

# **SN8P1700 Series**

## **USER'S MANUAL**

### **General Release Specification**

**SN8P1702**

**SN8P1704**

**SN8P1706**

**SN8P1707**

**SN8P1708**

## **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 use as components in systems intended, for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the SONiX product could create a situation where personal injury or death may occur. Should Buyer purchase or use SONiX products for any such unintended or unauthorized application. Buyer shall indemnify and hold SONiX and its officers, employees, subsidiaries, affiliates and distributors harmless against all claims, cost, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use even if such claim alleges that SONiX was negligent regarding the design or manufacture of the part.

### AMENDMENT HISTORY

| Version  | Date      | Description   |
|----------|-----------|---|
| VER 1.90 | Sep. 2002 | V1.90 first issue   |
| VER 1.93 | Feb. 2003 | <ol style="list-style-type: none"> <li>1. Extend chip operating temperature from "0°C ~ +70°C" to "-20°C ~ +70°C".</li> <li>2. Change the description of ADD M,A instruction from "M ← M+A" to "M ← A+M"</li> <li>3. Add ADC grade table.</li> <li>4. Remove "Support hardware multiplier (MUL)" in SN8P1702 FEATURES section.</li> <li>5. Change "Four internal interrupts" to "Three internal interrupts" in SN8P1704 FEATURES section.</li> <li>6. Change "ACC can't be access by "B0MOV" instruction" to "ACC can't be access by "B0MOV" instruction during the instant addressing mode".</li> <li>7. Correct the description of STKnH.</li> <li>8. Change "special register is located at 08h~FFh" to "special register is located at 80h~FFh".</li> <li>9. Correct the bit definition of INTEN register.</li> <li>10. Correct the description of "TC0 CLOCK FREQUENCY OUTPUT" section.</li> <li>11. Correct the description of "TC1 CLOCK FREQUENCY OUTPUT" section.</li> <li>12. SCKMD = 1 means SIO is in SLAVE mode. SCKMD = 0 means SIO is in MASTER mode.</li> <li>13. Remove "SIO clock and SPI clock are compatible".</li> <li>14. Modify ADB's output data table.</li> <li>15. Correct an error of template code: "b0bclr FWDRST" → "b0bset FWDRST".</li> <li>16. Add a notice about OSCM register access cycle.</li> <li>17. SN8P1702/SN8A1702A don't provide "MUL, PUSH, POP" instruction.</li> <li>18. Add a notice about OSCM register access cycle.</li> </ol> |
| VER 1.94 | Sep. 2003 | <ol style="list-style-type: none"> <li>1. Correct EOC description.</li> <li>2. Correct watchdog timer overflow time.</li> <li>3. Correct POP operand.</li> <li>4. Correct ADCKS table.</li> <li>5. Add new section about checksum calculate must avoid 04H~07H.</li> <li>6. Reserved Last 16 word ROM addresses</li> <li>7. Add SIOM table and SIO rate note</li> <li>8. Remove register bit description</li> <li>9. Modify TC0M description</li> <li>10. Modify TC1M description</li> <li>11. Modify PWM description</li> <li>12. Modify ADC Frequency description</li> <li>13. Change Code option table to Chapter 2</li> <li>14. Add ADC current consumption</li> </ol>  |

|  |  |  |
|--|--|--|
|  |  | <ul style="list-style-type: none"><li>15. Add LVD detect voltage</li><li>16. Remove approval sheet.</li><li>17. Remove PCB layout notice section.</li><li>18. Add MASK/OTP relative table.</li><li>19. Modify the description of INTRQ register.</li><li>20. Modify the calculation formula of SIOR and SIO clock.</li></ul> |
|--|--|--|

# Table of Contents

|   |           |
|---|-----------|
| AMENDMENT HISTORY .....                   | 2         |
| <b>1</b> <b>PRODUCT OVERVIEW .....</b>    | <b>11</b> |
| GENERAL DESCRIPTION .....                 | 11        |
| FEATURES SELECTION TABLE .....            | 11        |
| MASK/OTP RELATIVE TABLE .....             | 11        |
| ADC GRADE TABLE .....                     | 11        |
| SN8P1702 FEATURES .....                   | 12        |
| SN8P1704 FEATURES .....                   | 13        |
| SN8P1707/SN8P1708 FEATURES .....          | 15        |
| SYSTEM BLOCK DIAGRAM .....                | 16        |
| PIN ASSIGNMENT .....                      | 17        |
| PIN DESCRIPTIONS .....                    | 22        |
| PIN CIRCUIT DIAGRAMS .....                | 22        |
| <b>2</b> <b>CODE OPTION TABLE .....</b>   | <b>23</b> |
| <b>3</b> <b>ADDRESS SPACES .....</b>      | <b>24</b> |
| PROGRAM MEMORY (ROM) .....                | 24        |
| OVERVIEW .....                            | 24        |
| USER RESET VECTOR ADDRESS (0000H) .....   | 26        |
| INTERRUPT VECTOR ADDRESS (0008H) .....    | 26        |
| CHECKSUM CALCULATION .....                | 28        |
| GENERAL PURPOSE PROGRAM MEMORY AREA ..... | 29        |
| LOOKUP TABLE DESCRIPTION .....            | 29        |
| JUMP TABLE DESCRIPTION .....              | 31        |
| DATA MEMORY (RAM) .....                   | 33        |
| OVERVIEW .....                            | 33        |

|  |           |
|--|-----------|
| RAM BANK SELECTION .....                   | 35        |
| WORKING REGISTERS.....                     | 36        |
| H, L REGISTERS.....                        | 36        |
| Y, Z REGISTERS.....                        | 37        |
| X REGISTERS .....                          | 38        |
| R REGISTERS .....                          | 38        |
| PROGRAM FLAG .....                         | 39        |
| CARRY FLAG .....                           | 39        |
| DECIMAL CARRY FLAG.....                    | 39        |
| ZERO FLAG .....                            | 39        |
| ACCUMULATOR .....                          | 40        |
| STACK OPERATIONS.....                      | 41        |
| OVERVIEW.....                              | 41        |
| STACK REGISTERS.....                       | 42        |
| STACK OPERATION EXAMPLE.....               | 43        |
| PROGRAM COUNTER.....                       | 44        |
| ONE ADDRESS SKIPPING .....                 | 45        |
| MULTI-ADDRESS JUMPING .....                | 46        |
| <br>                                       |           |
| <b>4 ADDRESSING MODE.....</b>              | <b>47</b> |
| OVERVIEW.....                              | 47        |
| IMMEDIATE ADDRESSING MODE .....            | 47        |
| DIRECTLY ADDRESSING MODE .....             | 47        |
| INDIRECTLY ADDRESSING MODE.....            | 47        |
| TO ACCESS DATA in RAM BANK 0.....          | 48        |
| TO ACCESS DATA in RAM BANK 1.....          | 48        |
| <br>                                       |           |
| <b>5 SYSTEM REGISTER .....</b>             | <b>49</b> |
| OVERVIEW.....                              | 49        |
| SYSTEM REGISTER ARRANGEMENT (BANK 0) ..... | 49        |
| BYTES of SYSTEM REGISTER.....              | 49        |
| BITS of SYSTEM REGISTER .....              | 51        |

|          |   |           |
|----------|---|-----------|
| <b>6</b> | <b>POWER ON RESET .....</b>                               | <b>55</b> |
|          | OVERVIEW.....   | 55        |
|          | EXTERNAL RESET DESCRIPTION.....                           | 56        |
|          | LOW VOLTAGE DETECTOR (LVD) DESCRIPTION.....               | 57        |
| <b>7</b> | <b>OSCILLATORS.....</b>                                   | <b>58</b> |
|          | OVERVIEW.....   | 58        |
|          | <i>CLOCK BLOCK DIAGRAM .....</i>                          | <i>58</i> |
|          | <i>OSCM REGISTER DESCRIPTION.....</i>                     | <i>59</i> |
|          | <i>EXTERNAL HIGH-SPEED OSCILLATOR.....</i>                | <i>60</i> |
|          | <i>OSCILLATOR MODE CODE OPTION.....</i>                   | <i>60</i> |
|          | <i>OSCILLATOR DEVIDE BY 2 CODE OPTION.....</i>            | <i>60</i> |
|          | <i>OSCILLATOR SAFE GUARD CODE OPTION .....</i>            | <i>60</i> |
|          | <i>SYSTEM OSCILLATOR CIRCUITS .....</i>                   | <i>61</i> |
|          | <i>External RC Oscillator Frequency Measurement .....</i> | <i>62</i> |
|          | INTERNAL LOW-SPEED OSCILLATOR .....                       | 63        |
|          | SYSTEM MODE DESCRIPTION .....                             | 64        |
|          | OVERVIEW.....   | 64        |
|          | NORMAL MODE .....   | 64        |
|          | SLOW MODE.....  | 64        |
|          | POWER DOWN MODE.....                                      | 64        |
|          | SYSTEM MODE CONTROL.....                                  | 65        |
|          | <i>SN8P1700 SYSTEM MODE BLOCK DIAGRAM.....</i>            | <i>65</i> |
|          | <i>SYSTEM MODE SWITCHING .....</i>                        | <i>66</i> |
|          | WAKEUP TIME.....  | 67        |
|          | OVERVIEW.....   | 67        |
|          | HARDWARE WAKEUP.....                                      | 67        |
| <b>8</b> | <b>TIMERS COUNTERS.....</b>                               | <b>68</b> |
|          | WATCHDOG TIMER (WDT).....                                 | 68        |
|          | BASIC TIMER 0 (T0) .....                                  | 69        |

|   |    |
|---|----|
| OVERVIEW.....                             | 69 |
| T0M REGISTER DESCRIPTION .....            | 69 |
| T0C COUNTING REGISTER.....                | 70 |
| T0 BASIC TIMER OPERATION SEQUENCE.....    | 71 |
| TIMER COUNTER 0 (TC0).....                | 72 |
| OVERVIEW.....                             | 72 |
| TC0M MODE REGISTER.....                   | 73 |
| TC0C COUNTING REGISTER.....               | 74 |
| TC0R AUTO-LOAD REGISTER.....              | 75 |
| TC0 TIMER COUNTER OPERATION SEQUENCE..... | 76 |
| TC0 CLOCK FREQUENCY OUTPUT (BUZZER).....  | 78 |
| TC0OUT FREQUENCY TABLE.....               | 79 |
| TIMER COUNTER 1 (TC1).....                | 81 |
| OVERVIEW.....                             | 81 |
| TC1M MODE REGISTER.....                   | 82 |
| TC1C COUNTING REGISTER.....               | 83 |
| TC1R AUTO-LOAD REGISTER.....              | 84 |
| TC1 TIMER COUNTER OPERATION SEQUENCE..... | 85 |
| TC1 CLOCK FREQUENCY OUTPUT (BUZZER).....  | 87 |
| PWM FUNCTION DESCRIPTION .....            | 88 |
| OVERVIEW.....                             | 88 |
| PWM PROGRAM DESCRIPTION.....              | 89 |

## 9 INTERRUPT..... 90

|                                       |    |
|---------------------------------------|----|
| OVERVIEW.....                         | 90 |
| INTEN INTERRUPT ENABLE REGISTER ..... | 91 |
| INTRQ INTERRUPT REQUEST REGISTER..... | 91 |
| INTERRUPT OPERATION DESCRIPTION ..... | 92 |
| GIE GLOBAL INTERRUPT OPERATION .....  | 92 |
| INT0 (P0.0) INTERRUPT OPERATION ..... | 93 |
| INT1 (P0.1) INTERRUPT OPERATION ..... | 93 |
| INT2 (P0.2) INTERRUPT OPERATION ..... | 94 |
| T0 INTERRUPT OPERATION.....           | 95 |
| TC0 INTERRUPT OPERATION .....         | 96 |
| TC1 INTERRUPT OPERATION .....         | 97 |
| SIO INTERRUPT OPERATION.....          | 98 |
| MULTI-INTERRUPT OPERATION.....        | 99 |

|           |   |            |
|-----------|---|------------|
| <b>10</b> | <b>SERIAL INPUT/OUTPUT TRANSCEIVER (SIO).....</b>   | <b>101</b> |
|           | OVERVIEW.....                                       | 101        |
|           | SIOM MODE REGISTER.....                             | 102        |
|           | SIOB DATA BUFFER.....                               | 103        |
|           | SIOR REGISTER DESCRIPTION .....                     | 103        |
|           | SIO MASTER OPERATING DESCRIPTION .....              | 104        |
|           | <i>RISING EDGE TRANSMITTER/RECEIVER MODE</i> .....  | 104        |
|           | <i>FALLING EDGE TRANSMITTER/RECEIVER MODE</i> ..... | 105        |
|           | <i>RISING EDGE RECEIVER MODE</i> .....              | 106        |
|           | <i>FALLING EDGE RECEIVER MODE</i> .....             | 107        |
|           | SIO SLAVE OPERATING DESCRIPTION.....                | 108        |
|           | <i>RISING EDGE TRANSMITTER/RECEIVER MODE</i> .....  | 109        |
|           | <i>FALLING EDGE TRANSMITTER/RECEIVER MODE</i> ..... | 110        |
|           | <i>RISING EDGE RECEIVER MODE</i> .....              | 111        |
|           | <i>FALLING EDGE RECEIVER MODE</i> .....             | 112        |
|           | SIO INTERRUPT OPERATION DESCRIPTION.....            | 113        |
| <b>11</b> | <b>I/O PORT.....</b>                                | <b>114</b> |
|           | OVERVIEW.....                                       | 114        |
|           | I/O PORT FUNCTION TABLE .....                       | 115        |
|           | PULL-UP RESISTERS.....                              | 116        |
|           | I/O PORT DATA REGISTER .....                        | 119        |
| <b>12</b> | <b>8-CHANNEL ANALOG TO DIGITAL CONVERTER.....</b>   | <b>121</b> |
|           | OVERVIEW.....                                       | 121        |
|           | ADM REGISTER.....                                   | 122        |
|           | ADR REGISTERS.....                                  | 122        |
|           | ADB REGISTERS.....                                  | 122        |
|           | ADC CONVERTING TIME .....                           | 124        |
|           | ADC CIRCUIT.....                                    | 125        |



|           |  |            |
|-----------|--|------------|
| <b>13</b> | <b>7-BIT DIGITAL TO ANALOG CONVERTER .....</b> | <b>126</b> |
|           | OVERVIEW.....                                  | 126        |
|           | DAM REGISTER.....                              | 126        |
|           | D/A CONVERTER OPERATION .....                  | 127        |
| <b>14</b> | <b>CODING ISSUE .....</b>                      | <b>128</b> |
|           | TEMPLATE CODE.....                             | 128        |
|           | CHIP DECLARATION IN ASSEMBLER.....             | 133        |
|           | PROGRAM CHECK LIST .....                       | 133        |
| <b>15</b> | <b>INSTRUCTION SET TABLE .....</b>             | <b>134</b> |
| <b>16</b> | <b>ELECTRICAL CHARACTERISTIC .....</b>         | <b>135</b> |
|           | ABSOLUTE MAXIMUM RATING .....                  | 135        |
|           | STANDARD ELECTRICAL CHARACTERISTIC .....       | 135        |
|           | <i>SN8P1700 Series (OTP)</i> .....             | 135        |
| <b>17</b> | <b>PACKAGE INFORMATION .....</b>               | <b>136</b> |
|           | P-DIP18 PIN .....                              | 136        |
|           | SOP18 PIN .....                                | 137        |
|           | SSOP20 PIN .....                               | 138        |
|           | S-DIP28 PIN .....                              | 139        |
|           | SOP28 PIN .....                                | 140        |
|           | QFP 44 PIN.....                                | 141        |
|           | SSOP 48 PIN .....                              | 142        |

|                    |     |
|--------------------|-----|
| P-DIP 48 PIN ..... | 143 |
| P-DIP 40 PIN ..... | 144 |

# 1 PRODUCT OVERVIEW

## GENERAL DESCRIPTION

The SN8P1700 is a series of 8-bit micro-controller including SN8P1702, SN8P1704, SN8P1706, SN8P1707 and SN8P1708. This series is utilized with CMOS technology fabrication and featured with low power consumption and high performance by its unique electronic structure.

These chips are designed with the excellent IC structure including the large program memory OTP ROM, the massive data memory RAM, one 8-bit basic timer (T0), two 8-bit timer counters (TC0, TC1), a watchdog timer, up to seven interrupt sources (T0, TC0, TC1, SIO, INT0, INT1, INT2), a 7-bit DAC converter, an 8-channel ADC converter with 8-bit/12-bit resolution, two channel PWM output (PWM0, PWM1), two channel buzzer output (BZ0, BZ1) and 8-level stack buffers. Besides, the user can choose desired oscillator configurations for the controller. There are four oscillator configurations to select for generating system clock, including High/Low Speed crystal, ceramic resonator or cost-saving RC. SN8P1700 series also includes an internal RC oscillator for slow mode controlled by programming.

## FEATURES SELECTION TABLE

| CHIP     | ROM   | RAM | Stack | Timer |     |     | I/O | ADC | DAC | PWM    | SIO | Wakeup  | Package       |
|----------|-------|-----|-------|-------|-----|-----|-----|-----|-----|--------|-----|---------|---------------|
|          |       |     |       | T0    | TC0 | TC1 |     |     |     | Buzzer |     | Pin no. |               |
| SN8P1702 | 1K*16 | 64  | 8     | -     | V   | -   | 12  | 4ch | -   | 1      | -   | 3       | DIP18/SOP18   |
| SN8P1704 | 2K*16 | 128 |       | -     | V   | V   | 18  | 5ch | 1ch | 2      | 1   | 8       | SKDIP28/SOP28 |
| SN8P1706 | 4K*16 | 256 |       | V     | V   | V   | 30  | 8ch | 1ch | 2      | 1   | 9       | DIP40         |
| SN8P1707 |       |     |       | V     | V   | V   | 33  | 8ch | 1ch | 2      | 1   | 9       | QFP44         |
| SN8P1708 |       |     |       | V     | V   | V   | 33  | 8ch | 1ch | 2      | 1   | 9       | DIP48/SSOP48  |

Table 1-1. Selection Table of SN8P1700

## MASK/OTP Relative Table

| Mask Version | Package Form       | OTP Chip for Verification | Assembler Declaration |
|--------------|--------------------|---------------------------|-----------------------|
| SN8A1702A    | DIP18/SOP18/SSOP20 | SN8P1702                  | CHIP SN8P1702         |
| SN8A1704A    | SKDIP28/SOP28      | SN8P1704                  | CHIP SN8P1704         |
| SN8A1706A    | DIP40              | SN8P1706                  | CHIP SN8P1706         |
| SN8A1707A    | QFP44              | SN8P1707                  | CHIP SN8P1707         |
| SN8A1708A    | DIP48/SSOP48       | SN8P1708                  | CHIP SN8P1708         |

Note: Recommend SN8P1702A to replace SN8P1702 in new design. Refer SN8P1702A datasheet for details.

Table 1-2. MASK/OTP Relative Table

## ADC GRADE TABLE

| CHIP        | PARAMETER                       | MIN | MAX | UNITS | REMARK             |
|-------------|---------------------------------|-----|-----|-------|--------------------|
| SN8P170X    | Resolution                      |     | 12  | Bits  | 170X:<br>1702~1708 |
|             | No Mission Code                 | 8   | 12  | Bits  |                    |
|             | Differential Nonlinearity (DNL) |     | 16  | LSB   |                    |
| SN8P170X-12 | Resolution                      |     | 12  | Bits  | 170X:<br>1702~1708 |
|             | No Mission Code                 | 10  | 12  | Bits  |                    |
|             | Differential Nonlinearity (DNL) |     | 4   | LSB   |                    |

Table 1-3. ADC Grade Table

## SN8P1702 FEATURES

- ◆ **Memory configuration**  
OTP ROM size: 1K \* 16 bits.  
RAM size: 64 \* 8 bits.
- ◆ **I/O pin configuration (Total 12 pins)**  
Input only: P0  
Bi-directional: P1, P4, P5  
Wakeup: P0, P1  
Pull-up resistors: P0, P1, P4, P5  
External interrupt: P0  
P4 pins shared with ADC inputs.
- ◆ **One 8-bit timer counters. (TC0).**
- ◆ **On chip watchdog timer.**
- ◆ **Eight levels stack buffer.**
- ◆ **59 powerful instructions**  
Four clocks per instruction cycle  
All of instructions are one word length.  
Most of instructions are one cycle only.  
All ROM area lookup table function (MOVC)
- ◆ **Two interrupt sources**  
One internal interrupts: TC0.  
One external interrupts: INT0.
- ◆ **An 4-channel ADC with 8-bit/12-bit resolution**
- ◆ **One channel PWM output. (PWM0)**
- ◆ **One channel Buzzer output. (BZ0)**
- ◆ **Dual clock system offers three operating modes**  
External high clock: RC type up to 10 MHz  
External high clock: Crystal type up to 16 MHz  
Internal low clock: RC type 16KHz(3V), 32KHz(5V)  
Normal mode: Both high and low clock active  
Slow mode: Low clock only  
Sleep mode: Both high and low clock stop
- ◆ **Package (Chip form support)**  
PDIP 18 pins  
SOP 18 pins / SSOP20 (MASK type only)

### Notice:

1. Declare "CHIP SN8P1702" in assembler.
2. Use @SET\_PUR macro to control pull-up resistor. Refer I/O chapter for detailed information
3. Call @SET\_PUR macro at least one time to avoid sleep mode fail.

## SN8P1704 FEATURES

- ◆ **Memory configuration**  
OTP ROM size: 2K \* 16 bits.  
RAM size: 128 \* 8 bits.
- ◆ **I/O pin configuration (Total 18 pins)**  
Input only: P0  
Bi-directional: P1, P4, P5  
Wakeup: P0, P1  
Pull-up resistors: P0, P1, P4, P5  
External interrupt: P0  
P4 pins shared with ADC inputs.
- ◆ **Two 8-bit timer counters. (TC0, TC1).**
- ◆ **On chip watchdog timer.**
- ◆ **Eight levels stack buffer.**
- ◆ **60 powerful instructions**  
Four clocks per instruction cycle  
All of instructions are one word length.  
Most of instructions are one cycle only.  
All ROM area lookup table function (MOVCL)  
Support hardware multiplier (MUL).
- ◆ **Six interrupt sources**  
Three internal interrupts: TC0, TC1, SIO.  
Three external interrupts: INT0, INT1, INT2.
- ◆ **A 5-channel ADC with 8-bit/12-bit resolution.**
- ◆ **One channel DAC with 7-bit resolution.**
- ◆ **SIO function.**
- ◆ **Two channel PWM output. (PWM0, PWM1)**
- ◆ **Two channel Buzzer output. (BZ0, BZ1)**
- ◆ **Dual clock system offers three operating modes**  
External high clock: RC type up to 10 MHz  
External high clock: Crystal type up to 16 MHz  
Internal low clock: RC type 16KHz(3V), 32KHz(5V)  
Normal mode: Both high and low clock active  
Slow mode: Low clock only  
Sleep mode: Both high and low clock stop
- ◆ **Package (Chip form support)**  
SOP 28 pins  
SKDIP 28 pins

### Notice:

1. Declare "CHIP SN8P1704" in assembler.
2. Use @SET\_PUR macro to control pull-up resistor. Refer I/O chapter for detailed information
3. Call @SET\_PUR macro at least one time to avoid sleep mode fail.

## SN8P1706 FEATURES

- ◆ **Memory configuration**  
OTP ROM size: 4K \* 16 bits.  
RAM size: 256 \* 8 bits (bank 0 and bank 1).
- ◆ **I/O pin configuration (Total 30 pins)**  
Input only: P0  
Bi-directional: P1, P2, P4, P5  
Wakeup: P0, P1  
Pull-up resistors: P0, P1, P2, P4, P5  
External interrupt: P0  
P4 pins shared with ADC inputs.
- ◆ **An 8-bit basic timer. (T0).**
- ◆ **Two 8-bit timer counters. (TC0, TC1).**
- ◆ **On chip watchdog timer.**
- ◆ **Eight levels stack buffer.**
- ◆ **60 powerful instructions**  
Four clocks per instruction cycle  
All of instructions are one word length.  
Most of instructions are one cycle only.  
All ROM area lookup table function (MOVC)  
Support hardware multiplier (MUL).
- ◆ **Seven interrupt sources**  
Four internal interrupts: T0, TC0, TC1, SIO.  
Three external interrupts: INT0, INT1, INT2.
- ◆ **An 8-channel ADC with 8-bit/12-bit resolution.**
- ◆ **One channel DAC 7bit resolution.**
- ◆ **SIO function.**
- ◆ **Two channel PWM output. (PWM0, PWM1)**
- ◆ **Two channel Buzzer output. (BZ0, BZ1)**
- ◆ **Dual clock system offers three operating modes**  
External high clock: RC type up to 10 MHz  
External high clock: Crystal type up to 16 MHz  
Internal low clock: RC type 16KHz(3V), 32KHz(5V)  
Normal mode: Both high and low clock active  
Slow mode: Low clock only  
Sleep mode: Both high and low clock stop
- ◆ **Package (Chip form support)**  
P-DIP 40 pins

### Notice:

1. Declare "CHIP SN8P1706" in assembler.
2. Use @SET\_PUR macro to control pull-up resistor. Refer I/O chapter for detailed information

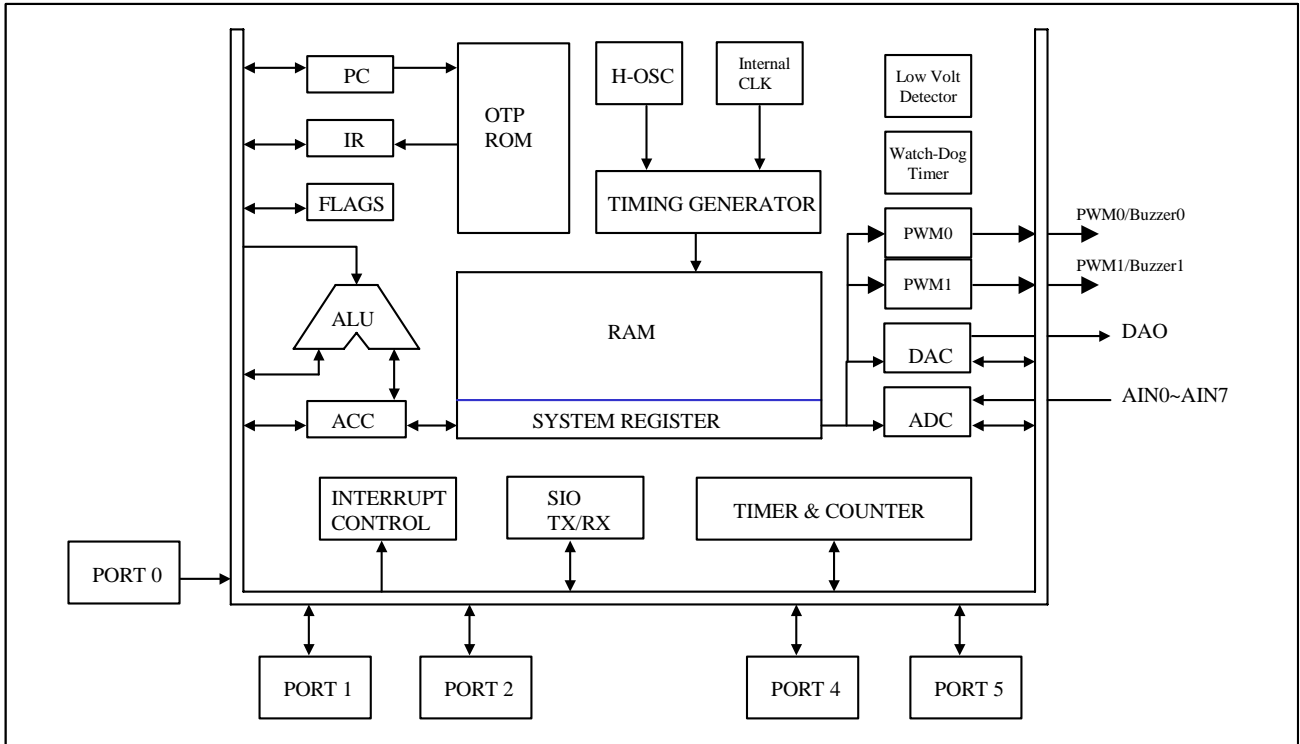
## SN8P1707/SN8P1708 FEATURES

- ◆ **Memory configuration**  
OTP ROM size: 4K \* 16 bits.  
RAM size: 256 \* 8 bits (bank 0 and bank 1).
- ◆ **I/O pin configuration (Total 33 pins)**  
Input only: P0  
Bi-directional: P1, P2, P4, P5  
Wakeup: P0, P1  
Pull-up resistors: P0, P1, P2, P4, P5  
External interrupt: P0  
P4 pins shared with ADC inputs.
- ◆ **An 8-bit basic timer. (T0).**
- ◆ **Two 8-bit timer counters. (TC0, TC1).**
- ◆ **On chip watchdog timer.**
- ◆ **Eight levels stack buffer.**
- ◆ **60 powerful instructions**  
Four clocks per instruction cycle  
All of instructions are one word length.  
Most of instructions are one cycle only.  
All ROM area lookup table function (MOVC)  
Support hardware multiplier (MUL).
- ◆ **Seven interrupt sources**  
Four internal interrupts: T0, TC0, TC1, SIO.  
Three external interrupts: INT0, INT1, INT2.
- ◆ **An 8-channel ADC with 8-bit/12-bit resolution.**
- ◆ **One channel DAC with 7-bit resolution.**
- ◆ **SIO function.**
- ◆ **Two channel PWM output. (PWM0, PWM1)**
- ◆ **Two channel Buzzer output. (BZ0, BZ1)**
- ◆ **Dual clock system offers three operating modes**  
External high clock: RC type up to 10 MHz  
External high clock: Crystal type up to 16 MHz  
Internal low clock: RC type 16KHz(3V), 32KHz(5V)  
Normal mode: Both high and low clock active  
Slow mode: Low clock only  
Sleep mode: Both high and low clock stop
- ◆ **Package (Chip form support)**  
QPF 44 pins (SN8P1707)  
SSOP 48 pins (SN8P1708)  
PDIP 48 pins (SN8P1708)

### Notice:

1. Declare "CHIP SN8P1707" for SN8P1707 in assembler.
2. Declare "CHIP SN8P1708" for SN8P1708 in assembler.
3. Use @SET\_PUR macro to control pull-up resistor. Refer I/O chapter for detailed information

## SYSTEM BLOCK DIAGRAM



**Figure 1-1.Simplified System Block Diagram**



## PIN ASSIGNMENT

Format Description : SN8P17XXY

Y = Q > QFP , P > PDIP , K > SKDIP , S > SOP , X > SSOP

### OTP Type:

SN8P1702 (SOP 18PIN)  
SN8P1702 (PDIP 18PIN)

|           |   |   |    |               |
|-----------|---|---|----|---------------|
| P0.0/INT0 | 1 | U | 18 | VDD/VPP       |
| RST       | 2 |   | 17 | XIN           |
| P1.1      | 3 |   | 16 | XOUT          |
| P1.0      | 4 |   | 15 | P5.0          |
| VSS       | 5 |   | 14 | P5.1          |
| P4.3/AIN3 | 6 |   | 13 | P5.2          |
| P4.2/AIN2 | 7 |   | 12 | P5.3          |
| P4.1/AIN1 | 8 |   | 11 | P5.4/BZ0/PWM0 |
| P4.0/AIN0 | 9 |   | 10 | VDD           |

SN8P1702P  
SN8P1702S

### MASK Type:

SN8A1702A (SOP 18PIN)  
SN8A1702A (PDIP 18PIN)  
SN8A1702A (SSOP 20PIN)

|           |   |   |    |               |
|-----------|---|---|----|---------------|
| P0.0/INT0 | 1 | U | 18 | VDD           |
| RST       | 2 |   | 17 | XIN           |
| P1.1      | 3 |   | 16 | XOUT          |
| P1.0      | 4 |   | 15 | P5.0          |
| VSS       | 5 |   | 14 | P5.1          |
| P4.3/AIN3 | 6 |   | 13 | P5.2          |
| P4.2/AIN2 | 7 |   | 12 | P5.3          |
| P4.1/AIN1 | 8 |   | 11 | P5.4/BZ0/PWM0 |
| P4.0/AIN0 | 9 |   | 10 | VDD           |

SN8A1702AP  
SN8A1702AS

|           |    |   |    |               |
|-----------|----|---|----|---------------|
| VSS       | 1  | U | 20 | P1.0          |
| VSS       | 2  |   | 19 | P1.1          |
| P4.3/AIN3 | 3  |   | 18 | RST           |
| P4.2/AIN2 | 4  |   | 17 | P0.0/INT0     |
| P4.1/AIN1 | 5  |   | 16 | VDD           |
| P4.0/AIN0 | 6  |   | 15 | XIN           |
| AVREFH    | 7  |   | 14 | XOUT          |
| VDD       | 8  |   | 13 | P5.0          |
| P5.3      | 9  |   | 12 | P5.1          |
| P5.2      | 10 |   | 11 | P5.4/BZ0/PWM0 |

SN8A1702AX

➤ Only MASK type support SSOP20 package

**OTP Type:**
**SN8P1704 (SOP 28PIN)**
**SN8P1704 (SKDIP 28PIN)**

|           |    |   |    |               |
|-----------|----|---|----|---------------|
| P1.4      | 1  | U | 28 | RST           |
| P1.3      | 2  |   | 27 | P0.2/INT2     |
| VDD       | 3  |   | 26 | P0.1/INT1     |
| P1.2      | 4  |   | 25 | P0.0/INT0     |
| P1.1      | 5  |   | 24 | VDD/VPP       |
| P1.0      | 6  |   | 23 | XIN           |
| VSS       | 7  |   | 22 | XOUT          |
| P4.4/AIN4 | 8  |   | 21 | VSS           |
| P4.3/AIN3 | 9  |   | 20 | P5.0/SCK      |
| P4.2/AIN2 | 10 |   | 19 | P5.1/SI       |
| P4.1/AIN1 | 11 |   | 18 | P5.2/SO       |
| P4.0/AIN0 | 12 |   | 17 | P5.3/BZ1/PWM1 |
| AVREFH    | 13 |   | 16 | P5.4/BZ0/PWM0 |
| VDD       | 14 |   | 15 | DAO           |

SN8P1704K

SN8P1704S

**MASK Type:**
**SN8A1704A (SOP 28PIN)**
**SN8A1704A (SKDIP 28PIN)**

|           |    |   |    |               |
|-----------|----|---|----|---------------|
| P1.4      | 1  | U | 28 | RST           |
| P1.3      | 2  |   | 27 | P0.2/INT2     |
| VDD       | 3  |   | 26 | P0.1/INT1     |
| P1.2      | 4  |   | 25 | P0.0/INT0     |
| P1.1      | 5  |   | 24 | VDD           |
| P1.0      | 6  |   | 23 | XIN           |
| VSS       | 7  |   | 22 | XOUT          |
| P4.4/AIN4 | 8  |   | 21 | VSS           |
| P4.3/AIN3 | 9  |   | 20 | P5.0/SCK      |
| P4.2/AIN2 | 10 |   | 19 | P5.1/SI       |
| P4.1/AIN1 | 11 |   | 18 | P5.2/SO       |
| P4.0/AIN0 | 12 |   | 17 | P5.3/BZ1/PWM1 |
| AVREFH    | 13 |   | 16 | P5.4/BZ0/PWM0 |
| VDD       | 14 |   | 15 | DAO           |

SN8A1704AK

SN8A1704AS

**OTP Type:**

SN8P1706 (P-DIP 40PIN)

|           |    |   |    |               |
|-----------|----|---|----|---------------|
| P1.5      | 1  | U | 40 | RST           |
| P1.4      | 2  |   | 39 | P0.2/INT2     |
| P1.3      | 3  |   | 38 | P0.1/INT1     |
| VDD       | 4  |   | 37 | P0.0/INT0     |
| P1.2      | 5  |   | 36 | VDD/VPP       |
| P1.1      | 6  |   | 35 | XIN           |
| P1.0      | 7  |   | 34 | XOUT          |
| P2.0      | 8  |   | 33 | VSS           |
| P2.1      | 9  |   | 32 | P2.4          |
| P2.2      | 10 |   | 31 | P5.0/SCK      |
| P2.3      | 11 |   | 30 | P5.1/SI       |
| VSS       | 12 |   | 29 | P5.2/SO       |
| P4.7/AIN7 | 13 |   | 28 | P5.3/BZ1/PWM1 |
| P4.6/AIN6 | 14 |   | 27 | P5.4/BZ0/PWM0 |
| P4.5/AIN5 | 15 |   | 26 | P5.5          |
| P4.4/AIN4 | 16 |   | 25 | P5.6          |
| P4.3/AIN3 | 17 |   | 24 | P5.7          |
| P4.2/AIN2 | 18 |   | 23 | DAO           |
| P4.1/AIN1 | 19 |   | 22 | VDD           |
| P4.0/AIN0 | 20 |   | 21 | AVREFH        |

SN8P1706P

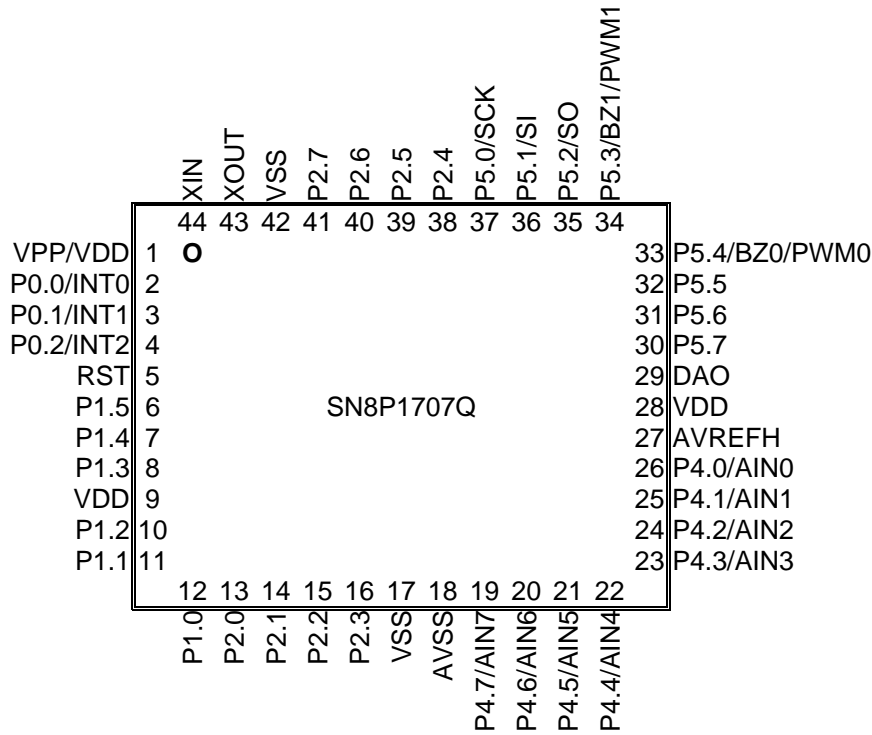
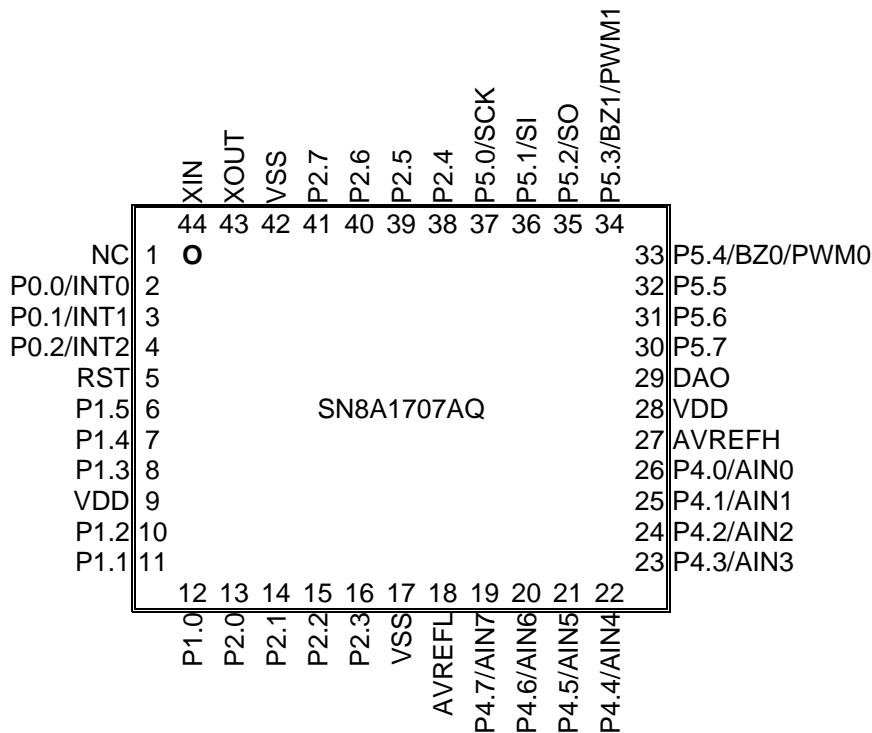
**MASK Type:**

SN8A1706A (P-DIP 40PIN)

|           |    |   |    |               |
|-----------|----|---|----|---------------|
| P1.5      | 1  | U | 40 | RST           |
| P1.4      | 2  |   | 39 | P0.2/INT2     |
| P1.3      | 3  |   | 38 | P0.1/INT1     |
| VDD       | 4  |   | 37 | P0.0/INT0     |
| P1.2      | 5  |   | 36 | NC            |
| P1.1      | 6  |   | 35 | XIN           |
| P1.0      | 7  |   | 34 | XOUT          |
| P2.0      | 8  |   | 33 | VSS           |
| P2.1      | 9  |   | 32 | P2.4          |
| P2.2      | 10 |   | 31 | P5.0/SCK      |
| P2.3      | 11 |   | 30 | P5.1/SI       |
| AVREFL    | 12 |   | 29 | P5.2/SO       |
| P4.7/AIN7 | 13 |   | 28 | P5.3/BZ1/PWM1 |
| P4.6/AIN6 | 14 |   | 27 | P5.4/BZ0/PWM0 |
| P4.5/AIN5 | 15 |   | 26 | P5.5          |
| P4.4/AIN4 | 16 |   | 25 | P5.6          |
| P4.3/AIN3 | 17 |   | 24 | P5.7          |
| P4.2/AIN2 | 18 |   | 23 | DAO           |
| P4.1/AIN1 | 19 |   | 22 | VDD           |
| P4.0/AIN0 | 20 |   | 21 | AVREFH        |

SN8A1706AP

- For OTP type (SN8P1706) compatible issue, please connect AVREFL pin of MASK type (SN8A1706A) to the analog ground of PCB. The voltage level of AVREFL pin is the valid lowest ADC input voltage. By the way, the AVREFH is the valid highest ADC input voltage.

**OTP Type:**
**SN8P1707 (QFP 44PIN)**

**MASK Type:**
**SN8A1707A (QFP 44PIN)**


➤ For OTP type (SN8P1707) compatible issue, please connect AVREFL pin of MASK type (SN8A1707A) to the analog ground of PCB. The voltage level of AVREFL pin is the valid lowest ADC input voltage. By the way, the AVREFH is the valid highest ADC input voltage.

**OTP Type:**
**SN8P1708 (SSOP 48PIN)**
**SN8P1708 (P-DIP 48PIN)**

|           |    |   |    |               |
|-----------|----|---|----|---------------|
| P2.5      | 1  | U | 48 | P2.4          |
| P2.6      | 2  |   | 47 | P5.0/SCK      |
| P2.7      | 3  |   | 46 | P5.1/SI       |
| VSS       | 4  |   | 45 | P5.2/SO       |
| VSS       | 5  |   | 44 | P5.3/BZ1/PWM1 |
| XOUT      | 6  |   | 43 | VSS           |
| XIN       | 7  |   | 42 | P5.4/BZ0/PWM0 |
| VPP/VDD   | 8  |   | 41 | P5.5          |
| P0.0/INT0 | 9  |   | 40 | P5.6          |
| P0.1/INT1 | 10 |   | 39 | P5.7          |
| P0.2/INT2 | 11 |   | 38 | DAO           |
| RST       | 12 |   | 37 | VDD           |
| P1.5      | 13 |   | 36 | AVDD          |
| P1.4      | 14 |   | 35 | AVREFH        |
| P1.3      | 15 |   | 34 | P4.0/AIN0     |
| VDD       | 16 |   | 33 | P4.1/AIN1     |
| VSS       | 17 |   | 32 | P4.2/AIN2     |
| P1.2      | 18 |   | 31 | P4.3/AIN3     |
| P1.1      | 19 |   | 30 | P4.4/AIN4     |
| P1.0      | 20 |   | 29 | P4.5/AIN5     |
| P2.0      | 21 |   | 28 | P4.6/AIN6     |
| P2.1      | 22 |   | 27 | P4.7/AIN7     |
| P2.2      | 23 |   | 26 | AVSS          |
| P2.3      | 24 |   | 25 | VSS           |

SN8P1708P

SN8P1708X

**MASK Type:**
**SN8A1708A (SSOP 48PIN)**
**SN8A1708A (P-DIP 48PIN)**

|           |    |   |    |               |
|-----------|----|---|----|---------------|
| P2.5      | 1  | U | 48 | P2.4          |
| P2.6      | 2  |   | 47 | P5.0/SCK      |
| P2.7      | 3  |   | 46 | P5.1/SI       |
| VSS       | 4  |   | 45 | P5.2/SO       |
| VSS       | 5  |   | 44 | P5.3/BZ1/PWM1 |
| XOUT      | 6  |   | 43 | VSS           |
| XIN       | 7  |   | 42 | P5.4/BZ0/PWM0 |
| NC        | 8  |   | 41 | P5.5          |
| P0.0/INT0 | 9  |   | 40 | P5.6          |
| P0.1/INT1 | 10 |   | 39 | P5.7          |
| P0.2/INT2 | 11 |   | 38 | DAO           |
| RST       | 12 |   | 37 | VDD           |
| P1.5      | 13 |   | 36 | AVDD          |
| P1.4      | 14 |   | 35 | AVREFH        |
| P1.3      | 15 |   | 34 | P4.0/AIN0     |
| VDD       | 16 |   | 33 | P4.1/AIN1     |
| VSS       | 17 |   | 32 | P4.2/AIN2     |
| P1.2      | 18 |   | 31 | P4.3/AIN3     |
| P1.1      | 19 |   | 30 | P4.4/AIN4     |
| P1.0      | 20 |   | 29 | P4.5/AIN5     |
| P2.0      | 21 |   | 28 | P4.6/AIN6     |
| P2.1      | 22 |   | 27 | P4.7/AIN7     |
| P2.2      | 23 |   | 26 | AVREFL        |
| P2.3      | 24 |   | 25 | VSS           |

SN8A1708AP

SN8A1708AX

- **For OTP type (SN8P1708) compatible issue, please connect AVREFL pin of MASK type (SN8A1708A) to the analog ground of PCB. The voltage level of AVREFL pin is the valid lowest ADC input voltage. By the way, the AVREFH is the valid highest ADC input voltage.**

## PIN DESCRIPTIONS

| PIN NAME          | TYPE | DESCRIPTION   |
|-------------------|------|---|
| VDD, VSS          | P    | Power supply input pins for digital circuit.  |
| AVDD, AVSS        | P    | Power supply input pins for analog circuit.   |
| VPP/VDD           | P    | OTP ROM programming pin. Connect to VDD in normal operation.  |
| RST               | I    | System reset input pin. Schmitt trigger structure, active "low", normal stay to "high".                         |
| XIN, XOUT         | I, O | External oscillator pins. RC mode from XIN.   |
| P0.0 / INT0       | I    | Port 0.0 and shared with INT0 trigger pin (Schmitt trigger) / Built-in pull-up resistors.                       |
| P0.1 / INT1       | I    | Port 0.1 and shared with INT1 trigger pin (Schmitt trigger) / Built-in pull-up resistors.                       |
| P0.2 / INT2       | I    | Port 0.2 and shared with INT2 trigger pin (Schmitt trigger) / Built-in pull-up resistors.                       |
| P1.0 ~ P1.5       | I/O  | Port 1.0~Port 1.5 bi-direction pins / Built-in pull-up resistors.   |
| P2.0 ~ P2.7       | I/O  | Port 2.0~Port 2.7 bi-direction pins / Built-in pull-up resistors.   |
| P4.0 ~ P4.7       | I/O  | Port 4.0~Port 4.7 bi-direction pins / Built-in pull-up resistors.   |
| P5.0 / SCK        | I/O  | Port 5.0 bi-direction pin and SIO's clock input/output / Built-in pull-up resistors.                            |
| P5.1 / SI         | I/O  | Port 5.1 bi-direction pin and SIO's data input / Built-in pull-up resistors.                                    |
| P5.2 / SO         | I/O  | Port 5.2 bi-direction pin and SIO's data output / Built-in pull-up resistors.                                   |
| P5.3 / BZ1 / PWM1 | I/O  | Port 5.3 bi-direction pin, TC1 ÷ 2 signal output pin for buzzer or PWM1 output pin. Built-in pull-up resistors. |
| P5.4 / BZ0 / PWM0 | I/O  | Port 5.4 bi-direction pin, TC0 ÷ 2 signal output pin for buzzer or PWM0 output pin. Built-in pull-up resistors. |
| P5.5 ~ P5.7       | I/O  | Port 5.5~Port 5.7 bi-direction pins / Built-in pull-up resistors.   |
| AVREFH            | I    | A/D converter high analog reference voltage.  |
| AIN0 ~ AIN7       | I    | Analog signal input pins for ADC converter.   |
| DAO               | O    | 5-bit DAC signal output pin.  |

Table 1-4. SN8P1700 Pin Description

## PIN CIRCUIT DIAGRAMS

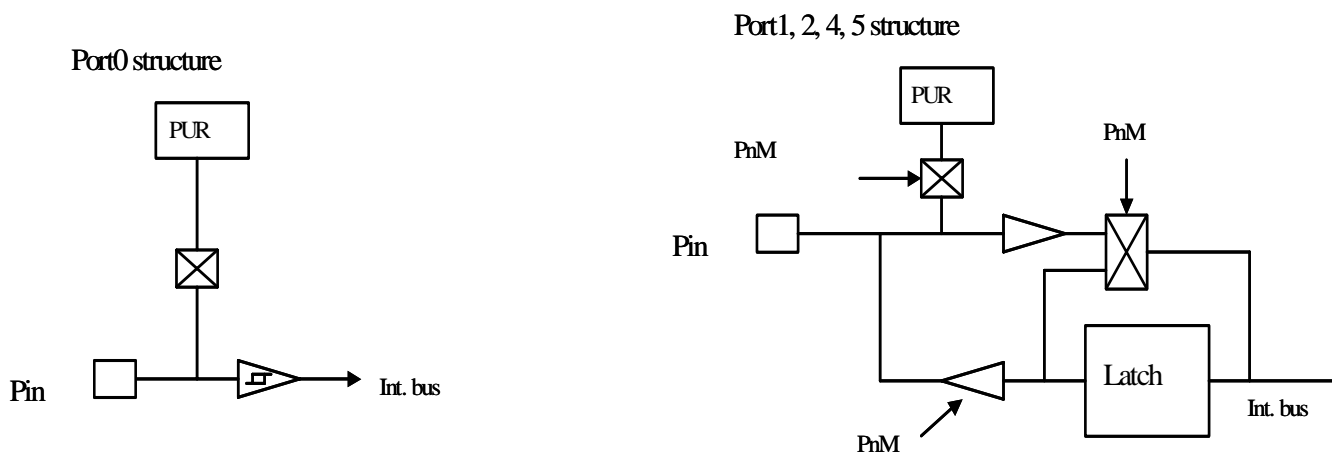


Figure 1-2. Pin Circuit Diagram

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

# 2 CODE OPTION TABLE

| Code Option  | Content   | Function Description  |
|--------------|-----------|---|
| High_Clk     | RC        | Low cost RC for external high clock oscillator  |
|              | 32K X'tal | Low frequency, power saving crystal (e.g. 32.768K) for external high clock oscillator |
|              | 12M X'tal | High speed crystal /resonator (e.g. 12M) for external high clock oscillator           |
|              | 4M X'tal  | Standard crystal /resonator (e.g. 3.58M) for external high clock oscillator           |
| High_Clk / 2 | Enable    | External high clock divided by two, Fosc = high clock / 2                             |
|              | Disable   | Fosc = high clock   |
| OSG          | Enable    | Enable Oscillator Safe Guard function   |
|              | Disable   | Disable Oscillator Safe Guard function  |
| Watch_Dog    | Enable    | Enable Watch Dog function   |
|              | Disable   | Disable Watch Dog function  |
| LVD          | Enable    | Enable the low voltage detect   |
|              | Disable   | Disable the low voltage detect  |
| Security     | Enable    | Enable ROM code Security function   |
|              | Disable   | Disable ROM code Security function  |

Table 2-1. Code Option Table of SN8P1700

### Notice : The OSG working voltage and the frequency relation table:

The min. working voltage will be affect by the OSG option. It is very important to check this code option. Turn on the OSG will improve the EMI performance. But the side effect is an increase in the working voltage.

| OSC. Freq.(Mhz) | OSG ON (Volt) | OSG OFF(Volt) |
|-----------------|---------------|---------------|
| 1               | 2.4           | 2.2           |
| 2               | 2.4           | 2.2           |
| 4               | 2.5           | 2.2           |
| 6               | 2.5           | 2.3           |
| 8               | 2.6           | 2.4           |
| 10              | 2.8           | 2.6           |
| 12              | 3             | 2.7           |
| 16              | 3.5           | 2.8           |
| 18              | 3.7           | 3             |
| 20              | 4.1           | 3.2           |

Notice : The system working frequency is only warranty under 16Mhz.

# 3 ADDRESS SPACES

## PROGRAM MEMORY (ROM)

### OVERVIEW

ROM Maps for SN8P1700 devices provide OTP memory that programmable by user. SN8P1702 has 1K x 16-bit program memory, SN8P1704 has 2K x 16-bit program memory and SN8P1706, SN8P1707 and SN8P1708 have 4K x 16-bit program memory. The SN8P1700 program memory is able to fetch instructions through 12-bit wide PC (Program Counter) and can look up ROM data by using ROM code registers (R, X, Y, Z). In standard configuration, the device's 4,096 x 16-bit program memory has four areas:

- 1-word reset vector addresses
- 1-word Interrupt vector addresses
- 5-words reserved area
- 4K words (SN8P1706, SN8P1707, SN8P1708)
- 2K words (SN8P1704)
- 1K words (SN8P1702)

All of the program memory is partitioned into three coding areas. The 1<sup>st</sup> area is located from 00H to 03H(The Reset vector area), the 2<sup>nd</sup> area is a reserved area 04H ~07H, the 3<sup>rd</sup> area is for the interrupt vector and the user code area from 0008H to 0FFEH. The address 08H is the interrupt enter address point.

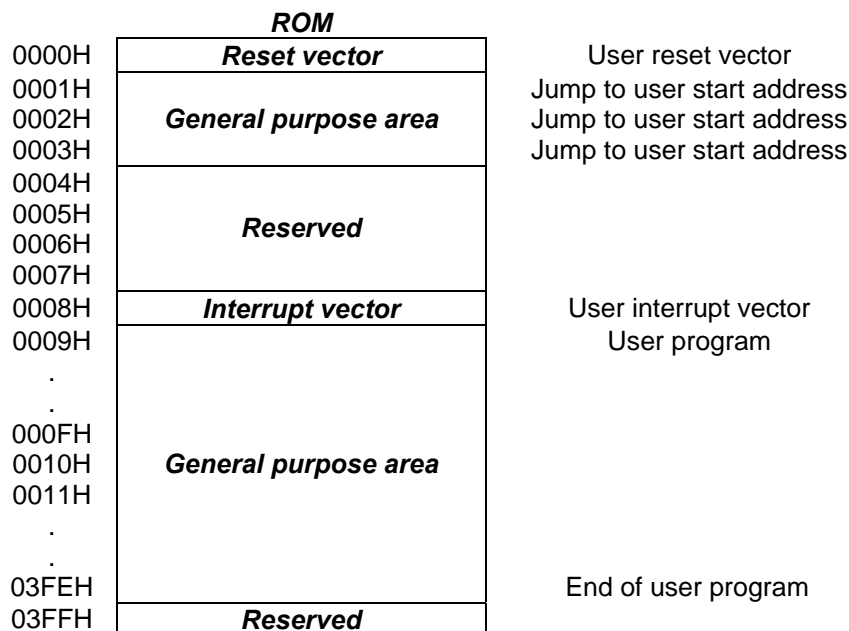
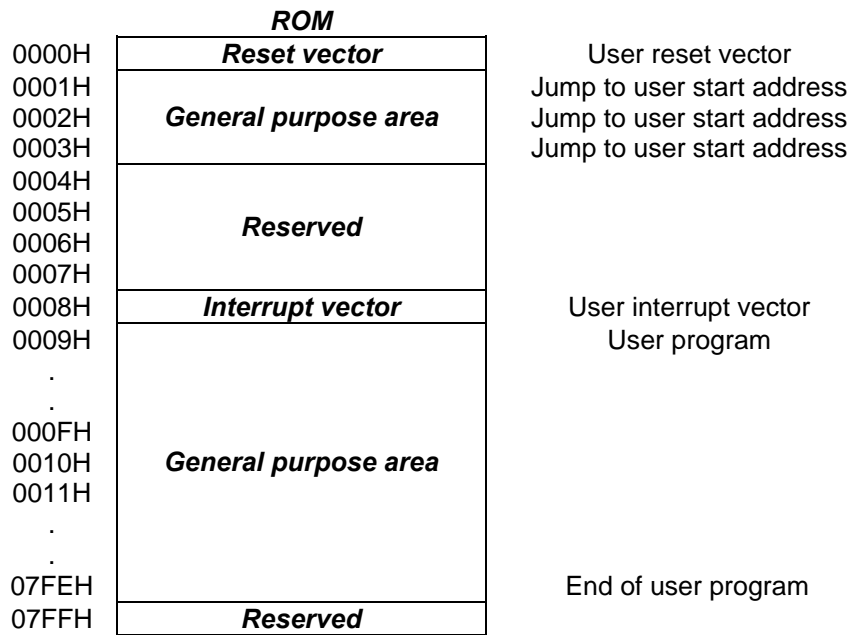
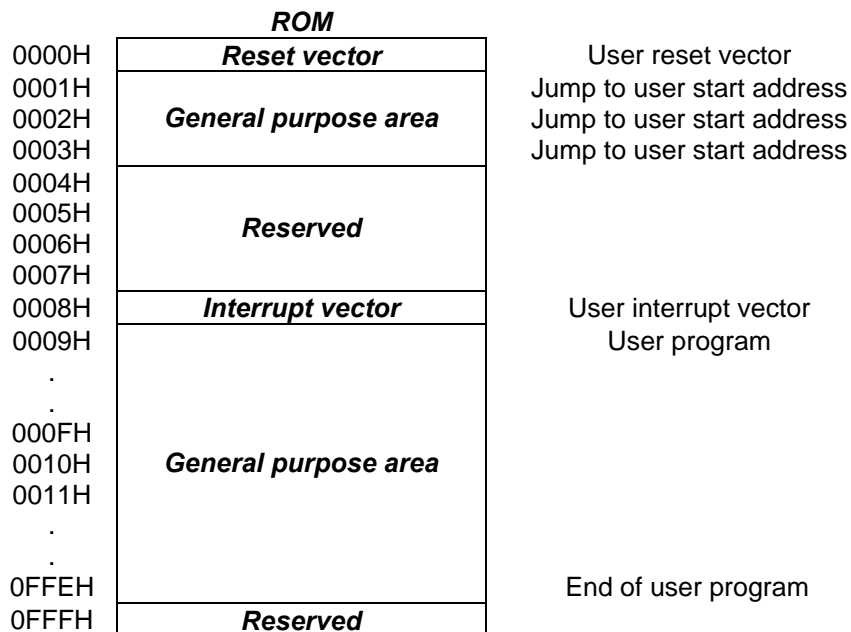


Figure 3-1. ROM Address Structure (SN8P1702)





**Figure 3-2. ROM Address Structure (SN8P1704)**



**Figure 3-3. ROM Address Structure (SN8P1706/SN8P1707/SN8P1708)**

## USER RESET VECTOR ADDRESS (0000H)

A 1-word vector address area is used to execute system reset. After power on 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. The following example shows the way to define the reset vector in the program memory.

⇒ **Example: After power on reset, external reset active or reset by watchdog timer overflow.**

CHIP SN8P1708

```

ORG      0          ; 0000H
JMP      START    ; Jump to user program address.
.          ; 0001H ~ 0007H are reserved

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

```

## INTERRUPT VECTOR ADDRESS (0008H)

A 1-word vector address area is used to execute interrupt request. If any interrupt service is executed, the program counter (PC) value is stored in stack buffer and points 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 1: This demo program includes interrupt service routine and the user program is behind the interrupt service routine.**

CHIP SN8P1708

```

ORG      0          ; 0000H
JMP      START    ; Jump to user program address.
.          ; 0001H ~ 0007H are reserved

ORG      8          ; Interrupt service routine
B0XCH    A, ACCBUF ; B0XCH doesn't change C, Z flag
PUSH                    ; Push 80H ~ 87H system registers
.
.
.
POP                    ; Pop 80H ~ 87H system registers
B0XCH    A, ACCBUF
RETI                    ; End of interrupt service routine

START:   .          ; The head of user program.
.          ; User program
.
.
.
JMP      START    ; End of user program

ENDP                    ; End of program

```

⇒ Example 2: The demo program includes interrupt service routine and the address of interrupt service routine is in a special address of general-purpose area.

CHIP SN8P1708

```

    ORG      0          ; 0000H
    JMP      START     ; Jump to user program address.
    .        ; 0001H ~ 0007H are reserved

    ORG      08
    JMP      MY_IRQ    ; 0008H, Jump to interrupt service routine address

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

MY_IRQ:
    B0XCH   A, ACCBUF ; The head of interrupt service routine
    PUSH   ; B0XCH doesn't change C, Z flag
    .      ; Push 80H ~ 87H system registers
    .
    .
    POP    ; Pop 80H ~ 87H system registers
    B0XCH   A, ACCBUF
    RETI   ; End of interrupt service routine

    ENDP          ; End of program

```

➤ Remark: It is easy to get the rules of SONiX program from demo programs given above. These points are as following.

1. The address 0000H is a "JMP" instruction to make the program go to general-purpose ROM area. The 0004H~0007H are reserved. Users have to skip 0004H~0007H addresses. It is very important and necessary.

2. The interrupt service starts from 0008H. Users can put the whole interrupt service routine from 0008H (Example1) or to put a "JMP" instruction in 0008H then place the interrupt service routine in other general-purpose ROM area (Example2) to get more modularized coding style.

## CHECKSUM CALCULATION

The ROM addresses 0004H~0007H and last address are reserved area. User should avoid these addresses (0004H~0007H and last address) when calculate the Checksum value.

⇒ **Example:**

The demo program shows how to avoid 0004H~0007H when calculated Checksum from 00H to the end of user's code

```

MOV      A,#END_USER_CODE$L
B0MOV   END_ADDR1,A      ;save low end address to end_addr1
MOV      A,#END_USER_CODE$
M
B0MOV   END_ADDR2,A      ;save middle end address to end_addr2
CLR     Y                ;set Y to 00H
CLR     Z                ;set Z to 00H

@@:
CALL    YZ_CHECK        ;call function of check yz value
MOVC
B0BSET  FC              ;clear C flag
ADD     DATA1,A        ;add A to Data1
MOV     A,R
ADC     DATA2,A        ;add R to Data2
JMP     END_CHECK      ;check if the YZ address = the end of code

AAA:
INCMS   Z               ;Z=Z+1
JMP     @B              ;if Z!= 00H calculate to next address
JMP     Y_ADD_1        ;if Z=00H increase Y

END_CHECK:
MOV     A,END_ADDR1
CMPRS   A,Z             ;check if Z = low end address
JMP     AAA            ;if Not jump to checksum calculate
MOV     A,END_ADDR2
CMPRS   A,Y            ;if Yes, check if Y = middle end address
JMP     AAA            ;if Not jump to checksum calculate
JMP     CHECKSUM_END  ;if Yes checksum calculated is done.
YZ_CHECK:
MOV     A,#04H
CMPRS   A,Z            ;check if Z=04H
RET     ;if Not return to checksum calculate
MOV     A,#00H
CMPRS   A,Y            ;if Yes, check if Y=00H
RET     ;if Not return to checksum calculate
INCMS   Z               ;if Yes, increase 4 to Z
INCMS   Z
INCMS   Z
INCMS   Z
RET     ;set YZ=0008H then return

Y_ADD_1:
INCMS   Y               ;increase Y
NOP
JMP     @B              ;jump to checksum calculate

CHECKSUM_END:
.....
.....

END_USER_CODE:        ;Label of program end

```

## GENERAL PURPOSE PROGRAM MEMORY AREA

The 4089<sup>1</sup>-word at ROM locations 0010H~0FFE<sup>H</sup> are used as general-purpose memory. The area is stored instruction's op-code and look-up table data. The SN8P1700 includes jump table function by using program counter (PC) and look-up table function by using ROM code registers (R, X, Y, Z).

The boundary of program memory is separated by the high-byte program counter (PCH) every 100H. In jump table function and look-up table function, the program counter can't leap over the boundary by program counter automatically. Users need to modify the PCH value to "PCH+1" as the PCL overflow (from 0FFH to 000H).

➤ **Notice: 1: The SN8P1702's ROM size is about 1K words and the SN8P1704's ROM size is about 2K words.**

## LOOKUP TABLE DESCRIPTION

In the ROM's data lookup function, the X register is pointed to the highest 8-bit, Y register 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 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              ; Not overflow
INCMS    Y               ; Z overflow (FFH → 00), → Y=Y+1
NOP      ; Not overflow
;
@@:      MOVC             ; To lookup data, R = 51H, ACC = 05H.
;
TABLE1:  .               ;
          DW      0035H   ; To define a word (16 bits) data.
          DW      5105H   ; "
          DW      2012H   ; "

```

➤ **CAUTION: The Y register can't increase automatically if Z register cross boundary from 0xFF to 0x00. Therefore, user must take care such situation to avoid loop-up table errors. If Z register overflow, Y register must be added one. The following INC\_YZ macro shows a simple method to process Y and Z registers automatically.**

➤ **Note: Because the program counter (PC) is only 12-bit, the X register is useless in the application. Users can omit "B0MOV X, #TABLE1\$H". SONiX ICE support more larger program memory addressing capability. So make sure X register is "0" to avoid unpredicted error in loop-up table operation.**

➤ **Example: INC\_YZ Macro**

```

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

          INCMS    Y               ; Y+1
          NOP      ; Not overflow

@@:
          ENDM

```

The other coding style of loop-up table is to add Y or Z index register by accumulator. Be careful if carry happen. Refer following example for detailed information:

➤ **Example: Increase Y and Z register by B0ADD/ADD instruction**

```

B0MOV    Y, #TABLE1$M    ; To set lookup table's middle address.
B0MOV    Z, #TABLE1$L    ; To set lookup table's low address.

B0MOV    A, BUF          ; Z = Z + BUF.
B0ADD    Z, A

B0BTS1   FC              ; Check the carry flag.
JMP      GETDATA        ; FC = 0
INCMS    Y               ; FC = 1. Y+1.
NOP

GETDATA:
MOV      ;
        ; To lookup data. If BUF = 0, data is 0x0035
        ; If BUF = 1, data is 0x5105
        ; If BUF = 2, data is 0x2012
        .
        .
        .
        .
        .
TABLE1:  .
        DW      0035H    ; To define a word (16 bits) data.
        DW      5105H    ; "
        DW      2012H    ; "

```

## 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. The new program counter (PC) points to a series jump instructions as a listing table. The way is easy to make a multi-stage program.

When carry flag occurs after executing of "ADD PCL, A", it will not affect PCH register. Users have to check if the jump table leaps over the ROM page boundary or the listing file generated by SONiX assembly software. If the jump table leaps over the ROM page boundary (e.g. from xxFFH to xx00H), move the jump table to the top of next program memory page (xx00H). **Here one page mean 256 words.**

⇒ **Example : If PC = 0323H (PCH = 03H, PCL = 23H)**

```

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

B0ADD    PCL, A      ; PCL = PCL + ACC, the PCH can't be changed.
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

```

In following example, the jump table starts at 0x00FD. When execute B0ADD PCL, A. If ACC = 0 or 1, the jump table points to the right address. If the ACC is larger then 1 will cause error because PCH doesn't increase one automatically. We can see the PCL = 0 when ACC = 2 but the PCH still keep in 0. The program counter (PC) will point to a wrong address 0x0000 and crash system operation. It is important to check whether the jump table crosses over the boundary (xxFFH to xx00H). A good coding style is to put the jump table at the start of ROM boundary (e.g. 0100H).

⇒ **Example: If "jump table" crosses over ROM boundary will cause errors.**

```

ROM Address
.
.
.
0X00FD    B0ADD    PCL, A      ; PCL = PCL + ACC, the PCH can't be changed.
0X00FE    JMP      A0POINT    ; ACC = 0
0X00FF    JMP      A1POINT    ; ACC = 1
0X0100    JMP      A2POINT    ; ACC = 2 ← jump table cross boundary here
0X0101    JMP      A3POINT    ; ACC = 3
.
.
.

```

SONiX provides a macro for safe jump table function. This macro will check the ROM boundary and move the jump table to the right position automatically. The side effect of this macro is maybe wasting some ROM size. Notice the maximum jmp table number for this macro is limited under 254.

```

@JMP_A    MACRO    VAL
IF        (($+1) !& 0XFF00) != (($+(VAL)) !& 0XFF00)
JMP      ($ | 0XFF)
ORG      ($ | 0XFF)
ENDIF
ADD      PCL, A
ENDM

```

➤ **Note: "VAL" is the number of the jump table listing number.**

⇒ 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     ; 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
JMP      A4POINT     ; ACC = 4, jump to A4POINT
```

If the jump table position is from 00FDH to 0101H, the “@JMP\_A” macro will make the jump table to start from 0100h.



## DATA MEMORY (RAM)

### OVERVIEW

The SN8P1700 has internally built-in the data memory up to 256 bytes for storing the general-purpose data.

For SN8P1702

- 48 \* 8-bit general purpose area in bank 0
- 128 \* 8-bit system special register area

For SN8P1704

- 128 \* 8-bit general purpose area in bank 0
- 128 \* 8-bit system special register area

For SN8P1706/SN8P1707/SN8P1708

- 128 \* 8-bit general purpose area in bank 0
- 128 \* 8-bit general purpose area in bank 1
- 128 \* 8-bit system special register area

The memory is separated into bank 0 and bank 1. The user can program RAM bank selection bits of RBANK register to access all data in any of the two RAM banks. The bank 0, using the first 128-byte location assigned as general-purpose area, and the remaining 128-byte in bank 0 as system register. The bank 1, using the first 128-byte location assigned as general-purpose area, and others useless.

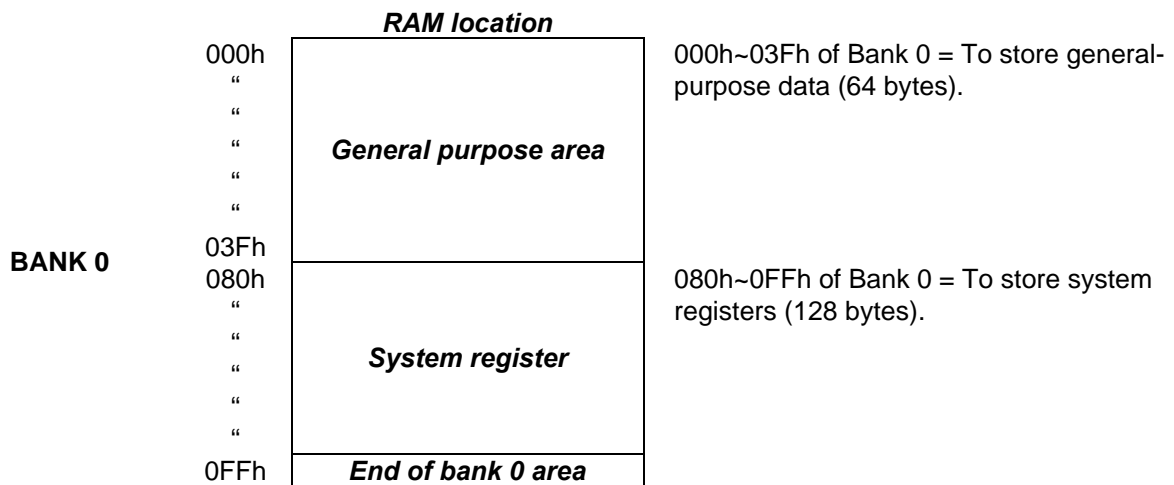
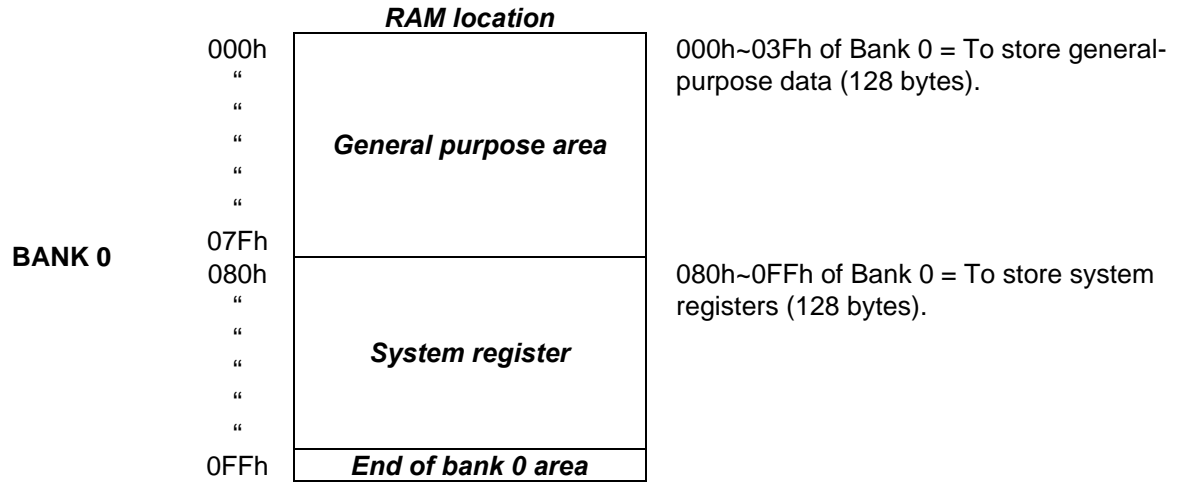
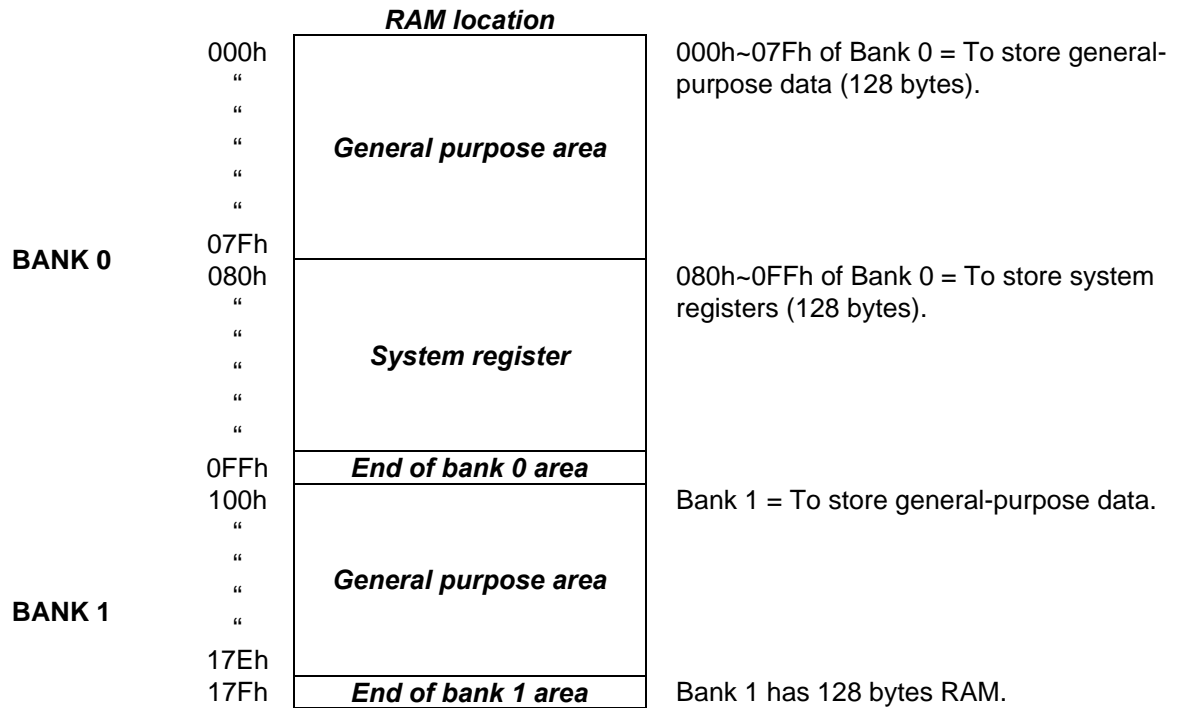


Figure 3-4. RAM Location of SN8P1702



**Figure 3-5. RAM Location of SN8P1704**



**Figure 3-6 RAM Location of SN8P1706/SN8P1707/SN8P1708**

➤ **Note:** The undefined locations of system register area are logic “high” after executing read instruction “MOV A, M”.

## RAM BANK SELECTION

The RBANK is a 1-bit register located at 87H in RAM bank 0. The user can access RAM data by using this register pointing to working RAM bank for ACC to read/write RAM data.

**RBANK initial value = xxxx xxx0**

| 087H         | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0  |
|--------------|-------|-------|-------|-------|-------|-------|-------|--------|
| <b>RBANK</b> | 0     | 0     | 0     | 0     | 0     | 0     | 0     | RBNKS0 |
|              | -     | -     | -     | -     | -     | -     | -     | R/W    |

RBNKS<sub>n</sub>: RAM bank selecting control bit. 0 = bank 0, 1 = bank 1.

### ➤ Example: RAM bank selecting.

```

; BANK 0
        CLR        RBANK        ; b0bclr FRBNKS0
        .
; BANK 1
        MOV        A, #1        ; b0bset FRBNKS0
        B0MOV     RBANK, A
        .
    
```

➤ **Note: “B0MOV” instruction can access the RAM of bank 0 in other bank situation directly.**

### ➤ Example: Access RAM bank 0 in RAM bank 1.

```

; BANK 1
        B0BSET    RBNKS0        ; Get into RAM bank 1
        B0MOV     A, BUF0        ; Read BUF0 data. BUF0 is in RAM bank0.
        MOV       BUF1, A        ; Write BUF0 data to BUF1. BUF1 is in RAM bank1.
        .
        .
        MOV       A, BUF1        ; Read BUF1(bank1) data and store in ACC.
        B0MOV     BUF0, A        ; Write ACC data to BUF0(bank0).
    
```

Under bank 1 situation, using “B0MOV” instruction is an easy way to access RAM bank 0 data. User can make a habit to read/write system register (0087H~00FFH). Then user can access system registers without switching RAM bank.

### ➤ Example: To Access the system registers in bank 1 situation.

```

; BANK 1
        B0BSET    RBNKS0        ; Switch the Ram Bank into bank 1
        .
        MOV       A, #0FFH       ; Set all pins of P1 to be logic high.
        B0MOV     P1, A          ; Operate the bank 0 special register by the b0mov instruction
        .                        ; while the RAM system in the bank1.
        B0MOV     A, P0          ; Read P0 data in the Bank 0 and store into BUF1 in the bank 1.
        MOV       BUF1, A        ;
    
```

## WORKING REGISTERS

The locations 80H to 85H of RAM bank 0 in data memory stores the specially defined registers such as register H, L, R, X, Y, Z, respectively shown in the following table. These registers can use as the general purpose of working buffer and be used to access ROM's and RAM's data. For instance, all of the ROM's table can be looked-up with R, X, Y and Z registers. The data of RAM memory can be indirectly accessed with H, L, Y and Z registers.

|            |     |     |     |     |     |     |  |
|------------|-----|-----|-----|-----|-----|-----|--|
|            | 80H | 81H | 82H | 83H | 84H | 85H |  |
| <b>RAM</b> | L   | H   | R   | Z   | Y   | X   |  |
|            | R/W | R/W | R/W | R/W | R/W | R/W |  |

## H, L REGISTERS

The H and L are 8-bit register with two major functions. One is to use the registers as working register. The other is to use the registers as data pointer to access RAM's data. The @HL that is data point\_0 index buffer located at address E6H in RAM bank\_0. It employs H and L registers to addressing RAM location in order to read/write data through ACC. The Lower 4-bit of H register is pointed to RAM bank number and L register is pointed to RAM address number, respectively. The higher 4-bit data of H register is truncated in RAM indirectly access mode.

**H initial value = 0000 0000**

|          |       |       |       |       |       |       |       |       |
|----------|-------|-------|-------|-------|-------|-------|-------|-------|
| 081H     | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| <b>H</b> | HBIT7 | HBIT6 | HBIT5 | HBIT4 | HBIT3 | HBIT2 | HBIT1 | HBIT0 |
|          | R/W   | R/W   | R/W   | R/W   | R/W   | R/W   | R/W   | R/W   |

**L initial value = 0000 0000**

|          |       |       |       |       |       |       |       |       |
|----------|-------|-------|-------|-------|-------|-------|-------|-------|
| 080H     | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| <b>L</b> | LBIT7 | LBIT6 | LBIT5 | LBIT4 | LBIT3 | LBIT2 | LBIT1 | LBIT0 |
|          | R/W   | R/W   | R/W   | R/W   | R/W   | R/W   | R/W   | R/W   |

⇒ **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    H, #00H        ; To set RAM bank 0 for H register
B0MOV    L, #20H        ; To set location 20H for L register
B0MOV    A, @HL         ; To read a data into ACC
    
```

⇒ **Example: Clear general-purpose data memory area of bank 0 using @HL register.**

```

CLR      H              ; H = 0, bank 0
MOV      A, #07FH
B0MOV    L, A           ; L = 7FH, the last address of the data memory area
CLR_HL_BUF:
CLR      @HL            ; Clear @HL to be zero
DECMS   L              ; L - 1, if L = 0, finish the routine
JMP     CLR_HL_BUF     ; Not zero

CLR      @HL
END_CLR: ; End of clear general purpose data memory area of bank 0
.
.
.
    
```

## 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 be address ROM location in order to look-up ROM data.

**Y initial value = 0000 0000**

| 084H     | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|----------|-------|-------|-------|-------|-------|-------|-------|-------|
| <b>Y</b> | YBIT7 | YBIT6 | YBIT5 | YBIT4 | YBIT3 | YBIT2 | YBIT1 | YBIT0 |
|          | R/W   | R/W   | R/W   | R/W   | R/W   | R/W   | R/W   | R/W   |

**Z initial value = 0000 0000**

| 083H     | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|----------|-------|-------|-------|-------|-------|-------|-------|-------|
| <b>Z</b> | ZBIT7 | ZBIT6 | ZBIT5 | ZBIT4 | ZBIT3 | ZBIT2 | ZBIT1 | ZBIT0 |
|          | R/W   | R/W   | R/W   | R/W   | R/W   | R/W   | R/W   | R/W   |

The @YZ that is data point\_1 index buffer located at address E7H in RAM bank 0. It employs Y and Z registers to addressing RAM location in order to read/write data through ACC. The Lower 4-bit of Y register is pointed to RAM bank number and Z register is pointed to RAM address number, respectively. The higher 4-bit data of Y register is truncated in RAM indirectly access mode.

➤ **Example: If want to read a data from RAM address 25H of bank 1, it can use indirectly addressing mode to access data as following.**

```
B0MOV    Y, #01H      ; To set RAM bank 1 for Y register
B0MOV    Z, #25H      ; To set location 25H for Z register
B0MOV    A, @YZ       ; To read a data into ACC
```

➤ **Example: Clear general-purpose data memory area of bank 1 using @YZ register.**

```
MOV      A, #1
B0MOV    Y, A          ; Y = 1, bank 1
MOV      A, #07FH
B0MOV    Z, A          ; Y = 7FH, the last address of the data memory area
```

CLR\_YZ\_BUF:

```
CLR      @YZ          ; Clear @YZ to be zero

DECMS    Z            ; Y - 1, if Y= 0, finish the routine
JMP      CLR_YZ_BUF   ; Not zero
```

```
CLR      @YZ
END_CLR: ; End of clear general purpose data memory area of bank 0
```

➤ **Note: Please consult the “LOOK-UP TABLE DESCRIPTION” about Y, Z register look-up table application.**

## X REGISTERS

There are two major functions of the X register. First, X register can be used as working registers. Second, the X registers must be clear in order to look-up the ROM data. The SN8P1700's program counter only has 12-bit. In look-up table function, the users can omit X register.

**X initial value = 0000 0000**

| 085H     | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|----------|-------|-------|-------|-------|-------|-------|-------|-------|
| <b>X</b> | XBIT7 | XBIT6 | XBIT5 | XBIT4 | XBIT3 | XBIT2 | XBIT1 | XBIT0 |
|          | R/W   | R/W   | R/W   | R/W   | R/W   | R/W   | R/W   | R/W   |

➤ **Note: Please consult the “LOOK-UP TABLE DESCRIPTION” about X register look-up table application.**

## R REGISTERS

There are two major functions of the R register. First, R register can be used as working registers. Second, the R registers can be store high-byte data of look-up ROM data. After MOVC instruction executed, the high-byte data of a ROM address will be stored in R register and the low-byte data stored in ACC.

**R initial value = 0000 0000**

| 082H     | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|----------|-------|-------|-------|-------|-------|-------|-------|-------|
| <b>R</b> | RBIT7 | RBIT6 | RBIT5 | RBIT4 | RBIT3 | RBIT2 | RBIT1 | RBIT0 |
|          | R/W   | R/W   | R/W   | R/W   | R/W   | R/W   | R/W   | R/W   |

➤ **Note: Please consult the “LOOK-UP TABLE DESCRIPTION” about R register look-up table application.**

## PROGRAM FLAG

The PFLAG includes carry flag (C), decimal carry flag (DC) and zero flag (Z). If the result of operating is zero or there is carry, borrow occurrence, then these flags will be set to PFLAG register.

**PFLAG initial value = xxxx x000**

| 086H         | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|--------------|-------|-------|-------|-------|-------|-------|-------|-------|
| <b>PFLAG</b> | -     | -     | -     | -     | -     | C     | DC    | Z     |
|              | -     | -     | -     | -     | -     | R/W   | R/W   | R/W   |

## 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".

## DECIMAL CARRY FLAG

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

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

## ZERO FLAG

Z = 1: After operation, the content of ACC is zero.

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

## 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.

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
.
```

The PUSH and POP instructions don't store ACC value as any interrupt service executed. ACC must be exchanged to another data memory defined by users. Thus, once interrupt occurs, these data must be stored in the data memory based on the user's program as follows.

### ⇒ Example: ACC and working registers protection.

```
ACCBUF     EQU     00H           ; ACCBUF is ACC data buffer in bank 0.
```

```
INT_SERVICE:
```

```
  B0XCH    A, ACCBUF           ; B0XCH doesn't change C, Z flag
```

```
  PUSH.    .                   ; Push instruction
```

```
  .
```

```
  .
```

```
  POP      .                   ; Pop instruction
```

```
  B0XCH    A, ACCBUF           ; Re-load ACC
```

```
  RETI     .                   ; Exit interrupt service vector
```

➤ **Notice:** To save and re-load ACC data must be used "B0XCH" instruction, or the PLAGE value maybe modified by ACC.

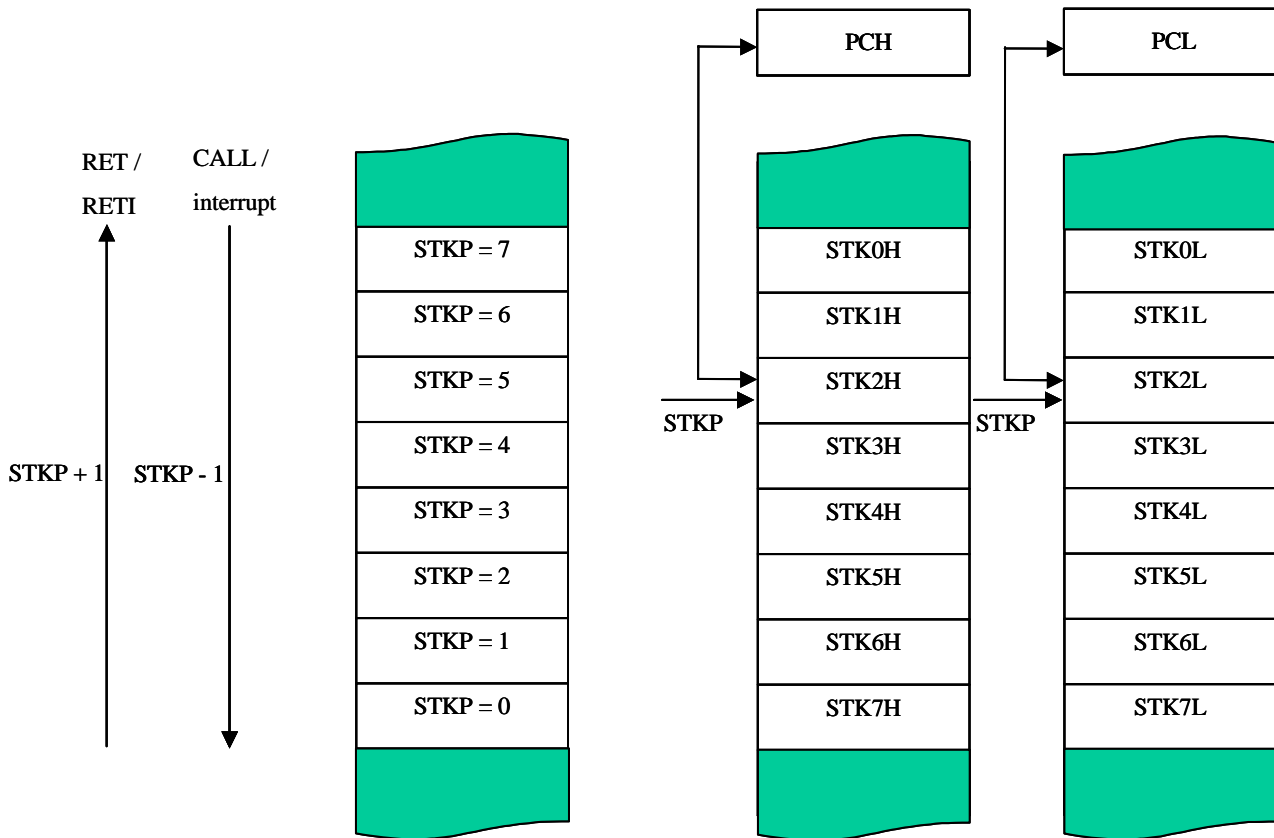


# STACK OPERATIONS

## OVERVIEW

The stack buffer of SN8P1700 has 8-level high area and each level is 12-bits length. This buffer is designed to save and restore program counter's (PC) data when interrupt service is executed. The STKP register is a pointer designed to point active level in order to save or restore data from stack buffer for kernel circuit. The STKnH and STKnL are the 12-bit stack buffers to store program counter (PC) data.

### STACK BUFFER



**Figure 3-7 Stack-Save and Stack-Restore Operation**

## STACK REGISTERS

The stack pointer (STKP) is a 4-bit register to store the address used to access the stack buffer, 12-bits 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 (Stack-Save) and reading (Stack-Restore) from the top of stack. Stack-Save operation decrements the STKP and the Stack-Restore operation increments one time. That makes the STKP always points to the top address of stack buffer and writes 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.

**STKP (stack pointer) initial value = 0xxx 1111**

| 0DFH        | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3  | Bit 2  | Bit 1  | Bit 0  |
|-------------|-------|-------|-------|-------|--------|--------|--------|--------|
| <b>STKP</b> | GIE   | -     | -     | -     | STKPB3 | STKPB2 | STKPB1 | STKPB0 |
|             | R/W   | -     | -     | -     | R/W    | R/W    | R/W    | R/W    |

STKPBn: Stack pointer. (n = 0 ~ 3)

GIE: Global interrupt control bit. 0 = disable, 1 = enable. More detail information is in interrupt chapter.

⇒ **Example: Stack pointer (STKP) reset routine.**

```
MOV      A, #00001111B
B0MOV   STKP, A
```

**STKn (stack buffer) initial value = xxxx xxxx xxxx xxxx, STKn = STKnH + STKnL (n = 7 ~ 0)**

| 0F0H~0FFH    | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3  | Bit 2  | Bit 1 | Bit 0 |
|--------------|-------|-------|-------|-------|--------|--------|-------|-------|
| <b>STKnH</b> | -     | -     | -     | -     | SnPC11 | SnPC10 | SnPC9 | SnPC8 |
|              | -     | -     | -     | -     | R/W    | R/W    | R/W   | R/W   |

| 0F0H~0FFH    | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|--------------|-------|-------|-------|-------|-------|-------|-------|-------|
| <b>STKnL</b> | SnPC7 | SnPC6 | SnPC5 | SnPC4 | SnPC3 | SnPC2 | SnPC1 | SnPC0 |
|              | R/W   | R/W   | R/W   | R/W   | R/W   | R/W   | R/W   | R/W   |

STKnH: Store PCH data as interrupt or call executing. The n expressed 0 ~7.

STKnL: Store PCL data as interrupt or call executing. The n expressed 0 ~7.

## STACK OPERATION EXAMPLE

The two kinds of Stack-Save operations to reference the stack pointer (STKP) and write the program counter contents (PC) into the stack buffer are CALL instruction and interrupt service. Under each condition, the STKP is decremented 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 following table.

| Stack Level | STKP Register |        |        |        | Stack Buffer |          | Description           |
|-------------|---------------|--------|--------|--------|--------------|----------|-----------------------|
|             | STKPB3        | STKPB2 | STKPB1 | STKPB0 | High Byte    | Low Byte |                       |
| 0           | 1             | 1      | 1      | 1      | STK0H        | STK0L    | -                     |
| 1           | 1             | 1      | 1      | 0      | STK1H        | STK1L    | -                     |
| 2           | 1             | 1      | 0      | 1      | STK2H        | STK2L    | -                     |
| 3           | 1             | 1      | 0      | 0      | STK3H        | STK3L    | -                     |
| 4           | 1             | 0      | 1      | 1      | STK4H        | STK4L    | -                     |
| 5           | 1             | 0      | 1      | 0      | STK5H        | STK5L    | -                     |
| 6           | 1             | 0      | 0      | 1      | STK6H        | STK6L    | -                     |
| 7           | 1             | 0      | 0      | 0      | STK7H        | STK7L    | -                     |
| >8          | -             | -      | -      | -      | -            | -        | <b>Stack Overflow</b> |

**Table 3-1. STKP, STKnH and STKnL relative of Stack-Save Operation**

There is a Stack-Restore operation corresponding each push operation to restore the program counter (PC). The RETI instruction is for interrupt service routine. The RET instruction is for CALL instruction. When a Stack-Restore operation occurs, the STKP is incremented and points to the next free stack location. The stack buffer restores the last program counter (PC) to the program counter registers. The Stack-Restore operation is as following table.

| Stack Level | STKP Register |        |        |        | Stack Buffer |          | Description |
|-------------|---------------|--------|--------|--------|--------------|----------|-------------|
|             | STKPB3        | STKPB2 | STKPB1 | STKPB0 | High Byte    | Low Byte |             |
| 7           | 1             | 0      | 0      | 0      | STK7H        | STK7L    | -           |
| 6           | 1             | 0      | 0      | 1      | STK6H        | STK6L    | -           |
| 5           | 1             | 0      | 1      | 0      | STK5H        | STK5L    | -           |
| 4           | 1             | 0      | 1      | 1      | STK4H        | STK4L    | -           |
| 3           | 1             | 1      | 0      | 0      | STK3H        | STK3L    | -           |
| 2           | 1             | 1      | 0      | 1      | STK2H        | STK2L    | -           |
| 1           | 1             | 1      | 1      | 0      | STK1H        | STK1L    | -           |
| 0           | 1             | 1      | 1      | 1      | STK0H        | STK0L    | -           |

**Table 3-2. STKP, STKnH and STKnL relative of Stack-Restore Operation**

## PROGRAM COUNTER

The program counter (PC) is a 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.

**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 |
|-----------|--------|--------|--------|--------|--------|--------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| <b>PC</b> | -      | -      | -      | -      | 0      | 0      | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     |
|           | PCH    |        |        |        |        |        |       |       | PCL   |       |       |       |       |       |       |       |

**PCH Initial value = xxxx 0000**

| 0CFH       | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|------------|-------|-------|-------|-------|-------|-------|-------|-------|
| <b>PCH</b> | -     | -     | -     | -     | PC11  | PC10  | PC9   | PC8   |
|            | -     | -     | -     | -     | R/W   | R/W   | R/W   | R/W   |

**PCL Initial value = 0000 0000**

| 0CEH       | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|------------|-------|-------|-------|-------|-------|-------|-------|-------|
| <b>PCL</b> | PC7   | PC6   | PC5   | PC4   | PC3   | PC2   | PC1   | PC0   |
|            | R/W   | R/W   | R/W   | R/W   | R/W   | R/W   | R/W   | R/W   |

## ONE ADDRESS SKIPPING

There are 9 instructions (CMPRS, INCS, INCMS, DECS, DECMS, BTS0, BTS1, B0BTS0, B0BTS1) with one address skipping function. If the result of these instructions is matched, the PC will add 2 steps to skip next instruction.

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

```

                B0BTS1   FC           ; Skip next instruction, if Carry_flag = 1
                JMP      C0STEP      ; Else jump to C0STEP.
C0STEP:        .
                NOP
                B0MOV    A, BUF0     ; Move BUF0 value to ACC.
                B0BTS0   FZ           ; Skip next instruction, if Zero flag = 0.
                JMP      C1STEP      ; Else jump to C1STEP.
C1STEP:        .
                NOP

```

*If the ACC is equal to the immediate data or memory, the PC will add 2 steps to skip next instruction.*

```

                CMPRS    A, #12H     ; Skip next instruction, if ACC = 12H.
                JMP      C0STEP      ; Else jump to C0STEP.
C0STEP:        .
                NOP

```

*If the result after increasing or decreasing by 1 is 0xFF or 0x00, 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 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 = 0323H (PCH = 03H, PCL = 23H)

```
; PC = 0323H
MOV      A, #28H
B0MOV   PCL, A           ; Jump to address 0328H
.
.
; PC = 0328H
MOV      A, #00H
B0MOV   PCL, A           ; Jump to address 0300H
```

⇒ Example: If PC = 0323H (PCH = 03H, PCL = 23H)

```
; PC = 0323H
B0ADD   PCL, A           ; PCL = PCL + ACC, the PCH cannot 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
.
.
;
```

# 4 ADDRESSING MODE

## OVERVIEW

The SN8P1700 provides three addressing modes to access RAM data, including immediate addressing mode, directly addressing mode and indirectly address mode. The main purpose of the three different modes is described in the following:

### IMMEDIATE ADDRESSING MODE

The immediate addressing mode uses an immediate data to set up the location (MOV A, #I, B0MOV M,#I) in ACC or specific RAM.

#### *Immediate addressing mode*

```
MOV      A, #12H      ; To set an immediate data 12H into ACC
```

### DIRECTLY ADDRESSING MODE

The directly addressing mode uses address number to access memory location (MOV A,12H, B0MOV 12H,A).

#### *Directly addressing mode*

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

### INDIRECTLY ADDRESSING MODE

The indirectly addressing mode is to set up an address in data pointer registers (Y/Z) and uses MOV instruction to read/write data between ACC and @YZ register (MOV A,@YZ, B0MOV @YZ,A).

#### ➔ Example: Indirectly addressing mode with @YZ register

```

CLR      Y              ; To clear Y register to access RAM bank 0.
B0MOV    Z, #12H       ; To set an immediate data 12H into Z register.
B0MOV    A, @YZ        ; Use data pointer @YZ reads a data from RAM location
                        ; 012H into ACC.

MOV      A, #01H       ; To set Y = 1 for accessing RAM bank 1.
B0MOV    Y, A          ; To set an immediate data 12H into Z register.
B0MOV    Z, #12H       ; Use data pointer @YZ reads a data from RAM location
B0MOV    A, @YZ        ; 012H into ACC.

MOV      A, #0FH       ; To set Y = 15 for accessing RAM bank 15.
B0MOV    Y, A          ; To set an immediate data 12H into Z register.
B0MOV    Z, #12H       ; Use data pointer @YZ reads a data from RAM location
B0MOV    A, @YZ        ; 012H into ACC.

```

## TO ACCESS DATA in RAM BANK 0

In the RAM bank 0, this area memory can be read/written by these three access methods.

⇒ **Example 1: To use RAM bank0 dedicate instruction (Such as B0xxx instruction).**

```
B0MOV    A, 12H           ; To move content from location 12H of RAM bank 0 to ACC
```

⇒ **Example 2: To use directly addressing mode (Through RBANK register).**

```
B0MOV    RBANK, #00H     ; To set RAM bank = 0
MOV      A, 12H          ; To move content from location 12H of RAM bank 0 to ACC
```

⇒ **Example 3: To use indirectly addressing mode with @YZ register.**

```
CLR      Y               ; To clear Y register for accessing RAM bank 0.
B0MOV    Z, #12H         ; To set an immediate data 12H into Z register.
B0MOV    A, @YZ          ; Use data pointer @YZ reads a data from RAM location
                        ; 012H into ACC.
```

## TO ACCESS DATA in RAM BANK 1

In the RAM bank 1, this area memory can be read/written by these two access methods.

⇒ **Example 1: To use directly addressing mode (Through RBANK register).**

```
B0MOV    RBANK, #01H     ; To set RAM bank = 1
MOV      A, 12H          ; To move content from location 12H of RAM bank 0 to ACC
```

⇒ **Example 2: To use indirectly addressing mode with @YZ register.**

```
MOV      A, #01H         ; To set Y = 1 for accessing RAM bank 1.
B0MOV    Y, A
B0MOV    Z, #12H         ; To set an immediate data 12H into Z register.
B0MOV    A, @YZ          ; Use data pointer @YZ reads a data from RAM location
                        ; 012H into ACC.
```



# 5 SYSTEM REGISTER

## OVERVIEW

The system special register is located at 80h~FFh. The main purpose of system registers is to control the peripheral hardware of the chip. Using system registers can control I/O ports, SIO, ADC, PWM, timers and counters by programming. The Memory map provides an easy and quick reference source for writing application program. To accessing these system registers is controlled by the select memory bank (RBANK = 0) or the bank 0 read/write instruction (B0MOV, B0BSET, B0BCLR...).

## SYSTEM REGISTER ARRANGEMENT (BANK 0)

### BYTES of SYSTEM REGISTER

SN8P1702

|   | 0    | 1    | 2    | 3    | 4    | 5    | 6     | 7    | 8     | 9     | A    | B    | C    | D    | E    | F    |
|---|------|------|------|------|------|------|-------|------|-------|-------|------|------|------|------|------|------|
| 8 | -    | -    | R    | Z    | Y    | -    | PFLAG | -    | -     | -     | -    | -    | -    | -    | -    | -    |
| 9 | -    | -    | -    | -    | -    | -    | -     | -    | -     | -     | -    | -    | -    | -    | -    | -    |
| A | -    | -    | -    | -    | -    | -    | -     | -    | -     | -     | -    | -    | -    | -    | -    | -    |
| B | -    | ADM  | ADB  | ADR  | -    | -    | -     | -    | -     | -     | -    | -    | -    | -    | -    | -    |
| C | P1W  | P1M  | -    | -    | P4M  | P5M  | -     | -    | INTRQ | INTEN | OSCM | -    | -    | TC0R | PCL  | PCH  |
| D | P0   | P1   | -    | -    | P4   | P5   | -     | -    | -     | -     | TC0M | TC0C | -    | -    | -    | STKP |
| E | -    | -    | -    | -    | -    | -    | -     | @YZ  | -     | -     | -    | -    | -    | -    | -    | -    |
| F | STK7 | STK7 | STK6 | STK6 | STK5 | STK5 | STK4  | STK4 | STK3  | STK3  | STK2 | STK2 | STK1 | STK1 | STK0 | STK0 |

Table 5-1. System Register Arrangement of SN8P1702

SN8P1704

|   | 0    | 1    | 2    | 3    | 4    | 5    | 6     | 7    | 8     | 9     | A    | B    | C    | D    | E    | F    |
|---|------|------|------|------|------|------|-------|------|-------|-------|------|------|------|------|------|------|
| 8 | -    | -    | R    | Z    | Y    | -    | PFLAG | -    | -     | -     | -    | -    | -    | -    | -    | -    |
| 9 | -    | -    | -    | -    | -    | -    | -     | -    | -     | -     | -    | -    | -    | -    | -    | -    |
| A | -    | -    | -    | -    | -    | -    | -     | -    | -     | -     | -    | -    | -    | -    | -    | -    |
| B | DAM  | ADM  | ADB  | ADR  | SIOM | SIOR | SIOB  | -    | -     | -     | -    | -    | -    | -    | -    | -    |
| C | P1W  | P1M  | -    | -    | P4M  | P5M  | -     | -    | INTRQ | INTEN | OSCM | -    | -    | TC0R | PCL  | PCH  |
| D | P0   | P1   | -    | -    | P4   | P5   | -     | -    | -     | -     | TC0M | TC0C | TC1M | TC1C | TC1R | STKP |
| E | -    | -    | -    | -    | -    | -    | -     | @YZ  | -     | -     | -    | -    | -    | -    | -    | -    |
| F | STK7 | STK7 | STK6 | STK6 | STK5 | STK5 | STK4  | STK4 | STK3  | STK3  | STK2 | STK2 | STK1 | STK1 | STK0 | STK0 |

Table 5-2. System Register Arrangement of SN8P1704

## SN8P1706/SN8P1707/SN8P1708

|   | 0     | 1     | 2     | 3     | 4     | 5     | 6     | 7     | 8     | 9     | A     | B     | C     | D     | E     | F     |
|---|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| 8 | L     | H     | R     | Z     | Y     | X     | PFLAG | RBANK | -     | -     | -     | -     | -     | -     | -     | -     |
| 9 | -     | -     | -     | -     | -     | -     | -     | -     | -     | -     | -     | -     | -     | -     | -     | -     |
| A | -     | -     | -     | -     | -     | -     | -     | -     | -     | -     | -     | -     | -     | -     | -     | -     |
| B | DAM   | ADM   | ADB   | ADR   | SIOM  | SIOR  | SIOB  | -     | -     | -     | -     | -     | -     | -     | -     | -     |
| C | P1W   | P1M   | P2M   | -     | P4M   | P5M   | -     | -     | INTRQ | INTEN | OSCM  | -     | -     | TC0R  | PCL   | PCH   |
| D | P0    | P1    | P2    | -     | P4    | P5    | -     | -     | T0M   | T0C   | TC0M  | TC0C  | TC1M  | TC1C  | TC1R  | STKP  |
| E | -     | -     | -     | -     | -     | -     | @HL   | @YZ   | -     | -     | -     | -     | -     | -     | -     | -     |
| F | STK7L | STK7H | STK6L | STK6H | STK5L | STK5H | STK4L | STK4H | STK3L | STK3H | STK2L | STK2H | STK1L | STK1H | STK0L | STK0H |

Table 5-3. System Register Arrangement of SN8P1706/SN8P1707/SN8P1708

**Description**

L, H = Working & @HL addressing register.  
 X = Working and ROM address register.  
 PFLAG = ROM page and special flag register.  
 DAM = DAC's mode register.  
 ADB = ADC's data buffer.  
 SIOM = SIO mode control register.  
 SIOB = SIO's data buffer.  
 PnM = Port n input/output mode register.  
 INTRQ = Interrupts' request register.  
 OSCM = Oscillator mode register.  
 T0M = Timer 0 mode register.  
 T0C = Timer 0 counting register.  
 TC1M = Timer/Counter 1 mode register.  
 TC1C = Timer/Counter 1 counting register.  
 STKP = Stack pointer buffer.  
 @HL = RAM HL indirect addressing index pointer.

R = Working register and ROM lookup data buffer.  
 Y, Z = Working, @YZ and ROM addressing register.  
 RBANK = RAM Bank Select register.  
 ADM = ADC's mode register.  
 ADR = ADC's resolution selects register.  
 SIOR = SIO's clock reload buffer.  
 P1W = Port 1 wakeup register.  
 Pn = Port n data buffer.  
 INTEN = Interrupts' enable register.  
 PCH, PCL = Program counter.  
 TC0M = Timer/Counter 0 mode register.  
 TC0C = Timer/Counter 0 counting register.  
 TC0R = Timer/Counter 0 auto-reload data buffer.  
 TC1R = Timer/Counter 1 auto-reload data buffer.  
 STK0~STK7 = Stack 0 ~ stack 7 buffer.  
 @YZ = RAM YZ indirect addressing index pointer.

**Note:**

- All of register names had been declared in SONiX 8-bit MCU assembler.
- One-bit name had been declared in SONiX 8-bit MCU assembler with "F" prefix code.
- It will get logic "H" data, when use instruction to check empty location.
- The low nibble of ADR register is read only.
- "b0bset", "b0bclr", "bset", "bclr" instructions only support "R/W" registers.

## BITS of SYSTEM REGISTER

**SN8P1702 System register table**

| 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    | -      | -        | -        | -        | -      | C      | DC     | Z       | R/W | PFLAG               |
| 0B1H    | ADENB  | ADS      | EOC      | GCHS     | -      | 0      | CHS1   | CHS0    | R/W | ADM mode register   |
| 0B2H    | ADB11  | ADB10    | ADB9     | ADB8     | ADB7   | ADB6   | ADB5   | ADB4    | R   | ADB data buffer     |
| 0B3H    | -      | ADCKS    | ADLEN    | 0        | ADB3   | ADB2   | ADB1   | ADB0    | R/W | ADR register        |
| 0C0H    | 0      | 0        | 0        | 0        | 0      | 0      | P11W   | P10W    | W   | P1W wakeup register |
| 0C1H    | 0      | 0        | 0        | 0        | 0      | 0      | P11M   | P10M    | R/W | P1M I/O direction   |
| 0C4H    | 0      | 0        | 0        | 0        | P43M   | P42M   | P41M   | P40M    | R/W | P4M I/O direction   |
| 0C5H    | 0      | 0        | 0        | P54M     | P53M   | P52M   | P51M   | P50M    | R/W | P5M I/O direction   |
| 0C8H    | 0      | 0        | TC0IRQ   | 0        | 0      | 0      | 0      | P00IRQ  | R/W | INTRQ               |
| 0C9H    | 0      | 0        | TC0IEN   | 0        | 0      | 0      | 0      | P00IEN  | R/W | INTEN               |
| 0CAH    | 0      | WDRST    | WDRate   | 0        | CPUM0  | CLKMD  | STPHX  | 0       | R/W | OSCM                |
| 0CDH    | TC0R7  | TC0R6    | TC0R5    | TC0R4    | TC0R3  | TC0R2  | TC0R1  | TC0R0   | W   | TC0R                |
| 0CEH    | PC7    | PC6      | PC5      | PC4      | PC3    | PC2    | PC1    | PC0     | R/W | PCL                 |
| 0CFH    | -      | -        | -        | -        | -      | PC10   | PC9    | PC8     | R/W | PCH                 |
| 0D0H    | -      | -        | -        | -        | -      | -      | -      | P00     | R   | P0 data buffer      |
| 0D1H    | -      | -        | -        | -        | -      | -      | P11    | P10     | R/W | P1 data buffer      |
| 0D4H    | -      | -        | -        | -        | P43    | P42    | P41    | P40     | R/W | P4 data buffer      |
| 0D5H    | -      | -        | -        | P54      | P53    | P52    | P51    | P50     | R/W | P5 data buffer      |
| 0DAH    | TC0ENB | TC0rate2 | TC0rate1 | TC0rate0 | 0      | ALOAD0 | TC0OUT | PWM0OUT | R/W | TC0M                |
| 0DBH    | TC0C7  | TC0C6    | TC0C5    | TC0C4    | TC0C3  | TC0C2  | TC0C1  | TC0C0   | R/W | TC0C                |
| 0DFH    | GIE    | -        | -        | -        | STKPB3 | STKPB2 | STKPB1 | STKPB0  | R/W | STKP stack pointer  |
| 0E7H    | @YZ7   | @YZ6     | @YZ5     | @YZ4     | @YZ3   | @YZ2   | @YZ1   | @YZ0    | R/W | @YZ index pointer   |
| 0F0H    | S7PC7  | S7PC6    | S7PC5    | S7PC4    | S7PC3  | S7PC2  | S7PC1  | S7PC0   | R/W | STK7L               |
| 0F1H    | -      | -        | -        | -        | -      | S7PC10 | S7PC9  | S7PC8   | R/W | STK7H               |
| 0F2H    | S6PC7  | S6PC6    | S6PC5    | S6PC4    | S6PC3  | S6PC2  | S6PC1  | S6PC0   | R/W | STK6L               |
| 0F3H    | -      | -        | -        | -        | -      | S6PC10 | S6PC9  | S6PC8   | R/W | STK6H               |
| "       | "      | "        | "        | "        | "      | "      | "      | "       | "   | "                   |
| "       | "      | "        | "        | "        | "      | "      | "      | "       | "   | "                   |
| "       | "      | "        | "        | "        | "      | "      | "      | "       | "   | "                   |
| 0FCH    | S1PC7  | S1PC6    | S1PC5    | S1PC4    | S1PC3  | S1PC2  | S1PC1  | S1PC0   | R/W | STK1L               |
| 0FDH    | -      | -        | -        | -        | -      | S1PC10 | S1PC9  | S1PC8   | R/W | STK1H               |
| 0FEH    | S0PC7  | S0PC6    | S0PC5    | S0PC4    | S0PC3  | S0PC2  | S0PC1  | S0PC0   | R/W | STK0L               |
| 0FFH    | -      | -        | -        | -        | -      | S0PC10 | S0PC9  | S0PC8   | R/W | STK0H               |

**Table 5-4. Bit System Register Table of SN8P1702**

**Note:**

- a). To avoid system error, please be sure to put all the "0" as it indicates in the above table
- b). All of register name had been declared in SONiX 8-bit MCU assembler.
- c). One-bit name had been declared in SONiX 8-bit MCU assembler with "F" prefix code.
- d). "b0bset", "b0bclr", "bset", "bclr" instructions only support "R/W" registers.

**SN8P1704 System register table**

| 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    | -      | -        | -        | -        | -      | C      | DC     | Z       | R/W | PFLAG               |
| 0B0H    | DAENB  | DAB6     | DAB5     | DAB4     | DAB3   | DAB2   | DAB1   | DAB0    | R/W | DAM data register   |
| 0B1H    | ADENB  | ADS      | EOC      | GCHS     | -      | CHS2   | CHS1   | CHS0    | R/W | ADM mode register   |
| 0B2H    | ADB11  | ADB10    | ADB9     | ADB8     | ADB7   | ADB6   | ADB5   | ADB4    | R   | ADB data buffer     |
| 0B3H    | -      | ADCKS    | ADLEN    | 0        | ADB3   | ADB2   | ADB1   | ADB0    | R/W | ADR register        |
| 0B4H    | SENB   | START    | SRATE1   | SRATE0   | 0      | SCKMD  | SEDGE  | TXRX    | R/W | SIOM mode register  |
| 0B5H    | SIOR7  | SIOR6    | SIOR5    | SIOR4    | SIOR3  | SIOR2  | SIOR1  | SIOR0   | W   | SIOR reload buffer  |
| 0B6H    | SIOB7  | SIOB6    | SIOB5    | SIOB4    | SIOB3  | SIOB2  | SIOB1  | SIOB0   | R/W | SIOB data buffer    |
| 0C0H    | 0      | 0        | 0        | P14W     | P13W   | P12W   | P11W   | P10W    | W   | P1W wakeup register |
| 0C2H    | 0      | 0        | 0        | 0        | 0      | 0      | 0      | 0       | R/W | P2M I/O direction   |
| 0C1H    | 0      | 0        | 0        | P14M     | P13M   | P12M   | P11M   | P10M    | R/W | P1M I/O direction   |
| 0C4H    | 0      | 0        | 0        | P44M     | P43M   | P42M   | P41M   | P40M    | R/W | P4M I/O direction   |
| 0C5H    | 0      | 0        | 0        | P54M     | P53M   | P52M   | P51M   | P50M    | R/W | P5M I/O direction   |
| 0C8H    | 0      | TC1IRQ   | TC0IRQ   | 0        | SIOIRQ | P02IRQ | P01IRQ | P00IRQ  | R/W | INTRQ               |
| 0C9H    | 0      | TC1IEN   | TC0IEN   | 0        | SIOIEN | P02IEN | P01IEN | P00IEN  | R/W | INTEN               |
| 0CAH    | 0      | WDRST    | WDRate   | 0        | CPUM0  | CLKMD  | STPHX  | 0       | R/W | OSCM                |
| 0CDH    | TC0R7  | TC0R6    | TC0R5    | TC0R4    | TC0R3  | TC0R2  | TC0R1  | TC0R0   | W   | TC0R                |
| 0CEH    | PC7    | PC6      | PC5      | PC4      | PC3    | PC2    | PC1    | PC0     | R/W | PCL                 |
| 0CFH    | -      | -        | -        | -        | PC11   | PC10   | PC9    | PC8     | R/W | PCH                 |
| 0D0H    | -      | -        | -        | -        | -      | P02    | P01    | P00     | R   | P0 data buffer      |
| 0D1H    | -      | -        | -        | P14      | P13    | P12    | P11    | P10     | R/W | P1 data buffer      |
| 0D4H    | -      | -        | -        | P44      | P43    | P42    | P41    | P40     | R/W | P4 data buffer      |
| 0D5H    | -      | -        | -        | P54      | P53    | P52    | P51    | P50     | R/W | P5 data buffer      |
| 0DAH    | TC0ENB | TC0rate2 | TC0rate1 | TC0rate0 | 0      | ALOAD0 | TC0OUT | PWM0OUT | R/W | TC0M                |
| 0DBH    | TC0C7  | TC0C6    | TC0C5    | TC0C4    | TC0C3  | TC0C2  | TC0C1  | TC0C0   | R/W | TC0C                |
| 0DCH    | TC1ENB | TC1rate2 | TC1rate1 | TC1rate0 | 0      | 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    | -        | -        | -        | STKPB3 | STKPB2 | STKPB1 | STKPB0  | R/W | STKP stack pointer  |
| 0E7H    | @YZ7   | @YZ6     | @YZ5     | @YZ4     | @YZ3   | @YZ2   | @YZ1   | @YZ0    | R/W | @YZ index pointer   |
| 0F0H    | S7PC7  | S7PC6    | S7PC5    | S7PC4    | S7PC3  | S7PC2  | S7PC1  | S7PC0   | R/W | STK7L               |
| 0F1H    | -      | -        | -        | -        | S7PC11 | S7PC10 | S7PC9  | S7PC8   | R/W | STK7H               |
| 0F2H    | S6PC7  | S6PC6    | S6PC5    | S6PC4    | S6PC3  | S6PC2  | S6PC1  | S6PC0   | R/W | STK6L               |
| 0F3H    | -      | -        | -        | -        | S6PC11 | S6PC10 | S6PC9  | S6PC8   | R/W | STK6H               |
| "       | "      | "        | "        | "        | "      | "      | "      | "       | "   | "                   |
| "       | "      | "        | "        | "        | "      | "      | "      | "       | "   | "                   |
| "       | "      | "        | "        | "        | "      | "      | "      | "       | "   | "                   |
| 0FCH    | S1PC7  | S1PC6    | S1PC5    | S1PC4    | S1PC3  | S1PC2  | S1PC1  | S1PC0   | R/W | STK1L               |
| 0FDH    | -      | -        | -        | -        | S1PC11 | S1PC10 | S1PC9  | S1PC8   | R/W | STK1H               |
| 0FEH    | S0PC7  | S0PC6    | S0PC5    | S0PC4    | S0PC3  | S0PC2  | S0PC1  | S0PC0   | R/W | STK0L               |
| 0FFH    | -      | -        | -        | -        | S0PC11 | S0PC10 | S0PC9  | S0PC8   | R/W | STK0H               |

**Table 5-5. Bit System Register Table of SN8P1704**

**Note:**

- a). To avoid system error, please be sure to put all the "0" as it indicates in the above table
- b). All of register name had been declared in SONiX 8-bit MCU assembler.
- c). One-bit name had been declared in SONiX 8-bit MCU assembler with "F" prefix code.
- d). "b0bset", "b0bclr", "bset", "bclr" instructions only support "R/W" registers.
- e). For detail description please refer file of "System Register Quick Reference Table"

**SN8P1706 System register table**

| Address | Bit7   | Bit6     | Bit5     | Bit4     | Bit3   | Bit2   | Bit1   | Bit0    | R/W | Remarks             |
|---------|--------|----------|----------|----------|--------|--------|--------|---------|-----|---------------------|
| 080H    | LBIT7  | LBIT6    | LBIT5    | LBIT4    | LBIT3  | LBIT2  | LBIT1  | LBIT0   | R/W | L                   |
| 081H    | HBIT7  | HBIT6    | HBIT5    | HBIT4    | HBIT3  | HBIT2  | HBIT1  | HBIT0   | R/W | H                   |
| 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    | XBIT7  | XBIT6    | XBIT5    | XBIT4    | XBIT3  | XBIT2  | XBIT1  | XBIT0   | R/W | X                   |
| 086H    | -      | -        | -        | -        | -      | C      | DC     | Z       | R/W | PFLAG               |
| 087H    | -      | -        | -        | -        | -      | -      | -      | RBNKS0  | R/W | RBANK               |
| 0B0H    | DAENB  | DAB6     | DAB5     | DAB4     | DAB3   | DAB2   | DAB1   | DAB0    | R/W | DAM data register   |
| 0B1H    | ADENB  | ADS      | EOC      | GCHS     | -      | CHS2   | CHS1   | CHS0    | R/W | ADM mode register   |
| 0B2H    | ADB11  | ADB10    | ADB9     | ADB8     | ADB7   | ADB6   | ADB5   | ADB4    | R   | ADB data buffer     |
| 0B3H    | -      | ADCKS    | ADLEN    | 0        | ADB3   | ADB2   | ADB1   | ADB0    | R/W | ADR register        |
| 0B4H    | SENB   | START    | SRATE1   | SRATE0   | 0      | SCKMD  | SEDGE  | TXRX    | R/W | SIOM mode register  |
| 0B5H    | SIOR7  | SIOR6    | SIOR5    | SIOR4    | SIOR3  | SIOR2  | SIOR1  | SIOR0   | W   | SIOR reload buffer  |
| 0B6H    | SIOB7  | SIOB6    | SIOB5    | SIOB4    | SIOB3  | SIOB2  | SIOB1  | SIOB0   | R/W | SIOB data buffer    |
| 0C0H    | 0      | 0        | P15W     | P14W     | P13W   | P12W   | P11W   | P10W    | W   | P1W wakeup register |
| 0C1H    | 0      | 0        | P15M     | P14M     | P13M   | P12M   | P11M   | P10M    | R/W | P1M I/O direction   |
| 0C2H    | 0      | 0        | 0        | P24M     | P23M   | P22M   | P21M   | P20M    | R/W | P2M I/O direction   |
| 0C4H    | P47M   | P46M     | P45M     | P44M     | P43M   | P42M   | P41M   | P40M    | R/W | P4M I/O direction   |
| 0C5H    | P57M   | P56M     | P55M     | P54M     | P53M   | P52M   | P51M   | P50M    | R/W | P5M I/O direction   |
| 0C8H    | 0      | TC1IRQ   | TC0IRQ   | T0IRQ    | SIOIRQ | P02IRQ | P01IRQ | P00IRQ  | R/W | INTRQ               |
| 0C9H    | 0      | TC1IEN   | TC0IEN   | T0IEN    | SIOIEN | P02IEN | P01IEN | P00IEN  | R/W | INTEN               |
| 0CAH    | 0      | WDRST    | Wdrate   | 0        | CPUM0  | CLKMD  | STPHX  | 0       | R/W | OSCM                |
| 0CDH    | TC0R7  | TC0R6    | TC0R5    | TC0R4    | TC0R3  | TC0R2  | TC0R1  | TC0R0   | W   | TC0R                |
| 0CEH    | PC7    | PC6      | PC5      | PC4      | PC3    | PC2    | PC1    | PC0     | R/W | PCL                 |
| 0CFH    | -      | -        | -        | -        | PC11   | PC10   | PC9    | PC8     | R/W | PCH                 |
| 0D0H    | -      | -        | -        | -        | -      | P02    | P01    | P00     | R   | P0 data buffer      |
| 0D1H    | -      | -        | P15      | P14      | P13    | P12    | P11    | P10     | R/W | P1 data buffer      |
| 0D2H    | -      | -        | -        | P24      | P23    | P22    | P21    | P20     | R/W | P2 data buffer      |
| 0D4H    | P47    | P46      | P45      | P44      | P43    | P42    | P41    | P40     | R/W | P4 data buffer      |
| 0D5H    | P57    | P56      | P55      | P54      | P53    | P52    | P51    | P50     | R/W | P5 data buffer      |
| 0D8H    | T0ENB  | T0rate2  | T0rate1  | T0rate0  | 0      | 0      | 0      | 0       | R/W | T0M                 |
| 0D9H    | T0C7   | T0C6     | T0C5     | T0C4     | T0C3   | T0C2   | T0C1   | T0C0    | R/W | T0C                 |
| 0DAH    | TC0ENB | TC0rate2 | TC0rate1 | TC0rate0 | 0      | ALOAD0 | TC0OUT | PWM0OUT | R/W | TC0M                |
| 0DBH    | TC0C7  | TC0C6    | TC0C5    | TC0C4    | TC0C3  | TC0C2  | TC0C1  | TC0C0   | R/W | TC0C                |
| 0DCH    | TC1ENB | TC1rate2 | TC1rate1 | TC1rate0 | 0      | 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    | -        | -        | -        | STKPB3 | STKPB2 | STKPB1 | STKPB0  | R/W | STKP stack pointer  |
| 0E6H    | @HL7   | @HL6     | @HL5     | @HL4     | @HL3   | @HL2   | @HL1   | @HL0    | R/W | @HL index pointer   |
| 0E7H    | @YZ7   | @YZ6     | @YZ5     | @YZ4     | @YZ3   | @YZ2   | @YZ1   | @YZ0    | R/W | @YZ index pointer   |
| 0F0H    | S7PC7  | S7PC6    | S7PC5    | S7PC4    | S7PC3  | S7PC2  | S7PC1  | S7PC0   | R/W | STK7L               |
| 0F1H    | -      | -        | -        | -        | S7PC11 | S7PC10 | S7PC9  | S7PC8   | R/W | STK7H               |
| 0F2H    | S6PC7  | S6PC6    | S6PC5    | S6PC4    | S6PC3  | S6PC2  | S6PC1  | S6PC0   | R/W | STK6L               |
| 0F3H    | -      | -        | -        | -        | S6PC11 | S6PC10 | S6PC9  | S6PC8   | R/W | STK6H               |
| "       | "      | "        | "        | "        | "      | "      | "      | "       | "   | "                   |
| "       | "      | "        | "        | "        | "      | "      | "      | "       | "   | "                   |
| 0FCH    | S1PC7  | S1PC6    | S1PC5    | S1PC4    | S1PC3  | S1PC2  | S1PC1  | S1PC0   | R/W | STK1L               |
| 0FDH    | -      | -        | -        | -        | S1PC11 | S1PC10 | S1PC9  | S1PC8   | R/W | STK1H               |
| 0FEH    | S0PC7  | S0PC6    | S0PC5    | S0PC4    | S0PC3  | S0PC2  | S0PC1  | S0PC0   | R/W | STK0L               |
| 0FFH    | -      | -        | -        | -        | S0PC11 | S0PC10 | S0PC9  | S0PC8   | R/W | STK0H               |

**Table 5-6. Bit System Register Table of SN8P1706**
**Note:**

- To avoid system error, please be sure to put all the "0" as it indicates in the above table
- All of register name had been declared in SONiX 8-bit MCU assembler.
- One-bit name had been declared in SONiX 8-bit MCU assembler with "F" prefix code.
- "b0bset", "b0bclr", "bset", "bclr" instructions only support "R/W" registers.
- For detail description please refer file of "System Register Quick Reference Table"

**SN8P1707/ SN8P1708 System register table**

| Address | Bit7   | Bit6     | Bit5     | Bit4     | Bit3   | Bit2   | Bit1   | Bit0    | R/W | Remarks             |
|---------|--------|----------|----------|----------|--------|--------|--------|---------|-----|---------------------|
| 080H    | LBIT7  | LBIT6    | LBIT5    | LBIT4    | LBIT3  | LBIT2  | LBIT1  | LBIT0   | R/W | L                   |
| 081H    | HBIT7  | HBIT6    | HBIT5    | HBIT4    | HBIT3  | HBIT2  | HBIT1  | HBIT0   | R/W | H                   |
| 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    | XBIT7  | XBIT6    | XBIT5    | XBIT4    | XBIT3  | XBIT2  | XBIT1  | XBIT0   | R/W | X                   |
| 086H    | -      | -        | -        | -        | -      | C      | DC     | Z       | R/W | PFLAG               |
| 087H    | -      | -        | -        | -        | -      | -      | -      | RBNKS0  | R/W | RBANK               |
| 0B0H    | DAENB  | DAB6     | DAB5     | DAB4     | DAB3   | DAB2   | DAB1   | DAB0    | R/W | DAM data register   |
| 0B1H    | ADENB  | ADS      | EOC      | GCHS     | -      | CHS2   | CHS1   | CHS0    | R/W | ADM mode register   |
| 0B2H    | ADB11  | ADB10    | ADB9     | ADB8     | ADB7   | ADB6   | ADB5   | ADB4    | R   | ADB data buffer     |
| 0B3H    | -      | ADCKS    | ADLEN    | 0        | ADB3   | ADB2   | ADB1   | ADB0    | R/W | ADR register        |
| 0B4H    | SENB   | START    | SRATE1   | SRATE0   | 0      | SCKMD  | SEDGE  | TXRX    | R/W | SIOM mode register  |
| 0B5H    | SIOR7  | SIOR6    | SIOR5    | SIOR4    | SIOR3  | SIOR2  | SIOR1  | SIOR0   | W   | SIOR reload buffer  |
| 0B6H    | SIOB7  | SIOB6    | SIOB5    | SIOB4    | SIOB3  | SIOB2  | SIOB1  | SIOB0   | R/W | SIOB data buffer    |
| 0C0H    | 0      | 0        | P15W     | P14W     | P13W   | P12W   | P11W   | P10W    | W   | P1W wakeup register |
| 0C1H    | 0      | 0        | P15M     | P14M     | P13M   | P12M   | P11M   | P10M    | R/W | P1M I/O direction   |
| 0C2H    | P27M   | P26M     | P25M     | P24M     | P23M   | P22M   | P21M   | P20M    | R/W | P2M I/O direction   |
| 0C4H    | P47M   | P46M     | P45M     | P44M     | P43M   | P42M   | P41M   | P40M    | R/W | P4M I/O direction   |
| 0C5H    | P57M   | P56M     | P55M     | P54M     | P53M   | P52M   | P51M   | P50M    | R/W | P5M I/O direction   |
| 0C8H    | 0      | TC1IRQ   | TC0IRQ   | T0IRQ    | SIOIRQ | P02IRQ | P01IRQ | P00IRQ  | R/W | INTRQ               |
| 0C9H    | 0      | TC1IEN   | TC0IEN   | T0IEN    | SIOIEN | P02IEN | P01IEN | P00IEN  | R/W | INTEN               |
| 0CAH    | 0      | WDRST    | Wdrate   | 0        | CPUM0  | CLKMD  | STPHX  | 0       | R/W | OSCM                |
| 0CDH    | TC0R7  | TC0R6    | TC0R5    | TC0R4    | TC0R3  | TC0R2  | TC0R1  | TC0R0   | W   | TC0R                |
| 0CEH    | PC7    | PC6      | PC5      | PC4      | PC3    | PC2    | PC1    | PC0     | R/W | PCL                 |
| 0CFH    | -      | -        | -        | -        | PC11   | PC10   | PC9    | PC8     | R/W | PCH                 |
| 0D0H    | -      | -        | -        | -        | -      | P02    | P01    | P00     | R   | P0 data buffer      |
| 0D1H    | -      | -        | 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      |
| 0D4H    | P47    | P46      | P45      | P44      | P43    | P42    | P41    | P40     | R/W | P4 data buffer      |
| 0D5H    | P57    | P56      | P55      | P54      | P53    | P52    | P51    | P50     | R/W | P5 data buffer      |
| 0D8H    | T0ENB  | T0rate2  | T0rate1  | T0rate0  | 0      | 0      | 0      | 0       | R/W | T0M                 |
| 0D9H    | T0C7   | T0C6     | T0C5     | T0C4     | T0C3   | T0C2   | T0C1   | T0C0    | R/W | T0C                 |
| 0DAH    | TC0ENB | TC0rate2 | TC0rate1 | TC0rate0 | 0      | ALOAD0 | TC0OUT | PWM0OUT | R/W | TC0M                |
| 0DBH    | TC0C7  | TC0C6    | TC0C5    | TC0C4    | TC0C3  | TC0C2  | TC0C1  | TC0C0   | R/W | TC0C                |
| 0DCH    | TC1ENB | TC1rate2 | TC1rate1 | TC1rate0 | 0      | 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    | -        | -        | -        | STKPB3 | STKPB2 | STKPB1 | STKPB0  | R/W | STKP stack pointer  |
| 0E6H    | @HL7   | @HL6     | @HL5     | @HL4     | @HL3   | @HL2   | @HL1   | @HL0    | R/W | @HL index pointer   |
| 0E7H    | @YZ7   | @YZ6     | @YZ5     | @YZ4     | @YZ3   | @YZ2   | @YZ1   | @YZ0    | R/W | @YZ index pointer   |
| 0F0H    | S7PC7  | S7PC6    | S7PC5    | S7PC4    | S7PC3  | S7PC2  | S7PC1  | S7PC0   | R/W | STK7L               |
| 0F1H    | -      | -        | -        | -        | S7PC11 | S7PC10 | S7PC9  | S7PC8   | R/W | STK7H               |
| 0F2H    | S6PC7  | S6PC6    | S6PC5    | S6PC4    | S6PC3  | S6PC2  | S6PC1  | S6PC0   | R/W | STK6L               |
| 0F3H    | -      | -        | -        | -        | S6PC11 | S6PC10 | S6PC9  | S6PC8   | R/W | STK6H               |
| "       | "      | "        | "        | "        | "      | "      | "      | "       | "   | "                   |
| "       | "      | "        | "        | "        | "      | "      | "      | "       | "   | "                   |
| 0FCH    | S1PC7  | S1PC6    | S1PC5    | S1PC4    | S1PC3  | S1PC2  | S1PC1  | S1PC0   | R/W | STK1L               |
| 0FDH    | -      | -        | -        | -        | S1PC11 | S1PC10 | S1PC9  | S1PC8   | R/W | STK1H               |
| 0FEH    | S0PC7  | S0PC6    | S0PC5    | S0PC4    | S0PC3  | S0PC2  | S0PC1  | S0PC0   | R/W | STK0L               |
| 0FFH    | -      | -        | -        | -        | S0PC11 | S0PC10 | S0PC9  | S0PC8   | R/W | STK0H               |

**Table 5-7. Bit System Register Table of SN8P1707/ SN8P1708**
**Note:**

- To avoid system error, please be sure to put all the "0" as it indicates in the above table
- All of register name had been declared in SONiX 8-bit MCU assembler.
- One-bit name had been declared in SONiX 8-bit MCU assembler with "F" prefix code.
- "b0bset", "b0bclr", "bset", "bclr" instructions only support "R/W" registers.
- For detail description please refer file of "System Register Quick Reference Table"

# 6 POWER ON RESET

## OVERVIEW

SN8P1700 provides two system resets. One is external reset and the other is low voltage detector (LVD). The external reset is a simple RC circuit connecting to the reset pin. The low voltage detector (LVD) is built in internal circuit. When one of the reset devices occurs, the system will reset and the system registers become initial value. The timing diagram is as following.

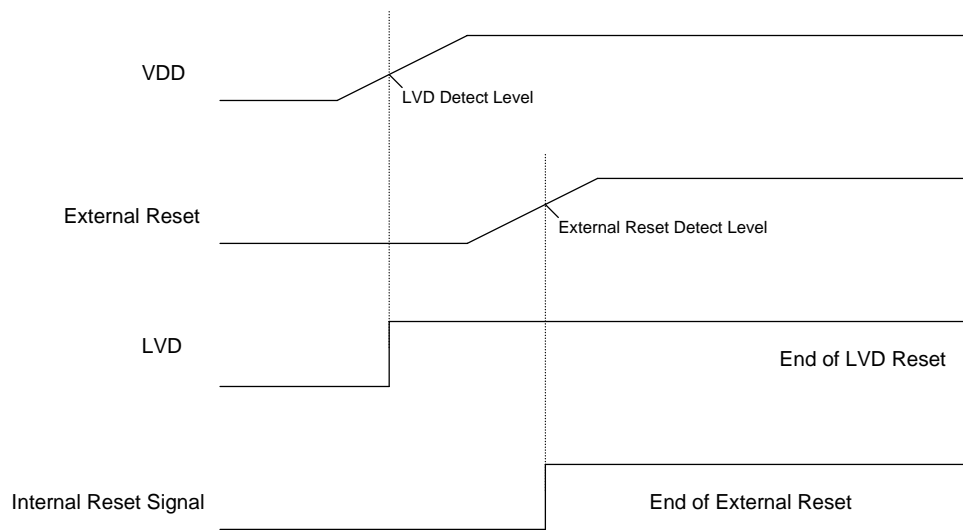


Figure 6-1 Power on Reset Timing Diagram

**Notice :** The working current of the LVD is about 100uA.

## EXTERNAL RESET DESCRIPTION

The external reset is a low level active device. The reset pin receives the low voltage and resets the system. When the voltage detects high level, it stops resetting the system. Users can use an external reset circuit to control system operation. It is necessary that the VDD must be stable.

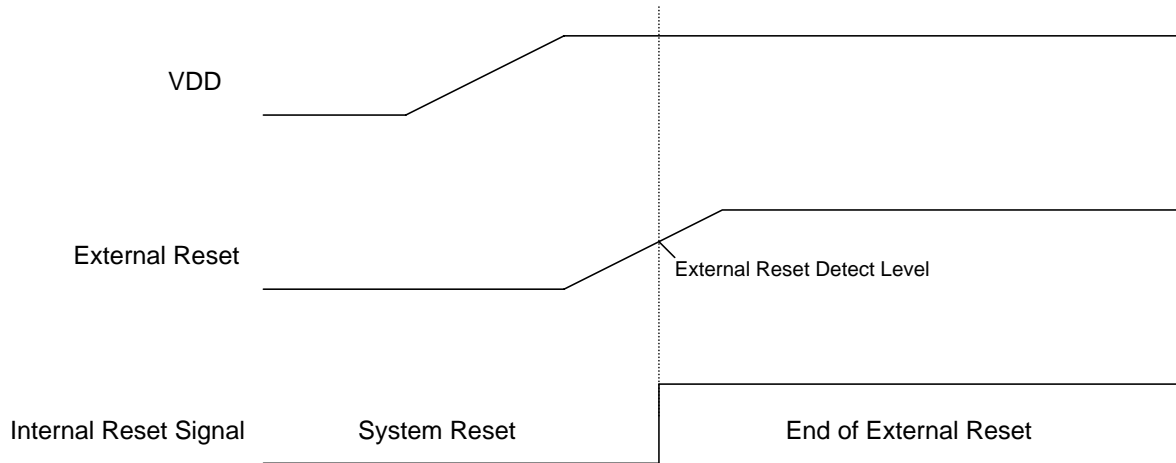


Figure 6-2 External Reset Timing Diagram

Users must be sure the VDD stable earlier than external reset (Figure 5-2) or the external reset will fail. The external reset circuit is a simple RC circuit as following.

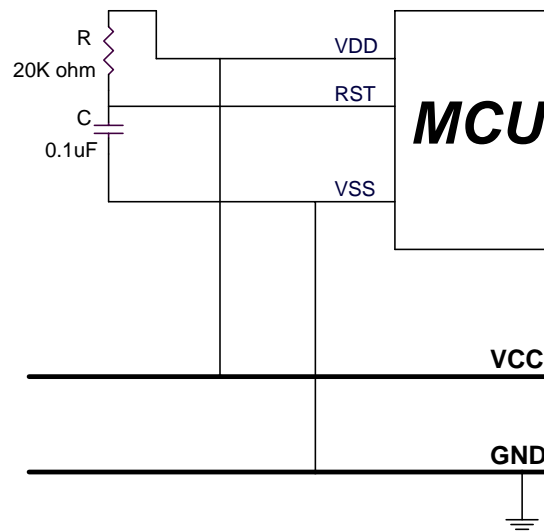


Figure 6-3. External Reset Circuit



In worse-power condition as brown out reset. The reset pin may keep high level but the VDD is low voltage. That makes the system reset fail and chip error. To connect a diode from reset pin to VDD is a good solution. The circuit can force the capacitor to release electric charge and drop the voltage, and solve the error.

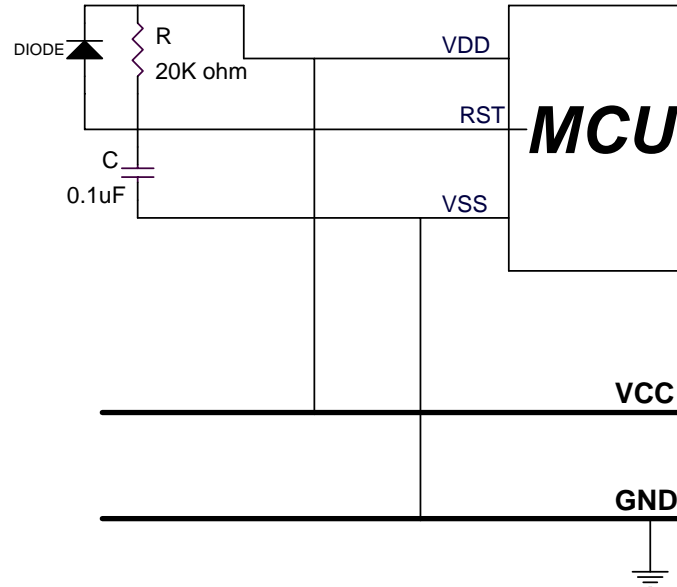


Figure 6-4. External Reset Circuit with Diode

## LOW VOLTAGE DETECTOR (LVD) DESCRIPTION

The LVD is a low voltage detector. It detects VDD level and reset the system as the VDD lower than the desired voltage. The detect level is 2.4V. If the VDD lower than 2.4V, the system resets. The LVD function is controlled by code option. Users can turn on it for special application like worse power condition. LVD work with external reset function. They are OR active.

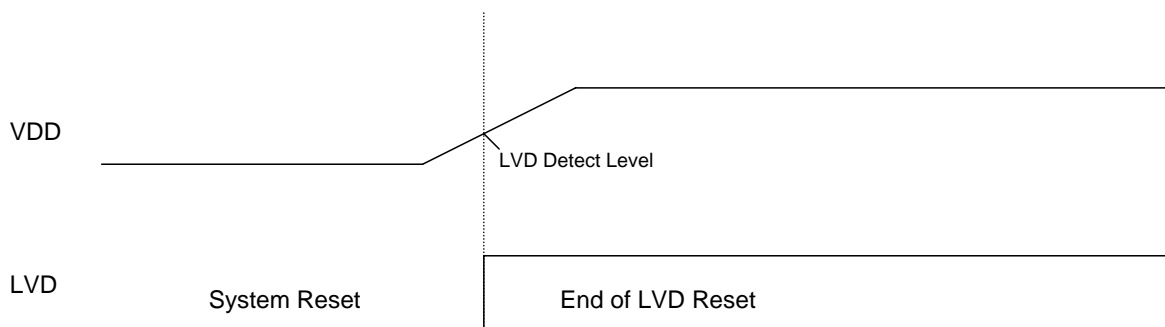


Figure 6-5. LVD Timing Diagram

The LVD can protect system to work well under brownout reset. But it is a high consumptive circuit. In 3V condition, the LVD consumes about 100uA. It is a very large consumption for battery system. So the LVD supports AC system well.

➤ **Notice:** LVD is selected by code option.

# 7 OSCILLATORS

## OVERVIEW

The SN8P1700 highly performs the dual clock micro-controller system. The dual clocks are high-speed clock and low-speed clock. The high-speed clock frequency is supplied through the external oscillator circuit. The low-speed clock frequency is supplied through on-chip RC oscillator circuit.

The external high-speed clock and the internal low-speed clock can be system clock ( $F_{osc}$ ). And the system clock is divided by 4 to be the instruction cycle ( $F_{cpu}$ ).

$$F_{cpu} = F_{osc} / 4$$

The system clock is required by the following peripheral modules:

- ✓ **Basic timer (T0)**
- ✓ **Timer counter 0 (TC0)**
- ✓ **Timer counter 1 (TC1)**
- ✓ **Watchdog timer**
- ✓ **Serial I/O interface (SIO)**
- ✓ **AD converter**
- ✓ **PWM output (PWM0, PWM1)**
- ✓ **Buzzer output (TC0OUT, TC1OUT)**

## CLOCK BLOCK DIAGRAM

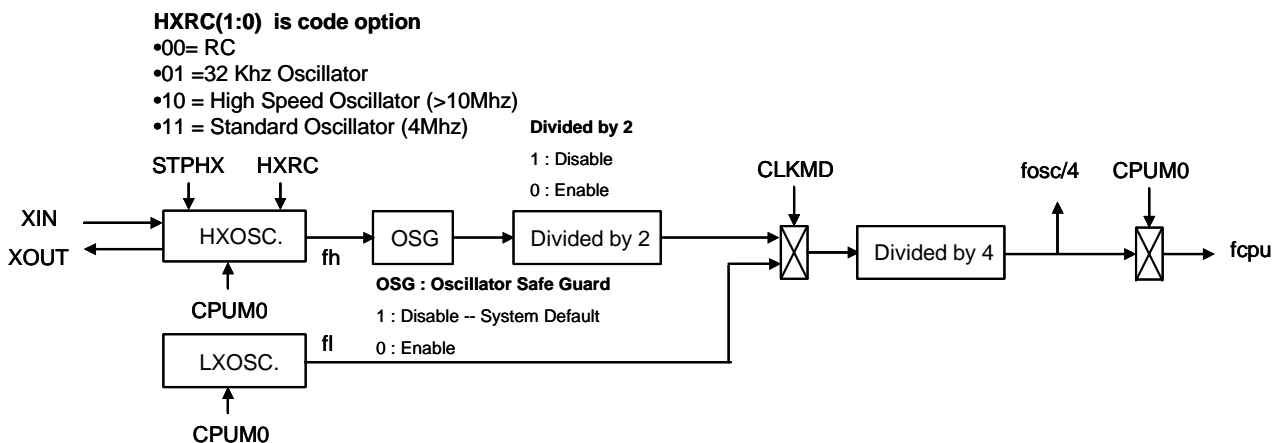


Figure 7-1. Clock Block Diagram

- HXOSC: External high-speed clock.
- LXOSC: Internal low-speed clock.
- OSG: Oscillator safe guard.

## OSCM REGISTER DESCRIPTION

The OSCM register is a oscillator control register. It can control oscillator select, system mode, watchdog timer clock source and rate.

**OSCM initial value = 000x 000x**

| 0CAH        | Bit 7 | Bit 6 | Bit 5   | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|-------------|-------|-------|---------|-------|-------|-------|-------|-------|
| <b>OSCM</b> | 0     | WDRST | Wdtrate | 0     | CPUM0 | CLKMD | STPHX | 0     |
|             | -     | R/W   | R/W     | -     | R/W   | R/W   | R/W   | -     |

STPHX: Eternal high-speed oscillator control bit. 0 = free run, 1 = stop. This bit just only controls external high-speed oscillator. If STPHX=1, the internal low-speed RC oscillator is still running.

CLKMD: System high/Low speed mode select bit. 0 = normal (dual) mode, 1 = slow mode.

CPUM0: CPU operating mode control bit. 0 = normal, 1 = sleep (power down) mode to turn off both high/low clock.

➤ **Notice: The bit 7 of OSCM register must be "0", or the system will be error.**

## EXTERNAL HIGH-SPEED OSCILLATOR

SN8P1700 can be operated in four different oscillator modes. There are external RC oscillator modes, high crystal/resonator mode (12M code option), standard crystal/resonator mode (4M code option) and low crystal mode (32K code option). For different application, the users can select one of suitable oscillator mode by programming code option to generate system high-speed clock source after reset.

⇒ **Example: Stop external high-speed oscillator.**

B0BSET    FSTPHX            ; To stop external high-speed oscillator only.

B0BSET    FCPUM0            ; To stop external high-speed oscillator and internal low-speed  
; oscillator called power down mode (sleep mode).

## OSCILLATOR MODE CODE OPTION

SN8P1700 has four oscillator modes for different applications. These modes are 4M, 12M, 32K and RC. The main purpose is to support different oscillator types and frequencies. High-speed crystal needs more current but the low one doesn't. For crystals, there are three steps to select. If the oscillator is RC type, to select "RC" and the system will divide the frequency by 2 automatically. User can select oscillator mode from Code Option table before compiling. The table is as follow.

| Code Option | Oscillator Mode | Remark                                     |
|-------------|-----------------|--|
| 00          | RC mode         | Output the Fcpu square wave from Xout pin. |
| 01          | 32K             | 32768Hz                                    |
| 10          | 12M             | 12MHz ~ 16MHz                              |
| 11          | 4M              | 3.58MHz                                    |

## OSCILLATOR DEVIDE BY 2 CODE OPTION

SN8P1700 has an external clock divide by 2 function. It is a code option called "High\_Clk / 2". If "High\_Clk / 2" is enabled, the external clock frequency is divided by 8 for the Fcpu. Fcpu is equal to Fosc/8. If "High\_Clk / 2" is disabled, the external clock frequency is divided by 4 for the Fcpu. The Fcpu is equal to Fosc/4.

➤ **Note: In RC mode, "High\_Clk / 2" is always enabled.**

## OSCILLATOR SAFE GUARD CODE OPTION

SN8P1700 builds in an oscillator safe guard (OSG) to make oscillator more stable. It is a low-pass filter circuit and stops high frequency noise into system from external oscillator circuit. This function makes system to work better under AC noisy conditions.

## SYSTEM OSCILLATOR CIRCUITS

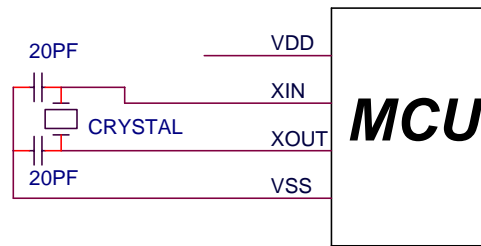


Figure 7-2. Crystal/Ceramic Oscillator

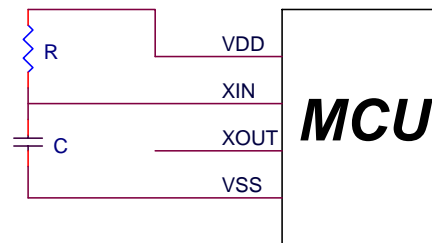


Figure 7-3. RC Oscillator

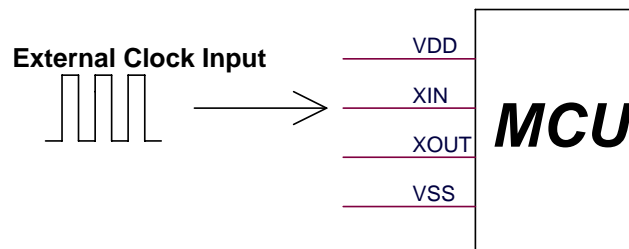


Figure 7-4. External clock input

- **Note1:** The VDD and VSS of external oscillator circuit must be from the micro-controller. Don't connect them from the neighbor power terminal.
- **Note2:** The external clock input mode can select RC type oscillator or crystal type oscillator of the code option and input the external clock into XIN pin.
- **Note3:** In RC type oscillator code option situation, the external clock's frequency is divided by 2.
- **Note4:** The power and ground of external oscillator circuit must be connected from the micro-controller's VDD and VSS. It is necessary to step up the performance of the whole system.

## External RC Oscillator Frequency Measurement

There are two ways to get the Fosc frequency of external RC oscillator. One measures the XOUT output waveform. Under external RC oscillator mode, the XOUT outputs the square waveform whose frequency is Fcpu. The other measures the external RC frequency by instruction cycle (Fcpu). The external RC frequency is the Fcpu multiplied by 4. We can get the Fosc frequency of external RC from the Fcpu frequency. The sub-routine to get Fcpu frequency of external oscillator is as the following.

➔ Example: Fcpu instruction cycle of external oscillator

```
BOBSET    P1M.0      ; Set P1.0 to be output mode for outputting Fcpu toggle
                    signal.
```

@ @:

```
BOBSET    P1.0      ; Output Fcpu toggle signal in low-speed clock mode.
BOBCLR    P1.0      ; Measure the Fcpu frequency by oscilloscope.
JMP       @B
```

## INTERNAL LOW-SPEED OSCILLATOR

The internal low-speed oscillator is built in the micro-controller. The low-speed clock's source is a RC type oscillator circuit. The low-speed clock can supplies clock for system clock, timer counter, watchdog timer, SIO clock source and so on.

➔ **Example: Stop internal low-speed oscillator.**

```
B0BSET    FCPUM0           ; To stop external high-speed oscillator and internal low-speed
                                ; oscillator called power down mode (sleep mode).
```

➤ **Note: The internal low-speed clock can't be turned off individually. It is controlled by CPUM0 bit of OSCM register.**

The low-speed oscillator uses RC type oscillator circuit. The frequency is affected by the voltage and temperature of the system. In common condition, the frequency of the RC oscillator is about 16KHz at 3V and 32KHz at 5V. The relative between the RC frequency and voltage is as following.

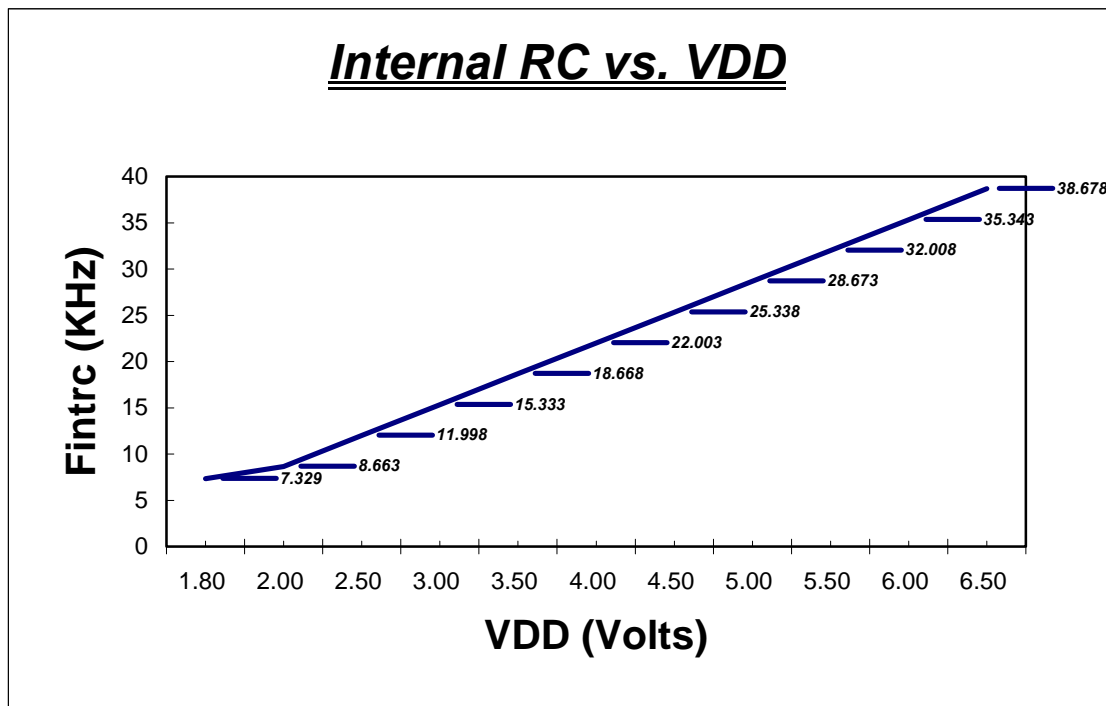


Figure 7-5. Internal RC vs. VDD Diagram

➔ **Example: To measure the internal RC frequency is by instruction cycle (Fcpu). The internal RC frequency is the Fcpu multiplied by 4. So we can get the Fosc frequency of internal RC from the Fcpu frequency.**

```
B0BSET    P1M.0           ; Set P1.0 to be output mode for outputting Fcpu toggle signal.
```

```
B0BSET    FCLKMD          ; Switch the system clock to internal low-speed clock mode.
```

@@:

```
B0BSET    P1.0           ; Output Fcpu toggle signal in low-speed clock mode.
```

```
B0BCLR    P1.0           ; Measure the Fcpu frequency by oscilloscope.
```

```
JMP      @B
```

## SYSTEM MODE DESCRIPTION

### OVERVIEW

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

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

In actual application, the user can adjust the chip's controller to work in these three modes by using OSCM register. At the high-speed mode, the instruction cycle ( $F_{cpu}$ ) is  $F_{osc}/4$ . At the low-speed mode and 3V, the  $F_{cpu}$  is 16KHz/4.

### NORMAL MODE

In normal mode, the system clock source is external high-speed clock. After power on, the system works under normal mode. The instruction cycle is  $f_{osc}/4$ . When the external high-speed oscillator is 3.58MHz, the instruction cycle is  $3.58\text{MHz}/4 = 895\text{KHz}$ . All software and hardware are executed and working. In normal mode, system can get into power down mode and slow mode.

### SLOW MODE

In slow mode, the system clock source is internal low-speed RC clock. To set  $CLKMD = 1$ , the system switch to slow mode. In slow mode, the system works as normal mode but the slower clock. The system in slow mode can get into normal mode and power down mode. To set  $STPHX = 1$  to stop the external high-speed oscillator, and then the system consumes less power.

### POWER DOWN MODE

The power down mode is also called sleep mode. The chip stops working as sleeping status. The power consumption is very less almost to zero. The power down mode is usually applied to low power consuming system as battery power productions. To set  $CUPM0 = 1$ , the system gets into power down mode. The external high-speed and low-speed oscillators are turned off. The system can be waked up by P0, P1 trigger signal.



# SYSTEM MODE CONTROL

## SN8P1700 SYSTEM MODE BLOCK DIAGRAM

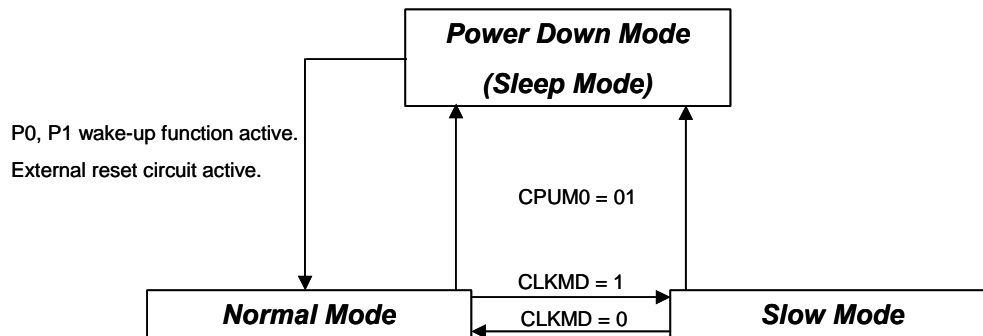


Figure 7-6. SN8P1700 System Mode Block Diagram

### Operating mode description

| MODE               | NORMAL     | SLOW       | POWER DOWN (SLEEP) | REMARK                |
|--------------------|------------|------------|--------------------|-----------------------|
| HX osc.            | Running    | By STPHX   | Stop               |                       |
| LX osc.            | Running    | Running    | Stop               |                       |
| CPU instruction    | Executing  | Executing  | Stop               |                       |
| T0 timer           | *Active    | *Active    | Inactive           | * Active by programm. |
| TC0 timer          | *Active    | *Active    | Inactive           |                       |
| TC1 timer          | *Active    | *Active    | Inactive           |                       |
| Watchdog timer     | Active     | Active     | Inactive           |                       |
| Internal interrupt | All active | All active | All inactive       |                       |
| External interrupt | All active | All active | All inactive       |                       |
| Wakeup source      | -          | -          | P0, P1, Reset      |                       |

Table 7-1. Operating Mode Description

## SYSTEM MODE SWITCHING

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

**CPUM0 = 1**

```

BOBSET      FCPUM0      ; set the system into power down mode.

```

During the sleep, only the wakeup pin and reset can wakeup the system back to the normal mode.

**Switch normal mode to slow mode.**

```

BOBSET      FCLKMD      ;To set CLKMD = 1, Change the system into slow mode
BOBSET      FSTPHX      ;To stop external high-speed oscillator for power saving.

```

**Switch slow mode to normal mode**

If external high clock stop and program want to switch back normal mode. It is necessary to delay at least 10mS for external clock stable.

```

BOBCLR      FSTPHX      ; Turn on the external high-speed oscillator.

@@:         B0MOV        Z, #27      ; If VDD = 5V, internal RC=32KHz (typical) will delay
            DECMS        Z          ; 0.125ms X 81 = 10.125ms for external clock stable
            JMP          @B

BOBCLR      FCLKMD      ; Change the system back to the normal mode

```

## WAKEUP TIME

### OVERVIEW

The external high-speed oscillator needs a delay time from stopping to operating. The delay is very necessary and makes the oscillator to work stably. Some conditions during system operating, the external high-speed oscillator often runs and stops. Under these condition, the delay time for external high-speed oscillator restart is called wakeup time.

There are two conditions need wakeup time. One is power down mode to normal mode. The other one is slow mode to normal mode. For the first case, SN8P1700 provides 2048 oscillator clocks to be the wakeup time. But in the last case, users need to make the wakeup time by themselves.

### HARDWARE WAKEUP

When the system is in power down mode (sleep mode), the external high-speed oscillator stops. For wakeup into normal, SN8P1700 provides 2048 external high-speed oscillator clocks to be the wakeup time for warming up the oscillator circuit. After the wakeup time, the system goes into the normal mode. The value of the wakeup time is as following.

$$\text{The wakeup time} = 1/F_{osc} * 2048 \text{ (sec)}$$

➤ **Example:** In power down mode (sleep mode), the system is waked up by P0 or P1 trigger signal. After the wakeup time, the system goes into normal mode. The wakeup time of P0, P1 wakeup function is as following.

$$\text{The wakeup time} = 1/F_{osc} * 2048 = 0.57 \text{ ms} \quad (F_{osc} = 3.58\text{MHz})$$

$$\text{The wakeup time} = 1/F_{osc} * 2048 = 62.5 \text{ ms} \quad (F_{osc}=32768\text{Hz})$$

Under power down mode (sleep mode), there are only I/O ports with wakeup function making the system to return normal mode. The Port 0 and Port 1 have wakeup function. Port 0's wakeup function always enables. The Port 1 controls by the P1W register.

**P1W initial value = xx00 0000**

| 0C0H       | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|------------|-------|-------|-------|-------|-------|-------|-------|-------|
| <b>P1W</b> | 0     | 0     | P15W  | P14W  | P13W  | P12W  | P11W  | P10W  |
|            | -     | -     | W     | W     | W     | W     | W     | W     |

P10W~P15W: Port 1 wakeup function control bits. 0 = none wakeup function, 1 = Enable each pin of Port 1 wakeup function.

➤ **Note:** For SN8P1702 the P1W register only obtains P10W and P11W. For SN8P1704 the P1W register only obtain P10W~P14W.

# 8 TIMERS COUNTERS

## WATCHDOG TIMER (WDT)

The watchdog timer (WDT) is a binary up counter designed for monitoring program execution. If the program get into the unknown status by noise interference, WDT's overflow signal will reset this chip and restart operation. The instruction that clear the watch-dog timer (BOBSET FWDRST) should be executed at proper points in a program within a given period. If an instruction that clears the watchdog timer is not executed within the period and the watchdog timer overflows, reset signal is generated and system is restarted with reset status. In order to generate different output timings, the user can control watchdog timer by modifying Wdrate control bits of OSCM register. The watchdog timer will be disabled at green and power down modes.

**OSCM initial value = 0000 000x**

| OCAH        | Bit 7 | Bit 6 | Bit 5  | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|-------------|-------|-------|--------|-------|-------|-------|-------|-------|
| <b>OSCM</b> | 0     | WDRST | Wdrate | -     | CPUM0 | CLKMD | STPHX | -     |
|             | -     | R/W   | R/W    | -     | R/W   | R/W   | R/W   | -     |

➤ **Notice: The bit 7 must be "0", or the system will be error.**

Wdrate: Watchdog timer rate select bit. 0 = 14<sup>th</sup>, 1 = 8<sup>th</sup>.

WDRST : Watch dog timer reset bit. 0 = Non reset, 1 = clear the watchdog timer's counter.

| Wdrate | Watchdog timer overflow time   |  |
|--------|--|--|
|        | External high-speed oscillator   |  |
| 0      | $1 / ( f_{cpu} \div 2^{14} \div 16 ) = 293 \text{ ms}, F_{osc}=3.58\text{MHz}$ |  |
|        | $1 / ( f_{cpu} \div 2^{14} \div 16 ) = 32 \text{ s}, F_{osc}=32768\text{Hz}$   |  |
| 1      | $1 / ( f_{cpu} \div 2^8 \div 16 ) = 4.5 \text{ ms}, F_{osc}=3.58\text{MHz}$    |  |
|        | $1 / ( f_{cpu} \div 2^8 \div 16 ) = 500 \text{ ms}, F_{osc}=32768\text{Hz}$    |  |

Figure 8-1. Watchdog timer overflow time table

➤ **Note: The watch dog timer can be enabled or disabled by the code option.**

⇒ **Example: An operation of watch-dog timer is as following. To clear the watchdog timer's counter in the top of the main routine of the program.**

```
Main:
      BOBSET      FWDRST      ; Clear the watchdog timer's counter.
      .
      CALL        SUB1
      CALL        SUB2
      .
      .
      .
      JMP        MAIN
```

## BASIC TIMER 0 (T0)

### OVERVIEW

The basic timer (T0) is an 8-bit binary up counter. It uses TOM register to select T0C's input clock for counting a precision time. If the T0 timer has occur an overflow (from FFH to 00H), it will continue counting and issue a time-out signal to trigger T0 interrupt to request interrupt service. The main purposes of the T0 basic timer is as following.

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

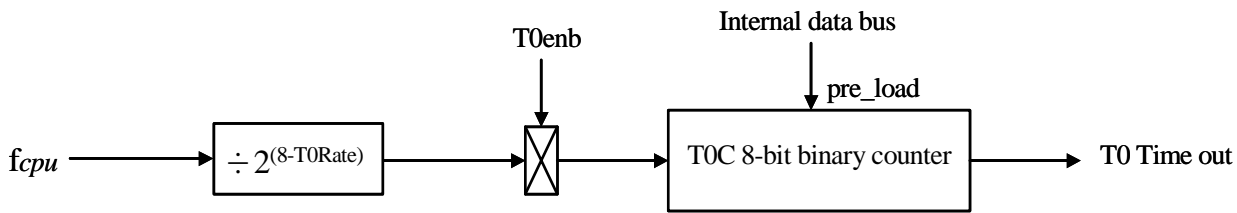


Figure 8-2. Basic Timer T0 Block Diagram

### T0M REGISTER DESCRIPTION

The T0M is the basic timer mode register which is a 8-bit read/write register and only used the high nibble. By loading different value into the T0M register, users can modify the basic timer clock dynamically as program executing.

Eight rates for T0 timer can be selected by T0RATE0 ~ T0RATE2 bits. The range is from fcpu/2 to fcpu/256. The T0M initial value is zero and the rate is fcpu/256. The bit7 of T0M called T0ENB is the control bit to start T0 timer. The combination of these bits is to determine the T0 timer clock frequency and the intervals.

**T0M initial value = 0000 xxxx**

| 0D8H       | Bit 7 | Bit 6   | Bit 5   | Bit 4   | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|------------|-------|---------|---------|---------|-------|-------|-------|-------|
| <b>T0M</b> | T0ENB | T0RATE2 | T0RATE1 | T0RATE0 | 0     | 0     | 0     | 0     |
|            | R/W   | R/W     | R/W     | R/W     | -     | -     | -     | -     |

T0ENB: T0 timer control bit. 0 = disable, 1 = enable.

T0RATE2~T0RATE0: The T0 timer's clock source select bits. 000 = fcpu/256, 001 = fcpu/128, ... , 110 = fcpu/4, 111 = fcpu/2.

## T0C COUNTING REGISTER

T0C is an 8-bit counter register for the basic timer (T0). T0C must be reset whenever the T0ENB is set "1" to start the basic timer. T0C is incremented by one with every clock pulse which frequency is determined by T0RATE0 ~ T0RATE2. When T0C has incremented to "0FFH", it will be cleared to "00H" in next clock and an overflow generated. Under T0 interrupt service request (T0IEN) enable condition, the T0 interrupt request flag will be set "1" and the system executes the interrupt service routine. The T0C has no auto reload function. After T0C overflow, the T0C is continuing counting. Users need to reset T0C value to get a accurate time.

**T0C initial value = xxxx xxxx**

|            |       |       |       |       |       |       |       |       |
|------------|-------|-------|-------|-------|-------|-------|-------|-------|
| 0D9H       | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| <b>T0C</b> | T0C7  | T0C6  | T0C5  | T0C4  | T0C3  | T0C2  | T0C1  | T0C0  |
|            | R/W   | R/W   | R/W   | R/W   | R/W   | R/W   | R/W   | R/W   |

| T0RATE | T0CLOCK  | High speed mode (fcpu = 3.58MHz / 4) |                    | Low speed mode (fcpu = 32768Hz / 4) |                    |
|--------|----------|--------------------------------------|--------------------|-------------------------------------|--------------------|
|        |          | Max overflow interval                | One step = max/256 | Max overflow interval               | One step = max/256 |
| 000    | fcpu/256 | 73.2 ms                              | 286us              | 8000 ms                             | 31.25 ms           |
| 001    | fcpu/128 | 36.6 ms                              | 143us              | 4000 ms                             | 15.63 ms           |
| 010    | fcpu/64  | 18.3 ms                              | 71.5us             | 2000 ms                             | 7.8 ms             |
| 011    | fcpu/32  | 9.15 ms                              | 35.8us             | 1000 ms                             | 3.9 ms             |
| 100    | fcpu/16  | 4.57 ms                              | 17.9us             | 500 ms                              | 1.95 ms            |
| 101    | fcpu/8   | 2.28 ms                              | 8.94us             | 250 ms                              | 0.98 ms            |
| 110    | fcpu/4   | 1.14 ms                              | 4.47us             | 125 ms                              | 0.49 ms            |
| 111    | fcpu/2   | 0.57 ms                              | 2.23us             | 62.5 ms                             | 0.24 ms            |

**Figure 8-3. The Timing Table of Basic Timer T0.**

The equation of T0C initial value is as following.

$$T0C \text{ initial value} = 256 - (T0 \text{ interrupt interval time} * \text{input clock})$$

⇒ Example : To set 10ms interval time for T0 interrupt at 3.58MHz high-speed mode. T0C value (74H) = 256 - (10ms \* fcpu/64)

$$\begin{aligned}
 T0C \text{ initial value} &= 256 - (T0 \text{ interrupt interval time} * \text{input clock}) \\
 &= 256 - (10\text{ms} * 3.58 * 10^6 / 4 / 64) \\
 &= 256 - (10^{-2} * 3.58 * 10^6 / 4 / 64) \\
 &= 116 \\
 &= 74H
 \end{aligned}$$

## T0 BASIC TIMER OPERATION SEQUENCE

The T0 basic timer's sequence of operation can be following.

- Set the T0C initial value to setup the interval time.
- Set the T0ENB to be "1" to enable T0 basic timer.
- T0C is incremented by one with each clock pulse which frequency is corresponding to T0M selection.
- T0C overflow when T0C from FFH to 00H.
- When T0C overflow occur, the T0IRQ flag is set to be "1" by hardware.
- Execute the interrupt service routine.
- Users reset the T0C value and resume the T0 timer operation.

### ⇒ Example: Setup the T0M and T0C.

```

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)
B0BSET    FT0IEN    ; To enable T0 interrupt service
B0BCLR    FT0IRQ    ; To clear T0 interrupt request
B0BSET    FT0ENB    ; To enable T0 timer

```

### ⇒ Example: T0 interrupt service routine.

```

INT_SERVICE:
ORG       8         ; Interrupt vector
JMP      INT_SERVICE

B0XCH    A, ACCBUF  ; B0XCH doesn't change C, Z flag
PUSH                                ; Push

B0BTS1   FT0IRQ    ; Check T0IRQ
JMP      EXIT_INT  ; T0IRQ = 0, exit interrupt vector

B0BCLR   FT0IRQ    ; Reset T0IRQ
MOV      A,#74H    ; Reload T0C
B0MOV    T0C,A     ;
.        .         ; T0 interrupt service routine
.        .         ;
JMP      EXIT_INT  ; End of T0 interrupt service routine and exit interrupt vector
.        .         ;
EXIT_INT:
POP      A         ; Pop
B0XCH    A, ACCBUF ; Restore ACC value

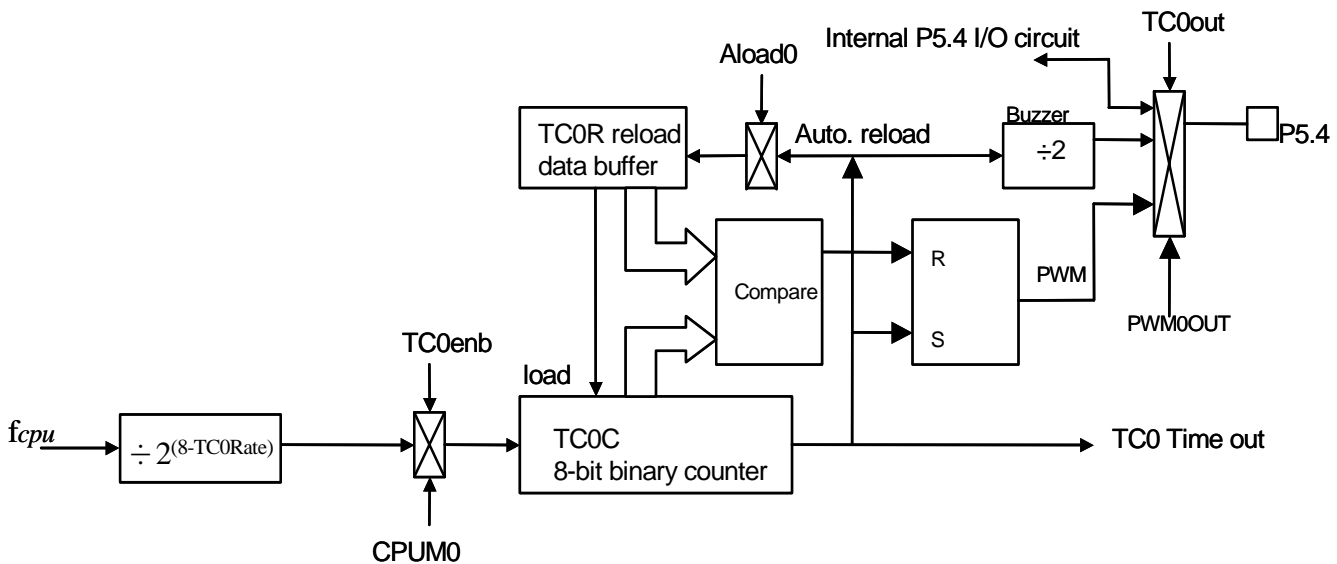
RETI                                ; Exit interrupt vector

```

## TIMER COUNTER 0 (TC0)

### OVERVIEW

The timer counter 0 (TC0) is used to generate an interrupt request when a specified time interval has elapsed. TC0 has a auto re-loadable counter that consists of two parts: an 8-bit reload register (TC0R) into which you write the



counter reference value, and an 8-bit counter register (TC0C) whose value is automatically incremented by counter logic.

Figure 8-4. Timer Count TC0 Block Diagram

The main purposes of the TC0 timer counter is as following.

- **8-bit programmable timer:** Generates interrupts at specific time intervals based on the selected clock frequency.
- **Arbitrary frequency output (Buzzer output):** Outputs selectable clock frequencies to the BZ0 pin (P5.4).
- **PWM function:** PWM output can be generated by the PWM1OUT bit and output to PWM0OUT pin (P5.4).



## TC0M MODE REGISTER

The TC0M is the timer counter mode register, which is an 8-bit read/write register. By loading different value into the TC0M register, users can modify the timer counter clock frequency dynamically when program executing.

Eight rates for TC0 timer can be selected by TC0RATE0 ~ TC0RATE2 bits. The range is from  $f_{cpu}/2$  to  $f_{cpu}/256$ . The TC0M initial value is zero and the rate is  $f_{cpu}/256$ . The bit7 of TC0M called TC0ENB is the control bit to start TC0 timer. The combination of these bits is to determine the TC0 timer clock frequency and the intervals.

**TC0M initial value = 0000 0000**

| 0DAH        | Bit 7  | Bit 6    | Bit 5    | Bit 4    | Bit 3 | Bit 2  | Bit 1  | Bit 0   |
|-------------|--------|----------|----------|----------|-------|--------|--------|---------|
| <b>TC0M</b> | TC0ENB | TC0RATE2 | TC0RATE1 | TC0RATE0 | 0     | ALOAD0 | TC0OUT | PWM0OUT |
|             | R/W    | R/W      | R/W      | R/W      | -     | R/W    | R/W    | R/W     |

TC0ENB: TC0 counter/BZ0/PWM0OUT enable bit. 0 = disable, 1 = enable.

TC0RATE2~TC0RATE0: TC0 internal clock select bits. 000 =  $f_{cpu}/256$ , 001 =  $f_{cpu}/128$ , ... , 110 =  $f_{cpu}/4$ , 111 =  $f_{cpu}/2$ .

ALOAD0: TC0 auto-reload function control bit. 0 = none auto-reload, 1 = auto-reload.

TC0OUT: TC0 time-out toggle signal output control bit. 0 = To disable TC0 signal output and to enable P5.4's I/O function, 1 = To enable TC0's signal output and to disable P5.4's I/O function. (Auto-disable the PWM0OUT function.)

PWM0OUT: TC0's PWM output control bit. 0 = To disable the PWM output, 1 = To enable the PWM output (The TC0OUT control bit must = 0 )

➤ **Note: Bit3 must set to 0..**

➤ **Note: The ICE S8KC do not support the PWM0OUT and TC0OUT Function. The PWM0OUT and TC0OUT must use the S8KD ICE (or later) to verify the function.**

## TC0C COUNTING REGISTER

TC0C is an 8-bit counter register for the timer counter (TC0). TC0C must be reset whenever the TC0ENB is set "1" to start the timer counter. TC0C is incremented by one with a clock pulse which the frequency is determined by TC0RATE0 ~ TC0RATE2. When TC0C has incremented to "0FFH", it will be cleared to "00H" in next clock and an overflow is generated. Under TC0 interrupt service request (TC0IEN) enable condition, the TC0 interrupt request flag will be set "1" and the system executes the interrupt service routine.

**TC0C initial value = xxxx xxxx**

|             |       |       |       |       |       |       |       |       |
|-------------|-------|-------|-------|-------|-------|-------|-------|-------|
| 0DBH        | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| <b>TC0C</b> | TC0C7 | TC0C6 | TC0C5 | TC0C4 | TC0C3 | TC0C2 | TC0C1 | TC0C0 |
|             | R/W   | R/W   | R/W   | R/W   | R/W   | R/W   | R/W   | R/W   |

| TC0RATE | TC0CLOCK | High speed mode (fcpu = 3.58MHz / 4) |                    | Low speed mode (fcpu = 32768Hz / 4) |                    |
|---------|----------|--------------------------------------|--------------------|-------------------------------------|--------------------|
|         |          | Max overflow interval                | One step = max/256 | Max overflow interval               | One step = max/256 |
| 000     | fcpu/256 | 73.2 ms                              | 286us              | 8000 ms                             | 31.25 ms           |
| 001     | fcpu/128 | 36.6 ms                              | 143us              | 4000 ms                             | 15.63 ms           |
| 010     | fcpu/64  | 18.3 ms                              | 71.5us             | 2000 ms                             | 7.8 ms             |
| 011     | fcpu/32  | 9.15 ms                              | 35.8us             | 1000 ms                             | 3.9 ms             |
| 100     | fcpu/16  | 4.57 ms                              | 17.9us             | 500 ms                              | 1.95 ms            |
| 101     | fcpu/8   | 2.28 ms                              | 8.94us             | 250 ms                              | 0.98 ms            |
| 110     | fcpu/4   | 1.14 ms                              | 4.47us             | 125 ms                              | 0.49 ms            |
| 111     | fcpu/2   | 0.57 ms                              | 2.23us             | 62.5 ms                             | 0.24 ms            |

**Table 8-1. The Timing Table of Timer Count TC0**

The equation of TC0C initial value is as following.

$$TC0C \text{ initial value} = 256 - (TC0 \text{ interrupt interval time} * \text{input clock})$$

⇒ **Example:** To set 10ms interval time for TC0 interrupt at 3.58MHz high-speed mode. TC0C value (74H) = 256 - (10ms \* fcpu/64)

$$\begin{aligned}
 TC0C \text{ initial value} &= 256 - (TC0 \text{ interrupt interval time} * \text{input clock}) \\
 &= 256 - (10ms * 3.58 * 10^6 / 4 / 64) \\
 &= 256 - (10^{-2} * 3.58 * 10^6 / 4 / 64) \\
 &= 116 \\
 &= 74H
 \end{aligned}$$

## TC0R AUTO-LOAD REGISTER

TC0R is an 8-bit register for the TC0 auto-reload function. TC0R's value applies to TC0OUT and PWM0OUT functions.. Under TC0OUT application, users must enable and set the TC0R register. The main purpose of TC0R is as following.

- Store the auto-reload value and set into TC0C when the TC0C overflow. (ALOAD0 = 1).
- Store the duty value of PWM0OUT function.

**TC0R initial value = xxxx xxxx**

| OCDH        | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|-------------|-------|-------|-------|-------|-------|-------|-------|-------|
| <b>TC0R</b> | TC0R7 | TC0R6 | TC0R5 | TC0R4 | TC0R3 | TC0R2 | TC0R1 | TC0R0 |
|             | W     | W     | W     | W     | W     | W     | W     | W     |

The equation of TC0R initial value is like TC0C as following.

$$TC0R \text{ initial value} = 256 - (TC0 \text{ interrupt interval time} * \text{input clock})$$

- **Note: The TC0R is write-only register can't be process by INCMS, DECMS instructions.**

## TC0 TIMER COUNTER OPERATION SEQUENCE

The TC0 timer counter's sequence of operation can be following.

- Set the TC0C initial value to setup the interval time.
- Set the TC0ENB to be "1" to enable TC0 timer counter.
- TC0C is incremented by one with each clock pulse which frequency is corresponding to T0M selection.
- TC0C overflow when TC0C from FFH to 00H.
- When TC0C overflow occur, the TC0IRQ flag is set to be "1" by hardware.
- Execute the interrupt service routine.
- Users reset the TC0C value and resume the TC0 timer operation.

### ⇒ Example: Setup the TC0M and TC0C without auto-reload function.

```

B0BCLR    FTC0IEN    ; To disable TC0 interrupt service
B0BCLR    FTC0ENB    ; To disable TC0 timer
MOV       A,#20H     ;
B0MOV     TC0M,A     ; To set TC0 clock = fcpu / 64
MOV       A,#74H     ; To set TC0C initial value = 74H
B0MOV     TC0C,A     ;(To set TC0 interval = 10 ms)

B0BSET    FTC0IEN    ; To enable TC0 interrupt service
B0BCLR    FTC0IRQ    ; To clear TC0 interrupt request
B0BSET    FTC0ENB    ; To enable TC0 timer

```

### ⇒ Example: Setup the TC0M and TC0C with auto-reload function.

```

B0BCLR    FTC0IEN    ; To disable TC0 interrupt service
B0BCLR    FTC0ENB    ; To disable TC0 timer
MOV       A,#20H     ;
B0MOV     TC0M,A     ; To set TC0 clock = fcpu / 64
MOV       A,#74H     ; To set TC0C initial value = 74H
B0MOV     TC0C,A     ; (To set TC0 interval = 10 ms)
B0MOV     TC0R,A     ; To set TC0R auto-reload register

B0BSET    FTC0IEN    ; To enable TC0 interrupt service
B0BCLR    FTC0IRQ    ; To clear TC0 interrupt request
B0BSET    FTC0ENB    ; To enable TC0 timer
B0BSET    ALOAD0     ; To enable TC0 auto-reload function.

```

⇒ Example: TC0 interrupt service routine without auto-reload function.

```

                ORG          8          ; Interrupt vector
INT_SERVICE:   JMP          INT_SERVICE

                BOXCH       A, ACCBUF  ; BOXCH doesn't change C, Z flag
                PUSH                               ; Push

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

                B0BCLR      FTC0IRQ    ; Reset TC0IRQ
                MOV         A,#74H     ; Reload TC0C
                B0MOV       TC0C,A

                .            .          ; TC0 interrupt service routine
                .            .
                JMP          EXIT_INT   ; End of TC0 interrupt service routine and exit interrupt
                                        ; vector

EXIT_INT:      .            .
                .            .
                POP         A          ; Pop
                BOXCH       A, ACCBUF  ; Restore ACC value.

                RETI                               ; Exit interrupt vector

```

⇒ Example: TC0 interrupt service routine with auto-reload.

```

                ORG          8          ; Interrupt vector
INT_SERVICE:   JMP          INT_SERVICE

                BOXCH       A, ACCBUF  ; BOXCH doesn't change C, Z flag
                PUSH                               ; Push

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

                B0BCLR      FTC0IRQ    ; Reset TC0IRQ
                .            .          ; TC0 interrupt service routine
                .            .
                JMP          EXIT_INT   ; End of TC0 interrupt service routine and exit interrupt
                                        ; vector

EXIT_INT:      .            .
                .            .
                POP         A          ; Pop
                BOXCH       A, ACCBUF  ; Restore ACC value.

                RETI                               ; Exit interrupt vector

```

## TC0 CLOCK FREQUENCY OUTPUT (BUZZER)

TC0 timer counter provides a frequency output function. By setting the TC0 clock frequency, the clock signal is output to P5.4 and the P5.4 general purpose I/O function is auto-disable. The TC0 output signal divides by 2. The TC0 clock has many combinations and easily to make difference frequency. This function applies as buzzer output to output multi-frequency.

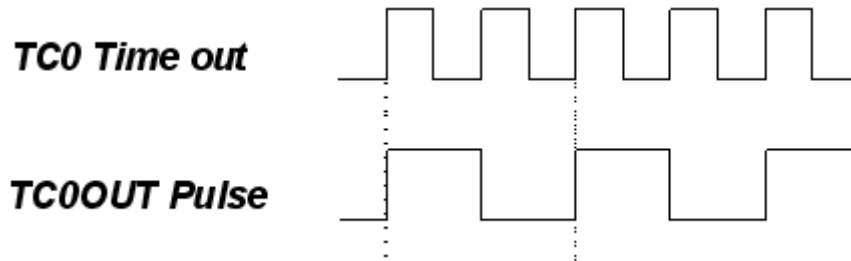


Figure 8-5. The TC0OUT Pulse Frequency

⇒ Example: Setup TC0OUT output from TC0 to TC0OUT (P5.4). The external high-speed clock is 4MHz. The TC0OUT frequency is 1KHz. Because the TC0OUT signal is divided by 2, set the TC0 clock to 2KHz. The TC0 clock source is from external oscillator clock. TC rate is  $F_{cpu}/4$ . The  $TC0RATE2-TC0RATE1 = 110$ .  $TC0C = TC0R = 131$ .

```

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

MOV      A,#131
BOBMV   TC0C,A
BOBMV   TC0R,A

BOBSET  FTC0OUT         ; Enable TC0 output to P5.4 and disable P5.4 I/O function
BOBSET  FALOAD0        ; Enable TC0 auto-reload function
BOBSET  FTC0ENB        ; Enable TC0 timer

```

## TC0OUT FREQUENCY TABLE

*Fosc* = 4MHz, TC0 Rate = *Fcpu*/8

| TC0R | TC0OUT (KHz) | TC0R | TC0OUT (KHz) | TC0R | TC0OUT (KHz) | TC0R | TC0OUT (KHz) | TC0R | TC0OUT (KHz) |
|------|--------------|------|--------------|------|--------------|------|--------------|------|--------------|
| 0    | 0.2441       | 56   | 0.3125       | 112  | 0.4340       | 168  | 0.7102       | 224  | 1.9531       |
| 1    | 0.2451       | 57   | 0.3141       | 113  | 0.4371       | 169  | 0.7184       | 225  | 2.0161       |
| 2    | 0.2461       | 58   | 0.3157       | 114  | 0.4401       | 170  | 0.7267       | 226  | 2.0833       |
| 3    | 0.2470       | 59   | 0.3173       | 115  | 0.4433       | 171  | 0.7353       | 227  | 2.1552       |
| 4    | 0.2480       | 60   | 0.3189       | 116  | 0.4464       | 172  | 0.7440       | 228  | 2.2321       |
| 5    | 0.2490       | 61   | 0.3205       | 117  | 0.4496       | 173  | 0.7530       | 229  | 2.3148       |
| 6    | 0.2500       | 62   | 0.3222       | 118  | 0.4529       | 174  | 0.7622       | 230  | 2.4038       |
| 7    | 0.2510       | 63   | 0.3238       | 119  | 0.4562       | 175  | 0.7716       | 231  | 2.5000       |
| 8    | 0.2520       | 64   | 0.3255       | 120  | 0.4596       | 176  | 0.7813       | 232  | 2.6042       |
| 9    | 0.2530       | 65   | 0.3272       | 121  | 0.4630       | 177  | 0.7911       | 233  | 2.7174       |
| 10   | 0.2541       | 66   | 0.3289       | 122  | 0.4664       | 178  | 0.8013       | 234  | 2.8409       |
| 11   | 0.2551       | 67   | 0.3307       | 123  | 0.4699       | 179  | 0.8117       | 235  | 2.9762       |
| 12   | 0.2561       | 68   | 0.3324       | 124  | 0.4735       | 180  | 0.8224       | 236  | 3.1250       |
| 13   | 0.2572       | 69   | 0.3342       | 125  | 0.4771       | 181  | 0.8333       | 237  | 3.2895       |
| 14   | 0.2583       | 70   | 0.3360       | 126  | 0.4808       | 182  | 0.8446       | 238  | 3.4722       |
| 15   | 0.2593       | 71   | 0.3378       | 127  | 0.4845       | 183  | 0.8562       | 239  | 3.6765       |
| 16   | 0.2604       | 72   | 0.3397       | 128  | 0.4883       | 184  | 0.8681       | 240  | 3.9063       |
| 17   | 0.2615       | 73   | 0.3415       | 129  | 0.4921       | 185  | 0.8803       | 241  | 4.1667       |
| 18   | 0.2626       | 74   | 0.3434       | 130  | 0.4960       | 186  | 0.8929       | 242  | 4.4643       |
| 19   | 0.2637       | 75   | 0.3453       | 131  | 0.5000       | 187  | 0.9058       | 243  | 4.8077       |
| 20   | 0.2648       | 76   | 0.3472       | 132  | 0.5040       | 188  | 0.9191       | 244  | 5.2083       |
| 21   | 0.2660       | 77   | 0.3492       | 133  | 0.5081       | 189  | 0.9328       | 245  | 5.6818       |
| 22   | 0.2671       | 78   | 0.3511       | 134  | 0.5123       | 190  | 0.9470       | 246  | 6.2500       |
| 23   | 0.2682       | 79   | 0.3531       | 135  | 0.5165       | 191  | 0.9615       | 247  | 6.9444       |
| 24   | 0.2694       | 80   | 0.3551       | 136  | 0.5208       | 192  | 0.9766       | 248  | 7.8125       |
| 25   | 0.2706       | 81   | 0.3571       | 137  | 0.5252       | 193  | 0.9921       | 249  | 8.9286       |
| 26   | 0.2717       | 82   | 0.3592       | 138  | 0.5297       | 194  | 1.0081       | 250  | 10.4167      |
| 27   | 0.2729       | 83   | 0.3613       | 139  | 0.5342       | 195  | 1.0246       | 251  | 12.5000      |
| 28   | 0.2741       | 84   | 0.3634       | 140  | 0.5388       | 196  | 1.0417       | 252  | 15.6250      |
| 29   | 0.2753       | 85   | 0.3655       | 141  | 0.5435       | 197  | 1.0593       | 253  | 20.8333      |
| 30   | 0.2765       | 86   | 0.3676       | 142  | 0.5482       | 198  | 1.0776       | 254  | 31.2500      |
| 31   | 0.2778       | 87   | 0.3698       | 143  | 0.5531       | 199  | 1.0965       | 255  | 62.5000      |
| 32   | 0.2790       | 88   | 0.3720       | 144  | 0.5580       | 200  | 1.1161       |      |              |
| 33   | 0.2803       | 89   | 0.3743       | 145  | 0.5631       | 201  | 1.1364       |      |              |
| 34   | 0.2815       | 90   | 0.3765       | 146  | 0.5682       | 202  | 1.1574       |      |              |
| 35   | 0.2828       | 91   | 0.3788       | 147  | 0.5734       | 203  | 1.1792       |      |              |
| 36   | 0.2841       | 92   | 0.3811       | 148  | 0.5787       | 204  | 1.2019       |      |              |
| 37   | 0.2854       | 93   | 0.3834       | 149  | 0.5841       | 205  | 1.2255       |      |              |
| 38   | 0.2867       | 94   | 0.3858       | 150  | 0.5896       | 206  | 1.2500       |      |              |
| 39   | 0.2880       | 95   | 0.3882       | 151  | 0.5952       | 207  | 1.2755       |      |              |
| 40   | 0.2894       | 96   | 0.3906       | 152  | 0.6010       | 208  | 1.3021       |      |              |
| 41   | 0.2907       | 97   | 0.3931       | 153  | 0.6068       | 209  | 1.3298       |      |              |
| 42   | 0.2921       | 98   | 0.3956       | 154  | 0.6127       | 210  | 1.3587       |      |              |
| 43   | 0.2934       | 99   | 0.3981       | 155  | 0.6188       | 211  | 1.3889       |      |              |
| 44   | 0.2948       | 100  | 0.4006       | 156  | 0.6250       | 212  | 1.4205       |      |              |
| 45   | 0.2962       | 101  | 0.4032       | 157  | 0.6313       | 213  | 1.4535       |      |              |
| 46   | 0.2976       | 102  | 0.4058       | 158  | 0.6378       | 214  | 1.4881       |      |              |
| 47   | 0.2990       | 103  | 0.4085       | 159  | 0.6443       | 215  | 1.5244       |      |              |
| 48   | 0.3005       | 104  | 0.4112       | 160  | 0.6510       | 216  | 1.5625       |      |              |
| 49   | 0.3019       | 105  | 0.4139       | 161  | 0.6579       | 217  | 1.6026       |      |              |
| 50   | 0.3034       | 106  | 0.4167       | 162  | 0.6649       | 218  | 1.6447       |      |              |
| 51   | 0.3049       | 107  | 0.4195       | 163  | 0.6720       | 219  | 1.6892       |      |              |
| 52   | 0.3064       | 108  | 0.4223       | 164  | 0.6793       | 220  | 1.7361       |      |              |
| 53   | 0.3079       | 109  | 0.4252       | 165  | 0.6868       | 221  | 1.7857       |      |              |
| 54   | 0.3094       | 110  | 0.4281       | 166  | 0.6944       | 222  | 1.8382       |      |              |
| 55   | 0.3109       | 111  | 0.4310       | 167  | 0.7022       | 223  | 1.8939       |      |              |

**Table 8-2. TC0OUT Frequency Table for *Fosc* = 4MHz, TC0 Rate = *Fcpu*/8**

**Fosc = 16MHz, TC0 Rate = Fcpu/8**

| TC0R | TC0OUT (KHz) | TC0R | TC0OUT (KHz) | TC0R | TC0OUT (KHz) | TC0R | TC0OUT (KHz) | TC0R | TC0OUT (KHz) |
|------|--------------|------|--------------|------|--------------|------|--------------|------|--------------|
| 0    | 0.9766       | 56   | 1.2500       | 112  | 1.7361       | 168  | 2.8409       | 224  | 7.8125       |
| 1    | 0.9804       | 57   | 1.2563       | 113  | 1.7483       | 169  | 2.8736       | 225  | 8.0645       |
| 2    | 0.9843       | 58   | 1.2626       | 114  | 1.7606       | 170  | 2.9070       | 226  | 8.3333       |
| 3    | 0.9881       | 59   | 1.2690       | 115  | 1.7730       | 171  | 2.9412       | 227  | 8.6207       |
| 4    | 0.9921       | 60   | 1.2755       | 116  | 1.7857       | 172  | 2.9762       | 228  | 8.9286       |
| 5    | 0.9960       | 61   | 1.2821       | 117  | 1.7986       | 173  | 3.0120       | 229  | 9.2593       |
| 6    | 1.0000       | 62   | 1.2887       | 118  | 1.8116       | 174  | 3.0488       | 230  | 9.6154       |
| 7    | 1.0040       | 63   | 1.2953       | 119  | 1.8248       | 175  | 3.0864       | 231  | 10.0000      |
| 8    | 1.0081       | 64   | 1.3021       | 120  | 1.8382       | 176  | 3.1250       | 232  | 10.4167      |
| 9    | 1.0121       | 65   | 1.3089       | 121  | 1.8519       | 177  | 3.1646       | 233  | 10.8696      |
| 10   | 1.0163       | 66   | 1.3158       | 122  | 1.8657       | 178  | 3.2051       | 234  | 11.3636      |
| 11   | 1.0204       | 67   | 1.3228       | 123  | 1.8797       | 179  | 3.2468       | 235  | 11.9048      |
| 12   | 1.0246       | 68   | 1.3298       | 124  | 1.8939       | 180  | 3.2895       | 236  | 12.5000      |
| 13   | 1.0288       | 69   | 1.3369       | 125  | 1.9084       | 181  | 3.3333       | 237  | 13.1579      |
| 14   | 1.0331       | 70   | 1.3441       | 126  | 1.9231       | 182  | 3.3784       | 238  | 13.8889      |
| 15   | 1.0373       | 71   | 1.3514       | 127  | 1.9380       | 183  | 3.4247       | 239  | 14.7059      |
| 16   | 1.0417       | 72   | 1.3587       | 128  | 1.9531       | 184  | 3.4722       | 240  | 15.6250      |
| 17   | 1.0460       | 73   | 1.3661       | 129  | 1.9685       | 185  | 3.5211       | 241  | 16.6667      |
| 18   | 1.0504       | 74   | 1.3736       | 130  | 1.9841       | 186  | 3.5714       | 242  | 17.8571      |
| 19   | 1.0549       | 75   | 1.3812       | 131  | 2.0000       | 187  | 3.6232       | 243  | 19.2308      |
| 20   | 1.0593       | 76   | 1.3889       | 132  | 2.0161       | 188  | 3.6765       | 244  | 20.8333      |
| 21   | 1.0638       | 77   | 1.3966       | 133  | 2.0325       | 189  | 3.7313       | 245  | 22.7273      |
| 22   | 1.0684       | 78   | 1.4045       | 134  | 2.0492       | 190  | 3.7879       | 246  | 25.0000      |
| 23   | 1.0730       | 79   | 1.4124       | 135  | 2.0661       | 191  | 3.8462       | 247  | 27.7778      |
| 24   | 1.0776       | 80   | 1.4205       | 136  | 2.0833       | 192  | 3.9063       | 248  | 31.2500      |
| 25   | 1.0823       | 81   | 1.4286       | 137  | 2.1008       | 193  | 3.9683       | 249  | 35.7143      |
| 26   | 1.0870       | 82   | 1.4368       | 138  | 2.1186       | 194  | 4.0323       | 250  | 41.6667      |
| 27   | 1.0917       | 83   | 1.4451       | 139  | 2.1368       | 195  | 4.0984       | 251  | 50.0000      |
| 28   | 1.0965       | 84   | 1.4535       | 140  | 2.1552       | 196  | 4.1667       | 252  | 62.5000      |
| 29   | 1.1013       | 85   | 1.4620       | 141  | 2.1739       | 197  | 4.2373       | 253  | 83.3333      |
| 30   | 1.1062       | 86   | 1.4706       | 142  | 2.1930       | 198  | 4.3103       | 254  | 125.0000     |
| 31   | 1.1111       | 87   | 1.4793       | 143  | 2.2124       | 199  | 4.3860       | 255  | 250.0000     |
| 32   | 1.1161       | 88   | 1.4881       | 144  | 2.2321       | 200  | 4.4643       |      |              |
| 33   | 1.1211       | 89   | 1.4970       | 145  | 2.2523       | 201  | 4.5455       |      |              |
| 34   | 1.1261       | 90   | 1.5060       | 146  | 2.2727       | 202  | 4.6296       |      |              |
| 35   | 1.1312       | 91   | 1.5152       | 147  | 2.2936       | 203  | 4.7170       |      |              |
| 36   | 1.1364       | 92   | 1.5244       | 148  | 2.3148       | 204  | 4.8077       |      |              |
| 37   | 1.1416       | 93   | 1.5337       | 149  | 2.3364       | 205  | 4.9020       |      |              |
| 38   | 1.1468       | 94   | 1.5432       | 150  | 2.3585       | 206  | 5.0000       |      |              |
| 39   | 1.1521       | 95   | 1.5528       | 151  | 2.3810       | 207  | 5.1020       |      |              |
| 40   | 1.1574       | 96   | 1.5625       | 152  | 2.4038       | 208  | 5.2083       |      |              |
| 41   | 1.1628       | 97   | 1.5723       | 153  | 2.4272       | 209  | 5.3191       |      |              |
| 42   | 1.1682       | 98   | 1.5823       | 154  | 2.4510       | 210  | 5.4348       |      |              |
| 43   | 1.1737       | 99   | 1.5924       | 155  | 2.4752       | 211  | 5.5556       |      |              |
| 44   | 1.1792       | 100  | 1.6026       | 156  | 2.5000       | 212  | 5.6818       |      |              |
| 45   | 1.1848       | 101  | 1.6129       | 157  | 2.5253       | 213  | 5.8140       |      |              |
| 46   | 1.1905       | 102  | 1.6234       | 158  | 2.5510       | 214  | 5.9524       |      |              |
| 47   | 1.1962       | 103  | 1.6340       | 159  | 2.5773       | 215  | 6.0976       |      |              |
| 48   | 1.2019       | 104  | 1.6447       | 160  | 2.6042       | 216  | 6.2500       |      |              |
| 49   | 1.2077       | 105  | 1.6556       | 161  | 2.6316       | 217  | 6.4103       |      |              |
| 50   | 1.2136       | 106  | 1.6667       | 162  | 2.6596       | 218  | 6.5789       |      |              |
| 51   | 1.2195       | 107  | 1.6779       | 163  | 2.6882       | 219  | 6.7568       |      |              |
| 52   | 1.2255       | 108  | 1.6892       | 164  | 2.7174       | 220  | 6.9444       |      |              |
| 53   | 1.2315       | 109  | 1.7007       | 165  | 2.7473       | 221  | 7.1429       |      |              |
| 54   | 1.2376       | 110  | 1.7123       | 166  | 2.7778       | 222  | 7.3529       |      |              |
| 55   | 1.2438       | 111  | 1.7241       | 167  | 2.8090       | 223  | 7.5758       |      |              |

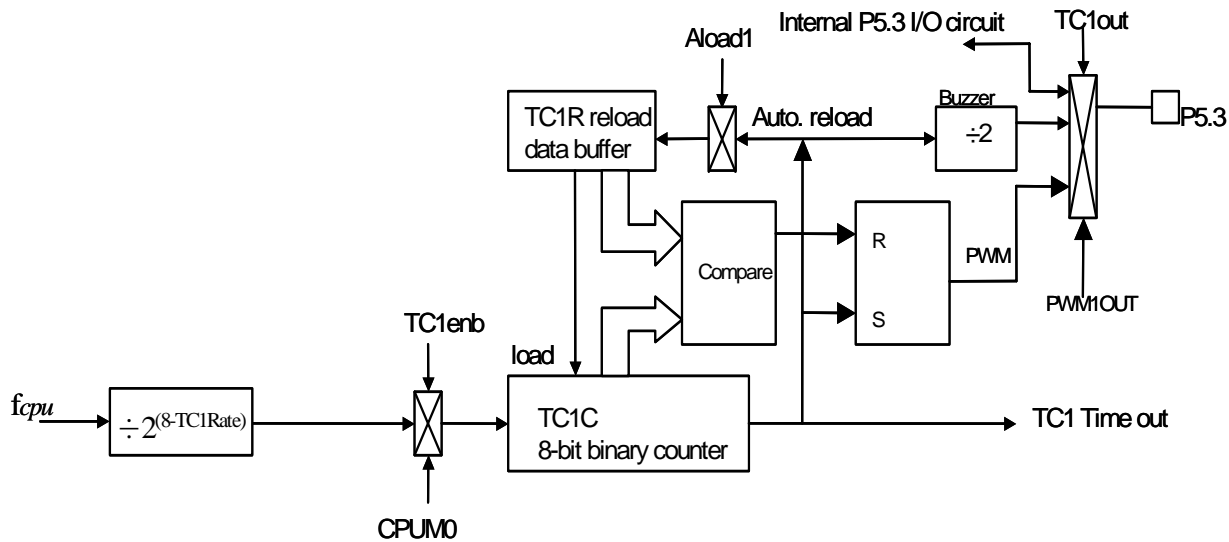
**Table 8-3. TC0OUT Frequency Table for Fosc = 16MHz, TC0 Rate = Fcpu/8**



## TIMER COUNTER 1 (TC1)

### OVERVIEW

The timer counter 1 (TC1) is used to generate an interrupt request when a specified time interval has elapsed. 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.

Figure 8-6. Timer Count TC1 Block Diagram

The main purposes of the TC1 timer is as following.

- **8-bit programmable timer:** Generates interrupts at specific time intervals based on the selected clock frequency.
- **Arbitrary frequency output (Buzzer output):** Outputs selectable clock frequencies to the BZ1 pin (P5.3).
- **PWM function:** PWM output can be generated by the PWM1OUT bit and output to PWM1OUT pin (P5.3).

## TC1M MODE REGISTER

The TC1M is an 8-bit read/write timer mode register. By loading different value into the TC1M register, users can modify the timer clock frequency dynamically as program executing.

Eight rates for TC1 timer can be selected by TC1RATE0 ~ TC1RATE2 bits. The range is from  $f_{cpu}/2$  to  $f_{cpu}/256$ . The TC1M initial value is zero and the rate is  $f_{cpu}/256$ . The bit7 of TC1M called TC1ENB is the control bit to start TC1 timer. The combination of these bits is to determine the TC1 timer clock frequency and the intervals.

**TC1M initial value = 0000 0000**

| 0DCH        | Bit 7  | Bit 6    | Bit 5    | Bit 4    | Bit 3 | Bit 2  | Bit 1  | Bit 0   |
|-------------|--------|----------|----------|----------|-------|--------|--------|---------|
| <b>TC1M</b> | TC1ENB | TC1RATE2 | TC1RATE1 | TC1RATE0 | 0     | ALOAD1 | TC1OUT | PWM1OUT |
|             | R/W    | R/W      | R/W      | R/W      | -     | R/W    | R/W    | R/W     |

TC1ENB: TC1 counter/BZ1/PWM1OUT enable bit. 0 = disable, 1 = enable.

TC1RATE2~TC1RATE0: TC1 internal clock select bits. 000 =  $f_{cpu}/256$ , 001 =  $f_{cpu}/128$ , ... , 110 =  $f_{cpu}/4$ , 111 =  $f_{cpu}/2$ .

ALOAD1: TC1 auto-reload function 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 function.)

PWM1OUT: TC1's PWM output control bit. 0 = To disable the PWM output, 1 = To enable the PWM output (The TC1OUT control bit must = 0 )

➤ **Note: Bit3 must set to 0..**

➤ **Note: The S8KC ICE do not support the PWM1OUT and TC1OUT Function. The PWM1OUT and TC1OUT must use the S8KD ICE (or later) to verify the function.**

## TC1C COUNTING REGISTER

TC1C is an 8-bit counter register for the timer counter (TC1). TC1C must be reset whenever the TC1ENB is set “1” to start the timer. TC0C is incremented by one with a clock pulse which the frequency is determined by TC0RATE0 ~ TC0RATE2. When TC0C has incremented to “0FFH”, it will be cleared to “00H” in next clock and an overflow is generated. Under TC1 interrupt service request (TC1IEN) enable condition, the TC1 interrupt request flag will be set “1” and the system executes the interrupt service routine.

**TC1C initial value = xxxx xxxx**

|             |       |       |       |       |       |       |       |       |
|-------------|-------|-------|-------|-------|-------|-------|-------|-------|
| 0DDH        | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| <b>TC1C</b> | TC1C7 | TC1C6 | TC1C5 | TC1C4 | TC1C3 | TC1C2 | TC1C1 | TC1C0 |
|             | R/W   | R/W   | R/W   | R/W   | R/W   | R/W   | R/W   | R/W   |

**The interval time of TC1 basic timer table.**

| TC1RATE | TC1CLOC<br>K | High speed mode (fcpu = 3.58MHz / 4) |                    | Low speed mode (fcpu = 32768Hz / 4) |                    |
|---------|--------------|--------------------------------------|--------------------|-------------------------------------|--------------------|
|         |              | Max overflow interval                | One step = max/256 | Max overflow interval               | One step = max/256 |
| 000     | fcpu/256     | 73.2 ms                              | 286us              | 8000 ms                             | 31.25 ms           |
| 001     | fcpu/128     | 36.6 ms                              | 143us              | 4000 ms                             | 15.63 ms           |
| 010     | fcpu/64      | 18.3 ms                              | 71.5us             | 2000 ms                             | 7.8 ms             |
| 011     | fcpu/32      | 9.15 ms                              | 35.8us             | 1000 ms                             | 3.9 ms             |
| 100     | fcpu/16      | 4.57 ms                              | 17.9us             | 500 ms                              | 1.95 ms            |
| 101     | fcpu/8       | 2.28 ms                              | 8.94us             | 250 ms                              | 0.98 ms            |
| 110     | fcpu/4       | 1.14 ms                              | 4.47us             | 125 ms                              | 0.49 ms            |
| 111     | fcpu/2       | 0.57 ms                              | 2.23us             | 62.5 ms                             | 0.24 ms            |

**Table 8-4. The Timing Table of Timer Count TC1**

The equation of TC1C initial value is as following.

$$\boxed{TC1C\ initial\ value = 256 - (TC1\ interrupt\ interval\ time * input\ clock)}$$

⇒ **Example: To set 10ms interval time for TC1 interrupt at 3.58MHz high-speed mode. TC1C value (74H) = 256 - (10ms \* fcpu/64)**

$$\begin{aligned}
 TC1C\ initial\ value &= 256 - (TC1\ interrupt\ interval\ time * input\ clock) \\
 &= 256 - (10ms * 3.58 * 10^6 / 4 / 64) \\
 &= 256 - (10^{-2} * 3.58 * 10^6 / 4 / 64) \\
 &= 116 \\
 &= 74H
 \end{aligned}$$

## TC1R AUTO-LOAD REGISTER

TC1R is an 8-bit register for the TC1 auto-reload function. TC1R's value applies to TC1OUT and PWM1OUT functions. Under TC1OUT application, users must enable and set the TC1R register. The main purpose of TC1R is as following.

- Store the auto-reload value and set into TC1C when the TC1C overflow. (ALOAD1 = 1).
- Store the duty value of PWM1OUT function.

**TC1R initial value = xxxx xxxx**

| 0DEH        | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|-------------|-------|-------|-------|-------|-------|-------|-------|-------|
| <b>TC1R</b> | TC1R7 | TC1R6 | TC1R5 | TC1R4 | TC1R3 | TC1R2 | TC1R1 | TC1R0 |
|             | W     | W     | W     | W     | W     | W     | W     | W     |

The equation of TC1R initial value is like TC1C as following.

$$TC1R \text{ initial value} = 256 - (TC1 \text{ interrupt interval time} * \text{input clock})$$

- **Note: The TC1R is write-only register can't be process by INCMS, DECMS instructions.**

## TC1 TIMER COUNTER OPERATION SEQUENCE

The TC1 timer's sequence of operation can be following.

- Set the TC1C initial value to setup the interval time.
- Set the TC1ENB to be "1" to enable TC1 timer counter.
- TC1C is incremented by one with each clock pulse which frequency is corresponding to TC1M selection.
- TC1C overflow if TC1C from FFH to 00H.
- When TC1C overflow occur, the TC1IRQ flag is set to be "1" by hardware.
- Execute the interrupt service routine.
- Users reset the TC1C value and resume the TC1 timer operation.

### ⇒ Example: Setup the TC1M and TC1C without auto-reload function.

```

B0BCLR    FTC1IEN    ; To disable TC1 interrupt service
B0BCLR    FTC1ENB    ; To disable TC1 timer
MOV       A,#20H    ;
B0MOV     TC1M,A     ; To set TC1 clock = fcpu / 64
MOV       A,#74H    ; To set TC1C initial value = 74H
B0MOV     TC1C,A     ;(To set TC1 interval = 10 ms)

B0BSET    FTC1IEN    ; To enable TC1 interrupt service
B0BCLR    FTC1IRQ    ; To clear TC1 interrupt request
B0BSET    FTC1ENB    ; To enable TC1 timer

```

### ⇒ Example: Setup the TC1M and TC1C with auto-reload function.

```

B0BCLR    FTC1IEN    ; To disable TC1 interrupt service
B0BCLR    FTC1ENB    ; To disable TC1 timer
MOV       A,#20H    ;
B0MOV     TC1M,A     ; To set TC1 clock = fcpu / 64
MOV       A,#74H    ; To set TC1C initial value = 74H
B0MOV     TC1C,A     ; (To set TC1 interval = 10 ms)
B0MOV     TC1R,A     ; To set TC1R auto-reload register

B0BSET    FTC1IEN    ; To enable TC1 interrupt service
B0BCLR    FTC1IRQ    ; To clear TC1 interrupt request
B0BSET    FTC1ENB    ; To enable TC1 timer
B0BSET    ALOAD1     ; To enable TC1 auto-reload function.

```

⇒ Example: TC1 interrupt service routine without auto-reload function.

```

                ORG          8          ; Interrupt vector
INT_SERVICE:   JMP          INT_SERVICE

                BOXCH       A, ACCBUF  ; BOXCH doesn't change C, Z flag
                PUSH                               ; Push

                B0BTS1      FTC1IRQ    ; Check TC1IRQ
                JMP          EXIT_INT   ; TC1IRQ = 0, exit interrupt vector

                B0BCLR      FTC1IRQ    ; Reset TC1IRQ
                MOV         A,#74H     ; Reload TC1C
                B0MOV       TC1C,A

                .            .          ; TC1 interrupt service routine
                .            .
                JMP          EXIT_INT   ; End of TC1 interrupt service routine and exit interrupt
                                        ; vector

EXIT_INT:     .            .
                .            .
                POP         A, ACCBUF  ; Pop
                BOXCH       A, ACCBUF  ; Restore ACC value.

                RETI                               ; Exit interrupt vector

```

⇒ Example: TC1 interrupt service routine with auto-reload.

```

                ORG          8          ; Interrupt vector
INT_SERVICE:   JMP          INT_SERVICE

                BOXCH       A, ACCBUF  ; BOXCH doesn't change C, Z flag
                PUSH                               ; Push

                B0BTS1      FTC1IRQ    ; Check TC1IRQ
                JMP          EXIT_INT   ; TC1IRQ = 0, exit interrupt vector

                B0BCLR      FTC1IRQ    ; Reset TC1IRQ
                .            .          ; TC1 interrupt service routine
                .            .
                JMP          EXIT_INT   ; End of TC1 interrupt service routine and exit interrupt
                                        ; vector

EXIT_INT:     .            .
                .            .
                POP         A, ACCBUF  ; Pop
                BOXCH       A, ACCBUF  ; Restore ACC value.

                RETI                               ; Exit interrupt vector

```

## TC1 CLOCK FREQUENCY OUTPUT (BUZZER)

TC1 timer counter provides a frequency output function. By setting the TC1 clock frequency, the clock signal is output to P5.3 and the P5.3 general purpose I/O function is auto-disable. The TC1 output signal divides by 2. The TC1 clock has many combinations and easily to make difference frequency. This function applies as buzzer output to output multi-frequency.

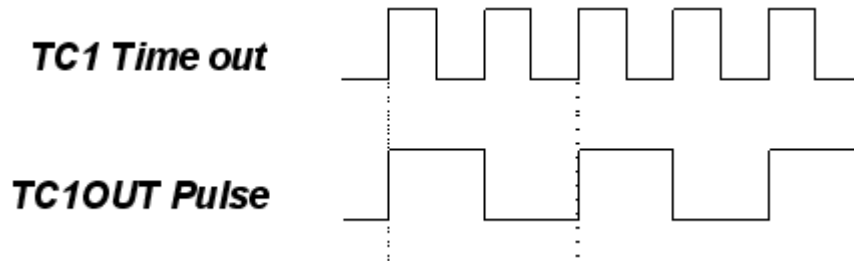


Figure 8-7. The TC1OUT Pulse Frequency

⇒ **Example:** Setup TC1OUT output from TC1 to TC1OUT (P5.3). The external high-speed clock is 4MHz. The TC1OUT frequency is 1KHz. Because the TC1OUT signal is divided by 2, set the TC1 clock to 2KHz. The TC1 clock source is from external oscillator clock. TC1 rate is  $F_{cpu}/4$ . The  $TC1RATE2-TC1RATE1 = 110$ .  $TC1C = TC1R = 131$ .

```

MOV      A,#01100000B
B0MOV   TC1M,A           ; Set the TC1 rate to Fcpu/4

MOV      A,#131
B0MOV   TC1C,A          ; Set the auto-reload reference value
B0MOV   TC1R,A

B0BSET  FTC1OUT         ; Enable TC1 output to P5.3 and disable P5.3 I/O function
B0BSET  FALOAD1        ; Enable TC1 auto-reload function
B0BSET  FTC1ENB        ; Enable TC1 timer

```

➤ **Note:** The TC1OUT frequency table is as TC0OUT frequency table. Please consult TC0OUT frequency table. (Table 7-2~7-5)

# PWM FUNCTION DESCRIPTION

## OVERVIEW

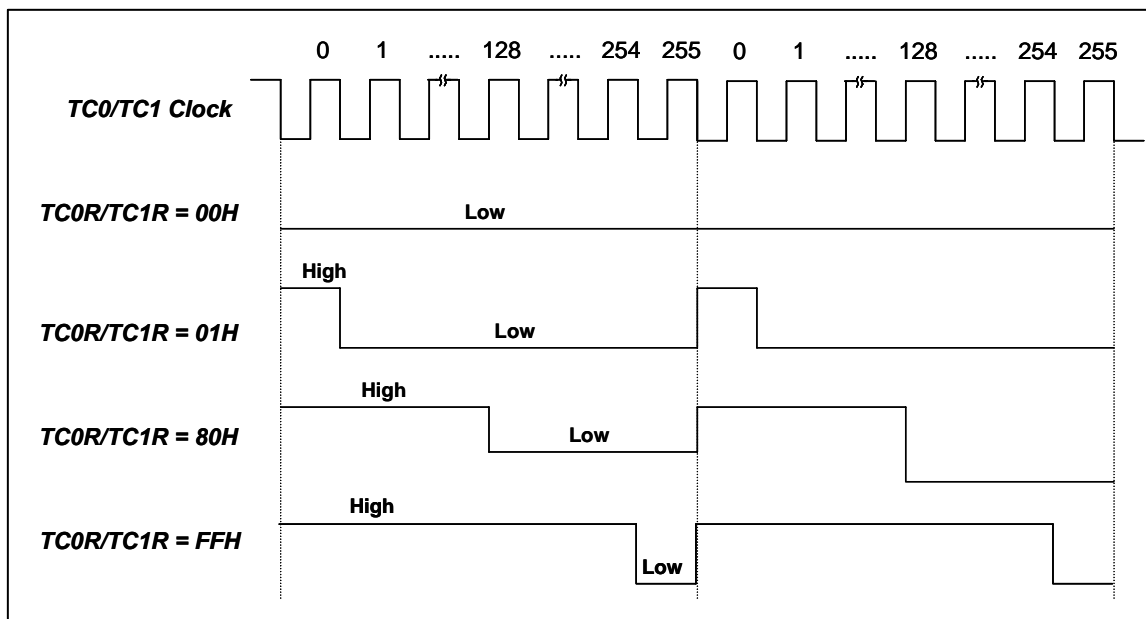
PWM function is generated by TC0/TC1 timer counter and output the PWM signal to PWM0OUT pin (P5.4)/ PWM1OUT pin (P5.3). The 8-bit counter counts modulus 256, from 0-255, inclusive. The value of the 8-bit counter is compared to the contents of the reference register (TC0R/TC1R). When the reference register value (TC0R/TC1R) is equal to the counter value (TC0C/TC1C), the PWM output goes low. When the counter reaches zero, the PWM output is forced high. The low-to-high ratio (duty) of the PWM0/PWM1 output is TC0R/256 and TC1R/256.

All PWM outputs remain inactive during the first 256 input clock signals. Then, when the counter value (TC0C/TC1C) changes from FFH back to 00H, the PWM output is forced to high level. The pulse width ratio (duty cycle) is defined by the contents of the reference register (TC0R/TC1R) and is programmed in increments of 1:256. The 8-bit PWM data register TC0R/TC1R is write only register.

PWM output can be held at low level by continuously loading the reference register with 00H. Under PWM operating, to change the PWM's duty cycle is to modify the TC0R/TC1R.

| Reference Register Value<br>(TC0R/TC1R) | 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 |

**Table 8-5. The PWM Duty Cycle Table**



**Figure 8-8 The Output of PWM with different TC0R/TC1R.**



## PWM PROGRAM DESCRIPTION

- ⇒ **Example:** Setup PWM0 output from TC0 to PWM0OUT (P5.4). The external high-speed oscillator clock is 4MHz. 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 TC0RATE2~TC0RATE1 = 110. TC0C = TC0R = 30.

```

MOV          A,#01100000B
B0MOV       TC0M,A           ; Set the TC0 rate to Fcpu/4
B0MOV       TC0M,A           ; Set the TC0 rate to Fcpu/4
MOV         A,#0x00          ;First Time Initial TC0

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

B0BCLR      FTC0OUT          ; Disable TC0OUT function.
B0BSET      FPWM0OUT         ; Enable PWM0 output to P5.4 and disable P5.4 I/O function
B0BSET      FTC0ENB          ; Enable TC0 timer

```

- **Note1:** The TC0R and TC1R are write-only registers. Don't process them using INCMS, DECMS instructions.
- **Note2:** Set TC0C at initial is to make first duty-cycle correct. After TC0 is enabled, don't modify TC0R value to avoid duty cycle error of PWM output.

- ⇒ **Example:** Modify TC0R/TC1R 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
B0MOV       A, BUF0          ; programming.
B0MOV       TC0R, A

```

- **Note3:** That is better to set the TC0C and TC0R value together when PWM0 duty modified. It protects the PWM0 signal no glitch as PWM0 duty changing. That is better to set the TC1C and TC1R value together when PWM1 duty modified. It protects the PWM1 signal no glitch as PWM1 duty changing.
- **Note4:** The TC0OUT function must be set "0" when PWM0 output enable. The TC1OUT function must be set "0" when PWM1 output enable.
- **Note5:** The PWM can work with interrupt request.

# 9 INTERRUPT

## OVERVIEW

The SN8P1700 provides 7<sup>1</sup> interrupt sources, including four internal interrupts (T0, TC0, TC1 & SIO) and three external interrupts (INT0 ~ INT2). These external interrupts can wakeup the chip from power down mode to high-speed normal mode. The external clock input pins of INT0/INT1/INT2 are shared with P0.0/P0.1/P0.2 pins. Once interrupt service is executed, the GIE bit in STKP register will clear to "0" for stopping other interrupt request. When interrupt service exits, the GIE 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.

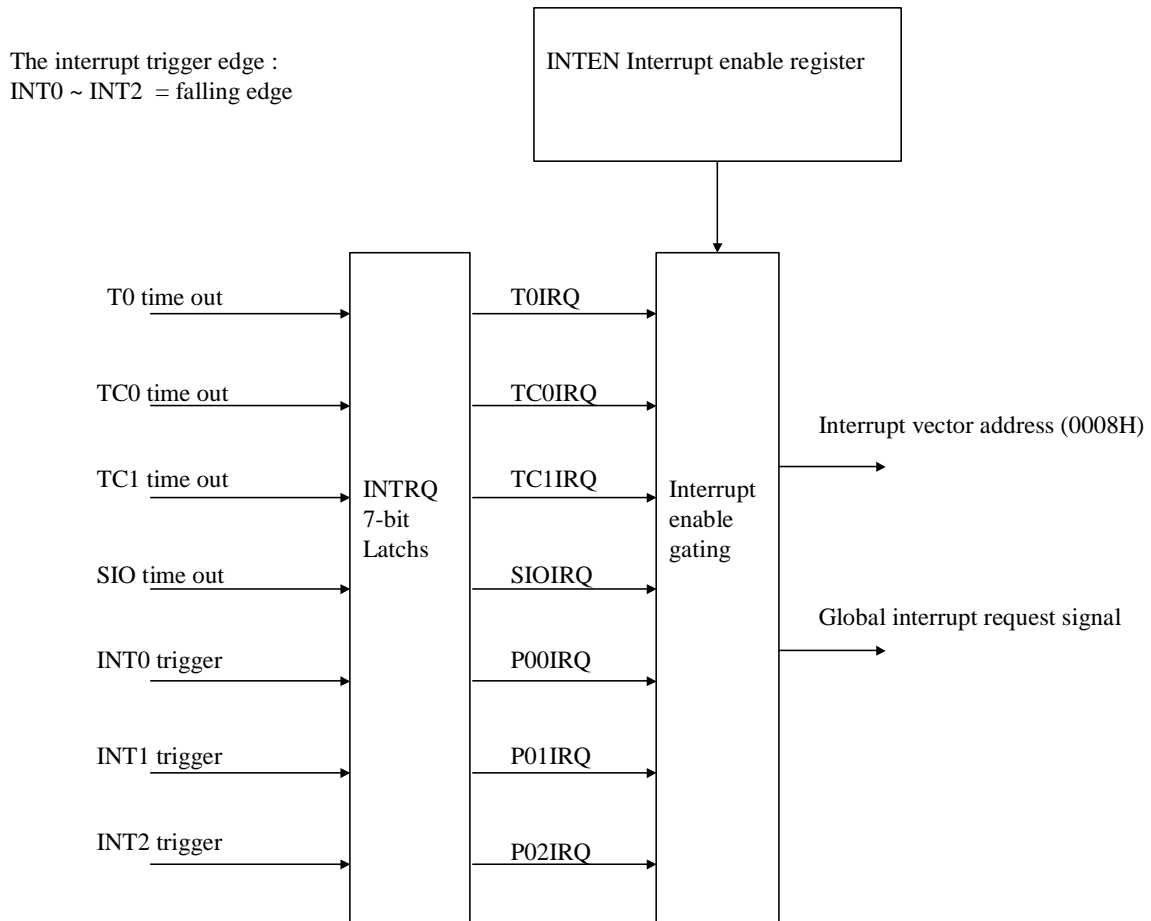


Figure 9-1. The 7 Interrupts of SN8P1700

- **Note: 1.** For SN8P1702 only obtain one internal interrupt P00 and one external interrupt TC0.
- **Note: 2.** The GIE bit must enable and all interrupt operations work.

## INTEN INTERRUPT ENABLE REGISTER

INTEN is the interrupt request control register including four internal interrupts, three external interrupts and SIO interrupt enable control bits. One of the register to be set "1" is to enable the interrupt request function. Once of the interrupt occur, the program jump to ORG 8 to execute interrupt service routines. The program exits the interrupt service routine when the returning interrupt service routine instruction (RETI) is executed.

**INTEN initial value = x000 0000**

| 0C9H         | Bit 7 | Bit 6  | Bit 5  | Bit 4 | Bit 3  | Bit 2  | Bit 1  | Bit 0  |
|--------------|-------|--------|--------|-------|--------|--------|--------|--------|
| <b>INTEN</b> | 0     | TC1IEN | TC0IEN | TOIEN | SIOIEN | P02IEN | P01IEN | P00IEN |
|              | -     | R/W    | R/W    | R/W   | R/W    | R/W    | R/W    | R/W    |

P00IEN : External P0.0 interrupt control bit. 0 = disable, 1 = enable.

P01IEN : External P0.1 interrupt control bit. 0 = disable, 1 = enable.

P02IEN : External P0.2 interrupt control bit. 0 = disable, 1 = enable.

SIOIEN : SIO interrupt control bit. 0 = disable, 1 = enable.

TOIEN : T0 timer interrupt control bit. 0 = disable, 1 = enable.

TC0IEN : Timer interrupt control bit. 0 = disable, 1 = enable.

TC1IEN : Timer interrupt control bit. 0 = disable, 1 = enable.

## INTRQ INTERRUPT REQUEST REGISTER

INTRQ is the interrupt request flag register. The register includes all interrupt request indication flags. Each one of these interrupt request occurs, the bit of the INTRQ register would be set "1". The INTRQ value needs to be clear by programming after detecting the flag. In the interrupt vector of program, users know the any interrupt requests occurring by the register and do the routine corresponding of the interrupt request.

**INTRQ initial value = x000 0000**

| 0C8H         | Bit 7 | Bit 6  | Bit 5  | Bit 4 | Bit 3  | Bit 2  | Bit 1  | Bit 0  |
|--------------|-------|--------|--------|-------|--------|--------|--------|--------|
| <b>INTRQ</b> | 0     | TC1IRQ | TC0IRQ | TOIRQ | SIOIRQ | P02IRQ | P01IRQ | P00IRQ |
|              | -     | R/W    | R/W    | R/W   | R/W    | R/W    | R/W    | R/W    |

P00IRQ : External P0.0 interrupt request bit. 0 = non-request, 1 = request.

P01IRQ : External P0.1 interrupt request bit. 0 = non-request, 1 = request.

P02IRQ : External P0.2 interrupt request bit. 0 = non-request, 1 = request.

SIOIRQ : SIO interrupt request bit. 0 = non-request, 1 = request.

TOIRQ : T0 timer interrupt request control bit. 0 = non request, 1 = request.

TC0IRQ : TC0 timer interrupt request controls bit. 0 = non request, 1 = request.

TC1IRQ : TC1 timer interrupt request controls bit. 0 = non request, 1 = request.

When interrupt occurs, the related request bit of INTRQ register will be set to "1" no matter the related enable bit of INTEN register is enabled or disabled. If the related bit of INTEN = 1 and the related bit of INTRQ is also set to be "1". As the result, the system will execute the interrupt vector (ORG 8). If the related bit of INTEN = 0, moreover, the system won't execute interrupt vector even when the related bit of INTRQ is set to be "1". Users need to be cautious with the operation under multi-interrupt situation.

## INTERRUPT OPERATION DESCRIPTION

SN8P1700 provides 7 interrupts. The operation of the 7 interrupts is as following.

### GIE GLOBAL INTERRUPT OPERATION

GIE is the global interrupt control bit. All interrupts start work after the GIE = 1. It is necessary for interrupt service request. One of the interrupt requests occurs, and the program counter (PC) points to the interrupt vector (ORG 8) and the stack add 1 level.

**STKP initial value = 0xxx 1111**

| ODFH        | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3  | Bit 2  | Bit 1  | Bit 0  |
|-------------|-------|-------|-------|-------|--------|--------|--------|--------|
| <b>STKP</b> | GIE   | -     | -     | -     | STKPB3 | STKPB2 | STKPB1 | STKPB0 |
|             | R/W   | -     | -     | -     | R/W    | R/W    | R/W    | R/W    |

GIE: Global interrupt control bit. 0 = disable, 1 = enable.

➔ **Example: Set global interrupt control bit (GIE).**

```
BOBSET      FGIE          ; Enable GIE
```

➤ **Note: The GIE bit must enable and all interrupt operations work.**

## INT0 (P0.0) INTERRUPT OPERATION

The INT0 is triggered by falling edge. When the INT0 trigger occurs, the P00IRQ will be set to "1" however the P00IEN is enable or disable. If the P00IEN = 1, the trigger event will make the P00IRQ to be "1" and the system enter interrupt vector. If the P00IEN = 0, the trigger event will make the P00IRQ to be "1" but the system will not enter interrupt vector. Users need to care for the operation under multi-interrupt situation.

### ⇒ Example: INT0 interrupt request setup.

```

BOBSET      FP00IEN      ; Enable INT0 interrupt service
BOBCLR      FP00IRQ      ; Clear INT0 interrupt request flag
BOBSET      FGIE         ; Enable GIE

```

### ⇒ Example: INT0 interrupt service routine.

```

INT_SERVICE:
    ORG      8            ; Interrupt vector
    JMP      INT_SERVICE

    BOXCH    A, ACCBUF    ; BOXCH doesn't change C, Z flag
    PUSH

    BOBTS1   FP00IRQ      ; Check P00IRQ
    JMP      EXIT_INT     ; P00IRQ = 0, exit interrupt vector

    BOBCLR   FP00IRQ      ; Reset P00IRQ
    .        .            ; INT0 interrupt service routine
    .

EXIT_INT:
    POP
    BOXCH    A, ACCBUF    ; Restore ACC value.

    RETI      ; Exit interrupt vector

```

➤ **Note:** The *PUSH* and *POP* instruction only save *L,H,R,Z,Y,X,PFLAG* and *RBANK* registers but *A* register. User must save register *A* by *BOXCH* instruction when *PUSH* command is used.

## INT1 (P0.1) INTERRUPT OPERATION

The INT1 is triggered by falling edge. When the INT1 trigger occurs, the P01IRQ will be set to "1" however the P01IEN is enable or disable. If the P01IEN = 1, the trigger event will make the P01IRQ to be "1" and the system enter interrupt vector. If the P01IEN = 0, the trigger event will make the P01IRQ to be "1" but the system will not enter interrupt vector. Users need to care for the operation under multi-interrupt situation.

### ⇒ Example: INT1 interrupt request setup.

```

BOBSET      FP01IEN      ; Enable INT1 interrupt service
BOBCLR      FP01IRQ      ; Clear INT1 interrupt request flag
BOBSET      FGIE         ; Enable GIE

```

⇒ Example: INT1 interrupt service routine.

```

                ORG          8          ; Interrupt vector
INT_SERVICE:   JMP          INT_SERVICE

                BOXCH       A, ACCBUF  ; BOXCH doesn't change C, Z flag
                PUSH                               ; Push

                BOBTS1      FP01IRQ   ; Check P01IRQ
                JMP          EXIT_INT  ; P01IRQ = 0, exit interrupt vector

                BOBCLR      FP01IRQ   ; Reset P01IRQ
                .           .         ; INT1 interrupt service routine
                .           .

EXIT_INT:     POP                               ; Pop
                BOXCH       A, ACCBUF  ; Restore ACC value.

                RETI                               ; Exit interrupt vector

```

## INT2 (P0.2) INTERRUPT OPERATION

The INT2 is triggered by falling edge. When the INT2 trigger occurs, the P02IRQ will be set to “1” however the P02IEN is enable or disable. If the P02IEN = 1, the trigger event will make the P02IRQ to be “1” and the system enter interrupt vector. If the P02IEN = 0, the trigger event will make the P02IRQ to be “1” but the system will not enter interrupt vector. Users need to care for the operation under multi-interrupt situation.

⇒ Example: INT2 interrupt request setup.

```

                BOBSET      FP02IEN   ; Enable INT2 interrupt service
                BOBCLR      FP02IRQ   ; Clear INT2 interrupt request flag
                BOBSET      FGIE      ; Enable GIE

```

⇒ Example: INT2 interrupt service routine.

```

                ORG          8          ; Interrupt vector
INT_SERVICE:   JMP          INT_SERVICE

                BOXCH       A, ACCBUF  ; BOXCH doesn't change C, Z flag
                PUSH                               ; Push

                BOBTS1      FP02IRQ   ; Check P02IRQ
                JMP          EXIT_INT  ; P02IRQ = 0, exit interrupt vector

                BOBCLR      FP02IRQ   ; Reset P02IRQ
                .           .         ; INT2 interrupt service routine
                .           .

EXIT_INT:     POP                               ; Pop
                BOXCH       A, ACCBUF  ; Restore ACC value.

                RETI                               ; Exit interrupt vector

```

## 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.

```

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

BOBSET    FT0IEN    ; Enable T0 interrupt service
BOBCLR    FT0IRQ    ; Clear T0 interrupt request flag
BOBSET    FT0ENB    ; Enable T0 timer

BOBSET    FGIE      ; Enable GIE

```

### ➔ Example: T0 interrupt service routine.

```

ORG       8          ; Interrupt vector
JMP      INT_SERVICE

INT_SERVICE:

BOXCH    A, ACCBUF   ; BOXCH doesn't change C, Z flag
PUSH

BOBTS1   FT0IRQ     ; Check T0IRQ
JMP      EXIT_INT   ; T0IRQ = 0, exit interrupt vector

BOBCLR   FT0IRQ     ; Reset T0IRQ
MOV      A, #74H    ;
BOMOV    T0C, A     ; Reset T0C.
.        .          ; T0 interrupt service routine
.        .

EXIT_INT:

POP      A          ; Pop
BOXCH    A, ACCBUF   ; Restore ACC value.

RETI     ; Exit interrupt vector

```

## TC0 INTERRUPT OPERATION

When the TC0C counter occurs overflow, the TC0IRQ will be set to "1" however the TC0IEN is enable or disable. If the TC0IEN = 1, the trigger event will make the TC0IRQ to be "1" and the system enter interrupt vector. If the TC0IEN = 0, the trigger event will make the TC0IRQ to be "1" but the system will not enter interrupt vector. Users need to care for the operation under multi-interrupt situation.

### ⇒ Example: TC0 interrupt request setup.

```

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

BOBSET    FTC0IEN    ; Enable TC0 interrupt service
BOBCLR    FTC0IRQ    ; Clear TC0 interrupt request flag
BOBSET    FTC0ENB    ; Enable TC0 timer

BOBSET    FGIE       ; Enable GIE

```

### ⇒ Example: TC0 interrupt service routine.

```

ORG       8          ; Interrupt vector
JMP       INT_SERVICE
INT_SERVICE:

BOXCH     A, ACCBUF  ; BOXCH doesn't change C, Z flag
PUSH

BOBTS1    FTC0IRQ    ; Check TC0IRQ
JMP       EXIT_INT  ; TC0IRQ = 0, exit interrupt vector

BOBCLR    FTC0IRQ    ; Reset TC0IRQ
MOV       A, #74H
BOMOV     TC0C, A    ; Reset TC0C.
.         .          ; TC0 interrupt service routine
.         .

EXIT_INT:

POP
BOXCH     A, ACCBUF  ; Restore ACC value.

RETI     ; Exit interrupt vector

```



## TC1 INTERRUPT OPERATION

When the TC1C counter occurs overflow, the TC1IRQ will be set to "1" however the TC1IEN is enable or disable. If the TC1IEN = 1, the trigger event will make the TC1IRQ to be "1" and the system enter interrupt vector. If the TC1IEN = 0, the trigger event will make the TC1IRQ to be "1" but the system will not enter interrupt vector. Users need to care for the operation under multi-interrupt situation.

### ⇒ Example: TC1 interrupt request setup.

```

B0BCLR    FTC1IEN    ; Disable TC1 interrupt service
B0BCLR    FT C1ENB   ; Disable TC1 timer
MOV       A, #20H    ;
B0MOV     TC1M, A    ; Set TC1 clock = Fcpu / 64
MOV       A, #74H    ; Set TC1C initial value = 74H
B0MOV     TC1C, A    ; Set TC1 interval = 10 ms

B0BSET    FTC1IEN    ; Enable TC1 interrupt service
B0BCLR    FTC1IRQ    ; Clear TC1 interrupt request flag
B0BSET    FTC1ENB    ; Enable TC1 timer

B0BSET    FGIE       ; Enable GIE

```

### ⇒ Example: TC1 interrupt service routine.

```

INT_SERVICE:
ORG       8          ; Interrupt vector
JMP      INT_SERVICE

B0XCH    A, ACCBUF   ; B0XCH doesn't change C, Z flag
PUSH

B0BTS1   FTC1IRQ    ; Check TC1IRQ
JMP      EXIT_INT   ; TC1IRQ = 0, exit interrupt vector

B0BCLR   FTC1IRQ    ; Reset TC1IRQ
MOV      A, #74H    ;
B0MOV    TC1C, A    ; Reset TC1C.
.        .          ; TC1 interrupt service routine
.        .

EXIT_INT:
POP      A          ; Pop
B0XCH    A, ACCBUF   ; Restore ACC value.

RETI     ; Exit interrupt vector

```

## SIO INTERRUPT OPERATION

When the SIO finished transmitting, the SIOIRQ will be set to "1" however the SIOIEN is enable or disable. If the SIOIEN = 1, the trigger event will make the SIOIRQ to be "1" and the system enter interrupt vector. If the SIOIEN = 0, the trigger event will make the SIOIRQ to be "1" but the system will not enter interrupt vector. Users need to care for the operation under multi-interrupt situation.

### ➔ Example: SIO interrupt request setup.

```

BOBSET      FSIOIEN      ; Enable SIO interrupt service
BOBCLR      FSIOIRQ      ; Clear SIO interrupt request flag
BOBSET      FGIE         ; Enable GIE

```

### ➔ Example: SIO interrupt service routine.

```

                                8      ; Interrupt vector
                                INT_SERVICE
INT_SERVICE:
                                8      ; B0XCH doesn't change C, Z flag
                                A, ACCBUF
                                PUSH    ; Push
                                B0BTS1  FSIOIRQ      ; Check SIOIRQ
                                JMP      EXIT_INT    ; SIOIRQ = 0, exit interrupt vector
                                B0BCLR  FSIOIRQ      ; Reset SIOIRQ
                                .        .          ; SIO interrupt service routine
                                .        .
EXIT_INT:
                                POP      ; Pop
                                B0XCH   A, ACCBUF    ; Restore ACC value.
                                RETI     ; Exit interrupt vector

```

## MULTI-INTERRUPT OPERATION

In most conditions, the software designer uses more than one interrupt request. Processing multi-interrupt request needs to set the priority of these interrupt requests. The IRQ flags of the 7 interrupt are controlled by the interrupt event occurring. But the IRQ flag set doesn't mean the system to execute the interrupt vector. The IRQ flags can be triggered by the events without interrupt enable. Just only any the event occurs and the IRQ will be logic "1". The IRQ and its trigger event relationship is as the below table.

| <b><i>Interrupt Name</i></b> | <b><i>Trigger Event Description</i></b> |
|------------------------------|---|
| P00IRQ                       | P0.0 trigger. Falling edge.             |
| P01IRQ                       | P0.1 trigger. Falling edge.             |
| P02IRQ                       | P0.2 trigger. Falling edge.             |
| T0IRQ                        | T0C overflow.                           |
| TC0IRQ                       | TC0C overflow.                          |
| TC1IRQ                       | TC1C overflow.                          |
| SIOIRQ                       | End of SIO transmitter operating.       |

There are two things need to do for multi-interrupt. One is to make a good priority for these interrupt requests. Two is using IEN and IRQ flags to decide executing interrupt service routine or not. Users have to check interrupt control bit and interrupt request flag in interrupt vector. There is a simple routine as following.

⇒ Example: How does users check the interrupt request in multi-interrupt situation?

```

ORG          8          ; Interrupt vector

BOXCH        A, ACCBUF  ; BOXCH doesn' t change C, Z flag
PUSH         ; Push
INTP00CHK:   ; Check INT0 interrupt request
B0BTS1       FP00IEN    ; Check P00IEN
JMP          INTP01CHK  ; Jump check to next interrupt
B0BTS0       FP00IRQ    ; Check P00IRQ
JMP          INTP00     ; Jump to INT0 interrupt service routine
INTP01CHK:   ; Check INT1 interrupt request
B0BTS1       FP01IEN    ; Check P01IEN
JMP          INTP02CHK  ; Jump check to next interrupt
B0BTS0       FP01IRQ    ; Check P01IRQ
JMP          INTP01     ; Jump to INT1 interrupt service routine
INTP02CHK:   ; Check INT2 interrupt request
B0BTS1       FP02IEN    ; Check P02IEN
JMP          INTT0CHK   ; Jump check to next interrupt
B0BTS0       FP02IRQ    ; Check P02IRQ
JMP          INTP02     ; Jump to INT2 interrupt service routine
INTT0CHK:    ; Check T0 interrupt request
B0BTS1       FT0IEN     ; Check T0IEN
JMP          INTTC0CHK  ; Jump check to next interrupt
B0BTS0       FT0IRQ     ; Check T0IRQ
JMP          INTT0      ; Jump to T0 interrupt service routine
INTTC0CHK:   ; Check TC0 interrupt request
B0BTS1       FTC0IEN    ; Check TC0IEN
JMP          INTTC1CHK  ; Jump check to next interrupt
B0BTS0       FTC0IRQ    ; Check TC0IRQ
JMP          INTTC0     ; Jump to TC0 interrupt service routine
INTTC1HK:    ; Check TC1 interrupt request
B0BTS1       FTC1IEN    ; Check TC1IEN
JMP          INTSIOCHK  ; Jump check to next interrupt
B0BTS0       FTC1IRQ    ; Check TC1IRQ
JMP          INTTC1     ; Jump to TC1 interrupt service routine
INTSIOCHK:   ; Check SIO interrupt request
B0BTS1       FSIOIEN    ; Check SIOIEN
JMP          INT_EXIT   ; Jump to exit of IRQ
B0BTS0       FSIOIRQ    ; Check SIOIRQ
JMP          INTSIO     ; Jump to SIO interrupt service routine
INT_EXIT:    ; Pop
POP          ; Restore ACC value.
BOXCH        A, ACCBUF
RETI         ; Exit interrupt vector

```

# 10 SERIAL INPUT/OUTPUT TRANSCEIVER (SIO)

## OVERVIEW

The SN8P1700 provides an 8-bit SIO interface circuit with clock rate selection. The SIOM register can control SIO operating function, such as: transmit/receive, clock rate, transfer edge and starting this circuit. This SIO circuit will TX or RX 8-bit data automatically by setting SENB and START bits in SIOM register. The SIOB is an 8-bit buffer, which is designed to store transfer data. SIOC and SIOR are designed to generate SIO's clock source with auto-reload function. The 3-bit I/O counter can monitor the operation of SIO and announce an interrupt request after transmitting/receiving 8 bits data. After transferring 8-bit data, this circuit will be disabled automatically and re-transfer data by programming SIOM register.

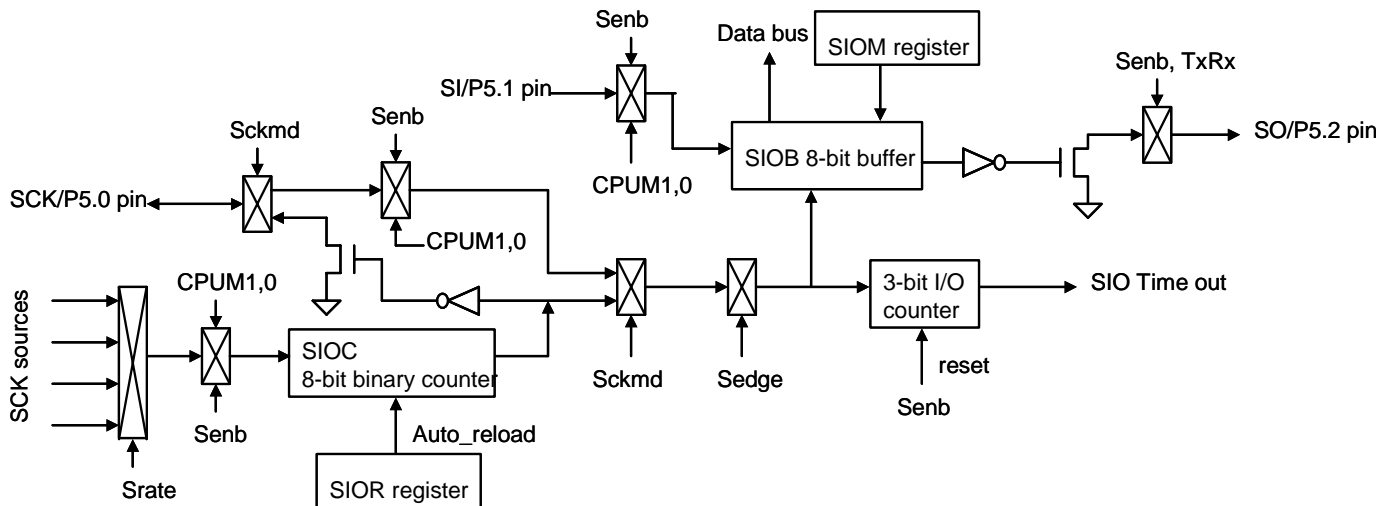
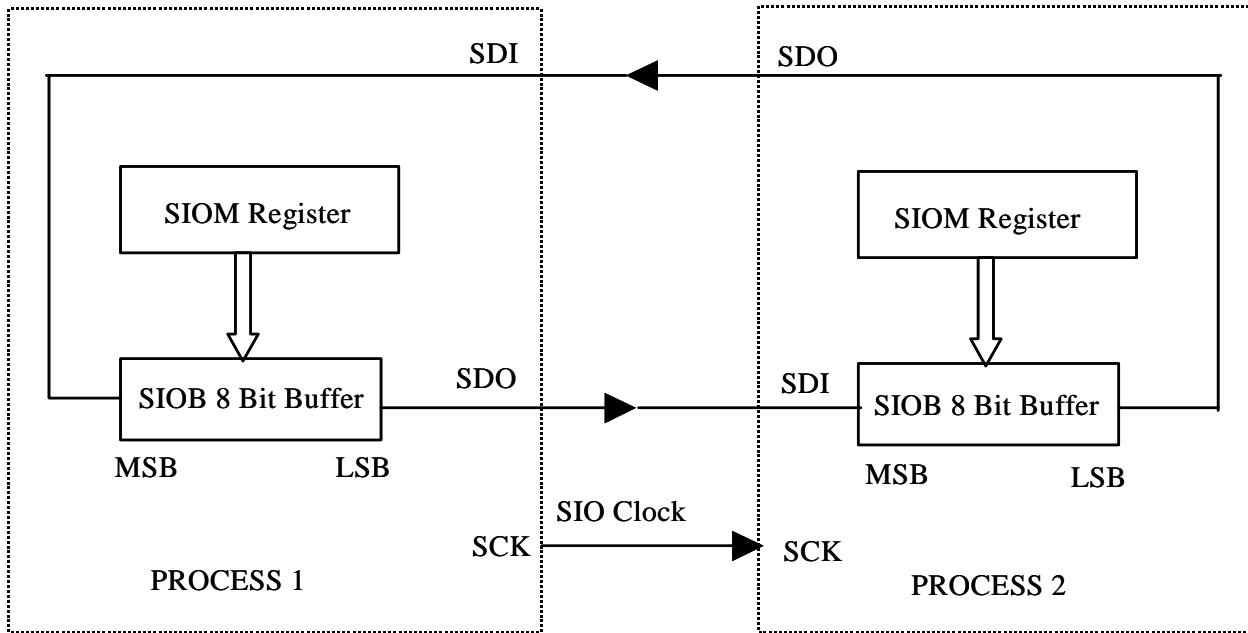


Figure 10-1. SIO Interface Circuit Diagram

Figure 9-2 shows a typical transfer between two microcontrollers. Process 1 sends SCK for initial the data transfer. Both processors must work in the same clock edge direction, then both controllers would send and receive data at the same time.



**Figure 10-2. SIO Data Transfer Diagram**

## SIOM MODE REGISTER

**SIOM initial value = 0000 x000**

| 0B4H        | Bit 7 | Bit 6 | Bit 5  | Bit 4  | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|-------------|-------|-------|--------|--------|-------|-------|-------|-------|
| <b>SIOM</b> | SENB  | START | SRATE1 | SRATE0 | 0     | SCKMD | SEdge | TXRX  |
|             | R/W   | R/W   | R/W    | R/W    | -     | R/W   | R/W   | R/W   |

SENB: SIO function control bit. 0 = disable (P5.0~P5.2 is general purpose port), 1 = enable (P5.0~P5.2 is SIO pins).

START: SIO progress control bit. 0 = End of transfer, 1 = progressing.

SRATE1, 0: SIO's transfer rate select bit. 00 = fcpu/32, 01 = fcpu/16, 10 = fcpu/8, 11 = fcpu/4.

(Note: These 2-bits are workless when SCKMD=1)

SCKMD: SIO's clock mode select bit. 0 = internal, 1 = external mode.

SEdge: SIO's transfer clock edge select bit. 0 = falling edge, 1 = raising edge.

TXRX: SIO's transfer direction select bit. 0 = receiver only , 1 = transmitter/receiver full duplex.

- **Note 1: If SCKMD=1 for external clock, the SIO is in SLAVE mode.  
If SCKMD=0 for internal clock, the SIO is in MASTER mode.**
- **Note 2: Don't set SENB and START bits in the same time. That makes the SIO function error.**

Because SIO function is shared with Port5 for P5.0 as SCK, P5.1 as SI and P5.2 as SO

The following table shown the Port5[2:0] I/O mode behavior and setting when SIO function enable and disable

| SENB=1 (SIO Function Enable)  |   |   |
|-------------------------------|---|---|
| P5.0/SCK                      | (SCKMD=1)<br>SIO source = External clock                                    | P5.0 will change to Input mode automatically, no matter what P5M setting  |
|                               | (SCKMD=0)<br>SIO source = Internal clock                                    | P5.0 will change to Output mode automatically, no matter what P5M setting |
| P5.1/SI                       | P5.1 must be set as Input mode in P5M ,or the SIO function will be abnormal |   |
| P5.2/SO                       | (TXRX=1)<br>SIO = Transmitter/Receiver                                      | P5.2 will change to Output mode automatically, no matter what P5M setting |
|                               | (TXRX=0)<br>SIO = Receiver only   | P5.2 will change to Input mode automatically, no matter what P5M setting  |
| SENB=0 (SIO Function Disable) |   |   |
| P5.0/P5.1/P5.2                | Port5[2:0] I/O mode are fully controlled by P5M when SIO function Disable   |   |

## SI0B DATA BUFFER

**SI0B initial value = 0000 0000**

| 0B6H        | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|-------------|-------|-------|-------|-------|-------|-------|-------|-------|
| <b>SI0B</b> | X     | X     | X     | X     | X     | X     | X     | X     |
|             | R/W   | R/W   | R/W   | R/W   | R/W   | R/W   | R/W   | R/W   |

SI0B is the SIO data buffer register. It stores serial I/O transmit and receive data.

## SI0R REGISTER DESCRIPTION

**SI0R initial value = 0000 0000**

| 0B5H        | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|-------------|-------|-------|-------|-------|-------|-------|-------|-------|
| <b>SI0R</b> | X     | X     | X     | X     | X     | X     | X     | X     |
|             | W     | W     | W     | W     | W     | W     | W     | W     |

The SI0R is designed for the SIO counter to reload the counted value when end of counting. It is like a post-scaler of SIO clock source and let SIO has more flexible to setting SCK range. Users can set the SI0R value to setup SIO transfer time. To setup SI0R value equation to desire transfer time is as following.

$$\text{SCK frequency} = \text{SIO rate} / (256 - \text{SI0R})$$

$$\text{SI0R} = 256 - (1 / (\text{SCK frequency}) * \text{SIO rate} / 2)$$

⇒ Example: Setup the SIO clock to be 5KHz. Fosc = 3.58MHz. SIO's rate = Fcpu = Fosc/4.

$$\begin{aligned} \text{SI0R} &= 256 - (1/(5\text{KHz}) * 3.58\text{MHz}/4) \\ &= 256 - 89 \\ &= 167 \end{aligned}$$

## SIO MASTER OPERATING DESCRIPTION

Under master-transmitter situation, the SCK has two directions as following.

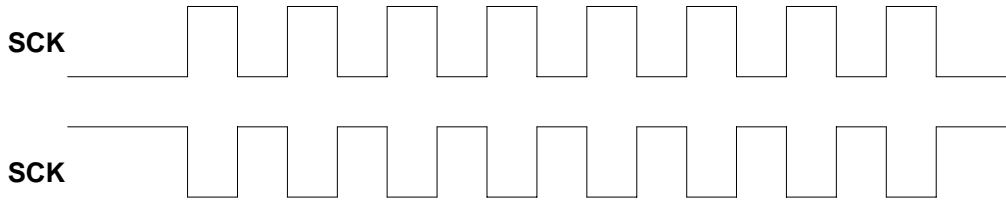


Figure 10-3. The Two SCK Directions of SIO Master Operation

### RISING EDGE TRANSMITTER/RECEIVER MODE

#### ⇒ Example: Master Tx/Rx rising edge

```

MOV          A, TXDATA          ; Load transmitted data into SIOB register.
B0MOV       SIOB, A
MOV         A, #0FFH           ; Set SIO clock with auto-reload function.
B0MOV       SIOR, A
MOV         A, #10000011B      ; Setup SIOM and enable SIO function. Rising edge.
B0MOV       SIOM, A
B0BSET      FSTART             ; Start transfer and receiving SIO data.

CHK_END:
B0BTS0      FSTART             ; Wait the end of SIO operation.
JMP         CHK_END
B0MOV       A, SIOB            ; Save SIOB data into RXDATA buffer.
MOV         RXDATA, A
  
```

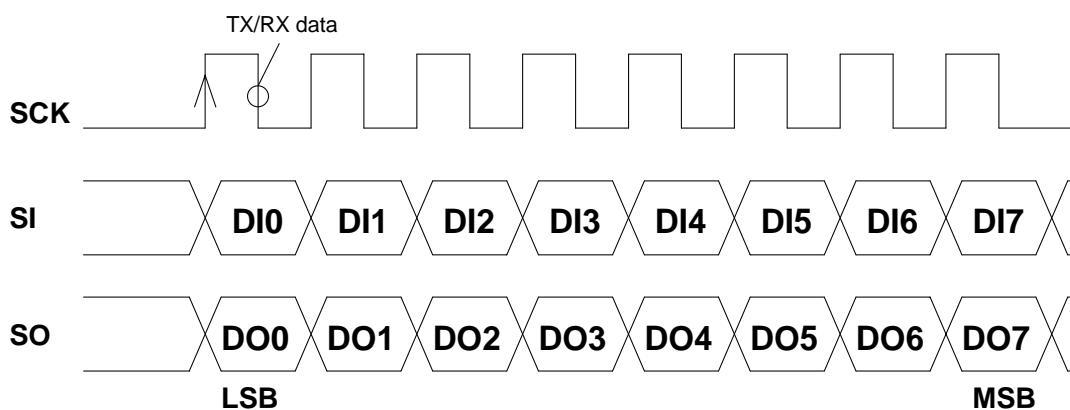


Figure 10-4. The Rising Edge Timing Diagram of Master Transfer and Receiving Operation



## FALLING EDGE TRANSMITTER/RECEIVER MODE

### ⇒ Example: Master Tx/Rx falling edge

```

MOV          A,TXDATA          ; Load transmitted data into SIOB register.
B0MOV       SIOB,A
MOV         A,#0FFH           ; Set SIO clock with auto-reload function.
B0MOV       SIOR,A
MOV         A,#10000001B      ; Setup SIOM and enable SIO function. Falling edge.
B0MOV       SIOM,A
B0BSET      FSTART           ; Start transfer and receiving SIO data.
CHK_END:
B0BTS0     FSTART           ; Wait the end of SIO operation.
JMP        CHK_END
B0MOV      A,SIOB           ; Save SIOB data into RXDATA buffer.
MOV        RXDATA,A

```

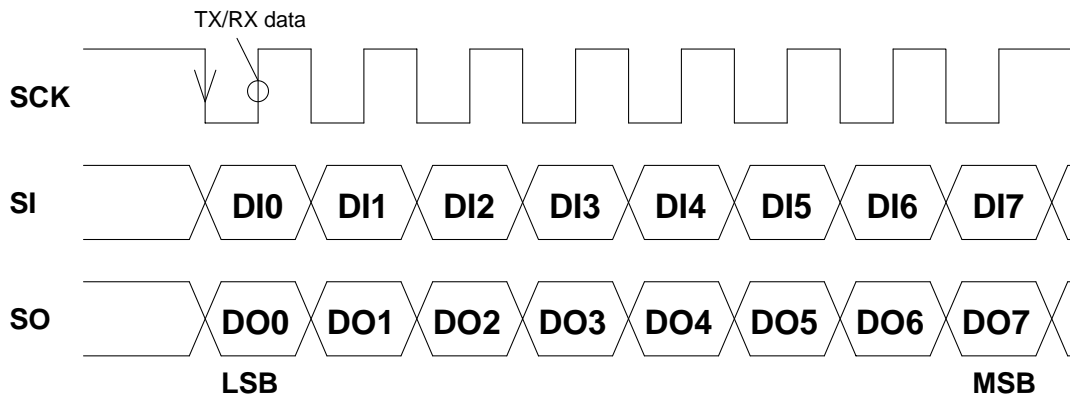


Figure 10-5. The Falling Edge Timing Diagram of Master Transfer and Receiving Operation

## RISING EDGE RECEIVER MODE

### ⇒ Example: Master Rx rising edge

```

MOV          A,#0FFH           ; Set SIO clock with auto-reload function.
B0MOV       SIOR,A
MOV          A,#10000010B      ; Setup SIOM and enable SIO function. Rising edge.
B0MOV       SIOM,A
B0BSET      FSTART            ; Start receiving SIO data.
CHK_END:
B0BTS0     FSTART            ; Wait the end of SIO operation.
JMP        CHK_END
B0MOV       A,SIOB            ; Save SIOB data into RXDATA buffer.
MOV         RXDATA,A

```

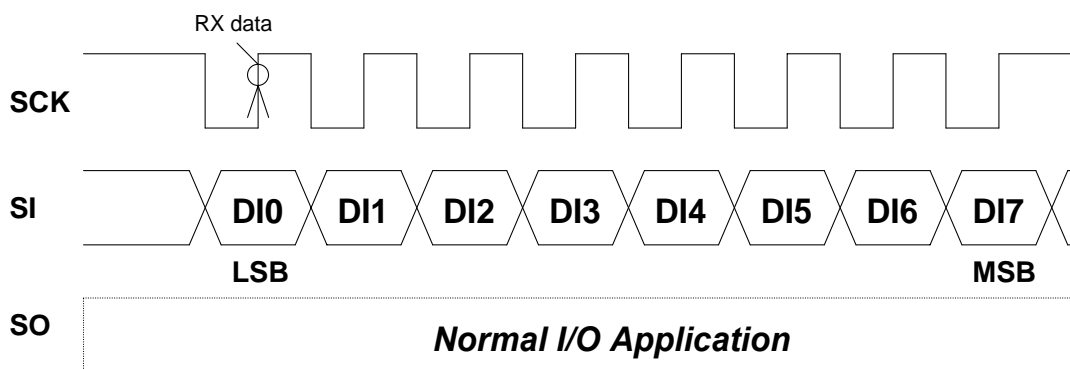


Figure 10-6. The Rising Edge Timing Diagram of Master Receiving Operation

## FALLING EDGE RECEIVER MODE

### ⇒ Example: Master Rx falling edge

```

MOV          A,#0FFH           ; Set SIO clock with auto-reload function.
B0MOV       SIOR,A
MOV          A,#10000000B      ; Setup SIOM and enable SIO function. Falling edge.
B0MOV       SIOM,A
B0BSET      FSTART            ; Start receiving SIO data.
CHK_END:
B0BTS0     FSTART            ; Wait the end of SIO operation.
JMP        CHK_END
B0MOV       A,SIOB            ; Save SIOB data into RXDATA buffer.
MOV         RXDATA,A

```

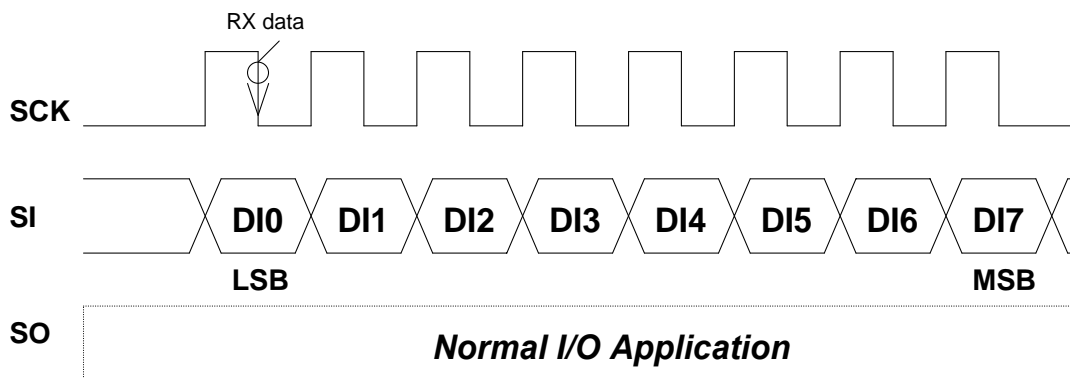
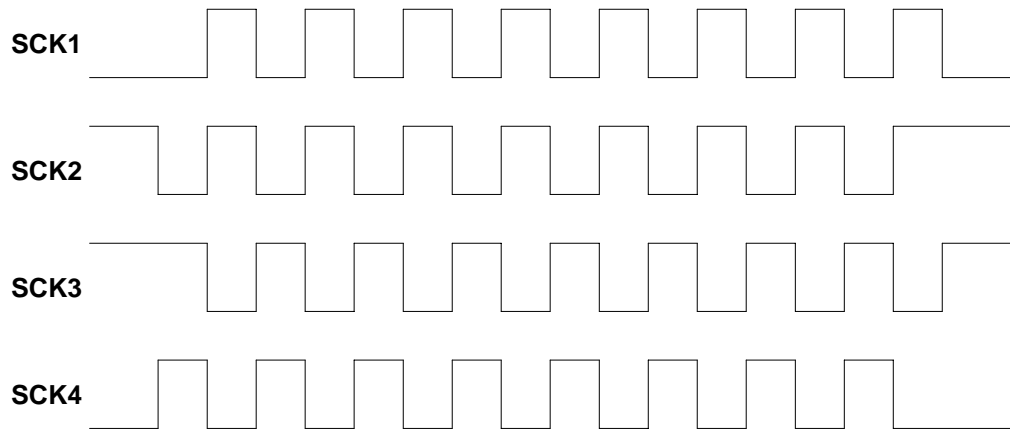


Figure 10-7. The Falling Edge Timing Diagram of Master Receiving Operation

## SIO SLAVE OPERATING DESCRIPTION

Under slave-receiver situation, the SCK has four phases as following.



**Figure 10-8. The Four Phases SCK clock of SIO Slave Operation.**

## RISING EDGE TRANSMITTER/RECEIVER MODE

⇒ Example: Slave Tx/Rx rising edge

```

MOV          A,TXDATA          ; Load transfer data into SIOB register.
B0MOV       SIOB,A
MOV         A,# 10000111B      ; Setup SIOM and enable SIO function. Rising edge.
B0MOV       SIOM,A
B0BSET      FSTART            ; Start transfer and receiving SIO data.
CHK_END:
B0BTS0     FSTART            ; Wait the end of SIO operation.
JMP        CHK_END
B0MOV      A,SIOB            ; Save SIOB data into RXDATA buffer.
MOV        RXDATA,A
    
```

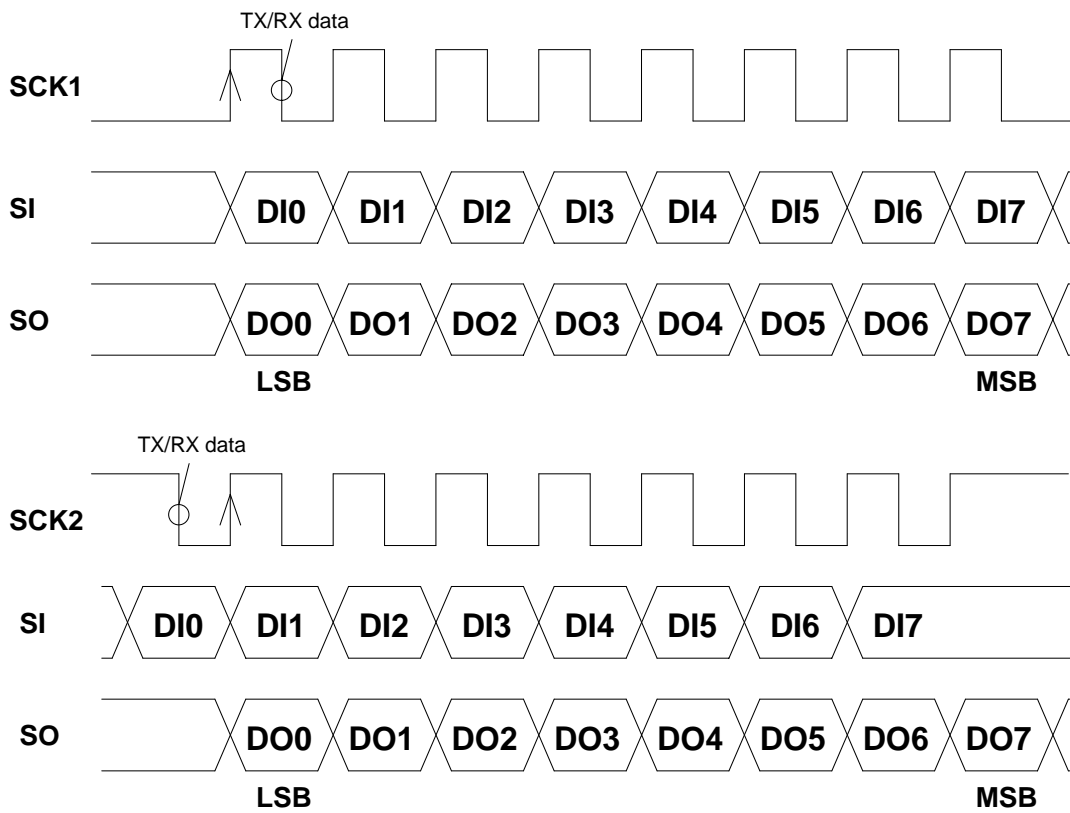


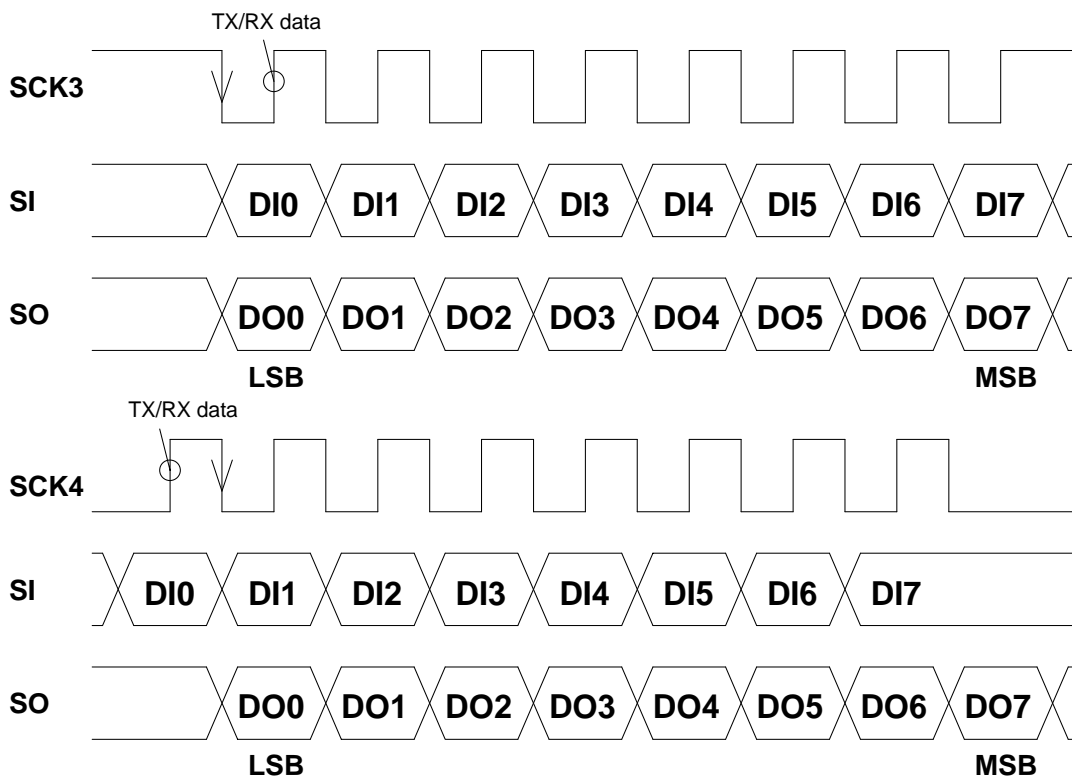
Figure 10-9. The Rising Edge Timing Diagram of Slave Transfer and Receiving Operation

## FALLING EDGE TRANSMITTER/RECEIVER MODE

⇒ **Example: Slave Tx/Rx falling edge**

```

MOV          A,TXDATA          ; Load transfer data into SIOB register.
B0MOV       SIOB,A
MOV          A,# 10000101B     ; Setup SIOM and enable SIO function. Falling edge.
B0MOV       SIOM,A
B0BSET      FSTART            ; Start transfer and receiving SIO data.
CHK_END:
B0BTS0     FSTART            ; Wait the end of SIO operation.
JMP        CHK_END
B0MOV      A,SIOB            ; Save SIOB data into RXDATA buffer.
MOV        RXDATA,A
    
```



**Figure 10-10. The Falling Edge Timing Diagram of Slave Transfer and Receiving Operation**

## RISING EDGE RECEIVER MODE

### ⇒ Example: Slave Rx rising edge

```

MOV          A,# 10000110B      ; Setup SIOM and enable SIO function. Rising edge.
B0MOV       SIOM,A
B0BSET      FSTART             ; Start receiving SIO data.
CHK_END:
B0BTS0     FSTART             ; Wait the end of SIO operation.
JMP        CHK_END
B0MOV       A,SIOB             ; Save SIOB data into RXDATA buffer.
MOV        RXDATA,A

```

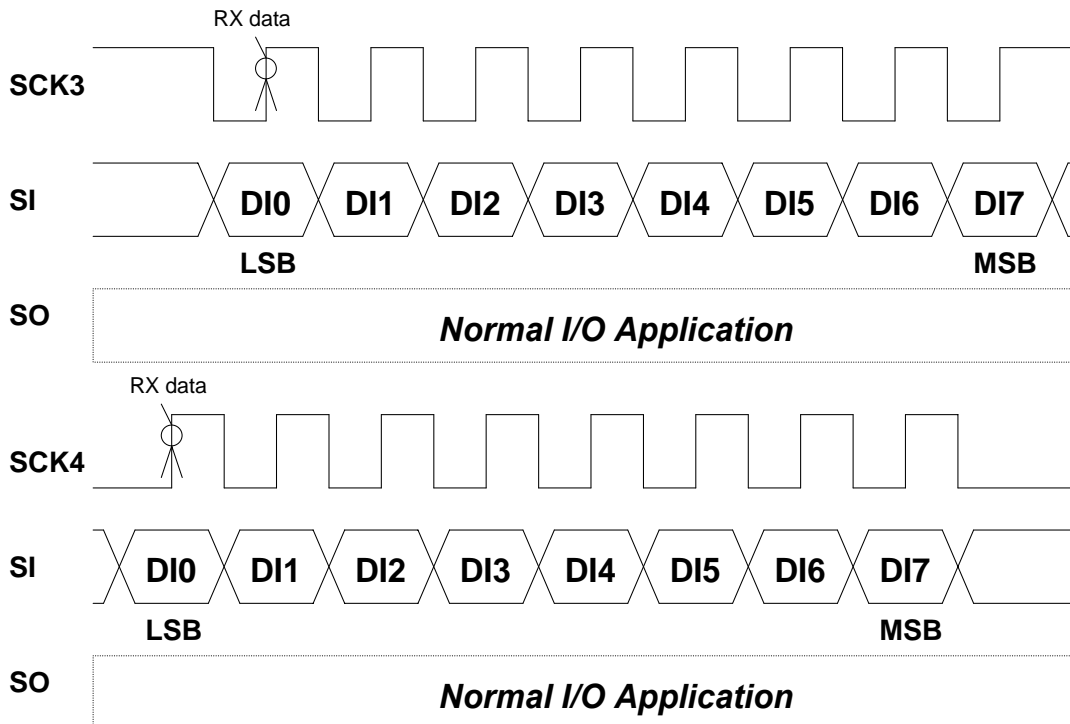


Figure 10-11. The Rising Edge Timing Diagram of Slave Receiving Operation

## FALLING EDGE RECEIVER MODE

### ⇒ Example: Slave Rx falling edge

```

MOV          A,# 10000100B      ; Setup SIOM and enable SIO function. Falling edge.
B0MOV       SIOM,A
B0BSET      FSTART              ; Start receiving SIO data.
CHK_END:
B0BTS0     FSTART              ; Wait the end of SIO operation.
JMP        CHK_END
B0MOV       A,SIOB              ; Save SIOB data into RXDATA buffer.
MOV        RXDATA,A

```

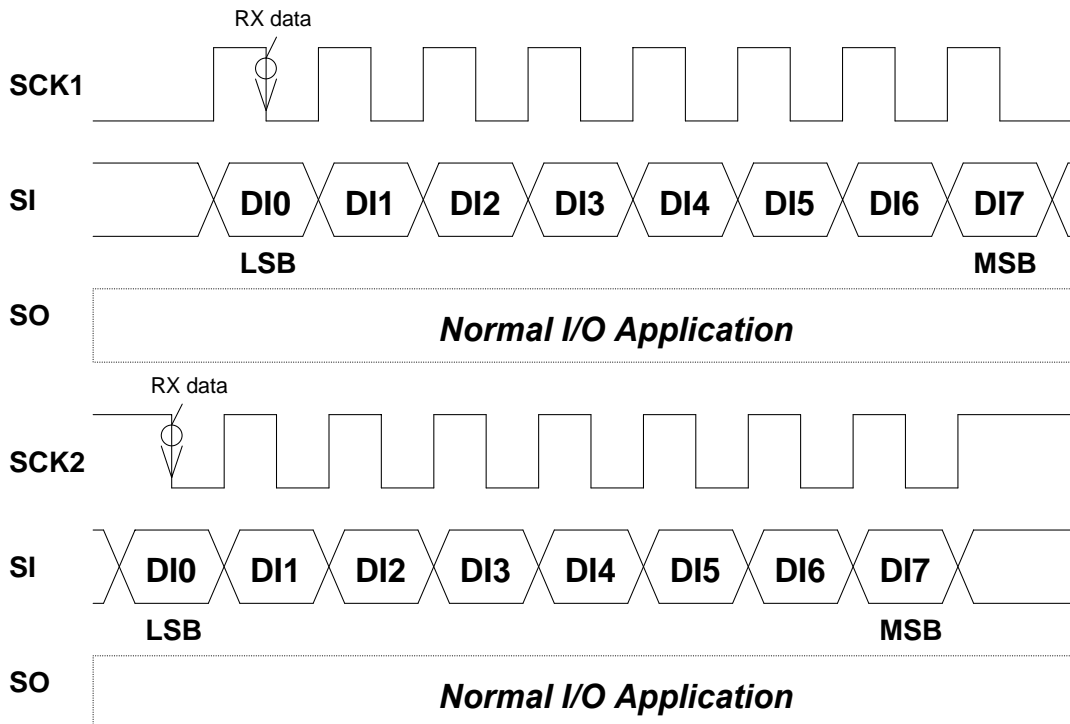


Figure 10-12. The Falling Edge Timing Diagram of Slave Receiving Operation



## SIO INTERRUPT OPERATION DESCRIPTION

The SIO provides an interrupt function. Users can process SIO data after the SIO interrupt request occurring. There is an example for the application as following.

### ➔ Example: SIO interrupt demo routine.

Main:

```

MOV      A,# 10000100B      ; Setup SIOM and enable SIO function. Falling edge.
B0MOV   SIOM,A
B0BSET  FSTART              ; Start transfer SIO data.
.
.
JMP     MAIN

```

```

ORG      8                  ; Interrupt vector

```

```

BOXCH   A, ACCBUF
PUSH

```

```

B0BTS1  FSIOIRQ
JMP     INT_EXIT
B0MOV   A,SIOB              ; Save SIOB data into RXDATA buffer.
MOV     RXDATA,A
B0BCLR  FSIOIRQ            ; Clear SIO interrupt request flag.

```

INT\_EXIT:

```

POP
BOXCH   A, ACCBUF

```

# 11 I/O PORT

## OVERVIEW

The SN8P1700 provides up to 5 ports for users' application, consisting of one input only port (P0), four I/O ports (P1, P2, P4, P5). The direction of I/O port is selected by PnM register and a macro @SET\_PUR is defined for user setting pull-up register. After the system resets, all ports work as input function without pull-up resistors.

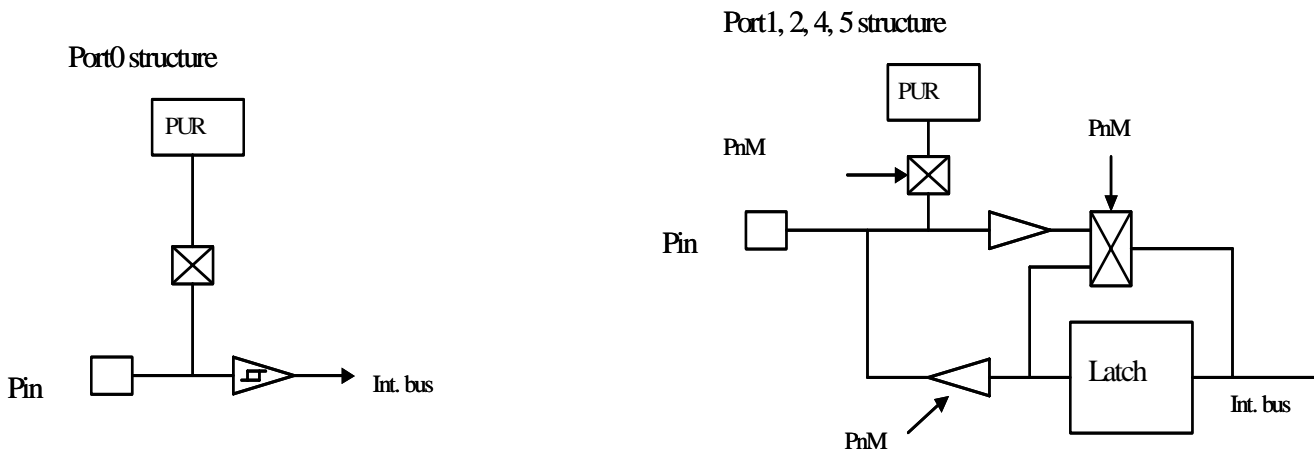


Figure 11-1. The I/O Port Block Diagram

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

**I/O PORT FUNCTION TABLE**

| Port/Pin  | I/O | Function Description                  | Remark                       |
|-----------|-----|---------------------------------------|------------------------------|
| P0.0~P0.2 | I   | General-purpose input function        |                              |
|           |     | External interrupt (INT0~INT2)        |                              |
|           |     | Wakeup for power down mode            |                              |
| P1.0~P1.5 | I/O | General-purpose input/output function |                              |
|           |     | Wakeup for power down mode            |                              |
| P2.0~P2.7 | I/O | General-purpose input/output function |                              |
| P4.0~P4.7 | I/O | General-purpose input/output function |                              |
|           |     | ADC analog signal input               |                              |
| P5.0      | I/O | General-purpose input/output function |                              |
|           |     | SIO clock pin.                        |                              |
| P5.1      | I/O | General-purpose input/output function |                              |
|           | I   | SIO data input pin.                   | <b>P5M.1 must be set "0"</b> |
| P5.2      | I/O | General-purpose input/output function |                              |
|           | O   | SIO data output pin.                  | <b>P5M.1 must be set "1"</b> |
| P5.3~P5.7 | I/O | General-purpose input/output function |                              |

Table 11-1. I/O Function Table

## PULL-UP RESISTERS

SN8P1700 series chips built-in pull-up resistors in port 0, port 1, port 4 and port 5. For MASK type compatible issues, SONiX 8-bit MCU assembler provide a @SET\_PUR macro to control pull-up resistors. @SET\_PUR macro only allows enable or disable pull-up resistors as a whole port.

### SN8P1702 / SN8P1704:

#### @SET\_PUR VAL

| I/O Port        | Port 7    | Port 6    | Port 5 | Port 4 | Port 3    | Port 2    | Port 1 | Port 0 |
|-----------------|-----------|-----------|--------|--------|-----------|-----------|--------|--------|
| VAL             | Bit 7     | Bit 6     | Bit 5  | Bit 4  | Bit 3     | Bit 2     | Bit 1  | Bit 0  |
| Disable Pull-up | Fixed "0" | Fixed "0" | 0      | 0      | Fixed "0" | Fixed "0" | 0      | 0      |
| Enable Pull-up  |           |           | 1      | 1      |           |           | 1      | 1      |

### SN8P1706 / SN8P1707 / SN8P1708:

#### @SET\_PUR VAL

| I/O Port        | Port 7    | Port 6    | Port 5 | Port 4 | Port 3    | Port 2 | Port 1 | Port 0 |
|-----------------|-----------|-----------|--------|--------|-----------|--------|--------|--------|
| VAL             | Bit 7     | Bit 6     | Bit 5  | Bit 4  | Bit 3     | Bit 2  | Bit 1  | Bit 0  |
| Disable Pull-up | Fixed "0" | Fixed "0" | 0      | 0      | Fixed "0" | 0      | 0      | 0      |
| Enable Pull-up  |           |           | 1      | 1      |           | 1      | 1      | 1      |

#### ➔ Example 1: Enable port 0 and port 1 pull-up resistors and disable others

##### CHIP SN8P1708

```

        ORG 0x10
Main:
        .
        .
        @SET_PUR      0x03                ; Enable port 0 and port 1 pull-up resistors

```

#### ➔ Example 2: Enable all pull-up resistors

##### CHIP SN8P1708

```

        ORG 0x10
Main:
        .
        .
        @SET_PUR      0x37                ; Enable port 0, port 1, port 4 and port 5 pull-up resistors

```

#### Note:

- Enable on-chip pull-up resistors of port 0 and port 1 to avoid unpredicted wakeup in sleep mode.
- SN8P1704 and SN8P1702 must call @SET\_PUR at least one time to avoid sleep mode fail.

## I/O PORT MODE

The port direction is programmed by PnM register. Port 0 is always input mode. Port 1,2,4 and 5 can select input or output direction.

**P1M initial value = xx00 0000**

| 0C1H       | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|------------|-------|-------|-------|-------|-------|-------|-------|-------|
| <b>P1M</b> | 0     | 0     | P15M  | P14M  | P13M  | P12M  | P11M  | P10M  |
|            | -     | -     | R/W   | R/W   | R/W   | R/W   | R/W   | R/W   |

P10M~P15M: P1.0~P1.5 I/O direction control bit. 0 = input mode, 1 = output mode.

**P2M initial value = 0000 0000**

| 0C2H       | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|------------|-------|-------|-------|-------|-------|-------|-------|-------|
| <b>P2M</b> | P27M  | P26M  | P25M  | P24M  | P23M  | P22M  | P21M  | P20M  |
|            | R/W   | R/W   | R/W   | R/W   | R/W   | R/W   | R/W   | R/W   |

P20M~P27M: P2.0~P2.7 I/O direction control bit. 0 = input mode, 1 = output mode.

**P4M initial value = 0000 0000**

| 0C4H       | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|------------|-------|-------|-------|-------|-------|-------|-------|-------|
| <b>P4M</b> | P47M  | P46M  | P45M  | P44M  | P43M  | P42M  | P41M  | P40M  |
|            | R/W   | R/W   | R/W   | R/W   | R/W   | R/W   | R/W   | R/W   |

P40M~P47M: P4.0~P4.7 I/O direction control bit. 0 = input mode, 1 = output mode.

**P5M initial value = 0000 0000**

| 0C5H       | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|------------|-------|-------|-------|-------|-------|-------|-------|-------|
| <b>P5M</b> | P57M  | P56M  | P55M  | P54M  | P53M  | P52M  | P51M  | P50M  |
|            | R/W   | R/W   | R/W   | R/W   | R/W   | R/W   | R/W   | R/W   |

P50M~P57M: P5.0~P5.7 I/O direction control bit. 0 = input mode, 1 = output mode.

The each bit of PnM is set to "0", the I/O pin is input mode. The each bit of PnM is set to "1", the I/O pin is output mode. Input mode is with pull-up resistor controlled by setting @SET\_UP macro. The output mode disables the pull-up resistors no matter pull-up resistors is set or not.

- **The PnM registers are read/write bi-direction registers. Users can program them by bit control instructions (B0BSET, B0BCLR).**

## ⇒ Example: I/O mode selecting.

```
CLR      P1M      ; Set all ports to be input mode.  
CLR      P2M  
CLR      P4M  
CLR      P5M
```

```
MOV      A, #0FFH ; Set all ports to be output mode.  
B0MOV   P1M, A  
B0MOV   P2M, A  
B0MOV   P4M, A  
B0MOV   P5M, A
```

```
B0BCLR  P1M.5    ; Set P1.5 to be input mode.
```

```
B0BSET  P1M.5    ; Set P1.5 to be output mode.
```

## I/O PORT DATA REGISTER

**P0 initial value = xxxx x000**

| 0D0H      | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|-----------|-------|-------|-------|-------|-------|-------|-------|-------|
| <b>P0</b> | -     | -     | -     | -     | -     | P02   | P01   | P00   |
|           | -     | -     | -     | -     | -     | R     | R     | R     |

**P1 initial value = xx00 0000**

| 0D1H      | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|-----------|-------|-------|-------|-------|-------|-------|-------|-------|
| <b>P1</b> | -     | -     | P15   | P14   | P13   | P12   | P11   | P10   |
|           | -     | -     | R/W   | R/W   | R/W   | R/W   | R/W   | R/W   |

**P2 initial value = 0000 0000**

| 0D2H      | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|-----------|-------|-------|-------|-------|-------|-------|-------|-------|
| <b>P2</b> | P27   | P26   | P25   | P24   | P23   | P22   | P21   | P20   |
|           | R/W   | R/W   | R/W   | R/W   | R/W   | R/W   | R/W   | R/W   |

**P4 initial value = 0000 0000**

| 0D4H      | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|-----------|-------|-------|-------|-------|-------|-------|-------|-------|
| <b>P4</b> | P47   | P46   | P45   | P44   | P43   | P42   | P41   | P40   |
|           | R/W   | R/W   | R/W   | R/W   | R/W   | R/W   | R/W   | R/W   |

**P5 initial value = 0000 0000**

| 0D5H      | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|-----------|-------|-------|-------|-------|-------|-------|-------|-------|
| <b>P5</b> | P57   | P56   | P55   | P54   | P53   | P52   | P51   | P50   |
|           | R/W   | R/W   | R/W   | R/W   | R/W   | R/W   | R/W   | R/W   |

⇒ **Example: Read data from input port.**

```

B0MOV      A, P0           ; Read data from Port 0
B0MOV      A, P1           ; Read data from Port 1
B0MOV      A, P2           ; Read data from Port 2
B0MOV      A, P4           ; Read data from Port 4
B0MOV      A, P5           ; Read data from Port 5
    
```

⇒ **Example: Write data to output port.**

```

MOV        A, #55H        ; Write data 55H to Port 1, Port2, Port 4, Port 5
B0MOV      P1, A
B0MOV      P2, A
B0MOV      P4, A
B0MOV      P5, A
    
```

➔ **Example: Write one bit data to output port.**

```
BOBSET      P1.3      ; Set P1.3 and P4.0 to be "1".
BOBSET      P4.0
BOBCLR      P2.3      ; Set P2.3 and P5.5 to be "0".
BOBCLR      P5.5
```

➔ **Example: Port bit test.**

```
BOBTS1      P0.0      ; Bit test 1 for P0.0
BOBTS0      P1.5      ; Bit test 0 for P1.5
```



# 12 8-CHANNEL ANALOG TO DIGITAL CONVERTER

## OVERVIEW

This analog to digital converter of SN8P1700 has 8-input sources with up to 4096-step resolution to transfer analog signal into 12-bits digital data. The sequence of ADC operation is to select input source (AIN0 ~ AIN7) at first, then set GCHS and ADS bit to "1" to start conversion. When the conversion is complete, the ADC circuit will set EOC bit to "1" and final value output in ADB register. This ADC circuit can select between 8-bit and 12-bit resolution operation by programming ADLEN bit in ADR register.

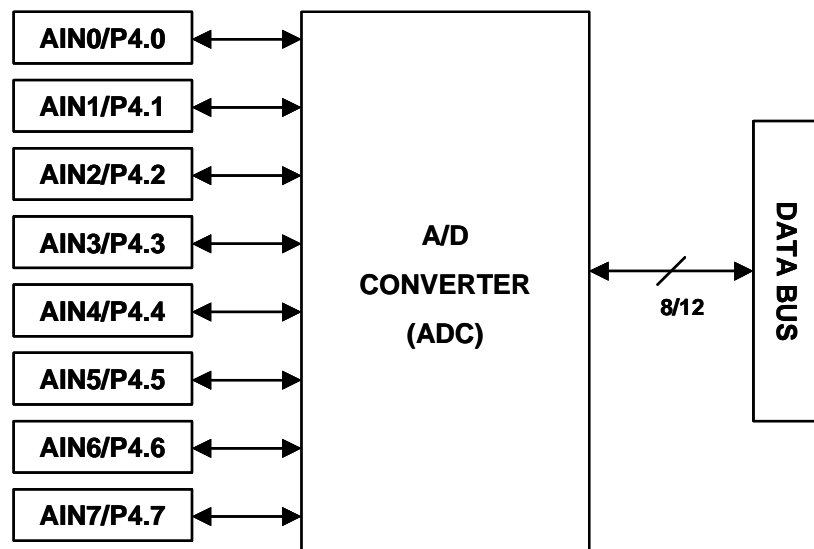


Figure 12-1. AD Converter Function Diagram

- **Note:** For 8-bit resolution the conversion time is 12 steps.  
For 12-bit resolution the conversion time is 16 steps.
- **Note:** The analog input level must be between the AVREFH and AVSS.
- **Note:** The AVREFH level must be between the AVDD and AVSS.

## ADM REGISTER

**ADM initial value = 0000 x000**

| 0B1H       | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|------------|-------|-------|-------|-------|-------|-------|-------|-------|
| <b>ADM</b> | ADENB | ADS   | EOC   | GCHS  | -     | CHS2  | CHS1  | CHS0  |
|            | R/W   | R/W   | R/W   | R/W   | -     | R/W   | R/W   | R/W   |

CHS2, 1, 0: ADC input channels select bit. 000 = AIN0, 001 = AIN1, 010 = AIN2, 011 = AIN3, .. , 111 = AIN7.

GCHS: Global channel select bit. 0 = To disable AIN channel, 1 = To enable AIN channel.

EOC: ADC status bit. 0 = Progressing, 1 = End of converting and reset ADENB bit.

ADS: ADC start bit. 0 = stop, 1 = starting.

ADENB: ADC control bit. 0 = disable, 1 = enable.

## ADR REGISTERS

**ADR initial value = x00x 0000**

| 0B3H       | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|------------|-------|-------|-------|-------|-------|-------|-------|-------|
| <b>ADR</b> | -     | ADCKS | ADLEN | 0     | ADB3  | ADB2  | ADB1  | ADB0  |
|            | -     | R/W   | R/W   | -     | R     | R     | R     | R     |

ADBn: ADC data buffer. ADB11~ADB4 bits for 8-bit ADC. ADB11~ADB0 bits for 12-bit ADC.

ADLEN: ADC's resolution select bits. 0 = 8-bit, 1 = 12-bit.

ADCKS: ADC's clock source select bit.

| ADCKS | ADC clock source | Note                                       |
|-------|------------------|--|
| 0     | Fcpu/4           | Both validate in Normal mode and Slow mode |
| 1     | Fhosc            | Only validate in Normal mode               |

## ADB REGISTERS

**ADB initial value = xxxx xxxx**

| 0B2H       | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|------------|-------|-------|-------|-------|-------|-------|-------|-------|
| <b>ADB</b> | ADB11 | ADB10 | ADB9  | ADB8  | ADB7  | ADB6  | ADB5  | ADB4  |
|            | R     | R     | R     | R     | R     | R     | R     | R     |

ADB is ADC data buffer to store AD converter result. The ADB is only 8-bit register including bit 4~bit11 ADC data. To combine ADB register and the low-nibble of ADR will get full 12-bit ADC data buffer. The ADC buffer is a read-only register. In 8-bit ADC mode, the ADC data is stored in ADB register. In 12-bit ADC mode, the ADC data is stored in ADB and ADR registers.

**The AIN's input voltage v.s. ADB's output data**

| AIN n            | ADB11 | ADB10 | ADB9 | ADB8 | ADB7 | ADB6 | ADB5 | ADB4 | ADB3 | ADB2 | ADB1 | ADB0 |
|------------------|-------|-------|------|------|------|------|------|------|------|------|------|------|
| 0/4096*AVREFH    | 0     | 0     | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    |
| 1/4096*AVREFH    | 0     | 0     | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 1    |
| .                | .     | .     | .    | .    | .    | .    | .    | .    | .    | .    | .    | .    |
| .                | .     | .     | .    | .    | .    | .    | .    | .    | .    | .    | .    | .    |
| .                | .     | .     | .    | .    | .    | .    | .    | .    | .    | .    | .    | .    |
| 4094/4096*AVREFH | 1     | 1     | 1    | 1    | 1    | 1    | 1    | 1    | 1    | 1    | 1    | 0    |
| 4095/4096*AVREFH | 1     | 1     | 1    | 1    | 1    | 1    | 1    | 1    | 1    | 1    | 1    | 1    |

For different applications, users may need more than 8-bit resolution but less than 12-bit ADC converter. To process the ADB and ADR data can make the job well. First, the AD resolution must be set 12-bit mode and then to execute ADC converter routine. Then delete the LSB of ADC data and get the new resolution result. The table is as following.

| ADC Resolution | ADB   |       |      |      |      |      |      |      | ADR  |      |      |      |
|----------------|-------|-------|------|------|------|------|------|------|------|------|------|------|
|                | ADB11 | ADB10 | ADB9 | ADB8 | ADB7 | ADB6 | ADB5 | ADB4 | ADB3 | ADB2 | ADB1 | ADB0 |
| 8-bit          | O     | O     | O    | O    | O    | O    | O    | O    | x    | x    | x    | x    |
| 9-bit          | O     | O     | O    | O    | O    | O    | O    | O    | O    | x    | x    | x    |
| 10-bit         | O     | O     | O    | O    | O    | O    | O    | O    | O    | O    | x    | x    |
| 11-bit         | O     | O     | O    | O    | O    | O    | O    | O    | O    | O    | O    | x    |
| 12-bit         | O     | O     | O    | O    | O    | O    | O    | O    | O    | O    | O    | O    |

**O = Selected, x = Delete**

## ADC CONVERTING TIME

$$12\text{-bit ADC conversion time} = 1/(\text{ADC clock}/4) * 16 \text{ sec}$$

$$8\text{-bit ADC conversion time} = 1/(\text{ADC clock}/4) * 12 \text{ sec}$$

**High clock (fosc) is @3.58MHz**

| ADLEN      | ADCKS0 | ADC Clock | ADC conversion time                                  |
|------------|--------|-----------|--|
| 0 (8-bit)  | 0      | Fcpu/4    | $1/((3.58\text{MHz}/4)/4/4) * 12 = 214.5 \text{ us}$ |
|            | 1      | Fhosc     | $1/(3.58\text{MHz}/4) * 12 = 13.4 \text{ us}$        |
| 1 (12-bit) | 0      | Fcpu/4    | $1/((3.58\text{MHz}/4)/4/4) * 16 = 286 \text{ us}$   |
|            | 1      | Fhosc     | $1/(3.58\text{MHz}/4) * 16 = 17.9 \text{ us}$        |

⇒ **Example : To set AIN0 ~ AIN1 for ADC input and executing 12-bit ADC**

```

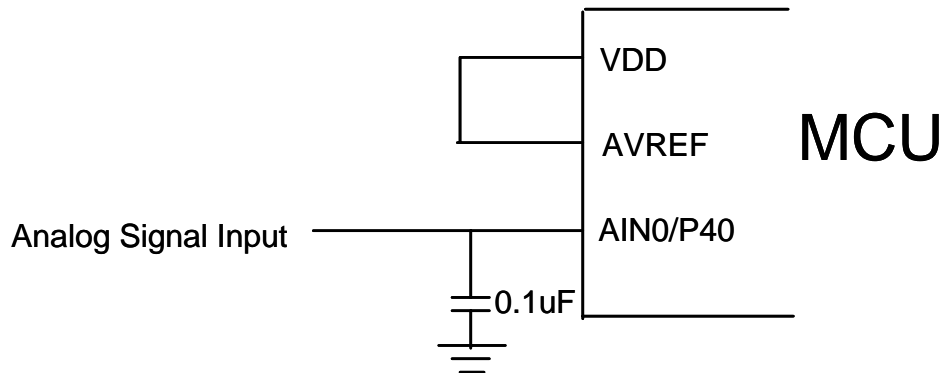
ADC0:
    MOV     A, #60H
    B0MOV  ADR, A           ; To set 12-bit ADC and ADC clock = Fosc.
    MOV     A, #90H
    B0MOV  ADM, A          ; To enable ADC and set AIN0 input
    B0BSET FADS            ; To start conversion

WADC0:
    B0BTS1 FEOC           ; To skip, if end of converting =1
    JMP     WADC0         ; else, jump to WADC0
    B0MOV  A, ADB         ; To get AIN0 input data

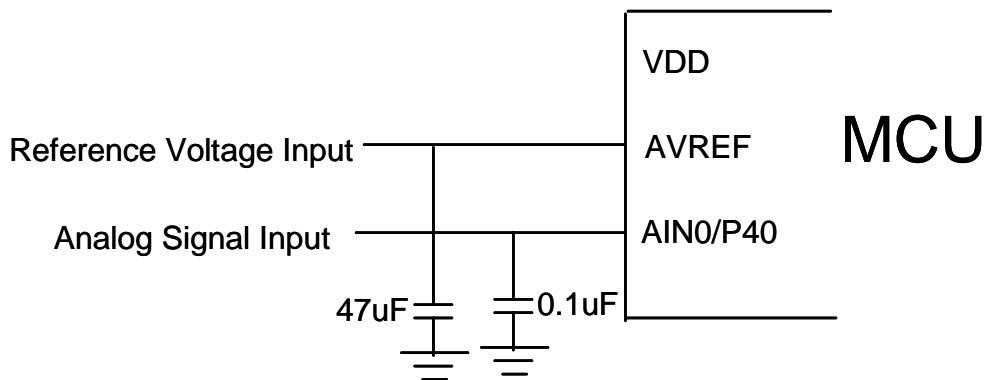
ADC1:
    MOV     A, #91H
    B0MOV  ADM, A          ; To enable ADC and set AIN1 input
    B0BSET FADS            ; To start conversion

QEXADC:
    B0BCLR FGCHS         ; To release AINx input channel
    
```

## ADC CIRCUIT



*AVREFH is connected to VDD.*



*AVREFH is connected to external AD reference voltage.*

Figure 12-2. The AINx and AVREFH Circuit of AD Converter

- **Note:** The capacitor between AIN and GND is a bypass capacitor. It is helpful to stable the analog signal. Users can omit it.

# 13 7-BIT DIGITAL TO ANALOG CONVERTER

## OVERVIEW

The D/A converter uses 7-bit structure to synthesize 128 steps' analog signal with current source output. After DAENB bit is set to "1", DAC circuit will turn to be enabled and the DAM register, from bit0 to bit6, will send digital signal to ladder resistors in order to generate analog signal on DAO pin.



Figure 13-1. The DA converter Block Diagram

In order to get a proper linear output, a Loading Resistor  $R_L$  is usually added between DAO and Ground. The example shows the result of  $V_{dd} = 5V$ ,  $R_L = 150\Omega$  and  $V_{dd} = 3V$ ,  $R_L = 150\Omega$ .

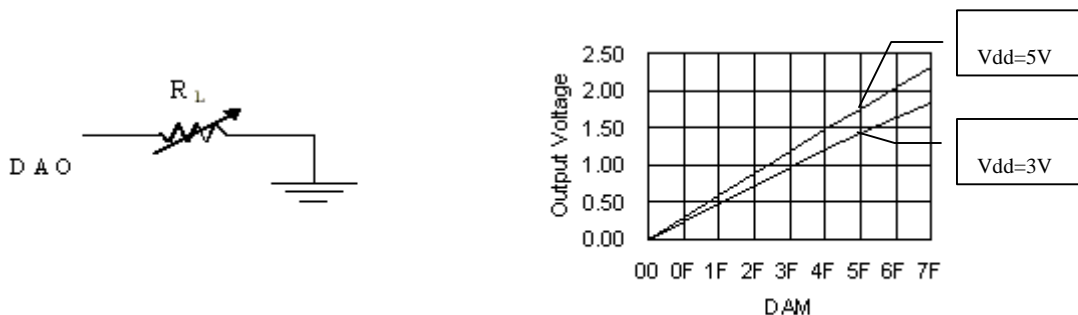


Figure 13-2 DAO Circuit with  $R_L$

Figure 13-3. DAC Output Voltage in  $V_{dd}=5V$  and  $3V$

The D/A converter is not designed for a precise DC voltage output and is suitable for a simple audio application e.g. Tone or Melody generation.

## DAM REGISTER

**DAM initial value = 0000 0000**

| 0B0H       | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|------------|-------|-------|-------|-------|-------|-------|-------|-------|
| <b>DAM</b> | DAENB | DAB6  | DAB5  | DAB4  | DAB3  | DAB2  | DAB1  | DAB0  |
|            | R/W   | R/W   | R/W   | R/W   | R/W   | R/W   | R/W   | R/W   |

DAENB: Digital to Analog converter control bit. 0 = disable, 1 = enable.

DABn: Digital input data.

## D/A CONVERTER OPERATION

When the DAENB = 0, the DAO pin is output floating status. After setting DAENB to “1”, the DAO output value is controlled by DAB bits.

⇒ **Example: Output 1/2 VDD from DAO pin.**

```
MOV      A, #00111111B
BOMOV   DAM, A           ; Set DAB to a half of the full scale.

BOBSET  FDAENB          ; Enable D/A function.
```

The DAB's data v.s. DAO's output voltage as following:

| DAB6 | DAB5 | DAB4 | DAB3 | DAB2 | DAB1 | DAB0 | DAO        |
|------|------|------|------|------|------|------|------------|
| 0    | 0    | 0    | 0    | 0    | 0    | 0    | VSS        |
| 0    | 0    | 0    | 0    | 0    | 0    | 1    | Idac       |
| 0    | 0    | 0    | 0    | 0    | 1    | 0    | 2 * Idac   |
| 0    | 0    | 0    | 0    | 0    | 1    | 1    | 3 * Idac   |
| .    | .    | .    | .    | .    | .    | .    | .          |
| .    | .    | .    | .    | .    | .    | .    | .          |
| .    | .    | .    | .    | .    | .    | .    | .          |
| 1    | 1    | 1    | 1    | 1    | 1    | 0    | 126 * Idac |
| 1    | 1    | 1    | 1    | 1    | 1    | 1    | 127 * Idac |

**Table 13-1. DAB and DAO Relative Table**

➤ **Note:**  $Idac = I_{FSO} / (2^7 - 1)$  ( $I_{FSO}$ : Full-scale Output Current)

# 14 CODING ISSUE

## TEMPLATE CODE

```

;*****
; FILENAME   : TEMPLATE.ASM
; AUTHOR     : SONiX
; PURPOSE    : Template Code for SN8X17XX
; REVISION   : 09/01/2002 V1.0   First issue
;*****
;* (c) Copyright 2002, SONiX TECHNOLOGY CO., LTD.
;*****
CHIP      SN8P1708                ; Select the CHIP

;-----
;
;                               Include Files
;-----
.nolist                            ; do not list the macro file

    INCLUDESTD  MACRO1.H
    INCLUDESTD  MACRO2.H
    INCLUDESTD  MACRO3.H

.list                               ; Enable the listing function

;-----
;
;                               Constants Definition
;-----
;   ONE      EQU      1

;-----
;
;                               Variables Definition
;-----
.DATA

    org      0h                    ;Bank 0 data section start from RAM address 0x000
    Wk00B0   DS      1              ;Temporary buffer for main loop
    Iwk00B0  DS      1              ;Temporary buffer for ISR
    AccBuf   DS      1              ;Accumulater buffer
    PflagBuf DS      1              ;PFLAG buffer

    org      100h                  ;Bank 1 data section start from RAM address 0x100
    BufB1    DS      20            ;Temporary buffer in bank 1

;-----
;
;                               Bit Flag Definition
;-----
    Wk00B0_0 EQU      Wk00B0.0    ;Bit 0 of Wk00B0
    Iwk00B0_1 EQU      Iwk00B0.1  ;Bit 1 of Iwk00
;-----

```



```

;                                     Code section
;-----

.CODE

    ORG        0                ;Code section start
    jmp        Reset            ;Reset vector
                                ;Address 4 to 7 are reserved

    ORG        8
    jmp        Isr              ;Interrupt vector

    ORG        10h
;-----
;   Program reset section
;-----
Reset:
    mov        A,#07Fh          ;Initial stack pointer and
    b0mov      STKP,A           ;disable global interrupt
    b0mov      PFLAG,#00h      ;pflag = x,x,x,x,x,c,dc,z
    b0mov      RBANK,#00h      ;Set initial RAM bank in bank 0
    mov        A,#40h          ;Clear watchdog timer and initial system mode
    b0mov      OSCM,A

    call       ClrRAM           ;Clear RAM
    call       SysInit         ;System initial
    b0bset     FGIE            ;Enable global interrupt

;-----
;   Main routine
;-----
Main:
    b0bset     FWDRST          ;Clear watchdog timer

    call       MnApp

    jmp        Main

;-----
;   Main application
;-----
MnApp:

    ; Put your main program here

    ret

;-----
;   Jump table routine
;-----
    ORG        0x0100          ;The jump table should start from the head
                                ;of boundary.

    b0mov      A,Wk00
    and        A,#3
    ADD        PCL,A
    jmp        JmpSub0
    jmp        JmpSub1
    jmp        JmpSub2
;-----

```

```

JumpSub0:
    ; Subroutine 1
    jmp          JumpExit

JumpSub1:
    ; Subroutine 2
    jmp          JumpExit

JumpSub2:
    ; Subroutine 3
    jmp          JumpExit

JumpExit:
    ret                      ;Return Main

;-----
; Isr (Interrupt Service Routine)
; Arguments :
; Returns   :
; Reg Change:
;-----
Isr:
;-----
;   Save ACC and system registers
;-----
    b0xch      A,AccBuf      ;B0xch instruction do not change C,Z flag
    push                       ;Remark this line in SN8P1702 registers
                               ;Save 80h ~ 87h system

    ;Following two lines for SN8X1702 only
    ;b0mov     A,PFLAG
    ;b0mov     PflagBuf,A

;-----
;   Check which interrupt happen
;-----

IntP00Chk:
    b0bts1    FP00IEN
    jmp       IntTc0Chk      ;Modify this line for another interrupt
    b0bts0    FP00IRQ
    jmp       P00isr

    ;If necessary, insert another interrupt checking here

IntTc0Chk:
    b0bts1    FTC0IEN
    jmp       IsrExit        ;Suppose TC0 is the last interrupt which you
    b0bts0    FTC0IRQ        ;want to check
    jmp       TC0isr

```

```
;-----  
; Exit interrupt service routine  
;-----  
  
IsrExit:  
  
    ; Following two lines for SN8X1702 only  
    ;b0mov     A,PFLAG  
    ;b0mov     PflagBuf,A  
  
    pop                               ;Remark this line in SN8P1702  
                                       ;Restore 80h ~ 87h system registers  
    b0xch     A,AccBuf                ;B0xch instruction do not change C,Z flag  
  
    reti                               ;Exit the interrupt routine  
  
;-----  
; INT0 interrupt service routine  
;-----  
P00isr:  
    b0bclr    FP00IRQ  
  
    ;Process P0.0 external interrupt here  
  
    jmp      IsrExit  
  
;-----  
; TC0 interrupt service routine  
;-----  
TC0isr:  
    b0bclr    FTC0IRQ  
  
    ;Process TC0 timer interrupt here  
  
    jmp      IsrExit  
  
;-----  
; SysInit  
; Initialize I/O, Timer, Interrupt, etc.  
;-----  
SysInit:  
  
    ret
```

```
-----  
; ClrRAM  
; Use index @YZ to clear RAM (00h~7Fh)  
-----  
  
ClrRAM:  
  
; RAM Bank 0  
  clr      Y          ;Select bank 0  
  b0mov    Z,#0x7f    ;Set @YZ address from 7fh  
  
ClrRAM10:  
  clr      @YZ        ;Clear @YZ content  
  decms    Z          ;z = z - 1 , skip next if z=0  
  jmp      ClrRAM10  
  clr      @YZ        ;Clear address 0x00  
  
; RAM Bank 1  
  mov      A,#1  
  b0mov    Y,A        ;Select bank 1  
  b0mov    Z,#0x7f    ;Set @YZ address from 17fh  
  
ClrRAM20:  
  clr      @YZ        ;Clear @YZ content  
  decms    Z          ;z = z - 1 , skip next if z=0  
  jmp      ClrRAM20  
  clr      @YZ        ;Clear address 0x100  
  ret  
  
-----  
  ENDP
```

## CHIP DECLARATION IN ASSEMBLER

| Assembler     | OTP Device Part Number | MASK Device Part Number |
|---------------|------------------------|-------------------------|
| CHIP SN8P1702 | SN8P1702               | SN8A1702A               |
| CHIP SN8P1704 | SN8P1704               | SN8A1704A               |
| CHIP SN8P1706 | SN8P1706               | SN8A1706A               |
| CHIP SN8P1707 | SN8P1707               | SN8A1707A               |
| CHIP SN8P1708 | SN8P1708               | SN8A1708A               |

## PROGRAM CHECK LIST

| Item                    | Description  |
|-------------------------|--|
| <b>Pull-up Resister</b> | Use @SET_PUR macro to enable or disable on-chip pull-up resistors. Refer I/O port chapter for detailed information.  |
| <b>Undefined Bits</b>   | All bits those are marked as "0" (undefined bits) in system registers should be set "0" to avoid unpredicted system errors.  |
| <b>ADC</b>              | Set ADC input pin I/O direction as input mode and disable pull-up resister of ADC input pin  |
| <b>SIO Master Mode</b>  | Set SCK (P5.0) and SO (P5.2) pin as output mode. Set SI (P5.1) pin as input mode.  |
| <b>SIO Slave Mode</b>   | Set SO (P5.2) pin as output mode. Set SCK (P5.0) and SI (P5.1) pin as input mode.  |
| <b>PWM0</b>             | Set PWM0 (P5.4) pin as output mode.  |
| <b>PWM1</b>             | Set PWM1 (P5.3) pin as output mode.  |
| <b>Interrupt</b>        | Do not enable interrupt before initializing RAM.   |
| <b>Non-Used I/O</b>     | Non-used I/O ports should be pull-up or pull-down in input mode, or be set as low in output mode to save current consumption.  |
| <b>Sleep Mode</b>       | Enable on-chip pull-up resistors of port 0 and port 1 to avoid unpredicted wakeup.   |
| <b>Stack Buffer</b>     | Be careful of function call and interrupt service routine operation. Don't let stack buffer overflow or underflow.   |
| <b>System Initial</b>   | <ol style="list-style-type: none"> <li>1. Write 0x7F into STKP register to initial stack pointer and disable global interrupt</li> <li>2. Clear all RAM.</li> <li>3. Initialize all system register even unused registers.</li> </ol>  |
| <b>Noisy Immunity</b>   | <ol style="list-style-type: none"> <li>1. Enable OSG and High_Clk / 2 code option together</li> <li>2. Enable the watchdog option to protect system crash.</li> <li>3. Non-used I/O ports should be set as output low mode</li> <li>4. Constantly refresh important system registers and variables in RAM to avoid system crash by a high electrical fast transient noise.</li> <li>5. Enable the LVD option to improve the power on reset or brown-out reset performance</li> </ol> |

# 15 INSTRUCTION SET TABLE

| Field        | Mnemonic   | Description  | C | DC | Z | Cycle |
|--------------|--|--|---|----|---|-------|
| MOV          | MOV A,M  | $A \leftarrow M$   | - | -  | √ | 1     |
|              | MOV M,A  | $M \leftarrow A$   | - | -  | - | 1     |
|              | B0MOV A,M  | $A \leftarrow M$ (bnak 0)  | - | -  | √ | 1     |
|              | B0MOV M,A  | $M$ (bank 0) $\leftarrow A$  | - | -  | - | 1     |
|              | MOV A,I  | $A \leftarrow I$   | - | -  | - | 1     |
|              | B0MOV M,I  | $M \leftarrow I$ , (M = only for Working registers R, Y, Z, RBANK & PFLAG)                                 | - | -  | - | 1     |
|              | XCH A,M  | $A \leftrightarrow M$  | - | -  | - | 1     |
|              | B0XCH A,M  | $A \leftrightarrow M$ (bank 0)   | - | -  | - | 1     |
| MOV R,A      | $R, A \leftarrow ROM [Y,Z]$  | -  | - | -  | 2 |       |
| ARITH        | ADC A,M  | $A \leftarrow A + M + C$ , if occur carry, then C=1, else C=0  | √ | √  | √ | 1     |
|              | ADC M,A  | $M \leftarrow A + M + C$ , if occur carry, then C=1, else C=0  | √ | √  | √ | 1     |
|              | ADD A,M  | $A \leftarrow A + M$ , if occur carry, then C=1, else C=0  | √ | √  | √ | 1     |
|              | ADD M,A  | $M \leftarrow A + M$ , if occur carry, then C=1, else C=0  | √ | √  | √ | 1     |
|              | B0ADD M,A  | $M$ (bank 0) $\leftarrow M$ (bank 0) + A, if occur carry, then C=1, else C=0                               | √ | √  | √ | 1     |
|              | ADD A,I  | $A \leftarrow A + I$ , if occur carry, then C=1, else C=0  | √ | √  | √ | 1     |
|              | SBC A,M  | $A \leftarrow A - M - /C$ , if occur borrow, then C=0, else C=1  | √ | √  | √ | 1     |
|              | SBC M,A  | $M \leftarrow A - M - /C$ , if occur borrow, then C=0, else C=1  | √ | √  | √ | 1     |
|              | SUB A,M  | $A \leftarrow A - M$ , if occur borrow, then C=0, else C=1   | √ | √  | √ | 1     |
|              | SUB M,A  | $M \leftarrow A - M$ , if occur borrow, then C=0, else C=1   | √ | √  | √ | 1     |
| C            | SUB A,I  | $A \leftarrow A - I$ , if occur borrow, then C=0, else C=1   | √ | √  | √ | 1     |
|              | DAA  | To adjust ACC's data format from HEX to DEC.   | √ | -  | - | 1     |
| MUL          | A,M  | $R, A \leftarrow A * M$ , The LB of product stored in Acc and HB stored in R register. ZF affected by Acc. | - | -  | √ | 2     |
| LOGIC        | AND A,M  | $A \leftarrow A$ and M   | - | -  | √ | 1     |
|              | AND M,A  | $M \leftarrow A$ and M   | - | -  | √ | 1     |
|              | AND A,I  | $A \leftarrow A$ and I   | - | -  | √ | 1     |
|              | OR A,M   | $A \leftarrow A$ or M  | - | -  | √ | 1     |
|              | OR M,A   | $M \leftarrow A$ or M  | - | -  | √ | 1     |
|              | OR A,I   | $A \leftarrow A$ or I  | - | -  | √ | 1     |
|              | XOR A,M  | $A \leftarrow A$ xor M   | - | -  | √ | 1     |
|              | XOR M,A  | $M \leftarrow A$ xor M   | - | -  | √ | 1     |
| XOR A,I      | $A \leftarrow A$ xor I   | -  | - | √  | 1 |       |
| PUSH         | SWAP M   | $A (b3\sim b0, b7\sim b4) \leftarrow M (b7\sim b4, b3\sim b0)$   | - | -  | - | 1     |
|              | SWAPM M  | $M (b3\sim b0, b7\sim b4) \leftarrow M (b7\sim b4, b3\sim b0)$   | - | -  | - | 1     |
|              | RRC M  | $A \leftarrow RRC M$   | √ | -  | - | 1     |
|              | RRCM M   | $M \leftarrow RRC M$   | √ | -  | - | 1     |
|              | RLC M  | $A \leftarrow RLC M$   | √ | -  | - | 1     |
|              | RLCM M   | $M \leftarrow RLC M$   | √ | -  | - | 1     |
|              | CLR M  | $M \leftarrow 0$   | - | -  | - | 1     |
|              | BCLR M.b   | $M.b \leftarrow 0$   | - | -  | - | 1     |
|              | BSET M.b   | $M.b \leftarrow 1$   | - | -  | - | 1     |
|              | BOBCLR M.b   | $M(bank 0).b \leftarrow 0$   | - | -  | - | 1     |
| BOBSET M.b   | $M(bank 0).b \leftarrow 1$   | -  | - | -  | 1 |       |
| BRANCH       | CMPSR A,I  | ZF,C $\leftarrow A - I$ , If A = I, then skip next instruction   | √ | -  | √ | 1+S   |
|              | CMPSR A,M  | ZF,C $\leftarrow A - M$ , If A = M, then skip next instruction   | √ | -  | √ | 1+S   |
|              | INCS M   | $A \leftarrow M + 1$ , If A = 0, then skip next instruction  | - | -  | - | 1+S   |
|              | INCMS M  | $M \leftarrow M + 1$ , If M = 0, then skip next instruction  | - | -  | - | 1+S   |
|              | DECS M   | $A \leftarrow M - 1$ , If A = 0, then skip next instruction  | - | -  | - | 1+S   |
|              | DECMS M  | $M \leftarrow M - 1$ , If M = 0, then skip next instruction  | - | -  | - | 1+S   |
|              | 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   |
|              | BOBTS0 M.b   | If M(bank 0).b = 0, then skip next instruction   | - | -  | - | 1+S   |
|              | BOBTS1 M.b   | If M(bank 0).b = 1, then skip next instruction   | - | -  | - | 1+S   |
|              | JMP d  | $PC15/14 \leftarrow RomPages1/0, PC13\sim PC0 \leftarrow d$  | - | -  | - | 2     |
|              | CALL d   | $Stack \leftarrow PC15\sim PC0, PC15/14 \leftarrow RomPages1/0, PC13\sim PC0 \leftarrow d$                 | - | -  | - | 2     |
| MISC         | RET  | $PC \leftarrow Stack$  | - | -  | - | 2     |
|              | RETI   | $PC \leftarrow Stack$ , and to enable global interrupt   | - | -  | - | 2     |
|              | PUSH   | To push working registers (080H-087H) into buffers   | - | -  | - | 1     |
|              | POP  | To pop working registers (080H-087H) from buffers  | √ | √  | √ | 1     |
|              | NOP  | No operation   | - | -  | - | 1     |
| @SET_PUR VAL | Enable or disable pull-up resistors. Bit N of VAL: "0" disable port N pull-up, "1" enable port N pull-up | -  | - | √  | - |       |

**Table 15-1. Instruction Set Table of SN8P1700**

**Note 1:** Any instruction that read/write from 0SCM, will add an extra cycle.)

**Note 2:** SN8P1702/SN8A1702 don't provide "MUL, PUSH, POP" instruction.

# 16 ELECTRICAL CHARACTERISTIC

## ABSOLUTE MAXIMUM RATING

|   |   |                         |
|---|---|-------------------------|
| Supply voltage (Vdd).....                 | (All of the voltages referenced to Vss) | - 0.3V ~ 6.0V           |
| Input in voltage (Vin).....               |   | Vss - 0.2V ~ Vdd + 0.2V |
| Operating ambient temperature (Topr)..... |   | -20°C ~ + 70°C          |
| Storage ambient temperature (Tstor).....  |   | -30°C ~ + 125°C         |
| Power consumption (Pc).....               |   | 500 mW                  |

## STANDARD ELECTRICAL CHARACTERISTIC

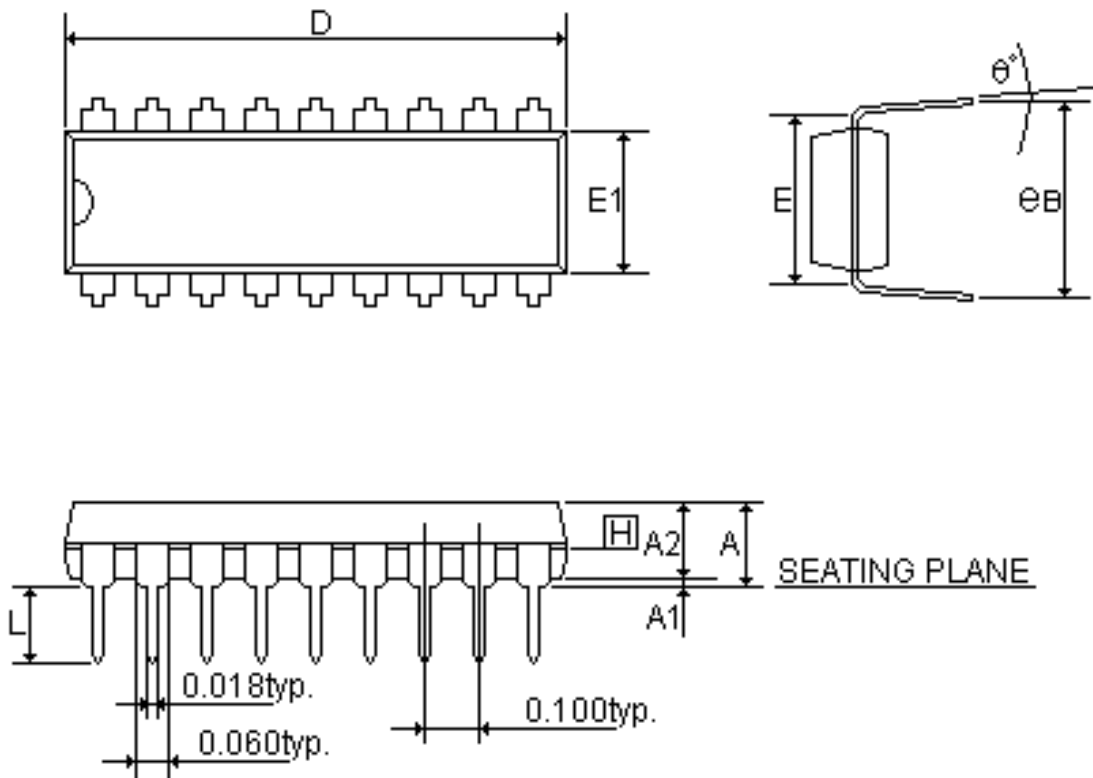
### SN8P1700 Series (OTP)

(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                          | 2.2                  | 5.0  | 5.5    | V     |    |
|   |                  | Programming mode, Vpp = 12.5V                   | 4.5                  | 5.0  | 5.5    |       |    |
| RAM Data Retention voltage              | Vdr              |   | -                    | 1.5  | -      | V     |    |
| Internal POR                            | Vpor             | Vdd rise rate to ensure internal power-on reset | -                    | 0.05 | -      | V/ms  |    |
| Input Low Voltage                       | ViL1             | All input pins except those specified below     | Vss                  | -    | 0.3Vdd | V     |    |
|   | ViL2             | Input with Schmitt trigger buffer - Port0       | Vss                  | -    | 0.2Vdd | V     |    |
|   | ViL3             | Reset pin ; Xin ( in RC mode )                  | Vss                  | -    | 0.2Vdd | V     |    |
|   | ViL4             | Xin ( in X'tal mode )                           | Vss                  | -    | 0.3Vdd | V     |    |
| Input High Voltage                      | ViH1             | All input pins except those specified below     | 0.7Vdd               | -    | Vdd    | V     |    |
|   | ViH2             | Input with Schmitt trigger buffer -Port0        | 0.8Vdd               | -    | Vdd    | V     |    |
|   | ViH3             | Reset pin ; Xin ( in RC mode )                  | 0.9Vdd               | -    | Vdd    | V     |    |
|   | ViH4             | Xin ( in X'tal mode )                           | 0.7Vdd               | -    | Vdd    | V     |    |
| Reset pin leakage current               | Ilekg            | Vin = Vdd                                       | -                    | -    | 2      | uA    |    |
| I/O port pull-up resistor               | Rup              | Vin = Vss , Vdd = 5V                            | -                    | 100  | -      | KΩ    |    |
| I/O port input leakage current          | Ilekg            | Pull-up resistor disable, Vin = Vdd             | -                    | -    | 2      | uA    |    |
| Port1 output source current             | IoH              | Vop = Vdd - 0.5V                                | -                    | 12   | -      | mA    |    |
|   | IoL              | Vop = Vss + 0.5V                                | -                    | 15   | -      |       |    |
| Port2 output source current             | IoH              | Vop = Vdd - 0.5V                                | -                    | 12   | -      | mA    |    |
|   | IoL              | Vop = Vss + 0.5V                                | -                    | 15   | -      |       |    |
| Port4 output source current             | IoH              | Vop = Vdd - 0.5V                                | -                    | 12   | -      | mA    |    |
|   | IoL              | Vop = Vss + 0.5V                                | -                    | 15   | -      |       |    |
| Port5 output source current             | IoH              | Vop = Vdd - 0.5V                                | -                    | 12   | -      | mA    |    |
|   | IoL              | Vop = Vss + 0.5V                                | -                    | 15   | -      |       |    |
| INTn trigger pulse width                | Tint0            | INT0 ~ INT2 interrupt request pulse width       | 2/fcpu               | -    | -      | cycle |    |
| AVREFH input voltage                    | Varef            | Vdd = 5.0V                                      | 1.2V                 | -    | Vdd    | V     |    |
| AIN0 ~ AIN7 input voltage               | Vani             |   | Vss+0.2              | -    | Avref  | V     |    |
| Oscillator Frequency                    | Fosc             | Crystal type or ceramic resonator               | 32768                | 4M   | 16M    | Hz    |    |
|   |                  | VDD = 3V, RC type for external mode             | -                    | 6M   | -      |       |    |
|   |                  | VDD = 5V, RC type for external mode             | -                    | 10M  | -      |       |    |
| Supply Current<br>(Disable ADC and LVD) | Idd1             | Run Mode  | Vdd= 5V 4Mhz         | -    | 7      | 15    | mA |
|   |                  |   | Vdd= 3V 4Mhz         | -    | 1.5    | 3     | mA |
|   |                  |   | Vdd= 3V 32768Hz      | -    | 50     | 100   | uA |
|   | Idd2             | Slow mode<br>(Stop High Clock)                  | Vdd= 5V 32KHz Int RC | -    | 80     | 150   | uA |
|   |                  |   | Vdd= 3V 16KHz Int RC | -    | 15     | 30    | uA |
|   | Idd3             | Sleep mode                                      | Vdd= 5V              | -    | 10     | 18    | uA |
| Vdd= 3V                                 |                  |   | -                    | 3    | 6      | uA    |    |
| LVD Detect Voltage                      | Vdet             | Low voltage detect level                        | -                    | 2.4  | -      | V     |    |
| Voltage detector current                | Ivdet            | LVD enable operating current                    | -                    | 100  | 180    | uA    |    |
| ADC current consumption                 | I <sub>ADC</sub> | Vdd=5.0V  | -                    | 0.6  | 1      | mA    |    |
|   |                  | Vdd=3.0V  | -                    | 0.4  | 0.8    | mA    |    |
| DAC Full-scale Output Current           | I <sub>FSO</sub> | Vdd=5V, RL =150ohm                              | -                    | 12   | -      | mA    |    |

# 17 PACKAGE INFORMATION

## P-DIP18 PIN

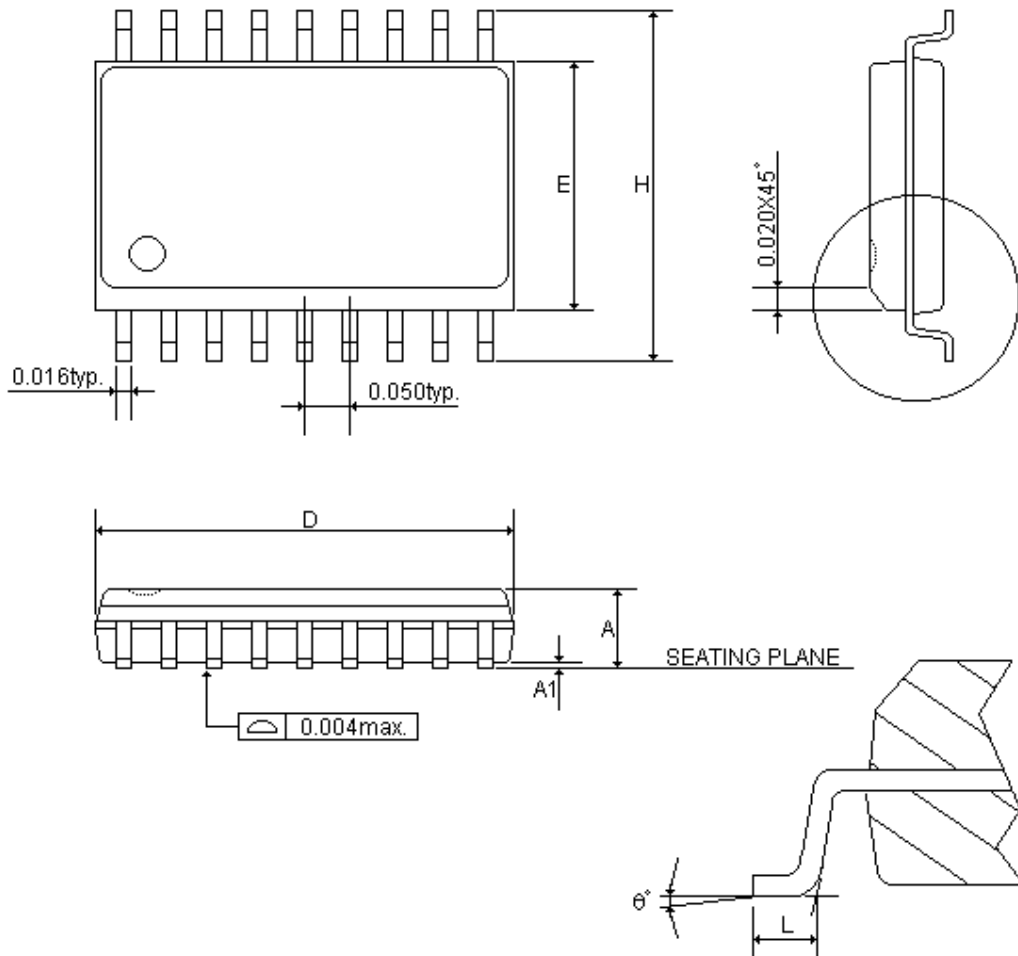


| Symbols        | MIN.      | NOR.  | MAX.  |
|----------------|-----------|-------|-------|
| A              | -         | -     | 0.210 |
| A1             | 0.015     | -     | -     |
| A2             | 0.125     | 0.130 | 0.135 |
| D              | 0.880     | 0.900 | 0.920 |
| E              | 0.300BSC. |       |       |
| E1             | 0.245     | 0.250 | 0.255 |
| L              | 0.115     | 0.130 | 0.150 |
| eB             | 0.335     | 0.355 | 0.375 |
| $\theta^\circ$ | 0         | 7     | 15    |

UNIT : INCH



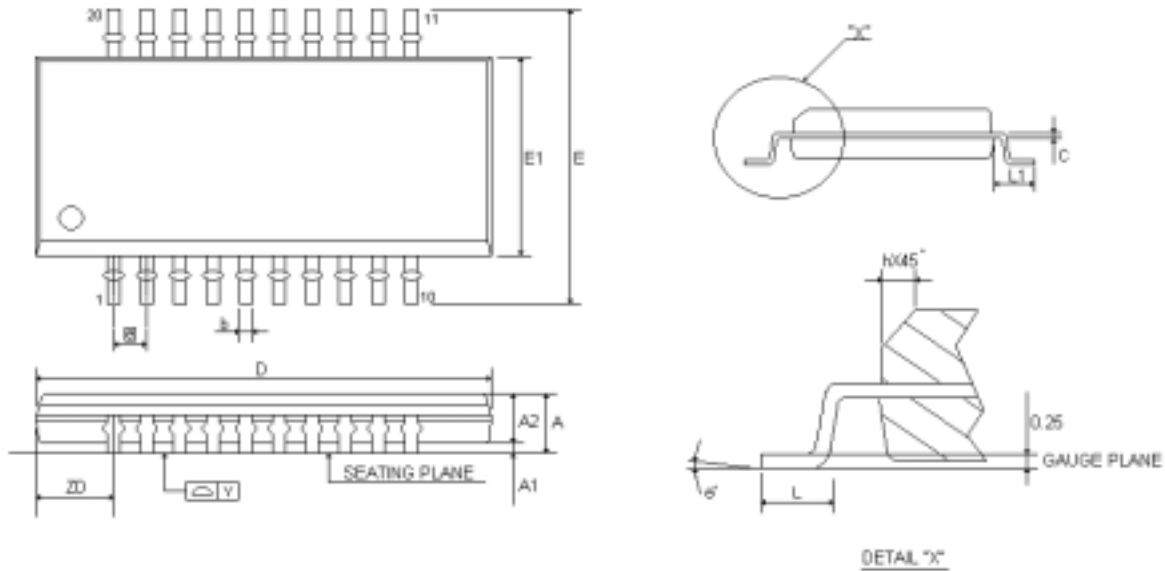
## SOP18 PIN



| Symbols        | MIN.  | MAX.  |
|----------------|-------|-------|
| A              | 0.093 | 0.104 |
| A1             | 0.004 | 0.012 |
| D              | 0.447 | 0.463 |
| E              | 0.291 | 0.299 |
| H              | 0.394 | 0.419 |
| L              | 0.016 | 0.050 |
| $\theta^\circ$ | 0     | 8     |

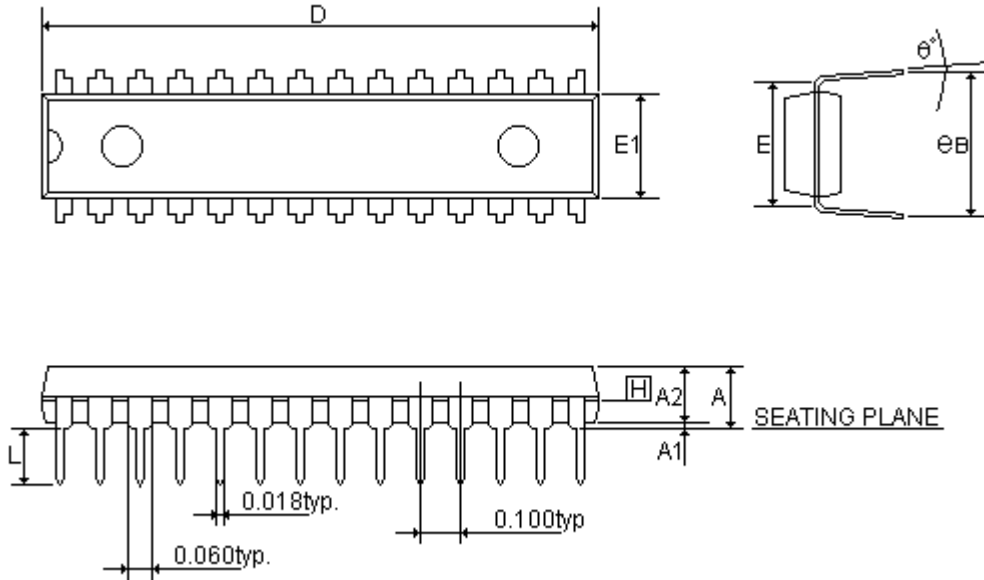
UNIT : INCH

## SSOP20 PIN



| Symbols    | DIMENSION (MM) |       |      | DIMENSION (MIL) |      |      |
|------------|----------------|-------|------|-----------------|------|------|
|            | MIN.           | NOM.  | MAX. | MIN.            | NOM. | MAX. |
| A          | 1.35           | 1.60  | 1.75 | 53              | 63   | 69   |
| A1         | 0.10           | 0.15  | 0.25 | 4               | 6    | 10   |
| A2         | -              | -     | 1.50 | -               | -    | 59   |
| b          | 0.20           | 0.254 | 0.30 | 8               | 10   | 12   |
| b1         | 0.20           | 0.254 | 0.28 | 8               | 11   | 11   |
| C          | 0.18           | 0.203 | 0.25 | 7               | 8    | 10   |
| C1         | 0.18           | 0.203 | 0.23 | 7               | 8    | 9    |
| D          | 8.56           | 8.66  | 8.74 | 337             | 341  | 344  |
| E          | 5.80           | 6.00  | 6.20 | 228             | 236  | 244  |
| E1         | 3.80           | 3.90  | 4.00 | 150             | 154  | 157  |
| e          | 0.635 BSC      |       |      | 25 BSC          |      |      |
| h          | 0.25           | 0.42  | 0.50 | 10              | 17   | 20   |
| L          | 0.40           | 0.635 | 1.27 | 16              | 25   | 50   |
| L1         | 1.00           | 1.05  | 1.10 | 39              | 41   | 43   |
| ZD         | 1.50 REF       |       |      | 58 REF          |      |      |
| Y          | -              | -     | 0.10 | -               | -    | 4    |
| $\theta$ ° | 0°             | -     | 8°   | 0°              | -    | 8°   |

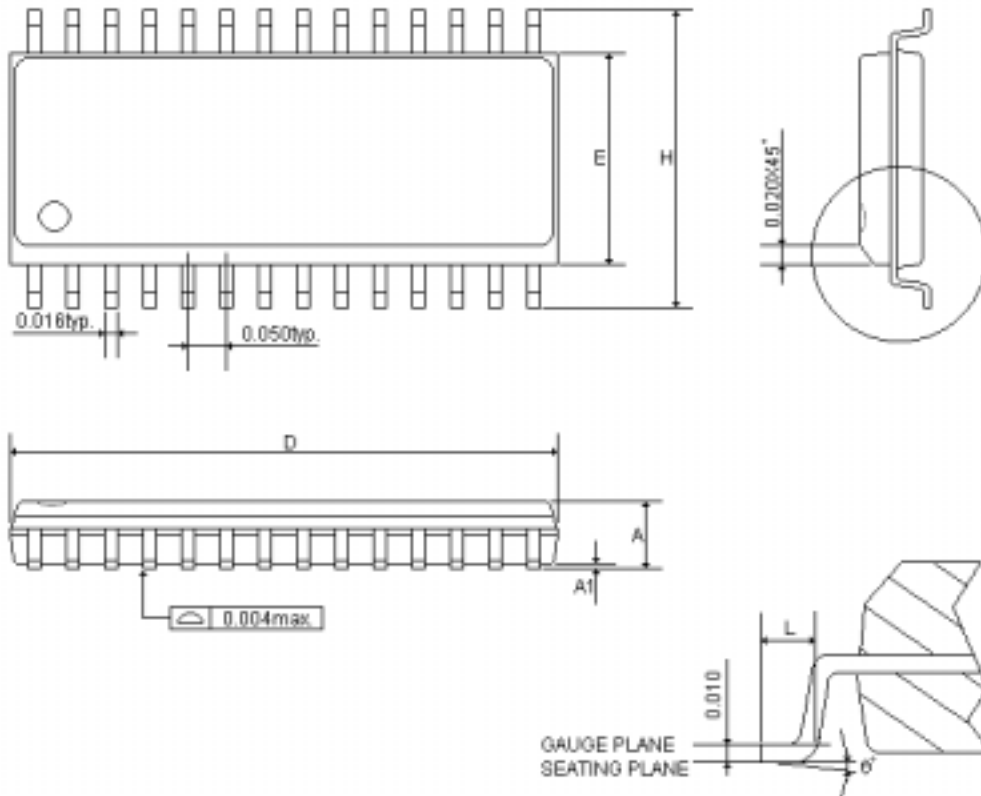
## S-DIP28 PIN



| 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 <sub>B</sub> | 0.330     | 0.350 | 0.370 |
| θ °            | 0         | 7     | 15    |

UNIT : INCH

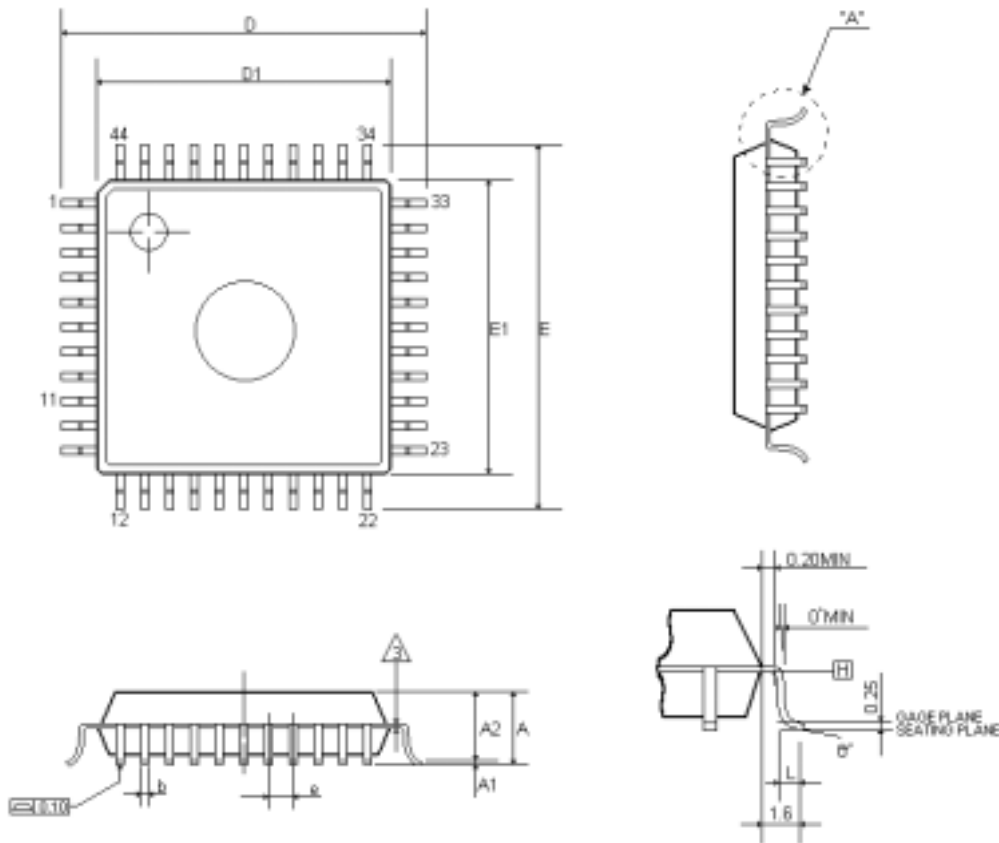
## SOP28 PIN



| 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     |

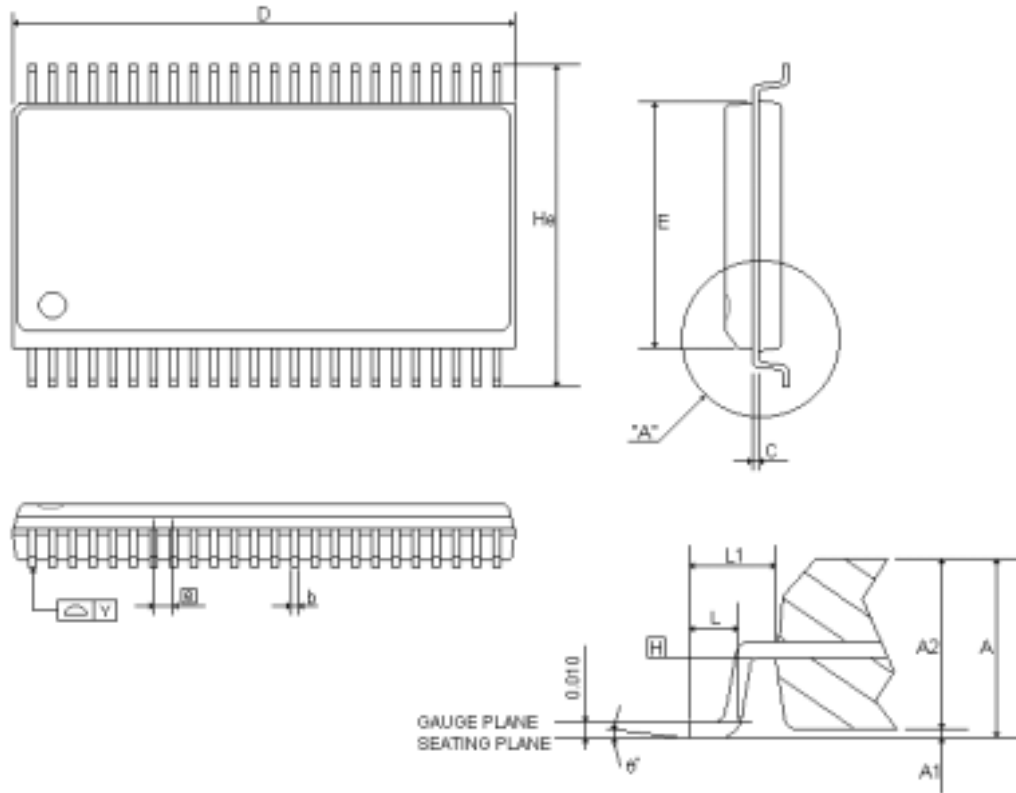
UNIT : INCH

## QFP 44 PIN



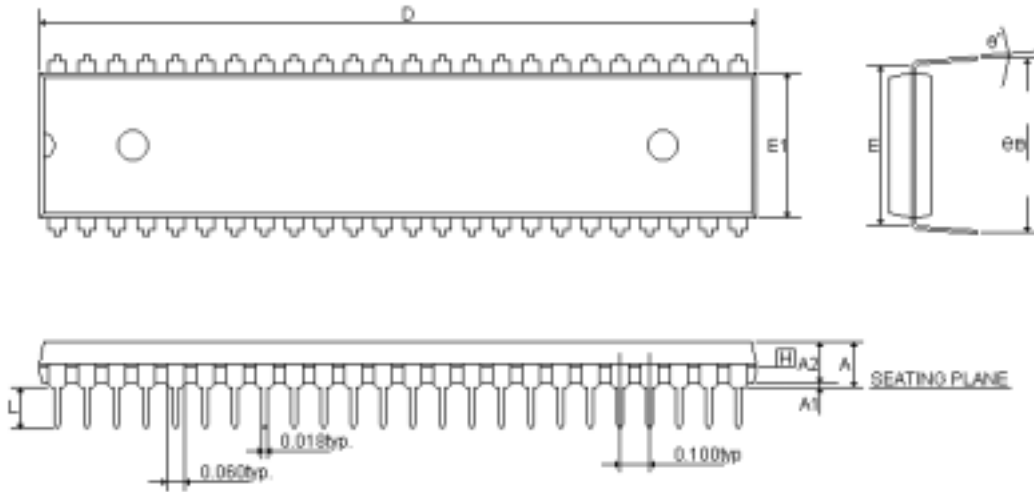
| SYMBOLS        | MIN    | NOR   | MAX   | MIN    | NOR    | MAX    |
|----------------|--------|-------|-------|--------|--------|--------|
|                | (inch) |       |       | (mm)   |        |        |
| A              | -      | -     | 0.106 | -      | -      | 2.700  |
| A1             | 0.010  | 0.012 | 0.014 | 0.250  | 0.300  | 0.350  |
| A2             | 0.075  | 0.079 | 0.087 | 1.900  | 2.000  | 2.200  |
| b              | 0.012  |       |       | 0.300  |        |        |
| C              | 0.004  | 0.006 | 0.008 | 0.100  | 0.150  | 0.200  |
| D              | 0.512  | 0.520 | 0.528 | 13.000 | 13.200 | 13.400 |
| D1             | 0.390  | 0.394 | 0.398 | 9.900  | 10.000 | 10.100 |
| E              | 0.512  | 0.520 | 0.528 | 13.000 | 13.200 | 13.400 |
| E1             | 0.390  | 0.394 | 0.398 | 9.900  | 10.000 | 10.100 |
| L              | 0.029  | 0.035 | 0.037 | 0.730  | 0.880  | 0.930  |
| [e]            | 0.031  |       |       | 0.800  |        |        |
| $\theta^\circ$ | 0°     | -     | 7°    | 0°     | -      | 7°     |

## SSOP 48 PIN



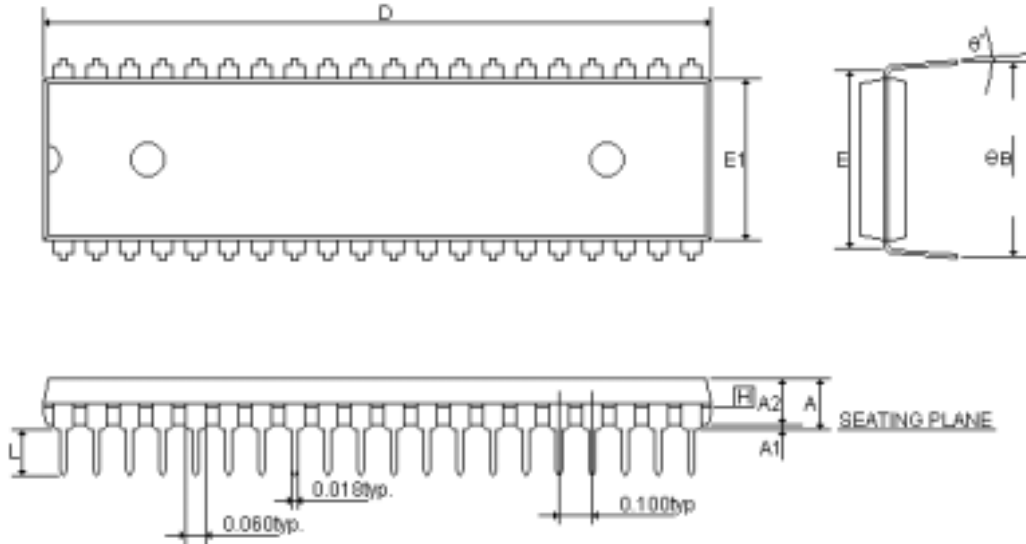
| SYMBOLS | MIN    | NOR   | MAX   | MIN    | NOR    | MAX    |
|---------|--------|-------|-------|--------|--------|--------|
|         | (inch) |       |       | (mm)   |        |        |
| A       | 0.095  | 0.102 | 0.110 | 2.413  | 2.591  | 2.794  |
| A1      | 0.008  | 0.012 | 0.016 | 0.203  | 0.305  | 0.406  |
| A2      | 0.089  | 0.094 | 0.099 | 2.261  | 2.388  | 2.515  |
| b       | 0.008  | 0.010 | 0.030 | 0.203  | 0.254  | 0.762  |
| C       | -      | 0.008 | -     | -      | 0.203  | -      |
| D       | 0.620  | 0.625 | 0.630 | 15.748 | 15.875 | 16.002 |
| E       | 0.291  | 0.295 | 0.299 | 7.391  | 7.493  | 7.595  |
| [e]     | -      | 0.025 | -     | -      | 0.635  | -      |
| He      | 0.396  | 0.406 | 0.416 | 10.058 | 10.312 | 10.566 |
| L       | 0.020  | 0.030 | 0.040 | 0.508  | 0.762  | 1.016  |
| L1      | -      | 0.056 | -     | -      | 1.422  | -      |
| Y       | -      | -     | 0.003 | -      | -      | 0.076  |
| θ°      | 0°     | -     | 8°    | 0°     | -      | 8°     |

## P-DIP 48 PIN



| SYMBOLS   | MIN          | NOR          | MAX          | MIN           | NOR           | MAX           |
|-----------|--------------|--------------|--------------|---------------|---------------|---------------|
|           | (inch)       |              |              | (mm)          |               |               |
| <b>A</b>  | -            | -            | <b>0.220</b> | -             | -             | <b>5.588</b>  |
| <b>A1</b> | <b>0.015</b> | -            | -            | <b>0.381</b>  | -             | -             |
| <b>A2</b> | <b>0.150</b> | <b>0.155</b> | <b>0.160</b> | <b>3.810</b>  | <b>3.937</b>  | <b>4.064</b>  |
| <b>D</b>  | <b>2.400</b> | <b>2.450</b> | <b>2.550</b> | <b>60.960</b> | <b>62.230</b> | <b>64.770</b> |
| <b>E</b>  | <b>0.600</b> |              |              | <b>15.240</b> |               |               |
| <b>E1</b> | <b>0.540</b> | <b>0.545</b> | <b>0.550</b> | <b>13.716</b> | <b>13.843</b> | <b>13.970</b> |
| <b>L</b>  | <b>0.115</b> | <b>0.130</b> | <b>0.150</b> | <b>2.921</b>  | <b>3.302</b>  | <b>3.810</b>  |
| <b>eB</b> | <b>0.630</b> | <b>0.650</b> | <b>0.067</b> | <b>16.002</b> | <b>16.510</b> | <b>1.702</b>  |
| <b>θ°</b> | <b>0°</b>    | <b>7°</b>    | <b>15°</b>   | <b>0°</b>     | <b>7°</b>     | <b>15°</b>    |

## P-DIP 40 PIN



| SYMBOLS   | MIN          | NOR          | MAX          | MIN           | NOR           | MAX           |
|-----------|--------------|--------------|--------------|---------------|---------------|---------------|
|           | (inch)       |              |              | (mm)          |               |               |
| <b>A</b>  | -            | -            | <b>0.220</b> | -             | -             | <b>5.588</b>  |
| <b>A1</b> | <b>0.015</b> | -            | -            | <b>0.381</b>  | -             | -             |
| <b>A2</b> | <b>0.150</b> | <b>0.115</b> | <b>0.160</b> | <b>3.810</b>  | <b>2.921</b>  | <b>4.064</b>  |
| <b>D</b>  | <b>2.055</b> | <b>2.060</b> | <b>2.070</b> | <b>52.197</b> | <b>52.324</b> | <b>52.578</b> |
| <b>E</b>  | <b>0.600</b> |              |              | <b>15.240</b> |               |               |
| <b>E1</b> | <b>0.540</b> | <b>0.545</b> | <b>0.550</b> | <b>13.716</b> | <b>13.843</b> | <b>13.970</b> |
| <b>L</b>  | <b>0.115</b> | <b>0.130</b> | <b>0.150</b> | <b>2.921</b>  | <b>3.302</b>  | <b>3.810</b>  |
| <b>eB</b> | <b>0.630</b> | <b>0.650</b> | <b>0.067</b> | <b>16.002</b> | <b>16.510</b> | <b>1.702</b>  |
| <b>θ°</b> | <b>0°</b>    | <b>7°</b>    | <b>15°</b>   | <b>0°</b>     | <b>7°</b>     | <b>15°</b>    |



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 use as components in systems intended, for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the SONIX product could create a situation where personal injury or death may occur. Should Buyer purchase or use SONIX products for any such unintended or unauthorized application. Buyer shall indemnify and hold SONIX and its officers, employees, subsidiaries, affiliates and distributors harmless against all claims, cost, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use even if such claim alleges that SONIX was negligent regarding the design or manufacture of the part.

**Main Office:**

Address: 9F, NO. 8, Hsien Cheng 5th St, Chupei City, Hsinchu, Taiwan R.O.C.

Tel: 886-3-551 0520

Fax: 886-3-551 0523

**Taipei Office:**

Address: 15F-2, NO. 171, Song Ted Road, Taipei, Taiwan R.O.C.

Tel: 886-2-2759 1980

Fax: 886-2-2759 8180

**Hong Kong Office:**

Address: Flat 3 9/F Energy Plaza 92 Granville Road, Tsimshatsui East Kowloon.

Tel: 852-2723 8086

Fax: 852-2723 9179

**Technical Support by Email:**

Sn8fae@sonix.com.tw