深圳市矽源特科技有限公司

ShenZhen ChipSourceTek Technology Co. ,Ltd.

DATA SHEET

NY8A050D

6 I/O 8-bit EPROM-Based MCU

Version 1.2

Jul. 7, 2021

Revision History

Version	Date	Description	Modified Page
1.0	2019/05/03	Formal release.	-
1.1	2020/03/23	Modify DAA instruction.	46
1.2	2021/07/07	Modified address mapping “Mapped to bank0” to be “Unused”. Minor correction.	14 17,19,56

Table of Contents

1. 概述

NY8A050D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。NY8A050D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。因此非常適合各種低記憶容量但又複雜的應用。

在I/O的資源方面，NY8A050D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。

NY8A050D有一組計時器，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。

NY8A050D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，NY8A050D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NY8A050D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：
  - 2.0V ~ 5.5V @系統頻率≦8MHz。
  - 2.2V ~ 5.5V @系統頻率＞8MHz。
- 寬廣的工作温度：-40°C ~ 85°C。
- 512x14 bits EPROM。
- 32 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下拉電阻。
- PB[5:0]可選擇上拉電阻。
- PB[5:4]及PB[2:0]可選擇開漏極輸出(Open-Drain)。
- PB[3]可選擇當作輸入或開漏極輸出(Open-Drain)。
- 4 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。
- 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)。
- 三種硬體中斷：
  - Timer0 溢位中斷。
  - PB 輸入狀態改變中斷。
  - 外部中斷輸入。
- NY8A050D在待機模式(Standby mode)下的三種喚醒中斷：
  - Timer0 溢位中斷。
  - PB 輸入狀態改變中斷。
  - 外部中斷輸入。
- NY8A050D在睡眠模式(Halt mode)下的二種喚醒中斷：
  - PB 輸入狀態改變中斷。
  - 外部中斷輸入。

1. General Description

NY8A050D 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. NY8A050D adopts advanced CMOS technology to provide customers remarkable solution with low cost and high performance benefits. RISC architecture is applied to NY8A050D 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, NY8A050D is very suitable for those applications that are sophisticated but compact program size is required.

As NY8A050D 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.

NY8A050D also provides 1 set of timer which can be used as regular timer based on system oscillation or event counter with external trigger clock.

NY8A050D 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, NY8A050D 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.

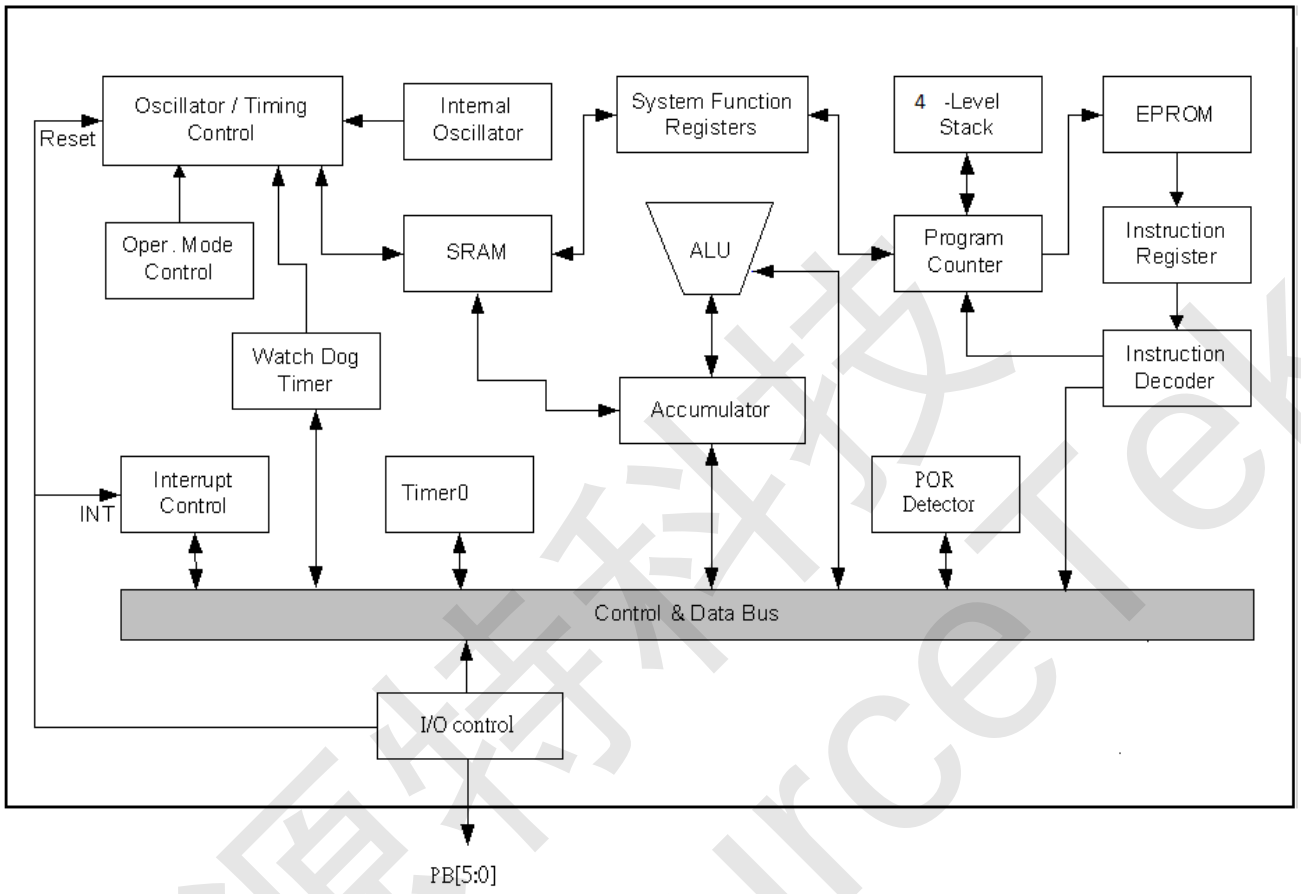
While NY8A050D operates in Standby mode and Halt mode, kinds of event will issue interrupt requests and can wake-up NY8A050D to enter Normal mode and Slow mode in order to process urgent events.

1.1 Features

- Wide operating voltage range:
  - 2.0V ~ 5.5V @system clock ≦8MHz.
  - 2.2V ~ 5.5V @system clock ＞8MHz.
- Wide operating temperature: -40°C ~ 85°C.
- 512 x 14 bits EPROM.
- 32 bytes SRAM.
- 6 general purpose I/O pins (GPIO), PB[5:0], with independent direction control.
- PB[3:0] have features of Pull-Low resistor for input pin.
- PB[5:0] have features of Pull-High resistor for input pin.
- PB[5:4] and PB[2:0] have features of open-drain output.
- PB[3] have feature of input or open-drain output.
- 4 level hardware Stack.
- Direct and indirect addressing modes for data access.

- One 8-bit up-count timer (Timer0) with programmable prescaler.
- 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.
- Three hardware interrupt events:
  - Timer0 overflow interrupt.
  - PB input change interrupt.
  - External interrupt.
- Three interrupt events to wake-up NY8A050D from Standby mode:
  - Timer0 overflow interrupt.
  - PB input change interrupt.
  - External interrupt.
- Two interrupt events to wake-up NY8A050D from Halt mode:
  - PB input change interrupt.
  - External interrupt.

1.2 Block Diagram



1.3 Pin Assignment

NY8A050D provides three kinds of package type which are SOP8 and SOT23-6.

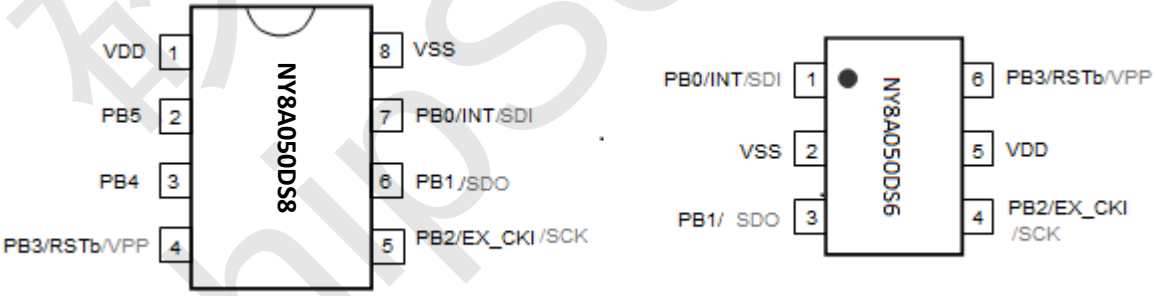


Figure 1 Package pin assignment

1.4 Pin Description

Pin Name	I/O	Description
PB0/ INT/ SDI	I/O	PB0 is a bidirectional I/O pin. PB0 is input pin of external interrupt when EIS=1 & INTIE=1. PB0 can be programming pad SDI.
PB1/ SDO	I/O	PB1 is a bidirectional I/O pin. PB1 can be programming pad SDO.
PB2 / EX_CKI / SCK	I/O	PB2 is a bidirectional I/O pin. It can also be timer clock source EX_CKI. PB2 can be programming pad SCK.
PB3/ RSTb/ VPP	I/O	PB3 is an input pin or open-drain output pin. It can be reset pin RSTb. If RSTb pin is low, it will reset NY8A050D. It can be programming pad VPP.
PB4	I/O	PB4 is a bidirectional I/O pin. 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

NY8A050D 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 NY8A050D is 512 words. Therefore, the Program Counter (PC) is 9 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.

NY8A050D provides instruction CALL, GOTOA, CALLA to address 256 location of program space. NY8A050D provides instruction GOTO to address 512 location of program space. NY8A050D 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.

NY8A050D program ROM address 0x1FE~0x1FF are reserved space, if user tries to write code in these addresses will get unexpected false functions.

NY8A050D program ROM address 0x00E~0x00F are preset rolling code can be released and used as normal program space.

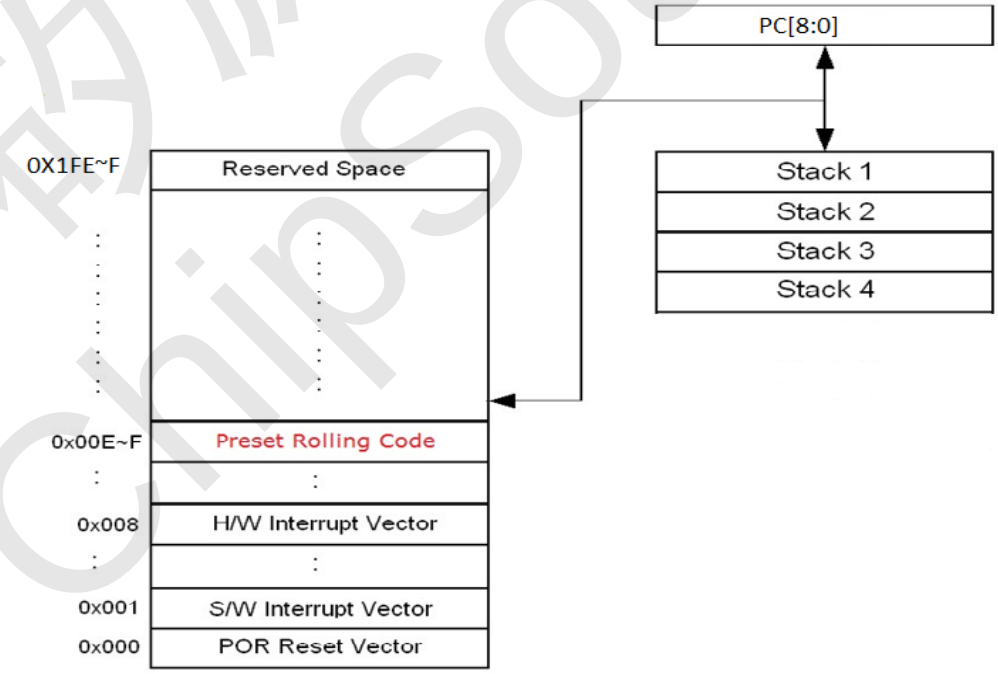


Figure 2 Program Memory Address Mapping

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].

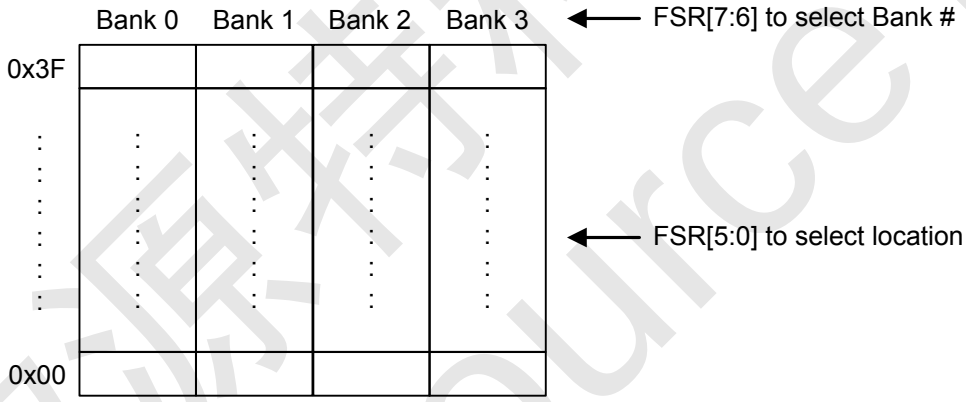


Figure 3 Indirect Addressing Mode of Data Memory Access

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.

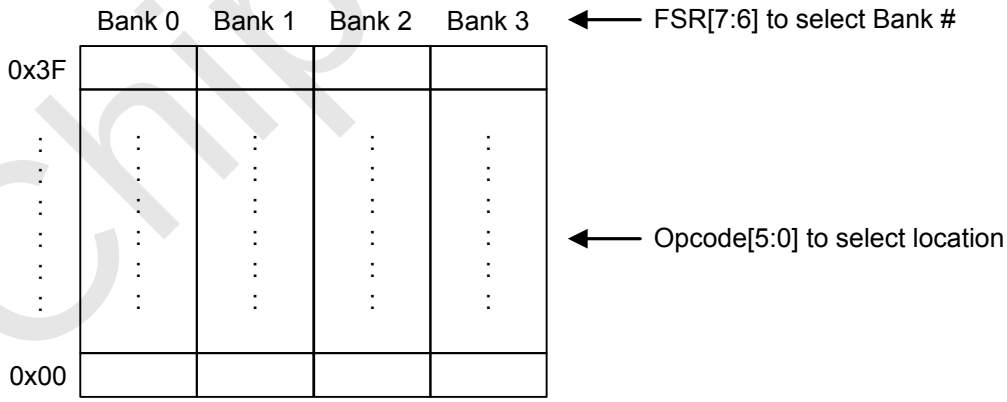


Figure 4 Direct Addressing Mode of Data Memory Access

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 0x2F of Bank and other banks in address from 0x10 to 0x2F are mapped back as the Table 1 shows.

The NY8A050D register name and address mapping of R-page SFR are described in the following table.

FSR[7:6] / Address	00 (Bank 0)	01 (Bank 1)	10 (Bank 2)	11 (Bank 3)
0x0	INDF	The same mapping as Bank 0		
0x1	TMR0			
0x2	PCL			
0x3	STATUS			
0x4	FSR			
0x5	-			
0x6	PORTB			
0x7	-			
0x8	PCON			
0x9	BWUCON			
0xA	PCHBUF			
0xB	BPLCON			
0xC	BPHCON			
0xD	-			
0xE	INTE			
0xF	INTF			
0x10 ~ 0x1F	General Purpose Register	Unused		
0x20 ~ 0x2F	General Purpose Register	Unused		

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	-	-
0x1	-	-
0x2	-	-
0x3	-	-
0x4	-	-
0x5	-	-
0x6	IOSTB	-
0x7	-	TBHP
0x8	-	TBHD
0x9	-	-
0xA	PS0CV	-
0xB	-	-
0xC	BODCON	-
0xD	-	-
0xE	-	-
0xF	PCON1	OSCCR

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

3. Function Description

This chapter will describe the detailed operations of NY8A050D.

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[8:0])

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 9-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[8], is not directly accessible. Update of PC[8] must be done through register PCHBUF.

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

For LCALL instruction, PC[8:0] 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 NY8A050D, bank register is not used, because there's only one bank in NY8A050D.

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	-	-	LVREN	-	-	-
R/W Property			R/W	R/W	-	-	R/W	-	-	-
Initial Value			1	0	X	X	1	X	X	X

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.

WDTEN: Enable/disable WDT.

WDTEN=1, enable WDT.

WDTEN=0, disable WDT.

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, $0 \le x \le 5$.

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	-	-	-	-	-	-	-	PCHBUF[0]
R/W Property			-	-	-	-	-	-	-	W
Initial Value			X	X	X	X	X	X	X	0

PCHBUF[0]: Buffer of the 8th bit of PC.

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	1	1	1	1

/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			0	0	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	-	-	-	-	-	INTIE	PBIE	T0IE
R/W Property			-	-	-	-	-	R/W	R/W	R/W
Initial Value			X	X	X	X	X	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.

3.1.13 INTF (Interrupt Flag Register)

Name	SFR Type	Addr.	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
INTF	R	0xF	-	-	-	-	-	INTIF	PBIF	T0IF
R/W Property			-	-	-	-	-	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.

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)
000	1:2	1:1
001	1:4	1:2
010	1:8	1:4
011	1:16	1:8
100	1:32	1:16
101	1:64	1:32
110	1:128	1:64
111	1:256	1:128

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 reset or timer interrupt, otherwise 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 Low Oscillator Frequency (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, Low Oscillator Frequency (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	-	-	IOPB5	IOPB4	IOPB3	IOPB2	IOPB1	IOPB0
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

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

IOPBx=1, PBx is input mode.

IOPBx=0, PBx is output mode.

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	-	ODPB2	ODPB1	ODPB0
R/W Property			-	-	R/W	R/W	-	R/W	R/W	R/W
Initial Value			X	X	0	0	X	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 PCON1 (Power Control Register1)

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

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

3.4 S-page Special Function Register

3.4.1 TBHP (Table Access High Byte Address Pointer Register)

Name	SFR Type	Addr.	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TBHP	S	0x7	-	-	-	-	-	-	-	TBHP0
R/W Property			-	-	-	-	-	-	-	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[0] and ACC. ACC is the Low Byte of PC[8:0] and TBHP[0] is the high byte of PC[8:0].

3.4.2 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.

3.4.3 OSCCR (Oscillation Control Register)

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

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

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

Table 4 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

NY8A050D 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. (Except PB[3], which is always in open-drain mode when configured as output port.)

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

Feature		PB[2:0]	PB[3]	PB[5:4]
Input	Pull-High Resistor	V	V	V
	Pull-Low Resistor	V	V	X
Output	Open-Drain	V	always	V

Table 5 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 NY8A050D. 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.

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 NY8A050D to enter reset process.

When NY8A050D 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.

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: 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.

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.

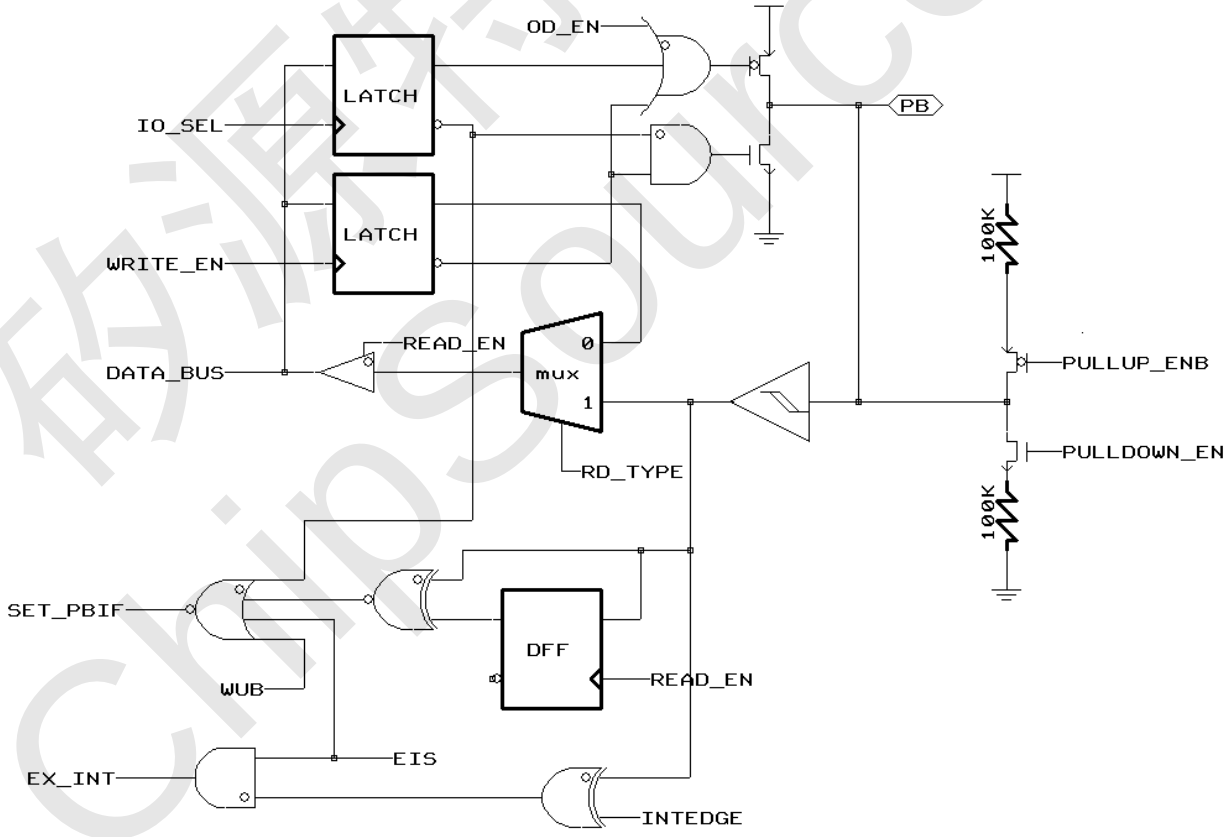


Figure 5 Block Diagram of PB0

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.

WUB: port B wake-up enable.

SET_PBIF: port B wake-up flag.

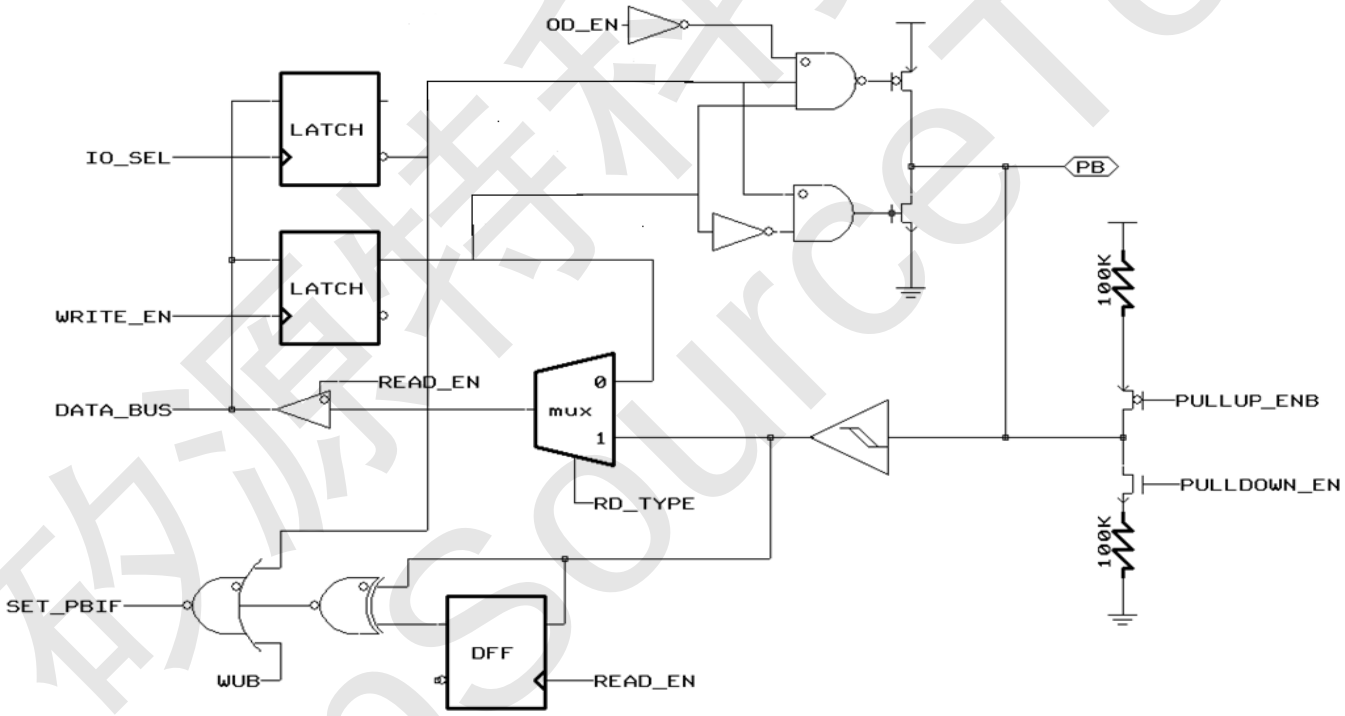


Figure 6 Block Diagram of PB1

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.

WUB: port B wake-up enable.

SET_PBIF: port B wake-up flag.

EX CKI: external clock input.

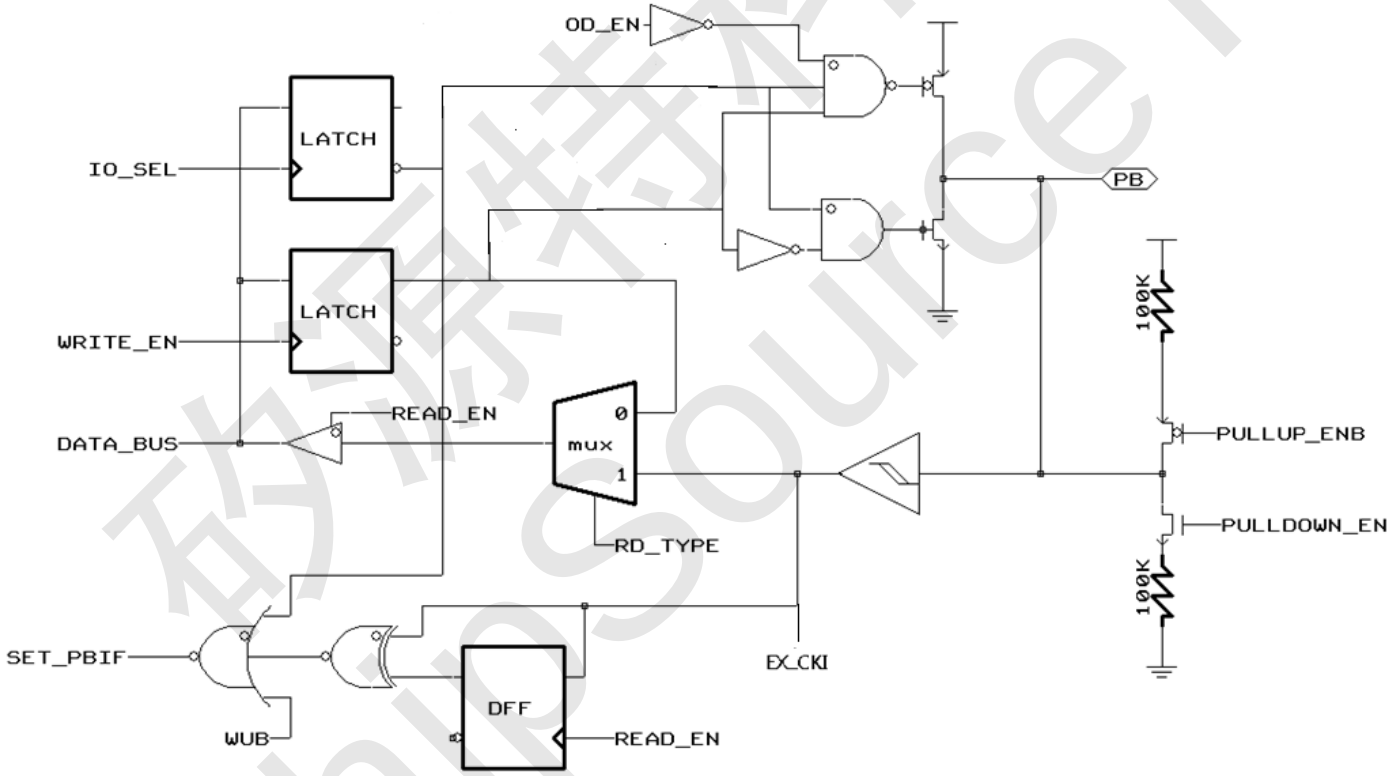


Figure 7 Block Diagram of PB2

IO_SEL: set pad attribute as input or output.

WRITE_EN: write data to pad.

READ_EN: read pad.

RSTPAD_EN: reset pad enable.

RSTB_IN: reset pad input.

PULLUP_ENB: enable Pull-High.

PULLDOWN_EN: enable Pull-Low.

RD_TYPE: select read pin or read latch.

WUB: port B wake-up enable.

SET_PBIF: port B wake-up flag.

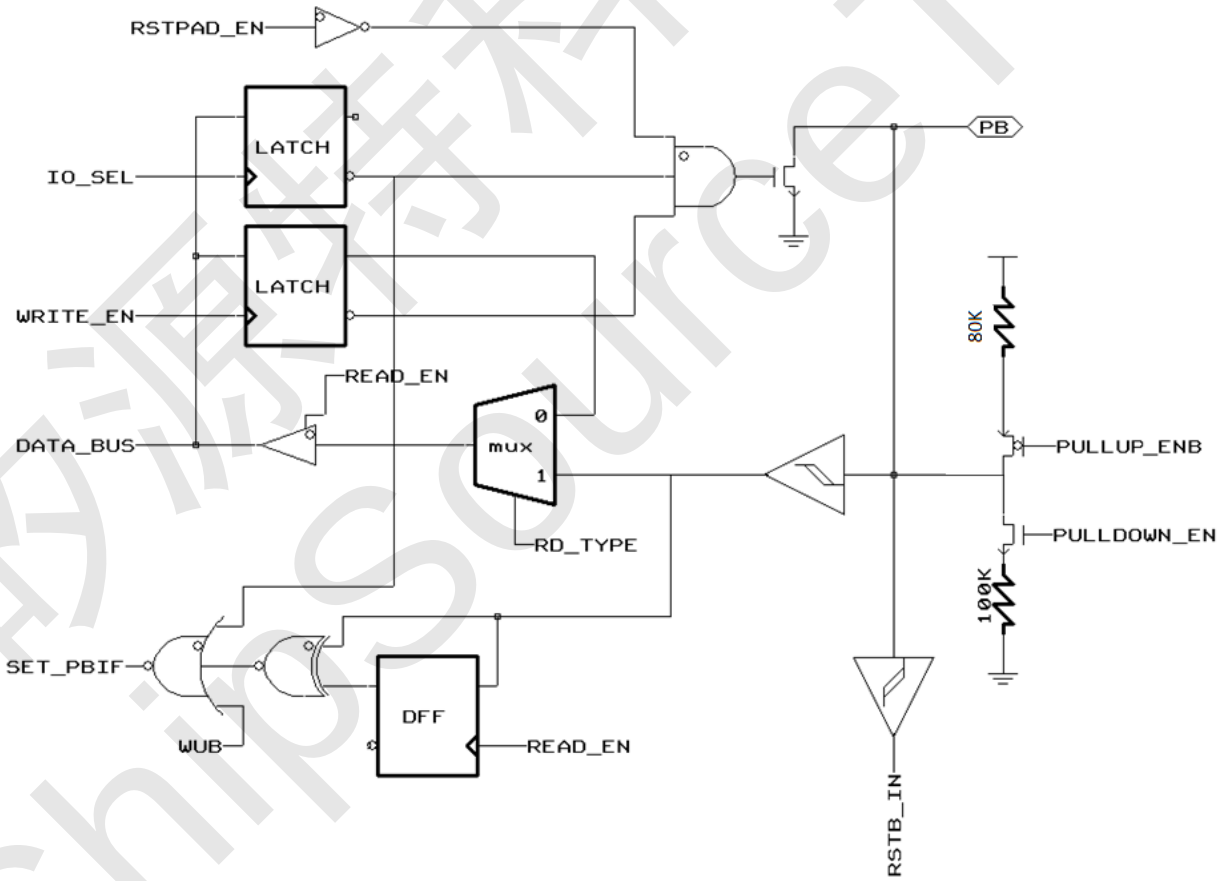


Figure 8 Block Diagram of PB3

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.

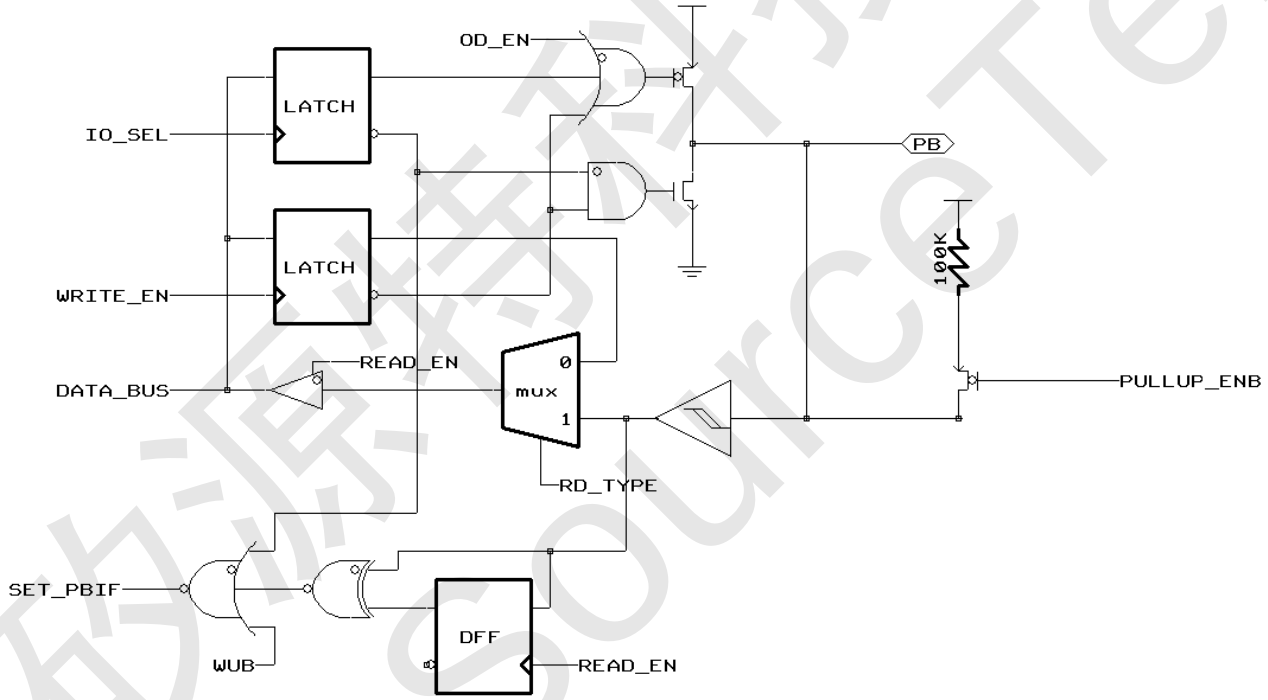


Figure 9 Block Diagram of PB4/PB5

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 6 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. 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, NY8A050D must assign prescaler0 to Timer0 and the prescaler0 dividing ratio must be no less than 4.

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.

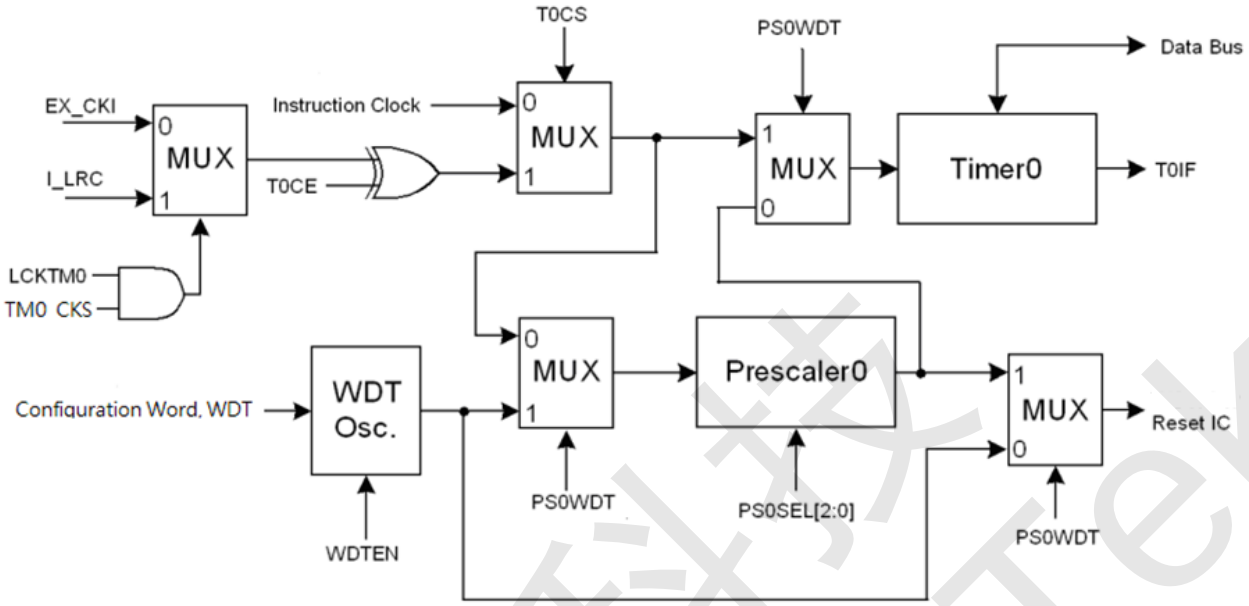


Figure 10 Block Diagram of Timer0 and WDT

3.7 Watch-Dog Timer (WDT)

There is an on-chip free-running oscillator in NY8A050D 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 will reset NY8A050D. 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 NY8A050D.

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

3.10 Interrupt

NY8A050D 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 3 hardware interrupts:

- Timer0 overflow interrupt.
- PB input change interrupt.
- External 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 NY8A050D 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 NY8A050D 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 NY8A050D 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.

3.10.1 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.

3.10.2 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.

3.10.3 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 Oscillation Configuration

Because NY8A050D is a dual-clock IC, there are high oscillation (F_{HOSC}) and low oscillation (F_{LOSC}) which can be selected as system oscillation (F_{OSC}). The oscillators which could be used as F_{HOSC} are internal high RC oscillator (I_HRC). The oscillators which could be used as F_{LOSC} are internal low RC oscillator (I_LRC).

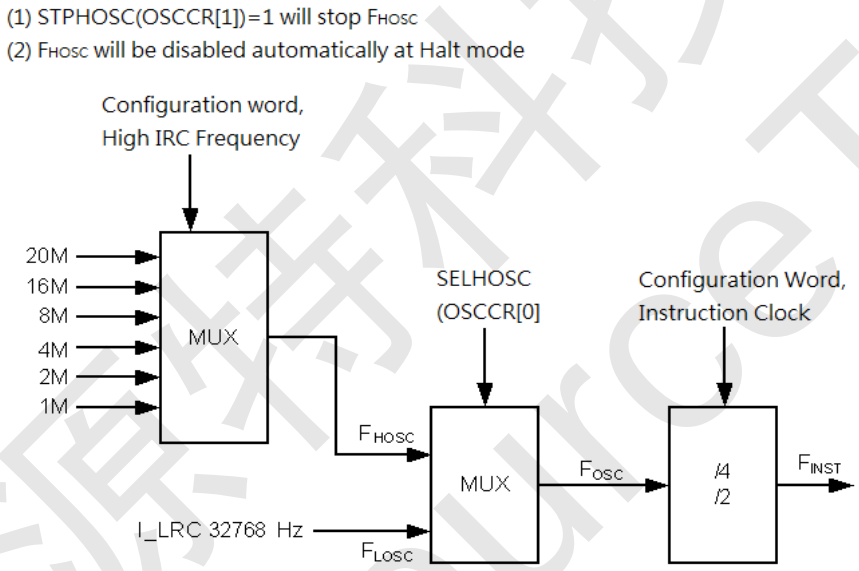


Figure 11 Oscillation Configuration of NY8A050D

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.

3.12 Operating Mode

NY8A050D 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, NY8A050D will stop almost all operations except Timer0 in order to wake-up periodically. At Halt mode, NY8A050D will sleep until external event to wake-up.

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

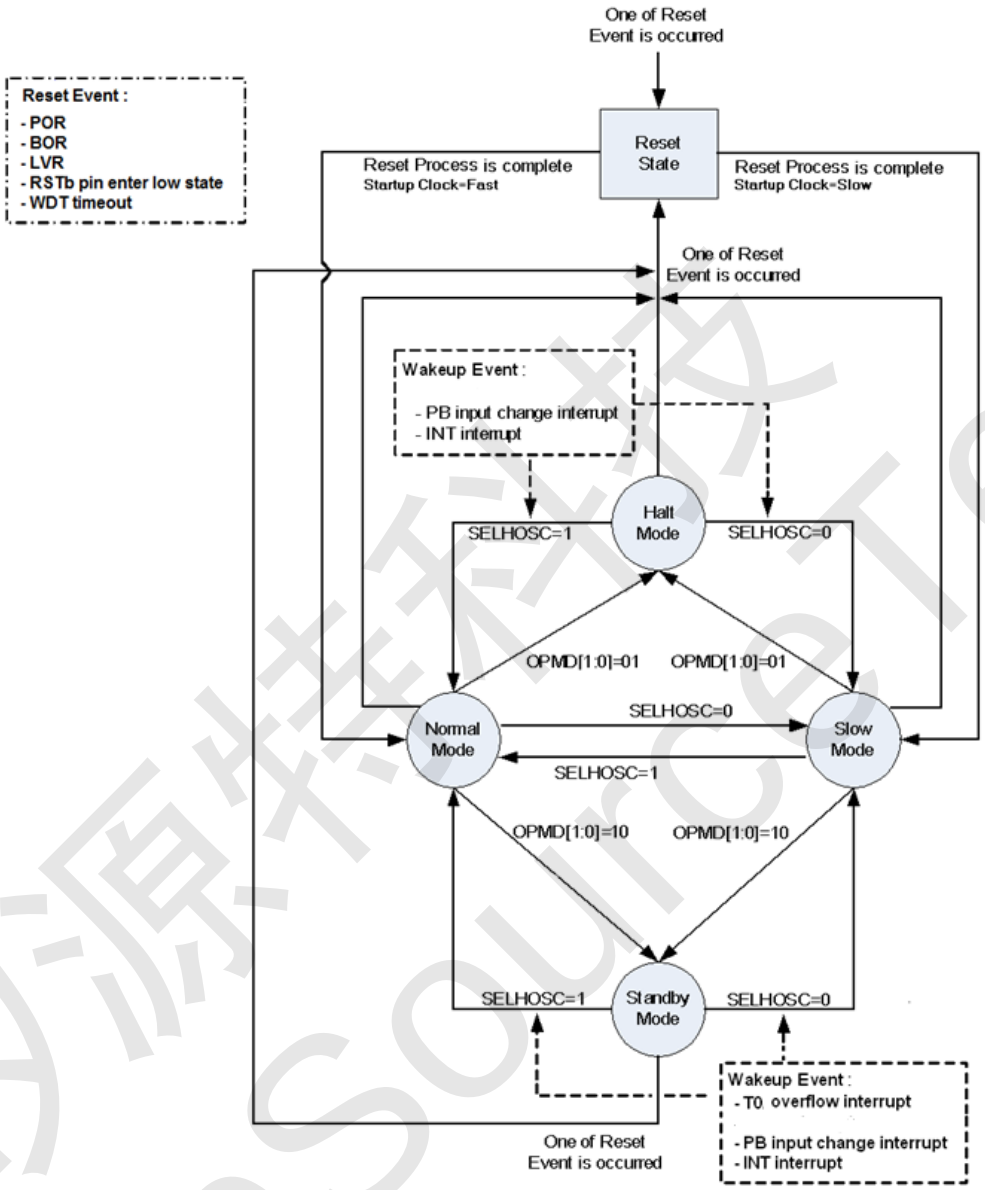


Figure 12 Four Operating Modes

3.12.1 Normal Mode

After any Reset Event is occurred and Reset Process is complete, NY8A050D 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, NY8A050D will enter Normal mode, if Startup Clock= I_LRC, NY8A050D 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, NY8A050D 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 NY8A050D 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.

3.12.2 Slow Mode

NY8A050D 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 NY8A050D. 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.

3.12.3 Standby Mode

NY8A050D 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 NY8A050D and user has to enter slow mode and write 1 to register bit STPHOSC in order to stop F_{HOSC} oscillation. Most of NY8A050D peripheral modules are disabled but Timer can be still active if register bit T0EN is set to 1. Therefore NY8A050D can wake-up after Timer0 is expired. The expiration period is determined by the register TMR0, F_{INST} and other configurations for Timer0.

- Instruction execution is stop and some peripheral modules may be active according to corresponding module enable bit.
- F_{HOSC} can be disabled by writing 1 to register bit STPHOSC.
- The F_{LOSC} is still active and running.
- IC can wake-up from Standby mode if any of (a) Timer0 overflow (b) PB input change interrupt or (c) INT external interrupt is happened.
- 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.

3.12.4 Halt Mode

NY8A050D 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 NY8A050D can only wake-up by some specific events. Therefore, Halt mode is the most power saving mode provided by NY8A050D.

- Instruction execution is stop and all peripheral modules are disabled.
- F_{HOSC} and F_{LOSC} are both disabled automatically.
- IC can wake-up from Halt mode if any of (a) PB input change interrupt or (b) 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

3.12.5 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 NY8A050D enters Standby mode or Halt mode, user may execute instruction ENI. At this condition, NY8A050D 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.

3.12.6 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/~~4~~	T0EN	T0EN	T0EN	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 - PB input change - INT	- PB input change - INT

Table 7 Summary of Operating Modes

3.13 Reset Process

NY8A050D 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 8 Summary of /TO & /PD Value and its Associated Event

After Reset Event is released, NY8A050D 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 4.5ms, 18ms, 72ms or 288ms. After oscillator is stable, NY8A050D will wait further 16 clock cycles of F_{OSC} (oscillator start-up time, OST) and Reset Process is complete.

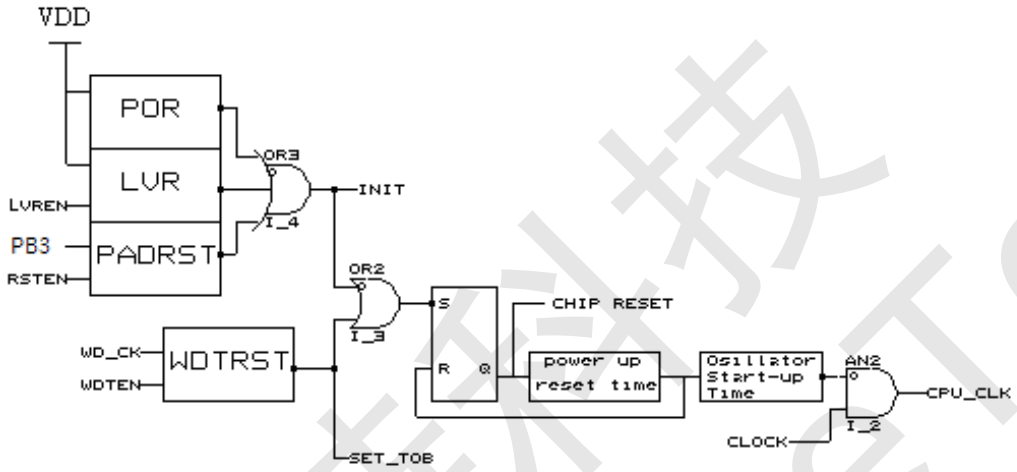


Figure 13 Block diagram of on-chip reset circuit

For slow V_{DD} 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.

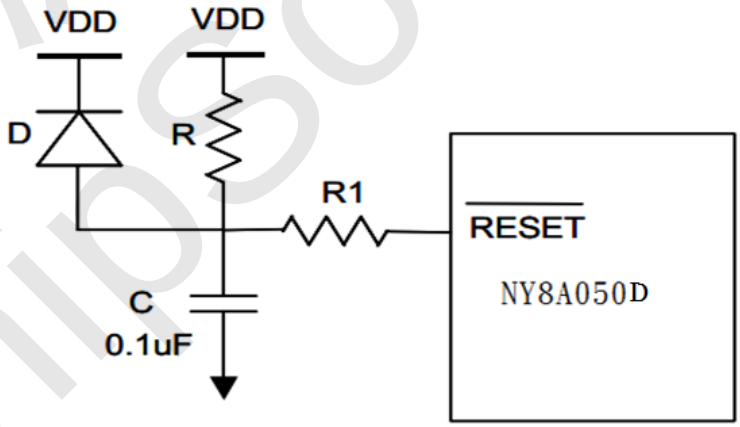


Figure 14 Block Diagram of Reset Application

4. Instruction Set

NY8A050D 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 9 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 0x6 ~ 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 ~0x2F.

S: S-page SFR, S is 0x7 ~ 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 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 before executing instruction: ACC=0x12, R=0x34, C=1, d=1. after executing instruction: R=0x47, ACC=0x12, C=0.

ADDAR	Add ACC and R
Syntax:	ADDAR R, d
Operand:	0 ≤ R ≤63 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 before executing instruction: ACC=0x12, R=0x34,C=1, d=1. after executing instruction: 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 before executing instruction: ACC=0x12, i=0x34, C=1. after executing instruction: 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 before executing instruction: ACC=0x12, i=0x34, C=1. after executing instruction: ACC=0x46, C=0.

ANDAR	AND ACC and R
Syntax:	ANDAR R, d
Operand:	0 ≤ R ≤ 63. 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 before executing instruction: ACC=0x5A, R=0xAF, d=1. after executing instruction: 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 before executing instruction: ACC=0x5A, i=0xAF. after executing instruction: ACC=0x0A, Z=0.

BCR	Clear Bit in R
Syntax:	BCR R, bit
Operand:	0 ≤ R ≤ 63 0 ≤ bit ≤ 7
Operation:	0 → R[bit]
Status affected:	--
Description:	Clear the bitth position in R.
Cycle:	1
Example:	BCR R,B2 before executing instruction: R=0x5A, B2=0x3. after executing instruction: R=0x52.

BSR	Set Bit in R
Syntax:	BSR R, bit
Operand:	0 ≤ R ≤ 63 0 ≤ bit ≤ 7
Operation:	1 → R[bit]
Status affected:	--
Description:	Set the bitth position in R.
Cycle:	1
Example:	BSR R,B2 before executing instruction: R=0x5A, B2=0x2. after executing instruction: R=0x5E.

BTRSC	Test Bit in R and Skip if Clear
Syntax:	BTRSC R, bit
Operand:	0 ≤ R ≤ 63 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 Instruction1 Instruction2 before executing instruction: R=0x5A, B2=0x2. after executing instruction: 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 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 Instruction2 Instruction3 before executing instruction: R=0x5A, B2=0x3. after executing instruction: 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 {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[0] is loaded into PC[8].
Cycle:	2
Example:	CALL SUB before executing instruction: PC=A0. Stack pointer=1 after executing instruction: PC=address of SUB, Stack[1] = A0+1, Stack pointer=2.

CALLA	Call Subroutine
Syntax:	CALLA
Operand:	--
Operation:	PC + 1 → Top of Stack {TBHP, ACC} → PC
Status affected:	--
Description:	The return address (PC + 1) is pushed onto top of Stack. The contents of TBHP[0] is loaded into PC[8] and ACC is loaded into PC[7:0].
Cycle:	2
Example:	CALLA before executing instruction: TBHP=0x01, ACC=0x34. PC=A0. Stack pointer=1. after executing instruction: PC=0x134, Stack[1]=A0+1, Stack pointer=2.

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

CLRWDT	Clear Watch-Dog Timer
Syntax:	CLRWDT
Operand:	--
Operation:	00h → WDT, 00h → WDT prescaler 1 → /TO 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 before executing instruction: /TO=0 after executing instruction: /TO=1

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

COMR	Complement R
Syntax:	COMR R, d
Operand:	0 ≤ R ≤ 63 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 before executing instruction: R=0xA6, d=1, Z=0. after executing instruction: R=0x59, Z=0.

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 before executing instruction: R=0x34, ACC=12, Z=0, C=0. after executing instruction: R=0x34, ACC=12, Z=0, C=1.

DECR	Decrease R
Syntax:	DECR R, d
Operand:	0 ≤ R ≤ 63 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 before executing instruction: R=0x01, d=1, Z=0. after executing instruction: R=0x00, Z=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 ADDAR R,d DAA ENI before executing instruction: ACC=0x28, R=0x25, d=0. after executing instruction: ACC=0x53, C=0.

DECRSZ	Decrease R, Skip if 0
Syntax:	DECRSZ R, d
Operand:	0 ≤ R ≤ 63 d = 0, 1.
Operation:	R - 1 → dest, 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. 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 instruction2 instruction3 before executing instruction: R=0x1, d=1, Z=0. after executing instruction: 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 before executing instruction: GIE=1. After executing instruction: 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 before executing instruction: GIE=0. After executing instruction: 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[0] is loaded into PC[8].
Cycle:	2
Example:	GOTO Level before executing instruction: PC=A0. after executing instruction: 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[0] is loaded into PC[8] and ACC is loaded into PC[7:0].
Cycle:	2
Example:	GOTOA before executing instruction: PC=A0. TBHP=0x01, ACC=0x34. after executing instruction: PC=0x134.

INCR	Increase R
Syntax:	INCR R, d
Operand:	0 ≤ R ≤ 63 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 before executing instruction: R=0xFF, d=1, Z=0. after executing instruction: R=0x00, Z=1.

INT	Software Interrupt
Syntax:	INT
Operand:	--
Operation:	PC + 1 → Top of Stack, 001h → PC
Status affected:	--
Description:	Software interrupt. First, return address (PC + 1) is pushed onto the Stack. The address 0x001 is loaded into PC[8:0].
Cycle:	3
Example:	INT before executing instruction: PC=address of INT code. after executing instruction: PC=0x01.

INCRSZ	Increase R, Skip if 0
Syntax:	INCRSZ R, d
Operand:	0 ≤ R ≤ 63 d = 0, 1.
Operation:	R + 1 → dest, 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. 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 instruction2, instruction3. before executing instruction: R=0xFF, d=1, Z=0. after executing instruction: R=0x00, Z=1. And the program will skip instruction2 execution because the operation result is zero.

IORAR	OR ACC with R
Syntax:	IORAR R, d
Operand:	0 ≤ R ≤ 63 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 before executing instruction: R=0x50, ACC=0xAA, d=1, Z=0. after executing instruction: 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 before executing instruction: i=0x50, ACC=0xAA, Z=0. after executing instruction: ACC=0xFA, Z=0.

IOSTR	Move F-page SFR to ACC
Syntax:	IOSTR F
Operand:	6 ≤ F ≤ 15
Operation:	F-page SFR → ACC
Status affected:	--
Description:	Move F-page SFR F to ACC.
Cycle:	1
Example:	IOSTR F before executing instruction: F=0x55, ACC=0xAA. after executing instruction: F=0x55, ACC=0x55.

IOST	Load F-page SFR from ACC
Syntax:	IOST F
Operand:	6 ≤ 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 before executing instruction: F=0x55, ACC=0xAA. after executing instruction: F=0xAA, ACC=0xAA.

LCALL	Call Subroutine
Syntax:	LCALL adr
Operand:	0 ≤ adr ≤ 511
Operation:	PC + 1 → Top of Stack, adr → PC[8:0]
Status affected:	--
Description:	The return address (PC + 1) is pushed onto top of Stack. The 9-bit immediate address adr is loaded into PC[8:0].
Cycle:	2
Example:	LCALL SUB before executing instruction: PC=A0. Stack level=1 after executing instruction: PC=address of SUB, Stack[1]= A0+1, Stack pointer =2.

LGOTO	Unconditional Branch
Syntax:	LGOTO adr
Operand:	0 ≤ adr ≤ 511
Operation:	adr → PC[8:0].
Status affected:	--
Description:	LGOTO is an unconditional branch instruction. The 9-bit immediate address adr is loaded into PC[8:0].
Cycle:	2
Example:	LGOTO Level before executing instruction: PC=A0. after executing instruction: 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 before executing instruction: i=0x55, ACC=0xAA. after executing instruction: 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 before executing instruction: R=0x55, ACC=0xAA. after executing instruction: R=0xAA, ACC=0xAA.

MOVR	Move to ACC or R
Syntax:	MOVR R, d
Operand:	0 ≤ R ≤ 63 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 before executing instruction: R=0x0, ACC=0xAA, Z=0, d=0. after executing instruction: R=0x0, ACC=0x00, Z=1.

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

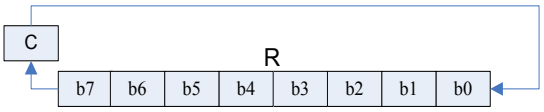
RETIE	Return from Interrupt and Enable Interrupt Globally
Syntax:	RETIE
Operand:	--
Operation:	Top of Stack → PC 1 → GIE
Status affected:	--
Description:	The PC is loaded from top of Stack as return address and GIE is set to 1.
Cycle:	2
Example:	RETIE before executing instruction: Stack level=2. after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

RETIA	Return with Data in ACC

RETIA	Return with Data in ACC
Syntax:	RETIA i
Operand:	0 ≤ i < 255
Operation:	i → ACC, 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 before executing instruction: GIE=0, Stack pointer =2, i=0x55, ACC=0xAA. after executing instruction: GIE=1, PC=Stack[2], Stack pointer =1, ACC=0x55.

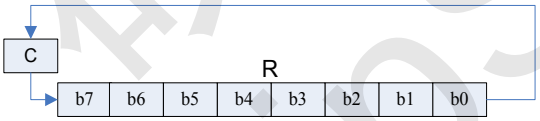
RET	Return from Subroutine
Syntax:	RET
Operand:	--
Operation:	Top of Stack → PC
Status affected:	--
Description:	PC is loaded from top of Stack as return address.
Cycle:	2
Example:	RET before executing instruction: Stack level=2. after executing instruction: PC=Stack[2], Stack level=1.

RLR	Rotate Left R Through Carry
Syntax:	RLR R, d
Operand:	0 ≤ R ≤ 63 d = 0, 1.
Operation:	C → dest[0], R[7] → C, 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 before executing instruction: R=0xA5, d=1, C=0. after executing instruction: R=0x4A, C=1.

RRR	Rotate Right R Through Carry
Syntax:	RRR R, d
Operand:	0 ≤ R ≤ 63 d = 0, 1.
Operation:	C → dest[7], R[7:1] → dest[6:0], 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 before executing instruction: R=0xA5, d=1, C=0. after executing instruction: R=0x52, C=1.

SBCAR	Subtract ACC and Carry from R
Syntax:	SBCAR R, d
Operand:	0 ≤ R ≤ 63 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 (a) before executing instruction: R=0x05, ACC=0x06, d=1, C=0. after executing instruction: R=0xFE, C=0. (-2) (b) before executing instruction: R=0x05, ACC=0x06, d=1, C=1. after executing instruction: R=0xFF, C=0. (-1) (c) before executing instruction: R=0x06, ACC=0x05, d=1, C=0. after executing instruction: R=0x00, C=1. (-0), Z=1. (d) before executing instruction: R=0x06, ACC=0x05, d=1, C=1. after executing instruction: 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 (a) before executing instruction: i=0x05, ACC=0x06, C=0. after executing instruction: ACC=0xFE, C=0. (-2) (b) before executing instruction: i=0x05, ACC=0x06, C=1. after executing instruction: ACC=0xFF, C=0. (-1) (c) before executing instruction: i=0x06, ACC=0x05, C=0. after executing instruction: ACC=0x00, C=1. (-0), Z=1. (d) before executing instruction: i=0x06, ACC=0x05, C=1. after executing instruction: ACC=0x1, C=1. (+1)

SFUN	Load S-page SFR from ACC
Syntax:	SFUN S
Operand:	7 ≤ 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 before executing instruction: S=0x55, ACC=0xAA. after executing instruction: S=0xAA, ACC=0xAA.

SFUNR	Move S-page SFR to ACC
Syntax:	SFUNR S
Operand:	7 ≤ S ≤ 15
Operation:	S-page SFR → ACC
Status affected:	--
Description:	Move S-page SFR S to ACC.
Cycle:	1
Example:	SFUNR S before executing instruction: S=0x55, ACC=0xAA. after executing instruction: S=0x55, ACC=0x55.

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

SUBAR	Subtract ACC from R
Syntax:	SUBAR R, d
Operand:	0 ≤ R ≤ 63 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 (a) before executing instruction: R=0x05, ACC=0x06, d=1. after executing instruction: R=0xFF, C=0. (-1) (b) before executing instruction: R=0x06, ACC=0x05, d=1. after executing instruction: 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 (a) before executing instruction: i=0x05, ACC=0x06. after executing instruction: ACC=0xFF, C=0. (-1) (b) before executing instruction: i=0x06, ACC=0x05, d=1. after executing instruction: ACC=0x01, C=1. (+1)

SWAPR	Swap High/Low Nibble in R
Syntax:	SWAPR R, d
Operand:	0 ≤ R ≤ 63 d = 0, 1.
Operation:	R[3:0] → dest[7:4]. 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 before executing instruction: R=0xA5, d=1. after executing instruction: R=0x5A.

TABLEA	Read ROM data
Syntax:	TABLEA
Operand:	--
Operation:	ROM data{ TBHP, ACC } [7:0] → ACC ROM data{TBHP, ACC} [13:8] → TBHD.
Status affected:	--
Description:	The 8 least significant bits of ROM pointed by {TBHP[0], ACC} is placed to ACC. The 6 most significant bits of ROM pointed by {TBHP[0], ACC} is placed to TBHD[5:0].
Cycle:	2
Example:	TABLEA before executing instruction: TBHP=0x01, ACC=0x34. TBHD=0x01. ROM data[0x134]= 0x35AA after executing instruction: 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 before executing instruction: T0MD=0x55, ACC=0xAA. after executing instruction: 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 before executing instruction T0MD=0x55, ACC=0xAA. after executing instruction ACC=0x55.

XORAR	Exclusive-OR ACC with R
Syntax:	XORAR R, d
Operand:	0 ≤ R ≤ 63 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 before executing instruction: R=0xA5, ACC=0xF0, d=1. after executing instruction: 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 before executing instruction: i=0xA5, ACC=0xF0. after executing instruction: ACC=0x55.

5. Configuration Words

Item	Name	Options
1	High IRC Frequency	1. 1MHz 2. 2MHz 3. 4MHz 4. 8MHz 5. 16MHz 6. 20MHz
2	Instruction Clock	1. 2 oscillator period 2. 4 oscillator period
3	WDT	1. Watchdog Enable (Software control) 2. Watchdog Disable (Always disable)
4	Timer0 source	1. EX_CKI 2. I_LRC
5	PB.3	1. PB.3 is I/O 2. PB.3 is reset
6	PB.4	1. PB.4 is I/O 2. PB.4 is instruction clock output
7	Startup Time	1. 4.5ms 2. 18ms 3. 72ms 4. 288ms
8	WDT Time Base	1. 3.5ms 2. 15ms 3. 60ms 4. 250ms
9	LVR Setting	1. Register Control 2. LVR Always On
10	LVR Voltage	1. 1.6V 2. 1.8V 3. 2.0V 4. 2.2V 5. 2.4V 6. 2.7V 7. 3.0V 8. 3.3V 9. 3.6V 10. 4.2V
11	VDD Voltage	1. 3.0V 2. 4.5V 3. 5.0V
12	Read Output Data	1. I/O port 2. Register
13	Startup Clock	1. I_HRC 2. I_LRC
14	Input High Voltage (VIH)	1. 0.7VDD 2. 0.5VDD
15	Input Low Voltage (VIL)	1. 0.3VDD 2. 0.2VDD
16	Input Voltage Schmitt Trigger	1. ENABLE (Check 14, 15) 2. Disable (14, 15 no USE)

Table 10 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

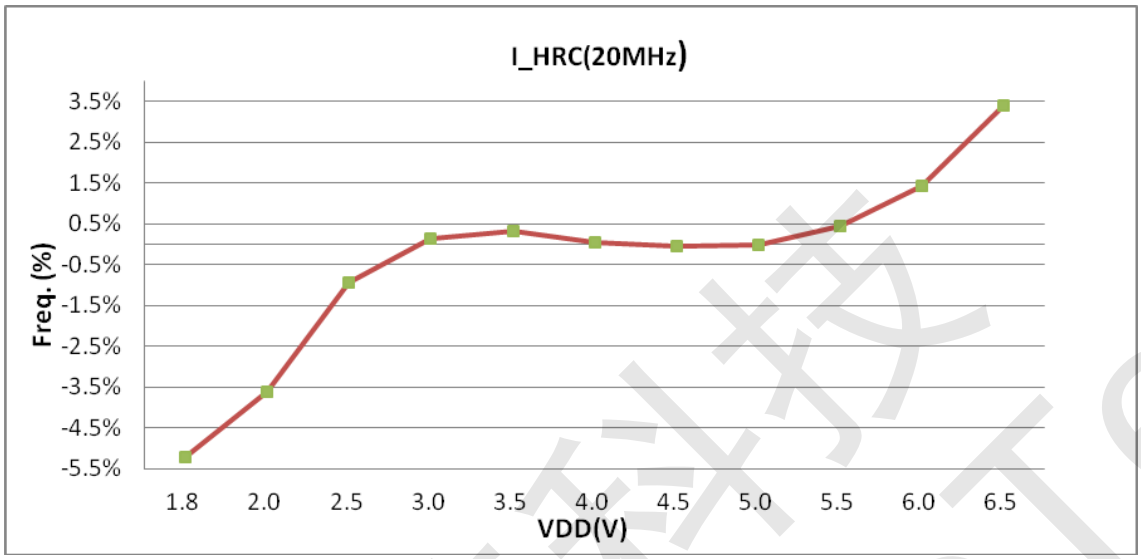
(All refer F_{INST}=F_{HOSC}/4, F_{HOSC}=16MHz@I_HRC, WDT enabled, ambient temperature T_A=25°C unless otherwise specified.)

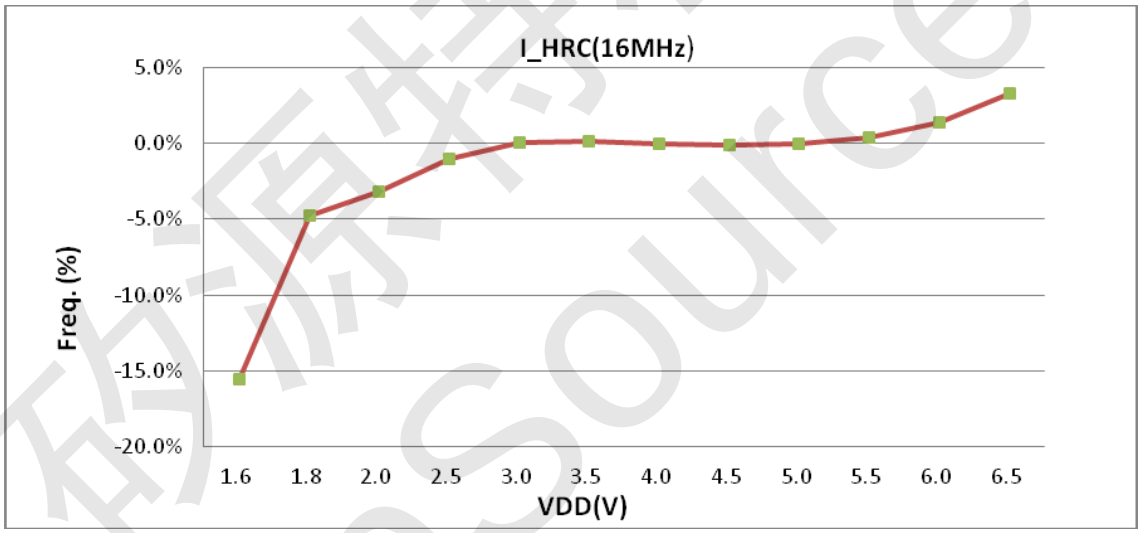
Symbol	Parameter	V_{DD}	Min.	Typ.	Max.	Unit	Condition
V_{DD}	Operating voltage	--	3.0	--	5.5	V	F_{INST}=20MHz @ I_HRC/2
			2.2				F_{INST}=20MHz @ I_HRC/4
			2.7				F_{INST}=16MHz @ I_HRC/2
			2.0				F_{INST}=16MHz @ I_HRC/4
			2.0				F_{INST}=8MHz @ I_HRC/2
			1.6				F_{INST}=8MHz @ I_HRC/4
			1.6				F_{INST}=4MHz @ I_HRC/4 & I_HRC/2
			1.6				F_{INST}=32KHz @ I_LRC/4 & I_LRC/2
V_{IH}	Input high voltage	5V	4.0	--	--	V	RSTb (0.8V_{DD})
		3V	2.4	--	--		
		5V	3.5	--	--	V	All other I/O pins, EX_CKI, INT CMOS (0.7V_{DD})
		3V	2.1	--	--		
		5V	2.5	--	--	V	All other I/O pins, EX_CKI, INT TTL (0.5V_{DD})
		3V	1.5	--	--		
		5V	2.5	--	--	V	All other I/O pins, EX_CKI, INT No Schmitt Trigger(0.5V_{DD})
		3V	1.5	--	--		
V_{IL}	Input low voltage	5V	--	--	1.0	V	RSTb (0.2V_{DD})
		3V	--	--	0.6		
		5V	--	--	1.5	V	All other I/O pins, EX_CKI, INT CMOS (0.3V_{DD})
		3V	--	--	0.9		
		5V	--	--	1.0	V	All other I/O pins, EX_CKI, INT TTL (0.2V_{DD})
		3V	--	--	0.6		
		5V	--	--	2.5	V	All other I/O pins, EX_CKI, INT No Schmitt Trigger(0.5V_{DD})
		3V	--	--	1.5		
I_{OH}	Output high current	5V	--	-18	--	mA	V_{OH}=4.0V
		3V	--	-10	--		V_{OH}=2.0V
I_{OL}	Output low current	5V	--	20	--	mA	V_{OL}=1.0V
		3V	--	12	--		
I_{OP}	Operating current	**Normal Mode**					
		5V	--	1.5	--	mA	F_{HOSC}=20MHz @ I_HRC/2

Symbol	Parameter	V_{DD}	Min.	Typ.	Max.	Unit	Condition
		3V	--	0.7	--		
		5V	--	1.3	--	mA	F_{HOSC}=20MHz @ I_HRC/4
		3V	--	0.5	--		
		5V	--	1.4	--	mA	F_{HOSC}=16MHz @ I_HRC/2
		3V	--	0.6	--		
		5V	--	1.2	--	mA	F_{HOSC}=16MHz @ I_HRC/4
		3V	--	0.4	--		
		5V	--	1.1	--	mA	F_{HOSC}=8MHz @ I_HRC/2
		3V	--	0.4	--		
		5V	--	0.8	--	mA	F_{HOSC}=8MHz @ I_HRC/4
		3V	--	0.3	--		
		5V	--	0.8	--	mA	F_{HOSC}=4MHz @ I_HRC/2
		3V	--	0.3	--		
		5V	--	0.6	--	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	--		
		5V	--	7.2	--	uA	F_{HOSC} disabled, F_{LOSC}=32KHz @ I_LRC/2
		3V	--	2.4	--		
		5V	--	4.4	--	uA	F_{HOSC} disabled, F_{LOSC}=32KHz @ I_LRC/4
		3V	--	1.4	--		
I_{STB}	Standby current	5V	--	1.8	--	uA	Standby mode, F_{HOSC} disabled, F_{LOSC}=32KHz @ I_LRC/4
		3V	--	0.4	--		
I_{HALT}	Halt current	5V	--	--	0.5	uA	Halt mode, WDT disabled.
		3V	--	--	0.2		
		5V	--	--	5	uA	Halt mode, WDT enabled.
		3V	--	--	2		
R_{PH}	Pull-High resistor	5V	--	60	--	kΩ	Pull-High resistor (not include PB3)
		3V	--	120	--		
		5V	--	85	--	kΩ	Pull-High resistor (PB3)
		3V	--	85	--		
R_{PL}	Pull-Low resistor	5V	--	60	--	kΩ	Pull-Low resistor
		3V	--	120	--		

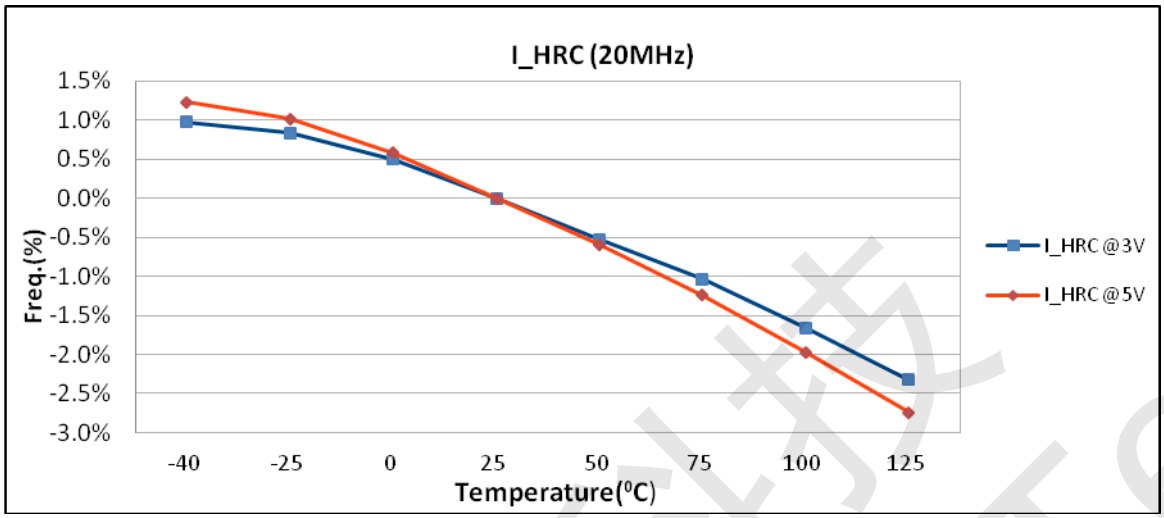
6.3 Characteristic Graph

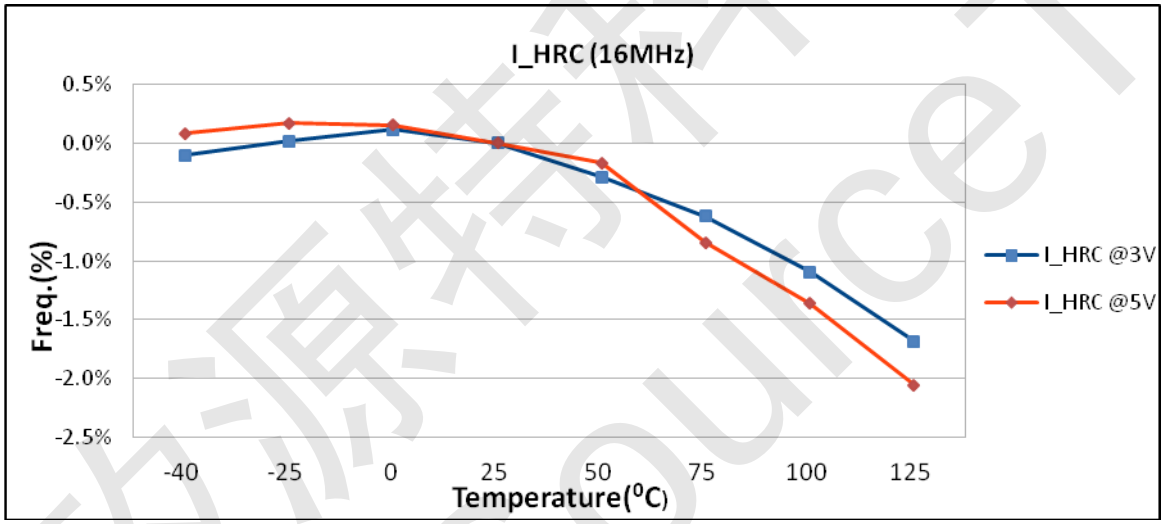
6.3.1 Frequency vs. V_{DD} of I_HRC



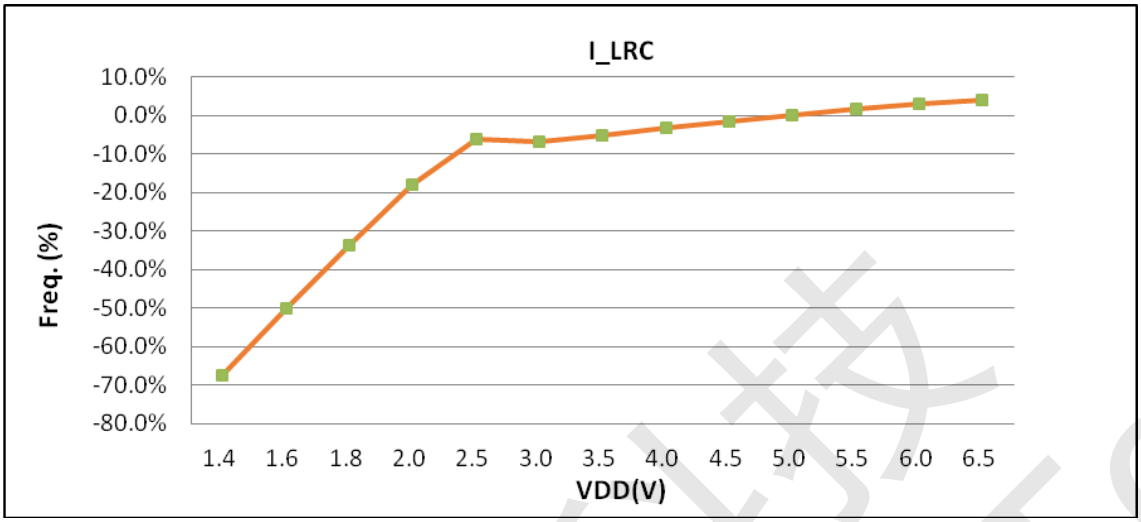


6.3.2 Frequency vs. Temperature of I_HRC

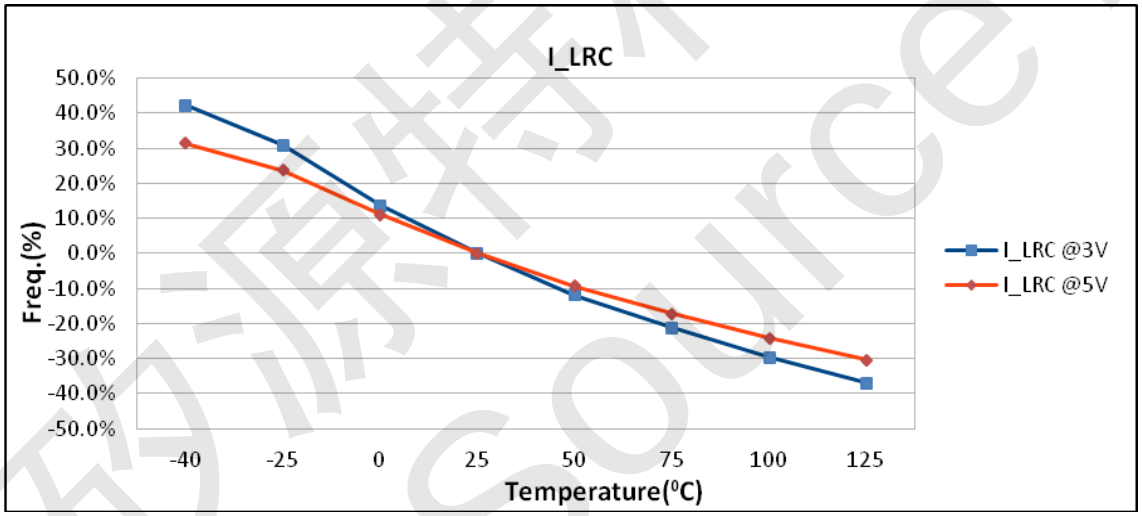




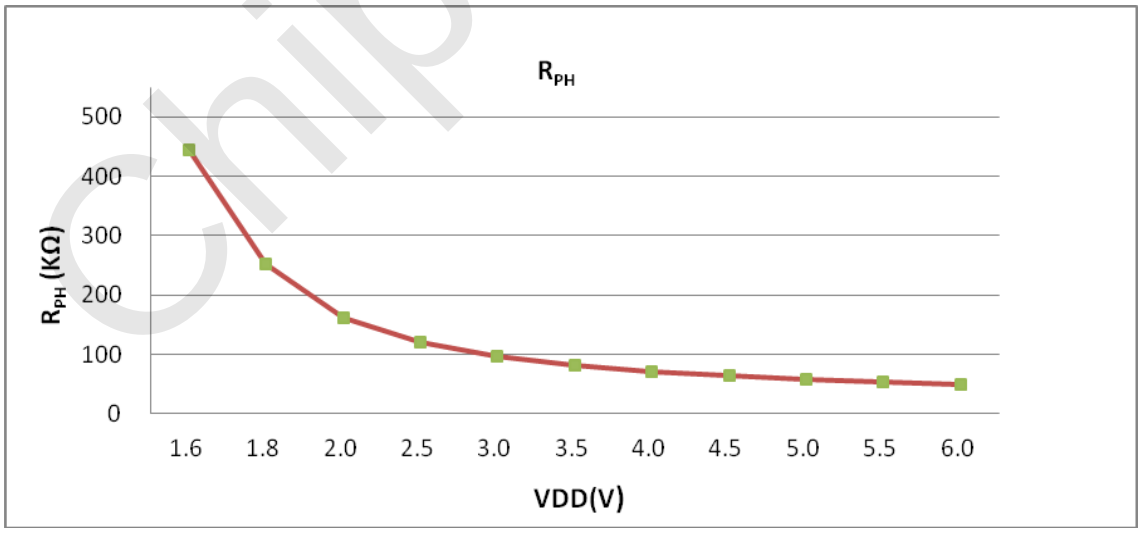
6.3.3 Frequency vs. V_{DD} of I_LRC



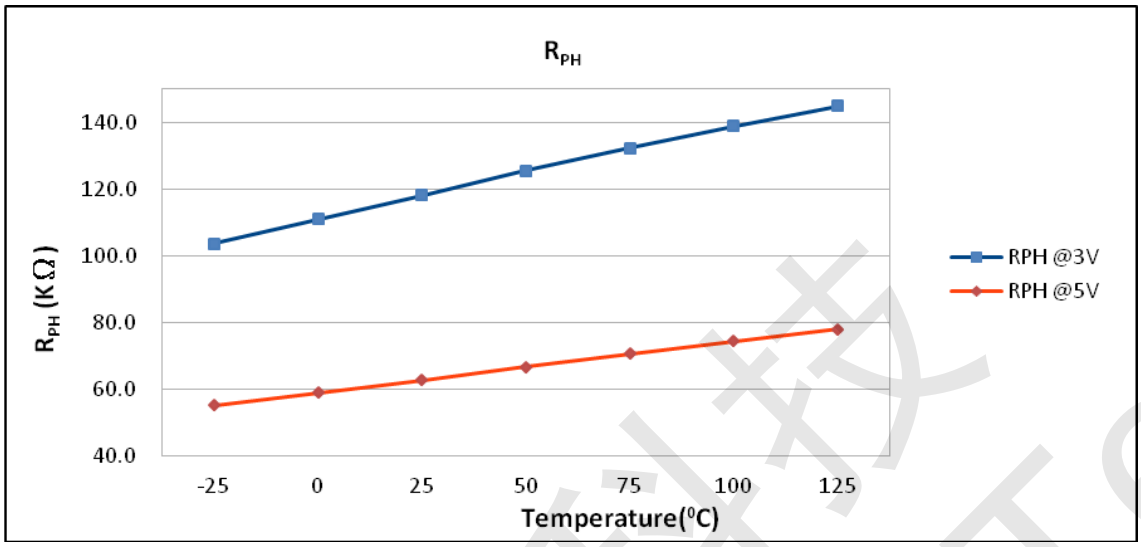
6.3.4 Frequency vs. Temperature of I_LRC



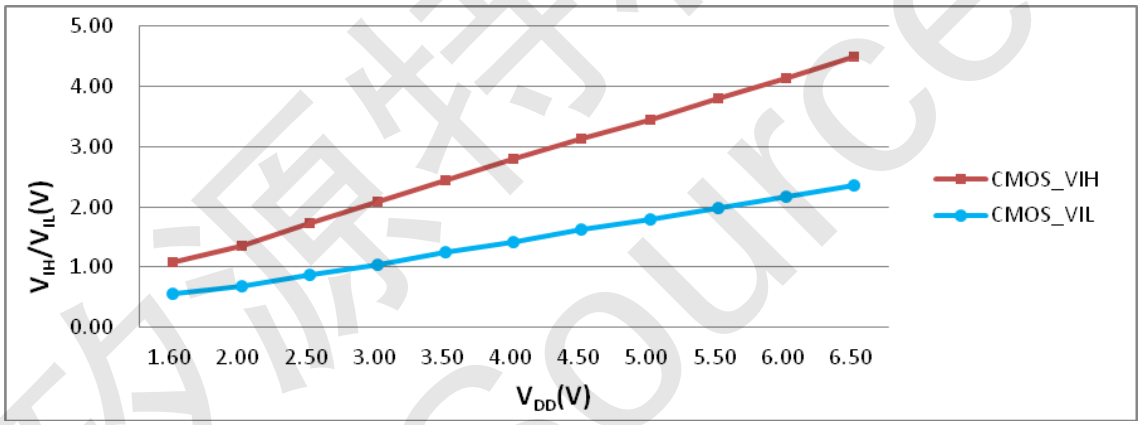
6.3.5 Pull High Resistor vs. VDD

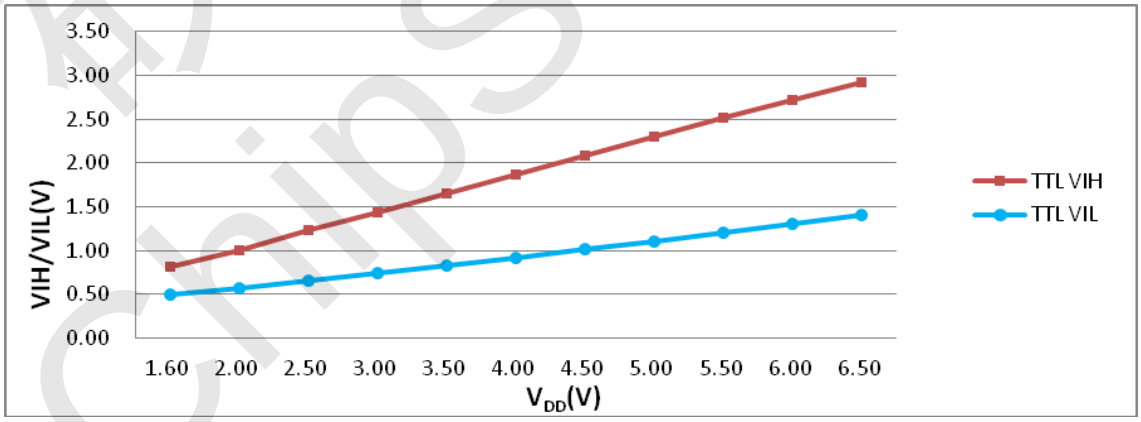


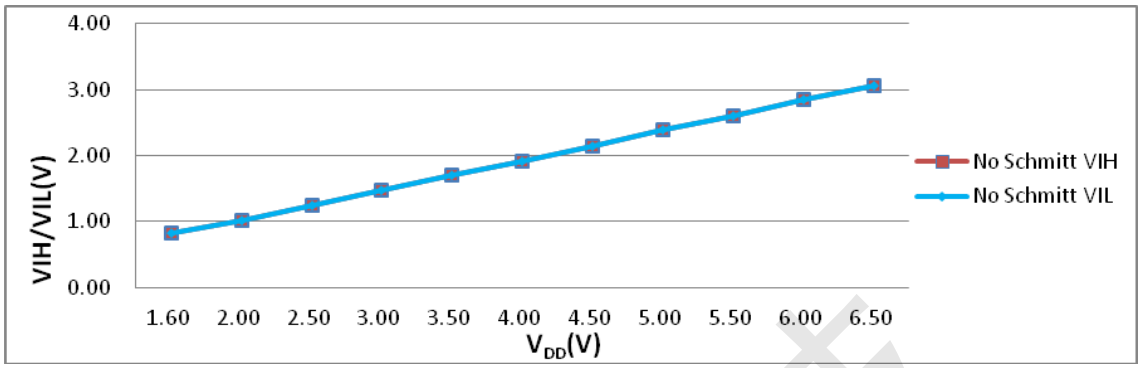
6.3.6 Pull High Resistor vs. Temperature

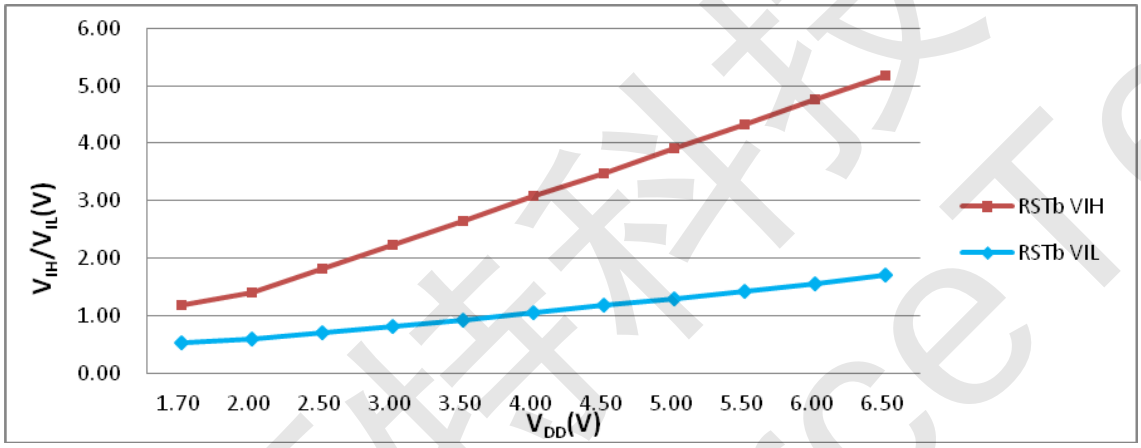


6.3.7 VIH/VIL vs. VDD

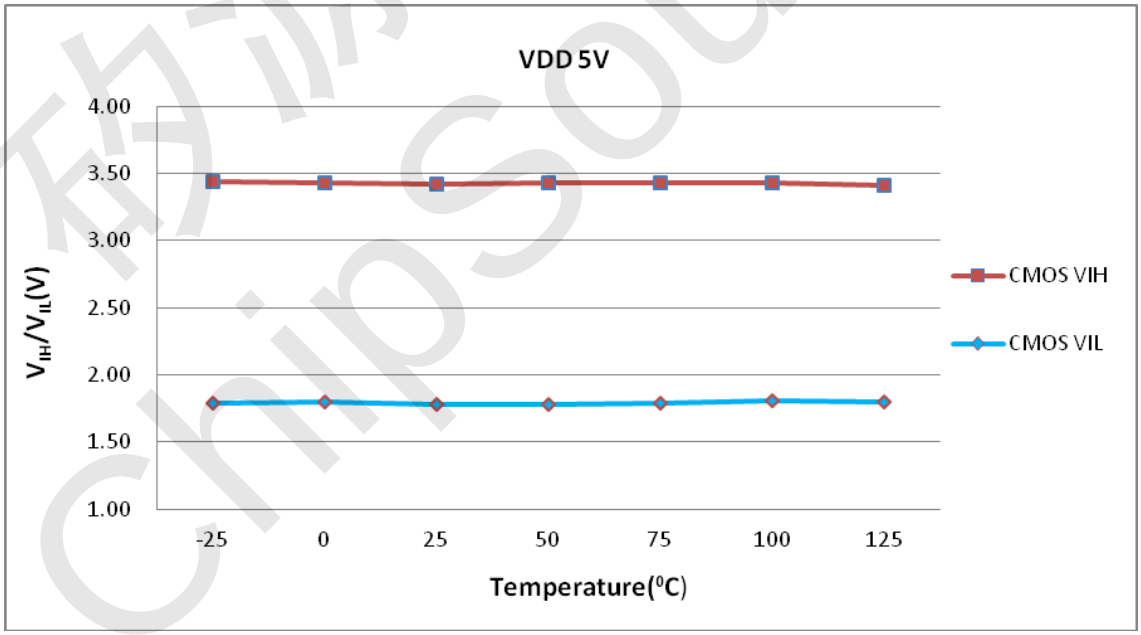


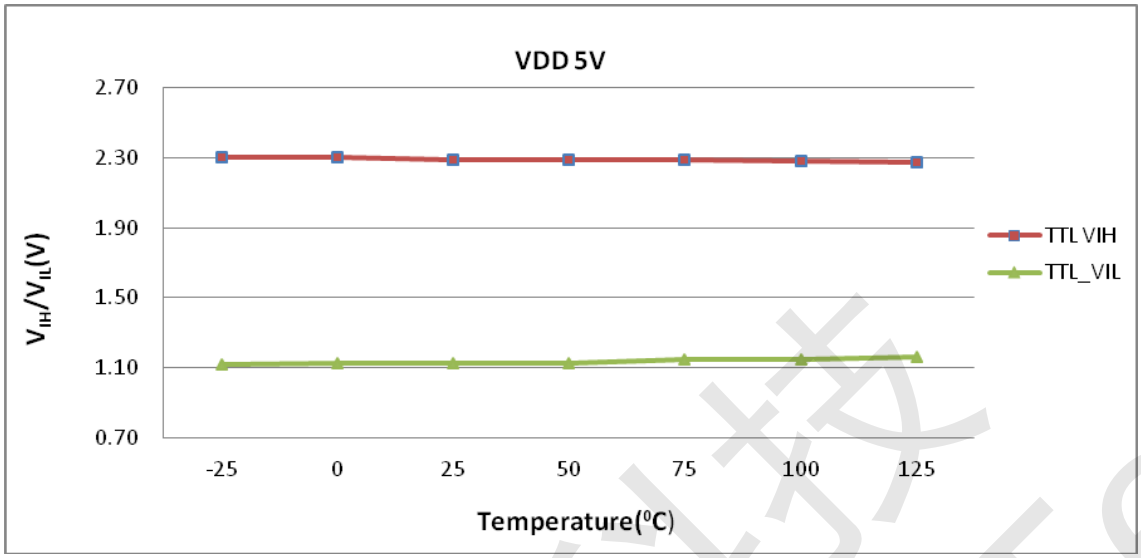


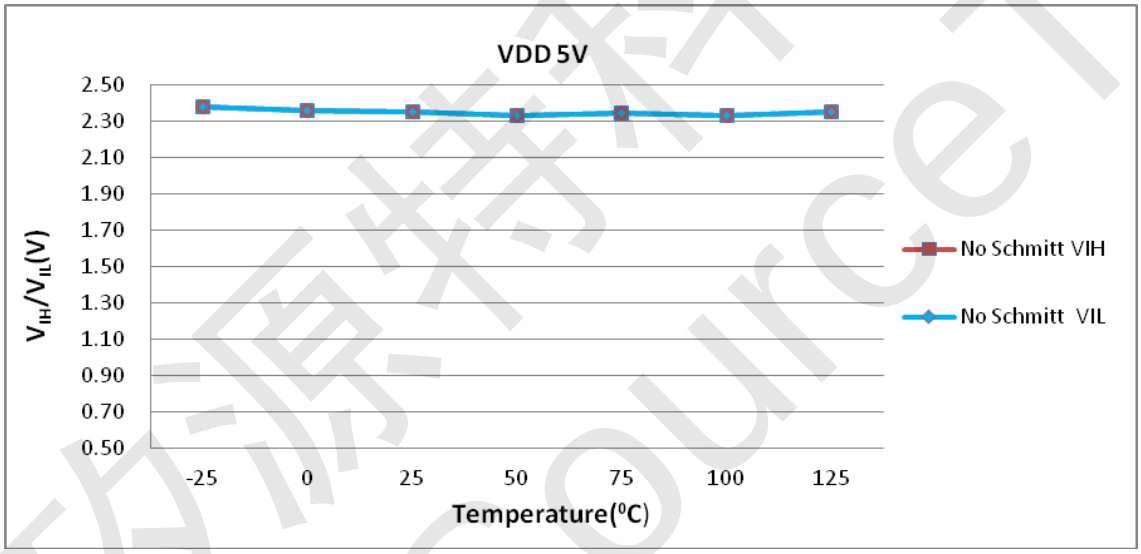


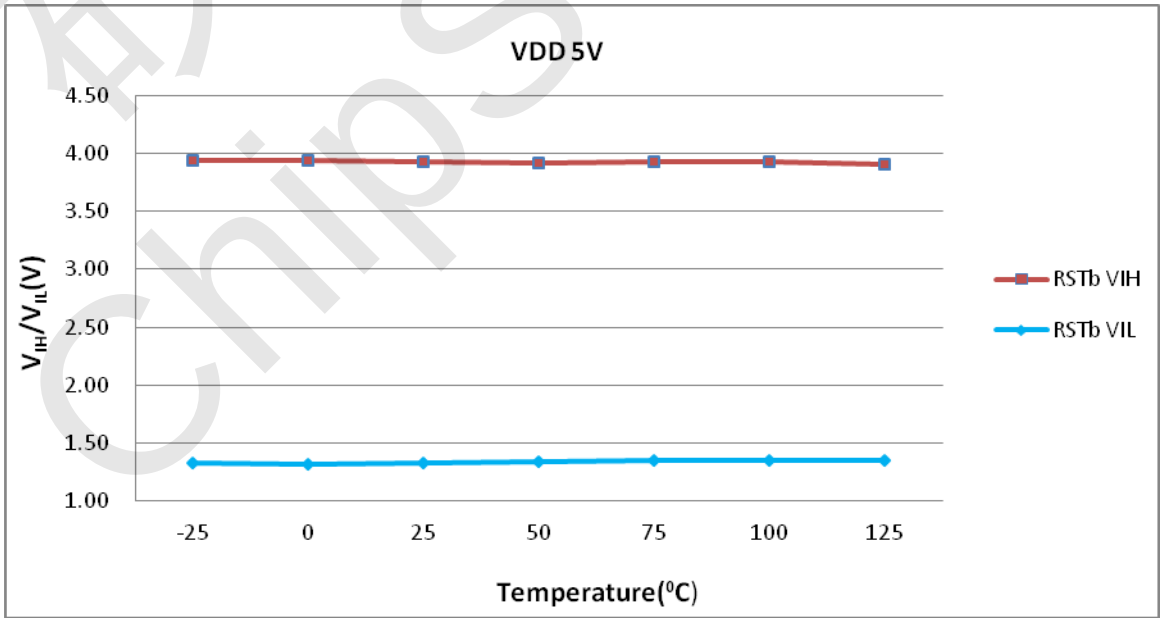


6.3.8 VIH/VIL vs. Temperature







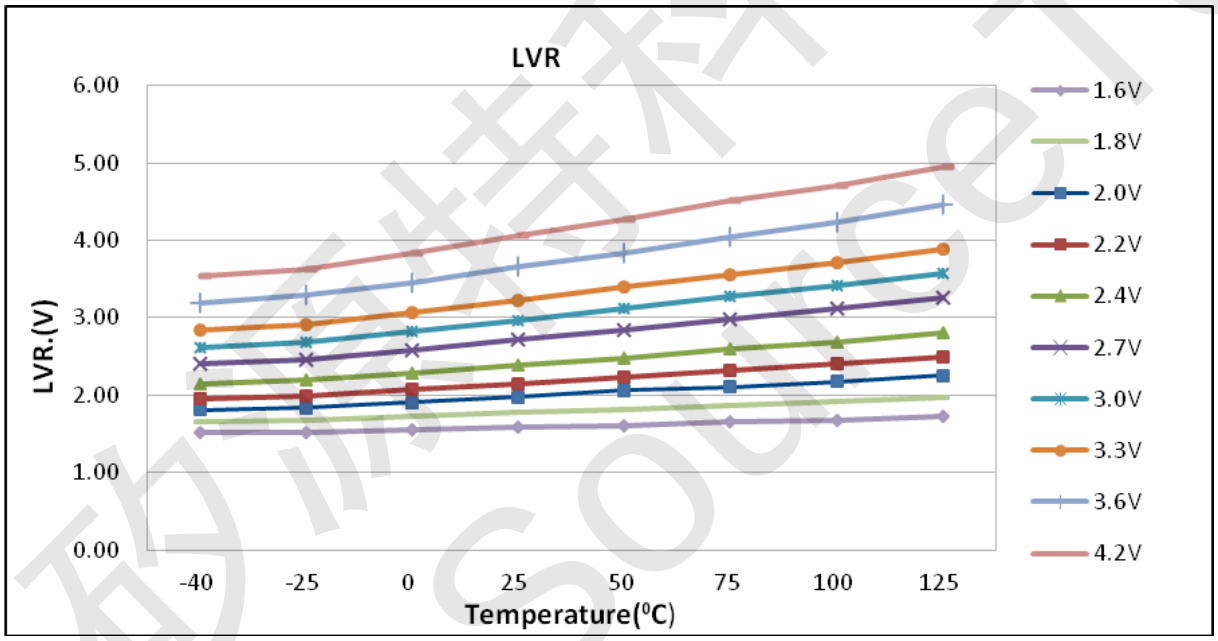


6.4 Recommended Operating Voltage

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

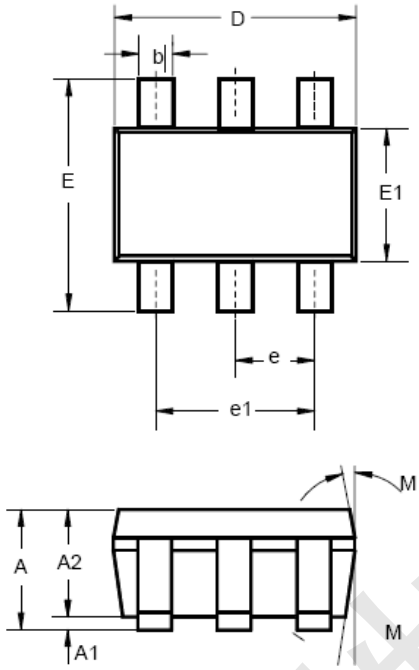
Frequency	Min. Voltage	Max. Voltage	LVR : default (25 °C)	LVR : Recommended (-40 °C ~ +85 °C)
20M/2T	3.0V	5.5V	3.0V	3.3V
16M/2T	2.7V	5.5V	2.7V	3.0V
20M/4T	2.2V	5.5V	2.2V	2.4V
16M/4T	2.0V	5.5V	2.0V	2.2V
8M(2T or 4T)	2.0V	5.5V	2.0V	2.2V
≦4M(2T or 4T)	1.6V	5.5V	1.6V	1.8V

6.5 LVR vs. Temperature



7. Package Dimension

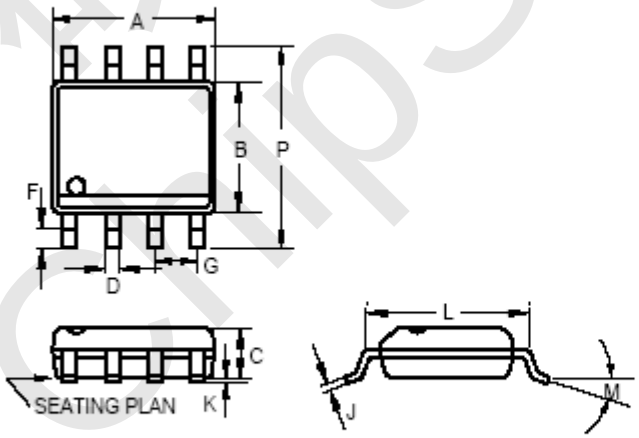
7.1 6-Pin Plastic SOT23-6 (63 mil)



	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.

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

8. Ordering Information

P/N	Package Type	Pin Count	Package Width	Shipping
NY8A050DS6	SOT23-6	6	63 mil	Tape & Reel: 3.0K pcs per Reel
NY8A050DS8	SOP	8	150 mil	Tape & Reel: 2.5K pcs per Reel Tube: 100 pcs per Tube