x | x | x | x | x | x | x | - |

Bit7-1        PORTB<7:1>: PORTBI/O Pin location
               1= Level>VIH
               0= Level<VIL


PORTB direction control register TRISB (86H)

| 86H | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|
| TRISB | TRISB7 | TRISB6 | TRISB5 | TRISB4 | TRISB3 | TRISB2 | TRISB1 | - |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | - |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | - |


Bit7-1          TRISB<7:1>: PORTB Tri state control bit
                1=Input (Tri state )
                0=Output


Example: PORTB hander

```
CLR   PORTB                    ;Clear data register
LDIA   B'00110000'            ;Set PORTB<5:4> as input and others as output
LD    TRISB,A
```

## 15.2.2  PORTB Pull Up Resistor

Each PORTB pin has an internal weak pull-up that can be configured separately. Control bit WPUB<7:1> to enable or disable each weak pull-up. When the port pin is configured as output, its weak pull will be automatically cut off. When the power is reset, the RBPU bit of the OPTION_REG register is not allowed.

PORTB pull up resistor register WPUB (95H)

| 95H | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|
| WPUB | WPUB7 | WPUB6 | WPUB5 | WPUB4 | WPUB3 | WPUB2 | WPUB1 | - |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | - |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | - |

Bit7-1          WPUB<7:1>: Weak pull up register bit
                1=Enable
                0=Disable

Notes: To enable either of these pull ups individually, RBPU bit of OPTION_REG must be cleared.

### 15.2.3 PORTB Level Change Interrupt

All PORTB pins can be individually configured as level change interrupt. The control bit IOCB<7:0> allows or prohibits the interrupt function of each pin. After power on reset, it is forbidden pin level change interrupt function.

For pin that have enable level change interrupt, the value on the pin is compared to the old value that was locked when reading PORTB last time. If mismatch, the level change interrupt flag bit in the INTCON register will set.

This interrupt can wake up the device from sleep mode. User can clear the interrupt in the interrupt service routine:

1)  Read and write operation on PORTB. This will end the pin level mismatch.

2)  Clear bit flag RBIF.

The mismatch status keeps the RBIF flag set. Reading or writing to PORTB will end the mismatch condition and allow the RBIF flag to be cleared. The latch will hold the value of the last read without the effect of the under voltage reset. After the reset, if the mismatch persists, the RBIF flag bit will continue to be 1.

Notes: If the level of I/O pins changes when the read operation is performed (the beginning of the Q2 cycle), the RBIF interrupt flag bit will not be set to 1. In addition, because the reading or writing of the port affects all the bits of the port, it is important to be careful when using multiple pins in the level change interrupt mode. When dealing with a pin level change, you may not notice the level change on the other pin.

PORTB level change interrupt register IOCB (96H)

| 96H | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|-----|------|------|------|------|------|------|------|------|
| IOCB | IOCB7 | IOCB6 | IOCB5 | IOCB4 | IOCB3 | IOCB2 | IOCB1 | - |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | - |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | - |

Bit7-1          IOCB<7:1>: PORTB level change interrupt control bit
                1=Enable
                0=Disable

## 15.3   PORTC

### 15.3.1 PORTC Data and Control

PORTC is a 3-bit wide bidirectional port. The corresponding data direction register is TRISC. Setting a bit in TRISC makes the corresponding PORTC pin an input. Clearing a bit in TRISC makes

the corresponding PORTC pin as an output.

Reading the PORTC register is read the state of the pin and writing to this register will write to the port latch. All write operation is read-modify-write operation. Therefore, writing a port means reading the port's pin level, modifying the read value, and then writing the changed value to the port data latch.

PORTC data register PORTC (07H)

| 07H | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|------|------|------|------|------|------|------|------|------|
| PORTC | - | - | - | - | - | RC2 | RC1 | RC0 |
| R/W | - | - | - | - | - | R/W | R/W | R/W |
| Reset | - | - | - | - | - | x | x | x |

Bit2-0        PORTC<2:0>: PORTC I/O Pin location
              1=Port pin level>VIH
              0=Port pin level<VIL

PORTC Direction register TRISC (87H)

| 87H | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|------|------|------|------|------|------|------|------|------|
| TRISC | - | - | - | - | - | TRISC2 | TRISC1 | TRISC0 |
| R/W | - | - | - | - | - | R/W | R/W | R/W |
| Reset | - | - | - | - | - | 1 | 1 | 1 |

Bit2-0        TRISC<2:0>: PORTC Tri state control bit
              1=PORTC pin configured as input (Tri state )
              0=PORTC Pin configured as output

Notes: When RC0 configured as output, only open-drain output function.

Example: PORTC hander.

```
CLR    PORTC              ;Clear date register
LDIA   B'01110000'        ;Set PORTC<3:0> as output
LD     TRISC,A
```

## 15.3.2 PORTC Pull Up Resistor

Each PORTC pin has an internal weak pull-up that can be configured separately. Control bit WPUC<7:0> to enable or disable each weak pull-up. When the port pin is configured as output, its weak pull will be automatically cut off.

PORTC Pull-up resistor register WPUC (18FH)

| 18FH | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|------|------|------|------|------|------|------|------|------|
| WPUC | - | - | - | - | - | WPUC2 | WPUC1 | - |
| R/W | - | - | - | - | - | R/W | R/W | - |

| Reset | - | - | - | - | - | 0 | 0 | - |

Bit2-1          WPUC<2:1>: Weak pull-up register bit
                1=Enable
                0=Disable
Notes: Weak pull-up is automatically disabled if the pin is configured as an output.

## 15.4   PORTE

PORTE is a 2-bit wide bidirectional port. The corresponding data direction register is TRISE. Setting a bit in TRISE makes the corresponding PORTE pin input. Clearing a bit in TRISC makes the corresponding PORTE pin as output.

Reading the PORTE register is read the state of the pin and writing to this register will write to the port latch. All write operation is read-modify-write operation. Therefore, writing a port means reading the port's pin level, modifying the read value, and then writing the changed value to the port data latch. The RA7 and RA5 ports must be set as input.

PORTE data register PORTE (09H)

| 09H | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|------|------|------|------|------|------|------|------|------|
| PORTE | - | - | - | - | - | RE2 | RE1 | - |
| R/W | - | - | - | - | - | R/W | R/W | - |
| Reset | - | - | - | - | - | x | x | - |

Bit2-1          PORTE<2:1>: PORTE I/O Pin location
                1=Port pin level>VIH
                0=Port pin level<VIL

PORTE direction register TRISE (89H)

| 87H | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|------|------|------|------|------|------|------|------|------|
| TRISE | - | - | - | - | - | TRISE2 | TRISE1 | - |
| R/W | - | - | - | - | - | R/W | R/W | - |
| Reset | - | - | - | - | - | 1 | 1 | - |

Bit2-1          TRISE<2:1>: PORTE Tri state control bit
                1=PORTE pin configured as input
                0=PORTE pin configured as output

## 15.5   I/O

### 15.5.1  Write I/O Port

The I/O port register of the chip, like the general purpose register, can be written by data transfer instruction, bit operation instruction and so on.

Example: Write I/O Port

| | | |
|---|---|---|
| LD | PORTA,A | ;ACC value assign to PORTA |
| CLRB | PORTB,1 | ;PORTB.1 clear |
| CLR | PORTC | ;PORTC clear |
| SET | PORTA | ;PORTA all port set |
| SETB | PORTB,1 | ;PORTB.1 set |

### 15.5.2  Read I/O port

Example: Read I/O Port

| | | |
|---|---|---|
| LD | A,PORTA | ;PORTA value assign to ACC |
| SNZB | PORTA,1 | ;If PORTA,1 was 1 then skip the next statement |
| SZB | PORTA,1 | ;If PORTA,1 was 0 then skip the next statement |

Notes: When user read I/O status, if the I/O port is an input port, the data read by the use will be the external level of the port. If the I/O is an output port, then read the value of the port will be the output data of the internal register.

### 15.5.3  I/O Usage Note

In operation I/O, the following aspects should be noted:

When I/O is converted from output to input, wait for several instruction cycles so that I/O port state is stable.

If the internal pull-up register is used, when the I/O is switched from output to input the internal level stabilization time is related to the capacitance connect to the I/O port. The use should set the wait time according to the actual situation to prevent scan error.

When the I/O port is the input port, its input level shall be between "VDD+ 0.7v" and "gnd-0.7 V". If the input voltage is not in this range, the following figure can be used.
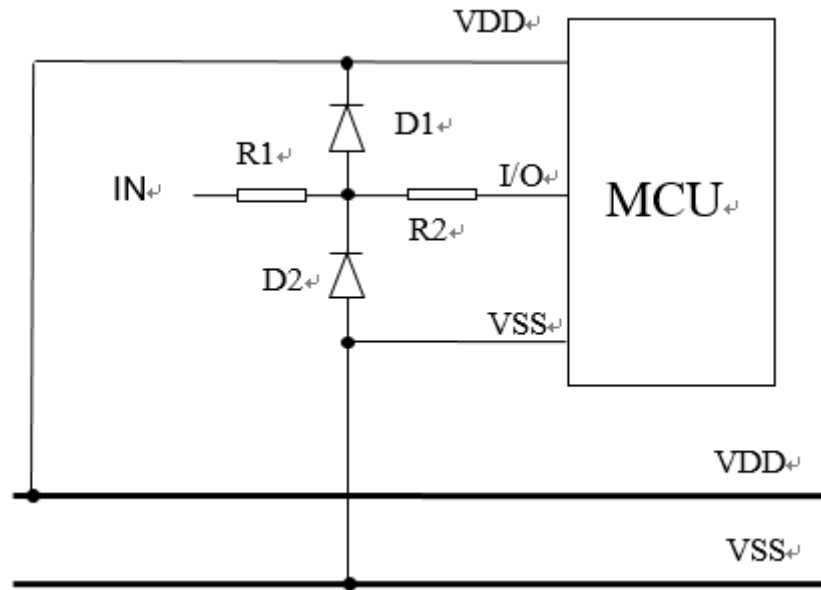
Figure 15-3 I / O ESD protection diagram

If a long connection line is connected online at the I/O port, please add a current-limiting resistor near the chip I/O to enhance MCU's anti-emc capability.

# 16 Interrupt

## 16.1 Interrupt Overview

The chip has the following kinds of interrupt sources.

− TIMER0 overflow interrupt

− TIMER1 overflow interrupt

− TIMER2 matching interrupt

− INT interrupt

− PORTB voltage change interrupt

− A/D interrupt

The interrupt control register (INTCON) and the peripheral interrupt request register (PIR1) record various interrupt requests in their respective flag bits. The INTCON register also includes each interrupt allowed bit and global interrupt allow bit.

Global interrupts allow bit GIE (INTCON<7>) to allow all unblocked interrupts when set 1, and all interrupts are forbidden at zero time. All interrupts can be prohibited by the corresponding allowable bits in the INTCON and PIE1 registers.

The execution "return from interrupt" command will exit the interrupt service program and place GIE position 1, thereby allow unblocked interrupts.

Interrupt execution diagram shown in Figure 16-1.
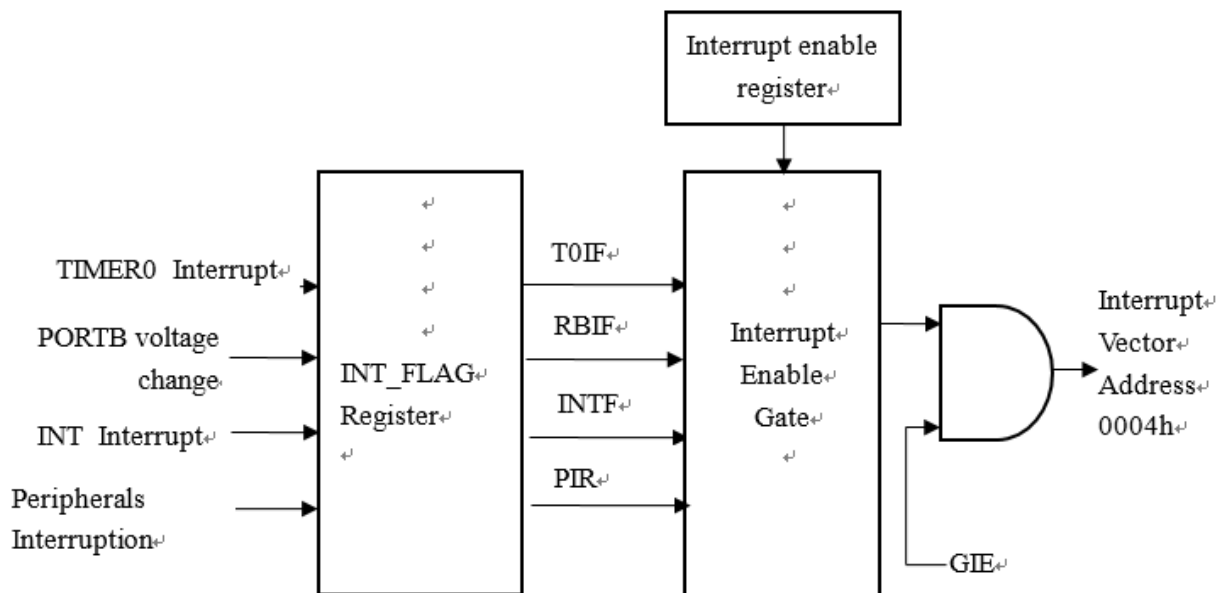


Figure 16-1 Interrupt execution diagram

## 16.2   Interrupt Control Register

### 16.2.1  Interrupt Control Register

The interrupt control register, INTCON, is a read-write register containing the allowable and mark bits of the TMR0 register overflow, the PORTB port level change interrupt, and so on.

When an interrupt condition occurs, the interrupt flag will be set regardless of the state of the corresponding interrupt enable bit or the global enable bit. The user software should ensure that the corresponding interrupt flag bit is cleared before allowing an interrupt.

Interrupt control register INTCON (0BH)

| 0BH | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|
| INTCON | GIE | PEIE | T0IE | INTE | RBIE | T0IF | INTF | RBIF |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Bit7      GIE: Global interrupt enable bit
              1=Enable
              0=Disable

Bit6      PEIE: Peripheral interrupt enable bit
              1=Enable
              0=Disable

Bit5      T0IE: TIMER0 overflow interrupt enable bit
              1=Enable
              0=Disable

Bit4      INTE: External interrupt enable bit
              1=Enable
              0=Disable

Bit3      RBIE: PORTB voltage change interrupt enable bit (1)
              1=Enable
              0=Disable

Bit2      T0IF: TIMER0 Overflow interrupt flag bit (2)
              1=TMR0 register overflow (Must be cleared by software )
              0=TMR0 register do not overflow

Bit1      INTF: External interrupt flag bit
              1= Interrupt occur (must be cleared by software)
              0=Not occur

Bit0      RBIF: PORTB voltage change interrupt flag
               1= The state of at least one pin in PORTB changed (must be cleared in software)
              0= None of PORTB I/O change

Notes:

1)   IOCB register must be enable, the corresponding port must set as input.

2) The T0IF bit is set when TMR0 rolls over to 0. The reset does not change TMR0, it should be initialized before the T0IF bit is cleared.

## 16.2.2 Peripheral Interrupt Enable Register

Peripheral Interrupt Enable Register PIE1. PEIF bit of INTCON register must be set before allowing any peripheral interrupt.

Peripheral interrupt Enable register PIE1 (8CH)

| 8CH | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|------|------|------|------|------|------|------|--------|--------|
| PIE1 | - | ADIE | - | - | - | - | TMR2IE | TMR1IE |
| R/W | - | R/W | - | - | - | - | R/W | R/W |
| Reset | - | 0 | - | - | - | - | 0 | 0 |

|       |                                                        |
|-------|--------------------------------------------------------|
| Bit7  | Reserve                                                |
| Bit6  | ADIE: ADC interrupt enable                             |
|       | 1=Enable                                               |
|       | 0=Disable                                              |
| Bit5-2 | Reserve                                               |
| Bit1  | TMR2IE: TIMER2 and PR2 match interrupt enable bit      |
|       | 1=Enable                                               |
|       | 0=Disable                                              |
| Bit0  | TMR1IE: TIMER1 Overflow interrupt enable bit           |
|       | 1=Enable                                               |
|       | 0=Disable                                              |

## 16.2.3 Peripheral Interrupt Request Register

The peripheral interrupt request register is PIR1 and PIR2. When an interrupt condition occurs, the interrupt flag bit will be set to 1 regardless of whether the corresponding interrupt allows bit or globally allowed bit status. The user software should ensure that the corresponding interrupt flag bit is cleared before allowing an interrupt.

Peripheral interrupt request register PIR1 (0CH)

| 0CH | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|------|------|------|------|------|------|------|--------|--------|
| PIR1 | - | ADIF | - | - | - | - | TMR2IF | TMR1IF |
| R/W | - | R/W | - | - | - | - | R/W | R/W |
| Reset | - | 0 | - | - | - | - | 0 | 0 |

|       |                                                        |
|-------|--------------------------------------------------------|
| Bit7  | Reserve                                                |
| Bit6  | ADIF: A/D converter interrupt bit                      |
|       | 1=A/D convent complete (Must be cleared by software )  |
|       | 0= A / D conversion is incomplete or has not yet started |
| Bit5-2 | Reserve                                               |
| Bit1  | TMR2IF: TIMER2 and PR2 match interrupt flag            |

1=Match (Must be cleared by software )

0=Mismatch

Bit0     TMR1IF: TIMER1 overflow interrupt flag

1=TMR1 register overflow (Must be cleared by software )

0=TMR1 Register did not overflow

## 16.3 Interrupt Protection Method

After the interrupt request occurs and is responded, the program moves to 0004H to execute the interrupt subroutine. The content of ACC and STATUS must be saved before the response is interrupted. The chip does not provide a dedicated warehousing and stack recovery instruction, and the user needs to protect the content of ACC and STATUS in order to avoid running errors after the interruption.

Example: ACC and STATUS stack protection

```
                ORG      0000H
                JP       START              ;Start address
                ORG      0004H
                JP       INT_SERVICE        ; Interrupt service routine
                ORG      0008H
        START:
                …
                …
    INT_SERVICE:
        PUSH:                               ;Interrupt service routine entrance, save ACC and STATUS
                LD       ACC_BAK,A          ;Save ACC value
                SWAPA    STATUS
                LD       STATUS_BAK,A       ;Save STATUS value
                …
                …
        POP:                                ;Interrupt service routine exit, restore ACC and status
                SWAPA    STATUS_BAK
                LD       STATUS,A           ;Restore STATUS value
                SWAPR    ACC_BAK            ;Restore ACC value
                SWAPA    ACC_BAK
                RETI
```

## 16.4 Interrupt Priority, and More Interrupt Nesting

The priority of the chip's interrupts is equal, and when an interrupt is in progress, it does not respond to another interrupt, and only when the "RETI" command is executed can it respond to the next interrupt.

MCU has no preset interrupt priority when multiple interrupts occur simultaneously. First of all, the

priority of each interrupt must be set in advance. Second, the control system responds to the interrupt by using interrupts to enable the power and interrupt control bits. In the program, the interrupt control bit and interrupt request flag must be detected.

# 17    TIMER0

## 17.1    TIMER0 Overview

The TIMER0 is composed of the following functions:

−    8-bit timer/counter register (TMR0).

−    8-bit prescaler (Shared with watchdog timer).

−    Programmable internal or external clock sources.

−    Programmable external clock edge selection.

−    Overflow interrupt.

Figure 17-1 TIMER0 Block Diagram.
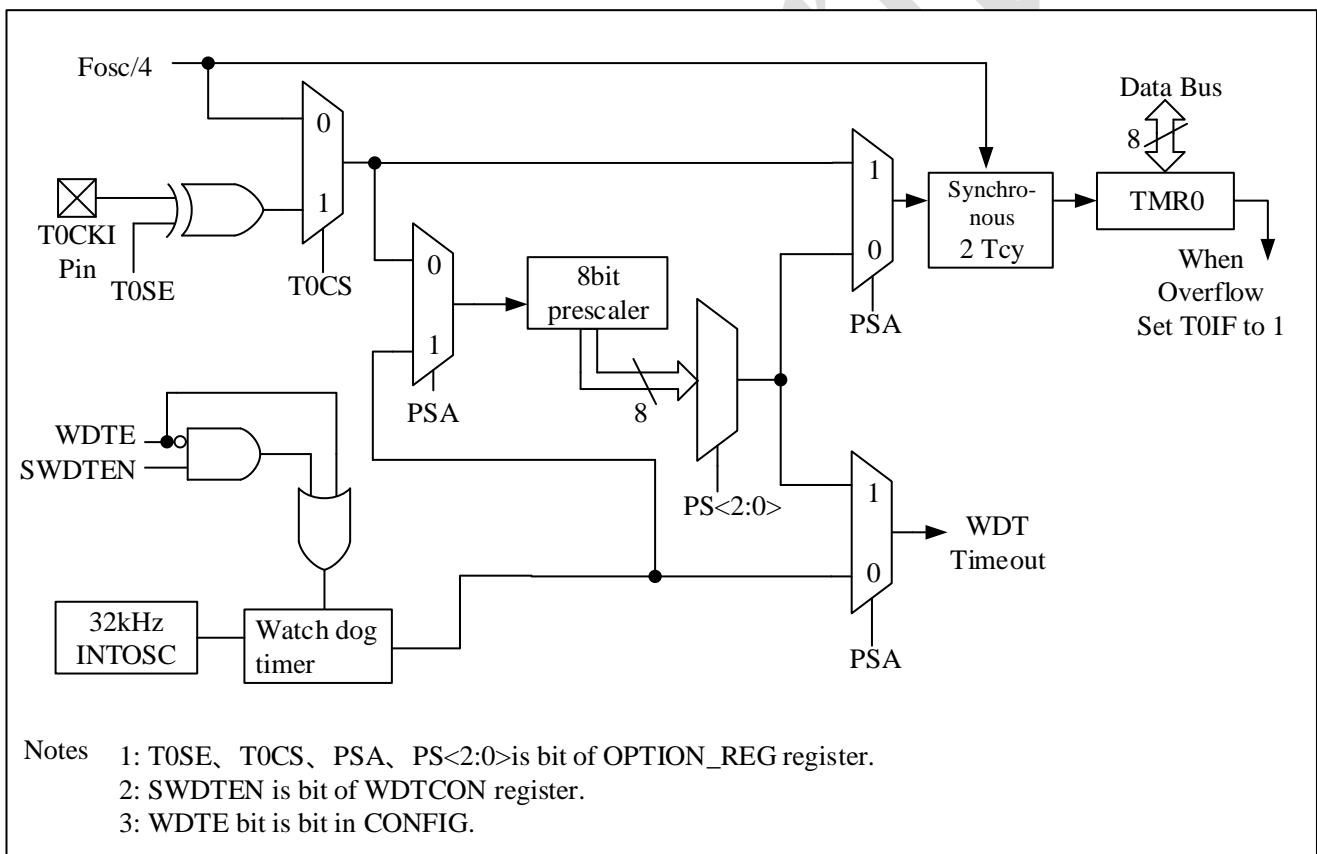


Figure 17-1 TIMER0/WDT Structure Diagram

## 17.2 TIMER0 Working Principle

The TIMER0 module can be used as an 8-bit timer or an 8-bit counter.

### 17.2.1 8-Bit Timer Mode

When used as a timer, the TIMER0 module will be incremented at each instruction cycle (without the prescaler). Select the timer mode by using the T0CS bit of the OPTION_REG register. If a write operation is performed on the TMR0 register, increments will be prohibited in the next two instruction cycles. Can be adjusted to the value of the TMR0 register, allowing the delay of two instruction cycles when TMR0 is written.

### 17.2.2 8-Bit Counter Mode

When used as a counter, The TIMER0 module will increase along each rising edge or falling edge of the T0CKI pin. The increasing edge depends on the T0SE bit of the OPTION_REG register. Select the counter mode by using the T0CS position 1 of the OPTION_REG register.

### 17.2.3 Software Programmer Prescaler

TIMER0 and the watchdog timer (WDT) share a software programmable predivider, but not at the same time. The distribution of the predivider is controlled by the PSA position of the OPTION_REG register. To assign the predivider to TIMER0, the PSA must be cleared.

TIMER0 module has 8 divider ratio choices, range from 1:2 to 1:256. The divider ratio can be selected through the PS<2:0> bit of the OPTION_REG register. To make the TIMER0 module have a 1:1 divider ratio, the prescaler must be assigned to the WDT module.

The predivider cannot read and write, when the prescaler is allocated to the TIMER0 module, all instructions written to the TMR0 register will clear the prescaler. When the predivider is assigned to WDT, the CLRWDT command will simultaneously clear the prescaler and WDT.

### 17.2.4 Switch the Predivider between the TIMER0 and WDT Modules

When the prescaler is allocated to TIMER0 or WDT, unintentional device reset may occur when switching prescaler. To change the prescaler from TIMER0 to the WDT module, you must perform the sequence of instructions as shown in example 18-1.

Example 18-1 Reallocate Prescaler (TMR0-WDT)

```
CLR        TMR0            ;TMR0 Clear
CLRWDT                     ;WDT Clear
LDIA       B'00xx1111'
LD         OPTION_REG,A
LDIA       B'00xx1xxx'     ; Reallocate
LD         OPTION_REG,A
```

Reallocate prescaler from WDT to the TIMER0 module, the following instructions should be executed.

| CLRWDT | | ;WDT Clear |
|--------|--------|-----------|

| LDIA | B'00xx0xxx' | ;Reallocate |
|------|-------------|-------------|
| LD | OPTION_REG,A | |

### 17.2.5 TIMER0 Interrupt

When the TMR0 register overflows from FFh to 00h, TIMER0 is interrupted. Every time the TMR0 register overflows, whether the TIMER0 interrupt is allowed or not, the T0IF interrupt flag position of the INTCON register is set to 1. The T0IF bit must be cleared in the software. The TIMER0 interrupt allows the T0IE bit of the INTCON register.

Notes: Because the timer is closed during hibernation, the TIMER0 interrupt does not wake the processor.

## 17.3   TIMER0 Related Register

There are two register relate to TMR0, 8-bit timer/counter (TMR0), 8-bit programmable control register (OPTION_REG).

TRM is an 8-bit read-write timer/counter, OPTION_REG is an 8-bit write only register. User can change the value of OPTION_REG to change the working mode of TMR0. Please refer to the application of the prescaler register (OPTION_REG).

8-bit timer/counter TMR0 (01H)

| 01H | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|-----|------|------|------|------|------|------|------|------|
| TMR0 | - | - | - | - | - | - | - | - |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | X | X | X | X | X | X | X | X |

OPTION_REG Register (81H)

| 81H | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|-----|------|------|------|------|------|------|------|------|
| OPTION_REG | RBPU | INTEDG | T0CS | T0SE | PSA | PS2 | PS1 | PS0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

Bit7        RBPU: PORTB Pull up enable bit
            1=Disable PORTB Pull up
            0= The various locking values of the port enable the ability to pull on the PORTB.
Bit6        INTEDG: Interrupt edge selection
            1= The rising edge of an INT trigger interrupts.
            0= The falling edge of an INT trigger interrupts.
Bit5        T0CS: TIMER Clock source selection bit
            1=T0CKI External clock input
            0=Internal instruction cycle clock (FOSC/4)

Bit4        T0SE: TIMER0 Clock edge select bit

                1=Falling edge

                0=Rising edge

Bit3        PSA: Prescaler assigned bit

                1= Prescaler assigned to WDT

                0= Prescaler assigned to TIMER0

Bit2-0      PS2～PS0: Prescaler parameter configuration bit

| PS2-PS1-PS0 | TMR0 divide ratio | WDT Divide ratio |
|---|---|---|
| 000 | 1: 2 | 1: 1 |
| 001 | 1: 4 | 1: 2 |
| 010 | 1: 8 | 1: 4 |
| 011 | 1: 16 | 1: 8 |
| 100 | 1: 32 | 1: 16 |
| 101 | 1: 64 | 1: 32 |
| 110 | 1: 128 | 1: 64 |
| 111 | 1: 256 | 1: 128 |

# 18 TIMER1

## 18.1 TIMER1 Overview

TIMER1 is a 16-bit timer/counter with the following feature:

− 16-bit timer/counter register (TMR1H:TMR1L)

− 3-bit prescaler

− Overflow interrupt

− Overflow wakeup (External clock asynchronous mode only)

− Capture/compare function of the time base

− Special event trigger function (with the ECCP)
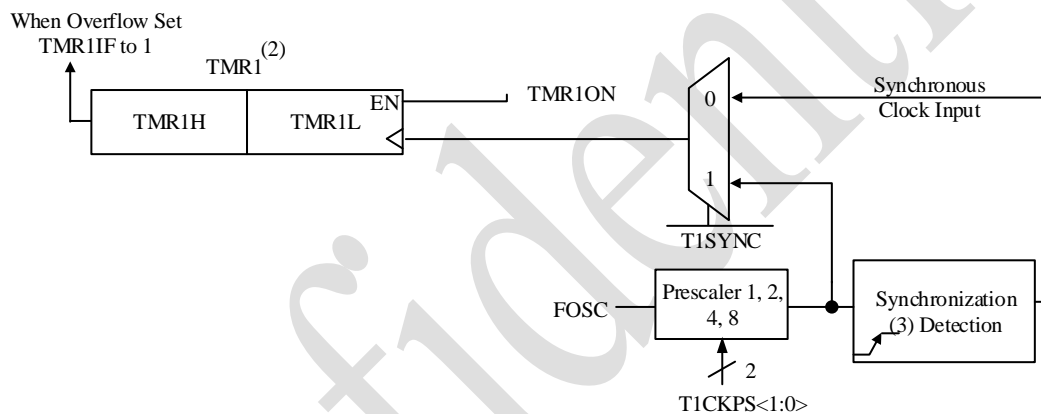


Figure 18-1 TIMER1 Block Diagram

## 18.2 TIMER1 Working Principle

TIMER1 is a 16-bit increment counter accessed by a pair of registers TMR1H:TMR1L. The counter can be updated directly by THR1H or TMR1L.

When TIMER1 works, TMR1H:TMR1L register will be incremented by the multiple of FOSC, and the specific multiple is determined by the TIMER1 prescaler.

## 18.3 TIMER1 Prescaler

TIMER1 has four prescaler option, allow 1, 2, 4 or 8 divide of the input clock. T1CKPS bit of the T1CON register control the prescaler counter. The prescaler counter cannot be read or writen directly.

## 18.4 TIMER1 Interrupt

After TIMER1 register (TMR1H: TMR1L) increase to FFFFh, it will overflow to 0000h. When the TIMER1 overflows, the TIMER1 interrupt flag bit of the PIR1 register is set to 1. To allow the overflow to be interrupted, the user should take the following position:

1) TIMER1 interrupt enable bit in the PIE1 register

2) The PEIE bit of the INTCON register

3) The GIE bit of the INTCON registers

The interrupt can be cleared by clearing the TMR1IF bit in the interrupt service routine.

Notes: Before enabling the interrupt again, clear the TMR1H:TMR1L register and the TMR1IF bit. The timer interrupt cannot wake up the processor because the timer is off during sleep mode.

## 18.5 TIMER1 Related Register

TIMER1 is controlled mainly by three register: control register T1CON, data register TMR1L, TMR1H. The data register must assign low (TMR1L) bit first.

TIMER1 Low bits of data Register TMR1L (0EH)

| 0EH | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|-----|------|------|------|------|------|------|------|------|
| TMR1L | - | - | - | - | - | - | - | - |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | x | x | x | x | x | x | x | x |

TIMER1 High bits of data register TMR1H (0FH)

| 0FH | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|-----|------|------|------|------|------|------|------|------|
| TMR1H | - | - | - | - | - | - | - | - |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | x | x | x | x | x | x | x | x |

TIMER1 Control register T1CON (10H)

| 10H | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|-----|------|------|------|------|------|------|------|------|
| T1CON | - | - | T1CKPS1 | T1CKPS0 | - | - | - | TMR1ON |
| R/W | - | - | R/W | R/W | - | - | - | R/W |
| Reset | - | - | 0 | 0 | - | - | - | 0 |

Bit 7      Reserve

Bit 6      Reserve    Must be 0

Bit 5-4      T1CKPS<1:0>: TIMER1 Input clock prescaler selection bit

           11 = 1:8

           10 = 1:4

           01 = 1:2

           00 = 1:1

Bit3      Reserve    Must be 0

Bit2      Reserve

Bit1      Reserve    Must be 0

Bit0      TMR1ON: TIMER1 Enable bit

           1 = Enable TIMER1

               0    = Disable TIMER1

# 19 TIMER2

## 19.1 TIMER2 Overview

The TIMER2 module is an 8-bit timer / counter with the following features:

− 8-bit timer register (TMR2)

− 8-bit period register (PR2)

− Interrupt on TMR2 match with PR2

− Software programmable prescaler (1:1, 1:4 and 1:16)

− Software programmable postscaler (1:1 to 1:16)

The block diagram of TIMER2 is shown in Figure 19-1。
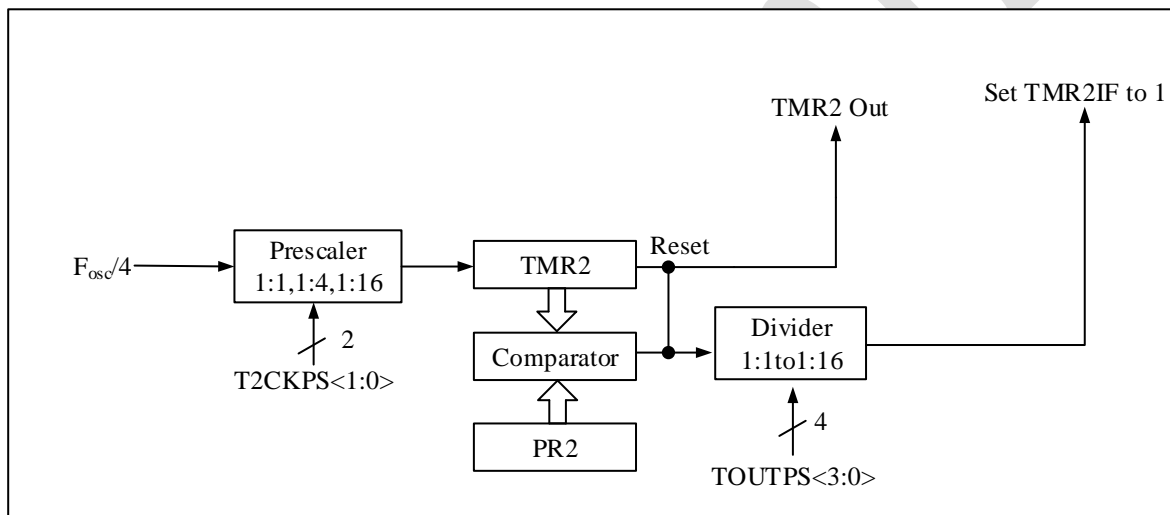


Figure 19-1 TIMER2 block diagram

## 19.2 TIMER2 Working Principle

The clock input of the TIMER2 module is the system instruction clock (FOSC/4). The clock is input to the TIMER2 prescaler and is available in the following division ratios: 1:1, 1:4, or 1:16. The prescaler output is used to increment the TMR2 register.

Continuously compare the values of TMR2 and PR2 to determine when they match. TMR2 will be incremented from 00h until it matches the value in PR2.When a match occurs, the following two events occur:

1) TMR2 is reset to 00h in the next increment cycle.

2) TIMER2 frequency divider increases.

The matching output of the TIMER2 and PR2 comparator is then input to the TIMER2 divider. The

frequency divider has a ratio of 1:1 to 1:16. The output of the TIMER2 frequency divider is used for the TMR2IF interrupt flag position 1 of the PIR1 register.

TMR2 and PR2 registers can be read and written. When reset, the TMR2 register is set to 00h and the PR2 register is set to FFh.

TMR2ON of T2CON register is set to enable TIMER2.

TMR2ON bit is cleared to disable TIMER2.

The TIMER2 prescaler is controlled by T2CKPS position of T2CON register.

The TIMER2 frequency divider is controlled by TOUTPS position of T2CON register.

The perscaler and the post-step counter are cleared in the following situations:

- Perform write operation on TMR2 register.

- Perform write operation on T2CON register.

- Any device reset (power reset, watchdog timer reset or under voltage reset).

Notes: Write T2CON will not clear TMR2.


## 19.3 TIMER2 Related Register

There are two registers associated with TIMER2, which are data storage register TMR2 and control register T2CON.

TIMER2 data register TMR2 (11H)

| 11H | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|------|------|------|------|------|------|------|------|------|
| TMR2 | - | - | - | - | - | - | - | - |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |


TIMER2 control register T2CON (12H)

| 12H | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|------|------|--------|--------|--------|--------|--------|---------|---------|
| T2CON | - | TOUTPS3 | TOUTPS2 | TOUTPS1 | TOUTPS0 | TMR2ON | T2CKPS1 | T2CKPS0 |
| R/W | - | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | - | 0 | 0 | 0 | 0 | 0 | 0 | 0 |


   Bit7      Reserve, Read as 0

   Bit6-3    TOUTPS<3:0>: TIMER2 Output post divider selection bit

               0000=1:1 Post Divider Ratio

               0001=1:2 Post Divider Ratio

               0010=1:3 Post Divider Ratio

               0011=1:4 Post Divider Ratio

               0100=1:5 Post Divider Ratio

0101=1:6 Post Divider Ratio
0110=1:7 Post Divider Ratio
0111=1:8 Post Divider Ratio
1000=1:9 Post Divider Ratio
1001=1:10 Post Divider Ratio
1010=1:11 Post Divider Ratio
1011=1:12 Post Divider Ratio
1100=1:13 Post Divider Ratio
1101=1:14 Post Divider Ratio
1110=1:15 Post Divider Ratio
1111=1:16 Post Divider Ratio

Bit2    TMR2ON: TIMER2 Enable bit
1=Enable TIMER2
0=Disable TIMER2

Bit1-0    T2CKPS<1:0>: TIMER2 Clock frequency ration selection
00=Prescaler value is 1
01= Prescaler value is 4
1x= Prescaler value is 16

# 20 ADC

## 20.1 ADC Overview

An analog-to-digital converter (ADC) converts an analog input signal into a 12-bit binary number that represents the signal. The analog input channels used by the device share a sample and hold circuit. The output of the sample-and-hold circuit is connected to the input of the analog-to-digital converter. The ADC uses a successive approximation to produce a 12-bit binary result and stores the result in the ADC result registers (ADRESL and ADRESH).

The ADC reference voltage is always generated internally.

The ADC can produce an interrupt after the transformation is complete.

Figure 20-1 ADC block diagram.
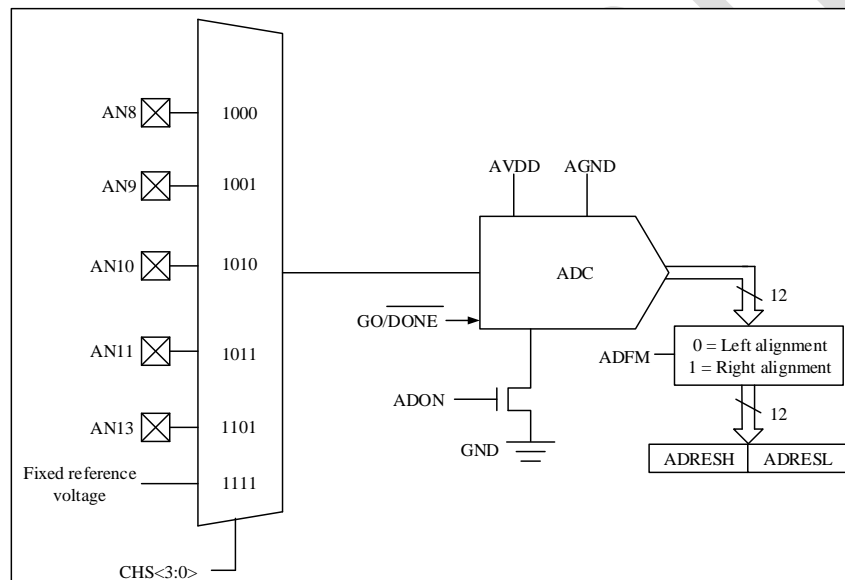


Figure 20-1 ADC block diagram

## 20.2 ADC Configuration

The following factors must be considered when configuring and using ADC:

1) Port configuration

2) Channel selection

3) ADC conversion clock source

4) Interrupt control

5) The result of the storage format

### 20.2.1 Port Configuration

ADC can convert analog signals as well as digital signals. When converting an analog signal, the I/O pin should be configured as an input by setting the corresponding TRIS bit. For more information, see the corresponding port chapter.

Notes: Applying an analog voltage to a pin that is defined as a digital input may result in an over-current in the input buffer.

### 20.2.2 Channel Selection

The CHS bit of the ADCON0 register determines which channel is connected to the sampling circuit. If the channel is changed, a certain delay is required before the next transformation begins.

### 20.2.3 ADC Reference Voltage

The reference voltage of ADC is always provided by VDD and GND of the chip.

### 20.2.4 Clock Transform

It is possible to select the clock source of the transformation by setting the ADCS bit of the AD-CON0 register. There are four possible clock frequencies to choose from:

− FOSC/8

− FOSC/16

− FOSC/32

− FRC (Internal oscillator)

The time to complete a transformation is defined as TAD. A complete 12-bit conversion requires 49 TAD cycles.

It is necessary to conform to the corresponding TAD specification to get the correct results.

Table 20-1 shows an example of choosing the correct ADC clock.

Notes: Unless FRC is used, any change in the system clock frequency will change the frequency of the ADC clock, thereby negatively affecting the ADC conversion results.

Table 20-1 Relationship between ADC CLOK CYCLE and device operating frequency

| ADC Clock Cycle | | Device frequency | | |
|---|---|---|---|---|
| ADC Clock Source | ADCS<1:0> | 8MHz | 4MHz | 1MHz |
| Fosc/8 | 00 | 49.0μs | 98.0μs | 392.0μs |
| Fosc/16 | 01 | 98.0μs | 196.0μs | 784.0μs |
| Fosc/32 | 10 | 196.0μs | 392.0μs | 1.5ms |
| FRC | 11 | 1-3ms | 1-3ms | 1-3ms |

Notes: It is recommended that you do not use values in shaded cells.

### 20.2.5 ADC Interrupt

The ADC module allows an interrupt to be generated after the A / D conversion is completed. The ADC interrupt flag is the ADIF bit in the PIR1 register and interrupt enable bit is the ADIE bit in the PIE1 register. The ADIF bit must be cleared with software. The ADIF bit will be set after each conversion, regardless of whether ADC interrupt is allowed.

Interrupt can occur regardless of whether the device is active or in sleep mode. If the device is in Sleep mode, this interrupt wakes up the device. When the device wakes from sleep mode, it always executes the next instruction after the STOP command. Global interrupts must be disabled if the user attempts to wake the device from sleep mode and resumes code execution in sequence. If the global interrupt is enabled, the program will jump to the interrupt service routine execution.

### 20.2.6 Result Formatting

The result of a 12-bit A/D conversion can take two formats: left-justified or right-justified. The output format is controlled by the ADFM bit in the ADCON0 register.

When ADFM=0, AD conversion result left justified and AD conversion result is 12Bit, when ADFM =1, AD conversion result right justified and AD conversion result is 10Bit.

## 20.3    ADC Working Principle

### 20.3.1 Start Conversion

To enable the ADC module, the ADON bit of the ADCON0 register must be set and the GO/DONE bit in of the ADCON0 register to begin analog-to-digital conversion.

Notes: The GO/DONE bit cannot be set by the same instruction that turns on the A/D module.

### 20.3.2 Complete the Conversion

When the conversion is complete, the ADC module will:

1)    Clear GO/DONE Bit

2)    Set the ADIF flag bit

3)    Update the ADRESH: ADRESL register with the new result of the transformation.

### 20.3.3 End the Conversion

If it is necessary to terminate the conversion before the conversion is completed, you can use the software clean GO/DONE bit. The ADRESH: ADRESL register is not updated with the uncompleted module conversion results. Therefore, ADRESH: the ADRESL register will maintain the value obtained from the last conversion. In addition, after the termination of A/D conversion, two TAD delays are required to begin the next collection. After the delay, the acquisition of the input signal of the selected channel will begin automatically.

Notes: Therefore, the reset closes the ADC module and terminates any pending transitions.

### 20.3.4 ADC Works in Sleep Mode

ADC module can work in sleep mode. This operation requires setting the ADC clock source to the FRC option. If you select the FRC clock source, the ADC waits one more instruction cycle before starting the conversion. This allows execution of the STOP instruction to reduce system noise during conversion. If ADC interrupts are enabled, the device wakes up from Sleep mode when the conversion is complete. If the ADC interrupt is disabled, the ADC module will still be turned off even after the conversion is completed, even if the ADON bit remains set. If the ADC clock source is not FRC, execution of the STOP instruction aborts the current conversion and turns off the A / D module even though the ADON bit remains set.

### 20.3.5 A/D Conversion Steps

The following steps give an example of using the ADC for analog-to-digital conversion:

1) Port configuration:

− Disable pin output driver (see TRIS register)

− Configure pins as analog input pins

2) Set ADC module:

− Select ADC conversion clock

− Select the ADC input channel

− Select the format of the result

− Start the ADC module

3) Set ADC interrupt (optional):

− Clear ADC interrupt flag

− Allow ADC interrupt

− Allow peripheral interrupt

− Allow global interrupts

4) The required collection time.

5) GO/DONE Set to 1 to initiate the conversion.

6) One of the following methods waits for the ADC conversion to finish:

− Check GO/DONE bit

− Wait for ADC interrupt (interrupt enabled)

7) ADC Result

8) ADC the ADC interrupt flag is cleared (this action is required if interrupts are enabled)

Notes: A global interrupt must be disabled if the user attempts to resume sequential code execution after waking the device from sleep mode.

Example 20-1 AD Conversion

```
LDIA        B'10000000'
LD          ADCON1,A
SETB        TRISA,0          ; Set PORTA.0 as input port
LDIA        B'11000001'
LD          ADCON0,A
CALL        DELAY            ; Delay for some time
SETB        ADCON0,GO
SZB         ADCON0,GO        ; Wait for AD conversion to finish
JP          $-1
LD          A,ADRESH         ; Save AD conversion result high
LD          RESULTH,A
LD          A,ADRESL         ; Save AD conversion result low
LD          RESULTL,A
```

## 20.4 ADC Related RAM

There are four main types of RAM associated with AD conversion, namely control register AD-CON0 and ADCON1, data register ADRESH and ADRESL.

AD control register ADCON0 (1FH)

| 1FH | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|-----|------|------|------|------|------|------|------|------|
| ADCON0 | ADCS1 | ADCS0 | CHS3 | CHS2 | CHS1 | CHS0 | GO/DONE | ADON |
| RW | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Bit7-6    ADCS<1:0>: A/D Convert clock select bit
00=FOSC/8
01=FOSC/16
10=FOSC/32
11=FRC (Generated by a dedicated internal oscillator clock frequency up to 32kHz)

Bit5-2    CHS<3:0>: Analog channel selection bit
1000=AN8
1001=AN9
1010=AN10
1011=AN11
1101=AN13
1111= Fixed reference voltage

Bit1    GO/DONE: A/D Conversion status bit
1=A/D Conversion is in progress. Setting this bit starts A/D conversion. This bit is automatically cleared by hardware when the A/D conversion is completed.
0=A/D Conversion completed or not in progress

Bit0    ADON: ADC Enable bit
1=Enable ADC
0=Disable ADC, Do not consume working current

AD control register ADCON1 (9FH)

| 9FH | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|-----|------|------|------|------|------|------|------|------|
| ADCON1 | ADFM | - | - | - | - | - | - | - |
| RW | R/W | - | - | - | - | - | - | - |
| Reset | 0 | - | - | - | - | - | - | - |

Bit7    ADFM: A/D Conversion result format selection bit
1= Align right
0= Align left

Bit6-0        Reserve, Read as 0

AD Data Register High Bits ADRESH (1EH), ADFM=0

| 1EH | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|-----|------|------|------|------|------|------|------|------|
| ADRESH | ADRES11 | ADRES10 | ADRES9 | ADRES8 | ADRES7 | ADRES6 | ADRES5 | ADRES4 |
| RW | R | R | R | R | R | R | R | R |
| Reset | x | x | x | x | x | x | x | x |

Bit7-0        ADRES<11:4>: ADC result register bit
The upper 8 bits of the 12-bit conversion result

AD Data Register Low Bits ADRESL (9EH), ADFM=0

| 9EH | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|-----|------|------|------|------|------|------|------|------|
| ADRESL | ADRES3 | ADRES2 | ADRES1 | ADRES0 | - | - | - | - |
| RW | R | R | R | R | - | - | - | - |
| Reset | x | x | x | x | - | - | - | - |

Bit7-4        ADRES<3:0>: ADC result register bit
The lower 4 Bits of the 12-Bit conversion result
Bit3-0        reserve

AD Data Register High Bits ADRESH (1EH), ADFM=1

| 1EH | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|-----|------|------|------|------|------|------|------|------|
| ADRESH | - | - | - | - | - | - | ADRES11 | ADRES10 |
| RW | - | - | - | - | - | - | R | R |
| Reset | - | - | - | - | - | - | x | x |

Bit7-2        reserve
Bit1-0        ADRES<11:10>: ADC result register bit
The Higher 2 Bits of the 12-Bit conversion result

AD Data Register Low Bits ADRESL (9EH), ADFM=1

| 9EH | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|-----|------|------|------|------|------|------|------|------|
| ADRESL | ADRES9 | ADRES8 | ADRES7 | ADRES6 | ADRES5 | ADRES4 | ADRES3 | ADRES2 |
| RW | R | R | R | R | R | R | R | R |
| Reset | x | x | x | x | x | x | x | x |

Bit7-0      ADRES<9:2>: ADC Result Register Bit

The 9-2 Bits of the 12-Bit conversion result

Notes: In the case of ADFM = 1, the result of the AD conversion holds only the upper 10 bits of the 12-bit result, where ADRESH stores the upper 2 bits and ADRESL stores bits 2 to 9.

# 21  PWM Module

The chip contains PWM1 and PWM2. The operation of PWM1 and PWM2 modules is basically the same.

## 21.1  PWM1

PWM mode generates a pulse width modulated signal with variable frequency and duty cycle. In PWM mode, timer TIMER2 is required.

CPP1 Control Register CCP1CON (17H)

| 17H | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|
| CCP1CON | - | - | DC1B1 | DC1B0 | CCP1M3 | CCP1M2 | CCP1M1 | CCP1M0 |
| RW | - | - | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset Value | - | - | 0 | 0 | 0 | 0 | 0 | 0 |

Bit7-6  reserve

Bit5-4  DC1B<1:0>: The lower two bits of the PWM duty cycle

These two bits are the lower 2 bits of the 10-bit PWM duty cycle control word.
The upper 8 bits of the duty cycle control word is in CCPR1L.

Bit3-0  CCP1M<3:0>: CCP1 mode select bit

0000= PWM Power down (Reset ECCP Module )

0001= reserve

0010= reserve

0011= reserve

0100= reserve

0101= reserve

0110= reserve

0111= reserve

1000= reserve

1001= reserve

1010= reserve

1011= reserve

11xx=PWM Mode is valid

## 21.2  PWM2

PWM mode generates a pulse width modulated signal with variable frequency and duty cycle. In PWM mode, timer TIMER2 is required.

CCP2 Control Register CCP2CON (1DH)

| 1DH | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|------|------|------|------|------|------|------|------|------|
| CCP2CON | - | - | DC2B1 | DC2B0 | CCP2M3 | CCP2M2 | CCP2M1 | CCP2M0 |
| RW | - | - | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset Value | - | - | 0 | 0 | 0 | 0 | 0 | 0 |

Bit7-6    reserve

Bit5-4    DC2B<1:0>: Two low PWM duty cycle of the control word

Capture mode:

Reserve.

Compare mode:

Reserve.

PWM Mode:

These two bits are the lower 2 bits of the 10-bit PWM duty cycle control word.

The upper 8 bits of the duty cycle control word is in CCPR2L.

Bit3-0    CCP2M<3:0>: CCP2 mode select bit

0000= PWM Power down (Reset ECCP Module )

0001= reserve

0010= reserve

0011= reserve

0100= reserve

0101= reserve

0110= reserve

0111= reserve

1000= reserve

1001= reserve

1010= reserve

1011= reserve

11xx=PWM Mode is valid

## 21.3    PWM Mode

PWM mode generates a pulse width modulated signal on the CCPx pin. The duty cycle, period and resolution are determined by the following registers:

-    PR2

-    T2CON

-    CCPRxL

-    CCPxCON

In Pulse Width Modulation (PWM) mode, the PWM module outputs a PWM signal with up to 10 bits of resolution on the CCPx pin. Because the CCPx pin is multiplexed with the port data latch, the corresponding TRIS bit must be cleared to enable the CCPx pin's output driver.

Notes: Clear CCPxCON register will give up the CCPx control of CCPx pins.
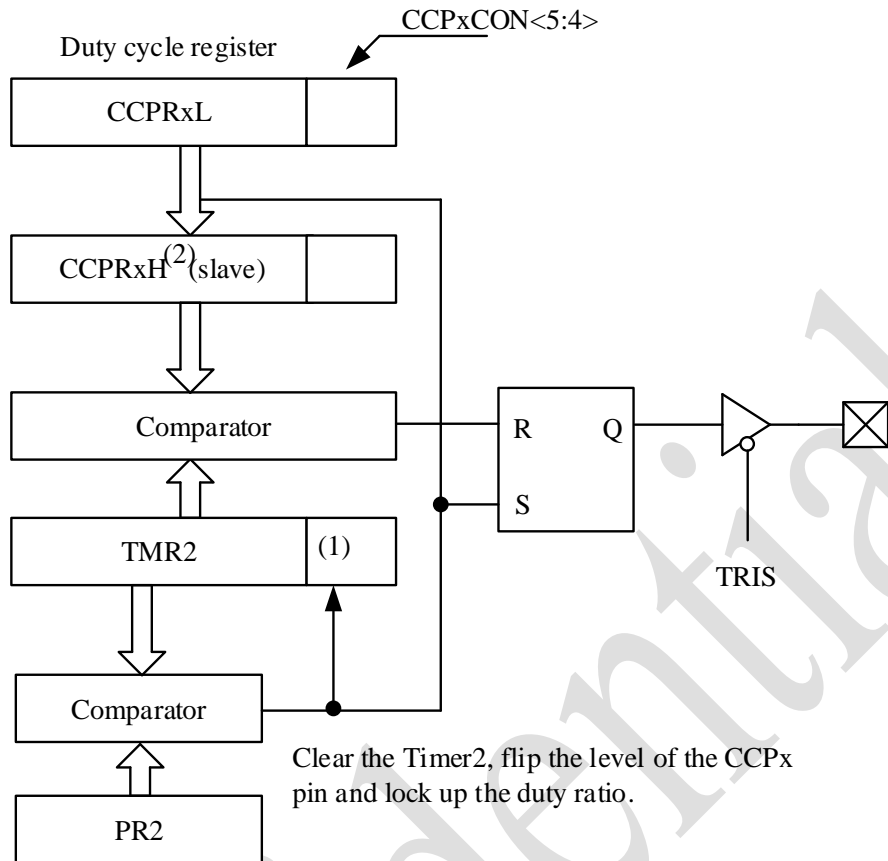
Notes: In the PWM mode, CCPRxH is a read-only register.

Figure 21-1 Simplified block diagram of PWM operation
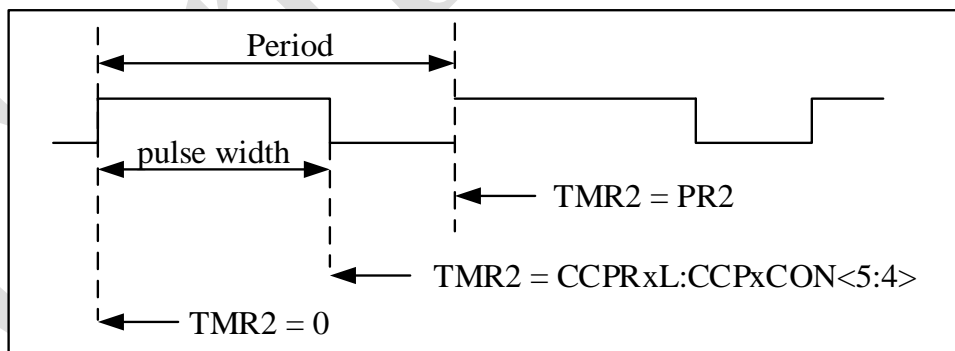


Figure 21-2 Typical waveform of PWM signal

### 21.3.1  PWM Period

The PWM period is specified by writing to the PR2 register of TIMER2.

The PWM period can be calculated using Equation 22-1:

Equation 21-1PWM period

PWM Period = [(PR2)+1]*4*TOSC* (TMR2 Prescaler value)

Notes: TOSC=1/FOSC

When TMR2 is equal to PR2, the following three events occur during the next up-count cycle:

1) TMR2 cleared

2) CCPxP is set (Exceptions: If PWM duty cycle =0%, The CCPx pin will not be set )

3) The PWM duty cycle is latched from CCPRxL to CCPRxH

Notes: TIMER2 Divide ratio is not used when determining the PWM frequency (see Section 19.2 "TIMER2 working principle").

## 21.3.2 PWM Duty Cycle

The PWM duty cycle can be specified by writing a 10-bit value to the following multiple registers: The CCPxL <1:0> bits of the CCPRxL and CCPxCON registers. CCPRxL holds the upper 8 bits of the duty cycle, the lower two bits of the duty cycle are stored in the DCxB <1: 0> bits of the CCPx-CON register. The CCPxCL <1: 0> bits of the CCPRxL and CCPxCON registers can be written at any time, but the duty cycle value is not latched into CCPRxH until the values in PR2 and TMR2 match (ie, the period ends). CCPRxH is a read-only register in PWM mode.

Equation 21-2 is used to calculate the width of the PWM pulse. Equation 12-3 is used to calculate the PWM duty cycle.

Equation 21-2 Pulse Width

Pulse Width = (CCPRxL: CCPxCON<5:4>)*TOSC*(TMR2 Prescaler value)

Equation 21-3 Duty cycle

Duty cycle $= \dfrac{(CCPRxL:CCPxCON<5:4>)}{4(PR2+1)}$

The CCPRxH register and a 2-bit internal latch are used to provide double buffering for the PWM duty cycle. This dual buffer structure is extremely important to avoid glitches during PWM operation.

The value of the 8-bit timer TMR2 register is combined with a 2-bit internal system clock (FOSC) or 2 bits of the prescaler to generate a 10-bit time base. The system clock is used when the TIMER2 prescaler is 1:1.

The CCPx pin is cleared when the 10-bit time base matches the CCPRxH and 2-bit latch combination.

## 21.3.3 PWM Resolution

The resolution determines the duty cycle in a given period. For example, a 10-bit resolution will produce 1024 discrete duty cycles while an 8-bit resolution will result in 256 discrete duty cycles.

When PR2 is 255, the maximum PWM resolution is 10 bits. As shown in Equation 12-4, the resolution is a function of the PR2 register value.

Equation 21-4 PWM Resolution

Resolution = (log [4(PR2+1)]) / (log (2))

Notes: If the pulse width is greater than the period value, the specified PWM pin will remain unchanged.

The following table shows the PWM frequency and resolution values for Fosc = 8M.

Table 21-1 Examples of PWM Frequency and Resolution (FOSC = 8MHz)

| PWM Frequency | 1.22kHz | 4.90kHz | 19.61kHz | 76.92kHz | 153.85kHz | 200.0kHz |
|---|---|---|---|---|---|---|
| Timer prescaler value (1, 4 or 16) | 16 | 4 | 1 | 1 | 1 | 1 |
| PR2 Value | 0x65 | 0x65 | 0x65 | 0x19 | 0x0C | 0x09 |
| Highest resolution (bit) | 8 | 8 | 8 | 6 | 5 | 5 |

### 21.3.4 Operation in Sleep Mode

In Sleep mode, the TMR2 register will not increment and the status of the module will remain unchanged. If the CCPx pin has output, the output will continue to be unchanged. When the device is awakened, TMR2 will continue to work from its original state.

### 21.3.5 System Clock Frequency Changes

The PWM frequency is generated by the system clock frequency. Any change in the system clock frequency will cause the PWM frequency to change.

### 21.3.6 The Effect of Reset

Any reset forces all ports to input mode and forces the CCP register to its reset state.

### 21.3.7 Set the PWM Operation

The following steps should be performed when configuring the CCP module for PWM operation:

1) The PWM pin (CCPx) output driver is disabled as an input by setting the appropriate TRIS bit.

2) Set the PWM period by loading the PR2 register.

3) The PWM mode of the CCP module is configured by loading the CCPxCON register with the appropriate value.

4) The PWM duty cycle is set by loading the CCPRxL register and the DCxB <1: 0> bits in the CCPxCON register.

5) Configure and start TIMER2:

− Clear the TMR2IF interrupt flag in the PIR1 register.

− The TIMER2 prescaler is set by loading the T2CKPS bit in the T2CON register.

− TIMER2 is enabled by setting the TMR2ON bit in the T2CON register.

6) PWM output enabled after new PWM period begins:

− Wait for TIMER2 to overflow (TMR2IF bit in PIR1 register is set).

− The CCPx pin output driver is enabled by clearing the corresponding TRIS bit.

# 22 MCU DC Characteristics

## 22.1 MCU DC Characteristics

| Symbol | Parameter | Test Conditions | | Min | Tpy | Max | Units |
|--------|-----------|------|------|-----|-----|-----|-------|
| | | VDD | condition | | | | |
| VDD | Operating Voltage | - | 8MHz | 2.2 | - | 3.3 | V |
| | | - | 4MHz | 2.2 | - | 3.3 | V |
| Idd | Operating current | 3V | ADC Enable | - | 2 | - | mA |
| | | 2V | ADC Enable | - | 1 | - | mA |
| Istb | Quiescent Current | 3V | - | - | 3 | 30 | μA |
| | | 2V | - | - | 3 | 30 | μA |
| Vil | Low level input voltage | - | - | - | - | 0.3VDD | V |
| Vih | High level input voltage | - | - | 0.7VDD | - | - | V |
| Voh | High-level output voltage | - | Without load | 0.9VDD | - | - | V |
| VoL | Low-level output voltage | - | Without load | - | - | 0.1VDD | V |
| VADI | ADC port input voltage | - | - | 0 | - | VDD | V |
| VAD | ADC operating voltage | - | - | 2 | - | 3.6 | V |
| EAD | ADC conversion error | - | - | - | ±2 | - | - |
| Rph | Pull-up resistor value | 3V | - | - | 35 | - | K |
| | | 2V | - | - | 65 | - | K |
| IoL | Output sink current | 3V | Vol=0.3VDD | - | 50 | - | mA |
| | | 2V | Vol=0.3VDD | - | 25 | - | mA |
| IoH | Output port pull current | 3V | Voh=0.7VDD | - | 15 | - | mA |
| | | 2V | Voh=0.7VDD | - | 10 | - | mA |

## 22.2 MCU AC Characteristics

| Symbol | Parameter | Test Conditions | | Min | Tpy | Max | Units |
|--------|-----------|------|------|-----|-----|-----|-------|
| | | VDD | Condition | | | | |
| TWDT | WDT reset time | 3V | - | - | 18 | - | ms |
| | | 2V | - | - | 36 | - | ms |
| TAD | ADC conversion time | 3V | - | - | 49 | - | CLK |
| | | 2V | - | - | 49 | - | CLK |
| FINTRC | 8MHz internal oscillator frequency | VDD=2 to 3.6V, 25℃ | | -1.5% | - | 1.5% | - |
| | | VDD=2 to 3.6V, -40 to 85℃ | | -2.5% | - | 2.5% | - |

## 22.3   Instruction List

| Mnemonic | | Operational | Instruction cycle | Sign |
|---|---|---|---|---|
| Control class -4 | NOP | Empty operation | 1 | None |
| | STOP | Enter sleep mode | 1 | TO, PD |
| | CLRWDT | Clear the watchdog counter | 1 | TO, PD |
| Data Transfer -4 | LD [R], A | Transfer ACC contents to R | 1 | NONE |
| | LD A, [R] | Send R content to ACC | 1 | Z |
| | TESTZ R | Pass data memory contents to data memory | 1 | Z |
| | LDIA i | Immediately i sent to the ACC | 1 | NONE |
| Logic Operation -16 | CLR A | Clear ACC | 1 | Z |
| | SET [R] | Set data memory R. | 1 | NONE |
| | CLR [R] | Clear data memory R | 1 | Z |
| | ORA [R] | R and ACC content to do or operation, the result is stored in the ACC | 1 | Z |
| | ORR [R] | R and ACC content to do "or" operation, the result into R | 1 | Z |
| | ANDA [R] | R and ACC content to do "and" operation, the result is stored in ACC | 1 | Z |
| | ANDR [R] | R and ACC contents of the "and" operation, the result into R | 1 | Z |
| | XORA [R] | R and ACC content to do "exclusive OR" operation, the result is stored ACC | 1 | Z |
| | XORR [R] | R and ACC content to do "exclusive OR" operation, the result is stored in R. | 1 | Z |
| | SWAPA [R] | R register contents of the high and low nibble conversion, the result is stored ACC | 1 | NONE |
| | SWAPR [R] | R register contents of the high and low nibble conversion, the result stored in R | 1 | NONE |
| | COMA [R] | The contents of the R register are inverted and the result is stored in ACC | 1 | Z |
| | COMR [R] | The contents of the R register are inverted and the result is stored in R | 1 | Z |
| | XORIA i | ACC and immediate i do XOR operation, the result stored ACC | 1 | Z |
| | ANDIA i | ACC and immediate i do "AND" operation, the result is stored in ACC | 1 | Z |
| | ORIA i | ACC and immediate i do "or" operation, the result is stored ACC | 1 | Z |

| Shift operation, 8 | RRCA [R] | The data memory is rotated right by one bit with carry bit and the result is stored in ACC | 1 | C |
|---|---|---|---|---|
| | RRCR [R] | The data memory is rotated right by one bit and the result is stored in R | 1 | C |
| | RLCA [R] | The data memory is rotated left by one bit with carry bit and the result is stored in ACC | 1 | C |
| | RLCR [R] | The data memory is rotated left one bit with carry bit and the result is stored in R | 1 | C |
| | RLA [R] | The data memory is rotated left by one bit without carry and the result is placed in ACC | 1 | NONE |
| | RLR [R] | The data memory is rotated left by one bit without carry and the result is stored in R | 1 | NONE |
| | RRA [R] | The data memory is rotated right by one bit without carry and the result is stored in ACC | 1 | NONE |
| | RRR [R] | The data memory is rotated right by one bit without carry and the result is stored in R | 1 | NONE |
| Incremental decline, 4 | INCA [R] | Increment Data Memory R and put the result in ACC | 1 | Z |
| | INCR [R] | Increment data memory R, the result into R | 1 | Z |
| | DECA [R] | Decrement Data Memory R and place the result in ACC | 1 | Z |
| | DECR [R] | Decrement data memory R, the result into R | 1 | Z |
| Bit operational, 2 | CLRB [R], b | Clear a bit in data memory R. | 1 | NONE |
| | SETB [R], b | One place in data memory R | 1 | NONE |
| Look up the table, 2 | TABLE [R] | Read OTP content results into TABLE_DATAH and R | 2 | NONE |
| | TABLEA | Read OTP content results into TABLE_DATAH and ACC | 2 | NONE |
| Mathematical operations, 16 | ADDA [R] | ACC+[R]→ACC | 1 | Z,C,DC,OV |
| | ADDR [R] | ACC+[R]→R | 1 | Z,C,DC,OV |
| | ADDCA [R] | ACC+[R]+C→ACC | 1 | Z,C,DC,OV |
| | ADDCR [R] | ACC+[R]+C→R | 1 | Z,C,DC,OV |
| | ADDIA i | ACC+i→ACC | 1 | Z,C,DC,OV |
| | SUBA [R] | [R]-ACC→ACC | 1 | Z,C,DC,OV |
| | SUBR [R] | [R]-ACC→R | 1 | Z,C,DC,OV |
| | SUBCA [R] | [R]-ACC-C→ACC | 1 | Z,C,DC,OV |

| | SUBCR [R] | [R]-ACC-C→R | 1 | Z,C,DC,OV |
|---|---|---|---|---|
| | SUBIA i | i-ACC→ACC | 1 | Z,C,DC,OV |
| | HSUBA [R] | ACC-[R]→ACC | 1 | Z,C,DC,OV |
| | HSUBR [R] | ACC-[R]→R | 1 | Z,C,DC,OV |
| | HSUBCA [R] | ACC-[R]-$\overline{C}$→ACC | 1 | Z,C,DC,OV |
| | HSUBCR [R] | ACC-[R]-$\overline{C}$→R | 1 | Z,C,DC,OV |
| | HSUBIA i | ACC-i→ACC | 1 | Z,C,DC,OV |
| Unconditional transfer, 5 | RET | Return from subroutine | 2 | NONE |
| | RET i | Returns from the subroutine and saves the literal I into ACC | 2 | NONE |
| | RETI | Return from interruption | 2 | NONE |
| | CALL ADD | Subroutine call | 2 | NONE |
| | JP ADD | Unconditional jump | 2 | NONE |
| Condition transfer, 8 | SZB [R], b | If the b bit of data memory R is "0", the next instruction is skipped | 1or2 | NONE |
| | SNZB [R], b | If the b bit of data memory R is "1", the next instruction is skipped | 1or2 | NONE |
| | SZA [R] | Data memory R is sent to ACC. If the content is "0", the next instruction is skipped | 1or2 | NONE |
| | SZR [R] | Data memory R contents "0", then skip the next instruction | 1or2 | NONE |
| | SZINCA [R] | Data memory R plus "1", the results into the ACC, if the result is "0", then skip the next instruction | 1or2 | NONE |
| | SZINCR [R] | Data memory R plus "1", the result into R, if the result is "0", then skip the next instruction | 1or2 | NONE |
| | SZDECA [R] | Data memory R minus "1", the results into the ACC, if the result is "0", then skip the next instruction | 1or2 | NONE |
| | SZDECR [R] | Data memory R minus "1", the result into R, if the result is "0", then skip the next instruction | 1or2 | NONE |

## 22.4 Instruction Description

ADDA            [R]

operation:        R plus ACC, the result into ACC

cycle:            1

Affect status bit:   C, DC, Z, OV

example:

|      | LDIA | 09H  | ; Assign 09H to ACC |
|------|------|------|---------------------|
|      | LD   | R01,A | ; Assign ACC (09H) value to register R01 |
|      | LDIA | 077H | ; Assign 77H to ACC |
|      | ADDA | R01  | ; Execution result: ACC = 09H + 77H = 80H |

ADDR       [R]

operation:    R plus ACC, the results into R

cycle:    1

Affect   status   C, DC, Z, OV
bit

example:

|      | LDIA | 09H  | ; Assign 09H to ACC |
|------|------|------|---------------------|
|      | LD   | R01,A | ; Assign ACC (09H) value to register R01 |
|      | LDIA | 077H | ; Assign 77H to ACC |
|      | ADDR | R01  | ; Execution result: R01 = 09H + 77H = 80H |

ADDCA      [R]

operation:    R plus ACC and C bit, the result into ACC

cycle:    1

Affect   status   C, DC, Z, OV
bit:

example:

|      | LDIA  | 09H  | ; Assign 09H to ACC |
|------|-------|------|---------------------|
|      | LD    | R01,A | ; Assign ACC value (09H) to the register R01 |
|      | LDIA  | 077H | ; Assign 77H to ACC |
|      | ADDCA | R01  | ; Execution result: ACC = 09H + 77H + C= 80H(C=0) |
|      |       |      | ACC=09H+77H+C=81H(C=1) |

ADDCR      [R]

operation:    R plus ACC and C bit, the result into R

cycle:    1

Affect   status   C, DC, Z, OV
bit:

example:

|      | LDIA  | 09H  | ; Assign 09H to ACC |
|------|-------|------|---------------------|
|      | LD    | R01,A | ; Assign ACC (09H) to custom register R01 |
|      | LDIA  | 077H | ; Assign 77H to ACC |
|      | ADDCR | R01  | ; Execution result: R01 = 09H + 77H + C = 80H (C = 0) |
|      |       |      | R01=09H+77H+C=81H(C=1) |

ADDIA      i

operation:    immediate data i plus ACC, the result into ACC

cycle:    1

Affect   status   C, DC, Z, OV
bit:

example:

| | | | |
|---|---|---|---|
| | LDIA | 09H | ; Assign 09H to ACC |
| | ADDIA | 077H | ; Execution result: ACC = ACC (09H) + i (77H) = 80H |

| | |
|---|---|
| ANDA | [R] |
| operation: | Register R with ACC logic and operation, the result into ACC |
| cycle: | 1 |
| Affect status bit: | Z |
| example: | |

| | | | |
|---|---|---|---|
| | LDIA | 0FH | ; Assign 0FH to ACC |
| | LD | R01,A | ; Assign the ACC value (0FH) to register R01 |
| | LDIA | 77H | ; Assign 77H to ACC |
| | ANDA | R01 | ; Execution result: ACC = (0FHand77H) = 07H |

| | |
|---|---|
| ANDR | [R] |
| operation: | Register R with ACC logic and operation, the result into R |
| cycle: | 1 |
| Affect assign bit: | Z |
| example: | |

| | | | |
|---|---|---|---|
| | LDIA | 0FH | ; Assign 0FH to ACC |
| | LD | R01,A | ; Assign the ACC value (0FH) to register R01 |
| | LDIA | 77H | ; Assign 77H to ACC |
| | ANDR | R01 | ; Execution result: R01 = (0FHand77H) = 07H |

| | |
|---|---|
| ANDIA | i |
| operation: | Immediate data i with ACC logic and operation, the result into ACC |
| cycle: | 1 |
| Affect assign bit: | Z |
| example: | |

| | | | |
|---|---|---|---|
| | LDIA | 0FH | ; Assign 0FH to ACC |
| | ANDIA | 77H | ; Execution result: ACC = (0FHand77H) = 07H |

| | |
|---|---|
| CALL | ADD |
| operation: | Call subroutine |
| cycle: | 2 |
| Affect assign bit: | no |
| example: | |

| | | | |
|---|---|---|---|
| | CALL | LOOP | ; Call the subroutine address whose name is defined as "LOOP" |

CLRA

| operation: | clear ACC |
| cycle: | 1 |
| Affect assign bit: | Z |
| example: | |

| | CLR | A | ; Execution result: ACC=0 |

CLR          [R]

| operation: | Clear register R |
| cycle: | 1 |
| Affect status bit: | Z |
| example: | |

| | CLR | R01 | ; Execution result: R01=0 |

CLRB          [R],b

| operation: | Clear bit b of register R |
| cycle: | 1 |
| Affect status bit: | No |
| example: | |

| | CLRB | R01,3 | ; Execution result: Bit 3 of R01 is zero |

CLRWDT

| operation: | Clear watchdog counter |
| cycle: | 1 |
| Affect status bit: | TO, PD |
| example: | |

| | CLRWDT | | ; Watchdog counter clear |

COMA          [R]

| operation: | Register R inverted, result into ACC |
| cycle: | 1 |
| Affect status bit: | Z |
| example: | |

| | LDIA | 0AH | ; Assign ACC to 0AH |
| | LD | R01,A | ; Assign ACC value (0AH) to register R01 |
| | COMA | R01 | ; Execution result: ACC=0F5H |

COMR          [R]

operation:  Register R inverted, the result into R

cycle:   1

Affect  status Z

bit:

example:

| | | | |
|---|---|---|---|
| | LDIA | 0AH | ; Assign ACC to 0AH |
| | LD | R01,A | ; Assign ACC value (0AH) to register R01 |
| | COMR | R01 | ; Execution result: R01=0F5H |

**DECA**   [R]

operation:  Register R self minus 1, result into ACC

cycle:   1

Affect  status Z

bit:

example:

| | | | |
|---|---|---|---|
| | LDIA | 0AH | ; Assign 0AH to ACC |
| | LD | R01,A | ; Assign ACC value (0AH) to register R01 |
| | DECA | R01 | ; Execution result: ACC=(0AH-1)=09H |

**DECR**   [R]

operation:  Register R self minus 1, result into R

cycle:   1

Affect  status Z

bit:

example:

| | | | |
|---|---|---|---|
| | LDIA | 0AH | ; Assign 0AH to ACC |
| | LD | R01,A | ;Assign the ACC value(0AH) to register R01 |
| | DECR | R01 | ; Execution result: R01 = (0AH-1) = 09H |

**HSUBA**   [R]

operation:  ACC minus R, result ACC

cycle:   1

Affect  status C,DC,Z,OV

bit:

example:

| | | | |
|---|---|---|---|
| | LDIA | 077H | ; Assign 077H to ACC |
| | LD | R01,A | ; Assign the ACC value (077H) to register R01 |
| | LDIA | 080H | ; Assign 080H to ACC |
| | HSUBA | R01 | ; Execution result: ACC = (80H-77H) = 09H |

**HSUBR**   [R]

operation:  ACC minus R, result into R

cycle:   1

Affect status bit: C,DC,Z,OV

example:

| | | | |
|---|---|---|---|
| | LDIA | 077H | ; Assign 077H to ACC |
| | LD | R01,A | ; Assign the ACC value (077H) to register R01 |
| | LDIA | 080H | ; Assign 080H to ACC |
| | HSUBR | R01 | ; Execution result: R01 = (80H-77H) = 09H |

HSUBCA     [R]

operation:

    ACC minus R minus $\overline{C}$ , result into ACC

cycle: 1

Affect status bit: C,DC,Z,OV

example:

| | | | |
|---|---|---|---|
| | LDIA | 077H | ; Assign 077H to ACC |
| | LD | R01,A | ; Assign the ACC value (077H) to register R01 |
| | LDIA | 080H | ; Assign 080H to ACC |
| | HSUBCA | R01 | ; Execution result: ACC = (80H-77H-) = 09H (C = 0) |

                                          ACC=(80H-77H-$\overline{C}$ )=08H(C=1)

HSUBCR     [R]

operation:

    ACC minus R minus $\overline{C}$ , result into R

cycle: 1

Affect status bit: C,DC,Z,OV

example:

| | | | |
|---|---|---|---|
| | LDIA | 077H | ; Assign 077H to ACC |
| | LD | R01,A | ; Assign the ACC value (077H) to register R01 |
| | LDIA | 080H | ; Assign 080H to ACC |
| | HSUBCR | R01 | ; Execution result: R01 = (80H-77H-) = 09H (C = 0) |

                                          R01=(80H-77H-$\overline{C}$ )=08H(C=1)

INCA          [R]

operation:    Register R self plus 1, results into the ACC

cycle: 1

Affect status bit: Z

example:

| | | | |
|---|---|---|---|
| | LDIA | 0AH | ; Assign 0AH to ACC |
| | LD | R01,A | ; Assign ACC value (0AH) to register R01 |
| | INCA | R01 | ; Execution result: ACC = (0AH + 1) = 0BH |

INCR            [R]

operation:      Register R self plus 1, result into ACC

cycle:          1

Affect    status    Z
bit:

example:

|  | LDIA | 0AH | ; Assign 0AH to ACC |
|  | LD | R01,A | ; Assign ACC value (0AH) to register R01 |
|  | INCR | R01 | ; Execution result: R01 = (0AH + 1) = 0BH |

JP            ADD

operation:      Jump to add address

cycle:          2

Affect    status    no
bit:

example:

|  | JP | LOOP | ; Jump to the subroutine address whose name is defined as "LOOP" |

LD            A, [R]

operation:      Assign R to ACC

cycle:          1

Affect    status    Z
bit:

example:

|  | LD | A,R01 | ; Assign the value of register R0 to ACC |
|  | LD | R02,A | ; The value of ACC is assigned to register R02, which realizes the data movement from R01 → R02 |

LD            [R], A

operation:      Assign ACC to R

cycle:          1

Affect    status    no
bit:

example:

|  | LDIA | 09H | ; Assign 09H to ACC |
|  | LD | R01,A | ; Execution result: R01 = 09H |

LDIA            i

operation:      Immediate data i assign to the ACC

cycle:          1

Affect    status    no
bit:

Example

        LDIA        0AH        ; Assign 0AH to ACC

**NOP**

| operation: | Empty instruction |
| --- | --- |
| cycle: | 1 |
| Affect status bit: | no |
| example: | |

        NOP

**ORIA        i**

| operation: | Immediate data and ACC logic or operation, result into ACC |
| --- | --- |
| cycle: | 1 |
| Affect status bit: | Z |
| example: | |

        LDIA        0AH        ; Assign 0AH to ACC
        ORIA        030H        ; Execution result: ACC = (0AHor30H) = 3AH

**ORA        [R]**

| operation: | Register R and ACC Logic or operation, result into ACC |
| --- | --- |
| cycle: | 1 |
| Affect status bit: | Z |
| example: | |

        LDIA        0AH        ; Assign 0AH to ACC
        LD        R01,A        ; Assign ACC (0AH) to register R01
        LDIA        30H        ;Assign 30H to ACC
        ORA        R01        ; Execution result: ACC=(0AHor30H)=3AH

**ORR        [R]**

| operation: | Register R with ACC logic OR operation, result into R |
| --- | --- |
| cycle: | 1 |
| Affect status bit: | Z |
| example: | |

        LDIA        0AH        ; Assign 0AH to ACC
        LD        R01,A        ; Assign ACC (0AH) to register R01
        LDIA        30H        ; Assign 30H to ACC
        ORR        R01        ; Execution result: R01=(0AHor30H)=3AH

**RET**

| operation: | Return from subroutine |
| --- | --- |

cycle:    2

Affect status bit:    no

example:

| | | | |
|---|---|---|---|
| | CALL | LOOP | ; Call subroutine LOOP |
| | NOP | | ; RET instruction return to execute this statement |
| | … | | ; Other programs |
| LOOP: | | | |
| | … | | ; Subroutine |
| | RET | | ; Subroutine return |

RET    i

operation:    Return from subroutine with parameters, parameters into ACC

cycle:    2

Affect status bit:    no

example:

| | | | |
|---|---|---|---|
| | CALL | LOOP | ; Call subroutine LOOP |
| | NOP | | ; RET instruction return to execute this statement |
| | … | | ; Other programs |
| LOOP: | | | |
| | … | | ; Subroutine |
| | RET | 35H | ; Subroutine return, ACC=35H |

RETI

operation:    Interrupt return

cycle:    2

Affect status bit:    no

example:

| | | | |
|---|---|---|---|
| INT_START | | | ; Interrupt program entry |
| | … | | ; Interrupt handler |
| | RETI | | ; Interrupt return |

RLCA    [R]

operation:    Register R with C loop left move one, result into ACC

cycle:    1

Affect status bit:    C

example:

| | | | |
|---|---|---|---|
| | LDIA | 03H | ; Assign 03H to ACC |
| | LD | R01,A | ; ACC value assigned to R01,R01=03H |
| | RLCA | R01 | ; Execution result: ACC=06H(C=0); |
| | | | ACC=07H(C=1) |

C=0

RLCR        [R]

operation:      Register R with C loop left move one, the result into R

cycle:          1

Affect status   C
bit:

example:

| | | |
|---|---|---|
| LDIA | 03H | ; Assign 03H to ACC |
| LD | R01,A | ; ACC value assign to R01, R01,R01=03H |
| RLCR | R01 | ; Execution result: R01=06H(C=0); |
| | | R01=07H(C=1); C=0 |

RLA         [R]

operation:      Register R without C loop left move one, result into ACC

cycle:          1

Affect status   no
bit:

example:

| | | |
|---|---|---|
| LDIA | 03H | ; Assign 03H to ACC |
| LD | R01,A | ; ACC value assigned to R01, R01,R01=03H |
| RLA | R01 | ; Execution result: ACC=06H |

RLR         [R]

operation:      Register R without C loop left move one, result into R

cycle:          1

Affect status   no
bit:

example:

| | | |
|---|---|---|
| LDIA | 03H | ; Assign 03H to ACC |
| LD | R01,A | ; ACC value assigned to R01, R01,R01=03H |
| RLR | R01 | ; Execution result: R01=06H |

RRCA       [R]

operation:      Register R with C loop right move one, result into ACC

cycle:          1

Affect status   C
bit:

example:

| | | |
|---|---|---|
| LDIA | 03H | ; Assign 03H to ACC |
| LD | R01,A | ; ACC value assigned to R01, R01,R01=03H |
| RRCA | R01 | ; Execution result: ACC=01H(C=0); |
| | | ACC=081H(C=1); |
| | | C=1 |

| RRCR | [R] | | |
|------|-----|---|---|
| operation: | Register R with C loop right move one, result into R | | |
| cycle: | 1 | | |
| Affect status bit: | C | | |
| example | | | |
| | LDIA | 03H | ; ACC Assign 03H to ACC |
| | LD | R01,A | ; ACC value assigned to R01, R01,R01=03H |
| | RRCR | R01 | ; Execution result: R01=01H(C=0); |
| | | | R01=81H(C=1); |
| | | | C=1 |

| RRA | [R] | | |
|-----|-----|---|---|
| operation: | Register R without C loop right move one, result into ACC | | |
| cycle: | 1 | | |
| Affect status bit: | no | | |
| example: | | | |
| | LDIA | 03H | ; Assign 03H to ACC |
| | LD | R01,A | ; ACC value assigned to R01, R01,R01=03H |
| | RRA | R01 | ; Execution result: ACC=81H |

| RRR | [R] | | |
|-----|-----|---|---|
| operation: | Register R without C loop right move one, the result into R | | |
| cycle: | 1 | | |
| Affect status bit: | No | | |
| example: | | | |
| | LDIA | 03H | ; Assign 03H to ACC |
| | LD | R01,A | ; Assign ACC to R01 |
| | RRR | R01 | ; Execution result: R01=03H |

| SET | [R] | | |
|-----|-----|---|---|
| operation: | Register R all positions is set to 1 | | |
| cycle: | 1 | | |
| Affect status bit: | No | | |
| example | | | |
| | SET | R01 | ; Execution result: R01=0FFH |

| SETB | [R],b | |
|------|-------|---|
| operation: | Bit b of register R is set | |
| cycle: | 1 | |

Affect    status    no
bit:

example:

|      | CLR  | R01   | ;R01=0                        |
|------|------|-------|-------------------------------|
|      | SETB | R01,3 | ; Execution result: R01=08H   |


**STOP**

operation:    Enter sleep mode

cycle:    1

Affect    status    TO, PD
bit:

example:

|      | STOP |       | ; Chip into power-saving mode, CPU, oscillator stop working, |
|------|------|-------|----------------------------------------------------|
|      |      |       |   IO port to maintain the original state.          |


**SUBIA        i**

operation:    Immediate data i minus ACC, result into ACC

cycle:    1

Affect    status    C,DC,Z,OV
bit:

example:

|      | LDIA  | 077H | ; Assign 077H to ACC                    |
|------|-------|------|-----------------------------------------|
|      | SUBIA | 80H  | ; Execution result: ACC=80H-77H=09H     |


**SUBA        [R]**

operation:    Register R minus ACC, result into ACC

cycle:    1

Affect    status    C,DC,Z,OV
bit:

example:

|      | LDIA | 080H  | ; Assign 080H to ACC                        |
|------|------|-------|---------------------------------------------|
|      | LD   | R01,A | ; ACC value assign to R01, R01 = 80H        |
|      | LDIA | 77H   | ; Assign 77H to ACC                         |
|      | SUBA | R01   | ; Execution result: ACC=80H-77H=09H         |


**SUBR        [R]**

operation:    Register R minus ACC, result into R

cycle:    1

Affect    status    C,DC,Z,OV
bit:

example:

|      | LDIA | 080H  | ; Assign 80H to ACC                         |
|------|------|-------|---------------------------------------------|
|      | LD   | R01,A | ; ACC value assign to R01, R01 = 80H        |
|      | LDIA | 77H   | ; Assign 77H to ACC                         |

|       | SUBR | R01 | ; Execution result: R01=80H-77H=09H |
|-------|------|-----|--------------------------------------|

**SUBCA**    [R]

operation:    Register R minus ACC and C, result into ACC

cycle:    1

Affect status bit:    C,DC,Z,OV

example:

|       | LDIA  | 080H | ; Assign 80H to ACC |
|-------|-------|------|---------------------|
|       | LD    | R01,A | ; ACC value assigned to R01, R01=80H |
|       | LDIA  | 77H  | ; Assign 77H to ACC |
|       | SUBCA | R01  | ; Execution result: ACC=80H-77H-C=09H(C=0); |
|       |       |      | ACC=80H-77H-C=08H(C=1); |

**SUBCR**    [R]

operation:    Register R minus ACC and C, result into R

cycle:    1

Affect status bit:    C,DC,Z,OV

example:

|       | LDIA  | 080H | ; Assign 80H to ACC |
|-------|-------|------|---------------------|
|       | LD    | R01,A | ; ACC value assigned to R01, R01=80H |
|       | LDIA  | 77H  | ; Assign 77H to ACC |
|       | SUBCR | R01  | ; Execution result: R01=80H-77H-C=09H(C=0) |
|       |       |      | R01=80H-77H-C=08H(C=1) |

**SWAPA**    [R]

operation:    Register R high and low nibble exchange, result into ACC

cycle:    1

Affect status bit:    no

example:

|       | LDIA  | 035H | ; Assign 35H to ACC |
|-------|-------|------|---------------------|
|       | LD    | R01,A | ; ACC value assigned to R01, R01=35H |
|       | SWAPA | R01  | ; Execution result: ACC=53H |

**SWAPR**    [R]

operation:    Register R high and low nibble exchange, result into ACC

cycle:    1

Affect status bit:    no

example:

|       | LDIA  | 035H | ; Assign 35H to ACC |
|-------|-------|------|---------------------|
|       | LD    | R01,A | ; ACC value assigned to R01, R01=35H |

| | | |
|---|---|---|
| SWAPR | R01 | ; Execution result: R01=53H |

**SZB**        [R],b

operation:     Judge the b-bit of register R., if it is 0 then jump, Otherwise, execution by sequence

cycle:         1or2

Affect status bit:     无

example:

| | | |
|---|---|---|
| SZB | R01,3 | ; Judge the third bit of register R01 |
| JP | LOOP | ; If R01 bit 3 is 1 then implementation this sentence, jump to loop |
| JP | LOOP1 | ; If R01 bit 3 is 0 time jump, execute this statement, jump to LOOP1 |

**SNZB**      [R],b

operation:     Judge the b-bit of register R., if it is 1 then jump, Otherwise, execution by sequence

cycle:         1or2

Affect status bit:     No

example:

| | | |
|---|---|---|
| SNZB | R01,3 | ; Judge the third bit of register R01 |
| JP | LOOP | ; If R01 bit 3 is 0 then implementation this sentence, jump to loop |
| JP | LOOP1 | ; If R01 bit 3 is 1 time jump, execute this statement, jump to LOOP1 |

**SZA**        [R]

operation:     Assign the value of register R to ACC, If the result is 0, then jump to next statement, otherwise execution by sequence

cycle:         1or2

Affect status bit:     No

example:

| | | |
|---|---|---|
| SZA | R01 | ;R01→ACC |
| JP | LOOP | ; When R01 is not 0, execute this statement and jump to LOOP |
| JP | LOOP1 | ; When R01 is 0, execute this statement and jump to LOOP1 |

**SZR**        [R]

operation:     Assign the value of register R to R, If the result is 0, then jump to next statement, otherwise execution by sequence

cycle:         1or2

Affect status bit:     No

example:

| | | |
|---|---|---|
| SZR | R01 | ;R01→R01 |
| JP | LOOP | ; When R01 is not 0, execute this statement and jump to LOOP |
| JP | LOOP1 | ; When R01 is 0, execute this statement and jump to LOOP1 |

SZINCA          [R]

operation:      Register R self plus 1, , Result into ACC, If the result is 0, then jump to the next statement,

                otherwise execution by sequence

cycle:          1or2

Affect status   No

bit:

example:

                SZINCA          R01                     ;R01+1→ACC

                JP              LOOP                    ; When R01 is not 0, execute this statement and jump to LOOP

                JP              LOOP1                   ; When R01 is 0, execute this statement and jump to LOOP1


SZINCR          [R]

operation:      Register R self plus 1, Result into R, If the result is 0, then jump the next statement,

                otherwise execution by sequence

cycle:          1or2

Affect flag bit:   No

example:

                SZINCR          R01                     ;R01+1→R01

                JP              LOOP                    ; When R01 is not 0, execute this statement and jump to LOOP

                JP              LOOP1                   ; When R01 is 0, execute this statement and jump to LOOP1


SZDECA          [R]

operation:      Register R self minus 1, Result into ACC, If the result is 0, then jump to the next statement,

                otherwise execution by sequence

cycle:          1or2

Affect status   No

bit:

example:

                SZDECA          R01                     ;R01-1→ACC

                JP              LOOP                    ; When R01 is not 0, execute this statement and jump to LOOP

                JP              LOOP1                   ; When R01 is 0, execute this statement and jump to LOOP1


SZDECR          [R]

operation:      Register R self minus 1, Result into R, If the result is 0, then jump to the next statement,

                otherwise execution by sequence

cycle:          1or2

Affect status   No

bit:

example:

                SZDECR          R01                     ;R01-1→R01

                JP              LOOP                    ; When R01 is not 0, execute this statement and jump to LOOP

                JP              LOOP1                   ; When R01 is 0, execute this statement and jump to LOOP1

TABLE        [R]

operation:        Look up table, look-up table low 8 bits into the R, high into the special register TABLE_SPH

cycle:        2

Affect    status    No
bit:

example:

| | | |
|---|---|---|
| LDIA | 01H | ; Assign 01H to ACC |
| LD | TABLE_SPH,A | ; ACC assignment to table High address, TABLE_SPH=1 |
| LDIA | 015H | ; Assign 15H to ACC |
| LD | TABLE_SPL,A | ; ACC assignment to table Low address, TABLE_SPL=15H |
| TABLE | R01 | ; Lookup Table address is 0115H, Execution result: TABLE_DATAH=12H, R01=34H |
| … | | |
| ORG | 0115H | |
| DW | 1234H | |

TABLEA

operation:        Look up table, look-up table low 8 bits into the ACC, high into the special register TABLE_SPH

cycle:        2

Affect    status    No
bit:

example:

| | | |
|---|---|---|
| LDIA | 01H | ; Assign 01H to ACC |
| LD | TABLE_SPH,A | ; ACC assignment to table High address, TABLE_SPH=1 |
| LDIA | 015H | ; Assign 15H to ACC |
| LD | TABLE_SPL,A | ; ACC assignment to table Low address, TABLE_SPL=15H |
| TABLEA | | ;Lookup Table address is 0115H, Execution result: TABLE_DATAH=12H, ACC=34H |
| … | | |
| ORG | 0115H | |
| DW | 1234H | |

TESTZ        [R]

operation:        Assign the value of register R0 to R0 to affect the Z flag

cycle:        1

Affect    status    Z
bit:

example:

| | | |
|---|---|---|
| TESTZ | R0 | ; Assign the value of register R0 to R0 to affect the Z flag |
| SZB | STATUS,Z | ; Judgment Z flag, If it is 0 then jump |
| JP | Add1 | ; If register R0 is 0 then jump to Add1 |
| JP | Add2 | ; If register R0 is not 0 then jump to Add2 |

XORIA              i

operation:        Immediate data Logical XOR operation with ACC and result in ACC

cycle:             1

Affect    status   Z
bit:

example:

| | | |
|---|---|---|
| LDIA | 0AH | ; Assign 0AH to ACC |
| XORIA | 0FH | ; Execution result: ACC=05H |

XORA              [R]

operation:        ACC logical XOR operation with register R, result into ACC

cycle:             1

Affect    status   Z
bit:

example:

| | | |
|---|---|---|
| LDIA | 0AH | ; Assign 0AH to ACC |
| LD | R01,A | ; ACC value assign to R01,R01=0AH |
| LDIA | 0FH | ; Assign 0FH to ACC |
| XORA | R01 | ; Execution result: ACC=05H |

XORR              [R]

operation:        ACC logical XOR operation with register R, result into R

cycle:             1

Affect    status   Z
bit:

example:

| | | |
|---|---|---|
| LDIA | 0AH | ; Assign 0AH to ACC |
| LD | R01,A | ; ACC value assign to R01,R01=0AH |
| LDIA | 0FH | ; Assign 0FH to ACC |
| XORR | R01 | ; Execution result: R01=05H |

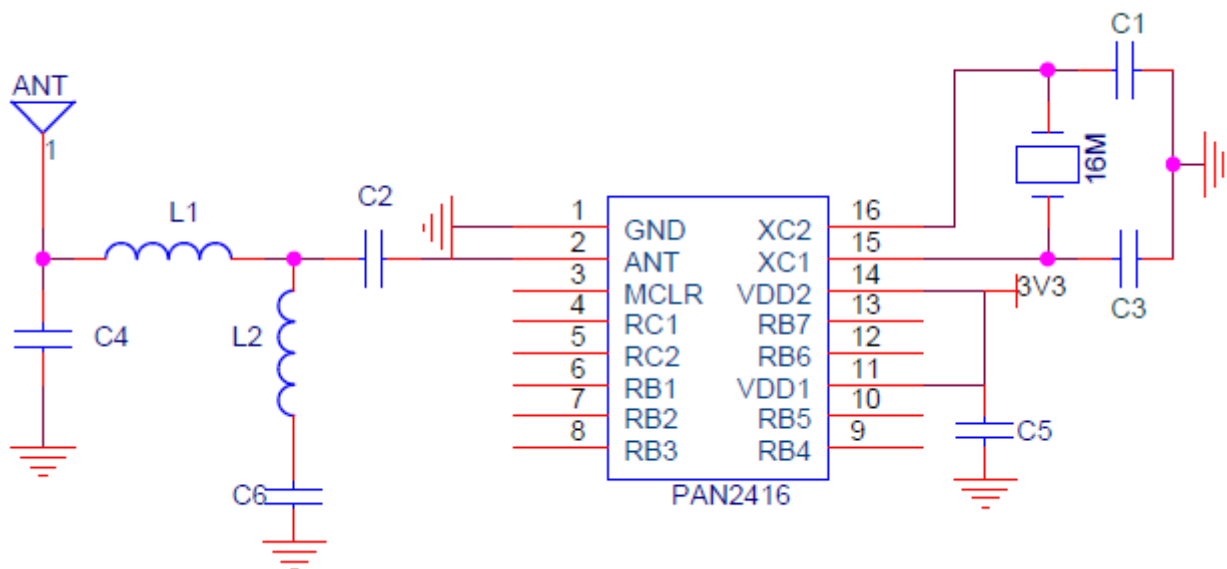# 23    Typical Application Circuit (Reference)



Figure 23-1 PAN2416AV application

Notes: RF matching part of the test for the single/double-sided test transmitter and receiver can be certified through the safety certification.

| BOM | Notes |
|---|---|
| C1 / C3 | Resonant capacitor, adjusting, Optional scope 15~36pF |
| C5 | 0.1uF |
| C2 | Recommended 3.3Pf, Optional scope 2~4pF |
| L1 | 5.6nH |
| L2 | 2.2nH |
| C4 | 0.5pF |
| C6 | 0.5pF |

# 24   Package Size

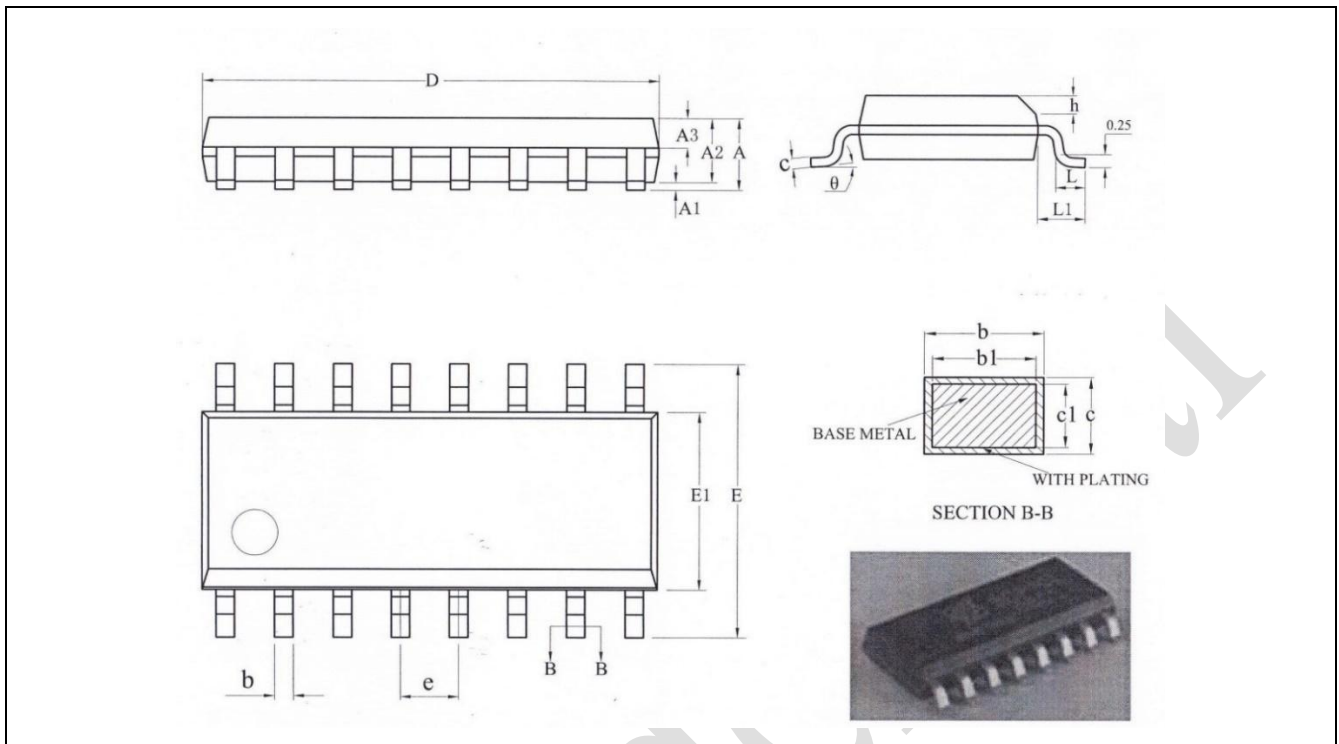

Figure 24-1 The SOP16 package size for PAN2416AV

Table 24-1 Package detail parameters for the SOP16

| SYMBOL | MIN | NOM | MAX |
|--------|-----|-----|-----|
| A | - | - | 1.75 |
| A1 | 0.10 | - | 0.225 |
| A2 | 1.30 | 1.40 | 1.50 |
| A3 | 0.60 | 0.65 | 0.70 |
| b | 0.39 | - | 0.47 |
| b1 | 0.38 | 0.41 | 0.44 |
| c | 0.20 | - | 0.24 |
| c1 | 0.19 | 0.20 | 0.21 |
| D | 9.80 | 9.90 | 10.00 |
| E | 5.80 | 6.00 | 6.20 |
| E1 | 3.80 | 3.90 | 4.00 |
| e | 1.27BSC | | |
| h | 0.25 | - | 0.50 |
| L | 0.50 | - | 0.80 |
| L1 | 1.05REF | | |
| Ø | 0 | - | 8° |

Note 1: Units of measure is millimeter.