



**Touch Proximity Sensing Flash MCU**

**BS45F3340**

Revision: V1.20 Date: January 10, 2024

[www.holtek.com](http://www.holtek.com)

## Table of Contents

|  |           |
|--|-----------|
| <b>Features</b> .....  | <b>7</b>  |
| CPU Features .....   | 7         |
| Peripheral Features.....   | 7         |
| <b>Development Tools</b> .....                                   | <b>7</b>  |
| <b>General Description</b> .....                                 | <b>8</b>  |
| <b>Block Diagram</b> .....                                       | <b>8</b>  |
| <b>Pin Assignment</b> .....                                      | <b>9</b>  |
| <b>Pin Description</b> .....                                     | <b>10</b> |
| <b>Absolute Maximum Ratings</b> .....                            | <b>12</b> |
| <b>D.C. Characteristics</b> .....                                | <b>13</b> |
| Operating Voltage Characteristics.....                           | 13        |
| Operating Current Characteristics.....                           | 13        |
| Standby Current Characteristics .....                            | 13        |
| <b>A.C. Characteristics</b> .....                                | <b>14</b> |
| High Speed Internal Oscillator – HIRC – Frequency Accuracy ..... | 14        |
| Low Speed Internal Oscillator – LIRC – Frequency Accuracy .....  | 14        |
| Operating Frequency Characteristic Curves .....                  | 14        |
| System Start Up Time Characteristics .....                       | 15        |
| <b>Input/Output Characteristics</b> .....                        | <b>15</b> |
| <b>Memory Characteristics</b> .....                              | <b>16</b> |
| <b>LVR Electrical Characteristics</b> .....                      | <b>16</b> |
| <b>Internal Reference Voltage Characteristics</b> .....          | <b>16</b> |
| <b>A/D Converter Electrical Characteristics</b> .....            | <b>17</b> |
| Temperature Sensor Characteristics .....                         | 17        |
| <b>Proximity Sensing AFE Electrical Characteristics</b> .....    | <b>18</b> |
| Comparator Characteristics .....                                 | 18        |
| Operational Amplifier Characteristics .....                      | 19        |
| D/A Converter Characteristics.....                               | 19        |
| <b>Sink Current Generator Electrical Characteristics</b> .....   | <b>20</b> |
| <b>Power-on Reset Characteristics</b> .....                      | <b>21</b> |
| <b>System Architecture</b> .....                                 | <b>21</b> |
| Clocking and Pipelining.....                                     | 21        |
| Program Counter.....   | 22        |
| Stack .....  | 23        |
| Arithmetic and Logic Unit – ALU .....                            | 23        |
| <b>Flash Program Memory</b> .....                                | <b>24</b> |
| Structure.....   | 24        |
| Special Vectors .....  | 24        |
| Look-up Table.....   | 24        |

|   |           |
|---|-----------|
| Table Program Example .....                       | 25        |
| In Circuit Programming – ICP .....                | 26        |
| On Chip Debug Support – OCDS .....                | 27        |
| <b>Data Memory .....</b>                          | <b>27</b> |
| Structure.....                                    | 27        |
| General Purpose Data Memory .....                 | 28        |
| Special Purpose Data Memory .....                 | 28        |
| <b>Special Function Register Description.....</b> | <b>30</b> |
| Indirect Addressing Registers – IAR0, IAR1 .....  | 30        |
| Memory Pointers – MP0, MP1 .....                  | 30        |
| Accumulator – ACC .....                           | 31        |
| Program Counter Low Byte Register – PCL.....      | 31        |
| Look-up Table Registers – TBLP, TBHP, TBLH .....  | 31        |
| Status Register – STATUS .....                    | 31        |
| Bank Pointer – BP.....                            | 33        |
| Option Memory Mapping Register – ORMC .....       | 33        |
| <b>EEPROM Data Memory.....</b>                    | <b>34</b> |
| EEPROM Data Memory Structure .....                | 34        |
| EEPROM Registers .....                            | 34        |
| Reading Data from the EEPROM .....                | 36        |
| Writing Data to the EEPROM.....                   | 36        |
| Write Protection.....                             | 36        |
| EEPROM Interrupt .....                            | 36        |
| Programming Considerations.....                   | 37        |
| <b>Oscillators .....</b>                          | <b>38</b> |
| Oscillator Overview .....                         | 38        |
| System Clock Configurations .....                 | 38        |
| Internal High Speed RC Oscillator – HIRC .....    | 39        |
| Internal 32kHz Oscillator – LIRC.....             | 39        |
| <b>Operating Modes and System Clocks .....</b>    | <b>39</b> |
| System Clocks .....                               | 39        |
| System Operation Modes.....                       | 40        |
| Control Registers .....                           | 42        |
| Operating Mode Switching.....                     | 43        |
| FAST Mode to SLOW Mode Switching .....            | 44        |
| Standby Current Considerations .....              | 46        |
| Wake-up.....                                      | 47        |
| <b>Watchdog Timer.....</b>                        | <b>48</b> |
| Watchdog Timer Clock Source.....                  | 48        |
| Watchdog Timer Control Register .....             | 48        |
| Watchdog Timer Operation .....                    | 49        |
| <b>Reset and Initialisation.....</b>              | <b>50</b> |
| Reset Functions .....                             | 50        |
| Reset Initial Conditions .....                    | 52        |

|   |            |
|---|------------|
| <b>Input/Output Ports .....</b>             | <b>56</b>  |
| Pull-high Resistors .....                   | 56         |
| Port A Wake-up .....                        | 57         |
| I/O Port Control Registers .....            | 57         |
| Pin-shared Functions .....                  | 58         |
| I/O Pin Structures .....                    | 62         |
| Programming Considerations.....             | 62         |
| <b>Timer Modules – TM .....</b>             | <b>63</b>  |
| Introduction .....                          | 63         |
| TM Operation .....                          | 63         |
| TM Clock Source.....                        | 63         |
| TM Interrupts.....                          | 64         |
| TM External Pins.....                       | 64         |
| Programming Considerations.....             | 65         |
| <b>Compact Type TM – CTM .....</b>          | <b>66</b>  |
| Compact Type TM Operation .....             | 66         |
| Compact Type TM Register Description.....   | 66         |
| Compact Type TM Operating Modes .....       | 70         |
| <b>Standard Type TM – STM .....</b>         | <b>76</b>  |
| Standard TM Operation.....                  | 76         |
| Standard Type TM Register Description ..... | 76         |
| Standard Type TM Operation Modes .....      | 81         |
| <b>Proximity Sensing Circuit.....</b>       | <b>89</b>  |
| Proximity Sensing Circuit Operation .....   | 89         |
| Proximity Sensing Circuit Registers .....   | 89         |
| Offset Calibration Procedure .....          | 95         |
| <b>Sink Current Generator .....</b>         | <b>97</b>  |
| Sink Current Generator Registers.....       | 97         |
| <b>Analog to Digital Converter .....</b>    | <b>99</b>  |
| A/D Converter Overview .....                | 99         |
| A/D Converter Register Description .....    | 100        |
| A/D Converter Operation.....                | 103        |
| A/D Converter Reference Voltage.....        | 104        |
| A/D Converter Input Signals.....            | 104        |
| Conversion Rate and Timing Diagram .....    | 105        |
| Summary of A/D Conversion Steps .....       | 106        |
| Programming Considerations.....             | 107        |
| A/D Conversion Function .....               | 107        |
| Temperature Sensor Function.....            | 107        |
| A/D Conversion Programming Examples.....    | 108        |
| <b>UART Interface.....</b>                  | <b>110</b> |
| UART External Pins .....                    | 111        |
| UART Single Wire Mode .....                 | 111        |
| UART Data Transfer Scheme.....              | 111        |

|  |            |
|--|------------|
| UART Status and Control Registers..... | 112        |
| Baud Rate Generator .....              | 119        |
| UART Setup and Control.....            | 120        |
| UART Transmitter.....                  | 121        |
| UART Receiver .....                    | 123        |
| Managing Receiver Errors .....         | 124        |
| UART Interrupt Structure.....          | 125        |
| UART Power Down and Wake-up.....       | 127        |
| <b>Touch Key Function .....</b>        | <b>128</b> |
| Touch Key Structure.....               | 128        |
| Touch Key Register Definition.....     | 128        |
| Touch Key Operation.....               | 132        |
| Touch Key Interrupt.....               | 133        |
| Programming Considerations.....        | 133        |
| <b>Interrupts .....</b>                | <b>134</b> |
| Interrupt Registers.....               | 134        |
| Interrupt Operation.....               | 137        |
| External Interrupt.....                | 138        |
| Proximity Sensing Interrupt.....       | 139        |
| Touch Key Interrupt.....               | 139        |
| A/D Converter Interrupt.....           | 139        |
| UART Interrupt.....                    | 139        |
| TM Interrupts.....                     | 140        |
| EEPROM Interrupt.....                  | 140        |
| Time Base Interrupts.....              | 140        |
| Interrupt Wake-up Function.....        | 141        |
| Programming Considerations.....        | 141        |
| <b>Configuration Options.....</b>      | <b>142</b> |
| <b>Application Circuits.....</b>       | <b>143</b> |
| <b>Instruction Set.....</b>            | <b>144</b> |
| Introduction .....                     | 144        |
| Instruction Timing.....                | 144        |
| Moving and Transferring Data.....      | 144        |
| Arithmetic Operations.....             | 144        |
| Logical and Rotate Operation .....     | 145        |
| Branches and Control Transfer .....    | 145        |
| Bit Operations .....                   | 145        |
| Table Read Operations .....            | 145        |
| Other Operations.....                  | 145        |
| <b>Instruction Set Summary .....</b>   | <b>146</b> |
| Table Conventions.....                 | 146        |
| <b>Instruction Definition.....</b>     | <b>148</b> |

|   |            |
|---|------------|
| <b>Package Information .....</b>                              | <b>157</b> |
| SAW Type 16-pin QFN (4mm×4mm×0.75mm) Outline Dimensions ..... | 158        |
| 16-pin NSOP (150mil) Outline Dimensions .....                 | 159        |
| 24-pin SSOP (150mil) Outline Dimensions .....                 | 160        |

## Features

### CPU Features

- Operating Voltage
  - ♦  $f_{SYS}=8\text{MHz}$ : 1.8V~5.5V
- Up to 0.5 $\mu\text{s}$  instruction cycle with 8MHz system clock at  $V_{DD}=5\text{V}$
- Power down and wake-up functions to reduce power consumption
- Oscillator types
  - ♦ Internal High Speed 8MHz RC – HIRC
  - ♦ Internal Low Speed 32kHz RC – LIRC
- Multi-mode operation: FAST, SLOW, IDLE and SLEEP
- Fully integrated internal oscillators require no external components
- All instructions executed in one or two instruction cycles
- Table read instructions
- 61 powerful instructions
- 6-level subroutine nesting
- Bit manipulation instruction

### Peripheral Features

- Flash Program Memory: 4K $\times$ 16
- RAM Data Memory: 192 $\times$ 8
- True EEPROM Memory: 32 $\times$ 8
- 4 touch key functions – fully integrated without requiring external components
- Watchdog Timer function
- Up to 20 bidirectional I/O lines
- Single external interrupt line shared with I/O pin
- Two Timer Modules for time measure, compare match output, PWM output or single pulse output function
- Dual Time Base functions for generation of fixed time interrupt signals
- Fully-duplex / Half-duplex Universal Asynchronous Receiver and Transmitter Interface – UART
- 8 external channel 12-bit resolution A/D converter with Temperature Sensor and Internal Reference Voltage  $V_R$
- Infrared LED constant current driver with a driving current of up to 384mA (Max.)
- Proximity Sensing Circuit
  - ♦ Two Operational Amplifiers
  - ♦ One Comparator
- Low Voltage Reset function
- Package types: 16-pin QFN, 16-pin NSOP, 24-pin SSOP

## Development Tools

For rapid product development and to simplify device parameter setting, Holtek has provided relevant development tools which users can download from the following link:

[https://www.holtek.com/page/detail/dev\\_plat/IR\\_Sensing\\_Workshop](https://www.holtek.com/page/detail/dev_plat/IR_Sensing_Workshop)

## General Description

The device is a Flash Memory A/D type 8-bit high performance RISC architecture microcontroller with integrated Proximity Sensing and Touch Key functions.

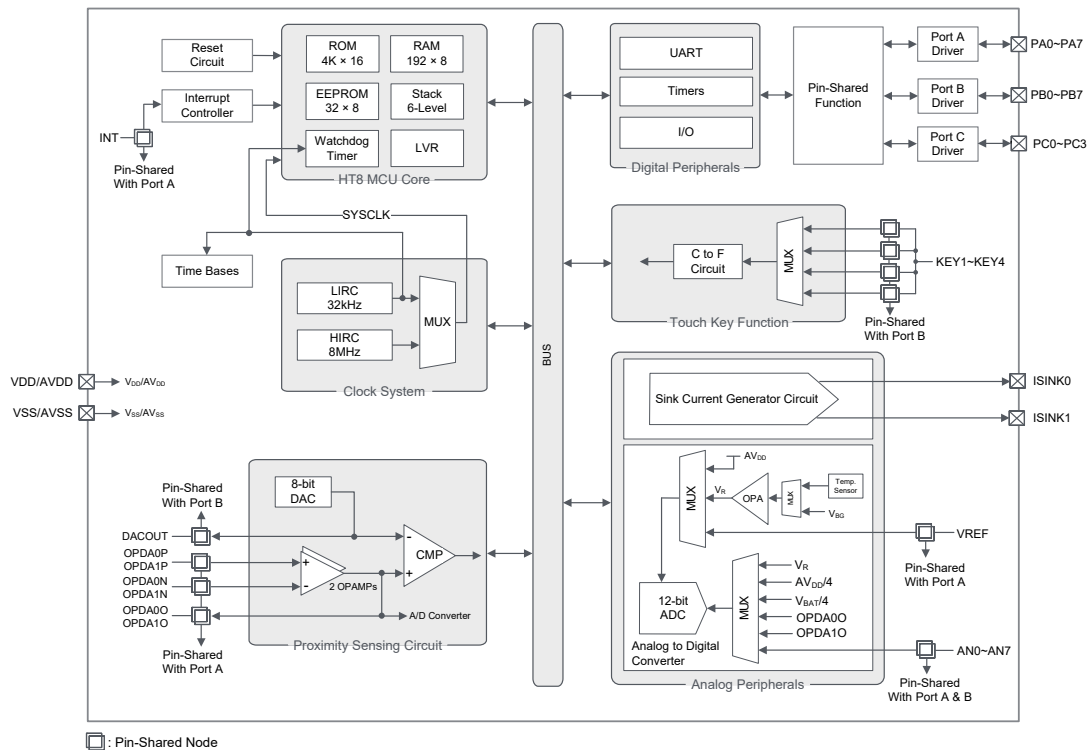
For memory features, the Flash Memory offers users the convenience of multi-programming features. Other memory includes an area of RAM Data Memory as well as an area of true EEPROM memory for storage of non-volatile data such as serial numbers, calibration data etc.

Analog features include a multi-channel 12-bit A/D converter, a Proximity Sensing Current mainly composed of two operational amplifiers and a comparator, and a 2-channel Sink Current Generator. With regard to internal timers, the device includes multiple and extremely flexible Timer Modules providing functions for timing, pulse generation and PWM output operations. Communication with the outside world is catered for by including a fully integrated UART interface function, a popular interface which provides designers with a means of easy communication with external peripheral hardware. Protective features such as an internal Watchdog Timer and Low Voltage Reset coupled with excellent noise immunity and ESD protection ensure that reliable operation is maintained in hostile electrical environments.

The device also includes fully integrated high and low speed oscillators which require no external components for their implementation. The ability to operate and switch dynamically between a range of operating modes using different clock sources gives users the ability to optimise microcontroller operation and minimise power consumption.

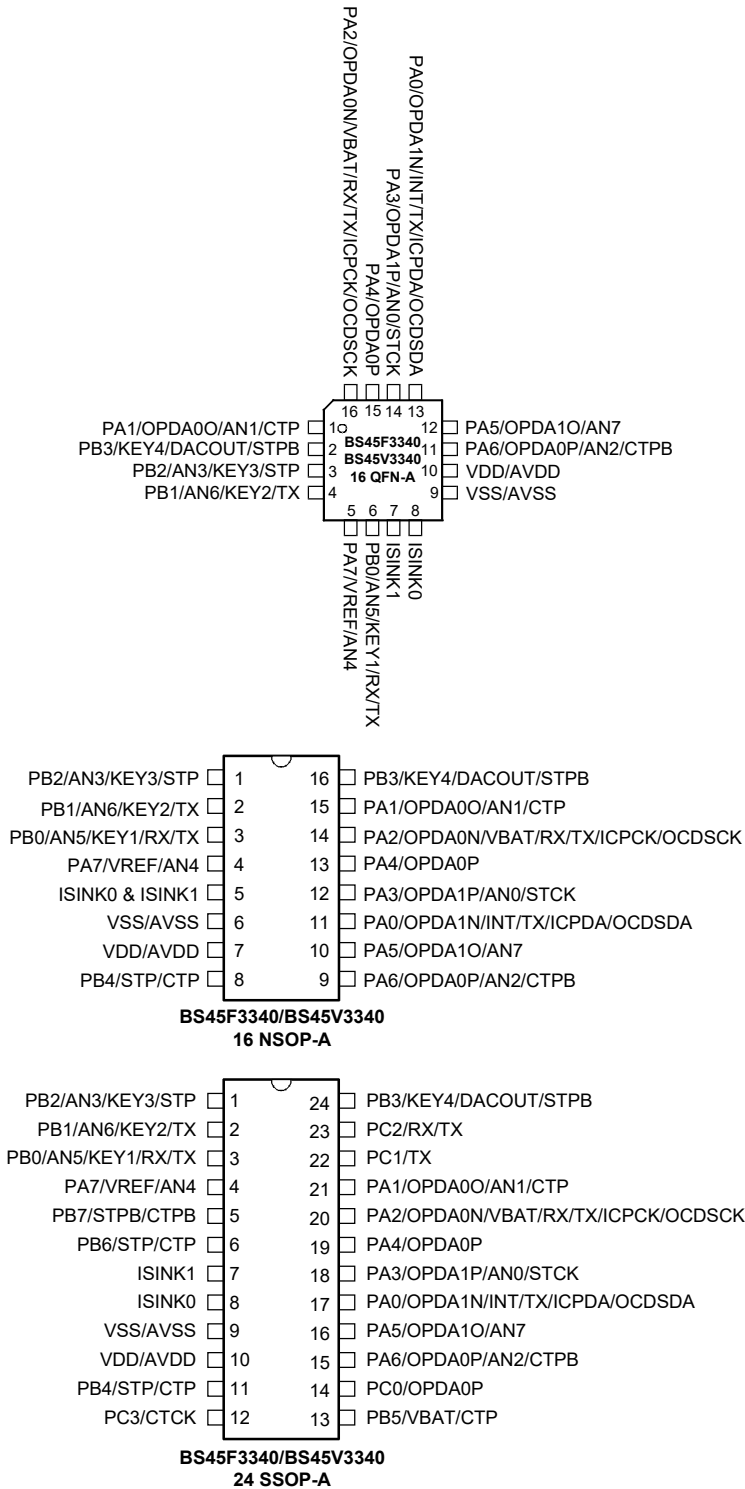
The inclusion of flexible I/O programming features, Time Base functions, touch keys along with many other features ensure that the device will find excellent use in various Proximity Sensing products.

## Block Diagram





**Pin Assignment**



Note: 1. If the pin-shared pin functions have multiple outputs, the desired pin-shared function is determined by the corresponding software control bits.

2. The OCSDSA and OCDSCK pins are supplied as OCDS dedicated pins and as such only available for the BS45V3340 device which is the OCDS EV chip for the BS45F3340 device.
3. When the ISINK0 and ISINK1 pins are bonded to the same pin location (16 NSOP-A) or externally connected together by users (16 QFN-A or 24 SSOP-A), their corresponding internal switches and current control registers should be properly configured to avoid current leakage problem. Refer to the “Sink Current Generator” section.
4. For less pin-count package types there will be unbonded pins which should be properly configured to avoid unwanted current consumption resulting from floating input conditions. Refer to the “Standby Current Considerations” and “Input/Output Ports” sections.

## Pin Description

The function of each pin is listed in the following table, however the details behind how each pin is configured is contained in other sections of the datasheet. As the Pin Description table shows the situation for the package type with the most pins, not all pins in the table will be available on smaller package size.

| Pin Name                               | Function | OPT                    | I/T | O/T  | Description  |
|--|----------|------------------------|-----|------|--|
| PA0/OPDA1N/INT/TX/<br>ICPDA/OCSDSA     | PA0      | PAPU<br>PAWU<br>PAS0   | ST  | CMOS | General purpose I/O. Register enabled pull-high and wake-up  |
|  | OPDA1N   | PAS0                   | AN  | —    | OPAMP1 inverting input   |
|  | INT      | PAS0<br>INTC0<br>INTEG | ST  | —    | External interrupt input   |
|  | TX       | PAS0                   | —   | CMOS | UART serial data output  |
|  | ICPDA    | —                      | ST  | CMOS | ICP data/address   |
|  | OCSDSA   | —                      | ST  | CMOS | OCDS data/address, for EV chip only  |
| PA1/OPDA0O/AN1/CTP                     | PA1      | PAPU<br>PAWU<br>PAS0   | ST  | CMOS | General purpose I/O. Register enabled pull-high and wake-up  |
|  | OPDA0O   | PAS0                   | —   | AN   | OPAMP0 output  |
|  | AN1      | PAS0                   | AN  | —    | A/D Converter external input channel 1   |
|  | CTP      | PAS0                   | —   | CMOS | CTM output   |
| PA2/OPDA0N/VBAT/RX/TX/<br>ICPCK/OCDSCK | PA2      | PAPU<br>PAWU<br>PAS0   | ST  | CMOS | General purpose I/O. Register enabled pull-high and wake-up  |
|  | OPDA0N   | PAS0                   | AN  | —    | OPAMP0 inverting input   |
|  | VBAT     | PAS0                   | AN  | —    | A/D Converter external input   |
|  | RX/TX    | PAS0<br>IFS            | ST  | CMOS | UART serial data input in full-duplex communication or UART serial data input / output in single wire mode communication |
|  | ICPCK    | —                      | ST  | —    | ICP clock pin  |
|  | OCDSCK   | —                      | ST  | —    | OCDS clock pin, for EV chip only   |
| PA3/OPDA1P/AN0/STCK                    | PA3      | PAPU<br>PAWU<br>PAS0   | ST  | CMOS | General purpose I/O. Register enabled pull-high and wake-up  |
|  | OPDA1P   | PAS0                   | AN  | —    | OPAMP1 Non-inverting input   |
|  | AN0      | PAS0                   | AN  | —    | A/D Converter external input channel 0   |
|  | STCK     | PAS0                   | ST  | —    | STM clock input  |
| PA4/OPDA0P                             | PA4      | PAPU<br>PAWU<br>PAS1   | ST  | CMOS | General purpose I/O. Register enabled pull-high and wake-up  |
|  | OPDA0P   | PAS1                   | AN  | —    | OPAMP0 Non-inverting input   |

| Pin Name             | Function | OPT                  | I/T | O/T  | Description  |
|----------------------|----------|----------------------|-----|------|--|
| PA5/OPDA10/AN7       | PA5      | PAPU<br>PAWU<br>PAS1 | ST  | CMOS | General purpose I/O. Register enabled pull-high and wake-up  |
|                      | OPDA10   | PAS1                 | —   | AN   | OPAMP1 output  |
|                      | AN7      | PAS1                 | AN  | —    | A/D Converter external input channel 7   |
| PA6/OPDA0P/AN2/CTPB  | PA6      | PAPU<br>PAWU<br>PAS1 | ST  | CMOS | General purpose I/O. Register enabled pull-high and wake-up  |
|                      | OPDA0P   | PAS1                 | AN  | —    | OPAMP0 Non-inverting input   |
|                      | AN2      | PAS1                 | AN  | —    | A/D Converter external input channel 2   |
|                      | CTPB     | PAS1                 | —   | CMOS | CTM inverting output   |
| PA7/VREF/AN4         | PA7      | PAPU<br>PAWU<br>PAS1 | ST  | CMOS | General purpose I/O. Register enabled pull-high and wake-up  |
|                      | VREF     | PAS1                 | AN  | —    | A/D Converter external reference input   |
|                      | AN4      | PAS1                 | AN  | —    | A/D Converter external input channel 4   |
| PB0/AN5/KEY1/RX/TX   | PB0      | PBPU<br>PBS0         | ST  | CMOS | General purpose I/O. Register enabled pull-high  |
|                      | AN5      | PBS0                 | AN  | —    | A/D Converter external input channel 5   |
|                      | KEY1     | PBS0                 | AN  | —    | Touch key input  |
|                      | RX/TX    | PBS0<br>IFS          | ST  | CMOS | UART serial data input in full-duplex communication or UART serial data input / output in single wire mode communication |
| PB1/AN6/KEY2/TX      | PB1      | PBPU<br>PBS0         | ST  | CMOS | General purpose I/O. Register enabled pull-high  |
|                      | AN6      | PBS0                 | AN  | —    | A/D Converter external input channel 6   |
|                      | KEY2     | PBS0                 | AN  | —    | Touch key input  |
|                      | TX       | PBS0                 | —   | CMOS | UART serial data output  |
| PB2/AN3/KEY3/STP     | PB2      | PBPU<br>PBS0         | ST  | CMOS | General purpose I/O. Register enabled pull-high  |
|                      | AN3      | PBS0                 | AN  | —    | A/D Converter external input channel 3   |
|                      | KEY3     | PBS0                 | AN  | —    | Touch key input  |
|                      | STP      | PBS0                 | —   | CMOS | STM output   |
| PB3/KEY4/DACOUT/STPB | PB3      | PBPU<br>PBS0         | ST  | CMOS | General purpose I/O. Register enabled pull-high  |
|                      | KEY4     | PBS0                 | AN  | —    | Touch key input  |
|                      | DACOUT   | PBS0                 | —   | AN   | DAC output   |
|                      | STPB     | PBS0                 | —   | CMOS | STM inverting output   |
| PB4/STP/CTP          | PB4      | PBPU<br>PBS1         | ST  | CMOS | General purpose I/O. Register enabled pull-high  |
|                      | STP      | PBS1                 | —   | CMOS | STM output   |
|                      | CTP      | PBS1                 | —   | CMOS | CTM output   |
| PB5/VBAT/CTP         | PB5      | PBPU<br>PBS1         | ST  | CMOS | General purpose I/O. Register enabled pull-high  |
|                      | VBAT     | PBS1                 | AN  | —    | A/D Converter external input   |
|                      | CTP      | PBS1                 | —   | CMOS | CTM output   |
| PB6/STP/CTP          | PB6      | PBPU<br>PBS1         | ST  | CMOS | General purpose I/O. Register enabled pull-high  |
|                      | STP      | PBS1                 | —   | CMOS | STM output   |
|                      | CTP      | PBS1                 | —   | CMOS | CTM output   |

| Pin Name      | Function | OPT          | I/T | O/T  | Description  |
|---------------|----------|--------------|-----|------|--|
| PB7/STPB/CTPB | PB7      | PBPU<br>PBS1 | ST  | CMOS | General purpose I/O. Register enabled pull-high  |
|               | STPB     | PBS1         | —   | CMOS | STM inverting output   |
|               | CTPB     | PBS1         | —   | CMOS | CTM inverting output   |
| PC0/OPDA0P    | PC0      | PCPU<br>PCS0 | ST  | CMOS | General purpose I/O. Register enabled pull-high  |
|               | OPDA0P   | PCS0         | AN  | —    | OPAMP0 non-inverting input   |
| PC1/TX        | PC1      | PCPU<br>PCS0 | ST  | CMOS | General purpose I/O. Register enabled pull-high  |
|               | TX       | PCS0         | —   | CMOS | UART serial data output  |
| PC2/RX/TX     | PC2      | PCPU<br>PCS0 | ST  | CMOS | General purpose I/O. Register enabled pull-high  |
|               | RX/TX    | PCS0<br>IFS  | ST  | CMOS | UART serial data input in full-duplex communication or UART serial data input / output in single wire mode communication |
| PC3/CTCK      | PC3      | PCPU         | ST  | CMOS | General purpose I/O. Register enabled pull-high  |
|               | CTCK     | —            | ST  | —    | CTM clock input  |
| ISINK0        | ISINK0   | —            | —   | AN   | Sink0 current source   |
| ISINK1        | ISINK1   | —            | —   | AN   | Sink1 current source   |
| VDD/AVDD      | VDD      | —            | PWR | —    | Digital positive power supply  |
|               | AVDD     | —            | PWR | —    | Analog positive power supply   |
| VSS/AVSS      | VSS      | —            | PWR | —    | Digital negative power supply  |
|               | AVSS     | —            | PWR | —    | Analog negative power supply   |

Legend: I/T: Input type;

OPT: Optional by register option;

ST: Schmitt Trigger input;

AN: Analog signal.

O/T: Output type;

PWR: Power;

CMOS: CMOS output;

## Absolute Maximum Ratings

|                               |                                  |
|-------------------------------|----------------------------------|
| Supply Voltage .....          | $V_{SS}-0.3V$ to $6.0V$          |
| Input Voltage .....           | $V_{SS}-0.3V$ to $V_{DD}+0.3V$   |
| Storage Temperature.....      | $-60^{\circ}C$ to $150^{\circ}C$ |
| Operating Temperature.....    | $-40^{\circ}C$ to $85^{\circ}C$  |
| $I_{OL}$ Total .....          | 80mA                             |
| $I_{OH}$ Total .....          | -80mA                            |
| Total Power Dissipation ..... | 500mW                            |

Note: These are stress ratings only. Stresses exceeding the range specified under “Absolute Maximum Ratings” may cause substantial damage to the device. Functional operation of the device at other conditions beyond those listed in the specification is not implied and prolonged exposure to extreme conditions may affect devices reliability.

## D.C. Characteristics

For data in the following tables, note that factors such as oscillator type, operating voltage, operating frequency, pin load conditions, temperature and program instruction type, etc., can all exert an influence on the measured values.

### Operating Voltage Characteristics

Ta=-40°C~85°C

| Symbol          | Parameter                | Test Conditions                            | Min. | Typ. | Max. | Unit |
|-----------------|--------------------------|--|------|------|------|------|
| V <sub>DD</sub> | Operating Voltage – HIRC | f <sub>sys</sub> =f <sub>HIRC</sub> =8MHz  | 1.8  | —    | 5.5  | V    |
|                 | Operating Voltage – LIRC | f <sub>sys</sub> =f <sub>LIRC</sub> =32kHz | 1.8  | —    | 5.5  | V    |

### Operating Current Characteristics

Ta=25°C

| Symbol          | Operating Mode   | Test Conditions |                         | Min. | Typ. | Max. | Unit |
|-----------------|------------------|-----------------|-------------------------|------|------|------|------|
|                 |                  | V <sub>DD</sub> | Conditions              |      |      |      |      |
| I <sub>DD</sub> | SLOW Mode – LIRC | 1.8V            | f <sub>sys</sub> =32kHz | —    | 8    | 16   | μA   |
|                 |                  | 3V              |                         | —    | 10   | 20   |      |
|                 |                  | 5V              |                         | —    | 30   | 50   |      |
|                 | FAST Mode – HIRC | 1.8V            | f <sub>sys</sub> =8MHz  | —    | 0.6  | 1.0  | mA   |
|                 |                  | 3V              |                         | —    | 0.8  | 1.2  |      |
|                 |                  | 5V              |                         | —    | 1.6  | 2.4  |      |

Note: When using the characteristic table data, the following notes should be taken into consideration:

1. Any digital inputs are setup in a non-floating condition.
2. All measurements are taken under conditions of no load and with all peripherals in an off state.
3. There are no DC current paths.
4. All Operating Current values are measured using a continuous NOP instruction program loop.

### Standby Current Characteristics

Ta=25°C, unless otherwise specified

| Symbol           | Standby Mode      | Test Conditions |   | Min. | Typ. | Max. | Max. @85°C | Unit |
|------------------|-------------------|-----------------|---|------|------|------|------------|------|
|                  |                   | V <sub>DD</sub> | Conditions                                  |      |      |      |            |      |
| I <sub>STB</sub> | SLEEP Mode        | 1.8V            | WDT off                                     | —    | 0.11 | 0.15 | 2.00       | μA   |
|                  |                   | 3V              |   | —    | 0.11 | 0.15 | 2.00       |      |
|                  |                   | 5V              |   | —    | 0.18 | 0.38 | 2.90       |      |
|                  | IDLE0 Mode – LIRC | 1.8V            | f <sub>sub</sub> on                         | —    | 2.4  | 4.0  | 4.8        | μA   |
|                  |                   | 3V              |   | —    | 3    | 5    | 6          |      |
|                  |                   | 5V              |   | —    | 5    | 10   | 12         |      |
|                  | IDLE1 Mode – HIRC | 1.8V            | f <sub>sub</sub> on, f <sub>sys</sub> =8MHz | —    | 288  | 400  | 480        | μA   |
|                  |                   | 3V              |   | —    | 360  | 500  | 600        |      |
|                  |                   | 5V              |   | —    | 600  | 800  | 960        |      |

Note: When using the characteristic table data, the following notes should be taken into consideration:

1. Any digital inputs are setup in a non-floating condition.
2. All measurements are taken under conditions of no load and with all peripherals in an off state.
3. There are no DC current paths.
4. All Standby Current values are taken after a HALT instruction execution thus stopping all instruction execution.

## A.C. Characteristics

For data in the following tables, note that factors such as oscillator type, operating voltage, operating frequency and temperature etc., can all exert an influence on the measured values.

### High Speed Internal Oscillator – HIRC – Frequency Accuracy

During the program writing operation the writer will trim the HIRC oscillator at a user-selected voltage of either 3V or 5V.

| Symbol            | Parameter                          | Test Conditions |            | Min.  | Typ. | Max.  | Unit |
|-------------------|------------------------------------|-----------------|------------|-------|------|-------|------|
|                   |                                    | V <sub>DD</sub> | Temp.      |       |      |       |      |
| f <sub>HIRC</sub> | 8MHz Writer Trimmed HIRC Frequency | 3V/5V           | 25°C       | -1%   | 8    | +1%   | MHz  |
|                   |                                    |                 | -40°C~85°C | -2.5% | 8    | +2.5% |      |
|                   |                                    | 2.2V~5.5V       | 25°C       | -2.5% | 8    | +2.5% |      |
|                   |                                    |                 | -40°C~85°C | -3%   | 8    | +3%   |      |
|                   |                                    | 1.8V~5.5V       | 25°C       | -5%   | 8    | +3%   |      |
|                   |                                    |                 | -40°C~85°C | -10%  | 8    | +5%   |      |

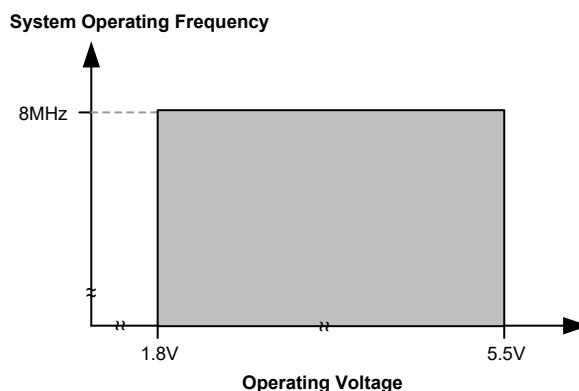
Note: 1. The 3V/5V values for V<sub>DD</sub> are provided as these are the two selectable fixed voltages at which the HIRC frequency is trimmed by the writer.

2. The row below the 3V/5V trim voltage row is provided to show the values for the full V<sub>DD</sub> range operating voltage. It is recommended that the trim voltage is fixed at 3V for application voltage ranges from 1.8V to 3.6V and fixed at 5V for application voltage ranges from 3.3V to 5.5V.

### Low Speed Internal Oscillator – LIRC – Frequency Accuracy

| Symbol             | Parameter          | Test Conditions |            | Min. | Typ. | Max. | Unit |
|--------------------|--------------------|-----------------|------------|------|------|------|------|
|                    |                    | V <sub>DD</sub> | Temp.      |      |      |      |      |
| f <sub>LIRC</sub>  | LIRC Frequency     | 2.2V~5.5V       | -40°C~85°C | -7%  | 32   | +7%  | kHz  |
| t <sub>START</sub> | LIRC Start-up Time | —               | -40°C~85°C | —    | —    | 100  | μs   |

### Operating Frequency Characteristic Curves



### System Start Up Time Characteristics

Ta=-40°C~85°C

| Symbol              | Parameter   | Test Conditions   | Min. | Typ. | Max. | Unit              |
|---------------------|---|---|------|------|------|-------------------|
| t <sub>SST</sub>    | System Start-up Time<br>(Wake-up from Condition where f <sub>sys</sub> is off)      | f <sub>sys</sub> =f <sub>H</sub> ~f <sub>H</sub> /64, f <sub>H</sub> =f <sub>HIRC</sub> | —    | 16   | —    | t <sub>HIRC</sub> |
|                     |   | f <sub>sys</sub> =f <sub>SUB</sub> =f <sub>LIRC</sub>                                   | —    | 2    | —    | t <sub>LIRC</sub> |
|                     | System Start-up Time<br>(Wake-up from Condition where f <sub>sys</sub> is on)       | f <sub>sys</sub> =f <sub>H</sub> ~f <sub>H</sub> /64, f <sub>H</sub> =f <sub>HIRC</sub> | —    | 2    | —    | t <sub>H</sub>    |
|                     |   | f <sub>sys</sub> =f <sub>SUB</sub> =f <sub>LIRC</sub>                                   | —    | 2    | —    | t <sub>SUB</sub>  |
|                     | System Speed Switch Time<br>(FAST to SLOW Mode or SLOW to FAST Mode)                | f <sub>HIRC</sub> switches from off → on  | —    | 16   | —    | t <sub>HIRC</sub> |
| t <sub>RSTD</sub>   | System Reset Delay Time<br>(Reset source from Power-on reset or LVR Hardware Reset) | RR <sub>POR</sub> =5V/ms  | 14   | 16   | 18   | ms                |
|                     | System Reset Delay Time<br>(LVRC/WDT Register Software Reset)                       | —   |      |      |      |                   |
|                     | System Reset Delay Time<br>(WDT Overflow Reset)                                     | —   | 14   | 16   | 18   |                   |
| t <sub>SRESET</sub> | Minimum Software Reset Width to Reset   | —   | 45   | 90   | 120  | μs                |

- Note: 1. For the System Start-up time values, whether f<sub>sys</sub> is on or off depends upon the mode type and the chosen f<sub>sys</sub> system oscillator. Details are provided in the System Operating Modes section.
2. The time units, shown by the symbols t<sub>HIRC</sub> etc., are the inverse of the corresponding frequency values as provided in the frequency tables. For example, t<sub>HIRC</sub>=1/f<sub>HIRC</sub>, t<sub>sys</sub>=1/f<sub>sys</sub> etc.
3. If the LIRC is used as the system clock and if it is off when in the SLEEP Mode, then an additional LIRC start up time, t<sub>START</sub>, as provided in the LIRC frequency table, must be added to the t<sub>SST</sub> time in the table above.
4. The System Speed Switch Time is effectively the time taken for the newly activated oscillator to start up.

### Input/Output Characteristics

Ta=-40°C~85°C

| Symbol          | Parameter   | Test Conditions |   | Min.               | Typ. | Max.               | Unit |
|-----------------|---|-----------------|---|--------------------|------|--------------------|------|
|                 |   | V <sub>DD</sub> | Conditions  |                    |      |                    |      |
| V <sub>IL</sub> | Input Low Voltage for I/O Ports                                   | 5V              | —   | 0                  | —    | 1.5                | V    |
|                 |   | —               |   | 0                  | —    | 0.2V <sub>DD</sub> |      |
| V <sub>IH</sub> | Input High Voltage for I/O Ports                                  | 5V              | —   | 3.5                | —    | 5.0                | V    |
|                 |   | —               |   | 0.8V <sub>DD</sub> | —    | V <sub>DD</sub>    |      |
| I <sub>OL</sub> | Sink Current for I/O Ports (except PB4)                           | 3V              | V <sub>OL</sub> =0.1V <sub>DD</sub>                           | 16                 | 32   | —                  | mA   |
|                 |   | 5V              |   | 32                 | 65   | —                  |      |
|                 | Sink Current for PB4  | 3V              | V <sub>OL</sub> =0.1V <sub>DD</sub>                           | 16                 | 32   | —                  |      |
|                 |   | 5V              |   | 40                 | 80   | —                  |      |
| I <sub>OH</sub> | Source Current for I/O Ports (except PB4)                         | 3V              | V <sub>OH</sub> =0.9V <sub>DD</sub>                           | -4                 | -8   | —                  | mA   |
|                 |   | 5V              |   | -8                 | -16  | —                  |      |
|                 | Source Current for PB4  | 3V              | V <sub>OH</sub> =0.9V <sub>DD</sub>                           | -16                | -32  | —                  |      |
|                 |   | 5V              |   | -40                | -80  | —                  |      |
| R <sub>PH</sub> | Pull-high Resistance for I/O Ports<br>(except PB4) <sup>(1)</sup> | 3V              | LVPU=0, P <sub>x</sub> PU=FFH<br>(P <sub>x</sub> =PA, PB, PC) | 20                 | 60   | 100                | kΩ   |
|                 |   | 5V              |   | 10                 | 30   | 50                 |      |
|                 |   | 3V              | LVPU=1, P <sub>x</sub> PU=FFH<br>(P <sub>x</sub> =PA, PB, PC) | 6.67               | 15   | 23                 |      |
|                 |   | 5V              |   | 3.5                | 7.5  | 12                 |      |
|                 | Pull-high Resistance for PB4 <sup>(1)</sup>                       | 3V              | —   | 20                 | 60   | 100                |      |
|                 |   | 5V              |   | 10                 | 30   | 50                 |      |

| Symbol            | Parameter                                  | Test Conditions |  | Min. | Typ. | Max. | Unit |
|-------------------|--|-----------------|--|------|------|------|------|
|                   |  | V <sub>DD</sub> | Conditions   |      |      |      |      |
| R <sub>PL</sub>   | Pull-low Resistance for PB4 <sup>(2)</sup> | 3V              | —  | 20   | 30   | 40   | kΩ   |
|                   |  | 5V              |  | 20   | 30   | 40   |      |
| I <sub>LEAK</sub> | Input Leakage Current                      | 5V              | V <sub>IN</sub> =V <sub>DD</sub> (only when R <sub>PL</sub> does not exist or is disabled) or V <sub>IN</sub> =V <sub>SS</sub> | —    | —    | ±1   | μA   |
| t <sub>TCK</sub>  | xTM xTCK Input Pin Minimum Pulse Width     | —               | —  | 0.3  | —    | —    | μs   |
| t <sub>INT</sub>  | Interrupt Pin Minimum Pulse Width          | —               | —  | 10   | —    | —    | μs   |

Note: 1. The R<sub>PH</sub> internal pull-high resistance value is calculated by connecting to ground and enabling the input pin with a pull-high resistor and then measuring the pin current at the specified supply voltage level. Dividing the voltage by this measured current provides the R<sub>PH</sub> value.

2. The R<sub>PL</sub> internal pull-low resistance value is calculated by connecting to V<sub>DD</sub> and enabling input pin with pull-low resistor and then measuring the pin current at the specified supply voltage level. Dividing the voltage by this measured current provides the R<sub>PL</sub> value.

## Memory Characteristics

Ta=-40°C~85°C

| Symbol                    | Parameter                   | Test Conditions |            | Min. | Typ. | Max. | Unit             |
|---------------------------|-----------------------------|-----------------|------------|------|------|------|------------------|
|                           |                             | V <sub>DD</sub> | Conditions |      |      |      |                  |
| <b>Data EEPROM Memory</b> |                             |                 |            |      |      |      |                  |
| V <sub>DD</sub>           | Operating Voltage for Read  | —               | —          | 1.8  | —    | 5.5  | V                |
|                           | Operating Voltage for Write | —               | —          | 2.2  | —    | 5.5  |                  |
| t <sub>EERD</sub>         | Read Cycle Time             | —               | —          | —    | —    | 4    | t <sub>sys</sub> |
| t <sub>EEWR</sub>         | Write Cycle Time            | 3.0V~5.5V       | —          | —    | 4    | 6    | ms               |

## LVR Electrical Characteristics

Ta=-40°C~85°C

| Symbol             | Parameter                          | Test Conditions |                                    | Min. | Typ. | Max. | Unit |
|--------------------|------------------------------------|-----------------|------------------------------------|------|------|------|------|
|                    |                                    | V <sub>DD</sub> | Conditions                         |      |      |      |      |
| V <sub>DD</sub>    | Operating Voltage                  | —               | —                                  | 1.8  | —    | 5.5  | V    |
| V <sub>LVR</sub>   | Low Voltage Reset Voltage          | —               | LVR enable                         | -5%  | 1.7  | +5%  | V    |
| I <sub>LVRBG</sub> | Operating Current                  | 3V              | LVR enable, V <sub>LVR</sub> =1.7V | —    | —    | 15   | μA   |
|                    |                                    | 5V              |                                    | —    | 15   | 25   |      |
| t <sub>LVR</sub>   | Minimum Low Voltage Width to Reset | —               | —                                  | 120  | 240  | 480  | μs   |
| I <sub>LVR</sub>   | Additional Current for LVR Enable  | 5V              | VBGEN=0                            | —    | —    | 25   | μA   |

## Internal Reference Voltage Characteristics

Ta=-40°C~85°C

| Symbol           | Parameter                                       | Test Conditions |                      | Min. | Typ. | Max. | Unit |
|------------------|---|-----------------|----------------------|------|------|------|------|
|                  |   | V <sub>DD</sub> | Conditions           |      |      |      |      |
| t <sub>BGS</sub> | V <sub>BG</sub> Turn On Stable Time             | —               | No load              | —    | —    | 50   | μs   |
| I <sub>BG</sub>  | Additional Current for Bandgap Reference Enable | —               | VBGEN=1, LVR disable | —    | —    | 2    | μA   |

Note: The V<sub>BG</sub> voltage is used as the A/D converter operational amplifier input.



## A/D Converter Electrical Characteristics

Ta=-40°C~85°C, unless otherwise specified

| Symbol             | Parameter  | Test Conditions |  | Min.                 | Typ. | Max.                 | Unit              |
|--------------------|--|-----------------|--|----------------------|------|----------------------|-------------------|
|                    |  | V <sub>DD</sub> | Conditions   |                      |      |                      |                   |
| V <sub>DD</sub>    | Operating Voltage                                    | —               | —  | 1.8                  | —    | 5.5                  | V                 |
| V <sub>ADI</sub>   | Input Voltage  | —               | —  | 0                    | —    | V <sub>REF</sub>     | V                 |
| V <sub>REF</sub>   | Reference Voltage                                    | —               | —  | 1.6                  | —    | V <sub>DD</sub>      | V                 |
| N <sub>R</sub>     | Resolution   | —               | —  | —                    | —    | 12                   | Bit               |
| DNL                | Differential Non-linearity                           | —               | V <sub>REF</sub> =V <sub>DD</sub> , t <sub>ADCK</sub> =0.5μs | -3                   | —    | 3                    | LSB               |
| INL                | Integral Non-linearity                               | —               | V <sub>REF</sub> =V <sub>DD</sub> , t <sub>ADCK</sub> =0.5μs | -4                   | —    | 4                    | LSB               |
| I <sub>ADC</sub>   | Additional Current for A/D Converter Enable          | 1.8V            | No load, t <sub>ADCK</sub> =0.5μs                            | —                    | 300  | 420                  | μA                |
|                    |  | 3V              |  | —                    | 340  | 500                  |                   |
|                    |  | 5V              |  | —                    | 500  | 700                  |                   |
| t <sub>ADCK</sub>  | Clock Period   | —               | A/D Converter input ≠ Temperature Sensor signal              | 0.5                  | —    | 10.0                 | μs                |
|                    |  | —               | A/D Converter input = Temperature Sensor signal              | 1                    | —    | 2                    |                   |
| t <sub>ON2ST</sub> | A/D Converter On-to-Start Time                       | —               | —  | 4                    | —    | —                    | μs                |
| t <sub>ADS</sub>   | Sampling Time  | —               | —  | —                    | 4    | —                    | t <sub>ADCK</sub> |
| t <sub>ADC</sub>   | Conversion Time (Including A/D Sample and Hold Time) | —               | —  | —                    | 16   | —                    | t <sub>ADCK</sub> |
| GERR               | A/D Conversion Gain Error                            | —               | V <sub>REF</sub> =V <sub>DD</sub>                            | -4                   | —    | 4                    | LSB               |
| OSRR               | A/D Conversion Offset Error                          | —               | V <sub>REF</sub> =V <sub>DD</sub>                            | -4                   | —    | 4                    | LSB               |
| I <sub>PGA</sub>   | Additional Current for OPA Enable                    | 3V              | No load  | —                    | 390  | 550                  | μA                |
|                    |  | 5V              |  | —                    | 500  | 650                  |                   |
| V <sub>OR</sub>    | OPA Maximum Output Voltage Range                     | 3V              | —  | V <sub>SS</sub> +0.1 | —    | V <sub>DD</sub> -0.1 | V                 |
|                    |  | 5V              |  | V <sub>SS</sub> +0.1 | —    | V <sub>DD</sub> -0.1 |                   |
| V <sub>R</sub>     | OPA Fix Voltage Output                               | 1.8V~5.5V       | Ta=25°C  | -2.5%                | 1.6  | +2.5%                | V                 |
|                    |  |                 | Ta=-40°C~85°C  | -10%                 | 1.6  | +10%                 |                   |
|                    |  | 3.2V~5.5V       | Ta=25°C  | -2.5%                | 3.0  | +2.5%                |                   |
|                    |  |                 | Ta=-40°C~85°C  | -10%                 | 3.0  | +10%                 |                   |
| 4.2V~5.5V          | Ta=25°C  | -2.5%           | 4.0  | +2.5%                |      |                      |                   |
|                    | Ta=-40°C~85°C  | -10%            | 4.0  | +10%                 |      |                      |                   |

## Temperature Sensor Characteristics

Ta=-40°C~85°C, unless otherwise specified

| Symbol                 | Parameter  | Test Conditions |                                     | Min. | Typ. | Max. | Unit  |
|------------------------|--|-----------------|-------------------------------------|------|------|------|-------|
|                        |  | V <sub>DD</sub> | Conditions                          |      |      |      |       |
| V <sub>DD</sub>        | Operating Voltage  | —               | —                                   | 1.8  | —    | 5.5  | V     |
| I <sub>TS</sub>        | Additional Current for Temperature Sensor Enable                       | 3V              | TSEN=0→1 (ADC and OPA not included) | —    | 20   | 50   | μA    |
|                        |  | 5V              |                                     | —    | 20   | 50   |       |
| t <sub>TSS</sub>       | Temperature Sensor Turn On Stable Time                                 | —               | —                                   | —    | —    | 150  | μs    |
| t <sub>VR_SW</sub>     | V <sub>R</sub> Stable Time When SATS Switching                         | —               | SATS=0→1 or 1→0                     | —    | —    | 150  | μs    |
| T <sub>OS_ERR</sub>    | Temperature Sensor Error at Normal Temperature Trimming <sup>(1)</sup> | 3V/5V           | TS_SEL=0 @ 3V                       | -2.0 | —    | 2.0  | °C    |
|                        |  |                 | TS_SEL=1 @ 5V                       | —    | —    | —    |       |
|                        |  | 2V~5V           | —                                   | -15  | —    | 15   |       |
| T <sub>S_L_P_ERR</sub> | Temperature Slope Error <sup>(2)</sup>                                 | 2V~5V           | Ta=-20°C~70°C                       | -30% | —    | 30%  | °C/°C |

| Symbol | Parameter                   | Test Conditions |                                  | Min. | Typ. | Max. | Unit |
|--------|-----------------------------|-----------------|----------------------------------|------|------|------|------|
|        |                             | V <sub>DD</sub> | Conditions                       |      |      |      |      |
| TLE    | Temperature Linearity Error | 2V~5V           | Ta=-20°C~70°C<br>Best fit method | -10  | —    | +10  | °C   |

Note: 1. “Normal Temperature” means the ambient temperature when calibrating by writer. It is recommended to calibrate at 25°C±5°C. Temperature is calculated as follows:

$$T_a(^{\circ}\text{C}) = 655 \times \frac{\text{ADC}_{\text{TS}}}{\text{ADC}_{\text{REF}}} - 460 + T_{\text{OS}}$$

The 3V/5V values for V<sub>DD</sub> are provided as these are the two selectable fixed voltages at which the temperature sensor is trimmed by the writer.

2. The temperature calculation slope error is calculated as follows:

$$\text{TS Slope Error} = \frac{\text{Measured Temp. Value 1} - \text{Measured Temp. Value 2}}{\text{Ambient Temp. Value 1} - \text{Ambient Temp. Value 2}} \times 100\% - 100\%$$

## Proximity Sensing AFE Electrical Characteristics

### Comparator Characteristics

T<sub>a</sub>=-40°C~85°C

| Symbol           | Parameter  | Test Conditions |   | Min.            | Typ. | Max.               | Unit |
|------------------|--|-----------------|---|-----------------|------|--------------------|------|
|                  |  | V <sub>DD</sub> | Conditions                                  |                 |      |                    |      |
| V <sub>DD</sub>  | Operating Voltage                                | —               | —   | 1.8             | 5.0  | 5.5                | V    |
| I <sub>CMP</sub> | Operating Current<br>(I <sub>DAC</sub> included) | —               | No load, OPDIS[1:0]=00B                     | —               | 255  | 310                | μA   |
|                  |  | —               | No load, OPDIS[1:0]=01B                     | —               | 275  | 340                |      |
|                  |  | —               | No load, OPDIS[1:0]=10B                     | —               | 300  | 380                |      |
|                  |  | —               | No load, OPDIS[1:0]=11B                     | —               | 330  | 420                |      |
| V <sub>OS</sub>  | Input Offset Voltage                             | 5V              | Without calibration<br>(OPDCOF[4:0]=10000B) | -10             | —    | +10                | mV   |
|                  |  |                 | With calibration, V <sub>IN</sub> ≥250mV    | -4              | —    | +4                 |      |
| V <sub>CM</sub>  | Common Mode Voltage Range                        | —               | —   | V <sub>SS</sub> | —    | V <sub>DD</sub> -1 | V    |
| t <sub>RP</sub>  | Response Time                                    | 3V              | With 10mV overdrive,<br>OPDIS[1:0]=00B      | —               | —    | 35                 | μs   |
|                  |  | 5V              |   | —               | —    | 35                 |      |
|                  |  | 3V              | With 10mV overdrive,<br>OPDIS[1:0]=01B      | —               | —    | 2.5                |      |
|                  |  | 5V              |   | —               | —    | 2.5                |      |
|                  |  | 3V              | With 10mV overdrive,<br>OPDIS[1:0]=10B      | —               | —    | 1                  |      |
|                  |  | 5V              |   | —               | —    | 1                  |      |
|                  |  | 3V              | With 10mV overdrive,<br>OPDIS[1:0]=11B      | —               | —    | 0.7                |      |
|                  |  | 5V              |   | —               | —    | 0.7                |      |
| V <sub>HYS</sub> | Hysteresis                                       | 3V              | OPDHYS[1:0]=00B,<br>OPDIS[1:0]=00B          | 0               | 0    | 5                  | mV   |
|                  |  | 5V              |   | 0               | 0    | 5                  |      |
|                  |  | 3V              | OPDHYS[1:0]=01B,<br>OPDIS[1:0]=01B          | 20              | 40   | 60                 |      |
|                  |  | 5V              |   | 20              | 40   | 60                 |      |
|                  |  | 3V              | OPDHYS[1:0]=10B,<br>OPDIS[1:0]=10B          | 50              | 100  | 150                |      |
|                  |  | 5V              |   | 50              | 100  | 150                |      |
|                  |  | 3V              | OPDHYS[1:0]=11B,<br>OPDIS[1:0]=11B          | 80              | 160  | 240                |      |
|                  |  | 5V              |   | 80              | 160  | 240                |      |

### Operational Amplifier Characteristics

Ta=-40°C~85°C

| Symbol           | Parameter                         | Test Conditions |  | Min.                    | Typ. | Max.                    | Unit |
|------------------|-----------------------------------|-----------------|--|-------------------------|------|-------------------------|------|
|                  |                                   | V <sub>DD</sub> | Conditions   |                         |      |                         |      |
| V <sub>DD</sub>  | Operating Voltage                 | —               | —  | 1.8                     | 5.0  | 5.5                     | V    |
| I <sub>OPA</sub> | Additional Current for OPA enable | —               | No load, OPDAnBW=0 (n=0, 1)  | —                       | 80   | 200                     | μA   |
|                  |                                   |                 | No load, OPDAnBW=1 (n=0, 1)  | —                       | 230  | 510                     |      |
| V <sub>OS</sub>  | Input Offset Voltage              | 5V              | Without calibration<br>(OPDAnOf[5:0]=100000B; n=0, 1)                  | -15                     | —    | +15                     | mV   |
|                  |                                   |                 | With calibration   | -2                      | —    | +2                      |      |
| I <sub>OS</sub>  | Input Offset Current              | 5V              | V <sub>IN</sub> =1/2 V <sub>CM</sub>                                   | —                       | 1    | 10                      | nA   |
| V <sub>CM</sub>  | Common Mode Voltage Range         | —               | —  | V <sub>SS</sub>         | —    | V <sub>DD</sub> -1.4    | V    |
| PSRR             | Power Supply Rejection Ratio      | 5V              | —  | 50                      | 70   | —                       | dB   |
| CMRR             | Common Mode Rejection Ratio       | 5V              | —  | 50                      | 80   | —                       | dB   |
| A <sub>OL</sub>  | Open Loop Gain                    | —               | —  | 60                      | 80   | —                       | dB   |
| SR               | Slew Rate                         | 5V              | R <sub>LOAD</sub> =1MΩ, C <sub>LOAD</sub> =60pF,<br>OPDAnBW=0 (n=0, 1) | 180                     | 500  | —                       | V/ms |
|                  |                                   |                 | R <sub>LOAD</sub> =1MΩ, C <sub>LOAD</sub> =60pF,<br>OPDAnBW=1 (n=0, 1) | 600                     | 1800 | —                       |      |
| GBW              | Gain Bandwidth                    | 5V              | R <sub>LOAD</sub> =1MΩ, C <sub>LOAD</sub> =60pF,<br>OPDAnBW=0 (n=0, 1) | 250                     | 600  | —                       | kHz  |
|                  |                                   |                 | R <sub>LOAD</sub> =1MΩ, C <sub>LOAD</sub> =60pF,<br>OPDAnBW=1 (n=0, 1) | 800                     | 2000 | —                       |      |
| V <sub>OR</sub>  | Maximum Output Voltage Range      | —               | R <sub>LOAD</sub> =5kΩ to V <sub>DD</sub> /2                           | V <sub>SS</sub><br>+120 | —    | V <sub>DD</sub><br>-180 | mV   |

### D/A Converter Characteristics

Ta=-40°C~85°C

| Symbol            | Parameter   | Test Conditions |                                   | Min.            | Typ. | Max.             | Unit |
|-------------------|---|-----------------|-----------------------------------|-----------------|------|------------------|------|
|                   |   | V <sub>DD</sub> | Conditions                        |                 |      |                  |      |
| V <sub>DD</sub>   | Operating Voltage   | —               | —                                 | 1.8             | 5.0  | 5.5              | V    |
| V <sub>DACO</sub> | Output Voltage Range                                      | —               | —                                 | V <sub>SS</sub> | —    | V <sub>REF</sub> | V    |
| V <sub>REF</sub>  | Reference Voltage   | —               | —                                 | V <sub>DD</sub> |      |                  | V    |
| I <sub>DAC</sub>  | Additional Current for D/A Converter Enable<br>(each DAC) | 3V              | —                                 | —               | —    | 200              | μA   |
|                   |   | 5V              | —                                 | —               | —    | 280              |      |
| t <sub>ST</sub>   | Settling Time   | 3V              | C <sub>LOAD</sub> =50pF           | —               | —    | 5                | μs   |
|                   |   | 5V              |                                   | —               | —    | 5                |      |
| DNL               | Differential Non-linearity                                | 3V              | V <sub>REF</sub> =V <sub>DD</sub> | -1              | —    | +1               | LSB  |
|                   |   | 5V              |                                   | -1              | —    | +1               |      |
| INL               | Integral Non-linearity                                    | 3V              | V <sub>REF</sub> =V <sub>DD</sub> | -1.5            | —    | +1.5             | LSB  |
|                   |   | 5V              |                                   | -1.5            | —    | +1.5             |      |
| R <sub>O</sub>    | R2R Output Resistor                                       | 3V              | —                                 | —               | 20   | —                | kΩ   |
|                   |   | 5V              | —                                 | —               | 20   | —                |      |
| OSRR              | Offset Error  | 3V              | —                                 | —               | —    | 6                | mV   |
|                   |   | 5V              | —                                 | —               | —    | 10               |      |
| GERR              | Gain Error  | 3V              | —                                 | —               | —    | 12               | mV   |
|                   |   | 5V              | —                                 | —               | —    | 20               |      |

## Sink Current Generator Electrical Characteristics

Ta=-40°C~85°C, unless otherwise specified

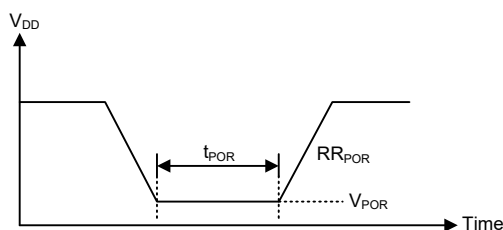
| Symbol             | Parameter                   | Test Conditions |  | Min. | Typ. | Max. | Unit |
|--------------------|-----------------------------|-----------------|--|------|------|------|------|
|                    |                             | V <sub>DD</sub> | Conditions   |      |      |      |      |
| V <sub>DD</sub>    | Operating Voltage           | —               | —  | 1.8  | —    | 5.5  | V    |
| I <sub>SINK0</sub> | Sink Current for ISINK0 Pin | 5V              | (After trimming)<br>Ta=25°C, V <sub>ISINK0</sub> =3.0V,<br>ISGDATA0[3:0]=0000B, ISST0=0            | 0.95 | 1.00 | 1.05 | mA   |
|                    |                             | —               | (After trimming)<br>Ta=-40°C~85°C, V <sub>ISINK0</sub> =1.0V~4.5V,<br>ISGDATA0[3:0]=0000B, ISST0=0 | 0.82 | 1.00 | 1.18 |      |
|                    |                             | —               | (After trimming)<br>Ta=-40°C~85°C, V <sub>ISINK0</sub> =0.7V~1.0V,<br>ISGDATA0[3:0]=0000B, ISST0=0 | 0.7  | 1.0  | 1.0  |      |
|                    |                             | 5V              | (After trimming)<br>Ta=25°C, V <sub>ISINK0</sub> =3.0V,<br>ISGDATA0[3:0]=1111B, ISST0=1            | 177  | 192  | 208  |      |
|                    |                             | —               | (After trimming)<br>Ta=-40°C~85°C, V <sub>ISINK0</sub> =1.0V~4.5V,<br>ISGDATA0[3:0]=1111B, ISST0=1 | 157  | 192  | 227  |      |
|                    |                             | —               | (After trimming)<br>Ta=-40°C~85°C, V <sub>ISINK0</sub> =0.7V~1.0V,<br>ISGDATA0[3:0]=1111B, ISST0=1 | 134  | 192  | 192  |      |
| I <sub>SINK1</sub> | Sink current for ISINK1 pin | 5V              | (After trimming)<br>Ta=25°C, V <sub>ISINK1</sub> =3.0V,<br>ISGDATA1[3:0]=0000B, ISST1=0            | 0.9  | 1.0  | 1.1  | mA   |
|                    |                             | —               | (After trimming)<br>Ta=-40°C~85°C, V <sub>ISINK1</sub> =1.0V~4.5V,<br>ISGDATA1[3:0]=0000B, ISST1=0 | 0.82 | 1.00 | 1.18 |      |
|                    |                             | —               | (After trimming)<br>Ta=-40°C~85°C, V <sub>ISINK1</sub> =0.7V~1.0V,<br>ISGDATA1[3:0]=0000B, ISST1=0 | 0.7  | 1.0  | 1.0  |      |
|                    |                             | 5V              | (After trimming)<br>Ta=25°C, V <sub>ISINK1</sub> =3.0V,<br>ISGDATA1[3:0]=1111B, ISST1=1            | 173  | 192  | 211  |      |
|                    |                             | —               | (After trimming)<br>Ta=-40°C~85°C, V <sub>ISINK1</sub> =1.0V~4.5V,<br>ISGDATA1[3:0]=1111B, ISST1=1 | 157  | 192  | 227  |      |
|                    |                             | —               | (After trimming)<br>Ta=-40°C~85°C, V <sub>ISINK1</sub> =0.7V~1.0V,<br>ISGDATA1[3:0]=1111B, ISST1=1 | 134  | 192  | 192  |      |

Note: When the ISINK0 and ISINK1 pins are bounded to the same pin location or externally connected together, their corresponding internal switches and current control registers should be properly configured to avoid current leakage problem. Refer to the “Sink Current Generator” section.

## Power-on Reset Characteristics

Ta=-40°C~85°C

| Symbol            | Parameter   | Test Conditions |            | Min.  | Typ. | Max. | Unit |
|-------------------|---|-----------------|------------|-------|------|------|------|
|                   |   | V <sub>DD</sub> | Conditions |       |      |      |      |
| V <sub>POR</sub>  | V <sub>DD</sub> Start Voltage to Ensure Power-on Reset                              | —               | —          | —     | —    | 100  | mV   |
| RR <sub>POR</sub> | V <sub>DD</sub> Rising Rate to Ensure Power-on Reset                                | —               | —          | 0.035 | —    | —    | V/ms |
| t <sub>POR</sub>  | Minimum Time for V <sub>DD</sub> Stays at V <sub>POR</sub> to Ensure Power-on Reset | —               | —          | 1     | —    | —    | ms   |



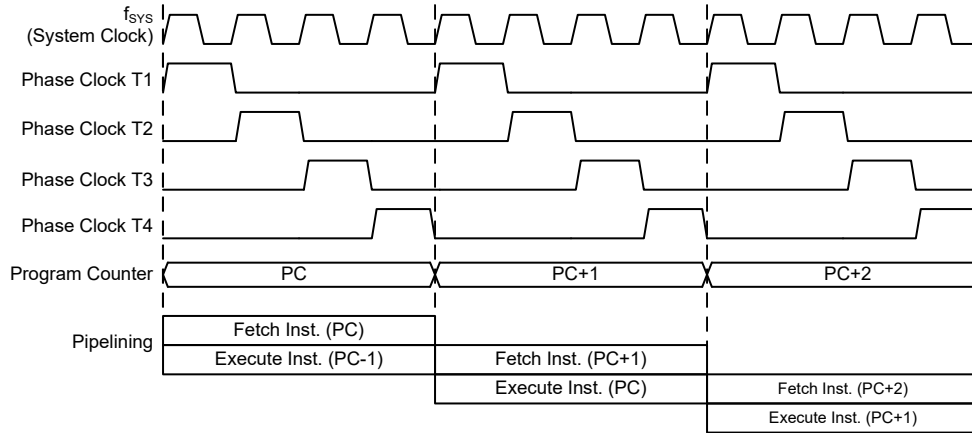
## System Architecture

A key factor in the high-performance features of the Holtek range of microcontrollers is attributed to their internal system architecture. The device takes advantage of the usual features found within RISC microcontrollers providing increased speed of operation and enhanced performance. The pipelining scheme is implemented in such a way that instruction fetching and instruction execution are overlapped, hence instructions are effectively executed in one cycle, with the exception of branch or call instructions which need one more cycle. An 8-bit wide ALU is used in practically all instruction set operations, which carries out arithmetic operations, logic operations, rotation, increment, decrement, branch decisions, etc. The internal data path is simplified by moving data through the Accumulator and the ALU. Certain internal registers are implemented in the Data Memory and can be directly or indirectly addressed. The simple addressing methods of these registers along with additional architectural features ensure that a minimum of external components is required to provide a functional I/O and A/D control system with maximum reliability and flexibility. This makes the device suitable for low-cost, high-volume production for controller applications.

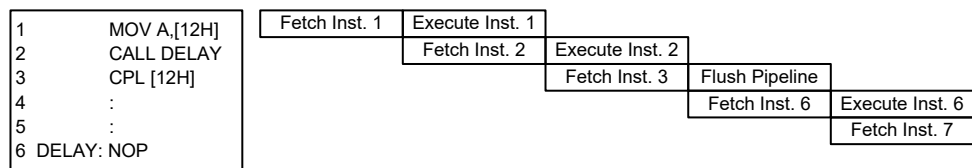
### Clocking and Pipelining

The main system clock, derived from either an HIRC or LIRC oscillator is subdivided into four internally generated non-overlapping clocks, T1~T4. The Program Counter is incremented at the beginning of the T1 clock during which time a new instruction is fetched. The remaining T2~T4 clocks carry out the decoding and execution functions. In this way, one T1~T4 clock cycle forms one instruction cycle. Although the fetching and execution of instructions takes place in consecutive instruction cycles, the pipelining structure of the microcontroller ensures that instructions are effectively executed in one instruction cycle. The exception to this are instructions where the contents of the Program Counter are changed, such as subroutine calls or jumps, in which case the instruction will take one more instruction cycle to execute.

For instructions involving branches, such as jump or call instructions, two machine cycles are required to complete instruction execution. An extra cycle is required as the program takes one cycle to first obtain the actual jump or call address and then another cycle to actually execute the branch. The requirement for this extra cycle should be taken into account by programmers in timing sensitive applications.



**System Clocking and Pipelining**



**Instruction Fetching**

### Program Counter

During program execution, the Program Counter is used to keep track of the address of the next instruction to be executed. It is automatically incremented by one each time an instruction is executed except for instructions, such as “JMP” or “CALL” that demands a jump to a non-consecutive Program Memory address. Only the lower 8 bits, known as the Program Counter Low Register, are directly addressable by the application program.

When executing instructions requiring jumps to non-consecutive addresses such as a jump instruction, a subroutine call, interrupt or reset, etc., the microcontroller manages program control by loading the required address into the Program Counter. For conditional skip instructions, once the condition has been met, the next instruction, which has already been fetched during the present instruction execution, is discarded and a dummy cycle takes its place while the correct instruction is obtained.

| Program Counter |                |
|-----------------|----------------|
| High Byte       | Low Byte (PCL) |
| PC11~PC8        | PCL7~PCL0      |

**Program Counter**

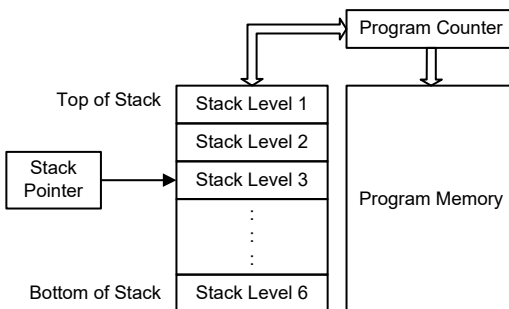
The lower byte of the Program Counter, known as the Program Counter Low register or PCL, is available for program control and is a readable and writeable register. By transferring data directly into this register, a short program jump can be executed directly; however, as only this low byte is available for manipulation, the jumps are limited to the present page of memory that is 256 locations. When such program jumps are executed it should also be noted that a dummy cycle will be inserted. Manipulating the PCL register may cause program branching, so an extra cycle is needed to pre-fetch.

## Stack

This is a special part of the memory which is used to save the contents of the Program Counter only. The stack is organized into multiple levels and neither part of the data nor part of the program space, and is neither readable nor writeable. The activated level is indexed by the Stack Pointer, and is neither readable nor writeable. At a subroutine call or interrupt acknowledge signal, the contents of the Program Counter are pushed onto the stack. At the end of a subroutine or an interrupt routine, signaled by a return instruction, RET or RETI, the Program Counter is restored to its previous value from the stack. After a device reset, the Stack Pointer will point to the top of the stack.

If the stack is full and an enabled interrupt takes place, the interrupt request flag will be recorded but the acknowledge signal will be inhibited. When the Stack Pointer is decremented, by RET or RETI, the interrupt will be serviced. This feature prevents stack overflow allowing the programmer to use the structure more easily. However, when the stack is full, a CALL subroutine instruction can still be executed which will result in a stack overflow. Precautions should be taken to avoid such cases which might cause unpredictable program branching.

If the stack is overflow, the first Program Counter save in the stack will be lost.



## Arithmetic and Logic Unit – ALU

The arithmetic-logic unit or ALU is a critical area of the microcontroller that carries out arithmetic and logic operations of the instruction set. Connected to the main microcontroller data bus, the ALU receives related instruction codes and performs the required arithmetic or logical operations after which the result will be placed in the specified register. As these ALU calculation or operations may result in carry, borrow or other status changes, the status register will be correspondingly updated to reflect these changes. The ALU supports the following functions:

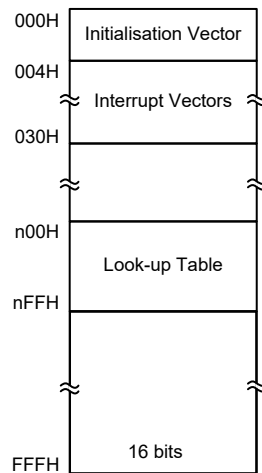
- Arithmetic operations:  
ADD, ADDM, ADC, ADCM, SUB, SUBM, SBC, SBCM, DAA
- Logic operations:  
AND, OR, XOR, ANDM, ORM, XORM, CPL, CPLA
- Rotation:  
RRA, RR, RRCA, RRC, RLA, RL, RLCA, RLC
- Increment and Decrement:  
INCA, INC, DECA, DEC
- Branch decision:  
JMP, SZ, SZA, SNZ, SIZ, SDZ, SIZA, SDZA, CALL, RET, RETI

## Flash Program Memory

The Program Memory is the location where the user code or program is stored. For the device the Program Memory is Flash type, which means it can be programmed and re-programmed a large number of times, allowing the user the convenience of code modification on the same device. By using the appropriate programming tools, the Flash device offers users the flexibility to conveniently debug and develop their applications while also offering a means of field programming and updating.

### Structure

The Program Memory has a capacity of 4K×16 bits. The Program Memory is addressed by the Program Counter and also contains data, table information and interrupt entries. Table data, which can be setup in any location within the Program Memory, is addressed by a separate table pointer register.



**Program Memory Structure**

### Special Vectors

Within the Program Memory, certain locations are reserved for the reset and interrupts. The location 000H is reserved for use by the device reset for program initialisation. After a device reset is initiated, the program will jump to this location and begin execution.

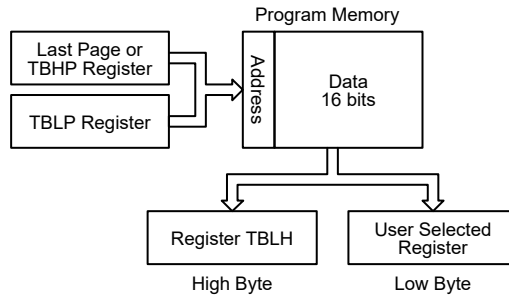
### Look-up Table

Any location within the Program Memory can be defined as a look-up table where programmers can store fixed data. To use the look-up table, the table pointer must first be setup by placing the address of the look up data to be retrieved in the table pointer registers, TBLP and TBHP. These registers define the total address of the look-up table.

After setting up the table pointer, the table data can be retrieved from the Program Memory using the “TABRD [m]” or “TABRDL [m]” instruction. When the instruction is executed, the lower order table byte from the Program Memory will be transferred to the user defined Data Memory register [m] as specified in the instruction. The higher order table data byte from the Program Memory will be transferred to the TBLH special register.



The accompanying diagram illustrates the addressing data flow of the look-up table.



### Table Program Example

The following example shows how the table pointer and table data is defined and retrieved from the microcontroller. This example uses raw table data located in the Program Memory which is stored there using the ORG statement. The value at this ORG statement is “0F00H” which refers to the start address of the last page within the 4K Program Memory of the device. The table pointer low byte register is setup here to have an initial value of “06H”. This will ensure that the first data read from the data table will be at the Program Memory address “0F06H” or 6 locations after the start of the last page. Note that the value for the table pointer is referenced to the first address specified by TBLP and TBHP if the “TABRD [m]” instruction is being used. The high byte of the table data which in this case is equal to zero will be transferred to the TBLH register automatically when the “TABRD [m]” instruction is executed.

Because the TBLH register is a read-only register and can not be restored, care should be taken to ensure its protection if both the main routine and Interrupt Service Routine use table read instructions. If using the table read instructions, the Interrupt Service Routines may change the value of the TBLH and subsequently cause errors if used again by the main routine. As a rule, it is recommended that simultaneous use of the table read instructions should be avoided. However, in situations where simultaneous use cannot be avoided, the interrupts should be disabled prior to the execution of any main routine table-read instructions. Note that all table related instructions require two instruction cycles to complete their operation.

### Table Read Program Example

```

tempreg1 db ?      ; temporary register #1
tempreg2 db ?      ; temporary register #2
:
:
:
mov a,06h          ; initialise low table pointer - note that this address is referenced
mov tblp,a        ; to the last page or the page that tbhp pointed
mov a,0Fh         ; initialise high table pointer
mov tbhp,a
:
:
tabrd tempreg1    ; transfers value in table referenced by table pointer data at program
                  ; memory address "0F06H" transferred to tempreg1 and TBLH
dec tblp          ; reduce value of table pointer by one
tabrd tempreg2    ; transfers value in table referenced by table pointer
                  ; data at program memory address "0F05H" transferred to tempreg2 and
                  ; TBLH
                  ; in this example the data "1AH" is transferred to tempreg1
                  ; and data "0FH" to register tempreg2 and the data "00H" is transferred
                  ; to TBLH
  
```

```

:
:
org 0F00h          ; sets initial address of program memory
dc 00Ah, 00Bh, 00Ch, 00Dh, 00Eh, 00Fh, 01Ah, 01Bh
:
:
:

```

### In Circuit Programming – ICP

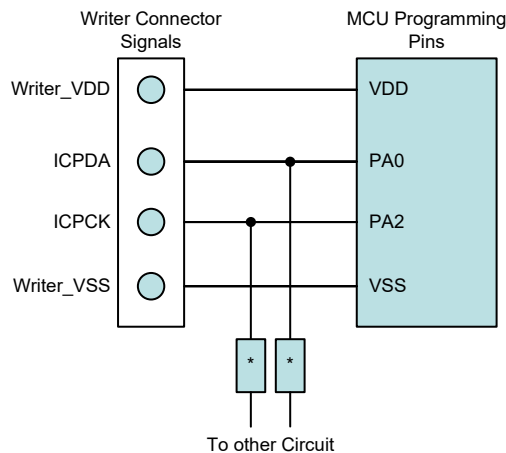
The provision of Flash Type Program Memory provides the user with a means of convenient and easy upgrades and modifications to their programs on the same device. As an additional convenience, a means of programming the microcontroller in-circuit has provided using a 4-pin interface. This provides manufacturers with the possibility of manufacturing their circuit boards complete with a programmed or un-programmed microcontroller, and then programming or upgrading the program at a later stage. This enables product manufacturers to easily keep their manufactured products supplied with the latest program releases without removal and re-insertion of the device.

The Holtek Flash MCU to Writer Programming Pin correspondence table is as follows:

| Holtek Writer Pins | MCU Programming Pins | Pin Description                 |
|--------------------|----------------------|---------------------------------|
| ICPDA              | PA0                  | Programming Serial Data/Address |
| ICPCK              | PA2                  | Programming Clock               |
| VDD                | VDD                  | Power Supply                    |
| VSS                | VSS                  | Ground                          |

The Program Memory can be programmed serially in-circuit using this 4-wire interface. Data is downloaded and uploaded serially on a single pin with an additional line for the clock. Two additional lines are required for the power supply. The technical details regarding the in-circuit programming of the device is beyond the scope of this document and will be supplied in supplementary literature.

During the programming process, taking control of the ICPDA and ICPCK pins for data and clock programming purposes. The user must there take care to ensure that no other outputs are connected to these two pins.



Note: \* may be resistor or capacitor. The resistance of \* must be greater than 1kΩ or the capacitance of \* must be less than 1nF.

## On Chip Debug Support – OCDS

An EV chip exists for the purposes of device emulation. This EV chip device also provides an “On-Chip Debug” function to debug the real MCU device during the development process. The EV chip and the real MCU device are almost functionally compatible except for “On-Chip Debug” function. Users can use the EV chip device to emulate the real chip device behavior by connecting the OCDSDA and OCDSCK pins to the Holtek HT-IDE development tools. The OCDSDA pin is the OCDS Data/Address input/output pin while the OCDSCK pin is the OCDS clock input pin. When users use the EV chip for debugging, other functions which are shared with the OCDSDA and OCDSCK pins in the device will have no effect in the EV chip. However, the two OCDS pins which are pin-shared with the ICP programming pins are still used as the Flash Memory programming pins for ICP. For more detailed OCDS information, refer to the corresponding document named “Holtek e-Link for 8-bit MCU OCDS User’s Guide”.

| Holtek e-Link Pins | EV Chip Pins | Pin Description                                 |
|--------------------|--------------|---|
| OCDSDA             | OCDSDA       | On-Chip Debug Support Data/Address input/output |
| OCDSCK             | OCDSCK       | On-Chip Debug Support Clock input               |
| VDD                | VDD          | Power Supply                                    |
| VSS                | VSS          | Ground  |

## Data Memory

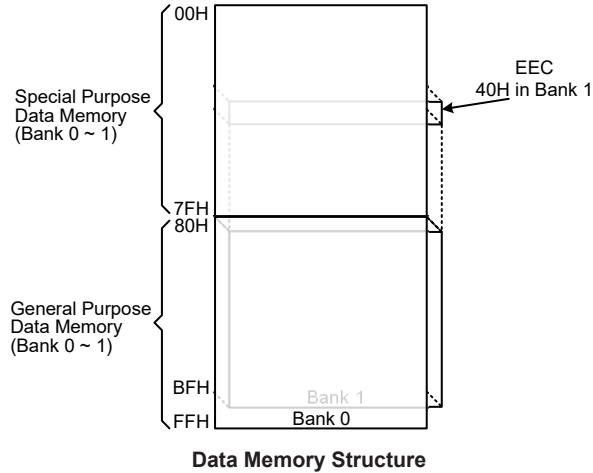
The Data Memory is a volatile area of 8-bit wide RAM internal memory and is the location where temporary information is stored.

### Structure

Categorized into two types, the first of these is an area of RAM where special function registers are located. These registers have fixed locations and are necessary for correct operation of the device. Many of these registers can be read from and written to directly under program control, however, some remain protected from user manipulation. The second area of Data Memory is reserved for general purpose use. All locations within this area are read and write accessible under program control.

The Data Memory is subdivided into two banks, all of which are implemented in 8-bit wide RAM. Switching between the different Data Memory banks is achieved by properly setting the Bank Pointer to the correct value. The address range of the Special Purpose Data Memory for the device is from 00H to 7FH while the General Purpose Data Memory address range is from 80H to FFH.

| Special Purpose Data Memory     | General Purpose Data Memory |                          |
|---------------------------------|-----------------------------|--------------------------|
| Located Banks                   | Capacity                    | Bank: Address            |
| 0: 00H~7FH<br>1: 40H (EEC only) | 192×8                       | 0: 80H~FFH<br>1: 80H~BFH |



### General Purpose Data Memory

All microcontroller programs require an area of read/write memory where temporary data can be stored and retrieved for use later. It is this area of RAM memory that is known as General Purpose Data Memory. This area of Data Memory is fully accessible by the user programming for both reading and writing operations. By using the bit operation instructions individual bits can be set or reset under program control giving the user a large range of flexibility for bit manipulation in the Data Memory.

### Special Purpose Data Memory

This area of Data Memory is where registers, necessary for the correct operation of the microcontroller, are stored. Most of the registers are both readable and writeable but some are protected and are readable only, the details of which are located under the relevant Special Function Register section. Note that for locations that are unused, any read instruction to these addresses will return the value “00H”.

|     | Bank 0   | Bank 1 |     | Bank 0   | Bank 1 |
|-----|----------|--------|-----|----------|--------|
| 00H | IAR0     |        | 40H | CTMC1    | EEC    |
| 01H | MP0      |        | 41H | CTMDL    |        |
| 02H | IAR1     |        | 42H | CTMDH    |        |
| 03H | MP1      |        | 43H | CTMAL    |        |
| 04H | BP       |        | 44H | CTMAH    |        |
| 05H | ACC      |        | 45H | STMC0    |        |
| 06H | PCL      |        | 46H | STMC1    |        |
| 07H | TBLP     |        | 47H | STMDL    |        |
| 08H | TBLH     |        | 48H | STMDH    |        |
| 09H | TBHP     |        | 49H | STMAL    |        |
| 0AH | STATUS   |        | 4AH | STMAH    |        |
| 0BH | SCC      |        | 4BH | OPSWA    |        |
| 0CH | HIRCC    |        | 4CH | OPSWB    |        |
| 0DH | LVRC     |        | 4DH | OPSWC    |        |
| 0EH | INTEG    |        | 4EH | OPSWD    |        |
| 0FH | RSTFC    |        | 4FH | OPSWE    |        |
| 10H | INTC0    |        | 50H | OPDC0    |        |
| 11H | INTC1    |        | 51H | OPDC1    |        |
| 12H | INTC2    |        | 52H | OPDDA    |        |
| 13H | INTC3    |        | 53H | OPDA0CAL |        |
| 14H | PA       |        | 54H | OPDA1CAL |        |
| 15H | PAC      |        | 55H | OPDCCAL  |        |
| 16H | PAPU     |        | 56H | SADC0    |        |
| 17H | PAWU     |        | 57H | SADC1    |        |
| 18H | PB       |        | 58H | SADC2    |        |
| 19H | PBC      |        | 59H | SADOL    |        |
| 1AH | PBPU     |        | 5AH | SADOH    |        |
| 1BH | PC       |        | 5BH | ISGENC   |        |
| 1CH | PCC      |        | 5CH | ISGDATA0 |        |
| 1DH | PCPU     |        | 5DH | ISGDATA1 |        |
| 1EH | VBGC     |        | 5EH | LVPUC    |        |
| 1FH | WDTC     |        | 5FH | ORMC     |        |
| 20H | PSC0R    |        | 60H | IFS      |        |
| 21H | TB0C     |        | 61H |          |        |
| 22H | PSC1R    |        |     |          |        |
| 23H | TB1C     |        |     |          |        |
| 24H | EEA      |        |     |          |        |
| 25H | EED      |        |     |          |        |
| 26H | PAS0     |        |     |          |        |
| 27H | PAS1     |        |     |          |        |
| 28H | PBS0     |        |     |          |        |
| 29H | PBS1     |        |     |          |        |
| 2AH | PCS0     |        |     |          |        |
| 2BH | TKTMR    |        |     |          |        |
| 2CH | TKC0     |        |     |          |        |
| 2DH | TKC1     |        |     |          |        |
| 2EH | TK16DL   |        |     |          |        |
| 2FH | TK16DH   |        |     |          |        |
| 30H | TKMOC0   |        |     |          |        |
| 31H | TKMOC1   |        |     |          |        |
| 32H | TKM016DL |        |     |          |        |
| 33H | TKM016DH |        |     |          |        |
| 34H | TKM0ROL  |        |     |          |        |
| 35H | TKM0ROH  |        |     |          |        |
| 36H | USR      |        |     |          |        |
| 37H | UCR1     |        |     |          |        |
| 38H | UCR2     |        |     |          |        |
| 39H | UCR3     |        |     |          |        |
| 3AH | BRDH     |        |     |          |        |
| 3BH | BRDL     |        |     |          |        |
| 3CH | UFCR     |        |     |          |        |
| 3DH | TXR_RXR  |        |     |          |        |
| 3EH | RxCNT    |        |     |          |        |
| 3FH | CTMC0    |        | 7FH |          |        |

□ : Unused, read as 00H

**Special Purpose Data Memory Structure**

## Special Function Register Description

Most of the Special Function Register details will be described in the relevant functional sections however several registers require a separate description in this section.

### Indirect Addressing Registers – IAR0, IAR1

The Indirect Addressing Registers, IAR0 and IAR1, although having their locations in normal RAM register space, do not actually physically exist as normal registers. The method of indirect addressing for RAM data manipulation uses these Indirect Addressing Registers and Memory Pointers, in contrast to direct memory addressing, where the actual memory address is specified. Actions on the IAR0 and IAR1 registers will result in no actual read or write operation to these registers but rather to the memory location specified by their corresponding Memory Pointers, MP0 or MP1. Acting as a pair, IAR0 and MP0 can together access data only from Bank 0 while the IAR1 register together with the MP1 register pair can access data from any Data Memory bank. As the Indirect Addressing Registers are not physically implemented, reading the Indirect Addressing Registers will return a result of “00H” and writing to the registers will result in no operation.

### Memory Pointers – MP0, MP1

Two Memory Pointers, known as MP0 and MP1 are provided. These Memory Pointers are physically implemented in the Data Memory and can be manipulated in the same way as normal registers providing a convenient way with which to address and track data. When any operation to the relevant Indirect Addressing Registers is carried out, the actual address that the microcontroller is directed to is the address specified by the related Memory Pointer. MP0, together with Indirect Addressing Register, IAR0, are used to access data from Bank 0, while MP1 and IAR1 are used to access data from all banks according to the BP register. Direct Addressing can only be used with Bank 0, all other banks must be addressed indirectly using MP1 and IAR1.

The following example shows how to clear a section of four Data Memory locations already defined as locations adres1 to adres4.

#### Indirect Addressing Program

```
data .section 'data'
adres1 db ?
adres2 db ?
adres3 db ?
adres4 db ?
block db ?
code .section at 0 'code'
org 00h
start:
    mov a, 04h          ; setup size of block
    mov block, a
    mov a, offset adres1 ; Accumulator loaded with first RAM address
    mov mp0, a         ; setup memory pointer with first RAM address
loop:
    clr IAR0           ; clear the data at address defined by MP0
    inc mp0            ; increase memory pointer
    sdz block          ; check if last memory location has been cleared
    jmp loop
continue:
```

The important point to note here is that in the example shown above, no reference is made to specific Data Memory addresses.

## **Accumulator – ACC**

The Accumulator is central to the operation of any microcontroller and is closely related with operations carried out by the ALU. The Accumulator is the place where all intermediate results from the ALU are stored. Without the Accumulator it would be necessary to write the result of each calculation or logical operation such as addition, subtraction, shift, etc., to the Data Memory resulting in higher programming and timing overheads. Data transfer operations usually involve the temporary storage function of the Accumulator; for example, when transferring data between one user-defined register and another, it is necessary to do this by passing the data through the Accumulator as no direct transfer between two registers is permitted.

## **Program Counter Low Byte Register – PCL**

To provide additional program control functions, the low byte of the Program Counter is made accessible to programmers by locating it within the Special Purpose area of the Data Memory. By manipulating this register, direct jumps to other program locations are easily implemented. Loading a value directly into this PCL register will cause a jump to the specified Program Memory location, however, as the register is only 8-bit wide, only jumps within the current Program Memory page are permitted. When such operations are used, note that a dummy cycle will be inserted.

## **Look-up Table Registers – TBLP, TBHP, TBLH**

These three special function registers are used to control operation of the look-up table which is stored in the Program Memory. TBLP and TBHP are the table pointers and indicate the location where the table data is located. Their value must be setup before any table read commands are executed. Their value can be changed, for example using the “INC” or “DEC” instructions, allowing for easy table data pointing and reading. TBLH is the location where the high order byte of the table data is stored after a table read data instruction has been executed. Note that the lower order table data byte is transferred to a user defined location.

## **Status Register – STATUS**

This 8-bit register contains the zero flag (Z), carry flag (C), auxiliary carry flag (AC), overflow flag (OV), power down flag (PDF), and watchdog time-out flag (TO). These arithmetic/logical operation and system management flags are used to record the status and operation of the microcontroller.

With the exception of the TO and PDF flags, bits in the status register can be altered by instructions like most other registers. Any data written into the status register will not change the TO or PDF flag. In addition, operations related to the status register may give different results due to the different instruction operations. The TO flag can be affected only by a system power-up, a WDT time-out or by executing the “CLR WDT” or “HALT” instruction. The PDF flag is affected only by executing the “HALT” or “CLR WDT” instruction or during a system power-up.

The Z, OV, AC and C flags generally reflect the status of the latest operations.

- C is set if an operation results in a carry during an addition operation or if a borrow does not take place during a subtraction operation; otherwise C is cleared. C is also affected by a rotate through carry instruction.
- AC is set if an operation results in a carry out of the low nibbles in addition, or no borrow from the high nibble into the low nibble in subtraction; otherwise AC is cleared.
- Z is set if the result of an arithmetic or logical operation is zero; otherwise Z is cleared.
- OV is set if an operation results in a carry into the highest-order bit but not a carry out of the highest-order bit, or vice versa; otherwise OV is cleared.

- PDF is cleared by a system power-up or executing the “CLR WDT” instruction. PDF is set by executing the “HALT” instruction.
- TO is cleared by a system power-up or executing the “CLR WDT” or “HALT” instruction. TO is set by a WDT time-out.

In addition, on entering an interrupt sequence or executing a subroutine call, the status register will not be pushed onto the stack automatically. If the contents of the status register are important and if the subroutine can corrupt the status register, precautions must be taken to correctly save it.

• **STATUS Register**

| Bit  | 7 | 6 | 5  | 4   | 3   | 2   | 1   | 0   |
|------|---|---|----|-----|-----|-----|-----|-----|
| Name | — | — | TO | PDF | OV  | Z   | AC  | C   |
| R/W  | — | — | R  | R   | R/W | R/W | R/W | R/W |
| POR  | — | — | 0  | 0   | x   | x   | x   | x   |

“x”: Unknown

- Bit 7~6      Unimplemented, read as “0”
- Bit 5        **TO**: Watchdog Time-out flag  
               0: After power up or executing the “CLR WDT” or “HALT” instruction  
               1: A watchdog time-out occurred
- Bit 4        **PDF**: Power down flag  
               0: After power up or executing the “CLR WDT” instruction  
               1: By executing the “HALT” instruction
- Bit 3        **OV**: Overflow flag  
               0: No overflow  
               1: An operation results in a carry into the highest-order bit but not a carry out of the highest-order bit or vice versa
- Bit 2        **Z**: Zero flag  
               0: The result of an arithmetic or logical operation is not zero  
               1: The result of an arithmetic or logical operation is zero
- Bit 1        **AC**: Auxiliary flag  
               0: No auxiliary carry  
               1: An operation results in a carry out of the low nibbles in addition, or no borrow from the high nibble into the low nibble in subtraction
- Bit 0        **C**: Carry flag  
               0: No carry-out  
               1: An operation results in a carry during an addition operation or if a borrow does not take place during a subtraction operation
- The “C” flag is also affected by a rotate through carry instruction.



### Bank Pointer – BP

For this device, the Data Memory is divided into two banks, Bank 0 and Bank 1. Selecting the required Data Memory area is achieved using the Bank Pointer. Bit 0 of the Bank Pointer is used to select Data Memory Banks 0~1.

The Data Memory is initialised to Bank 0 after a reset, except for a WDT time-out reset in the IDLE or SLEEP Mode, in which case, the Data Memory bank remains unaffected. Directly addressing the Data Memory will always result in Bank 0 being accessed irrespective of the value of the Bank Pointer. Accessing data from Bank1 must be implemented using Indirect Addressing.

#### • BP Register

| Bit  | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0     |
|------|---|---|---|---|---|---|---|-------|
| Name | — | — | — | — | — | — | — | DMBP0 |
| R/W  | — | — | — | — | — | — | — | R/W   |
| POR  | — | — | — | — | — | — | — | 0     |

Bit 7~1 Unimplemented, read as “0”

Bit 0 **DMBP0**: Data Memory Bank selection  
 0: Bank 0  
 1: Bank 1

### Option Memory Mapping Register – ORMC

The ORMC register is used to enable the Option Memory Mapping function. The Option Memory capacity is 32 words. When a specific pattern of 55H and AAH is consecutively written into this register, the Option Memory Mapping function will be enabled and then the Option Memory code can be read by using the table read instruction. The Option Memory addresses 00H~1FH will be mapped to Program Memory last page addresses E0H~FFH.

To successfully enable the Option Memory Mapping function, the specific pattern of 55H and AAH must be written into the ORMC register in two consecutive instruction cycles. It is therefore recommended that the global interrupt bit EMI should first be cleared before writing the specific pattern, and then set high again at a proper time according to users’ requirements after the pattern is successfully written. An internal timer will be activated when the pattern is successfully written. The mapping operation will be automatically finished after a period of  $4 \times T_{LIRC}$ . Therefore, users should read the data in time, otherwise the Option Memory Mapping function needs to be restarted. After the completion of each consecutive write operation to the ORMC register, the timer will recount.

When the table read instructions are used to read the Option Memory code, both “TABRD [m]” and “TABRDL [m]” instructions can be used. However, care must be taken if the “TABRD [m]” instruction is used, the table pointer defined by the TBHP register must be referenced to the last page. Refer to corresponding sections about the table read instruction for more details.

#### • ORMC Register

| Bit  | 7     | 6     | 5     | 4     | 3     | 2     | 1     | 0     |
|------|-------|-------|-------|-------|-------|-------|-------|-------|
| Name | ORMC7 | ORMC6 | ORMC5 | ORMC4 | ORMC3 | ORMC2 | ORMC1 | ORMC0 |
| R/W  | R/W   | R/W   | R/W   | R/W   | R/W   | R/W   | R/W   | R/W   |
| POR  | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     |

Bit 7~0 **ORMC7~ORMC0**: Option Memory Mapping specific pattern

When a specific pattern of 55H and AAH is written into this register, the Option Memory Mapping function will be enabled. Note that the register content will be cleared after the MCU is woken up from the IDLE/SLEEP mode.

## EEPROM Data Memory

The device contains an area of internal EEPROM Data Memory. EEPROM is by its nature a non-volatile form of re-programmable memory, with data retention even when its power supply is removed. By incorporating this kind of data memory, a whole new host of application possibilities are made available to the designer. The availability of EEPROM storage allows information such as product identification numbers, calibration values, specific user data, system setup data or other product information to be stored directly within the product microcontroller. The process of reading and writing data to the EEPROM memory has been reduced to a very trivial affair.

### EEPROM Data Memory Structure

The EEPROM Data Memory capacity is 32×8 bits for the device. Unlike the Program Memory and RAM Data Memory, the EEPROM Data Memory is not directly mapped into memory space and is therefore not directly addressable in the same way as the other types of memory. Read and Write operations to the EEPROM are carried out in single byte operations using an address register and a data register in Bank 0 and a single control register in Bank 1.

### EEPROM Registers

Three registers control the overall operation of the internal EEPROM Data Memory. These are the address register, EEA, the data register, EED and a single control register, EEC. As both the EEA and EED registers are located in Bank 0, they can be directly accessed in the same way as any other Special Function Register. The EEC register however, being located in Bank 1, can only be read from or written to indirectly using the MP1 Memory Pointer and Indirect Addressing Register, IAR1. Because the EEC control register is located at address 40H in Bank 1, the MP1 Memory Pointer must first be set to the value 40H and the Bank Pointer register, BP, set to the value, 01H, before any operations on the EEC register are executed.

| Register Name | Bit  |      |      |      |      |      |      |      |
|---------------|------|------|------|------|------|------|------|------|
|               | 7    | 6    | 5    | 4    | 3    | 2    | 1    | 0    |
| EEA           | —    | —    | —    | EEA4 | EEA3 | EEA2 | EEA1 | EEA0 |
| EED           | EED7 | EED6 | EED5 | EED4 | EED3 | EED2 | EED1 | EED0 |
| EEC           | D7   | —    | —    | —    | WREN | WR   | RDEN | RD   |

**EEPROM Register List**

#### • EEA Register

| Bit  | 7 | 6 | 5 | 4    | 3    | 2    | 1    | 0    |
|------|---|---|---|------|------|------|------|------|
| Name | — | — | — | EEA4 | EEA3 | EEA2 | EEA1 | EEA0 |
| R/W  | — | — | — | R/W  | R/W  | R/W  | R/W  | R/W  |
| POR  | — | — | — | 0    | 0    | 0    | 0    | 0    |

Bit 7~5 Unimplemented, read as “0”

Bit 4~0 **EEA4~EEA0**: Data EEPROM address bit 4 ~ bit 0

#### • EED Register

| Bit  | 7    | 6    | 5    | 4    | 3    | 2    | 1    | 0    |
|------|------|------|------|------|------|------|------|------|
| Name | EED7 | EED6 | EED5 | EED4 | EED3 | EED2 | EED1 | EED0 |
| R/W  | R/W  | R/W  | R/W  | R/W  | R/W  | R/W  | R/W  | R/W  |
| POR  | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    |

Bit 7~0 **EED7~EED0**: Data EEPROM data bit 7 ~ bit 0

• **EEC Register**

| Bit  | 7   | 6 | 5 | 4 | 3    | 2   | 1    | 0   |
|------|-----|---|---|---|------|-----|------|-----|
| Name | D7  | — | — | — | WREN | WR  | RDEN | RD  |
| R/W  | R/W | — | — | — | R/W  | R/W | R/W  | R/W |
| POR  | 0   | — | — | — | 0    | 0   | 0    | 0   |

Bit 7        **D7**: Reserved bit, must be fixed at “0”

Bit 6~4     Unimplemented, read as “0”

Bit 3        **WREN**: Data EEPROM Write Enable  
               0: Disable  
               1: Enable

This is the Data EEPROM Write Enable Bit which must be set high before Data EEPROM write operations are carried out. Clearing this bit to zero will inhibit Data EEPROM write operations.

Bit 2        **WR**: EEPROM Write Control  
               0: Write cycle has finished  
               1: Activate a write cycle

This is the Data EEPROM Write Control Bit and when set high by the application program will activate a write cycle. This bit will be automatically reset to zero by the hardware after the write cycle has finished. Setting this bit high will have no effect if the WREN has not first been set high.

Bit 1        **RDEN**: Data EEPROM Read Enable  
               0: Disable  
               1: Enable

This is the Data EEPROM Read Enable Bit which must be set high before Data EEPROM read operations are carried out. Clearing this bit to zero will inhibit Data EEPROM read operations.

Bit 0        **RD**: EEPROM Read Control  
               0: Read cycle has finished  
               1: Activate a read cycle

This is the Data EEPROM Read Control Bit and when set high by the application program will activate a read cycle. This bit will be automatically reset to zero by the hardware after the read cycle has finished. Setting this bit high will have no effect if the RDEN has not first been set high.

- Note: 1. The WREN, WR, RDEN and RD cannot be set high at the same time in one instruction. The WR and RD cannot be set high at the same time.  
 2. Ensure that the  $f_{SUB}$  clock is stable before executing the write operation.  
 3. Ensure that the write operation is totally complete before changing the contents of the EEPROM related registers.

### **Reading Data from the EEPROM**

To read data from the EEPROM, the EEPROM address of the data to be read must first be placed in the EEA register. Then the read enable bit, RDEN, in the EEC register must be set high to enable the read function. If the RD bit in the EEC register is now set high, a read cycle will be initiated. Setting the RD bit high will not initiate a read operation if the RDEN bit has not been set. When the read cycle terminates, the RD bit will be automatically cleared to zero, after which the data can be read from the EED register. The data will remain in the EED register until another read or write operation is executed. The application program can poll the RD bit to determine when the data is valid for reading.

### **Writing Data to the EEPROM**

To write data to the EEPROM, the EEPROM address of the data to be written must first be placed in the EEA register and the data placed in the EED register. To initiate a write cycle, the write enable bit, WREN, in the EEC register must first be set high to enable the write function. After this, the WR bit in the EEC register must be immediately set high to initiate a write cycle. These two instructions must be executed in two consecutive instruction cycles. The global interrupt bit EMI should also first be cleared before implementing any write operations, and then set again after the write cycle has started. Note that setting the WR bit high will not initiate a write cycle if the WREN bit has not been set. As the EEPROM write cycle is controlled using an internal timer whose operation is asynchronous to microcontroller system clock, a certain time will elapse before the data will have been written into the EEPROM. Detecting when the write cycle has finished can be implemented either by polling the WR bit in the EEC register or by using the EEPROM interrupt. When the write cycle terminates, the WR bit will be automatically cleared to zero by the microcontroller, informing the user that the data has been written to the EEPROM. The application program can therefore poll the WR bit to determine when the write cycle has ended.

### **Write Protection**

Protection against inadvertent write operation is provided in several ways. After the device is powered-on the Write Enable bit in the control register will be cleared preventing any write operations. Also at power-on the Bank Pointer register, BP, will be reset to zero, which means that Data Memory Bank 0 will be selected. As the EEPROM control register is located in Bank 1, this adds a further measure of protection against spurious write operations. During normal program operation, ensuring that the Write Enable bit in the control register is cleared will safeguard against incorrect write operations.

### **EEPROM Interrupt**

The EEPROM write interrupt is generated when an EEPROM write cycle has ended. The EEPROM interrupt must first be enabled by setting the DEE bit in the relevant interrupt register. When an EEPROM write cycle ends, the DEF request flag will be set. If the global and the EEPROM interrupts are enabled and the stack is not full, a jump to the associated EEPROM Interrupt vector will take place. When the interrupt is serviced the EEPROM interrupt flag, DEF, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts. More details can be obtained in the Interrupt section.

## Programming Considerations

Care must be taken that data is not inadvertently written to the EEPROM. Protection can be enhanced by ensuring that the Write Enable bit is normally cleared to zero when not writing. Also the Bank Pointer register, BP, could be normally cleared to zero as this would inhibit access to Bank 1 where the EEPROM control register exists. Although certainly not necessary, consideration might be given in the application program to the checking of the validity of new write data by a simple read back process.

When writing data, the WR bit must be set high immediately after the WREN bit has been set high, to ensure the write cycle executes correctly. The global interrupt bit EMI should also be cleared before a write cycle is executed and then re-enabled after the write cycle starts. Note that the device should not enter the IDLE or SLEEP mode until the EEPROM read or write operation is totally complete. Otherwise, the EEPROM read or write operation will fail.

## Programming Examples

### Reading data from the EEPROM - polling method

```
MOV A, EEPROM_ADRES      ; user defined address
MOV EEA, A
MOV A, 40H                ; setup memory pointer MP1
MOV MP1, A               ; MP1 points to EEC register
MOV A, 01H               ; setup Bank Pointer
MOV BP, A
SET IAR1.1               ; set RDEN bit, enable read operations
SET IAR1.0               ; start Read Cycle - set RD bit
BACK:
SZ IAR1.0                ; check for read cycle end
JMP BACK
CLR IAR1                  ; disable EEPROM read if no more read operations are required
CLR BP
MOV A, EED                ; move read data to register
MOV READ_DATA, A
```

Note: For each read operation, the address register should be re-specified followed by setting the RD bit high to activate a read cycle even if the target address is consecutive.

### Writing Data to the EEPROM - polling method

```
MOV A, EEPROM_ADRES      ; user defined address
MOV EEA, A
MOV A, EEPROM_DATA       ; user defined data
MOV EED, A
MOV A, 40H                ; setup memory pointer MP1
MOV MP1, A               ; MP1 points to EEC register
MOV A, 01H               ; setup Bank Pointer
MOV BP, A
CLR EMI
SET IAR1.3               ; set WREN bit, enable write operations
SET IAR1.2               ; start Write Cycle - set WR bit - executed immediately
                        ; after setting WREN bit

SET EMI
BACK:
SZ IAR1.2                ; check for write cycle end
JMP BACK
CLR BP
```

## Oscillators

Various oscillator options offer the user a wide range of functions according to their various application requirements. The flexible features of the oscillator functions ensure that the best optimisation can be achieved in terms of speed and power saving. Oscillator operations are selected through the relevant control registers.

### Oscillator Overview

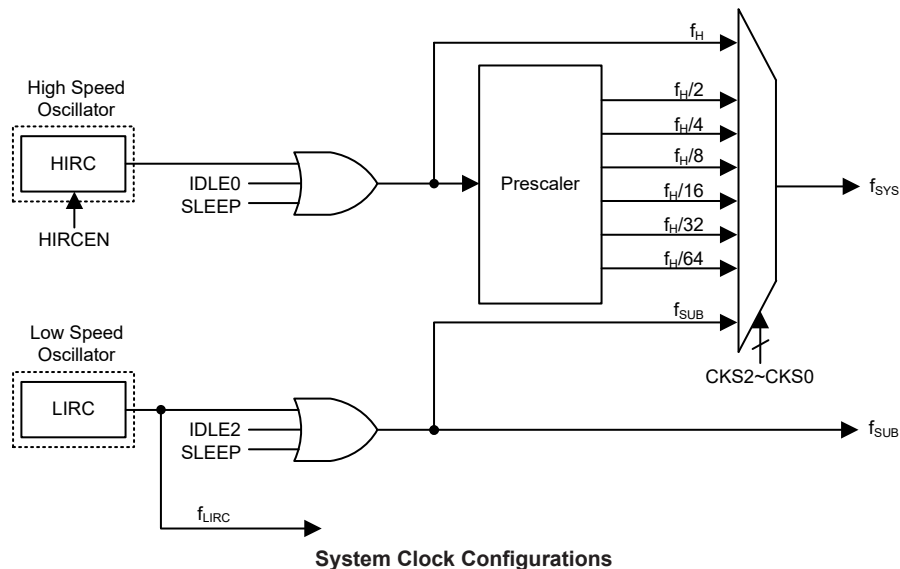
In addition to being the source of the main system clock the oscillators also provide clock sources for the Watchdog Timer and Time Base Interrupts. Two fully integrated internal oscillators, requiring no external components, are provided to form a wide range of both fast and slow system oscillators. The higher frequency oscillator provides higher performance but carry with it the disadvantage of higher power requirements, while the opposite is of course true for the lower frequency oscillator. With the capability of dynamically switching between fast and slow system clock, the device has the flexibility to optimize the performance/power ratio, a feature especially important in power sensitive portable applications.

| Type                   | Name | Frequency |
|------------------------|------|-----------|
| Internal High Speed RC | HIRC | 8MHz      |
| Internal Low Speed RC  | LIRC | 32kHz     |

Oscillator Types

### System Clock Configurations

There are two oscillator sources, a high speed oscillator and a low speed oscillator. The high speed system clock is sourced from the internal 8MHz RC oscillator, HIRC. The low speed oscillator is the internal 32kHz RC oscillator, LIRC. Selecting whether the low or high speed oscillator is used as the system oscillator is implemented using the CKS2~CKS0 bits in the SCC register and as the system clock can be dynamically selected.



### **Internal High Speed RC Oscillator – HIRC**

The high speed internal RC oscillator is a fully integrated system oscillator requiring no external components. The internal RC oscillator has a fixed frequency of 8MHz. Device trimming during the manufacturing process and the inclusion of internal frequency compensation circuits are used to ensure that the influence of the power supply voltage, temperature and process variations on the oscillation frequency are minimised.

### **Internal 32kHz Oscillator – LIRC**

The internal 32kHz System Oscillator is also a fully integrated RC oscillator with a typical frequency of 32kHz, requiring no external components for its implementation. Device trimming during the manufacturing process and the inclusion of internal frequency compensation circuits are used to ensure that the influence of the power supply voltage, temperature and process variations on the oscillation frequency are minimised.

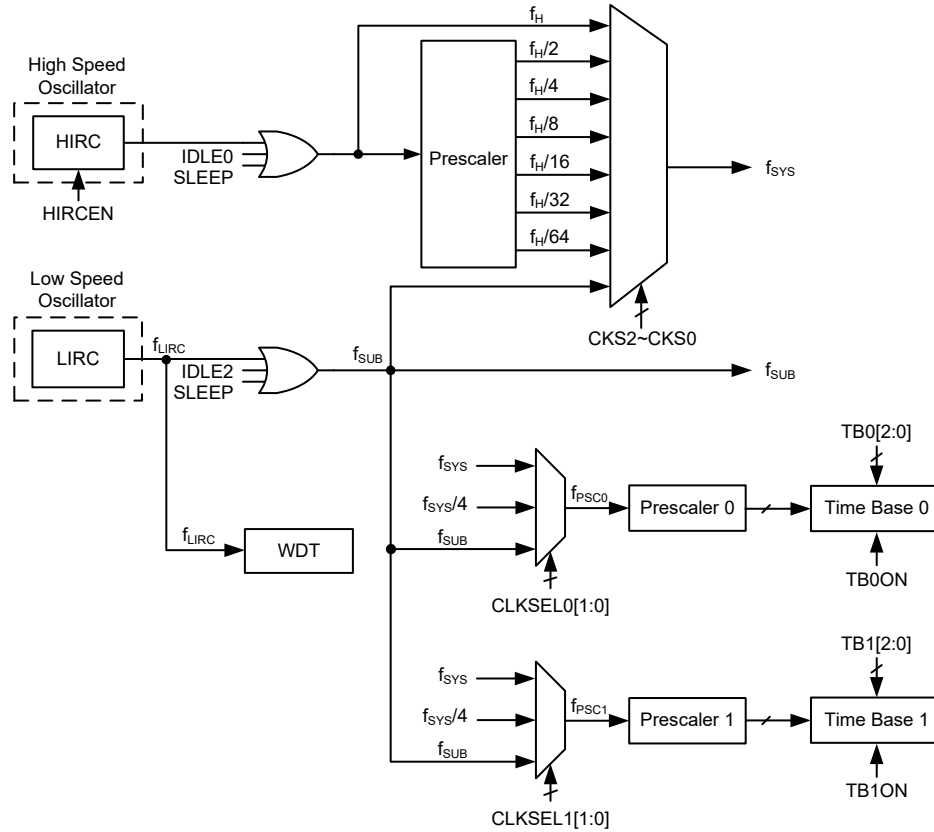
## **Operating Modes and System Clocks**

Present day applications require that their microcontrollers have high performance but often still demand that they consume as little power as possible, conflicting requirements that are especially true in battery powered portable applications. The fast clocks required for high performance will by their nature increase current consumption and of course vice versa, lower speed clocks reduce current consumption. As Holtek has provided the device with both high and low speed clock sources and the means to switch between them dynamically, the user can optimise the operation of their microcontroller to achieve the best performance/power ratio.

### **System Clocks**

The device has many different clock sources for both the CPU and peripheral function operation. By providing the user with a wide range of clock options using register programming, a clock system can be configured to obtain maximum application performance.

The main system clock, can come from a high frequency,  $f_H$ , or low frequency,  $f_{SUB}$ , source, and is selected using the CKS2~CKS0 bits in the SCC register. The high speed system clock is sourced from the HIRC oscillator. The low speed system clock source is sourced from the LIRC oscillator. The other choice, which is a divided version of the high speed system oscillator has a range of  $f_H/2 \sim f_H/64$ .



Device Clock Configurations

Note: When the system clock source  $f_{SYS}$  is switched to  $f_{SUB}$  from  $f_H$ , the high speed oscillator will stop to conserve the power or continue to oscillate to provide the clock source,  $f_H \sim f_H/64$ , for peripheral circuit to use, which is determined by configuring the corresponding high speed oscillator enable control bit.

### System Operation Modes

There are six different modes of operation for the microcontroller, each one with its own special characteristics and which can be chosen according to the specific performance and power requirements of the application. There are two modes allowing normal operation of the microcontroller, the FAST Mode and SLOW Mode. The remaining four modes, the SLEEP, IDLE0, IDLE1 and IDLE2 Modes are used when the microcontroller CPU is switched off to conserve power.



| Operation Mode | CPU | Register Setting |        |           | f <sub>sys</sub>                   | f <sub>H</sub>        | f <sub>SUB</sub> | f <sub>LIRC</sub>     |
|----------------|-----|------------------|--------|-----------|------------------------------------|-----------------------|------------------|-----------------------|
|                |     | FHIDEN           | FSIDEN | CKS2~CKS0 |                                    |                       |                  |                       |
| FAST           | On  | x                | x      | 000~110   | f <sub>H</sub> ~f <sub>H</sub> /64 | On                    | On               | On                    |
| SLOW           | On  | x                | x      | 111       | f <sub>SUB</sub>                   | On/Off <sup>(1)</sup> | On               | On                    |
| IDLE0          | Off | 0                | 1      | 000~110   | Off                                | Off                   | On               | On                    |
|                |     |                  |        | 111       | On                                 |                       |                  |                       |
| IDLE1          | Off | 1                | 1      | xxx       | On                                 | On                    | On               | On                    |
| IDLE2          | Off | 1                | 0      | 000~110   | On                                 | On                    | Off              | On                    |
|                |     |                  |        | 111       | Off                                |                       |                  |                       |
| SLEEP          | Off | 0                | 0      | xxx       | Off                                | Off                   | Off              | On/Off <sup>(2)</sup> |

"x": Don't care

Note: 1. The f<sub>H</sub> clock will be switched on or off by configuring the corresponding oscillator enable bit in the SLOW mode.

2. The f<sub>LIRC</sub> clock will be switched on or off which is controlled by the WDT function being enabled or disabled in the SLEEP mode.

### FAST Mode

This is one of the main operating modes where the microcontroller has all of its functions operational and where the system clock is provided by the high speed oscillator. This mode operates allowing the microcontroller to operate normally with a clock source come from the high speed oscillator, HIRC. The high speed oscillator will however first be divided by a ratio ranging from 1 to 64, the actual ratio being selected by the CKS2~CKS0 bits in the SCC register. Although a high speed oscillator is used, running the microcontroller at a divided clock ratio reduces the operating current.

### SLOW Mode

This is also a mode where the microcontroller operates normally although now with a slower speed clock source. The clock source used will be from f<sub>SUB</sub>. The f<sub>SUB</sub> clock is derived from the LIRC oscillator.

### SLEEP Mode

The SLEEP Mode is entered when a HALT instruction is executed and when the FHIDEN and FSIDEN bit both are low. In the SLEEP mode the CPU will be stopped. The f<sub>SUB</sub> clock provided to the peripheral function will also be stopped. However, the f<sub>LIRC</sub> clock can continue to operate if the WDT function is enabled.

### IDLE0 Mode

The IDLE0 Mode is entered when a HALT instruction is executed and when the FHIDEN bit in the SCC register is low and the FSIDEN bit in the SCC register is high. In the IDLE0 Mode the CPU will be switched off but the low speed oscillator will be turned on to drive some peripheral functions.

### IDLE1 Mode

The IDLE1 Mode is entered when a HALT instruction is executed and when the FHIDEN bit in the SCC register is high and the FSIDEN bit in the SCC register is high. In the IDLE1 Mode the CPU will be switched off but both the high and low speed oscillators will be turned on to provide a clock source to keep some peripheral functions operational.

### IDLE2 Mode

The IDLE2 Mode is entered when a HALT instruction is executed and when the FHIDEN bit in the SCC register is high and the FSIDEN bit in the SCC register is low. In the IDLE2 Mode the CPU will be switched off but the high speed oscillator will be turned on to provide a clock source to keep some peripheral functions operational.

### Control Registers

The SCC and HIRCC registers are used to control the system clock and the HIRC oscillator configurations.

| Register Name | Bit  |      |      |   |   |   |        |        |
|---------------|------|------|------|---|---|---|--------|--------|
|               | 7    | 6    | 5    | 4 | 3 | 2 | 1      | 0      |
| SCC           | CKS2 | CKS1 | CKS0 | — | — | — | FHIDEN | FSIDEN |
| HIRCC         | —    | —    | —    | — | — | — | HIRCF  | HIRCEN |

**System Operating Mode Control Register List**

#### • SCC Register

| Bit  | 7    | 6    | 5    | 4 | 3 | 2 | 1      | 0      |
|------|------|------|------|---|---|---|--------|--------|
| Name | CKS2 | CKS1 | CKS0 | — | — | — | FHIDEN | FSIDEN |
| R/W  | R/W  | R/W  | R/W  | — | — | — | R/W    | R/W    |
| POR  | 0    | 0    | 0    | — | — | — | 0      | 0      |

Bit 7~5     **CKS2~CKS0**: System clock selection

- 000:  $f_H$
- 001:  $f_H/2$
- 010:  $f_H/4$
- 011:  $f_H/8$
- 100:  $f_H/16$
- 101:  $f_H/32$
- 110:  $f_H/64$
- 111:  $f_{SUB}$

These three bits are used to select which clock is used as the system clock source. In addition to the system clock source directly derived from  $f_H$  or  $f_{SUB}$ , a divided version of the high speed system oscillator can also be chosen as the system clock source.

Bit 4~2     Unimplemented, read as “0”

Bit 1       **FHIDEN**: High Frequency oscillator control when CPU is switched off

- 0: Disable
- 1: Enable

This bit is used to control whether the high speed oscillator is activated or stopped when the CPU is switched off by executing a “HALT” instruction.

Bit 0       **FSIDEN**: Low Frequency oscillator control when CPU is switched off

- 0: Disable
- 1: Enable

This bit is used to control whether the low speed oscillator is activated or stopped when the CPU is switched off by executing a “HALT” instruction.

• **HIRCC Register**

| Bit  | 7 | 6 | 5 | 4 | 3 | 2 | 1     | 0      |
|------|---|---|---|---|---|---|-------|--------|
| Name | — | — | — | — | — | — | HIRCF | HIRCEN |
| R/W  | — | — | — | — | — | — | R     | R/W    |
| POR  | — | — | — | — | — | — | 0     | 1      |

Bit 7~2 Unimplemented, read as “0”

Bit 1 **HIRCF**: HIRC oscillator stable flag  
 0: HIRC unstable  
 1: HIRC stable

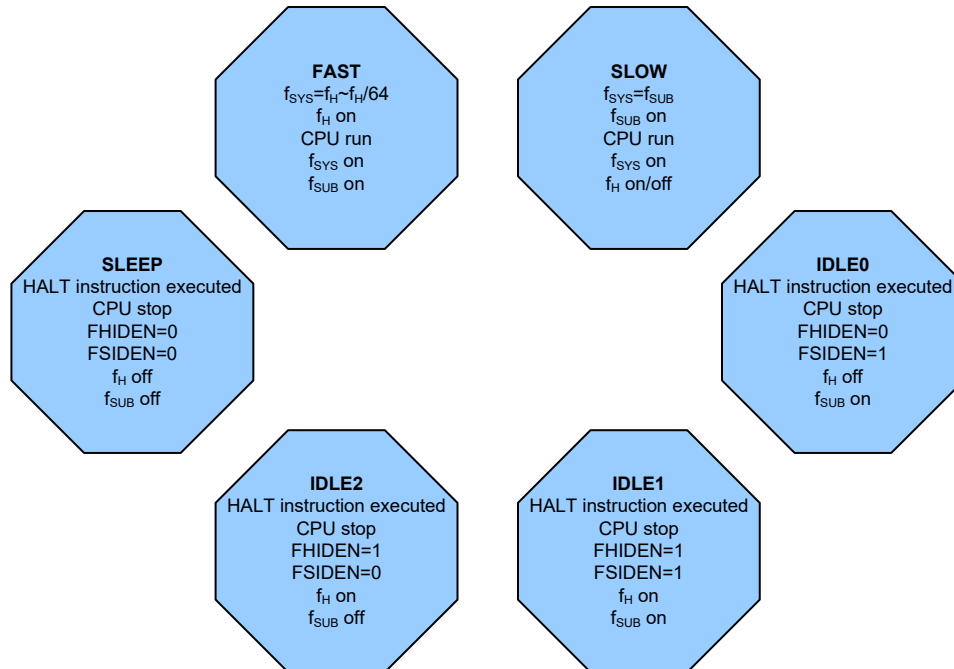
This bit is used to indicate whether the HIRC oscillator is stable or not. When the HIRCEN bit is set to 1 to enable the HIRC oscillator, the HIRCF bit will first be cleared to 0 and then set to 1 after the HIRC oscillator is stable.

Bit 0 **HIRCEN**: HIRC oscillator enable control  
 0: Disable  
 1: Enable

**Operating Mode Switching**

The device can switch between operating modes dynamically allowing the user to select the best performance/power ratio for the present task in hand. In this way microcontroller operations that do not require high performance can be executed using slower clocks thus requiring less operating current and prolonging battery life in portable applications.

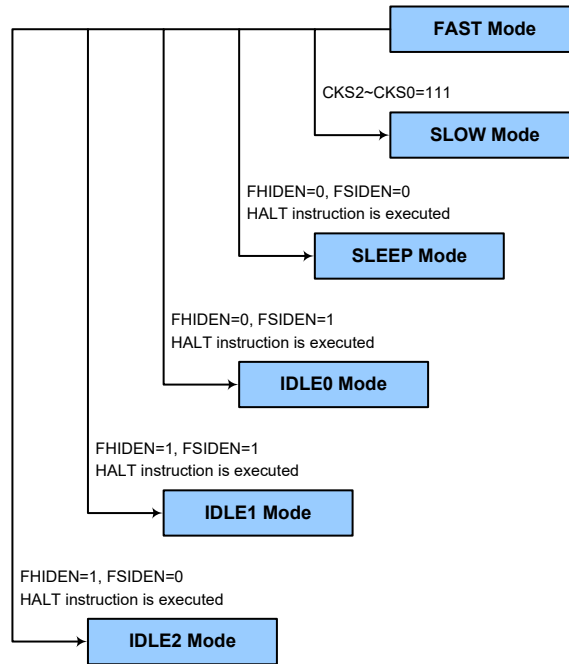
In simple terms, Mode Switching between the FAST Mode and SLOW Mode is executed using the CKS2~CKS0 bits in the SCC register while Mode Switching from the FAST/SLOW Modes to the SLEEP/IDLE Modes is executed via the HALT instruction. When a HALT instruction is executed, whether the device enters the IDLE Mode or the SLEEP Mode is determined by the condition of the FHIDEN and FSIDEN bits in the SCC register.



### FAST Mode to SLOW Mode Switching

When running in the FAST Mode, which uses the high speed system oscillator, and therefore consumes more power, the system clock can switch to run in the SLOW Mode by setting the CKS2~CKS0 bits to “111” in the SCC register. This will then use the low speed system oscillator which will consume less power. Users may decide to do this for certain operations which do not require high performance and can subsequently reduce power consumption.

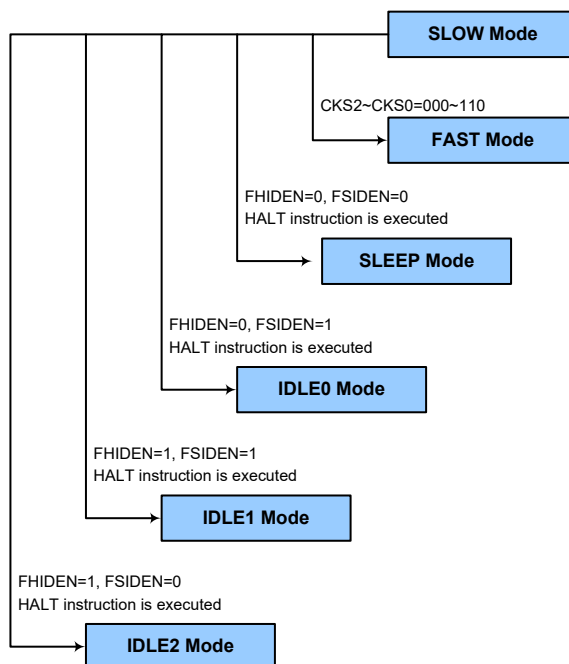
The SLOW Mode system clock is sourced from the LIRC oscillator and therefore requires this oscillator to be stable before full mode switching occurs.



### SLOW Mode to FAST Mode Switching

In SLOW mode the system clock is derived from  $f_{SUB}$ . When system clock is switched back to the FAST mode from  $f_{SUB}$ , the CKS2~CKS0 bits should be set to “000”~“110” and then the system clock will respectively be switched to  $f_H \sim f_H/64$ .

However, if  $f_H$  is not used in SLOW mode and thus switched off, it will take some time to re-oscillate and stabilise when switching to the FAST mode from the SLOW Mode. This is monitored using the HIRCF bit in the HIRCC register. The time duration required for the high speed system oscillator stabilization is specified in the System Start Up Time Characteristics.



#### Entering the SLEEP Mode

There is only one way for the device to enter the SLEEP Mode and that is to execute the “HALT” instruction in the application program with both the FHIDEN and FSIDEN bits in the SCC register equal to “0”. In this mode all the clocks and functions will be switched off except the WDT function. When this instruction is executed under the conditions described above, the following will occur:

- The system clock will be stopped and the application program will stop at the “HALT” instruction.
- The Data Memory contents and registers will maintain their present condition.
- The I/O ports will maintain their present conditions.
- In the status register, the Power Down flag, PDF, will be set and the Watchdog time-out flag, TO, will be cleared.
- The WDT will be cleared and resume counting if the WDT function is always enabled. If the WDT function is disabled, the WDT will be cleared and then stopped.

#### Entering the IDLE0 Mode

There is only one way for the device to enter the IDLE0 Mode and that is to execute the “HALT” instruction in the application program with the FHIDEN bit in the SCC register equal to “0” and the FSIDEN bit in the SCC register equal to “1”. When this instruction is executed under the conditions described above, the following will occur:

- The  $f_H$  clock will be stopped and the application program will stop at the “HALT” instruction, but the  $f_{SUB}$  clock will be on.
- The Data Memory contents and registers will maintain their present condition.
- The I/O ports will maintain their present conditions.
- In the status register, the Power Down flag, PDF, will be set and the Watchdog time-out flag, TO, will be cleared.
- The WDT will be cleared and resume counting if the WDT function is always enabled. If the WDT function is disabled, the WDT will be cleared and then stopped.

### Entering the IDLE1 Mode

There is only one way for the device to enter the IDLE1 Mode and that is to execute the “HALT” instruction in the application program with both the FHIDEN and FSIDEN bits in the SCC register equal to “1”. When this instruction is executed under the conditions described above, the following will occur:

- The  $f_{\text{H}}$  and  $f_{\text{SUB}}$  clocks will be on but the application program will stop at the “HALT” instruction.
- The Data Memory contents and registers will maintain their present condition.
- The I/O ports will maintain their present conditions.
- In the status register, the Power Down flag, PDF, will be set and the Watchdog time-out flag, TO, will be cleared.
- The WDT will be cleared and resume counting if the WDT function is always enabled. If the WDT function is disabled, the WDT will be cleared and then stopped.

### Entering the IDLE2 Mode

There is only one way for the device to enter the IDLE2 Mode and that is to execute the “HALT” instruction in the application program with the FHIDEN bit in the SCC register equal to “1” and the FSIDEN bit in the SCC register equal to “0”. When this instruction is executed under the conditions described above, the following will occur:

- The  $f_{\text{H}}$  clock will be on but the  $f_{\text{SUB}}$  clock will be off and the application program will stop at the “HALT” instruction.
- The Data Memory contents and registers will maintain their present condition.
- The I/O ports will maintain their present conditions.
- In the status register, the Power Down flag, PDF, will be set and the Watchdog time-out flag, TO, will be cleared.
- The WDT will be cleared and resume counting if the WDT function is always enabled. If the WDT function is disabled, the WDT will be cleared and then stopped.

### Standby Current Considerations

As the main reason for entering the SLEEP or IDLE Mode is to keep the current consumption of the device to as low a value as possible, perhaps only in the order of several micro-amps except in the IDLE1 and IDLE2 Modes, there are other considerations which must also be taken into account by the circuit designer if the power consumption is to be minimised. Special attention must be made to the I/O pins on the device. All high-impedance input pins must be connected to either a fixed high or low level as any floating input pins could create internal oscillations and result in increased current consumption. This also applies to the device which has different package types, as there may be unbonded pins. These pins must either be setup as outputs or if setup as inputs must have pull-high resistors connected.

Care must also be taken with the loads, which are connected to I/O pins, which are setup as outputs. These should be placed in a condition in which minimum current is drawn or connected only to external circuits that do not draw current, such as other CMOS inputs. Also note that additional standby current will also be required if the LIRC oscillator has been enabled.

In the IDLE1 and IDLE2 Modes the high speed oscillator is on, if the peripheral function clock source is derived from the high speed oscillator, the additional standby current will also be perhaps in the order of several hundred micro-amps.

## Wake-up

To minimise power consumption the device can enter the SLEEP or any IDLE Mode, where the CPU will be switched off. However, when the device is woken up again, it will take a considerable time for the original system oscillator to restart, stabilise and allow normal operation to resume.

After the system enters the SLEEP or IDLE Mode, it can be woken up from one of various sources listed as follows:

- An external falling edge on Port A
- A system interrupt
- A WDT overflow

When the device executes the “HALT” instruction, it will enter the IDLE or SLEEP mode and the PDF flag will be set high. The PDF flag is cleared to 0 if the device experiences a system power-up or executes the clear Watchdog Timer instruction.

If the system is woken up by a WDT overflow, a Watchdog Timer Time-out reset will be initiated and the TO flag will be set to 1. The TO flag is set high if a WDT time-out occurs, and causes a wake-up that only resets the Program Counter and Stack Pointer, the other flags remain in their original status.

Each pin on Port A can be setup using the PAWU register to permit a negative transition on the pin to wake up the system. When a pin wake-up occurs, the program will resume execution at the instruction following the “HALT” instruction.

If the system is woken up by an interrupt, then two possible situations may occur. The first is where the related interrupt is disabled or the interrupt is enabled but the stack is full, in which case the program will resume execution at the instruction following the “HALT” instruction. In this situation, the interrupt which woke up the device will not be immediately serviced, but will rather be serviced later when the related interrupt is finally enabled or when a stack level becomes free. The other situation is where the related interrupt is enabled and the stack is not full, in which case the regular interrupt response takes place. If an interrupt request flag is set high before entering the SLEEP or IDLE Mode, the wake-up function of the related interrupt will be disabled.

## Watchdog Timer

The Watchdog Timer is provided to prevent program malfunctions or sequences from jumping to unknown locations, due to certain uncontrollable external events such as electrical noise.

### Watchdog Timer Clock Source

The Watchdog Timer clock source is provided by the internal clock,  $f_{LIRC}$  which is sourced from the LIRC oscillator. The LIRC internal oscillator has an approximate frequency of 32kHz and this specified internal clock period can vary with  $V_{DD}$ , temperature and process variations. The Watchdog Timer source clock is then subdivided by a ratio of  $2^8$  to  $2^{15}$  to give longer timeouts, the actual value being chosen using the WS2~WS0 bits in the WDTC register.

### Watchdog Timer Control Register

A single register, WDTC, controls the required timeout period as well as the enable/disable and reset MCU operation.

#### • WDTC Register

| Bit  | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   |
|------|-----|-----|-----|-----|-----|-----|-----|-----|
| Name | WE4 | WE3 | WE2 | WE1 | WE0 | WS2 | WS1 | WS0 |
| R/W  | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| POR  | 0   | 1   | 0   | 1   | 0   | 0   | 1   | 1   |

Bit 7~3 **WE4~WE0**: WDT function software control

01010: Enable

10101: Disable

Other values: Reset MCU

When these bits are changed by the environmental noise or software setting to reset the microcontroller, the reset operation will be activated after a delay time,  $t_{SRESET}$  and the WRF bit in the RSTFC register will be set high.

Bit 2~0 **WS2~WS0**: WDT time-out period selection

000:  $2^8/f_{LIRC}$

001:  $2^9/f_{LIRC}$

010:  $2^{10}/f_{LIRC}$

011:  $2^{11}/f_{LIRC}$

100:  $2^{12}/f_{LIRC}$

101:  $2^{13}/f_{LIRC}$

110:  $2^{14}/f_{LIRC}$

111:  $2^{15}/f_{LIRC}$

These three bits determine the division ratio of the watchdog timer source clock, which in turn determines the time-out period.

#### • RSTFC Register

| Bit  | 7 | 6 | 5 | 4 | 3 | 2    | 1   | 0   |
|------|---|---|---|---|---|------|-----|-----|
| Name | — | — | — | — | — | LVRF | LRF | WRF |
| R/W  | — | — | — | — | — | R/W  | R/W | R/W |
| POR  | — | — | — | — | — | x    | 0   | 0   |

“x”: unknown

Bit 7~3 Unimplemented, read as “0”

Bit 2 **LVRF**: LVR function reset flag

Refer to the Low Voltage Reset section.

Bit 1 **LRF**: LVR control register software reset flag

Refer to the Low Voltage Reset section.



Bit 0      **WRF**: WDTC register software reset flag  
 0: Not occurred  
 1: Occurred

This bit is set to 1 by the WDTC register software reset and cleared to zero by the application program. Note that this bit can be cleared to zero only by the application program.

### Watchdog Timer Operation

The Watchdog Timer operates by providing a device reset when its timer overflows. This means that in the application program and during normal operation the user has to strategically clear the Watchdog Timer before it overflows to prevent the Watchdog Timer from executing a reset. This is done using the clear watchdog instruction. If the program malfunctions for whatever reason, jumps to an unknown location, or enters an endless loop, the clear instruction will not be executed in the correct manner, in which case the Watchdog Timer will overflow and reset the device. There are five bits, WE4~WE0, in the WDTC register to offer the enable/disable and reset control of the Watchdog Timer. The WDT function will be disabled when the WE4~WE0 bits are set to a value of 10101B while the WDT function will be enabled if the WE4~WE0 bits are equal to 01010B. If the WE4~WE0 bits are set to any other values, other than 01010B and 10101B, it will reset the device after a delay time,  $t_{SRESET}$ . After power-on these bits will have a value of 01010B.

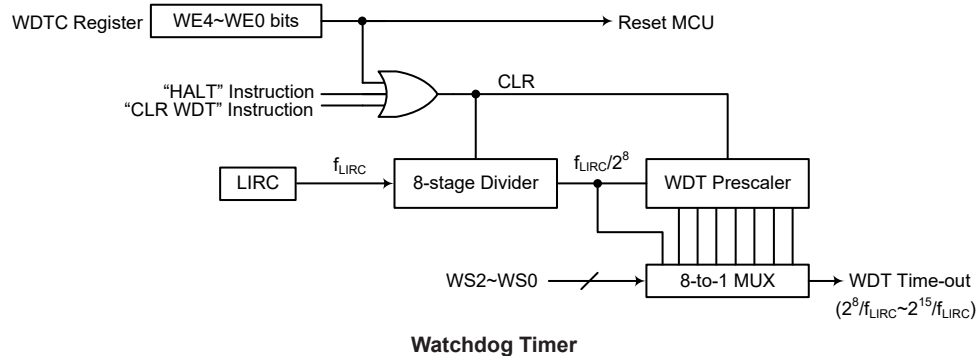
| WE4~WE0 Bits     | WDT Function |
|------------------|--------------|
| 10101B           | Disable      |
| 01010B           | Enable       |
| Any other values | Reset MCU    |

**Watchdog Timer Enable/Disable/Reset Control**

Under normal program operation, a Watchdog Timer time-out will initialise a device reset and set the status bit TO high. However, if the system is in the SLEEP or IDLE Mode, when a Watchdog Timer time-out occurs, the TO bit in the status register will be set high and only the Program Counter and Stack Pointer will be reset. Three methods can be adopted to clear the contents of the Watchdog Timer. The first is a WDTC software reset, which means a certain value except 01010B and 10101B written into the WE4~WE0 bits, the second is using the Watchdog Timer software clear instruction and the third is via a HALT instruction.

There is only one method of using software instruction to clear the Watchdog Timer. That is to use the single “CLR WDT” instruction to clear the WDT.

The maximum time out period is when the  $2^{15}$  division ratio is selected. As an example, with a 32kHz LIRC oscillator as its source clock, this will give a maximum watchdog period of around 1 second for the  $2^{15}$  division ratio, and a minimum timeout of 8ms for the  $2^8$  division ratio.



## Reset and Initialisation

A reset function is a fundamental part of any microcontroller ensuring that the device can be set to some predetermined condition irrespective of outside parameters. The most important reset condition is after power is first applied to the microcontroller. In this case, internal circuitry will ensure that the microcontroller, after a short delay, will be in a well-defined state and ready to execute the first program instruction. After this power-on reset, certain important internal registers will be set to defined states before the program commences. One of these registers is the Program Counter, which will be reset to zero forcing the microcontroller to begin program execution from the lowest Program Memory address.

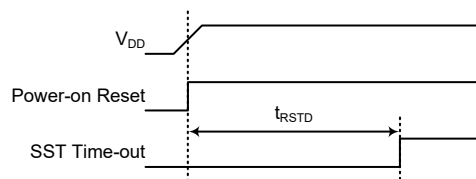
Another reset exists in the form of a Low Voltage Reset, LVR, where a full reset is implemented in situations where the power supply voltage falls below a certain threshold. Another type of reset is when the Watchdog Timer overflows and resets the microcontroller. All types of reset operations result in different register conditions being setup.

### Reset Functions

There are several ways in which a microcontroller reset can occur, through events occurring internally.

#### Power-on Reset

The most fundamental and unavoidable reset is the one that occurs after power is first applied to the microcontroller. As well as ensuring that the Program Memory begins execution from the first memory address, a power-on reset also ensures that certain other registers are preset to known conditions. All the I/O port and port control registers will power up in a high condition ensuring that all pins will be first set to inputs.

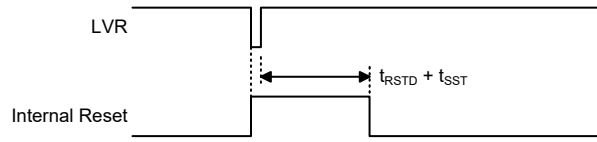


**Power-on Reset Timing Chart**

#### Low Voltage Reset – LVR

The microcontroller contains a low voltage reset circuit in order to monitor the supply voltage of the device and provide an MCU reset when the value falls below a certain predefined level. If the supply voltage of the device drops to within a range of  $0.9V \sim V_{LVR}$  such as might occur when changing the battery in battery powered applications, the LVR will automatically reset the device internally and the LVRF bit in the RSTFC register will also be set to 1. For a valid LVR signal, a low supply voltage, i.e., a voltage in the range between  $0.9V \sim V_{LVR}$  must exist for a time greater than that specified by  $t_{LVR}$  in the LVR Electrical Characteristics. If the low supply voltage state does not exceed this value, the LVR will ignore the low supply voltage and will not perform a reset function. If the LVS7~LVS0 bits are set to 01011010B, the LVR function is enabled with a fixed LVR voltage of 1.7V. If the LVS7~LVS0 bits are set to 10100101B, the LVR function is disabled. If the LVS7~LVS0 bits are changed to other values by environmental noise, the LVR will reset the device after a delay time,  $t_{SRESET}$ . When this happens, the LRF bit in the RSTFC register will be set high. After power-on the register will have the value of 01011010B.

Note that the LVR function will be automatically disabled when the device enters the SLEEP or IDLE mode.



**Low Voltage Reset Timing Chart**

• **LVRC Register**

| Bit  | 7    | 6    | 5    | 4    | 3    | 2    | 1    | 0    |
|------|------|------|------|------|------|------|------|------|
| Name | LVS7 | LVS6 | LVS5 | LVS4 | LVS3 | LVS2 | LVS1 | LVS0 |
| R/W  | R/W  | R/W  | R/W  | R/W  | R/W  | R/W  | R/W  | R/W  |
| POR  | 0    | 1    | 0    | 1    | 1    | 0    | 1    | 0    |

Bit 7~0 **LVS7~LVS0**: LVR voltage select  
 01011010: 1.7V  
 10100101: LVR disable

Other values: Generates a MCU reset – register is reset to POR value

When an actual low voltage condition occurs, as specified by the defined LVR voltage value above, an MCU reset will be generated. The reset operation will be activated after the low voltage condition keeps more than a  $t_{LVR}$  time. In this situation the register contents will remain the same after such a reset occurs.

Any register value, other than 01011010B and 10100101B values, will also result in the generation of an MCU reset. The reset operation will be activated after a delay time,  $t_{RESET}$ . However, in this situation the register contents will be reset to the POR value.

• **VBGC Register**

| Bit  | 7 | 6 | 5 | 4 | 3     | 2 | 1 | 0 |
|------|---|---|---|---|-------|---|---|---|
| Name | — | — | — | — | VBGEN | — | — | — |
| R/W  | — | — | — | — | R/W   | — | — | — |
| POR  | — | — | — | — | 0     | — | — | — |

Bit 7~4 Unimplemented, read as “0”

Bit 3 **VBGEN**: Bandgap Buffer Control  
 0: Disable  
 1: Enable

Note that the Bandgap circuit is enabled when the LVR function is enabled or when the VBGEN bit is set to 1.

Bit 2~0 Unimplemented, read as “0”

• **RSTFC Register**

| Bit  | 7 | 6 | 5 | 4 | 3 | 2    | 1   | 0   |
|------|---|---|---|---|---|------|-----|-----|
| Name | — | — | — | — | — | LVRF | LRF | WRF |
| R/W  | — | — | — | — | — | R/W  | R/W | R/W |
| POR  | — | — | — | — | — | x    | 0   | 0   |

“x”: unknown

Bit 7~3 Unimplemented, read as “0”

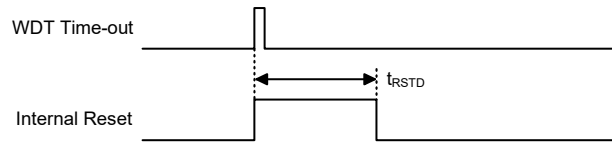
Bit 2 **LVRF**: LVR function reset flag  
 0: Not occurred  
 1: Occurred

This bit is set to 1 when an actual Low Voltage Reset situation condition occurs. This bit can only be cleared to zero by the application program.

- Bit 1     **LRF**: LVR control register software reset flag  
           0: Not occurred  
           1: Occurred  
           This bit is set to 1 by the LVRC control register containing any undefined LVR voltage register values. This in effect acts like a software reset function. Note that this bit can only be cleared to zero by the application program.
- Bit 0     **WRF**: WDTC register software reset flag  
           Refer to the Watchdog Timer Control Register section.

**Watchdog Time-out Reset during Normal Operation**

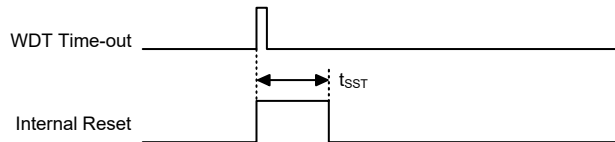
When the Watchdog Time-out Reset during normal operation in the FAST or SLOW mode occurs, the Watchdog time-out flag TO will be set to “1”.



**WDT Time-out Reset during Normal Operation Timing Chart**

**Watchdog Time-out Reset during SLEEP or IDLE Mode**

The Watchdog Time-out Reset during SLEEP or IDLE Mode is a little different from other kinds of reset. Most of the conditions remain unchanged except that the Program Counter and the Stack Pointer will be cleared to “0” and the TO flag will be set to “1”. Refer to the System Start Up Time Characteristics for  $t_{SST}$  details.



**WDT Time-out Reset during SLEEP or IDLE Timing Chart**

**Reset Initial Conditions**

The different types of reset described affect the reset flags in different ways. These flags, known as PDF and TO are located in the status register and are controlled by various microcontroller operations, such as the SLEEP or IDLE Mode function or Watchdog Timer. The reset flags are shown in the table:

| TO | PDF | Reset Conditions                                       |
|----|-----|--|
| 0  | 0   | Power-on reset   |
| u  | u   | LVR reset during FAST or SLOW Mode operation           |
| 1  | u   | WDT time-out reset during FAST or SLOW Mode operation  |
| 1  | 1   | WDT time-out reset during IDLE or SLEEP Mode operation |

“u” stands for unchanged

The following table indicates the way in which the various components of the microcontroller are affected after a power-on reset occurs.

| Item               | Condition after Reset                            |
|--------------------|--|
| Program Counter    | Reset to zero                                    |
| Interrupts         | All interrupts will be disabled                  |
| WDT, Time Bases    | Cleared after reset, WDT begins counting         |
| Timer Modules      | Timer Modules will be turned off                 |
| Input/Output Ports | I/O ports will be setup as inputs                |
| Stack Pointer      | Stack Pointer will point to the top of the stack |

The different kinds of resets all affect the internal registers of the microcontroller in different ways. To ensure reliable continuation of normal program execution after a reset occurs, it is important to know what condition the microcontroller is in after a particular reset occurs. The following table describes how each type of reset affects each of the microcontroller internal registers. Note that where more than one package type exists the table will reflect the situation for the larger package type.

| Register | Power-on Reset  | WDT Time-out (Normal Operation) | WDT Time-out (IDLE/SLEEP) |
|----------|-----------------|---------------------------------|---------------------------|
| IAR0     | x x x x x x x x | u u u u u u u u                 | u u u u u u u u           |
| MP0      | x x x x x x x x | u u u u u u u u                 | u u u u u u u u           |
| IAR1     | x x x x x x x x | u u u u u u u u                 | u u u u u u u u           |
| MP1      | x x x x x x x x | u u u u u u u u                 | u u u u u u u u           |
| BP       | - - - - - - 0   | - - - - - - 0                   | - - - - - - u             |
| ACC      | x x x x x x x x | u u u u u u u u                 | u u u u u u u u           |
| PCL      | 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0                 | 0 0 0 0 0 0 0 0           |
| TBLP     | x x x x x x x x | u u u u u u u u                 | u u u u u u u u           |
| TBLH     | x x x x x x x x | u u u u u u u u                 | u u u u u u u u           |
| TBHP     | - - - - x x x x | - - - - u u u u                 | - - - - u u u u           |
| STATUS   | --00 x x x x    | --1u u u u u                    | --11 u u u u              |
| SCC      | 000- - -00      | 000- - -00                      | u u u- - -u u             |
| HIRCC    | - - - - - - 0 1 | - - - - - - 0 1                 | - - - - - - u u           |
| LVRC     | 0101 1010       | 0101 1010                       | u u u u u u u u           |
| INTEG    | - - - - - - 0 0 | - - - - - - 0 0                 | - - - - - - u u           |
| RSTFC    | - - - - - x 0 0 | - - - - - u u u                 | - - - - - u u u           |
| INTC0    | -000 0000       | -000 0000                       | -u u u u u u u            |
| INTC1    | 0000 0000       | 0000 0000                       | u u u u u u u u           |
| INTC2    | 0000 0000       | 0000 0000                       | u u u u u u u u           |
| INTC3    | - - - 0 - - - 0 | - - - 0 - - - 0                 | - - - u - - - u           |
| PA       | 1111 1111       | 1111 1111                       | u u u u u u u u           |
| PAC      | 1111 1111       | 1111 1111                       | u u u u u u u u           |
| PAPU     | 0000 0000       | 0000 0000                       | u u u u u u u u           |
| PAWU     | 0000 0000       | 0000 0000                       | u u u u u u u u           |
| PB       | 1111 1111       | 1111 1111                       | u u u u u u u u           |
| PBC      | 1111 1111       | 1111 1111                       | u u u u u u u u           |
| PBPU     | 0000 0000       | 0000 0000                       | u u u u u u u u           |
| PC       | - - - - 1 1 1 1 | - - - - 1 1 1 1                 | - - - - u u u u           |
| PCC      | - - - - 1 1 1 1 | - - - - 1 1 1 1                 | - - - - u u u u           |
| PCPU     | - - - - 0 0 0 0 | - - - - 0 0 0 0                 | - - - - u u u u           |
| VBGC     | - - - - 0 - - - | - - - - 0 - - -                 | - - - - u - - -           |

| Register | Power-on Reset | WDT Time-out<br>(Normal Operation) | WDT Time-out<br>(IDLE/SLEEP) |
|----------|----------------|------------------------------------|------------------------------|
| WDTC     | 0101 0011      | 0101 0011                          | uuuu uuuu                    |
| PSC0R    | ---- --00      | ---- --00                          | ---- --uu                    |
| TB0C     | 0--- -000      | 0--- -000                          | u--- -uuu                    |
| PSC1R    | ---- --00      | ---- --00                          | ---- --uu                    |
| TB1C     | 0--- -000      | 0--- -000                          | u--- -uuu                    |
| EEA      | ---0 0000      | ---0 0000                          | ---u uuuu                    |
| EED      | 0000 0000      | 0000 0000                          | uuuu uuuu                    |
| PAS0     | 0000 0000      | 0000 0000                          | uuuu uuuu                    |
| PAS1     | 0000 0000      | 0000 0000                          | uuuu uuuu                    |
| PBS0     | 0000 0000      | 0000 0000                          | uuuu uuuu                    |
| PBS1     | 0000 0000      | 0000 0000                          | uuuu uuuu                    |
| PCS0     | --00 0000      | --00 0000                          | --uu uuuu                    |
| TKTMR    | 0000 0000      | 0000 0000                          | uuuu uuuu                    |
| TKC0     | -000 0-00      | -000 0-00                          | -uuu u-uu                    |
| TKC1     | ---- --11      | ---- --11                          | ---- --uu                    |
| TK16DL   | 0000 0000      | 0000 0000                          | uuuu uuuu                    |
| TK16DH   | 0000 0000      | 0000 0000                          | uuuu uuuu                    |
| TKM0C0   | 0000 0000      | 0000 0000                          | uuuu uuuu                    |
| TKM0C1   | 0-00 0000      | 0-00 0000                          | u-uu uuuu                    |
| TKM016DL | 0000 0000      | 0000 0000                          | uuuu uuuu                    |
| TKM016DH | 0000 0000      | 0000 0000                          | uuuu uuuu                    |
| TKM0ROL  | 0000 0000      | 0000 0000                          | uuuu uuuu                    |
| TKM0ROH  | ---- --00      | ---- --00                          | ---- --uu                    |
| USR      | 0000 1011      | 0000 1011                          | uuuu uuuu                    |
| UCR1     | 0000 00x0      | 0000 00x0                          | uuuu uuuu                    |
| UCR2     | 0000 0000      | 0000 0000                          | uuuu uuuu                    |
| UCR3     | ---- ---0      | ---- ---0                          | ---- ---u                    |
| BRDH     | 0000 0000      | 0000 0000                          | uuuu uuuu                    |
| BRDL     | 0000 0000      | 0000 0000                          | uuuu uuuu                    |
| UFCR     | --00 0000      | --00 0000                          | --uu uuuu                    |
| TXR_RXR  | xxxx xxxx      | xxxx xxxx                          | uuuu uuuu                    |
| RxCNT    | ---- -000      | ---- -000                          | ---- -uuu                    |
| CTMC0    | 0000 0000      | 0000 0000                          | uuuu uuuu                    |
| CTMC1    | 0000 0000      | 0000 0000                          | uuuu uuuu                    |
| CTMDL    | 0000 0000      | 0000 0000                          | uuuu uuuu                    |
| CTMDH    | ---- --00      | ---- --00                          | ---- --uu                    |
| CTMAL    | 0000 0000      | 0000 0000                          | uuuu uuuu                    |
| CTMAH    | ---- --00      | ---- --00                          | ---- --uu                    |
| STMC0    | 0000 0000      | 0000 0000                          | uuuu uuuu                    |
| STMC1    | 0000 0000      | 0000 0000                          | uuuu uuuu                    |
| STMDL    | 0000 0000      | 0000 0000                          | uuuu uuuu                    |
| STMDH    | ---- --00      | ---- --00                          | ---- --uu                    |
| STMAL    | 0000 0000      | 0000 0000                          | uuuu uuuu                    |
| STMAH    | ---- --00      | ---- --00                          | ---- --uu                    |
| OPSWA    | 0000 0000      | 0000 0000                          | uuuu uuuu                    |
| OPSWB    | 0000 0000      | 0000 0000                          | uuuu uuuu                    |
| OPSWC    | 0000 0000      | 0000 0000                          | uuuu uuuu                    |

| Register | Power-on Reset | WDT Time-out<br>(Normal Operation) | WDT Time-out<br>(IDLE/SLEEP) |
|----------|----------------|------------------------------------|------------------------------|
| OPSWD    | 0000 0000      | 0000 0000                          | uuuu uuuu                    |
| OPSWE    | 0000 0000      | 0000 0000                          | uuuu uuuu                    |
| OPDC0    | 0000 0000      | 0000 0000                          | uuuu uuuu                    |
| OPDC1    | 0000 0000      | 0000 0000                          | uuuu uuuu                    |
| OPDDA    | 0000 0000      | 0000 0000                          | uuuu uuuu                    |
| OPDA0CAL | 0010 0000      | 0010 0000                          | uuuu uuuu                    |
| OPDA1CAL | 0010 0000      | 0010 0000                          | uuuu uuuu                    |
| OPDCCAL  | 0001 0000      | 0001 0000                          | uuuu uuuu                    |
| SADC0    | 0000 0000      | 0000 0000                          | uuuu uuuu                    |
| SADC1    | 0000 0000      | 0000 0000                          | uuuu uuuu                    |
| SADC2    | 0000 0000      | 0000 0000                          | uuuu uuuu                    |
| SADOH    | xxxx xxxx      | xxxx xxxx                          | uuuu uuuu<br>(ADRFS=0)       |
|          |                |                                    | ---- uuuu<br>(ADRFS=1)       |
| SADOL    | xxxx ----      | xxxx ----                          | uuuu ----<br>(ADRFS=0)       |
|          |                |                                    | uuuu uuuu<br>(ADRFS=1)       |
| ISGENC   | 0 --- --00     | 0 --- --00                         | u --- --uu                   |
| ISGDATA0 | ---0 0000      | ---0 0000                          | ---u uuuu                    |
| ISGDATA1 | ---0 0000      | ---0 0000                          | ---u uuuu                    |
| LVPUUC   | ---- ---0      | ---- ---0                          | ---- ---u                    |
| ORMC     | 0000 0000      | 0000 0000                          | 0000 0000                    |
| IFS      | ---- --00      | ---- --00                          | ---- --uu                    |
| EEC      | 0 --- 0000     | 0 --- 0000                         | u --- uuuu                   |

Note: “u” stands for unchanged  
“x” stands for unknown  
“-” stands for unimplemented

## Input/Output Ports

Holtek microcontrollers offer considerable flexibility on their I/O ports. With the input or output designation of every pin fully under user program control, pull-high selections for all ports and wake-up selections on certain pins, the user is provided with an I/O structure to meet the needs of a wide range of application possibilities.

The device provides bidirectional input/output lines labeled with port names PA~PC. These I/O ports are mapped to the RAM Data Memory with specific addresses as shown in the Special Purpose Data Memory table. All of these I/O ports can be used for input and output operations. For input operation, these ports are non-latching, which means the inputs must be ready at the T2 rising edge of instruction “MOV A, [m]”, where m denotes the port address. For output operation, all the data is latched and remains unchanged until the output latch is rewritten.

| Register Name | Bit   |       |       |       |       |       |       |       |
|---------------|-------|-------|-------|-------|-------|-------|-------|-------|
|               | 7     | 6     | 5     | 4     | 3     | 2     | 1     | 0     |
| PA            | PA7   | PA6   | PA5   | PA4   | PA3   | PA2   | PA1   | PA0   |
| PAC           | PAC7  | PAC6  | PAC5  | PAC4  | PAC3  | PAC2  | PAC1  | PAC0  |
| PAPU          | PAPU7 | PAPU6 | PAPU5 | PAPU4 | PAPU3 | PAPU2 | PAPU1 | PAPU0 |
| PAWU          | PAWU7 | PAWU6 | PAWU5 | PAWU4 | PAWU3 | PAWU2 | PAWU1 | PAWU0 |
| PB            | PB7   | PB6   | PB5   | PB4   | PB3   | PB2   | PB1   | PB0   |
| PBC           | PBC7  | PBC6  | PBC5  | PBC4  | PBC3  | PBC2  | PBC1  | PBC0  |
| PBPU          | PBPU7 | PBPU6 | PBPU5 | PBPU4 | PBPU3 | PBPU2 | PBPU1 | PBPU0 |
| PC            | —     | —     | —     | —     | PC3   | PC2   | PC1   | PC0   |
| PCC           | —     | —     | —     | —     | PCC3  | PCC2  | PCC1  | PCC0  |
| PCPU          | —     | —     | —     | —     | PCPU3 | PCPU2 | PCPU1 | PCPU0 |
| LVPUC         | —     | —     | —     | —     | —     | —     | —     | LVPUC |

“—”: Unimplemented, read as “0”

### I/O Logic Function Register List

## Pull-high Resistors

Many product applications require pull-high resistors for their switch inputs usually requiring the use of an external resistor. To eliminate the need for these external resistors, all I/O pins, when configured as a digital input have the capability of being connected to an internal pull-high resistor. These pull-high resistors are selected using the PxPU and LVPUC registers, and are implemented using weak PMOS transistors. The PxPU register is used to determine whether the pull-high function is enabled or not while the LVPUC register is used to select the pull-high resistor value for low voltage power supply applications.

Note that the pull-high resistor can be controlled by the relevant pull-high control register only when the pin-shared functional pin is selected as a digital input or NMOS output. Otherwise, the pull-high resistors cannot be enabled.

### • PxPU Register

| Bit  | 7     | 6     | 5     | 4     | 3     | 2     | 1     | 0     |
|------|-------|-------|-------|-------|-------|-------|-------|-------|
| Name | PxPU7 | PxPU6 | PxPU5 | PxPU4 | PxPU3 | PxPU2 | PxPU1 | PxPU0 |
| R/W  | R/W   | R/W   | R/W   | R/W   | R/W   | R/W   | R/W   | R/W   |
| POR  | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     |

**PxPU**n: I/O Port x Pin pull-high function control

0: Disable

1: Enable



The PxPUn bit is used to control the pin pull-high function. Here the “x” can be A, B and C. However, the actual available bits for each I/O Port may be different.

For the PB4 pin, there is an internal pull-low resistor which is controlled by configuration option. If its pull-low and pull-high functions are both enabled, additional power consumption will be required.

• **LVPUC Register**

| Bit  | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0    |
|------|---|---|---|---|---|---|---|------|
| Name | — | — | — | — | — | — | — | LVPU |
| R/W  | — | — | — | — | — | — | — | R/W  |
| POR  | — | — | — | — | — | — | — | 0    |

Bit 7~1 Unimplemented, read as “0”

Bit 0 **LVPU**: Pull-high resistor selection for low voltage power supply

0: All pin pull-high resistors are 60kΩ @ 3V

1: All pin pull-high resistors are 15kΩ @ 3V

The LVPU bit is used to select the pull-high resistor value for low voltage power supply applications. Note that this bit is only available when the corresponding pin pull-high function is enabled. If the pull-high function is disabled, the LVPU bit has no effect on selecting the pull-high resistor value.

Note that the PB4 pin does not support this pull-high resistor value selection function. Refer to the Input/Output Characteristics section for its pull-high resistor value details.

**Port A Wake-up**

The HALT instruction forces the microcontroller into the SLEEP or IDLE Mode which preserves power, a feature that is important for battery and other low-power applications. Various methods exist to wake up the microcontroller, one of which is to change the logic condition on one of the Port A pins from high to low. This function is especially suitable for applications that can be woken up via external switches. Each pin on Port A can be selected individually to have this wake-up feature using the PAWU register.

Note that the wake-up function can be controlled by the wake-up control register only when the pin is selected as a general purpose input and the MCU enters the IDLE or SLEEP mode.

• **PAWU Register**

| Bit  | 7     | 6     | 5     | 4     | 3     | 2     | 1     | 0     |
|------|-------|-------|-------|-------|-------|-------|-------|-------|
| Name | PAWU7 | PAWU6 | PAWU5 | PAWU4 | PAWU3 | PAWU2 | PAWU1 | PAWU0 |
| R/W  | R/W   | R/W   | R/W   | R/W   | R/W   | R/W   | R/W   | R/W   |
| POR  | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     |

Bit 7~0 **PAWU7~PAWU0**: PA7~PA0 wake-up function control

0: Disable

1: Enable

**I/O Port Control Registers**

Each I/O port has its own control register known as PAC~PCC, to control the input/output configuration. With this control register, each CMOS output or input can be reconfigured dynamically under software control. Each pin of the I/O ports is directly mapped to a bit in its associated port control register. For the I/O pin to function as an input, the corresponding bit of the control register must be written as a “1”. This will then allow the logic state of the input pin to be directly read by instructions. When the corresponding bit of the control register is written as a “0”, the I/O pin will be setup as a CMOS output. If the pin is currently setup as an output, instructions can still be used to read the output register. However, it should be noted that the program will in fact only read the status of the output data latch and not the actual logic status of the output pin.

• **PxC Register**

| Bit  | 7    | 6    | 5    | 4    | 3    | 2    | 1    | 0    |
|------|------|------|------|------|------|------|------|------|
| Name | PxC7 | PxC6 | PxC5 | PxC4 | PxC3 | PxC2 | PxC1 | PxC0 |
| R/W  | R/W  | R/W  | R/W  | R/W  | R/W  | R/W  | R/W  | R/W  |
| POR  | 1    | 1    | 1    | 1    | 1    | 1    | 1    | 1    |

**PxCn:** I/O Port x Pin type selection

0: Output

1: Input

The PxCn bit is used to control the pin type selection. Here the “x” can be A, B and C. However, the actual available bits for each I/O Port may be different.

For the PB4 pin, there is an internal pull-low resistor which is controlled by configuration option. If it is configured to output high level with its pull-low resistor enabled, additional power consumption will be required.

**Pin-shared Functions**

The flexibility of the microcontroller range is greatly enhanced by the use of pins that have more than one function. Limited numbers of pins can force serious design constraints on designers but by supplying pins with multi-functions, many of these difficulties can be overcome. For these pins, the desired function of the multi-function I/O pins is selected by a series of registers via the application program control.

**Pin-shared Function Selection Registers**

The limited number of supplied pins in a package can impose restrictions on the amount of functions a certain device can contain. However, by allowing the same pins to share several different functions and providing a means of function selection, a wide range of different functions can be incorporated into even relatively small package sizes. The device includes Port “x” Output Function Selection register “n”, labeled as PxCn, and Input Function Selection register, labeled as IFS, which can select the desired functions of the multi-function pin-shared pins.

The most important point to note is to make sure that the desired pin-shared function is properly selected and also deselected. For most pin-shared functions, to select the desired pin-shared function, the pin-shared function should first be correctly selected using the corresponding pin-shared control register. After that the corresponding peripheral functional setting should be configured and then the peripheral function can be enabled. However, a special point must be noted for some digital input pins, such as INT, xTCK, etc, which share the same pin-shared control configuration with their corresponding general purpose I/O functions when setting the relevant pin-shared control bit fields. To select these pin functions, in addition to the necessary pin-shared control and peripheral functional setup aforementioned, they must also be set as an input by setting the corresponding bit in the I/O port control register. To correctly deselect the pin-shared function, the peripheral function should first be disabled and then the corresponding pin-shared function control register can be modified to select other pin-shared functions.

| Register Name | Bit   |       |       |       |       |       |       |       |
|---------------|-------|-------|-------|-------|-------|-------|-------|-------|
|               | 7     | 6     | 5     | 4     | 3     | 2     | 1     | 0     |
| PAS0          | PAS07 | PAS06 | PAS05 | PAS04 | PAS03 | PAS02 | PAS01 | PAS00 |
| PAS1          | PAS17 | PAS16 | PAS15 | PAS14 | PAS13 | PAS12 | PAS11 | PAS10 |
| PBS0          | PBS07 | PBS06 | PBS05 | PBS04 | PBS03 | PBS02 | PBS01 | PBS00 |
| PBS1          | PBS17 | PBS16 | PBS15 | PBS14 | PBS13 | PBS12 | PBS11 | PBS10 |
| PCS0          | —     | —     | PCS05 | PCS04 | PCS03 | PCS02 | PCS01 | PCS00 |
| IFS           | —     | —     | —     | —     | —     | —     | IFS1  | IFS0  |

**Pin-shared Function Selection Register List**

• **PAS0 Register**

| Bit  | 7     | 6     | 5     | 4     | 3     | 2     | 1     | 0     |
|------|-------|-------|-------|-------|-------|-------|-------|-------|
| Name | PAS07 | PAS06 | PAS05 | PAS04 | PAS03 | PAS02 | PAS01 | PAS00 |
| R/W  | R/W   | R/W   | R/W   | R/W   | R/W   | R/W   | R/W   | R/W   |
| POR  | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     |

- Bit 7~6     **PAS07~PAS06:** PA3 Pin-shared function selection  
 00: PA3/STCK  
 01: OPDA1P  
 10: AN0  
 11: PA3/STCK
- Bit 5~4     **PAS05~PAS04:** PA2 Pin-shared function selection  
 00: PA2  
 01: OPDA0N  
 10: VBAT  
 11: RX/TX
- Bit 3~2     **PAS03~PAS02:** PA1 Pin-shared function selection  
 00: PA1  
 01: OPDA0O  
 10: AN1  
 11: CTP
- Bit 1~0     **PAS01~PAS00:** PA0 Pin-shared function selection  
 00: PA0/INT  
 01: OPDA1N  
 10: TX  
 11: PA0/INT

• **PAS1 Register**

| Bit  | 7     | 6     | 5     | 4     | 3     | 2     | 1     | 0     |
|------|-------|-------|-------|-------|-------|-------|-------|-------|
| Name | PAS17 | PAS16 | PAS15 | PAS14 | PAS13 | PAS12 | PAS11 | PAS10 |
| R/W  | R/W   | R/W   | R/W   | R/W   | R/W   | R/W   | R/W   | R/W   |
| POR  | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     |

- Bit 7~6     **PAS17~PAS16:** PA7 Pin-shared function selection  
 00: PA7  
 01: VREF  
 10: AN4  
 11: PA7
- Bit 5~4     **PAS15~PAS14:** PA6 Pin-shared function selection  
 00: PA6  
 01: OPDA0P  
 10: AN2  
 11: CTPB
- Bit 3~2     **PAS13~PAS12:** PA5 Pin-shared function selection  
 00: PA5  
 01: OPDA1O  
 10: AN7  
 11: PA5
- Bit 1~0     **PAS11~PAS10:** PA4 Pin-shared function selection  
 00: PA4  
 01: OPDA0P  
 10: PA4  
 11: PA4

• **PBS0 Register**

| Bit  | 7     | 6     | 5     | 4     | 3     | 2     | 1     | 0     |
|------|-------|-------|-------|-------|-------|-------|-------|-------|
| Name | PBS07 | PBS06 | PBS05 | PBS04 | PBS03 | PBS02 | PBS01 | PBS00 |
| R/W  | R/W   | R/W   | R/W   | R/W   | R/W   | R/W   | R/W   | R/W   |
| POR  | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     |

Bit 7~6     **PBS07~PBS06:** PB3 Pin-shared function selection

00: PB3  
01: KEY4  
10: DACOUT  
11: STPB

Bit 5~4     **PBS05~PBS04:** PB2 Pin-shared function selection

00: PB2  
01: AN3  
10: KEY3  
11: STP

Bit 3~2     **PBS03~PBS02:** PB1 Pin-shared function selection

00: PB1  
01: AN6  
10: KEY2  
11: TX

Bit 1~0     **PBS01~PBS00:** PB0 Pin-shared function selection

00: PB0  
01: AN5  
10: KEY1  
11: RX/TX

• **PBS1 Register**

| Bit  | 7     | 6     | 5     | 4     | 3     | 2     | 1     | 0     |
|------|-------|-------|-------|-------|-------|-------|-------|-------|
| Name | PBS17 | PBS16 | PBS15 | PBS14 | PBS13 | PBS12 | PBS11 | PBS10 |
| R/W  | R/W   | R/W   | R/W   | R/W   | R/W   | R/W   | R/W   | R/W   |
| POR  | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     |

Bit 7~6     **PBS17~PBS16:** PB7 Pin-shared function selection

00: PB7  
01: STPB  
10: CTPB  
11: PB7

Bit 5~4     **PBS15~PBS14:** PB6 Pin-shared function selection

00: PB6  
01: STP  
10: CTP  
11: PB6

Bit 3~2     **PBS13~PBS12:** PB5 Pin-shared function selection

00: PB5  
01: VBAT  
10: CTP  
11: PB5

Bit 1~0     **PBS11~PBS10:** PB4 Pin-shared function selection

00: PB4  
01: STP  
10: CTP  
11: PB4

• **PCS0 Register**

| Bit  | 7 | 6 | 5     | 4     | 3     | 2     | 1     | 0     |
|------|---|---|-------|-------|-------|-------|-------|-------|
| Name | — | — | PCS05 | PCS04 | PCS03 | PCS02 | PCS01 | PCS00 |
| R/W  | — | — | R/W   | R/W   | R/W   | R/W   | R/W   | R/W   |
| POR  | — | — | 0     | 0     | 0     | 0     | 0     | 0     |

- Bit 7~6 Unimplemented, read as “0”
- Bit 5~4 **PCS05~PCS04**: PC2 Pin-shared function selection  
 00: PC2  
 01: RX/TX  
 10: PC2  
 11: PC2
- Bit 3~2 **PCS03~PCS02**: PC1 Pin-shared function selection  
 00: PC1  
 01: TX  
 10: PC1  
 11: PC1
- Bit 1~0 **PCS01~PCS00**: PC0 Pin-shared function selection  
 00: PC0  
 01: OPDA0P  
 10: PC0  
 11: PC0

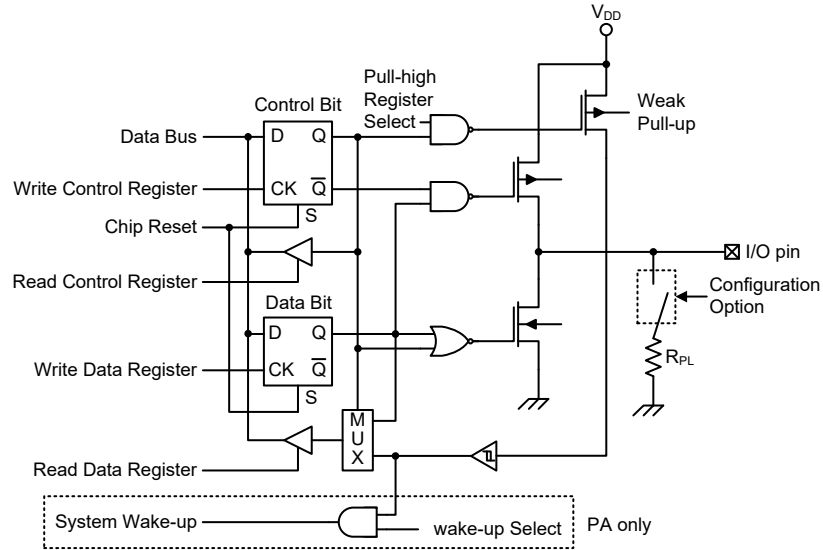
• **IFS Register**

| Bit  | 7 | 6 | 5 | 4 | 3 | 2 | 1    | 0    |
|------|---|---|---|---|---|---|------|------|
| Name | — | — | — | — | — | — | IFS1 | IFS0 |
| R/W  | — | — | — | — | — | — | R/W  | R/W  |
| POR  | — | — | — | — | — | — | 0    | 0    |

- Bit 7~2 Unimplemented, read as “0”
- Bit 1~0 **IFS1~IFS0**: RX/TX input source pin selection  
 00: PA2  
 01: PB0  
 10: PC2  
 11: PA2

### I/O Pin Structures

The accompanying diagram illustrates the internal structures of the I/O logic function. As the exact logical construction of the I/O pin will differ from this diagram, it is supplied as a guide only to assist with the functional understanding of the logic function I/O pins. The wide range of pin-shared structures does not permit all types to be shown.



Note: The R<sub>PL</sub> resistor which is controlled by configuration option is only available for the PB4 pin.

**Logic Function Input/Output Structure**

### Programming Considerations

Within the user program, one of the first things to consider is port initialisation. After a reset, all of the I/O data and port control registers will be set high. This means that all I/O pins will default to an input state, the level of which depends on the other connected circuitry and whether pull-high selections have been chosen. If the port control registers are then programmed to setup some pins as outputs, these output pins will have an initial high output value unless the associated port data registers are first programmed. Selecting which pins are inputs and which are outputs can be achieved byte-wide by loading the correct values into the appropriate port control register or by programming individual bits in the port control register using the “SET [m].i” and “CLR [m].i” instructions. Note that when using these bit control instructions, a read-modify-write operation takes place. The microcontroller must first read in the data on the entire port, modify it to the required new bit values and then rewrite this data back to the output ports.

Port A has the additional capability of providing wake-up functions. When the device is in the SLEEP or IDLE Mode, various methods are available to wake the device up. One of these is a high to low transition of any of the Port A pins. Single or multiple pins on Port A can be setup to have this function.

## Timer Modules – TM

One of the most fundamental functions in any microcontroller device is the ability to control and measure time. To implement time related functions the device includes several Timer Modules, abbreviated to the name TM. The TMs are multi-purpose timing units and serve to provide operations such as Timer/Counter, Compare Match Output and Single Pulse Output as well as being the functional unit for the generation of PWM signals. Each of the TMs has two individual interrupts. The addition of input and output pins for each TM ensures that users are provided with timing units with a wide and flexible range of features.

The common features of the different TM types are described here with more detailed information provided in the individual Compact and Standard Type TM sections.

### Introduction

The device contains two TMs and each individual TM can be categorised as a certain type, namely Compact Type TM and Standard Type TM. Although similar in nature, the different TM types vary in their feature complexity. The common features to all of the Compact and Standard Type TMs will be described in this section. The detailed operation regarding each of the TM types will be described in separate sections. The main features and differences between the two types of TMs are summarised in the accompanying table.

| Function                     | CTM            | STM            |
|------------------------------|----------------|----------------|
| Timer/Counter                | √              | √              |
| Compare Match Output         | √              | √              |
| PWM Output                   | √              | √              |
| Single Pulse Output          | —              | √              |
| PWM Alignment                | Edge           | Edge           |
| PWM Adjustment Period & Duty | Duty or Period | Duty or Period |

**TM Function Summary**

### TM Operation

The different types of TM offer a diverse range of functions, from simple timing operations to PWM signal generation. The key to understanding how the TM operates is to see it in terms of a free running counter whose value is then compared with the value of pre-programmed internal comparators. When the free running count-up counter has the same value as the pre-programmed comparator, known as a compare match situation, a TM interrupt signal will be generated which can clear the counter and perhaps also change the condition of the TM output pin. The internal TM counter is driven by a user selectable clock source, which can be an internal clock or an external pin.

### TM Clock Source

The clock source which drives the main counter in each TM can originate from various sources. The selection of the required clock source is implemented using the xTCK2~xTCK0 bits in the xTM control registers, where “x” stands for C or S type TM. The clock source can be a ratio of the system clock  $f_{SYS}$  or the internal high clock  $f_H$ , the  $f_{SUB}$  clock source or the external xTCK pin. The xTCK pin clock source is used to allow an external signal to drive the TM as an external clock source or for event counting.

**TM Interrupts**

The Compact and Standard type TMs each have two internal interrupts, one for each of the internal comparator A or comparator P, which generate a TM interrupt when a compare match condition occurs. When a TM interrupt is generated it can be used to clear the counter and also to change the state of the TM output pin.

**TM External Pins**

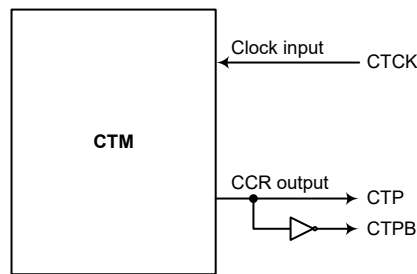
Each of the TMs, irrespective of what type, has one TM input pin, with the label xTCK. The xTM input pin, xTCK, is essentially a clock source for the xTM and is selected using the xTCK2~xTCK0 bits in the xTMC0 register. This external TM input pin allows an external clock source to drive the internal TM. The xTCK input pin can be chosen to have either a rising or falling active edge. The STCK pin is also used as the external trigger input pin in single pulse output mode for the STM.

The TMs each have two output pins, xTP and xTPB. The xTPB pin outputs the inverted signal of the xTP. When the TM is in the Compare Match Output Mode, these pins can be controlled by the xTM to switch to a high or low level or to toggle when a compare match situation occurs. The external xTP and xTPB output pins are also the pins where the xTM generates the PWM output waveform.

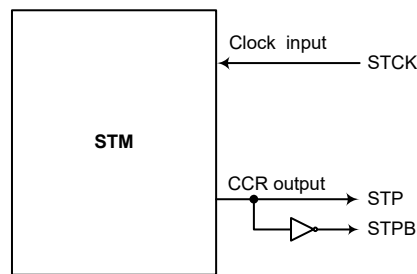
As the TM input and output pins are pin-shared with other functions, the TM input and output functions must first be setup using the relevant pin-shared function selection bits. The details of the pin-shared function selection are described in the pin-shared function section.

| CTM   |           | STM   |           |
|-------|-----------|-------|-----------|
| Input | Output    | Input | Output    |
| CTCK  | CTP, CTPB | STCK  | STP, STPB |

**TM External Pins**



**CTM Function Pin Block Diagram**



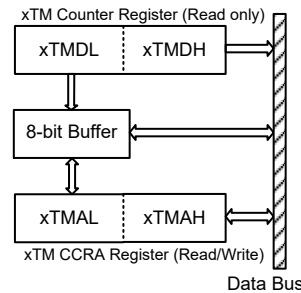
**STM Function Pin Block Diagram**



## Programming Considerations

The TM Counter Registers and the Compare CCRA registers, all have a low and high byte structure. The high bytes can be directly accessed, but as the low bytes can only be accessed via an internal 8-bit buffer, reading or writing to these register pairs must be carried out in a specific way. The important point to note is that data transfer to and from the 8-bit buffer and its related low byte only takes place when a write or read operation to its corresponding high byte is executed.

As the CCRA registers are implemented in the way shown in the following diagram and accessing these register pairs is carried out in a specific way as described above, it is recommended to use the “MOV” instruction to access the CCRA low byte register, named xTMAL, using the following access procedures. Accessing the CCRA low byte register without following these access procedures will result in unpredictable values.

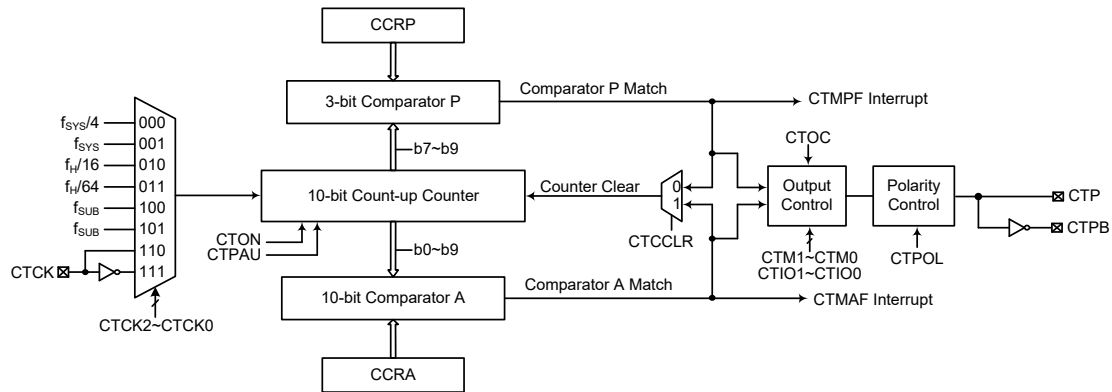


The following steps show the read and write procedures:

- Writing Data to CCRA
  - ♦ Step 1. Write data to Low Byte xTMAL
    - Note that here data is only written to the 8-bit buffer.
  - ♦ Step 2. Write data to High Byte xTMAH
    - Here data is written directly to the high byte registers and simultaneously data is latched from the 8-bit buffer to the Low Byte registers.
- Reading Data from the Counter Registers and CCRA
  - ♦ Step 1. Read data from the High Byte xTMDH, xTMAH
    - Here data is read directly from the High Byte registers and simultaneously data is latched from the Low Byte register into the 8-bit buffer.
  - ♦ Step 2. Read data from the Low Byte xTMDL, xTMAL
    - This step reads data from the 8-bit buffer.

## Compact Type TM – CTM

The Compact type TM contains three operating modes, which are Compare Match Output, Timer/Event Counter and PWM Output modes. The Compact type TM can also be controlled with an external input pin and can drive two external output pins.



- Note: 1. As the CTM external pins are pin-shared with other functions, the relevant pin-shared control bits should be properly configured before using these pins, except the CTCK pin which is selected only when the CTM counter clock is set to come from the CTCK pin by configuring the related CTM control register.  
2. The CTPB is the inverted signal of the CTP.

10-bit Compact Type TM Block Diagram

## Compact Type TM Operation

Its core is a 10-bit count-up counter which is driven by a user selectable internal or external clock source. There are also two internal comparators with the names, Comparator A and Comparator P. These comparators will compare the value in the counter with CCRP and CCRA registers. The CCRP is three bits wide whose value is compared with the highest three bits in the counter while the CCRA is the ten bits and therefore compares with all counter bits.

The only way of changing the value of the 10-bit counter using the application program, is to clear the counter by changing the CTON bit from low to high. The counter will also be cleared automatically by a counter overflow or a compare match with one of its associated comparators. When these conditions occur, a CTM interrupt signal will also usually be generated. The Compact Type TM can operate in a number of different operational modes, can be driven by different clock sources including an input pin and can also control two output pins. All operating setup conditions are selected using relevant internal registers.

## Compact Type TM Register Description

Overall operation of the Compact type TM is controlled using several registers. A read only register pair exists to store the internal counter 10-bit value, while a read/write register pair exists to store the internal 10-bit CCRA value. The remaining two registers are control registers which setup the different operating and control modes as well as the three CCRP bits.

| Register Name | Bit   |       |       |       |      |       |       |        |
|---------------|-------|-------|-------|-------|------|-------|-------|--------|
|               | 7     | 6     | 5     | 4     | 3    | 2     | 1     | 0      |
| CTMC0         | CTPAU | CTCK2 | CTCK1 | CTCK0 | CTON | CTRP2 | CTRP1 | CTRP0  |
| CTMC1         | CTM1  | CTM0  | CTIO1 | CTIO0 | CTOC | CTPOL | CTDPX | CTCCLR |
| CTMDL         | D7    | D6    | D5    | D4    | D3   | D2    | D1    | D0     |
| CTMDH         | —     | —     | —     | —     | —    | —     | D9    | D8     |

| Register Name | Bit |    |    |    |    |    |    |    |
|---------------|-----|----|----|----|----|----|----|----|
|               | 7   | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
| CTMAL         | D7  | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| CTMAH         | —   | —  | —  | —  | —  | —  | D9 | D8 |

**10-bit Compact Type TM Register List**

• **CTMC0 Register**

| Bit  | 7     | 6     | 5     | 4     | 3    | 2     | 1     | 0     |
|------|-------|-------|-------|-------|------|-------|-------|-------|
| Name | CTPAU | CTCK2 | CTCK1 | CTCK0 | CTON | CTRP2 | CTRP1 | CTRP0 |
| R/W  | R/W   | R/W   | R/W   | R/W   | R/W  | R/W   | R/W   | R/W   |
| POR  | 0     | 0     | 0     | 0     | 0    | 0     | 0     | 0     |

**Bit 7 CTPAU:** CTM counter pause control  
 0: Run  
 1: Pause  
 The counter can be paused by setting this bit high. Clearing the bit to zero restores normal counter operation. When in a Pause condition the CTM will remain powered up and continue to consume power. The counter will retain its residual value when this bit changes from low to high and resume counting from this value when the bit changes to a low value again.

**Bit 6~4 CTCK2~CTCK0:** Select CTM counter clock  
 000:  $f_{SYS}/4$   
 001:  $f_{SYS}$   
 010:  $f_H/16$   
 011:  $f_H/64$   
 100:  $f_{SUB}$   
 101:  $f_{SUB}$   
 110: CTCK rising edge clock  
 111: CTCK falling edge clock

These three bits are used to select the clock source for the CTM. The external pin clock source can be chosen to be active on the rising or falling edge. The clock source  $f_{SYS}$  is the system clock, while  $f_H$  and  $f_{SUB}$  are other internal clocks, the details of which can be found in the Operating Modes and System Clocks section.

**Bit 3 CTON:** CTM counter on/off control  
 0: Off  
 1: On  
 This bit controls the overall on/off function of the CTM. Setting the bit high enables the counter to run, clearing the bit to 0 disables the CTM. Clearing this bit to zero will stop the counter from counting and turn off the CTM which will reduce its power consumption. When the bit changes state from low to high the internal counter value will be reset to zero, however when the bit changes from high to low, the internal counter will retain its residual value until the bit returns high again.

If the CTM is in the Compare Match Output Mode or the PWM Output Mode then the CTM output pin will be reset to its initial condition, as specified by the CTOC bit, when the CTON bit changes from low to high.

**Bit 2~0 CTRP2~CTRP0:** CTM CCRP 3-bit register, compared with the CTM counter bit 9~bit 7  
 Comparator P Match Period=  
 000: 1024 CTM clocks  
 001: 128 CTM clocks  
 010: 256 CTM clocks  
 011: 384 CTM clocks  
 100: 512 CTM clocks  
 101: 640 CTM clocks  
 110: 768 CTM clocks  
 111: 896 CTM clocks

These three bits are used to setup the value on the internal CCRP 3-bit register, which are then compared with the internal counter's highest three bits. The result of this comparison can be selected to clear the internal counter if the CTCCLR bit is set to zero. Setting the CTCCLR bit to zero ensures that a compare match with the CCRP values will reset the internal counter. As the CCRP bits are only compared with the highest three counter bits, the compare values exist in 128 clock cycle multiples. Clearing all three bits to zero is in effect allowing the counter to overflow at its maximum value.

• **CTMC1 Register**

| Bit  | 7    | 6    | 5     | 4     | 3    | 2     | 1     | 0      |
|------|------|------|-------|-------|------|-------|-------|--------|
| Name | CTM1 | CTM0 | CTIO1 | CTIO0 | CTOC | CTPOL | CTDPX | CTCCLR |
| R/W  | R/W  | R/W  | R/W   | R/W   | R/W  | R/W   | R/W   | R/W    |
| POR  | 0    | 0    | 0     | 0     | 0    | 0     | 0     | 0      |

Bit 7~6     **CTM1~CTM0**: CTM operating mode selection  
             00: Compare Match Output Mode  
             01: Undefined  
             10: PWM Output Mode  
             11: Timer/Counter Mode

These bits setup the required operating mode for the CTM. To ensure reliable operation the CTM should be switched off before any changes are made to the CTM1 and CTM0 bits. In the Timer/Counter Mode, the CTM output pin state is undefined.

Bit 5~4     **CTIO1~CTIO0**: CTM external pin function selection

Compare Match Output Mode  
             00: No change  
             01: Output low  
             10: Output high  
             11: Toggle output  
 PWM Output Mode  
             00: PWM Output inactive state  
             01: PWM Output active state  
             10: PWM output  
             11: Undefined

Timer/Counter Mode  
             Unused

These two bits are used to determine how the CTM output pin changes state when a certain condition is reached. The function that these bits select depends upon in which mode the CTM is running.

In the Compare Match Output Mode, the CTIO1 and CTIO0 bits determine how the CTM output pin changes state when a compare match occurs from the Comparator A. The CTM output pin can be setup to switch high, switch low or to toggle its present state when a compare match occurs from the Comparator A. When the bits are both zero, then no change will take place on the output. The initial value of the CTM output pin should be setup using the CTOC bit in the CTMC1 register. Note that the output level requested by the CTIO1 and CTIO0 bits must be different from the initial value setup using the CTOC bit otherwise no change will occur on the CTM output pin when a compare match occurs. After the CTM output pin changes state it can be reset to its initial level by changing the level of the CTON bit from low to high.

In the PWM Output Mode, the CTIO1 and CTIO0 bits determine how the CTM output pin changes state when a certain compare match condition occurs. The PWM output function is modified by changing these two bits. It is necessary to change the values of the CTIO1 and CTIO0 bits only after the CTM has been switched off. Unpredictable PWM outputs will occur if the CTIO1 and CTIO0 bits are changed when The CTM is running.

- Bit 3**      **CTOC:** CTM CTP output control bit  
 Compare Match Output Mode  
     0: Initial low  
     1: Initial high  
 PWM Output Mode  
     0: Active low  
     1: Active high
- This is the output control bit for the CTP output pin. Its operation depends upon whether CTM is being used in the Compare Match Output Mode or in the PWM Output Mode. It has no effect if the CTM is in the Timer/Counter Mode. In the Compare Match Output Mode, it determines the logic level of the CTM output pin before a compare match occurs. In the PWM Output Mode, it determines if the PWM signal is active high or active low.
- Bit 2**      **CTPOL:** CTM CTP output polarity control  
     0: Non-invert  
     1: Invert
- This bit controls the polarity of the CTP output pin. When the bit is set high the CTM output pin will be inverted and not inverted when the bit is zero. It has no effect if the CTM is in the Timer/Counter Mode.
- Bit 1**      **CTDPX:** CTM PWM period/duty control  
     0: CCRP – period; CCRA – duty  
     1: CCRP – duty; CCRA – period
- This bit determines which of the CCRA and CCRP registers are used for period and duty control of the PWM waveform.
- Bit 0**      **CTCCLR:** CTM counter clear condition selection  
     0: CTM Comparatror P match  
     1: CTM Comparatror A match
- This bit is used to select the method which clears the counter. Remember that the Compact TM contains two comparators, Comparator A and Comparator P, either of which can be selected to clear the internal counter. With the CTCCLR bit set high, the counter will be cleared when a compare match occurs from the Comparator A. When the bit is low, the counter will be cleared when a compare match occurs from the Comparator P or with a counter overflow. A counter overflow clearing method can only be implemented if the CCRP bits are all cleared to zero. The CTCCLR bit is not used in the PWM Output Mode.

• **CTMDL Register**

| Bit  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|------|----|----|----|----|----|----|----|----|
| Name | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| R/W  | R  | R  | R  | R  | R  | R  | R  | R  |
| POR  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |

**Bit 7~0**      **D7~D0:** CTM Counter Low Byte Register bit 7 ~ bit 0  
 CTM 10-bit Counter bit 7 ~ bit 0

• **CTMDH Register**

| Bit  | 7 | 6 | 5 | 4 | 3 | 2 | 1  | 0  |
|------|---|---|---|---|---|---|----|----|
| Name | — | — | — | — | — | — | D9 | D8 |
| R/W  | — | — | — | — | — | — | R  | R  |
| POR  | — | — | — | — | — | — | 0  | 0  |

**Bit 7~2**      Unimplemented, read as “0”

**Bit 1~0**      **D9~D8:** CTM Counter High Byte Register bit 1 ~ bit 0  
 CTM 10-bit Counter bit 9 ~ bit 8

• **CTMAL Register**

| Bit  | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   |
|------|-----|-----|-----|-----|-----|-----|-----|-----|
| Name | D7  | D6  | D5  | D4  | D3  | D2  | D1  | D0  |
| R/W  | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| POR  | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |

Bit 7~0      **D7~D0**: CTM CCRA Low Byte Register bit 7 ~ bit 0  
CTM 10-bit CCRA bit 7 ~ bit 0

• **CTMAH Register**

| Bit  | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0   |
|------|---|---|---|---|---|---|-----|-----|
| Name | — | — | — | — | — | — | D9  | D8  |
| R/W  | — | — | — | — | — | — | R/W | R/W |
| POR  | — | — | — | — | — | — | 0   | 0   |

Bit 7~2      Unimplemented, read as “0”  
Bit 1~0      **D9~D8**: CTM CCRA High Byte Register bit 1 ~ bit 0  
CTM 10-bit CCRA bit 9 ~ bit 8

**Compact Type TM Operating Modes**

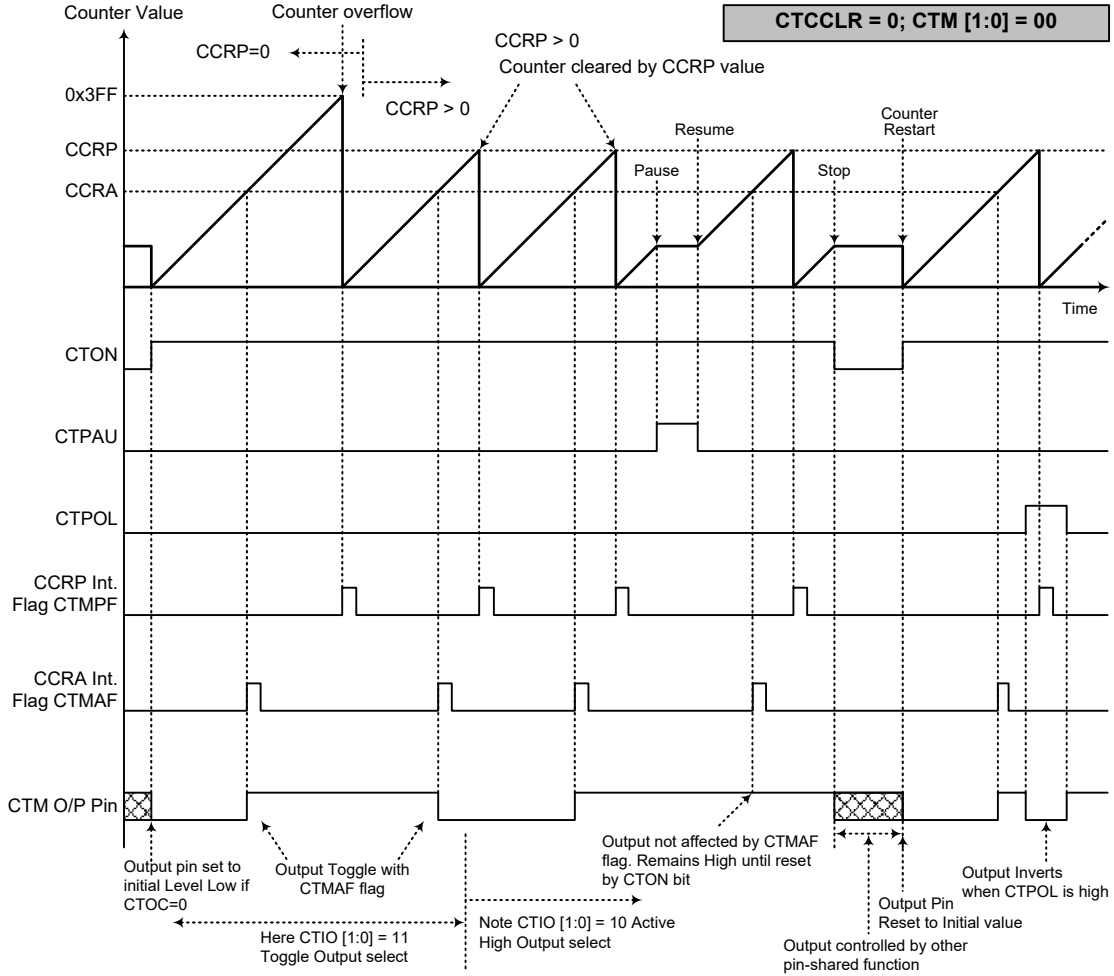
The Compact Type TM can operate in one of three operating modes, Compare Match Output Mode, PWM Output Mode or Timer/Counter Mode. The operating mode is selected using the CTM1 and CTM0 bits in the CTMC1 register.

**Compare Match Output Mode**

To select this mode, bits CTM1 and CTM0 in the CTMC1 register, should be set to “00” respectively. In this mode once the counter is enabled and running it can be cleared by three methods. These are a counter overflow, a compare match from Comparator A and a compare match from Comparator P. When the CTCCLR bit is low, there are two ways in which the counter can be cleared. One is when a compare match from Comparator P, the other is when the CCRP bits are all zero which allows the counter to overflow. Here both CTMAF and CTMPF interrupt request flags for Comparator A and Comparator P respectively, will both be generated.

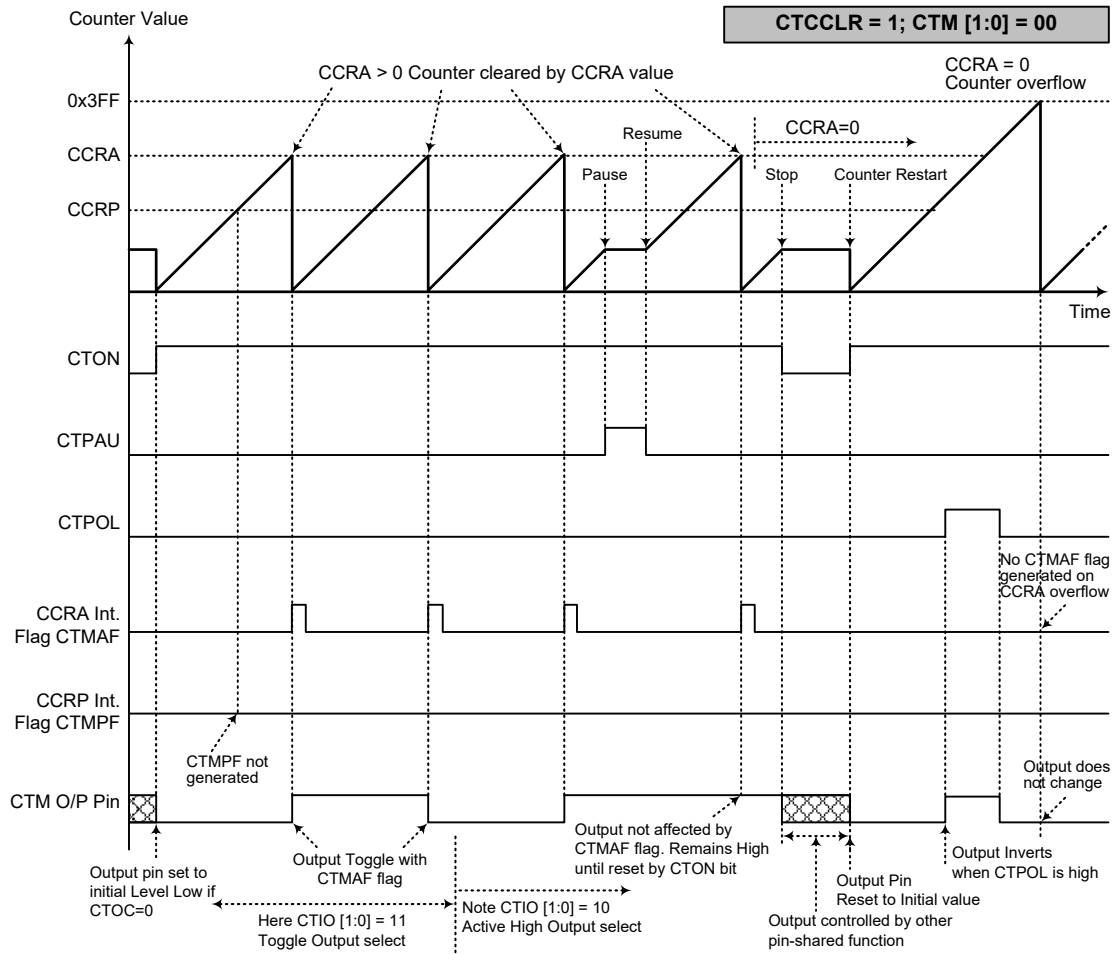
If the CTCCLR bit in the CTMC1 register is high, then the counter will be cleared when a compare match occurs from Comparator A. However, here only the CTMAF interrupt request flag will be generated even if the value of the CCRP bits is less than that of the CCRA registers. Therefore, when CTCCLR is high no CTMPF interrupt request flag will be generated. If the CCRA bits are all zero, the counter will overflow when it reaches its maximum 10-bit, 3FF Hex, value, however here the CTMAF interrupt request flag will not be generated.

As the name of the mode suggests, after a comparison is made, the CTM output pin will change state. The CTM output pin condition however only changes state when a CTMAF interrupt request flag is generated after a compare match occurs from Comparator A. The CTMPF interrupt request flag, generated from a compare match occurs from Comparator P, will have no effect on the CTM output pin. The way in which the CTM output pin changes state are determined by the condition of the CTIO1 and CTIO0 bits in the CTMC1 register. The CTM output pin can be selected using the CTIO1 and CTIO0 bits to go high, to go low or to toggle from its present condition when a compare match occurs from Comparator A. The initial condition of the CTM output pin, which is setup after the CTON bit changes from low to high, is setup using the CTOC bit. Note that if the CTIO1 and CTIO0 bits are zero then no pin change will take place.



**Compare Match Output Mode – CTCCLR=0**

- Note: 1. With CTCCLR=0, a Comparator P match will clear the counter  
 2. The CTM output pin controlled only by the CTMAF flag  
 3. The output pin reset to initial state by a CTON bit rising edge



**Compare Match Output Mode – CTCCLR=1**

- Note: 1. With CTCCLR=1, a Comparator A match will clear the counter
2. The CTM output pin controlled only by the CTMAF flag
3. The output pin reset to initial state by a CTON rising edge
4. The CTMPF flags is not generated when CTCCLR=1



**Timer/Counter Mode**

To select this mode, bits CTM1 and CTM0 in the CTMC1 register should be set to “11” respectively. The Timer/Counter Mode operates in an identical way to the Compare Match Output Mode generating the same interrupt flags. The exception is that in the Timer/Counter Mode the CTM output pin is not used. Therefore, the above description and Timing Diagrams for the Compare Match Output Mode can be used to understand its function. As the CTM output pin is not used in this mode, the pin can be used as a normal I/O pin or other pin-shared functions.

**PWM Output Mode**

To select this mode, bits CTM1 and CTM0 in the CTMC1 register should be set to “10” respectively. The PWM function within the CTM is useful for applications which require functions such as motor control, heating control, illumination control etc. By providing a signal of fixed frequency but of varying duty cycle on the CTM output pin, a square wave AC waveform can be generated with varying equivalent DC RMS values.

As both the period and duty cycle of the PWM waveform can be controlled, the choice of generated waveform is extremely flexible. In the PWM Output Mode, the CTCCLR bit has no effect on the PWM operation. Both of the CCRA and CCRP registers are used to generate the PWM waveform, one register is used to clear the internal counter and thus control the PWM waveform frequency, while the other one is used to control the duty cycle. Which register is used to control either frequency or duty cycle is determined using the CTD PX bit in the CTMC1 register. The PWM waveform frequency and duty cycle can therefore be controlled by the values in the CCRA and CCRP registers.

An interrupt flag, one for each of the CCRA and CCRP, will be generated when a compare match occurs from either Comparator A or Comparator P. The CTOC bit in the CTMC1 register is used to select the required polarity of the PWM waveform while the two CTIO1 and CTIO0 bits are used to enable the PWM output or to force the CTM output pin to a fixed high or low level. The CTPOL bit is used to reverse the polarity of the PWM output waveform.

• **10-bit CTM, PWM Output Mode, Edge-aligned Mode, CTD PX=0**

| CCRP   | 1~7      | 0    |
|--------|----------|------|
| Period | CCRP×128 | 1024 |
| Duty   | CCRA     |      |

If  $f_{SYS}=8\text{MHz}$ , CTM clock source is  $f_{SYS}/4$ , CCRP=2, CCRA=128,

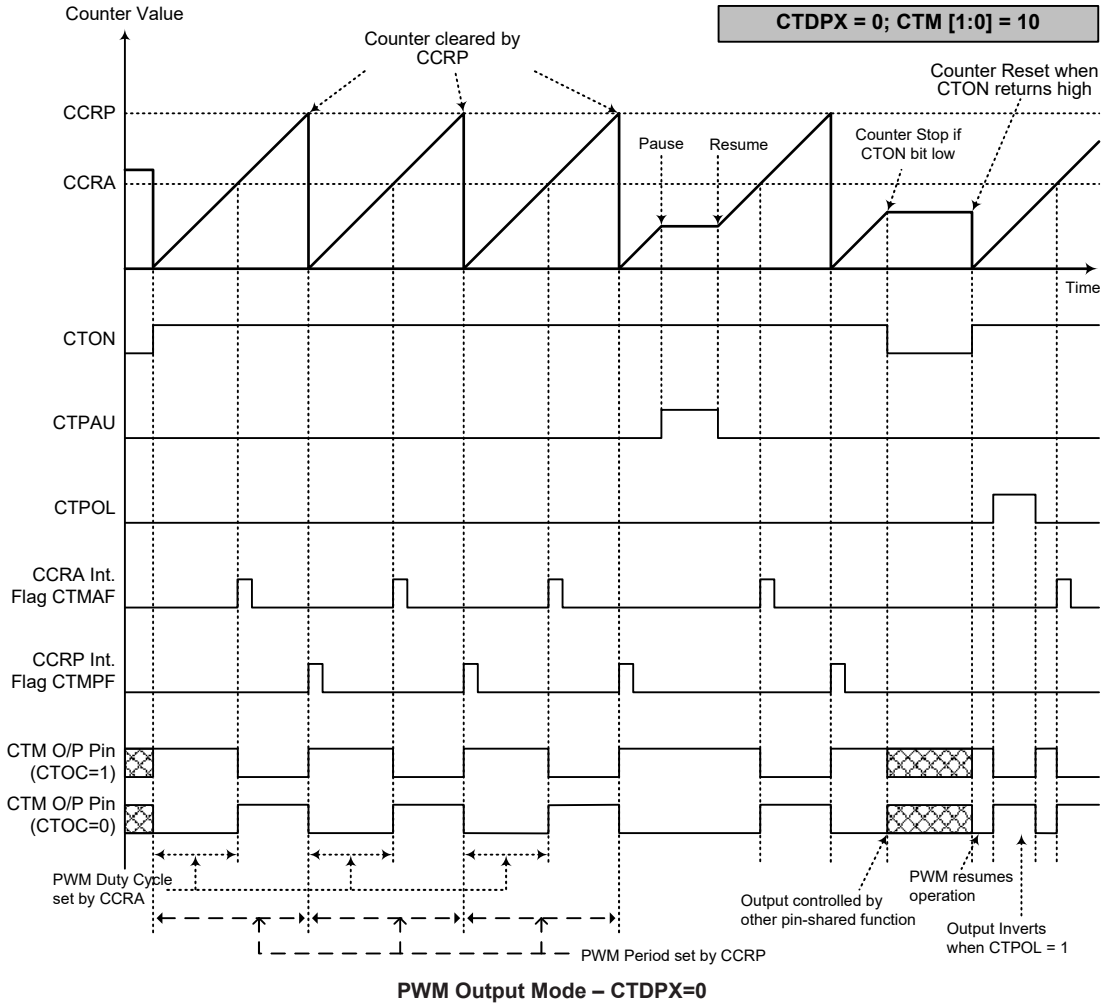
The CTM PWM output frequency= $(f_{SYS}/4)/(2\times 128)=f_{SYS}/1024=8\text{kHz}$ , duty= $128/(2\times 128)=50\%$ .

If the Duty value defined by the CCRA register is equal to or greater than the Period value, then the PWM output duty is 100%.

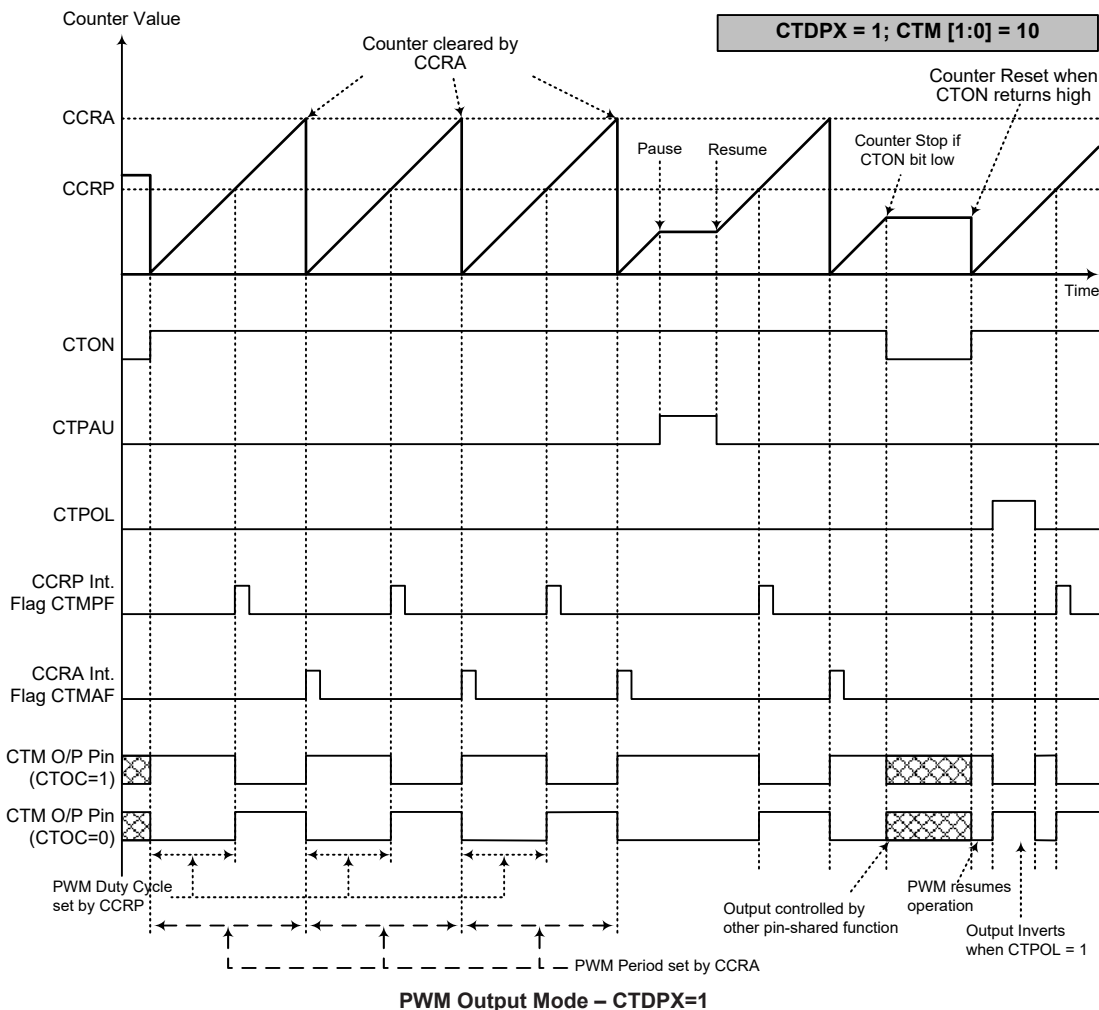
• **10-bit CTM, PWM Output Mode, Edge-aligned Mode, CTD PX=1**

| CCRP   | 1~7      | 0    |
|--------|----------|------|
| Period | CCRA     |      |
| Duty   | CCRP×128 | 1024 |

The PWM output period is determined by the CCRA register value together with the CTM clock while the PWM duty cycle is defined by the CCRP register value.



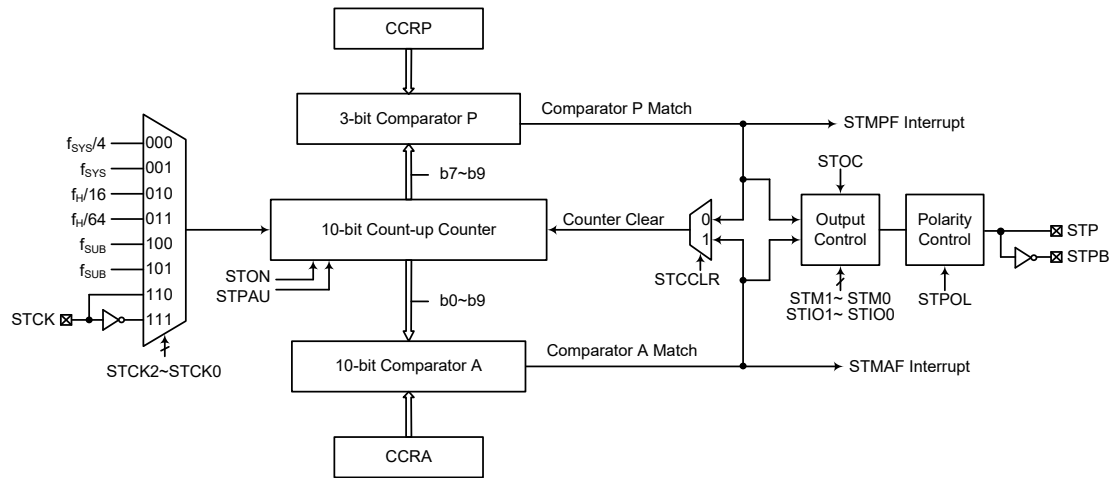
- Note: 1. Here CTD PX=0 – Counter cleared by CCRP  
 2. A counter clear sets PWM Period  
 3. The internal PWM function continues running even when CTIO[1:0]=00 or 01  
 4. The CTCCLR bit has no influence on PWM operation



- Note: 1. Here CTDPX=1 – Counter cleared by CCRA  
 2. A counter clear sets PWM Period  
 3. The internal PWM function continues even when CTIO[1:0]=00 or 01  
 4. The CTCCLR bit has no influence on PWM operation

## Standard Type TM – STM

The Standard Type TM contains four operating modes, which are Compare Match Output, Timer/Event Counter, Single Pulse Output and PWM Output modes. The Standard TM can also be controlled with one external input pin and can drive two external output pins.



- Note: 1. As the STM external pins are pin-shared with other functions, the relevant pin-shared control bits should be properly configured before using these pins.  
2. The STPB is the inverted signal of the STP.

**10-bit Standard Type TM Block Diagram**

### Standard TM Operation

The size of Standard TM is 10-bit wide and its core is a 10-bit count-up counter which is driven by a user selectable internal or external clock source. There are also two internal comparators with the names, Comparator A and Comparator P. These comparators will compare the value in the counter with the CCRP and CCRA registers. The CCRP comparator is 3-bit wide whose value is compared the with highest 3 bits in the counter while the CCRA is the ten bits and therefore compares all counter bits.

The only way of changing the value of the 10-bit counter using the application program, is to clear the counter by changing the STON bit from low to high. The counter will also be cleared automatically by a counter overflow or a compare match with one of its associated comparators. When these conditions occur, a STM interrupt signal will also usually be generated. The Standard Type TM can operate in a number of different operational modes, can be driven by different clock sources including an input pin and can also control the output pins. All operating setup conditions are selected using relevant internal registers.

### Standard Type TM Register Description

Overall operation of the Standard TM is controlled using a series of registers. A read only register pair exists to store the internal counter 10-bit value, while a read/write register pair exists to store the internal 10-bit CCRA value. The remaining two registers are control registers which set the different operating and control modes as well as the three CCRP bits.

| Register Name | Bit   |       |       |       |      |       |       |        |
|---------------|-------|-------|-------|-------|------|-------|-------|--------|
|               | 7     | 6     | 5     | 4     | 3    | 2     | 1     | 0      |
| STMC0         | STPAU | STCK2 | STCK1 | STCK0 | STON | STPR2 | STPR1 | STPR0  |
| STMC1         | STM1  | STM0  | STIO1 | STIO0 | STOC | STPOL | STDPX | STCCLR |
| STMDL         | D7    | D6    | D5    | D4    | D3   | D2    | D1    | D0     |
| STMDH         | —     | —     | —     | —     | —    | —     | D9    | D8     |
| STMAL         | D7    | D6    | D5    | D4    | D3   | D2    | D1    | D0     |
| STMAH         | —     | —     | —     | —     | —    | —     | D9    | D8     |

**10-bit Standard TM Register List**

• **STMC0 Register**

| Bit  | 7     | 6     | 5     | 4     | 3    | 2     | 1     | 0     |
|------|-------|-------|-------|-------|------|-------|-------|-------|
| Name | STPAU | STCK2 | STCK1 | STCK0 | STON | STPR2 | STPR1 | STPR0 |
| R/W  | R/W   | R/W   | R/W   | R/W   | R/W  | R/W   | R/W   | R/W   |
| POR  | 0     | 0     | 0     | 0     | 0    | 0     | 0     | 0     |

Bit 7 **STPAU**: STM counter pause control

0: Run  
 1: Pause

The counter can be paused by setting this bit high. Clearing the bit to zero restores normal counter operation. When in a Pause condition the STM will remain powered up and continue to consume power. The counter will retain its residual value when this bit changes from low to high and resume counting from this value when the bit changes to a low value again.

Bit 6~4 **STCK2~STCK0**: STM counter clock selection

000:  $f_{SYS}/4$   
 001:  $f_{SYS}$   
 010:  $f_H/16$   
 011:  $f_H/64$   
 100:  $f_{SUB}$   
 101:  $f_{SUB}$   
 110: STCK rising edge clock  
 111: STCK falling edge clock

These three bits are used to select the clock source for the STM. The external pin clock source can be chosen to be active on the rising or falling edge. The clock source  $f_{SYS}$  is the system clock, while  $f_H$  and  $f_{SUB}$  are other internal clocks, the details of which can be found in the Operating Modes and System Clocks section.

Bit 3 **STON**: STM counter on/off control

0: Off  
 1: On

This bit controls the overall on/off function of the STM. Setting the bit high enables the counter to run while clearing the bit disables the STM. Clearing this bit to zero will stop the counter from counting and turn off the STM which will reduce its power consumption. When the bit changes state from low to high the internal counter value will be reset to zero, however when the bit changes from high to low, the internal counter will retain its residual value until the bit returns high again. If the STM is in the Compare Match Output Mode, PWM Output Mode or Single Pulse Output Mode then the STM output pin will be reset to its initial condition, as specified by the STOC bit, when the STON bit changes from low to high.

Bit 2~0 **STRP2~STRP0**: STM CCRP 3-bit register, compared with the STM Counter bit 9 ~ bit 7

Comparator P Match Period:

- 000: 1024 STM clocks
- 001: 128 STM clocks
- 010: 256 STM clocks
- 011: 384 STM clocks
- 100: 512 STM clocks
- 101: 640 STM clocks
- 110: 768 STM clocks
- 111: 896 STM clocks

These three bits are used to setup the value on the internal CCRP 3-bit register, which are then compared with the internal counter's highest three bits. The result of this comparison can be selected to clear the internal counter if the STCCLR bit is set to zero. Setting the STCCLR bit to zero ensures that a compare match with the CCRP values will reset the internal counter. As the CCRP bits are only compared with the highest three counter bits, the compare values exist in 128 clock cycle multiples. Clearing all three bits to zero is in effect allowing the counter to overflow at its maximum value.

• **STMC1 Register**

| Bit  | 7    | 6    | 5     | 4     | 3    | 2     | 1     | 0      |
|------|------|------|-------|-------|------|-------|-------|--------|
| Name | STM1 | STM0 | STIO1 | STIO0 | STOC | STPOL | STDPX | STCCLR |
| R/W  | R/W  | R/W  | R/W   | R/W   | R/W  | R/W   | R/W   | R/W    |
| POR  | 0    | 0    | 0     | 0     | 0    | 0     | 0     | 0      |

Bit 7~6 **STM1~STM0**: STM operating mode selection

- 00: Compare Match Output Mode
- 01: Undefined
- 10: PWM Output Mode or Single Pulse Output Mode
- 11: Timer/Counter Mode

These bits are used to set the required operating mode for the STM. To ensure reliable operation the STM should be switched off before any changes are made to the STM1 and STM0 bits. In the Timer/Counter Mode, the STM output pin state is undefined.

Bit 5~4 **STIO1~STIO0**: STM external pin function selection

Compare Match Output Mode

- 00: No change
- 01: Output low
- 10: Output high
- 11: Toggle output

PWM Output Mode/Single Pulse Output Mode

- 00: PWM output inactive state
- 01: PWM output active state
- 10: PWM output
- 11: Single Pulse Output

Timer/Counter Mode

Unused

These two bits are used to determine how the STM external pin changes state when a certain condition is reached. The function that these bits select depends upon in which mode the STM is running.

In the Compare Match Output Mode, the STIO1 and STIO0 bits determine how the STM output pin changes state when a compare match occurs from the Comparator A. The STM output pin can be set to switch high, switch low or to toggle its present state when a compare match occurs from the Comparator A. When the bits are both zero, then no change will take place on the output. The initial value of the STM output pin should be set using the STOC bit in the STMC1 register. Note that the output level requested by the STIO1 and STIO0 bits must be different from the initial value setup using the STOC bit otherwise no change will occur on the STM output pin when a compare match occurs. After the STM output pin changes state, it can be reset to its initial level by changing the level of the STON bit from low to high.

In the PWM Output Mode, the STIO1 and STIO0 bits determine how the STM output pin changes state when a certain compare match condition occurs. The PWM output function is modified by changing these two bits. It is necessary to change the values of the STIO1 and STIO0 bits only after the STM has been switched off. Unpredictable PWM outputs will occur if the STIO1 and STIO0 bits are changed when the STM is running.

- Bit 3     **STOC:** STM STP output control  
Compare Match Output Mode  
    0: Initial low  
    1: Initial high  
PWM Output Mode/Single Pulse Output Mode  
    0: Active low  
    1: Active high

This is the output control bit for the STM output pin. Its operation depends upon whether STM is being used in the Compare Match Output Mode or in the PWM Output Mode/Single Pulse Output Mode. It has no effect if the STM is in the Timer/Counter Mode. In the Compare Match Output Mode, it determines the logic level of the STM output pin before a compare match occurs. In the PWM Output Mode it determines if the PWM signal is active high or active low. In the Single Pulse Output Mode, it determines the logic level of the STM output pin when the STON bit changes from low to high.

- Bit 2     **STPOL:** STM STP output polarity control  
    0: Non-invert  
    1: Invert

This bit controls the polarity of the STP output pin. When the bit is set high the STM output pin will be inverted and not inverted when the bit is zero. It has no effect if the STM is in the Timer/Counter Mode.

- Bit 1     **STDPX:** STM PWM duty/period control  
    0: CCRP – period; CCRA – duty  
    1: CCRP – duty; CCRA – period

This bit determines which of the CCRA and CCRP registers are used for period and duty control of the PWM waveform.

- Bit 0     **STCCLR:** STM counter clear condition selection  
    0: Comparator P match  
    1: Comparator A match

This bit is used to select the method which clears the counter. Remember that the Standard TM contains two comparators, Comparator A and Comparator P, either of which can be selected to clear the internal counter. With the STCCLR bit set high, the counter will be cleared when a compare match occurs from the Comparator A. When the bit is low, the counter will be cleared when a compare match occurs from the Comparator P or with a counter overflow. A counter overflow clearing method can only be implemented if the CCRP bits are all cleared to zero. The STCCLR bit is not used in the PWM Output Mode or Single Pulse Output Mode.

• **STMDL Register**

| Bit  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|------|----|----|----|----|----|----|----|----|
| Name | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| R/W  | R  | R  | R  | R  | R  | R  | R  | R  |
| POR  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |

Bit 7~0      **D7~D0**: STM Counter Low Byte Register bit 7 ~ bit 0  
STM 10-bit Counter bit 7 ~ bit 0

• **STMDH Register**

| Bit  | 7 | 6 | 5 | 4 | 3 | 2 | 1  | 0  |
|------|---|---|---|---|---|---|----|----|
| Name | — | — | — | — | — | — | D9 | D8 |
| R/W  | — | — | — | — | — | — | R  | R  |
| POR  | — | — | — | — | — | — | 0  | 0  |

Bit 7~2      Implemented, read as “0”  
Bit 1~0      **D9~D8**: STM Counter High Byte Register bit 1 ~ bit 0  
STM 10-bit Counter bit 9 ~ bit 8

• **STMAL Register**

| Bit  | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   |
|------|-----|-----|-----|-----|-----|-----|-----|-----|
| Name | D7  | D6  | D5  | D4  | D3  | D2  | D1  | D0  |
| R/W  | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| POR  | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |

Bit 7~0      **D7~D0**: STM CCRA Low Byte Register bit 7 ~ bit 0  
STM 10-bit CCRA bit 7 ~ bit 0

• **STMAH Register**

| Bit  | 7 | 6 | 5 | 4 | 3 | 2 | 1   | 0   |
|------|---|---|---|---|---|---|-----|-----|
| Name | — | — | — | — | — | — | D9  | D8  |
| R/W  | — | — | — | — | — | — | R/W | R/W |
| POR  | — | — | — | — | — | — | 0   | 0   |

Bit 7~2      Implemented, read as “0”  
Bit 1~0      **D9~D8**: STM CCRA High Byte Register bit 1 ~ bit 0  
STM 10-bit CCRA bit 9 ~ bit 8



## Standard Type TM Operation Modes

The Standard Type TM can operate in one of four operating modes, Compare Match Output Mode, PWM Output Mode, Single Pulse Output Mode or Timer/Counter Mode. The operating mode is selected using the STM1 and STM0 bits in the STMC1 register.

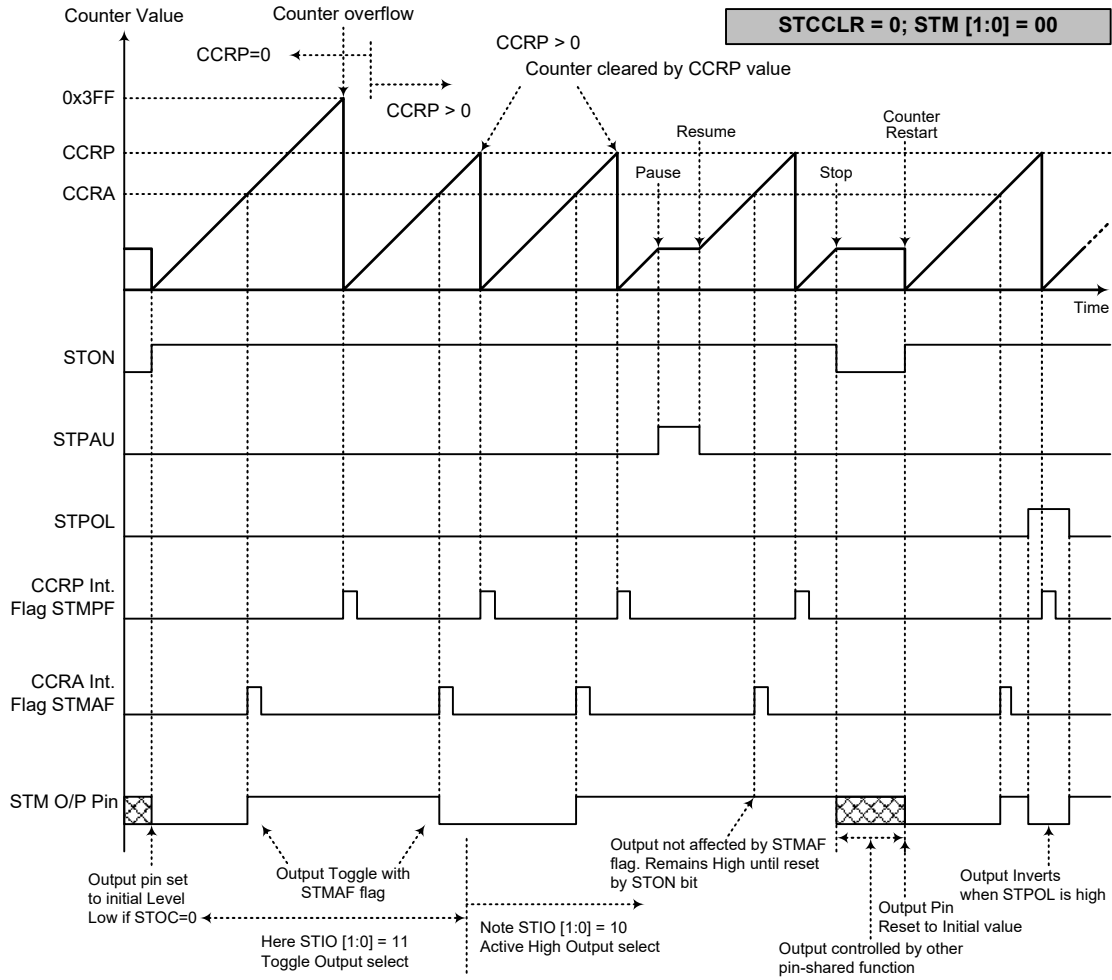
### Compare Match Output Mode

To select this mode, bits STM1 and STM0 in the STMC1 register, should be set to “00” respectively. In this mode once the counter is enabled and running it can be cleared by three methods. These are a counter overflow, a compare match from Comparator A and a compare match from Comparator P. When the STCCLR bit is low, there are two ways in which the counter can be cleared. One is when a compare match from Comparator P, the other is when the CCRP bits are all zero which allows the counter to overflow. Here both STMAF and STMPF interrupt request flags for Comparator A and Comparator P respectively, will both be generated.

If the STCCLR bit in the STMC1 register is high, then the counter will be cleared when a compare match occurs from Comparator A. However, here only the STMAF interrupt request flag will be generated even if the value of the CCRP bits is less than that of the CCRA registers. Therefore, when STCCLR is high no STMPF interrupt request flag will be generated. In the Compare Match Output Mode, the CCRA can not be set to “0”.

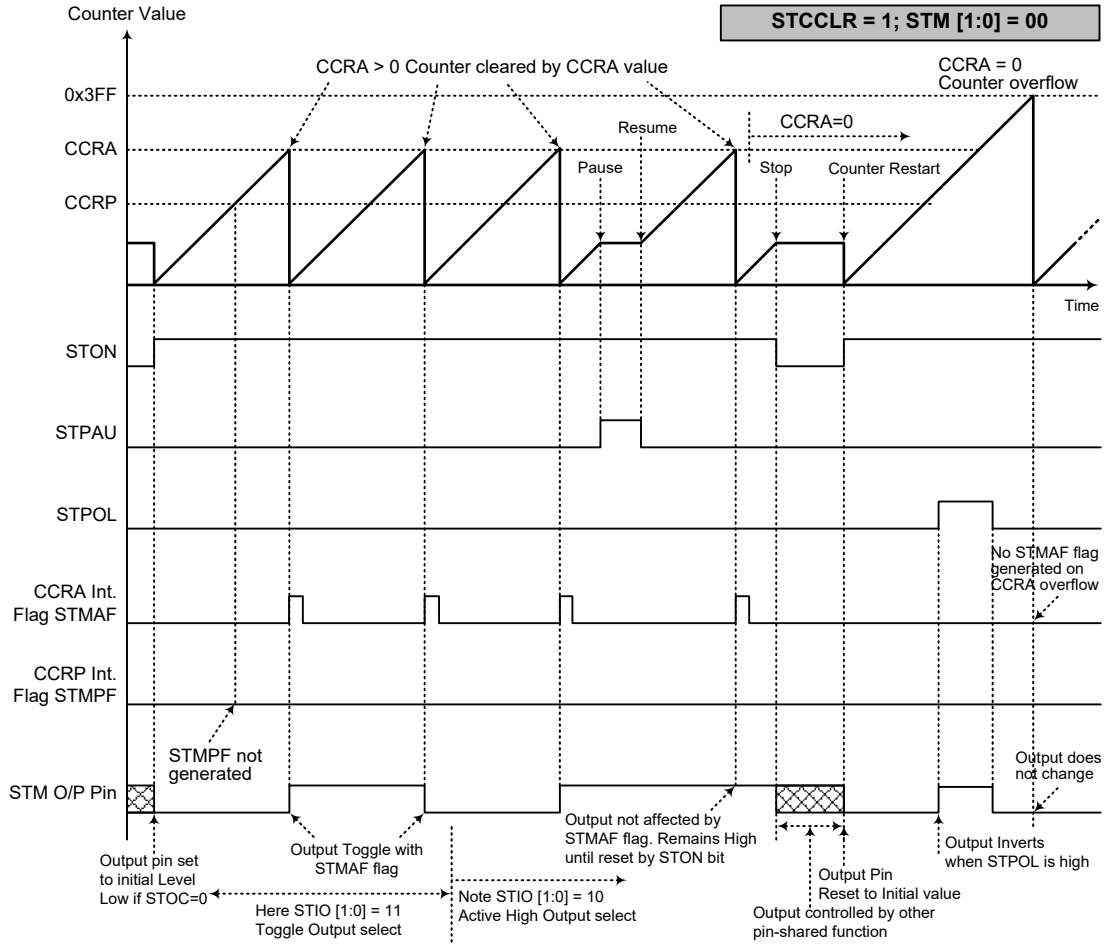
If the CCRA bits are all zero, the counter will overflow when it reaches its maximum 10-bit, 3FF Hex, value, however here the STMAF interrupt request flag will not be generated.

As the name of the mode suggests, after a comparison is made, the STM output pin, will change state. The STM output pin condition however only changes state when a STMAF interrupt request flag is generated after a compare match occurs from Comparator A. The STMPF interrupt request flag, generated from a compare match occurs from Comparator P, will have no effect on the STM output pin. The way in which the STM output pin changes state are determined by the condition of the STIO1 and STIO0 bits in the STMC1 register. The STM output pin can be selected using the STIO1 and STIO0 bits to go high, to go low or to toggle from its present condition when a compare match occurs from Comparator A. The initial condition of the STM output pin, which is configured after the STON bit changes from low to high, is configured using the STOC bit. Note that if the STIO1 and STIO0 bits are zero then no pin change will take place.



**Compare Match Output Mode – STCCLR=0**

- Note: 1. With STCCLR=0 a Comparator P match will clear the counter
2. The STM output pin is controlled only by the STMAF flag
3. The output pin is reset to its initial state by a STON bit rising edge



**Compare Match Output Mode – STCCLR=1**

- Note:
1. With  $STCCLR=1$  a Comparator A match will clear the counter
  2. The STM output pin is controlled only by the  $STMAF$  flag
  3. The output pin is reset to its initial state by a  $STON$  bit rising edge
  4. The  $STMPF$  flag is not generated when  $STCCLR=1$

### Timer/Counter Mode

To select this mode, bits STM1 and STM0 in the STMC1 register should be set to “11” respectively. The Timer/Counter Mode operates in an identical way to the Compare Match Output Mode generating the same interrupt flags. The exception is that in the Timer/Counter Mode the STM output pin is not used. Therefore, the above description and Timing Diagrams for the Compare Match Output Mode can be used to understand its function. As the STM output pin is not used in this mode, the pin can be used as a normal I/O pin or other pin-shared functions.

### PWM Output Mode

To select this mode, bits STM1 and STM0 in the STMC1 register should be set to “10” respectively. The PWM function within the STM is useful for applications which require functions such as motor control, heating control, illumination control etc. By providing a signal of fixed frequency but of varying duty cycle on the STM output pin, a square wave AC waveform can be generated with varying equivalent DC RMS values.

As both the period and duty cycle of the PWM waveform can be controlled, the choice of generated waveform is extremely flexible. In the PWM output mode, the STCCLR bit has no effect as the PWM period. Both of the CCRA and CCRP registers are used to generate the PWM waveform, one register is used to clear the internal counter and thus control the PWM waveform frequency, while the other one is used to control the duty cycle. Which register is used to control either frequency or duty cycle is determined using the STDPX bit in the STMC1 register. The PWM waveform frequency and duty cycle can therefore be controlled by the values in the CCRA and CCRP registers.

An interrupt flag, one for each of the CCRA and CCRP, will be generated when a compare match occurs from either Comparator A or Comparator P. The STOC bit in the STMC1 register is used to select the required polarity of the PWM waveform while the two STIO1 and STIO0 bits are used to enable the PWM output or to force the STM output pin to a fixed high or low level. The STPOL bit is used to reverse the polarity of the PWM output waveform.

• **10-bit STM, PWM Output Mode, Edge-aligned Mode, STDPX=0**

| CCRP   | 001b | 010b | 011b | 100b | 101b | 110b | 111b | 000b |
|--------|------|------|------|------|------|------|------|------|
| Period | 128  | 256  | 384  | 512  | 640  | 768  | 896  | 1024 |
| Duty   | CCRA |      |      |      |      |      |      |      |

If  $f_{SYS}=8\text{MHz}$ , STM clock source is  $f_{SYS}/4$ , CCRP=100b and CCRA=128,

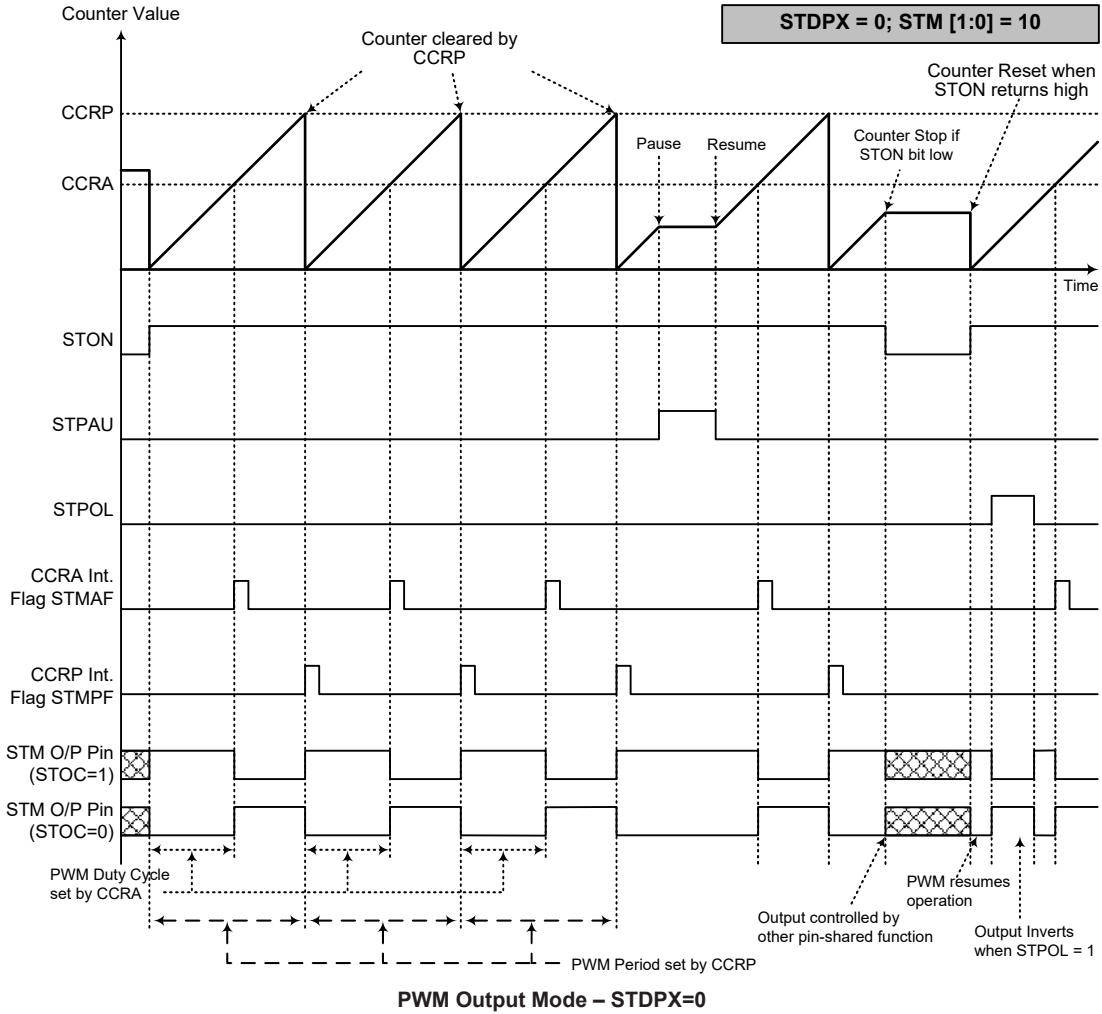
The STM PWM output frequency= $(f_{SYS}/4)/512=f_{SYS}/2048=4\text{kHz}$ , duty=128/512=25%.

If the Duty value defined by the CCRA register is equal to or greater than the Period value, then the PWM output duty is 100%.

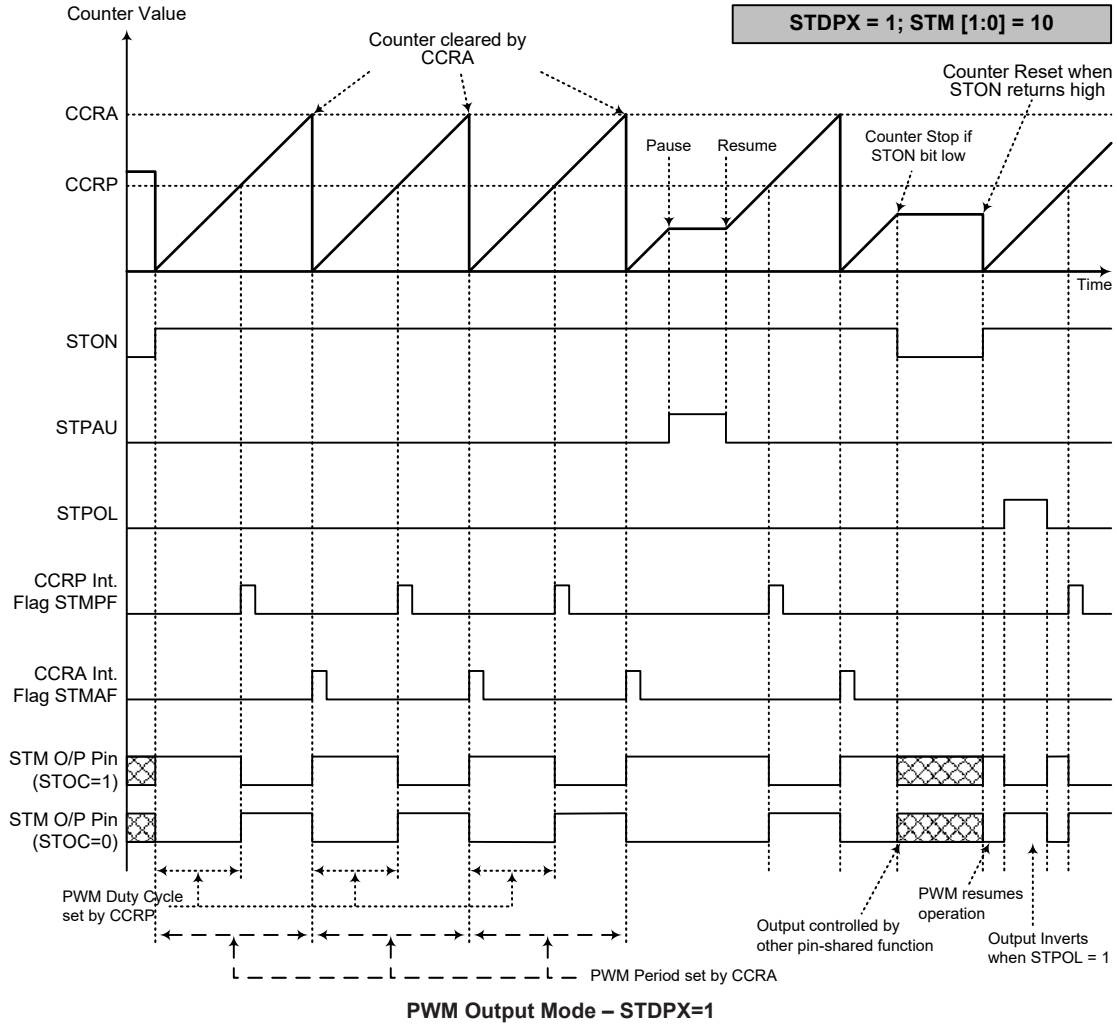
• **10-bit STM, PWM Output Mode, Edge-aligned Mode, STDPX=1**

| CCRP   | 001b | 010b | 011b | 100b | 101b | 110b | 111b | 000b |
|--------|------|------|------|------|------|------|------|------|
| Period | CCRA |      |      |      |      |      |      |      |
| Duty   | 128  | 256  | 384  | 512  | 640  | 768  | 896  | 1024 |

The PWM output period is determined by the CCRA register value together with the STM clock while the PWM duty cycle is defined by the CCRP register value except when the CCRP value is equal to 0.



- Note: 1. Here STDPX=0 – Counter cleared by CCRP  
 2. A counter clear sets the PWM Period  
 3. The internal PWM function continues running even when STIO[1:0]=00 or 01  
 4. The STCCLR bit has no influence on PWM operation



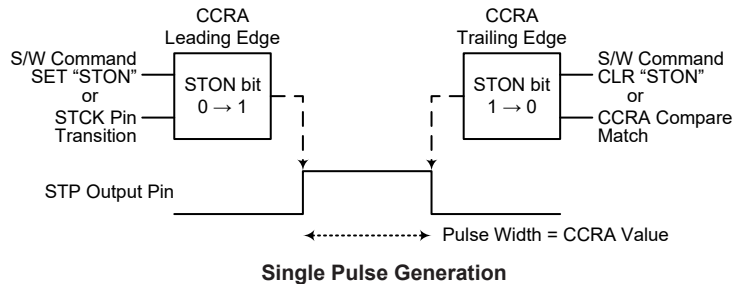
- Note: 1. Here STDPX=1 – Counter cleared by CCRA  
 2. A counter clear sets the PWM Period  
 3. The internal PWM function continues even when STIO[1:0]=00 or 01  
 4. The STCCLR bit has no influence on PWM operation

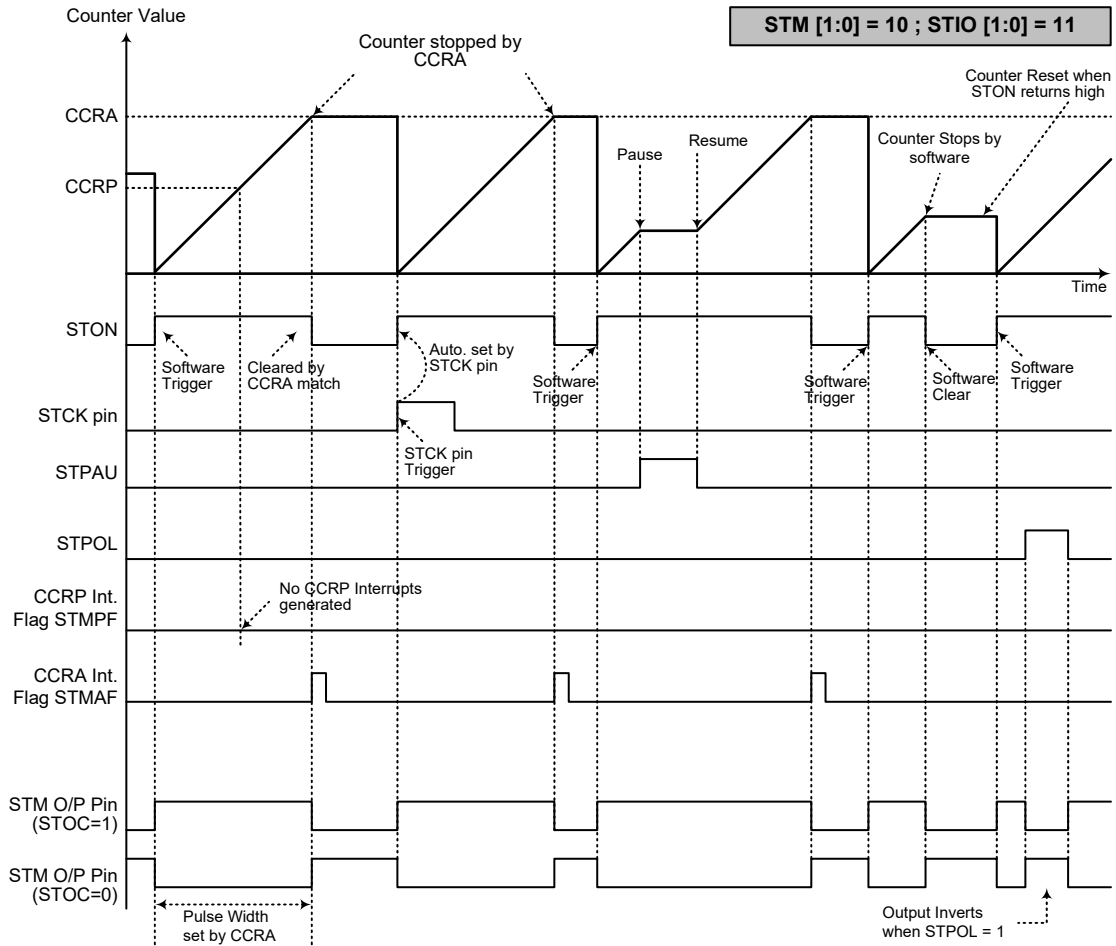
**Single Pulse Output Mode**

To select this mode, bits STM1 and STM0 in the STMC1 register should be set to “10” respectively and also the STIO1 and STIO0 bits should be set to “11” respectively. The Single Pulse Output Mode, as the name suggests, will generate a single shot pulse on the STM output pin.

The trigger for the pulse output leading edge is a low to high transition of the STON bit, which can be implemented using the application program. However, in the Single Pulse Output Mode, the STON bit can also be made to automatically change from low to high using the external STCK pin, which will in turn initiate the Single Pulse output. When the STON bit transitions to a high level, the counter will start running and the pulse leading edge will be generated. The STON bit should remain high when the pulse is in its active state. The generated pulse trailing edge will be generated when the STON bit is cleared to zero, which can be implemented using the application program or when a compare match occurs from Comparator A.

However, a compare match from Comparator A will also automatically clear the STON bit and thus generate the Single Pulse output trailing edge. In this way the CCRA value can be used to control the pulse width. A compare match from Comparator A will also generate a STM interrupt. The counter can only be reset back to zero when the STON bit changes from low to high when the counter restarts. In the Single Pulse Output Mode CCRP is not used. The STCCLR and STDPX bits are not used in this Mode.





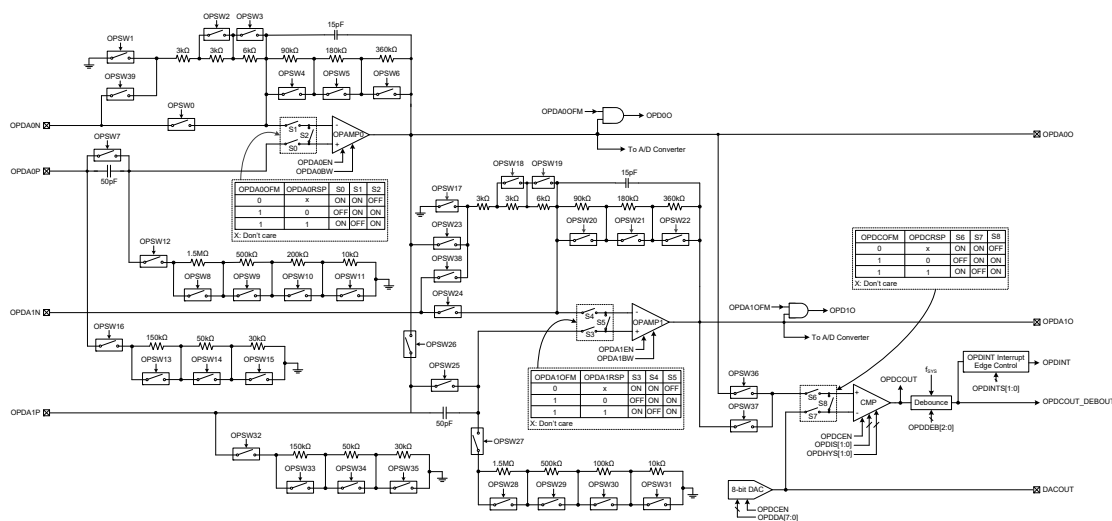
**Single Pulse Output Mode**

- Note:
1. Counter stopped by CCRA
  2. CCRP is not used
  3. The pulse triggered by the STCK pin or by setting the STON bit high
  4. A STCK pin active edge will automatically set the STON bit high
  5. In the Single Pulse Output Mode, STIO[1:0] must be set to "11" and can not be changed



## Proximity Sensing Circuit

The device includes a proximity sensing circuit which is composed of two operational amplifiers, a comparator and an 8-bit D/A converter. The two-stage operational amplifier circuit can amplify a small signal with a gain ranging from 1 to 44521. Users can choose different combinations according to different applications, no matter inverting or non-inverting amplification. And finally, the amplified analog signal will be compared with the D/A converter output reference voltage using a comparator.



**Proximity Sensing Circuit Block Diagram**

### Proximity Sensing Circuit Operation

The source voltage is input from OPDAnN and/or OPDAnP (n=0~1). The first stage operational amplifier chooses different amplifier modes through the switches OPDSW0~OPDSW16 and OPDSW39. Similarly, the second stage operational amplifier can also choose different modes through the switches OPDSW17~OPDSW35. The two OPAMPs both consist of a PGA function, the PGA gain can be positive or negative determined by the input voltage connecting to the positive input or negative input of PGA. The OPAMP0's gain can be 1~211 configured by OPDSW1~OPDSW6, the OPAMP1's gain can be 1~211 configured by OPDSW17~OPDSW22.

The input signal which is amplified by OPAMP0/OPAMP1 can be directly output on the OPDA00/OPDA10 pin, and also can be internally connected to the A/D converter selected by setting the relevant register for reading the amplified input voltage.

The D/A converter is used to generate reference voltage only for the comparator. The comparator compares the reference voltage and the amplified input voltage. The comparator output, OPDCOUT, will first be filtered with a certain debounce time period selected by the OPDDEB2~OPDDEB0 bits in the OPDC0 register. Then a filtered digital comparator output signal, OPDCOUT\_DEBOUT, is obtained. When this signal appears the correct edge type selected using the OPDINTS1~OPDINTS0 bits in the OPDC0 register, it will trigger an interrupt to inform the MCU.

### Proximity Sensing Circuit Registers

The overall operation of the Proximity Sensing Circuit is controlled using a series of registers. The OPDSWA~OPDSWE registers are used to control the analog switches. The OPDC0 register is used to control the operational amplifiers, comparator and D/A converter, select the OPDINT interrupt

edge and the comparator decoupling time. The OPDC1 register is used to control the comparator hysteresis voltage and bias current as well as the operational amplifiers' low current/high bandwidth selection. The OPDDA register is used to control the D/A converter output voltage. The OPDAnCAL and OPDCCAL registers are used to control the operational amplifiers and comparator input offset voltage calibration function.

| Register Name | Bit      |          |          |          |          |          |          |          |
|---------------|----------|----------|----------|----------|----------|----------|----------|----------|
|               | 7        | 6        | 5        | 4        | 3        | 2        | 1        | 0        |
| OPSWA         | OPSW7    | OPSW6    | OPSW5    | OPSW4    | OPSW3    | OPSW2    | OPSW1    | OPSW0    |
| OPSWB         | OPSW15   | OPSW14   | OPSW13   | OPSW12   | OPSW11   | OPSW10   | OPSW9    | OPSW8    |
| OPSWC         | OPSW23   | OPSW22   | OPSW21   | OPSW20   | OPSW19   | OPSW18   | OPSW17   | OPSW16   |
| OPSWD         | OPSW31   | OPSW30   | OPSW29   | OPSW28   | OPSW27   | OPSW26   | OPSW25   | OPSW24   |
| OPSWE         | OPSW39   | OPSW38   | OPSW37   | OPSW36   | OPSW35   | OPSW34   | OPSW33   | OPSW32   |
| OPDC0         | OPDA1EN  | OPDA0EN  | OPDCEN   | OPDINTS1 | OPDINTS0 | OPDDEB2  | OPDDEB1  | OPDDEB0  |
| OPDC1         | OPD1O    | OPD0O    | OPDHYS1  | OPDHYS0  | OPDIS1   | OPDIS0   | OPDA1BW  | OPDA0BW  |
| OPDDA         | D7       | D6       | D5       | D4       | D3       | D2       | D1       | D0       |
| OPDA0CAL      | OPDA0OFM | OPDA0RSP | OPDA0OF5 | OPDA0OF4 | OPDA0OF3 | OPDA0OF2 | OPDA0OF1 | OPDA0OF0 |
| OPDA1CAL      | OPDA1OFM | OPDA1RSP | OPDA1OF5 | OPDA1OF4 | OPDA1OF3 | OPDA1OF2 | OPDA1OF1 | OPDA1OF0 |
| OPDCCAL       | OPDCOUT  | OPDCOFM  | OPDCRSP  | OPDCOF4  | OPDCOF3  | OPDCOF2  | OPDCOF1  | OPDCOF0  |

**Proximity Sensing Circuit Register List**

• **OPSWA Register**

| Bit  | 7     | 6     | 5     | 4     | 3     | 2     | 1     | 0     |
|------|-------|-------|-------|-------|-------|-------|-------|-------|
| Name | OPSW7 | OPSW6 | OPSW5 | OPSW4 | OPSW3 | OPSW2 | OPSW1 | OPSW0 |
| R/W  | R/W   | R/W   | R/W   | R/W   | R/W   | R/W   | R/W   | R/W   |
| POR  | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     |

- Bit 7      **OPSW7:** OPSW7 switch on/off control  
0: Off  
1: On
- Bit 6      **OPSW6:** OPSW6 switch on/off control  
0: Off  
1: On
- Bit 5      **OPSW5:** OPSW5 switch on/off control  
0: Off  
1: On
- Bit 4      **OPSW4:** OPSW4 switch on/off control  
0: Off  
1: On
- Bit 3      **OPSW3:** OPSW3 switch on/off control  
0: Off  
1: On
- Bit 2      **OPSW2:** OPSW2 switch on/off control  
0: Off  
1: On
- Bit 1      **OPSW1:** OPSW1 switch on/off control  
0: Off  
1: On
- Bit 0      **OPSW0:** OPSW0 switch on/off control  
0: Off  
1: On

• **OPSWB Register**

| Bit  | 7      | 6      | 5      | 4      | 3      | 2      | 1     | 0     |
|------|--------|--------|--------|--------|--------|--------|-------|-------|
| Name | OPSW15 | OPSW14 | OPSW13 | OPSW12 | OPSW11 | OPSW10 | OPSW9 | OPSW8 |
| R/W  | R/W    | R/W    | R/W    | R/W    | R/W    | R/W    | R/W   | R/W   |
| POR  | 0      | 0      | 0      | 0      | 0      | 0      | 0     | 0     |

- Bit 7      **OPSW15:** OPSW15 switch on/off control  
             0: Off  
             1: On
- Bit 6      **OPSW14:** OPSW14 switch on/off control  
             0: Off  
             1: On
- Bit 5      **OPSW13:** OPSW13 switch on/off control  
             0: Off  
             1: On
- Bit 4      **OPSW12:** OPSW12 switch on/off control  
             0: Off  
             1: On
- Bit 3      **OPSW11:** OPSW11 switch on/off control  
             0: Off  
             1: On
- Bit 2      **OPSW10:** OPSW10 switch on/off control  
             0: Off  
             1: On
- Bit 1      **OPSW9:** OPSW9 switch on/off control  
             0: Off  
             1: On
- Bit 0      **OPSW8:** OPSW8 switch on/off control  
             0: Off  
             1: On

• **OPSWC Register**

| Bit  | 7      | 6      | 5      | 4      | 3      | 2      | 1      | 0      |
|------|--------|--------|--------|--------|--------|--------|--------|--------|
| Name | OPSW23 | OPSW22 | OPSW21 | OPSW20 | OPSW19 | OPSW18 | OPSW17 | OPSW16 |
| R/W  | R/W    | R/W    | R/W    | R/W    | R/W    | R/W    | R/W    | R/W    |
| POR  | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      |

- Bit 7      **OPSW23:** OPSW23 switch on/off control  
             0: Off  
             1: On
- Bit 6      **OPSW22:** OPSW22 switch on/off control  
             0: Off  
             1: On
- Bit 5      **OPSW21:** OPSW21 switch on/off control  
             0: Off  
             1: On
- Bit 4      **OPSW20:** OPSW20 switch on/off control  
             0: Off  
             1: On
- Bit 3      **OPSW19:** OPSW19 switch on/off control  
             0: Off  
             1: On

- Bit 2      **OPSW18:** OPSW18 switch on/off control  
            0: Off  
            1: On
- Bit 1      **OPSW17:** OPSW17 switch on/off control  
            0: Off  
            1: On
- Bit 0      **OPSW16:** OPSW16 switch on/off control  
            0: Off  
            1: On

• **OPSWD Register**

| Bit  | 7      | 6      | 5      | 4      | 3      | 2      | 1      | 0      |
|------|--------|--------|--------|--------|--------|--------|--------|--------|
| Name | OPSW31 | OPSW30 | OPSW29 | OPSW28 | OPSW27 | OPSW26 | OPSW25 | OPSW24 |
| R/W  | R/W    | R/W    | R/W    | R/W    | R/W    | R/W    | R/W    | R/W    |
| POR  | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      |

- Bit 7      **OPSW31:** OPSW31 switch on/off control  
            0: Off  
            1: On
- Bit 6      **OPSW30:** OPSW30 switch on/off control  
            0: Off  
            1: On
- Bit 5      **OPSW29:** OPSW29 switch on/off control  
            0: Off  
            1: On
- Bit 4      **OPSW28:** OPSW28 switch on/off control  
            0: Off  
            1: On
- Bit 3      **OPSW27:** OPSW27 switch on/off control  
            0: Off  
            1: On
- Bit 2      **OPSW26:** OPSW26 switch on/off control  
            0: Off  
            1: On
- Bit 1      **OPSW25:** OPSW25 switch on/off control  
            0: Off  
            1: On
- Bit 0      **OPSW24:** OPSW24 switch on/off control  
            0: Off  
            1: On

• **OPSW E Register**

| Bit  | 7      | 6      | 5      | 4      | 3      | 2      | 1      | 0      |
|------|--------|--------|--------|--------|--------|--------|--------|--------|
| Name | OPSW39 | OPSW38 | OPSW37 | OPSW36 | OPSW35 | OPSW34 | OPSW33 | OPSW32 |
| R/W  | R/W    | R/W    | R/W    | R/W    | R/W    | R/W    | R/W    | R/W    |
| POR  | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      |

- Bit 7      **OPSW39:** OPSW39 switch on/off control  
            0: Off  
            1: On
- Bit 6      **OPSW38:** OPSW38 switch on/off control  
            0: Off  
            1: On

- Bit 5      **OPSW37:** OPSW37 switch on/off control  
             0: Off  
             1: On
- Bit 4      **OPSW36:** OPSW36 switch on/off control  
             0: Off  
             1: On
- Bit 3      **OPSW35:** OPSW35 switch on/off control  
             0: Off  
             1: On
- Bit 2      **OPSW34:** OPSW34 switch on/off control  
             0: Off  
             1: On
- Bit 1      **OPSW33:** OPSW33 switch on/off control  
             0: Off  
             1: On
- Bit 0      **OPSW32:** OPSW32 switch on/off control  
             0: Off  
             1: On

• **OPDC0 Register**

| Bit  | 7       | 6       | 5      | 4        | 3        | 2       | 1       | 0       |
|------|---------|---------|--------|----------|----------|---------|---------|---------|
| Name | OPDA1EN | OPDA0EN | OPDCEN | OPDINTS1 | OPDINTS0 | OPDDEB2 | OPDDEB1 | OPDDEB0 |
| R/W  | R/W     | R/W     | R/W    | R/W      | R/W      | R/W     | R/W     | R/W     |
| POR  | 0       | 0       | 0      | 0        | 0        | 0       | 0       | 0       |

- Bit 7      **OPDA1EN:** OPD OPAMP1 enable or disable control  
             0: Disable – OPDA10 high impedance  
             1: Enable
- Bit 6      **OPDA0EN:** OPD OPAMP0 enable or disable control  
             0: Disable – OPDA00 high impedance  
             1: Enable
- Bit 5      **OPDCEN:** OPD Comparator and DAC enable or disable control  
             0: Disable – Comparator output pulled low; DACOUT high impedance  
             1: Enable
- Bit 4~3    **OPDINTS1~OPDINTS0:** OPDINT interrupt edge control  
             00: Disable  
             01: Rising edge  
             10: Falling edge  
             11: Rising and Falling edges
- Bit 2~0    **OPDDEB2~OPDDEB0:** OPD Comparator debounce time control  
             000: Bypass, without debounce  
             001:  $(1\sim 2) \times t_{DEB}$   
             010:  $(3\sim 4) \times t_{DEB}$   
             011:  $(7\sim 8) \times t_{DEB}$   
             100:  $(15\sim 16) \times t_{DEB}$   
             101:  $(31\sim 32) \times t_{DEB}$   
             110:  $(63\sim 64) \times t_{DEB}$   
             111:  $(127\sim 128) \times t_{DEB}$   
             Note:  $t_{DEB} = 1/f_{SYS}$

• **OPDC1 Register**

| Bit  | 7     | 6     | 5       | 4       | 3      | 2      | 1       | 0       |
|------|-------|-------|---------|---------|--------|--------|---------|---------|
| Name | OPD1O | OPD0O | OPDHYS1 | OPDHYS0 | OPDIS1 | OPDIS0 | OPDA1BW | OPDA0BW |
| R/W  | R     | R     | R/W     | R/W     | R/W    | R/W    | R/W     | R/W     |
| POR  | 0     | 0     | 0       | 0       | 0      | 0      | 0       | 0       |

- Bit 7      **OPD1O**: OPAMP1 output under offset calibration
- Bit 6      **OPD0O**: OPAMP0 output under offset calibration
- Bit 5~4    **OPDHYS1~OPDHYS0**: OPD Comparator hysteresis voltage window control  
Refer to Comparator Characteristics section.
- Bit 3~2    **OPDIS1~OPDIS0**: OPD Comparator bias current control  
Refer to Comparator Characteristic.
- Bit 1      **OPDA1BW**: OPD OPAMP1 low current / high bandwidth selection  
0: Low current  
1: High bandwidth
- Bit 0      **OPDA0BW**: OPD OPAMP0 low current / high bandwidth selection  
0: Low current  
1: High bandwidth

• **OPDDA Register**

| Bit  | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   |
|------|-----|-----|-----|-----|-----|-----|-----|-----|
| Name | D7  | D6  | D5  | D4  | D3  | D2  | D1  | D0  |
| R/W  | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| POR  | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |

- Bit 7~0    **D7~D0**: OPD DAC output voltage control  
DAC  $V_{OUT} = (V_{DD}/256) \times D[7:0]$

• **OPDA0CAL Register**

| Bit  | 7        | 6        | 5        | 4        | 3        | 2        | 1        | 0        |
|------|----------|----------|----------|----------|----------|----------|----------|----------|
| Name | OPDA0OFM | OPDA0RSP | OPDA0OF5 | OPDA0OF4 | OPDA0OF3 | OPDA0OF2 | OPDA0OF1 | OPDA0OF0 |
| R/W  | R/W      | R/W      | R/W      | R/W      | R/W      | R/W      | R/W      | R/W      |
| POR  | 0        | 0        | 1        | 0        | 0        | 0        | 0        | 0        |

- Bit 7      **OPDA0OFM**: OPD OPAMP0 normal operation or input offset voltage calibration mode selection  
0: Normal operation  
1: Offset calibration mode
- Bit 6      **OPDA0RSP**: OPD OPAMP0 input offset voltage calibration reference selection  
0: Select inverting input as the reference voltage input  
1: Select non-inverting input as the reference voltage input
- Bit 5~0    **OPDA0OF5~OPDA0OF0**: OPD OPAMP0 input offset voltage calibration control  
This 6-bit field is used to perform the OPD OPAMP0 input offset calibration operation and the value for the OPD OPAMP0 input offset calibration can be restored into this bit field. More detailed information is described in the “Operational Amplifier Input Offset Calibration” section.

• **OPDA1CAL Register**

| Bit  | 7        | 6        | 5        | 4        | 3        | 2        | 1        | 0        |
|------|----------|----------|----------|----------|----------|----------|----------|----------|
| Name | OPDA1OFM | OPDA1RSP | OPDA1OF5 | OPDA1OF4 | OPDA1OF3 | OPDA1OF2 | OPDA1OF1 | OPDA1OF0 |
| R/W  | R/W      | R/W      | R/W      | R/W      | R/W      | R/W      | R/W      | R/W      |
| POR  | 0        | 0        | 1        | 0        | 0        | 0        | 0        | 0        |

Bit 7 **OPDA1OFM**: OPD OPAMP1 normal operation or input offset voltage calibration mode selection

- 0: Normal operation
- 1: Offset calibration mode

Bit 6 **OPDA1RSP**: OPD OPAMP1 input offset voltage calibration reference selection

- 0: Select inverting input as the reference voltage input
- 1: Select non-inverting input as the reference voltage input

Bit 5~0 **OPDA1OF5~OPDA1OF0**: OPD OPAMP1 input offset voltage calibration control

This 6-bit field is used to perform the OPD OPAMP1 input offset calibration operation and the value for the OPD OPAMP1 input offset calibration can be restored into this bit field. More detailed information is described in the “Operational Amplifier Input Offset Calibration” section.

• **OPDCCAL Register**

| Bit  | 7       | 6       | 5       | 4       | 3       | 2       | 1       | 0       |
|------|---------|---------|---------|---------|---------|---------|---------|---------|
| Name | OPDCOUT | OPDCOFM | OPDCRSP | OPDCOF4 | OPDCOF3 | OPDCOF2 | OPDCOF1 | OPDCOF0 |
| R/W  | R       | R/W     | R/W     | R/W     | R/W     | R/W     | R/W     | R/W     |
| POR  | 0       | 0       | 0       | 1       | 0       | 0       | 0       | 0       |

Bit 7 **OPDCOUT**: OPD Comparator output

- 0: Non-inverting input voltage < DAC output voltage
- 1: Non-inverting input voltage > DAC output voltage

Bit 6 **OPDCOFM**: OPD Comparator normal operation or input offset voltage calibration mode selection

- 0: Normal operation
- 1: Offset calibration mode

Bit 5 **OPDCRSP**: OPD Comparator input offset voltage calibration reference selection

- 0: Select inverting input as the reference voltage input
- 1: Select non-inverting input as the reference voltage input

Bit 4~0 **OPDCOF4~OPDCOF0**: OPD Comparator input offset voltage calibration control

This 5-bit field is used to perform the OPD comparator input offset calibration operation and the value for the OPD Comparator input offset calibration can be restored into this bit field. More detailed information is described in the “Comparator Input Offset Calibration” section.

**Offset Calibration Procedure**

To operate in the input offset calibration mode for the Operational Amplifiers or the Comparator, the OPDAnOFM or OPDCOFM bit should first be set to “1” followed by the reference input selection by configuring the OPDAnRSP or OPDCRSP bit. Note that as the Operational Amplifier inputs are pin-shared with I/O or other functions, before the calibration, they should be configured as the OPD operational amplifier input pin function first by correctly setting the related pin-shared function register.

### Operational Amplifier Input Offset Calibration

- Step1. Set OPDAnOFM=1 and OPDAnRSP=1, the Operational Amplifier n is now under offset calibration mode. To make sure  $V_{AnOS}$  as minimal as possible after calibration, the input reference voltage in calibration should be the same as input DC operating voltage in normal mode operation.
- Step2. Set OPDAnOF[5:0]=000000 and then read OPDnO bit
- Step3. Increase the OPDAnOF[5:0] value by 1 and then read the OPDnO bit.
- If the OPDnO bit state has not changed, then repeat Step 3 until the OPDnO bit state has changed.
- If the OPDnO bit state has changed, record the OPDAnOF[5:0] value as  $V_{AnOS1}$  and then go to Step 4.
- Step4. Set OPDAnOF[5:0]=111111 then read OPDnO bit.
- Step5. Decrease the OPDAnOF[5:0] value by 1 and then read the OPDnO bit.
- If the OPDnO bit state has not changed, then repeat Step 5 until the OPDnO bit state has changed.
- If the OPDnO bit state has changed, record the OPDAnOF[5:0] value as  $V_{AnOS2}$  and then go to Step 6.
- Step6. Restore the Operational Amplifier n input offset calibration value  $V_{AnOS}$  into the OPDAnOF[5:0] bit field. The offset Calibration procedure is now finished.
- $V_{AnOS}=(V_{AnOS1}+V_{AnOS2})/2$ . If  $(V_{AnOS1}+V_{AnOS2})/2$  is not integral, discard the decimal.

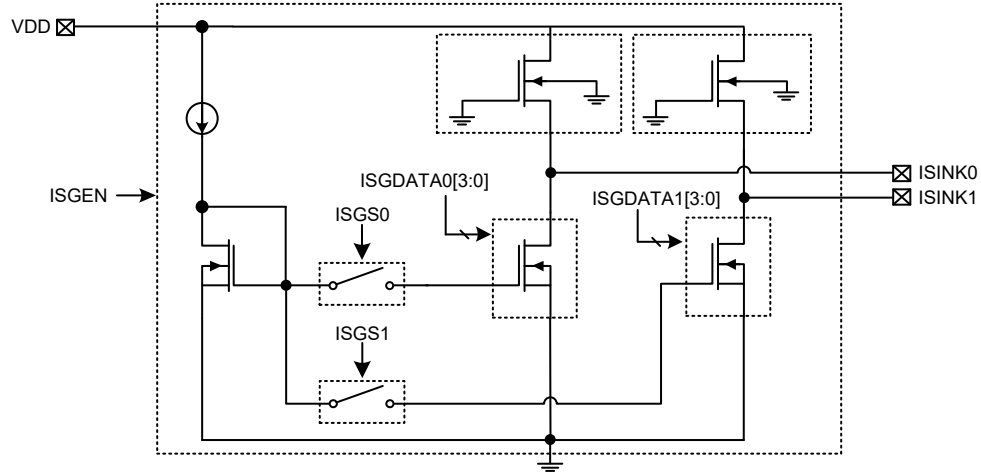
### Comparator Input Offset Calibration

- Step1. Set OPDCOFM=1 and OPDCRSP=1, the Comparator is now operating in the comparator input offset calibration mode. To make sure  $V_{COS}$  as minimal as possible after calibration, the input reference voltage in calibration should be the same as input DC operating voltage in normal operation.
- Step2. Set OPDCOF[4:0]=00000 and read the OPDCOUT bit.
- Step3. Increase the OPDCOF[4:0] value by 1 and then read the OPDCOUT bit.
- If the OPDCOUT bit state has not changed, then repeat Step 3 until the OPDCOUT bit state has changed.
- If the OPDCOUT bit state has changed, record the OPDCOF[4:0] value as  $V_{COS1}$  and then go to Step 4.
- Step4. Set OPDCOF[4:0]=11111 and then read the OPDCOUT bit.
- Step5. Decrease the OPDCOF[4:0] value by 1 and then read the OPDCOUT bit.
- If the OPDCOUT bit state has not changed, then repeat Step 5 until the OPDCOUT bit state has changed.
- If the OPDCOUT bit state has changed, record the OPDCOF[4:0] value as  $V_{COS2}$  and then go to Step 6.
- Step6. Restore the Comparator input offset calibration value  $V_{COS}$  into the OPDCOF[4:0] bit field. The offset Calibration procedure is now finished.
- $V_{COS}=(V_{COS1}+V_{COS2})/2$ . If  $(V_{COS1}+V_{COS2})/2$  is not integral, discard the decimal.



## Sink Current Generator

The sink current source generator could provide constant current no matter what  $V_{ISINK}$  voltage is from 0.7V to 4.5V. The constant current value is controlled by the ISGDATA0/ISGDATA1 register, and the sink current range is 1mA~192mA.



Note: For the 16-pin NSOP package where ISINK0 and ISINK1 are bounded to the same pin location, or the 16-pin QFN/24-pin SSOP package where these two pins may be externally bounded together by users, when ISGS0/ISGS1=0, the corresponding ISGDATA0/ISGDATA1 register must be cleared to zero to avoid current leakage problem.

**Sink Current Generator Block Diagram**

## Sink Current Generator Registers

There is a series of registers controlling the overall operation of the Sink Current Generator function.

| Register Name | Bit   |   |   |       |    |    |       |       |
|---------------|-------|---|---|-------|----|----|-------|-------|
|               | 7     | 6 | 5 | 4     | 3  | 2  | 1     | 0     |
| ISGENC        | ISGEN | — | — | —     | —  | —  | ISGS1 | ISGS0 |
| ISGDATA0      | —     | — | — | ISST0 | D3 | D2 | D1    | D0    |
| ISGDATA1      | —     | — | — | ISST1 | D3 | D2 | D1    | D0    |

**Sink Current Generator Register List**

### • ISGENC Register

| Bit  | 7     | 6 | 5 | 4 | 3 | 2 | 1     | 0     |
|------|-------|---|---|---|---|---|-------|-------|
| Name | ISGEN | — | — | — | — | — | ISGS1 | ISGS0 |
| R/W  | R/W   | — | — | — | — | — | R/W   | R/W   |
| POR  | 0     | — | — | — | — | — | 0     | 0     |

Bit 7 **ISGEN**: Sink current generator enable control  
 0: Disable  
 1: Enable

When the ISGEN bit is cleared to zero to disable the sink current generator, the ISINK0 and ISINK1 pin statuses are  $V_{ISINK0/1}$ =floating,  $I_{ISINK0/1}$ =0.

Bit 6~2 Unimplemented, read as “0”

- Bit 1      **ISGS1**: ISINK1 switch on/off control  
                  0: Off  
                  1: On
- Bit 1      **ISGS0**: ISINK0 switch on/off control  
                  0: Off  
                  1: On

• **ISGDATA0 Register**

| Bit  | 7 | 6 | 5 | 4     | 3   | 2   | 1   | 0   |
|------|---|---|---|-------|-----|-----|-----|-----|
| Name | — | — | — | ISST0 | D3  | D2  | D1  | D0  |
| R/W  | — | — | — | R/W   | R/W | R/W | R/W | R/W |
| POR  | — | — | — | 0     | 0   | 0   | 0   | 0   |

- Bit 7~5      Unimplemented, read as “0”
- Bit 4      **ISST0**: Switch first stage or second stage current control  
                  0: First stage  
                  1: Second stage
- Bit 3~0      **D3~D0**: Sink current control for the ISINK0 pin  
                  First stage current value (mA)=1+1×D[3:0], 1mA/step  
                  Second stage current value (mA)=16+11+11×D[3:0], 11mA/step  
                  Maximum current (mA)=16mA+176mA=192mA

• **ISGDATA1 Register**

| Bit  | 7 | 6 | 5 | 4     | 3   | 2   | 1   | 0   |
|------|---|---|---|-------|-----|-----|-----|-----|
| Name | — | — | — | ISST1 | D3  | D2  | D1  | D0  |
| R/W  | — | — | — | R/W   | R/W | R/W | R/W | R/W |
| POR  | — | — | — | 0     | 0   | 0   | 0   | 0   |

- Bit 7~5      Unimplemented, read as “0”
- Bit 4      **ISST1**: Switch first stage or second stage current control  
                  0: First stage  
                  1: Second stage
- Bit 3~0      **D3~D0**: Sink current control for the ISINK1 pin  
                  First stage current value (mA)=1+1×D[3:0], 1mA/step  
                  Second stage current value (mA)=16+11+11×D[3:0], 11mA/step  
                  Maximum current (mA)=16mA+176mA=192mA

## Analog to Digital Converter

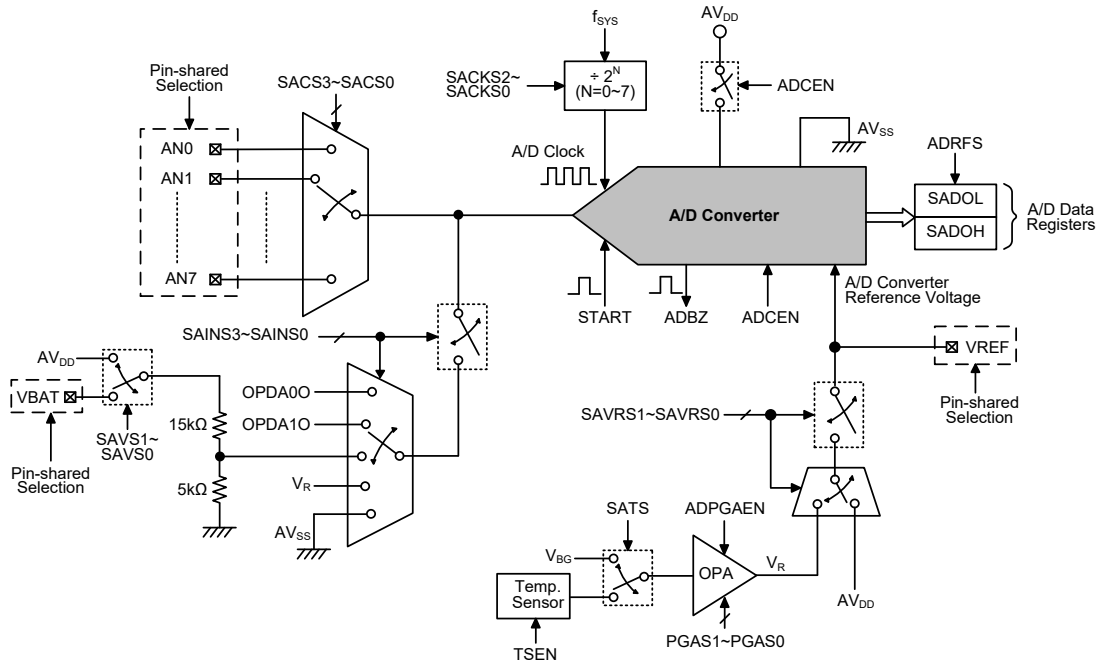
The need to interface to real world analog signals is a common requirement for many electronic systems. However, to properly process these signals by a microcontroller, they must first be converted into digital signals by A/D converters. By integrating the A/D conversion electronic circuitry into the microcontroller, the need for external components is reduced significantly with the corresponding follow-on benefits of lower costs and reduced component space requirements.

### A/D Converter Overview

The device contains a multi-channel analog to digital converter which can directly interface to external analog signals, such as that from sensors or other control signals and convert these signals directly into a 12-bit digital value. It also can convert the internal signals, such as the Proximity Sensing Circuit operational amplifier outputs, into a 12-bit digital value. The external or internal analog signal to be converted is determined by the SAINS2~SAINS0 bits together with the SACS3~SACS0 bits. More detailed information about the A/D input signal is described in the “A/D Converter Control Registers” and “A/D Converter Input Signals” sections respectively.

| External Input Channels | Internal Signals  | Channel Select Bits           |
|-------------------------|---|-------------------------------|
| 8: AN0~AN7              | 6: OPDA00, OPDA10,<br>AV <sub>DD</sub> /4, V <sub>BAT</sub> /4, V <sub>R</sub> , AV <sub>SS</sub> | SAINS2~SAINS0,<br>SACS3~SACS0 |

The accompanying block diagram shows the overall internal structure of the A/D converter, together with its associated registers.



**A/D Converter Structure**

### A/D Converter Register Description

Overall operation of the A/D converter is controlled using several registers. A read only register pair exists to store the A/D converter data 12-bit value. The remaining three registers are control registers which setup the operating and control function of the A/D converter.

| Register Name   | Bit     |        |        |        |        |        |        |        |
|-----------------|---------|--------|--------|--------|--------|--------|--------|--------|
|                 | 7       | 6      | 5      | 4      | 3      | 2      | 1      | 0      |
| SADOL (ADRF5=0) | D3      | D2     | D1     | D0     | —      | —      | —      | —      |
| SADOL (ADRF5=1) | D7      | D6     | D5     | D4     | D3     | D2     | D1     | D0     |
| SAD0H (ADRF5=0) | D11     | D10    | D9     | D8     | D7     | D6     | D5     | D4     |
| SAD0H (ADRF5=1) | —       | —      | —      | —      | D11    | D10    | D9     | D8     |
| SADC0           | START   | ADBZ   | ADCEN  | ADRF5  | SACS3  | SACS2  | SACS1  | SACS0  |
| SADC1           | SAINS2  | SAINS1 | SAINS0 | SAVRS1 | SAVRS0 | SACKS2 | SACKS1 | SACKS0 |
| SADC2           | ADPGAEN | PGAGS1 | PGAGS0 | D4     | SAVS1  | SAVS0  | TS_SEL | TSEN   |

**A/D Converter Register List**

### A/D Converter Data Registers – SADOL, SAD0H

As the device contains an internal 12-bit A/D converter, it requires two data registers to store the converted value. These are a high byte register, known as SAD0H, and a low byte register, known as SADOL. After the conversion process takes place, these registers can be directly read by the microcontroller to obtain the digitised conversion value. As only 12 bits of the 16-bit register space is utilised, the format in which the data is stored is controlled by the ADRF5 bit in the SADC0 register as shown in the accompanying table. D0~D11 are the A/D conversion result data bits. Any unused bits will be read as zero. Note that A/D Converter data register contents will be unchanged if the A/D converter is disabled.

| ADRF5 | SAD0H |     |    |    |     |     |    |    | SADOL |    |    |    |    |    |    |    |
|-------|-------|-----|----|----|-----|-----|----|----|-------|----|----|----|----|----|----|----|
|       | 7     | 6   | 5  | 4  | 3   | 2   | 1  | 0  | 7     | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
| 0     | D11   | D10 | D9 | D8 | D7  | D6  | D5 | D4 | D3    | D2 | D1 | D0 | 0  | 0  | 0  | 0  |
| 1     | 0     | 0   | 0  | 0  | D11 | D10 | D9 | D8 | D7    | D6 | D5 | D4 | D3 | D2 | D1 | D0 |

**A/D Converter Data Registers**

### A/D Converter Control Registers – SADC0, SADC1, SADC2

To control the function and operation of the A/D converter, three control registers known as SADC0, SADC1 and SADC2 are provided. These 8-bit registers define functions such as the selection of which analog channel is connected to the internal A/D converter, the digitised data format, the A/D clock source as well as controlling the start function and monitoring the A/D converter busy status, etc. As the device contains only one actual analog to digital converter hardware circuit, each of the external or internal analog signal inputs must be routed to the converter. The SACS3~SACS0 bits in the SADC0 register are used to determine which external channel input is selected to be converted. The SAINS2~SAINS0 bits in the SADC1 register are used to determine that the analog signal to be converted comes from the internal analog signal or external analog channel input.

The relevant pin-shared function selection bits determine which pins on I/O Ports are used as analog inputs for the A/D converter input and which pins are not to be used as the A/D converter input. When the pin is selected to be an A/D input, its original function whether it is an I/O or other pin-shared function will be removed. In addition, any internal pull-high resistor connected to the pin will be automatically removed if the pin is selected to be an A/D converter input.

• **SADC0 Register**

| Bit  | 7     | 6    | 5     | 4     | 3     | 2     | 1     | 0     |
|------|-------|------|-------|-------|-------|-------|-------|-------|
| Name | START | ADBZ | ADCEN | ADRFS | SACS3 | SACS2 | SACS1 | SACS0 |
| R/W  | R/W   | R    | R/W   | R/W   | R/W   | R/W   | R/W   | R/W   |
| POR  | 0     | 0    | 0     | 0     | 0     | 0     | 0     | 0     |

- Bit 7**     **START**: Start the A/D conversion  
0→1→0: Start  
This bit is used to initiate an A/D conversion process. The bit is normally low but if set high and then cleared low again, the A/D converter will initiate a conversion process.
- Bit 6**     **ADBZ**: A/D converter busy flag  
0: No A/D conversion is in progress  
1: A/D conversion is in progress  
This read only flag is used to indicate whether the A/D conversion is in progress or not. When the START bit is set from low to high and then to low again, the ADBZ flag will be set to 1 to indicate that the A/D conversion is initiated. The ADBZ flag will be cleared to 0 after the A/D conversion is complete.
- Bit 5**     **ADCEN**: A/D converter function enable control  
0: Disable  
1: Enable  
This bit controls the A/D internal function. This bit should be set to one to enable the A/D converter. If the bit is set low, then the A/D converter will be switched off reducing the device power consumption. When the A/D converter function is disabled, the contents of the A/D data register pair known as SADOH and SADOL will be unchanged.
- Bit 4**     **ADRFS**: A/D converter data format selection  
0: A/D converter data format → SADOH=D[11:4]; SADOL=D[3:0]  
1: A/D converter data format → SADOH=D[11:8]; SADOL=D[7:0]  
This bit controls the format of the 12-bit converted A/D value in the two A/D data registers. Details are provided in the A/D data register section.
- Bit 3~0**   **SACS3~SACS0**: A/D converter external analog channel input selection  
0000: AN0  
0001: AN1  
0010: AN2  
0011: AN3  
0100: AN4  
0101: AN5  
0110: AN6  
0111: AN7  
1000~1111: Non-existed channel, the input will be floating

• **SADC1 Register**

| Bit  | 7      | 6      | 5      | 4      | 3      | 2      | 1      | 0      |
|------|--------|--------|--------|--------|--------|--------|--------|--------|
| Name | SAINS2 | SAINS1 | SAINS0 | SAVRS1 | SAVRS0 | SACKS2 | SACKS1 | SACKS0 |
| R/W  | R/W    | R/W    | R/W    | R/W    | R/W    | R/W    | R/W    | R/W    |
| POR  | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      |

- Bit 7~5**   **SAINS2~SAINS0**: A/D converter input signal selection  
000: External input – External analog channel input  
001: Internal input – Proximity Sensing Circuit OPAMP0 output, OPDA00  
010: Internal input – Proximity Sensing Circuit OPAMP1 output, OPDA10  
011: Internal input – Ground, AV<sub>SS</sub>  
100: Internal input – A/D converter power supply voltage AV<sub>DD</sub>/4 or external input voltage V<sub>BAT</sub>/4  
101: Internal input – A/D converter OPA output voltage, V<sub>R</sub>  
110~111: External input – External analog channel input

Care must be taken if the SAINS2~SAINS0 bits are set to “001”~“101” to select the internal analog signal to be converted. When the internal analog signal is selected to be converted, the external input pin must never be selected as the A/D input signal by properly setting the SACS3~SACS0 bits with a value from “1000” to “1111”. Otherwise, the external channel input will be connected together with the internal analog signal. This will result in unpredictable situations such as an irreversible damage.

When the SAINS2~SAINS0 bits are set to “100”, the actual voltage input source is selected by the SAVS1~SAVS0 bits in the SADC2 register.

- Bit 4~3    **SAVRS1~SAVRS0**: A/D converter reference voltage selection
- 00: From external VREF pin
  - 01: From internal A/D converter power  $AV_{DD}$
  - 10: From internal A/D converter OPA output voltage  $V_R$
  - 11: From internal A/D converter power  $AV_{DD}$

These bits are used to select the A/D converter reference voltage. Care must be taken if the SAVRS1~SAVRS0 bits are set to “01”~“11” to select the internal A/D converter power or OPA output as the reference voltage source. When the internal A/D converter power or OPA output is selected as the reference voltage, the VREF pin cannot be configured as the reference voltage input by properly configuring the corresponding pin-shared function control bits. Otherwise, the external input voltage on VREF pin will be connected to the internal A/D converter power.

- Bit 2~0    **SACKS2~SACKS0**: A/D conversion clock source selection
- 000:  $f_{SYS}$
  - 001:  $f_{SYS}/2$
  - 010:  $f_{SYS}/4$
  - 011:  $f_{SYS}/8$
  - 100:  $f_{SYS}/16$
  - 101:  $f_{SYS}/32$
  - 110:  $f_{SYS}/64$
  - 111:  $f_{SYS}/128$

These three bits are used to select the clock source for the A/D converter.

• **SADC2 Register**

| Bit  | 7       | 6      | 5      | 4   | 3     | 2     | 1      | 0    |
|------|---------|--------|--------|-----|-------|-------|--------|------|
| Name | ADPGAEN | PGAGS1 | PGAGS0 | D4  | SAVS1 | SAVS0 | TS_SEL | TSEN |
| R/W  | R/W     | R/W    | R/W    | R/W | R/W   | R/W   | R/W    | R/W  |
| POR  | 0       | 0      | 0      | 0   | 0     | 0     | 0      | 0    |

- Bit 7    **ADPGAEN**: PGA enable control
- 0: Disable
  - 1: Enable

- Bit 6~5    **PGAGS1~PGAGS0**: PGA gain selection
- 00:  $V_R=V_{TS}$ , SATS=1
  - 01:  $V_R=1.6V$ , SATS=0
  - 10:  $V_R=3.0V$ , SATS=0
  - 11:  $V_R=4.0V$ , SATS=0

When these bits are set to 01/10/11, the Bandgap must be enabled by setting the VBGEN bit in the VBGC register to 1.

- Bit 4    **D4**: Reserved bit, must be fixed at “0”
- Bit 3~2    **SAVS1~SAVS0**: Voltage input selection
- 00: Floating
  - 01:  $AV_{DD}$
  - 10: VBAT
  - 11: Cannot be used

As the VBAT pin is pin-shared with other functions, when the SAVS bit field is set to “10”, the VBAT pin-shared function control bit should be properly configured to disable other pin function.

Bit 1 **TS\_SEL**: Application option for better temperature resolution when SAVRS[1:0]=01 or 11  
0: Suitable for  $V_{DD}(\min.) > 1.8V$  situation.

When  $TS\_SEL=0$ , select  $V_R=V_{TS}$  and  $V_R=1.6V$  as a pair to measure the temperature value; select  $V_{DD}=3V$  when using the writer to calibrate  $T_{OS}$  value.

1: Suitable for for  $V_{DD}(\min.) > 3.3V$  situation

When  $TS\_SEL=1$ , select  $V_R=V_{TS}$  and  $V_R=3.0V$  as a pair to measure the temperature value; select  $V_{DD}=5V$  when using the writer to calibrate  $T_{OS}$  value.

To achieve a better temperature resolution, it is recommended to switch this bit setting and select the corresponding  $V_R$  according to the actual  $V_{DD}$  situation.

Bit 0 **TSEN**: Temperature Sensor enable control

0: Disable

1: Enable

This bit controls the internal temperature sensor circuitry. If the temperature sensor output will be converted or the temperature sensor output will be selected as the A/D conversion reference voltage, the temperature sensor circuitry should be turned on by setting the TSEN bit high first. When the temperature sensor is enabled by setting the TSEN bit to 1, a time named as  $t_{TSS}$  should be allowed for the temperature sensor circuit to stabilise before implementing relevant temperature sensor operation.

## A/D Converter Operation

The START bit in the SADC0 register is used to start the AD conversion. When the microcontroller sets this bit from low to high and then low again, an analog to digital conversion cycle will be initiated.

The ADBZ bit in the SADC0 register is used to indicate whether the analog to digital conversion process is in progress or not. This bit will be automatically set to 1 by the microcontroller after an A/D conversion is successfully initiated. When the A/D conversion is complete, the ADBZ will be cleared to 0. In addition, the corresponding A/D interrupt request flag will be set in the interrupt control register, and if the interrupts are enabled, an appropriate internal interrupt signal will be generated. This A/D internal interrupt signal will direct the program flow to the associated A/D internal interrupt address for processing. If the A/D internal interrupt is disabled, the microcontroller can poll the ADBZ bit in the SADC0 register to check whether it has been cleared as an alternative method of detecting the end of an A/D conversion cycle.

The clock source for the A/D converter, which originates from the system clock  $f_{SYS}$ , can be chosen to be either  $f_{SYS}$  or a subdivided version of  $f_{SYS}$ . The division ratio value is determined by the SACKS2~SACKS0 bits in the SADC1 register. Although the A/D clock source is determined by the system clock  $f_{SYS}$  and by bits SACKS2~SACKS0, there are some limitations on the maximum A/D clock source speed that can be selected. As the recommended range of permissible A/D clock period,  $t_{ADCK}$ , is from 0.5 $\mu s$  to 10 $\mu s$ , care must be taken for system clock frequencies. For example, as the system clock operates at a frequency of 8MHz, the SACKS2~SACKS0 bits should not be set to 000, 001 or 111. Doing so will give A/D clock periods that are less or larger than the minimum or maximum A/D clock period which may result in inaccurate A/D conversion values. Refer to the following table for examples, where values marked with an asterisk \* show where special care must be taken, as the values may be less or larger than the specified A/D Clock Period range.

| f <sub>sys</sub> | A/D Clock Period (t <sub>ADCK</sub> )     |   |   |   |  |  |  |   |
|------------------|---|---|---|---|--|--|--|---|
|                  | SACKS[2:0]<br>=000<br>(f <sub>sys</sub> ) | SACKS[2:0]<br>=001<br>(f <sub>sys</sub> /2) | SACKS[2:0]<br>=010<br>(f <sub>sys</sub> /4) | SACKS[2:0]<br>=011<br>(f <sub>sys</sub> /8) | SACKS[2:0]<br>=100<br>(f <sub>sys</sub> /16) | SACKS[2:0]<br>=101<br>(f <sub>sys</sub> /32) | SACKS[2:0]<br>=110<br>(f <sub>sys</sub> /64) | SACKS[2:0]<br>=111<br>(f <sub>sys</sub> /128) |
| 1MHz             | 1μs                                       | 2μs   | 4μs   | 8μs   | 16μs *                                       | 32μs *                                       | 64μs *                                       | 128μs *                                       |
| 2MHz             | 500ns                                     | 1μs   | 2μs   | 4μs   | 8μs  | 16μs *                                       | 32μs *                                       | 64μs *  |
| 4MHz             | 250ns *                                   | 500ns                                       | 1μs   | 2μs   | 4μs  | 8μs  | 16μs *                                       | 32μs *  |
| 8MHz             | 125ns *                                   | 250ns *                                     | 500ns                                       | 1μs   | 2μs  | 4μs  | 8μs  | 16μs *  |

**A/D Clock Period Examples**

Controlling the power on/off function of the A/D converter circuitry is implemented using the ADCEN bit in the SADC0 register. This bit must be set high to power on the A/D converter. When the ADCEN bit is set high to power on the A/D converter internal circuitry a certain delay, as indicated in the timing diagram, must be allowed before an A/D conversion is initiated. Even if no pins are selected for use as A/D inputs, if the ADCEN bit is high, then some power will still be consumed. In power conscious applications it is therefore recommended that the ADCEN is set low to reduce power consumption when the A/D converter function is not being used.

### A/D Converter Reference Voltage

The reference voltage supply to the A/D converter can be supplied from the power supply AV<sub>DD</sub>, or from the A/D converter OPA output V<sub>R</sub>, or from an external reference source supplied on pin VREF. The desired selection is made using the SAVRS1~SAVRS0 bits. When the SAVRS bit field is set to “01” or “11”, the A/D converter reference voltage will come from the AV<sub>DD</sub>. When the SAVRS bit field is set to “10”, the A/D converter reference voltage will come from the OPA output. Otherwise, the A/D converter reference voltage will come from the VREF pin. As the VREF pin is pin-shared with other functions, when the VREF pin is selected as the reference voltage supply pin, the VREF pin-shared function control bit should be properly configured to disable other pin function. However, if the internal A/D converter power AV<sub>DD</sub> or the OPA output V<sub>R</sub> is selected as the reference voltage, the VREF pin must not be configured as the reference voltage input function to avoid the internal connection between the VREF pin and the internal reference signal.

The analog input values must not be allowed to exceed the value of the selected A/D reference voltage.

| SAVRS[1:0] | Reference        | Description                                 |
|------------|------------------|---|
| 00         | VREF pin         | External A/D converter reference pin VREF   |
| 01         | AV <sub>DD</sub> | Internal A/D converter power supply voltage |
| 10         | V <sub>R</sub>   | Internal A/D converter OPA output voltage   |
| 11         | AV <sub>DD</sub> | Internal A/D converter power supply voltage |

**A/D Converter Reference Voltage Selection**

### A/D Converter Input Signals

All the external A/D analog channel input pins are pin-shared with the I/O pins as well as other functions. The corresponding control bits for each A/D external input pin in the PxS0 and PxS1 registers determine whether the input pins are setup as A/D converter analog input channel or whether they have other functions. If the pin is setup to be as an A/D analog channel input, the original pin functions will be disabled. In this way, pins can be changed under program control to change their function between A/D inputs and other functions. All pull high resistors, which are setup through register programming, will be automatically disconnected if the pins are setup as A/D inputs. Note that it is not necessary to first setup the A/D pin as an input in the port control register



to enable the A/D input as when the pin-shared function control bits enable an A/D input, the status of the port control register will be overridden.

There are some internal analog signals derived from the proximity sensing circuit OPAMP outputs, A/D converter power supply or A/D converter OPA output, which can be connected to the A/D converter as the analog input signal by configuring the SAINS2~SAINS0 bits. If the external channel input is selected to be converted, the SAINS2~SAINS0 bits should be set to “000”, “110” or “111” and the SACS3~SACS0 bits can determine which external channel is selected. If the internal analog signal is selected to be converted, the SACS3~SACS0 bits must be configured with an appropriate value to switch off the external analog channel input. Otherwise, the internal analog signal will be connected together with the external channel input. This will result in unpredictable situations.

| SAINS[2:0]    | SACS[3:0] | Input Signals                              | Description  |
|---------------|-----------|--|--|
| 000, 110, 111 | 0000~0111 | AN0~AN7                                    | External pin analog input  |
|               | 1000~1111 | —  | Non-existed channel, input is floating   |
| 001           | 1000~1111 | OPDA00                                     | Proximity Sensing Circuit OPAMP0 output voltage  |
| 010           | 1000~1111 | OPDA10                                     | Proximity Sensing Circuit OPAMP1 output voltage  |
| 011           | 1000~1111 | AV <sub>SS</sub>                           | Connected to ground  |
| 100           | 1000~1111 | AV <sub>DD</sub> /4 or V <sub>BAT</sub> /4 | A/D converter power supply voltage AV <sub>DD</sub> /4 or external input V <sub>BAT</sub> /4 |
| 101           | 1000~1111 | V <sub>R</sub>                             | A/D converter OPA output voltage V <sub>R</sub>  |

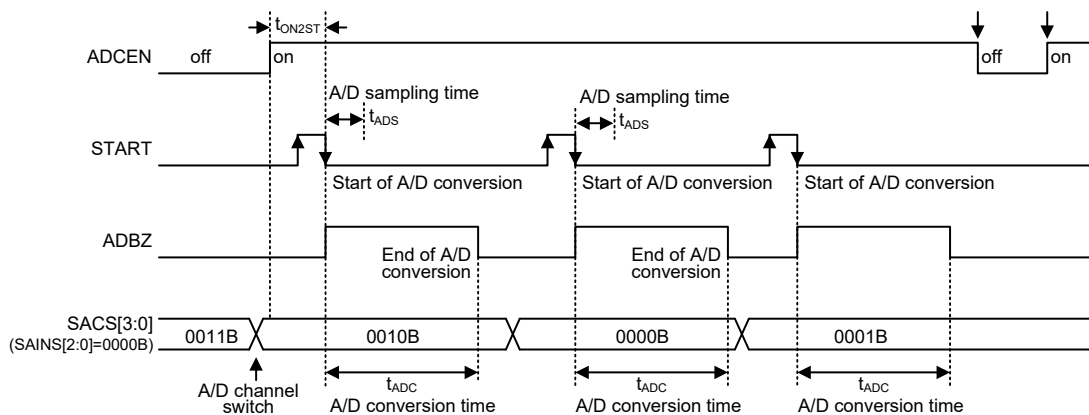
**A/D Converter Input Signal Selection**

### Conversion Rate and Timing Diagram

A complete A/D conversion contains two parts, data sampling and data conversion. The data sampling which is defined as  $t_{ADS}$  takes 4 A/D clock periods and the data conversion takes 12 A/D clock periods. Therefore, a total of 16 A/D clock periods for an external input A/D conversion which is defined as  $t_{ADC}$  are necessary.

$$\text{Maximum single A/D conversion rate} = 1 \div (\text{A/D clock period} \times 16)$$

The accompanying diagram shows graphically the various stages involved in an analog to digital conversion process and its associated timing. After an A/D conversion process has been initiated by the application program, the microcontroller internal hardware will begin to carry out the conversion, during which time the program can continue with other functions. The time taken for the A/D conversion is  $16 t_{ADCK}$  where  $t_{ADCK}$  is equal to the A/D clock period.



**A/D Conversion Timing – External Channel Input**

## Summary of A/D Conversion Steps

The following summarises the individual steps that should be executed in order to implement an A/D conversion process.

- Step 1  
Select the required A/D conversion clock by correctly programming bits SACKS2~SACKS0 in the SADC1 register.
- Step 2  
Enable the A/D converter by setting the ADCEN bit in the SADC0 register to 1.
- Step 3  
Select which signal is to be connected to the internal A/D converter by correctly configuring the SAINS2~SAINS0 bits.  
Select the external channel input to be converted, go to Step 4.  
Select the internal analog signal to be converted, go to Step 5.
- Step 4  
If the A/D input signal comes from the external channel input selected by configuring the SAINS2~SAINS0 bit field, the corresponding pins should be configured as A/D input function by configuring the relevant pin-shared function control bits. The desired analog channel then should be selected by configuring the SACS3~SACS0 bit field. After this step, go to Step 6.
- Step 5  
Before the A/D input signal is selected to come from the internal analog signal by configuring the SAINS2~SAINS0 bit field, the corresponding external input must be switched to a non-existent channel input by properly configured the SACS3~SACS0 bits. The desired internal analog signal then can be selected by configuring the SAINS2~SAINS0 bit field. After this step, go to Step 6.
- Step 6  
Select the reference voltage source by configuring the SAVRS1~SAVRS0 bits in the SADC1 register. Care should be taken in this step which can refer to the A/D Converter Reference Voltage section for details.
- Step 7  
Select A/D converter output data format by setting the ADRFS bit in the SADC0 register.
- Step 8  
If A/D conversion interrupt is used, the interrupt control registers must be correctly configured to ensure the A/D interrupt function is active. The master interrupt control bit, EMI, and the A/D conversion interrupt control bit, ADE, must both be set high in advance.
- Step 9  
The A/D conversion procedure can now be initialized by setting the START bit from low to high and then low again.
- Step 10  
If A/D conversion is in progress, the ADBZ flag will be set high. After the A/D conversion process is complete, the ADBZ flag will go low and then the output data can be read from SADOH and SADOL registers.

Note: When checking for the end of the conversion process, if the method of polling the ADBZ bit in the SADC0 register is used, the interrupt enable step above can be omitted.

### Programming Considerations

During microcontroller operations where the A/D converter is not being used, the A/D internal circuitry can be switched off to reduce power consumption, by clearing bit ADCEN to 0 in the SADC0 register. When this happens, the internal A/D converter circuits will not consume power irrespective of what analog voltage is applied to their input lines. If the A/D converter input lines are used as normal I/O pins, then care must be taken as if the input voltage is not at a valid logic level, then this may lead to some increase in power consumption.

### A/D Conversion Function

As the device contains a 12-bit A/D converter, its full-scale converted digitised value is equal to FFFH. Since the full-scale analog input value is equal to the actual A/D converter reference voltage,  $V_{REF}$ , this gives a single bit analog input value of  $V_{REF}$  divided by 4096.

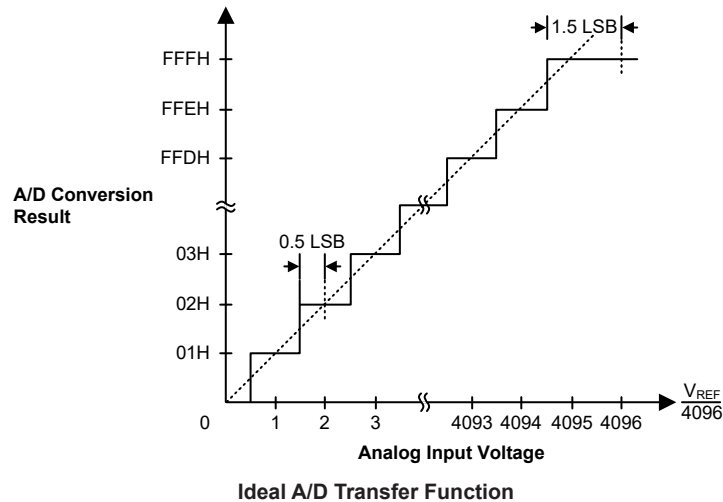
$$1 \text{ LSB} = V_{REF} \div 4096$$

The A/D Converter input voltage value can be calculated using the following equation:

$$\text{A/D input voltage} = \text{A/D output digital value} \times (V_{REF} \div 4096)$$

The diagram shows the ideal transfer function between the analog input value and the digitised output value for the A/D converter. Except for the digitised zero value, the subsequent digitised values will change at a point 0.5 LSB below where they would change without the offset, and the last full scale digitised value will change at a point 1.5 LSB below the  $V_{REF}$  level.

Note that here the  $V_{REF}$  voltage is the actual A/D converter reference voltage determined by the SAVRS field.



### Temperature Sensor Function

The A/D converter integrates a temperature sensor circuit. The temperature sensor output signal is amplified by the OPA and input to the A/D converter for conversion, then the current ambient temperature value can be obtained using a specific formula. According to the actual  $V_{DD}$  operating range, users should properly setup the TS\_SEL bit and select the corresponding  $V_R$ .

- Step 1  
 Select  $V_R$  as the A/D converter input (SAINS[2:0]=101b) and  $AV_{DD}$  as the A/D converter reference voltage (SAVRS[1:0]=11b).

- Step 2  
Enable the temperature sensor (TSEN=1). A time of  $t_{TSS}$  should be allowed for temperature sensor to stabilise. Select  $V_R$  (1.6V or 3.0V) to be converted.
- Step 3  
Start A/D conversion and read the converted value. It is recommended to read several times and take the average value, then record it as  $ADC_{REF}$ .  
Note: If  $AV_{DD}$  is fixed, there is no need to repeat this step everytime the temperature is measured if this step has been executed once and the  $ADC_{REF}$  value has been recorded.
- Step 4  
Select  $V_R=V_{TS}$  (PGAGS[1:0]=00), wait for a stable time of 150 $\mu$ s.
- Step 5  
Start A/D conversion and read the converted value. It is recommended to read several times and take the average value, then record it as  $ADC_{TS}$ .
- Step 6  
The measured temperature value is obtained using the following formula:

$$T_a(^{\circ}C) = 655 \times \frac{ADC_{TS}}{ADC_{REF}} - 460 + T_{OS}$$

Where the  $T_{OS}$  value has been previously calibrated by writer and stored in the Option Memory.

- Step 7  
If it is required to measure the temperature again, restart from Step 2; wait for a time of 150 $\mu$ s.
- Note: 1. Users should follow the above steps to measure  $T_a$  under a known temperature condition. The measured  $T_a$  value minus the known temperature value is a new  $T_{OS}$  value. Then store it in the Option Memory to correct the offset of the formula.  
2. The  $T_{OS}$  format stored in the Option Memory is 2's complement, 7 bits for the integer including the sign bit and 1 bit for the decimal.  
3. When using the writer for programming, a dedicated temperature module (EMDE001A) developed by Holtek must be externally connected to obtain the measurement data of TS,  $ADC_{TS}$  and  $T_{OS}$ , which are then stored in the Option Memory by writer. They can be read from the Program Memory last page using the table read instruction when the Option Memory mapping function is enabled.

| Name       | Mapped Address in Program Memory | Description   |
|------------|----------------------------------|---|
| TS         | FF5H                             | TS code (00H(0 $^{\circ}$ C)~FFH(51 $^{\circ}$ C))<br>Temperature value can be converted from the code with 0.2 $^{\circ}$ C/step |
| $ADC_{TS}$ | FF6H                             | 12-bit TS A/D converted value bit 11~bit 4  |
|            | FF7H                             | 12-bit TS A/D converted value bit 3~bit 0   |
| $T_{OS}$   | FF8H                             | Temperature sensor single point calibration compensation value  |

#### Temperature Measurement Reference Items

The Option Memory mapping function is enabled using the ORMC register. For more details, refer to the "Option Memory Mapping Register – ORMC" in the Special Function Register Description section.

### A/D Conversion Programming Examples

The following two programming examples illustrate how to setup and implement an A/D conversion. In the first example, the method of polling the ADBZ bit in the SADC0 register is used to detect when the conversion cycle is complete, whereas in the second example, the A/D interrupt is used to determine when the conversion is complete.

**Example: using an ADBZ polling method to detect the end of conversion**

```
clr ADE          ; disable ADC interrupt
mov a, 0BH
mov SADC1, a     ; select input signal from external channel input, reference
                 ; voltage from AVDD, fSYS/8 as A/D clock
mov a, 80h      ; setup PAS0 register to configure pin AN0
mov PAS0, a
mov a, 20h
mov SADC0, a     ; enable A/D converter and connect AN0 channel to A/D converter
:
start_conversion:
clr START       ; high pulse on start bit to initiate conversion
set START      ; reset A/D
clr START      ; start A/D
polling_EOC:
sz ADBZ        ; poll the SADC0 register ADBZ bit to detect end of A/D conversion
jmp polling_EOC ; continue polling
mov a, SADOL    ; read low byte conversion result value
mov SADOL_buffer, a ; save result to user defined register
mov a, SADOH    ; read high byte conversion result value
mov SADOH_buffer, a ; save result to user defined register
:
:
jmp start_conversion ; start next A/D conversion
```

**Example: using the interrupt method to detect the end of conversion**

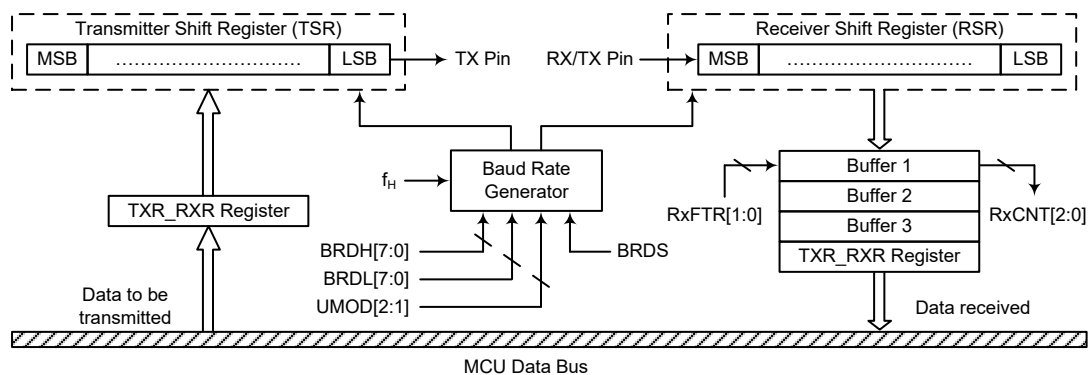
```
clr ADE          ; disable ADC interrupt
mov a, 0BH
mov SADC1, a     ; select input signal from external channel input, reference voltage
                 ; from AVDD, fSYS/8 as A/D clock
mov a, 80h      ; setup PAS0 register to configure pin AN0
mov PAS0, a
mov a, 20h
mov SADC0, a     ; enable A/D converter and connect AN0 channel to A/D converter
Start_conversion:
clr START       ; high pulse on START bit to initiate conversion
set START      ; reset A/D
clr START      ; start A/D
clr ADF        ; clear ADC interrupt request flag
set ADE        ; enable ADC interrupt
set EMI        ; enable global interrupt
:
:
; ADC interrupt service routine
ADC_ISR:
mov acc_stack, a ; save ACC to user defined memory
mov a, STATUS
mov status_stack, a ; save STATUS to user defined memory
:
:
mov a, SADOL    ; read low byte conversion result value
mov SADOL_buffer, a ; save result to user defined register
mov a, SADOH    ; read high byte conversion result value
mov SADOH_buffer, a ; save result to user defined register
:
:
EXIT_INT_ISR:
mov a, status_stack
mov STATUS, a ; restore STATUS from user defined memory
mov a, acc_stack ; restore ACC from user defined memory
reti
```

## UART Interface

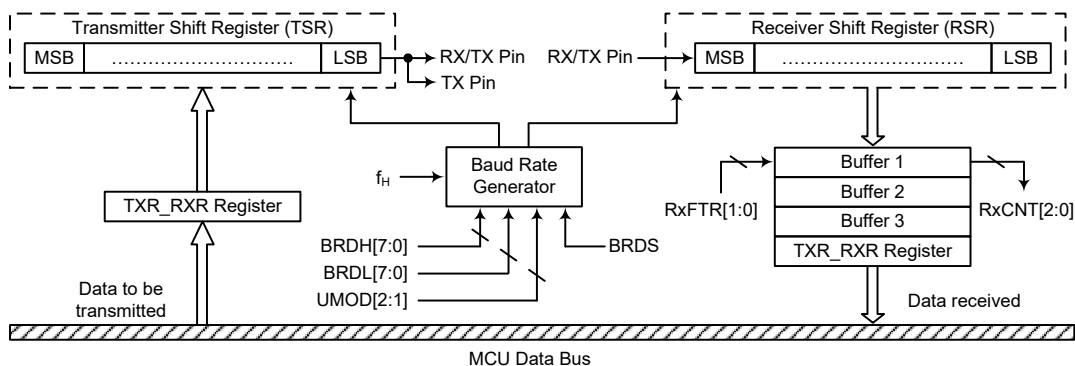
The device contains an integrated full-duplex or half-duplex asynchronous serial communications UART interface that enables communication with external devices that contain a serial interface. The UART function has many features and can transmit and receive data serially by transferring a frame of data with eight or nine data bits per transmission as well as being able to detect errors when the data is overwritten or incorrectly framed. The UART function possesses its own internal interrupt which can be used to indicate when a reception occurs or when a transmission terminates.

The integrated UART function contains the following features:

- Full-duplex or half-duplex (single wire mode) asynchronous communication
- 8 or 9 bits character length
- Even, odd, mark, space or no parity options
- One or two stop bits configurable for receiver
- Two stop bits for transmitter
- Baud rate generator with 16-bit prescaler
- Parity, framing, noise and overrun error detection
- Support for interrupt on address detect (last character bit=1)
- Separately enabled transmitter and receiver
- 4-byte Deep FIFO Receive Data Buffer
- 1-byte Deep FIFO Transmit Data Buffer
- RX/TX pin wake-up function
- Transmit and receive interrupts
- Interrupts can be triggered by the following conditions:
  - ♦ Transmitter Empty
  - ♦ Transmitter Idle
  - ♦ Receiver reaching FIFO trigger level
  - ♦ Receiver Overrun
  - ♦ Address Mode Detect



**UART Data Transfer Block Diagram – SWM=0**



**UART Data Transfer Block Diagram – SWM=1**

### UART External Pins

To communicate with an external serial interface, the internal UART has two external pins known as TX and RX/TX. The TX and RX/TX pins are the UART transmitter and receiver pins respectively. The TX and RX/TX pin function should first be selected by the corresponding pin-shared function selection register before the UART function is used. Along with the UARTEN bit, the TXEN and RXEN bits, if set, will setup these pins to their respective TX output and RX/TX input conditions and disable any pull-high resistor option which may exist on the TX and RX/TX pins. When the TX or RX/TX pin function is disabled by clearing the UARTEN, TXEN or RXEN bit, the TX or RX/TX pin will be set to a floating state. At this time whether the internal pull-high resistor is connected to the TX or RX/TX pin or not is determined by the corresponding I/O pull-high function control bit.

### UART Single Wire Mode

The UART function also supports the Single Wire Mode communication which is selected using the SWM bit in the UCR3 register. When the SWM bit is set high, the UART function will be in the single wire mode. In the single wire mode, a single RX/TX pin can be used to transmit and receive data depending upon the corresponding control bits. When the RXEN bit is set high, the RX/TX pin is used as a receiver pin. When the RXEN bit is cleared to zero and the TXEN bit is set high, the RX/TX pin will act as a transmitter pin.

It is recommended not to set both the RXEN and TXEN bits high in the single wire mode. If both the RXEN and TXEN bits are set high, the RXEN bit will have the priority and the UART will act as a receiver.

It is important to note that the functional description in this UART chapter, which is described from the full-duplex communication standpoint, also applies to the half-duplex (single wire mode) communication except the pin usage. In the single wire mode, the TX pin mentioned in this chapter should be replaced by the RX/TX pin to understand the whole UART single wire mode function.

In the single wire mode, the data can also be transmitted on the TX pin in a transmission operation with proper software configurations. Therefore, the data will be output on the RX/TX and TX pins.

### UART Data Transfer Scheme

The UART Data Transfer Block Diagrams show the overall data transfer structure arrangement for the UART. The actual data to be transmitted from the MCU is first transferred to the TXR\_RXR register by the application program. The data will then be transferred to the Transmit Shift Register from where it will be shifted out, LSB first, onto the TX pin at a rate controlled by the Baud Rate Generator. Only the TXR\_RXR register is mapped onto the MCU Data Memory, the Transmit Shift Register is not mapped and is therefore inaccessible to the application program.

Data to be received by the UART is accepted on the external RX/TX pin, from where it is shifted in, LSB first, to the Receiver Shift Register at a rate controlled by the Baud Rate Generator. When the shift register is full, the data will then be transferred from the shift register to the internal TXR\_RXR register, where it is buffered and can be manipulated by the application program. Only the TXR\_RXR register is mapped onto the MCU Data Memory, the Receiver Shift Register is not mapped and is therefore inaccessible to the application program.

It should be noted that the actual register for data transmission and reception only exists as a single shared register, TXR\_RXR, in the Data Memory.

### UART Status and Control Registers

There are nine control registers associated with the UART function. The USR, UCR1, UCR2, UCR3, UFCR and RxCNT registers control the overall function of the UART, while the BRDH and BRDL registers control the Baud rate. The actual data to be transmitted and received on the serial interface is managed through the TXR\_RXR data register.

| Register Name | Bit    |       |       |       |       |       |        |        |
|---------------|--------|-------|-------|-------|-------|-------|--------|--------|
|               | 7      | 6     | 5     | 4     | 3     | 2     | 1      | 0      |
| USR           | PERR   | NF    | FERR  | OERR  | RIDLE | RXIF  | TIDLE  | TXIF   |
| UCR1          | UARTEN | BNO   | PREN  | PRT1  | PRT0  | TXBRK | RX8    | TX8    |
| UCR2          | TXEN   | RXEN  | STOPS | ADDEN | WAKE  | RIE   | TIIE   | TEIE   |
| UCR3          | —      | —     | —     | —     | —     | —     | —      | SWM    |
| UFCR          | —      | —     | UMOD2 | UMOD1 | UMOD0 | BRDS  | RxFTR1 | RxFTR0 |
| RxCNT         | —      | —     | —     | —     | —     | D2    | D1     | D0     |
| TXR_RXR       | TXRX7  | TXRX6 | TXRX5 | TXRX4 | TXRX3 | TXRX2 | TXRX1  | TXRX0  |
| BRDH          | D7     | D6    | D5    | D4    | D3    | D2    | D1     | D0     |
| BRDL          | D7     | D6    | D5    | D4    | D3    | D2    | D1     | D0     |

**UART Register List**

#### • USR Register

The USR register is the status register for the UART, which can be read by the program to determine the present status of the UART. All flags within the USR register are read only. Further explanation on each of the flags is given below:

| Bit  | 7    | 6  | 5    | 4    | 3     | 2    | 1     | 0    |
|------|------|----|------|------|-------|------|-------|------|
| Name | PERR | NF | FERR | OERR | RIDLE | RXIF | TIDLE | TXIF |
| R/W  | R    | R  | R    | R    | R     | R    | R     | R    |
| POR  | 0    | 0  | 0    | 0    | 1     | 0    | 1     | 1    |

Bit 7 **PERR**: Parity error flag  
 0: No parity error is detected  
 1: Parity error is detected

The PERR flag is the parity error flag. When this read only flag is “0”, it indicates a parity error has not been detected. When the flag is “1”, it indicates that the parity of the received word is incorrect. This error flag is applicable only if the parity is enabled and the parity type (odd, even, mark or space) is selected. The flag can also be cleared to zero by a software sequence which involves a read to the status register USR followed by an access to the TXR\_RXR data register.

Bit 6 **NF**: Noise flag  
 0: No noise is detected  
 1: Noise is detected

The NF flag is the noise flag. When this read only flag is “0”, it indicates no noise condition. When the flag is “1”, it indicates that the UART has detected noise on the



receiver input. The NF flag is set during the same cycle as the RXIF flag but will not be set in the case of an overrun. The NF flag can be cleared to zero by a software sequence which will involve a read to the status register USR followed by an access to the TXR\_RXR data register.

- Bit 5     **FERR:** Framing error flag  
          0: No framing error is detected  
          1: Framing error is detected

The FERR flag is the framing error flag. When this read only flag is “0”, it indicates that there is no framing error. When the flag is “1”, it indicates that a framing error has been detected for the current character. The flag can also be cleared to zero by a software sequence which will involve a read to the status register USR followed by an access to the TXR\_RXR data register.

- Bit 4     **OERR:** Overrun error flag  
          0: No overrun error is detected  
          1: Overrun error is detected

The OERR flag is the overrun error flag which indicates when the receiver buffer has overflowed. When this read only flag is “0”, it indicates that there is no overrun error. When the flag is “1”, it indicates that an overrun error occurs which will inhibit further transfers to the TXR\_RXR receive data register. The flag is cleared to zero by a software sequence, which is a read to the status register USR followed by an access to the TXR\_RXR data register.

- Bit 3     **RIDLE:** Receiver status  
          0: Data reception is in progress (Data being received)  
          1: No data reception is in progress (Receiver is idle)

The RIDLE flag is the receiver status flag. When this read only flag is “0”, it indicates that the receiver is between the initial detection of the start bit and the completion of the stop bit. When the flag is “1”, it indicates that the receiver is idle. Between the completion of the stop bit and the detection of the next start bit, the RIDLE bit is “1” indicating that the UART receiver is idle and the RX/TX pin stays in logic high condition.

- Bit 2     **RXIF:** Receive TXR\_RXR data register status  
          0: TXR\_RXR data register is empty  
          1: TXR\_RXR data register has available data and reaches receiver FIFO trigger level

The RXIF flag is the receive data register status flag. When this read only flag is “0”, it indicates that the TXR\_RXR read data register is empty. When the flag is “1”, it indicates that the TXR\_RXR read data register contains new data and reaches the Receiver FIFO trigger level. When the contents of the shift register are transferred to the TXR\_RXR register, and reach receiver FIFO trigger level, an interrupt is generated if RIE=1 in the UCR2 register. If one or more errors are detected in the received word, the appropriate receive-related flags NF, FERR, and/or PERR are set within the same clock cycle. The RXIF flag will eventually be cleared to zero when the USR register is read with RXIF set, followed by a read from the TXR\_RXR register, and if the TXR\_RXR register has no more new data available.

- Bit 1     **TIDLE:** Transmission idle  
          0: Data transmission is in progress (Data being transmitted)  
          1: No data transmission is in progress (Transmitter is idle)

The TIDLE flag is known as the transmission complete flag. When this read only flag is “0”, it indicates that a transmission is in progress. This flag will be set high when the TXIF flag is “1” and when there is no transmit data or break character being transmitted. When TIDLE is equal to “1”, the TX pin becomes idle with the pin state in logic high condition. The TIDLE flag is cleared to zero by reading the USR register with TIDLE set and then writing to the TXR\_RXR register. The flag is not generated when a data character or a break is queued and ready to be sent.

Bit 0      **TXIF**: Transmit TXR\_RXR data register status  
             0: Character is not transferred to the transmit shift register  
             1: Character has transferred to the transmit shift register (TXR\_RXR data register is empty)

The TXIF flag is the transmit data register empty flag. When this read only flag is “0”, it indicates that the character is not transferred to the transmitter shift register. When the flag is “1”, it indicates that the transmitter shift register has received a character from the TXR\_RXR data register. The TXIF flag is cleared to zero by reading the UART status register (USR) with TXIF set and then writing to the TXR\_RXR data register. Note that when the TXEN bit is set, the TXIF flag bit will also be set since the transmit data register is not yet full.

• **UCR1 Register**

The UCR1 register together with the UCR2 register are the two UART control registers that are used to set the various options for the UART function, such as overall on/off control, parity control, data transfer bit length, etc. Further explanation on each of the bits is given below:

| Bit  | 7      | 6   | 5    | 4    | 3    | 2     | 1   | 0   |
|------|--------|-----|------|------|------|-------|-----|-----|
| Name | UARTEN | BNO | PREN | PRT1 | PRT0 | TXBRK | RX8 | TX8 |
| R/W  | R/W    | R/W | R/W  | R/W  | R/W  | R/W   | R   | W   |
| POR  | 0      | 0   | 0    | 0    | 0    | 0     | x   | 0   |

“x”: unknown

Bit 7      **UARTEN**: UART function enable control  
             0: Disable UART. TX and RX/TX pins are in a floating state  
             1: Enable UART. TX and RX/TX pins can function as UART pins

The UARTEN bit is the UART enable bit. When this bit is equal to “0”, the UART will be disabled and the RX/TX pin as well as the TX pin will be in a floating state. When the bit is equal to “1”, the UART will be enabled and the TX and RX/TX pins will function as defined by the SWM mode selection bit together with the TXEN and RXEN enable control bits.

When the UART is disabled, it will empty the buffer so any character remaining in the buffer will be discarded. In addition, the value of the baud rate counter will be reset. If the UART is disabled, all error and status flags will be reset. Also the TXEN, RXEN, TXBRK, RXIF, OERR, FERR, PERR and NF bits as well as the RxCNT register will be cleared to zero, while the TIDLE, TXIF and RIDLE bits will be set high. Other control bits in UCR1, UCR2, UCR3, UFCR, BRDH and BRDL registers will remain unaffected. If the UART is active and the UARTEN bit is cleared to zero, all pending transmissions and receptions will be terminated and the module will be reset as defined above. When the UART is re-enabled, it will restart in the same configuration.

Bit 6      **BNO**: Number of data transfer bits selection  
             0: 8-bit data transfer  
             1: 9-bit data transfer

This bit is used to select the data length format, which can have a choice of either 8-bit or 9-bit format. When this bit is equal to “1”, a 9-bit data length format will be selected. If the bit is equal to “0”, then an 8-bit data length format will be selected. If 9-bit data length format is selected, then bits RX8 and TX8 will be used to store the 9th bit of the received and transmitted data respectively.

Note that the 9th bit of data if BNO=1, or the 8th bit of data if BNO=0, which is used as the parity bit, does not transfer to RX8 or TXRX7 (i.e. TXR\_RXR.7) respectively when the parity function is enabled.

Bit 5      **PREN**: Parity function enable control  
             0: Parity function is disabled  
             1: Parity function is enabled

This is the parity enable bit. When this bit is equal to “1”, the parity function will be enabled. If the bit is equal to “0”, then the parity function will be disabled.

- Bit 4~3 **PRT1~PRT0**: Parity type selection bits  
 00: Even parity for parity generator  
 01: Odd parity for parity generator  
 10: Mark parity for parity generator  
 11: Space parity for parity generator  
 These bits are the parity type selection bits. When these bits are equal to 00b, even parity type will be selected. If these bits are equal to 01b, then odd parity type will be selected. If these bits are equal to 10b, then a 1 (Mark) in the parity bit location will be selected. If these bits are equal to 11b, then a 0 (Space) in the parity bit location will be selected.
- Bit 2 **TXBRK**: Transmit break character  
 0: No break character is transmitted  
 1: Break characters transmit  
 The TXBRK bit is the Transmit Break Character bit. When this bit is “0”, there are no break characters and the TX pin operates normally. When the bit is “1”, there are transmit break characters and the transmitter will send logic zeros. When this bit is equal to “1”, after the buffered data has been transmitted, the transmitter output is held low for a minimum of a 13-bit length and until the TXBRK bit is reset.
- Bit 1 **RX8**: Receive data bit 8 for 9-bit data transfer format (read only)  
 This bit is only used if 9-bit data transfers are used, in which case this bit location will store the 9th bit of the received data known as RX8. The BNO bit is used to determine whether data transfers are in 8-bit or 9-bit format.
- Bit 0 **TX8**: Transmit data bit 8 for 9-bit data transfer format (write only)  
 This bit is only used if 9-bit data transfers are used, in which case this bit location will store the 9th bit of the transmitted data known as TX8. The BNO bit is used to determine whether data transfers are in 8-bit or 9-bit format.

• **UCR2 Register**

The UCR2 register is the second of the UART control registers and serves several purposes. One of its main functions is to control the basic enable/disable operation of the UART Transmitter and Receiver as well as enabling the various UART interrupt sources. The register also serves to control the receiver STOP bit number selection, receiver wake-up enable and the address detect enable. Further explanation on each of the bits is given below:

| Bit  | 7    | 6    | 5     | 4     | 3    | 2   | 1    | 0    |
|------|------|------|-------|-------|------|-----|------|------|
| Name | TXEN | RXEN | STOPS | ADDEN | WAKE | RIE | TIIE | TEIE |
| R/W  | R/W  | R/W  | R/W   | R/W   | R/W  | R/W | R/W  | R/W  |
| POR  | 0    | 0    | 0     | 0     | 0    | 0   | 0    | 0    |

- Bit 7 **TXEN**: UART Transmitter enabled control  
 0: UART transmitter is disabled  
 1: UART transmitter is enabled  
 The bit named TXEN is the Transmitter Enable Bit. When this bit is equal to “0”, the transmitter will be disabled with any pending data transmissions being aborted. In addition, the buffers will be reset. In this situation the TX pin will be in a floating state. If the TXEN bit is equal to “1” and the UARTEN bit is also equal to “1”, the transmitter will be enabled and the TX pin will be controlled by the UART. Clearing the TXEN bit during a transmission will cause the data transmission to be aborted and will reset the transmitter. If this situation occurs, the TX pin will be in a floating state.
- Bit 6 **RXEN**: UART Receiver enabled control  
 0: UART receiver is disabled  
 1: UART receiver is enabled  
 The bit named RXEN is the Receiver Enable Bit. When this bit is equal to “0”, the receiver will be disabled with any pending data receptions being aborted. In addition, the receive buffers will be reset. In this situation the RX/TX pin will be in a floating

state. If the RXEN bit is equal to “1” and the UARTEN bit is also equal to “1”, the receiver will be enabled and the RX/TX pin will be controlled by the UART. Clearing the RXEN bit during a reception will cause the data reception to be aborted and will reset the receiver. If this situation occurs, the RX/TX pin will be in a floating state.

- Bit 5     **STOPS:** Number of Stop bits selection for receiver  
           0: One stop bit format is used  
           1: Two stop bits format is used

This bit determines if one or two stop bits are to be used for receiver. When this bit is equal to “1”, two stop bits are used. If this bit is equal to “0”, then only one stop bit is used. Two stop bits are used for transmitter.

- Bit 4     **ADDEN:** Address detect function enable control  
           0: Address detect function is disabled  
           1: Address detect function is enabled

The bit named ADDEN is the address detect function enable control bit. When this bit is equal to “1”, the address detect function is enabled. When it occurs, if the 8th bit, which corresponds to TXRX7 if BNO=0 or the 9th bit, which corresponds to RX8 if BNO=1, has a value of “1”, then the received word will be identified as an address, rather than data. If the corresponding interrupt is enabled, an interrupt request will be generated each time the received word has the address bit set, which is the 8th or 9th bit depending on the value of BNO. If the address bit known as the 8th or 9th bit of the received word is “0” with the address detect function being enabled, an interrupt will not be generated and the received data will be discarded.

- Bit 3     **WAKE:** RX/TX pin wake-up UART function enable control  
           0: RX/TX pin wake-up UART function is disabled  
           1: RX/TX pin wake-up UART function is enabled

This bit is used to control the wake-up UART function when a falling edge on the RX/TX pin occurs. Note that this bit is only available when the UART clock ( $f_{H1}$ ) is switched off. There will be no RX/TX pin wake-up UART function if the UART clock ( $f_{H1}$ ) exists. If the WAKE bit is set to 1 as the UART clock ( $f_{H1}$ ) is switched off, a UART wake-up request will be initiated when a falling edge on the RX/TX pin occurs. When this request happens and the corresponding interrupt is enabled, an RX/TX pin wake-up UART interrupt will be generated to inform the MCU to wake up the UART function by switching on the UART clock ( $f_{H1}$ ) via the application program. Otherwise, the UART function can not resume even if there is a falling edge on the RX/TX pin when the WAKE bit is cleared to 0.

- Bit 2     **RIE:** Receiver interrupt enable control  
           0: Receiver related interrupt is disabled  
           1: Receiver related interrupt is enabled

This bit enables or disables the receiver interrupt. If this bit is equal to “1” and when the receiver overrun flag OERR or receive data available flag RXIF is set, the UART interrupt request flag will be set. If this bit is equal to “0”, the UART interrupt request flag will not be influenced by the condition of the OERR or RXIF flags.

- Bit 1     **TIE:** Transmitter Idle interrupt enable control  
           0: Transmitter idle interrupt is disabled  
           1: Transmitter idle interrupt is enabled

This bit enables or disables the transmitter idle interrupt. If this bit is equal to “1” and when the transmitter idle flag TIDLE is set, due to a transmitter idle condition, the UART interrupt request flag will be set. If this bit is equal to “0”, the UART interrupt request flag will not be influenced by the condition of the TIDLE flag.

- Bit 0     **TEIE:** Transmitter Empty interrupt enable control  
           0: Transmitter empty interrupt is disabled  
           1: Transmitter empty interrupt is enabled

This bit enables or disables the transmitter empty interrupt. If this bit is equal to “1” and when the transmitter empty flag TXIF is set, due to a transmitter empty condition, the UART interrupt request flag will be set. If this bit is equal to “0”, the UART interrupt request flag will not be influenced by the condition of the TXIF flag.

• **UCR3 Register**

The UCR3 register is used to enable the UART Single Wire Mode communication. As the name suggests in the single wire mode the UART communication can be implemented in one single line, RX/TX, together with the control of the RXEN and TXEN bits in the UCR2 register.

| Bit  | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0   |
|------|---|---|---|---|---|---|---|-----|
| Name | — | — | — | — | — | — | — | SWM |
| R/W  | — | — | — | — | — | — | — | R/W |
| POR  | — | — | — | — | — | — | — | 0   |

Bit 7~1 Unimplemented, read as “0”

Bit 0 **SWM**: Single Wire Mode enable control  
 0: Disable, the RX/TX pin is used as UART receiver function only  
 1: Enable, the RX/TX pin can be used as UART receiver or transmitter function controlled by the RXEN and TXEN bits

Note that when the Single Wire Mode is enabled, if both the RXEN and TXEN bits are high, the RX/TX pin will only be used as UART receiver input.

• **TXR\_RXR Register**

The TXR\_RXR register is the data register which is used to store the data to be transmitted on the TX pin or being received from the RX/TX pin.

| Bit  | 7     | 6     | 5     | 4     | 3     | 2     | 1     | 0     |
|------|-------|-------|-------|-------|-------|-------|-------|-------|
| Name | TXRX7 | TXRX6 | TXRX5 | TXRX4 | TXRX3 | TXRX2 | TXRX1 | TXRX0 |
| R/W  | R/W   | R/W   | R/W   | R/W   | R/W   | R/W   | R/W   | R/W   |
| POR  | x     | x     | x     | x     | x     | x     | x     | x     |

“x”: unknown

Bit 7~0 **TXRX7~TXRX0**: UART Transmit/Receive Data bit 7~bit 0

• **BRDH Register**

| Bit  | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   |
|------|-----|-----|-----|-----|-----|-----|-----|-----|
| Name | D7  | D6  | D5  | D4  | D3  | D2  | D1  | D0  |
| R/W  | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| POR  | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |

Bit 7~0 **D7~D0**: Baud rate divider high byte  
 The baud rate divider BRD (BRDH/BRDL) defines the UART clock divider ratio.  
 $Baud\ Rate = f_{it} / (BRD + UMOD / 8)$   
 $BRD = 16 \sim 65535$  or  $8 \sim 65535$  depending on BRDS  
 Note: 1. BRD value should not be set to less than 16 when BRDS=0 or less than 8 when BRDS=1, otherwise errors may occur.  
 2. The BRDL must be written first and then BRDH, otherwise errors may occur.  
 3. The BRDH register should not be modified during data transmission process.

• **BRDL Register**

| Bit  | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   |
|------|-----|-----|-----|-----|-----|-----|-----|-----|
| Name | D7  | D6  | D5  | D4  | D3  | D2  | D1  | D0  |
| R/W  | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| POR  | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |

Bit 7~0 **D7~D0**: Baud rate divider low byte  
 The baud rate divider BRD (BRDH/BRDL) defines the UART clock divider ratio.  
 $Baud\ Rate = f_{it} / (BRD + UMOD / 8)$

BRD=16~65535 or 8~65535 depending on BRDS

Note: 1. BRD value should not be set to less than 16 when BRDS=0 or less than 8 when BRDS=1, otherwise errors may occur.

2. The BRDL must be written first and then BRDH, otherwise errors may occur.

3. The BRDL register should not be modified during data transmission process.

• **UFCR Register**

The UFCR register is the FIFO control register which is used for UART modulation control, BRD range selection and trigger level selection for RXIF and interrupt.

| Bit  | 7 | 6 | 5     | 4     | 3     | 2    | 1      | 0      |
|------|---|---|-------|-------|-------|------|--------|--------|
| Name | — | — | UMOD2 | UMOD1 | UMOD0 | BRDS | RxFTR1 | RxFTR0 |
| R/W  | — | — | R/W   | R/W   | R/W   | R/W  | R/W    | R/W    |
| POR  | — | — | 0     | 0     | 0     | 0    | 0      | 0      |

Bit 7~6 Unimplemented, read as “0”

Bit 5~3 **UMOD2~UMOD0**: UART Modulation Control bits

The modulation control bits are used to correct the baud rate of the received or transmitted UART signal. These bits determine if the extra UART clock cycle should be added in a UART bit time. The UMOD2~UMOD0 value will be added to internal accumulator for every UART bit time. Until a carry to bit 3, the corresponding UART bit time increases a UART clock cycle.

Bit 2 **BRDS**: BRD range selection

0: BRD range is from 16 to 65535

1: BRD range is from 8 to 65535

The BRDS bit is used to control the sampling point in a UART bit time. If the BRDS bit is cleared to zero, the sampling point will be  $BRD/2$ ,  $BRD/2+1 \times f_{IH}$ , and  $BRD/2+2 \times f_{IH}$  in a UART bit time. If the BRDS bit is set high, the sampling point will be  $BRD/2-1 \times f_{IH}$ ,  $BRD/2$ , and  $BRD/2+2 \times f_{IH}$  in a UART bit time.

Note that the BRDS bit should not be modified during data transmission process.

Bit 1~0 **RxFTR1~RxFTR0**: Receiver FIFO trigger level (bytes)

00: 4 bytes in RX FIFO

01: 1 or more bytes in RX FIFO

10: 2 or more bytes in RX FIFO

11: 3 or more bytes in RX FIFO

For the receiver these bits define the number of received data bytes in the RX FIFO that will trigger the RXIF bit being set high, an interrupt will also be generated if the RIE bit is enabled. To prevent OERR from being set high, the receiver FIFO trigger level can be set to 2 bytes, avoiding an overrun state that cannot be processed by the program in time when more than 4 data bytes are received. After the reset the receiver FIFO is empty.

• **RxCNT Register**

The RxCNT register is the counter used to indicate the number of received data bytes in the RX FIFO which have not been read by the MCU. This register is read only.

| Bit  | 7 | 6 | 5 | 4 | 3 | 2  | 1  | 0  |
|------|---|---|---|---|---|----|----|----|
| Name | — | — | — | — | — | D2 | D1 | D0 |
| R/W  | — | — | — | — | — | R  | R  | R  |
| POR  | — | — | — | — | — | 0  | 0  | 0  |

Bit 7~3 Unimplemented, read as “0”

Bit 2~0 **D2~D0**: RX FIFO counter

The RxCNT register is the counter used to indicate the number of received data bytes in the RX FIFO which is not read by the MCU. When RX FIFO receives one byte of data, the RxCNT will increase by one; when the MCU reads one byte of data from RX

FIFO, the RxCNTn will decrease by one. If there are 4 bytes of data in the RX FIFO, the 5th data will be saved in the shift register. If there is 6th data, the 6th data will be saved in the shift register. But the RxCNT remains the value of 4. The RxCNT will be cleared when reset occurs or UARTEN=1. This register is read only.

### Baud Rate Generator

To setup the speed of the serial data communication, the UART function contains its own dedicated baud rate generator. The baud rate is controlled by its own internal free running 16-bit timer, the period of which is determined by two factors. The first of these is the value placed in the BRDH and BRDL baud rate registers and the second is the UART modulation control bits UMOD2~UMOD0. To prevent accumulated error of the receiver baud rate frequency, it is recommended to use two stop bits for resynchronization after each byte is received. If a baud rate BR is required with UART clock  $f_H$ .

$$f_H/BR = \text{Integer Part} + \text{Fractional Part}$$

The integer part is loaded into BRD (BRDH/BRDL). The fractional part is multiplied by 8 and rounded, then loaded into UMOD bit field as following:

$$BRD = \text{TRUNC}(f_H/BR)$$

$$UMOD = \text{ROUND}[\text{MOD}(f_H/BR) \times 8]$$

Therefore, the actual baud rate is as following:

$$\text{Baud Rate} = f_H / [BRD + (UMOD/8)]$$

### Calculating the Baud Rate and Error Values

For a clock frequency of 4MHz, determine the BRDH/BRDL register value, the actual baud rate and the error value for a desired baud rate of 230400.

$$\text{From the above formula, the } BRD = \text{TRUNC}(f_H/BR) = \text{TRUNC}(17.36111) = 17$$

$$\text{The } UMOD = \text{ROUND}[\text{MOD}(f_H/BR) \times 8] = \text{ROUND}(0.36111 \times 8) = \text{ROUND}(2.88888) = 3$$

$$\text{The actual Baud Rate} = f_H / [BRD + (UMOD/8)] = 230215.83$$

Therefore, the error is equal to  $(230215.83 - 230400) / 230400 = -0.08\%$

### Modulation Control Example

To get the best-fitting bit sequence for UART modulation control bits UMOD2~UMOD0, the following algorithm can be used: Firstly, the fractional part of the theoretical division factor is multiplied by 8. Then the product will be rounded and the UMOD2~UMOD0 bits will be filled with the rounded value. The UMOD2~UMOD0 value will be added to internal accumulator for every UART bit time. Until a carry to bit 3, the corresponding UART bit time increases by a UART clock cycle. The following is an example using the fraction 0.36111 previously calculated:  $UMOD[2:0] = \text{ROUND}(0.36111 \times 8) = 011b$

| Fraction Addition | Carry to Bit 3 | UART Bit Time Sequence | Extra UART Clock Cycle |
|-------------------|----------------|------------------------|------------------------|
| 0000b+0011b=0011b | No             | Start bit              | No                     |
| 0011b+0011b=0110b | No             | D0                     | No                     |
| 0110b+0011b=1001b | Yes            | D1                     | Yes                    |
| 1001b+0011b=1100b | No             | D2                     | No                     |
| 1100b+0011b=1111b | No             | D3                     | No                     |
| 1111b+0011b=0010b | Yes            | D4                     | Yes                    |
| 0010b+0011b=0101b | No             | D5                     | No                     |
| 0101b+0011b=1000b | Yes            | D6                     | Yes                    |
| 1000b+0011b=1011b | No             | D7                     | No                     |
| 1011b+0011b=1110b | No             | Parity bit             | No                     |
| 1110b+0011b=0001b | Yes            | Stop bit               | Yes                    |

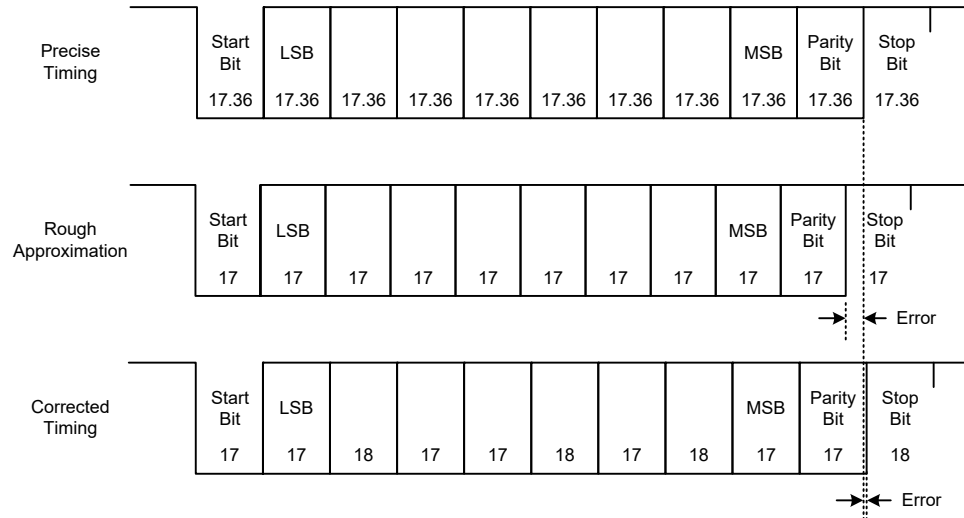


### Baud Rate Correction Example

The following figure presents an example using a baud rate of 230400 generated with UART clock  $f_H$ . The data format for the following figure is: eight data bits, parity enabled, no address bit, two stop bits.

The following figure shows three different frames:

- The upper frame is the correct one, with a bit length of 17.36  $f_H$  cycles ( $4000000/230400=17.36$ ).
- The middle frame uses a rough estimate, with 17  $f_H$  cycles for the bit length.
- The lower frame shows a corrected frame using the best fit for the UART modulation control bits UMOD2~UMOD0.



### UART Setup and Control

For data transfer, the UART function utilizes a non-return-to-zero, more commonly known as NRZ, format. This is composed of one start bit, eight or nine data bits, and one or two stop bits. Parity is supported by the UART hardware, and can be setup to be even, odd, mark, space or no parity. For the most common data format, 8 data bits along with no parity and one stop bit, denoted as 8, N, 1, is used as the default setting, which is the setting at power-on. The number of data bits along with the parity are setup by programming the BNO, PRT1~PRT0 and PREN bits. The transmitter always uses two stop bits while the receiver uses one or two stop bits which is determined by the STOPS bit.

The baud rate used to transmit and receive data is setup using the internal 16-bit baud rate generator, while the data is transmitted and received LSB first. Although the UART transmitter and receiver are functionally independent, they both use the same data format and baud rate. In all cases stop bits will be used for data transmission.

#### Enabling/Disabling the UART Interface

The basic on/off function of the internal UART function is controlled using the UARTEN bit in the UCR1 register. If the UARTEN, TXEN and RXEN bits are set, then these two UART pins will act as normal TX output pin and RX/TX input pin respectively. If no data is being transmitted on the TX pin, then it will default to a logic high value.

Clearing the UARTEN bit will disable the TX and RX/TX pins and allow these two pins to be used as normal I/O or other pin-shared functional pins. When the UART function is disabled the buffer will be reset to an empty condition, at the same time discarding any remaining residual data. Disabling the UART will also reset the error and status flags with bits TXEN, RXEN, TXBRK,



RXIF, OERR, FERR, PERR and NF as well as register RxCNT being cleared while bits TIDLE, TXIF and RIDLE will be set. The remaining control bits in the UCR1, UCR2, UCR3, UFCR, BRDH and BRDL registers will remain unaffected. If the UARTEN bit in the UCR1 register is cleared while the UART is active, then all pending transmissions and receptions will be immediately suspended and the UART will be reset to a condition as defined above. If the UART is then subsequently re-enabled, it will restart again in the same configuration.

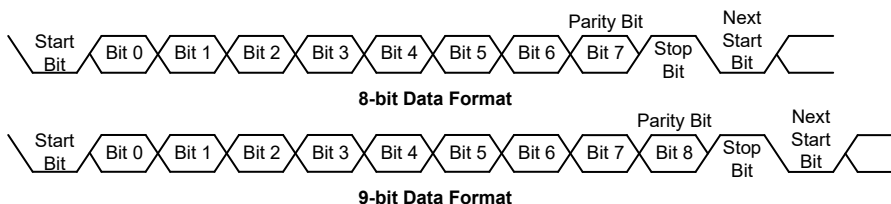
### Data, Parity and Stop Bit Selection

The format of the data to be transferred is composed of various factors such as data bit length, parity on/off, parity type, address bits and the number of stop bits. These factors are determined by the setup of various bits within the UCR1 and UCR2 registers. The BNO bit controls the number of data bits which can be set to either 8 or 9, the PRT1~PRT0 bits control the choice of odd, even, mark or space parity, the PREN bit controls the parity on/off function and the STOPS bit decides whether one or two stop bits are to be used for the receiver, while the transmitter always uses two stop bits. The following table shows various formats for data transmission. The address bit, which is the MSB of the data byte, identifies the frame as an address character or data if the address detect function is enabled. The number of stop bits, which can be either one or two, is independent of the data length and is only configurable for the receiver. The transmitter uses two stop bits.

| Start Bit                            | Data Bits | Address Bit | Parity Bit | Stop Bit |
|--------------------------------------|-----------|-------------|------------|----------|
| <b>Example of 8-bit Data Formats</b> |           |             |            |          |
| 1                                    | 8         | 0           | 0          | 1 or 2   |
| 1                                    | 7         | 0           | 1          | 1 or 2   |
| 1                                    | 7         | 1           | 0          | 1 or 2   |
| <b>Example of 9-bit Data Formats</b> |           |             |            |          |
| 1                                    | 9         | 0           | 0          | 1 or 2   |
| 1                                    | 8         | 0           | 1          | 1 or 2   |
| 1                                    | 8         | 1           | 0          | 1 or 2   |

**Transmitter Receiver Data Format**

The following diagram shows the transmit and receive waveforms for both 8-bit and 9-bit data formats.



### UART Transmitter

Data word lengths of either 8 or 9 bits can be selected by programming the BNO bit in the UCR1 register. When BNO bit is set, the word length will be set to 9 bits. In this case the 9th bit, which is the MSB, needs to be stored in the TX8 bit in the UCR1 register. At the transmitter core lies the Transmitter Shift Register, more commonly known as the TSR, whose data is obtained from the transmit data register, which is known as the TXR\_RXR register. The data to be transmitted is loaded into this TXR\_RXR register by the application program. The TSR register is not written to with new data until the stop bit from the previous transmission has been sent out. As soon as this stop bit has been transmitted, the TSR can then be loaded with new data from the TXR\_RXR register, if it is available. It should be noted that the TSR register, unlike many other registers, is not directly mapped into the Data Memory area and as such is not available to the application program for direct read/write operations. An actual transmission of data will normally be enabled when the

TXEN bit is set, but the data will not be transmitted until the TXR\_RXR register has been loaded with data and the baud rate generator has defined a shift clock source. However, the transmission can also be initiated by first loading data into the TXR\_RXR register, after which the TXEN bit can be set. When a transmission of data begins, the TSR is normally empty, in which case a transfer to the TXR\_RXR register will result in an immediate transfer to the TSR. If during a transmission the TXEN bit is cleared, the transmission will immediately cease and the transmitter will be reset. The TX output pin can then be configured as the I/O or other pin-shared function by configuring the corresponding pin-shared control bits.

### Transmitting Data

When the UART is transmitting data, the data is shifted on the TX pin from the shift register, with the least significant bit first. In the transmit mode, the TXR\_RXR register forms a buffer between the internal bus and the transmitter shift register. It should be noted that if 9-bit data format has been selected, then the MSB will be taken from the TX8 bit in the UCR1 register. The steps to initiate a data transfer can be summarized as follows:

- Make the correct selection of the BNO, PRT1~PRT0 and PREN bits to define the required word length and parity type. Two stop bits are used for the transmitter.
- Setup the BRDH and BRDL registers and the UMOD2~UMOD0 bits to select the desired baud rate.
- Set the TXEN bit to ensure that the TX pin is used as a UART transmitter pin.
- Access the USR register and write the data that is to be transmitted into the TXR\_RXR register. Note that this step will clear the TXIF bit.

This sequence of events can now be repeated to send additional data.

It should be noted that when TXIF=0, data will be inhibited from being written to the TXR\_RXR register. Clearing the TXIF flag is always achieved using the following software sequence:

1. A USR register access
2. A TXR\_RXR register write execution

The read-only TXIF flag is set by the UART hardware and if set indicates that the TXR\_RXR register is empty and that other data can now be written into the TXR\_RXR register without overwriting the previous data. If the TEIE bit is set, then the TXIF flag will generate an interrupt.

During a data transmission, a write instruction to the TXR\_RXR register will place the data into the TXR\_RXR register, which will be copied to the shift register at the end of the present transmission. When there is no data transmission in progress, a write instruction to the TXR\_RXR register will place the data directly into the shift register, resulting in the commencement of data transmission, and the TXIF bit being immediately set. When a frame transmission is complete, which happens after stop bits are sent or after the break frame, the TIDLE bit will be set. To clear the TIDLE bit the following software sequence is used:

1. A USR register access
2. A TXR\_RXR register write execution

Note that both the TXIF and TIDLE bits are cleared by the same software sequence.

### Transmitting Break

If the TXBRK bit is set high and the state keeps for a time of greater than  $[(BRD+1) \times t_{H}]$  while TIDLE=1, then break characters will be sent on the next transmission. Break character transmission consists of a start bit, followed by  $13 \times N$  '0' bits and stop bits, where  $N=1, 2, \text{etc.}$  If a break character is to be transmitted then the TXBRK bit must be first set by the application program, and

then cleared to generate the stop bits. Transmitting a break character will not generate a transmit interrupt. Note that a break condition length is at least 13 bits long. If the TXBRK bit is continually kept at a logic high level, then the transmitter circuitry will transmit continuous break characters. After the application program has cleared the TXBRK bit, the transmitter will finish transmitting the last break character and subsequently send out two stop bits. The automatic logic highs at the end of the last break character will ensure that the start bit of the next frame is recognized.

## UART Receiver

The UART is capable of receiving word lengths of either 8 or 9 bits. If the BNO bit is set, the word length will be set to 9 bits with the MSB being stored in the RX8 bit of the UCR1 register. At the receiver core lies the Receive Serial Shift Register, commonly known as the RSR. The data which is received on the RX/TX pin is sent to the data recovery block. The data recovery block operating speed is 16 times that of the baud rate, while the main receive serial shifter operates at the baud rate. After the RX/TX pin is sampled for the stop bit, the received data in RSR is transferred to the receive data register, if the register is empty. The data which is received on the external RX/TX pin is sampled three times by a majority detect circuit to determine the logic level that has been placed onto the RX/TX pin. It should be noted that the RSR register, unlike many other registers, is not directly mapped into the Data Memory area and as such is not available to the application program for direct read/write operations.

## Receiving Data

When the UART receiver is receiving data, the data is serially shifted in on the external RX/TX pin, LSB first. In the read mode, the TXR\_RXR register forms a buffer between the internal bus and the receiver shift register. The TXR\_RXR register is a four-byte deep FIFO data buffer, where four bytes can be held in the FIFO while a fifth byte can continue to be received. Note that the application program must ensure that the data is read from TXR\_RXR before the fifth byte has been completely shifted in, otherwise this fifth byte will be discarded and an overrun error OERR will be subsequently indicated. For continuous multi-byte data transmission, it is strongly recommended that the receiver uses two stop bits to avoid a receiving error caused by the accumulated error of the receiver baud rate frequency.

The steps to initiate a data transfer can be summarized as follows:

- Make the correct selection of BNO, PRT1~PRT0, PREN and STOPS bits to define the word length, parity type and number of stop bits.
- Setup the BRDH and BRDL registers and the UMOD2~UMOD0 bits to select the desired baud rate.
- Set the RXEN bit to ensure that the RX/TX pin is used as a UART receiver pin.

At this point the receiver will be enabled which will begin to look for a start bit.

When a character is received the following sequence of events will occur:

- The RXIF bit in the USR register will be set when the TXR\_RXR register has data available, the number of the available data bytes can be checked by polling the RxCNT register content.
- When the contents of the shift register have been transferred to the TXR\_RXR register and reach receiver FIFO trigger level, then if the RIE bit is set, an interrupt will be generated.
- If during reception, a frame error, noise error, parity error, or an overrun error has been detected, then the error flags can be set.

The RXIF bit can be cleared using the following software sequence:

1. A USR register access
2. A TXR\_RXR register read execution

### Receiving Break

Any break character received by the UART will be managed as a framing error. The receiver will count and expect a certain number of bit times as specified by the values programmed into the BNO bit plus one or two stop bits. If the break is much longer than 13 bit times, the reception will be considered as complete after the number of bit times specified by BNO plus one or two stop bits. The RXIF bit is set, FERR is set, zeros are loaded into the receive data register, interrupts are generated if appropriate and the RIDLE bit is set. A break is regarded as a character that contains only zeros with the FERR flag set. If a long break signal has been detected, the receiver will regard it as a data frame including a start bit, data bits and the invalid stop bit and the FERR flag will be set. The receiver must wait for a valid stop bit before looking for the next start bit. The receiver will not make the assumption that the break condition on the line is the next start bit. The break character will be loaded into the buffer and no further data will be received until one or two stop bits are received. It should be noted that the RIDLE read only flag will go high when the stop bits have not yet been received. The reception of a break character on the UART registers will result in the following:

- The framing error flag, FERR, will be set.
- The receive data register, TXR\_RXR, will be cleared.
- The OERR, NF, PERR, RIDLE or RXIF flags will possibly be set.

### Idle Status

When the receiver is reading data, which means it will be in between the detection of a start bit and the reading of a stop bit, the receiver status flag in the USR register, otherwise known as the RIDLE flag, will have a zero value. In between the reception of a stop bit and the detection of the next start bit, the RIDLE flag will have a high value, which indicates the receiver is in an idle condition.

### Receiver Interrupt

The read only receive interrupt flag RXIF in the USR register is set by an edge generated by the receiver. An interrupt is generated if RIE=1, when a word is transferred from the Receive Shift Register, RSR, to the Receive Data Register, TXR\_RXR. An overrun error can also generate an interrupt if RIE=1.

When a subroutine will be called with an execution time longer than the time for UART to receive five data bytes, if the UART received data could not be read in time during the subroutine execution, clear the RXEN bit to zero in advance to suspend data reception. If the UART interrupt could not be served in time to process the overrun error during the subroutine execution, ensure that both EMI and RXEN bits are disabled during this period, and then enable EMI and RXEN again after the subroutine execution has been completed to continue the UART data reception.

## Managing Receiver Errors

Several types of reception errors can occur within the UART module, the following section describes the various types and how they are managed by the UART.

### Overrun Error – OERR

The TXR\_RXR register is composed of a four-byte deep FIFO data buffer, where four bytes can be held in the FIFO register, while a fifth byte can continue to be received. Before this fifth byte has been entirely shifted in, the data should be read from the TXR\_RXR register. If this is not done, the overrun error flag OERR will be consequently indicated.

In the event of an overrun error occurring, the following will happen:

- The OERR flag in the USR register will be set.

- The TXR\_RXR contents will not be lost.
- The shift register will be overwritten.
- An interrupt will be generated if the RIE bit is set.

When the OERR flag is set to “1”, it is necessary to read five data bytes from the four-byte deep receiver FIFO and the shift register immediately to avoid unexpected errors, such as the UART is unable to receive data. If such an error occurs, clear the RXEN bit to “0” then set it to “1” again to continue data reception.

The OERR flag can be cleared by an access to the USR register followed by a read to the TXR\_RXR register.

#### **Noise Error – NF**

Over-sampling is used for data recovery to identify valid incoming data and noise. If noise is detected within a frame the following will occur:

- The read only noise flag, NF, in the USR register will be set on the rising edge of the RXIF bit.
- Data will be transferred from the Shift register to the TXR\_RXR register.
- No interrupt will be generated. However, this bit rises at the same time as the RXIF bit which itself generates an interrupt.

Note that the NF flag is reset by a USR register read operation followed by a TXR\_RXR register read operation.

#### **Framing Error – FERR**

The read only framing error flag, FERR, in the USR register, is set if a zero is detected instead of stop bits. If two stop bits are selected, both stop bits must be high; otherwise the FERR flag will be set. The FERR flag and the received data will be recorded in the USR and TXR\_RXR registers respectively, and the flag is cleared in any reset.

#### **Parity Error – PERR**

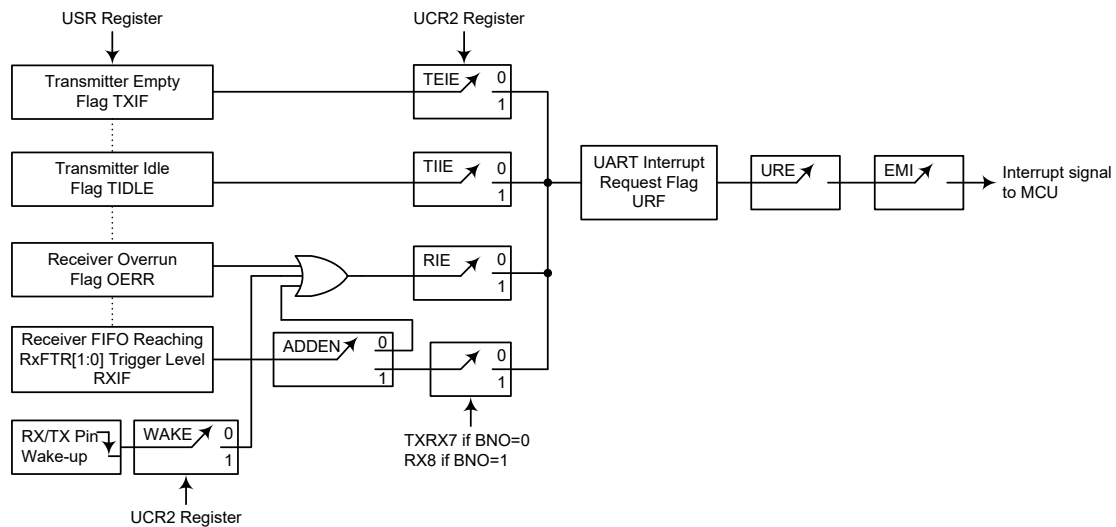
The read only parity error flag, PERR, in the USR register, is set if the parity of the received word is incorrect. This error flag is only applicable if the parity is enabled, PREN=1, and if the parity type, odd, even, mark or space, is selected. The read only PERR flag and the received data will be recorded in the USR and TXR\_RXR registers respectively. It is cleared on any reset, it should be noted that the flags, FERR and PERR, in the USR register should first be read by the application program before reading the data word.

### **UART Interrupt Structure**

Several individual UART conditions can generate a UART interrupt. When these conditions exist, a low pulse will be generated to get the attention of the microcontroller. These conditions are a transmitter data register empty, transmitter idle, receiver reaching FIFO trigger level, receiver overrun, address detect and an RX/TX pin wake-up. When any of these conditions are created, if the global interrupt enable bit and its corresponding interrupt control bit are enabled and the stack is not full, the program will jump to its corresponding interrupt vector where it can be serviced before returning to the main program. Four of these conditions have the corresponding USR register flags which will generate a UART interrupt if its associated interrupt enable control bit in the UCR2 register is set. The two transmitter interrupt conditions have their own corresponding enable control bits, while the two receiver interrupt conditions have a shared enable control bit. These enable bits can be used to mask out individual UART interrupt sources.

The address detect condition, which is also a UART interrupt source, does not have an associated flag, but will generate a UART interrupt when an address detect condition occurs if its function is enabled by setting the ADDEN bit in the UCR2 register. An RX/TX pin wake-up, which is also a UART interrupt source, does not have an associated flag, but will generate a UART interrupt if the UART clock ( $f_H$ ) source is switched off and the WAKE and RIE bits in the UCR2 register are set when a falling edge on the RX/TX pin occurs.

Note that the USR register flags are read only and cannot be cleared or set by the application program, neither will they be cleared when the program jumps to the corresponding interrupt servicing routine, as is the case for some of the other interrupts. The flags will be cleared automatically when certain actions are taken by the UART, the details of which are given in the UART register section. The overall UART interrupt can be disabled or enabled by the related interrupt enable control bits in the interrupt control registers of the microcontroller to decide whether the interrupt requested by the UART module is masked out or allowed.



UART Interrupt Structure

### Address Detect Mode

Setting the Address Detect Mode bit, ADDEN, in the UCR2 register, enables this special mode. If this bit is enabled then an additional qualifier will be placed on the generation of a Receiver Data Available interrupt, which is requested by the RXIF flag. If the ADDEN bit is enabled, then when data is available, an interrupt will only be generated if the highest received bit has a high value. Note that the URE and EMI interrupt enable bits must also be enabled for correct interrupt generation. This highest address bit is the 9th bit if BNO=1 or the 8th bit if BNO=0. If this bit is high, then the received word will be defined as an address rather than data. A Data Available interrupt will be generated every time the last bit of the received word is set. If the ADDEN bit is not enabled, then a Receiver Data Available interrupt will be generated each time the RXIF flag is set, irrespective of the data last bit status. The address detect mode and parity enable are mutually exclusive functions. Therefore, if the address detect mode is enabled, then to ensure correct operation, the parity function should be disabled by resetting the parity enable bit PREN to zero.

| ADDEN | 9th bit if BNO=1,<br>8th bit if BNO=0 | UART Interrupt<br>Generated |
|-------|---------------------------------------|-----------------------------|
| 0     | 0                                     | √                           |
|       | 1                                     | √                           |
| 1     | 0                                     | ×                           |
|       | 1                                     | √                           |

**ADDEN Bit Function**

### UART Power Down and Wake-up

When the UART clock,  $f_{H}$ , is switched off, the UART will cease to function. If the MCU switches off the UART clock,  $f_{H}$ , and enters the power down mode while a transmission is still in progress, then the transmission will be paused until the UART clock source derived from the microcontroller is activated. In a similar way, if the MCU switches off the UART clock  $f_{H}$  and enters the IDLE or SLEEP mode by executing the “HALT” instruction while receiving data, then the reception of data will likewise be paused. When the MCU enters the IDLE or SLEEP mode, note that the USR, UCR1, UCR2, UCR3, UFCR, RxCNT, TXR\_RXR as well as the BRDH and BRDL registers will not be affected. It is recommended to make sure first that the UART data transmission or reception has been finished before the microcontroller enters the IDLE or SLEEP mode.

The UART function contains a receiver RX/TX pin wake-up function, which is enabled or disabled by the WAKE bit in the UCR2 register. If this bit, along with the UART enable bit, UARTEN, the receiver enable bit, RXEN and the receiver interrupt bit, RIE, are all set when the MCU enters the power down mode with the UART clock  $f_{H}$  being switched off, then a falling edge on the RX/TX pin will trigger an RX/TX pin wake-up UART interrupt. Note that as it takes certain system clock cycles after a wake-up, before normal microcontroller operation resumes, any data received during this time on the RX/TX pin will be ignored.

For a UART wake-up interrupt to occur, in addition to the bits for the wake-up being set, the global interrupt enable bit, EMI, and the UART interrupt enable bit, URE, must be set. If the EMI and URE bits are not set, then only a wake up event will occur and no interrupt will be generated. Note also that as it takes certain system clock cycles after a wake-up before normal microcontroller resumes, the UART interrupt will not be generated until after this time has elapsed.

## Touch Key Function

The device provides multiple touch key functions. The touch key function is fully integrated and requires no external components, allowing touch key functions to be implemented by the simple manipulation of internal registers.

### Touch Key Structure

The touch keys are pin-shared with the I/O pins, with the desired function chosen via the corresponding pin-shared selection register bits. Keys are organised into one group, known as Module 0. The module is a fully independent set of four Touch Keys and each key has its own oscillator. The module contains its own control logic circuits and register set.

| Total Key Number | Touch Key | Shared I/O Pin |
|------------------|-----------|----------------|
| 4                | KEY1~KEY4 | PB0~PB3        |

**Touch Key Structure**

### Touch Key Register Definition

The touch key module 0, which contains four touch key functions, is controlled using several registers. The following table shows the register set for the touch key module 0.

| Register Name | Description   |
|---------------|---|
| TKTMR         | Touch key time slot 8-bit counter preload register                    |
| TKC0          | Touch key function Control register 0                                 |
| TKC1          | Touch key function Control register 1                                 |
| TK16DL        | Touch key function 16-bit counter low byte                            |
| TK16DH        | Touch key function 16-bit counter high byte                           |
| TKM016DL      | Touch key module 0 16-bit C/F counter low byte                        |
| TKM016DH      | Touch key module 0 16-bit C/F counter high byte                       |
| TKM0ROL       | Touch key module 0 reference oscillator capacitor selection low byte  |
| TKM0ROH       | Touch key module 0 reference oscillator capacitor selection high byte |
| TKM0C0        | Touch key module 0 Control register 0                                 |
| TKM0C1        | Touch key module 0 Control register 1                                 |

**Touch Key Function Register Definition**

| Register Name | Bit    |        |        |         |        |        |        |        |
|---------------|--------|--------|--------|---------|--------|--------|--------|--------|
|               | 7      | 6      | 5      | 4       | 3      | 2      | 1      | 0      |
| TKTMR         | D7     | D6     | D5     | D4      | D3     | D2     | D1     | D0     |
| TKC0          | —      | TKRCOV | TKST   | TKCFOV  | TK16OV | —      | TK16S1 | TK16S0 |
| TKC1          | —      | —      | —      | —       | —      | —      | TKFS1  | TKFS0  |
| TK16DL        | D7     | D6     | D5     | D4      | D3     | D2     | D1     | D0     |
| TK16DH        | D15    | D14    | D13    | D12     | D11    | D10    | D9     | D8     |
| TKM016DL      | D7     | D6     | D5     | D4      | D3     | D2     | D1     | D0     |
| TKM016DH      | D15    | D14    | D13    | D12     | D11    | D10    | D9     | D8     |
| TKM0ROL       | D7     | D6     | D5     | D4      | D3     | D2     | D1     | D0     |
| TKM0ROH       | —      | —      | —      | —       | —      | —      | D9     | D8     |
| TKM0C0        | M0MXS1 | M0MXS0 | M0DFEN | M0FILEN | M0SOFC | M0SOF2 | M0SOF1 | M0SOF0 |
| TKM0C1        | M0TSS  | —      | M0ROEN | M0KOEEN | M0K4EN | M0K3EN | M0K2EN | M0K1EN |

**Touch Key Function Register List**



• **TKTMR Register**

| Bit  | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   |
|------|-----|-----|-----|-----|-----|-----|-----|-----|
| Name | D7  | D6  | D5  | D4  | D3  | D2  | D1  | D0  |
| R/W  | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| POR  | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |

Bit 7~0 **D7~D0:** Touch key time slot 8-bit counter preload register

The touch key time slot counter preload register is used to determine the touch key time slot overflow time. The time slot unit period is obtained by a 5-bit counter and is equal to 32 time slot clock cycles. Therefore, the time slot counter overflow time is equal to the following equation shown.

Time slot counter overflow time =  $(256 - \text{TKTMR}[7:0]) \times 32 \text{ } t_{\text{rsc}}$ , where  $t_{\text{rsc}}$  is the time slot counter clock period.

• **TKC0 Register**

| Bit  | 7 | 6      | 5    | 4      | 3      | 2 | 1      | 0      |
|------|---|--------|------|--------|--------|---|--------|--------|
| Name | — | TKRCOV | TKST | TKCFOV | TK16OV | — | TK16S1 | TK16S0 |
| R/W  | — | R/W    | R/W  | R/W    | R/W    | — | R/W    | R/W    |
| POR  | — | 0      | 0    | 0      | 0      | — | 0      | 0      |

Bit 7 Unimplemented, read as “0”

Bit 6 **TKRCOV:** Touch key time slot counter overflow flag

0: No overflow occurs

1: Overflow occurs

When this bit is set by module 0 time slot counter overflow, the touch key interrupt request flag TKMF will be set and all module key oscillators and reference oscillators will automatically stop. The touch key module 16-bit C/F counter, touch key function 16-bit counter, 5-bit time slot unit period counter and 8-bit time slot counter will be automatically switched off. However, if this bit is set by application program, the touch key interrupt request flag will not be affected. Therefore, this bit cannot be set by application program but must be cleared to 0 by application program.

Bit 5 **TKST:** Touch key detection start control

0: Stopped or no operation

0→1: Start detection

In the module, the 16-bit C/F counter, touch key function 16-bit counter and 5-bit time slot unit period counter will automatically be cleared when this bit is cleared to zero. However, the 8-bit programmable time slot counter will not be cleared. When this bit is changed from low to high, the 16-bit C/F counter, touch key function 16-bit counter, 5-bit time slot unit period counter and 8-bit time slot counter will be switched on together with the key and reference oscillators to drive the corresponding counters.

Bit 4 **TKCFOV:** Touch key module 16-bit C/F counter overflow flag

0: No overflow occurs

1: Overflow occurs

This bit is set high by the touch key module 16-bit C/F counter overflow and must be cleared to 0 by application programs.

Bit 3 **TK16OV:** Touch key function 16-bit counter overflow flag

0: No overflow occurs

1: Overflow occurs

This bit is set high by the touch key function 16-bit counter overflow and must be cleared to 0 by application programs.

Bit 2 Unimplemented, read as “0”

Bit 1~0 **TK16S1~TK16S0:** Touch key function 16-bit counter clock source select

00:  $f_{\text{SYS}}$

01:  $f_{\text{SYS}}/2$

10:  $f_{\text{SYS}}/4$

11:  $f_{sys}/8$

• **TKC1 Register**

| Bit  | 7 | 6 | 5 | 4 | 3 | 2 | 1     | 0     |
|------|---|---|---|---|---|---|-------|-------|
| Name | — | — | — | — | — | — | TKFS1 | TKFS0 |
| R/W  | — | — | — | — | — | — | R/W   | R/W   |
| POR  | — | — | — | — | — | — | 1     | 1     |

Bit 7~2 Unimplemented, read as “0”

Bit 1~0 **TKFS1~TKFS0**: Touch Key oscillator and Reference oscillator frequency selection  
 00: 1MHz  
 01: 3MHz  
 10: 7MHz  
 11: 11MHz

• **TK16DH/TK16DL – Touch Key Function 16-bit Counter Register Pair**

| Register | TK16DH |     |     |     |     |     |    |    | TK16DL |    |    |    |    |    |    |    |
|----------|--------|-----|-----|-----|-----|-----|----|----|--------|----|----|----|----|----|----|----|
| Bit      | 7      | 6   | 5   | 4   | 3   | 2   | 1  | 0  | 7      | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
| Name     | D15    | D14 | D13 | D12 | D11 | D10 | D9 | D8 | D7     | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| R/W      | R      | R   | R   | R   | R   | R   | R  | R  | R      | R  | R  | R  | R  | R  | R  | R  |
| POR      | 0      | 0   | 0   | 0   | 0   | 0   | 0  | 0  | 0      | 0  | 0  | 0  | 0  | 0  | 0  | 0  |

This register pair is used to store the touch key function 16-bit counter value. This 16-bit counter can be used to calibrate the reference or key oscillator frequency. When the touch key time slot counter overflows, this 16-bit counter will be stopped and the counter content will be unchanged. This register pair will be cleared to zero when the TKST bit is cleared to zero.

• **TKM016DH/TKM016DL – Touch Key Module 0 16-bit C/F Counter Register Pair**

| Register | TKM016DH |     |     |     |     |     |    |    | TKM016DL |    |    |    |    |    |    |    |
|----------|----------|-----|-----|-----|-----|-----|----|----|----------|----|----|----|----|----|----|----|
| Bit      | 7        | 6   | 5   | 4   | 3   | 2   | 1  | 0  | 7        | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
| Name     | D15      | D14 | D13 | D12 | D11 | D10 | D9 | D8 | D7       | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| R/W      | R        | R   | R   | R   | R   | R   | R  | R  | R        | R  | R  | R  | R  | R  | R  | R  |
| POR      | 0        | 0   | 0   | 0   | 0   | 0   | 0  | 0  | 0        | 0  | 0  | 0  | 0  | 0  | 0  | 0  |

This register pair is used to store the touch key module 0 16-bit C/F counter value. This 16-bit C/F counter will be stopped and the counter content will be kept unchanged when the touch key time slot counter overflows. This register pair will be cleared to zero when the TKST bit is cleared to zero.

• **TKM0ROH/TKM0ROL – Touch Key Module 0 Reference Oscillator Capacitor Selection Register Pair**

| Register | TKM0ROH |   |   |   |   |   |     |     | TKM0ROL |     |     |     |     |     |     |     |
|----------|---------|---|---|---|---|---|-----|-----|---------|-----|-----|-----|-----|-----|-----|-----|
| Bit      | 7       | 6 | 5 | 4 | 3 | 2 | 1   | 0   | 7       | 6   | 5   | 4   | 3   | 2   | 1   | 0   |
| Name     | —       | — | — | — | — | — | D9  | D8  | D7      | D6  | D5  | D4  | D3  | D2  | D1  | D0  |
| R/W      | —       | — | — | — | — | — | R/W | R/W | R/W     | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| POR      | —       | — | — | — | — | — | 0   | 0   | 0       | 0   | 0   | 0   | 0   | 0   | 0   | 0   |

This register pair is used to store the touch key module 0 reference oscillator capacitor value.

The reference oscillator internal capacitor value= $(TKM0RO[9:0] \times 50pF) / 1024$

• **TKM0C0 Register**

| Bit  | 7      | 6      | 5      | 4       | 3      | 2      | 1      | 0      |
|------|--------|--------|--------|---------|--------|--------|--------|--------|
| Name | M0MXS1 | M0MXS0 | M0DFEN | M0FILEN | M0SOFC | M0SOF2 | M0SOF1 | M0SOF0 |
| R/W  | R/W    | R/W    | R/W    | R/W     | R/W    | R/W    | R/W    | R/W    |
| POR  | 0      | 0      | 0      | 0       | 0      | 0      | 0      | 0      |

Bit 7~6 **M0MXS1~M0MXS0**: Multiplexer Key Selection

- 00: KEY1
- 01: KEY2
- 10: KEY3
- 11: KEY4

Bit 5 **M0DFEN**: Touch key module 0 multi-frequency control

- 0: Disable
- 1: Enable

This bit is used to control the touch key oscillator frequency doubling function. When this bit is set to 1, the key oscillator frequency will be doubled.

Bit 4 **M0FILEN**: Touch key module 0 filter function control

- 0: Disable
- 1: Enable

Bit 3 **M0SOFC**: Touch key module 0 C/F oscillator frequency hopping function control selection

- 0: Controlled by M0SOF2~M0SOF0
- 1: Controlled by hardware circuit

This bit is used to select the touch key oscillator frequency hopping function control method. When this bit is set to 1, the key oscillator frequency hopping function is controlled by the hardware circuit regardless of the M0SOF2~M0SOF0 bits value.

Bit 2~0 **M0SOF2~M0SOF0**: Touch key module 0 Reference and Key oscillators hopping frequency selection (when M0SOFC=0)

- 000: 1.020MHz
- 001: 1.040MHz
- 010: 1.059MHz
- 011: 1.074MHz
- 100: 1.085MHz
- 101: 1.099MHz
- 110: 1.111MHz
- 111: 1.125MHz

These bits are used to select the touch key oscillator frequency for the hopping function. Note that these bits are only available when the M0SOFC bit is cleared to 0.

The frequency mentioned here will be changed when the external or internal capacitor has different values. If the touch key operates at 1MHz frequency, users can adjust the frequency in scale when any other frequency is selected.

• **TKM0C1 Register**

| Bit  | 7     | 6 | 5      | 4      | 3      | 2      | 1      | 0      |
|------|-------|---|--------|--------|--------|--------|--------|--------|
| Name | M0TSS | — | M0ROEN | M0KOEN | M0K4EN | M0K3EN | M0K2EN | M0K1EN |
| R/W  | R/W   | — | R/W    | R/W    | R/W    | R/W    | R/W    | R/W    |
| POR  | 0     | — | 0      | 0      | 0      | 0      | 0      | 0      |

Bit 7 **M0TSS**: Touch key module 0 time slot counter clock source selection

- 0: Touch key module 0 reference oscillator
- 1:  $f_{sys}/4$

Bit 6 Unimplemented, read as “0”

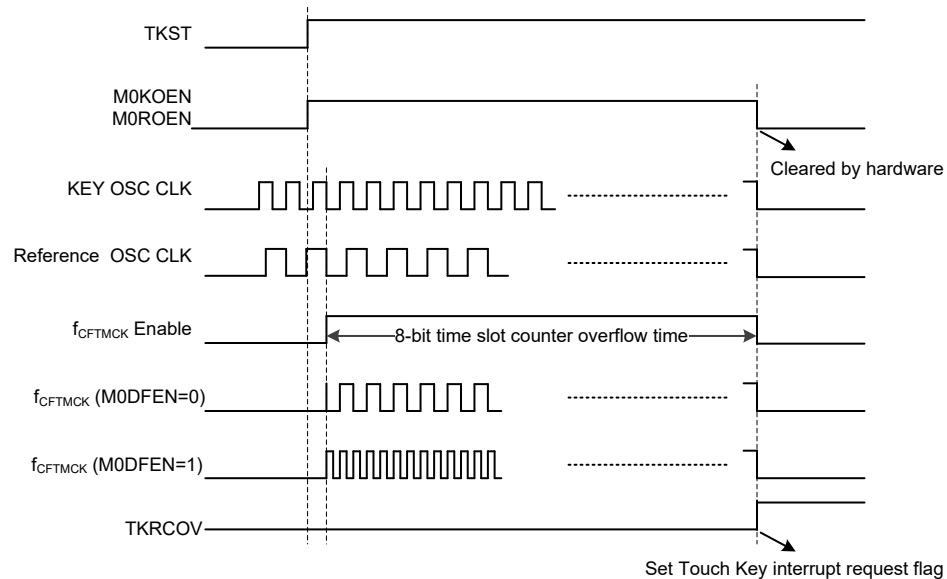
Bit 5 **M0ROEN**: Touch key module 0 Reference oscillator enable control

- 0: Disable
- 1: Enable

- Bit 4      **M0KOEN**: Touch key module 0 Key oscillator enable control  
             0: Disable  
             1: Enable
- Bit 3      **M0K4EN**: Touch key module 0 KEY4 enable control  
             0: Disable  
             1: Enable
- Bit 2      **M0K3EN**: Touch key module 0 KEY3 enable control  
             0: Disable  
             1: Enable
- Bit 1      **M0K2EN**: Touch key module 0 KEY2 enable control  
             0: Disable  
             1: Enable
- Bit 0      **M0K1EN**: Touch key module 0 KEY1 enable control  
             0: Disable  
             1: Enable

**Touch Key Operation**

When a finger touches or is in proximity to a touch pad, the capacitance of the pad will increase. By using this capacitance variation to change slightly the frequency of the internal sense oscillator, touch actions can be sensed by measuring these frequency changes. Using an internal programmable divider, the reference clock is used to generate a fixed time period. By counting a number of generated clock cycles from the sense oscillator during this fixed time period touch key actions can be determined.



**Touch Key Timing Diagram**

The touch key module 0 contains four touch key inputs, KEY1~KEY4, which are shared with logical I/O pins, and the desired function is selected using the relevant pin-shared control register bits. Each touch key has its own independent sense oscillator. Therefore, there are four sense oscillators within the touch key module 0.

During this reference clock fixed interval, the number of clock cycles generated by the sense oscillator is measured, and it is this value that is used to determine if a touch action has been made or not. At the end of the fixed reference clock time interval a Touch Key interrupt signal will be generated.

The touch key module 0 16-bit C/F counter, touch key function 16-bit counter, 5-bit time slot unit period counter will be automatically cleared when the TKST bit is cleared to zero, but the 8-bit programmable time slot counter will not be cleared. The overflow time is set by user. When the TKST bit changes from low to high, the 16-bit C/F counter, touch key function 16-bit counter, 5-bit time slot unit period counter and 8-bit time slot timer counter will be automatically switched on.

The key oscillator and reference oscillator in module 0 will be automatically stopped and the 16-bit C/F counter, touch key function 16-bit counter, 5-bit time slot unit period counter and 8-bit time slot timer counter will be automatically switched off when the time slot counter overflows. The clock source for the time slot counter is sourced from the reference oscillator or  $f_{SYS}/4$  which is selected using the M0TSS bit in the TKM0C1 register. The reference oscillator and key oscillator will be enabled by setting the M0ROEN bit and M0KOEN bits in the TKM0C1 register.

When the time slot counter in the touch key module 0 overflows, an actual touch key interrupt will take place. The touch keys mentioned here are the keys which are enabled.

### **Touch Key Interrupt**

The touch key only has a single interrupt, when the time slot counter in the touch key module 0 overflows, an actual touch key interrupt will take place. The touch keys mentioned here are the keys which are enabled. The 16-bit C/F counter, 16-bit counter, 5-bit time slot unit period counter and 8-bit time slot counter will be automatically cleared. More details regarding the touch key interrupt is located in the interrupt section of the datasheet.

### **Programming Considerations**

After the relevant registers are set, the touch key detection process is initiated by changing the TKST bit from low to high. This will enable and synchronise all relevant oscillators. The TKRCOV flag which is the time slot counter flag will go high when the counter overflows. When this happens an interrupt signal will be generated. As the TKRCOV flag will not be automatically cleared, it has to be cleared by the application program.

The TKCFOV flag which is the 16-bit C/F counter overflow flag will go high when the Touch Key Module 16-bit C/F counter overflows. As this flag will not be automatically cleared, it has to be cleared by the application program. The TK16OV flag which is the 16-bit counter overflow flag will go high when the 16-bit counter overflows. As this flag will not be automatically cleared, it has to be cleared by the application program.

When the external touch key size and layout are defined, their related capacitances will then determine the sensor oscillator frequency.

## Interrupts

Interrupts are an important part of any microcontroller system. When an external event or an internal function such as a Timer Module or an A/D converter requires microcontroller attention, their corresponding interrupt will enforce a temporary suspension of the main program allowing the microcontroller to direct attention to their respective needs. The device contains several external interrupt and internal interrupt functions. The external interrupt is generated by the action of the external INT pin, while the internal interrupts are generated by various internal functions including the TMs, Time Bases, EEPROM, UART, Touch Key, Proximity Sensing Circuit and the A/D converter.

### Interrupt Registers

Overall interrupt control, which basically means the setting of request flags when certain microcontroller conditions occur and the setting of interrupt enable bits by the application program, is controlled by a series of registers, located in the Special Purpose Data Memory. The registers fall into two categories. The first is the INTC0~INTC3 registers which setup the primary interrupts, the second is the INTEG register which sets the external interrupt trigger edge type.

Each register contains a number of enable bits to enable or disable individual registers as well as interrupt flags to indicate the presence of an interrupt request. The naming convention of these follows a specific pattern. First is listed an abbreviated interrupt type, then the (optional) number of that interrupt followed by either an “E” for enable/disable bit or “F” for request flag.

| Function                  | Enable Bit | Request Flag | Notes |
|---------------------------|------------|--------------|-------|
| Global                    | EMI        | —            | —     |
| INT Pin                   | INTE       | INTF         | —     |
| Proximity Sensing Circuit | OPDE       | OPDF         | —     |
| Touch Key                 | TKME       | TKMF         | —     |
| A/D Converter             | ADE        | ADF          | —     |
| UART                      | URE        | URF          | —     |
| CTM                       | CTMPE      | CTMPF        | —     |
|                           | CTMAE      | CTMAF        |       |
| STM                       | STMPE      | STMPF        | —     |
|                           | STMAE      | STMAF        |       |
| EEPROM                    | DEE        | DEF          | —     |
| Time Bases                | TBnE       | TBnF         | n=0~1 |

**Interrupt Register Bit Naming Conventions**

| Register Name | Bit   |       |       |       |       |       |       |       |
|---------------|-------|-------|-------|-------|-------|-------|-------|-------|
|               | 7     | 6     | 5     | 4     | 3     | 2     | 1     | 0     |
| INTEG         | —     | —     | —     | —     | —     | —     | INTS1 | INTS0 |
| INTC0         | —     | TKMF  | OPDF  | INTF  | TKME  | OPDE  | INTE  | EMI   |
| INTC1         | CTMAF | CTMPF | URF   | ADF   | CTMAE | CTMPE | URE   | ADE   |
| INTC2         | TB0F  | DEF   | STMAF | STMPF | TB0E  | DEE   | STMAE | STMPE |
| INTC3         | —     | —     | —     | TB1F  | —     | —     | —     | TB1E  |

**Interrupt Register List**

• **INTEG Register**

| Bit  | 7 | 6 | 5 | 4 | 3 | 2 | 1     | 0     |
|------|---|---|---|---|---|---|-------|-------|
| Name | — | — | — | — | — | — | INTS1 | INTS0 |
| R/W  | — | — | — | — | — | — | R/W   | R/W   |
| POR  | — | — | — | — | — | — | 0     | 0     |

Bit 7~2 Unimplemented, read as “0”

Bit 1~0 **INTS1~INTS0**: interrupt edge control for INT pin  
 00: Disable  
 01: Rising edge  
 10: Falling edge  
 11: Rising and falling edges

• **INTC0 Register**

| Bit  | 7 | 6    | 5    | 4    | 3    | 2    | 1    | 0   |
|------|---|------|------|------|------|------|------|-----|
| Name | — | TKMF | OPDF | INTF | TKME | OPDE | INTE | EMI |
| R/W  | — | R/W  | R/W  | R/W  | R/W  | R/W  | R/W  | R/W |
| POR  | — | 0    | 0    | 0    | 0    | 0    | 0    | 0   |

Bit 7 Unimplemented, read as “0”

Bit 6 **TKMF**: Touch Key interrupt request flag  
 0: No request  
 1: Interrupt request

Bit 5 **OPDF**: Proximity Sensing interrupt request flag  
 0: No request  
 1: Interrupt request

Bit 4 **INTF**: INT interrupt request flag  
 0: No request  
 1: Interrupt request

Bit 3 **TKME**: Touch Key interrupt control  
 0: Disable  
 1: Enable

Bit 2 **OPDE**: Proximity Sensing interrupt control  
 0: Disable  
 1: Enable

Bit 1 **INTE**: INT interrupt control  
 0: Disable  
 1: Enable

Bit 0 **EMI**: Global interrupt control  
 0: Disable  
 1: Enable

• **INTC1 Register**

| Bit  | 7     | 6     | 5   | 4   | 3     | 2     | 1   | 0   |
|------|-------|-------|-----|-----|-------|-------|-----|-----|
| Name | CTMAF | CTMPF | URF | ADF | CTMAE | CTMPE | URE | ADE |
| R/W  | R/W   | R/W   | R/W | R/W | R/W   | R/W   | R/W | R/W |
| POR  | 0     | 0     | 0   | 0   | 0     | 0     | 0   | 0   |

Bit 7 **CTMAF**: CTM Comparator A match interrupt request flag  
 0: No request  
 1: Interrupt request

Bit 6 **CTMPF**: CTM Comparator P match interrupt request flag  
 0: No request  
 1: Interrupt request

- Bit 5      **URF**: UART interrupt request flag  
0: No request  
1: Interrupt request
- Bit 4      **ADF**: A/D Converter interrupt request flag  
0: No request  
1: Interrupt request
- Bit 3      **CTMAE**: CTM Comparator A match interrupt control  
0: Disable  
1: Enable
- Bit 2      **CTMPE**: CTM Comparator P match interrupt control  
0: Disable  
1: Enable
- Bit 1      **URE**: UART interrupt control  
0: Disable  
1: Enable
- Bit 0      **ADE**: A/D Converter interrupt control  
0: Disable  
1: Enable

• **INTC2 Register**

| Bit  | 7    | 6   | 5     | 4     | 3    | 2   | 1     | 0     |
|------|------|-----|-------|-------|------|-----|-------|-------|
| Name | TB0F | DEF | STMAF | STMPF | TB0E | DEE | STMAE | STMPE |
| R/W  | R/W  | R/W | R/W   | R/W   | R/W  | R/W | R/W   | R/W   |
| POR  | 0    | 0   | 0     | 0     | 0    | 0   | 0     | 0     |

- Bit 7      **TB0F**: Time Base 0 interrupt request flag  
0: No request  
1: Interrupt request
- Bit 6      **DEF**: Data EEPROM interrupt request flag  
0: No request  
1: Interrupt request
- Bit 5      **STMAF**: STM Comparator A match interrupt request flag  
0: No request  
1: Interrupt request
- Bit 4      **STMPF**: STM Comparator P match interrupt request flag  
0: No request  
1: Interrupt request
- Bit 3      **TB0E**: Time Base 0 interrupt control  
0: Disable  
1: Enable
- Bit 2      **DEE**: Data EEPROM interrupt control  
0: Disable  
1: Enable
- Bit 1      **STMAE**: STM Comparator A match interrupt control  
0: Disable  
1: Enable
- Bit 0      **STMPE**: STM Comparator P match interrupt control  
0: Disable  
1: Enable



• **INTC3 Register**

| Bit  | 7 | 6 | 5 | 4    | 3 | 2 | 1 | 0    |
|------|---|---|---|------|---|---|---|------|
| Name | — | — | — | TB1F | — | — | — | TB1E |
| R/W  | — | — | — | R/W  | — | — | — | R/W  |
| POR  | — | — | — | 0    | — | — | — | 0    |

- Bit 7~5      Unimplemented, read as “0”
- Bit 4        **TB1F**: Time Base 1 interrupt request flag  
                  0: No request  
                  1: Interrupt request
- Bit 3~1      Unimplemented, read as “0”
- Bit 0        **TB1E**: Time Base 1 interrupt control  
                  0: Disable  
                  1: Enable

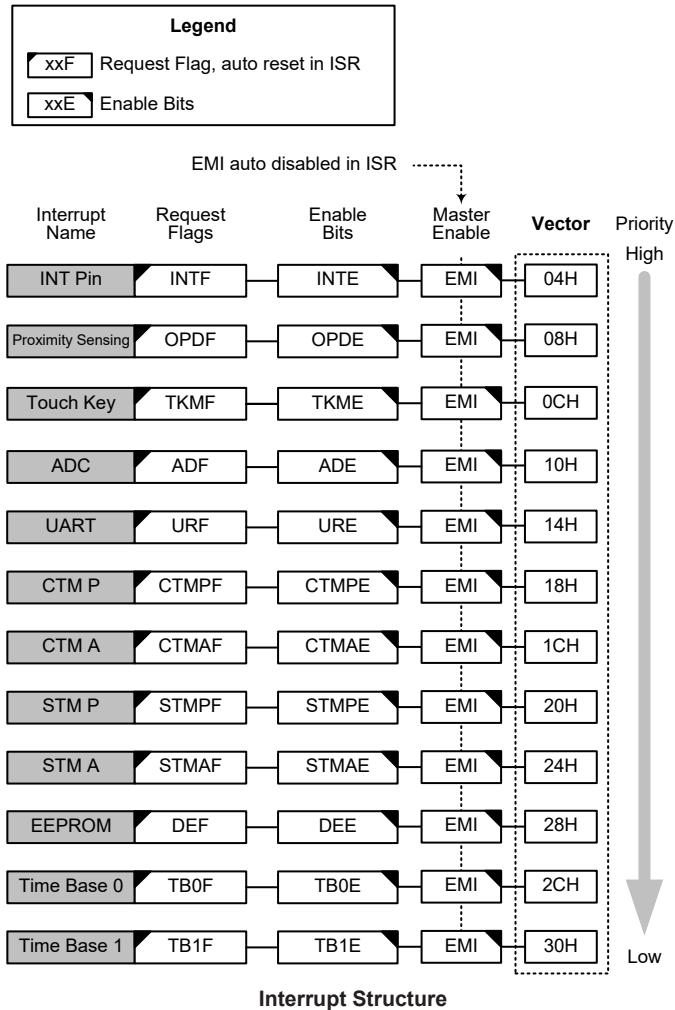
**Interrupt Operation**

When the conditions for an interrupt event occur, such as a TM Comparator P, Comparator A match or A/D conversion completion etc., the relevant interrupt request flag will be set. Whether the request flag actually generates a program jump to the relevant interrupt vector is determined by the condition of the interrupt enable bit. If the enable bit is set high, then the program will jump to its relevant vector; if the enable bit is zero then although the interrupt request flag is set an actual interrupt will not be generated and the program will not jump to the relevant interrupt vector. The global interrupt enable bit, if cleared to zero, will disable all interrupts.

When an interrupt is generated, the Program Counter, which stores the address of the next instruction to be executed, will be transferred onto the stack. The Program Counter will then be loaded with a new address which will be the value of the corresponding interrupt vector. The microcontroller will then fetch its next instruction from this interrupt vector. The instruction at this vector will usually be a “JMP” which will jump to another section of program which is known as the interrupt service routine. Here is located the code to control the appropriate interrupt. The interrupt service routine must be terminated with a “RETI”, which retrieves the original Program Counter address from the stack and allows the microcontroller to continue with normal execution at the point where the interrupt occurred.

The various interrupt enable bits, together with their associated request flags, are shown in the accompanying diagrams with their order of priority. All interrupt sources have their own individual vector. Once an interrupt subroutine is serviced, all the other interrupts will be blocked, as the global interrupt enable bit, EMI bit will be cleared automatically. This will prevent any further interrupt nesting from occurring. However, if other interrupt requests occur during this interval, although the interrupt will not be immediately serviced, the request flag will still be recorded.

If an interrupt requires immediate servicing while the program is already in another interrupt service routine, the EMI bit should be set after entering the routine, to allow interrupt nesting. If the stack is full, the interrupt request will not be acknowledged, even if the related interrupt is enabled, until the Stack Pointer is decremented. If immediate service is desired, the stack must be prevented from becoming full. In case of simultaneous requests, the accompanying diagram shows the priority that is applied. All of the interrupt request flags when set will wake up the device if it is in SLEEP or IDLE Mode, however to prevent a wake-up from occurring the corresponding flag should be set before the device are in SLEEP or IDLE Mode.



### External Interrupt

The external interrupt is controlled by signal transitions on the INT pin. An external interrupt request will take place when the external interrupt request flag, INTF, is set, which will occur when a transition, whose type is chosen by the edge selection bits, appears on the external interrupt pin. To allow the program to branch to its interrupt vector address, the global interrupt enable bit, EMI, and the external interrupt enable bit, INTE, must first be set. Additionally, the correct interrupt edge type must be selected using the INTEG register to enable the external interrupt function and to choose the trigger edge type. As the external interrupt pin is pin-shared with I/O pin, it can only be configured as external interrupt pin if its external interrupt enable bit in the corresponding interrupt register has been set and the external interrupt pin is selected by the corresponding pin-shared function selection bits. The pin must also be setup as an input by setting the corresponding bit in the port control register. When the interrupt is enabled, the stack is not full and the correct transition type appears on the external interrupt pin, a subroutine call to the external interrupt vector, will take place. When the interrupt is serviced, the external interrupt request flag, INTF, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts. Note that the pull-high resistor selection on the external interrupt pin will remain valid even if the pin is used as an external interrupt input.

The INTEG register is used to select the type of active edge that will trigger the external interrupt. A choice of either rising or falling or both edge types can be chosen to trigger an external interrupt. Note that the INTEG register can also be used to disable the external interrupt function.

### **Proximity Sensing Interrupt**

A Proximity Sensing Interrupt request will take place when the Proximity Sensing Interrupt request flag, OPDF, is set, which occurs when the Proximity Sensing circuit has detected a valid proximity signal. To allow the program to branch to its interrupt vector address, the global interrupt enable bit, EMI, and the Proximity Sensing Interrupt enable bit, OPDE, must first be set. When the interrupt is enabled, the stack is not full and a valid proximity signal is detected, a subroutine call to the Proximity Sensing Interrupt vector, will take place. When the interrupt is serviced, the Proximity Sensing Interrupt flag, OPDF, will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts.

### **Touch Key Interrupt**

A Touch Key Interrupt request will take place when the Touch Key Interrupt request flag, TKMF, is set, which occurs when the touch key module 0 time slot counter overflows. To allow the program to branch to its interrupt vector address, the global interrupt enable bit, EMI, and the Touch Key interrupt enable bit, TKME, must be first set. When the interrupt is enabled, the stack is not full and the Touch Key module 0 time slot counter overflow occurs, a subroutine call to the relevant interrupt vector, will take place. When the interrupt is serviced, the Touch Key interrupt request flag will be automatically reset and the EMI bit will also be automatically cleared to disable other interrupts.

### **A/D Converter Interrupt**

The A/D Converter Interrupt is controlled by the termination of an A/D conversion process. An A/D Converter Interrupt request will take place when the A/D Converter Interrupt request flag, ADF, is set, which occurs when the A/D conversion process finishes. To allow the program to branch to its interrupt vector address, the global interrupt enable bit, EMI, and A/D Interrupt enable bit, ADE, must first be set. When the interrupt is enabled, the stack is not full and the A/D conversion process has ended, a subroutine call to the A/D Converter Interrupt vector, will take place. When the interrupt is serviced, the A/D Converter Interrupt flag, ADF, will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts.

### **UART Interrupt**

Several individual UART conditions can generate a UART interrupt. When one of these conditions occurs, an interrupt signal will be generated to get the attention of the microcontroller. These conditions are a transmitter data register empty, transmitter idle, receiver reaching FIFO trigger level, receiver overrun, address detect and an RX/TX pin wake-up. To allow the program to branch to its interrupt vector addresses, the global interrupt enable bit, EMI, and the UART interrupt enable bit, URE, must first be set. When the interrupt is enabled, the stack is not full and any of these conditions are created, a subroutine call to the UART interrupt vector will take place. When the UART Interrupt is serviced, the UART interrupt request flag, URF, will be automatically cleared and the EMI bit will be automatically cleared to disable other interrupts. However, the USR register flags will only be cleared when certain actions are taken by the UART, the details of which are given in the UART section.

### TM Interrupts

The Compact and Standard Type TMs each have two interrupts, one comes from the comparator A match situation and the other comes from the comparator P match situation. For all of the TM types there are two interrupt request flags, xTMAF and xTMPF, and two enable control bits, xTMAE and xTMPE. A TM interrupt request will take place when any of the TM request flags are set, a situation which occurs when a TM comparator P or A match situation happens.

To allow the program to branch to their respective interrupt vector addresses, the global interrupt enable bit, EMI, and the respective TM Interrupt enable bit, xTMAE or xTMPE, must first be set. When the interrupt is enabled, the stack is not full and a TM comparator match situation occurs, a subroutine call to the corresponding TM Interrupt vector location, will take place. When the TM interrupt is serviced, the TM interrupt request flag, xTMAF or xTMPF, will be automatically cleared and the EMI bit will be also automatically cleared to disable other interrupts.

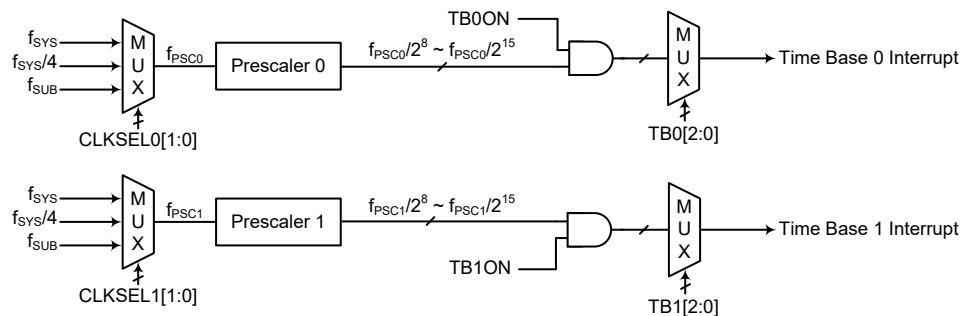
### EEPROM Interrupt

An EEPROM Interrupt request will take place when the EEPROM Interrupt request flag, DEF, is set, which occurs when an EEPROM Write cycle ends. To allow the program to branch to its interrupt vector address, the global interrupt enable bit, EMI, and the EEPROM Interrupt enable bit, DEE, must first be set. When the interrupt is enabled, the stack is not full and an EEPROM Write cycle ends, a subroutine call to the EEPROM Interrupt vector will take place. When the EEPROM Interrupt is serviced, the DEF flag will be automatically cleared and the EMI bit will also be automatically cleared to disable other interrupts.

### Time Base Interrupts

The function of the Time Base Interrupts is to provide regular time signals in the form of an internal interrupt. They are controlled by the overflow signals from their respective timer functions. When these happens their respective interrupt request flags, TB0F or TB1F will be set. To allow the program to branch to their respective interrupt vector addresses, the global interrupt enable bit, EMI and Time Base enable bits, TB0E or TB1E, must first be set. When the interrupt is enabled, the stack is not full and the Time Base overflows, a subroutine call to their respective vector locations will take place. When the interrupt is serviced, the respective interrupt request flag, TB0F or TB1F, will be automatically reset and the EMI bit will be cleared to disable other interrupts.

The purpose of the Time Base Interrupts is to provide an interrupt signal at fixed time periods. Its clock source,  $f_{PSC0}$  or  $f_{PSC1}$ , originates from the internal clock source  $f_{SYS}$ ,  $f_{SYS}/4$  or  $f_{SUB}$  and then passes through a divider, the division ratio of which is selected by programming the appropriate bits in the TB0C or TB1C register to obtain longer interrupt periods whose value ranges. The clock source that generates  $f_{PSC0}$  or  $f_{PSC1}$ , which in turn controls the Time Base interrupt period, is selected using the CLKSEL0[1:0] and CLKSEL1[1:0] bits in the PSC0R and PSC1R register respectively.



**Time Base Interrupts**

• **PSCnR Register (n=0~1)**

| Bit  | 7 | 6 | 5 | 4 | 3 | 2 | 1        | 0        |
|------|---|---|---|---|---|---|----------|----------|
| Name | — | — | — | — | — | — | CLKSELn1 | CLKSELn0 |
| R/W  | — | — | — | — | — | — | R/W      | R/W      |
| POR  | — | — | — | — | — | — | 0        | 0        |

Bit 7~2 Unimplemented, read as “0”

Bit 1~0 **CLKSELn1~CLKSELn0**: Prescaler n clock source selection

00:  $f_{SYS}$

01:  $f_{SYS}/4$

1x:  $f_{SUB}$

• **TBnC Register (n=0~1)**

| Bit  | 7     | 6 | 5 | 4 | 3 | 2    | 1    | 0    |
|------|-------|---|---|---|---|------|------|------|
| Name | TBnON | — | — | — | — | TBn2 | TBn1 | TBn0 |
| R/W  | R/W   | — | — | — | — | R/W  | R/W  | R/W  |
| POR  | 0     | — | — | — | — | 0    | 0    | 0    |

Bit 7 **TBnON**: Time Base n Control

0: Disable

1: Enable

Bit 6~3 Unimplemented, read as “0”

Bit 2~0 **TBn2~TBn0**: Time Base n Time-out Period Selection

000:  $2^8/f_{PSC}$

001:  $2^9/f_{PSC}$

010:  $2^{10}/f_{PSC}$

011:  $2^{11}/f_{PSC}$

100:  $2^{12}/f_{PSC}$

101:  $2^{13}/f_{PSC}$

110:  $2^{14}/f_{PSC}$

111:  $2^{15}/f_{PSC}$

## Interrupt Wake-up Function

Each of the interrupt functions has the capability of waking up the microcontroller when in the SLEEP or IDLE Mode. A wake-up is generated when an interrupt request flag changes from low to high and is independent of whether the interrupt is enabled or not. Therefore, even though the device is in the SLEEP or IDLE Mode and its system oscillator stopped, situations such as external edge transitions on the external interrupt pin may cause their respective interrupt flag to be set high and consequently generate an interrupt. Care must therefore be taken if spurious wake-up situations are to be avoided. If an interrupt wake-up function is to be disabled, then the corresponding interrupt request flag should be set high before the device enter the SLEEP or IDLE Mode. The interrupt enable bits have no effect on the interrupt wake-up function.

## Programming Considerations

By disabling the relevant interrupt enable bits, a requested interrupt can be prevented from being serviced, however, once an interrupt request flag is set, it will remain in this condition in the interrupt register until the corresponding interrupt is serviced or until the request flag is cleared by the application program.

It is recommended that programs do not use the “CALL” instruction within the interrupt service subroutine. Interrupts often occur in an unpredictable manner or need to be serviced immediately. If only one stack is left and the interrupt is not well controlled, the original control sequence will be damaged once a CALL subroutine is executed in the interrupt subroutine.

Every interrupt has the capability of waking up the microcontroller when it is in SLEEP or IDLE Mode, the wake up being generated when the interrupt request flag changes from low to high. If it is required to prevent a certain interrupt from waking up the microcontroller then its respective request flag should be first set high before enter SLEEP or IDLE Mode.

As only the Program Counter is pushed onto the stack, then when the interrupt is serviced, if the contents of the accumulator, status register or other registers are altered by the interrupt service program, their contents should be saved to the memory at the beginning of the interrupt service routine.

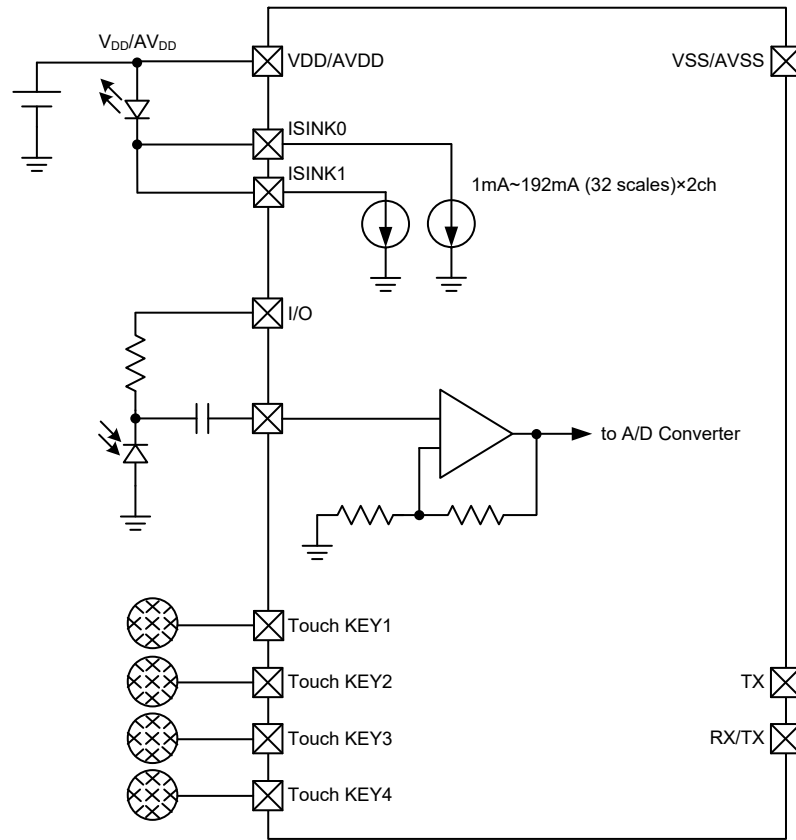
To return from an interrupt subroutine, either a RET or RETI instruction may be executed. The RETI instruction in addition to executing a return to the main program also automatically sets the EMI bit high to allow further interrupts. The RET instruction however only executes a return to the main program leaving the EMI bit in its present zero state and therefore disabling the execution of further interrupts.

## Configuration Options

Configuration options refer to certain options within the MCU that are programmed into the device during the programming process. During the development process, these options are selected using the HT-IDE software development tools. All options must be defined for proper system function, the details of which are shown in the table.

| No.                                 | Option   |
|-------------------------------------|--|
| <b>PB4 Pull-low Resistor Option</b> |  |
| 1                                   | PB4 pull-low resistor function:<br>1. Disable<br>2. Enable |

### Application Circuits



## Instruction Set

### Introduction

Central to the successful operation of any microcontroller is its instruction set, which is a set of program instruction codes that directs the microcontroller to perform certain operations. In the case of Holtek microcontroller, a comprehensive and flexible set of over 60 instructions is provided to enable programmers to implement their application with the minimum of programming overheads.

For easier understanding of the various instruction codes, they have been subdivided into several functional groupings.

### Instruction Timing

Most instructions are implemented within one instruction cycle. The exceptions to this are branch, call, or table read instructions where two instruction cycles are required. One instruction cycle is equal to 4 system clock cycles, therefore in the case of an 8MHz system oscillator, most instructions would be implemented within 0.5 $\mu$ s and branch or call instructions would be implemented within 1 $\mu$ s. Although instructions which require one more cycle to implement are generally limited to the JMP, CALL, RET, RETI and table read instructions, it is important to realize that any other instructions which involve manipulation of the Program Counter Low register or PCL will also take one more cycle to implement. As instructions which change the contents of the PCL will imply a direct jump to that new address, one more cycle will be required. Examples of such instructions would be "CLR PCL" or "MOV PCL, A". For the case of skip instructions, it must be noted that if the result of the comparison involves a skip operation then this will also take one more cycle, if no skip is involved then only one cycle is required.

### Moving and Transferring Data

The transfer of data within the microcontroller program is one of the most frequently used operations. Making use of three kinds of MOV instructions, data can be transferred from registers to the Accumulator and vice-versa as well as being able to move specific immediate data directly into the Accumulator. One of the most important data transfer applications is to receive data from the input ports and transfer data to the output ports.

### Arithmetic Operations

The ability to perform certain arithmetic operations and data manipulation is a necessary feature of most microcontroller applications. Within the Holtek microcontroller instruction set are a range of add and subtract instruction mnemonics to enable the necessary arithmetic to be carried out. Care must be taken to ensure correct handling of carry and borrow data when results exceed 255 for addition and less than 0 for subtraction. The increment and decrement instructions INC, INCA, DEC and DECA provide a simple means of increasing or decreasing by a value of one of the values in the destination specified.



## Logical and Rotate Operation

The standard logical operations such as AND, OR, XOR and CPL all have their own instruction within the Holtek microcontroller instruction set. As with the case of most instructions involving data manipulation, data must pass through the Accumulator which may involve additional programming steps. In all logical data operations, the zero flag may be set if the result of the operation is zero. Another form of logical data manipulation comes from the rotate instructions such as RR, RL, RRC and RLC which provide a simple means of rotating one bit right or left. Different rotate instructions exist depending on program requirements. Rotate instructions are useful for serial port programming applications where data can be rotated from an internal register into the Carry bit from where it can be examined and the necessary serial bit set high or low. Another application which rotate data operations are used is to implement multiplication and division calculations.

## Branches and Control Transfer

Program branching takes the form of either jumps to specified locations using the JMP instruction or to a subroutine using the CALL instruction. They differ in the sense that in the case of a subroutine call, the program must return to the instruction immediately when the subroutine has been carried out. This is done by placing a return instruction "RET" in the subroutine which will cause the program to jump back to the address right after the CALL instruction. In the case of a JMP instruction, the program simply jumps to the desired location. There is no requirement to jump back to the original jumping off point as in the case of the CALL instruction. One special and extremely useful set of branch instructions are the conditional branches. Here a decision is first made regarding the condition of a certain data memory or individual bits. Depending upon the conditions, the program will continue with the next instruction or skip over it and jump to the following instruction. These instructions are the key to decision making and branching within the program perhaps determined by the condition of certain input switches or by the condition of internal data bits.

## Bit Operations

The ability to provide single bit operations on Data Memory is an extremely flexible feature of all Holtek microcontrollers. This feature is especially useful for output port bit programming where individual bits or port pins can be directly set high or low using either the "SET [m].i" or "CLR [m].i" instructions respectively. The feature removes the need for programmers to first read the 8-bit output port, manipulate the input data to ensure that other bits are not changed and then output the port with the correct new data. This read-modify-write process is taken care of automatically when these bit operation instructions are used.

## Table Read Operations

Data storage is normally implemented by using registers. However, when working with large amounts of fixed data, the volume involved often makes it inconvenient to store the fixed data in the Data Memory. To overcome this problem, Holtek microcontrollers allow an area of Program Memory to be set as a table where data can be directly stored. A set of easy to use instructions provides the means by which this fixed data can be referenced and retrieved from the Program Memory.

## Other Operations

In addition to the above functional instructions, a range of other instructions also exist such as the "HALT" instruction for Power-down operations and instructions to control the operation of the Watchdog Timer for reliable program operations under extreme electric or electromagnetic environments. For their relevant operations, refer to the functional related sections.

## Instruction Set Summary

The following table depicts a summary of the instruction set categorised according to function and can be consulted as a basic instruction reference using the following listed conventions.

### Table Conventions

x: Bits immediate data  
 m: Data Memory address  
 A: Accumulator  
 i: 0~7 number of bits  
 addr: Program memory address

| Mnemonic                         | Description   | Cycles            | Flag Affected |
|----------------------------------|---|-------------------|---------------|
| <b>Arithmetic</b>                |   |                   |               |
| ADD A,[m]                        | Add Data Memory to ACC  | 1                 | Z, C, AC, OV  |
| ADDM A,[m]                       | Add ACC to Data Memory  | 1 <sup>Note</sup> | Z, C, AC, OV  |
| ADD A,x                          | Add immediate data to ACC                                       | 1                 | Z, C, AC, OV  |
| ADC A,[m]                        | Add Data Memory to ACC with Carry                               | 1                 | Z, C, AC, OV  |
| ADCM A,[m]                       | Add ACC to Data memory with Carry                               | 1 <sup>Note</sup> | Z, C, AC, OV  |
| SUB A,x                          | Subtract immediate data from the ACC                            | 1                 | Z, C, AC, OV  |
| SUB A,[m]                        | Subtract Data Memory from ACC                                   | 1                 | Z, C, AC, OV  |
| SUBM A,[m]                       | Subtract Data Memory from ACC with result in Data Memory        | 1 <sup>Note</sup> | Z, C, AC, OV  |
| SBC A,[m]                        | Subtract Data Memory from ACC with Carry                        | 1                 | Z, C, AC, OV  |
| SBCM A,[m]                       | Subtract Data Memory from ACC with Carry, result in Data Memory | 1 <sup>Note</sup> | Z, C, AC, OV  |
| DAA [m]                          | Decimal adjust ACC for Addition with result in Data Memory      | 1 <sup>Note</sup> | C             |
| <b>Logic Operation</b>           |   |                   |               |
| AND A,[m]                        | Logical AND Data Memory to ACC                                  | 1                 | Z             |
| OR A,[m]                         | Logical OR Data Memory to ACC                                   | 1                 | Z             |
| XOR A,[m]                        | Logical XOR Data Memory to ACC                                  | 1                 | Z             |
| ANDM A,[m]                       | Logical AND ACC to Data Memory                                  | 1 <sup>Note</sup> | Z             |
| ORM A,[m]                        | Logical OR ACC to Data Memory                                   | 1 <sup>Note</sup> | Z             |
| XORM A,[m]                       | Logical XOR ACC to Data Memory                                  | 1 <sup>Note</sup> | Z             |
| AND A,x                          | Logical AND immediate Data to ACC                               | 1                 | Z             |
| OR A,x                           | Logical OR immediate Data to ACC                                | 1                 | Z             |
| XOR A,x                          | Logical XOR immediate Data to ACC                               | 1                 | Z             |
| CPL [m]                          | Complement Data Memory  | 1 <sup>Note</sup> | Z             |
| CPLA [m]                         | Complement Data Memory with result in ACC                       | 1                 | Z             |
| <b>Increment &amp; Decrement</b> |   |                   |               |
| INCA [m]                         | Increment Data Memory with result in ACC                        | 1                 | Z             |
| INC [m]                          | Increment Data Memory   | 1 <sup>Note</sup> | Z             |
| DECA [m]                         | Decrement Data Memory with result in ACC                        | 1                 | Z             |
| DEC [m]                          | Decrement Data Memory   | 1 <sup>Note</sup> | Z             |
| <b>Rotate</b>                    |   |                   |               |
| RRA [m]                          | Rotate Data Memory right with result in ACC                     | 1                 | None          |
| RR [m]                           | Rotate Data Memory right  | 1 <sup>Note</sup> | None          |
| RRCA [m]                         | Rotate Data Memory right through Carry with result in ACC       | 1                 | C             |
| RRC [m]                          | Rotate Data Memory right through Carry                          | 1 <sup>Note</sup> | C             |
| RLA [m]                          | Rotate Data Memory left with result in ACC                      | 1                 | None          |
| RL [m]                           | Rotate Data Memory left   | 1 <sup>Note</sup> | None          |
| RLCA [m]                         | Rotate Data Memory left through Carry with result in ACC        | 1                 | C             |
| RLC [m]                          | Rotate Data Memory left through Carry                           | 1 <sup>Note</sup> | C             |

| Mnemonic                    | Description  | Cycles            | Flag Affected |
|-----------------------------|--|-------------------|---------------|
| <b>Data Move</b>            |  |                   |               |
| MOV A,[m]                   | Move Data Memory to ACC  | 1                 | None          |
| MOV [m],A                   | Move ACC to Data Memory  | 1 <sup>Note</sup> | None          |
| MOV A,x                     | Move immediate data to ACC   | 1                 | None          |
| <b>Bit Operation</b>        |  |                   |               |
| CLR [m].i                   | Clear bit of Data Memory   | 1 <sup>Note</sup> | None          |
| SET [m].i                   | Set bit of Data Memory   | 1 <sup>Note</sup> | None          |
| <b>Branch Operation</b>     |  |                   |               |
| JMP addr                    | Jump unconditionally   | 2                 | None          |
| SZ [m]                      | Skip if Data Memory is zero  | 1 <sup>Note</sup> | None          |
| SZA [m]                     | Skip if Data Memory is zero with data movement to ACC              | 1 <sup>Note</sup> | None          |
| SZ [m].i                    | Skip if bit i of Data Memory is zero                               | 1 <sup>Note</sup> | None          |
| SNZ [m].i                   | Skip if bit i of Data Memory is not zero                           | 1 <sup>Note</sup> | None          |
| SIZ [m]                     | Skip if increment Data Memory is zero                              | 1 <sup>Note</sup> | None          |
| SDZ [m]                     | Skip if decrement Data Memory is zero                              | 1 <sup>Note</sup> | None          |
| SIZA [m]                    | Skip if increment Data Memory is zero with result in ACC           | 1 <sup>Note</sup> | None          |
| SDZA [m]                    | Skip if decrement Data Memory is zero with result in ACC           | 1 <sup>Note</sup> | None          |
| CALL addr                   | Subroutine call  | 2                 | None          |
| RET                         | Return from subroutine   | 2                 | None          |
| RET A,x                     | Return from subroutine and load immediate data to ACC              | 2                 | None          |
| RETI                        | Return from interrupt  | 2                 | None          |
| <b>Table Read Operation</b> |  |                   |               |
| TABRD [m]                   | Read table (specific page or current page) to TBLH and Data Memory | 2 <sup>Note</sup> | None          |
| TABRDL [m]                  | Read table (last page) to TBLH and Data Memory                     | 2 <sup>Note</sup> | None          |
| <b>Miscellaneous</b>        |  |                   |               |
| NOP                         | No operation   | 1                 | None          |
| CLR [m]                     | Clear Data Memory  | 1 <sup>Note</sup> | None          |
| SET [m]                     | Set Data Memory  | 1 <sup>Note</sup> | None          |
| CLR WDT                     | Clear Watchdog Timer   | 1                 | TO, PDF       |
| SWAP [m]                    | Swap nibbles of Data Memory  | 1 <sup>Note</sup> | None          |
| SWAPA [m]                   | Swap nibbles of Data Memory with result in ACC                     | 1                 | None          |
| HALT                        | Enter power down mode  | 1                 | TO, PDF       |

Note: 1. For skip instructions, if the result of the comparison involves a skip then two cycles are required, if no skip takes place only one cycle is required.

2. Any instruction which changes the contents of the PCL will also require 2 cycles for execution.

## Instruction Definition

|                   |   |
|-------------------|---|
| <b>ADC A,[m]</b>  | Add Data Memory to ACC with Carry   |
| Description       | The contents of the specified Data Memory, Accumulator and the carry flag are added. The result is stored in the Accumulator.               |
| Operation         | $ACC \leftarrow ACC + [m] + C$  |
| Affected flag(s)  | OV, Z, AC, C  |
| <b>ADCM A,[m]</b> | Add ACC to Data Memory with Carry   |
| Description       | The contents of the specified Data Memory, Accumulator and the carry flag are added. The result is stored in the specified Data Memory.     |
| Operation         | $[m] \leftarrow ACC + [m] + C$  |
| Affected flag(s)  | OV, Z, AC, C  |
| <b>ADD A,[m]</b>  | Add Data Memory to ACC  |
| Description       | The contents of the specified Data Memory and the Accumulator are added. The result is stored in the Accumulator.                           |
| Operation         | $ACC \leftarrow ACC + [m]$  |
| Affected flag(s)  | OV, Z, AC, C  |
| <b>ADD A,x</b>    | Add immediate data to ACC   |
| Description       | The contents of the Accumulator and the specified immediate data are added. The result is stored in the Accumulator.                        |
| Operation         | $ACC \leftarrow ACC + x$  |
| Affected flag(s)  | OV, Z, AC, C  |
| <b>ADDM A,[m]</b> | Add ACC to Data Memory  |
| Description       | The contents of the specified Data Memory and the Accumulator are added. The result is stored in the specified Data Memory.                 |
| Operation         | $[m] \leftarrow ACC + [m]$  |
| Affected flag(s)  | OV, Z, AC, C  |
| <b>AND A,[m]</b>  | Logical AND Data Memory to ACC  |
| Description       | Data in the Accumulator and the specified Data Memory perform a bitwise logical AND operation. The result is stored in the Accumulator.     |
| Operation         | $ACC \leftarrow ACC \text{ "AND" } [m]$   |
| Affected flag(s)  | Z   |
| <b>AND A,x</b>    | Logical AND immediate data to ACC   |
| Description       | Data in the Accumulator and the specified immediate data perform a bit wise logical AND operation. The result is stored in the Accumulator. |
| Operation         | $ACC \leftarrow ACC \text{ "AND" } x$   |
| Affected flag(s)  | Z   |
| <b>ANDM A,[m]</b> | Logical AND ACC to Data Memory  |
| Description       | Data in the specified Data Memory and the Accumulator perform a bitwise logical AND operation. The result is stored in the Data Memory.     |
| Operation         | $[m] \leftarrow ACC \text{ "AND" } [m]$   |
| Affected flag(s)  | Z   |

|                  |  |
|------------------|--|
| <b>CALL addr</b> | Subroutine call  |
| Description      | Unconditionally calls a subroutine at the specified address. The Program Counter then increments by 1 to obtain the address of the next instruction which is then pushed onto the stack. The specified address is then loaded and the program continues execution from this new address. As this instruction requires an additional operation, it is a two cycle instruction.  |
| Operation        | Stack ← Program Counter + 1<br>Program Counter ← addr  |
| Affected flag(s) | None   |
| <b>CLR [m]</b>   | Clear Data Memory  |
| Description      | Each bit of the specified Data Memory is cleared to 0.   |
| Operation        | [m] ← 00H  |
| Affected flag(s) | None   |
| <b>CLR [m].i</b> | Clear bit of Data Memory   |
| Description      | Bit i of the specified Data Memory is cleared to 0.  |
| Operation        | [m].i ← 0  |
| Affected flag(s) | None   |
| <b>CLR WDT</b>   | Clear Watchdog Timer   |
| Description      | The TO, PDF flags and the WDT are all cleared.   |
| Operation        | WDT cleared<br>TO ← 0<br>PDF ← 0   |
| Affected flag(s) | TO, PDF  |
| <b>CPL [m]</b>   | Complement Data Memory   |
| Description      | Each bit of the specified Data Memory is logically complemented (1's complement). Bits which previously contained a 1 are changed to 0 and vice versa.   |
| Operation        | [m] ← $\overline{[m]}$   |
| Affected flag(s) | Z  |
| <b>CPLA [m]</b>  | Complement Data Memory with result in ACC  |
| Description      | Each bit of the specified Data Memory is logically complemented (1's complement). Bits which previously contained a 1 are changed to 0 and vice versa. The complemented result is stored in the Accumulator and the contents of the Data Memory remain unchanged.  |
| Operation        | ACC ← $\overline{[m]}$   |
| Affected flag(s) | Z  |
| <b>DAA [m]</b>   | Decimal-Adjust ACC for addition with result in Data Memory   |
| Description      | Convert the contents of the Accumulator value to a BCD (Binary Coded Decimal) value resulting from the previous addition of two BCD variables. If the low nibble is greater than 9 or if AC flag is set, then a value of 6 will be added to the low nibble. Otherwise the low nibble remains unchanged. If the high nibble is greater than 9 or if the C flag is set, then a value of 6 will be added to the high nibble. Essentially, the decimal conversion is performed by adding 00H, 06H, 60H or 66H depending on the Accumulator and flag conditions. Only the C flag may be affected by this instruction which indicates that if the original BCD sum is greater than 100, it allows multiple precision decimal addition. |
| Operation        | [m] ← ACC + 00H or<br>[m] ← ACC + 06H or<br>[m] ← ACC + 60H or<br>[m] ← ACC + 66H  |
| Affected flag(s) | C  |

|                  |  |
|------------------|--|
| <b>DEC [m]</b>   | Decrement Data Memory  |
| Description      | Data in the specified Data Memory is decremented by 1.   |
| Operation        | $[m] \leftarrow [m] - 1$   |
| Affected flag(s) | Z  |
| <br>             |  |
| <b>DECA [m]</b>  | Decrement Data Memory with result in ACC   |
| Description      | Data in the specified Data Memory is decremented by 1. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.  |
| Operation        | $ACC \leftarrow [m] - 1$   |
| Affected flag(s) | Z  |
| <br>             |  |
| <b>HALT</b>      | Enter power down mode  |
| Description      | This instruction stops the program execution and turns off the system clock. The contents of the Data Memory and registers are retained. The WDT and prescaler are cleared. The power down flag PDF is set and the WDT time-out flag TO is cleared.        |
| Operation        | $TO \leftarrow 0$<br>$PDF \leftarrow 1$  |
| Affected flag(s) | TO, PDF  |
| <br>             |  |
| <b>INC [m]</b>   | Increment Data Memory  |
| Description      | Data in the specified Data Memory is incremented by 1.   |
| Operation        | $[m] \leftarrow [m] + 1$   |
| Affected flag(s) | Z  |
| <br>             |  |
| <b>INCA [m]</b>  | Increment Data Memory with result in ACC   |
| Description      | Data in the specified Data Memory is incremented by 1. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.  |
| Operation        | $ACC \leftarrow [m] + 1$   |
| Affected flag(s) | Z  |
| <br>             |  |
| <b>JMP addr</b>  | Jump unconditionally   |
| Description      | The contents of the Program Counter are replaced with the specified address. Program execution then continues from this new address. As this requires the insertion of a dummy instruction while the new address is loaded, it is a two cycle instruction. |
| Operation        | Program Counter $\leftarrow$ addr  |
| Affected flag(s) | None   |
| <br>             |  |
| <b>MOV A,[m]</b> | Move Data Memory to ACC  |
| Description      | The contents of the specified Data Memory are copied to the Accumulator.   |
| Operation        | $ACC \leftarrow [m]$   |
| Affected flag(s) | None   |
| <br>             |  |
| <b>MOV A,x</b>   | Move immediate data to ACC   |
| Description      | The immediate data specified is loaded into the Accumulator.   |
| Operation        | $ACC \leftarrow x$   |
| Affected flag(s) | None   |
| <br>             |  |
| <b>MOV [m],A</b> | Move ACC to Data Memory  |
| Description      | The contents of the Accumulator are copied to the specified Data Memory.   |
| Operation        | $[m] \leftarrow ACC$   |
| Affected flag(s) | None   |

|                  |  |
|------------------|--|
| <b>NOP</b>       | No operation   |
| Description      | No operation is performed. Execution continues with the next instruction.  |
| Operation        | No operation   |
| Affected flag(s) | None   |
| <br>             |  |
| <b>OR A,[m]</b>  | Logical OR Data Memory to ACC  |
| Description      | Data in the Accumulator and the specified Data Memory perform a bitwise logical OR operation. The result is stored in the Accumulator.   |
| Operation        | ACC ← ACC "OR" [m]   |
| Affected flag(s) | Z  |
| <br>             |  |
| <b>OR A,x</b>    | Logical OR immediate data to ACC   |
| Description      | Data in the Accumulator and the specified immediate data perform a bitwise logical OR operation. The result is stored in the Accumulator.  |
| Operation        | ACC ← ACC "OR" x   |
| Affected flag(s) | Z  |
| <br>             |  |
| <b>ORM A,[m]</b> | Logical OR ACC to Data Memory  |
| Description      | Data in the specified Data Memory and the Accumulator perform a bitwise logical OR operation. The result is stored in the Data Memory.   |
| Operation        | [m] ← ACC "OR" [m]   |
| Affected flag(s) | Z  |
| <br>             |  |
| <b>RET</b>       | Return from subroutine   |
| Description      | The Program Counter is restored from the stack. Program execution continues at the restored address.   |
| Operation        | Program Counter ← Stack  |
| Affected flag(s) | None   |
| <br>             |  |
| <b>RET A,x</b>   | Return from subroutine and load immediate data to ACC  |
| Description      | The Program Counter is restored from the stack and the Accumulator loaded with the specified immediate data. Program execution continues at the restored address.  |
| Operation        | Program Counter ← Stack<br>ACC ← x   |
| Affected flag(s) | None   |
| <br>             |  |
| <b>RETI</b>      | Return from interrupt  |
| Description      | The Program Counter is restored from the stack and the interrupts are re-enabled by setting the EMI bit. EMI is the master interrupt global enable bit. If an interrupt was pending when the RETI instruction is executed, the pending Interrupt routine will be processed before returning to the main program. |
| Operation        | Program Counter ← Stack<br>EMI ← 1   |
| Affected flag(s) | None   |
| <br>             |  |
| <b>RL [m]</b>    | Rotate Data Memory left  |
| Description      | The contents of the specified Data Memory are rotated left by 1 bit with bit 7 rotated into bit 0.   |
| Operation        | [m].(i+1) ← [m].i; (i=0~6)<br>[m].0 ← [m].7  |
| Affected flag(s) | None   |

|                  |   |
|------------------|---|
| <b>RLA [m]</b>   | Rotate Data Memory left with result in ACC  |
| Description      | The contents of the specified Data Memory are rotated left by 1 bit with bit 7 rotated into bit 0. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.  |
| Operation        | $ACC.(i+1) \leftarrow [m].i; (i=0\sim6)$<br>$ACC.0 \leftarrow [m].7$  |
| Affected flag(s) | None  |
| <br>             |   |
| <b>RLC [m]</b>   | Rotate Data Memory left through Carry   |
| Description      | The contents of the specified Data Memory and the carry flag are rotated left by 1 bit. Bit 7 replaces the Carry bit and the original carry flag is rotated into bit 0.   |
| Operation        | $[m].(i+1) \leftarrow [m].i; (i=0\sim6)$<br>$[m].0 \leftarrow C$<br>$C \leftarrow [m].7$  |
| Affected flag(s) | C   |
| <br>             |   |
| <b>RLCA [m]</b>  | Rotate Data Memory left through Carry with result in ACC  |
| Description      | Data in the specified Data Memory and the carry flag are rotated left by 1 bit. Bit 7 replaces the Carry bit and the original carry flag is rotated into the bit 0. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged. |
| Operation        | $ACC.(i+1) \leftarrow [m].i; (i=0\sim6)$<br>$ACC.0 \leftarrow C$<br>$C \leftarrow [m].7$  |
| Affected flag(s) | C   |
| <br>             |   |
| <b>RR [m]</b>    | Rotate Data Memory right  |
| Description      | The contents of the specified Data Memory are rotated right by 1 bit with bit 0 rotated into bit 7.   |
| Operation        | $[m].i \leftarrow [m].(i+1); (i=0\sim6)$<br>$[m].7 \leftarrow [m].0$  |
| Affected flag(s) | None  |
| <br>             |   |
| <b>RRA [m]</b>   | Rotate Data Memory right with result in ACC   |
| Description      | Data in the specified Data Memory is rotated right by 1 bit with bit 0 rotated into bit 7. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.  |
| Operation        | $ACC.i \leftarrow [m].(i+1); (i=0\sim6)$<br>$ACC.7 \leftarrow [m].0$  |
| Affected flag(s) | None  |
| <br>             |   |
| <b>RRC [m]</b>   | Rotate Data Memory right through Carry  |
| Description      | The contents of the specified Data Memory and the carry flag are rotated right by 1 bit. Bit 0 replaces the Carry bit and the original carry flag is rotated into bit 7.  |
| Operation        | $[m].i \leftarrow [m].(i+1); (i=0\sim6)$<br>$[m].7 \leftarrow C$<br>$C \leftarrow [m].0$  |
| Affected flag(s) | C   |



|                   |   |
|-------------------|---|
| <b>RRCA [m]</b>   | Rotate Data Memory right through Carry with result in ACC   |
| Description       | Data in the specified Data Memory and the carry flag are rotated right by 1 bit. Bit 0 replaces the Carry bit and the original carry flag is rotated into bit 7. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.  |
| Operation         | ACC.i ← [m].(i+1); (i=0~6)<br>ACC.7 ← C<br>C ← [m].0  |
| Affected flag(s)  | C   |
| <b>SBC A,[m]</b>  | Subtract Data Memory from ACC with Carry  |
| Description       | The contents of the specified Data Memory and the complement of the carry flag are subtracted from the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.   |
| Operation         | ACC ← ACC – [m] – $\bar{C}$   |
| Affected flag(s)  | OV, Z, AC, C  |
| <b>SBCM A,[m]</b> | Subtract Data Memory from ACC with Carry and result in Data Memory  |
| Description       | The contents of the specified Data Memory and the complement of the carry flag are subtracted from the Accumulator. The result is stored in the Data Memory. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.   |
| Operation         | [m] ← ACC – [m] – $\bar{C}$   |
| Affected flag(s)  | OV, Z, AC, C  |
| <b>SDZ [m]</b>    | Skip if decrement Data Memory is 0  |
| Description       | The contents of the specified Data Memory are first decremented by 1. If the result is 0 the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.  |
| Operation         | [m] ← [m] – 1<br>Skip if [m]=0  |
| Affected flag(s)  | None  |
| <b>SDZA [m]</b>   | Skip if decrement Data Memory is zero with result in ACC  |
| Description       | The contents of the specified Data Memory are first decremented by 1. If the result is 0, the following instruction is skipped. The result is stored in the Accumulator but the specified Data Memory contents remain unchanged. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0, the program proceeds with the following instruction. |
| Operation         | ACC ← [m] – 1<br>Skip if ACC=0  |
| Affected flag(s)  | None  |
| <b>SET [m]</b>    | Set Data Memory   |
| Description       | Each bit of the specified Data Memory is set to 1.  |
| Operation         | [m] ← FFH   |
| Affected flag(s)  | None  |
| <b>SET [m].i</b>  | Set bit of Data Memory  |
| Description       | Bit i of the specified Data Memory is set to 1.   |
| Operation         | [m].i ← 1   |
| Affected flag(s)  | None  |

|                   |  |
|-------------------|--|
| <b>SIZ [m]</b>    | Skip if increment Data Memory is 0   |
| Description       | The contents of the specified Data Memory are first incremented by 1. If the result is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.  |
| Operation         | $[m] \leftarrow [m] + 1$<br>Skip if $[m]=0$  |
| Affected flag(s)  | None   |
| <br>              |  |
| <b>SIZA [m]</b>   | Skip if increment Data Memory is zero with result in ACC   |
| Description       | The contents of the specified Data Memory are first incremented by 1. If the result is 0, the following instruction is skipped. The result is stored in the Accumulator but the specified Data Memory contents remain unchanged. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction. |
| Operation         | $ACC \leftarrow [m] + 1$<br>Skip if $ACC=0$  |
| Affected flag(s)  | None   |
| <br>              |  |
| <b>SNZ [m].i</b>  | Skip if bit i of Data Memory is not 0  |
| Description       | If bit i of the specified Data Memory is not 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is 0 the program proceeds with the following instruction.  |
| Operation         | Skip if $[m].i \neq 0$   |
| Affected flag(s)  | None   |
| <br>              |  |
| <b>SUB A,[m]</b>  | Subtract Data Memory from ACC  |
| Description       | The specified Data Memory is subtracted from the contents of the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.  |
| Operation         | $ACC \leftarrow ACC - [m]$   |
| Affected flag(s)  | OV, Z, AC, C   |
| <br>              |  |
| <b>SUBM A,[m]</b> | Subtract Data Memory from ACC with result in Data Memory   |
| Description       | The specified Data Memory is subtracted from the contents of the Accumulator. The result is stored in the Data Memory. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.  |
| Operation         | $[m] \leftarrow ACC - [m]$   |
| Affected flag(s)  | OV, Z, AC, C   |
| <br>              |  |
| <b>SUB A,x</b>    | Subtract immediate data from ACC   |
| Description       | The immediate data specified by the code is subtracted from the contents of the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.   |
| Operation         | $ACC \leftarrow ACC - x$   |
| Affected flag(s)  | OV, Z, AC, C   |
| <br>              |  |
| <b>SWAP [m]</b>   | Swap nibbles of Data Memory  |
| Description       | The low-order and high-order nibbles of the specified Data Memory are interchanged.  |
| Operation         | $[m].3\sim[m].0 \leftrightarrow [m].7\sim[m].4$  |
| Affected flag(s)  | None   |

|                   |  |
|-------------------|--|
| <b>SWAPA [m]</b>  | Swap nibbles of Data Memory with result in ACC   |
| Description       | The low-order and high-order nibbles of the specified Data Memory are interchanged. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.   |
| Operation         | ACC.3~ACC.0 ← [m].7~[m].4<br>ACC.7~ACC.4 ← [m].3~[m].0   |
| Affected flag(s)  | None   |
| <b>SZ [m]</b>     | Skip if Data Memory is 0   |
| Description       | The contents of the specified Data Memory are read out and then written to the specified Data Memory again. If the contents of the specified Data Memory is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction. |
| Operation         | Skip if [m]=0  |
| Affected flag(s)  | None   |
| <b>SZA [m]</b>    | Skip if Data Memory is 0 with data movement to ACC   |
| Description       | The contents of the specified Data Memory are copied to the Accumulator. If the value is zero, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.   |
| Operation         | ACC ← [m]<br>Skip if [m]=0   |
| Affected flag(s)  | None   |
| <b>SZ [m].i</b>   | Skip if bit i of Data Memory is 0  |
| Description       | If bit i of the specified Data Memory is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0, the program proceeds with the following instruction.   |
| Operation         | Skip if [m].i=0  |
| Affected flag(s)  | None   |
| <b>TABRD [m]</b>  | Read table (specific page or current page) to TBLH and Data Memory   |
| Description       | The low byte of the program code addressed by the table pointer (TBHP and TBLP or only TBLP if no TBHP) is moved to the specified Data Memory and the high byte moved to TBLH.   |
| Operation         | [m] ← program code (low byte)<br>TBLH ← program code (high byte)   |
| Affected flag(s)  | None   |
| <b>TABRDL [m]</b> | Read table (last page) to TBLH and Data Memory   |
| Description       | The low byte of the program code (last page) addressed by the table pointer (TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.  |
| Operation         | [m] ← program code (low byte)<br>TBLH ← program code (high byte)   |
| Affected flag(s)  | None   |
| <b>XOR A,[m]</b>  | Logical XOR Data Memory to ACC   |
| Description       | Data in the Accumulator and the specified Data Memory perform a bitwise logical XOR operation. The result is stored in the Accumulator.  |
| Operation         | ACC ← ACC "XOR" [m]  |
| Affected flag(s)  | Z  |

|                   |  |
|-------------------|--|
| <b>XORM A,[m]</b> | Logical XOR ACC to Data Memory   |
| Description       | Data in the specified Data Memory and the Accumulator perform a bitwise logical XOR operation. The result is stored in the Data Memory.    |
| Operation         | $[m] \leftarrow \text{ACC} \text{ "XOR" } [m]$   |
| Affected flag(s)  | Z  |
| <br>              |  |
| <b>XOR A,x</b>    | Logical XOR immediate data to ACC  |
| Description       | Data in the Accumulator and the specified immediate data perform a bitwise logical XOR operation. The result is stored in the Accumulator. |
| Operation         | $\text{ACC} \leftarrow \text{ACC} \text{ "XOR" } x$  |
| Affected flag(s)  | Z  |

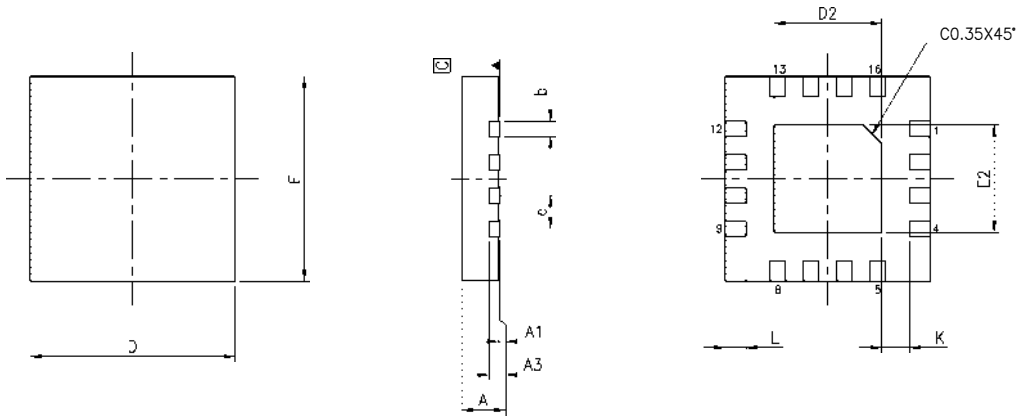
## Package Information

Note that the package information provided here is for consultation purposes only. As this information may be updated at regular intervals users are reminded to consult the [Holtek website](#) for the latest version of the [Package/Carton Information](#).

Additional supplementary information with regard to packaging is listed below. Click on the relevant section to be transferred to the relevant website page.

- Package Information (include Outline Dimensions, Product Tape and Reel Specifications)
- The Operation Instruction of Packing Materials
- Carton information

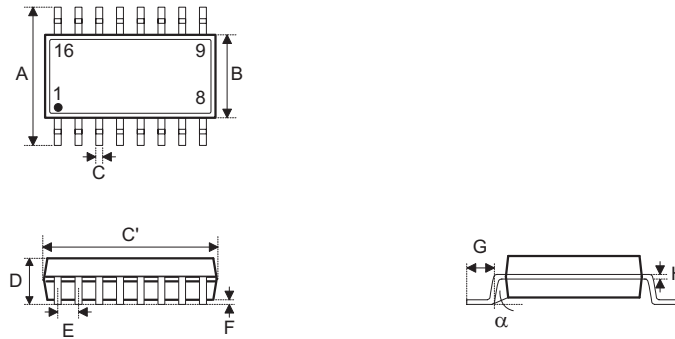
**SAW Type 16-pin QFN (4mm×4mm×0.75mm) Outline Dimensions**



| Symbol | Dimensions in inch |       |       |
|--------|--------------------|-------|-------|
|        | Min.               | Nom.  | Max.  |
| A      | 0.028              | 0.030 | 0.031 |
| A1     | 0.000              | 0.001 | 0.002 |
| A3     | 0.080 REF          |       |       |
| b      | 0.010              | 0.012 | 0.014 |
| D      | 0.157 BSC          |       |       |
| E      | 0.157 BSC          |       |       |
| e      | 0.026 BSC          |       |       |
| D2     | 0.079              | —     | 0.106 |
| E2     | 0.079              | —     | 0.106 |
| L      | 0.014              | 0.016 | 0.018 |
| K      | 0.008              | —     | —     |

| Symbol | Dimensions in mm |      |      |
|--------|------------------|------|------|
|        | Min.             | Nom. | Max. |
| A      | 0.70             | 0.75 | 0.80 |
| A1     | 0.00             | 0.02 | 0.05 |
| A3     | 0.203 REF        |      |      |
| b      | 0.25             | 0.30 | 0.35 |
| D      | 4.00 BSC         |      |      |
| E      | 4.00 BSC         |      |      |
| e      | 0.65 BSC         |      |      |
| D2     | 2.00             | —    | 2.70 |
| E2     | 2.00             | —    | 2.70 |
| L      | 0.35             | 0.40 | 0.45 |
| K      | 0.20             | —    | —    |

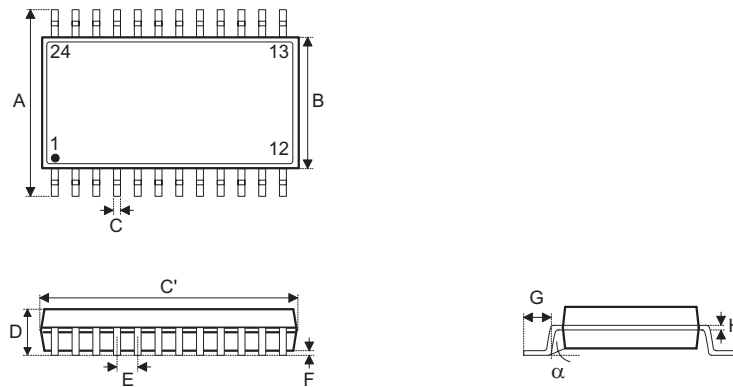
**16-pin NSOP (150mil) Outline Dimensions**



| Symbol   | Dimensions in inch |      |       |
|----------|--------------------|------|-------|
|          | Min.               | Nom. | Max.  |
| A        | 0.236 BSC          |      |       |
| B        | 0.154 BSC          |      |       |
| C        | 0.012              | —    | 0.020 |
| C'       | 0.390 BSC          |      |       |
| D        | —                  | —    | 0.069 |
| E        | 0.050 BSC          |      |       |
| F        | 0.004              | —    | 0.010 |
| G        | 0.016              | —    | 0.050 |
| H        | 0.004              | —    | 0.010 |
| $\alpha$ | 0°                 | —    | 8°    |

| Symbol   | Dimensions in mm |      |      |
|----------|------------------|------|------|
|          | Min.             | Nom. | Max. |
| A        | 6.00 BSC         |      |      |
| B        | 3.90 BSC         |      |      |
| C        | 0.31             | —    | 0.51 |
| C'       | 9.90 BSC         |      |      |
| D        | —                | —    | 1.75 |
| E        | 1.27 BSC         |      |      |
| F        | 0.10             | —    | 0.25 |
| G        | 0.40             | —    | 1.27 |
| H        | 0.10             | —    | 0.25 |
| $\alpha$ | 0°               | —    | 8°   |

**24-pin SSOP (150mil) Outline Dimensions**



| Symbol | Dimensions in inch |      |       |
|--------|--------------------|------|-------|
|        | Min.               | Nom. | Max.  |
| A      | 0.236 BSC          |      |       |
| B      | 0.154 BSC          |      |       |
| C      | 0.008              | —    | 0.012 |
| C'     | 0.341 BSC          |      |       |
| D      | —                  | —    | 0.069 |
| E      | 0.025 BSC          |      |       |
| F      | 0.004              | —    | 0.010 |
| G      | 0.016              | —    | 0.050 |
| H      | 0.004              | —    | 0.010 |
| α      | 0°                 | —    | 8°    |

| Symbol | Dimensions in mm |      |      |
|--------|------------------|------|------|
|        | Min.             | Nom. | Max. |
| A      | 6.00 BSC         |      |      |
| B      | 3.90 BSC         |      |      |
| C      | 0.20             | —    | 0.30 |
| C'     | 8.66 BSC         |      |      |
| D      | —                | —    | 1.75 |
| E      | 0.635 BSC        |      |      |
| F      | 0.10             | —    | 0.25 |
| G      | 0.41             | —    | 1.27 |
| H      | 0.10             | —    | 0.25 |
| α      | 0°               | —    | 8°   |



Copyright© 2023 by HOLTEK SEMICONDUCTOR INC. All Rights Reserved.

The information provided in this document has been produced with reasonable care and attention before publication, however, HOLTEK does not guarantee that the information is completely accurate. The information contained in this publication is provided for reference only and may be superseded by updates. HOLTEK disclaims any expressed, implied or statutory warranties, including but not limited to suitability for commercialization, satisfactory quality, specifications, characteristics, functions, fitness for a particular purpose, and non-infringement of any third-party's rights. HOLTEK disclaims all liability arising from the information and its application. In addition, HOLTEK does not recommend the use of HOLTEK's products where there is a risk of personal hazard due to malfunction or other reasons. HOLTEK hereby declares that it does not authorize the use of these products in life-saving, life-sustaining or safety critical components. Any use of HOLTEK's products in life-saving/sustaining or safety applications is entirely at the buyer's risk, and the buyer agrees to defend, indemnify and hold HOLTEK harmless from any damages, claims, suits, or expenses resulting from such use. The information provided in this document, including but not limited to the content, data, examples, materials, graphs, and trademarks, is the intellectual property of HOLTEK (and its licensors, where applicable) and is protected by copyright law and other intellectual property laws. No license, express or implied, to any intellectual property right, is granted by HOLTEK herein. HOLTEK reserves the right to revise the information described in the document at any time without prior notice. For the latest information, please contact us.