Preliminary V 1.3

SN8P2613
SN8P2612
SN8P2611

SONiX 8-Bit Micro-Controller

SONIX reserves the right to make change without further notice to any products herein to improve reliability, function or design. SONIX does not assume any liability arising out of the application or use of any product or circuit described herein; neither does it convey any license under its patent rights nor the rights of others. SONIX products are not designed, intended, or authorized for us as components in systems intended, for surgical implant into the body, or other applications intended to suppor

AMENDMENT HISTORY

Version	Date	Description
VER 0.1	Jan. 2005	First issue
VER 0.2	Jun. 2005	1. Add SN8P2611 item. 2. Add SN8P2611 programming pin mapping for Writer connection. 3. Add P-DIP 14 pins and SOP 14 pins package outline diagram.
	Aug. 2005	1. Remove Writer V2.5 information. 2. ADD P92 Note. Use M2IDE V1.06 (or after version) to simulation. 3. ADD P92 Note. Use 16M Hz Crystal to simulation internal 16M RC. 4. ADD P92 Note. Use 16M Hz Crystal to programming with EZ-Writer. 5. Modify P89 Internal Hihg RC.
	Nov.2005	1. ADD Brown-Out reset circuit. 2. Working Voltage vs. Frequency graphs.
VER 1.0	Nov. 2005	1. Modify Topr value. 2. ADD IHRC curve.
VER 1.1	Dec 2005	1. Modify T0 RTC interrupt service routine and T0IRQ operation description. 2. Modify Brown-Out Reset description 3. Remove power consumption(Pc) 4. Remove Noise Filter Enable Working Voltage 5. Modify PIN DESCRIPTIONS(P1 wakeup function) 6. Modify IHRC_RTC code option description 7. Remove High clock 32K mode 8. Modify M2IDE 1.07 9. Add Fcpu limitation by noise filter. 10. Modify ELECTRICAL CHARACTERISTIC. 11. Remove RTC function.
VER 1.2	May 2005	1. Modify Programming Pin Mapping
VER 1.3	Feb 2007	1. Add Marking Definition. 2. Modify ELECTRICAL CHARACTERISTIC. 3. Modify RST/P1.5/VPP PIN DISCRIPTION.

Table of Content

AMENDMENT HISTORY 2

3.1 OVERVIEW 37
3.2 POWER ON RESET 38
3.3 WATCHDOG RESET 38
3.4 BROWN OUT RESET 39
3.4.1 BROWN OUT DESCRIPTION 39
3.4.2 THE SYSTEM OPERATING VOLTAGE DECSRIPTION 40
3.4.3 BROWN OUT RESET IMPROVEMENT 40
3.5 EXTERNAL RESET 43
3.6 EXTERNAL RESET CIRCUIT 43
3.6.1 Simply RC Reset Circuit 43
3.6.2 Diode & RC Reset Circuit 44
3.6.3 Zener Diode Reset Circuit 44
3.6.4 Voltage Bias Reset Circuit 45
3.6.5 External Reset IC 46

4 SYSTEM CLOCK 47

4.1 OVERVIEW 47
4.2 CLOCK BLOCK DIAGRAM 47
4.3 OSCM REGISTER 48
4.4 SYSTEM HIGH CLOCK 49
4.4.1 INTERNAL HIGH RC 49
4.4.2 EXTERNAL HIGH CLOCK 49
4.4.2.1 CRYSTAL/CERAMIC 50
4.4.2.2 RC 50
4.4.2.3 EXTERNAL CLOCK SIGNAL 51
4.5 SYSTEM LOW CLOCK 52
4.5.1 SYSTEM CLOCK MEASUREMENT 53

5 SYSTEM OPERATION MODE 54

5.1 OVERVIEW 54
5.2 SYSTEM MODE SWITCHING EXAMPLE 55
5.3 WAKEUP 57
5.3.1 OVERVIEW 57
5.3.2 WAKEUP TIME 57
5.3.3 P1W WAKEUP CONTROL REGISTER 57

6 INTERRUPT 58

6.1 OVERVIEW 58
6.2 INTEN INTERRUPT

6.5 PUSH, POP ROUTINE 60
6.6 INT0 (P0.0) INTERRUPT OPERATION 61
6.7 INT1 (P0.1) INTERRUPT OPERATION 63
6.8 T0 INTERRUPT OPERATION 64
6.9 TC0 INTERRUPT OPERATION 65
6.10 MULTI-INTERRUPT OPERATION 66

7 I/O PORT 67

7.1 I/O PORT MODE 67
7.2 I/O PULL UP REGISTER 68
7.3 I/O OPEN-DRAIN REGISTER 69
7.4 I/O PORT DATA REGISTER 70

8 TIMERS 71

8.1 WATCHDOG TIMER 71
8.2 TIMER0 (T0) 73
8.2.1 OVERVIEW 73
8.2.2 T0M MODE REGISTER 73
8.2.3 T0C COUNTING REGISTER 74
8.2.4 T0 TIMER OPERATION SEQUENCE 75
8.3 TIMER/COUNTER0 (TC0) 76
8.3.1 OVERVIEW 76
8.3.2 TC0M MODE REGISTER 77
8.3.3 TC0C COUNTING REGISTER 78
8.3.4 TC0R AUTO-LOAD REGISTER 79
8.3.5 TC0 CLOCK FREQUENCY OUTPUT (BUZZER) 80
8.3.6 TC0 TIMER OPERATION SEQUENCE 81
8.4 PWM0 MODE 83
8.4.1 OVERVIEW 83
8.4.2 TCxIRQ and PWM Duty 84
8.4.3 PWM Duty with TCxR Changing 85
8.4.4 PWM PROGRAM EXAMPLE 86

9 INSTRUCTION TABLE 87

10 ELECTRICAL CHARACTERISTIC 88

10.1 ABSOLUTE MAXIMUM RATING 88
10.2 ELECTRICAL CHARACTERISTIC 88
10.3 CHARACTERISTIC GRAPHS 89

11 OTP PROGRAMMING PIN 90

11.1 THE PIN ASSIGNMENT OF EASY WRITER TRANSITION BOARD SOCKET: 90
11.2 THE PIN ASSIGNMENT OF W

1 PRODUCT OVERVIEW

1.1 FEATURES

Memory configuration
OTP ROM size: 2K * 16 bits.
RAM size: 64 * 8 bits.
Four levels stack buffer

I/O pin configuration
Bi-directional: P0, P1, P5.
Input only: P1.5.
Programmable open-drain: P1.0.
Wakeup: P0, P1 level change trigger
Pull-up resisters: P0, P1, P5.
External Interrupt trigger edge:
P0.0 controlled by PEDGE register.
P0.1 is falling edge trigger only.

3-Level LVD.
Reset system and power monitor.

Four interrupt sources
Two internal interrupts: T0, TC0.
One external interrupts: INT0, INT1.

Powerful instructions
One clocks per instruction cycle (1T)
Most of instructions are one cycle only.
All ROM area JMP instruction.
All ROM area CALL address instruction.
All ROM area lookup table function (MOVC)

Features Selection Table

Two 8-bit Timer/Counter
T0: Basic timer
TC0: Auto-reload timer/Counter/PWM0/Buzzer output

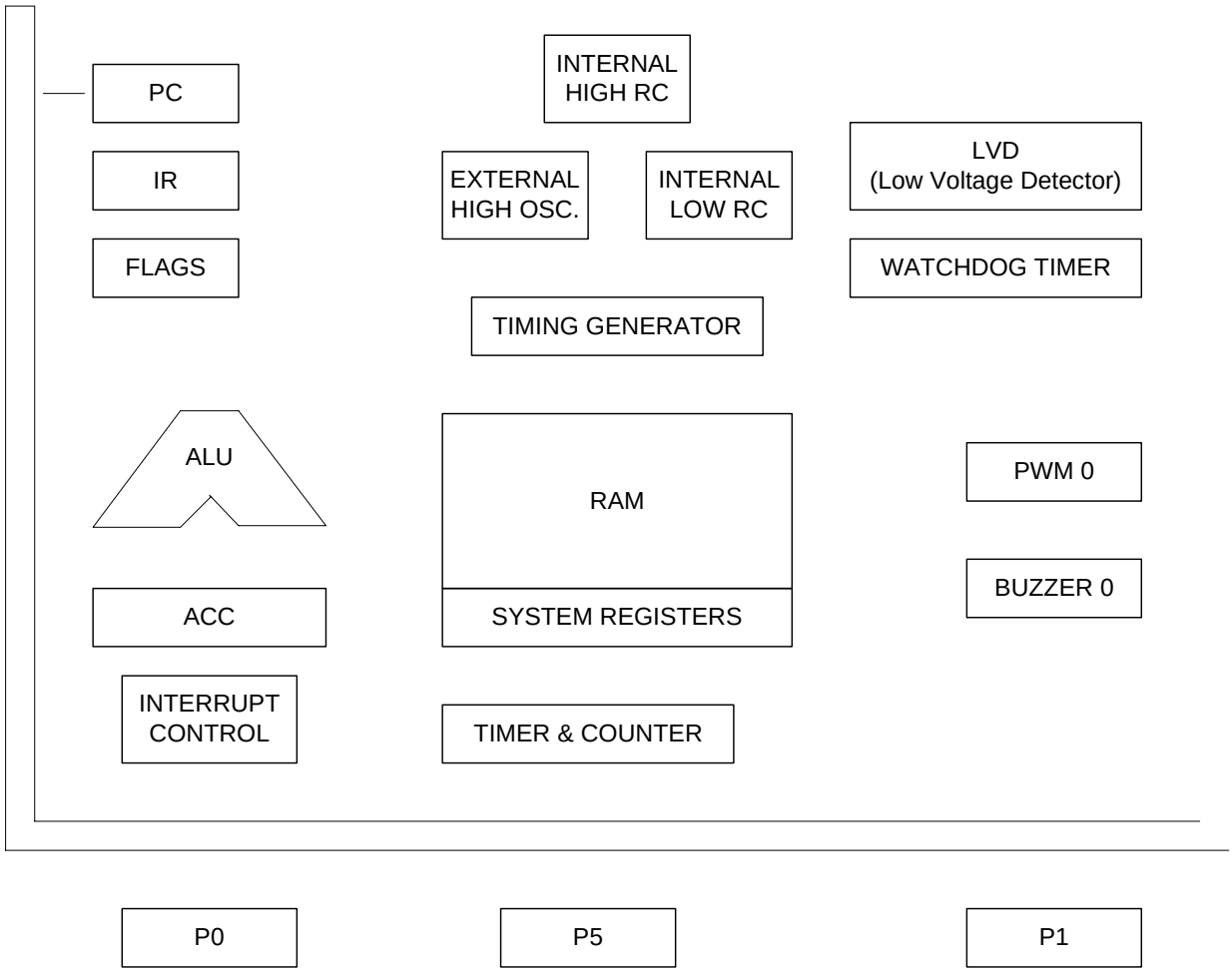
On chip watchdog timer and clock source is internal low clock RC type (16KHz@3V, 32KHz@5V).

Dual system clocks
External high clock: RC type up to 10 MHz
External high clock: Crystal type up to 16 MHz
Internal high clock: 16MHz RC type.
Internal low clock: RC type 16KHz(3V), 32KHz(5V)

Operating modes
Normal mode: Both high and low clock active
Slow mode: Low clock only
Sleep mode: Both high and low clock stop
Green mode: Periodical wakeup by T0 Timer

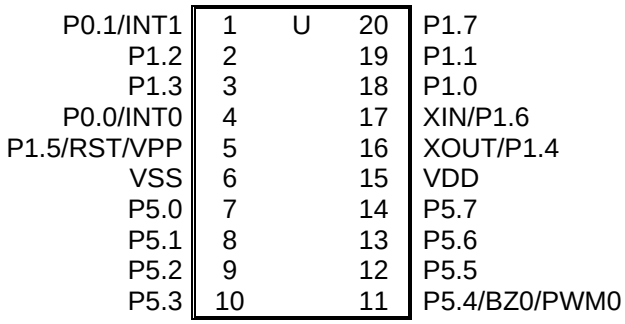
Package (Chip form support)
PDIP 20 pins
PDIP 18 pins
PDIP 14 pins
SOP 20 pins
SOP 18 pins
SOP 14 pins
SSOP 20 pins

1.2 SYSTEM BLOCK DIAGRAM



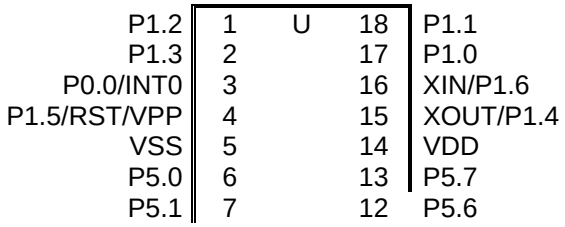
1.3 PIN ASSIGNMENT

SN8P2613P (P-DIP 20 pins)
SN8P2613S (SOP 20 pins)
SN8P2613X (SSOP 20 pins)

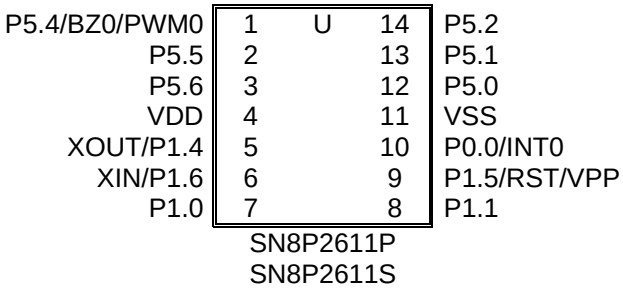


SN8P2613P
SN8P2613S
SN8P2613X

SN8P2612P (P-DIP 18 pins)
SN8P2612S (SOP 18 pins)
SN8P2612X (SSOP 20 pins)



SN8P2611P (P-DIP 14 pins)
SN8P2611S (SOP 14 pins)

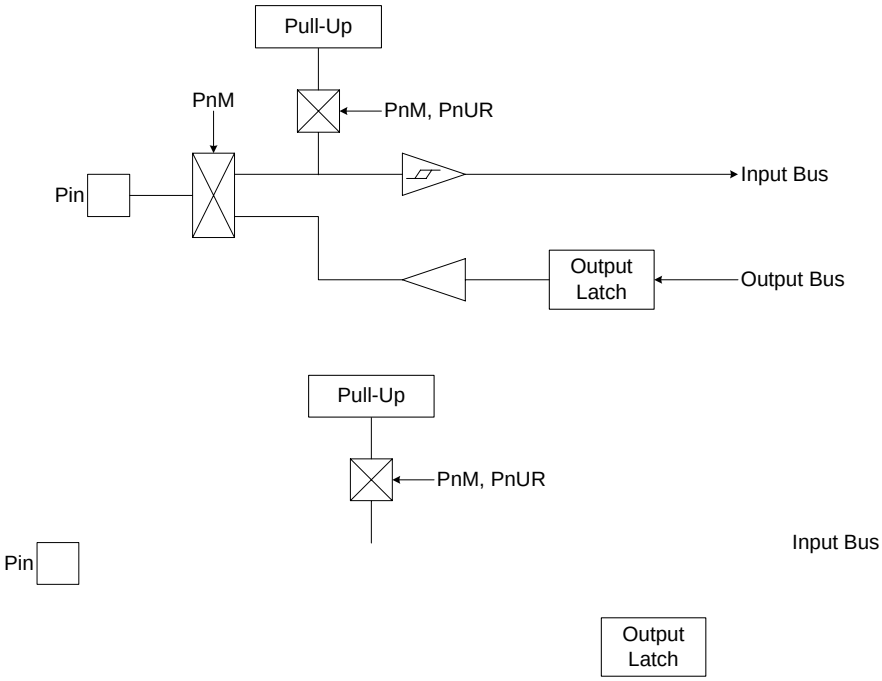


SN8P2611P
SN8P2611S

1.4 PIN DESCRIPTIONS

VDD, VSS	P	Power supply input pins for digital circuit.
P1.5/RST/VPP	I, P	P1.5: Input only pin (Schmitt trigger) if disable external reset function. **P1.5 without build-in pull-up resister.** **P1.5 is input only pin without pull-up resistor under P1.5 mode. Add the 100 ohm external resistor on P1.5, when it is set to be input pin.**

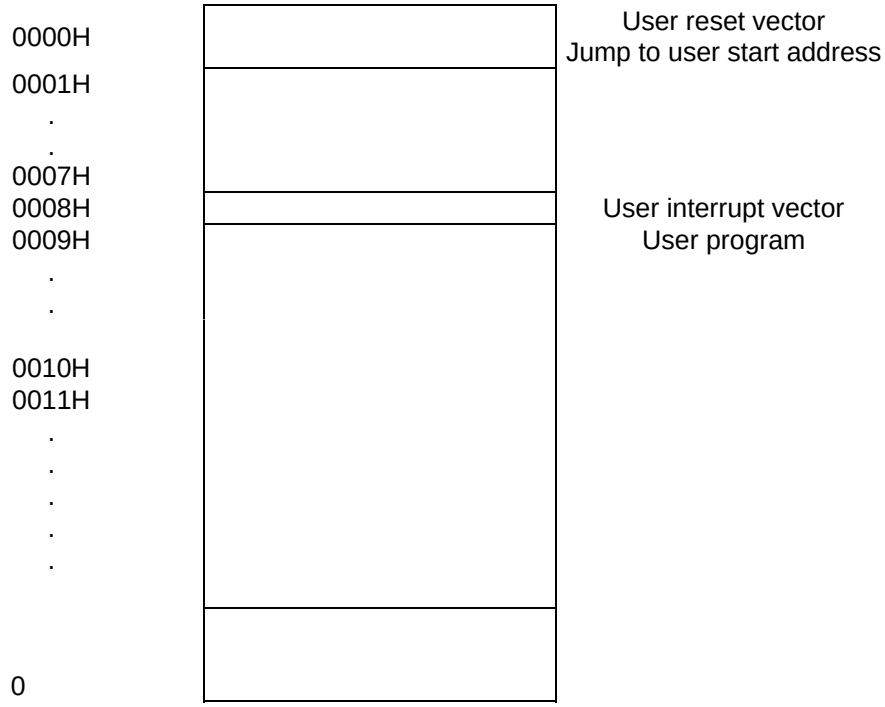
1.5 PIN CIRCUIT DIAGRAMS



2 CENTRAL PROCESSOR UNIT (CPU)

2.1 MEMORY MAP

2.1.1 PROGRAM MEMORY (ROM)



2.1.1.1 RESET VECTOR (0000H)

A one-word vector address area is used to execute system reset.

Power On Reset (NT0=1, NPD=0).
Watchdog Reset (NT0=0, NPD=0).
External Reset (NT0=1, NPD=1).

After power on reset, external reset or watchdog timer overflow reset, then the chip will restart the program from address 0000h and all system registers will be set as default values. It is easy to know reset status from NT0, NPD flags of PFLAG register. The following example shows the way to define the reset vector in the program memory.

Example: Defining Reset Vector

```
        ORG     0           ; 0000H
        JMP     START       ; Jump to user program address.
        …

        ORG     10H
START:                      ; 0010H, The head of user program.
        …                   ; User program
        …

        ENDP                ; End of program
```

2.1.1.2 INTERRUPT VECTOR (0008H)

A 1-word vector address area is used to execute interrupt request. If any interrupt service executes, the program counter (PC) value is stored in stack buffer and jump to 0008h of program memory to execute the vectored interrupt. Users have to define the interrupt vector. The following example shows the way to define the interrupt vector in the program memory.

Example: Defining Interrupt Vector. The interrupt service routine is following user program.

```
.CODE
        ORG     0           ; 0000H
        JMP     START       ; Jump to user program address.
        …
        ORG     8           ; Interrupt vector.
        JMP     MY_IRQ      ; 0008H, Jump to interrupt service routine address.

        ORG     10H
START:                      ; 0010H, The head of user program.
        …                   ; User program.
        …
        …
        JMP     START       ; End of user program.
        …
MY_IRQ:                     ;The head of interrupt service routine.
        PUSH                ; Save ACC and PFLAG register to buffers.
        …
        …
        POP                 ; Load ACC and PFLAG register from buffers.
        RETI                ; End of interrupt service routine.
        …

        ENDP                ; End of program.
```

2.1.1.3 LOOK-UP TABLE DESCRIPTION

In the ROM's data lookup function, Y register is pointed to middle byte address (bit 8~bit 15) and Z register is pointed to low byte address (bit 0~bit 7) of ROM. After MOVC instruction executed, the low-byte data will be stored in ACC and high-byte data stored in R register.

Example: To look up the ROM data located "TABLE1".

```
                B0MOV   Y, #TABLE1$M    ; To set lookup table1's middle address
                B0MOV   Z, #TABLE1$L    ; To set lookup table1's low address.
                MOVC                    ; To lookup data, R = 00H, ACC = 35H

                                        ; Increment the index address for next address.
                INCMS   Z               ; Z+1
                JMP     @F              ; Z is not overflow.
                INCMS   Y               ; Z overflow (FFH    00),    Y=Y+1
                NOP                     ;
                                        ;
@@:             MOVC                    ; To lookup data, R = 51H, ACC = 05H.
                …                       ;
TABLE1:         DW      0035H           ; To define a word (16 bits) data.
                DW      5105H
                DW      2012H
                …
```

Example: INC_YZ macro.

```
INC_YZ          MACRO
                INCMS   Z               ; Z+1
                JMP     @F              ; Not overflow

                INCMS   Y               ; Y+1
                NOP                     ; Not overflow
@@:
                ENDM
```

Example: Modify above exam

2.1.1.4 JUMP TABLE DESCRIPTION

The jump table operation is one of multi-address jumping function. Add low-byte program counter (PCL) and ACC value to get one new PCL. If PCL is overflow after PCL+ACC, PCH adds one automatically. The new program counter (PC) points to a series jump instructions as a listing table. It is easy to make a multi-jump program depends on the value of the accumulator (A).

Example: Jump table.

```
ORG     0X0100      ; The jump table is from the head of the ROM boundary

B0ADD   PCL, A      ; PCL = PCL + ACC, PCH + 1 when PCL overflow occurs.
JMP     A0POINT     ; ACC = 0, jump to A0POINT
JMP     A1POINT     ; ACC = 1, jump to A1POINT
JMP     A2POINT     ; ACC = 2, jump to A2POINT
JMP     A3POINT     ; ACC = 3, jump to A3POINT
```

SONIX provides a macro for safe jump table function. This macro will check the ROM boundary and move the jump

Example: "@JMP_A" application in SONIX macro file called "MACRO3.H".

```
B0MOV     A, BUF0       ; "BUF0" is from 0 to 4.
@JMP_A    5             ; The number of the jump table listing is five.
JMP       A0POINT       ; ACC = 0, jump to A0POINT
JMP       A1POINT       ; ACC = 1, jump to A1POINT
JMP       A2POINT       ; ACC = 2, jump to A2POINT
JMP       A3POINT       ; ACC = 3, jump to A3POINT
JMP       A4POINT       ; ACC = 4, jump to A4POINT
```

If the jump table position is across a ROM boundary (0x00FF~0x0100), the "@JMP_A" macro will adjust the jump table routine begin from next RAM boundary (0x0100).

Example: "@JMP_A" operation.

; Before compiling program.

```
ROM address
          B0MOV     A, BUF0       ; "BUF0" is from 0 to 4.
          @JMP_A    5             ; The number of the jump table listing is five.
0X00FD    JMP       A0POINT       ; ACC = 0, jump to A0POINT
0X00FE    JMP       A1POINT       ; ACC = 1, jump to A1POINT
0X00FF    JMP       A2POINT       ; ACC = 2, jump to A2POINT
0X0100    JMP       A3POINT       ; ACC = 3, jump to A3POINT
0X0101    JMP       A4POINT       ; ACC = 4, jump to A4POINT
```

; After compiling program.

```
ROM address
          B0MOV     A, BUF0       ; "BUF0" is from 0 to 4.
          @JMP_A    5             ; The number of the jump table listing is five.
0X0100    JMP       A0POINT       ; ACC = 0, jump to A0POINT
0X0101    JMP       A1POINT       ; ACC = 1, jump to A1POINT
0X0102    JMP       A2POINT       ; ACC = 2, jump to A2POINT
0X0103    JMP       A3POINT       ; ACC = 3, jump to A3POINT
0X0104    JMP       A4POINT       ; ACC = 4, jump to A4POINT
```

2.1.1.5 CHECKSUM CALCULATION

The last ROM address are reserved area. User should avoid these addresses (last address) when calculate the Checksum value.

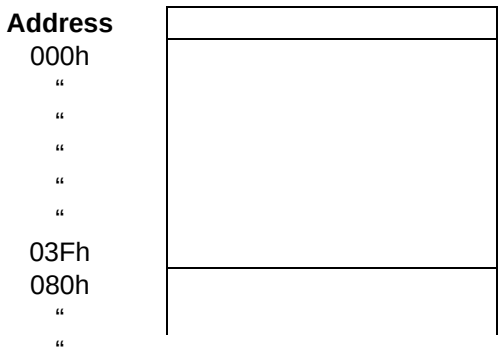
Example: The demo program shows how to calculated Checksum from 00H to the end of user's code.

```
MOV     A,#END_USER_CODE$L
```

2.1.2 CODE OPTION TABLE

Code Option	Content	Function Description
High_Clk	IHRC_16M	High speed internal 16MHz RC. XIN/XOUT become to P1.6/P1.4 bi-direction I/O pins.
	RC	Low cost RC for external high clock oscillator and XOUT becomes to P1.4 bit-direction I/O pin.
	12M X'tal	High speed crystal /resonator (e.g. 12MHz) for external high clock oscillator.
	4M X'tal	Standard crystal /resonator (e.g. 4M) for external high clock oscillator.
Watch_Dog	Always_On	Watchdog timer is always on enable even in power down and green mode.
	Enable	Enable watchdog timer. Watchdog timer stops in power down mode and green mode.
	Disable	Disable Watchdog function.
	Fhosc/1	Instruction cycle is oscillator clock. Notice: In Fosc/1, Noise Filter must be disabled.
	Fhosc/2	Instruction cycle is 2 oscillator clocks. Notice: In Fosc/2, Noise Filter must be disabled.
	Fhosc/4	Instruction cycle is 4 oscillator clocks.

2.1.3 DATA MEMORY (RAM)



80h~FFh of Bank 0 store system registers (128 bytes).

2.1.4.3 BIT DEFINITION of SYSTEM REGISTER

Address	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	R/W	Remarks
082H	RBIT7	RBIT6	RBIT5	RBIT4	RBIT3	RBIT2	RBIT1	RBIT0	R/W	R
083H	ZBIT7	ZBIT6	ZBIT5	ZBIT4	ZBIT3	ZBIT2	ZBIT1	ZBIT0	R/W	Z
084H	YBIT7	YBIT6	YBIT5	YBIT4	YBIT3	YBIT2	YBIT1	YBIT0	R/W	Y
086H	NT0	NPD	LVD36	LVD24	-	C	DC	Z	R/W	PFLAG

2.1.4.4 ACCUMULATOR

The ACC is an 8-bit data register responsible for transferring or manipulating data between ALU and data memory. If the result of operating is zero (Z) or there is carry (C or DC) occurrence, then these flags will be set to PFLAG register. ACC is not in data memory (RAM), so ACC can't be access by "B0MOV" instruction during the instant addressing mode.

Example: Read and write ACC value.

```
; Read ACC data and store in BUF data memory.
                MOV         BUF, A

; Write a immediate data into ACC.
                MOV         A, #0FH

; Write ACC data from BUF data memory.
                MOV         A, BUF
; or
                B0MOV       A, BUF
```

The system doesn't store ACC and PFLAG value when interrupt executed. ACC and PFLAG data must be saved to other data memories. "PUSH", "POP" save and load ACC, PFLAG data into buffers.

Example: Protect ACC and working registers.

```
INT_SERVICE:
                PUSH                ; Save ACC and PFLAG to buffers.
                …           .
                …
                POP                 ; Load ACC and PFLAG from buffers.

                RETI                ; Exit interrupt service vector
```

2.1.4.5 PROGRAM FLAG

The PFLAG register contains the arithmetic status of ALU operation, system reset status and LVD detecting status. NT0, NPD bits indicate system reset status including power on reset, LVD reset, reset by external pin active and watchdog reset. C, DC, Z bits indicate the result status of ALU operation. LVD24, LVD36 bits indicate LVD detecting power voltage status.

086H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PFLAG	NT0	NPD	LVD36	LVD24	-	C	DC	Z
Read/Write	R/W	R/W	R	R	-	R/W	R/W	R/W
After reset	-	-	0	0	-	0	0	0

Bit [7:6] **NT0, NPD:** Reset status flag.

NT0	NPD	Reset Status
0	0	Watch-dog time out
0	1	Reserved
1	0	Reset by LVD
1	1	Reset by external Reset Pin

Bit 5 **LVD36:** LVD 3.6V operating flag and only support LVD code option is LVD_H.
0 = Inactive (VDD > 3.6V).
1 = Active (VDD <= 3.6V).

Bit 4 **LVD24:** LVD 2.4V operating flag and only support LVD code option is LVD_M.
0 = Inactive (VDD > 2.4V).
1 = Active (VDD <= 2.4V).

Bit 2 **C:** Carry flag
1 = Addition with carry, subtraction without borrowing, rotation with shifting out logic “1”, comparison result ≥ 0.
0 = Addition without carry, subtraction with borrowing signal, rotation with shifting out logic “0”, comparison result < 0.

Bit 1 **DC:** Decimal carry flag
1 = Addition with carry from low nibble, subtraction without borrow from high nibble.
0 = Addition without carry from low nibble, subtraction with borrow from high nibble.

Bit 0 **Z:** Zero flag
1 = The result of an arithmetic/logic/branch operation is zero.
0 = The result of an arithmetic/logic/branch operation is not zero.

2.1.4.6 PROGRAM COUNTER

The program counter (PC) is a 11-bit binary counter separated into the high-byte 3 and the low-byte 8 bits. This counter is responsible for pointing a location in order to fetch an instruction for kernel circuit. Normally, the program counter is automatically incremented with each instruction during program execution.

Besides, it can be replaced with specific address by executing CALL or JMP instruction. When JMP or CALL instruction is executed, the destination address will be inserted to bit 0 ~ bit 10.

	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PC	-	-	-	-	-	PC10	PC9	PC8	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0
After reset	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0
	PCH								PCL							

ONE ADDRESS SKIPPING

There are nine instructions (CMPRS, INCS, INCMS, DECS, DECMS, BTS0, BTS1, B0BTS0, B0BTS1) with one address skipping function. If the result of these instructions is true, the PC will add 2 steps to skip next instruction.

INCS instruction:

```
        INCS    BUF0
        JMP     C0STEP      ; Jump to C0STEP if ACC is not zero.
        …
        …
C0STEP: NOP
```

INCMS instruction:

```
        INCMS   BUF0
        JMP     C0STEP      ; Jump to C0STEP if BUF0 is not zero.
        …
        …
C0STEP: NOP
```

DECS instruction:

```
        DECS    BUF0
        JMP     C0STEP      ; Jump to C0STEP if ACC is not zero.
        …
        …
C0STEP: NOP
```

DECMS instruction:

```
        DECMS   BUF0
        JMP     C0STEP      ; Jump to C0STEP if BUF0 is not zero.
        …
        …
C0STEP: NOP
```

MULTI-ADDRESS JUMPING

Users can jump around the multi-address by either JMP instruction or ADD M, A instruction (M = PCL) to activate multi-address jumping function. Program Counter supports "**ADD M,A**" , "**ADC M,A**" and "**B0ADD M,A**" instructions for carry to PCH when PCL overflow automatically. For jump table or others applications, users can calculate PC value by the three instructions and don't care PCL overflow problem.

2.1.4.7 Y, Z REGISTERS

The Y and Z registers are the 8-bit buffers. There are three major functions of these registers.
- can be used as general working registers
- can be used as RAM data pointers with @YZ register
- can be used as ROM data pointer with the MOVC instruction for look-up table

084H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Y	YBIT7	YBIT6	YBIT5	YBIT4	YBIT3	YBIT2	YBIT1	YBIT0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	-	-	-	-	-	-	-	-

083H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Z	ZBIT7	ZBIT6	ZBIT5	ZBIT4	ZBIT3	ZBIT2	ZBIT1	ZBIT0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	-	-	-	-	-	-	-	-

2.1.4.8 R REGISTERS

R register is an 8-bit buffer. There are two major functions of the register.

Can be used as working register

For store high-byte data of look-up table

(MOVC instruction executed, the high-byte data of specified ROM address will be stored in R register and the low-byte data will be stored in ACC).

082H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
R	RBIT7	RBIT6	RBIT5	RBIT4	RBIT3	RBIT2	RBIT1	RBIT0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	-	-	-	-	-	-	-	-

2.3 STACK OPERATION

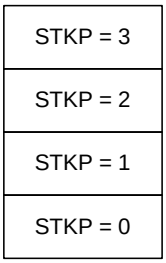
2.3.1 OVERVIEW

The stack buffer has 4-level. These buffers are designed to push and pop up program counter's (PC) data when interrupt service routine and "CALL" instruction are executed. The STKP register is a pointer designed to point active level in order to push or pop up data from stack buffer. The STKnH and STKnL are the stack buffers to store program counter (PC) data.

RET /
RETI

CALL /
INTERRUPT

STACK LevelCALL /



2.3.2 STACK REGISTERS

The stack pointer (STKP) is a 3-bit register to store the address used to access the stack buffer, 10-bit data memory (STKnH and STKnL) set aside for temporary storage of stack addresses.

The two stack operations are writing to the top of the stack (push) and reading from the top of stack (pop). Push operation decrements the STKP and the pop operation increments each time. That makes the STKP always point to the top address of stack buffer and write the last program counter value (PC) into the stack buffer.

The program counter (PC) value is stored in the stack buffer before a CALL instruction executed or during interrupt service routine. Stack operation is a LIFO type (Last in and first out). The stack pointer (STKP) and stack buffer (STKnH and STKnL) are located in the system register area bank 0.

0DFH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
STKP	GIE	-	-	-	-	STKPB2	STKPB1	STKPB0
Read/Write	R/W	-	-	-	-	R/W	R/W	R/W
After reset	0	-	-	-	-	1	1	1

Bit[2:0] **STKPBn:** Stack pointer (n = 0 ~ 2)

Bit 7 **GIE:** Global interrupt control bit.

2.3.3 STACK OPERATION EXAMPLE

The two kinds of Stack-Save operations refer to the stack pointer (STKP) and write the content of program counter (PC) to the stack buffer are CALL instruction and interrupt service. Under each condition, the STKP decreases and points to the next available stack location. The stack buffer stores the program counter about the op-code address. The Stack-Save operation is as the following table.

	STKPB2	**STKPB1**	**STKPB0**	**High Byte**	**Low Byte**	

3 RESET

3.1 OVERVIEW

The system would be reset in three conditions as following.

Power on reset

3.2 POWER ON RESET

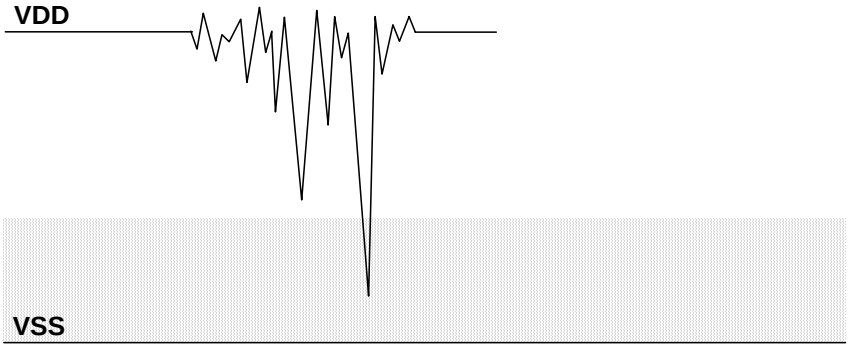
The power on reset depend no LVD operation for most power-up situations. The power supplying to system is a rising curve and needs some time to achieve the normal voltage. Power on reset sequence is as following.

Power-up: System detects the power voltage up and waits for power stable.

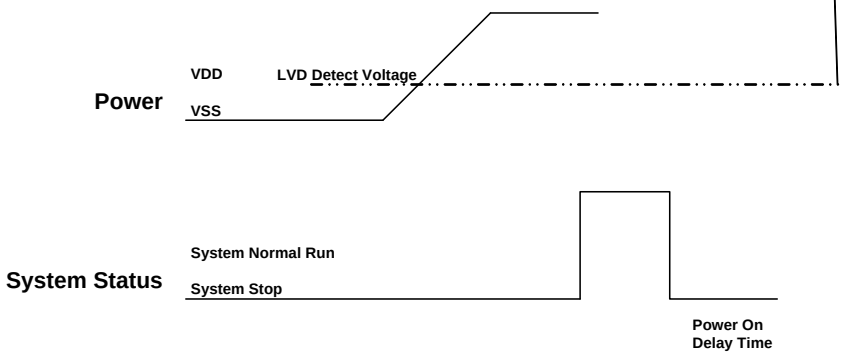
3.4 BROWN OUT RESET

3.4.1 BROWN OUT DESCRIPTION

The brown out reset is a power dropping condition. The power drops from normal voltage to low voltage by external factors (e.g. EFT interference or external loading changed). The brown out reset would make the system not work well or executing program error.



LVD reset:


Power
VDD
LVD Detect Voltage
VSS
System Status
System Normal Run
System Stop
Power On
Delay Time

LVD	LVD Code Option		
	LVD_L	LVD_M	LVD_H
2.0V Reset	Available	Available	Available
2.4V Flag	-	Available	-
2.4V Reset	-	-	Available
3.6V Flag	-	-	Available

LVD_L
If VDD < 2.0V, system will be reset.
Disable LVD24 and LVD36 bit of PFLAG register

LVD_M
If VDD < 2.0V, system will be reset.
Enable LVD24 bit of PFLAG register. If VDD > 2.4V, LVD24 is “0”. If VDD <= 2.4V, LVD24 flag is “1”
Disable LVD36 bit of PFLAG register
LVD2_H
If VDD < 2.4V, system will be reset.
Enable LVD24 bit of PFLAG register. If VDD > 2.4V, LVD24 is “0”. If VDD <= 2.4V, LVD24 flag is “1”
Enable LVD36 bit of PFLAG register. If VDD > 3.6V, LVD36 is “0”. If VDD <= 3.6V, LVD36 flag is “1”

Watchdog reset:
The watchdog timer is a protection to make sure the system executes well. Normally the watchdog timer would be clear at one point of program. Don’t clear the watchdog timer in several addresses. The system executes normally and the watchdog won’t reset system. When the system is under dead-band and the execution error, the watchdog timer can’t be clear by program. The watchdog is continuously counting until overflow occurrence. The overflow signal of watchdog timer triggers the system to reset, and the system return to normal mode after reset sequence. This method also can improve brown out reset condition and make sure the system to return normal mode.
If the system reset by watchdog and the power is still in dead-band, the system reset sequence won’t be successful and the system stays in reset status until the power return to normal range. Watchdog timer application note is as following.
Reduce the system executing rate:
If the system rate is fast and the dead-band exists, to reduce the system executing rate can improve the dead-band. The lower system rate is with lower minimum operating voltage. Select the power voltage that’s no dead-band issue and find out the mapping system rate. Adjust the system rate to the value and the system exits the dead-band issue. This way needs to modify whole program timing to fit the application requirement.
External reset circuit:
The external reset methods also can improve brown out reset and is the complete solution. There are three external reset circuits to improve brown out reset including “Zener diode reset circuit”, “Voltage bias reset circuit” and “External reset IC”. These three reset structures use external reset signal and control to make sure the MCU be reset under power dropping and under dead-band. The external rese

3.5 EXTERNAL RESET

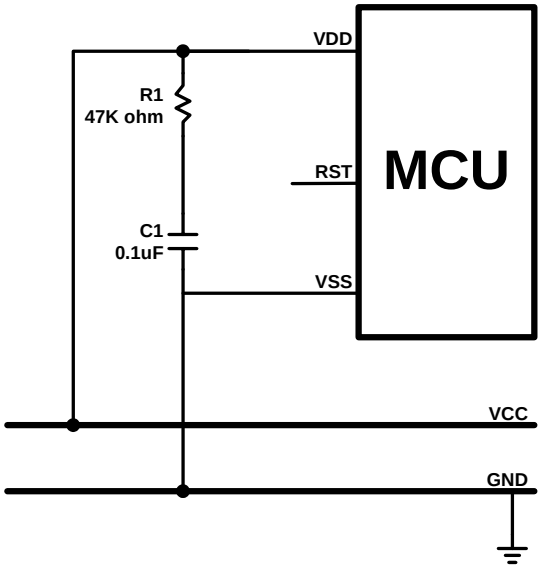
External reset function is controlled by “Reset_Pin” code option. Set the code option as “Reset” option to enable external reset function. External reset pin is Schmitt Trigger structure and low level active. The system is running when reset pin is high level voltage input. The reset pin receives the low voltage and the system is reset. The external reset operation actives in power on and normal running mode. During system power-up, the external reset pin must be high level input, or the system keeps in reset status. External reset sequence is as following.

External reset (only external reset pin enable): System checks external reset pin status. If external reset pin is not high level, the system keeps reset status and waits external reset pin released.
System initialization: All system registers is set as initial conditions and system is ready.
Oscillator warm up: Oscillator operation is successfully and supply to system clock.
Program executing: Power on sequence is finished and program executes from ORG 0.

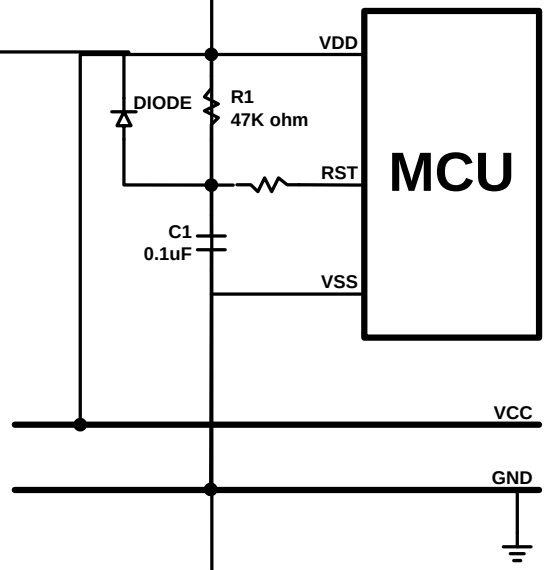
The external reset can reset the system during power on duration, and good external reset circuit can protect the system to avoid working at unusual power condition, e.g. brown out reset in AC power application…

3.6 EXTERNAL RESET CIRCUIT

3.6.1 Simply RC Reset Circuit



3.6.2 Diode & RC Reset Circuit



4 SYSTEM CLOCK

4.1 OVERVIEW

The micro-controller is a dual clock system. There are high-speed clock and low-speed clock. The high-speed clock is generated from the external oscillator circuit or on-chip

4.3 OSCM REGISTER

The OSCM register is an oscillator control register. It controls oscillator status, system mode.

0CAH

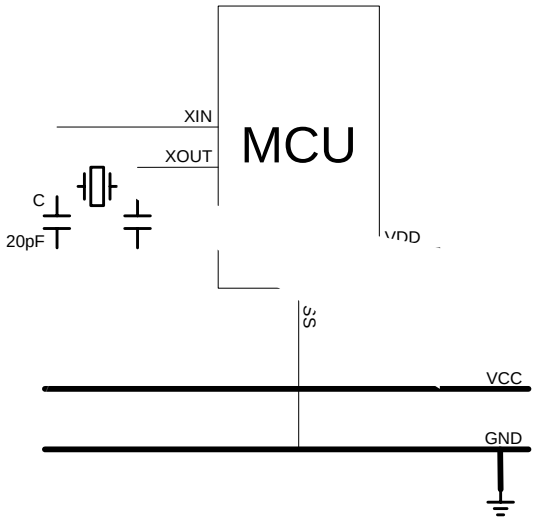
4.4 SYSTEM HIGH CLOCK

The system high clock is from internal 16MHz oscillator RC type or external oscillator. The high clock type is controlled by “High_Clk” code option.

High_Clk Code Option	Description
IHRC_16M	The high clock is internal 16MHz oscillator RC type. XIN and XOUT pins are general purpose I/O pins.
RC	The high clock is external RC type oscillator. XOUT pin is general purpose I/O pin.
12M	The high clock is external high speed oscillator. The typical frequency is 12MHz.
4M	The high clock is external oscillator. The typical frequency is 4MHz.

4.4.2.1 CRYSTAL/CERAMIC

Crystal/Ceramic devices are driven by XIN, XOUT pins. For high/normal/low frequency, the driving currents are different. High_Clk code option supports different frequencies. 12M option is for high speed (ex. 12MHz). 4M option is for normal speed (ex. 4MHz).



4.5 SYSTEM LOW CLOCK

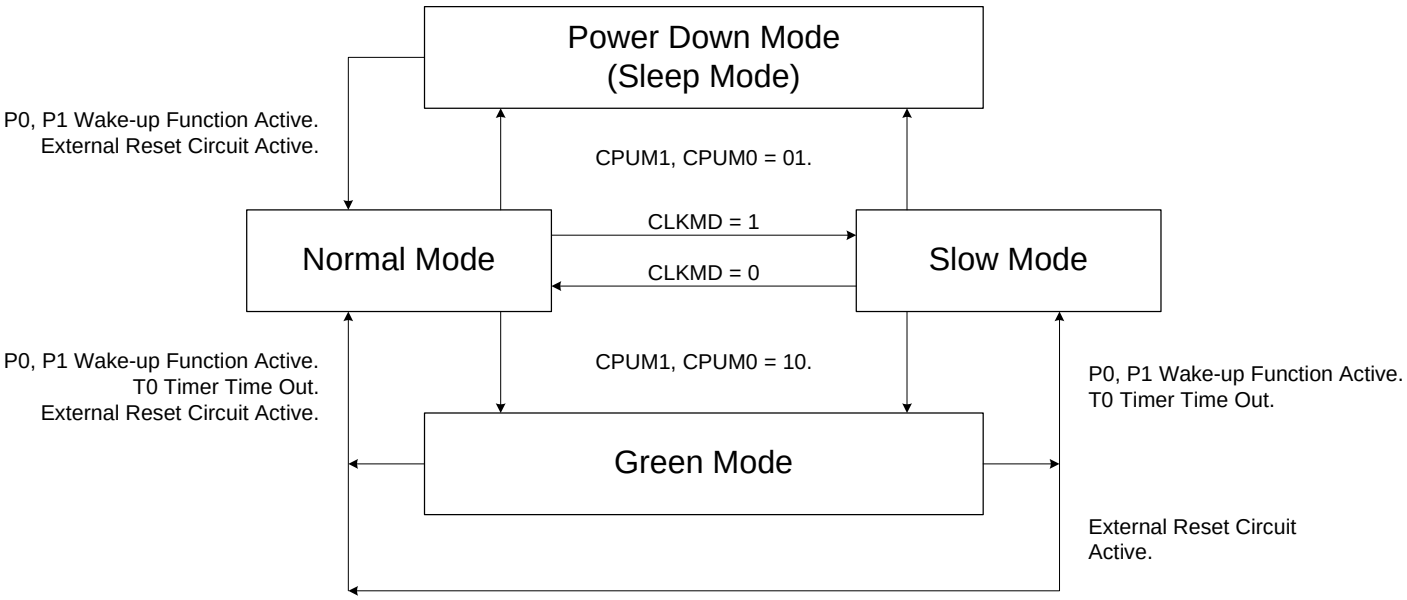
The system low clock source is the internal low-speed oscillator built in the micro-controller. The low-speed oscillator

5 SYSTEM OPERATION MODE

5.1 OVERVIEW

The chip is featured with low power consumption by switching around four different modes as following.

High-speed mode
Low-speed mode
Power-down mode (Sleep mode)
Green mode



5.2 SYSTEM MODE SWITCHING EXAMPLE

Example: Switch normal/slow mode to power down (sleep) mode.

```
B0BSET      FCPUM0          ; Set CPUM0 = 1.
```

Example: Switch normal mode to slow mode.

Example: Switch normal/slow mode to green mode and enable T0 wake-up function.

```
; Set T0 timer wakeup function.
        B0BCLR      FT0IEN      ; To disable T0 interrupt service
        B0BCLR      FT0ENB      ; To disable T0 timer
        MOV         A,#20H      ;
        B0MOV       T0M,A       ; To set T0 clock = Fcpu / 64
        MOV         A,#74H
        B0MOV       T0C,A       ; To set T0C initial value = 74H (To set T0 interval = 10 ms)
        B0BCLR      FT0IEN      ; To disable T0 interrupt service
        B0BCLR      FT0IRQ      ; To clear T0 interrupt request
        B0BSET      FT0ENB      ; To enable T0 timer
; Go into green mode
        B0BCLR      FCPUM0      ;To set CPUMx = 10
        B0BSET      FCPUM1
```

Example: INT0 interrupt service routine.

```
        ORG     8               ; Interrupt vector
        JMP     INT_SERVICE
INT_SERVICE:
        …                       ; Push routine to save ACC and PFLAG to buffers.

        B0BTS1  FP00IRQ         ; Check P00IRQ
        JMP     EXIT_INT        ; P00IRQ = 0, exit interrupt vector

        B0BCLR  FP00IRQ         ; Reset P00IRQ
        …                       ; INT0 interrupt service routine
        …
EXIT_INT:
        …                       ; Pop routine to load ACC and PFLAG from buffers.

        RETI                    ; Exit interrupt vector
```

6.7 INT1 (P0.1) INTERRUPT OPERATION

When the INT1 trigger occurs, the P01IRQ will be set to “1” no matter the P01IEN is enable or disable. If the P01IEN = 1 and the trigger event P01IRQ is also set to be “1”. As the result, the system will execute the interrupt vector (ORG 8). If the P01IEN = 0 and the trigger event P01IRQ is still se

6.8 T0 INTERRUPT OPERATION

When the T0C counter occurs overflow, the T0IRQ will be set to “1” however the T0IEN is enable or disable. If the T0IEN = 1, the trigger event will make the T0IRQ to be “1” and the system enter interrupt vector. If the T0IEN = 0, the trigger event will make the T0IRQ to be “1” but the system will not enter interrupt vector. Users need to care for the operation under multi-interrupt situation.

Example: T0 interrupt request setup.

```
B0BCLR      FT0IEN      ; Disable T0 interrupt service
B0BCLR      FT0ENB      ; Disable T0 timer
MOV         A, #20H     ;
B0MOV       T0M, A      ; Set T0 clock = Fcpu / 64
MOV         A, #74H     ; Set T0C initial value = 74H
B0MOV       T0C, A      ; Set T0 interval = 10 ms

B0BSET      FT0IEN      ; Enable T0 interrupt service
B0BCLR      FT0IRQ      ; Clear T0 interrupt request flag
B0BSET      FT0ENB      ; Enable T0 timer

B0BSET      FGIE        ; Enable GIE
```

Example: T0 interrupt service routine.

```
            ORG         8               ; Interrupt vector
            JMP         INT_SERVICE
INT_SERVICE:5.149CE:9-2.3952  -om520  -1.-.091/TTD7.2791  Tc0  Tw[(    )-592.8(  )-2341.3(  )]TJT*-0.001  Tc-0.0014  T
```

6.9 TC0 INTERRUPT OPERATION

When the TC0C counter overflows, the TC0IRQ will be set to “1” no matter the TC0IEN is enable or disable. If the TC0IEN and the trigger event TC0IRQ is set to be “1”. As the result, the system will execute the interrupt vector. If the TC0IEN = 0, the trigger event TC0IRQ is still set to be “1”. Moreover, the system won’t execute interrupt vector even when the TC0IEN is set to be “1”. Users need to be cautious with the operation under multi-interrupt situation.

Example: TC0 interrupt request setup.

```
        B0BCLR      FTC0IEN         ; Disable TC0 interrupt service
        B0BCLR      FTC0ENB         ; Disable TC0 timer
        MOV         A, #20H         ;
        B0MOV       TC0M, A         ; Set TC0 clock = Fcpu / 64
        MOV         A, #74H         ; Set TC0C initial value = 74H
        B0MOV       TC0C, A         ; Set TC0 interval = 10 ms

        B0BSET      FTC0IEN         ; Enable TC0 interrupt service
        B0BCLR      FTC0IRQ         ; Clear TC0 interrupt request flag
        B0BSET      FTC0ENB         ; Enable TC0 timer

        B0BSET      FGIE            ; Enable GIE
```

Example: TC0 interrupt service routine.

```
        ORG         8               ; Interrupt vector
        JMP         INT_SERVICE
INT_SERVICE:

        …                           ; Push routine to save ACC and PFLAG to buffers.

        B0BTS1      FTC0IRQ         ; Check TC0IRQ
        JMP         EXIT_INT        ; TC0IRQ = 0, exit interrupt vector

        B0BCLR      FTC0IRQ         ; Reset TC0IRQ
        MOV         A, #74H
        B0MOV       TC0C, A         ; Reset TC0C.
        …                           ; TC0 interrupt service routine
        …
EXIT_INT:
        …                           ; Pop routine to load ACC and PFLAG from buffers.

        RETI                        ; Exit interrupt vector
```

7 I/O PORT

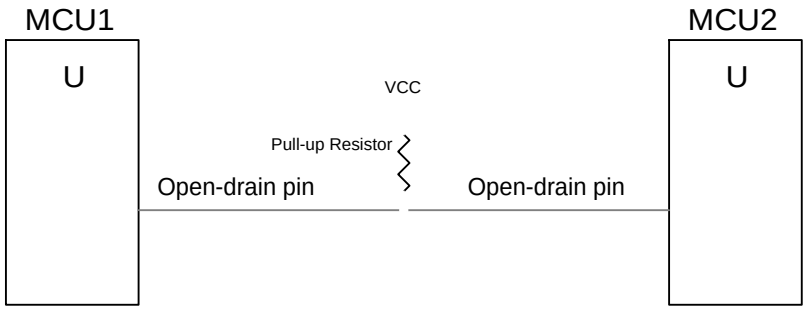
7.1 I/O PORT MODE

The port direction is programmed by PnM register. All I/O ports can select input or output direction.

0B8H	

7.3 I/O OPEN-DRAIN REGISTER

P1.0 is built-in open-drain function. P1.0 must be set as output mode when enable P1.0 open-drain function. Open-drain external circuit is as following.



The pull-up resistor is necessary. Open-drain output high is driven by pull-up resistor. Output low is sunken by MCU's pin.

0E9H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P1OC	-	-	-	-	-	-	-	P10OC
Read/Write	-	-	-	-	-	-	-	W
After reset	-	-	-	-	-	-	-	0

Bit 0 P10OC: P1.0 open-drain control bit
0 = Disable open-drain mode
1 = Enable open-drain mode

¾ Example: Enable P1.0 to open-drain mode and output high.

```
B0BSET    P1.0      ; Set P1.0 buffer high.

B0BSET    P10M      ; Enable P1.0 output mode.
MOV       A, #01H   ; Enable P1.0 open-drain function.
B0MOV     P1OC, A
```

' Note: P1OC is write only register. Setting P10OC must be used "MOV" instructions.

¾ Example: Disable P1.0 to open-drain mode and output low.

```
MOV       A, #0     ; Disable P1.0 open-drain function.
B0MOV     P1OC, A
```

' Note: After disable P1.0 open-drain function, P1.0 mode returns to last I/O mode.

7.4 I/O PORT DATA REGISTER

0D0H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P0	-	-	-	-	-	-	P01	P00
Read/Write	-	-	-	-	-	-	R/W	R/W
After reset	-	-	-	-	-	-	0	0

0D1H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P1	P17	P16	P15	P14	P13	P12	P11	P10
Read/Write	R	R/W	R	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

0D5H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P5	P57	P56	P55	P54	P53	P52	P51	P50
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
After reset	0	0	0	0	0	0	0	0

8 TIMERS

8.1 WATCHDOG TIMER

The watchdog timer (WDT) is a binary up counter designed for monitoring program execution. If the program goes into the unknown status by noise interference, WDT overflow signal raises and resets MCU. Watchdog clock controlled by code option and the clock source is internal low-speed oscillator (16KHz @3V, 32KHz @5V).

3V	16KHz	512ms
5V	32KHz	256ms

Watchdog clear is controlled by WDTR register. Moving **0x5A** data into WDTR is to reset watchdog timer.

0CCH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
WDTR	WDTR7	WDTR6	WDTR5	WDTR4	WDTR3	WDTR2	WDTR1	WDTR0
Read/Write	W	W	W	W	W	W	W	W

Watchdog timer application note is as following.

Before clearing watchdog timer, check I/O status and check RAM contents can improve system error.

Don't clear watchdog timer in interrupt vector and interrupt service routine. That can improve main routine fail. Clearing watchdog timer program is only at one part of the program. This way is the best structure to enhance the watchdog timer function.

Example: An operation of watchdog timer is as following. To clear the watchdog timer counter in the top of the main routine of the program.

```
Main:
            …                               ; Check I/O.
            …                               ; Check RAM
Err:        JMP $                           ; I/O or RAM error. Program jump here and don't
                                            ; clear watchdog. Wait watchdog timer overflow to reset IC.

Correct:                                    ; I/O and RAM are correct. Clear watchdog timer and
                                            ; execute program.
            B0BSET      FWDRST              ; Only one clearing watchdog timer of whole program.
            …
            CALL        SUB1
            CALL        SUB2
            …
            …
            …
            JMP         MAIN
```

8.2 TIMER 0 (T0)

8.2.1 OVERVIEW

The T0 is an 8-bit binary up timer and event counter. If T0 timer occurs an overflow (from FFH to 00H), it will continue

8.2.3 T0C COUNTING REGISTER

T0C is an 8-bit counter register for T0 interval time control.

8.3 TIMER/COUNTER 0 (TC0)

8.3.1 OVERVIEW

The TC0 is an 8-bit binary up counting timer with double buffers. TC0 has two clock sources including internal clock and external clock for counting a precision time. The internal clock source is from Fcpu. The external clock is INT0 from P0.0 pin (Falling edge trigger). Using TC0M register selects TC0C's clock source from internal or external. If TC0 timer occurs an overflow, it will continue counting and issue a time-out signal to trigger TC0 interrupt to request interrupt service. TC0 overflow time is 0xFF to 0X00 normally. Under PWM mode, TC0 overflow is decided by PWM cycle controlled by ALOAD0 and TC0OUT bits.

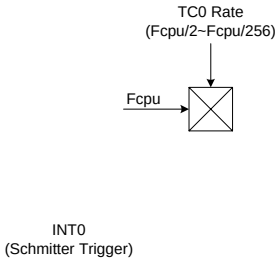
The main purposes of the TC0 timer is as following.

8-bit programmable up counting timer: Generates interrupts at specific time intervals based on the selected clock frequency.

External event counter: Counts system "events" based on falling edge detection of external clock signals at the INT0 input pin.

Buzzer output

PWM output



8.3.2 TC0M MODE REGISTER

8.3.3 TC0C COUNTING REGISTER

TC0C is an 8-bit counter register for TC0 interval time control.

0DBH	Bit 7	Bit 6	Bit 5		

8.3.4 TC0R AUTO-LOAD REGISTER

TC0 timer is with auto-load function controlled by ALOAD0 bit of TC0M. When TC0C overflow occurring, TC0R value will load to TC0C by system. It is easy to generate an accurate time, and users don't reset TC0C during interrupt service routine.

TC0 is double buffer design. If new TC0R value is set by program, the new value is stored in 1st buffer. Until TC0 overflow occurs, the new value moves to real TC0R buffer. This way can avoid TC0 interval time error and glitch in PWM and Buzzer output.

0CDH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0

8.3.5 TC0 CLOCK FREQUENCY OUTPUT (BUZZER)

8.3.6 TC0 TIMER OPERATION SEQUENCE

Set TC0 timer function mode.

```
        B0BSET      FTC0IEN         ; Enable TC0 interrupt function.
or
        B0BSET      FTC0OUT         ; Enable TC0OUT (Buzzer) function.
or
        B0BSET      FPWM0OUT        ; Enable PWM function.
```

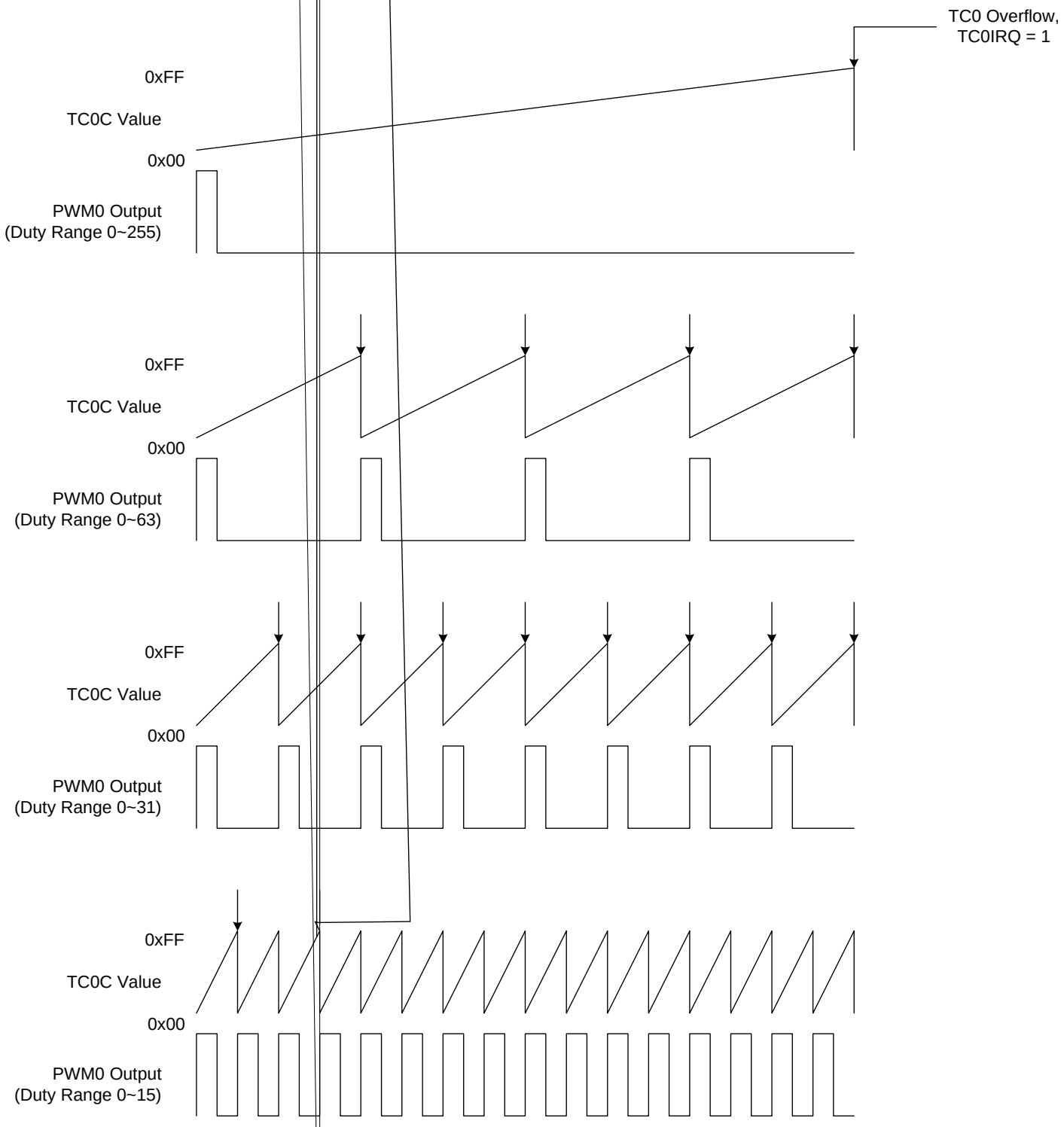
Enable TC0 timer.

```
        B0BSET      FTC0ENB         ; Enable TC0 timer.
```

8.4 PWM0 MODE

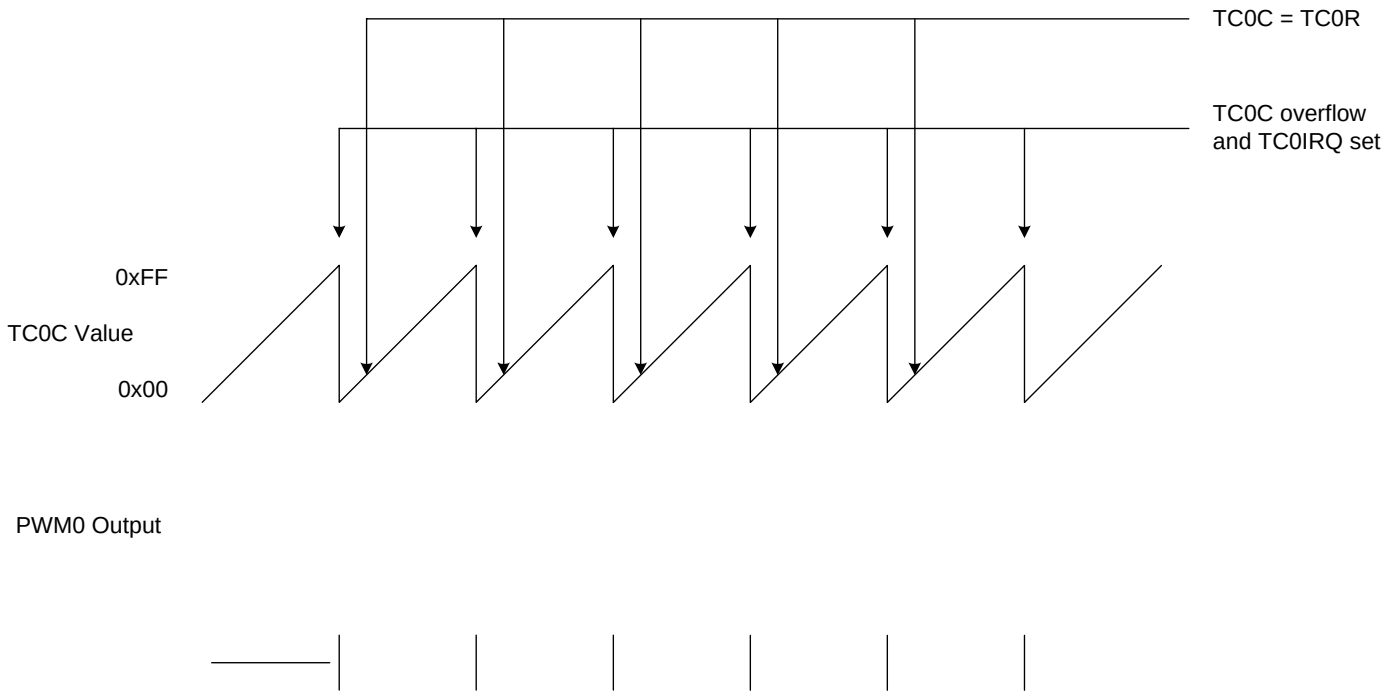
8.4.2 TCxIRQ and PWM Duty

In PWM mode, the frequency of TC0IRQ is depended on PWM duty range. From following diagram, the TC0IRQ frequency is related with PWM duty.



8.4.3 PWM Duty with TCxR Changing

In PWM mode, the system will compare TC0C and TC0R all the time. When TC0C<TC0R, the PWM will output logic "High", when TC0C TC0R, the PWM will output logic "Low". If TC0C is changed in certain period, the PWM duty will change in next PWM period. If TC0R is fixed all the time, the PWM waveform is also the same.



8.4.4 PWM PROGRAM EXAMPLE

Example: Setup PWM0 output from TC0 to PWM0OUT (P5.4). The external high-speed oscillator clock is 4MHz Fcpu = Fosc/4. The duty of PWM is 30/256. The PWM frequency is about 1KHz The PWM clock source is from external oscillator clock. TC0 rate is Fcpu/4. The TC0RATE2ŦC0RATE1 = 110. TC0C = TC0R = 30.

```
MOV     A,#01100000B
B0MOV   TC0M,A          ; Set the TC0 rate to Fcpu/4

MOV     A,#30           ; Set the PWM duty to 30/256
B0MOV   TC0C,A
B0MOV   TC0R,A

B0BCLR  FTC0OUT         ; Set duty range as 0/256~255/256.
B0BCLR  FALOAD0
B0BSET  FPWM0OUT        ; Enable PWM0 output to P5.4 and disable P5.4 I/O function
B0BSET  FTC0ENB         ; Enable TC0 timer
```

Example: Modify TC0R registers'value.

```
MOV     A, #30H         ; Input a number using B0MOV instruction.
B0MOV   TC0R, A

INCMS   BUF0            ; Get the new TC0R value from the BUF0 buffer defined by
NOP                     ; programming.
B0MOV   A, BUF0
B0MOV   TC0R, A
```

9 INSTRUCTION TABLE

M O V E	MOV	A,M	A M	-	-		1
	MOV	M,A	M A	-	-	-	1
	B0MOV	A,M	A M (bank 0)	-	-		1
	B0MOV	M,A	M (bank 0) A	-	-	-	1
	MOV	A,I	A I	-	-	-	1
	B0MOV	M,I	M I, “M” only supports 0x80~0x87 registers (e.g. PFLAG,R,Y,Z…)	-	-	-	1
	XCH	A,M	A M	-	-	-	1+N
	B0XCH	A,M	A M (bank 0)	-	-	-	1+N
	MOVC		R, A ROM [Y,Z]	-	-	-	2
A	ADC	A,M	A A + M + C, if occur carry, then C=1, else C=0				1
	ADC	M,A	M A + M + C, if occur carry, then C=1, else C=0				1+N

10 ELECTRICAL CHARACTERISTIC

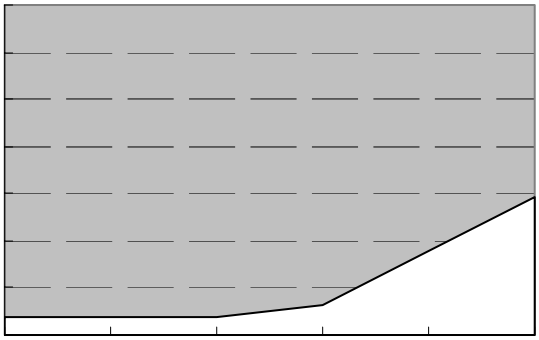
10.1 ABSOLUTE MAXIMUM RATING

Supply voltage (Vdd)..... - 0.3V ~ 6.0V
Input in voltage (Vin)..... Vss – 0.2V ~ Vdd + 0.2V
Operating ambient temperature (Topr)
SN8P2613/12P, SN8P2613/12S, SN8P2613/12X ..... 0 C ~ + 70 C
SN8P2613/12PD, SN8P2613/12SD, SN8P2613/12XD .....–40 C ~ + 85 C
Storage ambient temperature (Tstor) ..... –40 C ~ + 125 C

10.2 ELECTRICAL CHARACTERISTIC

(All of voltages refer to Vss, Vdd = 5.0V, fosc = 4MHz, Fcpu=1MHZ,ambient temperature is 25 C unless otherwise note.)

Operating voltage	Vdd	Normal mode, Vpp = Vdd, 25 C Normal mode, Vpp = Vdd,	2.4 2.4	5.0 5.0	5.5 5.5	V



11 OTP PROGRAMMING PIN

11.1 The pin assignment of Easy Writer transition board socket:

Easy Writer JP1/JP2

VSS	2	1	VDD
CE	4	3	CLK/PGCLK
OE/ShiftDat	6	5	PGM/OTPCLK
D0	8	7	D1
D2	10	9	D3
D4	12	11	D5
D6	14	13	D7
VPP	16	15	VDD

Easy Writer JP3 (Mapping to 48-pin text tool)

DIP1	1	48	DIP48
DIP2	2	47	DIP47
DIP3	3	46	DIP46
DIP4	4	45	DIP45
DIP5	5	44	DIP44
DIP6	6	43	DIP43
DIP7	7	42	DIP42
DIP8	8	41	DIP41

11.2 The pin assignment of Writer V3.0 transition board socket:

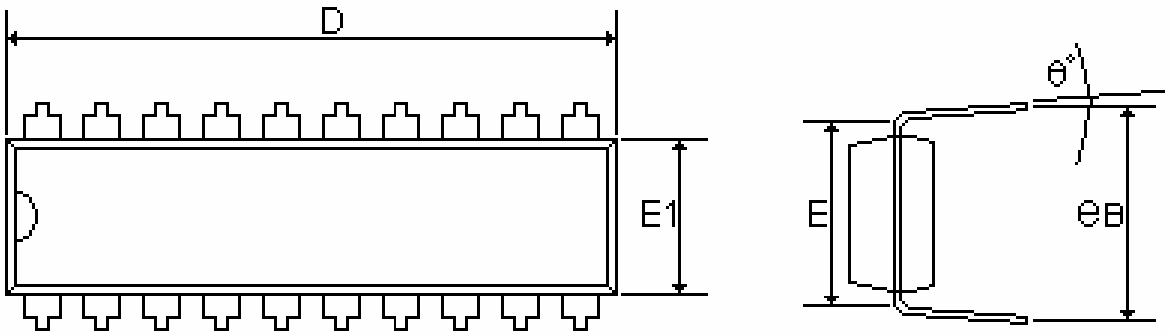
GND	2	1	VDD
CE	4	3	CLK
OE	6	5	PGM
D0	8	7	D1
D2	10	9	D3
D4	12	11	D5
D6	14	13	D7
VPP	16	15	VDD
RST	18	17	HLS
	20	19	

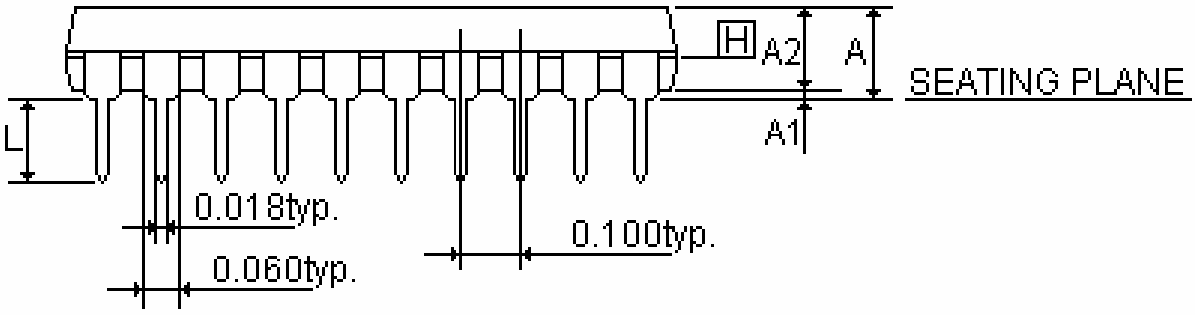
Writer V3.0 JP1 Pin Assignment

11.3 Programming Pin Mapping:

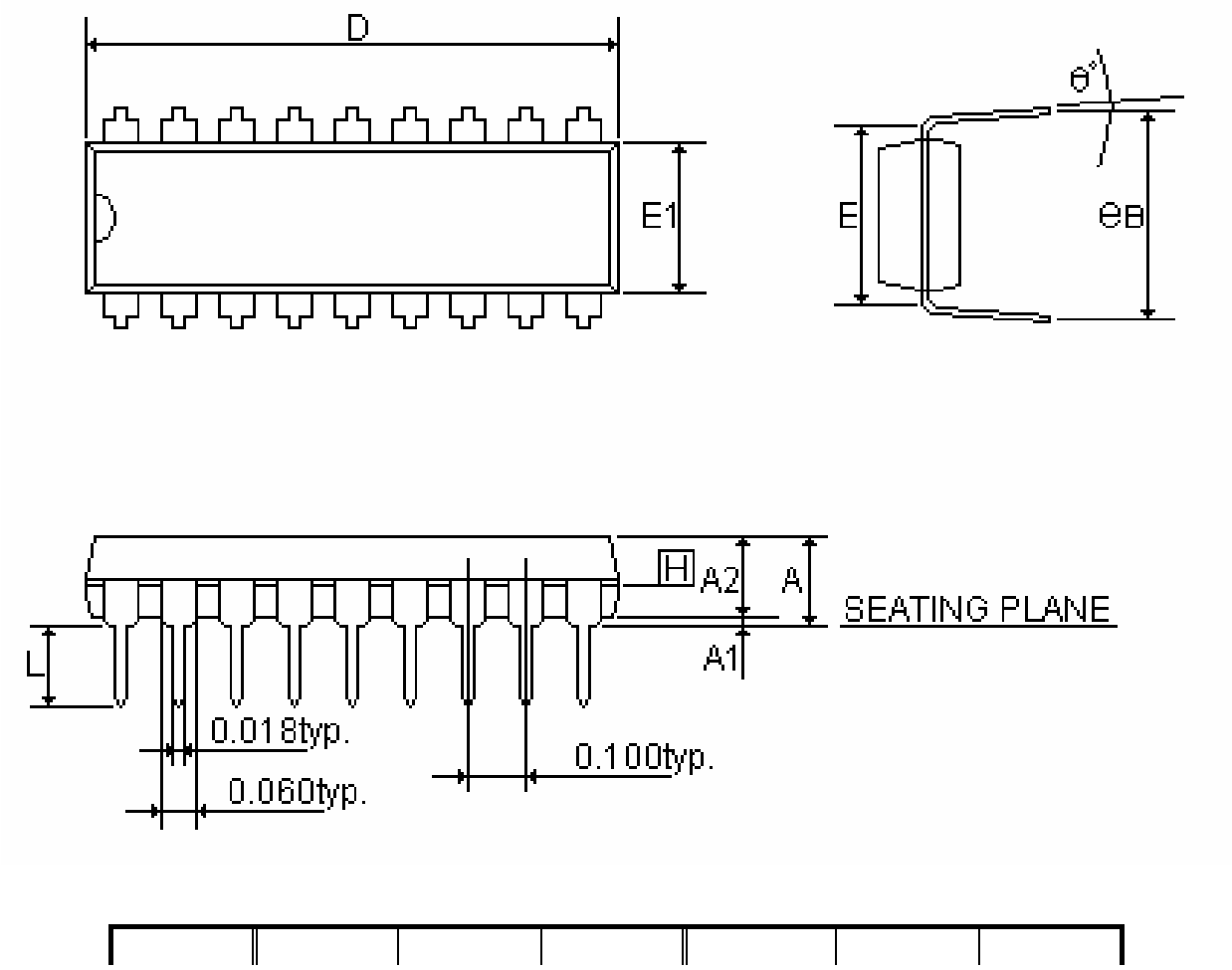
12 PACKAGE INFORMATION

12.1 P-DIP 20 PIN

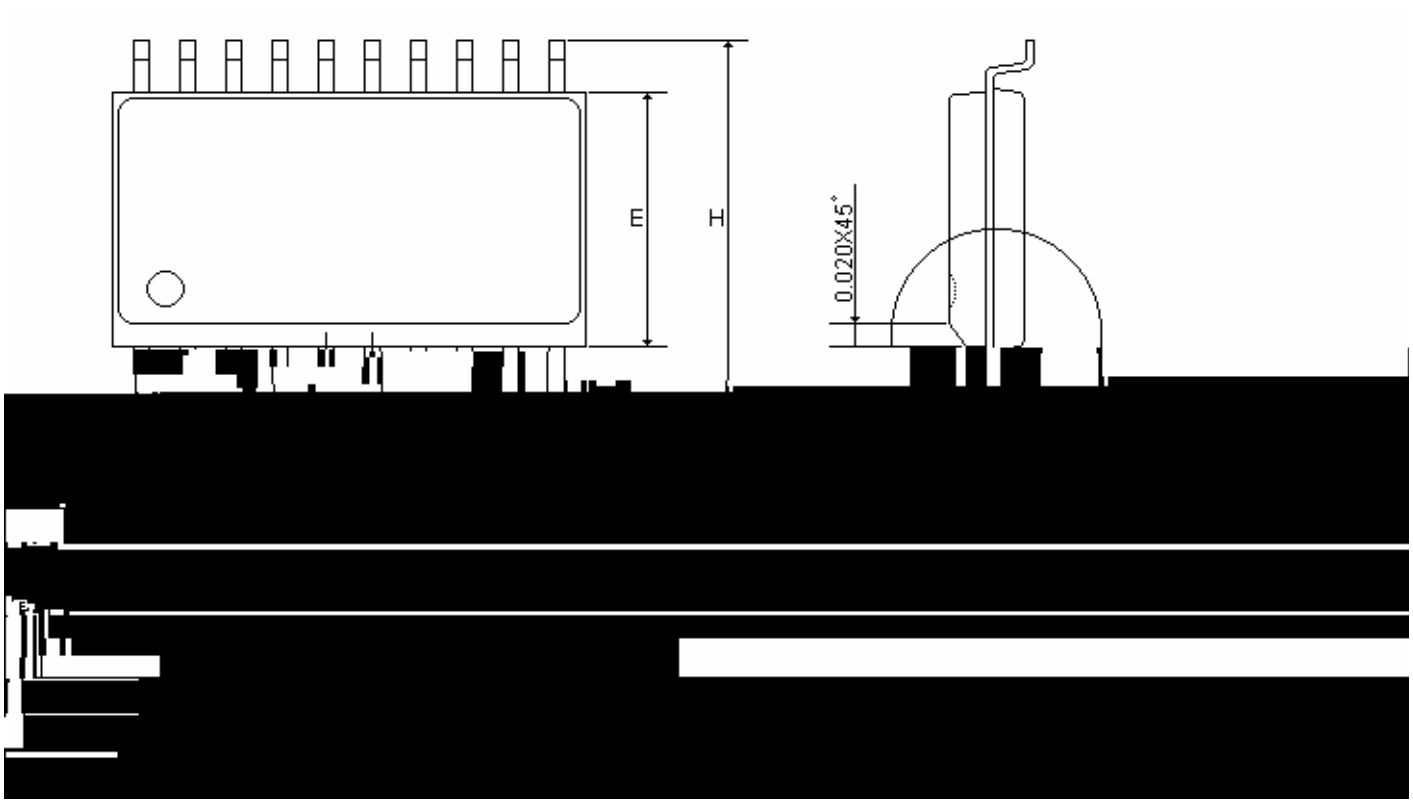




12.2 P-DIP 18 PIN

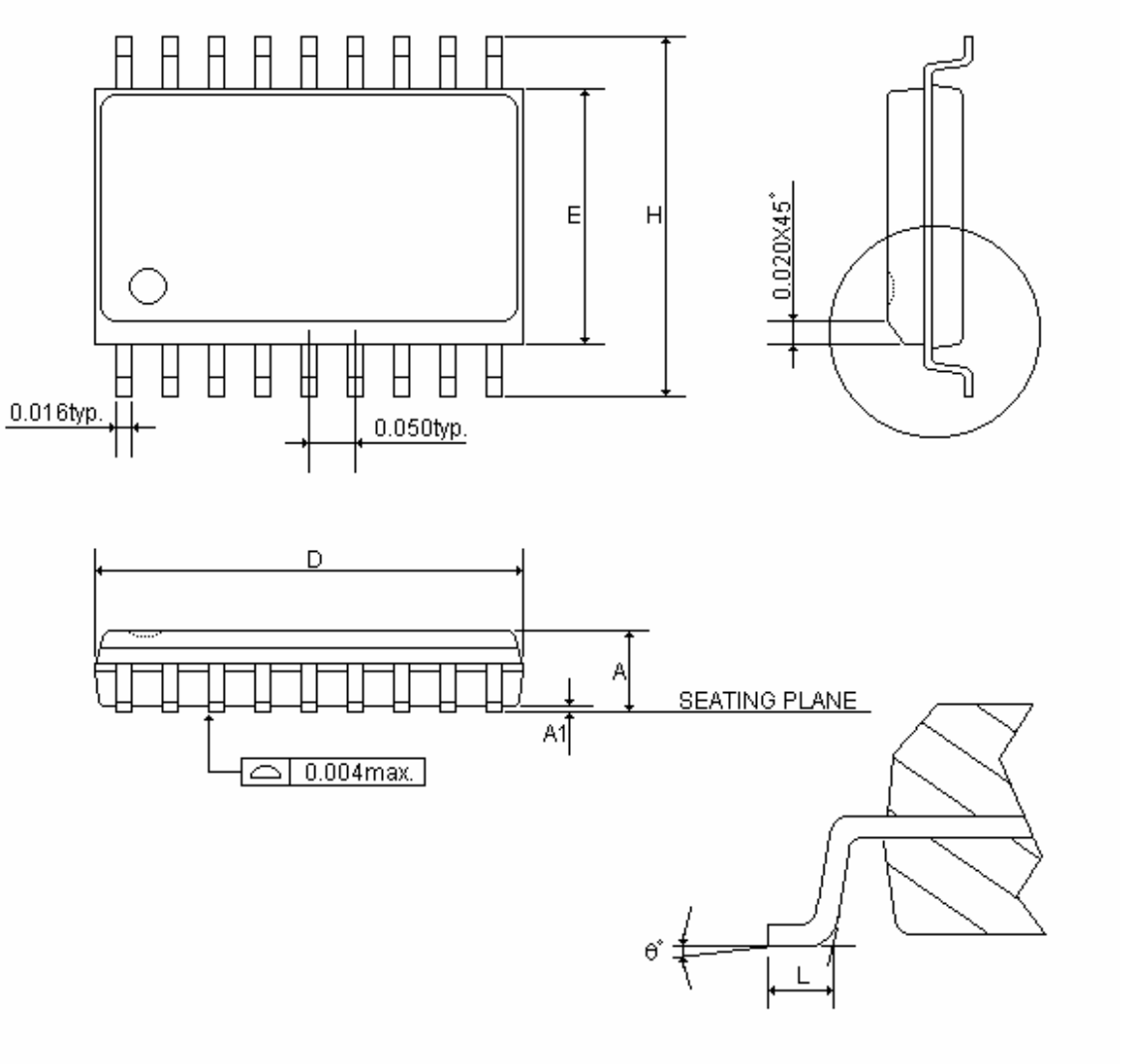


12.4 SOP 20 PIN

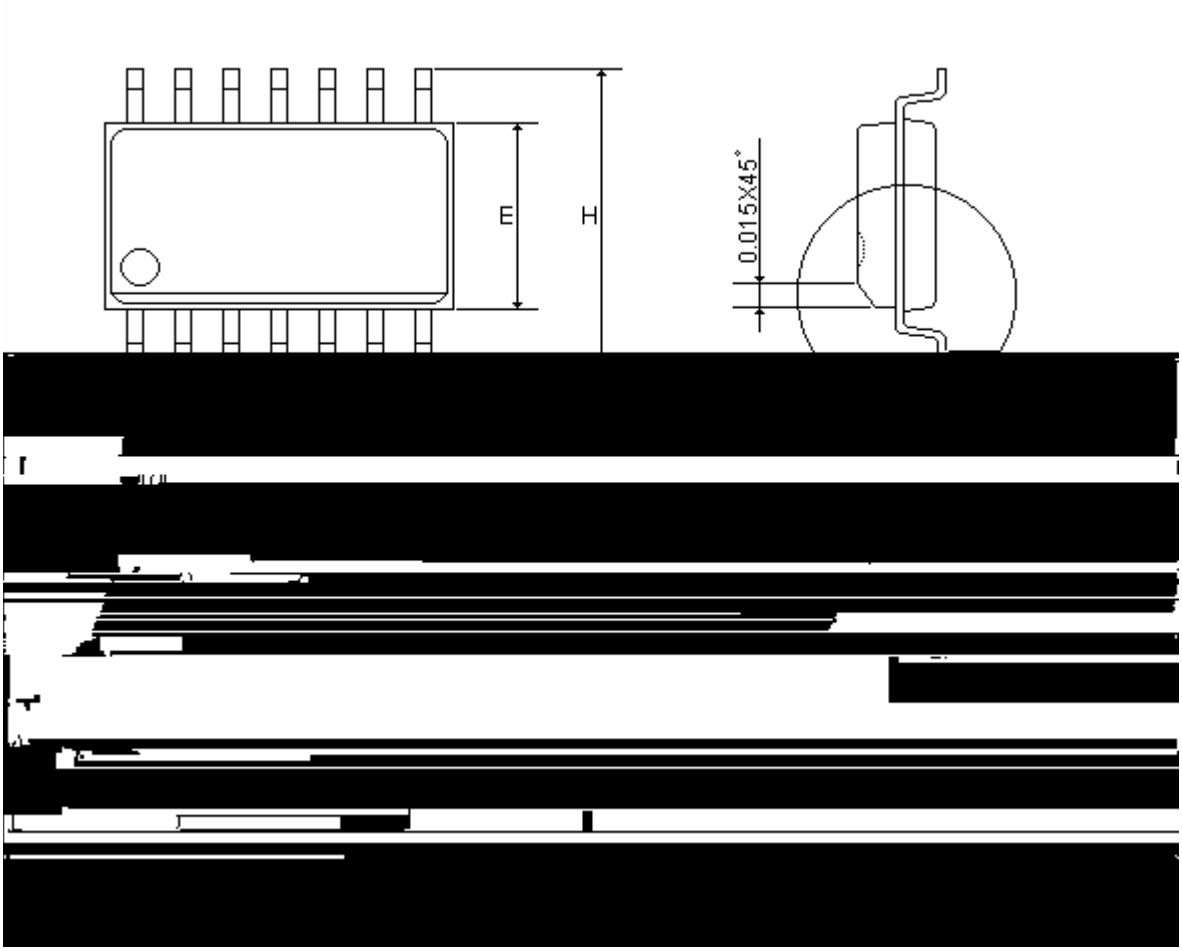


θ						

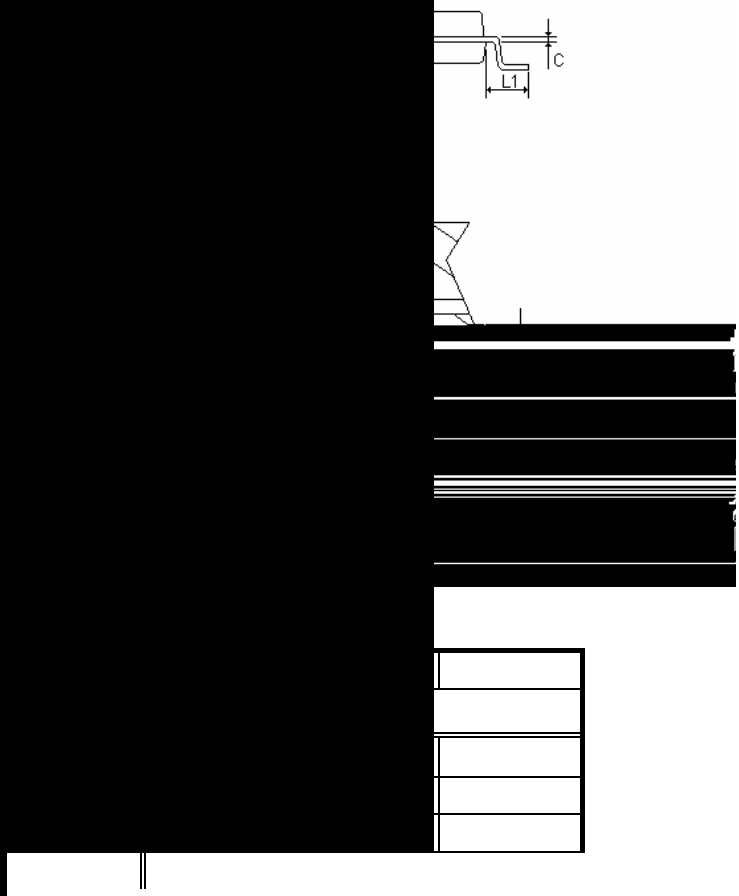
12.5 SOP 18 PIN



12.6 SOP 14 PIN



θ						

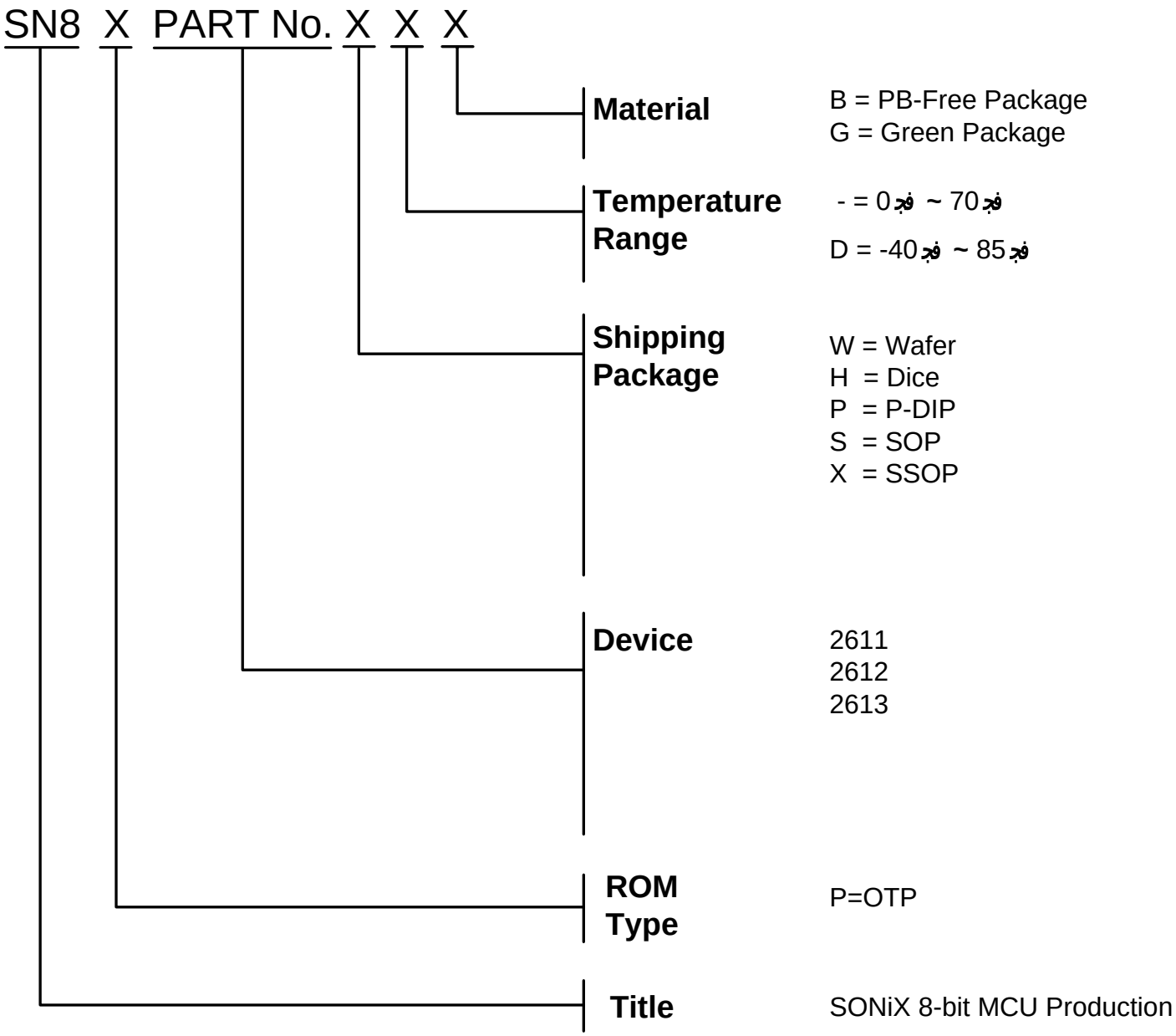

c
L1

13 Marking Definition

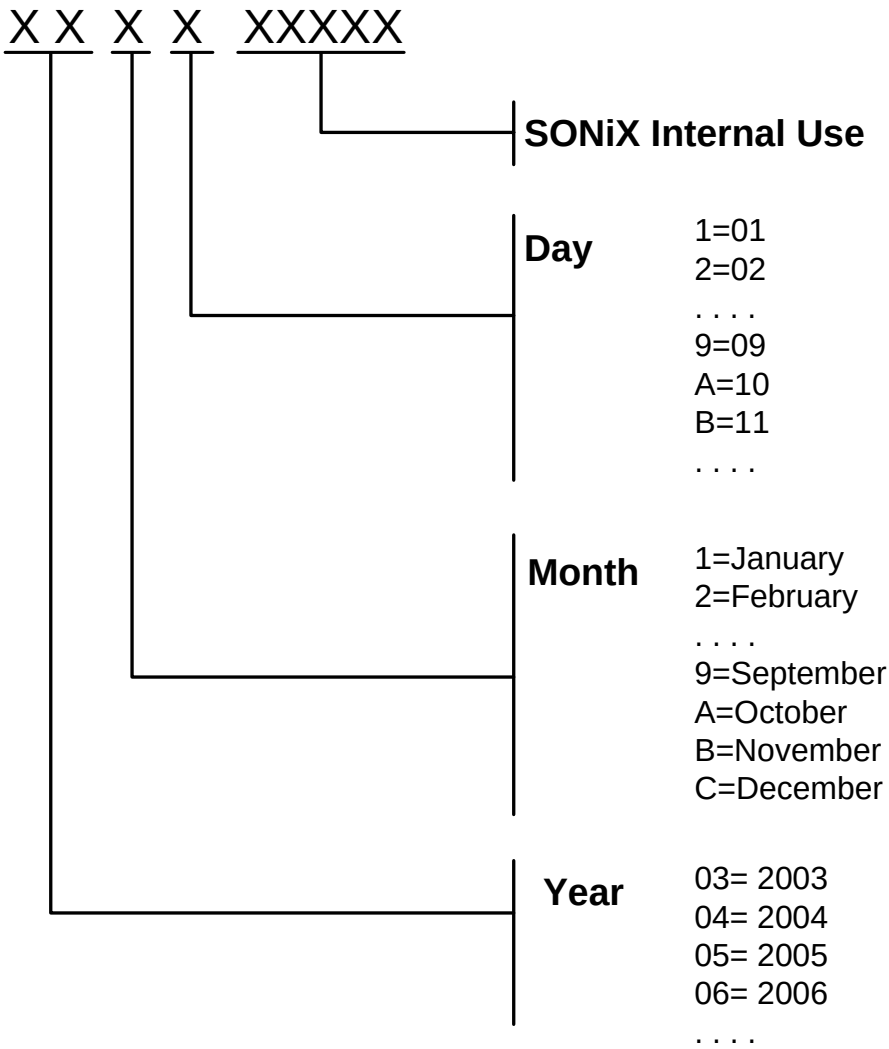
13.1 INTRODUCTION

There are many different types in Sonix 8-bit MCU production line. This note listed the production definition of all 8-bit MCU for order or obtain information. This definition is only for Blank OTP MCU.

13.2 MARKING INDETIFICATION SYSTEM



13.3 MARKING EXAMPLE

Name	ROM Type	Device	Package	Temperature	Material
SN8P2611PB	OTP	2611	P-DIP	0 ~70	PB-Free Package
SN8P2611SB	OTP	2611	SOP	0 ~70	PB-Free Package

13.4 DATECODE SYSTEM