# SN8P1604 PRODUCT SPECIFICATION

# SN8P1604 8-bit microcontroller

## 1. GENERAL DESCRIPTION

The SN8P1604 is an 8-bit micro-controller utilized with CMOS technology fabrication and featured with low power consumption and high performance by its unique electronic structure. This chip is designed with the excellent IC structure, including the program memory up to 4096-word OTP ROM, 128 bytes of the data memory, one 8-bit timer/event counter, watchdog timer, two interrupt sources (TC1, INT0), 22 I/O pins and 4 levels stack buffer. Besides, the user can choose desired oscillator configurations for the controller. There are four external oscillator configurations to select for generating system clock, including high-performing crystal, ceramic resonator, cost-saving RC and internal RC oscillator.

## 2. FEATURES

- ***Memory configuration***
  OTP ROM size : 4096 * 16 bits.
  RAM size : 128 * 8 bits.
- ***I/O pin configuration*** *(Total 22 pins)*
  One input pin with interrupt function.
  10 pins with wake-up function.
  Two intput/output port : 12 pins for general purpose.
- ***Built-in voltage detector for the Reset function***
- ***56 powerful instructions***
  All of instructions are 1 word with 1 or 2 cycles' execution.
  Execution time : 1 cycle uses 4 clocks of oscillator.
  All ROM area **JMP** instruction.
  All ROM area Subroutine **CALL** instruction.
  All ROM area lookup table function.(**MOVC** instruction)
- ***Two interrupt sources:***
  One internal interrupt : TC1
  One external interrupt : INT0
- ***Four levels stack buffer***
- An 8-bit timer/event counter.
  An 8-bit PWM output or one Buzzer output
- A watchdog timer.
- **Acceptable oscillator type**
  Crystal or ceramic resonator speed up to 20MHz
  RC oscillator type speed up to 10MHz
  Internal RC oscillator 16KHz
- ***Package :***
  SKDIP : 28
  SOP : 28

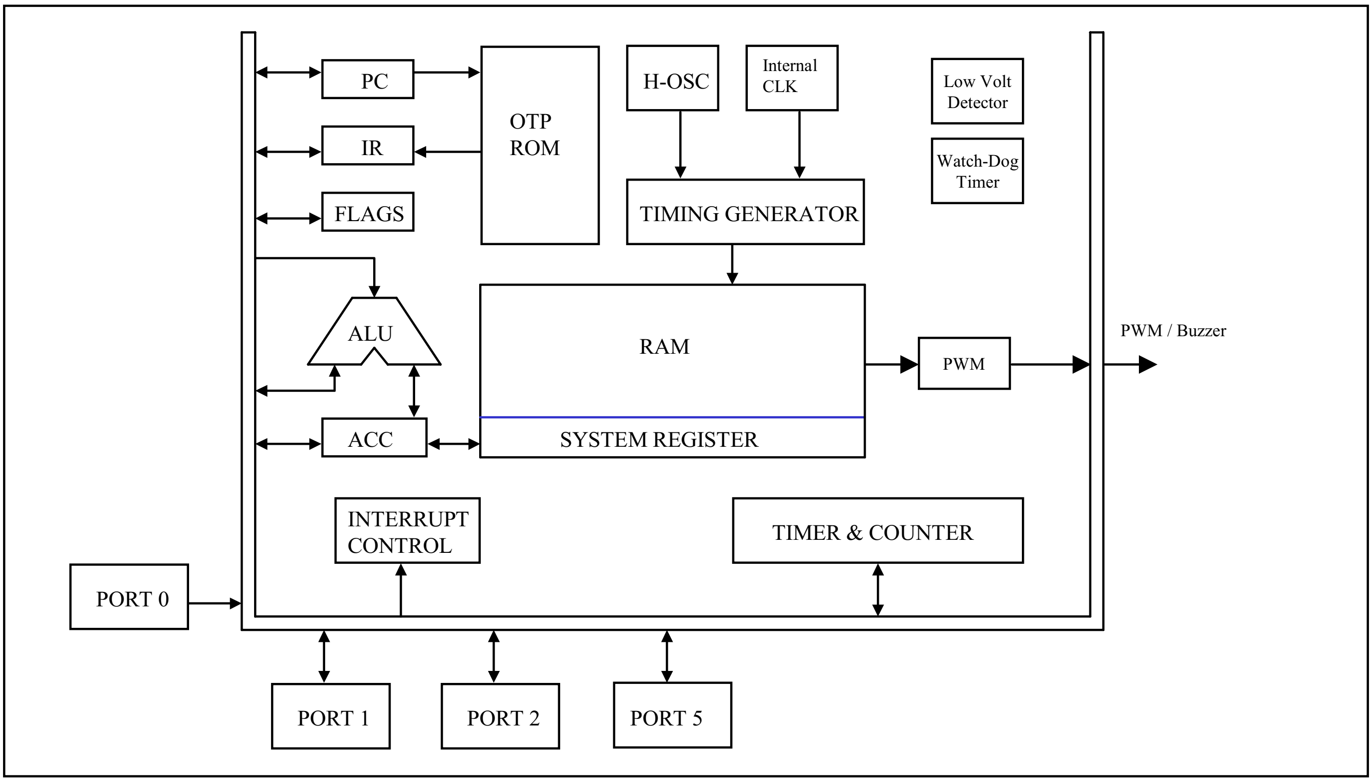
## 3. PIN ASSIGNMENT

| | | | | |
|---|---|---|---|---|
| P0.1 | 1 | U | 28 | RESET |
| VDD | 2 | | 27 | XIN |
| VPP | 3 | | 26 | XOUT/Fcpu |
| VSS | 4 | | 25 | P2.7 |
| P0.0/INT0 | 5 | | 24 | P2.6 |
| P5.0 | 6 | | 23 | P2.5 |
| P5.1 | 7 | | 22 | P2.4 |
| P5.2 | 8 | | 21 | P2.3 |
| P5.3/TC1/PWM1 | 9 | | 20 | P2.2 |
| P1.0 | 10 | | 19 | P2.1 |
| P1.1 | 11 | | 18 | P2.0 |
| P1.2 | 12 | | 17 | P1.7 |
| P1.3 | 13 | | 16 | P1.6 |
| P1.4 | 14 | | 15 | P1.5 |

SN8P1604K
SN8P1604S

## 4. BLOCK DIAGRAM

1604PA system block




## 5. PIN DESCRIPTION

| PIN NAME | TYPE | DESCRIPTION |
|---|---|---|
| VDD, VSS | P | Power supply input pins. |
| VPP/NC | I | During program op-code, this pin be pull to 12.5Vdc to reset internal address counter and to write data into OTP-ROM. This pin must be kept no connection during normal operation. |
| RST | I | System reset inputs pin. Schmitt trigger structure, active “low”, normal stay to “high”. |
| XIN | I | Oscillator input pin. |
| XOUT/Fcpu | I/O | Oscillator output pin. RC Mode as the Fcpu output |
| P0.0/INT0 | I | Port 0.0 and INT0 trigger pin with Schmitt trigger structure or wake-up from sleep mode |
| P0.1 | I | P0.1 with wake-up function. The P0.1 can be the clock input for the TC1 |
| P1.0 ~ P1.7 | I/O | Port 1.0 ~ Port 1.7 bi-direction pins with sleep mode wake-up function |
| P2.0 ~ P2.7 | I/O | Port 2.0 ~ Port 2.7 bi-direction pins. |
| P5.0 ~ P5.3 | I/O | P5.0 ~ P5.3 bi-direction pin, P5.3 as TC1 output for PWM and Buzzer function |

## 6. PROGRAM MEMORY (ROM)

The SN8P1604 provides the program memory up to 4096-word (4096 * 16 bits) to be addressed and is able to fetch instructions through 12-bit wide PC (Program Counter). It also can lookup ROM data by using ROM code registers (R, Y, Z). All of the program memory is partitioned into two coding areas, located from 000H to 00FH and from 010H to FFFH. The former area is assigned for executing interrupt vector. And the later area is for storing instruction's OP-code and lookup table's data. *The last location (FFFH) of OTP ROM had been reserved, it can not be used by programming.*

| | OTP ROM |
|---|---|
| 000h | Reset vector |
| 001h | |
| 002h | |
| 003h | |
| 004h | Reserved |
| 005h | " |
| 006h | " |
| 007h | " |
| 008h | Interrupt vector |
| 009h | . |
| 00Ah | . |
| . | . |
| . | . |
| 010h | General purpose area |
| . | . |
| . | . |
| . | . |
| . | . |
| . | . |
| . | . |
| FFEh | . |
| FFFh | Reserved |

## 7. DATA MEMORY (RAM)

The SN8P1604 has built-in 128 bytes memory location to store general purpose data and built-in special purpose memory to work as system registers. These memory locations are allocated in RAM bank 0, first 128-byte (00H ~ 7FH) shared for general data memory and last 128 bytes (80H ~ FFH) shared for system registers.

### 7.1 RAM BANK LOCATION

| | RAM location |
|---|---|
| 00h | General purpose area |
| 7Fh | End of Ram |
| 80h | System registers |
| | " |
| | " |
| | " |
| | " |
| FFh | " |

## 7.2 SYSTEM REGISTER ARRANGEMENT (BANK 0)

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | | | | | | | | | | | | | | | | |
| 1 | | | | | | | | | | | | | | | | |
| 8 | - | - | R | Z | Y | - | PFLAG | - | - | - | - | - | - | - | - | - |
| 9 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| A | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| B | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| C | P1W | P1M | P2M | - | - | P5M | - | - | INTRQ | INTEN | OSCM | - | - | - | PCL | PCH |
| D | P0 | P1 | P2 | - | - | P5 | - | - | - | - | - | - | TC1M | TC1C | TC1R | STKP |
| E | - | - | - | - | - | - | - | @YZ | - | - | - | - | - | - | - | - |
| F | - | - | - | - | - | - | - | | STK3 | STK3 | STK2 | STK2 | STK1 | STK1 | STK0 | STK0 |

PFLAG = ROM page and special flag register
R = Working register and ROM lookup data buffer
Y, Z = Working, @YZ and ROM addressing register
P0 ~ P5 = Port 0,1,2,5 data buffer
P1M ~ P5M = Port 1,2,5 input/output mode register
INTRQ = Interrupts' request register
INTEN = Interrupts' enable register
OSCM = Oscillator mode register
PCH, PCL = Program counter
TC1M = Timer/Event counter 1 mode register
TC1C = Timer/Event counter 1 counting register
STKP = Stack pointer buffer
STK0~STK3 = Stack 0 ~ stack 3 buffer
@YZ = RAM YZ indirect addressing index pointer

System register table

| Address | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 | R/W | Remarks |
|---|---|---|---|---|---|---|---|---|---|---|
| 080H | - | - | - | - | - | - | - | - | - | - |
| 081H | - | - | - | - | - | - | - | - | - | - |
| 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 |
| 085H | - | - | - | - | - | - | - | - | - | - |
| 086H | - | - | - | - | - | C | DC | Z | R/W | PFLAG |
| 087H | - | - | - | - | - | - | - | - | - | - |
| 0C0H | P17W | P16W | P15W | P14W | P13W | P12W | P11W | P10W | W | P1W wakeup register |
| 0C1H | P17M | P16M | P15M | P14M | P13M | P12M | P11M | P10M | W | P1M I/O direction |
| 0C2H | P27M | P26M | P25M | P24M | P23M | P22M | P21M | P20M | W | P2M I/O direction |
| 0C3H | - | - | - | - | - | - | - | - | - | - |
| 0C4H | - | - | - | - | - | - | - | - | - | - |
| 0C5H | - | - | - | - | P53M | P52M | P51M | P50M | W | P5M I/O direction |
| 0C6H | - | - | - | - | - | - | - | - | - | - |
| 0C7H | - | - | - | - | - | - | - | - | - | - |
| 0C8H | - | TC1IRQ | - | - | - | - | - | P00IRQ | R/W | INTRQ |
| 0C9H | - | TC1IEN | - | - | - | - | - | P00IEN | R/W | INTEN |
| 0CAH | WTCKS | WDRST | - | - | CPUM0 | CLKMD | STPHX | - | R/W | OSCM |
| 0CBH | - | - | - | - | - | - | - | - | - | - |
| 0CCH | - | - | - | - | - | - | - | - | - | - |
| 0CDH | - | - | - | - | - | - | - | - | - | - |
| 0CEH | PC7 | PC6 | PC5 | PC4 | PC3 | PC2 | PC1 | PC0 | R/W | PCL |
| 0CFH | - | - | - | - | PC11 | PC10 | PC9 | PC8 | R/W | PCH |

(To be continued)

System register table

| Address | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 | R/W | Remarks |
|---|---|---|---|---|---|---|---|---|---|---|
| 0D0H | - | - | - | - | - | - | P01 | P00 | R | P0 data buffer |
| 0D1H | P17 | P16 | P15 | P14 | P13 | P12 | P11 | P10 | R/W | P1 data buffer |
| 0D2H | P27 | P26 | P25 | P24 | P23 | P22 | P21 | P20 | R/W | P2 data buffer |
| 0D3H | - | - | - | - | - | - | - | - | - | - |
| 0D4H | - | - | - | - | - | - | - | - | - | - |
| 0D5H | - | - | - | - | P53 | P52 | P51 | P50 | R/W | P5 data buffer |
| 0D6H | - | - | - | - | - | - | - | - | - | - |
| 0D7H | - | - | - | - | - | - | - | - | - | - |
| 0D8H | - | - | - | - | - | - | - | - | - | - |
| 0D9H | - | - | - | - | - | - | - | - | - | - |
| 0DAH | - | - | - | - | - | - | - | - | - | - |
| 0DBH | - | - | - | - | - | - | - | - | - | - |
| 0DCH | TC1ENB | TC1rate2 | TC1rate1 | TC1rate0 | TC1CKS | ALOAD1 | TC1OUT | PWM1OUT | R/W | TC1M |
| 0DDH | TC1C7 | TC1C6 | TC1C5 | TC1C4 | TC1C3 | TC1C2 | TC1C1 | TC1C0 | R/W | TC1C |
| 0DEH | TC1R7 | TC1R6 | TC1R5 | TC1R4 | TC1R3 | TC1R2 | TC1R1 | TC1R0 | W | TC1R |
| 0DFH | GIE | - | - | - | - | STKPB2 | STKPB1 | STKPB0 | R/W | STKP stack pointer |
| 0E0H | - | - | - | - | - | - | - | - | - | - |
| 0E1H | - | - | - | - | - | - | - | - | - | - |
| 0E2H | - | - | - | - | - | - | - | - | - | - |
| 0E3H | - | - | - | - | - | - | - | - | - | - |
| 0E4H | - | - | - | - | - | - | - | - | - | - |
| 0E5H | - | - | - | - | - | - | - | - | - | - |
| 0E6H | - | - | - | - | - | - | - | - | - | - |
| 0E7H | @YZ7 | @YZ6 | @YZ5 | @YZ4 | @YZ3 | @YZ2 | @YZ1 | @YZ0 | R/W | @YZ index pointer |
| 0F0H | - | - | - | - | - | - | - | - | - | - |
| 0F1H | - | - | - | - | - | - | - | - | - | - |
| 0F2H | - | - | - | - | - | - | - | - | - | - |
| 0F3H | - | - | - | - | - | - | - | - | - | - |
| 0F4H | - | - | - | - | - | - | - | - | - | - |
| 0F5H | - | - | - | - | - | - | - | - | - | - |
| 0F6H | - | - | - | - | - | - | - | - | - | - |
| 0F7H | - | - | - | - | - | - | - | - | - | - |
| 0F8H | S3PC7 | S3PC6 | S3PC5 | S3PC4 | S3PC3 | S3PC2 | S3PC1 | S3PC0 | - | STK3L |
| 0F9H | - | - | - | - | - | - | S3PC9 | S3PC8 | - | STK3H |
| 0FAH | S2PC7 | S2PC6 | S2PC5 | S2PC4 | S2PC3 | S2PC2 | S2PC1 | S2PC0 | - | STK2L |
| 0FBH | - | - | - | - | - | - | S2PC9 | S2PC8 | - | STK2H |
| 0FCH | S1PC7 | S1PC6 | S1PC5 | S1PC4 | S1PC3 | S1PC2 | S1PC1 | S1PC0 | - | STK1L |
| 0FDH | - | - | - | - | - | - | S1PC9 | S1PC8 | - | STK1H |
| 0FEH | S0PC7 | S0PC6 | S0PC5 | S0PC4 | S0PC3 | S0PC2 | S0PC1 | S0PC0 | - | STK0L |
| 0FFH | - | - | - | - | - | - | S0PC9 | S0PC8 | - | STK0H |

Note :

a). All of register name had been declared in SN8ASM assembler.

b). One-bit name had been declared in SN8ASM assembler with “F” prefix code.

c). It will get a logic “H” data, when use instruction to check empty location.

## 8. ACCUMULATOR

The ACC is an 8-bits 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.

PFLAG initial value = xxxx xxxx

| | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| PFLAG | - | - | - | - | - | C | DC | Z |

### 8.1 CARRY FLAG

C = 1 : If executed arithmetic addition with occurring carry signal or executed arithmetic subtraction without borrowing signal or executed rotation instruction with shifting out logic "1".
C = 0 : If executed arithmetic addition without occurring carry signal or executed arithmetic subtraction with borrowing signal or executed rotation instruction with shifting out logic "0".

### 8.2 DECIMAL CARRY FLAG

DC = 1 : If executed arithmetic addition with occurring signal from low nibble or executed arithmetic subtraction without borrow signal from high nibble.
DC = 0 : If executed arithmetic addition without occurring signal from low nibble or executed arithmetic subtraction with borrow signal from high nibble.

### 8.3 ZERO FLAG

Z = 1 : After operation, the content of ACC is zero.
Z = 0 : After operation, the content of ACC is not zero.

## 9. WORKING REGISTERS

The locations 82H to 86H of RAM bank 0 in data memory stores the specially defined registers such as register R, Y, Z and PFLAG, respectively shown in the following table. These registers can use as the general purpose of working buffer and can also be used to access ROM's and RAM's data. For instance, all of the ROM's table can be looked-up with R, Y and Z registers. And the data of RAM memory can be indirectly accessed with Y and Z registers.

| RAM | 80H | 81H | 82H | 83H | 84H | 85H | 86H | 87H |
|---|---|---|---|---|---|---|---|---|
| | - | - | R | Z | Y | - | PFLAG | - |

### 9.1 Y, Z REGISTERS

The Y and Z registers are the 8-bit buffers. There are three major functions of these registers. First, Y and Z registers can be used as working registers. Second, these two registers can be used as data pointers for @YZ register. Third, the registers can address ROM location in order to lookup ROM data.

Example: If want to read a data from RAM address 20H of bank_0, it can use indirectly addressing mode to access data as following

```
B0MOV   Y,#00H          ; To set RAM bank 0 for Y register
B0MOV   Z,#20H          ; To set location 20H for Z register
B0MOV   A,@YZ           ; To read a data into ACC
```

### 9.2 LOOK-UP TABLE

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

Example: To lookup ROM's data from location table_1

```
                B0MOV   Y, #TABLE1$M      ; To set lookup table's middle address.
                B0MOV   Z, #TABLE1$L      ; To set lookup table's low address.
                MOVC                      ; To lookup data,   R = 01H,   ACC = 35H
                .                         ;
                INCMS   Z                 ; To lookup next ROM's data
                NOP                       ;
                MOVC                      ; To lookup data,   R = 51H,   ACC = 05H
                .       .                 ;
                .       .                 ;
TABLE1:         DW      0135H             ; To define a word (16 bits) data
                DW      5105H             ;   "
                DW      2012H             ;   "
                .       .
```

Note : The high-byte data of ROM can't be "**00**xxh".

## 9.3 ADDRESSING MODE

The SN8P1604 provides three addressing modes to access RAM data, including immediate mode, direct mode and indirect mode. The main purpose of these three different modes are described in the following table. The immediate addressing mode uses an immediate data to set up the location in (MOV A,#I, B0MOV M,#I) in ACC or specific RAM location. The directly addressing mode uses address number to access memory location (MOV A,12H, MOV 12H,A). The indirectly addressing mode is set an address in data pointer registers (Y/Z) and uses MOV instruction to read/write data between ACC and @YZ register (MOV A,@YZ, MOV @YZ,A).

Immediate addressing mode

```
        MOV     A,#12H      ; To set an immediate data 12H into ACC
        B0MOV   R,#28H      ; To set an immediate data 28H into R register
```

Directly addressing mode

```
        B0MOV   A,12H       ; To get a content of location 12H of bank 0 and save in ACC
```

Indirectly addressing mode with @YZ register

```
        CLR     Y           ; To clear Y register
        B0MOV   Z,#16H      ; To set an immediate data 16H into Z register
        B0MOV   A,@YZ       ; Use data pointer @YZ reads a data from RAM location 016H into ACC
```

## 10. PROGRAM COUNTER

The program counter (PC) is an 12-bit binary counter separated into the high-byte 4 bits 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 11.

| | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| PCH | - | - | - | - | PC11 | PC10 | PC9 | PC8 |

| | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| PCL | PC7 | PC6 | PC5 | PC4 | PC3 | PC2 | PC1 | PC0 |

PC Initial value = xxxx 0000 0000 0000

| | 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 | - | - | - | - | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 10.1 ONE ADDRESS SKIPPING

If the condition of bit test instruction is match, the PC will add 2 steps to skip next instruction.

```
            B0BTS1  FC              ; To skip, if Carry_flag = 1
            JMP     C0STEP          ;    else jump to C0STEP.
            .
            .
C0STEP:     NOP
```

### 10.2 MULTI-ADDRESS JUMPING

Users can jump round multi-address by either JMP instruction or ADD M,A instruction (M = PCL) to activate multi-address jumping function. If carry signal occurs after execution of ADD PCL,A, the carry signal will not affect PCH register.

Example : If PC = 0023H (PCH = 00H 、PCL = 23H)

```
;PC = 0023H
            MOV     A,#28H
            B0MOV   PCL,A           ; Jump to address 0328H
            .       .
;PC = 0028H .       .
            MOV     A,#00H
            B0MOV   PCL,A           ; Jump to address 0300H
```

Example : If PC = 0023H (PCH = 00H 、PCL = 23H)

```
;PC = 0023H
            B0ADD   PCL,A           ; PCL = PCL + ACC, the PCH can not be changed.
            JMP     A0POINT         ; If 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
            .       .               ;
            .       .               ;
```

## 11. STACK BUFFER

The stack buffer has up to 4-level areas and each level is 12-bit length. This buffer is designed to store PC’s value while the interrupt service or call subroutine is executed. The STKP register is a pointer designed to point active level in order for kernel circuit to push or pop up PC’s data from stack buffer. The STKP will decrease one level after the data is pushed into stack buffer and increase one level before data is popped up from stack buffer. Once interrupt occurs, the global interrupts will turn the enable bit (GIE) of STKP to be disable. And GIE will turn to be enable again, after RETI instruction is executed.

STKP initial value = 0xxx x111

| | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| STKP | GIE | - | - | - | - | STKPB2 | STKPB1 | STKPB0 |

STKn initial value = xxxx xxxx xxxx xxxx, STKn = STKnH + STKnL (n = 7H ~ 4H)

| | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| STKnH | - | - | - | - | SnPC11 | SnPC10 | SnPC9 | SnPC8 |

| | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| STKnL | SnPC7 | SnPC6 | SnPC5 | SnPC4 | SnPC3 | SnPC2 | SnPC1 | SnPC0 |

### 11.1 ACC & WORKING REGISTERS PROTECTION

The SN8P1604 does not push ACC and working registers into stack buffer during interrupt execution. Thus, once interrupt occurs, these data must be stored in the data memory based on the user's program as follows:

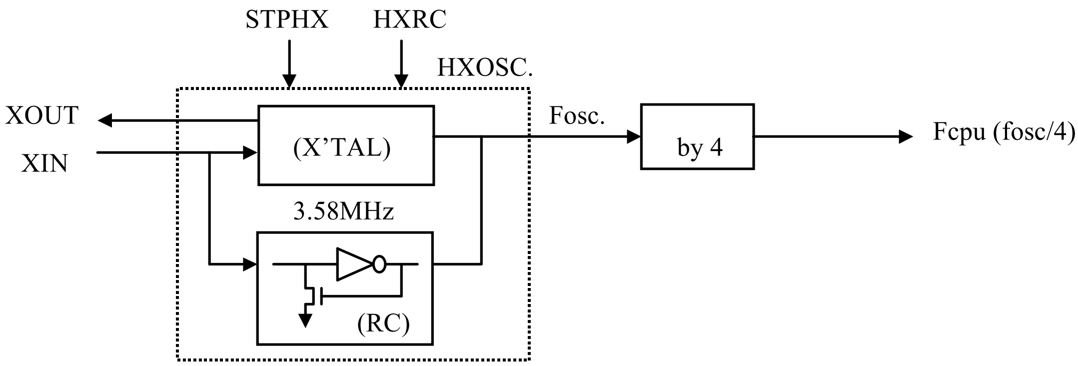
```
; To declare variables in source program
ACCBUF      EQU     00H             ; To declare ACC_buffer at 00H in bank 0
PFLAGBUF    EQU     07H             ; To declare Page_flags_buffer at 07H in bank 0

Example : To push ACC and working registers
PUSHBUF:    B0MOV   ACCBUF,A        ; To push ACC into ACC_buffer
            B0MOV   A,PFLAG         ; A ← PFLAG
            B0MOV   PFLAGBUF,A      ; To push PFLAG into PFLAGBUF

Example : To pop ACC and working registers
POPBUF:     B0MOV   A,PFLAGBUF      ; To pop PFLAG register
            B0MOV   PFLAG,A         ;
            B0XCH   A,ACCBUF        ; To pop ACC
```

## 12. OSCILLATOR

The SN8P1604 has both the internal and external oscillator which can be RC, crystal, ceramic resonator and internal clock to generate system clock source. The user can select desired one of them to be the oscillator configuration of the chip. The chip featured with low power consumption by using its power down mode or internal low clock mode. The SN8P1604 will switch the system between normal mode to internal slow clock mode by setting CLKMD = 1. The chip can be awakened from the power down mode into normal mode by triggering Port 0 and Port 1. After the system wakeups, the STPHX bit and the CPUM0 bit will reset to "0", and turn on both the internal low clock and the external clock automatically.




* HXRC is code option, 0 = crystal or resonator, 1 = RC

### 12.1 OSCM REGISTER

OSCM initial value = 0XXX 000X

| | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| OSCM | WTCKS | WDRST | - | - | CPUM0 | CLKMD | STPHX | - |

WTCKS: Watch-dog clock source select bit. 0 = fcpu , 1 = internal RC low clock.
CLKMD: High/Low speed mode selects bit. 0 = normal (dual) mode, 1 = internal RC mode.
STPHX : To stop high-speed oscillator controls bit. 0 = free run, 1 = stop.
CPUM0: CPU operating mode control bit. 0 = operating, 1 = sleep (power down), turn off both the hi-clock and low-clock

The oscillator's warm up time can be set by the SWARM Code option of OSC Code option , 01(Low Speed 32Khz) = 13$^{th}$ , other = 18$^{th}$

*Note : To recommend execution a NOP instruction after changing CPU operating mode.*

Example : To switch cpu operating from the normal operation mode to the power down mode.

```
N2SLEEP:                                    ; The warm up time must be set by the code(mask) option for wakeup.
            B0BSET    CPUM0                 ; Turn off both the internal RC clock and the external clock
            NOP                             ;
```

## *12.2 Oscillator Option*

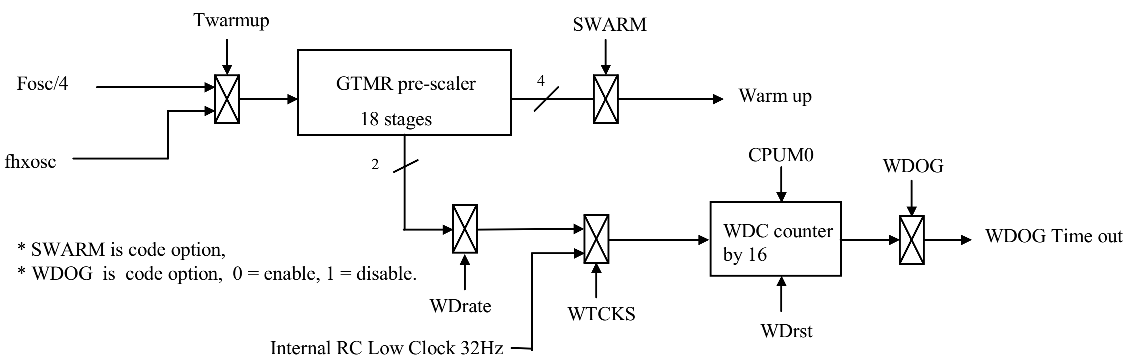
- Code Option content (Setting in the Assembler software)

**The SN8P1604 can be operated in four different oscillator modes. The user can program two configuration bits (FOSC<1:0>) to select one of these four modes:**

| | | |
|---|---|---|
| **• RC** | **00** | |
| **• Low Speed X'tal 32Khz ~ 200Khz** | **01** | **affect the warm-up and watch-dog** |
| **• High Speed X'tal 12Mhz** | **10** | |
| **• 4Mhz X'tal** | **11** | |

## *13.GTMR PRESCALER*

The GTMR pre-scaler is an 18-bit binary up counter designed to count a precision timing for watchdog timer and HX oscillator warm up time applications. There are two clock sources for GTMR pre-scaler selection. One is Fosc/4 for all of function operation and the other is fhxosc for operating CPU warm up.




## *13.1 WARMUP TIME*

There are two type warm-up timer designed for the different speed of oscillator. In actual application, user can set SWARM control bit in the Code Option. The SWARM control bit sets the Wdrate .

Example 1: Set the Swarm bit = 0 for the High Speed Oscillator, --> the Wdrate = 0

The Swarm bit = 0 --> the Warm-up time is $1/(fhxosc \div 2^{18)}) = 73.2mS$ @ 3.58 MHz
The Wdrate bit =0 --> the Watch-Dog timer is count by $1/(fcpu \div 2^{14} \div 16) = 293mS$ @ 3.58 MHz

Example 2 : Set the Swarm bit = 1 for the Low Speed Oscillator, --> the Wdrate = 1

The Swarm bit = 0 --> the Warm-up time is $1/(fhxosc \div 2^{13)}) = 250mS$ @32768 Hz
The Wdrate bit =0 --> the Watch-Dog timer is count by $1/(fcpu \div 2^{8} \div 16) = 0.5S$ @32768 Hz

Example 3 :

| HX_osc | SWARM | Warm up time |
|---|---|---|
| 3.58 MHz | 0 | $1/(\text{fhxosc} \div 2^{18}) = 73.2$ mS |
| 32768Hz | 1 | $1/(\text{fhxosc} \div 2^{13}) = 250$ ms |

### 13.2WAKEUP TIME

After power on, oscillator changes stopping status to running status. System needs 2048 oscillator clocks to detect oscillator stable. The period is called wake-up time. System can work in normal mode after wake-up time over.

***wake-up time = 1/Fosc * 2048 (sec)***

Example : 1. The wakeup time of P0, P1 wakeup function is as following. System will enter normal mode from power down mode.

@3.58MHz wake-up time = 1/Fosc * 2048 = 0.57 mS
@32768 wake-up time = 1/Fosc * 2048 = 62.5 mS

Example : 2. System is in slow mode (internal low clock status). The external oscillator stops (STPHX = 1). If system will go to normal mode, users have to make a delay routine by programming. The delay time is equal to 1/Fosc * 2048. External oscillator frequency is 3.58MHz. The routine is as following.

```
; Slow mode to normal mode routine

        B0BCLR      FSTPHX          ; Enable external oscillator (high clock)

        NOP                         ; Wakeup time routine
        NOP
        NOP

        B0BCLR      FCLKMD          ; Swtich to normal mode (high speed mode)
        .           .
        .           .
        .           .
```

Wake-up time of @3.58MHz is equal to 0.57ms. One instruction cycle of internal low clock (16KHz) is equal to 0.25ms. The wake-up time is about 3 instruction cycles of internal low clock. That is 3 **NOP**s.

### 13.3 WATCH DOG (WDOG) TIMER

The watchdog timer (WDTMR) is a 4-bit binary up counter designed for monitoring program execution. If the program is operated into the unknown status by noise interference, WDC's overflow signal will reset this chip to restart operation. In normal operation flow, the user must insert an instruction before overflow occurs to reset WDC timer to prevent the program from unexpected system reset. In order to generate different output timings, the user can control WDC by modifying WDrate bits of the Code_Option . This timer will be disabled at sleep modes.

OSCM initial value = 0XXX 000X

| | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| OSCM | WTCKS | WDRST | - | - | CPUM0 | CLKMD | STPHX | - |

WDRST : Watch dog timer reset bit. 0 = Non reset, 1 = clear the watchdog timer's counter
WTCKS: Watch-dog clock source select bit 0 : fcpu , 1: internal RC low clock (32Hz)

Example :

| Wdrate | Watchdog overflow time |
|---|---|
| 0 | $1/(fcpu \div 2^{14} \div 16) = 293$ ms @3.58 MHz |
| 0 | $1/(fcpu \div 2^{14} \div 16) = 64$ S @16384Hz |
| 1 | $1/(fcpu \div 2^{8} \div 16) = 4.5$ ms @3.58MHz |
| 1 | $1/(fcpu \div 2^{8} \div 16) = 1$S @16384Hz |
| Watch-dog clock is internal RC low clock | $1/(32 \div 16) = 0.5$S @32Hz |

Note : The watch dog timer can be enabled and be disabled by mask option.
The Wdrate can be set by the code option.

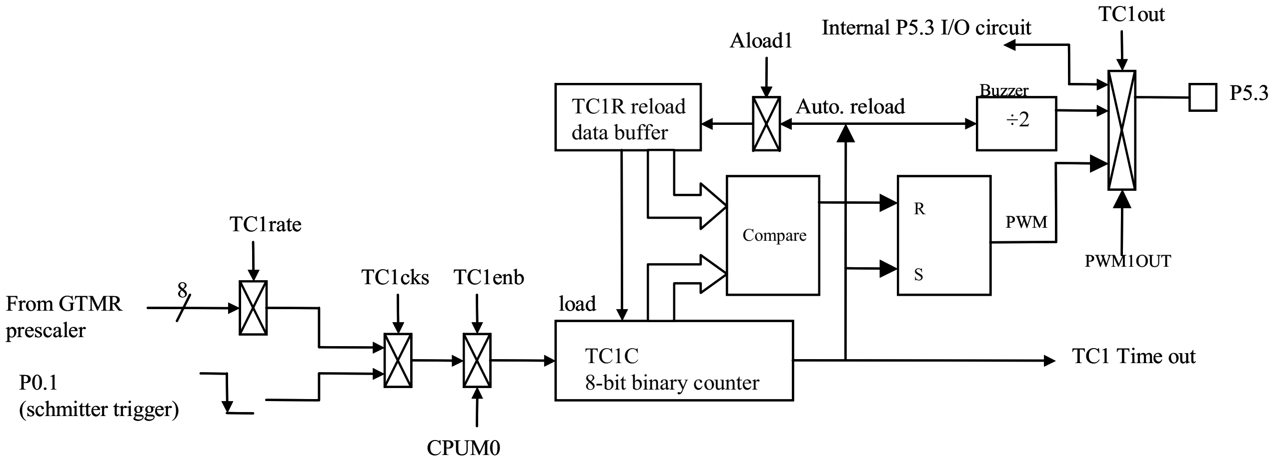
An operation of watch-dog timer is as follows:

```
Main:
        B0BSET      FWDRST          ; Clear the watchdog timer's counter
        Call        sub1
        Call        sub2
        XXX
        Jmp         main
```

## 14. TIMER/EVENT COUNTER (TC1)

Timer/Event Counter 1 (TC1) is used to count system ‘event’ by identifying the transition (high-to-low) of incoming square wave signals. To indicate that an event has occurred, or that a specified time interval has elapsed, TC1 generates an interrupt request. By counting signal transitions and comparing the current counter value with the reference register value, TC1 can be used to measure specific time intervals.

TC1 has a auto re-loadable counter that consists of two parts: an 8-bit reload register (TC1R) into which you write the counter reference value, and an 8-bit counter register (TC1C) whose value is automatically incremented by counter logic.




## TC1 Function Summary

| | |
|---|---|
| 8-bit programmable timer | Generates interrupts at specific time intervals based on the selected clock frequency. |
| External event counter | Count various system “events” based on edge detection of external clock signals at the P0.1 input pin. |
| Arbitrary frequency output | Outputs selectable clock frequencies to the TC1 output pin, TC1OUT |
| External signal divider | Divides the frequency of an incoming external clock signal according to a modifiable reference value (TC1R), and outputs the modified frequency to the TC1OUT |
| PWM function | PWM output can be generated by the PWM1OUT bit and output to TC1OUT. |

### 14.1 TC1M MODE REGISTER

TC1M initial value = 0xxx xxxx

| | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| TC1M | TC1ENB | TC1RATE2 | TC1RATE1 | TC1RATE0 | TC1CKS | ALOAD1 | TC1OUT | PWM1OUT |

TC1ENB : TC1 counter/BUZZER1/PWM1OUT enable bit. 0 = disable, 1 = enable.
TC1RATE : TC1 internal clock select bits. 000 = fcpu/256, 001 = fcpu/128, … , 110 = fcpu/4, 111 = fcpu/2.
TC1CKS : TC1 clock source select bit. 0 = Internal clock source, 1 = External clock source (INTP0.1).
ALOAD : Auto-reload control bit. 0 = none auto-reload, 1 = auto-reload.
TC1OUT : TC1 time-out toggle signal output control bit.
  0 = To disable TC1 signal output and to enable P5.3’s I/O function,
  1 = To enable TC1’s signal output and to disable P5.3’s I/O function. ( Auto-disable the PWM1OUT fun.)
PWM1OUT : PWM output control bit
  0 = To disable the PWM output
  1 = To enable the PWM output (The TC1OUT control bit must = 0 )

## 14.2 TC1C COUNTING REGISTER

TC1C initial value = xxxx xxxx

| | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| TC1C | x | x | x | x | x | x | x | x |

## 14.3 TC1R AUTO-LOAD REGISTER

TC1R initial value = xxxx xxxx

| | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| TC1R | x | x | x | x | x | x | x | x |

Note 1: The initial value of TC1C register is calculated as follows

***TC1C initial value = 256 - (TC1 interrupt interval time * input clock)***

Note 2: The TC1 timer must be disabled to modify TC1C's value.

## 14.4 PWM1 FUNCTION DESCRIPTION

The 8-bit counter counts modulus 256, that is, from 0-255, inclusive. The value of the 8-bit counter is compared to the contents of the reference register, RPWM1. When the reference register value equals the counter value, the PWM output goes low. When the counter reaches zero, the PWM output is forced high. The low-to-high ratio (duty) of the PWM output is RPWM1/256.

All PWM outputs remain inactive during the first 256 input clock signals. Then, when the counter value changes from FFH back to 00H, the PWM outputs are forced to high level. The pulse width ratio (duty cycle) is defined by the contents of the reference register and is programmed in increments of 1:256. The 8-bit PWM data register RPWM1 is read and written using 8-bit RAM control instruction only.

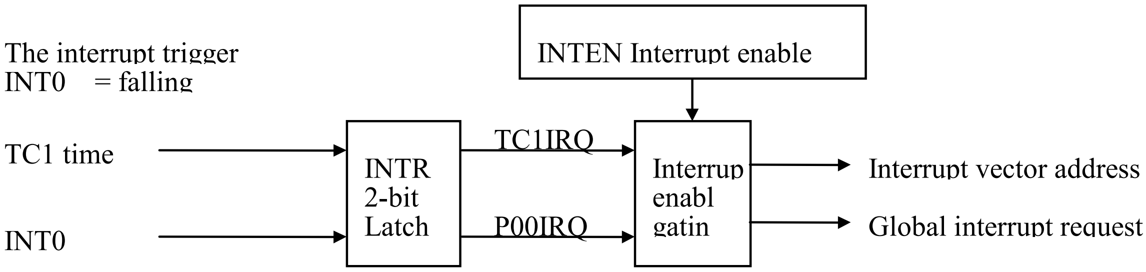
PWM output can be held at low level by continuously loading the reference register with 00H. By continuously loading the reference register with FFH, you can hold the PWM output to high level, except for the last pulse of the clock source, which sends the output low.

| Reference Register Value (RPWM1) | Duty |
|---|---|
| 0000 0000 | 0/256 |
| 0000 0001 | 1/256 |
| 0000 0010 | 2/256 |
| --- | --- |
| --- | --- |
| 1000 0000 | 128/256 |
| 1000 0001 | 129/256 |
| --- | --- |
| --- | --- |
| 1111 1110 | 254/256 |
| 1111 1111 | 255/256 |

Note : RPWM1 value is stored in TC1R register

## 15. INTERRUPT

The SN8P1604 provides two interrupt sources, including one internal interrupt (TC1) and one external interrupt (INT0). The external interrupts can warm up the chip while the system is switched from power-down to high-speed mode. The external clock input pins of TC1 is shared with P0.1 pin. Beside this, P0.0 pin can work with warm up function. Once interrupt service is executed, the GIE bit in STKP register will clear to "0" for stopping other interrupt request. On the contrast, when interrupt service exits, this bit will set to "1" to accept the next interrupts' request. All of the interrupt request signals are stored in INTRQ register. The user can program the chip to check INTRQ's content for setting executive priority.




### 15.1 INTEN INTERRUPT ENABLE REGISTER

INTEN initial value = x0xx xxx0

| | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| INTEN | - | TC1IEN | - | - | - | - | - | P00IEN |

P00IEN : External INT0 interrupt control bit. 0 = disable, 1 = enable.
TC1IEN : Timer/event counter 1 interrupt control bit. 0 = disable, 1 = enable.

### 15.2 INTRQ INTERRUPT REQUEST REGISTER

INTRQ initial value = x0xx xxx0

| | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| INTRQ | - | TC1IRQ | - | - | - | - | - | P00IRQ |

P00IRQ : External INT0 interrupt request control bit. 0 = non request, 1 = request.
TC1IRQ : TC1 timer/event counter interrupt request controls bit. 0 = non request, 1 = request.

Example : Interrupt service routine

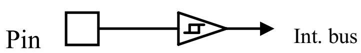
```
                ORG      0008H              ;
INTRS:          B0MOV    ACCBUF,A           ; To push ACC
                B0MOV    A,PFLAG            ; To push PFLAG register
                B0MOV    PFLAGBUF,A         ;    "
                B0BTS0   FP00IRQ            ; To skip, if P0.0 did not have interrupt request
                JMP      P00INTR            ;
                B0BTS0   FTC1IRQ            ; To skip, if TC1 did not have interrupt request
                JMP      TC1INTR            ;
                .        .                  ;
                .        .                  ;
                .        .                  ;
QINTRS:         B0MOV    A,PFLAGBUF         ; To pop PFLAG register
                B0MOV    PFLAG,A            ;    "
                B0XCH    A,ACCBUF           ; To pop ACC
                RETI     .                  ; To exit interrupt routine
                .        .                  ;
                .        .                  ;
```

```
P00INTR:     B0BCLR   FP00IRQ       ; To clear P0.0IRQ bit for waiting next time request
             .        .             ;
             .        .             ;
             .        .             ;
             JMP      QINTRS        ;
             .        .             ;
             .        .             ;
             .        .             ;
TC1INTR:     B0BCLR   FTC1IRQ       ; To clear TC1 interrupt request bit for waiting next time request
             .        .             ;
             .        .             ;
             .        .             ;
             JMP      QINTRS        ;
             .        .             ;
             .        .             ;
```

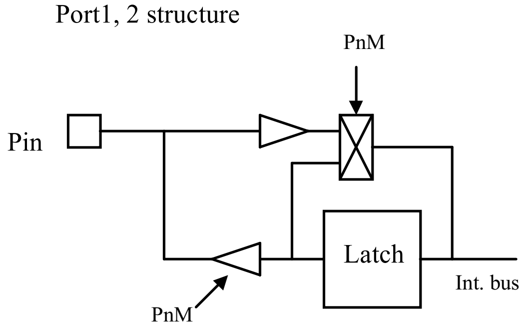
## 16. I/O PORT

The SN8P1604 provides 4 ports for users' application, consisting of one input port (P0) and three I/O ports (P1,P2,P5). The direction of I/O port is selected by PnM register. After the system resets, these ports work as input port without pull up resistors. If the user want to read-in a signal from I/O pin, it recommends to switch I/O pin as input mode to execute read-in instruction. (B0BTS0 M.b, B0BTS1 M.b or B0MOV A,M). The pull-up resistor can be set up by the code option in the programming phase.

Port0 structure




Port1, 2 structure




Note : All of the latch output circuits are push-pull structures.

### 16.1 PORT MODE (PnM) REGISTER

PnM initial value = 0000 0000

| | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| PnM | Pn7M | Pn6M | Pn5M | Pn4M | Pn3M | Pn2M | Pn1M | Pn0M |

Pn.M : The n expressed 1,2,5.
Pn7M ~ Pn0M : Port n.7 ~ Port n.0 input/output mode control bit. 0 = input, 1 = output.

### 16.2 PORT (Pn) DATA REGISTER

Pn initial value = xxxx xxxx

| | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Pn | Pn7 | Pn6 | Pn5 | Pn4 | Pn3 | Pn2 | Pn1 | Pn0 |

Pn : The n expressed the i/o port no.
Pn7 ~ Pn0 : Port n.7 ~ Port n.0 input/output data bit. 0 = logic "Low", 1 = Logic "High".

### 16.3 PORT 1 WAKEUP (P1W) REGISTER

In the power down mode , any one pin of port 1 has a logic “L” signal, it can wakeup this chip into normal mode operation. In this case, the P1.n pin must be set to input mode by P1M control and its wakeup function is programmed by P1W register.

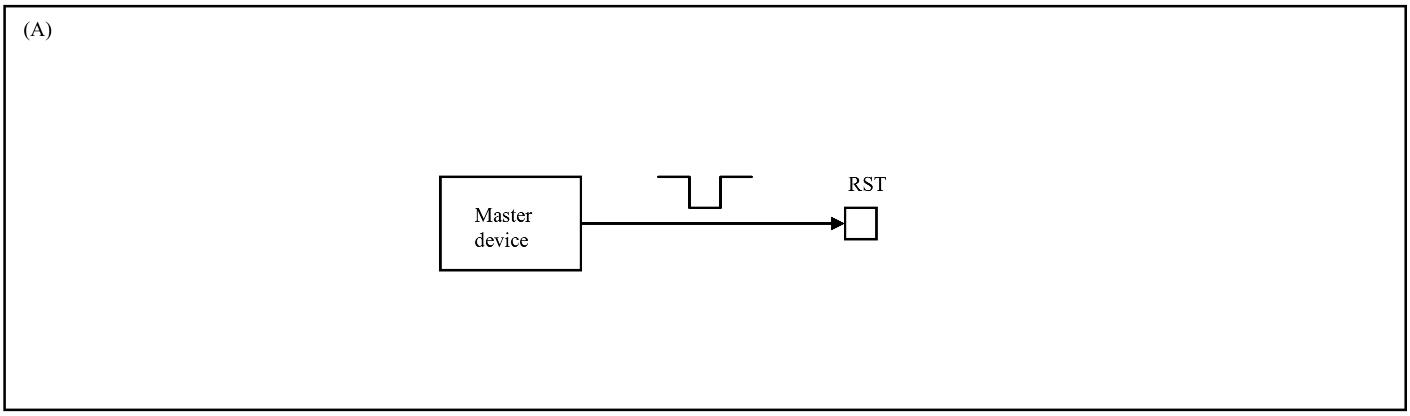
P1W initial value = 0000 0000

| | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| P1W | P17W | P16W | P15W | P14W | P13W | P12W | P11W | P10W |

P1nW : Port 1.n wakeup control bit. 0 = without wakeup function, 1 = with wakeup function.

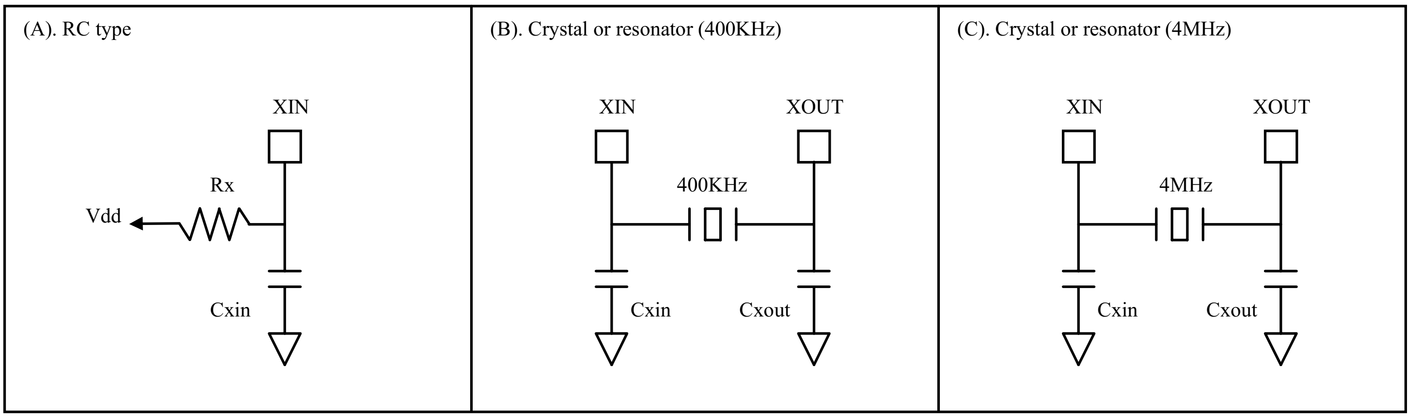
## 17.APPLICATION NOTE

TYPICAL RESET APPLICATION CIRCUIT FOR EXTERNAL MODE




The SN8P1604 had been built-in voltage detector, also it can be worded as slave to accept “low” pulse signal from external master device to reset this chip.

OSCILLATOR APPLICATION CIRCUIT FOR EXTERNAL MODE




The output frequency of RC type oscillator as follow:

| Cxin | Rx | fosc. (Vdd = 3.0V) | fosc. (Vdd = 5.0V) |
|---|---|---|---|
| 15pf | 4.7KΩ | 3MHz | 4MHz |
| | 10KΩ | 1.5MHz | 2MHz |
| | 47KΩ | 300KHz | 400KHz |
| | 100KΩ | 150KHz | 200KHz |
| | 300KΩ | 50KHz | 70KHz |

Mask option table

| Function | Option value = 0 | Option value = 1 | Remark |
|---|---|---|---|
| HXRC 00 | RC type | - | RC Oscillator and divide by 2 |
| HXRC 01 | X'TAL type | - | 32Khz |
| HXRC 10 | X'TAL type | - | 12Mhz |
| HXRC 11 | X'TAL type | - | 4Mhz |
| HXRC 01 | fhxosc ÷ $2^{18}$ | fhxosc÷ $2^{13}$ | The Warm-up timer |
| | fcpu ÷ $2^{14}$ ÷ 16 | fcpu ÷ $2^{8}$ ÷ 16 | The Watch-dog timer |
| Watch dog timer | Enable | Disable | |
| OTP security | Enable | Disable | |
| Voltage detector | Enable | Disable | LVD Level = 2.4 V |
| Pull-Up Resistor for input mode | Disable | Enable | Pull-up all the input pin |
| DIV 2 | Enable | Disable | System clock Divide by 2 |

## 18. ABSOLUTE MAXIMUM RATING

*(All of the voltages referenced to Vss)*

Supply voltage (Vdd) ........................................................ - 0.3V ~ 6.0V
Input in voltage (Vin) ........................................................ Vss - 0.2V ~ Vdd + 0.2V
Operating ambient temperature (Topr) ........................................................ 0°C ~ + 70°C
Storage ambient temperature (Tstor) ........................................................ -30°C ~ + 125°C
Power consumption (Pc) ........................................................ 500 mW

## 19. ELECTRICAL CHARACTERISTIC

*(All of voltages referenced to Vss, Vdd = 5.0V, fosc = 3.579545 MHz, ambient temperature is 25°C unless otherwise note.)*

| PARAMETER | SYM. | DESCRIPTION | | MIN. | TYP. | MAX. | UNIT |
|---|---|---|---|---|---|---|---|
| Operating voltage | Vdd | Normal mode, Vpp = Vdd | | 2.4 | 5.0 | 5.5 | V |
| | | Programming mode, Vpp = 12.5V | | 4.5 | 5.0 | 5.5 | |
| Reset pin input voltage | ViH | Reset pin high level | | 0.7Vdd | - | - | V |
| | ViL | Reset pin low level | | - | - | 0.3Vdd | |
| Reset pin leakage current | ILekg | Vin = Vdd | | - | - | 1 | uA |
| I/P port input voltage | ViH | Port 0 input voltage. | | - | 0.7Vdd | - | V |
| | ViL | | | - | 0.3Vdd | - | |
| | ViH | Port 1 and Port 2 input voltage. | | - | 0.6Vdd | - | V |
| | ViL | | | - | 0.4Vdd | - | |
| I/P port input leakage current | Ilekg | Vin = Vdd or Vin = Vss | | - | - | 2 | uA |
| Port1 output source current | IoH | Vop = Vdd - 0.5V | | - | 15 | - | mA |
| sink current | IoL | Vop = Vss + 0.5V | | - | 15 | - | |
| Port2 output source current | IoH | Vop = Vdd - 0.5V | | - | 15 | - | mA |
| sink current | IoL | Vop = Vss + 0.5V | | - | 15 | - | |
| INTn trigger pulse width | Tint0 | INT0 interrupt request pulse width | | 2/fcpu | - | - | cycle |
| Oscillator frequency | fhxosc | Crystal type or ceramic resonator | | 0.03 | 20 | - | MHz |
| | | Vdd = 3V, RC type for external mode | | 0.03 | 8 | - | |
| | | Vdd = 5V, RC type for external mode | | 0.03 | 15 | - | |
| Supply Current | Idd1 | Run Mode | Vdd= 5V 4Mhz | - | 5 | 8.5 | mA |
| | | | Vdd= 3V 4Mhz | - | 1.5 | 3 | mA |
| | | | Vdd= 3V 32768Hz | - | 45 | 90 | uA |
| | Idd2 | Internal RC mode (16KHz) | Vdd= 5V | - | 18 | 40 | uA |
| | | | Vdd= 3V | - | 15 | 30 | uA |
| | Idd3 | Stop mode | Vdd= 5V | - | 9 | 15 | uA |
| | | | Vdd= 3V | - | 2.5 | 6 | uA |
| Low Voltage Detect | Vdet | Low voltage detect level | | - | 2.4 | 2.8 | V |
| Voltage detector current | Ivdet | LVD enable operating current | | - | 100 | 180 | uA |

The current of the power down mode is dominated by the low voltage detector.

Notes:

1. The SN8P1604 LVD function must be turn on/off by the assembler option. Enable the LVD function to protect the Brown-out Reset Status. The user must provide the reset circuit for the power-on reset.

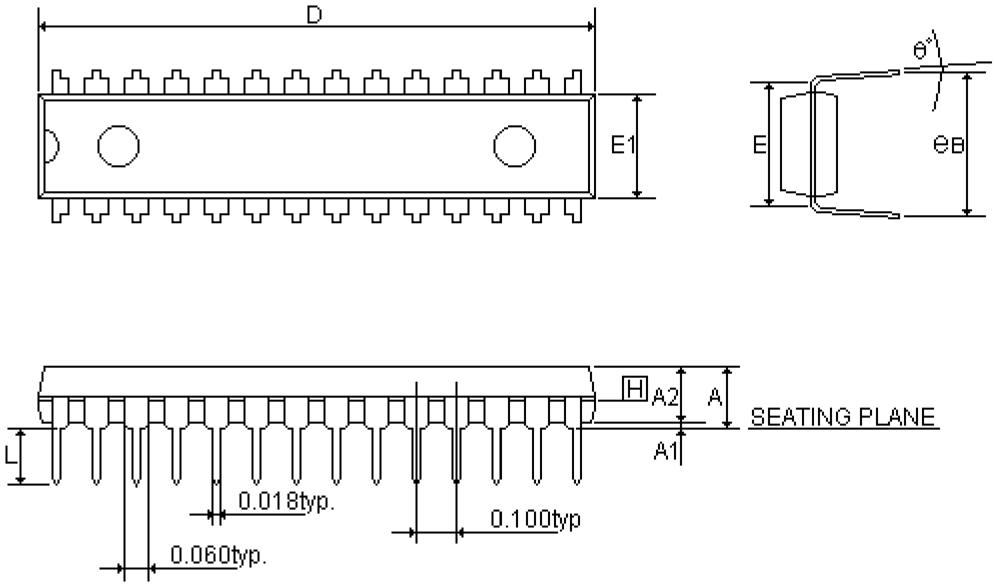
## 20. INSTRUCTION SET

| Field | Mnemonic | | Description | C | DC | Z | Cycle |
|---|---|---|---|---|---|---|---|
| | MOV | A,M | A ← M | - | - | √ | 1 |
| M | MOV | M,A | M ← A | - | - | - | 1 |
| O | B0MOV | A,M | A ← M (bnak 0) | - | - | √ | 1 |
| V | B0MOV | M,A | M (bank 0) ← A | - | - | - | 1 |
| E | MOV | A,I | A ← I | - | - | - | 1 |
| | B0MOV | M,I | M ← I, (M = Working registers, RBANK & PFLAG) | - | - | - | 1 |
| | XCH | A,M | A ←→M | - | - | - | 1 |
| | B0XCH | A,M | A ←→M (bank 0) | - | - | - | 1 |
| | MOVC | | R, A ← ROM [Y,Z] | - | - | - | 2 |
| | ADC | A,M | A ← A + M + C, if occur carry, then C=1, else C=0 | √ | √ | √ | 1 |
| A | ADC | M,A | M ← A + M + C, if occur carry, then C=1, else C=0 | √ | √ | √ | 1 |
| R | ADD | A,M | A ← A + M, if occur carry, then C=1, else C=0 | √ | √ | √ | 1 |
| I | ADD | M,A | M ← M + A, if occur carry, then C=1, else C=0 | √ | √ | √ | 1 |
| T | B0ADD | M,A | M (bank 0) ← M (bank 0) + A, if occur carry, then C=1, else C=0 | √ | √ | √ | 1 |
| H | ADD | A,I | A ← A + I, if occur carry, then C=1, else C=0 | √ | √ | √ | 1 |
| M | SBC | A,M | A ← A - M - /C, if occur borrow, then C=0, else C=1 | √ | √ | √ | 1 |
| E | SBC | M,A | M ← A - M - /C, if occur borrow, then C=0, else C=1 | √ | √ | √ | 1 |
| T | SUB | A,M | A ← A - M, if occur borrow, then C=0, else C=1 | √ | √ | √ | 1 |
| I | SUB | M,A | M ← A - M, if occur borrow, then C=0, else C=1 | √ | √ | √ | 1 |
| C | SUB | A,I | A ← A - I, if occur borrow, then C=0, else C=1 | √ | √ | √ | 1 |
| | DAA | | To adjust ACC's data format from HEX to DEC. | √ | - | - | 1 |
| | AND | A,M | A ← A and M | - | - | √ | 1 |
| L | AND | M,A | M ← A and M | - | - | √ | 1 |
| O | AND | A,I | A ← A and I | - | - | √ | 1 |
| G | OR | A,M | A ← A or M | - | - | √ | 1 |
| I | OR | M,A | M ← A or M | - | - | √ | 1 |
| C | OR | A,I | A ← A or I | - | - | √ | 1 |
| | XOR | A,M | A ← A xor M | - | - | √ | 1 |
| | XOR | M,A | M ← A xor M | - | - | √ | 1 |
| | XOR | A,I | A ← A xor I | - | - | √ | 1 |
| | SWAP | M | A (b3~b0, b7~b4) ←M(b7~b4, b3~b0) | - | - | - | 1 |
| P | SWAPM | M | M(b3~b0, b7~b4) ← M(b7~b4, b3~b0) | - | - | - | 1 |
| R | RRC | M | A ← RRC M | √ | - | - | 1 |
| O | RRCM | M | M ← RRC M | √ | - | - | 1 |
| C | RLC | M | A ← RLC M | √ | - | - | 1 |
| E | RLCM | M | M ← RLC M | √ | - | - | 1 |
| S | CLR | M | M ← 0 | - | - | - | 1 |
| S | BCLR | M.b | M.b ← 0 | - | - | - | 1 |
| | BSET | M.b | M.b ← 1 | - | - | - | 1 |
| | B0BCLR | M.b | M(bank 0).b ← 0 | - | - | - | 1 |
| | B0BSET | M.b | M(bank 0).b ← 1 | - | - | - | 1 |
| | CMPRS | A,I | ZF,C ← A - I, If A = I, then skip next instruction | √ | - | √ | 1 + S |
| B | CMPRS | A,M | ZF,C ← A - M, If A = M, then skip next instruction | √ | - | √ | 1 + S |
| R | INCS | M | A ← M + 1, If A = 0, then skip next instruction | - | - | - | 1 + S |
| A | INCMS | M | M ← M + 1, If M = 0, then skip next instruction | - | - | - | 1 + S |
| N | DECS | M | A ← M - 1, If A = 0, then skip next instruction | - | - | - | 1 + S |
| C | DECMS | M | M ← M - 1, If M = 0, then skip next instruction | - | - | - | 1 + S |
| H | BTS0 | M.b | If M.b = 0, then skip next instruction | - | - | - | 1 + S |
| | BTS1 | M.b | If M.b = 1, then skip next instruction | - | - | - | 1 + S |
| | B0BTS0 | M.b | If M(bank 0).b = 0, then skip next instruction | - | - | - | 1 + S |
| | B0BTS1 | M.b | If M(bank 0).b = 1, then skip next instruction | - | - | - | 1 + S |
| | JMP | d | PC15/14 ← RomPages1/0, PC13~PC0 ← d | - | - | - | 2 |
| | CALL | d | Stack ← PC15~PC0, PC15/14 ← RomPages1/0, PC13~PC0 ← d | - | - | - | 2 |
| M | RET | | PC ← Stack | - | - | - | 2 |
| I | RETI | | PC ← Stack, and to enable global interrupt | - | - | - | 2 |
| S | NOP | | No operation | - | - | - | 1 |

**Note :** a). Working registers = R, Y and Z
b). The memory is access to location RAM[Y,Z], if M = @YZ (located at address E7H in RAM bank 0).
c). All instructions are one cycle except for program branch and PC update which are two cycles.

## 21.PACKAGE INFORMATION

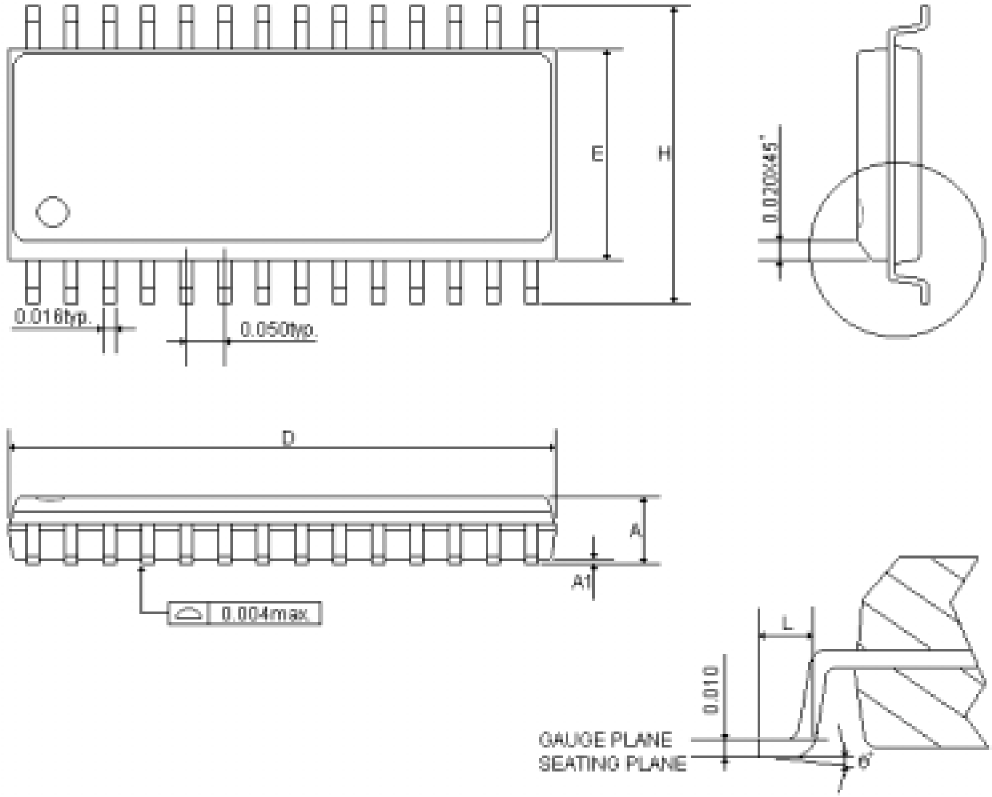
S-DIP28 pin :




| Symbols | MIN. | NOR. | MAX. |
|---|---|---|---|
| A | - | - | 0.210 |
| A1 | 0.015 | - | - |
| A2 | 0.114 | 0.130 | 0.135 |
| D | 1.390 | 1.390 | 1.400 |
| E | 0.310BSC. | | |
| E1 | 0.283 | 0.288 | 0.293 |
| L | 0.115 | 0.130 | 0.150 |
| $e_B$ | 0.330 | 0.350 | 0.370 |
| $\theta^\circ$ | 0 | 7 | 15 |

UNIT : INCH

SOP28 pin :




| Symbols | MIN. | MAX. |
|---|---|---|
| A | 0.093 | 0.104 |
| A1 | 0.004 | 0.012 |
| D | 0.697 | 0.713 |
| E | 0.291 | 0.299 |
| H | 0.394 | 0.419 |
| L | 0.016 | 0.050 |
| $\theta$° | 0 | 8 |

UNIT : INCH