# AT8A513H

## 6 I/O 8-bit EPROM-Based MCU

**Version 2.1**

**Aug. 26, 2025**

# Revision History

| Version | Date | Description | Modified Page |
|---|---|---|---|
| 1.0 | 2022/01/14 | Formal release. | - |
| 1.1 | 2022/02/28 | 1. Modified 3.3.4 CMPCR[3:2] initial value<br>2. Updated reference voltage Vref selection table | 22<br>41 |
| 1.2 | 2022/05/31 | 1. Updated PB3 description<br>2. Modified LVD table | 11<br>23, 39, 40 |
| 1.3 | 2022/07/29 | Update operating temperature range | 6, 8 |
| 1.4 | 2023/02/28 | Add LVD control flow | 40 |
| 1.5 | 2023/03/10 | Modified I_LRC deviation range | 74 |
| 1.6 | 2024/02/27 | Modified Pin Assignment description | 10 |
| 1.7 | 2024/07/05 | 1. Modified T1CR1 (Timer1 Control Register1) initial value.<br>2. Update Table 13 The reference voltage Vref selection table. | 24<br>41 |
| 1.8 | 2024/11/28 | Revised Comparator Voltage Reference Table. | 42 |
| 1.9 | 2025/06/11 | 1. Updated Comparator section, Optimized text in some chapters.<br>2. Optimized text in some chapters.<br>3. Update VIH value. | 39<br>22, 31, 32<br>73, 74, 75 |
| 2.0 | 2025/07/09 | Updated LVD voltage select table. | 10, 24, 45 |
| 2.1 | 2025/08/26 | 1. Updated Wide operating voltage range<br>2. Updated Pin Description<br>3. Vref hardware connection & LVD block diagram<br>4. Updated LVD voltage table.<br>5. Updated $F_{INST}$ define of VDD (Operating voltage)<br>6. Update Recommended Operating Voltage. | 6, 8<br>11<br>42<br>43<br>74<br>82 |

# Table of Contents

# 1. 概述

AT8A513H是以EPROM作為記憶體的 8 位元微控制器，專為多IO產品的應用而設計，例如遙控器、風扇/燈光控制或是遊樂器周邊等等。採用CMOS製程並同時提供客戶低成本、高性能等顯著優勢。AT8A513H核心建立在RISC精簡指令集架構可以很容易地做編輯和控制，共有 55 條指令。除了少數指令需要 2 個時序，大多數指令都是 1 個時序即能完成，可以讓使用者輕鬆地以程式控制完成不同的應用。因此非常適合各種中低記憶容量但又複雜的應用。

在I/O的資源方面，AT8A513H有 6 根彈性的雙向I/O腳，每個I/O腳都有單獨的暫存器控制為輸入或輸出腳。而且每一個I/O腳位都有附加的程式控制功能如上拉或下拉電阻或開漏極(Open-Drain) 輸出。此外針對紅外線搖控的產品方面，AT8A513H內建了可選擇頻率的紅外載波發射口。

AT8A513H有兩組計時器，可用系統頻率當作一般的計時的應用或者從外部訊號觸發來計數。另外AT8A513H提供 1 組 8 位元解析度的PWM輸出或者蜂鳴器輸出，可用來驅動馬達、LED、或蜂鳴器等等。

AT8A513H採用雙時鐘機制，高速振盪或者低速振盪都由內部RC振盪輸入。在雙時鐘機制下，AT8A513H可選擇多種工作模式如正常模式(Normal)、慢速模式(Slow mode)、待機模式(Standby mode) 與睡眠模式(Halt mode)可節省電力消耗延長電池壽命。

在省電的模式下如待機模式(Standby mode)與睡眠模式(Halt mode)中，有多種事件可以觸發中斷喚醒AT8A513H進入正常操作模式(Normal) 或 慢速模式(Slow mode) 來處理突發事件。

## 1.1 功能

- 寬廣的工作電壓：
  - 3.3V ~ 5.5V @ 20MHz/2T 。
  - 1.6V ~ 5.5V @ 4MHz/4T
- 寬廣的工作温度：-40°C ~ 85°C。
- 高達 ±5KV 的ESD。
- 內建 14 階準確的低電壓偵測電路功能。
- 1Kx14 bits EPROM。
- 48 bytes SRAM。
- 6 根可分別單獨控制輸入輸出方向的I/O腳(GPIO)、PB[5:0]。
- PB[3:0]可選擇輸入時使用內建上拉及下拉電阻。
- 輸入有三種組態可選(TTL/CMOS/Without Schmitt) 。
- PB[3]內建上拉電阻及輸出高推功能。
- PB[5:0]可選擇上拉電阻或開漏極輸出(Open-Drain)。
- 所有I/O腳輸出可選擇小灌電流(Small Sink Current)或一般灌電流(Normal Sink Current)。
- 所有I/O腳輸出可選擇小推電流(Small Drive Current)或一般推電流(Normal Drive Current)，除PB3 外。
- 8 層程式堆棧(Stack)。

- 存取資料有直接或間接定址模式。
- 一組 8 位元上數計時器(Timer0)包含可程式化的頻率預除線路。
- 一組 8 位元下數計時器(Timer1)可選重複載入或連續下數計時。
- 一個 8 位元的脈衝寬度調變輸出(PWM1)。
- 一個蜂鳴器輸出(BZ1)。
- 38/57KHz紅外線載波頻率可供選擇，同時載波之極性也可以根據數據作選擇。
- 內建上電復位電路(POR)。
- 內建低壓復位功能(LVR)。
- 內建看門狗計時(WDT)，可由程式韌體控制開關。
- 雙時鐘機制，系統可以隨時切換高速振盪或者低速振盪。
  - 高速振盪: I_HRC (1~20MHz內部高速RC振盪)
  - 低速振盪: I_LRC (內部 32KHz低速RC振盪)
- 四種工作模式可隨系統需求調整電流消耗：正常模式(Normal)、慢速模式(Slow mode)、待機模式(Standby mode) 與 睡眠模式(Halt mode)。
- 六種硬體中斷：
  - Timer0 溢位中斷。
  - Timer1 借位中斷。
  - WDT 中斷。
  - PB 輸入狀態改變中斷。
  - 外部中斷輸入。
  - 低電壓偵測中斷。
- AT8A513H在待機模式(Standby mode)下的六種喚醒中斷：
  - Timer0 溢位中斷。
  - Timer1 借位中斷。
  - WDT 中斷。
  - PB 輸入狀態改變中斷。
  - 外部中斷輸入。
  - 低電壓偵測中斷。
- AT8A513H在睡眠模式(Halt mode)下的三種喚醒中斷：
  - WDT 中斷。
  - PB 輸入狀態改變中斷。
  - 外部中斷輸入。

# 1. General Description

AT8A513H is an EPROM based 8-bit MCU tailored for I/O based applications like remote controllers, fan/light controller, game controllers, toy and various controllers. AT8A513H adopts advanced CMOS technology to provide customers remarkable solution with low cost and high performance benefits. RISC architecture is applied to AT8A513H and it provides 55 instructions. All instructions are executed in single instruction cycle except program branch and skip instructions which will take two instruction cycles. Therefore, AT8A513H is very suitable for those applications that are sophisticated but compact program size is required.

As AT8A513H address I/O type applications, it can provide 6 I/O pins for applications which require abundant input and output functionality. Moreover, each I/O pin may have additional features, like Pull-High/Pull-Low resistor and open-drain output type through programming. Moreover, AT8A513H has built-in infrared (IR) carrier generator with selectable IR carrier frequency and polarity for applications which demand remote control feature.

AT8A513H also provides 2 sets of timers which can be used as regular timer based on system oscillation or event counter with external trigger clock. Moreover, AT8A513H provides 1 set of 8-bit resolution Pulse Width Modulation (PWM) output and buzzer output in order to drive motor/LED and buzzer.

AT8A513H employs dual-clock oscillation mechanism, both high oscillation or low oscillation can be derived from internal resistor/capacitor oscillator. Moreover, based on dual-clock mechanism, AT8A513H provides kinds of operation mode like Normal mode, Slow mode, Standby mode and Halt mode in order to save power consumption and lengthen battery operation life.

While AT8A513H operates in Standby mode and Halt mode, kinds of event will issue interrupt requests and can wake-up AT8A513H to enter Normal mode and Slow mode in order to process urgent events.

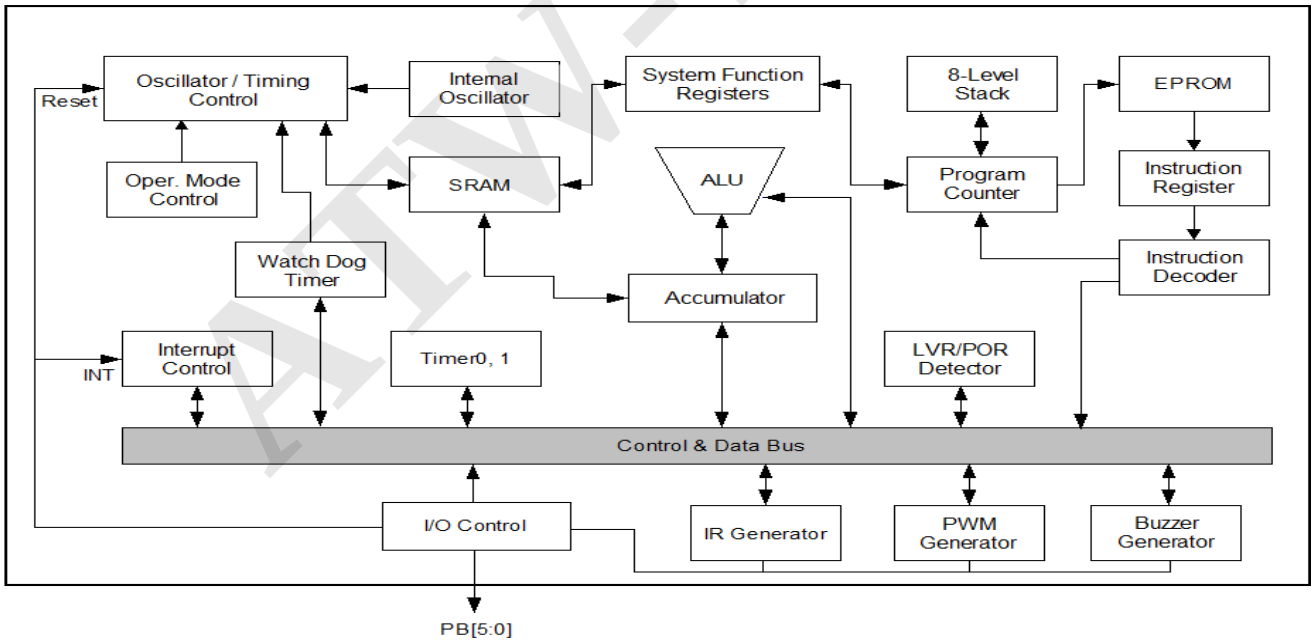
## 1.1 Features

- Wide operating voltage range:
  - 3.3V ~ 5.5V @ 20MHz/2T ◦
  - 1.6V ~ 5.5V @ 4MHz/4T
- Wide operating temperature: -40°C ~ 85°C.
- High ESD over ±5KV.
- Built-in high-precision Low-Voltage Detector (LVD).
- 1K x 14 bits EPROM.
- 48 bytes SRAM.
- 6 general purpose I/O pins (GPIO), PB[5:0], with independent direction control.
- PB[3:0] has features of Pull-High and Pull-Low resistor for input pin.
- PB[5:0] has features of Pull-High resistor, and open-drain output.

- $V_{IH}$ has three types to choose, which are $0.8V_{DD}$, $0.6V_{DD}$ and $0.5V_{DD}$ (No Schmitt). $V_{IL}$ has three types to choose, which are $0.3V_{DD}$, $0.2V_{DD}$ and $0.5V_{DD}$ (No Schmitt).
- PB[3] built-in Pull-High resistor and add output high function.
- 8-level hardware Stack.
- Direct and indirect addressing modes for data access.
- One 8-bit up-count timer (Timer0) with programmable pre-scalar.
- One 8-bit reload or continuous down-count timers (Timer1).
- One 8-bit resolution PWM (PWM1) output.
- One buzzer (BZ1) output.
- Selectable 38/57KHz IR carrier frequency and high/low polarity according to data value.
- Built-in Power-On Reset (POR).
- Built-in Low-Voltage Reset (LVR).
- Built-in Watch-Dog Timer (WDT) enabled/disabled by firmware control.
- Dual-clock oscillation: System clock can switch between high oscillation and low oscillation.
  - High oscillation: I_HRC (Internal High Resistor/Capacitor Oscillator ranging from 1M~20MHz)
  - Low oscillation: I_LRC (Internal 32KHz oscillator)
- Four kinds of operation mode to reduce system power consumption:
  - Normal mode, Slow mode, Standby mode and Halt mode.
- Six hardware interrupt events:
  - Timer0 overflow interrupt.
  - Timer1 underflow interrupt.
  - WDT timeout interrupt.
  - PB input change interrupt.
  - External interrupt.
  - LVD interrupt.
- Six interrupt events to wake-up AT8A513H from Standby mode:
  - Timer0 overflow interrupt.
  - Timer1 underflow interrupt.
  - WDT timeout interrupt.
  - PB input change interrupt.
  - External interrupt.
  - LVD interrupt.
- Three interrupt events to wake-up AT8A513H from Halt mode:
  - WDT timeout interrupt.

- PB input change interrupt.
- External interrupt.

## 1.2 Block Diagram




## 1.3 The Main Differences Between AT8A513H and AT8A513F

| Item | Function | AT8A513H | AT8A513F |
|---|---|---|---|
| 1 | PB3 Output High | Yes (by option) | -- |
| 2 | Comparator Output to PB2 | Yes | -- |
| 3 | Comparator structure | shared with LVD | --- |
| 4 | Low Voltage Detector (LVD) | 14 levels | 11 levels |

## 1.4 Pin Assignment

AT8A513H provides two kinds of package type which are SOP8 and SOT23-6.

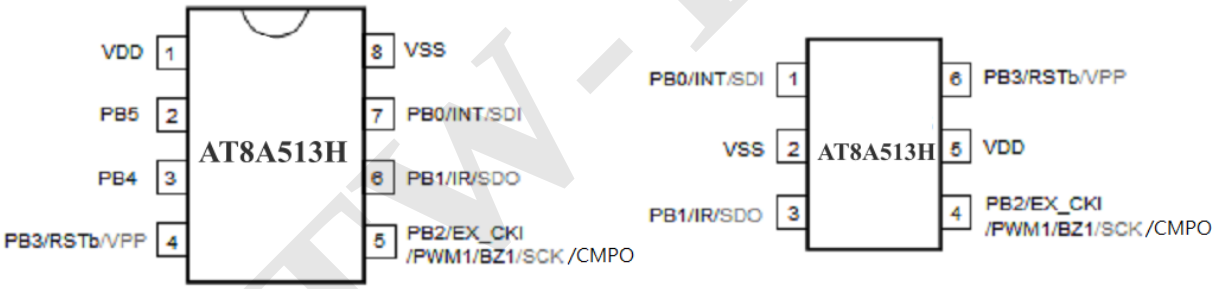



Figure 1 Package pin assignment

## 1.5 Pin Description

| Pin Name | I/O | Description |
|---|---|---|
| PB0<br>INT<br>SDI<br>CMP+ | I/O | PB0 is a bidirectional I/O pin.<br>PB0 is input pin of external interrupt when EIS=1 & INTIE=1.<br>PB0 can be programming pad SDI.<br>PB0 can be comparator non-inverse input. |
| PB1<br>IR<br>SDO<br>CMP- | I/O | PB1 is a bidirectional I/O pin.<br>If IR mode is enabled, this pin is IR carrier output.<br>PB1 can be programming pad SDO.<br>PB1 can be comparator inverse input. |
| PB2<br>EX_CKI<br>PWM1<br>BZ1<br>SCK<br>Comparator | I/O | PB2 is a bidirectional I/O pin.<br>PB2 can also be timer clock source EX_CKI.<br>PB2 can also be PWM output.<br>PB2 can also be BUZZER output.<br>PB2 can also be programming pad SCK.<br>PB2 can also be comparator output |
| PB3<br>RSTb<br>Vpp | I/O | PB3 is a bidirectional I/O pin.<br>PB3 can be reset pin RSTb. If RSTb pin is low, it will reset AT8A513H.<br>PB3 can be programming pad VPP. |
| PB4<br>$F_{INST}$_OUT | I/O | PB4 is a bidirectional I/O pin.<br>PB4 also can be output of instruction clock. |
| PB5 | I/O | PB5 is a bidirectional I/O pin. |
| VDD | - | Positive power supply. |
| VSS | - | Ground. |

## 2. Memory Organization

AT8A513H memory is divided into two categories: one is program memory and the other is data memory.

### 2.1 Program Memory

The program memory space of AT8A513H is 1K words. Therefore, the Program Counter (PC) is 10-bit wide in order to address any location of program memory.

Some locations of program memory are reserved as interrupt entrance. Power-On Reset vector is located at 0x000. Software interrupt vector is located at 0x001. Internal and external hardware interrupt vector is located at 0x008.

AT8A513H provides instruction CALL, GOTOA, CALLA to address 256 location of program space. AT8A513H provides instruction GOTO to address 512 location of program space. AT8A513H also provides instructions LCALL and LGOTO to address any location of program space.

When a call or interrupt is happening, next ROM address is written to top of the stack, when RET, RETIA or RETIE instruction is executed, the top of stack data is read and load to PC.

AT8A513H program ROM address 0x3FE~0x3FF are reserved space, if user tries to write code in these addresses will get unexpected false functions.

AT8A513H program ROM address 0x00E~0x00F are preset rolling code can be released and used as normal program space.

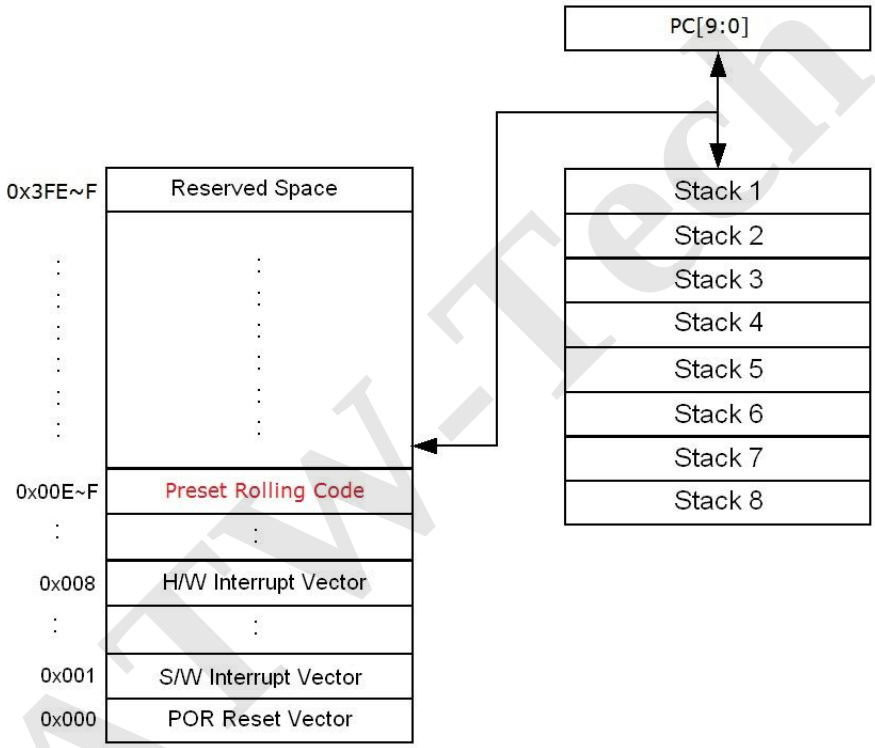



Figure 2 Program Memory Address Mapping

## 2.2 Data Memory

According to instructions used to access data memory, the data memory can be divided into three kinds of categories: one is R-page Special-function Register (SFR) + General Purpose Register (GPR), another is F-page SFR and the other is S-page SFR. GPR are made of SRAM and user can use them to store variables or intermediate results.

R-page data memory is divided into 4 banks and can be accessed directly or indirectly through a SFR register which is File Select Register (FSR). FSR[7:6] are used as Bank register BK[1:0] to select one bank out of the 4 banks.

R-page register can be divided into addressing mode: direct addressing mode and indirect addressing mode.

The indirect addressing mode of data memory access is described in the following graph. This indirect addressing mode is implied by accessing register INDF. The bank selection is determined by FSR[7:6] and the location selection is from FSR[5:0].

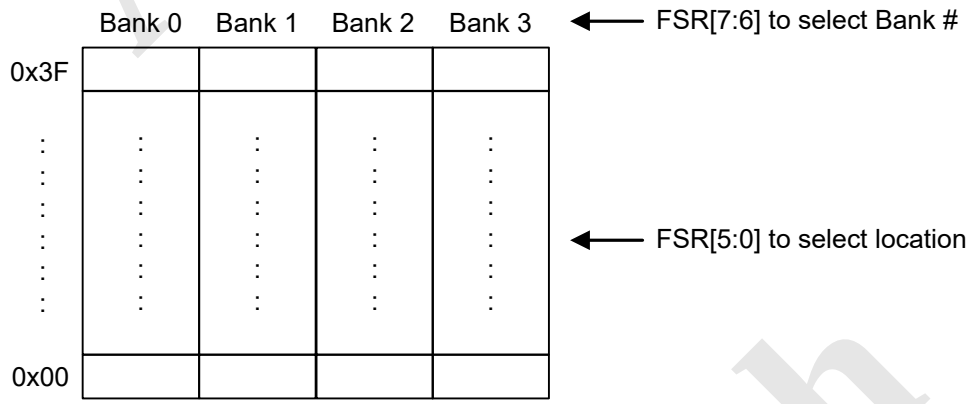



Figure 3 Indirect Addressing Mode of Data Memory Access

The direct addressing mode of data memory access is described below. The bank selection is determined by FSR[7:6] and the location selection is from instruction op-code[5:0] immediately.

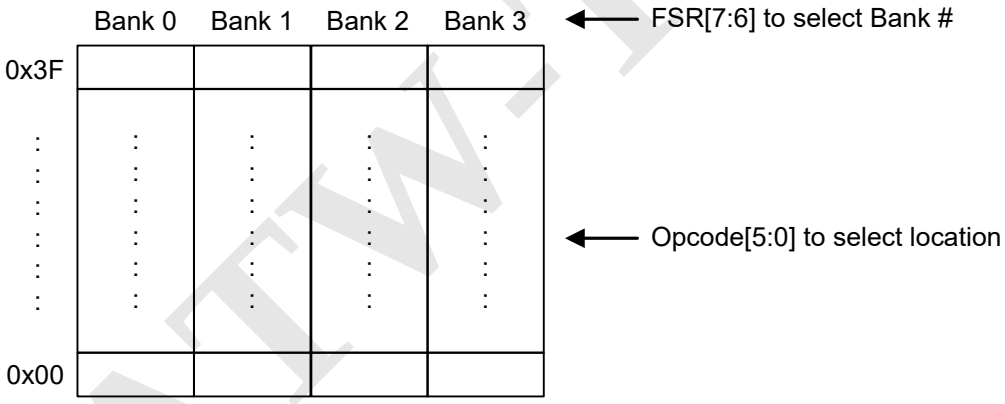



Figure 4 Direct Addressing Mode of Data Memory Access

R-page SFR can be accessed by general instructions like arithmetic instructions and data movement instructions. The R-page SFR occupy address from 0x0 to 0xF of Bank 0. However, the same address range of Bank 1, Bank 2 and Bank 3 are mapped back to Bank 0. In other words, R-page SFR physically existed at Bank 0. The GPR physically occupy address from 0x10 to 0x3F of Bank and other banks in address from 0x10 to 0x3F are mapped back as the Table 1 shows.

The AT8A513H register name and address mapping of R-page SFR are described in the following table.

<table>
<tr><th>FSR[7:6]<br>Address</th><th>00<br>(Bank 0)</th><th>01<br>(Bank 1)</th><th>10<br>(Bank 2)</th><th>11<br>(Bank 3)</th></tr>
<tr><td>0x0</td><td>INDF</td><td colspan="3" rowspan="16">The same mapping as Bank 0</td></tr>
<tr><td>0x1</td><td>TMR0</td></tr>
<tr><td>0x2</td><td>PCL</td></tr>
<tr><td>0x3</td><td>STATUS</td></tr>
<tr><td>0x4</td><td>FSR</td></tr>
<tr><td>0x5</td><td>-</td></tr>
<tr><td>0x6</td><td>PORTB</td></tr>
<tr><td>0x7</td><td>-</td></tr>
<tr><td>0x8</td><td>PCON</td></tr>
<tr><td>0x9</td><td>BWUCON</td></tr>
<tr><td>0xA</td><td>PCHBUF</td></tr>
<tr><td>0xB</td><td>BPLCON</td></tr>
<tr><td>0xC</td><td>BPHCON</td></tr>
<tr><td>0xD</td><td>-</td></tr>
<tr><td>0xE</td><td>INTE</td></tr>
<tr><td>0xF</td><td>INTF</td></tr>
<tr><td>0x10 ~ 0x1F</td><td>General Purpose Register</td><td colspan="3">Unused</td></tr>
<tr><td>0x20 ~ 0x3F</td><td>General Purpose Register</td><td colspan="3">Unused</td></tr>
</table>

Table 1 R-page SFR Address Mapping

F-page SFR can be accessed only by instructions IOST and IOSTR. S-page SFR can be accessed only by instructions SFUN and SFUNR. FSR[7:6] bank select bits are ignored while F-page and S-page register is accessed. The register name and address mapping of F-page and S-page are depicted in the following table.

| SFR Category / Address | F-page SFR | S-page SFR |
|---|---|---|
| 0x0 | - | TMR1 |
| 0x1 | - | T1CR1 |
| 0x2 | - | T1CR2 |
| 0x3 | - | PWM1DUTY |
| 0x4 | - | PS1CV |
| 0x5 | - | BZ1CR |
| 0x6 | IOSTB | IRCR |
| 0x7 | - | TBHP |
| 0x8 | - | TBHD |
| 0x9 | - | - |
| 0xA | PS0CV | - |
| 0xB | - | - |
| 0xC | BODCON | - |
| 0xD | - | - |
| 0xE | CMPCR | - |
| 0xF | PCON1 | OSCCR |

Table 2 F-page and S-page SFR Address Mapping

# 3. Function Description

This chapter will describe the detailed operations of AT8A513H.

## 3.1 R-page Special Function Register

### 3.1.1 INDF (Indirect Addressing Register)

| Name | SFR Type | Addr. | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|---|---|
| INDF | R | 0x0 | INDF[7:0] | | | | | | | |
| R/W Property | | | R/W | | | | | | | |
| Initial Value | | | xxxxxxxx | | | | | | | |

The register INDF is not physically existed and it is used as indirect addressing mode. Any instruction accessing INDF actually accesses the register pointed by register FSR

### 3.1.2 TMR0 (Timer0 Register)

| Name | SFR Type | Addr. | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|---|---|
| TMR0 | R | 0x1 | TMR0[7:0] | | | | | | | |
| R/W Property | | | R/W | | | | | | | |
| Initial Value | | | xxxxxxxx | | | | | | | |

When read the register TMR0, it actually read the current running value of Timer0.

Write the register TMR0 will change the current value of Timer0.

Timer0 clock source can be from instruction clock $F_{INST}$, or from external pin EX_CKI, or from Low Oscillator Frequency according to T0MD and configuration word setting.

### 3.1.3 PCL (Low Byte of PC[9:0])

| Name | SFR Type | Addr. | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|---|---|
| PCL | R | 0x2 | PCL[7:0] | | | | | | | |
| R/W Property | | | R/W | | | | | | | |
| Initial Value | | | 0x00 | | | | | | | |

The register PCL is the least significant byte (LSB) of 10-bit PC. PCL will be increased by one after one instruction is executed except some instructions which will change PC directly. The high byte of PC, i.e. PC[9:8], is not directly accessible. Update of PC[9:8] must be done through register PCHBUF.

For GOTO instruction, PC[8:0] is from instruction word and PC[9] is loaded from PCHBUF[1]. For CALL instruction, PC[7:0] is from instruction word and PC[9:8] is loaded from PCHBUF[1:0]. Moreover the next PC address, i.e. PC+1, will push onto top of Stack. For LGOTO instruction, PC[9:0] is from instruction word.

For LCALL instruction, PC[9:0] is from instruction word. Moreover the next PC address, i.e. PC+1, will push onto top of Stack.

### 3.1.4 STATUS (Status Register)

| Name | SFR Type | Addr. | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|---|---|
| STATUS | R | 0x3 | GP7 | GP6 | GP5 | /TO | /PD | Z | DC | C |
| R/W Property | | | R/W | R/W | R/W | R/W(*2) | R/W(*1) | R/W | R/W | R/W |
| Initial Value | | | 0 | 0 | 0 | 1 | 1 | X | X | X |

The register STATUS contains result of arithmetic instructions and reasons to cause reset.

**C:** Carry/Borrow bit

C=1, carry is occurred for addition instruction or borrow is not occurred for subtraction instruction.

C=0, carry is not occurred for addition instruction or borrow is occurred for subtraction instruction.

**DC:** Half Carry/half Borrow bit

DC=1, carry from the 4th LSB is occurred for addition instruction or borrow from the 4th LSB is not occurred for subtraction instruction.

DC=0, carry from the 4th LSB is not occurred for addition instruction or borrow from the 4th LSB is occurred for subtraction instruction.

**Z:** Zero bit

Z=1, result of logical operation is zero.

Z=0, result of logical operation is not zero.

**/PD:** Power down flag bit

/PD=1, after power-up or after instruction CLRWDT is executed.

/PD=0, after instruction SLEEP is executed.

**/TO:** Time overflow flag bit

/TO=1, after power-up or after instruction CLRWDT or SLEEP is executed.

/TO=0, WDT timeout is occurred.

GP7, GP6, GP5: General purpose read/write register bit.

(*1) can be cleared by sleep instruction.

(*2) can be set by clrwdt instruction.

### 3.1.5 FSR (Register File Selection Register)

| Name | SFR Type | Addr. | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|---|---|
| FSR | R | 0x4 | BK[1:0] | | FSR[5:0] | | | | | |
| R/W Property | | | R/W | | | | | | | |
| Initial Value | | | 0 | 0 | X | X | X | X | X | X |

**FSR[5:0]:** Select one register out of 64 registers of specific Bank.

**BK[1:0] must be 00.** For AT8A513H, bank register is not used, because there's only one bank

.

### 3.1.6 PortB (PortB Data Register)

| Name | SFR Type | Addr. | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|---|---|
| PortB | R | 0x6 | GP7 | GP6 | PB5 | PB4 | PB3 | PB2 | PB1 | PB0 |
| R/W Property | | | R/W | | | | | | | |
| Initial Value | | | Data latch value is xxxxxx, read value is xxxxxx port value(PB5~PB0) | | | | | | | |

While reading PortB, it will get the status of the specific pin if that pin is configured as input pin. However, if that pin is configured as output pin, whether it will get the status of the pin or the value of the corresponding output data latch is depend on the configuration word RD_OPT. While writing to PortB, data is written to PB's output data latch.

**GP7, GP6:** General purpose read/write register bit.

### 3.1.7 PCON (Power Control Register)

| Name | SFR Type | Addr. | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|---|---|
| PCON | R | 0x8 | WDTEN | EIS | LVDEN | GP4 | LVREN | CMPEN | GP1 | GP0 |
| R/W Property | | | R/W | | | | | | | |
| Initial Value | | | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |

**GP5~0:** General read/write register bits.

**LVREN:** Enable/disable LVR.

LVREN=1, enable LVR.

LVREN=0, disable LVR.

**EIS:** External interrupt select bit

EIS=1, PB0 is external interrupt.

EIS=0, PB0 is GPIO.

**LVDEN:** Enable/disable LVD.

LVDEN=1, enable LVD.

LVDEN=0, disable LVD.

**WDTEN:** Enable/disable WDT.

WDTEN=1, enable WDT.

WDTEN=0, disable WDT.

**CMPEN:** Enable/disable CMP

CMPEN=1, enable CMP.

CMPEN=0, disable CMP.

### 3.1.8 BWUCON (PortB Wake-up Control Register)

| Name | SFR Type | Addr. | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|---|---|
| BWUCON | R | 0x9 | - | - | WUPB5 | WUPB4 | WUPB3 | WUPB2 | WUPB1 | WUPB0 |
| R/W Property | | | - | - | R/W | R/W | R/W | R/W | R/W | R/W |
| Initial Value | | | X | X | 1 | 1 | 1 | 1 | 1 | 1 |

**WUPBx:** Enable/disable PBx wake-up function, 0 ≤ x ≤ 5.

WUPBx=1, enable PBx wake-up function.

WUPBx=0, disable PBx wake-up function.

### 3.1.9 PCHBUF (High Byte of PC)

| Name | SFR Type | Addr. | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|---|---|
| PCHBUF | R | 0xA | - | - | - | - | - | GP5 | PCHBUF[1:0] | |
| R/W Property | | | - | - | - | - | - | R/W | W | |
| Initial Value | | | X | X | X | X | X | 0 | 00 | |

**PCHBUF[1:0]:** Buffer of the 9th bit, 8th bit of PC.

**GP5:** General read/write register bit.

### 3.1.10 BPLCON (PortB Pull-Low Resistor Control Register)

| Name | SFR Type | Addr. | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|---|---|
| BPLCON | R | 0xB | /PLPB3 | /PLPB2 | /PLPB1 | /PLPB0 | - | - | - | - |
| R/W Property | | | R/W | R/W | R/W | R/W | - | - | - | - |
| Initial Value | | | 1 | 1 | 1 | 1 | X | X | X | X |

**/PLPBx:** Disable/enable PBx Pull-Low resistor, 0 ≤ x ≤ 3.

/PLPBx=1, disable PBx Pull-Low resistor.

/PLPBx=0, enable PBx Pull-Low resistor.

### 3.1.11 BPHCON (PortB Pull-High Resistor Control Register)

| Name | SFR Type | Addr | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|---|---|
| BPHCON | R | 0xC | - | - | /PHPB5 | /PHPB4 | /PHPB3 | /PHPB2 | /PHPB1 | /PHPB0 |
| R/W Property | | | - | - | R/W | R/W | R/W | R/W | R/W | R/W |
| Initial Value | | | X | X | 1 | 1 | 1 | 1 | 1 | 1 |

**/PHPBx:** Disable/enable PBx Pull-High resistor, 0 ≤ x ≤ 5.

/PHPBx=1, disable PBx Pull-High resistor.

/PHPBx=0, enable PBx Pull-High resistor.

### 3.1.12 INTE (Interrupt Enable Register)

| Name | SFR Type | Addr. | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|---|---|
| INTE | R | 0xE | - | WDTIE | - | LVDIE | T1IE | INTIE | PBIE | T0IE |
| R/W Property | | | - | R/W | - | R/W | R/W | R/W | R/W | R/W |
| Initial Value | | | X | 0 | X | 0 | 0 | 0 | 0 | 0 |

**T0IE:** Timer0 overflow interrupt enable bit.

T0IE=1, enable Timer0 overflow interrupt.

T0IE=0, disable Timer0 overflow interrupt.

**PBIE:** PortB input change interrupt enable bit.

PBIE=1, enable PortB input change interrupt.

PBIE=0, disable PortB input change interrupt.

**INTIE:** External interrupt enable bit.

INTIE=1, enable external interrupt.

INTIE=0, disable external interrupt.

**T1IE:** Timer1 underflow interrupt enable bit.

T1IE=1, enable Timer1 underflow interrupt.

T1IE=0, disable Timer1 underflow interrupt.

**LVDIE:** Low-voltage detector interrupt enable bit.

LVDIE=1, enable low-voltage detector interrupt.

LVDIE=0, disable low-voltage detector interrupt.

**WDTIE:** WDT timeout interrupt enable bit.

WDTIE=1, enable WDT timeout interrupt.

WDTIE=0, disable WDT timeout interrupt.

### 3.1.13 INTF (Interrupt Flag Register)

| Name | SFR Type | Addr. | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|---|---|
| INTF | R | 0xF | - | WDTIF | - | LVDIF | T1IF | INTIF | PBIF | T0IF |
| R/W Property | | | - | R/W | - | R/W | R/W | R/W | R/W | R/W |
| Initial Value(note*) | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**T0IF:** Timer0 overflow interrupt flag bit.

T0IF=1, Timer0 overflow interrupt is occurred.

T0IF must be clear by firmware.

**PBIF:** PortB input change interrupt flag bit.

PBIF=1, PortB input change interrupt is occurred.

PBIF must be clear by firmware.

**INTIF:** External interrupt flag bit.

INTIF=1, external interrupt is occurred.

INTIF must be clear by firmware.

**T1IF:** Timer1 underflow interrupt flag bit.

T1IF=1, Timer1 underflow interrupt is occurred.

T1IF must be clear by firmware.

**LVDIF:** Low-voltage detector interrupt flag bit.

LVDIF=1, Low-voltage detector interrupt is occurred.

LVDIF must be clear by firmware.

**WDTIF:** WDT timeout interrupt flag bit.

WDTIF=1, WDT timeout interrupt is occurred.

WDTIF must be clear by firmware.

***Note: When corresponding INTE bit is not enabled, the read interrupt flag is 0.***

## 3.2 T0MD Register

T0MD is a readable/writeable register which is only accessed by instruction T0MD / T0MDR.

| Name | SFR Type | Addr. | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|---|---|
| T0MD | - | - | LCKTM0 | INTEDG | T0CS | T0CE | PS0WDT | PS0SEL[2:0] | | |
| R/W Property | | | R/W | | | | | | | |
| Initial Value(note*) | | | 0 | 0 | 1 | 1 | 1 | 111 | | |

**PS0SEL[2:0]:** Prescaler0 dividing rate selection. The rate depends on Prescaler0 is assigned to Timer0 or WDT. When Prescaler0 is assigned to WDT, the dividing rate is dependent on which timeout mechanism is selected.

| PS0SEL[2:0] | Dividing Rate | | |
|---|---|---|---|
| | PS0WDT=0 (Timer0) | PS0WDT=1 (WDT Reset) | PS0WDT=1 (WDT Interrupt) |
| 000 | 1:2 | 1:1 | 1:2 |
| 001 | 1:4 | 1:2 | 1:4 |
| 010 | 1:8 | 1:4 | 1:8 |
| 011 | 1:16 | 1:8 | 1:16 |
| 100 | 1:32 | 1:16 | 1:32 |
| 101 | 1:64 | 1:32 | 1:64 |
| 110 | 1:128 | 1:64 | 1:128 |
| 111 | 1:256 | 1:128 | 1:256 |

Table 3 Prescaler0 Dividing Rate

**PS0WDT:** Prescaler0 assignment.

PS0WDT=1, Prescaler0 is assigned to WDT.

PS0WDT=0, Prescaler0 is assigned to Timer0.

***Note: Always set PS0WDT and PS0SEL[2:0] before enabling watchdog or timer interrupt, or reset or interrupt may be falsely triggered.***

**T0CE:** Timer0 external clock edge selection.

T0CE=1, Timer0 will increase one while high-to-low transition occurs on pin EX_CKI.

T0CE=0, Timer0 will increase one while low-to-high transition occurs on pin EX_CKI.

***Note: T0CE is also applied to Low Oscillator Frequency as timer0 clock source condition.***

**T0CS:** Timer0 clock source selection.

T0CS=1, External clock on pin EX_CKI or I_LRC is selected.

T0CS=0, Instruction clock $F_{INST}$ is selected.

**INTEDG:** Edge selection of external interrupt.

INTEDG=1, INTIF will be set while rising edge occurs on pin PB0.

INTEDG=0, INTIF will be set while falling edge occurs on pin PB0.

**LCKTM0:** When T0CS=1, timer 0 clock source can be optionally selected to be low-frequency oscillator.

T0CS=0, Instruction clock $F_{INST}$ is selected as timer0 clock source.

T0CS=1, LCKTM0=0, external clock on pin EX_CKI is selected as timer0 clock source.

T0CS=1, LCKTM0=1, I_LRC output replaces pin EX_CKI as timer0 clock source.

***Note: For more detail descriptions of timer0 clock source select, please see timer0 section.***

## 3.3 F-page Special Function Register

### 3.3.1 IOSTB (PortB I/O Control Register)

| Name | SFR Type | Addr. | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|---|---|
| IOSTB | F | 0x6 | GP7 | GP6 | IOPB5 | IOPB4 | IOPB3 | IOPB2 | IOPB1 | IOPB0 |
| R/W Property | | | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Initial Value | | | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |

**IOPBx:** PBx I/O mode selection, 0 ≤ x ≤ 5.

IOPBx=1, PBx is input mode.

IOPBx=0, PBx is output mode.

**GP7, GP6:** General purpose register bit.

### 3.3.2 PS0CV (Prescaler0 Counter Value Register)

| Name | SFR Type | Addr. | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|---|---|
| PS0CV | F | 0xA | PS0CV[7:0] | | | | | | | |
| R/W Property | | | R | | | | | | | |
| Initial Value | | | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

While reading PS0CV, it will get current value of Prescaler0 counter.

### 3.3.3 BODCON (PortB Open-Drain Control Register)

| Name | SFR Type | Addr | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|---|---|
| BODCON | F | 0xC | - | - | ODPB5 | ODPB4 | ODPB3 | ODPB2 | ODPB1 | ODPB0 |
| R/W Property | | | - | - | R/W | R/W | R/W | R/W | R/W | R/W |
| Initial Value | | | X | X | 0 | 0 | 0 | 0 | 0 | 0 |

**ODPBx:** Enable/disable open-drain of PBx, 0 ≤ x ≤ 5.

ODPBx=1, enable open-drain of PBx.

ODPBx=0, disable open-drain of PBx.

### 3.3.4 CMPCR (Comparator voltage select Control Register)

| Name | SFR Type | Addr. | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|---|---|
| CMPCR | F | 0xE | GP7 | RBIAS_H | RBIAS_L | CMPF_INV | PS1 | PS0 | NS1 | NS0 |
| R/W Property | | | R/W | | | | | | | |
| Initial Value | | | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |

**NS[1:0]:** Comparator inverting input select.

| NS[1:0] | Inverting input |
|---|---|
| 00 | PB1 |
| 01 | -- |
| 10 | Bandgap (0.65V) |
| 11 | Vref |

**PS[1:0]:** Comparator non-inverting input select

| PS[1:0] | Non-inverting input |
|---|---|
| 00 | PB0 |
| 01 | -- |
| 10 | Vref |
| 11 | --- |

**CMPF_INV:** Comparator output inverse control bit.

CMPF_INV = 1, Inverse comparator output.

CMPF_INV = 0, Non-inverse comparator output.

**RBIAS_L, RBIAS_H:** Set corresponding voltage reference levels

***(please refer to chapter 3.10.1)***

### 3.3.5 PCON1 (Power Control Register1)

| Name | SFR Type | Addr. | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|---|---|
| PCON1 | F | 0xF | GIE | LVDOUT | LVDS3 | LVDS2 | LVDS1 | LVDS0 | GP1 | T0EN |
| R/W Property | | | R/W(1*) | R | R/W | R/W | R/W | R/W | R/W | R/W |
| Initial Value | | | 0 | X | 1 | 1 | 1 | 1 | 0 | 1 |

**T0EN:** Enable/disable Timer0.

T0EN=1, enable Timer0.

T0EN=0, disable Timer0.

**GIE:** Global interrupt enable bit.

GIE=1, enable all unmasked interrupts.

GIE=0, disable all interrupts.

**LVDOUT:** Low voltage detector output, read-only.

**LVDS3~0:** Select one of the 14 LVD voltages.

| RBIAS[H:L] | LVRS[3:0] | LVD Voltage (V) |
|---|---|---|
| 00 | 0001 | 2.20 |
| 00 | 0010 | 2.40 |
| 00 | 1011 | 2.60 |
| 00 | 0011 | 2.80 |
| 00 | 0100 | 2.90 |
| 00 | 0101 | 3.00 |
| 00 | 1101 | 3.15 |
| 00 | 0110 | 3.30 |
| 00 | 1110 | 3.45 |
| 00 | 0111 | 3.60 |
| 00 | 1111 | 3.75 |
| 00 | 1000 | 3.90 |
| 00 | 1010 | 4.05 |
| 00 | 1100 | 4.15 |

Table4 LVD voltage select

**The hysteresis window between the VDD voltage changing from high to low and the VDD voltage changing from low to high is about ±0.1V.**

**GP1:** General purpose read/write register.

**(1*)** : set by instruction ENI, clear by instruction DISI, read by instruction IOSTR.

## 3.4 S-page Special Function Register

### 3.4.1 TMR1 (Timer1 Register)

| Name | SFR Type | Addr. | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|---|---|
| TMR1 | S | 0x0 | TMR1[7:0] | | | | | | | |
| R/W Property | | | R/W | | | | | | | |
| Initial Value | | | XXXXXXXX | | | | | | | |

When reading register TMR1, it will obtain current value of 8-bit down-count Timer1. When writing register TMR1, it will both write data to timer1 reload register and update Timer1 current content.

### 3.4.2 T1CR1 (Timer1 Control Register1)

| Name | SFR Type | Addr. | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|---|---|
| T1CR1 | S | 0x1 | PWM1OEN | PWM1OAL | - | - | - | T1OS | T1RL | T1EN |
| R/W Property | | | R/W | R/W | - | - | - | R/W | R/W | R/W |
| Initial Value | | | 0 | 0 | X | X | X | 0 | 0 | 0 |

This register is used to configure Timer1 functionality.

**T1EN:** Enable/disable Timer1.

T1EN=1, enable Timer1.

T1EN=0, disable Timer1.

**T1RL:**Configure Timer1 down-count mechanism while Non-Stop mode is selected (T1OS=0).

T1RL=1, initial value is reloaded from reload register TMR1.

T1RL=0, continuous down-count from 0xFF when underflow is occurred.

**T1OS:** Configure Timer1 operating mode while underflow is reached.

T1OS=1, One-Shot mode. Timer1 will count once from the initial value to 0x00.

T1OS=0, Non-Stop mode. Timer1 will keep down-count after underflow.

| T1OS | T1RL | Timer1 Down-Count Functionality |
|---|---|---|
| 0 | 0 | Timer1 will count from reload value down to 0x00.<br>When underflow is reached, 0xFF is reloaded and continues down-count. |
| 0 | 1 | Timer1 will count from reload value down to 0x00.<br>When underflow is reached, reload value is reloaded and continues to down-count. |
| 1 | x | Timer1 will count from initial value down to 0x00.<br>When underflow is reached, Timer1 will stop down-count. |

Table 5 Timer1 Functionality

**PWM1OAL:** Define PWM1 output active state.

PWM1OAL=1, PWM1 output is active low.

PWM1OAL=0, PWM1 output is active high.

**PWM1OEN:** Enable/disable PWM1 output.

PWM1OEN=1, PWM1 output will be present on PB2.

PWM1OEN=0, PB2 is GPIO.

### 3.4.3 T1CR2 (Timer1 Control Register2)

| Name | SFR Type | Addr. | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|---|---|
| T1CR2 | S | 0x2 | - | - | T1CS | T1CE | /PS1EN | PS1SEL[2:0] | | |
| R/W Property | | | - | - | R/W | R/W | R/W | R/W | R/W | R/W |
| Initial Value | | | X | X | 1 | 1 | 1 | 1 | 1 | 1 |

This register is used to configure Timer1 functionality.

**PS1SEL[2:0]:** Prescaler1 dividing rate selection.

| PS1SEL[2:0] | Dividing Rate |
|---|---|
| 000 | 1:2 |
| 001 | 1:4 |
| 010 | 1:8 |
| 011 | 1:16 |
| 100 | 1:32 |
| 101 | 1:64 |
| 110 | 1:128 |
| 111 | 1:256 |

Table 6 Prescaler1 Dividing Rate

***Note: Always set PS1SEL[2:0] at /PS1EN=1, or interrupt may be falsely triggered.***

**/PS1EN:** Disable/enable Prescaler1.

/PS1EN=1, disable Prescaler1.

/PS1EN=0, enable Prescaler1.

**T1CE:** Timer1 external clock edge selection.

T1CE=1, Timer1 will decrease one while high-to-low transition occurs on pin EX_CKI.

T1CE=0, Timer1 will decrease one while low-to-high transition occurs on pin EX_CKI.

**T1CS:** Timer1 clock source selection.

T1CS=1, External clock on pin EX_CKI is selected.

T1CS=0, Instruction clock is selected.

### 3.4.4 PWM1DUTY (PWM1 Duty Register)

| Name | SFR Type | Addr. | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|---|---|
| PWM1DUTY | S | 0x3 | PWM1DUTY[7:0] | | | | | | | |
| R/W Property | | | W | | | | | | | |
| Initial Value | | | XXXXXXXX | | | | | | | |

This register is write-only. After Timer1 is enabled and start down-count, PWM1 output will keep at inactive state. While Timer1 value is equal to PWM1DUTY, PWM1 output will become active state until underflow is occurred.

Moreover, the reload value of Timer1 stored on register TMR1 is used to define the PWM1 frame rate and register PWM1DUTY is used to define the duty cycle of PWM1.

### 3.4.5 PS1CV (Prescaler1 Counter Value Register)

| Name | SFR Type | Addr. | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|---|---|
| PS1CV | S | 0x4 | PS1CV[7:0] | | | | | | | |
| R/W Property | | | R | | | | | | | |
| Initial Value | | | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

While reading PS1CV, it will get current value of Prescaler1 counter.

### 3.4.6 BZ1CR (Buzzer1 Control Register)

| Name | SFR Type | Addr. | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|---|---|
| BZ1CR | S | 0x5 | BZ1EN | - | - | - | BZ1FSEL[3:0] | | | |
| R/W Property | | | W | - | - | - | W | | | |
| Initial Value | | | 0 | X | X | X | 1 | 1 | 1 | 1 |

BZ1FSEL[3:0]:Frequency selection of BZ1 output.

| BZ1FSEL[3:0] | BZ1 Frequency Selection | |
|---|---|---|
| | Clock Source | Dividing Rate |
| 0000 | Prescaler1 output | 1:2 |
| 0001 | | 1:4 |
| 0010 | | 1:8 |
| 0011 | | 1:16 |
| 0100 | | 1:32 |
| 0101 | | 1:64 |
| 0110 | | 1:128 |
| 0111 | | 1:256 |
| 1000 | Timer1 output | Timer1 bit 0 |
| 1001 | | Timer1 bit 1 |
| 1010 | | Timer1 bit 2 |
| 1011 | | Timer1 bit 3 |
| 1100 | | Timer1 bit 4 |
| 1101 | | Timer1 bit 5 |
| 1110 | | Timer1 bit 6 |
| 1111 | | Timer1 bit 7 |

Table 7 Buzzer1 Output (PB2) Frequency Selection

**BZ1EN:** Enable/Disable BZ1 output.

BZ1EN=1, enable Buzzer1.

BZ1EN=0, disable Buzzer1.

### 3.4.7 IRCR (IR Control Register)

| Name | SFR Type | Addr. | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|---|---|
| IRCR | S | 0x6 | - | - | - | - | - | IRCSEL | IRF57K | IREN |
| R/W Property | | | - | - | - | - | - | W | W | W |
| Initial Value | | | X | X | X | X | X | 0 | 0 | 0 |

**IREN:** Enable/Disable IR carrier output.

IREN=1, enable IR carrier output.

IREN=0, disable IR carrier output.

**IRF57K:** Selection of IR carrier frequency.

IRF57K=1, IR carrier frequency is 57KHz.

IRF57K=0, IR carrier frequency is 38KHz.

**IRCSEL:** Polarity selection of IR carrier.

IRCSEL=0, IR carrier will be generated when I/O pin data is 1.

IRCSEL=1, IR carrier will be generated when I/O pin data is 0.

***Note:***

***1. Only high oscillation ($F_{HOSC}$) (See section 3.12) can be used as IR clock source.***

***2. Division ratio for different oscillation type.***

| OSC. Type | 57KHz | 38KHz | Conditions |
|---|---|---|---|
| High IRC | 64 | 96 | HIRC mode (the input to IR module is set to 4MHz no matter what system clock is) |

Table 8 Division ratio for different oscillation type

### 3.4.8 TBHP (Table Access High Byte Address Pointer Register)

| Name | SFR Type | Addr. | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|---|---|
| TBHP | S | 0x7 | - | - | - | - | - | TBHP2 | TBHP1 | TBHP0 |
| R/W Property | | | - | - | - | - | - | R/W | R/W | R/W |
| Initial Value | | | X | X | X | X | X | X | X | X |

When instruction CALLA, GOTOA or TABLEA is executed, the target address is constituted by TBHP[2:0] and ACC. ACC is the Low Byte of PC[9:0] and TBHP[1:0] is the high byte of PC[9:0]. TBHP[2] is general register for AT8A513H.

### 3.4.9 TBHD (Table Access High Byte Data Register)

| Name | SFR Type | Addr. | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|---|---|
| TBHD | S | 0x8 | - | - | TBHD5 | TBHD4 | TBHD3 | TBHD2 | TBHD1 | TBHD0 |
| R/W Property | | | - | - | R | R | R | R | R | R |
| Initial Value | | | X | X | X | X | X | X | X | X |

When instruction TABLEA is executed, high byte of content of addressed ROM is loaded into TBHD[5:0] register. The Low Byte of content of addressed ROM is loaded to ACC.

### 3.4.10 OSCCR (Oscillation Control Register)

| Name | SFR Type | Addr. | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|---|---|
| OSCCR | S | 0xF | - | CMPOEN | - | - | OPMD[1:0] | | STPHOSC | SELHOSC |
| R/W Property | | | - | R/W | - | - | R/W | | R/W | R/W |
| Initial Value | | | X | 0 | X | X | 00 | | 0 | 1 |

**SELHOSC:** Selection of system oscillation ($F_{OSC}$).

SELHOSC=1, $F_{OSC}$ is high-frequency oscillation ($F_{HOSC}$).

SELHOSC=0, $F_{OSC}$ is low-frequency oscillation ($F_{LOSC}$).

**STPHOSC:** Disable/enable high-frequency oscillation ($F_{HOSC}$).

STPHOSC=1, $F_{HOSC}$ will stop oscillation and be disabled.

STPHOSC=0, $F_{HOSC}$ keep oscillation.

**CMPOEN:** Comparator output enable / disable

CMPOEN=1, Comparator output enable

CMPOEN=0, Comparator output disable

**OPMD[1:0]:** Selection of operating mode.

| OPMD[1:0] | Operating Mode |
|---|---|
| 00 | Normal mode |
| 01 | Halt mode |
| 10 | Standby mode |
| 11 | reserved |

Table 9 Selection of Operating Mode by OPMD[1:0]

***Note: STPHOSC cannot be changed with SELHOSC or OPMD at the same time. STPHOSC cannot be changed with OPMD at the same time during SELHOSC=1.***

## 3.5 I/O Port

AT8A513H provide 6 I/O pins which are PB[5:0]. User can read/write these I/O pins through register PORTB. Each I/O pin has a corresponding register bit to define it is input pin or output pin. Register IOSTB[5:0] define the input/output direction of PB[5:0].

When an I/O pin is configured as input pin, it may have Pull-High resistor or Pull-Low resistor which is enabled or disabled through registers. Register BPHCON[5:0] are used to enable or disable Pull-High resistor of PB[5:0]. Register BPLCON[7:4] are used to enable or disable Pull-Low resistor of PB[3:0].

When an I/O pin is configured as output pin, there is a corresponding and individual register to select as Open-Drain output pin. Register BODCON[5:0] determine PB[5:0] is Open-Drain or not.

The summary of Pad I/O feature is listed in the table below.

| Feature | | PB[3:0] | PB[5:4] |
|---|---|---|---|
| Input | Pull-High Resistor | V | V |
| | Pull-Low Resistor | V | X |
| Output | Open-Drain | V | V |

Table 10 Summary of Pad I/O Feature

The level change on each I/O pin of PB may generate interrupt request. Register BWUCON[5:0] will select which I/O pin of PB may generate this interrupt. As long as any pin of PB is selected by corresponding bit of BWUCON, the register bit PBIF (INTF[1]) will set to 1 if there is a level change occurred on any selected pin. An interrupt request will occur and interrupt service routine will be executed if register bit PBIE (INTE[1]) and GIE (PCON1[7]) are both set to 1.

There is one external interrupt provided by AT8A513H. When register bit EIS (PCON[6]) is set to 1, PB0 is used as input pin for external interrupt.

***Note: When PB0 is both set as level change operation and external interrupt, the external interrupt will have higher priority, and the PB0 level change operation will be disabled. But PB5~PB1 level change function are not affected.***

AT8A513H can provide IR carrier generation. IR carrier generation is enabled by register bit IREN (IRCR[0]) and carrier will be present on a PB1 pin.

PB3 can be used as external reset input determined by a configuration word. When an active-low signal is applied to PB3, it will cause AT8A513H to enter reset process.

When AT8A513H is in Normal mode or Standby mode, instruction clock is observable on PB4 if a configuration word is enabled.

Moreover, PB2 can be timer 0 external clock source EX_CKI if T0MD T0CS=1 and LCK_TM0=0. PB2 can be timer 1 external clock source if T1CS=1.

Moreover, PB2 can be PWM output if T1CR1[7] PWM1OEN=1 and corresponding PB.2 configuration word. PB2 can also be Buzzer output if BZ1CR[7] BZ1EN=1 and corresponding PB.2 configuration word.

### 3.5.1 Block Diagram of IO Pins

IO_SEL: Set pad attribute as input or output.
WRITE_EN: Write data to pad.
READ_EN: RRead pad.
OD_EN: Enable open-Drain.
PULLUP_ENB: Enable Pull-High.
PULLDOWN_EN: Enable Pull-Low.
RD_TYPE: select Read pin or read latch.
EIS: External interrupt function enable.
INTEDGE: External interrupt edge select.
EX_INT: External interrupt signal.
WUB: Port B wake-up enable.
SET_PBIF: Port B wake-up flag.

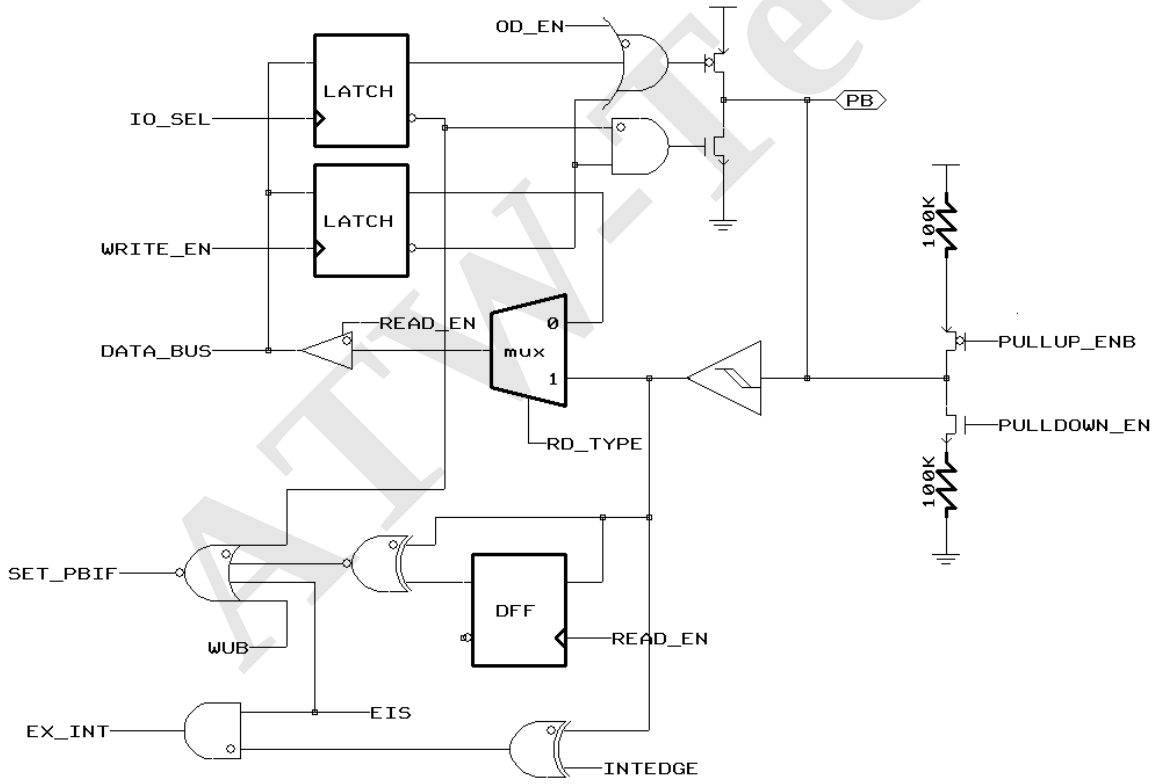



Figure 5 Block Diagram of PB0

| | |
|---|---|
| IO_SEL: | Set pad attribute as input or output. |
| WRITE_EN: | Write data to pad. |
| READ_EN: | Read pad. |
| OD_EN: | Enable open-Drain. |
| PULLUP_ENB: | Enable Pull-High. |
| PULLDOWN_EN: | Enable Pull-Low. |
| RD_TYPE: | Select read pin or read latch. |
| IREN: | IR function enable. |
| IRDT: | IR data. |
| WUB: | Port B wake-up enable. |
| SET_PBIF: | Port B wake-up flag. |

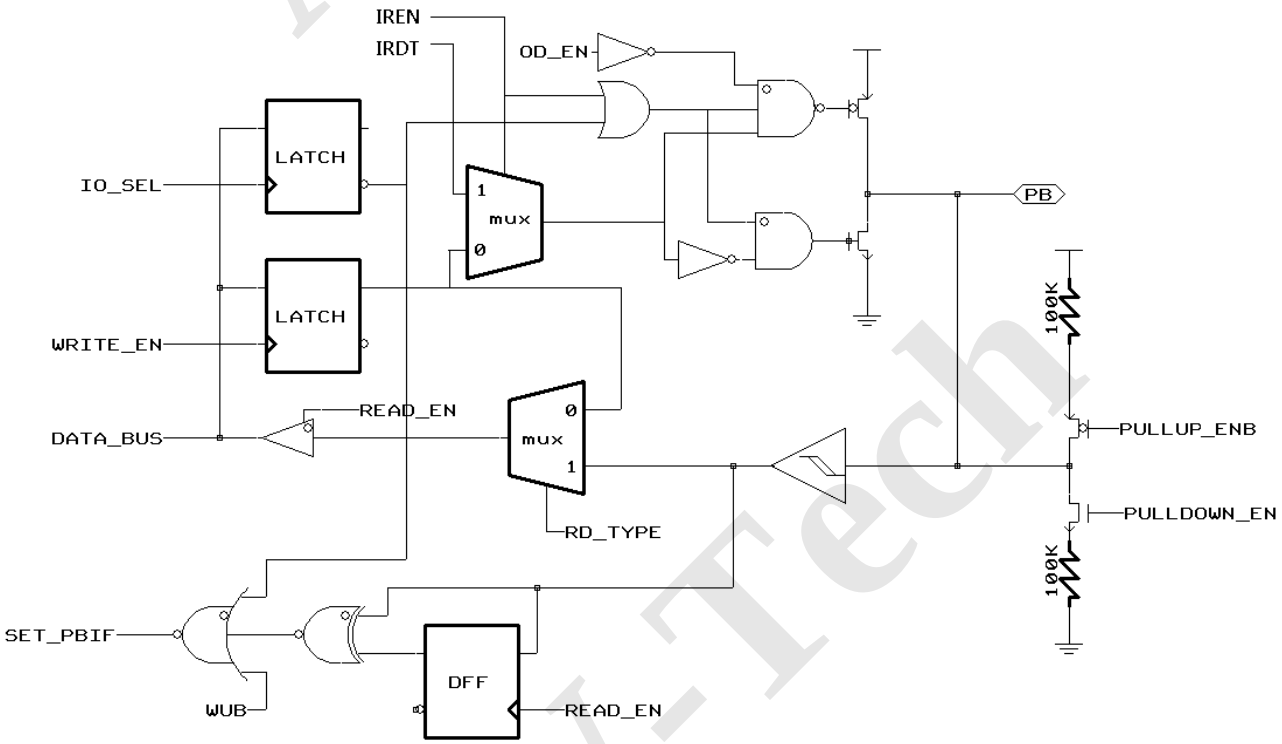



Figure 6 Block Diagram of PB1

| | |
|---|---|
| IO_SEL: | Set pad attribute as input or output. |
| WRITE_EN: | Write data to pad. |
| READ_EN: | Read pad. |
| OD_EN: | Enable open-Drain. |
| PULLUP_ENB: | Enable Pull-High. |
| PULLDOWN_EN: | Enable Pull-Low. |
| RD_TYPE: | Select read pin or read latch. |
| PBEN: | PWM/BUZZER enable. |
| PBDT: | PWM/BUZZER data. |
| WUB: | Port B wake-up enable. |
| SET_PBIF: | Port B wake-up flag. |
| EX CKI: | External clock input. |

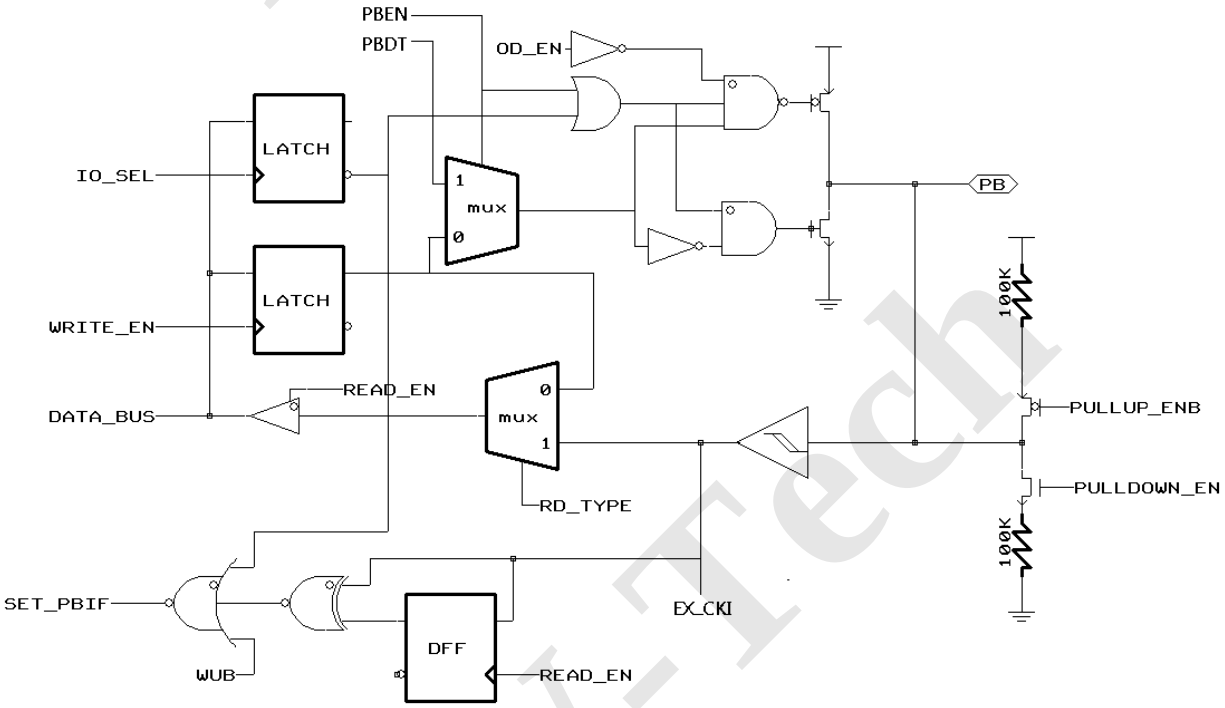



Figure 7 Block Diagram of PB2

| | |
|---|---|
| IO_SEL: | Set pad attribute as input or output. |
| WRITE_EN: | Write data to pad. |
| READ_EN: | Read pad. |
| OD_EN: | Enable open-Drain. |
| PULLUP_ENB: | Enable Pull-High. |
| PULLDOWN_EN: | Enable Pull-Low. |
| RD_TYPE: | Select read pin or read latch. |
| PBEN: | PWM/BUZZER enable. |
| PBDT: | PWM/BUZZER data. |
| WUB: | Port B wake-up enable. |
| SET_PBIF: | Port B wake-up flag. |

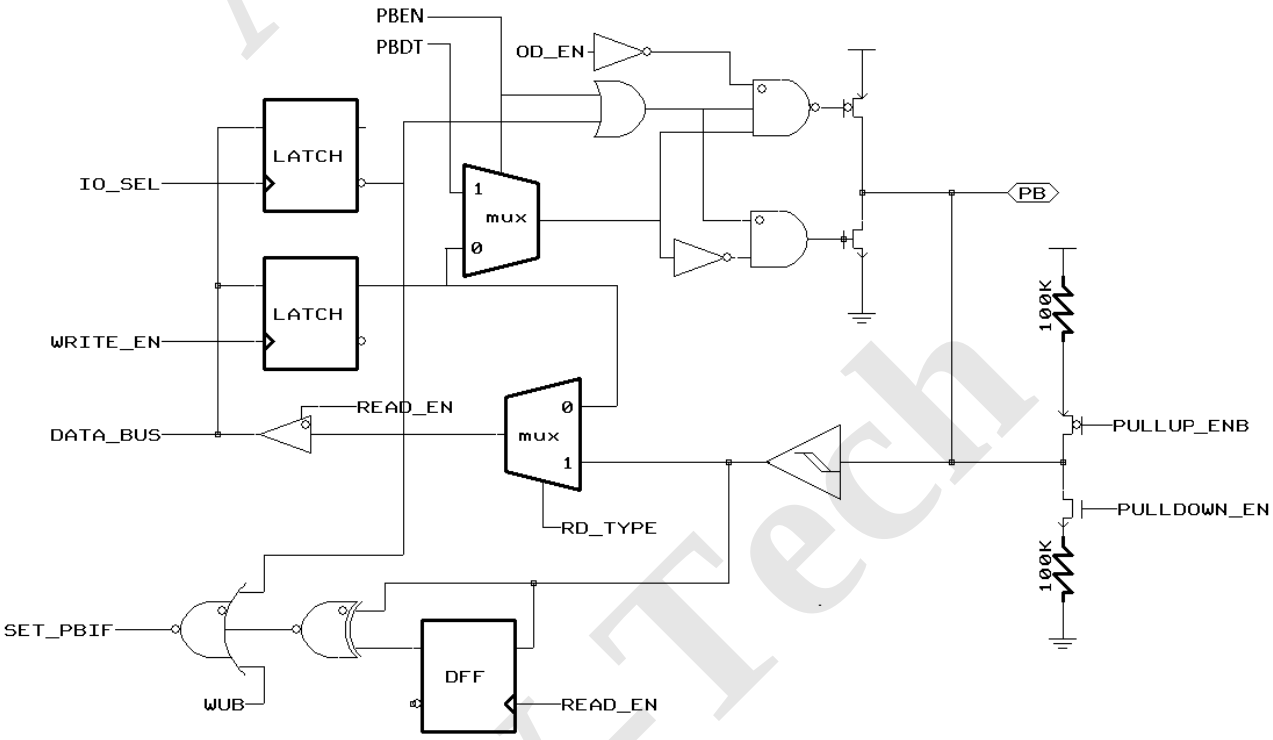



Figure 8 Block Diagram of PB3

***Note: While PB3 is set as input, the input voltage should not larger than VDD.***

| | |
|---|---|
| IO_SEL: | Set pad attribute as input or output. |
| WRITE_EN: | Write data to pad. |
| READ_EN: | Read pad. |
| OD_EN: | Enable open-Drain. |
| PULLUP_ENB: | Enable Pull-High. |
| RD_TYPE: | Select read pin or read latch. |
| WUB: | Port B wake-up enable. |
| SET_PBIF: | Port B wake-up flag. |

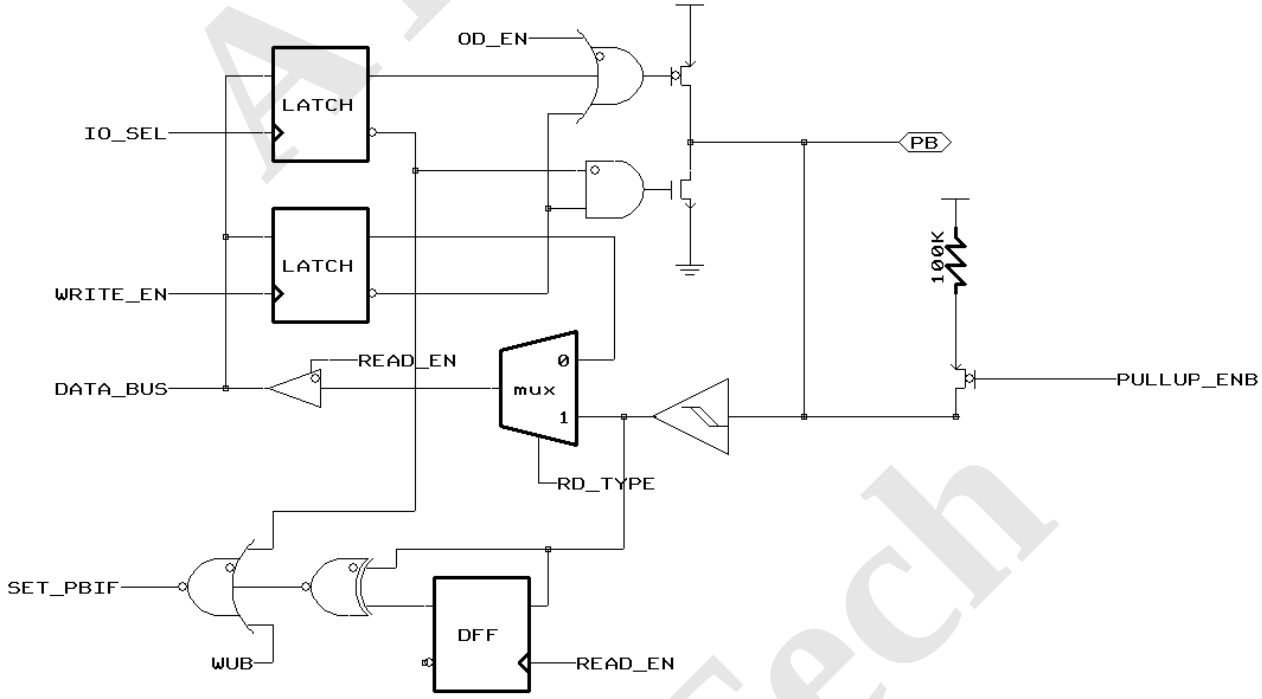



Figure 9 Block Diagram of PB4/PB5

### 3.6 Timer0

Timer0 is an 8-bit up-count timer and its operation is enabled by register bit T0EN (PCON1[0]). Writing to Timer0 will set its initial value. Reading from Timer0 will show its current count value.

The clock source to Timer0 can be from instruction clock, external pin EX_CKI or low speed clock Low Oscillator Frequency according to register bit T0CS and LCK_TM0 (T0MD[5] and T0MD[7]). When T0CS is 0, instruction clock is selected as Timer0 clock source. When T0CS is 1 and LCK_TM0 is 0, EX_CKI is selected as Timer0 clock source. When T0CS is 1 and LCK_TM0 is 1 (and Timer0 source must set to 1), Low Oscillator Frequency (I_LRC) output is selected. Summarized table is shown below. (Also check Figure. 10)

| Timer0 clock source | T0CS | LCKTM0 | Timer0 source |
|---|---|---|---|
| Instruction clock | 0 | X | X |
| EX_CKI | 1 | 0 | X |
| | | X | 0 |
| I_LRC | 1 | 1 | 1 |

Table 11 Summary of Timer0 clock source control

Moreover the active edge of EX_CKI or Low Oscillator Frequency to increase Timer0 can be selected by register bit T0CE (T0MD[4]). When T0CE is 1, high-to-low transition on EX_CKI or Low Oscillator Frequency will increase Timer0. When T0CE is 0, low-to-high transition on EX_CKI or Low Oscillator Frequency will increase Timer0.

Before Timer0 clock source is supplied to Timer0, it can be divided by Prescaler0 if register bit PS0WDT (T0MD[3]) is clear to 0. When writing 0 to PS0WDT by instruction, Prescaler0 is assigned to Timer0 and Prescaler0 will be clear after this instruction is executed. The dividing rate of Prescaler0 is determined by register bits PS0SEL[2:0] which is from 1:2 to 1:256.

Before entering Timer0, the Timer0 clock source synchronize with instruction clock in default. If EX_CKI or Low Oscillator Frequency is used as Timer0 clock source, care must be taken that their frequency can not exceed instruction clock frequency, or missing count may happen. When Low Oscillator Frequency is both used as Timer0 clock source and instruction clock, AT8A513H must assign prescaler0 to Timer0 and the prescaler0 dividing ratio must be no less than 4. There's a configuration word (EX_CKI to Inst. Clock) which can relieve this constraint. When this option is set to Async, the Timer0 clock source does not synchronize with the instruction clock, so the input frequency of EX_CKI can be higher than the instruction clock. The maximum frequency the EX_CKI can input is depended on process variation.

When Timer0 is overflow, the register bit T0IF (INTF[0]) will be set to 1 to indicate Timer0 overflow event is occurred. If register bit T0IE (INTE[0]) and GIE are both set to 1, interrupt request will occur and interrupt service routine will be executed. T0IF will not be clear until firmware writes 0 to T0IF.

The block diagram of Timer0 and WDT is shown in the figure below.

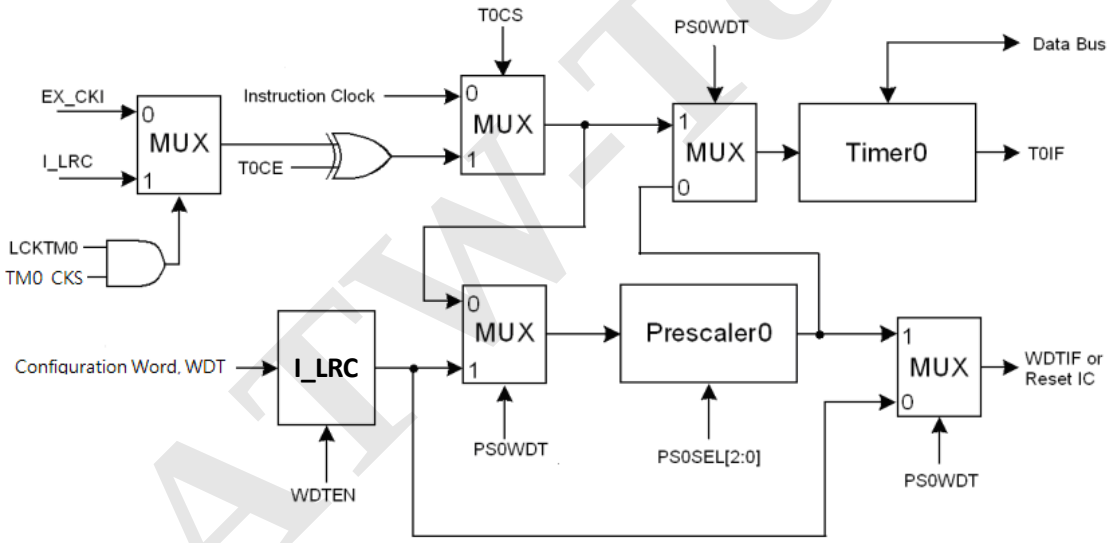



Figure 10 Block Diagram of Timer0 and WDT

## 3.7 Timer1/PWM1/Buzzer1

Timer1 is an 8-bit down-count timer with Prescaler1 whose dividing rate is programmable. The output of Timer1 can be used to generate PWM1 output and Buzzer1 output. A write to the Timer1 will both write to a timer1 reload register (T1rld) and timer1 counter. A read to the timer1 will show the content of the Timer1 current count value.

The block diagram of Timer1 is shown in the figure below.

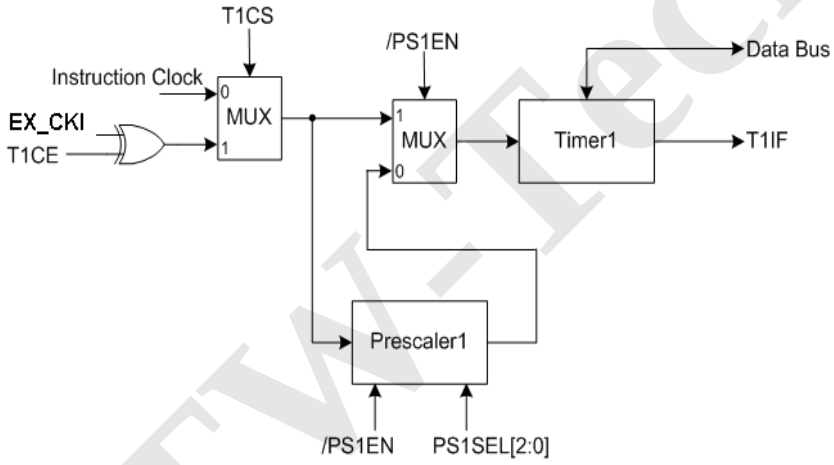



Figure 11 Block Diagram of Timer1

The operation of Timer1 can be enabled or disabled by register bit T1EN (T1CR1[0]). After Timer1 is enabled, its clock source can be instruction clock or pin EX_CKI which is determined by register bit T1CS (T1CR2[5]). When T1CS is 1, EX_CKI is selected as clock source. When T1CS is 0, instruction clock is selected as clock source.

When EX_CKI is selected, the active edge to decrease Timer1 is determined by register bit T1CE (T1CR2[4]). When T1CE is 1, high-to-low transition on EX_CKI will decrease Timer1. When T1CE is 0, low-to-high transition on EX_CKI will decrease Timer1. The selected clock source can be divided further by Prescaler1 before it is applied to Timer1. Prescaler1 is enabled by writing 0 to register bit /PS1EN (T1CR2[3]) and the dividing rate is from 1:2 to 1:256 determined by register bits PS1SEL[2:0] (T1CR2[2:0]). Current value of Prescaler1 can be obtained by reading register PS1CV.

Timer1 provide two kinds of operating mode: one is One-Shot mode and the other is Non-Stop mode. When register bit T1OS (T1CR1[2]) is 1, One-Shot mode is selected. Timer1 will count down once from initial value stored on register TMR1 to 0x00, i.e. underflow is occurred. When register bit T1OS (T1CR1[2]) is 0, Non-Stop mode is selected. When underflow is occurred, there are two selections to start next down-count which is determined by register bit T1RL (T1CR1[1]). When T1RL is 1, the initial value stored on register TMR1 will be restored and start next down-count from this initial value. When T1RL is 0, Timer1 will start next down-count from 0xFF.

When Timer1 is underflow, the register bit T1IF (INTF[3]) will be set to 1 to indicate Timer1 underflow event is occurred. If register bit T1IE (INTE[3]) and GIE are both set to 1, interrupt request will occur and interrupt service routine will be executed. T1IF will not be clear until firmware writes 0 to T1IF.

The timing chart of Timer1 is shown in the following figure.

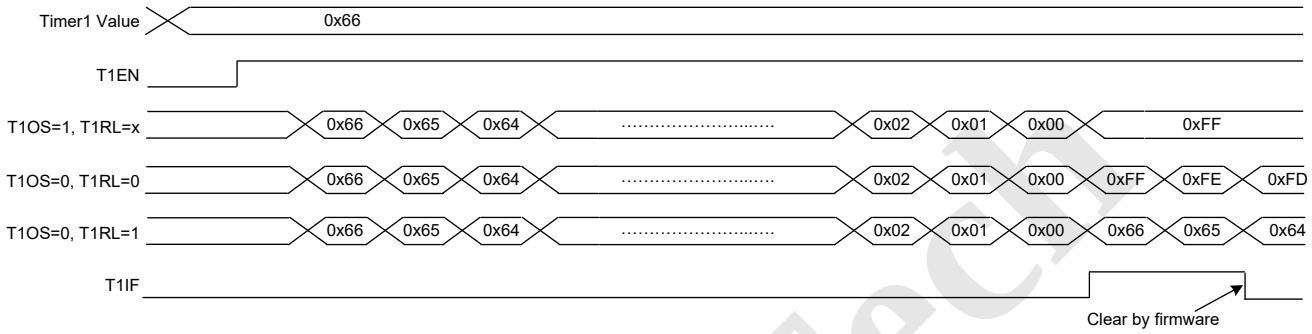



Figure 12 Timer1 Timing Chart

The PWM1 output can be available on I/O pin PB2 when register bit PWM1OEN (T1CR1[7]) is set to 1 and configuration word PB.2 is PWM. When PWM1OEN=1, PB2 will become output pin automatically. The active state of PWM1 output is determined by register bit PWM1OAL (T1CR1[6]). When PWM1OAL is 1, PWM1 output is active low. When PWM1OAL is 0, PWM1 output is active high. Moreover, the duty cycle and frame rate of PWM1 are both programmable. The duty cycle is determined by register PWM1DUTY. When PWM1DUTY is 0, PWM1 output will be never active. When PWM1DUTY is 0xFF, PWM1 output will be active for 255 Timer1 input clocks. The frame rate is determined by TMR1 initial value. Therefore, PWM1DUTY value must be less than or equal to TMR1. The block diagram of PWM1 is illustrated in the following figure.

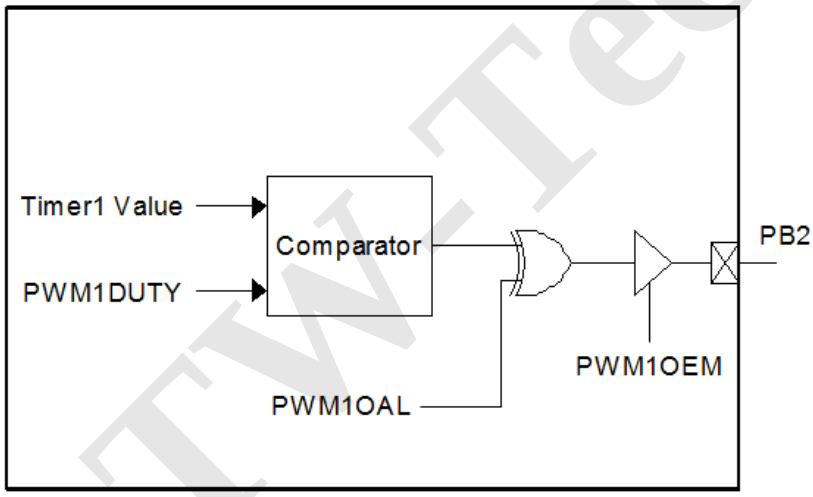



Figure 13 PWM1 Block Diagram

The Buzzer1 output (BZ1) can be available on I/O pin PB2, when register bit BZ1EN (BZ1CR1[7]) is set to 1 and corresponding configuration word PB.2 is BUZZER. When BZ1EN is set to 1, PB2 will become output pin automatically. The frequency of BZ1 can be derived from Timer1 output or Prescaler1 output and dividing rate is determined by register bits BZ1FSEL[3:0] (BZ1CR[3:0]). When BZ1FSEL[3] is 0, Prescaler1 output is selected to generate BZ1 output. When BZ1FSEL[3] is 1, Timer1 output is selected to generate BZ1 output. The dividing rate can be from 1:2 to 1:256 in order to generate all kinds of frequency. The block diagram of Buzzer1 is illustrated in the following figure.

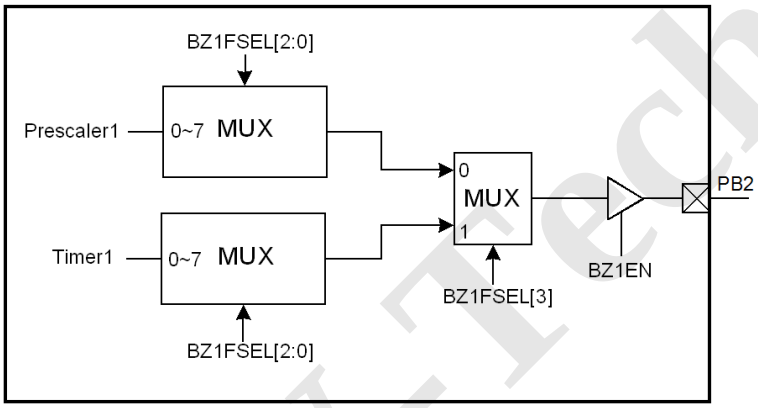



Figure 14 Buzzer1 Block Diagram

### 3.8 IR Carrier

The IR carrier will be generated after register bit IREN (IRCR[0]) is set to 1. Moreover, PB1 will become output pin automatically. When IREN is clear to 0, PB1 will become general I/O pin as it was configured.

The IR carrier frequency is selectable by register bit IRF57K (IRCR[1]). When IRF57K is 1, IR carrier frequency is 57KHz. When IRF57K is 0, IR carrier frequency is 38KHz.

The active state (polarity) of IR carrier is selectable according to PB1 output data. When register bit IRCSEL (IRCR[2]) is 1, IR carrier will be present on pin PB1 when its output data is 0. When register bit IRCSEL (IRCR[2]) is 0, IR carrier will be present on pin PB1 when its output data is 1. The polarity of IR carrier is shown in the following figure.

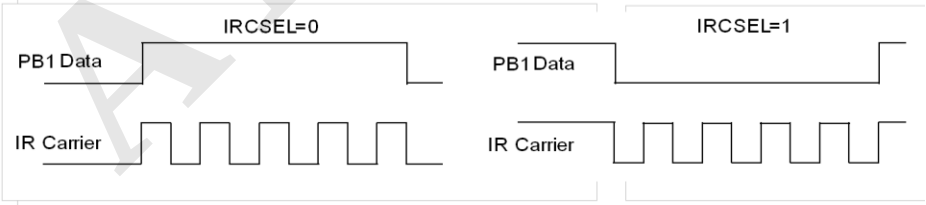



Figure 15 Polarity of IR Carrier vs. Output Data

### 3.9 Comparator-Related Functions

#### 3.9.1 Function Overview

- Voltage Comparator;
- Low Voltage Detection (LVD);
- Built-in resistor voltage divider module to monitor power supply voltage VDD;
- LVD interrupt

#### 3.9.2 Voltage Comparator

AT8A513H provides voltage comparator and internal reference voltage with various analog comparing mode. The comparator non-inverting and inverting input can share with GPIO.

CMPEN (register PCON[2]) is used to enable and disable comparator. When CMPEN=0(default), comparator is disabled. When CMPEN=1, the comparator is enabled. In halt mode the comparator is disabled automatically.

The structure of comparator is shown in the following figure:

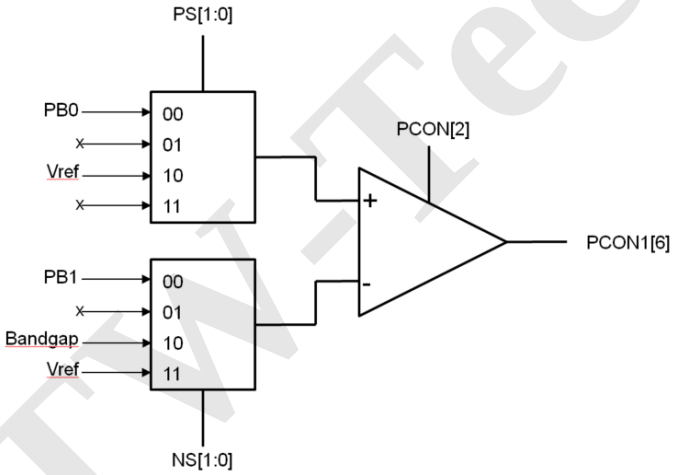



Figure 16 Comparator block diagram

The non-inverting input of the comparator is determined by PS[1:0] (register CMPCR[3:2]).

The table is shown below

| PS[1:0] | Non-inverting input |
|---|---|
| 00 | PB0 |
| 01 | -- |
| 10 | Vref |
| 11 | --- |

Table 12 Non-inverting input select

The inverting input of the comparator is determined by NS[1:0] (register CMPCR[1:0]).

The table is shown below

| NS[1:0] | Inverting input |
|---|---|
| 00 | PB1 |
| 01 | -- |
| 10 | Bandgap (0.65V) |
| 11 | Vref |

Table 13 Inverting input select

There are two ways to get the comparator output result: one is through register polling, the other is through probing output pad.

Comparator output can be polled by LVDOUT (register PCON1[6] ).

To probe comparator output at output pad, set CMPOE (register OSCCR[6]) to 1, then PB2 will be the real-time state of the comparator output.

### 3.9.3 Comparator Reference Voltage (Vref)

The internal reference voltage Vref is built by series resistance to provide different level of reference voltage. RBIAS_H and RBIAS_L are used to select the maximum and minimum values of Vref, and LVDS[3:0] are used to select one of 64 voltage levels. When VDD level falls below Vref voltage, LVD flag will go from high to low, and set the register bit LVDIF=1. And when the comparator result differs from the previous value, a comparator interrupt is generated. This interrupt request will be serviced if LVDIE and GIE are set to 1.

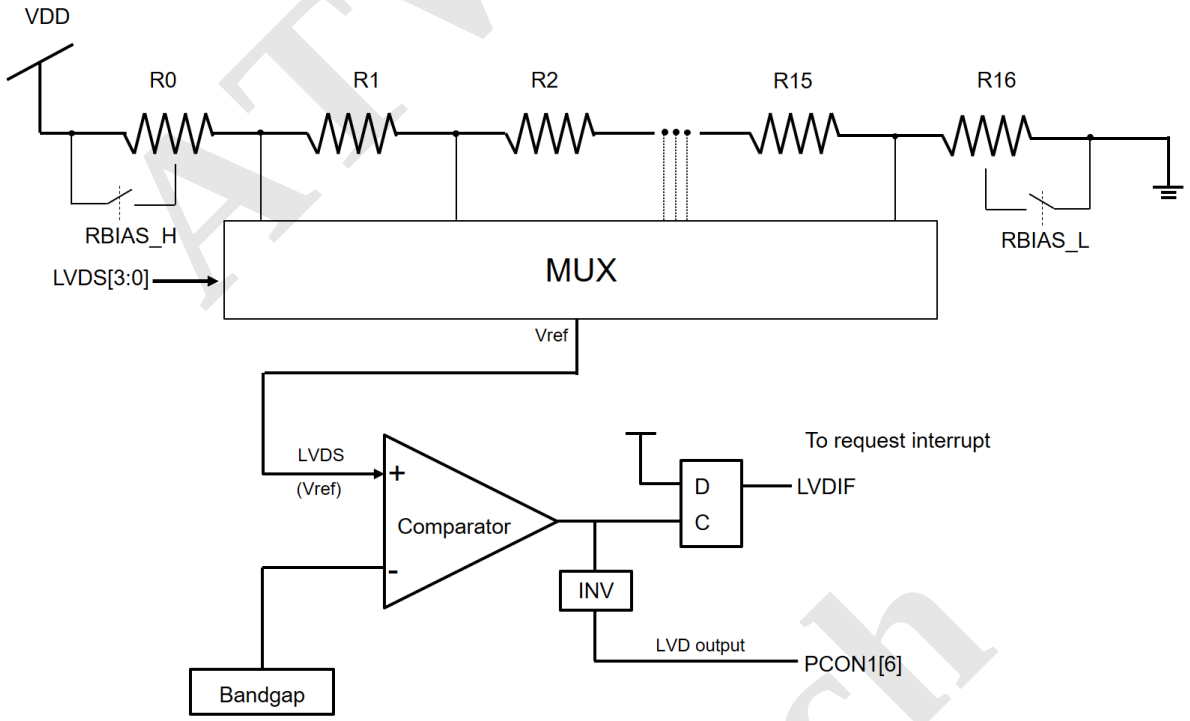



Figure 17 Vref hardware connection & LVD block diagram

The **Vref** is determined by RBIAS_H, RBIAS_L and LVDS[3:0]. The LVDS[3:0] is used to select one out of 64 reference voltages, the table shown below.

| **Condition** | | | | |
|---|---|---|---|---|
| | $V_{IN-}$ = Vref(V) | | | |
| LVRS[3:0] | RBIAS[H:L]=[0:0] | RBIAS[H:L]=[1:0] | RBIAS[H:L]=[0:1] | RBIAS[H:L]=[1:1] |
| 1100 | 0.161*VDD | 0.257*VDD | 0.075*VDD | 0.127*VDD |
| 1010 | 0.164*VDD | 0.262*VDD | 0.078*VDD | 0.133*VDD |
| 1000 | 0.170*VDD | 0.271*VDD | 0.084*VDD | 0.143*VDD |
| 1111 | 0.177*VDD | 0.283*VDD | 0.093*VDD | 0.157*VDD |
| 0111 | 0.183*VDD | 0.292*VDD | 0.099*VDD | 0.168*VDD |
| 1110 | 0.190*VDD | 0.304*VDD | 0.107*VDD | 0.182*VDD |

| Condition | | | | |
|---|---|---|---|---|
| | $V_{IN-}$ = Vref(V) | | | |
| LVRS[3:0] | RBIAS[H:L]=[0:0] | RBIAS[H:L]=[1:0] | RBIAS[H:L]=[0:1] | RBIAS[H:L]=[1:1] |
| 0110 | 0.199*VDD | 0.318*VDD | 0.117*VDD | 0.199*VDD |
| 1101 | 0.207*VDD | 0.330*VDD | 0.125*VDD | 0.213*VDD |
| 0101 | 0.216*VDD | 0.345*VDD | 0.135*VDD | 0.230*VDD |
| 0100 | 0.224*VDD | 0.356*VDD | 0.144*VDD | 0.244*VDD |
| 0011 | 0.231*VDD | 0.369*VDD | 0.152*VDD | 0.258*VDD |
| 1011 | 0.248*VDD | 0.395*VDD | 0.170*VDD | 0.289*VDD |
| 0010 | 0.268*VDD | 0.427*VDD | 0.192*VDD | 0.327*VDD |
| 0001 | 0.291*VDD | 0.463*VDD | 0.218*VDD | 0.370*VDD |
| 0000 | 0.317*VDD | 0.505*VDD | 0.246*VDD | 0.419*VDD |
| 1001 | 0.327*VDD | 0.521*VDD | 0.258*VDD | 0.438*VDD |

Table 14 The reference voltage Vref selection table

***Note: The deviation of Vref is ±0.1V.***

Based on an internal reference voltage (Vref) generated by a series resistor divider, the comparator can be used to monitor the supply voltage. When the inverting input is set to the internal bandgap reference (0.65V) and the non-inverting input is connected to the internal resistor divider output (Vref), the comparator compares the supply voltage against the configured threshold. The figure 17 is Vref hardware connection & LVD block diagram.

- The comparator output is 0 when the supply voltage is below the threshold.
- The comparator output is 1 when the supply voltage is above the threshold.

### 3.9.4 Low Voltage Detector (LVD)

AT8A513H low voltage detector (LVD) built-in precise band-gap reference for accurately detecting VDD level. Built-in 14-stage precise low-voltage detection circuit, allowing users to directly select the desired detection voltage point. If LVDEN(register PCON[5])=1 and VDD voltage value falls below LVD voltage which is selected by LVDS[3:0] as table shown below, the LVD output will become low. If the LVD interrupt is enabled, the LVD interrupt flag will be high and if GIE=1 it will force the program to execute interrupt service routine. Moreover, LVD real-state output can be polled by register PCON1[6].

The following table is LVD voltage select table.

| RBIAS[H:L] | LVDS[3:0] | LVD Voltage (V) |
|---|---|---|
| 00 | 0001 | 2.20 |
| 00 | 0010 | 2.40 |
| 00 | 1011 | 2.60 |
| 00 | 0011 | 2.80 |
| 00 | 0100 | 2.90 |
| 00 | 0101 | 3.00 |
| 00 | 1101 | 3.15 |
| 00 | 0110 | 3.30 |
| 00 | 1110 | 3.45 |
| 00 | 0111 | 3.60 |
| 00 | 1111 | 3.75 |
| 00 | 1000 | 3.90 |
| 00 | 1010 | 4.05 |
| 00 | 1100 | 4.15 |

Table 16 LVD voltage select

The LVD control flow is as the following:

*Step1: Select LVD voltage by LVDS[3:0]*

*Step2: Set CMPCR = 0x0A*

*Step3: Set PCON[5]=1 (enable LVD)*

*Step4: Check LVD status by PCON1[6]*

***Note: If LVD voltage LVDS[3:0] is changed, user must wait at least 50us(@$F_{HOSC}$=1MHz) to get correct LVD status by PCON1[6]***

### 3.9.5 LVD Interrupt

When VDD level falls below LVD voltage, LVD flag will go from high to low, and set the register bit LVDIF=1. And when the comparator result differs from the previous value, a comparator interrupt is generated. This interrupt request will be serviced if LVDIE and GIE are set to 1.

### 3.10 Watch-Dog Timer (WDT)

There is an on-chip free-running oscillator in AT8A513H which is used by WDT. As this oscillator is independent of other oscillation circuits, WDT may still keep working during Standby mode and Halt mode.

WDT can be enabled or disabled by a configuration word. When WDT is enabled by configuration word, its operation still can be controlled by register bit WDTEN (PCON[7]) during program execution. Moreover, the mechanism after WDT time-out can reset AT8A513H or issue an interrupt request which is determined by another configuration word. At the same time, register bit /TO (STATUS[4]) will be clear to 0 after WDT time-out.

The baseline of WDT time-out period can be 3.5 ms, 15 ms, 60 ms or 250 ms which is determined by two configuration words. The time-out period can be lengthened if Prescaler0 is assigned to WDT. Prescaler0 will be assigned to WDT by writing 1 to register bit PS0WDT. The dividing rate of Prescaler0 for WDT is determined by register bits PS0SEL[2:0] and depends on WDT time-out mechanism. The dividing rate is from 1:1 to 1:128 if WDT time-out will reset AT8A513H and dividing rate is from 1:2 to 1:256 if WDT time-out will interrupt AT8A513H.

When Prescaler0 is assigned to WDT, the execution of instruction CLRWDT will clear WDT, Prescaler0 and set /TO flag to 1.

If user selects interrupt for WDT time-out mechanism, register bit WDTIF (INTF[6]) will set to 1 after WDT is expired. It may generate an interrupt request if register bit WDTIE (INTE[6]) and GIE both set to 1. WDTIF will not be clear until firmware writes 0 to WDTIF.

### 3.11 Interrupt

AT8A513H provide two kinds of interrupt: one is software interrupt and the other is hardware interrupt. Software interrupt is caused by execution of instruction INT. There are 6 hardware interrupts:

- Timer0 overflow interrupt.
- Timer1 underflow interrupt.
- WDT timeout interrupt.
- PB input change interrupt.
- External interrupt..
- LVD interrupt

GIE is global interrupt enable flag. It has to be 1 to enable hardware interrupt functions. GIE can be set by ENI instruction and clear to 0 by DISI instruction.

After instruction INT is executed, no matter GIE is set or clear, the next instruction will be fetched from address 0x001. At the same time, GIE will be clear to 0 by AT8A513H automatically. This will prevent nested interrupt from happening. The last instruction of interrupt service routine of software interrupt has to be RETIE. Execution of this instruction will set GIE to 1 and return to original execution sequence.

While any of hardware interrupts is occurred, the corresponding bit of Interrupt Flag Register INTF will be set to 1. This bit will not be clear until firmware writes 0 to this bit. Therefore user can obtain information of which event

causes hardware interrupt by polling register INTF. Note that only when the corresponding bit of Interrupt Enable register INTE is set to 1, will the corresponding interrupt flag be read. And if the corresponding bit of Interrupt Enable Register INTE is set to 1 and GIE is also 1, hardware interrupt will occur and next instruction will be fetched from 0x008. At the same time, the register bit GIE will be clear by AT8A513H automatically. If user wants to implement nested interrupt, instruction ENI can be used as the first instruction of interrupt service routine which will set GIE to 1 again and allow other interrupt events to interrupt AT8A513H again. Instruction RETIE has to be the last instruction of interrupt service routine which will set GIE to 1 and return to original execution sequence.

It should be noted that ENI instruction cannot be placed right before RETIE instruction because ENI instruction in interrupt service routine will trigger nested interrupt, but RETIE will clear internal interrupt processing after jump out of ISR, so it is possible for interrupt flag to be falsely cleared.

### 3.11.1 Timer0 Overflow Interrupt

Timer0 overflow (from 0x00 to 0xFF) will set register bit T0IF. This interrupt request will be serviced if T0IE and GIE are set to 1.

### 3.11.2 Timer1 Underflow Interrupt

Timer1 underflow (from 0xFF to 0x00) will set register bit T1IF. This interrupt request will be serviced if T1IE and GIE are set to 1.

### 3.11.3 WDT Timeout Interrupt

When WDT is timeout and the configuration word selects WDT timeout will generate interrupt request, it will set register bit WDTIF. This interrupt request will be serviced if WDTIE and GIE are set to 1.

### 3.11.4 PB Input Change Interrupt

When PBx, $0 \le x \le 5$, is configured as input pin and corresponding register bit WUPBx is set to 1, a level change on these selected I/O pin(s) will set register bit PBIF. This interrupt request will be serviced if PBIE and GIE are set to 1. Note when PB0 is both set as level change interrupt and external interrupt, the external interrupt flag EIS will disable PB0 level change operation.

### 3.11.5 External Interrupt

According to the configuration of EIS=1 and INTEDG, the selected active edge on I/O pin PB0 will set register bit INTIF and this interrupt request will be served if INTIE and GIE are set to 1.

### 3.11.6 LVD Interrupt

When VDD level falls below LVD voltage, LVD flag will go from high to low, and set the register bit LVDIF=1. This interrupt request will be serviced if LVDIE and GIE are set to 1.

## 3.12 Oscillation Configuration

Because AT8A513H is a dual-clock IC, there are high oscillation (FHOSC) and low oscillation (FLOSC) which can be selected as system oscillation (FOSC). The oscillators which could be used as FHOSC are internal high RC oscillator (I_HRC). The oscillators which could be used as FLOSC are internal low RC oscillator (I_LRC).

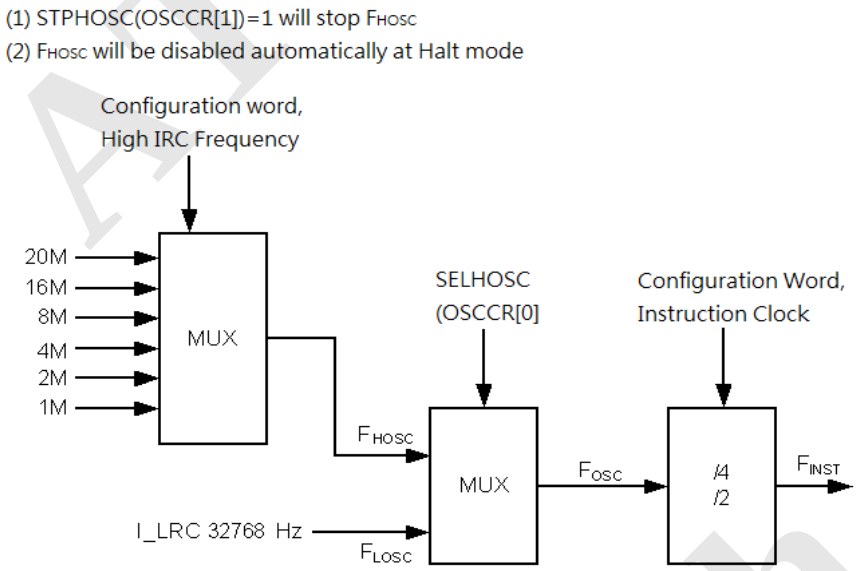



Figure 18 Oscillation Configuration of AT8A513H

I_HRC output frequency is determined by three configuration words and it can be 1M, 2M, 4M, 8M, 16M or 20MHz.

When I_LRC is selected, its frequency is centered on 32768Hz.

Either $F_{HOSC}$ or $F_{LOSC}$ can be selected as system oscillation $F_{OSC}$ according to the value of register bit SELHOSC (OSCCR[0]). When SELHOSC is 1, $F_{HOSC}$ is selected as $F_{OSC}$. When SELHOSC is 0, $F_{LOSC}$ is selected as $F_{OSC}$. Once $F_{OSC}$ is determined, the instruction clock $F_{INST}$ can be $F_{OSC}/2$ or $F_{OSC}/4$ according to value of a configuration word.

## 3.13 Operating Mode

AT8A513H provides four kinds of operating mode to tailor all kinds of application and save power consumptions. These operating modes are Normal mode, Slow mode, Standby mode and Halt mode. Normal mode is designated for high-speed operating mode. Slow mode is designated for low-speed mode in order to save power consumption. At Standby mode, AT8A513H will stop almost all operations except Timer0/Timer1/WDT in

order to wake-up periodically. At Halt mode, AT8A513H will sleep until external event or WDT trigger IC to wake-up.

The block diagram of four operating modes is described in the following figure.

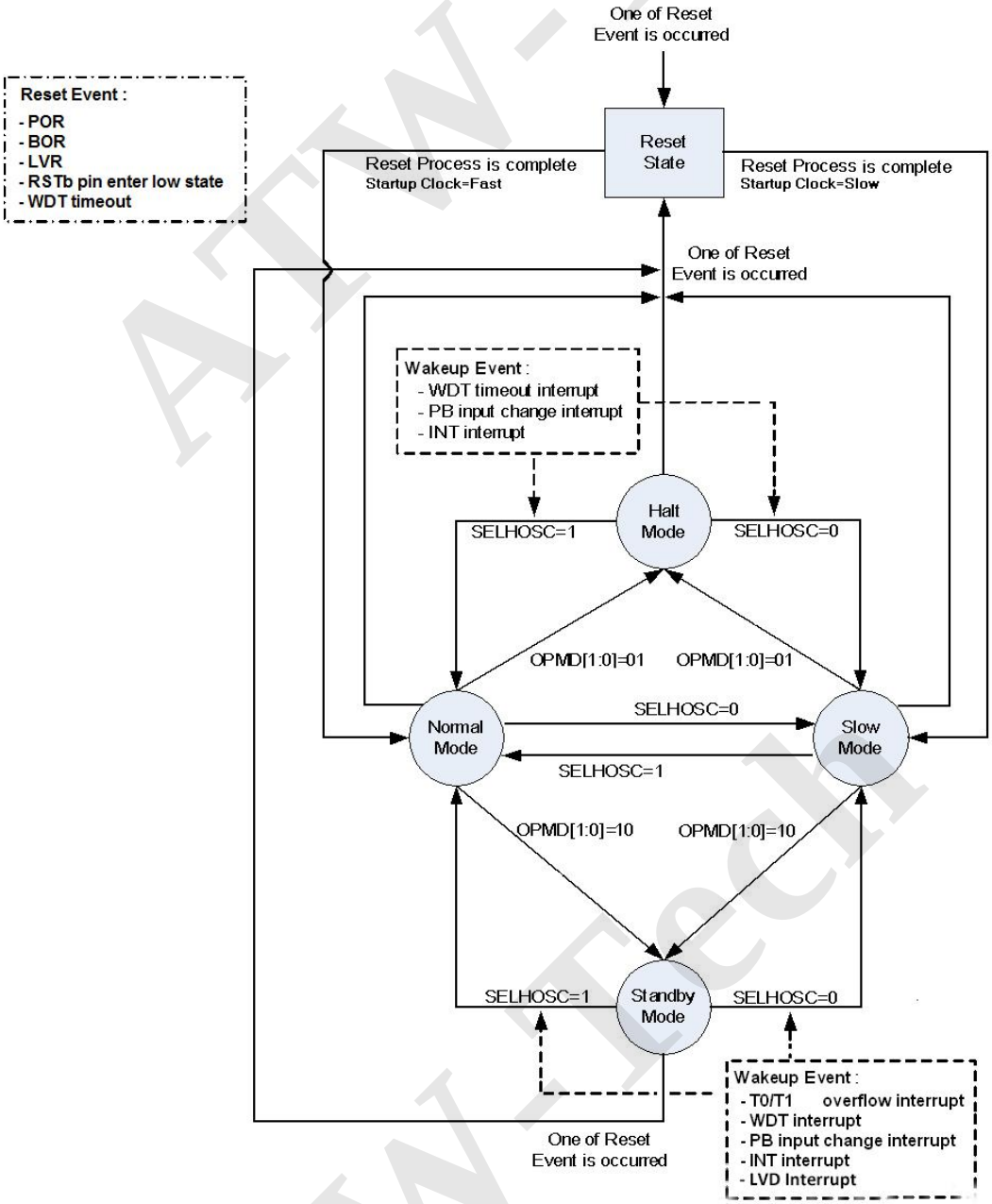



Figure 19 Four Operating Modes

### 3.13.1 Normal Mode

After any Reset Event is occurred and Reset Process is complete, AT8A513H will begin to execute program under Normal mode or Slow mode. Which mode is selected after Reset Process is determined by the Startup Clock configuration word. If Startup Clock= I_HRC, AT8A513H will enter Normal mode, if Startup Clock= I_LRC, AT8A513H will enter Slow mode. At Normal mode, $F_{HOSC}$ is selected as system oscillation in

order to provide highest performance and its power consumption will be the largest among four operating modes. After power on or any reset trigger is released, AT8A513H will enter Normal mode after reset process is complete.

Instruction execution is based on $F_{HOSC}$ and all peripheral modules may be active according to corresponding module enable bit.

The $F_{LOSC}$ is still active and running.

IC can switch to Slow mode by writing 0 to register bit SELHOSC (OSCCR[0]).

IC can switch to Standby mode or Halt mode by programming register bits OPMD[1:0] (OSCCR[3:2]).

For real time clock applications, the AT8A513H can run in normal mode, at the same time the low-frequency clock. Low Oscillator Frequency connects to timer0 clock. This is made possible by setting LCKTM0 to 1 and corresponding configuration word Timer0 source setting to 1.

### 3.13.2 Slow Mode

AT8A513H will enter Slow mode by writing 0 to register bit SELHOSC. At Slow mode, $F_{LOSC}$ is selected as system oscillation in order to save power consumption but still keep IC running. However, $F_{HOSC}$ will not be disabled automatically by AT8A513H. Therefore user can write 0 to register bit STPHOSC (OSCCR[1]) in slow mode to reduce power consumption further. But it is noted that it is forbidden to enter slow mode and stop $F_{HOSC}$ at the same time, one must enter slow mode first, then disable $F_{HOSC,}$ or the program may hang on.

Instruction execution is based on $F_{LOSC}$ and all peripheral modules may be active according to corresponding module enable bit.

$F_{HOSC}$ can be disabled by writing 1 to register bit STPHOSC.

IC can switch to Standby mode or Halt mode by programming register bits OPMD[1:0].

IC can switch to Normal mode by writing 1 to SELHOSC.

### 3.13.3 Standby Mode

AT8A513H will enter Standby mode by writing 10b to register bits OPMD[1:0]. At Standby mode, however, $F_{HOSC}$ will not be disabled automatically by AT8A513H and user has to enter slow mode and write 1 to register bit STPHOSC in order to stop $F_{HOSC}$ oscillation. Most of AT8A513H peripheral modules are disabled but Timer can be still active if register bit T0EN/T1EN is set to 1. Therefore AT8A513H can wake-up after Timer0/Timer1 is expired. The expiration period is determined by the register TMR0/TMR1, $F_{INST}$ and other configurations for Timer0/Timer1.

Instruction execution is stop and some peripheral modules may be active according to corresponding module enable bit.

FHOSC can be disabled by writing 1 to register bit STPHOSC.

The FLOSC is still active and running.

IC can wake-up from Standby mode if any of (a) Timer0 overflow /Timer1 underflow interrupt, (b) WDT timeout interrupt, (c) PB input change interrupt or (d) INT external interrupt is happened, (e) LVD Interrupt.

After wake-up from Standby mode, IC will return to Normal mode if SELHOSC=1, IC will return to Slow mode if SELHOSC=0.

It is not recommended to change oscillator mode (normal to slow / slow to normal) and enter standby mode at the same time.

### 3.13.4 Halt Mode

AT8A513H will enter Halt mode by executing instruction SLEEP or writing 01b to register bits OPMD[1:0]. After entering Halt mode, register bit /PD (STATUS[3]) will be clear to 0, register bit /TO (STATUS[4]) will be set to 1 and WDT will be clear but keep running.

At Halt mode, all of peripheral modules are disabled, instruction execution is stop and AT8A513H can only wake-up by some specific events. Therefore, Halt mode is the most power saving mode provided by AT8A513H.

Instruction execution is stop and all peripheral modules are disabled.

FHOSC and FLOSC are both disabled automatically.

IC can wake-up from Halt mode if any of (a) WDT timeout interrupt, (b) PB input change interrupt or (c) INT or external interrupt is happened.

After wake-up from Halt mode, IC will return to Normal mode if SELHOSC=1, IC will return to Slow mode if SELHOSC=0.

***Note: you can change STPHOSC and enter Halt mode in the same instruction***.

It is not recommended to change oscillator mode (normal to slow / slow to normal) and enter standby mode at the same time

### 3.13.5 Wake-up Stable Time

The wake-up stable time of Halt mode is 16*Fosc. There is no need of wake-up stable time for Standby mode because either $F_{HOSC}$ or $F_{LOSC}$ is still running at Standby mode.

Before AT8A513H enters Standby mode or Halt mode, user may execute instruction ENI. At this condition, AT8A513H will branch to address 0x008 in order to execute interrupt service routine after wake-up. If instruction DISI is executed before entering Standby mode or Halt mode, the next instruction will be executed after wake-up.

### 3.13.6 Summary of Operating Mode

The summary of four operating modes is described in the following table.

| Mode | Normal | Slow | Standby | Halt |
|---|---|---|---|---|
| $F_{HOSC}$ | Enabled | STPHOSC | STPHOSC | Disabled |
| $F_{LOSC}$ | Enabled | Enabled | Enabled | Disabled |
| Instruction Execution | Executing | Executing | Stop | Stop |
| Timer0/1 | T0EN / T1EN | T0EN / T1EN | T0EN / T1EN | Disabled |
| WDT | Option and WDTEN | Option and WDTEN | Option and WDTEN | Option and WDTEN |
| Other Modules | Module enable bit | Module enable bit | Module enable bit | All disabled |
| Wake-up Source | - | - | - Timer0 overflow<br>- Timer1 underflow<br>- WDT timeout<br>- PB input change<br>- INT<br>- LVD | - WDT timeout<br>- PB input change<br>- INT |

Table 17 Summary of Operating Modes

## 3.14 Reset Process

AT8A513H will enter Reset State and start Reset Process when one of following Reset Event is occurred:

Power-On Reset (POR) is occurred when $V_{DD}$ rising is detected.

Low-Voltage Reset (LVR) is occurred when operating $V_{DD}$ is below pre-defined voltage.

Pin RSTb is low state.

WDT timeout reset.

Moreover, value of all registers will be initialized to their initial value or unchanged if its initial value is unknown. The status bits /TO and /PD could be initialized according to which event causes reset. The /TO and /PD value and its associated event is summarized in the table below.

| Event | /TO | /PD |
|---|---|---|
| POR, LVR | 1 | 1 |
| RSTb reset from non-Halt mode | unchanged | unchanged |
| RSTb reset from Halt mode | 1 | 1 |
| WDT reset from non-Halt mode | 0 | 1 |
| WDT reset from Halt mode | 0 | 0 |
| SLEEP executed | 1 | 0 |
| CLRWDT executed | 1 | 1 |

Table 18 Summary of /TO & /PD Value and its Associated Event

After Reset Event is released, AT8A513H will start Reset Process. It will wait certain amount of period for oscillation stable no matter what kind of oscillator is adopted. This period is called power-up reset time and is determined by three-bit configuration words which can be 140u, 4.5ms, 18ms, 72ms or 288ms. After oscillator is stable, AT8A513H will wait further 16 clock cycles of FOSC (oscillator start-up time, OST) and Reset Process is complete.

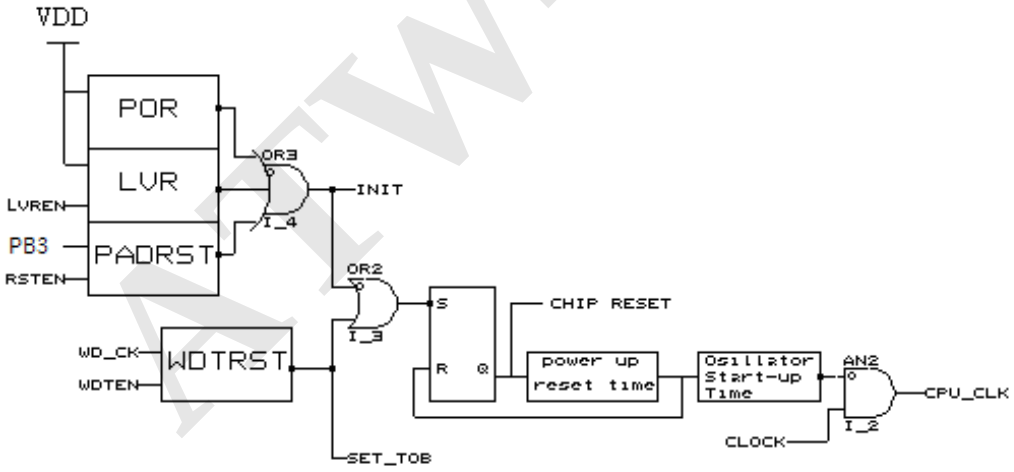



Figure 20 Block diagram of on-chip reset circuit

For slow VDD power-up, it is recommended to use RSTb reset, as the following figure.

It is recommended the R value should be not greater than 40kΩ.

The R1 value=100Ω to 1kΩ will prevent high current, ESD or Electrical overstress flowing into reset pin.

The diode helps discharge quickly when power down.

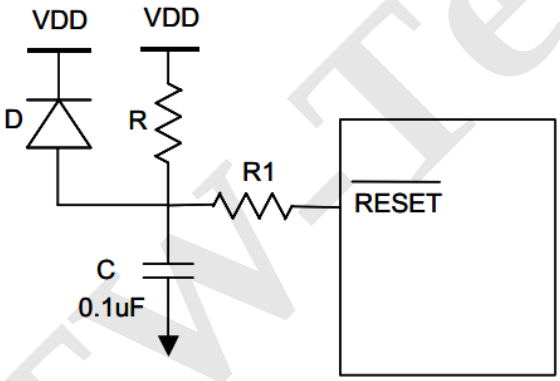



Figure 21 Block Diagram of Reset Application

## 4. Instruction Set

AT8A513H provides 55 powerful instructions for all kinds of applications.

| Inst. | OP 1 | OP 2 | Operation | Cyc. | Flag |
|---|---|---|---|---|---|
| ***Arithmetic Instructions*** | | | | | |
| ANDAR | R | d | dest = ACC & R | 1 | Z |
| IORAR | R | d | dest = ACC \| R | 1 | Z |
| XORAR | R | d | dest = ACC ⊕ R | 1 | Z |
| ANDIA | i | | ACC = ACC & i | 1 | Z |
| IORIA | i | | ACC = ACC \| i | 1 | Z |
| XORIA | i | | ACC = ACC ⊕ i | 1 | Z |
| RRR | R | d | Rotate right R | 1 | C |
| RLR | R | d | Rotate left R | 1 | C |
| BSR | R | bit | Set bit in R | 1 | - |
| BCR | R | bit | Clear bit in R | 1 | - |
| INCR | R | d | Increase R | 1 | Z |
| DECR | R | d | Decrease R | 1 | Z |
| COMR | R | d | dest = ~R | 1 | Z |
| ***Conditional Instructions*** | | | | | |
| BTRSC | R | bit | Test bit in R, skip if clear | 1 or 2 | - |
| BTRSS | R | bit | Test bit in R, skip if set | 1 or 2 | - |
| INCRSZ | R | d | Increase R, skip if 0 | 1 or 2 | - |
| DECRSZ | R | d | Decrease R, skip if 0 | 1 or 2 | - |
| ***Data Transfer Instructions*** | | | | | |
| MOVAR | R | | Move ACC to R | 1 | - |
| MOVR | R | d | Move R | 1 | Z |
| MOVIA | i | | Move immediate to ACC | 1 | - |
| SWAPR | R | d | Swap halves R | 1 | - |
| IOST | F | | Load ACC to F-page SFR | 1 | - |
| IOSTR | F | | Move F-page SFR to ACC | 1 | - |
| SFUN | S | | Load ACC to S-page SFR | 1 | - |
| SFUNR | S | | Move S-page SFR to ACC | 1 | - |
| T0MD | | | Load ACC to T0MD | 1 | - |
| T0MDR | | | Move T0MD to ACC | 1 | - |
| TABLEA | | | Read ROM | 2 | - |

| Inst. | OP 1 | OP 2 | Operation | Cyc. | Flag |
|---|---|---|---|---|---|
| ***Arithmetic Instructions*** | | | | | |
| ADDAR | R | d | dest = R + ACC | 1 | Z, DC, C |
| SUBAR | R | d | dest = R + (~ACC) | 1 | Z, DC, C |
| ADCAR | R | d | dest = R + ACC + C | 1 | Z, DC, C |
| SBCAR | R | d | dest = R + (~ACC) + C | 1 | Z, DC, C |
| ADDIA | i | | ACC = i + ACC | 1 | Z, DC, C |
| SUBIA | i | | ACC = i + (~ACC) | 1 | Z, DC, C |
| ADCIA | i | | ACC = i + ACC + C | 1 | Z, DC, C |
| SBCIA | i | | ACC = i + (~ACC) + C | 1 | Z, DC, C |
| DAA | | | Decimal adjust for ACC | 1 | C |
| CMPAR | R | | Compare R with ACC | 1 | Z, C |
| CLRA | | | Clear ACC | 1 | Z |
| CLRR | | | Clear R | 1 | Z |
| ***Other Instructions*** | | | | | |
| NOP | | | No operation | 1 | - |
| SLEEP | | | Go into Halt mode | 1 | /TO, /PD |
| CLRWDT | | | Clear Watch-Dog Timer | 1 | /TO, /PD |
| ENI | | | Enable interrupt | 1 | - |
| DISI | | | Disable interrupt | 1 | - |
| INT | | | Software Interrupt | 3 | - |
| RET | | | Return from subroutine | 2 | - |
| RETIE | | | Return from interrupt and enable interrupt | 2 | - |
| RETIA | i | | Return, place immediate in ACC | 2 | - |
| CALLA | | | Call subroutine by ACC | 2 | - |
| GOTOA | | | unconditional branch by ACC | 2 | - |
| CALL | adr | | Call subroutine | 2 | - |
| GOTO | adr | | unconditional branch | 2 | - |
| LCALL | adr | | Call subroutine | 2 | - |
| LGOTO | adr | | unconditional branch | 2 | - |

Table 19 Instruction Set

ACC: Accumulator.

adr: immediate address.

bit: bit address within an 8-bit register R.

**C:** Carry/Borrow bit

C=1, carry is occurred for addition instruction or borrow is **NOT** occurred for subtraction instruction.

C=0, carry is not occurred for addition instruction or borrow **IS** occurred for subtraction instruction.

d: Destination

If d is “0”, the result is stored in the ACC.

If d is “1”, the result is stored back in register R.

DC: Digital carry flag.

dest: Destination.

F: F-page SFR, F is 0x5 ~ 0xF.

i: 8-bit immediate data.

PC: Program Counter.

PCHBUF: High Byte Buffer of Program Counter.

**/PD:** Power down flag bit

/PD=1, after power-up or after instruction CLRWDT is executed.

/PD=0, after instruction SLEEP is executed.

Prescaler: Prescaler0 dividing rate.

R: R-page SFR, R is 0x00 ~0x3F.

S: S-page SFR, S is 0x0 ~ 0xF.

T0MD: T0MD register.

TBHP: The high-Byte at target address in ROM.

TBHD: Store the high-Byte data at target address in ROM.

**/TO:** Time overflow flag bit

/TO=1, after power-up or after instruction CLRWDT or SLEEP is executed.

/TO=0, WDT timeout is occurred.

WDT: Watchdog Timer Counter.

Z: Zero flag.

.

| ADCAR | Add ACC and R with Carry |
|---|---|
| Syntax: | ADCAR R, d |
| Operand: | 0 ≤ R ≤ 63<br>d = 0, 1. |
| Operation: | R + ACC + C → dest |
| Status affected: | Z, DC, C |
| Description: | Add the contents of ACC and register R with Carry. If d is 0, the result is stored in ACC. If d is 1, the result is stored back to R. |
| Cycle | 1 |
| Example: | ADCAR R, d<br>before executing instruction:<br>ACC=0x12, R=0x34, C=1, d=1.<br>after executing instruction:<br>R=0x47, ACC=0x12, C=0. |

| ADDAR | Add ACC and R |
|---|---|
| Syntax: | ADDAR R, d |
| Operand: | 0 ≤ R ≤63<br>d = 0, 1. |
| Operation: | ACC + R → dest |
| Status affected: | Z, DC, C |
| Description: | Add the contents of ACC and R. If d is 0, the result is stored in ACC. If d is 1, the result is stored back to R. |
| Cycle: | 1 |
| Example: | ADDAR R, d<br>before executing instruction:<br>ACC=0x12, R=0x34,C=1, d=1.<br>after executing instruction:<br>R=0x46, ACC=0x12, C=0. |

| ADCIA | Add ACC and Immediate with Carry |
|---|---|
| Syntax: | ADCIA i |
| Operand: | 0 ≤ i < 255 |
| Operation: | ACC + i + C → ACC |
| Status affected: | Z, DC, C |
| Description: | Add the contents of ACC and the 8-bit immediate data i with Carry. The result is placed in ACC. |
| Cycle: | 1 |
| Example: | ADCIA i<br>before executing instruction:<br>ACC=0x12, i=0x34, C=1.<br>after executing instruction:<br>ACC=0x47, C=0. |

| ADDIA | Add ACC and Immediate |
|---|---|
| Syntax: | ADDIA i |
| Operand: | 0 ≤ i < 255 |
| Operation: | ACC + i → ACC |
| Status affected: | Z, DC, C |
| Description: | Add the contents of ACC with the 8-bit immediate data i. The result is placed in ACC. |
| Cycle: | 1 |
| Example: | ADDIA i<br>before executing instruction:<br>ACC=0x12, i=0x34, C=1.<br>after executing instruction:<br>ACC=0x46, C=0. |

| ANDAR | AND ACC and R |
|---|---|
| Syntax: | ANDAR R, d |
| Operand: | 0 ≤ R ≤ 63.<br>d = 0, 1. |
| Operation: | ACC & R → dest |
| Status affected: | Z |
| Description: | The content of ACC is AND'ed with R. If d is 0, the result is stored in ACC. If d is 1, the result is stored back to R. |
| Cycle: | 1 |
| Example: | ANDAR R, d<br>before executing instruction:<br>ACC=0x5A, R=0xAF, d=1.<br>after executing instruction:<br>R=0x0A, ACC=0x5A, Z=0. |

| ANDIA | AND Immediate with ACC |
|---|---|
| Syntax: | ANDIA i |
| Operand: | 0 ≤ i < 255 |
| Operation: | ACC & i → ACC |
| Status affected: | Z |
| Description: | The content of ACC register is AND'ed with the 8-bit immediate data i. The result is placed in ACC. |
| Cycle: | 1 |
| Example: | ANDIA i<br>before executing instruction:<br>ACC=0x5A, i=0xAF.<br>after executing instruction:<br>ACC=0x0A, Z=0. |

| BCR | Clear Bit in R |
|---|---|
| Syntax: | BCR R, bit |
| Operand: | 0 ≤ R ≤ 63<br>0 ≤ bit ≤ 7 |
| Operation: | 0 → R[bit] |
| Status affected: | -- |
| Description: | Clear the bit$^{th}$ position in R. |
| Cycle: | 1 |
| Example: | BCR R,B2<br>before executing instruction:<br>R=0x5A, B2=0x3.<br>after executing instruction:<br>R=0x52. |

| BSR | Set Bit in R |
|---|---|
| Syntax: | BSR R, bit |
| Operand: | 0 ≤ R ≤ 63<br>0 ≤ bit ≤ 7 |
| Operation: | 1 → R[bit] |
| Status affected: | -- |
| Description: | Set the bit$^{th}$ position in R. |
| Cycle: | 1 |
| Example: | BSR R,B2<br>before executing instruction:<br>R=0x5A, B2=0x2.<br>after executing instruction:<br>R=0x5E. |

| BTRSC | Test Bit in R and Skip if Clear |
|---|---|
| Syntax: | BTRSC R, bit |
| Operand: | 0 ≤ R ≤ 63<br>0 ≤ bit ≤ 7 |
| Operation: | Skip next instruction, if R[bit] = 0. |
| Status affected: | -- |
| Description: | If R[bit] = 0, the next instruction which is already fetched is discarded and a NOP is executed instead. Therefore it makes this instruction a two-cycle instruction. |
| Cycle: | 1 or 2(skip) |
| Example: | BTRSC R, B2<br>Instruction1<br>Instruction2<br>before executing instruction:<br>R=0x5A, B2=0x2.<br>after executing instruction:<br>because R[B2]=0, instruction1 will not be executed, the program will start execute instruction from instruction2. |

| BTRSS | Test Bit in R and Skip if Set |
|---|---|
| Syntax: | BTRSS R, bit |
| Operand: | 0 ≤ R ≤ 63<br>0 ≤ bit ≤ 7 |
| Operation: | Skip next instruction, if R[bit] = 1. |
| Status affected: | -- |
| Description: | If R[bit] = 1, the next instruction which is already fetched is discarded and a NOP is executed instead. Therefore it makes this instruction a two-cycle instruction. |
| Cycle: | 1 or 2(skip) |
| Example: | BTRSS R, B2<br>Instruction2<br>Instruction3<br>before executing instruction:<br>R=0x5A, B2=0x3.<br>after executing instruction:<br>because R[B2]=1, instruction2 will not be executed, the program will start execute instruction from instruction3. |

| CALL | Call Subroutine |
|---|---|
| Syntax: | CALL adr |
| Operand: | 0 ≤ adr < 255 |
| Operation: | PC + 1 → Top of Stack<br>{PCHBUF, adr} → PC |
| Status affected: | -- |
| Description: | The return address (PC + 1) is pushed onto top of Stack. The 8-bit immediate address adr is loaded into PC[7:0] and PCHBUF[1:0] is loaded into PC[9:8]. |
| Cycle: | 2 |
| Example: | CALL SUB<br>before executing instruction:<br>PC=A0. Stack pointer=1<br>after executing instruction:<br>PC=address of SUB, Stack[1] = A0+1, Stack pointer=2. |

| CALLA | Call Subroutine |
|---|---|
| Syntax: | CALLA |
| Operand: | -- |
| Operation: | PC + 1 → Top of Stack<br>{TBHP, ACC} → PC |
| Status affected: | -- |
| Description: | The return address (PC + 1) is pushed onto top of Stack. The contents of TBHP[1:0] is loaded into PC[9:8] and ACC is loaded into PC[7:0]. |
| Cycle: | 2 |
| Example: | CALLA<br>before executing instruction:<br>TBHP=0x02, ACC=0x34. PC=A0. Stack pointer=1.<br>after executing instruction:<br>PC=0x234, Stack[1]=A0+1, Stack pointer=2. |

| CLRA | Clear ACC |
|---|---|
| Syntax: | CLRA |
| Operand: | -- |
| Operation: | 00h → ACC<br>1 → Z |
| Status affected: | Z |
| Description: | ACC is clear and Z is set to 1. |
| Cycle: | 1 |
| Example: | CLRA<br>before executing instruction:<br>ACC=0x55, Z=0.<br>after executing instruction:<br>ACC=0x00, Z=1. |

| CLRR | Clear R |
|---|---|
| Syntax: | CLRR R |
| Operand: | 0 ≤ R ≤ 63 |
| Operation: | 00h → R<br>1 → Z |
| Status affected: | Z |
| Description: | The content of R is clear and Z is set to 1. |
| Cycle: | 1 |
| Example: | CLRR R<br>before executing instruction:<br>R=0x55, Z=0.<br>after executing instruction:<br>R=0x00, Z=1. |

| COMR | Complement R |
|---|---|
| Syntax: | COMR R, d |
| Operand: | 0 ≤ R ≤ 63<br>d = 0, 1. |
| Operation: | ~R → dest |
| Status affected: | Z |
| Description: | The content of R is complemented. If d is 0, the result is stored in ACC. If d is 1, the result is stored back to R. |
| Cycle: | 1 |
| Example: | COMR, d<br>before executing instruction:<br>R=0xA6, d=1, Z=0.<br>after executing instruction:<br>R=0x59, Z=0. |

| CLRWDT | Clear Watch-Dog Timer |
|---|---|
| Syntax: | CLRWDT |
| Operand: | -- |
| Operation: | 00h → WDT,<br>00h → WDT prescaler<br>1 → /TO<br>1 → /PD |
| Status affected: | /TO, /PD |
| Description: | Executing CLRWDT will reset WDT, Prescaler0 if it is assigned to WDT. Moreover, status bits /TO and /PD will be set to 1. |
| Cycle: | 1 |
| Example: | CLRWDT<br>before executing instruction:<br>/TO=0<br>after executing instruction:<br>/TO=1 |

| CMPAR | Compare ACC and R |
|---|---|
| Syntax: | CMPAR R |
| Operand: | 0 ≤ R ≤ 63 |
| Operation: | R - ACC → (No restore) |
| Status affected: | Z, C |
| Description: | Compare ACC and R by subtracting ACC from R with 2's complement representation. The content of ACC and R is not changed. |
| Cycle: | 1 |
| Example: | CMPAR R<br>before executing instruction:<br>R=0x34, ACC=12, Z=0, C=0.<br>after executing instruction:<br>R=0x34, ACC=12, Z=0, C=1. |

| DAA | Convert ACC Data Format from Hexadecimal to Decimal |
|---|---|
| Syntax: | DAA |
| Operand: | -- |
| Operation: | ACC(hex) → ACC(dec) |
| Status affected: | C |
| Description: | Convert ACC data format from hexadecimal to decimal after addition operation and restore result to ACC. DAA instruction must be placed immediately after addition operation if decimal format is required. Please note that interrupt should be disabled before addition instruction and enabled after DAA instruction to avoid unexpected result. |
| Cycle: | 1 |
| Example: | DISI<br>ADDAR R,d<br>DAA<br>ENI<br>before executing instruction:<br>ACC=0x28, R=0x25, d=0.<br>after executing instruction:<br>ACC=0x53, C=0. |

| DECR | Decrease R |
|---|---|
| Syntax: | DECR R, d |
| Operand: | 0 ≤ R ≤ 63<br>d = 0, 1. |
| Operation: | R - 1 → dest |
| Status affected: | Z |
| Description: | Decrease R. If d is 0, the result is stored in ACC. If d is 1, the result is stored back to R. |
| Cycle: | 1 |
| Example: | DECR R, d<br>before executing instruction:<br>R=0x01, d=1, Z=0.<br>after executing instruction:<br>R=0x00, Z=1. |

| DECRSZ | Decrease R, Skip if 0 |
|---|---|
| Syntax: | DECRSZ R, d |
| Operand: | 0 ≤ R ≤ 63<br>d = 0, 1. |
| Operation: | R - 1 → dest,<br>Skip if result = 0 |
| Status affected: | -- |
| Description: | Decrease R first. If d is 0, the result is stored in ACC. If d is 1, the result is stored back to R.<br>If result is 0, the next instruction which is already fetched is discarded and a NOP is executed instead. Therefore it makes this instruction a two-cycle instruction. |
| Cycle: | 1 or 2(skip) |
| Example: | DECRSZ R, d<br>instruction2<br>instruction3<br>before executing instruction:<br>R=0x1, d=1, Z=0.<br>after executing instruction:<br>R=0x0, Z=1, and instruction will skip instruction2 execution because the operation result is zero. |

| DISI | Disable Interrupt Globally |
|---|---|
| Syntax: | DISI |
| Operand: | -- |
| Operation: | Disable Interrupt, 0 → GIE |
| Status affected: | -- |
| Description: | GIE is clear to 0 in order to disable all interrupt requests. |
| Cycle: | 1 |
| Example: | DISI<br>before executing instruction:<br>GIE=1.<br>After executing instruction:<br>GIE=0. |

| ENI | Enable Interrupt Globally |
|---|---|
| Syntax: | ENI |
| Operand: | -- |
| Operation: | Enable Interrupt, 1 → GIE |
| Status affected: | -- |
| Description: | GIE is set to 1 in order to enable all interrupt requests. |
| Cycle: | 1 |
| Example: | ENI<br>before executing instruction:<br>GIE=0.<br>After executing instruction:<br>GIE=1. |

| GOTO | Unconditional Branch |
|---|---|
| Syntax: | GOTO adr |
| Operand: | 0 ≤ adr < 511 |
| Operation: | {PCHBUF, adr} → PC |
| Status affected: | -- |
| Description: | GOTO is an unconditional branch instruction. The 9-bit immediate address adr is loaded into PC[8:0] and PCHBUF[1] is loaded into PC[9]. |
| Cycle: | 2 |
| Example: | GOTO Level<br>before executing instruction:<br>PC=A0.<br>after executing instruction:<br>PC=address of Level. |

| GOTOA | Unconditional Branch |
|---|---|
| Syntax: | GOTOA |
| Operand: | -- |
| Operation: | {TBHP, ACC} → PC |
| Status affected: | -- |
| Description: | GOTOA is an unconditional branch instruction. The content of TBHP[1:0] is loaded into PC[9:8] and ACC is loaded into PC[7:0]. |
| Cycle: | 2 |
| Example: | GOTOA<br>before executing instruction:<br>PC=A0. TBHP=0x02, ACC=0x34.<br>after executing instruction:<br>PC=0x234. |

| INCR | Increase R |
|---|---|
| Syntax: | INCR R, d |
| Operand: | 0 ≤ R ≤ 63<br>d = 0, 1. |
| Operation: | R + 1 → dest. |
| Status affected: | Z |
| Description: | Increase R. If d is 0, the result is stored in ACC. If d is 1, the result is stored back to R. |
| Cycle: | 1 |
| Example: | INCR R, d<br>before executing instruction:<br>R=0xFF, d=1, Z=0.<br>after executing instruction:<br>R=0x00, Z=1. |

| INCRSZ | Increase R, Skip if 0 |
|---|---|
| Syntax: | INCRSZ R, d |
| Operand: | 0 ≤ R ≤ 63<br>d = 0, 1. |
| Operation: | R + 1 → dest,<br>Skip if result = 0 |
| Status affected: | -- |
| Description: | Increase R first. If d is 0, the result is stored in ACC. If d is 1, the result is stored back to R.<br>If result is 0, the next instruction which is already fetched is discarded and a NOP is executed instead. Therefore it makes this instruction a two-cycle instruction. |
| Cycle: | 1 or 2(skip) |
| Example: | INCRSZ R, d<br>instruction2,<br>instruction3.<br>before executing instruction:<br>R=0xFF, d=1, Z=0.<br>after executing instruction:<br>R=0x00, Z=1. And the program will skip instruction2 execution because the operation result is zero. |

| INT | Software Interrupt |
|---|---|
| Syntax: | INT |
| Operand: | -- |
| Operation: | PC + 1 → Top of Stack,<br>001h → PC |
| Status affected: | -- |
| Description: | Software interrupt. First, return address (PC + 1) is pushed onto the Stack. The address 0x001 is loaded into PC[9:0]. |
| Cycle: | 3 |
| Example: | INT<br>before executing instruction:<br>PC=address of INT code.<br>after executing instruction:<br>PC=0x01. |

| IORAR | OR ACC with R |
|---|---|
| Syntax: | IORAR R, d |
| Operand: | 0 ≤ R ≤ 63<br>d = 0, 1. |
| Operation: | ACC \| R → dest |
| Status affected: | Z |
| Description: | OR ACC with R. If d is 0, the result is stored in ACC. If d is 1, the result is stored back to R. |
| Cycle: | 1 |
| Example: | IORAR R, d<br>before executing instruction:<br>R=0x50, ACC=0xAA, d=1, Z=0.<br>after executing instruction:<br>R=0xFA, ACC=0xAA, Z=0. |

| IORIA | OR Immediate with ACC |
|---|---|
| Syntax: | IORIA i |
| Operand: | 0 ≤ i < 255 |
| Operation: | ACC \| i → ACC |
| Status affected: | Z |
| Description: | OR ACC with 8-bit immediate data i. The result is stored in ACC. |
| Cycle: | 1 |
| Example: | IORIA i<br>before executing instruction:<br>i=0x50, ACC=0xAA, Z=0.<br>after executing instruction:<br>ACC=0xFA, Z=0. |

| IOSTR | Move F-page SFR to ACC |
|---|---|
| Syntax: | IOSTR F |
| Operand: | 0 ≤ F ≤ 15 |
| Operation: | F-page SFR → ACC |
| Status affected: | -- |
| Description: | Move F-page SFR F to ACC. |
| Cycle: | 1 |
| Example: | IOSTR F<br>before executing instruction:<br>F=0x55, ACC=0xAA.<br>after executing instruction:<br>F=0x55, ACC=0x55. |

| IOST | Load F-page SFR from ACC |
|---|---|
| Syntax: | IOSTF |
| Operand: | 0 ≤ F≤ 15 |
| Operation: | ACC → F-page SFR |
| Status affected: | -- |
| Description: | F-page SFR F is loaded by content of ACC. |
| Cycle: | 1 |
| Example: | IOST F<br>before executing instruction:<br>F=0x55, ACC=0xAA.<br>after executing instruction:<br>F=0xAA, ACC=0xAA. |

| LCALL | Call Subroutine |
|---|---|
| Syntax: | LCALL adr |
| Operand: | 0 ≤ adr ≤ 1023 |
| Operation: | PC + 1 → Top of Stack,<br>adr → PC[9:0] |
| Status affected: | -- |
| Description: | The return address (PC + 1) is pushed onto top of Stack. The 10-bit immediate address adr is loaded into PC[9:0]. |
| Cycle: | 2 |
| Example: | LCALL SUB<br>before executing instruction:<br>PC=A0. Stack level=1<br>after executing instruction:<br>PC=address of SUB, Stack[1]= A0+1, Stack pointer =2. |

| LGOTO | Unconditional Branch |
|---|---|
| Syntax: | LGOTO adr |
| Operand: | 0 ≤ adr ≤ 1023 |
| Operation: | adr → PC[9:0]. |
| Status affected: | -- |
| Description: | LGOTO is an unconditional branch instruction. The 10-bit immediate address adr is loaded into PC[9:0]. |
| Cycle: | 2 |
| Example: | LGOTO Level<br>before executing instruction:<br>PC=A0.<br>after executing instruction:<br>PC=address of Level. |

| MOVIA | Move Immediate to ACC |
|---|---|
| Syntax: | MOVIA i |
| Operand: | 0 ≤ i < 255 |
| Operation: | i → ACC |
| Status affected: | -- |
| Description: | The content of ACC is loaded with 8-bit immediate data i. |
| Cycle: | 1 |
| Example: | MOVIA i<br>before executing instruction:<br>i=0x55, ACC=0xAA.<br>after executing instruction:<br>ACC=0x55. |

| MOVAR | Move ACC to R |
|---|---|
| Syntax: | MOVAR R |
| Operand: | 0 ≤ R ≤ 63 |
| Operation: | ACC → R |
| Status affected: | -- |
| Description: | Move content of ACC to R. |
| Cycle: | 1 |
| Example: | MOVAR R<br>before executing instruction:<br>R=0x55, ACC=0xAA.<br>after executing instruction:<br>R=0xAA, ACC=0xAA. |

| MOVR | Move to ACC or R |
|---|---|
| Syntax: | MOVR R, d |
| Operand: | 0 ≤ R ≤ 63<br>d = 0, 1. |
| Operation: | R → dest |
| Status affected: | Z |
| Description: | The content of R is move to destination. If d is 0, destination is ACC. If d is 1, destination is R and it can be used to check whether R is zero according to status flag Z after execution. |
| Cycle: | 1 |
| Example: | MOVR R, d<br>before executing instruction:<br>R=0x0, ACC=0xAA, Z=0, d=0.<br>after executing instruction:<br>R=0x0, ACC=0x00, Z=1. |

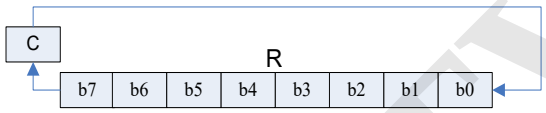
| NOP | No Operation |
|---|---|
| Syntax: | NOP |
| Operand: | -- |
| Operation: | No operation. |
| Status affected: | -- |
| Description: | No operation. |
| Cycle: | 1 |
| Example: | NOP<br>before executing instruction:<br>PC=A0<br>after executing instruction:<br>PC=A0+1 |

| RETIA | Return with Data in ACC |
|---|---|
| Syntax: | RETIA i |
| Operand: | 0 ≤ i < 255 |
| Operation: | i → ACC,<br>Top of Stack → PC |
| Status affected: | -- |
| Description: | ACC is loaded with 8-bit immediate data i and PC is loaded from top of Stack as return address and GIE is set to 1. |
| Cycle: | 2 |
| Example: | RETIA i<br>before executing instruction:<br>Stack pointer =2, i=0x55, ACC=0xAA.<br>after executing instruction:<br>PC=Stack[2], Stack pointer =1, ACC=0x55. |

| RETIE | Return from Interrupt and Enable Interrupt Globally |
|---|---|
| Syntax: | RETIE |
| Operand: | -- |
| Operation: | Top of Stack → PC<br>1 → GIE |
| Status affected: | -- |
| Description: | The PC is loaded from top of Stack as return address and GIE is set to 1. |
| Cycle: | 2 |
| Example: | RETIE<br>before executing instruction:<br>GIE=0, Stack level=2.<br>after executing instruction:<br>GIE=1, PC=Stack[2], Stack level =1. |

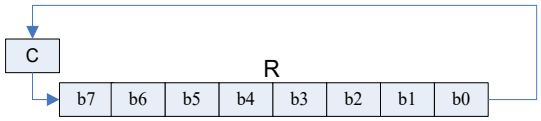
| RET | Return from Subroutine |
|---|---|
| Syntax: | RET |
| Operand: | -- |
| Operation: | Top of Stack → PC |
| Status affected: | -- |
| Description: | PC is loaded from top of Stack as return address. |
| Cycle: | 2 |
| Example: | RET<br>before executing instruction:<br>Stack level=2.<br>after executing instruction:<br>PC=Stack[2], Stack level=1. |

| RLR | Rotate Left R Through Carry |
|---|---|
| Syntax: | RLR R, d |
| Operand: | 0 ≤ R ≤ 63<br>d = 0, 1. |
| Operation: | C → dest[0], R[7] → C,<br>R[6:0] → dest[7:1] |




| | |
|---|---|
| Status affected: | C |
| Description: | The content of R is rotated one bit to the left through flag Carry. If d is 0, the result is placed in ACC. If d is 1, the result is stored back to R. |
| Cycle: | 1 |
| Example: | RLR R, d<br>before executing instruction:<br>R=0xA5, d=1, C=0.<br>after executing instruction:<br>R=0x4A, C=1. |

| RRR | Rotate Right R Through Carry |
|---|---|
| Syntax: | RRR R, d |
| Operand: | 0 ≤ R ≤ 63<br>d = 0, 1. |
| Operation: | C → dest[7], R[7:1] → dest[6:0],<br>R[0] → C |




| | |
|---|---|
| Status affected: | C |
| Description: | The content of R is rotated one bit to the right through flag Carry. If d is 0, the result is placed in ACC. If d is 1, the result is stored back to R. |
| Cycle: | 1 |
| Example: | RRR R, d<br>before executing instruction:<br>R=0xA5, d=1, C=0.<br>after executing instruction:<br>R=0x52, C=1. |

| SBCAR | Subtract ACC and Carry from R |
|---|---|
| Syntax: | SBCAR R, d |
| Operand: | 0 ≤ R ≤ 63<br>d = 0, 1. |
| Operation: | R + (~ACC) + C → dest |
| Status affected: | Z, DC, C |
| Description: | Subtract ACC and Carry from R with 2's complement representation. If d is 0, the result is placed in ACC. If d is 1, the result is stored back to R. |
| Cycle: | 1 |
| Example: | SBCAR R, d<br>(a) before executing instruction:<br>R=0x05, ACC=0x06, d=1, C=0.<br>after executing instruction:<br>R=0xFE, C=0. (-2)<br>(b) before executing instruction:<br>R=0x05, ACC=0x06, d=1, C=1.<br>after executing instruction:<br>R=0xFF, C=0. (-1)<br>(c) before executing instruction:<br>R=0x06, ACC=0x05, d=1, C=0.<br>after executing instruction:<br>R=0x00, C=1. (-0), Z=1.<br>(d) before executing instruction:<br>R=0x06, ACC=0x05, d=1, C=1.<br>after executing instruction:<br>R=0x1, C=1. (+1) |

| SBCIA | Subtract ACC and Carry from Immediate |
|---|---|
| Syntax: | SBCIA i |
| Operand: | 0 ≤ i < 255 |
| Operation: | i + (~ACC) + C → dest |
| Status affected: | Z, DC, C |
| Description: | Subtract ACC and Carry from 8-bit immediate data i with 2's complement representation. The result is placed in ACC. |
| Cycle: | 1 |
| Example: | SBCIA i<br>(a) before executing instruction:<br>i=0x05, ACC=0x06, C=0.<br>after executing instruction:<br>ACC=0xFE, C=0. (-2)<br>(b) before executing instruction:<br>i=0x05, ACC=0x06, C=1.<br>after executing instruction:<br>ACC=0xFF, C=0. (-1)<br>(c) before executing instruction:<br>i=0x06, ACC=0x05, C=0.<br>after executing instruction:<br>ACC=0x00, C=1. (-0), Z=1.<br>(d) before executing instruction:<br>i=0x06, ACC=0x05, C=1.<br>after executing instruction:<br>ACC=0x1, C=1. (+1) |

| SFUN | Load S-page SFR from ACC |
|---|---|
| Syntax: | SFUN S |
| Operand: | 0 ≤ S ≤15 |
| Operation: | ACC → S-page SFR |
| Status affected: | -- |
| Description: | S-page SFR S is loaded by content of ACC. |
| Cycle: | 1 |
| Example: | SFUN S<br>before executing instruction:<br>S=0x55, ACC=0xAA.<br>after executing instruction:<br>S=0xAA, ACC=0xAA. |

| SFUNR | Move S-page SFR to ACC |
|---|---|
| Syntax: | SFUNR S |
| Operand: | 0 ≤ S ≤ 15 |
| Operation: | S-page SFR → ACC |
| Status affected: | -- |
| Description: | Move S-page SFR S to ACC. |
| Cycle: | 1 |
| Example: | SFUNR S<br>before executing instruction:<br>S=0x55, ACC=0xAA.<br>after executing instruction:<br>S=0x55, ACC=0x55. |

| SLEEP | Enter Halt Mode |
|---|---|
| Syntax: | SLEEP |
| Operand: | -- |
| Operation: | 00h → WDT,<br>00h → WDT prescaler<br>1 → /TO<br>0 → /PD |
| Status affected: | /TO, /PD |
| Description: | WDT and Prescaler0 are clear to 0.<br>/TO is set to 1 and /PD is clear to 0.<br>IC enter Halt mode. |
| Cycle: | 1 |
| Example: | SLEEP<br>before executing instruction:<br>/PD=1, /TO=0.<br>after executing instruction:<br>/PD=0, /TO=1. |

| SUBAR | Subtract ACC from R |
|---|---|
| Syntax: | SUBAR R, d |
| Operand: | 0 ≤ R ≤ 63<br>d = 0, 1. |
| Operation: | R – ACC → dest |
| Status affected: | Z, DC, C |
| Description: | Subtract ACC from R with 2's complement representation. If d is 0, the result is placed in ACC. If d is 1, the result is stored back to R. |
| Cycle: | 1 |
| Example: | SUBAR R, d<br>(a) before executing instruction:<br>R=0x05, ACC=0x06, d=1.<br>after executing instruction:<br>R=0xFF, C=0. (-1)<br>(b) before executing instruction:<br>R=0x06, ACC=0x05, d=1.<br>after executing instruction:<br>R=0x01, C=1. (+1) |

| SUBIA | Subtract ACC from Immediate |
|---|---|
| Syntax: | SUBIA i |
| Operand: | 0 ≤ i < 255 |
| Operation: | i – ACC → ACC |
| Status affected: | Z, DC, C |
| Description: | Subtract ACC from 8-bit immediate data i with 2's complement representation. The result is placed in ACC. |
| Cycle: | 1 |
| Example: | SUBIA i<br>(a) before executing instruction:<br>i=0x05, ACC=0x06.<br>after executing instruction:<br>ACC=0xFF, C=0. (-1)<br>(b) before executing instruction:<br>i=0x06, ACC=0x05, d=1.<br>after executing instruction:<br>ACC=0x01, C=1. (+1) |

| SWAPR | Swap High/Low Nibble in R |
|---|---|
| Syntax: | SWAPR R, d |
| Operand: | 0 ≤ R ≤ 63<br>d = 0, 1. |
| Operation: | R[3:0] → dest[7:4].<br>R[7:4] → dest[3:0] |
| Status affected: | -- |
| Description: | The high nibble and low nibble of R is exchanged. If d is 0, the result is placed in ACC. If d is 1, the result is stored back to R. |
| Cycle: | 1 |
| Example: | SWAPR R, d<br>before executing instruction:<br>R=0xA5, d=1.<br>after executing instruction:<br>R=0x5A. |

| TABLEA | Read ROM data |
|---|---|
| Syntax: | TABLEA |
| Operand: | -- |
| Operation: | ROM data{ TBHP, ACC } [7:0] → ACC<br>ROM data{TBHP, ACC} [13:8] → TBHD. |
| Status affected: | -- |
| Description: | The 8 least significant bits of ROM pointed by {TBHP[2:0], ACC} is placed to ACC.<br>The 6 most significant bits of ROM pointed by {TBHP[2:0], ACC} is placed to TBHD[5:0]. |
| Cycle: | 2 |
| Example: | TABLEA<br>before executing instruction:<br>TBHP=0x02, CC=0x34.<br>TBHD=0x01.<br>ROM data[0x234]= 0x35AA<br>after executing instruction:<br>TBHD=0x35, ACC=0xAA. |

| T0MD | Load ACC to T0MD |
|---|---|
| Syntax: | T0MD |
| Operand: | -- |
| Operation: | ACC → T0MD |
| Status affected: | -- |
| Description: | The content of T0MD is loaded by ACC. |
| Cycle: | 1 |
| Example: | T0MD<br>before executing instruction:<br>T0MD=0x55, ACC=0xAA.<br>after executing instruction:<br>T0MD=0xAA. |

| T0MDR | Move T0MD to ACC |
|---|---|
| Syntax: | T0MDR |
| Operand: | -- |
| Operation: | T0MD → ACC |
| Status affected: | -- |
| Description: | Move the content of T0MD to ACC. |
| Cycle: | 1 |
| Example: | T0MDR<br>before executing instruction<br>T0MD=0x55, ACC=0xAA.<br>after executing instruction<br>ACC=0x55. |

| XORAR | Exclusive-OR ACC with R |
|---|---|
| Syntax: | XORAR R, d |
| Operand: | $0 \leq R \leq 63$<br>d = 0, 1. |
| Operation: | ACC ⊕ R → dest |
| Status affected: | Z |
| Description: | Exclusive-OR ACC with R. If d is 0, the result is placed in ACC. If d is 1, the result is stored back to R. |
| Cycle: | 1 |
| Example: | XORAR R, d<br>before executing instruction:<br>R=0xA5, ACC=0xF0, d=1.<br>after executing instruction:<br>R=0x55. |

| XORIA | Exclusive-OR Immediate with ACC |
|---|---|
| Syntax: | XORIA i |
| Operand: | 0 ≤ i < 255 |
| Operation: | ACC ⊕ i → ACC |
| Status affected: | Z |
| Description: | Exclusive-OR ACC with 8-bit immediate data i. The result is stored in ACC. |
| Cycle: | 1 |
| Example: | XORIA i<br>before executing instruction:<br>i=0xA5, ACC=0xF0.<br>after executing instruction:<br>ACC=0x55. |

## 5. Configuration Words

| Item | Name | Options |
|---|---|---|
| 1 | High IRC Frequency | 1. 1MHz 2. 2MHz 3. 4MHz<br>4. 8MHz 5. 16MHz 6. 20MHz |
| 2 | Instruction Clock | 1. 2 oscillator period 2. 4 oscillator period |
| 3 | WDT | 1. Watchdog Enable (Software control)<br>2. Watchdog Disable (Always disable) |
| 4 | WDT Event | 1. Watchdog Reset 2. Watchdog Interrupt |
| 5 | Timer0 source | 1. EX_CKI 2. I_LRC |
| 6 | PB.2 | 1. PB.2 is I/O 2. PB.2 is PWM 3. PB.2 is Buzzer |
| 7 | PB.3 | 1. PB.3 is I/O 2. PB.3 is reset |
| 8 | PB.4 | 1. PB.4 is I/O 2. PB.4 is instruction clock output |
| 9 | Startup Time | 1. 140us 2. 4.5ms 3. 18ms 4. 72ms 5. 288ms |
| 10 | WDT Time Base | 1. 3.5ms 2. 15ms 3. 60ms 4. 250ms |
| 11 | LVR Setting | 1. Register Control 2. Register Control + Halt mode Off<br>3. Always On 4. Operation mode On + Halt mode Off |
| 12 | LVR Voltage | 1. 1.6V 2. 1.8V 3. 2.0V 4. 2.2V 5. 2.4V<br>6. 2.7V 7. 3.0V 8. 3.3V 9. 3.6V ~~10. 4.2V~~ |
| 13 | VDD Voltage | 1. 3.0V 2. 4.5V 3. 5.0V |
| 14 | Read Output Data | 1. I/O port 2. Register |
| 15 | EX_CKI to Inst. Clock | 1. Sync 2. Async |
| 16 | Startup Clock | 1. I_HRC 2. I_LRC |
| 17 | Input High Voltage (VIH) | 1. 0.8VDD 2. 0.6VDD |
| 18 | Input Low Voltage (VIL) | 1. 0.3VDD 2. 0.2VDD |
| 19 | Input Voltage Schmitt Trigger | 1. Enable (depend on 17, 18) 2. Disable (17, 18 no use) |
| 20 | Sink / Drive Current | 1. Small 2. Normal |

Table 20 Configuration Words

# 6. Electrical Characteristics

## 6.1 Absolute Maximum Rating

| Symbol | Parameter | Rated Value | Unit |
|---|---|---|---|
| $V_{DD}$ - $V_{SS}$ | Supply voltage | -0.5 ~ +6.0 | V |
| $V_{IN}$ | Input voltage | $V_{SS}$-0.3V ~ $V_{DD}$+0.3 | V |
| $T_{OP}$ | Operating Temperature | -40 ~ +85 | °C |
| $T_{ST}$ | Storage Temperature | -40 ~ +125 | °C |

## 6.2 DC Characteristics

(All refer $F_{INST}$=$F_{HOSC}$/4, $F_{HOSC}$=16MHz@I_HRC, WDT enabled, ambient temperature $T_A$=25°C unless otherwise specified.)

| Symbol | Parameter | $V_{DD}$ | Min. | Typ. | Max. | Unit | Condition |
|---|---|---|---|---|---|---|---|
| $V_{DD}$ | Operating voltage | -- | 3.3 | -- | 5.5 | V | $F_{INST}$=10MHz @ I_HRC 20MHz/2T |
| | | | 2.7 | | | | $F_{INST}$=8MHz @ I_HRC 16MHz/2T |
| | | | 2.0 | | | | $F_{INST}$=5MHz @ I_HRC 20MHz/4T |
| | | | 1.8 | | | | $F_{INST}$=4MHz @ I_HRC 16MHz/4T & 8MHz/2T |
| | | | 1.6 | | | | $F_{INST}$=2MHz @ I_HRC 8MHz/4T & 4MHz/2T |
| | | | 1.6 | | | | $F_{INST}$=1MHz @ I_HRC 4MHz/4T |
| | | | 1.6 | | | | $F_{INST}$=8KHz @ I_LRC 32KHz/4T |
| $V_{IH}$ | Input high voltage | 5V | 4.0 | -- | -- | V | RSTb (0.8$V_{DD}$) |
| | | 3V | 2.4 | -- | -- | | |
| | | 5V | 3.5 | -- | -- | V | All other I/O pins, EX_CKI, INT |
| | | 3V | 2.1 | -- | -- | | 0.8$V_{DD}$ |
| | | 5V | 2.5 | -- | -- | V | All other I/O pins, EX_CKI, INT |
| | | 3V | 1.5 | -- | -- | | 0.6$V_{DD}$ |
| | | 5V | -- | 2.5 | -- | V | All other I/O pins, EX_CKI, INT |
| | | 3V | -- | 1.5 | -- | | No Schmitt Trigger (0.5$V_{DD}$) |
| $V_{IL}$ | Input low voltage | 5V | -- | -- | 1.0 | V | RSTb (0.2$V_{DD}$) |
| | | 3V | -- | -- | 0.6 | | |
| | | 5V | -- | -- | 1.5 | V | All other I/O pins, EX_CKI, INT |
| | | 3V | -- | -- | 0.9 | | 0.3$V_{DD}$ |
| | | 5V | -- | -- | 1.0 | V | All other I/O pins, EX_CKI, INT |
| | | 3V | -- | -- | 0.6 | | 0.2$V_{DD}$ |
| | | 5V | -- | 2.5 | -- | V | All other I/O pins, EX_CKI, INT |
| | | 3V | -- | 1.5 | -- | | No Schmitt Trigger (0.5$V_{DD}$) |

| Symbol | Parameter | $V_{DD}$ | Min. | Typ. | Max. | Unit | Condition |
|---|---|---|---|---|---|---|---|
| $I_{OH}$ | Output high current (Small current) | 5V | -- | 2.2 | -- | mA | $V_{OH}$=4.0V |
| | | 3V | -- | 1.2 | -- | | $V_{OH}$=2.0V |
| | Output high current (Normal current) | 5V | -- | 20 | -- | mA | $V_{OH}$=4.0V |
| | | 3V | -- | 12 | -- | | $V_{OH}$=2.0V |
| $I_{OL}$ | Output low current (Small current) | 5V | -- | 7.8 | -- | mA | $V_{OL}$=1.0V |
| | | 3V | -- | 4.5 | -- | | |
| | Output low current (Normal current) | 5V | -- | 42 | -- | mA | $V_{OL}$=1.0V |
| | | 3V | -- | 26 | -- | | |
| $I_{IR}$ | IR sink current | 5V | -- | 45 | -- | mA | $V_{OL}$=1.0V, Normal IR |
| | | 3V | -- | 26 | -- | | |
| $I_{OP}$ | Operating current | **Normal Mode** | | | | | |
| | | 5V | -- | 1.7 | -- | mA | $F_{HOSC}$=20MHz @ I_HRC/2 |
| | | 3V | -- | 0.7 | -- | | |
| | | 5V | -- | 1.4 | -- | mA | $F_{HOSC}$=20MHz @ I_HRC/4 |
| | | 3V | -- | 0.5 | -- | | |
| | | 5V | -- | 1.5 | -- | mA | $F_{HOSC}$=16MHz @ I_HRC/2 |
| | | 3V | -- | 0.6 | -- | | |
| | | 5V | -- | 1.3 | -- | mA | $F_{HOSC}$=16MHz @ I_HRC/4 |
| | | 3V | -- | 0.5 | -- | | |
| | | 5V | -- | 1.2 | -- | mA | $F_{HOSC}$=8MHz @ I_HRC/2 |
| | | 3V | -- | 0.4 | -- | | |
| | | 5V | -- | 0.9 | -- | mA | $F_{HOSC}$=8MHz @ I_HRC/4 |
| | | 3V | -- | 0.3 | -- | | |
| | | 5V | -- | 0.8 | -- | mA | $F_{HOSC}$=4MHz @ I_HRC/2 |
| | | 3V | -- | 0.3 | -- | | |
| | | 5V | -- | 0.7 | -- | mA | $F_{HOSC}$=4MHz @ I_HRC/4 |
| | | 3V | -- | 0.2 | -- | | |
| | | 5V | -- | 0.5 | -- | mA | $F_{HOSC}$=1MHz @ I_HRC/2 |
| | | 3V | -- | 0.2 | -- | | |
| | | 5V | -- | 0.5 | -- | mA | $F_{HOSC}$=1MHz @ I_HRC/4 |
| | | 3V | -- | 0.2 | -- | | |
| | | **Slow mode** | | | | | |
| | | 5V | -- | 9.9 | -- | uA | $F_{HOSC}$ disabled, $F_{LOSC}$=32KHz @ I_LRC/2 |
| | | 3V | -- | 4.9 | -- | | |
| | | 5V | -- | 6.6 | -- | uA | $F_{HOSC}$ disabled, $F_{LOSC}$=32KHz @ I_LRC/4 |
| | | 3V | -- | 3.6 | -- | | |

| Symbol | Parameter | $V_{DD}$ | Min. | Typ. | Max. | Unit | Condition |
|---|---|---|---|---|---|---|---|
| $I_{STB}$ | Standby current | 5V | -- | 3.5 | -- | uA | Standby mode, $F_{HOSC}$ disabled, $F_{LOSC}$=32KHz @ I_LRC/4 |
| | | 3V | -- | 2.5 | -- | | |
| $I_{HALT}$ | Halt current | 5V | -- | -- | 0.5 | uA | Halt mode, WDT disabled. |
| | | 3V | -- | -- | 0.2 | | |
| | | 5V | -- | -- | 5.0 | uA | Halt mode, WDT enabled. |
| | | 3V | -- | -- | 3.0 | | |
| $R_{PH}$ | Pull-High resistor | 5V | -- | 65 | -- | kΩ | Pull-High resistor (not include PB3) |
| | | 3V | -- | 120 | -- | | |
| | | 5V | -- | 85 | -- | kΩ | Pull-High resistor (PB3) |
| | | 3V | -- | 85 | -- | | |
| $R_{PL}$ | Pull-Low resistor | 5V | -- | 55 | -- | kΩ | Pull-Low resistor |
| | | 3V | -- | 105 | -- | | |

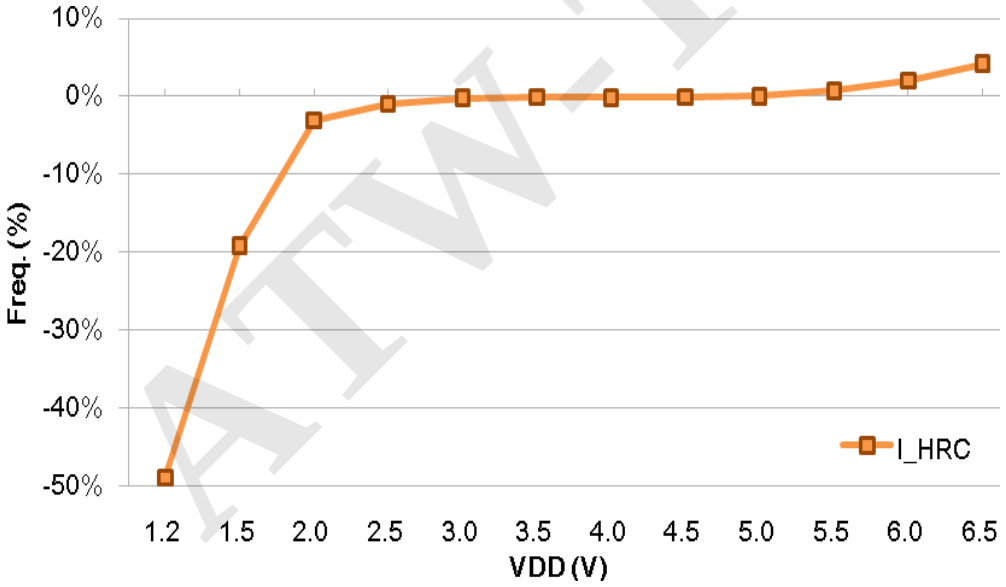
## 6.3 OSC Characteristic

Measurement conditions $V_{DD}$ Voltage, $T_A$ Temperature are equal to programming conditions.)

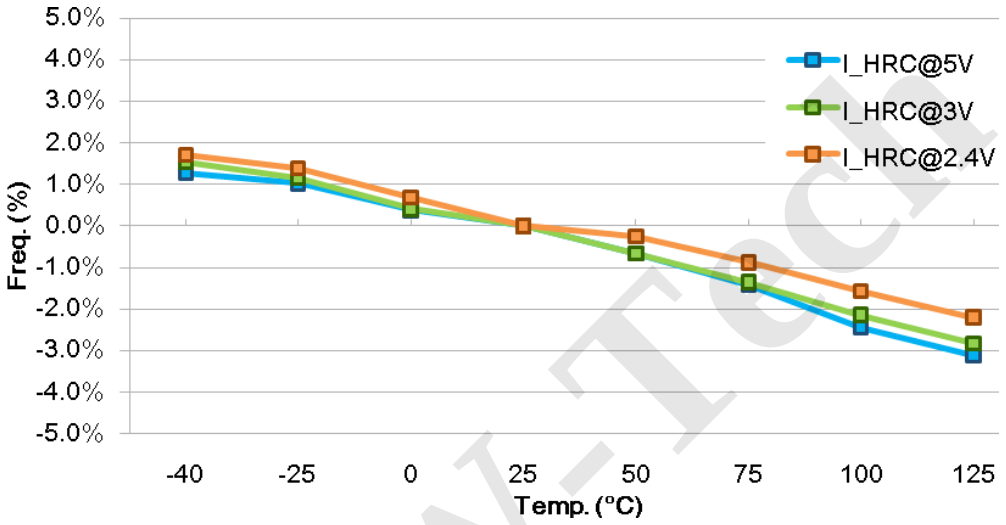
| Parameter | Min. | Typ. | Max. | Unit | Condition |
|---|---|---|---|---|---|
| I_HRC deviation by socket | | | ±1 | % | Socket installed directly on writer. |
| I_HRC deviation by handler | | | ±3 | % | Handler condition with correct setup. |
| I_LRC deviation by handler | | | ±20 | % | |

## 6.4 Characteristic Graph

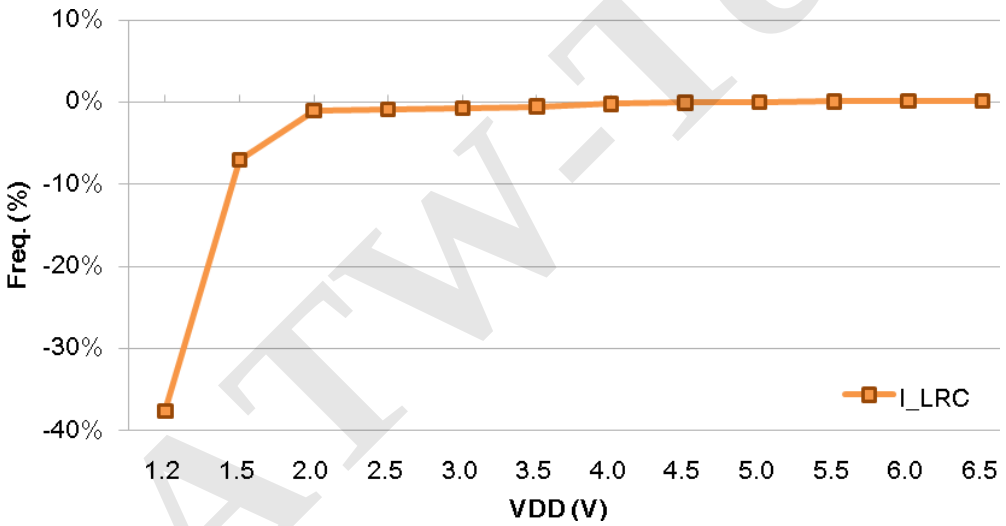
### 6.4.1 Frequency vs. VDD of I_HRC



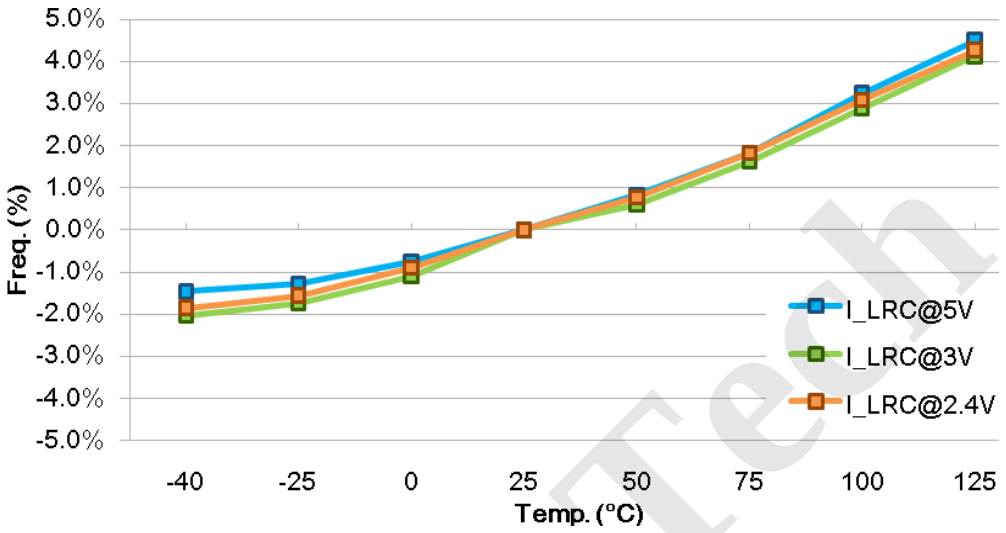

### 6.4.2 Frequency vs. Temperature of I_HRC



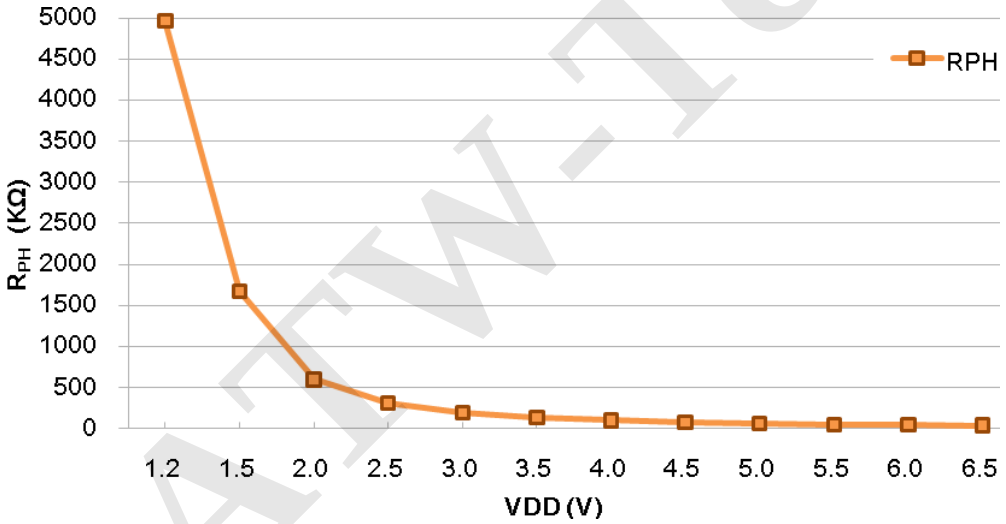
### 6.4.3 Frequency vs. VDD of I_LRC



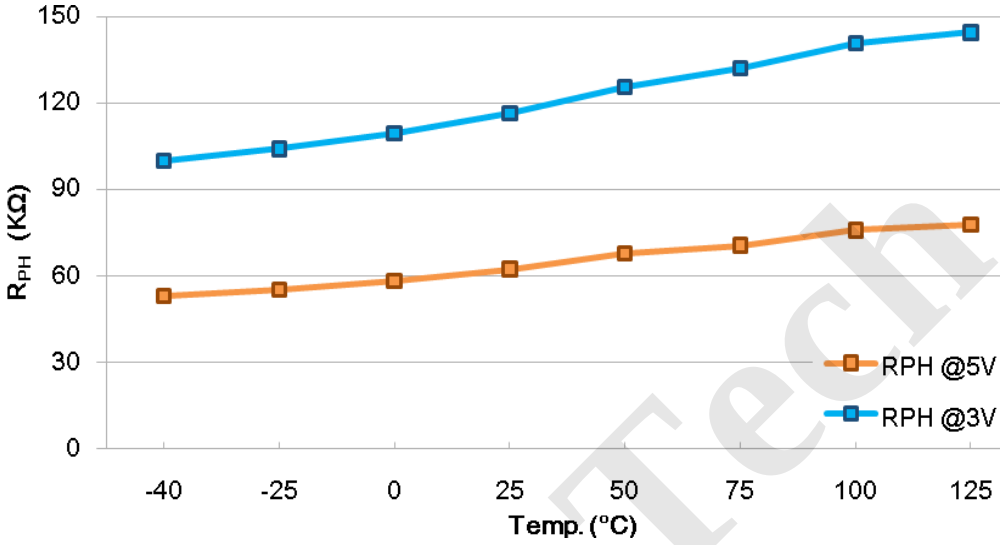

### 6.4.4 Frequency vs. Temperature of I_LRC



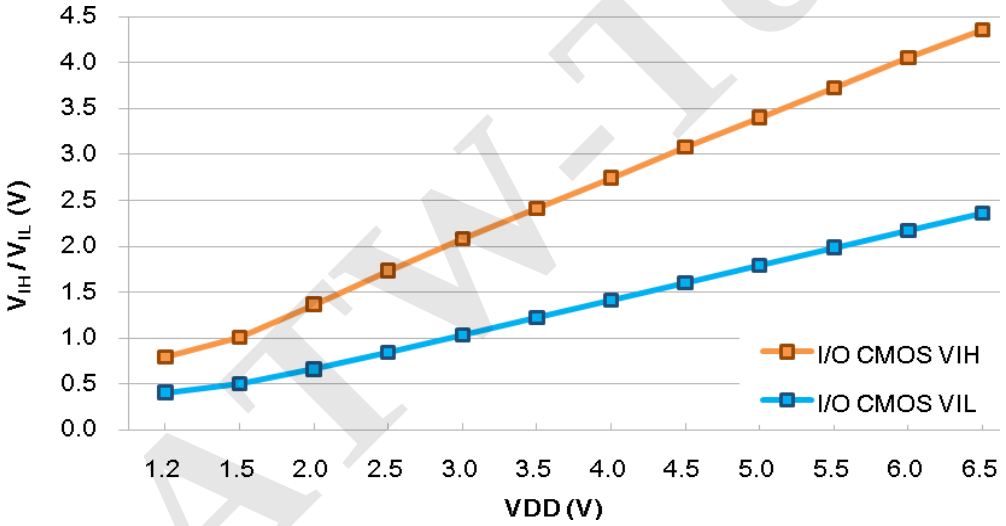
### 6.4.5 Pull High Resistor vs. VDD

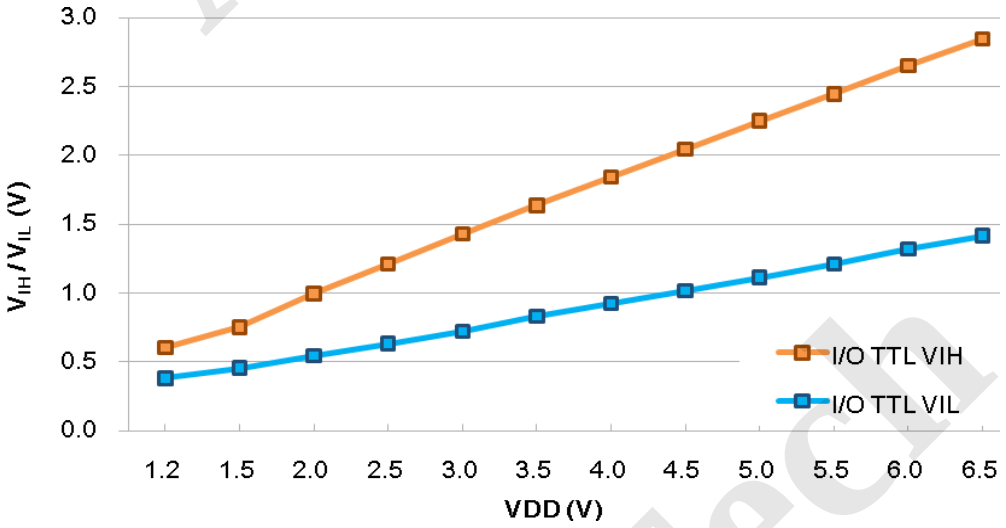


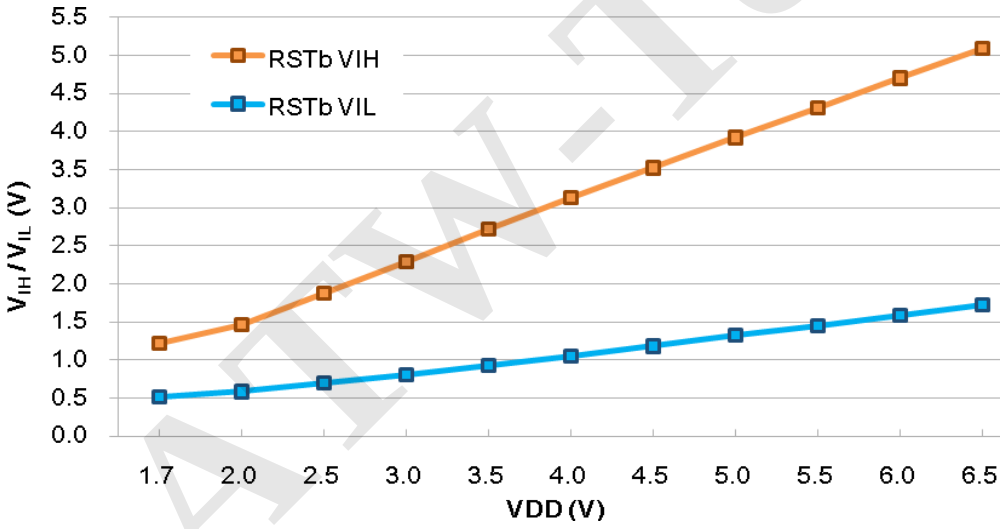

### 6.4.6 Pull High Resistor vs. Temperature



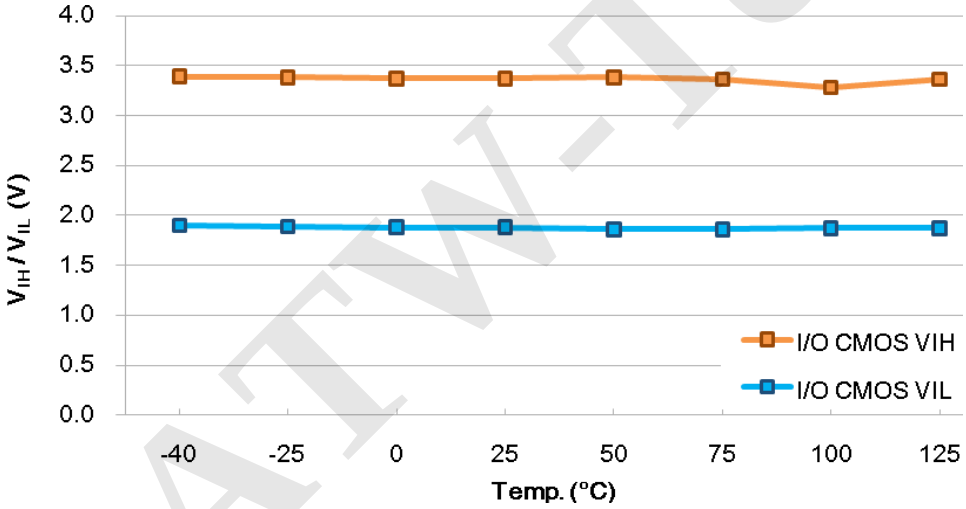
### 6.4.7 VIH/VIL vs. VDD

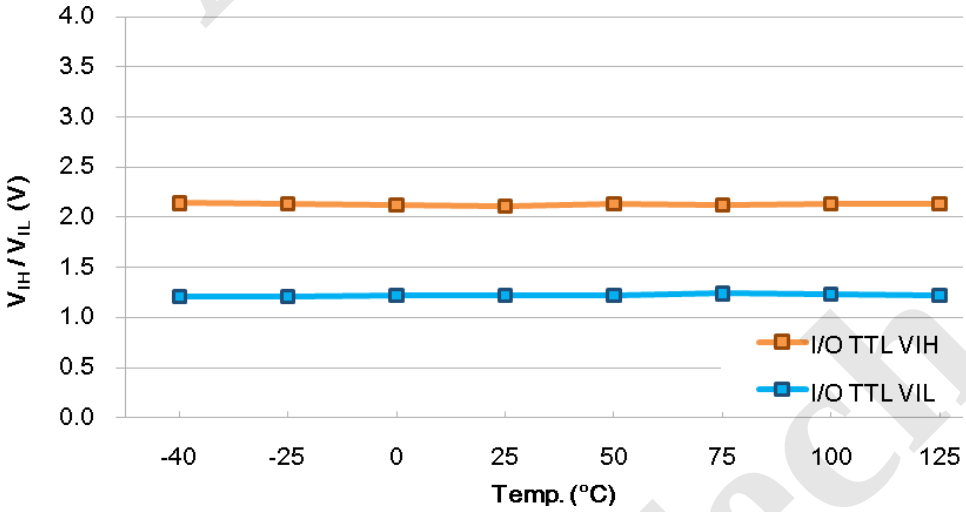


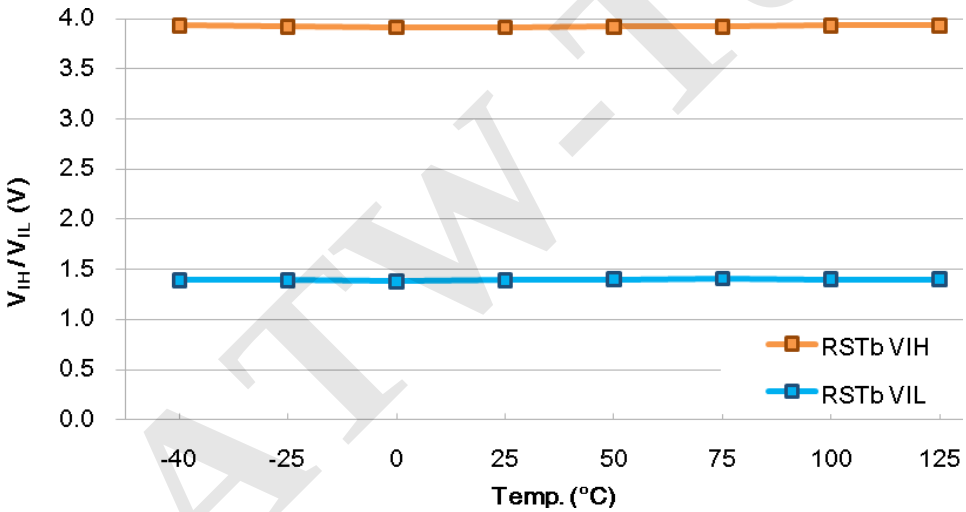



### 6.4.8 VIH/VIL vs. Temperature



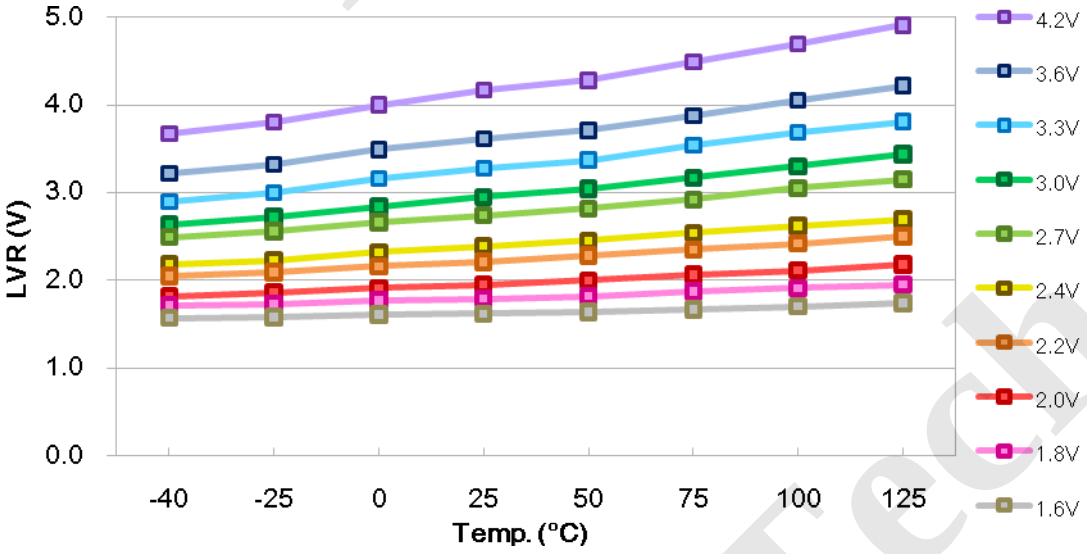



## 6.5 Recommended Operating Voltage

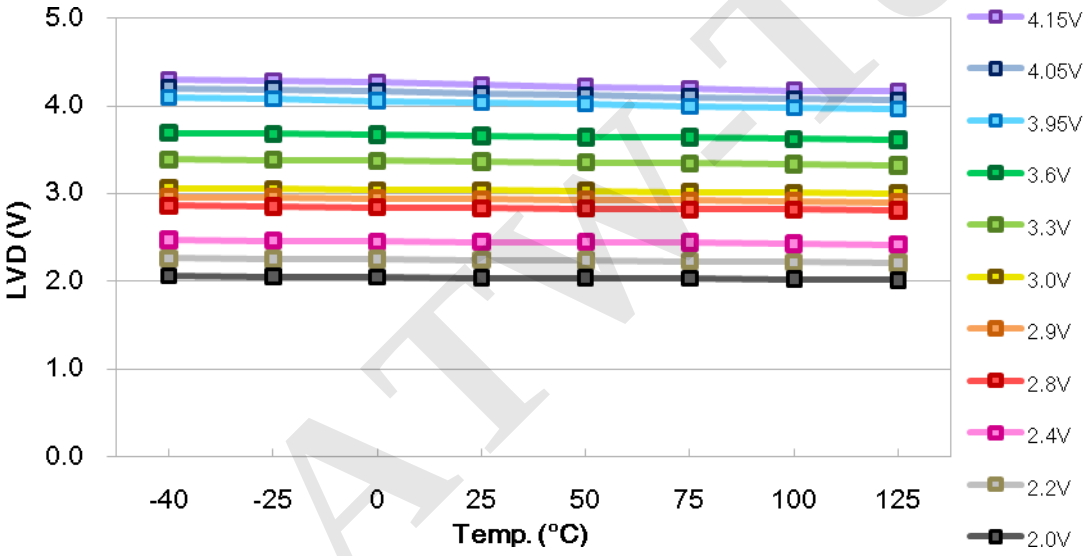
Recommended Operating Voltage (Temperature range: -40 °C ~ +85 °C)

| Frequency | Min. Voltage | Max. Voltage | LVR Setting | LVR Setting (25°C) |
|---|---|---|---|---|
| 20M/2T | 3.3V | 5.5V | 3.6V | 3.3V |
| 16M/2T | 2.7V | 5.5V | 3.0V | 2.7V |
| 20M/4T | 2.0V | 5.5V | 2.4V | 2.0V |
| 16M/4T | 1.8V | 5.5V | 2.2V | 1.8V |
| 8M/2T | 1.6V | 5.5V | 1.8V | 1.6V |
| 4M/4T | 1.6V | 5.5V | 1.8V | 1.6V |

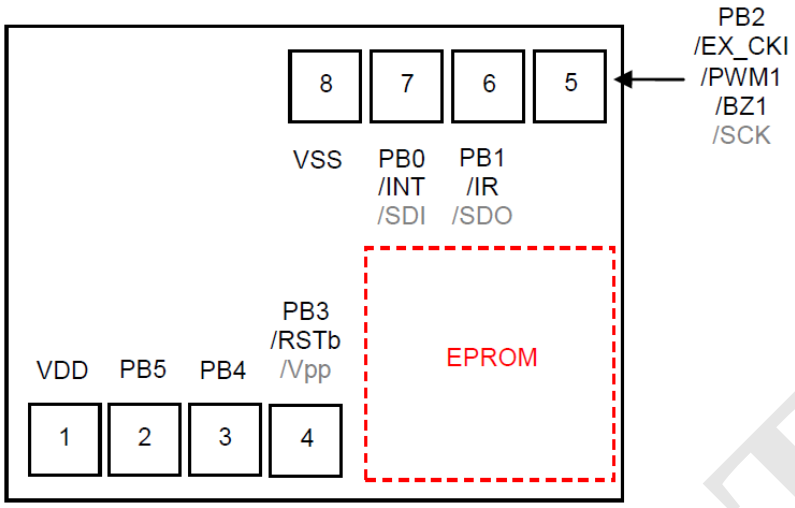
## 6.6 LVR vs. Temperature



### 6.7 LVD vs. Temperature

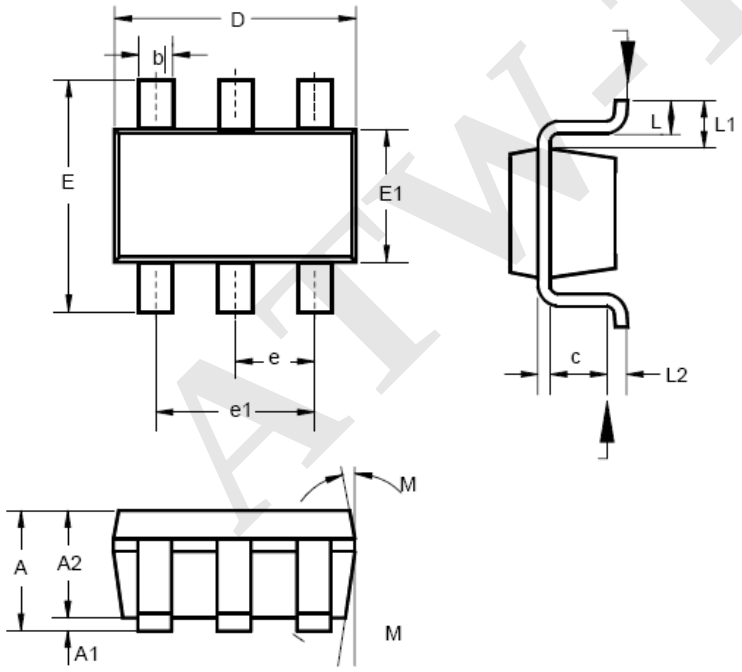



## 7. Die Pad Diagram



## 8. Package Dimension

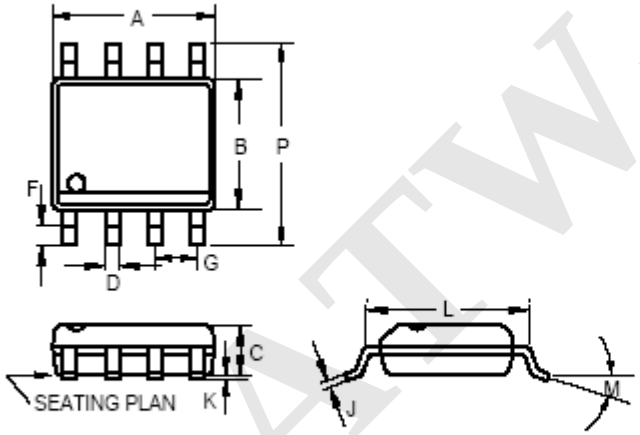
### 8.1 6-Pin Plastic SOT23-6 (63 mil)




| | INCHES | | | MILLIMETERS | | |
|---|---|---|---|---|---|---|
| | MIN | TYP | MAX | MIN | TYP | MAX |
| A | - | - | 0.057 | - | - | 1.45 |
| A1 | - | - | 0.006 | - | - | 0.15 |
| A2 | 0.035 | 0.045 | 0.051 | 0.90 | 1.15 | 1.30 |
| b | 0.012 | - | 0.020 | 0.30 | - | 0.50 |
| c | 0.003 | - | 0.009 | 0.08 | - | 0.22 |
| D | 0.114 BSC | | | 2.90 BSC | | |
| E | 0.110 BSC | | | 2.80 BSC | | |
| E1 | 0.063`BSC | | | 1.60 BSC | | |
| e | 0.037 BSC | | | 0.95 BSC | | |
| e1 | 0.075BSC | | | 1.90 BSC | | |
| L | 0.012 | 0.018 | 0.024 | 0.30 | 0.45 | 0.60 |
| L1 | 0.024 REF | | | 0.60 REF | | |
| L2 | 0.010 BSC | | | 0.25 BSC | | |
| °M | 5° | 10° | 15° | 5° | 10° | 15° |

**Note: For SOT23-6 IC, 3000 units per reel.**

### 8.2 8-Pin Plastic SOP (150 mil)




| | INCHES | | | MILLIMETERS | | |
|---|---|---|---|---|---|---|
| | MIN | TYP | MAX | MIN | TYP | MAX |
| A | 0.183 | - | 0.202 | 4.65 | - | 5.13 |
| B | 0.144 | - | 0.163 | 3.66 | - | 4.14 |
| C | 0.068 | - | 0.074 | 1.35 | - | 1.88 |
| D | 0.010 | - | 0.020 | 0.25 | - | 0.51 |
| F | 0.015 | - | 0.035 | 0.38 | - | 0.89 |
| G | 0.050 BSC | | | 1.27 BSC | | |
| J | 0.007 | - | 0.010 | 0.19 | - | 0.25 |
| K | 0.005 | - | 0.010 | 0.13 | - | 0.25 |
| L | 0.189 | - | 0.205 | 4.80 | - | 5.21 |
| M | - | - | 8° | - | - | 8° |
| P | 0.228 | - | 0.244 | 5.79 | - | 6.20 |

Note: f

**Note: For 8-pin SOP, 100 units per tube.**

## 9. Ordering Information

| *P/N* | *Package Type* | *Pin Count* | *Package Width* | *Shipping* |
|---|---|---|---|---|
| AT8A513H | Die | -- | -- | -- |
| AT8A513HS6 | SOT23-6 | 6 | 63 mil | Tape & Reel: 3.0K pcs per Reel |
| AT8A513HS8 | SOP | 8 | 150 mil | Tape & Reel: 2.5K pcs per Reel<br>Tube: 100 pcs per Tube |