



**Enhanced A/D MCU with EEPROM and UART**

**HT66F3197**

Revision: V1.10 Date: November 24, 2016

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

## Table of Contents

|  |           |
|--|-----------|
| <b>Features</b> .....                                  | <b>6</b>  |
| CPU Features .....                                     | 6         |
| Peripheral Features.....                               | 6         |
| <b>General Description</b> .....                       | <b>7</b>  |
| <b>Block Diagram</b> .....                             | <b>7</b>  |
| <b>Pin Assignment</b> .....                            | <b>8</b>  |
| <b>Pin Description</b> .....                           | <b>9</b>  |
| <b>Absolute Maximum Ratings</b> .....                  | <b>13</b> |
| <b>D.C. Characteristics</b> .....                      | <b>13</b> |
| <b>A.C. Characteristics</b> .....                      | <b>15</b> |
| <b>A/D Converter Electrical Characteristics</b> .....  | <b>18</b> |
| <b>LVD &amp; LVR Electrical Characteristics</b> .....  | <b>20</b> |
| <b>Comparator Electrical Characteristics</b> .....     | <b>21</b> |
| <b>LCD Electrical Characteristics</b> .....            | <b>23</b> |
| <b>Power on Reset Characteristics</b> .....            | <b>23</b> |
| <b>System Architecture</b> .....                       | <b>24</b> |
| Clocking and Pipelining.....                           | 24        |
| Program Counter.....                                   | 25        |
| Stack .....  | 26        |
| Arithmetic and Logic Unit – ALU .....                  | 26        |
| <b>Flash Program Memory</b> .....                      | <b>27</b> |
| Structure.....   | 27        |
| Special Vectors .....                                  | 27        |
| Look-up Table.....                                     | 27        |
| Table Program Example.....                             | 28        |
| In Circuit Programming – ICP .....                     | 29        |
| On-Chip Debug Support – OCDS .....                     | 30        |
| <b>Data Memory</b> .....                               | <b>31</b> |
| Structure.....   | 31        |
| Data Memory Addressing.....                            | 32        |
| General Purpose Data Memory .....                      | 32        |
| Special Purpose Data Memory .....                      | 32        |
| <b>Special Function Register Description</b> .....     | <b>34</b> |
| Indirect Addressing Registers – IAR0, IAR1, IAR2 ..... | 34        |
| Memory Pointers – MP0, MP1L/MP1H, MP2L/MP2H.....       | 34        |
| Accumulator – ACC.....                                 | 36        |
| Program Counter Low Register – PCL.....                | 36        |
| Look-up Table Registers – TBLP, TBHP, TBLH.....        | 36        |
| Status Register – STATUS.....                          | 36        |

|   |           |
|---|-----------|
| <b>EEPROM Data Memory</b> .....                   | <b>38</b> |
| EEPROM Data Memory Structure .....                | 38        |
| EEPROM Registers .....                            | 38        |
| Reading Data from the EEPROM .....                | 40        |
| Writing Data to the EEPROM.....                   | 40        |
| Write Protection.....                             | 40        |
| EEPROM Interrupt .....                            | 40        |
| Programming Considerations.....                   | 41        |
| <b>Oscillators</b> .....                          | <b>42</b> |
| Oscillator Overview .....                         | 42        |
| System Clock Configurations .....                 | 42        |
| External Crystal/Ceramic Oscillator – HXT .....   | 43        |
| Internal RC Oscillator – HIRC .....               | 44        |
| External 32.768kHz Crystal Oscillator – LXT ..... | 44        |
| Internal 32kHz Oscillator – LIRC .....            | 45        |
| Supplementary Oscillators .....                   | 45        |
| <b>Operating Modes and System Clocks</b> .....    | <b>46</b> |
| System Clocks .....                               | 46        |
| System Operation Modes.....                       | 47        |
| Control Register .....                            | 48        |
| Operating Mode Switching .....                    | 51        |
| Standby Current Considerations .....              | 55        |
| Wake-up .....                                     | 55        |
| Programming Considerations .....                  | 56        |
| <b>Watchdog Timer</b> .....                       | <b>57</b> |
| Watchdog Timer Clock Source.....                  | 57        |
| Watchdog Timer Control Register .....             | 57        |
| Watchdog Timer Operation .....                    | 58        |
| <b>Reset and Initialisation</b> .....             | <b>59</b> |
| Reset Functions .....                             | 59        |
| Reset Initial Conditions .....                    | 62        |
| <b>Input/Output Ports</b> .....                   | <b>66</b> |
| Pull-high Resistors .....                         | 67        |
| Port A Wake-up .....                              | 68        |
| I/O Port Control Registers .....                  | 68        |
| I/O Port Source Current Control.....              | 69        |
| Pin-shared Functions .....                        | 71        |
| I/O Pin Structures.....                           | 79        |
| Programming Considerations .....                  | 79        |
| <b>Timer Modules – TM</b> .....                   | <b>80</b> |
| Introduction .....                                | 80        |
| TM Operation .....                                | 80        |
| TM Clock Source.....                              | 80        |
| TM Interrupts.....                                | 81        |

|  |            |
|--|------------|
| TM External Pins .....                                     | 81         |
| TM Input/Output Pin Selection .....                        | 82         |
| Programming Considerations.....                            | 83         |
| <b>Compact Type TM – CTM .....</b>                         | <b>84</b>  |
| Compact TM Operation.....                                  | 84         |
| Compact Type TM Register Description.....                  | 84         |
| Compact Type TM Operating Modes .....                      | 88         |
| <b>Standard Type TM – STM .....</b>                        | <b>94</b>  |
| Standard TM Operation.....                                 | 94         |
| Standard Type TM Register Description .....                | 95         |
| Standard Type TM Operation Modes .....                     | 99         |
| <b>Periodic Type TM – PTM.....</b>                         | <b>109</b> |
| Periodic TM Operation .....                                | 109        |
| Periodic Type TM Register Description.....                 | 110        |
| Periodic Type TM Operation Modes.....                      | 114        |
| <b>Analog to Digital Converter – ADC.....</b>              | <b>123</b> |
| A/D Converter Overview .....                               | 123        |
| A/D Converter Register Description .....                   | 124        |
| A/D Converter Data Registers – SADOL, SADOH.....           | 124        |
| A/D Converter Control Registers – SADC0, SADC1, SADC2..... | 124        |
| A/D Converter Operation.....                               | 127        |
| A/D Converter Reference Voltage.....                       | 128        |
| A/D Converter Input Signal .....                           | 129        |
| Conversion Rate and Timing Diagram .....                   | 130        |
| Summary of A/D Conversion Steps.....                       | 131        |
| Programming Considerations.....                            | 132        |
| A/D Converter Transfer Function .....                      | 132        |
| A/D Converter Programming Examples .....                   | 133        |
| <b>Comparator .....</b>                                    | <b>135</b> |
| Comparator Operation .....                                 | 135        |
| Comparator Registers.....                                  | 135        |
| Offset calibration procedure .....                         | 137        |
| <b>Serial Interface Module – SIM .....</b>                 | <b>138</b> |
| SPI Interface .....  | 138        |
| I <sup>2</sup> C Interface .....                           | 144        |
| <b>UART Interface.....</b>                                 | <b>154</b> |
| UART External Pin.....                                     | 155        |
| UART Data Transfer Scheme.....                             | 155        |
| UART Status and Control Registers.....                     | 155        |
| Baud Rate Generator.....                                   | 160        |
| UART Setup and Control.....                                | 161        |
| UART Transmitter.....                                      | 163        |
| UART Receiver .....  | 164        |
| Managing Receiver Errors .....                             | 166        |

|  |            |
|--|------------|
| UART Interrupt Structure.....                  | 167        |
| UART Power Down and Wake-up.....               | 168        |
| <b>SCOM Function for LCD.....</b>              | <b>169</b> |
| LCD Operation .....                            | 169        |
| LCD Bias Current Control .....                 | 169        |
| <b>Interrupts .....</b>                        | <b>170</b> |
| Interrupt Registers.....                       | 170        |
| Interrupt Operation .....                      | 175        |
| External Interrupt.....                        | 176        |
| Comparator Interrupt.....                      | 177        |
| Multi-function Interrupt .....                 | 177        |
| A/D Converter Interrupt.....                   | 177        |
| Time Base Interrupt.....                       | 178        |
| Serial Interface Module Interrupt.....         | 179        |
| UART Transfer Interrupt.....                   | 180        |
| EEPROM Write Interrupt.....                    | 180        |
| LVD Interrupt .....                            | 180        |
| TM Interrupts .....                            | 180        |
| Interrupt Wake-up Function.....                | 181        |
| Programming Considerations.....                | 181        |
| <b>Low Voltage Detector – LVD .....</b>        | <b>182</b> |
| LVD Register .....                             | 182        |
| LVD Operation.....                             | 183        |
| <b>Configuration Option.....</b>               | <b>184</b> |
| <b>Application Circuits.....</b>               | <b>185</b> |
| <b>Instruction Set.....</b>                    | <b>186</b> |
| Introduction .....                             | 186        |
| Instruction Timing .....                       | 186        |
| Moving and Transferring Data.....              | 186        |
| Arithmetic Operations.....                     | 186        |
| Logical and Rotate Operation .....             | 187        |
| Branches and Control Transfer .....            | 187        |
| Bit Operations .....                           | 187        |
| Table Read Operations .....                    | 187        |
| Other Operations.....                          | 187        |
| <b>Instruction Set Summary .....</b>           | <b>188</b> |
| Table Conventions.....                         | 188        |
| Extended Instruction Set.....                  | 190        |
| <b>Instruction Definition.....</b>             | <b>192</b> |
| Extended Instruction Definition .....          | 201        |
| <b>Package Information .....</b>               | <b>208</b> |
| 48-pin LQFP (7mm×7mm) Outline Dimensions ..... | 209        |

## Features

### CPU Features

- Operating Voltage:
  - ♦  $f_{SYS}=4\text{MHz}$ : 1.8V~5.5V
  - ♦  $f_{SYS}=8\text{MHz}$ : 2.0V~5.5V
  - ♦  $f_{SYS}=12\text{MHz}$ : 2.7V~5.5V
  - ♦  $f_{SYS}=16\text{MHz}$ : 3.3V~5.5V
- Up to 0.25 $\mu\text{s}$  instruction cycle with 16MHz system clock at  $V_{DD}=5\text{V}$
- Power down and wake-up functions to reduce power consumption
- Oscillators:
  - ♦ External High Speed Crystal – HXT
  - ♦ Internal High Speed RC – HIRC
  - ♦ External 32.768kHz Crystal – LXT
  - ♦ Internal 32kHz RC – LIRC
- Multi-mode operation: NORMAL, SLOW, IDLE and SLEEP
- Fully integrated internal 4/8/12MHz oscillator requires no external components
- All instructions executed in one to three instruction cycles
- Table read instructions
- 115 powerful instructions
- 16-level subroutine nesting
- Bit manipulation instruction

### Peripheral Features

- Flash Program Memory: 8K  $\times$  16
- RAM Data Memory: 256  $\times$  8
- True EEPROM Memory: 64  $\times$  8
- Watchdog Timer function
- 44 bidirectional I/O lines
- Software controlled 4-SCOM lines LCD driver with 1/2 bias
- Two external interrupt lines shared with I/O pins
- Multiple Timer Modules for time measure, input capture, compare match output, PWM output or single pulse output functions
- Comparator function
- Dual Time-Base functions for generation of fixed time interrupt signals
- 16-channel 12-bit resolution A/D converter
- Serial Interfaces – SPI, I<sup>2</sup>C and full-duplex UART
- PMOS source current adjustable
- Low Voltage Reset function
- Low Voltage Detect function
- Package type: 48-pin LQFP

## General Description

The device is a Flash Memory type 8-bit high performance RISC architecture microcontroller. Offering users the convenience of Flash Memory multi-programming features, the device also includes a wide range of functions and 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.

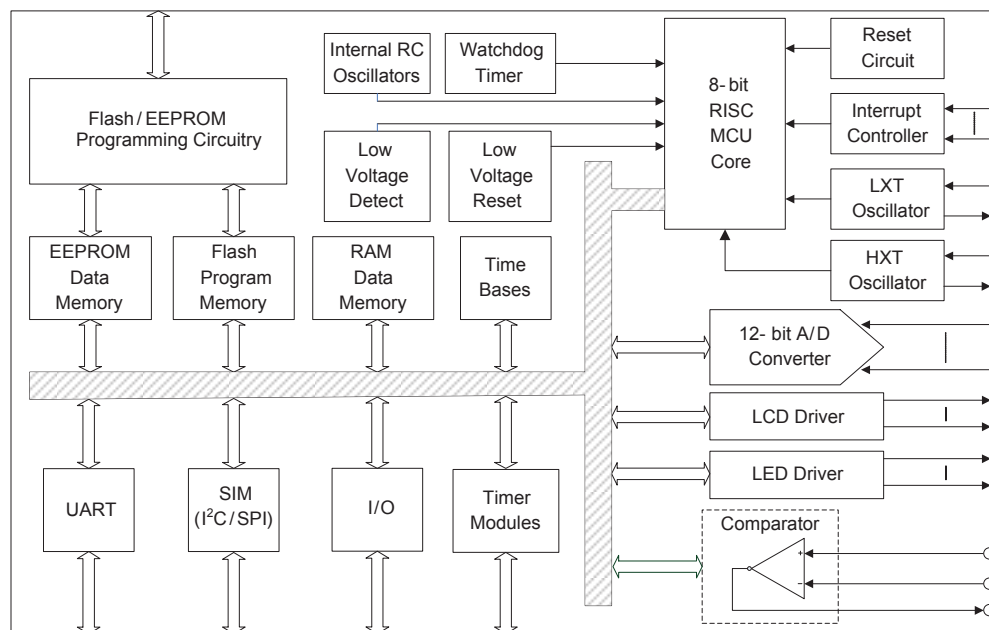
Analog features include a multi-channel 12-bit A/D converter and a comparator function. Multiple and extremely flexible Timer Modules provide timing, pulse generation and PWM generation functions. Communication with the outside world is catered for by including fully integrated SPI, I<sup>2</sup>C and UART interface functions, popular interfaces which provide designers with a means of easy communication with external peripheral hardware. Protective features such as an internal Watchdog Timer, Low Voltage Reset and Low Voltage Detector coupled with excellent noise immunity and ESD protection ensure that reliable operation is maintained in hostile electrical environments.

A full choice of external and internal, high speed and low speed oscillator functions are provided including a fully integrated system oscillator which requires no external components for its 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 minimize power consumption.

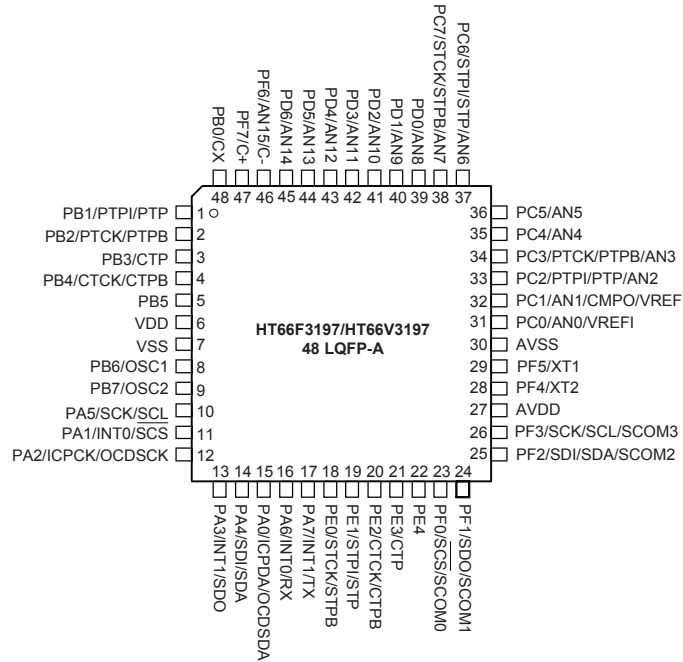
The UART module can support applications such as data communication, networking between microcontrollers, low-cost data links between PCs and peripheral devices, portable and battery operated device communication, etc.

The inclusion of flexible I/O programming features, timebase functions along with many other features ensure that the device will find excellent use in applications such as electronic metering, environmental monitoring, handheld instruments, household appliances, electronically controlled tools, motor driving in addition to many others.

## Block Diagram



## Pin Assignment



- Note: 1. If the pin-shared pin functions have multiple outputs simultaneously, the desired pin-shared function is determined by the corresponding software control bits.
2. The actual device and its equivalent OCDS EV device share the same package type, however the OCDS EV device part number is HT66V3197. Pins OCDSCK and OCSDSDA which are pin-shared with PA2 and PA0 are only used for the OCDS EV device.



## Pin Description

With the exception of the power pins, all pins on the device can be referenced by its Port name, e.g. PA0, PA1 etc., which refer to the digital I/O function of the pins. However these Port pins are also shared with other function such as the Analog to Digital Converter, Timer Module pins etc. 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.

| Pin Name                          | Function                | OPT                   | I/T | O/T  | Description  |
|-----------------------------------|-------------------------|-----------------------|-----|------|--|
| PA0/ICPDA/<br>OCDSDA              | PA0                     | PAPU<br>PAWU<br>PAS0  | ST  | CMOS | General purpose I/O. Register enabled pull-high and wake-up. |
|                                   | ICPDA                   | —                     | ST  | CMOS | ICP Address/Data   |
|                                   | OCDSDA                  | —                     | ST  | CMOS | OCDS Address/Data, for EV chip only                          |
| PA1/INT0/ $\overline{\text{SCS}}$ | PA1                     | PAPU<br>PAWU<br>PAS0  | ST  | CMOS | General purpose I/O. Register enabled pull-high and wake-up. |
|                                   | INT0                    | PAS0<br>IFS1<br>INTEG | ST  | —    | External Interrupt 0   |
|                                   | $\overline{\text{SCS}}$ | PAS0<br>IFS0          | ST  | CMOS | SPI slave select   |
| PA2/ICPCK/<br>OCDSCK              | PA2                     | PAPU<br>PAWU<br>PAS0  | ST  | CMOS | General purpose I/O. Register enabled pull-high and wake-up. |
|                                   | ICPCK                   | —                     | ST  | —    | ICP Clock pin  |
|                                   | OCDSCK                  | —                     | ST  | —    | OCDS Clock pin, for EV chip only                             |
| PA3/INT1/SDO                      | PA3                     | PAPU<br>PAWU<br>PAS0  | ST  | CMOS | General purpose I/O. Register enabled pull-high and wake-up. |
|                                   | INT1                    | PAS0<br>IFS1<br>INTEG | ST  | —    | External Interrupt 1   |
|                                   | SDO                     | PAS0                  | —   | CMOS | SPI data output  |
| PA4/SDI/SDA                       | PA4                     | PAPU<br>PAWU<br>PAS1  | ST  | CMOS | General purpose I/O. Register enabled pull-high and wake-up. |
|                                   | SDI                     | PAS1<br>IFS0          | ST  | —    | SPI data input   |
|                                   | SDA                     | PAS1<br>IFS0          | ST  | NMOS | I <sup>2</sup> C data line                                   |
| PA5/SCK/SCL                       | PA5                     | PAPU<br>PAWU<br>PAS1  | ST  | CMOS | General purpose I/O. Register enabled pull-high and wake-up. |
|                                   | SCK                     | PAS1<br>IFS0          | ST  | CMOS | SPI serial Clock   |
|                                   | SCL                     | PAS1<br>IFS0          | ST  | NMOS | I <sup>2</sup> C clock line                                  |
| PA6/INT0/RX                       | PA6                     | PAPU<br>PAWU<br>PAS1  | ST  | CMOS | General purpose I/O. Register enabled pull-high and wake-up. |
|                                   | INT0                    | PAS1<br>IFS1<br>INTEG | ST  | —    | External Interrupt 0   |
|                                   | RX                      | PAS1<br>UCR1          | ST  | —    | UART RX serial data input                                    |

| Pin Name              | Function | OPT                   | I/T | O/T  | Description  |
|-----------------------|----------|-----------------------|-----|------|--|
| PA7/INT1/TX           | PA7      | PAPU<br>PAWU<br>PAS1  | ST  | CMOS | General purpose I/O. Register enabled pull-high and wake-up. |
|                       | INT1     | PAS1<br>IFS1<br>INTEG | ST  | —    | External Interrupt 1   |
|                       | TX       | PAS1<br>UCR1          | —   | CMOS | UART TX serial data output                                   |
| PB0/CX                | PB0      | PBPU<br>PBS0          | ST  | CMOS | General purpose I/O. Register enabled pull-high.             |
|                       | CX       | PBS0                  | —   | CMOS | Comparator output  |
| PB1/PTPI/PTP          | PB1      | PBPU<br>PBS0          | ST  | CMOS | General purpose I/O. Register enabled pull-high.             |
|                       | PTPI     | PBS0<br>IFS0          | ST  | —    | PTM capture input  |
|                       | PTP      | PBS0                  | —   | CMOS | PTM output   |
| PB2/PTCK/PTPB         | PB2      | PBPU<br>PBS0          | ST  | CMOS | General purpose I/O. Register enabled pull-high.             |
|                       | PTCK     | PBS0<br>IFS0          | ST  | —    | PTM clock input  |
|                       | PTPB     | PBS0                  | —   | CMOS | PTM inverted output  |
| PB3/CTP               | PB3      | PBPU<br>PBS0          | ST  | CMOS | General purpose I/O. Register enabled pull-high.             |
|                       | CTP      | PBS0                  | —   | CMOS | CTM output   |
| PB4/CTCK/CTPB         | PB4      | PBPU<br>PBS1          | ST  | CMOS | General purpose I/O. Register enabled pull-high.             |
|                       | CTCK     | PBS1<br>IFS0          | ST  | —    | CTM clock input  |
|                       | CTPB     | PBS1                  | —   | CMOS | CTM inverted output  |
| PB5                   | PB5      | PBPU<br>PBS1          | ST  | CMOS | General purpose I/O. Register enabled pull-high.             |
| PB6/OSC1              | PB6      | PBPU<br>PBS1          | ST  | CMOS | General purpose I/O. Register enabled pull-high.             |
|                       | OSC1     | PBS1                  | HXT | —    | HXT oscillator pin   |
| PB7/OSC2              | PB7      | PBPU<br>PBS1          | ST  | CMOS | General purpose I/O. Register enabled pull-high.             |
|                       | OSC2     | PBS1                  | —   | HXT  | HXT oscillator pin   |
| PC0/AN0/VREF1         | PC0      | PCPU<br>PCS0          | ST  | CMOS | General purpose I/O. Register enabled pull-high.             |
|                       | AN0      | PCS0                  | AN  | —    | A/D Converter analog input                                   |
|                       | VREF1    | PCS0<br>SADC2         | AN  | —    | A/D Converter PGA voltage input                              |
| PC1/AN1/CMPO/<br>VREF | PC1      | PCPU<br>PCS0          | ST  | CMOS | General purpose I/O. Register enabled pull-high.             |
|                       | AN1      | PCS0                  | AN  | —    | A/D Converter analog input                                   |
|                       | CMPO     | PCS0<br>CMPC          | —   | CMOS | Comparator output  |
|                       | VREF     | PCS0<br>SADC2         | AN  | —    | A/D Converter reference voltage input                        |

| Pin Name              | Function | OPT          | I/T | O/T  | Description                                      |
|-----------------------|----------|--------------|-----|------|--|
| PC2/PTPI/PTP/<br>AN2  | PC2      | PCPU<br>PCS0 | ST  | CMOS | General purpose I/O. Register enabled pull-high. |
|                       | PTPI     | PCS0<br>IFS0 | ST  | —    | PTM capture input                                |
|                       | PTP      | PCS0         | —   | CMOS | PTM output                                       |
|                       | AN2      | PCS0         | AN  | —    | A/D Converter analog input                       |
| PC3/PTCK/PTPB/<br>AN3 | PC3      | PCPU<br>PCS0 | ST  | CMOS | General purpose I/O. Register enabled pull-high. |
|                       | PTCK     | PCS0<br>IFS0 | ST  | —    | PTM clock input                                  |
|                       | PTPB     | PCS0         | —   | CMOS | PTM inverted output                              |
| PC4/AN4               | AN3      | PCS0         | AN  | —    | A/D Converter analog input                       |
|                       | PC4      | PCPU<br>PCS1 | ST  | CMOS | General purpose I/O. Register enabled pull-high. |
| PC5/AN5               | AN4      | PCS1         | AN  | —    | A/D Converter analog input                       |
|                       | PC5      | PCPU<br>PCS1 | ST  | CMOS | General purpose I/O. Register enabled pull-high. |
| PC6/STPI/STP/<br>AN6  | AN5      | PCS1         | AN  | —    | A/D Converter analog input                       |
|                       | PC6      | PCPU<br>PCS1 | ST  | CMOS | General purpose I/O. Register enabled pull-high. |
|                       | STPI     | PCS1<br>IFS0 | ST  | —    | STM capture input                                |
|                       | STP      | PCS1         | —   | CMOS | STM output                                       |
| PC7/STCK/STPB/<br>AN7 | AN6      | PCS1         | AN  | —    | A/D Converter analog input                       |
|                       | PC7      | PCPU<br>PCS1 | ST  | CMOS | General purpose I/O. Register enabled pull-high. |
|                       | STCK     | PCS1<br>IFS0 | ST  | —    | STM clock input                                  |
|                       | STPB     | PCS1         | —   | CMOS | STM inverted output                              |
| PD0/AN8               | AN7      | PCS1         | AN  | —    | A/D Converter analog input                       |
|                       | PD0      | PDP<br>PDS0  | ST  | CMOS | General purpose I/O. Register enabled pull-high. |
| PD1/AN9               | AN8      | PDS0         | AN  | —    | A/D Converter analog input                       |
|                       | PD1      | PDP<br>PDS0  | ST  | CMOS | General purpose I/O. Register enabled pull-high. |
| PD2/AN10              | AN9      | PDS0         | AN  | —    | A/D Converter analog input                       |
|                       | PD2      | PDP<br>PDS0  | ST  | CMOS | General purpose I/O. Register enabled pull-high. |
| PD3/AN11              | AN10     | PDS0         | AN  | —    | A/D Converter analog input                       |
|                       | PD3      | PDP<br>PDS0  | ST  | CMOS | General purpose I/O. Register enabled pull-high. |
| PD4/AN12              | AN11     | PDS0         | AN  | —    | A/D Converter analog input                       |
|                       | PD4      | PDP<br>PDS1  | ST  | CMOS | General purpose I/O. Register enabled pull-high. |
| PD5/AN13              | AN12     | PDS1         | AN  | —    | A/D Converter analog input                       |
|                       | PD5      | PDP<br>PDS1  | ST  | CMOS | General purpose I/O. Register enabled pull-high. |
| PD6/AN14              | AN13     | PDS1         | AN  | —    | A/D Converter analog input                       |
|                       | PD6      | PDP<br>PDS1  | ST  | CMOS | General purpose I/O. Register enabled pull-high. |
| AN14                  | AN14     | PDS1         | AN  | —    | A/D Converter analog input                       |

| Pin Name                            | Function                | OPT           | I/T | O/T  | Description                                      |
|-------------------------------------|-------------------------|---------------|-----|------|--|
| PE0/STCK/STPB                       | PE0                     | PEPU<br>PES0  | ST  | CMOS | General purpose I/O. Register enabled pull-high. |
|                                     | STCK                    | PES0<br>IFS0  | ST  | —    | STM clock input                                  |
|                                     | STPB                    | PES0          | —   | CMOS | STM inverted output                              |
| PE1/STPI/STP                        | PE1                     | PEPU<br>PES0  | ST  | CMOS | General purpose I/O. Register enabled pull-high. |
|                                     | STPI                    | PES0<br>IFS0  | ST  | —    | STM capture input                                |
|                                     | STP                     | PES0          | —   | CMOS | STM output                                       |
| PE2/CTCK/CTPB                       | PE2                     | PEPU<br>PES0  | ST  | CMOS | General purpose I/O. Register enabled pull-high. |
|                                     | CTCK                    | PES0<br>IFS0  | ST  | —    | CTM clock input                                  |
|                                     | CTPB                    | PES0          | —   | CMOS | CTM inverted output                              |
| PE3/CTP                             | PE3                     | PEPU<br>PES0  | ST  | CMOS | General purpose I/O. Register enabled pull-high. |
|                                     | CTP                     | PES0          | ST  | —    | CTM clock input                                  |
| PE4                                 | PE4                     | PEPU<br>PES1  | ST  | CMOS | General purpose I/O. Register enabled pull-high. |
| PF0/ $\overline{\text{SCS}}$ /SCOM0 | PF0                     | PFPU<br>PFS0  | ST  | CMOS | General purpose I/O. Register enabled pull-high. |
|                                     | $\overline{\text{SCS}}$ | PFS0<br>IFS0  | ST  | CMOS | SPI slave select                                 |
|                                     | SCOM0                   | PFS0<br>SCOMC | —   | SCOM | Software controlled LCD common output            |
| PF1/SDO/SCOM1                       | PF1                     | PFPU<br>PFS0  | ST  | CMOS | General purpose I/O. Register enabled pull-high. |
|                                     | SDO                     | PFS0          | —   | CMOS | SPI data output                                  |
|                                     | SCOM1                   | PFS0<br>SCOMC | —   | SCOM | Software controlled LCD common output            |
| PF2/SDI/SDA/<br>SCOM2               | PF2                     | PFPU<br>PFS0  | ST  | CMOS | General purpose I/O. Register enabled pull-high. |
|                                     | SDI                     | PFS0<br>IFS0  | ST  | —    | SPI data input                                   |
|                                     | SDA                     | PFS0<br>IFS0  | ST  | NMOS | I <sup>2</sup> C data line                       |
|                                     | SCOM2                   | PFS0<br>SCOMC | —   | SCOM | Software controlled LCD common output            |
| PF3/SCK/SCL/<br>SCOM3               | PF3                     | PFPU<br>PFS0  | ST  | CMOS | General purpose I/O. Register enabled pull-high. |
|                                     | SCK                     | PFS0<br>IFS0  | ST  | CMOS | SPI serial Clock                                 |
|                                     | SCL                     | PFS0<br>IFS0  | ST  | NMOS | I <sup>2</sup> C clock line                      |
|                                     | SCOM3                   | PFS0<br>SCOMC | —   | SCOM | Software controlled LCD common output            |
| PF4/XT2                             | PF4                     | PFPU<br>PFS1  | ST  | CMOS | General purpose I/O. Register enabled pull-high. |
|                                     | XT2                     | PFS1          | —   | LXT  | LXT oscillator pin                               |
| PF5/XT1                             | PF5                     | PFPU<br>PFS1  | ST  | CMOS | General purpose I/O. Register enabled pull-high. |
|                                     | XT1                     | PFS1          | LXT | —    | LXT oscillator pin                               |

| Pin Name    | Function | OPT                    | I/T | O/T  | Description                                      |
|-------------|----------|------------------------|-----|------|--|
| PF6/AN15/C- | PF6      | PFPU<br>PFS1           | ST  | CMOS | General purpose I/O. Register enabled pull-high. |
|             | AN15     | PFS1                   | AN  | —    | A/D Converter analog input                       |
|             | C-       | PFS1<br>CMPC<br>CMPVOS | AN  | —    | Comparator negative input                        |
| PF7/C+      | PF7      | PFPU<br>PFS1           | ST  | CMOS | General purpose I/O. Register enabled pull-high. |
|             | C+       | PFS1<br>CMPC<br>CMPVOS | AN  | —    | Comparator positive input                        |
| VDD         | VDD      | —                      | PWR | —    | Power Supply                                     |
| AVDD        | AVDD     | —                      | PWR | —    | A/D Converter Power Supply                       |
| VSS         | VSS      | —                      | PWR | —    | Ground   |
| AVSS        | AVSS     | —                      | PWR | —    | A/D Converter Ground                             |

Legend: I/T: Input type; O/T: Output type;  
 OP: Optional by register option;  
 PWR: Power; ST: Schmitt Trigger input;  
 CMOS: CMOS output; NMOS: NMOS output;  
 SCOM: Software controlled LCD COM; AN: Analog signal;  
 HXT: High frequency crystal oscillator; LXT: Low frequency crystal oscillator

### Absolute Maximum Ratings

|                               |                                |
|-------------------------------|--------------------------------|
| Supply Voltage .....          | $V_{SS}-0.3V$ to $V_{SS}+6.0V$ |
| Input Voltage .....           | $V_{SS}-0.3V$ to $V_{DD}+0.3V$ |
| Storage Temperature.....      | -50°C to 125°C                 |
| Operating Temperature.....    | -40°C to 85°C                  |
| I <sub>OL</sub> Total .....   | 80mA                           |
| I <sub>OH</sub> 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 this device at other conditions beyond those listed in the specification is not implied and prolonged exposure to extreme conditions may affect device reliability.

### D.C. Characteristics

Ta=25°C

| Symbol                   | Parameter                | Test Conditions                            |   | Min. | Typ. | Max. | Unit |
|--------------------------|--------------------------|--|---|------|------|------|------|
|                          |                          | V <sub>DD</sub>                            | Conditions                                    |      |      |      |      |
| V <sub>DD</sub>          | Operating Voltage (HXT)  | —  | f <sub>SYS</sub> =f <sub>HXT</sub> =4MHz      | 1.8  | —    | 5.5  | V    |
|                          |                          |  | f <sub>SYS</sub> =f <sub>HXT</sub> =8MHz      | 2.0  | —    | 5.5  | V    |
|                          |                          |  | f <sub>SYS</sub> =f <sub>HXT</sub> =12MHz     | 2.7  | —    | 5.5  | V    |
|                          |                          |  | f <sub>SYS</sub> =f <sub>HXT</sub> =16MHz     | 3.3  | —    | 5.5  | V    |
|                          | Operating Voltage (HIRC) | —  | f <sub>SYS</sub> =f <sub>HIRC</sub> =4MHz     | 1.8  | —    | 5.5  | V    |
|                          |                          |  | f <sub>SYS</sub> =f <sub>HIRC</sub> =8MHz     | 2.0  | —    | 5.5  | V    |
|                          |                          |  | f <sub>SYS</sub> =f <sub>HIRC</sub> =12MHz    | 2.7  | —    | 5.5  | V    |
|                          | Operating Voltage (LXT)  | —  | f <sub>SYS</sub> =f <sub>LXT</sub> =32.768kHz | 1.8  | —    | 5.5  | V    |
| Operating Voltage (LIRC) | —                        | f <sub>SYS</sub> =f <sub>LIRC</sub> =32kHz | 1.8   | —    | 5.5  | V    |      |

| Symbol           | Parameter                                      | Test Conditions  |   | Min. | Typ. | Max. | Unit |
|------------------|--|--|---|------|------|------|------|
|                  |  | V <sub>DD</sub>  | Conditions  |      |      |      |      |
| I <sub>DD</sub>  | Operating Current (HXT)                        | 3V   | No load, all peripherals off,                                   | —    | 0.5  | 0.8  | mA   |
|                  |  | 5V   | f <sub>sys</sub> =f <sub>HXT</sub> =4MHz                        | —    | 1.0  | 1.5  | mA   |
|                  |  | 3V   | No load, all peripherals off,                                   | —    | 0.8  | 1.2  | mA   |
|                  |  | 5V   | f <sub>sys</sub> =f <sub>HXT</sub> =8MHz                        | —    | 1.6  | 2.4  | mA   |
|                  |  | 3V   | No load, all peripherals off,                                   | —    | 1.2  | 1.8  | mA   |
|                  |  | 5V   | f <sub>sys</sub> =f <sub>HXT</sub> =12MHz                       | —    | 2.4  | 3.6  | mA   |
|                  | Operating Current (HIRC)                       | 3V   | No load, all peripherals off,                                   | —    | 0.5  | 0.8  | mA   |
|                  |  | 5V   | f <sub>sys</sub> =f <sub>HIRC</sub> =4MHz                       | —    | 1.0  | 1.5  | mA   |
|                  |  | 3V   | No load, all peripherals off,                                   | —    | 0.8  | 1.2  | mA   |
|                  |  | 5V   | f <sub>sys</sub> =f <sub>HIRC</sub> =8MHz                       | —    | 1.6  | 2.4  | mA   |
|                  |  | 3V   | No load, all peripherals off,                                   | —    | 1.2  | 1.8  | mA   |
|                  |  | 5V   | f <sub>sys</sub> =f <sub>HIRC</sub> =12MHz                      | —    | 2.4  | 3.6  | mA   |
|                  | Operating Current (LXT)                        | 3V   | No load, all peripherals off,                                   | —    | 10   | 20   | μA   |
|                  |  | 5V   | f <sub>sys</sub> =f <sub>LXT</sub> =32768Hz                     | —    | 30   | 50   | μA   |
|                  | Operating Current (LIRC)                       | 3V   | No load, all peripherals off,                                   | —    | 10   | 20   | μA   |
|                  |  | 5V   | f <sub>sys</sub> =f <sub>LIRC</sub> =32kHz                      | —    | 30   | 50   | μA   |
|                  | Operating Current, f <sub>H</sub> =8MHz (HIRC) | 3V   | No load, all peripherals off,                                   | —    | 0.5  | 1.0  | mA   |
|                  |  | 5V   | f <sub>sys</sub> =f <sub>H</sub> /2                             | —    | 1.0  | 2.0  | mA   |
|                  |  | 3V   | No load, all peripherals off,                                   | —    | 0.25 | 0.5  | mA   |
|                  |  | 5V   | f <sub>sys</sub> =f <sub>H</sub> /64                            | —    | 0.5  | 1.0  | mA   |
|                  | Operating Current, f <sub>H</sub> =12MHz (HXT) | 3V   | No load, all peripherals off,                                   | —    | 0.7  | 1.4  | mA   |
|                  |  | 5V   | f <sub>sys</sub> =f <sub>H</sub> /2                             | —    | 1.4  | 2.8  | mA   |
|                  |  | 3V   | No load, all peripherals off,                                   | —    | 0.35 | 0.7  | mA   |
|                  |  | 5V   | f <sub>sys</sub> =f <sub>H</sub> /64                            | —    | 0.7  | 1.4  | mA   |
| I <sub>STB</sub> | Standby Current (SLEEP Mode)                   | 3V   | No load, all peripherals off,                                   | —    | 0.2  | 0.8  | μA   |
|                  |  | 5V   | WDT off   | —    | 0.5  | 1.0  | μA   |
|                  |  | 3V   | No load, all peripherals off,                                   | —    | —    | 3    | μA   |
|                  |  | 5V   | WDT on  | —    | —    | 5    | μA   |
|                  | Standby Current (IDLE0 Mode)                   | 3V   | No load, all peripherals off,                                   | —    | 3    | 5    | μA   |
|                  |  | 5V   | f <sub>SUB</sub> on   | —    | 5    | 10   | μA   |
|                  | Standby Current (IDLE1 Mode, HIRC)             | 3V   | No load, all peripherals off,                                   | —    | 0.25 | 0.5  | mA   |
|                  |  | 5V   | f <sub>SUB</sub> on, f <sub>sys</sub> =f <sub>HIRC</sub> =4MHz  | —    | 0.5  | 1.0  | mA   |
|                  |  | 3V   | No load, all peripherals off,                                   | —    | 0.5  | 1.0  | mA   |
|                  |  | 5V   | f <sub>SUB</sub> on, f <sub>sys</sub> =f <sub>HIRC</sub> =8MHz  | —    | 1.0  | 2.0  | mA   |
|                  |  | 3V   | No load, all peripherals off,                                   | —    | 0.7  | 1.4  | mA   |
|                  |  | 5V   | f <sub>SUB</sub> on, f <sub>sys</sub> =f <sub>HIRC</sub> =12MHz | —    | 1.4  | 2.8  | mA   |
|                  | Standby Current (IDLE1 Mode, HXT)              | 3V   | No load, all peripherals off,                                   | —    | 0.25 | 0.5  | mA   |
|                  |  | 5V   | f <sub>SUB</sub> on, f <sub>sys</sub> =f <sub>HXT</sub> =4MHz   | —    | 0.5  | 1.0  | mA   |
|                  |  | 3V   | No load, all peripherals off,                                   | —    | 0.5  | 1.0  | mA   |
|                  |  | 5V   | f <sub>SUB</sub> on, f <sub>sys</sub> =f <sub>HXT</sub> =8MHz   | —    | 1.0  | 2.0  | mA   |
|                  |  | 3V   | No load, all peripherals off,                                   | —    | 0.7  | 1.4  | mA   |
|                  |  | 5V   | f <sub>SUB</sub> on, f <sub>sys</sub> =f <sub>HXT</sub> =12MHz  | —    | 1.5  | 3.0  | mA   |
| 5V               | No load, all peripherals off,                  | —  | 2.0   | 4.0  | mA   |      |      |
|                  |  | f <sub>SUB</sub> on, f <sub>sys</sub> =f <sub>HXT</sub> =16MHz |   |      |      |      |      |

| 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    |
| 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>    | V    |
| I <sub>OL</sub> | Sink Current for I/O Port   | 1.8V            | V <sub>OL</sub> =0.1V <sub>DD</sub>   | 7                  | 14   | —                  | mA   |
|                 |   | 3V              | V <sub>OL</sub> =0.1V <sub>DD</sub>   | 16                 | 32   | —                  | mA   |
|                 |   | 5V              | V <sub>OL</sub> =0.1V <sub>DD</sub>   | 32                 | 64   | —                  | mA   |
| I <sub>OH</sub> | Source Current for I/O Port   | 3V              | V <sub>OH</sub> =0.9V <sub>DD</sub> , SLEDC <sub>n</sub> [m+1, m]=00B (n=0, 1, 2, m=0 or 2 or 4 or 6) | -1.0               | -2.0 | —                  | mA   |
|                 |   | 5V              | V <sub>OH</sub> =0.9V <sub>DD</sub> , SLEDC <sub>n</sub> [m+1, m]=00B (n=0, 1, 2, m=0 or 2 or 4 or 6) | -2.0               | -4.0 | —                  | mA   |
|                 |   | 3V              | V <sub>OH</sub> =0.9V <sub>DD</sub> , SLEDC <sub>n</sub> [m+1, m]=01B (n=0, 1, 2, m=0 or 2 or 4 or 6) | -1.75              | -3.5 | —                  | mA   |
|                 |   | 5V              | V <sub>OH</sub> =0.9V <sub>DD</sub> , SLEDC <sub>n</sub> [m+1, m]=01B (n=0, 1, 2, m=0 or 2 or 4 or 6) | -3.5               | -7.0 | —                  | mA   |
|                 |   | 3V              | V <sub>OH</sub> =0.9V <sub>DD</sub> , SLEDC <sub>n</sub> [m+1, m]=10B (n=0, 1, 2, m=0 or 2 or 4 or 6) | -2.5               | -5.0 | —                  | mA   |
|                 |   | 5V              | V <sub>OH</sub> =0.9V <sub>DD</sub> , SLEDC <sub>n</sub> [m+1, m]=10B (n=0, 1, 2, m=0 or 2 or 4 or 6) | -5.0               | -10  | —                  | mA   |
|                 |   | 3V              | V <sub>OH</sub> =0.9V <sub>DD</sub> , SLEDC <sub>n</sub> [m+1, m]=11B (n=0, 1, 2, m=0 or 2 or 4 or 6) | -5.5               | -11  | —                  | mA   |
| 5V              | V <sub>OH</sub> =0.9V <sub>DD</sub> , SLEDC <sub>n</sub> [m+1, m]=11B (n=0, 1, 2, m=0 or 2 or 4 or 6) | -11             | -22   | —                  | mA   |                    |      |
| R <sub>PH</sub> | Pull-high Resistance for I/O Ports  | 3V              | LVPU=0  | 20                 | 60   | 100                | kΩ   |
|                 |   | 5V              | LVPU=0  | 10                 | 30   | 50                 | kΩ   |
|                 |   | 3V              | LVPU=1  | 6.67               | 15   | 23                 | kΩ   |
|                 |   | 5V              | LVPU=1  | 3.5                | 7.5  | 12                 | kΩ   |

## A.C. Characteristics

T<sub>a</sub>=25°C

| Symbol           | Parameter           | Test Conditions |   | Min. | Typ.   | Max. | Unit |
|------------------|---------------------|-----------------|---|------|--------|------|------|
|                  |                     | V <sub>DD</sub> | Conditions                                    |      |        |      |      |
| f <sub>sys</sub> | System Clock (HXT)  | 1.8V~5.5V       | f <sub>sys</sub> =f <sub>HXT</sub> =4MHz      | —    | 4      | —    | MHz  |
|                  |                     | 2.0V~5.5V       | f <sub>sys</sub> =f <sub>HXT</sub> =8MHz      | —    | 8      | —    | MHz  |
|                  |                     | 2.7V~5.5V       | f <sub>sys</sub> =f <sub>HXT</sub> =12MHz     | —    | 12     | —    | MHz  |
|                  |                     | 3.3V~5.5V       | f <sub>sys</sub> =f <sub>HXT</sub> =16MHz     | —    | 16     | —    | MHz  |
|                  | System Clock (HIRC) | 1.8V~5.5V       | f <sub>sys</sub> =f <sub>HIRC</sub> =4MHz     | —    | 4      | —    | MHz  |
|                  |                     | 2.0V~5.5V       | f <sub>sys</sub> =f <sub>HIRC</sub> =8MHz     | —    | 8      | —    | MHz  |
|                  |                     | 2.7V~5.5V       | f <sub>sys</sub> =f <sub>HIRC</sub> =12MHz    | —    | 12     | —    | MHz  |
|                  | System Clock (LXT)  | 1.8V~5.5V       | f <sub>sys</sub> =f <sub>LXT</sub> =32.768kHz | —    | 32.768 | —    | kHz  |
|                  | System Clock (LIRC) | 1.8V~5.5V       | f <sub>sys</sub> =f <sub>LIRC</sub> =32kHz    | —    | 32     | —    | kHz  |

| Symbol            | Parameter   | Test Conditions |                | Min. | Typ. | Max. | Unit |
|-------------------|---|-----------------|----------------|------|------|------|------|
|                   |   | V <sub>DD</sub> | Conditions     |      |      |      |      |
| f <sub>HIRC</sub> | High Speed Internal RC Oscillator<br>(HIRC=4MHz, trim 4MHz @ V <sub>DD</sub> =3V)   | 3.0V            | Ta=25°C        | -2%  | 4    | +2%  | MHz  |
|                   |   | 2.2V~5.5V       | Ta=25°C        | -5%  | 4    | 5%   | MHz  |
|                   |   | 3.0V            | Ta=0°C~70°C    | -5%  | 4    | +5%  | MHz  |
|                   |   | 3.0V            | Ta= -40°C~85°C | -5%  | 4    | 5%   | MHz  |
|                   |   | 2.2V~5.5V       | Ta=0°C~70°C    | -7%  | 4    | +7%  | MHz  |
|                   |   | 2.2V~5.5V       | Ta= -40°C~85°C | -10% | 4    | +10% | MHz  |
|                   | High Speed Internal RC Oscillator<br>(HIRC=4MHz, trim 4MHz @ V <sub>DD</sub> =5V)   | 5.0V            | Ta=25°C        | -2%  | 4    | +2%  | MHz  |
|                   |   | 2.2V~5.5V       | Ta=25°C        | -5%  | 4    | 5%   | MHz  |
|                   |   | 5.0V            | Ta=0°C~70°C    | -5%  | 4    | +5%  | MHz  |
|                   |   | 5.0V            | Ta= -40°C~85°C | -5%  | 4    | 5%   | MHz  |
|                   |   | 2.2V~5.5V       | Ta=0°C~70°C    | -7%  | 4    | +7%  | MHz  |
|                   |   | 2.2V~5.5V       | Ta= -40°C~85°C | -10% | 4    | +10% | MHz  |
|                   | High Speed Internal RC Oscillator<br>(HIRC=8MHz, trim 8MHz @ V <sub>DD</sub> =3V)   | 5.0V            | Ta=25°C        | -20% | 8    | +20% | MHz  |
|                   |   | 5.0V            | Ta=25°C        | -20% | 12   | +20% | MHz  |
|                   |   | 3.0V            | Ta=25°C        | -2%  | 8    | +2%  | MHz  |
|                   |   | 3.0V~5.5V       | Ta=25°C        | -5%  | 8    | 5%   | MHz  |
|                   |   | 3.0V            | Ta=0°C~70°C    | -5%  | 8    | +5%  | MHz  |
|                   |   | 3.0V            | Ta= -40°C~85°C | -5%  | 8    | 5%   | MHz  |
|                   |   | 3.0V~5.5V       | Ta=0°C~70°C    | -7%  | 8    | +7%  | MHz  |
|                   |   | 3.0V~5.5V       | Ta= -40°C~85°C | -10% | 8    | +10% | MHz  |
|                   | High Speed Internal RC Oscillator<br>(HIRC=8MHz, trim 8MHz @ V <sub>DD</sub> =5V)   | 3.0V            | Ta=25°C        | -20% | 4    | +20% | MHz  |
|                   |   | 3.0V            | Ta=25°C        | -20% | 12   | +20% | MHz  |
|                   |   | 5.0V            | Ta=25°C        | -2%  | 8    | +2%  | MHz  |
|                   |   | 3.0V~5.5V       | Ta=25°C        | -5%  | 8    | 5%   | MHz  |
|                   |   | 5.0V            | Ta=0°C~70°C    | -5%  | 8    | +5%  | MHz  |
|                   |   | 5.0V            | Ta= -40°C~85°C | -5%  | 8    | 5%   | MHz  |
|                   |   | 3.0V~5.5V       | Ta=0°C~70°C    | -7%  | 8    | +7%  | MHz  |
|                   |   | 3.0V~5.5V       | Ta= -40°C~85°C | -10% | 8    | +10% | MHz  |
|                   | High Speed Internal RC Oscillator<br>(HIRC=12MHz, trim 12MHz @ V <sub>DD</sub> =5V) | 5.0V            | Ta=25°C        | -20% | 4    | +20% | MHz  |
|                   |   | 5.0V            | Ta=25°C        | -20% | 12   | +20% | MHz  |
|                   |   | 5.0V            | Ta=25°C        | -2%  | 12   | +2%  | MHz  |
|                   |   | 4.0V~5.5V       | Ta=25°C        | -5%  | 12   | 5%   | MHz  |
|                   |   | 5.0V            | Ta=0°C~70°C    | -5%  | 12   | +5%  | MHz  |
|                   |   | 5.0V            | Ta= -40°C~85°C | -5%  | 12   | 5%   | MHz  |
|                   |   | 4.0V~5.5V       | Ta=0°C~70°C    | -7%  | 12   | +7%  | MHz  |
|                   |   | 4.0V~5.5V       | Ta= -40°C~85°C | -10% | 12   | +10% | MHz  |



| Symbol  | Parameter  | Test Conditions   |  | Min. | Typ. | Max.             | Unit              |
|---|--|---|--|------|------|------------------|-------------------|
|   |  | V <sub>DD</sub>   | Conditions   |      |      |                  |                   |
| f <sub>LIRC</sub>   | Low Speed Internal RC Oscillator (LIRC)  | 2.2V~5.5V   | Ta=25°C  | -5%  | 32   | +5%              | kHz               |
|   |  |   | Ta= -40°C~85°C   | -10% | 32   | +10%             | kHz               |
| t <sub>TCK</sub>  | CTCK, STCK and PTCK Pin Minimum Pulse Width  | —   | —  | 0.3  | —    | —                | µs                |
| t <sub>TPI</sub>  | STPI, PTPI Pin Minimum Pulse Width   | —   | —  | 0.3  | —    | —                | µs                |
| t <sub>INT</sub>  | External Interrupt Minimum Pulse Width   | —   | —  | 10   | —    | —                | µs                |
| t <sub>SRESET</sub>   | Minimum Software Reset Width to Reset  | —   | —  | 45   | 90   | 120              | µs                |
| t <sub>EERD</sub>   | EEPROM Read Time   | —   | —  | —    | —    | 4                | t <sub>SYS</sub>  |
| t <sub>EEWR</sub>   | EEPROM Write Time  | —   | —  | —    | 2    | 4                | ms                |
| t <sub>RSTD</sub>   | System Reset Delay Time (POR Reset, LVR Hardware Reset, LVR Software Reset, WDT Software Reset, Reset Control Register Software Reset)                                       | —   | —  | 25   | 50   | 100              | ms                |
|   | System Reset Delay Time (WDT Time-out Hardware Cold Reset)   | —   | —  | 8.3  | 16.7 | 33.3             | ms                |
| t <sub>SST</sub>  | System Start-up Timer Period (Wake-up from Power Down Mode and f <sub>SYS</sub> Off)   | —   | f <sub>SYS</sub> =f <sub>SUB</sub> =f <sub>LXT</sub>   | 1024 | —    | —                | t <sub>LXT</sub>  |
|   |  | —   | f <sub>SYS</sub> =f <sub>H</sub> ~f <sub>H</sub> / 64, f <sub>H</sub> =f <sub>HXT</sub>                        | 128  | —    | —                | t <sub>HXT</sub>  |
|   |  | —   | 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 Timer Period (Slow Mode ↔ Normal Mode, or f <sub>H</sub> =f <sub>HIRC</sub> ↔ f <sub>HXT</sub> , or f <sub>SUB</sub> =f <sub>LIRC</sub> ↔ f <sub>LXT</sub> ) | —   | f <sub>HXT</sub> off → on (HXTF=1)   | 1024 | —    | —                | t <sub>HXT</sub>  |
|   |  | —   | f <sub>HIRC</sub> off → on (HIRCF=1)   | 16   | —    | —                | t <sub>HIRC</sub> |
|   |  | —   | f <sub>LXT</sub> off → on (LXTF=1)   | 1024 | —    | —                | t <sub>LXT</sub>  |
|   | System Start-up Timer Period (Wake-up from Power Down Mode and f <sub>SYS</sub> On)  | —   | f <sub>SYS</sub> =f <sub>H</sub> ~f <sub>H</sub> / 64, f <sub>SYS</sub> =f <sub>HXT</sub> or f <sub>HIRC</sub> | 2    | —    | —                | t <sub>H</sub>    |
| —   |  | f <sub>SYS</sub> =f <sub>LXT</sub> or f <sub>LIRC</sub> | 2  | —    | —    | t <sub>SUB</sub> |                   |
| System Start-up Timer Period (WDT Time-out Hardware Cold Reset) | —  | —   | 0  | —    | —    | t <sub>H</sub>   |                   |

Note: 1. t<sub>SYS</sub>=1/f<sub>SYS</sub>.

- To maintain the accuracy of the internal HIRC oscillator frequency, a 0.1µF decoupling capacitor should be connected between VDD and VSS and located as close to the device as possible.

## A/D Converter Electrical Characteristics

Ta=25°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>ADI</sub>   | Input Voltage                                      | —               | —  | 0    | —    | V <sub>REF</sub> | V                 |
| V <sub>REF</sub>   | Reference Voltage                                  | —               | —  | 1.8  | —    | V <sub>DD</sub>  | V                 |
| DNL                | Differential Nonlinearity                          | 1.8V            | SAINS[3:0]=0000B, SAVRS[1:0]=01B, V <sub>REF</sub> =V <sub>DD</sub> , t <sub>ADCK</sub> =2.0μs | —    | —    | ±3               | LSB               |
|                    |  | 2V              | SAINS[3:0]=0000B, SAVRS[1:0]=01B, V <sub>REF</sub> =V <sub>DD</sub> , t <sub>ADCK</sub> =0.5μs |      |      |                  |                   |
|                    |  | 3V              | SAINS[3:0]=0000B, SAVRS[1:0]=01B, V <sub>REF</sub> =V <sub>DD</sub> , t <sub>ADCK</sub> =0.5μs |      |      |                  |                   |
|                    |  | 5V              | SAINS[3:0]=0000B, SAVRS[1:0]=01B, V <sub>REF</sub> =V <sub>DD</sub> , t <sub>ADCK</sub> =0.5μs |      |      |                  |                   |
|                    |  | 1.8V            | SAINS[3:0]=0000B, SAVRS[1:0]=01B, V <sub>REF</sub> =V <sub>DD</sub> , t <sub>ADCK</sub> =10μs  |      |      |                  |                   |
|                    |  | 3V              | SAINS[3:0]=0000B, SAVRS[1:0]=01B, V <sub>REF</sub> =V <sub>DD</sub> , t <sub>ADCK</sub> =10μs  |      |      |                  |                   |
|                    |  | 5V              | SAINS[3:0]=0000B, SAVRS[1:0]=01B, V <sub>REF</sub> =V <sub>DD</sub> , t <sub>ADCK</sub> =10μs  |      |      |                  |                   |
| INL                | Integral Nonlinearity                              | 1.8V            | SAINS[3:0]=0000B, SAVRS[1:0]=01B, V <sub>REF</sub> =V <sub>DD</sub> , t <sub>ADCK</sub> =2.0μs | —    | —    | ±4               | LSB               |
|                    |  | 2V              | SAINS[3:0]=0000B, SAVRS[1:0]=01B, V <sub>REF</sub> =V <sub>DD</sub> , t <sub>ADCK</sub> =0.5μs |      |      |                  |                   |
|                    |  | 3V              | SAINS[3:0]=0000B, SAVRS[1:0]=01B, V <sub>REF</sub> =V <sub>DD</sub> , t <sub>ADCK</sub> =0.5μs |      |      |                  |                   |
|                    |  | 5V              | SAINS[3:0]=0000B, SAVRS[1:0]=01B, V <sub>REF</sub> =V <sub>DD</sub> , t <sub>ADCK</sub> =0.5μs |      |      |                  |                   |
|                    |  | 1.8V            | SAINS[3:0]=0000B, SAVRS[1:0]=01B, V <sub>REF</sub> =V <sub>DD</sub> , t <sub>ADCK</sub> =10μs  |      |      |                  |                   |
|                    |  | 3V              | SAINS[3:0]=0000B, SAVRS[1:0]=01B, V <sub>REF</sub> =V <sub>DD</sub> , t <sub>ADCK</sub> =10μs  |      |      |                  |                   |
|                    |  | 5V              | SAINS[3:0]=0000B, SAVRS[1:0]=01B, V <sub>REF</sub> =V <sub>DD</sub> , t <sub>ADCK</sub> =10μs  |      |      |                  |                   |
| I <sub>ADC</sub>   | Additional Current for ADC Enable                  | 1.8V            | No load (t <sub>ADCK</sub> =2.0μs)   | —    | 0.5  | 1.0              | mA                |
|                    |  | 3V              | No load (t <sub>ADCK</sub> =0.5μs)   | —    | 1.0  | 2.0              | mA                |
|                    |  | 5V              | No load (t <sub>ADCK</sub> =0.5μs)   | —    | 1.5  | 3.0              | mA                |
| t <sub>ADCK</sub>  | Clock Period                                       | —               | 1.8V ≤ V <sub>DD</sub> < 2.0V  | 2.0  | —    | 10               | μs                |
|                    |  | —               | 2.0V ≤ V <sub>DD</sub> ≤ 5.5V  | 0.5  | —    | 10               | μs                |
| t <sub>ON2ST</sub> | ADC on to ADC Start                                | —               | —  | 4    | —    | —                | μs                |
| t <sub>ADS</sub>   | Sampling Time                                      | —               | —  | —    | 4    | —                | t <sub>ADCK</sub> |
| t <sub>ADC</sub>   | Conversion Time (Include ADC Sample and Hold Time) | —               | —  | —    | 16   | —                | t <sub>ADCK</sub> |

| Symbol           | Parameter                         | Test Conditions |            | Min.                 | Typ. | Max.                 | Unit |
|------------------|-----------------------------------|-----------------|------------|----------------------|------|----------------------|------|
|                  |                                   | V <sub>DD</sub> | Conditions |                      |      |                      |      |
| I <sub>PGA</sub> | Additional Current for PGA Enable | 2.2V            | No load    | —                    | 250  | 400                  | μA   |
|                  |                                   | 3V              | No load    | —                    | 300  | 450                  | μA   |
|                  |                                   | 5V              | No load    | —                    | 400  | 550                  | μA   |
| V <sub>CM</sub>  | PGA Common Mode Voltage Range     | 2.2V            | —          | V <sub>SS</sub>      | —    | V <sub>DD</sub> -1.4 | V    |
|                  |                                   | 3V              | —          | V <sub>SS</sub>      | —    | V <sub>DD</sub> -1.4 | V    |
|                  |                                   | 5V              | —          | V <sub>SS</sub>      | —    | V <sub>DD</sub> -1.4 | V    |
| V <sub>OR</sub>  | PGA Maximum Output Voltage Range  | 2.2V            | —          | V <sub>SS</sub> +0.1 | —    | V <sub>DD</sub> -0.1 | V    |
|                  |                                   | 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    |
| V <sub>VR</sub>  | Fix Voltage Output of PGA         | 5V              | Ta=25°C    | -1%                  | 2    | +1%                  | V    |
|                  |                                   | 5V              | Ta=25°C    | -1%                  | 3    | +1%                  | V    |
|                  |                                   | 5V              | Ta=25°C    | -1%                  | 4    | +1%                  | V    |

Note: ADC conversion time (t<sub>ADC</sub>)=n (bits ADC) +4 (sampling time), the conversion for each bit needs one ADC clock (t<sub>ADCK</sub>).

### LVD & LVR Electrical Characteristics

Ta=25°C

| Symbol                 | Parameter                              | Test Conditions |  | Min. | Typ. | Max. | Unit |
|------------------------|--|-----------------|--|------|------|------|------|
|                        |  | V <sub>DD</sub> | Conditions                               |      |      |      |      |
| V <sub>LVR</sub>       | Low Voltage Reset Voltage              | —               | LVR enable, voltage select 1.7V          | -5%  | 1.7  | +5%  | V    |
|                        |  | —               | LVR enable, voltage select 1.9V          | -5%  | 1.9  | +5%  |      |
|                        |  | —               | LVR enable, voltage select 2.55V         | -3%  | 2.55 | +3%  |      |
|                        |  | —               | LVR enable, voltage select 3.15V         | -3%  | 3.15 | +3%  |      |
|                        |  | —               | LVR enable, voltage select 3.8V          | -3%  | 3.8  | +3%  |      |
| V <sub>LVD</sub>       | Low Voltage Detection Voltage          | —               | LVD enable, voltage select 1.8V          | -5%  | 1.8  | +5%  | V    |
|                        |  | —               | LVD enable, voltage select 2.0V          | -5%  | 2.0  | +5%  |      |
|                        |  | —               | LVD enable, voltage select 2.4V          | -5%  | 2.4  | +5%  |      |
|                        |  | —               | LVD enable, voltage select 2.7V          | -5%  | 2.7  | +5%  |      |
|                        |  | —               | LVD enable, voltage select 3.0V          | -5%  | 3.0  | +5%  |      |
|                        |  | —               | LVD enable, voltage select 3.3V          | -5%  | 3.3  | +5%  |      |
|                        |  | —               | LVD enable, voltage select 3.6V          | -5%  | 3.6  | +5%  |      |
|                        |  | —               | LVD enable, voltage select 4.0V          | -5%  | 4.0  | +5%  |      |
| I <sub>LVR/LVDBG</sub> | Operating Current                      | 3V              | LVD enable, LVR enable, LVR=1.9V, LVD=2V | —    | —    | 10   | μA   |
|                        |  | 5V              | LVD enable, LVR enable, LVR=1.9V, LVD=2V | —    | 8    | 15   | μA   |
| t <sub>LVDS</sub>      | LVDO Stable Time                       | —               | For LVR enable, LVD off → on             | —    | —    | 15   | μs   |
|                        |  | —               | For LVR disable, LVD off → on            | —    | —    | 150  | μs   |
| t <sub>LVR</sub>       | Minimum Low Voltage Width to Reset     | —               | —  | 120  | 240  | 480  | μs   |
| t <sub>LVD</sub>       | Minimum Low Voltage Width to Interrupt | —               | —  | 60   | 120  | 240  | μs   |
| I <sub>LVR</sub>       | Additional Current for LVR Enable      | 5V              | LVD disable                              | —    | —    | 8    | μA   |
| I <sub>LVD</sub>       | Additional Current for LVD Enable      | 5V              | LVR disable                              | —    | —    | 8    | μA   |

## Comparator Electrical Characteristics

Ta=25°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    |
| I <sub>COMP</sub> | Operating Current         | 1.8V            | CNVT[1:0]=00B  | —               | —    | 3                    | μA   |
|                   |                           | 3V              | CNVT[1:0]=00B  | —               | —    | 3                    | μA   |
|                   |                           | 5V              | CNVT[1:0]=00B  | —               | 1    | 3                    | μA   |
|                   |                           | 1.8V            | CNVT[1:0]=01B  | —               | —    | 22                   | μA   |
|                   |                           | 3V              | CNVT[1:0]=01B  | —               | —    | 22                   | μA   |
|                   |                           | 5V              | CNVT[1:0]=01B  | —               | 14   | 22                   | μA   |
|                   |                           | 1.8V            | CNVT[1:0]=10B  | —               | —    | 57                   | μA   |
|                   |                           | 3V              | CNVT[1:0]=10B  | —               | —    | 57                   | μA   |
|                   |                           | 5V              | CNVT[1:0]=10B  | —               | 36   | 57                   | μA   |
|                   |                           | 1.8V            | CNVT[1:0]=11B  | —               | —    | 92                   | μA   |
|                   |                           | 3V              | CNVT[1:0]=11B  | —               | —    | 92                   | μA   |
|                   |                           | 5V              | CNVT[1:0]=11B  | —               | 58   | 92                   | μA   |
| V <sub>OS</sub>   | Input Offset Voltage      | 1.8V            | Without calibration<br>(COF[4:0]=10000B),<br>CNVT[1:0]=00B | -10             | —    | 10                   | mV   |
|                   |                           | 3V              | Without calibration<br>(COF[4:0]=10000B),<br>CNVT[1:0]=00B | -10             | —    | 10                   | mV   |
|                   |                           | 5V              | Without calibration<br>(COF[4:0]=10000B),<br>CNVT[1:0]=00B | -10             | —    | 10                   | mV   |
|                   |                           | 1.8V            | With calibration,<br>CNVT[1:0]=00B<br>(Note 1)             | -4              | —    | 4                    | mV   |
|                   |                           | 3V              | With calibration,<br>CNVT[1:0]=00B<br>(Note 1)             | -4              | —    | 4                    | mV   |
|                   |                           | 5V              | With calibration,<br>CNVT[1:0]=00B<br>(Note 1)             | -4              | —    | 4                    | mV   |
| V <sub>CM</sub>   | Common Mode Voltage Range | —               | CNVT[1:0]=00B  | V <sub>SS</sub> | —    | V <sub>DD</sub> -1.4 | V    |
| A <sub>OL</sub>   | Open Loop Gain            | 1.8V            | CNVT[1:0]=00B  | 60              | —    | —                    | dB   |
|                   |                           | 3V              | CNVT[1:0]=00B  | 60              | —    | —                    | dB   |
|                   |                           | 5V              | CNVT[1:0]=00B  | 60              | 80   | —                    | dB   |
| V <sub>HYS</sub>  | Hysteresis                | 1.8V            | CNVT[1:0]=00B  | 10              | —    | 30                   | mV   |
|                   |                           | 3V              | CNVT[1:0]=00B  | 10              | —    | 30                   | mV   |
|                   |                           | 5V              | CNVT[1:0]=00B  | 10              | 24   | 30                   | mV   |

| Symbol          | Parameter     | Test Conditions |  | Min. | Typ. | Max. | Unit |
|-----------------|---------------|-----------------|--|------|------|------|------|
|                 |               | V <sub>DD</sub> | Conditions   |      |      |      |      |
| t <sub>RP</sub> | Response Time | 1.8V            | With 100mV overdrive,<br>C <sub>LOAD</sub> =3pF, CNVT[1:0]=00B | —    | 20   | 40   | μs   |
|                 |               | 3V              | With 100mV overdrive,<br>C <sub>LOAD</sub> =3pF, CNVT[1:0]=00B | —    | 20   | 40   | μs   |
|                 |               | 5V              | With 100mV overdrive,<br>C <sub>LOAD</sub> =3pF, CNVT[1:0]=00B | —    | 20   | 40   | μs   |
|                 |               | 1.8V            | With 100mV overdrive,<br>C <sub>LOAD</sub> =3pF, CNVT[1:0]=01B | —    | 1.2  | 3.0  | μs   |
|                 |               | 3V              | With 100mV overdrive,<br>C <sub>LOAD</sub> =3pF, CNVT[1:0]=01B | —    | 1.2  | 3.0  | μs   |
|                 |               | 5V              | With 100mV overdrive,<br>C <sub>LOAD</sub> =3pF, CNVT[1:0]=01B | —    | 1.2  | 3.0  | μs   |
|                 |               | 1.8V            | With 100mV overdrive,<br>C <sub>LOAD</sub> =3pF, CNVT[1:0]=10B | —    | 0.5  | 1.5  | μs   |
|                 |               | 3V              | With 100mV overdrive,<br>C <sub>LOAD</sub> =3pF, CNVT[1:0]=10B | —    | 0.5  | 1.5  | μs   |
|                 |               | 5V              | With 100mV overdrive,<br>C <sub>LOAD</sub> =3pF, CNVT[1:0]=10B | —    | 0.5  | 1.5  | μs   |
|                 |               | 1.8V            | With 100mV overdrive,<br>C <sub>LOAD</sub> =3pF, CNVT[1:0]=11B | —    | 0.3  | 1.0  | μs   |
|                 |               | 3V              | With 100mV overdrive,<br>C <sub>LOAD</sub> =3pF, CNVT[1:0]=11B | —    | 0.3  | 1.0  | μs   |
|                 |               | 5V              | With 100mV overdrive,<br>C <sub>LOAD</sub> =3pF, CNVT[1:0]=11B | —    | 0.3  | 1.0  | μs   |

Note 1. If Vin comparing threshold is lower than 250mV, the offset voltage could be over 4mV. It is recommended that when the comparing level is lower than 250mV, please first recalibrate the offset voltage.

2. All measurement is under C+ input voltage=(V<sub>DD</sub>-1.4)/2 and remain constant.

## LCD Electrical Characteristics

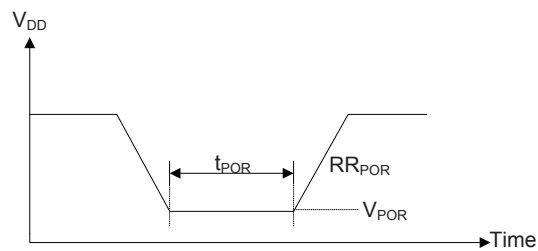
Ta=25°C

| Symbol            | Parameter                                   | Test Conditions |               | Min.            | Typ.            | Max.            | Unit |
|-------------------|---|-----------------|---------------|-----------------|-----------------|-----------------|------|
|                   |   | V <sub>DD</sub> | Conditions    |                 |                 |                 |      |
| I <sub>BIAS</sub> | V <sub>DD</sub> /2 bias Current for LCD     | 3V              | ISEL[1:0]=00B | 10.5            | 15              | 19.5            | μA   |
|                   |   | 5V              | ISEL[1:0]=00B | 17.5            | 25              | 32.5            | μA   |
|                   |   | 3V              | ISEL[1:0]=01B | 21              | 30              | 39              | μA   |
|                   |   | 5V              | ISEL[1:0]=01B | 35              | 50              | 65              | μA   |
|                   |   | 3V              | ISEL[1:0]=10B | 42              | 60              | 78              | μA   |
|                   |   | 5V              | ISEL[1:0]=10B | 70              | 100             | 130             | μA   |
|                   |   | 3V              | ISEL[1:0]=11B | 82.6            | 118             | 153.4           | μA   |
| V <sub>SCOM</sub> | V <sub>DD</sub> /2 Voltage for LCD COM Port | 2.2V~5.5V       | No load       | 0.475           | 0.5             | 0.525           | V    |
|                   |   |                 |               | V <sub>DD</sub> | V <sub>DD</sub> | V <sub>DD</sub> |      |

## Power on Reset Characteristics

Ta=25°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   |

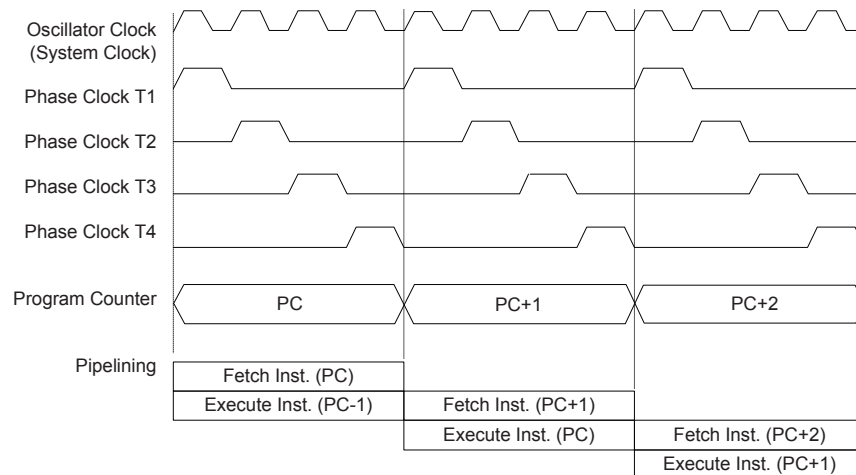


## System Architecture

A key factor in the high-performance features of the Holtek range of microcontrollers is attributed to their internal system architecture. The range of the device take 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 or two cycles for most of the standard or extended instructions respectively. The exceptions to this are 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

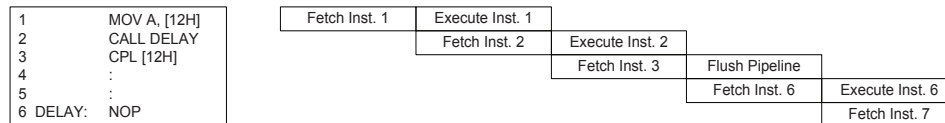
The main system clock, derived from either a HXT, LXT, HIRC or LIRC oscillator is subdivided into four internally generated non-overlapping clocks, T1~T4. The Program Counter is incremented at the beginning of the T1 clock during which time a new instruction is fetched. The remaining T2~T4 clocks carry 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.



**System Clocking and Pipelining**



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.



**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           |              |
|---------------------------|--------------|
| Program Counter High Byte | PCL Register |
| PC12~PC8                  | PCL7~PCL0    |

**Program Counter**

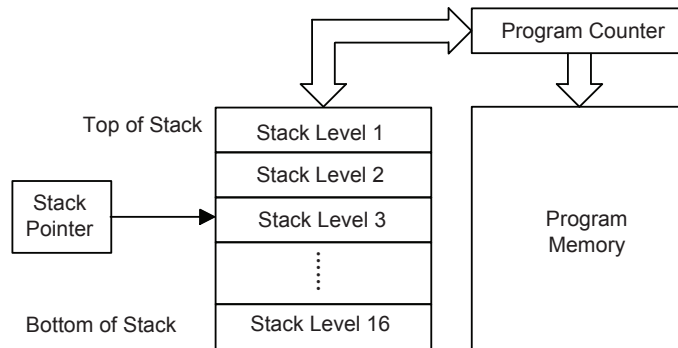
The lower byte of the Program Counter, known as the Program Counter Low register or PCL, is available for program control and is a readable and writeable register. By transferring data directly into this register, a short program jump can be executed directly. However, as only this low byte is available 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 16 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:

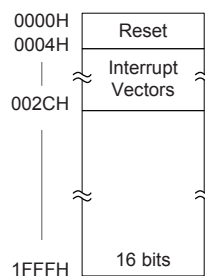
- Arithmetic operations: ADD, ADDM, ADC, ADCM, SUB, SUBM, SBC, SBCM, DAA, LADD, LADDM, LADC, LADCM, LSUB, LSUBM, LSBC, LSBCM, LDAA
- Logic operations: AND, OR, XOR, ANDM, ORM, XORM, CPL, CPLA, LAND, LANDM, LOR, LORM, LXOR, LXORM, LCPL, LCPLA
- Rotation: RRA, RR, RRCA, RRC, RLA, RL, RLCA, RLC, LRR, LRRCA, LRRCA, LRRCA, LRRCA, LRLA, LRL, LRLCA, LRLC
- Increment and Decrement: INCA, INC, DECA, DEC, LINCA, LINC, LDECA, LDEC
- Branch decision: JMP, SZ, SZA, SNZ, SIZ, SDZ, SIZA, SDZA, CALL, RET, RETI, LSNZ, LSZ, LSZA, LSIZ, LSIZA, LSDZ, LSDZA

## Flash Program Memory

The Program Memory is the location where the user code or program is stored. For this 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 offer 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  $8K \times 16$  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

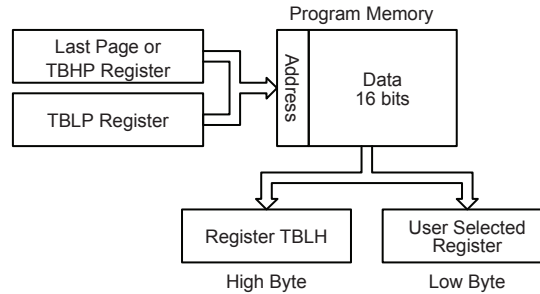
Within the Program Memory, certain locations are reserved for the reset and interrupts. The location 0000H is reserved for use by the device reset for program initialisation. After a device reset is initiated, the program will jump to this location and begin execution.

### 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 register, TBLP and TBHP. These registers define the total address of the look-up table.

After setting up the table pointer pair, the table data can be retrieved from the Program Memory using the corresponding table read instruction such as "TABRD [m]" or "TABRDL [m]" respectively when the memory [m] is located in sector 0. If the memory [m] is located in other sectors, the data can be retrieved from the program memory using the corresponding extended table read instruction such as "LTABRD [m]" or "LTABRDL [m]" respectively. 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 "1F00H" which refers to the start address of the last page within the 8K words Program Memory. The table pointer 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 "1F06H" or 6 locations after the start of the last page. Note that the value for the table pointer is referenced to the first address of the specific page 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/write register and can 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
mov a,1Fh          ; initialise high table pointer
mov tbhp,a
:
:
tabrd tempreg1     ; transfers value in table referenced by table pointer,
                  ; data at program memory address "1F06H" 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 "1F05H" transferred to tempreg2 and TBLH
                  ; in this example the data "1AH" is transferred to tempreg1 and data "0FH" to
                  ; register tempreg2
:
:
org 1F00h          ; sets initial address of program memory
dc 00Ah, 00Bh, 00Ch, 00Dh, 00Eh, 00Fh, 01Ah, 01Bh
:
  
```

### In Circuit Programming – ICP

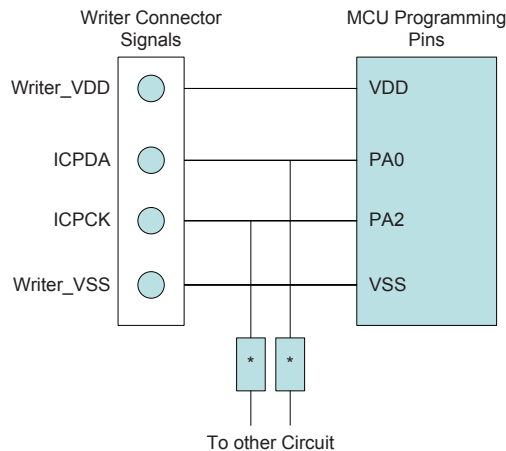
The provision of Flash type Program Memory provides the user with a means of convenient and easy upgrades and modifications to their programs on the same device. As an additional convenience, Holtek has provided a means of programming the microcontroller in-circuit using a 4-pin interface. This provides 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 and EEPROM data Memory can both 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 are beyond the scope of this document and will be supplied in supplementary literature.

During the programming process, the user must take control of the ICPDA and ICPCK pins for data and clock programming purposes 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

There is an EV chip named HT66V3197, which is used to emulate HT66F3197 device. The EV chip device also provides an "On-Chip Debug" function to debug the device during the development process. The EV chip and the actual MCU device are almost functionally compatible except for the "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 actual MCU 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 a more detailed OCDS description, 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.

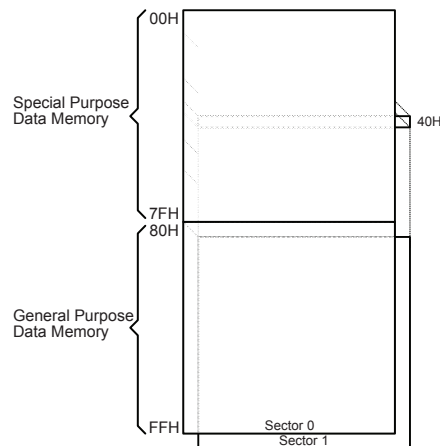
### Structure

Divided into two types, the first of these is an area of RAM, known as the Special Function Data Memory. Here are located registers which 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 known as the General Purpose Data Memory, which is reserved for general purpose use. All locations within this area are read and write accessible under program control.

The overall Data Memory is subdivided into several sectors for the device. The Special Purpose Data Memory registers addressed from 00H~7FH in Data Memory are common and accessible in sector 0 and the EEC register at the address 40H in sector 1. Switching between the different Data Memory sectors is achieved by properly setting the Memory Pointers to the correct value. The start address of the Data Memory for the device is the address 00H.

| Special Purpose Data Memory |  | General Purpose Data Memory |  |
|-----------------------------|--|-----------------------------|--|
| Available Sectors           | Sectors                                  | Capacity                    | Sectors                                |
| 0,1                         | Sector 0: 00H~7FH<br>Sector 1: 40H (EEC) | 256 × 8                     | Sector 0: 80H~FFH<br>Sector 1: 80H~FFH |

**Data Memory Summary**



**Data Memory Structure**

### **Data Memory Addressing**

For this device that supports the extended instructions, there is no Bank Pointer for Data Memory. For Data Memory the desired Sector is pointed by the MP1H or MP2H register and the certain Data Memory address in the selected sector is specified by the MP1L or MP2L register when using indirect addressing access.

Direct Addressing can be used in all sectors using the corresponding instruction which can address all available data memory space. For the accessed data memory which is located in any data memory sectors except sector 0, the extended instructions can be used to access the data memory instead of using the indirect addressing access. The main difference between standard instructions and extended instructions is that the data memory address "m" in the extended instructions has 9 valid bits, the high byte indicates a sector and the low byte indicates a specific address.

### **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".



| Sector 0 |        | Sector 1 | Sector 0 |            | Sector 1 |
|----------|--------|----------|----------|------------|----------|
| 00H      | IAR0   |          | 40H      |            | EEC      |
| 01H      | MP0    |          | 41H      | EEA        |          |
| 02H      | IAR1   |          | 42H      | EED        |          |
| 03H      | MP1L   |          | 43H      |            |          |
| 04H      | MP1H   |          | 44H      | PTMC0      |          |
| 05H      | ACC    |          | 45H      | PTMC1      |          |
| 06H      | PCL    |          | 46H      | PTMDL      |          |
| 07H      | TBLP   |          | 47H      | PTMDH      |          |
| 08H      | TBLH   |          | 48H      | PTMAL      |          |
| 09H      | TBHP   |          | 49H      | PTMAH      |          |
| 0AH      | STATUS |          | 4AH      | PTMRPL     |          |
| 0BH      |        |          | 4BH      | PTMRPH     |          |
| 0CH      | IAR2   |          | 4CH      | STMC0      |          |
| 0DH      | MP2L   |          | 4DH      | STMC1      |          |
| 0EH      | MP2H   |          | 4EH      | STMDL      |          |
| 0FH      | RSTFC  |          | 4FH      | STMDH      |          |
| 10H      | INTC0  |          | 50H      | STMAL      |          |
| 11H      | INTC1  |          | 51H      | STMAH      |          |
| 12H      | INTC2  |          | 52H      | STMRP      |          |
| 13H      |        |          | 53H      | SLEDC0     |          |
| 14H      | PA     |          | 54H      | SLEDC1     |          |
| 15H      | PAC    |          | 55H      | SLEDC2     |          |
| 16H      | PAPU   |          | 56H      |            |          |
| 17H      | PAWU   |          | 57H      | TB0C       |          |
| 18H      | PB     |          | 58H      | TB1C       |          |
| 19H      | PBC    |          | 59H      | PSC0R      |          |
| 1AH      | PBPU   |          | 5AH      | PSC1R      |          |
| 1BH      | PC     |          | 5BH      | SADOL      |          |
| 1CH      | PCC    |          | 5CH      | SADOH      |          |
| 1DH      | PCPU   |          | 5DH      | SADC0      |          |
| 1EH      | PD     |          | 5EH      | SADC1      |          |
| 1FH      | PDC    |          | 5FH      | SADC2      |          |
| 20H      | PDPU   |          | 60H      | SIMC0      |          |
| 21H      | PE     |          | 61H      | SIMC1      |          |
| 22H      | PEC    |          | 62H      | SIMD       |          |
| 23H      | PEPU   |          | 63H      | SIMC2/SIMA |          |
| 24H      | PF     |          | 64H      | SIMTOC     |          |
| 25H      | PFC    |          | 65H      | SCOMC      |          |
| 26H      | PFPU   |          | 66H      | USR        |          |
| 27H      | RSTC   |          | 67H      | UCR1       |          |
| 28H      | VBGRC  |          | 68H      | UCR2       |          |
| 29H      |        |          | 69H      | TXR_RXR    |          |
| 2AH      | MF10   |          | 6AH      | BRG        |          |
| 2BH      | MF11   |          | 6BH      |            |          |
| 2CH      | MF12   |          | 6CH      |            |          |
| 2DH      | INTEG  |          | 6DH      | IFS0       |          |
| 2EH      |        |          | 6EH      | IFS1       |          |
| 2FH      | SCC    |          | 6FH      |            |          |
| 30H      | HIRCC  |          | 70H      | PAS0       |          |
| 31H      | HXTC   |          | 71H      | PAS1       |          |
| 32H      | LXTC   |          | 72H      | PBS0       |          |
| 33H      | WDTC   |          | 73H      | PBS1       |          |
| 34H      | LVRC   |          | 74H      | PCS0       |          |
| 35H      | LVDC   |          | 75H      | PCS1       |          |
| 36H      | LVPUC  |          | 76H      | PDS0       |          |
| 37H      | CMPC   |          | 77H      | PDS1       |          |
| 38H      | CMPVOS |          | 78H      | PES0       |          |
| 39H      | CTMC0  |          | 79H      | PES1       |          |
| 3AH      | CTMC1  |          | 7AH      | PFS0       |          |
| 3BH      | CTMDL  |          | 7BH      | PFS1       |          |
| 3CH      | CTMDH  |          | 7CH      |            |          |
| 3DH      | CTMAL  |          |          |            |          |
| 3EH      | CTMAH  |          |          |            |          |
| 3FH      | CTMRP  |          |          |            |          |
|          |        |          | 7FH      |            |          |

□ : unused, read as 00H

**Special Purpose Data Memory**

## Special Function Register Description

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

### Indirect Addressing Registers – IAR0, IAR1, IAR2

The Indirect Addressing Registers, IAR0, IAR1 and IAR2, 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, IAR1 and IAR2 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, MP1L/MP1H or MP2L/MP2H. Acting as a pair, IAR0 and MP0 can together access data from Sector 0 while the IAR1 register together with MP1L/MP1H register pair and IAR2 register together with MP2L/MP2H register pair can access data from any sector. As the Indirect Addressing Registers are not physically implemented, reading the Indirect Addressing Registers directly will return a result of "00H" and writing to the registers directly will result in no operation.

### Memory Pointers – MP0, MP1L/MP1H, MP2L/MP2H

Five Memory Pointers, known as MP0, MP1L, MP1H, MP2L and MP2H 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 Sector 0, while MP1L/MP1H together with IAR1 and MP2L/MP2H together with IAR2 are used to access data from all sectors according to the corresponding MP1H or MP2H register. Direct Addressing can be used in all sectors using the corresponding instruction which can address all available data memory space.

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

### Indirect Addressing Program Example

- **Example 1**

```
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                 ; increment memory pointer
    sdz block               ; check if last memory location has been cleared
    jmp loop
continue:
```

• **Example 2**

```
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,01h ; setup the memory sector
mov mp1h,a
mov a,offset adres1 ; Accumulator loaded with first RAM address
mov mp1l,a ; setup memory pointer with first RAM address
loop:
clr IAR1 ; clear the data at address defined by MP1L
inc mp1l ; increment memory pointer MP1L
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.

**Direct Addressing Program Example using extended instructions**

```
data .section 'data'
temp db ?
code .section at 0 code
org 00h
start:
lmov a,[m] ; move [m] data to acc
lsub a, [m+1] ; compare [m] and [m+1] data
snz c ; [m]>[m+1]?
jmp continue ; no
lmov a,[m] ; yes, exchange [m] and [m+1] data
mov temp,a
lmov a,[m+1]
lmov [m],a
mov a,temp
lmov [m+1],a
continue:
:
```

Note: here "m" is a data memory address located in any data memory sectors. For example, m=1F0H, it indicates address 0F0H in Sector 1.

### **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 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), SC flag, CZ flag, 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, C SC and CZ 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.
- SC is the result of the "XOR" operation which is performed by the OV flag and the MSB of the current instruction operation result.
- CZ is the operational result of different flags for different instructions. Refer to register definitions for more details.

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 registers 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 | SC | CZ | TO | PDF | OV  | Z   | AC  | C   |
| R/W  | R  | R  | R  | R   | R/W | R/W | R/W | R/W |
| POR  | x  | x  | 0  | 0   | x   | x   | x   | x   |

"x": unknown

- Bit 7     **SC**: The result of the "XOR" operation which is performed by the OV flag and the MSB of the instruction operation result.
- Bit 6     **CZ**: The the operational result of different flags for different instructions.  
 For SUB/SUBM/LSUB/LSUBM instructions, the CZ flag is equal to the Z flag.  
 For SBC/SBCM/LSBC/LSBCM instructions, the CZ flag is the "AND" operation result which is performed by the previous operation CZ flag and current operation zero flag.  
 For other instructions, the CZ flag will not be affected.
- 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  
 C is also affected by a rotate through carry instruction.

## EEPROM Data Memory

The device contains an area of internal EEPROM Data Memory. EEPROM, which stands for Electrically Erasable Programmable Read Only Memory, is by its nature a non-volatile form of 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 64×8 bits. Unlike the Program Memory and RAM Data Memory, the EEPROM Data Memory is not directly mapped and is therefore not directly accessible 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 and data register in Sector 0 and a single control register in Sector 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 Sector 0, they can be directly accessed in the same way as any other Special Function Register. The EEC register however, being located in Sector 1, cannot be directly addressed directly and can only be read from or written to indirectly using the MP1H/MP1L or MP2H/MP2L Memory Pointer and Indirect Addressing Register, IAR1 or IAR2. Because the EEC control register is located at address 40H in Sector 1, the Memory Pointer low byte register, MP1L or MP2L, must first be set to the value 40H and the Memory Pointer high byte register, MP1H or MP2H, 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           | —   | —  | D5 | D4 | D3   | D2 | D1   | D0 |
| EED           | D7  | D6 | D5 | D4 | D3   | D2 | D1   | D0 |
| EEC           | —   | —  | —  | —  | WREN | WR | RDEN | RD |

**EEPROM Control Registers List**

### EEA Register

| Bit  | 7 | 6 | 5   | 4   | 3   | 2   | 1   | 0   |
|------|---|---|-----|-----|-----|-----|-----|-----|
| Name | — | — | D5  | D4  | D3  | D2  | D1  | D0  |
| 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 ~ 0    Data EEPROM address  
                   Data EEPROM address bit 5 ~ bit 0

**EED 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     Data EEPROM data  
 Data EEPROM data bit 7 ~ bit 0

**EED Register**

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

Bit 7 ~ 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: The WREN, WR, RDEN and RD can not be set to "1" at the same time in one instruction. The WR and RD can not be set to "1" at the same time.

### Reading Data from the EEPROM

To read data from the EEPROM, the read enable bit, RDEN, in the EEC register must first be set high to enable the read function. The EEPROM address of the data to be read must then be placed in the EEA register. 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. Then 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 initial a write cycle. These two instructions must be executed consecutively. 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 Memory Pointer high byte register, MP1H or MP2H, will be reset to zero, which means that Data Memory Sector 0 will be selected. As the EEPROM control register is located in Sector 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. However, as the EEPROM is contained within a Multi-function Interrupt, the associated multi-function interrupt enable bit must also be set. When an EEPROM write cycle ends, the DEF request flag and its associated multi-function interrupt request flag will both be set. If the global, EEPROM and Multi-function interrupts are enabled and the stack is not full, a jump to the associated Multi-function Interrupt vector will take place. When the interrupt is serviced only the Multi-function interrupt flag will be automatically reset, the EEPROM interrupt flag must be manually reset by the application program.



## 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 Memory Pointer high byte register could be normally cleared to zero as this would inhibit access to Sector 1 where the EEPROM control register exist. 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 completed. 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, 040H              ; setup memory pointer low byte MP1L
MOV MP1L, A              ; MP1L points to EEC register
MOV A, 01H               ; setup Memory Pointer high byte MP1H
MOV MP1H, 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/write
CLR MP1H
MOV A, EED               ; move read data to register
MOV READ_DATA, A
```

### • 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, 040H              ; setup memory pointer low byte MP1L
MOV MP1L, A              ; MP1L points to EEC register
MOV A, 01H               ; setup Memory Pointer high byte MP1H
MOV MP1H, A
CLR EMI
SET IAR1.3               ; set WREN bit, enable write operations
SET IAR1.2               ; start Write Cycle - set WR bit - executed immediately
                        ; after set WREN bit

SET EMI
BACK:
SZ IAR1.2                ; check for write cycle end
JMP BACK
CLR IAR1                 ; disable EEPROM read/write
CLR MP1H
```

## 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 selections and operation are selected through a combination of configuration options and 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. External oscillators requiring some external components as well as fully integrated internal oscillators, requiring no external components, are provided to form a wide range of both fast and slow system oscillators. All oscillator options are selected through the register programming. The higher frequency oscillators provide higher performance but carry with it the disadvantage of higher power requirements, while the opposite is of course true for the lower frequency oscillators. 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.

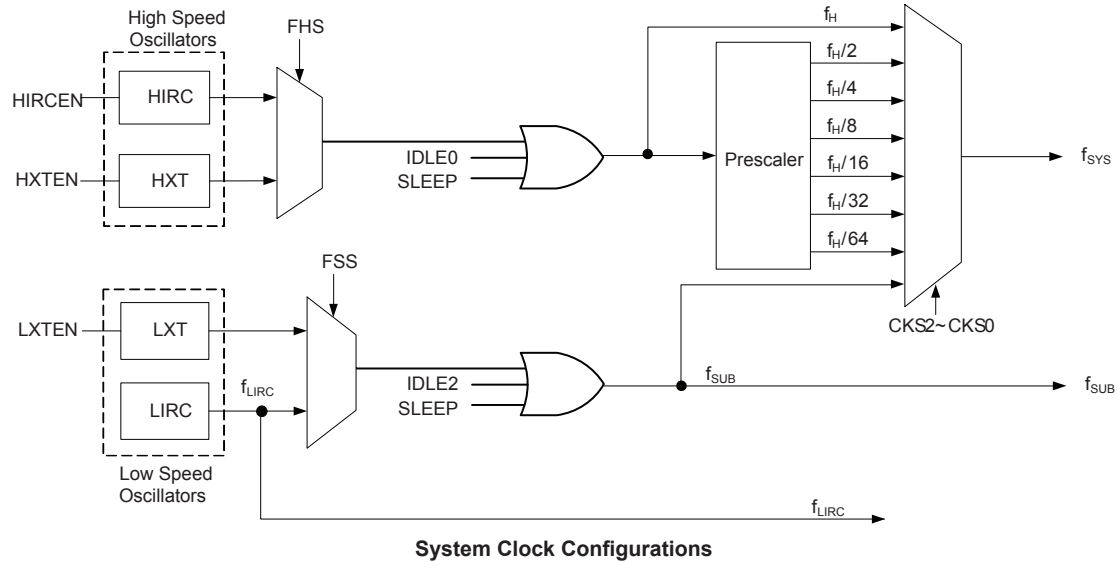
| Type                        | Name | Freq.        | Pins      |
|-----------------------------|------|--------------|-----------|
| External High Speed Crystal | HXT  | 400kHz~16MHz | OSC1/OSC2 |
| Internal High Speed RC      | HIRC | 4, 8, 12MHz  | —         |
| External Low Speed Crystal  | LXT  | 32.768kHz    | XT1/XT2   |
| Internal Low Speed RC       | LIRC | 32kHz        | —         |

Oscillator Types

### System Clock Configurations

There are four methods of generating the system clock, two high speed oscillators and two low speed oscillators. The high speed oscillators are the external crystal/ceramic oscillator - HXT and the internal 4MHz, 8MHz, 12MHz RC oscillator - HIRC. The two low speed oscillators are the internal 32kHz RC oscillator - LIRC and the external 32.768kHz crystal oscillator - LXT. 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.

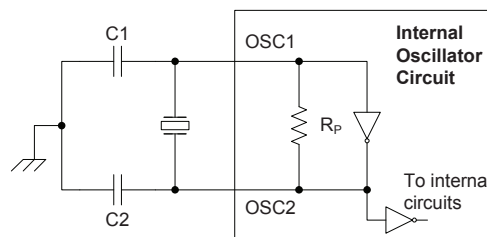
The actual source clock used for the low speed oscillators is chosen by the FSS bit in the SCC register while for the high speed oscillator the source clock is selected by the FHS bit in the SCC register. The frequency of the slow speed or high speed system clock is determined using the CKS2~CKS0 bits in the SCC register. Note that two oscillator selections must be made namely one high speed and one low speed system oscillators. It is not possible to choose a no-oscillator selection for either the high or low speed oscillator.



### External Crystal/Ceramic Oscillator – HXT

The External Crystal/Ceramic System Oscillator is one of the high frequency oscillator choices, which is selected via a software control bit, FHS. For most crystal oscillator configurations, the simple connection of a crystal across OSC1 and OSC2 will create the necessary phase shift and feedback for oscillation, without requiring external capacitors. However, for some crystal types and frequencies, to ensure oscillation, it may be necessary to add two small value capacitors, C1 and C2. Using a ceramic resonator will usually require two small value capacitors, C1 and C2, to be connected as shown for oscillation to occur. The values of C1 and C2 should be selected in consultation with the crystal or resonator manufacturer's specification.

For oscillator stability and to minimise the effects of noise and crosstalk, it is important to ensure that the crystal and any associated resistors and capacitors along with interconnecting lines are all located as close to the MCU as possible.



- Note: 1.  $R_P$  is normally not required. C1 and C2 are required.  
 2. Although not shown OSC1/OSC2 pins have a parasitic capacitance of around 7pF.

### Crystal/Resonator Oscillator – HXT

| Crystal Oscillator C1 and C2 Values           |       |       |
|---|-------|-------|
| Crystal Frequency                             | C1    | C2    |
| 16MHz   | 0pF   | 0pF   |
| 12MHz   | 0pF   | 0pF   |
| 8MHz  | 0pF   | 0pF   |
| 4MHz  | 0pF   | 0pF   |
| 1MHz  | 100pF | 100pF |
| Note: C1 and C2 values are for guidance only. |       |       |

**Crystal Recommended Capacitor Values**

### Internal RC Oscillator – HIRC

The internal RC oscillator is a fully integrated system oscillator requiring no external components. The internal RC oscillator has three fixed frequencies of 4MHz, 8MHz, and 12MHz. 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. As a result, at a power supply of either 3V or 5V and at a temperature of 25°C degrees, the fixed oscillation frequency of 4MHz, 8MHz or 12MHz will have a tolerance within 2%. Note that if this internal system clock option is selected, as it requires no external pins for its operation, I/O pins are free for use as normal I/O pins.

### External 32.768kHz Crystal Oscillator – LXT

The External 32.768kHz Crystal System Oscillator is one of the low frequency oscillator choices, which is selected via a software control bit, FSS. This clock source has a fixed frequency of 32.768kHz and requires a 32.768kHz crystal to be connected between pins XT1 and XT2. The external resistor and capacitor components connected to the 32.768kHz crystal are necessary to provide oscillation. For applications where precise frequencies are essential, these components may be required to provide frequency compensation due to different crystal manufacturing tolerances. After the LXT oscillator is enabled by setting the LXTEN bit to 1, there is a time delay associated with the LXT oscillator waiting for it to start-up.

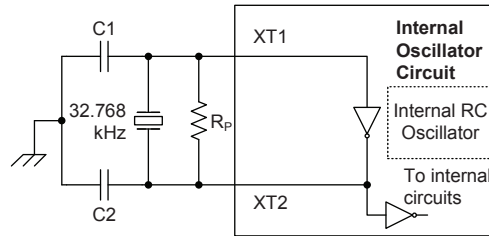
When the microcontroller enters the SLEEP or IDLE Mode, the system clock is switched off to stop microcontroller activity and to conserve power. However, in many microcontroller applications it may be necessary to keep the internal timers operational even when the microcontroller is in the SLEEP or IDLE Mode. To do this, another clock, independent of the system clock, must be provided.

However, for some crystals, to ensure oscillation and accurate frequency generation, it is necessary to add two small value external capacitors, C1 and C2. The exact values of C1 and C2 should be selected in consultation with the crystal or resonator manufacturer specification. The external parallel feedback resistor,  $R_p$ , is required.

The pin-shared software control bits determine if the XT1/XT2 pins are used for the LXT oscillator or as logic I/O or other pin-shared functional pins.

- If the LXT oscillator is not used for any clock source, the XT1/XT2 pins can be used as normal I/O or other pin-shared functional pins.
- If the LXT oscillator is used for any clock source, the 32.768kHz crystal should be connected to the XT1/XT2 pins.

For oscillator stability and to minimise the effects of noise and crosstalk, it is important to ensure that the crystal and any associated resistors and capacitors along with interconnecting lines are all located as close to the MCU as possible.



Note: 1.  $R_p$ , C1 and C2 are required.  
 2. Although not shown XT1/XT2 pins have a parasitic capacitance of around 7pF.

**External LXT Oscillator**

| LXT Oscillator C1 and C2 Values |      |      |
|---------------------------------|------|------|
| Crystal Frequency               | C1   | C2   |
| 32.768kHz                       | 10pF | 10pF |

Note: 1. C1 and C2 values are for guidance only.  
 2.  $R_p=5M\sim 10M\Omega$  is recommended.

**32.768kHz Crystal Recommended Capacitor Values**

**LXT Oscillator Low Power Function**

The LXT oscillator can function in one of two modes, the Speed-up Mode and the Low-power Mode. The mode selection is executed using the LXTSP bit in the LXTC register.

| LXTSP Bit | LXT Operating Mode |
|-----------|--------------------|
| 0         | Low-power          |
| 1         | Speed-up           |

When the LXTSP bit is set to high, the LXT Speed-up Mode will be enabled. In the Speed-up Mode the LXT oscillator will power up and stabilize quickly. However, after the LXT oscillator has fully powered up, it can be placed into the Low-power Mode by clearing the LXTSP bit to zero and the oscillator will continue to run but with reduced current consumption. It is important to note that the LXT operating mode switching must be properly controlled before the LXT oscillator clock is selected as the system clock source. Once the LXT oscillator clock is selected as the system clock source using the CKS bit field and FSS bit in the SCC register, the LXT oscillator operating mode can not be changed.

It should be note that no matter what condition the LXTSP is set to the LXT oscillator will always function normally. The only difference is that it will take more time to start up if in the Low-power Mode.

**Internal 32kHz Oscillator – LIRC**

The Internal 32 kHz System Oscillator is one of the low frequency oscillator choices, which is selected via a software control bit, FSS. It is a fully integrated RC oscillator with a typical frequency of 32kHz at 5V, 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. As a result, at a power supply of 5V and at a temperature of 25°C degrees, the fixed oscillation frequency of 32kHz will have a tolerance within 0.5%.

**Supplementary Oscillators**

The low speed oscillators, in addition to providing a system clock source are also used to provide a clock source to two other device functions. These are the Watchdog Timer and the Time Base Interrupts.

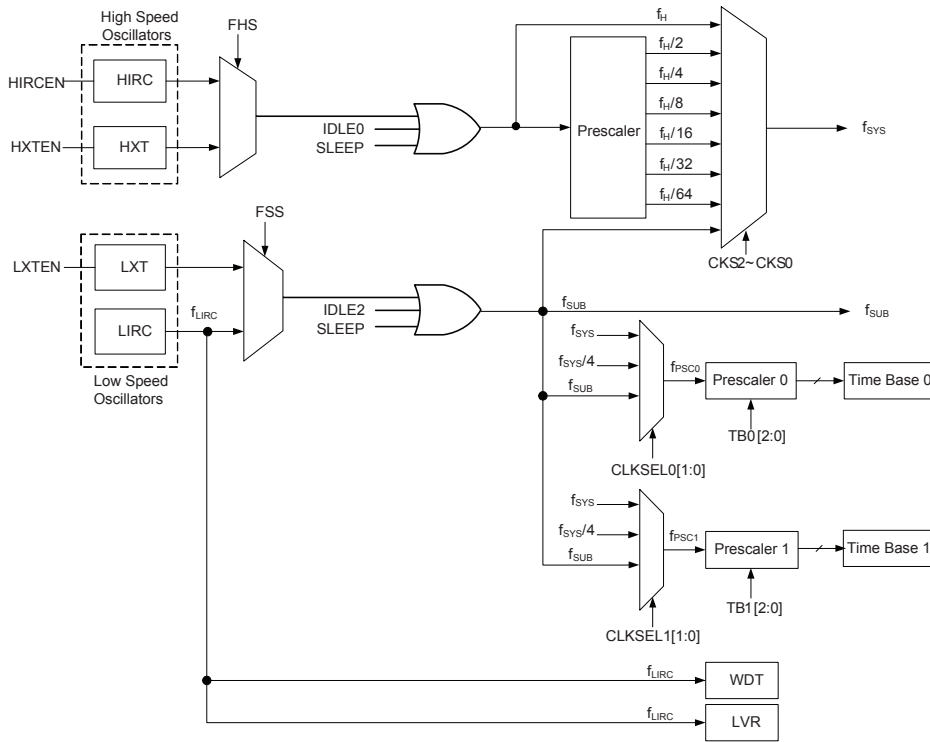
## Operating Modes and System Clocks

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

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

The main system clock, can come from either 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 an HXT or HIRC oscillator, selected via configuring the FHS bit in the SCC register. The low speed system clock source can be sourced from the internal clock  $f_{SUB}$ . If  $f_{SUB}$  is selected then it can be sourced by either the LXT or LIRC oscillators, selected via configuring the FSS bit in the SCC register. 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 can be stopped 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 NORMAL Mode and SLOW Mode. The remaining four modes, the SLEEP, IDLE0, IDLE1 and IDLE2 Mode are used when the microcontroller CPU is switched off to conserve power.

| Operation Mode | CPU | Related Register value |        |          | f <sub>sys</sub> | f <sub>H</sub>        | f <sub>SUB</sub> | f <sub>LIRC</sub>     |
|----------------|-----|------------------------|--------|----------|------------------|-----------------------|------------------|-----------------------|
|                |     | FHIDEN                 | FSIDEN | CKS[2:0] |                  |                       |                  |                       |
| Normal Mode    | On  | x                      | x      | 000~110  | On               | On                    | On               | On                    |
| Slow Mode      | On  | x                      | x      | 111      | On               | On/Off <sup>(1)</sup> | On               | On                    |
| Idle0 Mode     | Off | 0                      | 1      | 000~110  | Off              | Off                   | On               | On                    |
|                |     |                        |        | 111      | On               |                       |                  |                       |
| Idle1 Mode     | Off | 1                      | 1      | xxx      | On               | On                    | On               | On                    |
| Idle 2 Mode    | Off | 1                      | 0      | 000~110  | On               | On                    | Off              | On                    |
|                |     |                        |        | 111      | Off              |                       |                  |                       |
| Sleep Mode     | 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 can be switched on or off which is controlled by the WDT function being enabled or disabled in the SLEEP mode.

### NORMAL Mode

As the name suggests 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 one of the high speed oscillators. This mode operates allowing the microcontroller to operate normally with a clock source will come from one of the high speed oscillators, either the HXT or HIRC oscillators. 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 either the LIRC or LXT oscillator.

### SLEEP Mode

The SLEEP Mode is entered when an HALT instruction is executed and when the FHIDEN and FSIDEN bit are low. In the SLEEP mode the CPU will be stopped, and the f<sub>SUB</sub> clock to peripheral will be stopped too, but the Watchdog Timer function is decided by user application.

### IDLE0 Mode

The IDLE0 Mode is entered when an 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 an 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 an 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 Register

The registers, SCC, HIRCC, HXTC and LXTC, are used to control the system clock and the corresponding oscillator configurations.

| Register Name | Bit  |      |      |   |       |       |        |        |
|---------------|------|------|------|---|-------|-------|--------|--------|
|               | 7    | 6    | 5    | 4 | 3     | 2     | 1      | 0      |
| SCC           | CKS2 | CKS1 | CKS0 | — | FHS   | FSS   | FHIDEN | FSIDEN |
| HIRCC         | —    | —    | —    | — | HIRC1 | HIRC0 | HIRCF  | HIRCEN |
| HXTC          | —    | —    | —    | — | —     | HXTM  | HXTF   | HXTEN  |
| LXTC          | —    | —    | —    | — | —     | LXTSP | LXTF   | LXTEN  |

**System Operating Mode Control Registers List**

### SCC Register

| Bit  | 7    | 6    | 5    | 4 | 3   | 2   | 1      | 0      |
|------|------|------|------|---|-----|-----|--------|--------|
| Name | CKS2 | CKS1 | CKS0 | — | FHS | FSS | FHIDEN | FSIDEN |
| 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 7~5     **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     Unimplemented, read as "0".

Bit 3     **FHS**: High Frequency clock selection

0: HIRC  
1: HXT

Bit 2     **FSS**: Low Frequency clock selection

0: LIRC  
1: LXT



- 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. The LIRC oscillator is controlled by this bit together with the WDT function enable control when the LIRC is selected to be the low speed oscillator clock source or the WDT function is enabled respectively. If this bit is cleared to 0 but the WDT function is enabled, the LIRC oscillator will also be enabled.

**HIRCC Register**

| Bit  | 7 | 6 | 5 | 4 | 3     | 2     | 1     | 0      |
|------|---|---|---|---|-------|-------|-------|--------|
| Name | — | — | — | — | HIRC1 | HIRC0 | HIRCF | HIRCEN |
| R/W  | — | — | — | — | R/W   | R/W   | R     | R/W    |
| POR  | — | — | — | — | 0     | 0     | 0     | 1      |

- Bit 7~4     Unimplemented, read as "0".
- Bit 3~2     **HIRC1~HIRC0**: HIRC frequency selection  
           00: 4 MHz  
           01: 8 MHz  
           10: 12 MHz  
           11: 4 MHz  
           When the HIRC oscillator is enabled or the HIRC frequency selection is changed by application program, the clock frequency will automatically be changed after the HIRCF flag is set to 1.
- 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 or the HIRC frequency selection is changed by application program, 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

**HXTC Register**

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

- Bit 7~3     Unimplemented, read as "0".
- Bit 2     **HXTM**: HXT mode selection  
           0: HXT frequency ≤ 10 MHz  
           1: HXT frequency >10 MHz  
           This bit is used to select the HXT oscillator operating mode. Note that this bit must be properly configured before the HXT is enabled. When the HXTEN bit is set to 1 to enable the HXT oscillator, it is invalid to change the value of this bit.

- Bit 1     **HXTF**: HXT oscillator stable flag  
           0: HXT unstable  
           1: HXT stable  
           This bit is used to indicate whether the HXT oscillator is stable or not. When the HXTEN bit is set to 1 to enable the HXT oscillator, the HXTF bit will first be cleared to 0 and then set to 1 after the HXT oscillator is stable.
- Bit 0     **HXTEN**: HXT oscillator enable control  
           0: Disable  
           1: Enable

**LXTC Register**

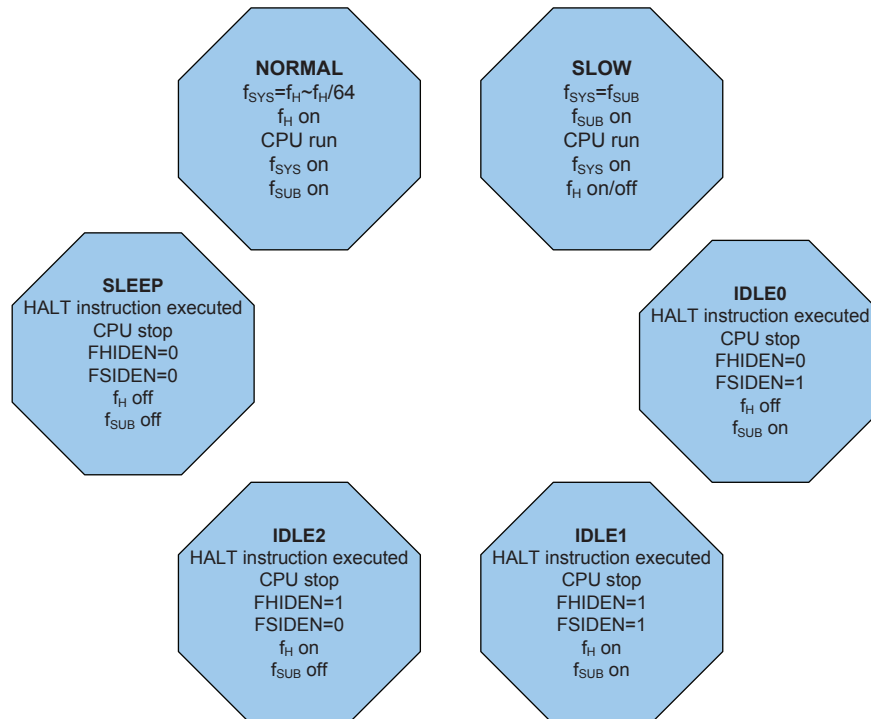
| Bit  | 7 | 6 | 5 | 4 | 3 | 2     | 1    | 0     |
|------|---|---|---|---|---|-------|------|-------|
| Name | — | — | — | — | — | LXTSP | LXTF | LXTEN |
| R/W  | — | — | — | — | — | RW    | R    | R/W   |
| POR  | — | — | — | — | — | 0     | 0    | 0     |

- Bit 7~3    Unimplemented, read as "0".
- Bit 2     **LXTSP**: LXT oscillator speed-up control  
           0: Disable – Low power  
           1: Enable – Speed up  
           This bit is used to control whether the LXT oscillator is operating in the low power or Speed-up mode. When the LXTSP bit is set to 1, the LXT oscillator will oscillate quickly but consume more power. If the LXTSP bit is cleared to 0, the LXT oscillator will consume less power but take longer time to stabilise. It is important to note that this bit can not be changed after the LXT oscillator is selected as the system clock source using the CKS2~CKS0 and FSS bits in the SCC register.
- Bit 1     **LXTF**: LXT oscillator stable flag  
           0: LXT unstable  
           1: LXT stable  
           This bit is used to indicate whether the LXT oscillator is stable or not. When the LXTEN bit is set to 1 to enable the LXT oscillator, the LXTF bit will first be cleared to 0 and then set to 1 after the LXT oscillator is stable.
- Bit 0     **LXTEN**: LXT 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.

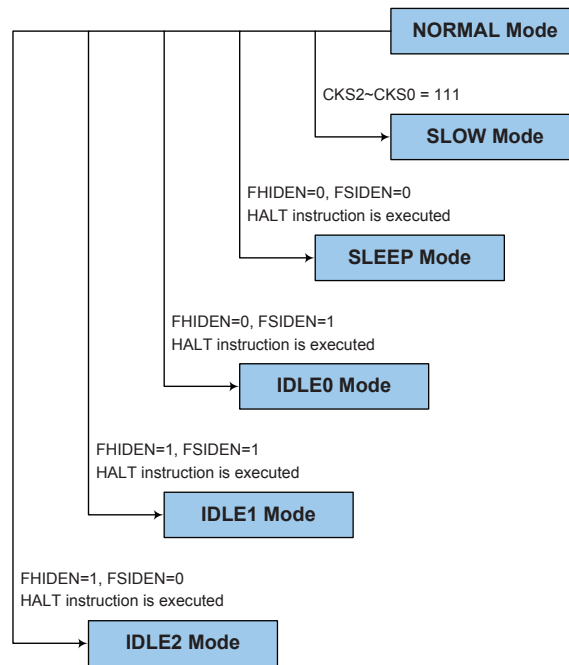
In simple terms, Mode Switching between the NORMAL Mode and SLOW Mode is executed using the CKS2~CKS0 bits in the SCC register while Mode Switching from the NORMAL/SLOW Modes to the SLEEP/IDLE Modes is executed via the HALT instruction. When an HALT instruction is executed, whether the device enters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



**NORMAL Mode to SLOW Mode Switching**

When running in the NORMAL 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 set 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.

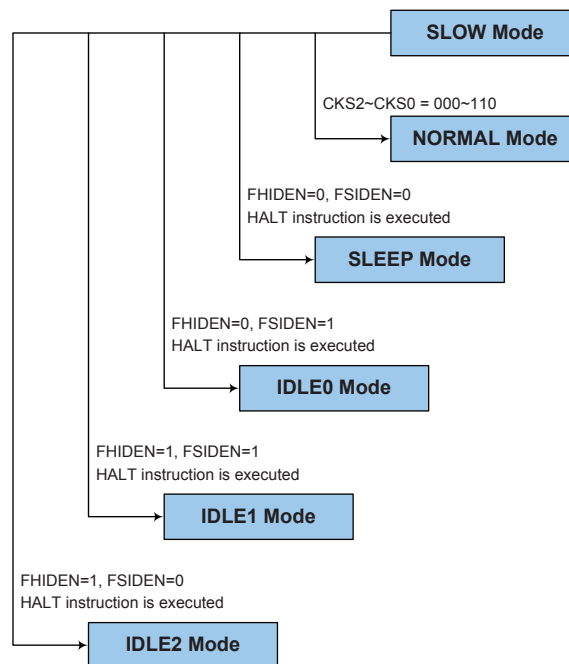
The SLOW Mode is sourced from the LXT or LIRC oscillator determined by the FSS bit in the SCC register and therefore requires this oscillator to be stable before full mode switching occurs.



### SLOW Mode to NORMAL Mode Switching

In SLOW mode the system clock is derived from  $f_{SUB}$ . When system clock is switched back to the NORMAL 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 NORMAL mode from the SLOW Mode. This is monitored using the HXTF bit in the HXTC register or the HIRCF bit in the HIRCC register. The time duration required for the high speed system oscillator stabilization is specified in the A.C. 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". 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 WDT timeout flag TO will be cleared.
- The WDT will be cleared and resume counting as the WDT is enabled. If the WDT is disabled then WDT will be cleared and 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 WDT timeout flag TO will be cleared.
- The WDT will be cleared and resume counting as the WDT is enabled. If the WDT is disabled then WDT will be cleared and 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 the FHIDEN bit in SCC register equal to "1" 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  $f_H$  and  $f_{SUB}$  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 WDT timeout flag TO will be cleared.
- The WDT will be cleared and resume counting as the WDT is enabled. If the WDT is disabled then WDT will be cleared and 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 SCC register equal to "0". When this instruction is executed under the conditions described above, the following will occur:

- The  $f_H$  clock will be on but the  $f_{SUB}$  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 WDT timeout flag TO will be cleared.
- The WDT will be cleared and resume counting as the WDT is enabled. If the WDT is disabled then WDT will be cleared and 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 Mode, 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. These 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 LXT or LIRC oscillator has enabled.

In the IDLE1 and IDLE 2 Mode 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, the PDF flag will be set to 1. The PDF flag will be 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 reset will be initiated and the TO flag will be set to 1. The TO flag is set if a WDT time-out occurs and causes a wake-up that only resets the Program Counter and Stack Pointer, 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 Port 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.

### Programming Considerations

The HXT and LXT oscillators use different SST counter. For example, if the system is woken up from the SLEEP Mode and both the HXT and LXT oscillators need to start-up from an off state.

- If the device is woken up from the SLEEP Mode to the NORMAL Mode, the high speed system oscillator needs an SST period. The device will execute first instruction after HIRCF/HXTF is "1". At this time, the LXT oscillator may not be stability if  $f_{SUB}$  is from LXT oscillator. The same situation occurs in the power-on state. The LXT oscillator is not ready yet when the first instruction is executed.
- There are peripheral functions, such as TMs, for which the  $f_{SYS}$  is used. If the system clock source is switched from  $f_H$  to  $f_{SUB}$ , the clock source to the peripheral functions mentioned above will change accordingly.



## 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 sourced from 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  $2^{18}$  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. This register controls the overall operation of the Watchdog Timer.

#### 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  
 10101: Disable  
 01010: Enable  
 Other values: Reset MCU(The reset operation will be activated after  $t_{SRESET}$  time.)  
 When these bits are changed by the environmental noise or software setting to reset the microcontroller, the reset operation will be activated after 2~3 LIRC clock cycles and the WRF bit in the RSTFC register will be set to 1.

Bit 2~0 **WS2~WS0**: WDT time-out period selection  
 000:  $2^8/f_{LIRC}$   
 001:  $2^{10}/f_{LIRC}$   
 010:  $2^{12}/f_{LIRC}$   
 011:  $2^{14}/f_{LIRC}$   
 100:  $2^{15}/f_{LIRC}$   
 101:  $2^{16}/f_{LIRC}$   
 110:  $2^{17}/f_{LIRC}$   
 111:  $2^{18}/f_{LIRC}$   
 These three bits determine the division ratio of the Watchdog Timer source clock, which in turn determines the timeout period.

#### RSTFC Register

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

"x": unknown

Bit 7~4 Unimplemented, read as "0"  
 Bit 3 **RSTF**: Reset control register software reset flag  
 Described elsewhere.  
 Bit 2 **LVRF**: LVR function reset flag  
 Described elsewhere.

- Bit 1     **LRF**: LVR control register software reset flag  
Described elsewhere.
- Bit 0     **WRF**: WDT control register software reset flag  
0: Not occurred  
1: Occurred

This bit is set to 1 by the WDT control register software reset and cleared by the application program. Note that this bit can only be cleared to 0 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 instructions. If the program malfunctions for whatever reason, jumps to an unknown location, or enters an endless loop, these clear instructions will not be executed in the correct manner, in which case the Watchdog Timer will overflow and reset the device. With regard to the Watchdog Timer enable/disable function, there are five bits, WE4~WE0, in the WDTC register to offer additional 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. The WDT function will be enabled if the WE4~WE0 bits value is equal to 01010B. If the WE4~WE0 bits are set to any other values by the environmental noise or software setting, except 01010B and 10101B, it will reset the device after 2~3 LIRC clock cycles. After power on these bits will have the value of 01010B.

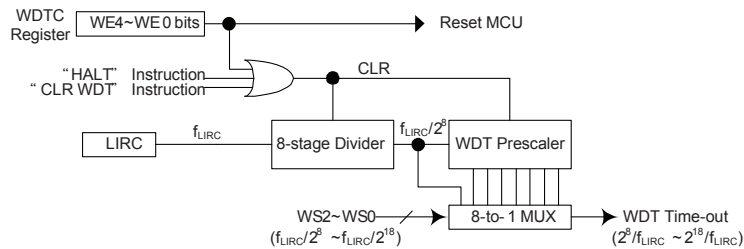
| WE4~WE0 Bits    | WDT Function |
|-----------------|--------------|
| 10101B          | Disable      |
| 01010B          | Enable       |
| Any other value | Reset MCU    |

#### Watchdog Timer Enable/Disable Control

Under normal program operation, a Watchdog Timer time-out will initialise a device reset and set the status bit TO. 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 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 WDT reset, which means a certain value except 01010B and 10101B written into the WE4~WE0 bit filed, the second is using the Watchdog Timer software clear instructions 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  $2^{18}$  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 8 seconds for the  $2^{18}$  division ratio, and a minimum timeout of 7.8ms for the  $2^8$  division ration.



**Watchdog Timer**

## 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.

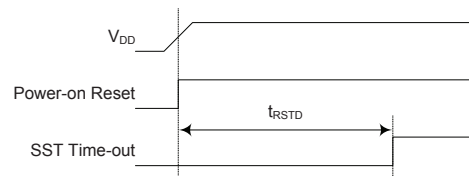
Another type of reset is when the Watchdog Timer overflows and resets. All types of reset operations result in different register conditions being setup. Another reset exists in the form of a Low Voltage Reset, LVR, where a full reset, is implemented in situations where the power supply voltage falls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

### Reset Functions

There are several ways in which a reset can occur, each of which will be described as follows.

#### 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 I/O ports will be first set to inputs.



Note: t<sub>RSTD</sub> is power-on delay with typical time=50 ms

**Power-On Reset Timing Chart**

#### Internal Reset Control

There is an internal reset control register, RSTC, which is used to provide a reset when the device operates abnormally due to the environmental noise interference. If the content of the RSTC register is set to any value other than 01010101B or 10101010B, it will reset the device after 2~3 f<sub>LIRC</sub> clock cycles. After power on the register will have a value of 01010101B.

| RSTC7 ~ RSTC0 Bits | Reset Function |
|--------------------|----------------|
| 01010101B          | No operation   |
| 10101010B          | No operation   |
| Any other value    | Reset MCU      |

**Internal Reset Function Control**

• **RSTC Register**

| Bit  | 7     | 6     | 5     | 4     | 3     | 2     | 1     | 0     |
|------|-------|-------|-------|-------|-------|-------|-------|-------|
| Name | RSTC7 | RSTC6 | RSTC5 | RSTC4 | RSTC3 | RSTC2 | RSTC1 | RSTC0 |
| R/W  | R/W   | R/W   | R/W   | R/W   | R/W   | R/W   | R/W   | R/W   |
| POR  | 0     | 1     | 0     | 1     | 0     | 1     | 0     | 1     |

Bit 7~0 **RSTC7~RSTC0**: Reset function control  
 01010101: No operation  
 10101010: No operation  
 Other values: Reset MCU

If these bits are changed due to adverse environmental conditions, the microcontroller will be reset. The reset operation will be activated after 2~3 LIRC clock cycles and the RSTF bit in the RSTFC register will be set to 1.

• **RSTFC Register**

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

"x": unknown

Bit 7~4 Unimplemented, read as "0"

Bit 3 **RSTF**: Reset control register software reset flag  
 0: Not occurred  
 1: Occurred

This bit is set to 1 by the RSTC control register software reset and cleared by the application program. Note that this bit can only be cleared to 0 by the application program.

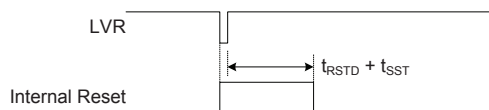
Bit 2 **LVRF**: LVR function reset flag  
 Described elsewhere.

Bit 1 **LRF**: LVR control register software reset flag  
 Described elsewhere.

Bit 0 **WRF**: WDT control register software reset flag  
 Described elsewhere.

**Low Voltage Reset – LVR**

The microcontroller contains a low voltage reset circuit in order to monitor the supply voltage of the device. The LVR function is always enabled with a specific LVR voltage  $V_{LVR}$ . 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, 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 LVD & 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. The actual  $V_{LVR}$  value can be selected by the LVS bits in the LVRC register. If the LVS7~LVS0 bits are changed to some certain values by the environmental noise or software setting, the LVR will reset the device after 2~3 LIRC clock cycles. When this happens, the LRF bit in the RSTFC register will be set to 1. After power on the register will have the value of 01100110B. Note that the LVR function will be automatically disabled when the device enters the power down mode.



Note:  $t_{RSTD}$  is power-on delay with typical time=50 ms

**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    | 1    | 0    | 0    | 1    | 1    | 0    |

Bit 7~0 **LVS7~LVS0**: LVR voltage select  
 01100110: 1.7V  
 01010101: 1.9V  
 00110011: 2.55V  
 10011001: 3.15V  
 10101010: 3.8V  
 11110000: LVR disable

Any other value: Generates MCU reset -- register is reset to POR value

When an actual low voltage condition occurs, as specified by one of the five defined LVR voltage values above, an MCU reset will be generated. In this situation the register contents will remain the same after such a reset occurs.

Any register value, other than the five defined LVR values above, will also result in the generation of an MCU reset. The reset operation will be activated after 2~3 LIRC clock cycles. However in this situation the register contents will be reset to the POR value.

• **RSTFC Register**

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

"x": unknown

Bit 7~4 Unimplemented, read as "0"

Bit 3 **RSTF**: Reset control register software reset flag  
 Described elsewhere.

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

This bit is set to 1 when a specific low voltage reset condition occurs. Note that this bit can only be cleared to 0 by the application program.

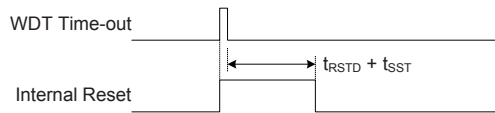
Bit 1 **LRF**: LVR control register software reset flag  
 0: Not occurred  
 1: Occurred

This bit is set to 1 by the LVRC control register contains any undefined LVR voltage register values. This in effect acts like a software-reset function. Note that this bit can only be cleared to 0 by the application program.

Bit 0 **WRF**: WDT control register software reset flag  
 Described elsewhere.

### Watchdog Time-out Reset during Normal Operation

The Watchdog time-out Reset during normal operation is the same as the hardware LVR reset except that the Watchdog time-out flag TO will be set to "1".

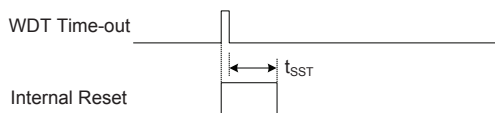


Note:  $t_{rSTD}$  is power-on delay with typical time=16.7 ms

**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 A.C. 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 Normal or SLOW Mode operation          |
| 1  | u   | WDT time-out reset during Normal or SLOW Mode operation |
| 1  | 1   | WDT time-out reset during IDLE or SLEEP Mode operation  |

Note: "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                | Clear 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.

| Register        | Reset<br>(Power On) | LVR Reset<br>(Normal Operation) | WDT Time-out<br>(Normal Operation) | WDT Time-out<br>(IDLE or SLEEP) |
|-----------------|---------------------|---------------------------------|------------------------------------|---------------------------------|
| Program Counter | 0000H               | 0000H                           | 0000H                              | 0000H                           |
| IAR0            | 0000 0000           | 0000 0000                       | 0000 0000                          | uuuu uuuu                       |
| MP0             | 0000 0000           | 0000 0000                       | 0000 0000                          | uuuu uuuu                       |
| IAR1            | 0000 0000           | 0000 0000                       | 0000 0000                          | uuuu uuuu                       |
| MP1L            | 0000 0000           | 0000 0000                       | 0000 0000                          | uuuu uuuu                       |
| MP1H            | 0000 0000           | 0000 0000                       | 0000 0000                          | uuuu uuuu                       |
| ACC             | xxxx xxxx           | uuuu uuuu                       | uuuu uuuu                          | uuuu uuuu                       |
| PCL             | 0000 0000           | 0000 0000                       | 0000 0000                          | 0000 0000                       |
| TBLP            | xxxx xxxx           | uuuu uuuu                       | uuuu uuuu                          | uuuu uuuu                       |
| TBLH            | xxxx xxxx           | uuuu uuuu                       | uuuu uuuu                          | uuuu uuuu                       |
| TBHP            | ---x xxxx           | ---u uuuu                       | ---u uuuu                          | ---u uuuu                       |
| STATUS          | xx00 xxxx           | uuuu uuuu                       | xx1u uuuu                          | uu11 uuuu                       |
| IAR2            | 0000 0000           | 0000 0000                       | 0000 0000                          | uuuu uuuu                       |
| MP2L            | 0000 0000           | 0000 0000                       | 0000 0000                          | uuuu uuuu                       |
| MP2H            | 0000 0000           | 0000 0000                       | 0000 0000                          | uuuu uuuu                       |
| RSTFC           | ---- 0x00           | ---- uuuu                       | ---- uuuu                          | ---- uuuu                       |
| INTC0           | -000 0000           | -000 0000                       | -000 0000                          | -uuu uuuu                       |
| INTC1           | 0000 0000           | 0000 0000                       | 0000 0000                          | uuuu uuuu                       |
| INTC2           | 0000 0000           | 0000 0000                       | 0000 0000                          | uuuu uuuu                       |
| PA              | 1111 1111           | 1111 1111                       | 1111 1111                          | uuuu uuuu                       |
| PAC             | 1111 1111           | 1111 1111                       | 1111 1111                          | uuuu uuuu                       |
| PAPU            | 0000 0000           | 0000 0000                       | 0000 0000                          | uuuu uuuu                       |
| PAWU            | 0000 0000           | 0000 0000                       | 0000 0000                          | uuuu uuuu                       |
| PB              | 1111 1111           | 1111 1111                       | 1111 1111                          | uuuu uuuu                       |
| PBC             | 1111 1111           | 1111 1111                       | 1111 1111                          | uuuu uuuu                       |
| PBPU            | 0000 0000           | 0000 0000                       | 0000 0000                          | uuuu uuuu                       |
| PC              | 1111 1111           | 1111 1111                       | 1111 1111                          | uuuu uuuu                       |
| PCC             | 1111 1111           | 1111 1111                       | 1111 1111                          | uuuu uuuu                       |
| PCPU            | 0000 0000           | 0000 0000                       | 0000 0000                          | uuuu uuuu                       |
| PD              | -111 1111           | -111 1111                       | -111 1111                          | -uuu uuuu                       |
| PDC             | -111 1111           | -111 1111                       | -111 1111                          | -uuu uuuu                       |
| PDPU            | -000 0000           | -000 0000                       | -000 0000                          | -uuu uuuu                       |
| PE              | ---1 1111           | ---1 1111                       | ---1 1111                          | ---u uuuu                       |
| PEC             | ---1 1111           | ---1 1111                       | ---1 1111                          | ---u uuuu                       |
| PEPU            | ---0 0000           | ---0 0000                       | ---0 0000                          | ---u uuuu                       |
| PF              | 1111 1111           | 1111 1111                       | 1111 1111                          | uuuu uuuu                       |
| PFC             | 1111 1111           | 1111 1111                       | 1111 1111                          | uuuu uuuu                       |
| PFPU            | 0000 0000           | 0000 0000                       | 0000 0000                          | uuuu uuuu                       |
| RSTC            | 0101 0101           | 0101 0101                       | 0101 0101                          | uuuu uuuu                       |
| VBGRC           | ---- ---0           | ---- ---0                       | ---- ---0                          | ---- ---u                       |
| MF10            | --00 --00           | --00 --00                       | --00 --00                          | --uu --uu                       |
| MF11            | 0000 0000           | 0000 0000                       | 0000 0000                          | uuuu uuuu                       |
| MF12            | --00 --00           | --00 --00                       | --00 --00                          | --uu --uu                       |
| INTEG           | ---- 0000           | ---- 0000                       | ---- 0000                          | ---- uuuu                       |
| SCC             | 000- 0000           | 000- 0000                       | 000- 0000                          | uuu- uuuu                       |
| HIRCC           | ---- 0001           | ---- 0001                       | ---- 0001                          | ---- uuuu                       |
| HXTC            | ---- -000           | ---- -000                       | ---- -000                          | ---- -uuu                       |

| Register           | Reset<br>(Power On) | LVR Reset<br>(Normal Operation) | WDT Time-out<br>(Normal Operation) | WDT Time-out<br>(IDLE or SLEEP) |
|--------------------|---------------------|---------------------------------|------------------------------------|---------------------------------|
| LXTC               | ---- -000           | ---- -000                       | ---- -000                          | ---- -uuu                       |
| WDTC               | 0101 0011           | 0101 0011                       | 0101 0011                          | uuuu uuuu                       |
| LVRC               | 0110 0110           | 0110 0110                       | 0110 0110                          | uuuu uuuu                       |
| LVDC               | --00 0000           | --00 0000                       | --00 0000                          | --uu uuuu                       |
| LVPUC              | ---- ---0           | ---- ---0                       | ---- ---0                          | ---- ---u                       |
| CMPC               | -000 00--           | -000 00--                       | -000 00--                          | -uuu uu--                       |
| CMPVOS             | -001 0000           | -001 0000                       | -001 0000                          | -uuu uuuu                       |
| CTMC0              | 0000 0---           | 0000 0---                       | 0000 0---                          | uuuu u---                       |
| CTMC1              | 0000 0000           | 0000 0000                       | 0000 0000                          | uuuu uuuu                       |
| CTMDL              | 0000 0000           | 0000 0000                       | 0000 0000                          | uuuu uuuu                       |
| CTMDH              | 0000 0000           | 0000 0000                       | 0000 0000                          | uuuu uuuu                       |
| CTMAL              | 0000 0000           | 0000 0000                       | 0000 0000                          | uuuu uuuu                       |
| CTMAH              | 0000 0000           | 0000 0000                       | 0000 0000                          | uuuu uuuu                       |
| CTMRP              | 0000 0000           | 0000 0000                       | 0000 0000                          | uuuu uuuu                       |
| EEA                | --00 0000           | --00 0000                       | --00 0000                          | --uu uuuu                       |
| EED                | 0000 0000           | 0000 0000                       | 0000 0000                          | uuuu uuuu                       |
| PTMC0              | 0000 0---           | 0000 0---                       | 0000 0---                          | uuuu u---                       |
| PTMC1              | 0000 0000           | 0000 0000                       | 0000 0000                          | uuuu uuuu                       |
| PTMDL              | 0000 0000           | 0000 0000                       | 0000 0000                          | uuuu uuuu                       |
| PTMDH              | ---- --00           | ---- --00                       | ---- --00                          | ---- --uu                       |
| PTMAL              | 0000 0000           | 0000 0000                       | 0000 0000                          | uuuu uuuu                       |
| PTMAH              | ---- --00           | ---- --00                       | ---- --00                          | ---- --uu                       |
| PTMRPL             | 0000 0000           | 0000 0000                       | 0000 0000                          | uuuu uuuu                       |
| PTMRPH             | ---- --00           | ---- --00                       | ---- --00                          | ---- --uu                       |
| STMC0              | 0000 0---           | 0000 0---                       | 0000 0---                          | uuuu u---                       |
| STMC1              | 0000 0000           | 0000 0000                       | 0000 0000                          | uuuu uuuu                       |
| STMDL              | 0000 0000           | 0000 0000                       | 0000 0000                          | uuuu uuuu                       |
| STMDH              | 0000 0000           | 0000 0000                       | 0000 0000                          | uuuu uuuu                       |
| STMAL              | 0000 0000           | 0000 0000                       | 0000 0000                          | uuuu uuuu                       |
| STMAH              | 0000 0000           | 0000 0000                       | 0000 0000                          | uuuu uuuu                       |
| STMRP              | 0000 0000           | 0000 0000                       | 0000 0000                          | uuuu uuuu                       |
| SLEDC0             | 0000 0000           | 0000 0000                       | 0000 0000                          | uuuu uuuu                       |
| SLEDC1             | 0000 0000           | 0000 0000                       | 0000 0000                          | uuuu uuuu                       |
| SLEDC2             | 0000 0000           | 0000 0000                       | 0000 0000                          | uuuu uuuu                       |
| TB0C               | 0--- -000           | 0--- -000                       | 0--- -000                          | u--- -uuu                       |
| TB1C               | 0--- -000           | 0--- -000                       | 0--- -000                          | u--- -uuu                       |
| PSC0R              | ---- --00           | ---- --00                       | ---- --00                          | ---- --uu                       |
| PSC1R              | ---- --00           | ---- --00                       | ---- --00                          | ---- --uu                       |
| SADOL<br>(ADRF5=0) | xxxx ----           | xxxx ----                       | xxxx ----                          | uuuu ----                       |
| SADOL<br>(ADRF5=1) | xxxx xxxxx          | xxxx xxxxx                      | xxxx xxxxx                         | uuuu uuuu                       |
| SADOH<br>(ADRF5=0) | xxxx xxxxx          | xxxx xxxxx                      | xxxx xxxxx                         | uuuu uuuu                       |
| SADOH<br>(ADRF5=1) | ---- xxxxx          | ---- xxxxx                      | ---- xxxxx                         | ---- uuuu                       |
| SADC0              | 0000 0000           | 0000 0000                       | 0000 0000                          | uuuu uuuu                       |



| Register   | Reset<br>(Power On) | LVR Reset<br>(Normal Operation) | WDT Time-out<br>(Normal Operation) | WDT Time-out<br>(IDLE or SLEEP) |
|------------|---------------------|---------------------------------|------------------------------------|---------------------------------|
| SADC1      | 0000 -000           | 0000 -000                       | 0000 -000                          | uuuu -uuu                       |
| SADC2      | 0--0 0000           | 0--0 0000                       | 0--0 0000                          | u--u uuuuu                      |
| SIMC0      | 111- 0000           | 111- 0000                       | 111- 0000                          | uuu- uuuuu                      |
| SIMC1      | 1000 0001           | 1000 0001                       | 1000 0001                          | uuuu uuuu                       |
| SIMD       | xxxx xxxx           | xxxx xxxx                       | xxxx xxxx                          | uuuu uuuu                       |
| SIMA/SIMC2 | 0000 0000           | 0000 0000                       | 0000 0000                          | uuuu uuuu                       |
| SIMTOC     | 0000 0000           | 0000 0000                       | 0000 0000                          | uuuu uuuu                       |
| SCOMC      | -000 ----           | -000 ----                       | -000 ----                          | -uuu ----                       |
| USR        | 0000 1011           | 0000 1011                       | 0000 1011                          | uuuu uuuu                       |
| UCR1       | 0000 00x0           | 0000 00x0                       | 0000 00x0                          | uuuu uuuu                       |
| UCR2       | 0000 0000           | 0000 0000                       | 0000 0000                          | uuuu uuuu                       |
| TXR_RXR    | xxxx xxxx           | xxxx xxxx                       | xxxx xxxx                          | uuuu uuuu                       |
| BRG        | xxxx xxxx           | xxxx xxxx                       | xxxx xxxx                          | uuuu uuuu                       |
| IFS0       | 0000 0000           | 0000 0000                       | 0000 0000                          | uuuu uuuu                       |
| IFS1       | ---- --00           | ---- --00                       | ---- --00                          | ---- --uu                       |
| PAS0       | 0000 0000           | 0000 0000                       | 0000 0000                          | uuuu uuuu                       |
| PAS1       | 0000 0000           | 0000 0000                       | 0000 0000                          | uuuu uuuu                       |
| PBS0       | 0000 0000           | 0000 0000                       | 0000 0000                          | uuuu uuuu                       |
| PBS1       | 0000 0000           | 0000 0000                       | 0000 0000                          | uuuu uuuu                       |
| PCS0       | 0000 0000           | 0000 0000                       | 0000 0000                          | uuuu uuuu                       |
| PCS1       | 0000 0000           | 0000 0000                       | 0000 0000                          | uuuu uuuu                       |
| PDS0       | 0000 0000           | 0000 0000                       | 0000 0000                          | uuuu uuuu                       |
| PDS1       | --00 0000           | --00 0000                       | --00 0000                          | --uu uuuu                       |
| PES0       | 0000 0000           | 0000 0000                       | 0000 0000                          | uuuu uuuu                       |
| PES1       | ---- --00           | ---- --00                       | ---- --00                          | ---- --uu                       |
| PFS0       | 0000 0000           | 0000 0000                       | 0000 0000                          | uuuu uuuu                       |
| PFS1       | 0000 0000           | 0000 0000                       | 0000 0000                          | uuuu uuuu                       |
| EEC        | ---- 0000           | ---- 0000                       | ---- 0000                          | ---- 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~PF. 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            | PC7   | PC6   | PC5   | PC4   | PC3   | PC2   | PC1   | PC0   |
| PCC           | PCC7  | PCC6  | PCC5  | PCC4  | PCC3  | PCC2  | PCC1  | PCC0  |
| PCPU          | PCPU7 | PCPU6 | PCPU5 | PCPU4 | PCPU3 | PCPU2 | PCPU1 | PCPU0 |
| PD            | —     | PD6   | PD5   | PD4   | PD3   | PD2   | PD1   | PD0   |
| PDC           | —     | PDC6  | PDC5  | PDC4  | PDC3  | PDC2  | PDC1  | PDC0  |
| PDPU          | —     | PDPU6 | PDPU5 | PDPU4 | PDPU3 | PDPU2 | PDPU1 | PDPU0 |
| PE            | —     | —     | —     | PE4   | PE3   | PE2   | PE1   | PE0   |
| PEC           | —     | —     | —     | PEC4  | PEC3  | PEC2  | PEC1  | PEC0  |
| PEPU          | —     | —     | —     | PEPU4 | PEPU3 | PEPU2 | PEPU1 | PEPU0 |
| PF            | PF7   | PF6   | PF5   | PF4   | PF3   | PF2   | PF1   | PF0   |
| PFC           | PFC7  | PFC6  | PFC5  | PFC4  | PFC3  | PFC2  | PFC1  | PFC0  |
| PFPU          | PFPU7 | PFPU6 | PFPU5 | PFPU4 | PFPU3 | PFPU2 | PFPU1 | PFPU0 |
| LVPUC         | —     | —     | —     | —     | —     | —     | —     | LVPU  |

"—": 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 an input have the capability of being connected to an internal pull-high resistor. These pull-high resistors are selected using the relevant pull-high control registers PxPU~PFPU and LVPUC and are implemented using weak PMOS transistors. Note that the pull-high resistor can be controlled by the relevant pull-high control registers only when the pin-shared functional pin is selected as an input or NMOS output. Otherwise, the pull-high resistors can not 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     |

**PxPUn:** 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, C, D, E and F. However, the actual available bits for each I/O Port may be different.

### 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:** Low Voltage pull-high resistor control
- 0: All pin pull-high resistors are 30K $\Omega$
  - 1: All pin pull-high resistors are 7.5K $\Omega$

Note that as the pull high resistors are formed using long PMOS transistors, lower operating voltages will result in higher pull high resistor impedances. It is therefore recommended that for lower voltage applications the lower pull high resistor value is chosen.

### Port A Wake-up

The HALT instruction forces the microcontroller into the SLEEP or IDLE Modes 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 registers only when the pin-shared functional pin is selected as general purpose input/output and the MCU enters the Power down 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     |

**PAWUn**: Port A Pin 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~PFC, 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, C, D, E and F. However, the actual available bits for each I/O Port may be different.

## I/O Port Source Current Control

The device supports different source current driving capability for each I/O port. With the corresponding selection registers, SLEDC0, SLEDC1 and SLEDC2, each I/O port can support four levels of the source current driving capability. Users should refer to the D.C. characteristics section to select the desired source current for different applications.

| Register Name | Bit     |         |         |         |         |         |         |         |
|---------------|---------|---------|---------|---------|---------|---------|---------|---------|
|               | 7       | 6       | 5       | 4       | 3       | 2       | 1       | 0       |
| SLEDC0        | SLEDC07 | SLEDC06 | SLEDC05 | SLEDC04 | SLEDC03 | SLEDC02 | SLEDC01 | SLEDC00 |
| SLEDC1        | SLEDC17 | SLEDC16 | SLEDC15 | SLEDC14 | SLEDC13 | SLEDC12 | SLEDC11 | SLEDC10 |
| SLEDC2        | SLEDC27 | SLEDC26 | SLEDC25 | SLEDC24 | SLEDC23 | SLEDC22 | SLEDC21 | SLEDC20 |

I/O Port Source Current Control Registers List

### SLEDC0 Register

| Bit  | 7       | 6       | 5       | 4       | 3       | 2       | 1       | 0       |
|------|---------|---------|---------|---------|---------|---------|---------|---------|
| Name | SLEDC07 | SLEDC06 | SLEDC05 | SLEDC04 | SLEDC03 | SLEDC02 | SLEDC01 | SLEDC00 |
| 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 **SLEDC07~SLEDC06**: PB7~PB4 source current selection  
 00: Source current=Level 0 (min.)  
 01: Source current=Level 1  
 10: Source current=Level 2  
 11: Source current=Level 3 (max.)
- Bit 5~4 **SLEDC05~SLEDC04**: PB3~PB0 source current selection  
 00: Source current=Level 0 (min.)  
 01: Source current=Level 1  
 10: Source current=Level 2  
 11: Source current=Level 3 (max.)
- Bit 3~2 **SLEDC03~SLEDC02**: PA7~PA4 source current selection  
 00: Source current=Level 0 (min.)  
 01: Source current=Level 1  
 10: Source current=Level 2  
 11: Source current=Level 3 (max.)
- Bit 1~0 **SLEDC01~SLEDC00**: PA3~PA0 source current selection  
 00: Source current=Level 0 (min.)  
 01: Source current=Level 1  
 10: Source current=Level 2  
 11: Source current=Level 3 (max.)

### SLEDC1 Register

| Bit  | 7       | 6       | 5       | 4       | 3       | 2       | 1       | 0       |
|------|---------|---------|---------|---------|---------|---------|---------|---------|
| Name | SLEDC17 | SLEDC16 | SLEDC15 | SLEDC14 | SLEDC13 | SLEDC12 | SLEDC11 | SLEDC10 |
| 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 **SLEDC17~SLEDC16**: PD6~PD4 source current select  
 00: source=Level 0 (min.)  
 01: source=Level 1  
 10: source=Level 2  
 11: source=Level 3 (max.)

- Bit 5~4    **SLEDC15~SLEDC14**: PD3~PD0 source current selection  
           00: Source current=Level 0 (min.)  
           01: Source current=Level 1  
           10: Source current=Level 2  
           11: Source current=Level 3 (max.)
- Bit 3~2    **SLEDC13~SLEDC12**: PC7~PC4 source current selection  
           00: Source current=Level 0 (min.)  
           01: Source current=Level 1  
           10: Source current=Level 2  
           11: Source current=Level 3 (max.)
- Bit 1~0    **SLEDC11~SLEDC10**: PC3~PC0 source current selection  
           00: Source current=Level 0 (min.)  
           01: Source current=Level 1  
           10: Source current=Level 2  
           11: Source current=Level 3 (max.)

**SLEDC2 Register**

| Bit  | 7       | 6       | 5       | 4       | 3       | 2       | 1       | 0       |
|------|---------|---------|---------|---------|---------|---------|---------|---------|
| Name | SLEDC27 | SLEDC26 | SLEDC25 | SLEDC24 | SLEDC23 | SLEDC22 | SLEDC21 | SLEDC20 |
| 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    **SLEDC27~SLEDC26**: PF7~PF4 source current select  
           00: source=Level 0 (min.)  
           01: source=Level 1  
           10: source=Level 2  
           11: source=Level 3 (max.)
- Bit 5~4    **SLEDC25~SLEDC24**: PF3~PF0 source current selection  
           00: Source current=Level 0 (min.)  
           01: Source current=Level 1  
           10: Source current=Level 2  
           11: Source current=Level 3 (max.)
- Bit 3~2    **SLEDC23~SLEDC22**: PE4 source current selection  
           00: Source current=Level 0 (min.)  
           01: Source current=Level 1  
           10: Source current=Level 2  
           11: Source current=Level 3 (max.)
- Bit 1~0    **SLEDC21~SLEDC20**: PE3~PE0 source current selection  
           00: Source current=Level 0 (min.)  
           01: Source current=Level 1  
           10: Source current=Level 2  
           11: Source current=Level 3 (max.)

## 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 P<sub>x</sub>S<sub>n</sub>, and Input Function Selection register "i", labeled as IFS<sub>i</sub>, which can select the desired functions of the multi-function pin-shared pins.

When the pin-shared input function is selected to be used, the corresponding input and output functions selection should be properly managed. For example, if the I<sup>2</sup>C SDA line is used, the corresponding output pin-shared function should be configured as the SDI/SDA function by configuring the P<sub>x</sub>S<sub>n</sub> register and the SDA signal input should be properly selected using the IFS<sub>0</sub> register. However, if the external interrupt function is selected to be used, the relevant output pin-shared function should be selected as an I/O function and the interrupt input signal should be selected.

The most important point to note is to make sure that the desired pin-shared function is properly selected and also deselected. 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. 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          | PCS07  | PCS06    | PCS05    | PCS04  | PCS03  | PCS02  | PCS01  | PCS00  |
| PCS1          | PCS17  | PCS16    | PCS15    | PCS14  | PCS13  | PCS12  | PCS11  | PCS10  |
| PDS0          | PDS07  | PDS06    | PDS05    | PDS04  | PDS03  | PDS02  | PDS01  | PDS00  |
| PDS1          | —      | —        | PDS15    | PDS14  | PDS13  | PDS12  | PDS11  | PDS10  |
| PES0          | PES07  | PES06    | PES05    | PES04  | PES03  | PES02  | PES01  | PES00  |
| PES1          | —      | —        | —        | —      | —      | —      | PES11  | PES10  |
| PFS0          | PFS07  | PFS06    | PFS05    | PFS04  | PFS03  | PFS02  | PFS01  | PFS00  |
| PFS1          | PFS17  | PFS16    | PFS15    | PFS14  | PFS13  | PFS12  | PFS11  | PFS10  |
| IFS0          | SCSBPS | SDISDAPS | SCKSCLPS | STPIPS | PTPIPS | STCKPS | CTCKPS | PTCKPS |
| IFS1          | —      | —        | —        | —      | —      | —      | INT1PS | INT0PS |

**Pin-shared Function Selection Registers 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 function selection  
             00: PA3/INT1  
             01: PA3/INT1  
             10: PA3/INT1  
             11: SDO
- Bit 5~4     **PAS05~PAS04:** PA2 pin function selection  
             00: PA2  
             01: PA2  
             10: PA2  
             11: PA2
- Bit 3~2     **PAS03~PAS02:** PA1 pin function selection  
             00: PA1/INT0  
             01: PA1/INT0  
             10: PA1/INT0  
             11:  $\overline{SCS}$
- Bit 1~0     **PAS01~PAS00:** PA0 pin function selection  
             00: PA0  
             01: PA0  
             10: PA0  
             11: PA0

**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 function selection  
             00: PA7/INT1  
             01: PA7/INT1  
             10: PA7/INT1  
             11: TX
- Bit 5~4     **PAS15~PAS14:** PA6 pin function selection  
             00: PA6/INT0  
             01: PA6/INT0  
             10: PA6/INT0  
             11: RX
- Bit 3~2     **PAS13~PAS12:** PA5 pin function selection  
             00: PA5  
             01: PA5  
             10: PA5  
             11: SCK/SCL
- Bit 1~0     **PAS11~PAS10:** PA4 pin function selection  
             00: PA4  
             01: PA4  
             10: PA4  
             11: SDI/SDA



**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 function selection

- 00: PB3
- 01: PB3
- 10: PB3
- 11: CTP

Bit 5~4 **PBS05~PBS04**: PB2 pin function selection

- 00: PB2/PTCK
- 01: PB2/PTCK
- 10: PB2/PTCK
- 11: PTPB

Bit 3~2 **PBS03~PBS02**: PB1 pin function selection

- 00: PB1/PTPI
- 01: PB1/PTPI
- 10: PB1/PTPI
- 11: PTP

Bit 1~0 **PBS01~PBS00**: PB0 pin function selection

- 00: PB0
- 01: PB0
- 10: PB0
- 11: CMPO

**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 function selection

- 00: PB7
- 01: PB7
- 10: PB7
- 11: OSC2

Bit 5~4 **PBS15~PBS14**: PB6 pin function selection

- 00: PB6
- 01: PB6
- 10: PB6
- 11: OSC1

Bit 3~2 **PBS13~PBS12**: PB5 pin function selection

- 00: PB5
- 01: PB5
- 10: PB5
- 11: PB5

Bit 1~0 **PBS11~PBS10**: PB4 pin function selection

- 00: PB4/CTCK
- 01: PB4/CTCK
- 10: PB4/CTCK
- 11: CTPB

**PCS0 Register**

| Bit  | 7     | 6     | 5     | 4     | 3     | 2     | 1     | 0     |
|------|-------|-------|-------|-------|-------|-------|-------|-------|
| Name | PCS07 | PCS06 | PCS05 | PCS04 | PCS03 | PCS02 | PCS01 | PCS00 |
| 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     **PCS07~PCS06:** PC3 pin function selection  
             00: PC3/PTCK  
             01: PC3/PTCK  
             10: PTPB  
             11: AN3
- Bit 5~4     **PCS05~PCS04:** PC2 pin function selection  
             00: PC2/PTPI  
             01: PC2/PTPI  
             10: PTP  
             11: AN2
- Bit 3~2     **PCS03~PCS02:** PC1 pin function selection  
             00: PC1  
             01: CMPO  
             10: VREF  
             11: AN1
- Bit 1~0     **PCS01~PCS00:** PC0 pin function selection  
             00: PC0  
             01: PC0  
             10: VREFI  
             11: AN0

**PCS1 Register**

| Bit  | 7     | 6     | 5     | 4     | 3     | 2     | 1     | 0     |
|------|-------|-------|-------|-------|-------|-------|-------|-------|
| Name | PCS17 | PCS16 | PCS15 | PCS14 | PCS13 | PCS12 | PCS11 | PCS10 |
| 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     **PCS17~PCS16:** PC7 pin function selection  
             00: PC7/STCK  
             01: PC7/STCK  
             10: STPB  
             11: AN7
- Bit 5~4     **PCS15~PCS14:** PC6 pin function selection  
             00: PC6/STPI  
             01: PC6/STPI  
             10: STP  
             11: AN6
- Bit 3~2     **PCS13~PCS12:** PC5 pin function selection  
             00: PC5  
             01: PC5  
             10: PC5  
             11: AN5
- Bit 1~0     **PCS11~PCS10:** PC4 pin function selection  
             00: PC4  
             01: PC4  
             10: PC4  
             11: AN4

**PDS0 Register**

| Bit  | 7     | 6     | 5     | 4     | 3     | 2     | 1     | 0     |
|------|-------|-------|-------|-------|-------|-------|-------|-------|
| Name | PDS07 | PDS06 | PDS05 | PDS04 | PDS03 | PDS02 | PDS01 | PDS00 |
| 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     **PDS07~PDS06**: PD3 pin function selection  
00: PD3  
01: PD3  
10: PD3  
11: AN11
- Bit 5~4     **PDS05~PDS04**: PD2 pin function selection  
00: PD2  
01: PD2  
10: PD2  
11: AN10
- Bit 3~2     **PDS03~PDS02**: PD1 pin function selection  
00: PD1  
01: PD1  
10: PD1  
11: AN9
- Bit 1~0     **PDS01~PDS00**: PD0 pin function selection  
00: PD0  
01: PD0  
10: PD0  
11: AN8

**PDS1 Register**

| Bit  | 7 | 6 | 5     | 4     | 3     | 2     | 1     | 0     |
|------|---|---|-------|-------|-------|-------|-------|-------|
| Name | — | — | PDS15 | PDS14 | PDS13 | PDS12 | PDS11 | PDS10 |
| 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     **PDS15~PDS14**: PD6 pin function selection  
00: PD6  
01: PD6  
10: PD6  
11: AN14
- Bit 3~2     **PDS13~PDS12**: PD5 pin function selection  
00: PD5  
01: PD5  
10: PD5  
11: AN13
- Bit 1~0     **PDS11~PDS10**: PD4 pin function selection  
00: PD4  
01: PD4  
10: PD4  
11: AN12

**PES0 Register**

| Bit  | 7     | 6     | 5     | 4     | 3     | 2     | 1     | 0     |
|------|-------|-------|-------|-------|-------|-------|-------|-------|
| Name | PES07 | PES06 | PES05 | PES04 | PES03 | PES02 | PES01 | PES00 |
| 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     **PES07~PES06:** PE3 pin function selection  
             00: PE3  
             01: PE3  
             10: PE3  
             11: CTP
- Bit 5~4     **PES05~PES04:** PE2 pin function selection  
             00: PE2/CTCK  
             01: PE2/CTCK  
             10: PE2/CTCK  
             11: CTPB
- Bit 3~2     **PES03~PES02:** PE1 pin function selection  
             00: PE1/STPI  
             01: PE1/STPI  
             10: PE1/STPI  
             11: STP
- Bit 1~0     **PES01~PES00:** PE0 pin function selection  
             00: PE0/STCK  
             01: PE0/STCK  
             10: PE0/STCK  
             11: STPB

**PES1 Register**

| Bit  | 7 | 6 | 5 | 4 | 3 | 2 | 1     | 0     |
|------|---|---|---|---|---|---|-------|-------|
| Name | — | — | — | — | — | — | PES11 | PES10 |
| R/W  | — | — | — | — | — | — | R/W   | R/W   |
| POR  | — | — | — | — | — | — | 0     | 0     |

- Bit 7~2     Unimplemented, read as "0"
- Bit 1~0     **PES11~PES10:** PE4 pin function selection  
             00: PE4  
             01: PE4  
             10: PE4  
             11: PE4

**PFS0 Register**

| Bit  | 7     | 6     | 5     | 4     | 3     | 2     | 1     | 0     |
|------|-------|-------|-------|-------|-------|-------|-------|-------|
| Name | PFS07 | PFS06 | PFS05 | PFS04 | PFS03 | PFS02 | PFS01 | PFS00 |
| 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     **PFS07~PFS06:** PF3 pin function selection  
             00: PF3  
             01: PF3  
             10: SCK/SCL  
             11: SCOM3

- Bit 5~4    **PFS05~PFS04**: PF2 pin function selection
  - 00: PF2
  - 01: PF2
  - 10: SDI/SDA
  - 11: SCOM2
- Bit 3~2    **PFS03~PFS02**: PF1 pin function selection
  - 00: PF1
  - 01: PF1
  - 10: SDO
  - 11: SCOM1
- Bit 1~0    **PFS01~PFS00**: PF0 pin function selection
  - 00: PF0
  - 01: PF0
  - 10: SCS
  - 11: SCOM0

**PFS1 Register**

| Bit  | 7     | 6     | 5     | 4     | 3     | 2     | 1     | 0     |
|------|-------|-------|-------|-------|-------|-------|-------|-------|
| Name | PFS17 | PFS16 | PFS15 | PFS14 | PFS13 | PFS12 | PFS11 | PFS10 |
| 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    **PFS17~PFS16**: PF7 pin function selection
  - 00: PF7
  - 01: PF7
  - 10: PF7
  - 11: CMPINP
- Bit 5~4    **PFS15~PFS14**: PF6 pin function selection
  - 00: PF6
  - 01: PF6
  - 10: CMPINN
  - 11: AN15
- Bit 3~2    **PFS13~PFS12**: PF5 pin function selection
  - 00: PF5
  - 01: PF5
  - 10: PF5
  - 11: XT1
- Bit 1~0    **PFS11~PFS10**: PF4 pin function selection
  - 00: PF4
  - 01: PF4
  - 10: PF4
  - 11: XT2

**IFS0 Register**

| Bit  | 7      | 6        | 5        | 4      | 3      | 2      | 1      | 0      |
|------|--------|----------|----------|--------|--------|--------|--------|--------|
| Name | SCSBPS | SDISDAPS | SCKSCLPS | STPIPS | PTPIPS | STCKPS | CTCKPS | PTCKPS |
| 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      **SCSBPS**: SCS input source pin selection  
0: PA1  
1: PF0
- Bit 6      **SDISDAPS**: SDI/SDA input source pin selection  
0: PA4  
1: PF2
- Bit 5      **SCKSCLPS**: SCK/SCL input source pin selection  
0: PA5  
1: PF3
- Bit 4      **STPIPS**: STPI input source pin selection  
0: PC6  
1: PE1
- Bit 3      **PTPIPS**: PTPI input source pin selection  
0: PC2  
1: PB1
- Bit 2      **STCKPS**: STCK input source pin selection  
0: PC7  
1: PE0
- Bit 1      **CTCKPS**: CTCK input source pin selection  
0: PB4  
1: PE2
- Bit 0      **PTCKPS**: PTCK input source pin selection  
0: PC3  
1: PB2

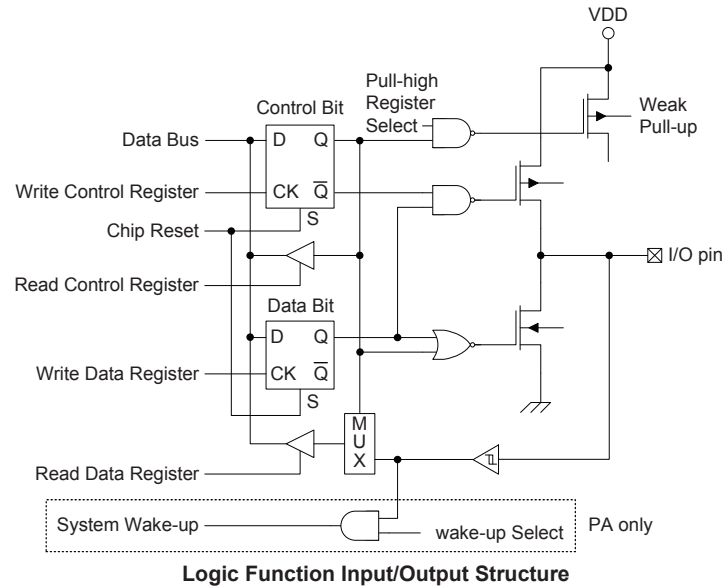
**IFS1 Register**

| Bit  | 7 | 6 | 5 | 4 | 3 | 2 | 1      | 0      |
|------|---|---|---|---|---|---|--------|--------|
| Name | — | — | — | — | — | — | INT1PS | INT0PS |
| R/W  | — | — | — | — | — | — | R/W    | R/W    |
| POR  | — | — | — | — | — | — | 0      | 0      |

- Bit 7~2    Unimplemented, read as "0"
- Bit 1      **INT1PS**: INT1 input source pin selection  
0: PA3  
1: PA7
- Bit 0      **INT0PS**: INT0 input source pin selection  
0: PA1  
1: PA6

## I/O Pin Structures

The accompanying diagram illustrates the internal structure 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.



## 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, PAC~PFC, 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, PA~PF, 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 devices is the ability to control and measure time. To implement time related functions the device includes several Timer Modules, generally abbreviated to the name TM. The TMs are multi-purpose timing units and serve to provide operations such as Timer/Counter, Input Capture, 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 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, Standard and Periodic TM sections.

### Introduction

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

| TM Function                  | CTM            | STM            | PTM            |
|------------------------------|----------------|----------------|----------------|
| Timer/Counter                | √              | √              | √              |
| Input Capture                | —              | √              | √              |
| Compare Match Output         | √              | √              | √              |
| PWM Channels                 | 1              | 1              | 1              |
| Single Pulse Output          | —              | 1              | 1              |
| PWM Alignment                | Edge           | Edge           | Edge           |
| PWM Adjustment Period & Duty | Duty or Period | 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 count-up 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, S or P 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 for event counting.



### TM Interrupts

The Compact Type, Standard Type and Periodic 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 or two TM input pins, with the label xTCK and xTPI respectively. 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 and PTCK pins are also used as the external trigger input pin in single pulse output mode for the STM and PTM respectively.

The other xTM input pin, STPI or PTPI, is the capture input whose active edge can be a rising edge, a falling edge or both rising and falling edges and the active edge transition type is selected using the STIO1~STIO0 or PTIO1~PTIO0 bits in the STMC1 or PTMC1 register respectively. There is another capture input, PTCK, for PTM capture input mode, which can be used as the external trigger input source except the PTPI pin.

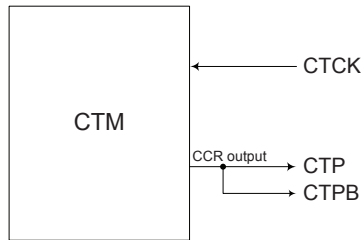
The TMs each have two output pins, xTP and xTPB. The xTPB is the inverted signal of the xTP output. The TM output pins can be selected using the corresponding pin-shared function selection bits described in the Pin-shared Function section. When the TM is in the Compare Match Output Mode, these pins can be controlled by the TM to switch to a high or low level or to toggle when a compare match situation occurs. The external xTP or xTPB output pin is also the pin where the TM generates the PWM output waveform. As the TM output pins are pin-shared with other functions, the TM output function must first be setup using relevant pin-shared function selection register.

| CTM   |          | STM        |           | PTM        |           |
|-------|----------|------------|-----------|------------|-----------|
| Input | Output   | Input      | Output    | Input      | Output    |
| CTCK  | CTP,CTPB | STCK, STPI | STP, STPB | PTCK, PTPI | PTP, PTPB |

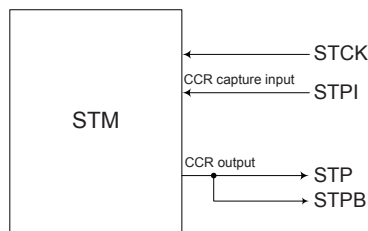
**TM External Pins**

**TM Input/Output Pin Selection**

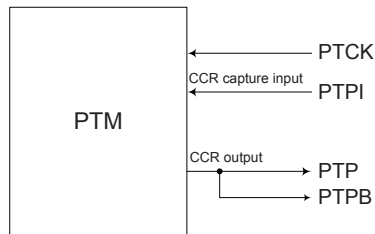
Selecting to have a TM input/output or whether to retain its other shared function is implemented using the relevant pin-shared function selection registers, with the corresponding selection bits in each pin-shared function register corresponding to a TM input/output pin. Configuring the selection bits correctly will setup the corresponding pin as a TM input/output. The details of the pin-shared function selection are described in the pin-shared function section.



**CTM Function Pin Control Block Diagram**



**STM Function Pin Control Block Diagram**

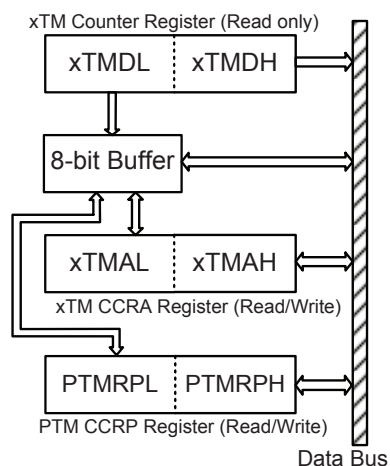


**PTM Function Pin Control Block Diagram**

## Programming Considerations

The TM Counter Registers and the Capture/Compare CCRA and CCRP 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 and CCRP 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 and CCRP low byte registers, named xTMAL and PTMRPL, using the following access procedures. Accessing the CCRA or CCRP low byte registers without following these access procedures will result in unpredictable values.



The following steps show the read and write procedures:

- Writing Data to CCRA or CCRP
  - ♦ Step 1. Write data to Low Byte xTMAL or PTMRPL
    - note that here data is only written to the 8-bit buffer.
  - ♦ Step 2. Write data to High Byte xTMAH or PTMRPH
    - 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 or CCRP
  - ♦ Step 1. Read data from the High Byte xTMnDH, xTMAH or PTMRPH
    - 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 xTMnDL, xTMAL or PTMRPL
    - this step reads data from the 8-bit buffer.

## Compact Type TM – CTM

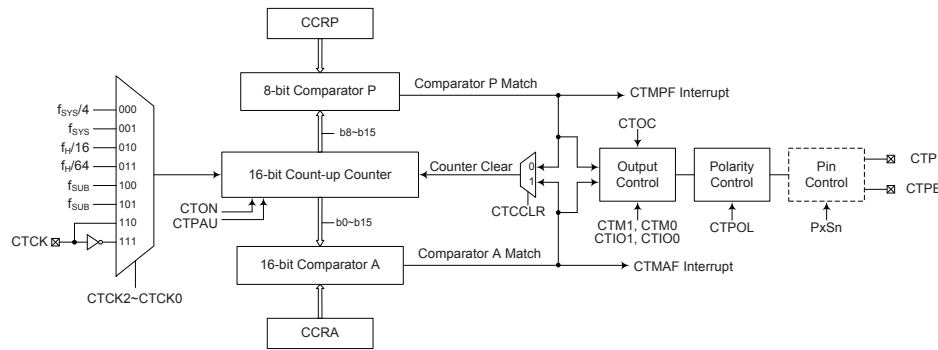
Although the simplest form of the three TM types, the Compact TM type still contains three operating modes, which are Compare Match Output, Timer/Event Counter and PWM Output modes. The Compact TM can also be controlled with an external input pin and can drive two external output pin.

| CTM Core   | CTM Input Pin | CTM Output Pin |
|------------|---------------|----------------|
| 16-bit CTM | CTCK          | CTP, CTPB      |

### Compact TM Operation

At its core is a 16-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 8-bit wide whose value is compared with the highest eight bits in the counter while the CCRA is 16-bit wide and therefore compares with all counter bits.

The only way of changing the value of the 16-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 an output pin. All operating setup conditions are selected using relevant internal registers.



**Compact Type TM Block Diagram**

### Compact Type TM Register Description

Overall operation of the Compact TM is controlled using a series of registers. A read only register pair exists to store the internal counter 16-bit value, while a read/write register pair exists to store the internal 16-bit CCRA value. The CTMRP register is used to store the 8-bit CCRP value. The remaining two registers are control registers which setup the different operating and control modes.

| Register Name | Bit   |       |       |       |      |       |       |        |
|---------------|-------|-------|-------|-------|------|-------|-------|--------|
|               | 7     | 6     | 5     | 4     | 3    | 2     | 1     | 0      |
| CTMC0         | CTPAU | CTCK2 | CTCK1 | CTCK0 | CTON | —     | —     | —      |
| CTMC1         | CTM1  | CTM0  | CTIO1 | CTIO0 | CTOC | CTPOL | CTDPX | CTCCLR |
| CTMDL         | D7    | D6    | D5    | D4    | D3   | D2    | D1    | D0     |
| CTMDH         | D15   | D14   | D13   | D12   | D11  | D10   | D9    | D8     |
| CTMAL         | D7    | D6    | D5    | D4    | D3   | D2    | D1    | D0     |
| CTMAH         | D15   | D14   | D13   | D12   | D11  | D10   | D9    | D8     |
| CTMRP         | D7    | D6    | D5    | D4    | D3   | D2    | D1    | D0     |

**16-bit Compact TM Registers List**

**CTMC0 Register**

| Bit  | 7     | 6     | 5     | 4     | 3    | 2 | 1 | 0 |
|------|-------|-------|-------|-------|------|---|---|---|
| Name | CTPAU | CTCK2 | CTCK1 | CTCK0 | CTON | — | — | — |
| R/W  | R/W   | R/W   | R/W   | R/W   | R/W  | — | — | — |
| POR  | 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 oscillator 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 while clearing the bit 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 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** Unimplemented, read as "0"

**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:** Select CTM Operating Mode  
 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**: Select CTM function

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 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 only 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**: CTP Output control

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 CTM output pin. Its operation depends upon whether CTM is being used in the Compare Match Output Mode or in the PWM 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 Mode it determines if the PWM signal is active high or active low.

Bit 2     **CTPOL**: CTP Output polarity control

- 0: Non-inverted
- 1: Inverted

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 TM is in the Timer/Counter Mode.

- Bit 1     **CTDPX**: CTM 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     **CTCCLR**: CTM Counter Clear condition selection  
 0: CTM Comparator P match  
 1: CTM Comparator 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     CTM Counter Low Byte Register bit 7 ~ bit 0  
 CTM 16-bit Counter bit 7 ~ bit 0

**CTMDH Register**

| Bit  | 7   | 6   | 5   | 4   | 3   | 2   | 1  | 0  |
|------|-----|-----|-----|-----|-----|-----|----|----|
| Name | D15 | D14 | D13 | D12 | D11 | D10 | D9 | D8 |
| R/W  | R   | R   | R   | R   | R   | R   | R  | R  |
| POR  | 0   | 0   | 0   | 0   | 0   | 0   | 0  | 0  |

- Bit 7~0     CTM Counter High Byte Register bit 7 ~ bit 0  
 CTM 16-bit Counter bit 15 ~ 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     CTM CCRA Low Byte Register bit 7 ~ bit 0  
 CTM 16-bit CCRA bit 7 ~ bit 0

**CTMAH Register**

| Bit  | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   |
|------|-----|-----|-----|-----|-----|-----|-----|-----|
| Name | D15 | D14 | D13 | D12 | D11 | D10 | D9  | D8  |
| 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     CTM CCRA High Byte Register bit 7 ~ bit 0  
 CTM 16-bit CCRA bit 15 ~ bit 8

### CTMRP 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      CTM CCRP Register bit 7 ~ bit 0  
 CTM CCRP 8-bit register, compared with the CTM Counter bit 15 ~ bit 8.  
 Comparator P Match Period  
 0: 65536 CTM clocks  
 1~255: 256 x (1~255) CTM clocks

These eight bits are used to setup the value on the internal CCRP 8-bit register, which are then compared with the internal counter's highest eight 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 eight counter bits, the compare values exist in 256 clock cycle multiples. Clearing all eight bits to zero is in effect allowing the counter to overflow at its maximum value.

### Compact Type TM Operating Modes

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

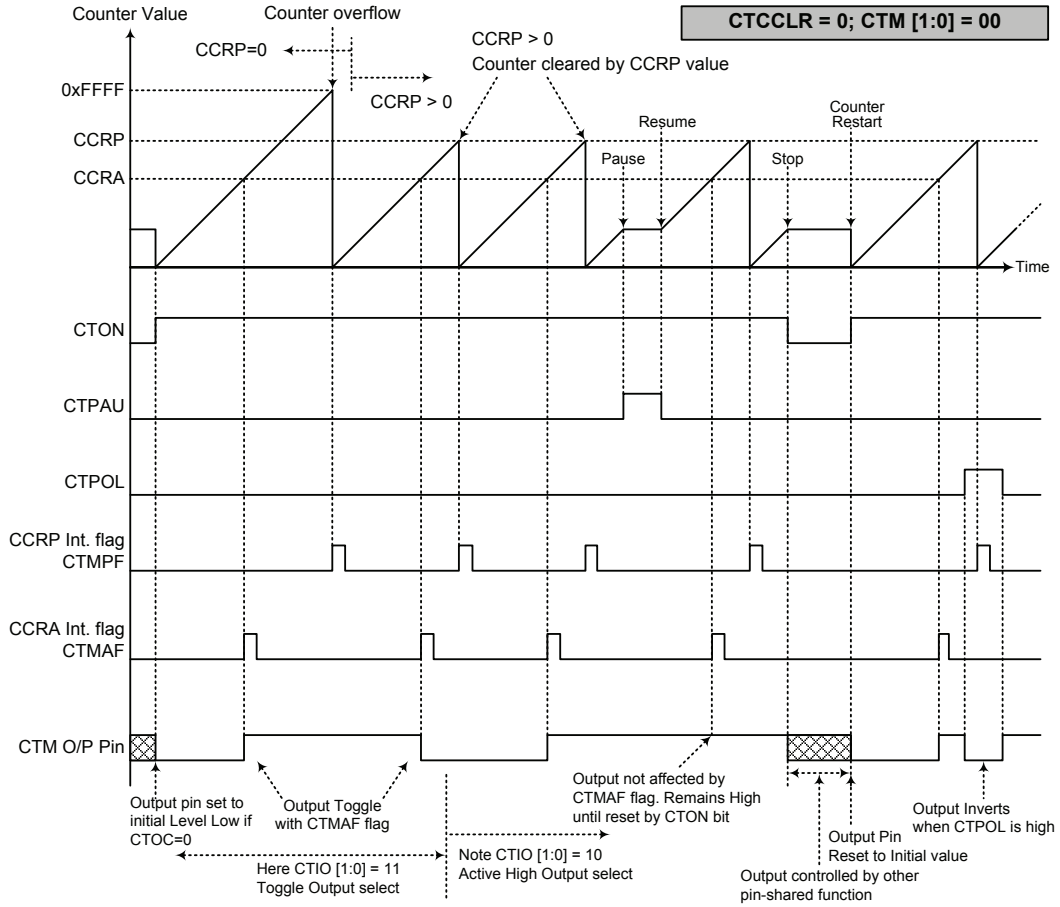
#### Compare Match Output Mode

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 occurs 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 the 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 16-bit, FFFF Hex, value, however here the CTMAF interrupt request flag will not be generated.

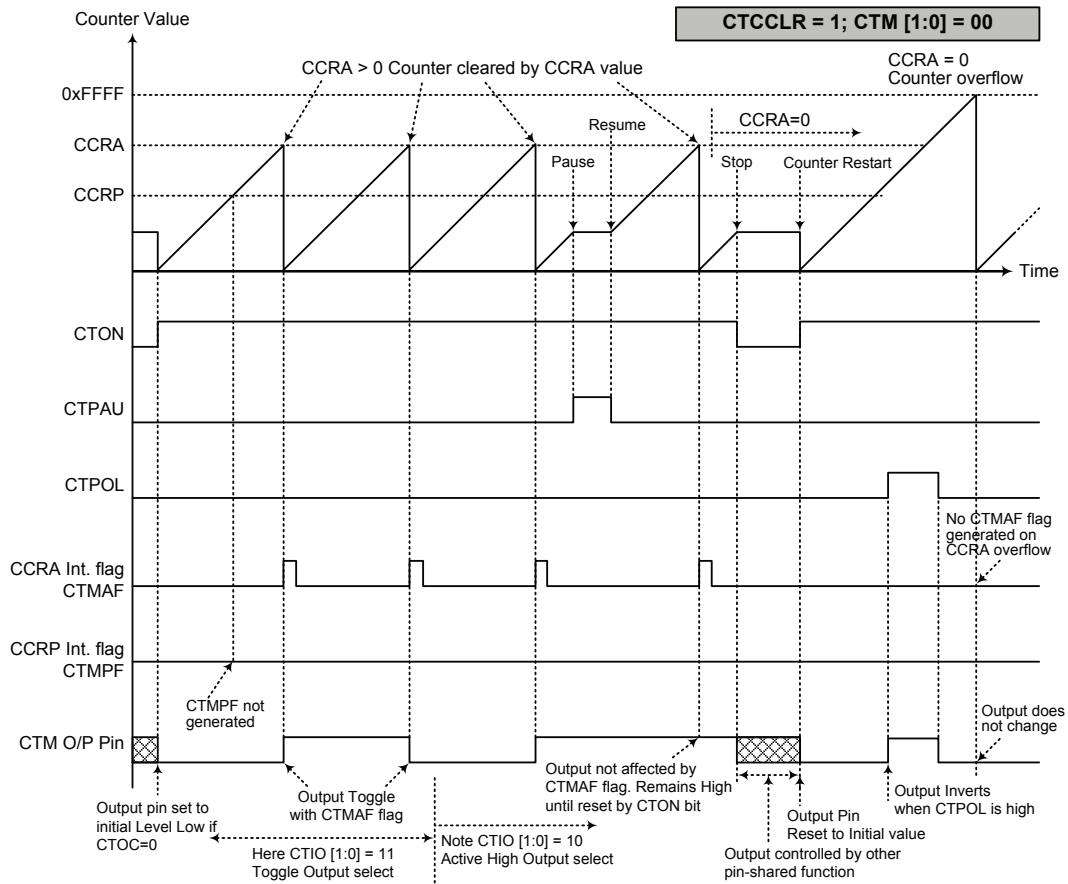
As the name of the mode suggests, after a comparison is made, the CTM output pin will change state. The CTM output pin condition however only changes state when a CTMAF interrupt request flag is generated after a compare match occurs from Comparator A. The CTMPF interrupt request flag, generated from 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 is controlled only by the CTMAF flag  
 3. The output pin is reset to its 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 is controlled only by the CTMAF flag  
 3. The output pin is reset to its initial state by a CTON bit rising edge  
 4. The CTMPF flag 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 function.

**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 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 CTDPX 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 TM output pin to a fixed high or low level. The CTPOL bit is used to reverse the polarity of the PWM output waveform.

• **16-bit CTM, PWM Mode, Edge-aligned Mode, CTDPX=0**

| CCRP   | 1~255    | 0     |
|--------|----------|-------|
| Period | CCRP×256 | 65536 |
| Duty   | CCRA     |       |

If  $f_{SYS}=16\text{MHz}$ , CTM clock source select  $f_{SYS}/4$ , CCRP=2 and CCRA=128,

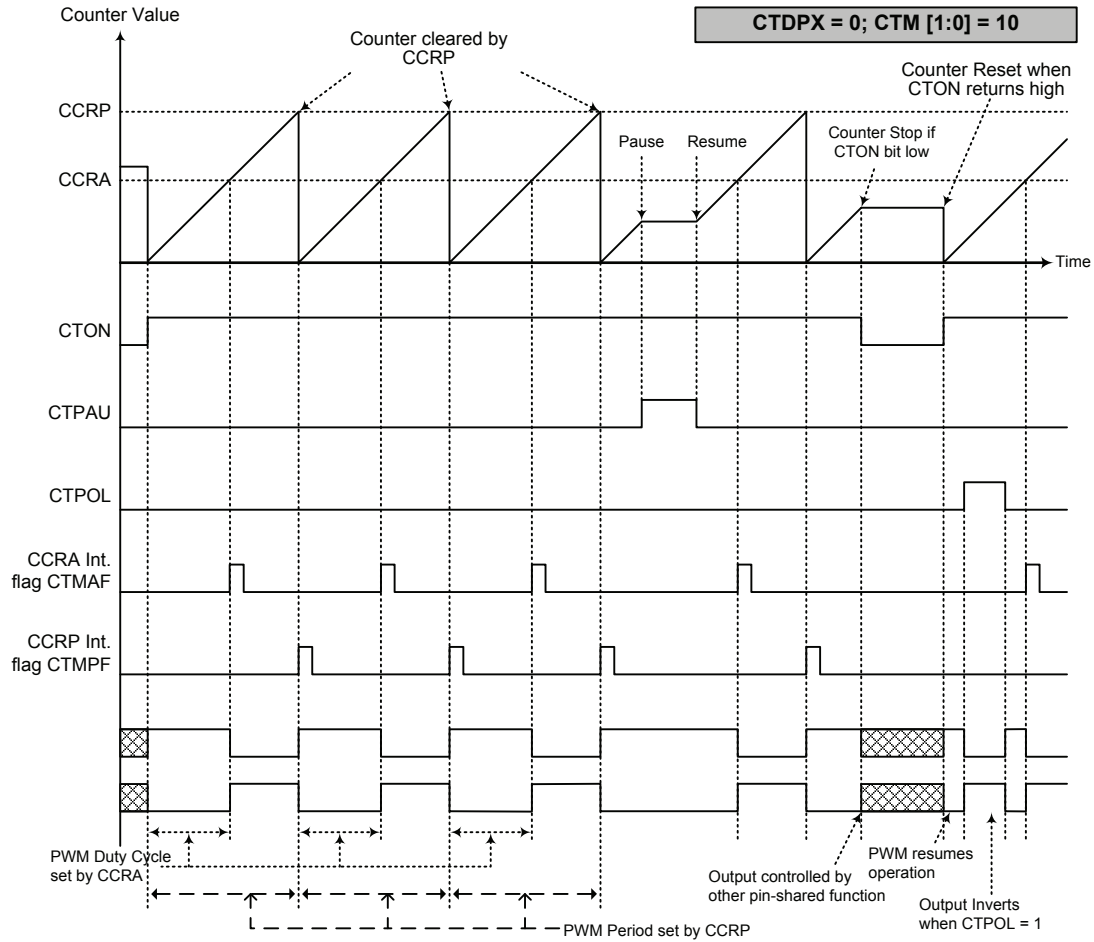
The CTM PWM output frequency= $(f_{SYS}/4)/(2 \times 256)=f_{SYS}/2048=7.8125\text{kHz}$ , duty= $128/(2 \times 256)=25\%$ .

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%.

• **16-bit CTM, PWM Mode, Edge-aligned Mode, CTDPX=1**

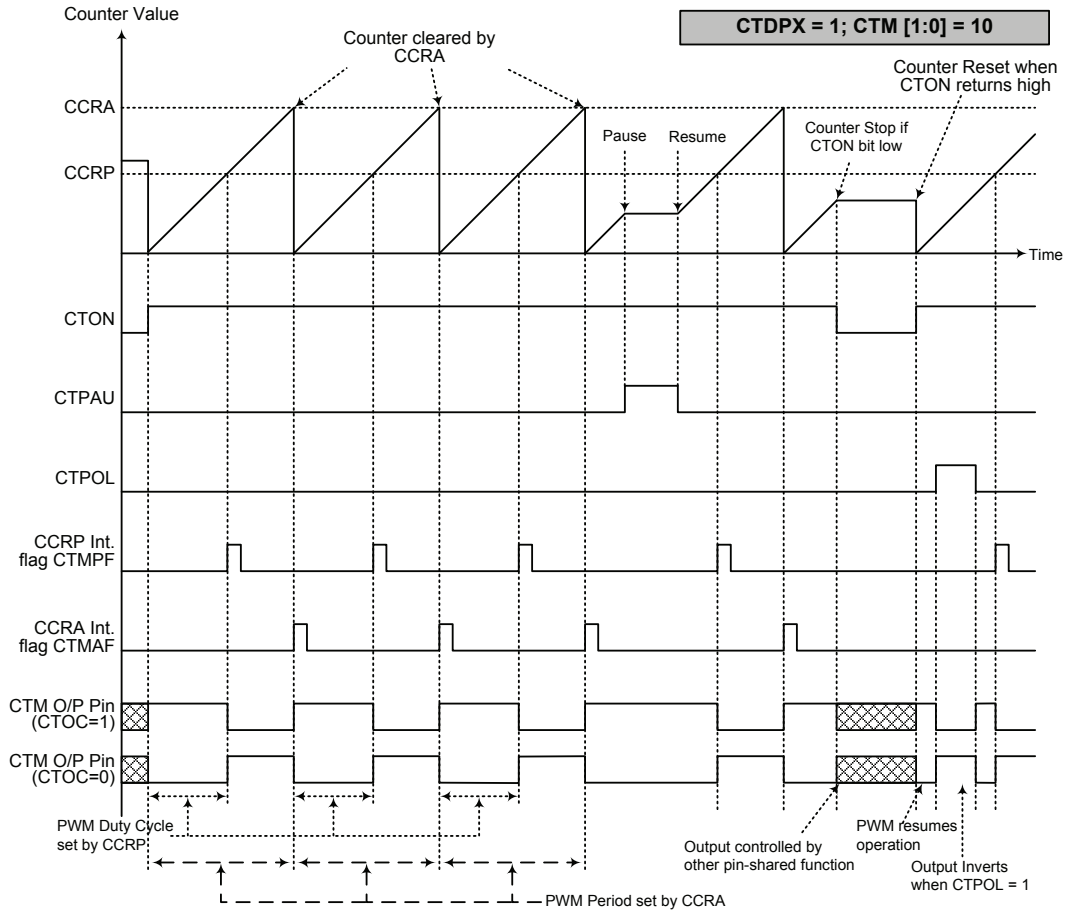
| CCRP   | 1~255    | 0     |
|--------|----------|-------|
| Period | CCRA     |       |
| Duty   | CCRP×256 | 65536 |

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 except when the CCRP value is equal to 0.



**PWM Mode – CTDPX=0**

- Note: 1. Here CTDPX=0 – Counter cleared by CCRP  
 2. A counter clear sets the 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



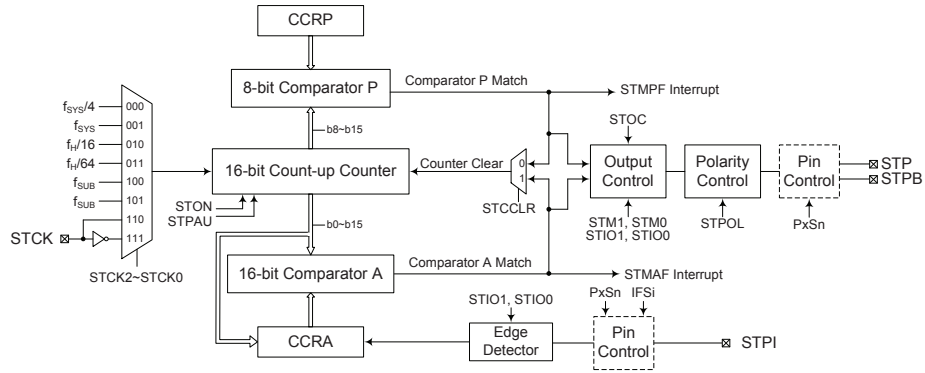
**PWM Mode – CTD PX=1**

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

## Standard Type TM – STM

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

| STM Core   | STM Input Pin | STM Output Pin |
|------------|---------------|----------------|
| 16-bit STM | STCK, STPI    | STP, STPB      |



**Standard Type TM Block Diagram**

### Standard TM Operation

The size of Standard TM is 16-bit wide and its core is a 16-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 comparator is 8-bit wide whose value is compared the with highest 8 bits in the counter while the CCRA is the sixteen bits and therefore compares all counter bits.

The only way of changing the value of the 16-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 an output pin. 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 16-bit value, while a read/write register pair exists to store the internal 16-bit CCRA value. The STMRP register is used to store the 8-bit CCRP value. The remaining two registers are control registers which setup the different operating and control modes.

| Register Name | Bit   |       |       |       |      |       |       |        |
|---------------|-------|-------|-------|-------|------|-------|-------|--------|
|               | 7     | 6     | 5     | 4     | 3    | 2     | 1     | 0      |
| STMC0         | STPAU | STCK2 | STCK1 | STCK0 | STON | —     | —     | —      |
| STMC1         | STM1  | STM0  | STIO1 | STIO0 | STOC | STPOL | STDPX | STCCLR |
| STMDL         | D7    | D6    | D5    | D4    | D3   | D2    | D1    | D0     |
| STMDH         | D15   | D14   | D13   | D12   | D11  | D10   | D9    | D8     |
| STMAL         | D7    | D6    | D5    | D4    | D3   | D2    | D1    | D0     |
| STMAH         | D15   | D14   | D13   | D12   | D11  | D10   | D9    | D8     |
| STMRP         | D7    | D6    | D5    | D4    | D3   | D2    | D1    | D0     |

16-bit Standard TM Registers List

#### STMC0 Register

| Bit  | 7     | 6     | 5     | 4     | 3    | 2 | 1 | 0 |
|------|-------|-------|-------|-------|------|---|---|---|
| Name | STPAU | STCK2 | STCK1 | STCK0 | STON | — | — | — |
| R/W  | R/W   | R/W   | R/W   | R/W   | R/W  | — | — | — |
| POR  | 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**: Select STM Counter clock  
 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 oscillator 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 or the 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      Unimplemented, read as "0"

**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**: Select STM Operating Mode  
             00: Compare Match Output Mode  
             01: Capture Input Mode  
             10: PWM Mode or Single Pulse Output Mode  
             11: Timer/Counter Mode

These bits setup 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**: Select STM function  
             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

            Capture Input Mode  
             00: Input capture at rising edge of STPI  
             01: Input capture at falling edge of STPI  
             10: Input capture at rising/falling edge of STPI  
             11: Input capture disabled

            Timer/Counter Mode  
             Unused

These two bits are used to determine how the STM output 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 TM 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 STM output pin should be setup 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 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 only 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 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 Mode/Single Pulse Output Mode it determines if the PWM signal is active high or active low.

- Bit 2 **STPOL**: STM STP Output polarity control  
     0: Non-inverted  
     1: Inverted

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, Single Pulse Output or Capture Input 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 STM Counter Low Byte Register bit 7 ~ bit 0  
 STM 16-bit Counter bit 7 ~ bit 0

**STMDH Register**

| Bit  | 7   | 6   | 5   | 4   | 3   | 2   | 1  | 0  |
|------|-----|-----|-----|-----|-----|-----|----|----|
| Name | D15 | D14 | D13 | D12 | D11 | D10 | D9 | D8 |
| R/W  | R   | R   | R   | R   | R   | R   | R  | R  |
| POR  | 0   | 0   | 0   | 0   | 0   | 0   | 0  | 0  |

Bit 7~0      STM Counter High Byte Register bit 7 ~ bit 0  
 STM 16-bit Counter bit 15 ~ 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      STM CCRA Low Byte Register bit 7 ~ bit 0  
 STM 16-bit CCRA bit 7 ~ bit 0

**STMAH Register**

| Bit  | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   |
|------|-----|-----|-----|-----|-----|-----|-----|-----|
| Name | D15 | D14 | D13 | D12 | D11 | D10 | D9  | D8  |
| 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      STM CCRA High Byte Register bit 7 ~ bit 0  
 STM 16-bit CCRA bit 15 ~ bit 8

**STM RP 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      STM CCRP Register bit 7 ~ bit 0  
 STM CCRP 8-bit register, compared with the STM counter bit 15~bit 8  
 Comparator P match period  
 0: 65536 STM clocks  
 1~255: (1~255) × 256 STM clocks

These eight bits are used to setup the value on the internal CCRP 8-bit register, which are then compared with the internal counter's highest eight 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 eight counter bits, the compare values exist in 256 clock cycle multiples. Clearing all eight bits to zero is in effect allowing the counter to overflow at its maximum value.

## **Standard Type TM Operation Modes**

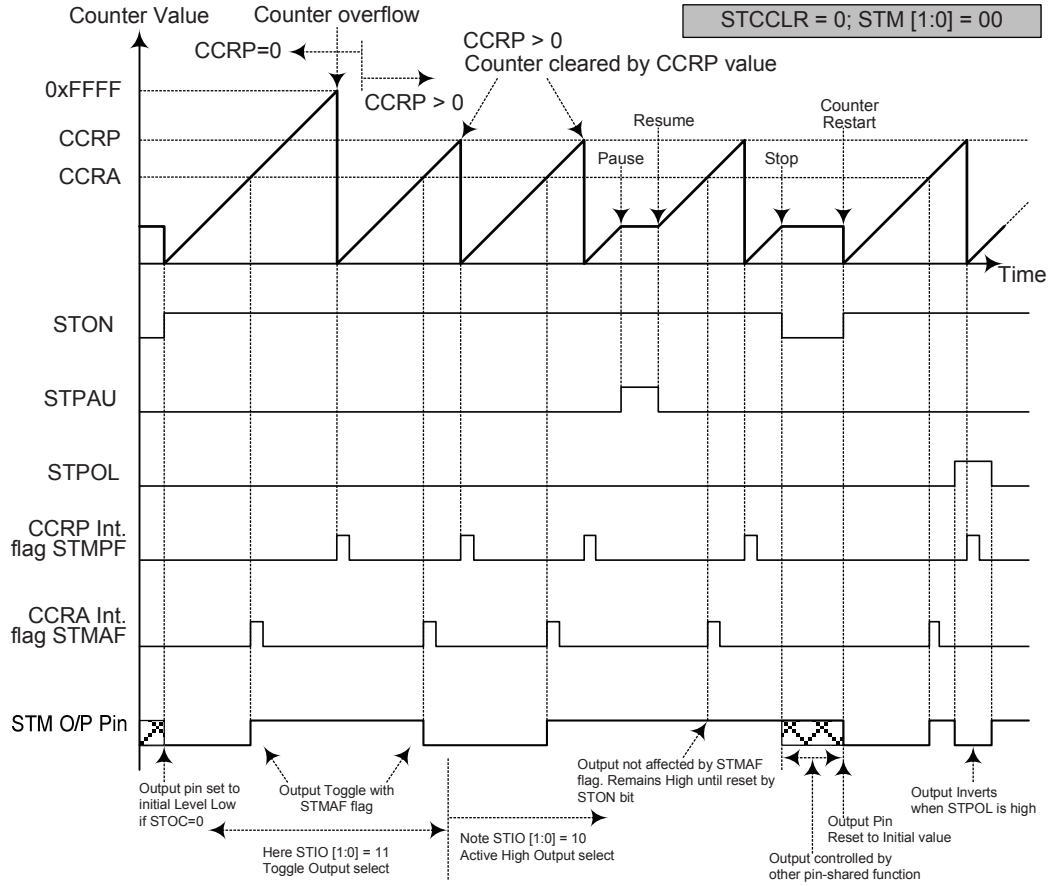
The Standard Type TM can operate in one of five operating modes, Compare Match Output Mode, PWM Output Mode, Single Pulse Output Mode, Capture Input Mode or Timer/Counter Mode. The operating mode is selected using the STM1 and STM0 bits in the STMC1 register.

### **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.

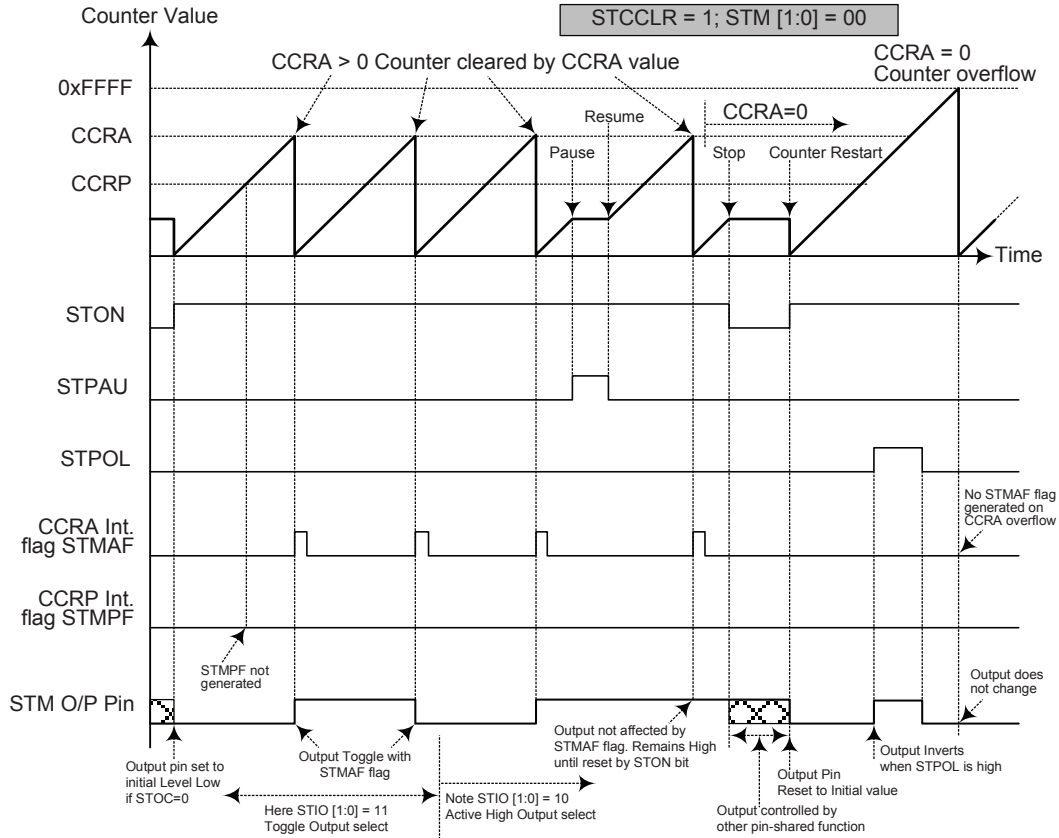
If the STCCLR bit in the STMC1 register is high then the counter will be cleared when a compare match occurs from Comparator A. However, here only the STMAF interrupt request flag will be generated even if the value of the CCRP bits is less than that of the CCRA registers. Therefore when STCCLR is high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

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 setup after the STON bit changes from low to high, is setup 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. A 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 function.

**PWM 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 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 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.

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

| CCRP   | 1~255    | 0     |
|--------|----------|-------|
| Period | CCRP×256 | 65536 |
| Duty   | CCRA     |       |

If  $f_{SYS}=16\text{MHz}$ , STM clock source is  $f_{SYS}/4$ ,  $CCRP=2$  and  $CCRA=128$ ,

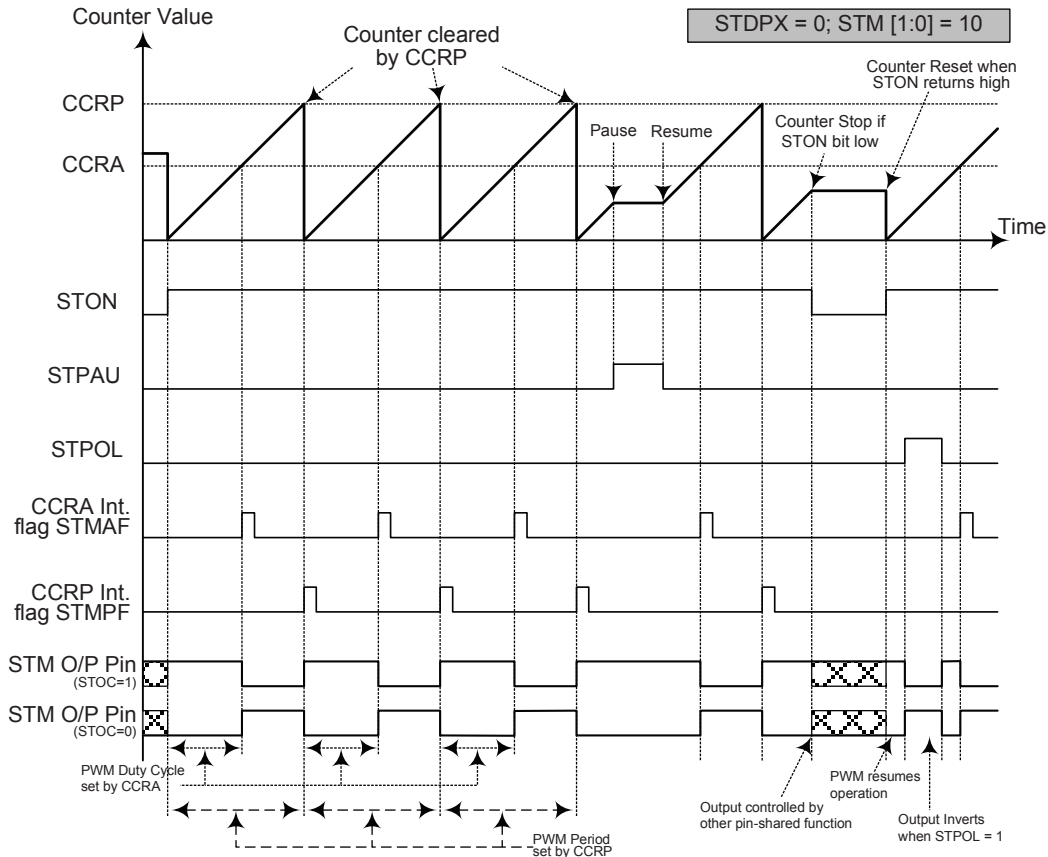
The STM PWM output frequency= $(f_{SYS}/4)/(2\times 256)=f_{SYS}/2048=7.8125\text{kHz}$ , duty= $128/(2\times 256)=25\%$ .

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%.

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

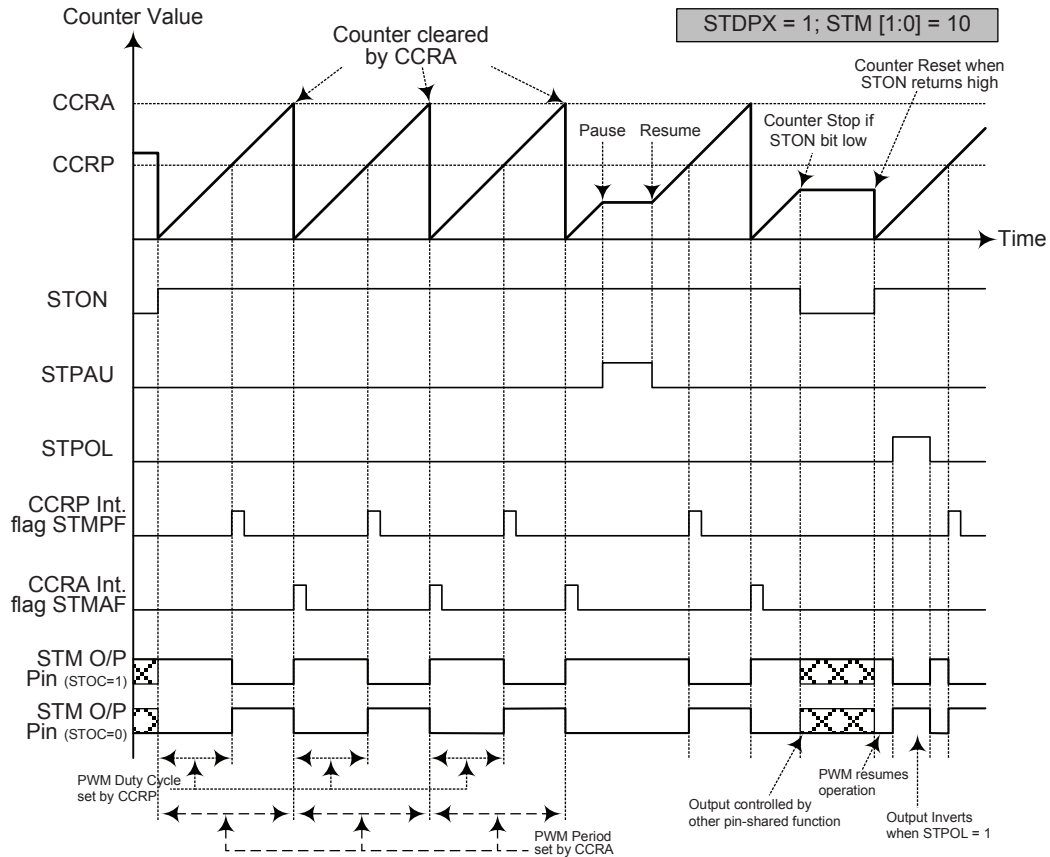
| CCRP   | 1~255    | 0     |
|--------|----------|-------|
| Period | CCRA     |       |
| Duty   | CCRP×256 | 65536 |

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



**PWM Output Mode – STDPX=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



**PWM Output Mode – STDPX=1**

- 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

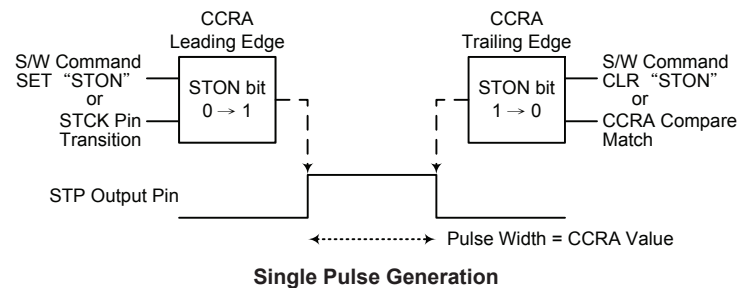


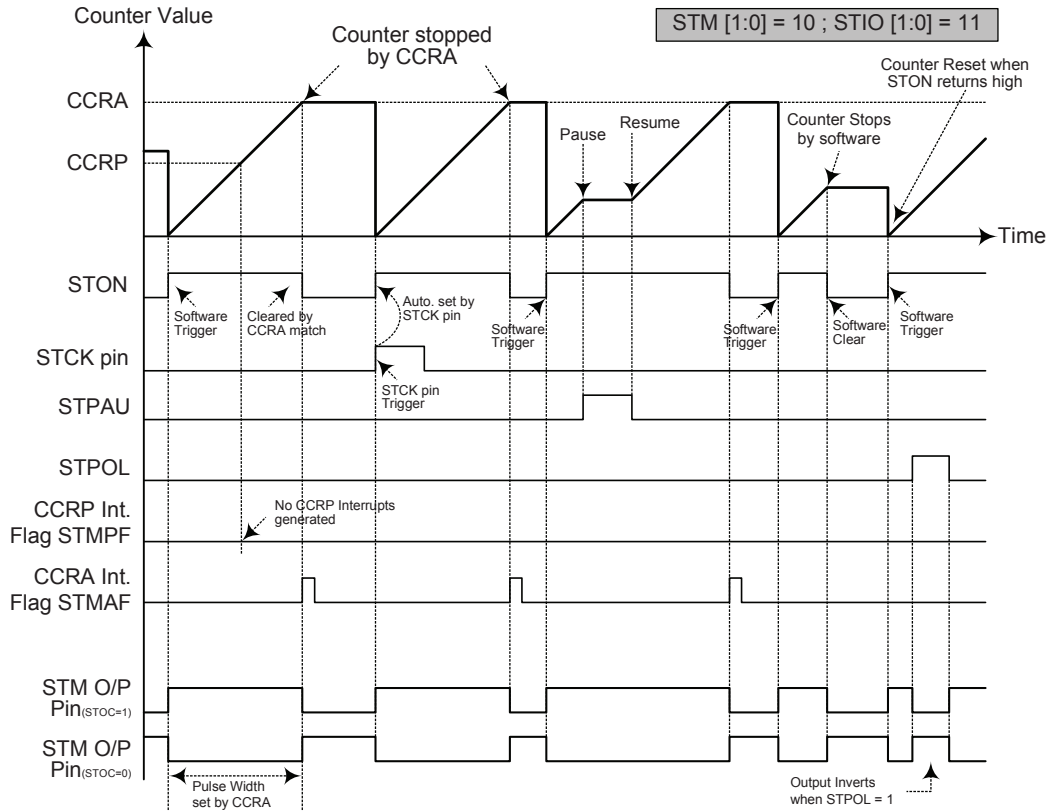
**Single Pulse Output Mode**

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 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 Mode CCRP is not used. The STCCLR and STDPX bits are not used in this Mode.





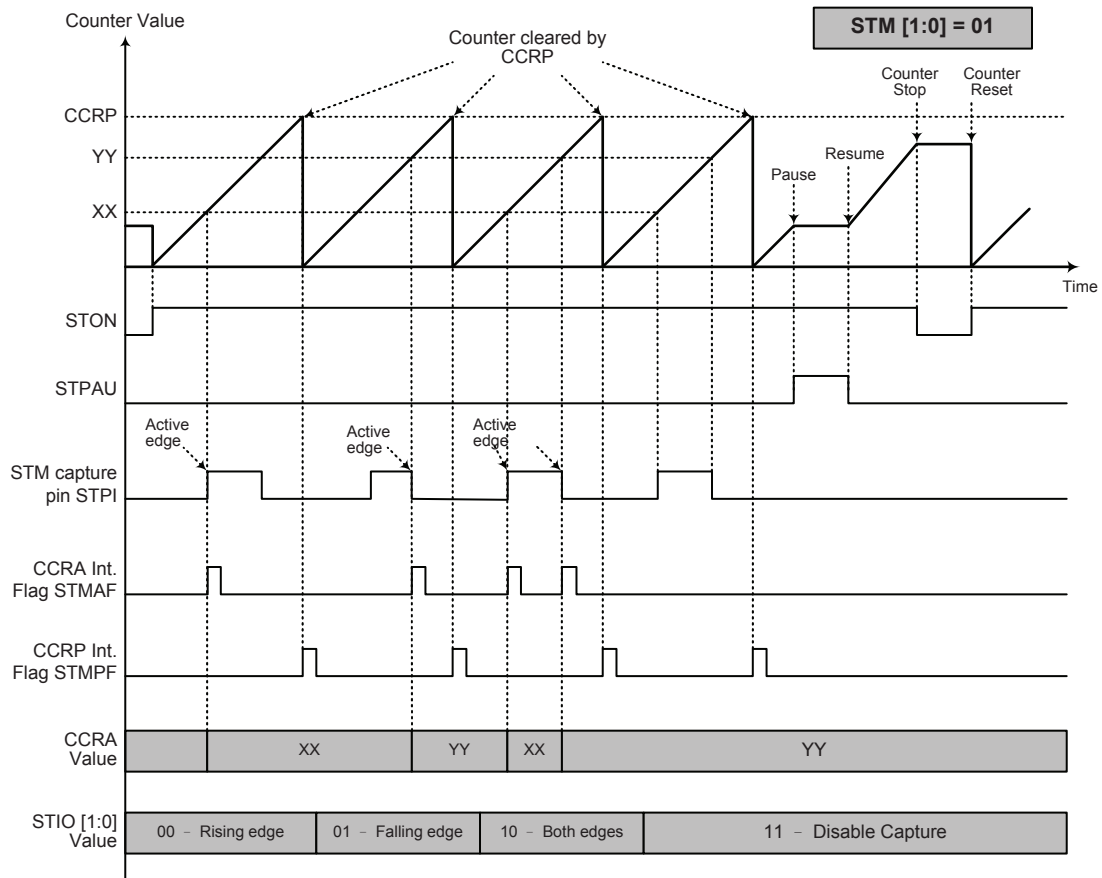
**Single Pulse Mode**

- 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 Mode, STIO [1:0] must be set to "11" and can not be changed.

### **Capture Input Mode**

To select this mode bits STM1 and STM0 in the STMC1 register should be set to 01 respectively. This mode enables external signals to capture and store the present value of the internal counter and can therefore be used for applications such as pulse width measurements. The external signal is supplied on the STPI pin, whose active edge can be a rising edge, a falling edge or both rising and falling edges; the active edge transition type is selected using the STIO1 and STIO0 bits in the STMC1 register. The counter is started when the STON bit changes from low to high which is initiated using the application program.

When the required edge transition appears on the STPI pin the present value in the counter will be latched into the CCRA registers and a STM interrupt generated. Irrespective of what events occur on the STPI pin the counter will continue to free run until the STON bit changes from high to low. When a CCRP compare match occurs the counter will reset back to zero; in this way the CCRP value can be used to control the maximum counter value. When a CCRP compare match occurs from Comparator P, a STM interrupt will also be generated. Counting the number of overflow interrupt signals from the CCRP can be a useful method in measuring long pulse widths. The STIO1 and STIO0 bits can select the active trigger edge on the STPI pin to be a rising edge, falling edge or both edge types. If the STIO1 and STIO0 bits are both set high, then no capture operation will take place irrespective of what happens on the STPI pin, however it must be noted that the counter will continue to run. The STCCLR and STDPX bits are not used in this Mode.



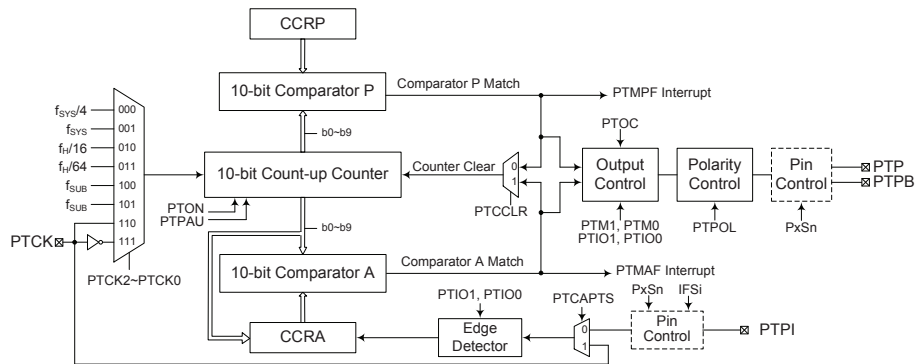
**Capture Input Mode**

- Note: 1. STM [1:0]=01 and active edge set by the STIO [1:0] bits  
 2. A STM Capture input pin active edge transfers the counter value to CCRA  
 3. STCCLR bit not used  
 4. No output function -- STOC and STPOL bits are not used  
 5. CCRP determines the counter value and the counter has a maximum count value when CCRP is equal to zero.

## Periodic Type TM – PTM

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

| PTM Core   | PTM Input Pin | PTM Output Pin |
|------------|---------------|----------------|
| 10-bit PTM | PTCK, PTPI    | PTP, PTPB      |



**Periodic Type TM Block Diagram**

### Periodic TM Operation

The size of Periodic 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 CCRP and CCRA registers. The CCRP and CCRA comparators are 10-bit wide whose value is respectively compared 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 PTON 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 PTM interrupt signal will also usually be generated. The Periodic 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.

### Periodic Type TM Register Description

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

| Register Name | Bit   |       |       |       |       |       |         |        |
|---------------|-------|-------|-------|-------|-------|-------|---------|--------|
|               | 7     | 6     | 5     | 4     | 3     | 2     | 1       | 0      |
| PTMC0         | PTPAU | PTCK2 | PTCK1 | PTCK0 | PTON  | —     | —       | —      |
| PTMC1         | PTM1  | PTM0  | PTIO1 | PTIO0 | PTOC  | PTPOL | PTCAPTS | PTCCLR |
| PTMDL         | D7    | D6    | D5    | D4    | D3    | D2    | D1      | D0     |
| PTMDH         | —     | —     | —     | —     | —     | —     | D9      | D8     |
| PTMAL         | D7    | D6    | D5    | D4    | D3    | D2    | D1      | D0     |
| PTMAH         | —     | —     | —     | —     | —     | —     | D9      | D8     |
| PTMRPL        | PTRP7 | PTRP6 | PTRP5 | PTRP4 | PTRP3 | PTRP2 | PTRP1   | PTRP0  |
| PTMRPH        | —     | —     | —     | —     | —     | —     | PTRP9   | PTRP8  |

**Periodic TM Registers List**

#### PTMC0 Register

| Bit  | 7     | 6     | 5     | 4     | 3    | 2 | 1 | 0 |
|------|-------|-------|-------|-------|------|---|---|---|
| Name | PTPAU | PTCK2 | PTCK1 | PTCK0 | PTON | — | — | — |
| R/W  | R/W   | R/W   | R/W   | R/W   | R/W  | — | — | — |
| POR  | 0     | 0     | 0     | 0     | 0    | — | — | — |

Bit 7 **PTPAU**: PTM 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 PTM 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 **PTCK2~PTCK0**: Select PTM Counter clock  
 000:  $f_{SYS}/4$   
 001:  $f_{SYS}$   
 010:  $f_H/16$   
 011:  $f_H/64$   
 100:  $f_{SUB}$   
 101:  $f_{SUB}$   
 110: PTCK rising edge clock  
 111: PTCK falling edge clock

These three bits are used to select the clock source for the PTM. 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 oscillator section.

Bit 3 **PTON**: PTM Counter On/Off control  
 0: Off  
 1: On

This bit controls the overall on/off function of the PTM. Setting the bit high enables the counter to run while clearing the bit disables the PTM. Clearing this bit to zero will stop the counter from counting and turn off the PTM 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 PTM is in the Compare Match Output Mode or the PWM output Mode or Single Pulse Output Mode, then the PTM output pin will be reset to its initial condition, as specified by the PTOC bit, when the PTON bit changes from low to high.

Bit 2~0 Unimplemented, read as "0"

**PTMC1 Register**

| Bit  | 7    | 6    | 5     | 4     | 3    | 2     | 1       | 0      |
|------|------|------|-------|-------|------|-------|---------|--------|
| Name | PTM1 | PTM0 | PTIO1 | PTIO0 | PTOC | PTPOL | PTCAPTS | PTCCLR |
| 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 **PTM1~PTM0**: Select PTM Operating Mode  
 00: Compare Match Output Mode  
 01: Capture Input Mode  
 10: PWM Mode or Single Pulse Output Mode  
 11: Timer/Counter Mode

These bits setup the required operating mode for the PTM. To ensure reliable operation the PTM should be switched off before any changes are made to the PTM1 and PTM0 bits. In the Timer/Counter Mode, the PTM output pin state is undefined.

Bit 5~4 **PTIO1~PTIO0**: Select PTM external pin PTP or PTPI function

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

Capture Input Mode

- 00: Input capture at rising edge of PTPI or PTCK
- 01: Input capture at falling edge of PTPI or PTCK
- 10: Input capture at rising/falling edge of PTPI or PTCK
- 11: Input capture disabled

Timer/Counter Mode

Unused

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

In the Compare Match Output Mode, the PTIO1 and PTIO0 bits determine how the PTM output pin changes state when a compare match occurs from the Comparator A. The PTM 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 PTM output pin should be setup using the PTOC bit in the PTMC1 register. Note that the output

level requested by the PTIO1 and PTIO0 bits must be different from the initial value setup using the PTOC bit otherwise no change will occur on the PTM output pin when a compare match occurs. After the PTM output pin changes state, it can be reset to its initial level by changing the level of the PTON bit from low to high.

In the PWM Mode, the PTIO1 and PTIO0 bits determine how the TM output pin changes state when a certain compare match condition occurs. The PTM output function is modified by changing these two bits. It is necessary to only change the values of the PTIO1 and PTIO0 bits only after the PTM has been switched off. Unpredictable PWM outputs will occur if the PTIO1 and PTIO0 bits are changed when the PTM is running.

Bit 3 **PTOC**: PTM PTP 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 PTM output pin. Its operation depends upon whether PTM is being used in the Compare Match Output Mode or in the PWM Mode/Single Pulse Output Mode. It has no effect if the PTM is in the Timer/Counter Mode. In the Compare Match Output Mode it determines the logic level of the PTM output pin before a compare match occurs. In the PWM Mode/Single Pulse Output Mode it determines if the PWM signal is active high or active low.

Bit 2 **PTPOL**: PTM PTP Output polarity control

- 0: Non-inverted
- 1: Inverted

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

Bit 1 **PTCAPTS**: PTM Capture Trigger Source selection

- 0: From PTPI pin
- 1: From PTCK pin

Bit 0 **PTCCLR**: PTM 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 Periodic TM contains two comparators, Comparator A and Comparator P, either of which can be selected to clear the internal counter. With the PTCCLR 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 PTCCLR bit is not used in the PWM Output, Single Pulse Output or Capture Input Mode.

**PTMDL 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 PTM Counter Low Byte Register bit 7 ~ bit 0  
PTM 10-bit Counter bit 7 ~ bit 0



**PTMDH 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 PTM Counter High Byte Register bit 1 ~ bit 0  
 PTM 10-bit Counter bit 9 ~ bit 8

**PTMAL 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 PTM CCRA Low Byte Register bit 7 ~ bit 0  
 PTM 10-bit CCRA bit 7 ~ bit 0

**PTMAH 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 PTM CCRA High Byte Register bit 1 ~ bit 0  
 PTM 10-bit CCRA bit 9 ~ bit 8

**PTMRPL Register**

| Bit  | 7     | 6     | 5     | 4     | 3     | 2     | 1     | 0     |
|------|-------|-------|-------|-------|-------|-------|-------|-------|
| Name | PTRP7 | PTRP6 | PTRP5 | PTRP4 | PTRP3 | PTRP2 | PTRP1 | PTRP0 |
| 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 PTM CCRP Low Byte Register bit 7 ~ bit 0  
 PTM 10-bit CCRP bit 7 ~ bit 0

**PTMRPH Register**

| Bit  | 7 | 6 | 5 | 4 | 3 | 2 | 1     | 0     |
|------|---|---|---|---|---|---|-------|-------|
| Name | — | — | — | — | — | — | PTRP9 | PTRP8 |
| R/W  | — | — | — | — | — | — | R/W   | R/W   |
| POR  | — | — | — | — | — | — | 0     | 0     |

Bit 7~2 Unimplemented, read as "0"

Bit 1~0 PTM CCRP High Byte Register bit 1 ~ bit 0  
 PTM 10-bit CCRP bit 9 ~ bit 8

## Periodic Type TM Operation Modes

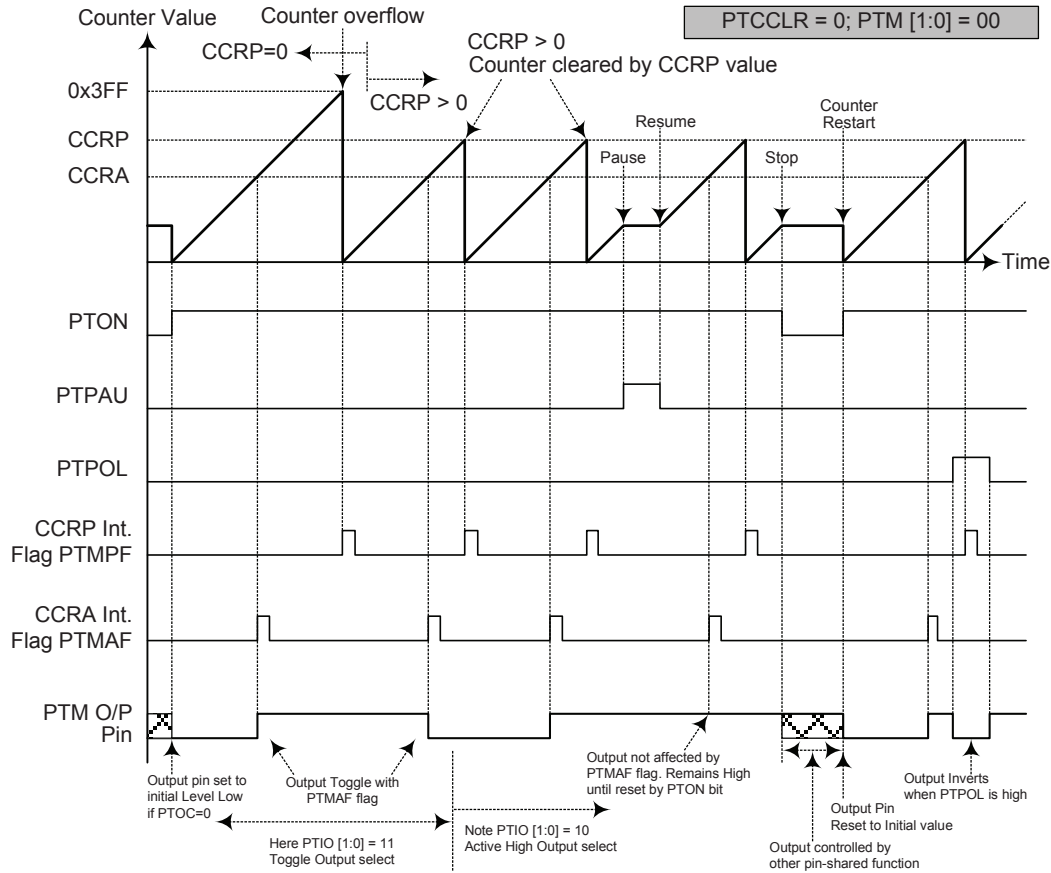
The Periodic Type TM can operate in one of five operating modes, Compare Match Output Mode, PWM Output Mode, Single Pulse Output Mode, Capture Input Mode or Timer/Counter Mode. The operating mode is selected using the PTM1 and PTM0 bits in the PTMC1 register.

### Compare Match Output Mode

To select this mode, bits PTM1 and PTM0 in the PTMC1 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 PTCCLR 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 PTMAF and PTMPF interrupt request flags for Comparator A and Comparator P respectively, will both be generated.

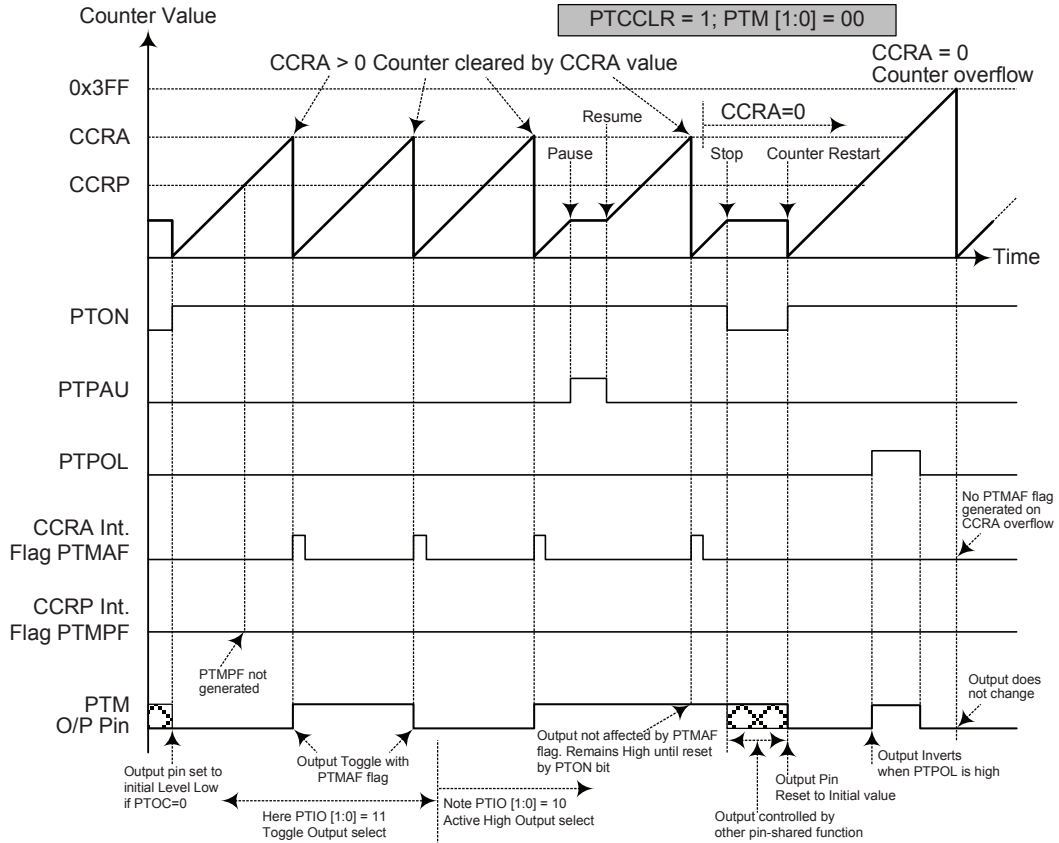
If the PTCCLR bit in the PTMC1 register is high then the counter will be cleared when a compare match occurs from Comparator A. However, here only the PTMAF interrupt request flag will be generated even if the value of the CCRP bits is less than that of the CCRA registers. Therefore when PTCCLR is high no PTMPF interrupt request flag will be generated. In the Compare Match Output Mode, the CCRA can not be set to "0".

As the name of the mode suggests, after a comparison is made, the PTM output pin will change state. The PTM output pin condition however only changes state when a PTMAF interrupt request flag is generated after a compare match occurs from Comparator A. The PTMPF interrupt request flag, generated from a compare match occurs from Comparator P, will have no effect on the PTM output pin. The way in which the PTM output pin changes state are determined by the condition of the PTIO1 and PTIO0 bits in the PTMC1 register. The PTM output pin can be selected using the PTIO1 and PTIO0 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 PTM output pin, which is setup after the PTON bit changes from low to high, is setup using the PTOC bit. Note that if the PTIO1 and PTIO0 bits are zero then no pin change will take place.



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

- Note: 1. With PTCCCLR=0, a Comparator P match will clear the counter  
 2. The PTM output pin is controlled only by the PTMAF flag  
 3. The output pin is reset to its initial state by a PTON bit rising edge



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

- Note: 1. With PTCCLR=1, a Comparator A match will clear the counter  
 2. The PTM output pin is controlled only by the PTMAF flag  
 3. The output pin is reset to its initial state by a PTON bit rising edge  
 4. A PTMPF flag is not generated when PTCCLR =1

**Timer/Counter Mode**

To select this mode, bits PTM1 and PTM0 in the PTMC1 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 PTM 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 PTM output pin is not used in this mode, the pin can be used as a normal I/O pin or other pin-shared function.

**PWM Output Mode**

To select this mode, bits PTM1 and PTM0 in the PTMC1 register should be set to 10 respectively and also the PTIO1 and PTIO0 bits should be set to 10 respectively. The PWM function within the PTM 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 PTM 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 mode, the PTCCLR bit has no effect as the PWM period. Both of the CCRP and CCRA 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. 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 PTOC bit in the PTMC1 register is used to select the required polarity of the PWM waveform while the two PTIO1 and PTIO0 bits are used to enable the PWM output or to force the PTM output pin to a fixed high or low level. The PTPOL bit is used to reverse the polarity of the PWM output waveform.

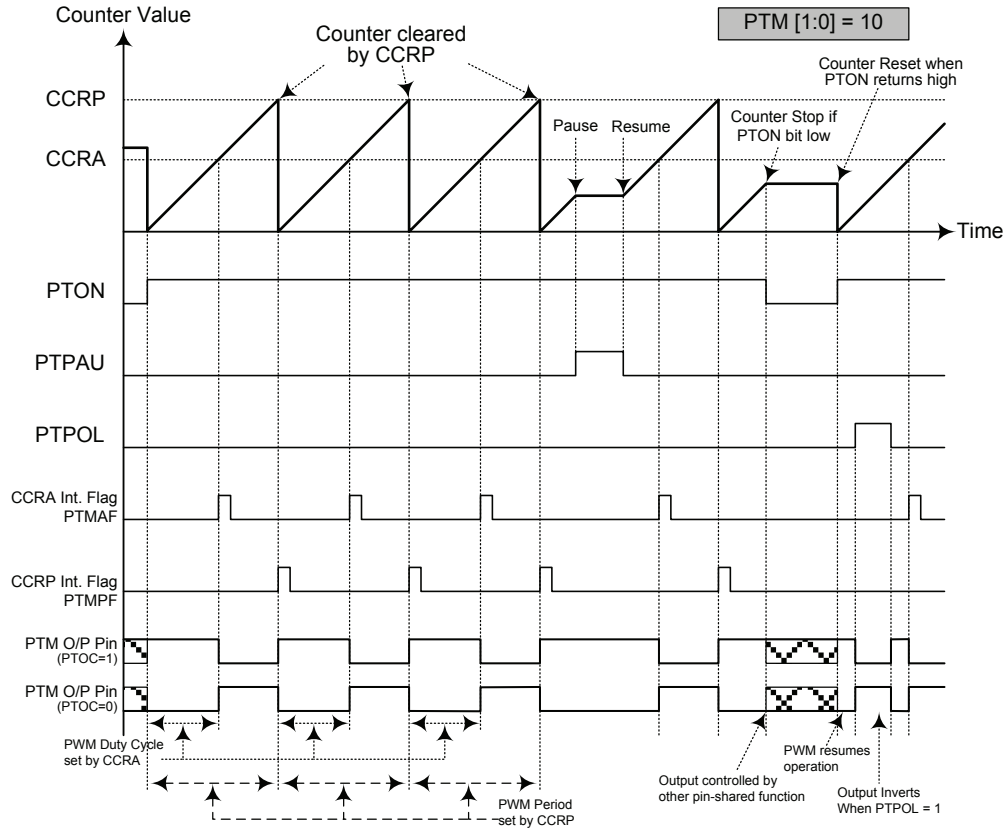
• **10-bit PWM Output Mode, Edge-aligned Mode**

| CCRP   | 1~1023 | 0    |
|--------|--------|------|
| Period | 1~1023 | 1024 |
| Duty   | CCRA   |      |

If  $f_{SYS}=16\text{MHz}$ , TM clock source select  $f_{SYS}/4$ , CCRP=512 and CCRA=128,

The PTM PWM output frequency= $(f_{SYS}/4)/512=f_{SYS}/2048=7.8125\text{kHz}$ , 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%.



**PWM Mode**

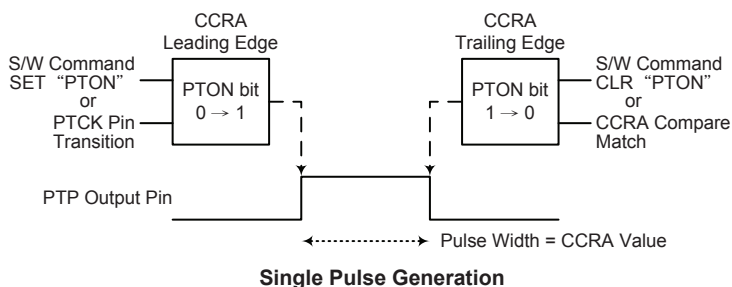
- Note: 1. The counter is cleared by CCRP.  
 2. A counter clear sets the PWM Period  
 3. The internal PWM function continues running even when PTIO [1:0]=00 or 01  
 4. The PTCCCLR bit has no influence on PWM operation

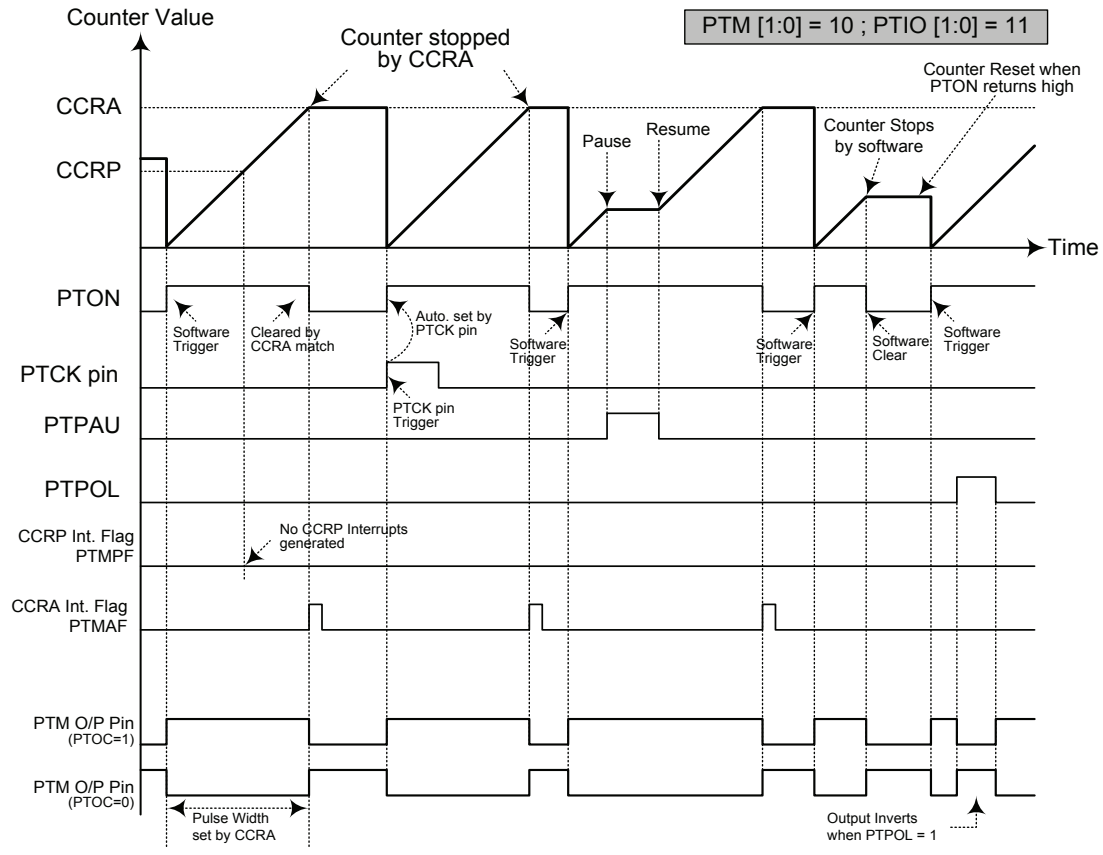
### Single Pulse Output Mode

To select this mode, bits PTM1 and PTM0 in the PTMC1 register should be set to 10 respectively and also the PTIO1 and PTIO0 bits should be set to 11 respectively. The Single Pulse Output Mode, as the name suggests, will generate a single shot pulse on the PTM output pin.

The trigger for the pulse output leading edge is a low to high transition of the PTON bit, which can be implemented using the application program. However in the Single Pulse Mode, the PTON bit can also be made to automatically change from low to high using the external PTCK pin, which will in turn initiate the Single Pulse output. When the PTON bit transitions to a high level, the counter will start running and the pulse leading edge will be generated. The PTON bit should remain high when the pulse is in its active state. The generated pulse trailing edge will be generated when the PTON 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 PTON 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 PTM interrupt. The counter can only be reset back to zero when the PTON bit changes from low to high when the counter restarts. In the Single Pulse Mode CCRP is not used. The PTCLLR is not used in this Mode.





**Single Pulse Mode**

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

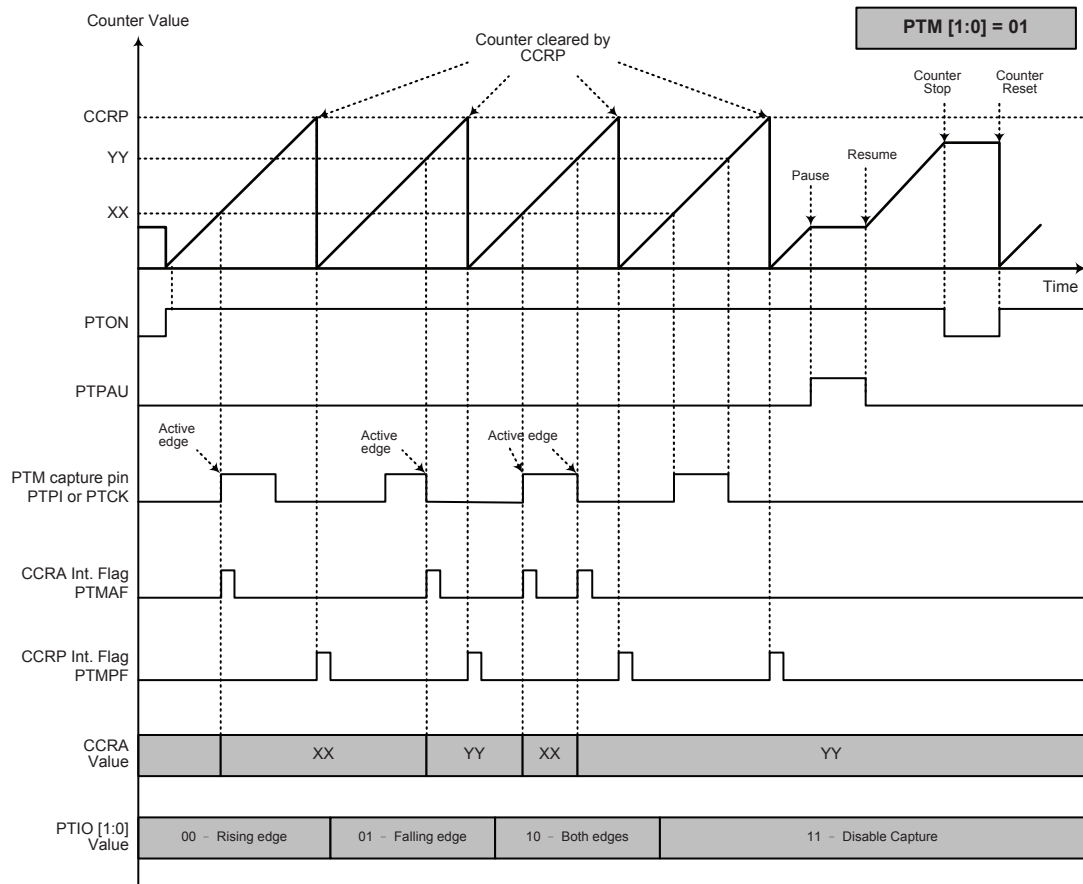


### **Capture Input Mode**

To select this mode bits PTM1 and PTM0 in the PTMC1 register should be set to 01 respectively. This mode enables external signals to capture and store the present value of the internal counter and can therefore be used for applications such as pulse width measurements. The external signal is supplied on the PTPI or PTCK pin, selected by the PTCAPTS bit in the PTMC1 register. The input pin active edge can be either a rising edge, a falling edge or both rising and falling edges; the active edge transition type is selected using the PTIO1 and PTIO0 bits in the PTMC1 register. The counter is started when the PTON bit changes from low to high which is initiated using the application program.

When the required edge transition appears on the PTPI or PTCK pin the present value in the counter will be latched into the CCRA registers and a PTM interrupt generated. Irrespective of what events occur on the PTPI or PTCK pin the counter will continue to free run until the PTON bit changes from high to low. When a CCRP compare match occurs the counter will reset back to zero; in this way the CCRP value can be used to control the maximum counter value. When a CCRP compare match occurs from Comparator P, a PTM interrupt will also be generated. Counting the number of overflow interrupt signals from the CCRP can be a useful method in measuring long pulse widths. The PTIO1 and PTIO0 bits can select the active trigger edge on the PTPI or PTCK pin to be a rising edge, falling edge or both edge types. If the PTIO1 and PTIO0 bits are both set high, then no capture operation will take place irrespective of what happens on the PTPI or PTCK pin, however it must be noted that the counter will continue to run.

As the PTPI or PTCK pin is pin shared with other functions, care must be taken if the PTM is in the Input Capture Mode. This is because if the pin is setup as an output, then any transitions on this pin may cause an input capture operation to be executed. The PTCCLR, PTOC and PTPOL bits are not used in this Mode.



**Capture Input Mode**

- Note: 1. PTM [1:0]=01 and active edge set by the PTIO [1:0] bits  
 2. A PTM Capture input pin active edge transfers the counter value to CCRA  
 3. PTCLR bit not used  
 4. No output function – PTOC and PTPOL bits are not used  
 5. CCRP determines the counter value and the counter has a maximum count value when CCRP is equal to zero.

## Analog to Digital Converter – ADC

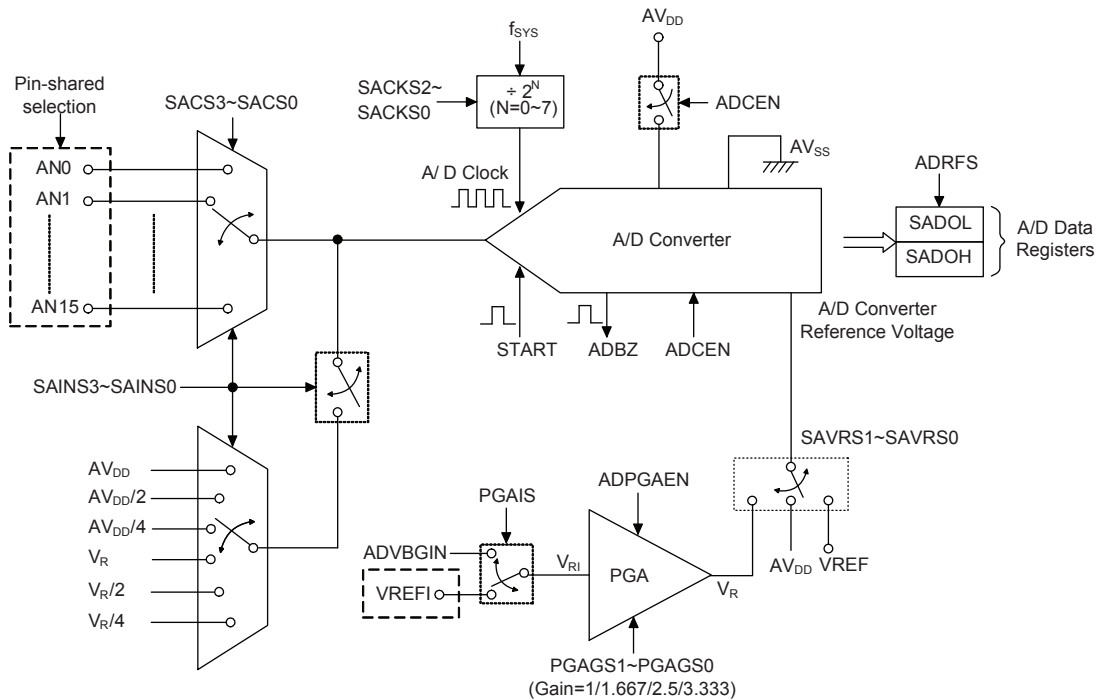
The need to interface to real world analog signals is a common requirement for many electronic systems. However, to properly process these signals by a microcontroller, they must first be converted into digital signals by A/D converters. By integrating the A/D conversion electronic circuitry into the 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. The external or internal analog signal to be converted is determined by the SAINS and SACS bit fields. Note that when the internal analog signal is to be converted, the selected external input channel will automatically be disconnected to avoid malfunction. More detailed information about the A/D converter input signal is described in the "A/D Converter Control Registers" and "A/D Converter Input Signal" sections respectively.

| External Input Channels | Internal Analog Signals   | A/D Converter Signal Select Bits |
|-------------------------|---|----------------------------------|
| 16: AN0~AN15            | 6: AV <sub>DD</sub> , AV <sub>DD</sub> /2, AV <sub>DD</sub> /4, V <sub>R</sub> , V <sub>R</sub> /2, V <sub>R</sub> /4 | SAINS3~SAINS0, 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 six registers. A read only register pair exists to store the ADC data 12-bit value. The remaining four 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     |
| SADOH(ADRF5=0) | D11     | D10    | D9     | D8     | D7     | D6     | D5     | D4     |
| SADOH(ADRF5=1) | —       | —      | —      | —      | D11    | D10    | D9     | D8     |
| SADC0          | START   | ADBZ   | ADCEN  | ADRF5  | SACS3  | SACS2  | SACS1  | SACS0  |
| SADC1          | SAINS3  | SAINS2 | SAINS1 | SAINS0 | —      | SACKS2 | SACKS1 | SACKS0 |
| SADC2          | ADPGAEN | —      | —      | PGAIS  | SAVRS1 | SAVRS0 | PGAGS1 | PGAGS0 |
| VBGRC          | —       | —      | —      | —      | —      | —      | —      | VBGREN |

**A/D Converter Register List**

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

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 SADOH, 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 the A/D converter data register contents will keep unchanged if the A/D converter is disabled.

| ADRF5 | SADOH |     |    |    |     |     |    |    | 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, several control registers known as SADC0, SADC1, 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 converter clock source as well as controlling the start function and monitoring the A/D converter busy status. As the device contains only one actual analog to digital converter hardware circuit, each of the external and internal analog signals 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 SAINS3~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 converter 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 | ADRF5 | 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 A/D conversion  
0→1: Reset the A/D converter and set ADBZ to 0  
1→0: Start A/D conversion and set ADBZ to 1  
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 high to indicate that the A/D conversion is initiated. The ADBZ flag will be cleared to zero 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 converter internal function. This bit should be set high 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 converter data register pair, SADOH and SADOL, will keep unchanged.
- Bit 4**     **ADRF5:** A/D Converter data format control  
0: ADC output data format → SADOH=D[11:4]; SADOL=D[3:0]  
1: ADC output data format → SADOH=D[11:8]; SADOL=D[7:0]  
This bit controls the format of the 12-bit converted A/D converter value in the two A/D converter data registers. Details are provided in the A/D converter data register section.
- Bit 3~0**   **SACS3~SACS0:** A/D converter external analog input channel selection  
0000: AN0  
0001: AN1  
0010: AN2  
0011: AN3  
0100: AN4  
0101: AN5  
0110: AN6  
0111: AN7  
1000: AN8  
1001: AN9  
1010: AN10  
1011: AN11  
1100: AN12  
1101: AN13  
1110: AN14  
1111: AN15

### SADC1 Register

| Bit  | 7      | 6      | 5      | 4      | 3 | 2      | 1      | 0      |
|------|--------|--------|--------|--------|---|--------|--------|--------|
| Name | SAINS3 | SAINS2 | SAINS1 | SAINS0 | — | SACKS2 | SACKS1 | SACKS0 |
| 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 7~4 **SAINS3~SAINS0**: A/D converter input signal selection  
 0000, 0100, 1100~1111: External signal – External analog channel input  
 0001: Internal signal – Internal A/D converter power supply voltage  $AV_{DD}$   
 0010: Internal signal – Internal A/D converter power supply voltage  $AV_{DD}/2$   
 0011: Internal signal – Internal A/D converter power supply voltage  $AV_{DD}/4$   
 0101: Internal signal – Internal A/D converter power supply voltage  $V_R$   
 0110: Internal signal – Internal A/D converter power supply voltage  $V_R/2$   
 0111: Internal signal – Internal A/D converter power supply voltage  $V_R/4$   
 When the internal analog signal is selected to be converted, the external channel input signal will automatically be switched off regardless of the SACKS3~SACKS0 bit field value. The internal reference voltage can be derived from various sources selected using the SAVRS1~SAVRS0 bits in the SADC2 register.

Bit 3 Unimplemented, read as "0"

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:  $f_{SYS}/128$

### SADC2 Register

| Bit  | 7       | 6 | 5 | 4     | 3      | 2      | 1      | 0      |
|------|---------|---|---|-------|--------|--------|--------|--------|
| Name | ADPGAEN | — | — | PGAIS | SAVRS1 | SAVRS0 | PGAGS1 | PGAGS0 |
| R/W  | R/W     | — | — | R/W   | R/W    | R/W    | R/W    | R/W    |
| POR  | 0       | — | — | 0     | 0      | 0      | 0      | 0      |

Bit 7 **ADPGAEN**: A/D converter PGA enable/disable control  
 0: Disable  
 1: Enable  
 This bit controls the internal PGA function to provide various reference voltage for the A/D converter. When the bit is set high, the internal reference voltage,  $V_R$ , can be used as the internal converter signal or reference voltage by the A/D converter. If the internal reference voltage is not used by the A/D converter, then the PGA function should be properly configured to conserve power.

Bit 6~5 Unimplemented, read as "0"

Bit 4 **PGAIS**: PGA input ( $V_{RI}$ ) selection  
 0: VREF1  
 1: ADVBGIN(ADVBGIN comes from the external source, and ADVBGIN voltage is equal to 1.2V)

Bit 3~2 **SAVRS1~SAVRS0**: ADC reference voltage selection  
 00:  $AV_{DD}$   
 01: VREF  
 1x:  $V_R$  (PGA output)  
 When the A/D converter reference voltage source is selected to derive from the internal  $V_{BG}$  voltage, the reference voltage which comes from the AVDD or VREF pin will be automatically switched off.

Bit 1~0     **PGAGS1~ PGAGS0**: PGA gain selection  
 00: Gain=1  
 01: Gain=1.667 ( $V_R=2V$  if  $V_{RI}=1.2V$ )  
 10: Gain=2.5 ( $V_R=3V$  if  $V_{RI}=1.2V$ )  
 11: Gain=3.333 ( $V_R=4V$  if  $V_{RI}=1.2V$ )

### A/D Converter Operation

The START bit is used to start and reset the A/D converter. When the microcontroller sets this bit from low to high and then low again, an analog to digital conversion cycle will be initiated. When the START bit is brought from low to high but not low again, the ADBZ bit in the SADC0 register will be cleared to zero and the analog to digital converter will be reset. It is the START bit that is used to control the overall start operation of the internal analog to digital converter.

The ADBZ bit in the SADC0 register is used to indicate whether the analog to digital conversion process is in process or not. When the A/D converter is reset by setting the START bit from low to high, the ADBZ flag will be cleared to "0". 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 converter 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 converter internal interrupt signal will direct the program flow to the associated A/D converter internal interrupt address for processing. If the A/D converter internal interrupt is disabled, the microcontroller can be used to 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 conversion clock source is determined by the system clock  $f_{SYS}$ , and by bits SACKS2~SACKS0, there are some limitations on the maximum A/D conversion clock source speed that can be selected. As the recommended value of permissible A/D conversion clock period,  $t_{ADCK}$ , is from 0.5 $\mu$ s to 10 $\mu$ s, care must be taken for system clock frequencies. For example, if the system clock operates at a frequency of 4MHz, the SACKS2~SACKS0 bits should not be set to "000", "110" or "111". Doing so will give A/D conversion clock periods that are less than the minimum A/D conversion clock period or greater than the maximum A/D conversion 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, depending upon the device, special care must be taken, as the values may be less than the specified minimum A/D conversion Clock Period.

| $f_{SYS}$ | A/D Conversion Clock Period ( $t_{ADCK}$ ) |                                 |                                 |                                 |                                  |                                  |                                  |                                   |
|-----------|--|---------------------------------|---------------------------------|---------------------------------|----------------------------------|----------------------------------|----------------------------------|-----------------------------------|
|           | SACKS [2:0]=000 ( $f_{SYS}$ )              | SACKS [2:0]=001 ( $f_{SYS}/2$ ) | SACKS [2:0]=010 ( $f_{SYS}/4$ ) | SACKS [2:0]=011 ( $f_{SYS}/8$ ) | SACKS [2:0]=100 ( $f_{SYS}/16$ ) | SACKS [2:0]=101 ( $f_{SYS}/32$ ) | SACKS [2:0]=110 ( $f_{SYS}/64$ ) | SACKS [2:0]=111 ( $f_{SYS}/128$ ) |
| 1MHz      | 1 $\mu$ s                                  | 2 $\mu$ s                       | 4 $\mu$ s                       | 8 $\mu$ s                       | 16 $\mu$ s*                      | 32 $\mu$ s*                      | 64 $\mu$ s*                      | 128 $\mu$ s*                      |
| 2MHz      | 500ns                                      | 1 $\mu$ s                       | 2 $\mu$ s                       | 4 $\mu$ s                       | 8 $\mu$ s                        | 16 $\mu$ s*                      | 32 $\mu$ s*                      | 64 $\mu$ s*                       |
| 4MHz      | 250ns*                                     | 500ns                           | 1 $\mu$ s                       | 2 $\mu$ s                       | 4 $\mu$ s                        | 8 $\mu$ s                        | 16 $\mu$ s*                      | 32 $\mu$ s*                       |
| 8MHz      | 125ns*                                     | 250ns*                          | 500ns                           | 1 $\mu$ s                       | 2 $\mu$ s                        | 4 $\mu$ s                        | 8 $\mu$ s                        | 16 $\mu$ s*                       |
| 12MHz     | 83ns*                                      | 167ns*                          | 333ns*                          | 667ns                           | 1.33 $\mu$ s                     | 2.67 $\mu$ s                     | 5.33 $\mu$ s                     | 10.67 $\mu$ s*                    |
| 16MHz     | 62.5ns*                                    | 125ns*                          | 250ns*                          | 500ns                           | 1 $\mu$ s                        | 2 $\mu$ s                        | 4 $\mu$ s                        | 8 $\mu$ s                         |

**A/D Conversion Clock Period Examples**

Controlling the power on/off function of the A/D conversion 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 conversion 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 converter inputs by configuring the corresponding pin control bits, 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 positive power supply pin, AVDD, an external reference source supplied on pin VREF or an internal reference source derived from the PGA output  $V_R$ . The desired selection is made using the SAVRS1~SAVRS0 bits and the corresponding pin-shared function selection bits. The PGA gain can be equal to 1, 1.667, 2.5 or 3.333 selected by the PGAGS1~PGAGS0 bits in the SADC2 register. 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 bits should first be properly configured to disable other pin-shared functions. When  $V_R$  (PGA output) is selected by ADC input or ADC reference voltage, the PGA should be enabled by setting ADPGAEN=1. Otherwise the PGA should be disabled by setting ADPGAEN=0 to conserve the power. The PGA input comes from VREFI or ADVBGIN selected by PGAIS bit in the SADC2 register. Note that the analog input signal values must not be allowed to exceed the value of the selected A/D Converter reference voltage.

| SAVRS[1:0] | Reference        | Description  |
|------------|------------------|--|
| 00         | AV <sub>DD</sub> | ADC Reference Voltage comes from AV <sub>DD</sub>  |
| 01         | VREF             | ADC Reference Voltage comes from external VREF pin |
| 1x         | $V_R$            | ADC Reference Voltage comes from $V_R$             |

**A/D Converter Reference Voltage Selection**

### VBGRC Register

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

Bit 7~1 Unimplemented, read as "0"

Bit 0 **VBGREN**: Bandgap enable control

0: Disable

1: Enable

When VBGREN bit is set to zero, VBGREF output is with high impedance.



### A/D Converter Input Signal

All of the external A/D Converter analog input pins are pin-shared with the I/O pins as well as other functions. The corresponding pin-shared function selection bits in the PxS0 and PxS1 registers, determine whether the external input pins are setup as A/D converter analog channel inputs or whether they have other functions. If the corresponding pin is setup to be an A/D converter 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 Converter 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 Converter inputs. Note that it is not necessary to first setup the A/D converter pin as an input in the port control register to enable the A/D Converter input as when the relevant A/D converter input function selection bits enable an A/D converter input, the status of the port control register will be overridden.

If the SAINS3~SAINS0 bits are set to "0000", "0100", "1100", "1101", "1110" and "1111", the external analog channel input is selected to be converted and the SACS3~SACS0 bits can determine which external channel is selected to be converted. If the SAINS3~SAINS0 bits are set to any other values except "0000", "0100", "1100", "1101", "1110" and "1111", one of the internal analog signals is selected to be converted. The internal analog signals can be derived from the A/D converter supply power,  $AV_{DD}$ , or internal reference voltage,  $V_R$ , with a specific ratio of 1, 1/2 or 1/4. If the internal analog signal is selected to be converted, the external channel signal input will automatically be switched off to avoid the signal contention.

| SAINS [3:0]                        | SACS [3:0] | Input Signals | Description                          |
|------------------------------------|------------|---------------|--------------------------------------|
| 0000, 0100, 1100, 1101, 1110, 1111 | 0000~1111  | AN0~AN15      | External channel analog input        |
| 0001                               | xxxx       | $AV_{DD}$     | A/D converter power supply voltage   |
| 0010                               | xxxx       | $AV_{DD}/2$   | A/D converter power supply voltage/2 |
| 0011                               | xxxx       | $AV_{DD}/4$   | A/D converter power supply voltage/4 |
| 0101                               | xxxx       | $V_R$         | Internal reference voltage           |
| 0110                               | xxxx       | $V_R/2$       | Internal reference voltage/2         |
| 0111                               | xxxx       | $V_R/4$       | Internal reference voltage/4         |

"x": Don't care

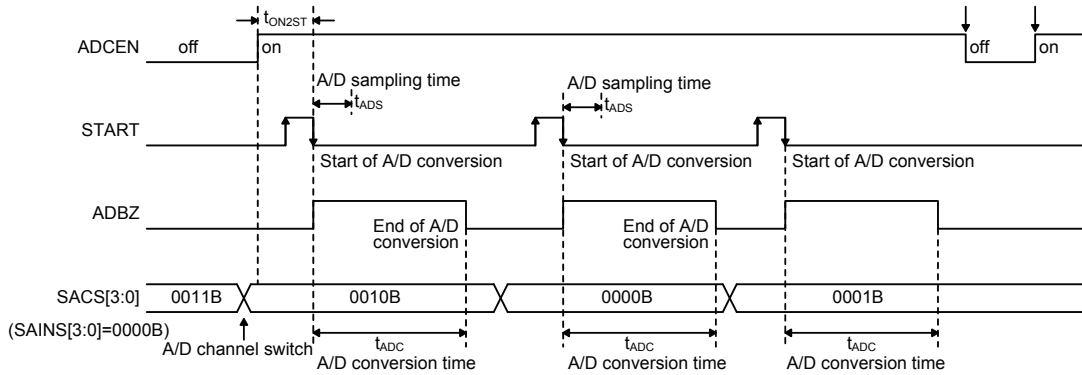
#### A/D Converter Input Signal Selection

**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 conversion clock cycles and the data conversion takes 12 A/D converter clock cycles. Therefore a total of 16 A/D conversion clock cycles for an A/D conversion which is defined as  $t_{ADC}$  are necessary.

$$\text{Maximum single A/D conversion rate} = \text{A/D conversion clock period} / 16$$

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  $16t_{ADCK}$  clock cycles where  $t_{ADCK}$  is equal to the A/D conversion clock period.



**A/D Conversion Timing**

## 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 properly programming the SACKS2~SACKS0 bits 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 SAINS3~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 converter input signal comes from the external channel input selecting by configuring the SAINS bit field, the corresponding pin should first be configured as A/D converter input function by configuring the corresponding pin-shared function selection bits in the PxS0 and PxS1 registers. The desired analog channel then should be selected by configuring the SACS bit field. After this step, go to Step 6.
- Step 5  
Before the A/D converter input signal is selected to come from the internal analog signal by configuring the SAINS bit field, the relevant control bit must be first cleared to zero to disable the external channel input. The desired internal analog signal then can be selected by configuring the SAINS bit field. After this step, go to Step 6.
- Step 6  
Select the reference voltage source by configuring the SAVRS1~SAVRS0 bits. If the reference voltage comes from  $V_R$ , the PGA output can be selected by configuring the PGAIS and PGAGS[1:0] bits.
- Step 7  
Select A/D converter output data format by configuring the ADRFS bit.
- Step 8  
If A/D converter interrupt is used, the interrupt control registers must be correctly configured to ensure the A/D converter interrupt function is active. The master interrupt control bit, EMI, and the A/D converter interrupt bits, ADE, must both set high in advance.
- Step 9  
The A/D conversion procedure can now be initialised 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 completed, the ADBZ flag will go low and then output data can be read from the SADOH and SADOL registers. If the ADC interrupt is enabled and the stack is not full, data can be acquired by interrupt service program.

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 conversion internal circuitry can be switched off to reduce power consumption, by clearing the ADCEN bit in the SADC0 register. When this happens, the internal A/D conversion 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/Os, 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 Converter Transfer Function**

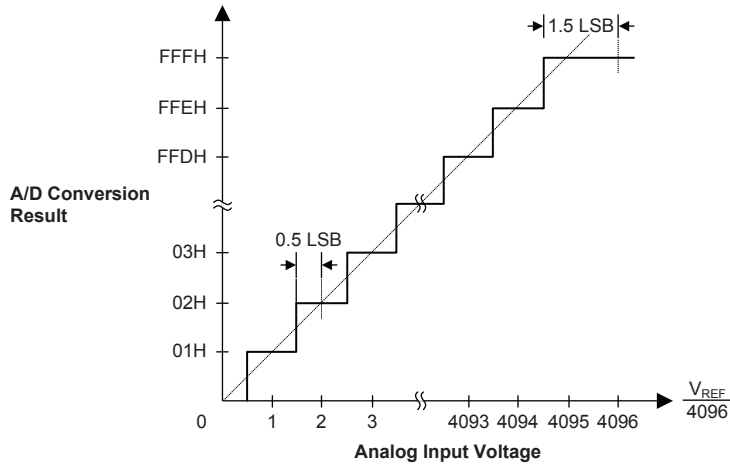
As the device contains a 12-bit A/D converter, its full-scale converted digitised value is equal to FFFH. Since the full-scale analog input value is equal to the actual A/D converter reference voltage,  $V_{REF}$ , this gives a single bit analog input value of  $V_{REF}$  divided by 4096.

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

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

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

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.



**Ideal A/D converter Transfer Function**

## A/D Converter 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 converter interrupt is used to determine when the conversion is complete.

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

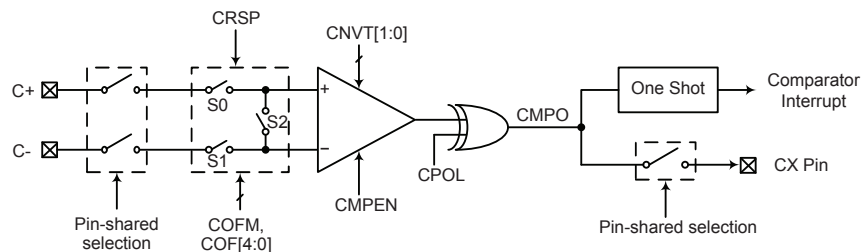
```
clr ADE          ; disable ADC interrupt
mov a,03H
mov SADC1,a      ; select fsys/8 as ADC clock
set ADCEN
mov a,03h        ; setup PCS0 to configure pin AN0
mov PCS0,a
mov a,20h
mov SADC0,a      ; enable and connect AN0 channel to A/D converter
:
start_conversion:
clr START        ; high pulse on start bit to initiate conversion
set START        ; reset A/D converter
clr START        ; start A/D conversion
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,03H
mov SADC1,a      ; select fsys/8 as ADC clock
set ADCEN
mov a,03h        ; setup PSC0 to configure pin AN0
mov PCS0,a
mov a,20h
mov SADC0,a      ; enable 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 converter
clr START        ; start A/D conversion
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
```

## Comparator

An analog comparator is contained within the device. The comparator function offers flexibility via their register controlled features such as power-down, polarity select, hysteresis etc. In sharing their pins with normal I/O pins the comparators do not waste precious I/O pins if these functions are otherwise unused.



**Comparator**

### Comparator Operation

The device contains a comparator function which is used to compare two analog voltages and provide an output based on their input difference.

Any pull-high resistors connected to the shared comparator input pins will be automatically disconnected when the corresponding comparator functional pins are selected. As the comparator inputs approach their switching level, some spurious output signals may be generated on the comparator output due to the slow rising or falling nature of the input signals. This can be minimised by the hysteresis function. Ideally the comparator should switch at the point where the positive and negative inputs signals are at the same voltage level. However, unavoidable input offsets introduce some uncertainties here. The hysteresis function will also increase the switching offset value. The hysteresis window will be changed for different comparator response time selections. The comparator response time is shown in the comparator electrical characteristics.

### Comparator Registers

Full control over each internal comparator is provided via two control registers, CMPC and CMPVOS. The comparator output is recorded via a bit in the respective control register, but can also be transferred out onto a shared I/O pin. Additional comparator functions include output polarity, response time and power down control.

| Register Name | Bit |       |      |      |       |       |      |      |
|---------------|-----|-------|------|------|-------|-------|------|------|
|               | 7   | 6     | 5    | 4    | 3     | 2     | 1    | 0    |
| CMPC          | —   | CMPEN | CPOL | CMPO | CNVT1 | CNVT0 | —    | —    |
| CMPVOS        | —   | COFM  | CRSP | COF4 | COF3  | COF2  | COF1 | COF0 |

**Comparator Registers List**

**CMPC Register**

| Bit  | 7 | 6     | 5    | 4    | 3     | 2     | 1 | 0 |
|------|---|-------|------|------|-------|-------|---|---|
| Name | — | CMPEN | CPOL | CMPO | CNVT1 | CNVT0 | — | — |
| R/W  | — | R/W   | R/W  | R    | R/W   | R/W   | — | — |
| POR  | — | 0     | 0    | 0    | 0     | 0     | — | — |

- Bit 7 Unimplemented, read as "0"
- Bit 6 **CMPEN**: Comparator Enable Control  
 0: Disable  
 1: Enable  
 This is the Comparator on/off control bit. If the bit is zero the comparator will be switched off and no power consumed even if analog voltages are applied to its inputs. For power sensitive applications this bit should be cleared to zero if the comparator is not used or before the devices enter the SLEEP or IDLE mode. Note that the comparator output will be set low when this bit is cleared to zero.
- Bit 5 **CPOL**: Comparator output polarity Control  
 0: Output is not inverted  
 1: Output is inverted  
 This is the Comparator polarity control bit. If the bit is zero then the comparator output bit, CMPnO, will reflect the non-inverted output condition of the comparator. If the bit is high the comparator output bit will be inverted.
- Bit 4 **CMPO**: Comparator output bit  
 If CPOL=0,  
 0: C+ < C-  
 1: C+ > C-  
 If CPOL=1,  
 0: C+ > C-  
 1: C+ < C-  
 This bit stores the Comparator output bit. The polarity of the bit is determined by the voltages on the comparator inputs and by the condition of the CPOL bit.
- Bit 3~2 **CNVT1~CNVT0**: Comparator response time select  
 These bits are used to select the Comparator response time. The response time is also determined by the overdriven voltage applied on the inputs except these bits. Refer to the comparator electrical characteristics for details.
- Bit 1~0 Unimplemented, read as "0"

**CMPVOS register**

| Bit  | 7 | 6    | 5    | 4    | 3    | 2    | 1    | 0    |
|------|---|------|------|------|------|------|------|------|
| Name | — | COFM | CRSP | COF4 | COF3 | COF2 | COF1 | COF0 |
| R/W  | — | R/W  | R/W  | R/W  | R/W  | R/W  | R/W  | R/W  |
| POR  | — | 0    | 0    | 1    | 0    | 0    | 0    | 0    |

- Bit 7 Unimplemented, read as "0"
- Bit 6 **COFM**: Comparator input offset calibration mode select  
 0: Normal operation mode  
 1: Offset calibration mode  
 This is the Comparator offset calibration mode selection bit. If the bit is zero the comparator will operate normally. The comparator will enter the offset calibration mode if this bit is cleared to zero.



- Bit 5      **CRSP**: Comparator input offset calibration reference select  
            0: From C<sup>-</sup> input  
            1: From C<sup>+</sup> input  
            This is the Comparator offset calibration reference selection bit. When the comparator is in the offset calibration mode, the calibration reference input can come from the comparator inputs, C<sup>-</sup> or C<sup>+</sup>, determined by this bit. This bit is only available when the comparator operates in normal mode.
- Bit 4~0    **COF4~COF0**: Comparator input offset calibration control  
            0: From C<sup>-</sup> input  
            1: From C<sup>+</sup> input  
            These bits are used to calibrate the input offset according to the selected reference input when the Comparator is in the offset calibration mode. Refer to the corresponding "Offset Calibration Procedure" section for the detailed calibration procedure.

### Offset calibration procedure

To operate in the input offset calibration mode for the Comparator, the COFM bit should first be set to "1" followed by the reference input selection by configuring the CRSP bit. For comparator n input offset calibration, the procedures are summarized as the following.

- Step 1. Set COFM=1 and configure the CRSP bit, the comparator n will now operate in the comparator input offset calibration mode.
- Step 2. Set COF [4:0]=00000B and read the CMPO bit.
- Step 3. Increase the COF [4:0] value by 1 and then read the CMPO bit.  
If the CMPO bit state has not changed, then repeat Step 3 until the CMPO bit state has changed.  
If the CMPO bit state has changed, record the COF field value as  $V_{OS1}$  and then go to Step 4.
- Step 4. Set COF [4:0]=11111B and read the CMPO bit.
- Step 5. Decrease the COF [4:0] value by 1 and then read the CMPO bit.  
If the CMPO bit state has not changed, then repeat Step 5 until the CMPO bit state has changed.  
If the CMPO bit state has changed, record the COF field value as  $V_{OS2}$  and then go to Step 6.
- Step 6. Restore the comparator input offset calibration value,  $V_{OS}$ , into the COF [4:0] bit field. The offset calibration procedure is now finished and the input offset value is calculated by the following equation.

$$V_{OS} = \frac{V_{OS1} + V_{OS2}}{2}$$

## Serial Interface Module – SIM

The device contains a Serial Interface Module, which includes both the four-line SPI interface or two-line I<sup>2</sup>C interface types, to allow an easy method of communication with external peripheral hardware. Having relatively simple communication protocols, these serial interface types allow the microcontroller to interface to external SPI or I<sup>2</sup>C based hardware such as sensors, Flash or EEPROM memory, etc. The SIM interface pins are pin-shared with other I/O pins and therefore the SIM interface functional pins must first be selected using the corresponding pin-shared function selection bits. As both interface types share the same pins and registers, the choice of whether the SPI or I<sup>2</sup>C type is used is made using the SIM operating mode control bits, named SIM2~SIM0, in the SIMC0 register. These pull-high resistors of the SIM pin-shared I/O pins are selected using pull-high control registers when the SIM function is enabled and the corresponding pins are used as SIM input pins.

### SPI Interface

The SPI interface is often used to communicate with external peripheral devices such as sensors, Flash or EEPROM memory devices, etc. Originally developed by Motorola, the four line SPI interface is a synchronous serial data interface that has a relatively simple communication protocol simplifying the programming requirements when communicating with external hardware devices.

The communication is full duplex and operates as a slave/master type, where the devices can be either master or slave. Although the SPI interface specification can control multiple slave devices from a single master, the device provided only one  $\overline{\text{SCS}}$  pin. If the master needs to control multiple slave devices from a single master, the master can use I/O pin to select the slave devices.

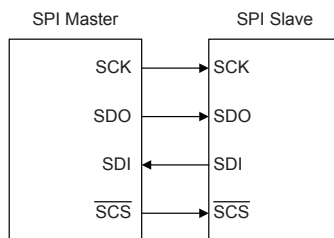
### SPI Interface Operation

The SPI interface is a full duplex synchronous serial data link. It is a four line interface with pin names SDI, SDO, SCK and  $\overline{\text{SCS}}$ . Pins SDI and SDO are the Serial Data Input and Serial Data Output lines, SCK is the Serial Clock line and  $\overline{\text{SCS}}$  is the Slave Select line. As the SPI interface pins are pin-shared with normal I/O pins and with the I<sup>2</sup>C function pins, the SPI interface pins must first be selected by configuring the pin-shared function selection bits and setting the correct bits in the SIMC0 and SIMC2 registers. After the desired SPI configuration has been set it can be disabled or enabled using the SIMEN bit in the SIMC0 register. Communication between devices connected to the SPI interface is carried out in a slave/master mode with all data transfer initiations being implemented by the master. The Master also controls the clock signal. As the device only contains a single  $\overline{\text{SCS}}$  pin only one slave device can be utilized. The  $\overline{\text{SCS}}$  pin is controlled by software, set CSEN bit to 1 to enable  $\overline{\text{SCS}}$  pin function, set CSEN bit to 0 the  $\overline{\text{SCS}}$  pin will be floating state.

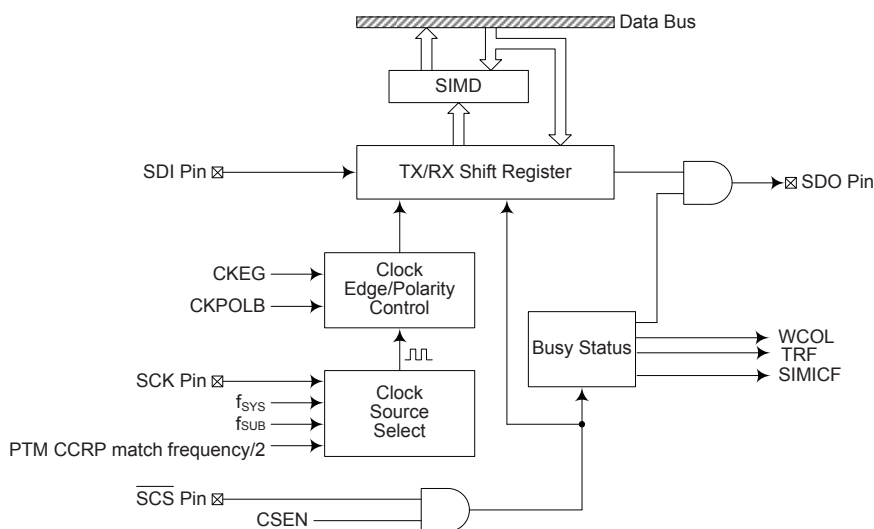
The SPI function in this device offers the following features:

- Full duplex synchronous data transfer
- Both Master and Slave modes
- LSB first or MSB first data transmission modes
- Transmission complete flag
- Rising or falling active clock edge

The status of the SPI interface pins is determined by a number of factors such as whether the device is in the master or slave mode and upon the condition of certain control bits such as CSEN and SIMEN.



**SPI Master/Slave Connection**



**SPI Block Diagram**

**SPI Registers**

There are three internal registers which control the overall operation of the SPI interface. These are the SIMD data register and two registers SIMC0 and SIMC2. Note that the SIMC1 register is only used by the I<sup>2</sup>C interface.

| Register Name | Bit  |      |        |      |         |         |       |        |
|---------------|------|------|--------|------|---------|---------|-------|--------|
|               | 7    | 6    | 5      | 4    | 3       | 2       | 1     | 0      |
| SIMC0         | SIM2 | SIM1 | SIM0   | —    | SIMDEB1 | SIMDEB0 | SIMEN | SIMICF |
| SIMC2         | D7   | D6   | CKPOLB | CKEG | MLS     | CSEN    | WCOL  | TRF    |
| SIMD          | D7   | D6   | D5     | D4   | D3      | D2      | D1    | D0     |

**SPI Registers List**

• **SIMD Register**

The SIMD register is used to store the data being transmitted and received. The same register is used by both the SPI and I<sup>2</sup>C functions. Before the device writes data to the SPI bus, the actual data to be transmitted must be placed in the SIMD register. After the data is received from the SPI bus, the device can read it from the SIMD register. Any transmission or reception of data from the SPI bus must be made via the SIMD 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  | x   | x   | x   | x   | x   | x   | x   | x   |

"x": unknown

There are also two control registers for the SPI interface, SIMC0 and SIMC2. Note that the SIMC2 register also has the name SIMA which is used by the I<sup>2</sup>C function. The SIMC1 register is not used by the SPI function, only by the I<sup>2</sup>C function. Register SIMC0 is used to control the enable/disable function and to set the data transmission clock frequency. Register SIMC2 is used for other control functions such as LSB/MSB selection, write collision flag, etc.

• **SIMC0 Register**

| Bit  | 7    | 6    | 5    | 4 | 3       | 2       | 1     | 0      |
|------|------|------|------|---|---------|---------|-------|--------|
| Name | SIM2 | SIM1 | SIM0 | — | SIMDEB1 | SIMDEB0 | SIMEN | SIMICF |
| R/W  | R/W  | R/W  | R/W  | — | R/W     | R/W     | R/W   | R/W    |
| POR  | 1    | 1    | 1    | — | 0       | 0       | 0     | 0      |

Bit 7~5 **SIM2~SIM0**: SIM Operating Mode Control  
 000: SPI master mode; SPI clock is  $f_{SYS}/4$   
 001: SPI master mode; SPI clock is  $f_{SYS}/16$   
 010: SPI master mode; SPI clock is  $f_{SYS}/64$   
 011: SPI master mode; SPI clock is  $f_{SUB}$   
 100: SPI master mode; SPI clock is PTM CCRP match frequency/2  
 101: SPI slave mode  
 110: I<sup>2</sup>C slave mode  
 111: Unused mode

These bits setup the overall operating mode of the SIM function. As well as selecting if the I<sup>2</sup>C or SPI function, they are used to control the SPI Master/Slave selection and the SPI Master clock frequency. The SPI clock is a function of the system clock but can also be chosen to be sourced from PTM and  $f_{SUB}$ . If the SPI Slave Mode is selected then the clock will be supplied by an external Master device.

Bit 4 Unimplemented, read as "0"

Bit 3~2 **SIMDEB1~SIMDEB0**: I<sup>2</sup>C Debounce Time Selection  
 Described elsewhere.

Bit 1 **SIMEN**: SIM enable control  
 0: Disable  
 1: Enable

The bit is the overall on/off control for the SIM interface. When the SIMEN bit is cleared to zero to disable the SIM interface, the SDI, SDO, SCK and  $\overline{SCS}$ , or SDA and SCL lines will lose their SPI or I<sup>2</sup>C function and the SIM operating current will be reduced to a minimum value. When the bit is high the SIM interface is enabled. The SIM configuration option must have first enabled the SIM interface for this bit to be effective. If the SIM is configured to operate as an SPI interface via the SIM2~SIM0 bits, the contents of the SPI control registers will remain at the previous settings when the SIMEN bit changes from low to high and should therefore be first initialised by the application program. If the SIM is configured to operate as an I<sup>2</sup>C interface via the SIM2~SIM0 bits and the SIMEN bit changes from low to high, the contents of the I<sup>2</sup>C control bits such as HTX and TXAK will remain at the previous settings and should therefore be first initialised by the application program while the relevant I<sup>2</sup>C flags such as HCF, HAAS, HBB, SRW and RXAK will be set to their default states.

Bit 0 **SIMICF**: SIM SPI slave mode Incomplete Transfer Flag  
 0: SIM SPI slave mode incomplete condition not occurred  
 1: SIM SPI slave mode incomplete condition occurred

This bit is only available when the SIM is configured to operate in an SPI slave mode. If the SPI operates in the slave mode with the SIMEN and CSEN bits both being set to 1 but the  $\overline{SCS}$  line is pulled high by the external master device before the SPI data transfer is completely finished, the SIMICF bit will be set to 1 together with the TRF bit. When this condition occurs, the corresponding interrupt will occur if the interrupt function is enabled. However, the TRF bit will not be set to 1 if the SIMICF bit is set to 1 by software application program.

• **SIMC2 Register**

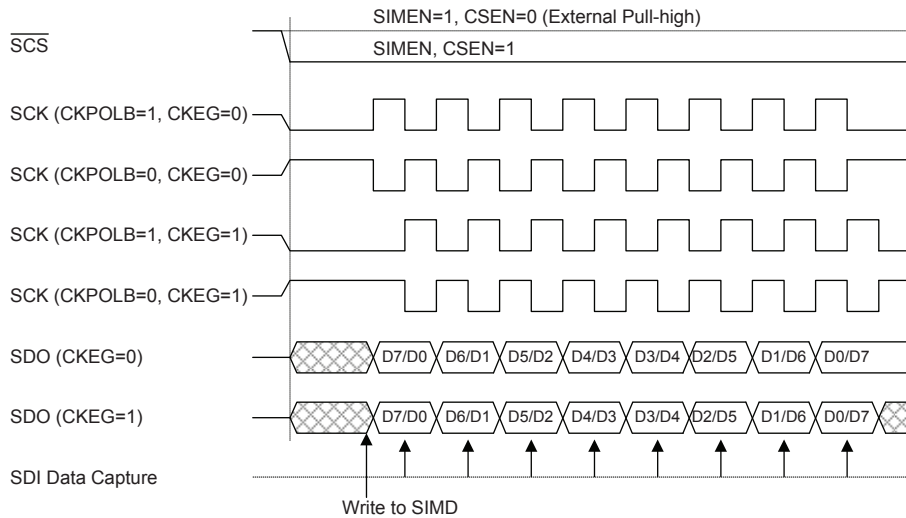
| Bit  | 7   | 6   | 5      | 4    | 3   | 2    | 1    | 0   |
|------|-----|-----|--------|------|-----|------|------|-----|
| Name | D7  | D6  | CKPOLB | CKEG | MLS | CSEN | WCOL | TRF |
| R/W  | R/W | R/W | R/W    | R/W  | R/W | R/W  | R/W  | R/W |
| POR  | 0   | 0   | 0      | 0    | 0   | 0    | 0    | 0   |

- Bit 7~6     Undefined bits  
 These bits can be read or written by the application program.
- Bit 5     **CKPOLB**: SPI clock line base condition selection  
           0: The SCK line will be high when the clock is inactive.  
           1: The SCK line will be low when the clock is inactive.  
 The CKPOLB bit determines the base condition of the clock line, if the bit is high, then the SCK line will be low when the clock is inactive. When the CKPOLB bit is low, then the SCK line will be high when the clock is inactive.
- Bit 4     **CKEG**: SPI SCK clock active edge type selection  
 CKPOLB=0  
           0: SCK is high base level and data capture at SCK rising edge  
           1: SCK is high base level and data capture at SCK falling edge  
 CKPOLB=1  
           0: SCK is low base level and data capture at SCK falling edge  
           1: SCK is low base level and data capture at SCK rising edge  
 The CKEG and CKPOLB bits are used to setup the way that the clock signal outputs and inputs data on the SPI bus. These two bits must be configured before data transfer is executed otherwise an erroneous clock edge may be generated. The CKPOLB bit determines the base condition of the clock line, if the bit is high, then the SCK line will be low when the clock is inactive. When the CKPOLB bit is low, then the SCK line will be high when the clock is inactive. The CKEG bit determines active clock edge type which depends upon the condition of CKPOLB bit.
- Bit 3     **MLS**: SPI data shift order  
           0: LSB first  
           1: MSB first  
 This is the data shift select bit and is used to select how the data is transferred, either MSB or LSB first. Setting the bit high will select MSB first and low for LSB first.
- Bit 2     **CSEN**: SPI  $\overline{\text{SCS}}$  pin control  
           0: Disable  
           1: Enable  
 The CSEN bit is used as an enable/disable for the  $\overline{\text{SCS}}$  pin. If this bit is low, then the  $\overline{\text{SCS}}$  pin will be disabled and placed into I/O pin or other pin-shared functions. If the bit is high, the  $\overline{\text{SCS}}$  pin will be enabled and used as a select pin.
- Bit 1     **WCOL**: SPI write collision flag  
           0: No collision  
           1: Collision  
 The WCOL flag is used to detect whether a data collision has occurred or not. If this bit is high, it means that data has been attempted to be written to the SIMD register during a data transfer operation. This writing operation will be ignored if data is being transferred. This bit can be cleared by the application program.
- Bit 0     **TRF**: SPI Transmit/Receive complete flag  
           0: SPI data is being transferred  
           1: SPI data transfer is completed  
 The TRF bit is the Transmit/Receive Complete flag and is set to 1 automatically when an SPI data transfer is completed, but must be cleared to 0 by the application program. It can be used to generate an interrupt.

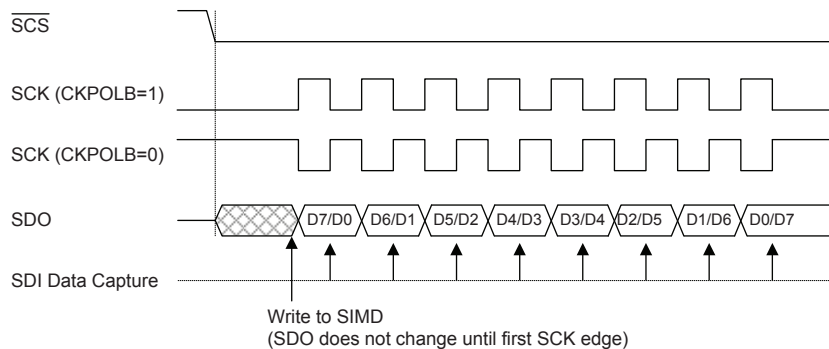
**SPI Communication**

After the SPI interface is enabled by setting the SIMEN bit high, then in the Master Mode, when data is written to the SIMD register, transmission/reception will begin simultaneously. When the data transfer is complete, the TRF flag will be set automatically, but must be cleared using the application program. In the Slave Mode, when the clock signal from the master has been received, any data in the SIMD register will be transmitted and any data on the SDI pin will be shifted into the SIMD register. The master should output a  $\overline{SCS}$  signal to enable the slave devices before a clock signal is provided. The slave data to be transferred should be well prepared at the appropriate moment relative to the  $\overline{SCS}$  signal depending upon the configurations of the CKPOLB bit and CKEG bit. The accompanying timing diagram shows the relationship between the slave data and  $\overline{SCS}$  signal for various configurations of the CKPOLB and CKEG bits.

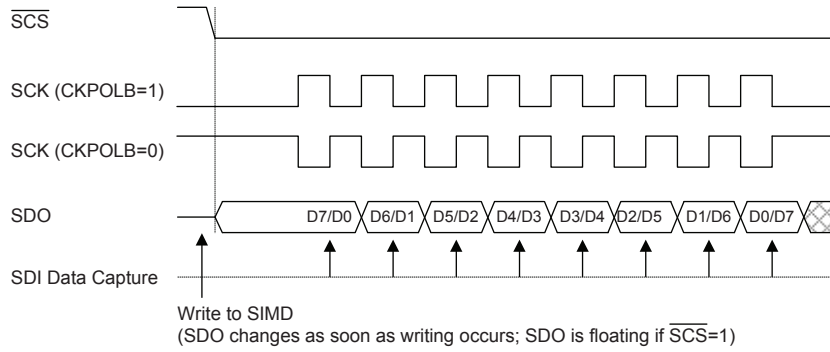
The SPI will continue to function in special IDLE Modes if the clock source used by the SPI interface is still active.



**SPI Master Mode Timing**



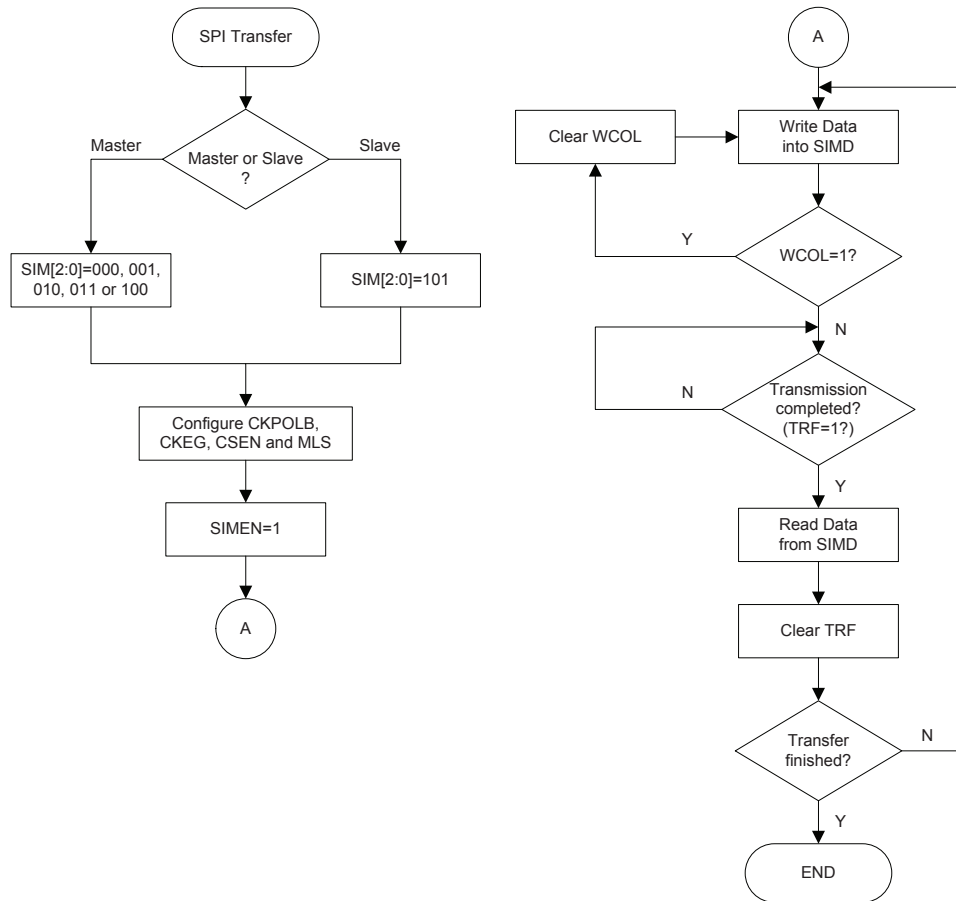
**SPI Slave Mode Timing – CKEG=0**



Note: For SPI slave mode, if SIMEN=1 and CSEN=0, SPI is always enabled and ignores the SCS level.

Note: For SPI slave mode, if SIMEN=1 and CSEN=0, the SPI is always enabled and ignores the SCS level.

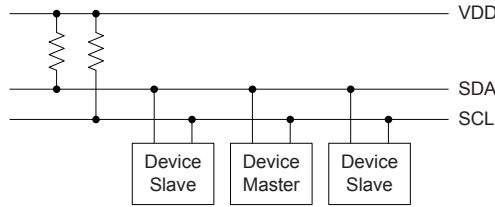
**SPI Slave Mode Timing – CKEG=1**



**SPI Transfer Control Flow Chart**

### I<sup>2</sup>C Interface

The I<sup>2</sup>C interface is used to communicate with external peripheral devices such as sensors, EEPROM memory etc. Originally developed by Philips, it is a two line low speed serial interface for synchronous serial data transfer. The advantage of only two lines for communication, relatively simple communication protocol and the ability to accommodate multiple devices on the same bus has made it an extremely popular interface type for many applications.

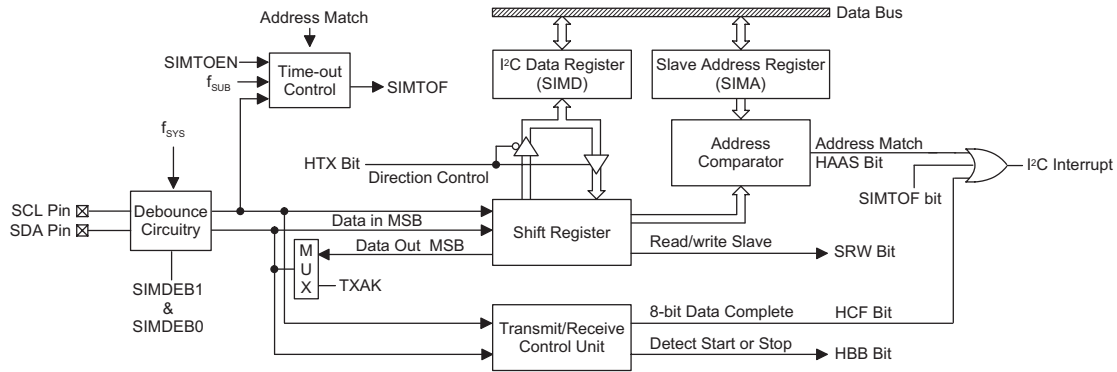


**I<sup>2</sup>C Master/Slave Bus Connection**

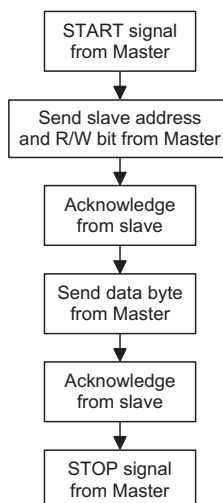
### I<sup>2</sup>C interface Operation

The I<sup>2</sup>C serial interface is a two line interface, a serial data line, SDA, and serial clock line, SCL. As many devices may be connected together on the same bus, their outputs are both open drain types. For this reason it is necessary that external pull-high resistors are connected to these outputs. Note that no chip select line exists, as each device on the I<sup>2</sup>C bus is identified by a unique address which will be transmitted and received on the I<sup>2</sup>C bus.

When two devices communicate with each other on the bidirectional I<sup>2</sup>C bus, one is known as the master device and one as the slave device. Both master and slave can transmit and receive data, however, it is the master device that has overall control of the bus. For the device, which only operate in slave mode, there are two methods of transferring data on the I<sup>2</sup>C bus, the slave transmit mode and the slave receive mode.







The SIMDEB1 and SIMDEB0 bits determine the debounce time of the I<sup>2</sup>C interface. This uses the system clock to in effect add a debounce time to the external clock to reduce the possibility of glitches on the clock line causing erroneous operation. The debounce time, if selected, can be chosen to be either 2 or 4 system clocks. To achieve the required I<sup>2</sup>C data transfer speed, there exists a relationship between the system clock,  $f_{SYS}$ , and the I<sup>2</sup>C debounce time. For either the I<sup>2</sup>C Standard or Fast mode operation, users must take care of the selected system clock frequency and the configured debounce time to match the criterion shown in the following table.

| I <sup>2</sup> C Debounce Time Selection | I <sup>2</sup> C Standard Mode (100kHz) | I <sup>2</sup> C Fast Mode (400kHz) |
|--|---|-------------------------------------|
| No Debounce                              | $f_{SYS} > 2 \text{ MHz}$               | $f_{SYS} > 5 \text{ MHz}$           |
| 2 system clock debounce                  | $f_{SYS} > 4 \text{ MHz}$               | $f_{SYS} > 10 \text{ MHz}$          |
| 4 system clock debounce                  | $f_{SYS} > 8 \text{ MHz}$               | $f_{SYS} > 20 \text{ MHz}$          |

I<sup>2</sup>C Minimum  $f_{SYS}$  Frequency

### I<sup>2</sup>C Registers

There are three control registers associated with the I<sup>2</sup>C bus, SIMC0, SIMC1 and SIMTOC, one address register SIMA, and one data register, SIMD. The SIMD register, which is shown in the above SPI section, is used to store the data being transmitted and received on the I<sup>2</sup>C bus. Before the microcontroller writes data to the I<sup>2</sup>C bus, the actual data to be transmitted must be placed in the SIMD register. After the data is received from the I<sup>2</sup>C bus, the microcontroller can read it from the SIMD register. Any transmission or reception of data from the I<sup>2</sup>C bus must be made via the SIMD register.

Note that the SIMA register also has the name SIMC2 which is used by the SPI function. Bit SIMEN and bits SIM2~SIM0 in register SIMC0 are used by the I<sup>2</sup>C interface.

| Register Name | Bit     |        |         |         |         |         |         |         |
|---------------|---------|--------|---------|---------|---------|---------|---------|---------|
|               | 7       | 6      | 5       | 4       | 3       | 2       | 1       | 0       |
| SIMC0         | SIM2    | SIM1   | SIM0    | —       | SIMDEB1 | SIMDEB0 | SIMEN   | SIMICF  |
| SIMC1         | HCF     | HAAS   | HBB     | HTX     | TXAK    | SRW     | IAMWU   | RXAK    |
| SIMA          | A6      | A5     | A4      | A3      | A2      | A1      | A0      | D0      |
| SIMD          | D7      | D6     | D5      | D4      | D3      | D2      | D1      | D0      |
| SIMTOC        | SIMTOEN | SIMTOF | SIMTOS5 | SIMTOS4 | SIMTOS3 | SIMTOS2 | SIMTOS1 | SIMTOS0 |

I<sup>2</sup>C Registers List

• **SIMD Register**

The SIMD register is used to store the data being transmitted and received. The same register is used by both the SPI and I<sup>2</sup>C functions. Before the device writes data to the I<sup>2</sup>C bus, the actual data to be transmitted must be placed in the SIMD register. After the data is received from the I<sup>2</sup>C bus, the device can read it from the SIMD register. Any transmission or reception of data from the I<sup>2</sup>C bus must be made via the SIMD 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  | x   | x   | x   | x   | x   | x   | x   | x   |

"x": unknown

• **SIMA Register**

The SIMA register is also used by the SPI interface but has the name SIMC2. The SIMA register is the location where the 7-bit slave address of the slave device is stored. Bits 7~1 of the SIMA register define the device slave address. Bit 0 is not defined.

When a master device, which is connected to the I<sup>2</sup>C bus, sends out an address, which matches the slave address in the SIMA register, the slave device will be selected. Note that the SIMA register is the same register address as SIMC2 which is used by the SPI interface.

| Bit  | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   |
|------|-----|-----|-----|-----|-----|-----|-----|-----|
| Name | A6  | A5  | A4  | A3  | A2  | A1  | A0  | 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 7~1 I<sup>2</sup>C slave address bit 7 ~ bit 1

Bit 0 Undefined bit

The bit can be read or written by the application program.

There are also three control registers for the I<sup>2</sup>C interface, SIMC0, SIMC1 and SIMTOC. The register SIMC0 is used to control the enable/disable function and to set the data transmission clock frequency. The SIMC1 register contains the relevant flags which are used to indicate the I<sup>2</sup>C communication status. Another register, SIMTOC, is used to control the I<sup>2</sup>C time-out function and described in the corresponding section.

• **SIMC0 Register**

| Bit  | 7    | 6    | 5    | 4 | 3       | 2       | 1     | 0      |
|------|------|------|------|---|---------|---------|-------|--------|
| Name | SIM2 | SIM1 | SIM0 | — | SIMDEB1 | SIMDEB0 | SIMEN | SIMICF |
| R/W  | R/W  | R/W  | R/W  | — | R/W     | R/W     | R/W   | R/W    |
| POR  | 1    | 1    | 1    | — | 0       | 0       | 0     | 0      |

Bit 7~5 **SIM2~SIM0**: SIM Operating Mode Control

- 000: SPI master mode; SPI clock is  $f_{SYS}/4$
- 001: SPI master mode; SPI clock is  $f_{SYS}/16$
- 010: SPI master mode; SPI clock is  $f_{SYS}/64$
- 011: SPI master mode; SPI clock is  $f_{SUB}$
- 100: SPI master mode; SPI clock is PTM CCRP match frequency/2
- 101: SPI slave mode
- 110: I<sup>2</sup>C slave mode
- 111: Unused mode

These bits setup the overall operating mode of the SIM function. As well as selecting if the I<sup>2</sup>C or SPI function, they are used to control the SPI Master/Slave selection and the SPI Master clock frequency. The SPI clock is a function of the system clock but can also be chosen to be sourced from PTM and  $f_{SUB}$ . If the SPI Slave Mode is selected then the clock will be supplied by an external Master device.

- Bit 4 Unimplemented, read as "0"
- Bit 3~2 **SIMDEB1~SIMDEB0**: I<sup>2</sup>C Debounce Time Selection
  - 00: No debounce
  - 01: 2 system clock debounce
  - 1x: 4 system clock debounce

- Bit 1 **SIMEN**: SIM Enable Control
  - 0: Disable
  - 1: Enable

The bit is the overall on/off control for the SIM interface. When the SIMEN bit is cleared to zero to disable the SIM interface, the SDI, SDO, SCK and  $\overline{SCS}$ , or SDA and SCL lines will lose their SPI or I<sup>2</sup>C function and the SIM operating current will be reduced to a minimum value. When the bit is high the SIM interface is enabled. The SIM configuration option must have first enabled the SIM interface for this bit to be effective. If the SIM is configured to operate as an SPI interface via the SIM2~SIM0 bits, the contents of the SPI control registers will remain at the previous settings when the SIMEN bit changes from low to high and should therefore be first initialised by the application program. If the SIM is configured to operate as an I<sup>2</sup>C interface via the SIM2~SIM0 bits and the SIMEN bit changes from low to high, the contents of the I<sup>2</sup>C control bits such as HTX and TXAK will remain at the previous settings and should therefore be first initialised by the application program while the relevant I<sup>2</sup>C flags such as HCF, HAAS, HBB, SRW and RXAK will be set to their default states.

- Bit 0 **SIMICF**: SIM SPI slave mode Incomplete Transfer Flag  
 Described elsewhere.

• **SIMC1 Register**

| Bit  | 7   | 6    | 5   | 4   | 3    | 2   | 1     | 0    |
|------|-----|------|-----|-----|------|-----|-------|------|
| Name | HCF | HAAS | HBB | HTX | TXAK | SRW | IAMWU | RXAK |
| R/W  | R   | R    | R   | R/W | R/W  | R   | R/W   | R    |
| POR  | 1   | 0    | 0   | 0   | 0    | 0   | 0     | 1    |

- Bit 7 **HCF**: I<sup>2</sup>C Bus data transfer completion flag
  - 0: Data is being transferred
  - 1: Completion of an 8-bit data transfer

The HCF flag is the data transfer flag. This flag will be zero when data is being transferred. Upon completion of an 8-bit data transfer the flag will go high and an interrupt will be generated.
- Bit 6 **HAAS**: I<sup>2</sup>C Bus data transfer completion flag
  - 0: Not address match
  - 1: Address match

The HAAS flag is the address match flag. This flag is used to determine if the slave device address is the same as the master transmit address. If the addresses match then this bit will be high, if there is no match then the flag will be low.
- Bit 5 **HBB**: I<sup>2</sup>C Bus busy flag
  - 0: I<sup>2</sup>C Bus is not busy
  - 1: I<sup>2</sup>C Bus is busy

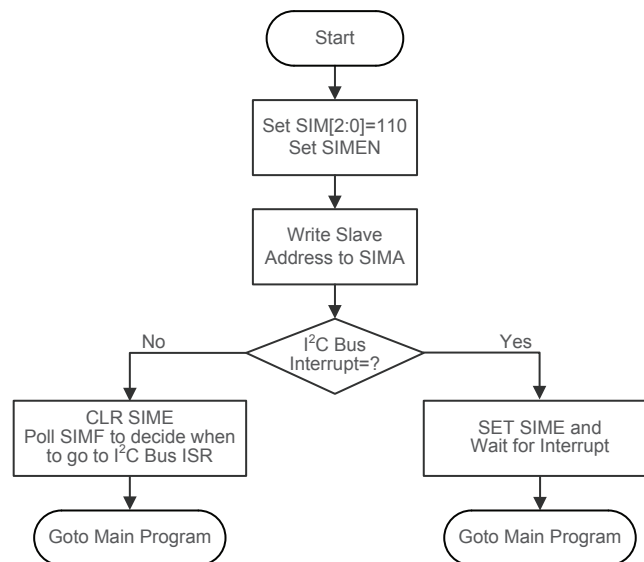
The HBB flag is the I<sup>2</sup>C busy flag. This flag will be "1" when the I<sup>2</sup>C bus is busy which will occur when a START signal is detected. The flag will be set to "0" when the bus is free which will occur when a STOP signal is detected.
- Bit 4 **HTX**: I<sup>2</sup>C slave device transmitter/receiver selection
  - 0: Slave device is the receiver
  - 1: Slave device is the transmitter

- Bit 3      **TXAK:** I<sup>2</sup>C bus transmit acknowledge flag  
            0: Slave send acknowledge flag  
            1: Slave does not send acknowledge flag  
The TXAK bit is the transmit acknowledge flag. After the slave device receipt of 8-bits of data, this bit will be transmitted to the bus on the 9<sup>th</sup> clock from the slave device. The slave device must always set TXAK bit to "0" before further data is received.
- Bit 2      **SRW:** I<sup>2</sup>C slave read/write flag  
            0: Slave device should be in receive mode  
            1: Slave device should be in transmit mode  
The SRW flag is the I<sup>2</sup>C Slave Read/Write flag. This flag determines whether the master device wishes to transmit or receive data from the I<sup>2</sup>C bus. When the transmitted address and slave address is match, that is when the HAAS flag is set high, the slave device will check the SRW flag to determine whether it should be in transmit mode or receive mode. If the SRW flag is high, the master is requesting to read data from the bus, so the slave device should be in transmit mode. When the SRW flag is zero, the master will write data to the bus, therefore the slave device should be in receive mode to read this data.
- Bit 1      **IAMWU:** I<sup>2</sup>C Address Match Wake-Up control  
            0: Disable  
            1: Enable – must be cleared by the application program after wake-up  
This bit should be set to 1 to enable the I<sup>2</sup>C address match wake up from the SLEEP or IDLE Mode. If the IAMWU bit has been set before entering either the SLEEP or IDLE mode to enable the I<sup>2</sup>C address match wake up, then this bit must be cleared by the application program after wake-up to ensure correction device operation.
- Bit 0      **RXAK:** I<sup>2</sup>C bus receive acknowledge flag  
            0: Slave receives acknowledge flag  
            1: Slave does not receive acknowledge flag  
The RXAK flag is the receiver acknowledge flag. When the RXAK flag is "0", it means that a acknowledge signal has been received at the 9<sup>th</sup> clock, after 8 bits of data have been transmitted. When the slave device in the transmit mode, the slave device checks the RXAK flag to determine if the master receiver wishes to receive the next byte. The slave transmitter will therefore continue sending out data until the RXAK flag is "1". When this occurs, the slave transmitter will release the SDA line to allow the master to send a STOP signal to release the I<sup>2</sup>C Bus.

**I<sup>2</sup>C Bus Communication**

Communication on the I<sup>2</sup>C bus requires four separate steps, a START signal, a slave device address transmission, a data transmission and finally a STOP signal. When a START signal is placed on the I<sup>2</sup>C bus, all devices on the bus will receive this signal and be notified of the imminent arrival of data on the bus. The first seven bits of the data will be the slave address with the first bit being the MSB. If the address of the slave device matches that of the transmitted address, the HAAS bit in the SIMC1 register will be set and an I<sup>2</sup>C interrupt will be generated. After entering the interrupt service routine, the slave device must first check the condition of the HAAS and SIMTOF bits to determine whether the interrupt source originates from an address match, 8-bit data transfer completion or I<sup>2</sup>C bus time-out occurrence. During a data transfer, note that after the 7-bit slave address has been transmitted, the following bit, which is the 8<sup>th</sup> bit, is the read/write bit whose value will be placed in the SRW bit. This bit will be checked by the slave device to determine whether to go into transmit or receive mode. Before any transfer of data to or from the I<sup>2</sup>C bus, the microcontroller must initialise the bus, the following are steps to achieve this:

- Step 1  
 Set the SIM2~SIM0 bits to "110" and SIMEN bit to "1" in the SIMC0 register to enable the I<sup>2</sup>C bus.
- Step 2  
 Write the slave address of the device to the I<sup>2</sup>C bus address register SIMA.
- Step 3  
 Set the SIME interrupt enable bit of the interrupt control register to enable the SIM interrupt.



**I<sup>2</sup>C Bus Initialisation Flow Chart**

**I<sup>2</sup>C Bus Start Signal**

The START signal can only be generated by the master device connected to the I<sup>2</sup>C bus and not by the slave device. This START signal will be detected by all devices connected to the I<sup>2</sup>C bus. When detected, this indicates that the I<sup>2</sup>C bus is busy and therefore the HBB bit will be set. A START condition occurs when a high to low transition on the SDA line takes place when the SCL line remains high.

**I<sup>2</sup>C Slave Address**

The transmission of a START signal by the master will be detected by all devices on the I<sup>2</sup>C bus. To determine which slave device the master wishes to communicate with, the address of the slave device will be sent out immediately following the START signal. All slave devices, after receiving this 7-bit address data, will compare it with their own 7-bit slave address. If the address sent out by the master matches the internal address of the microcontroller slave device, then an internal I<sup>2</sup>C bus interrupt signal will be generated. The next bit following the address, which is the 8<sup>th</sup> bit, defines the read/write status and will be saved to the SRW bit of the SIMC1 register. The slave device will then transmit an acknowledge bit, which is a low level, as the 9<sup>th</sup> bit. The slave device will also set the status flag HAAS when the addresses match.

As an I<sup>2</sup>C bus interrupt can come from three sources, when the program enters the interrupt subroutine, the HAAS and SIMTOF bits should be examined to see whether the interrupt source has come from a matching slave address, the completion of a data byte transfer or the I<sup>2</sup>C bus time-out occurrence. When a slave address is matched, the device must be placed in either the transmit mode and then write data to the SIMD register, or in the receive mode where it must implement a dummy read from the SIMD register to release the SCL line.

**I<sup>2</sup>C Bus Read/Write Signal**

The SRW bit in the SIMC1 register defines whether the slave device wishes to read data from the I<sup>2</sup>C bus or write data to the I<sup>2</sup>C bus. The slave device should examine this bit to determine if it is to be a transmitter or a receiver. If the SRW flag is "1" then this indicates that the master device wishes to read data from the I<sup>2</sup>C bus, therefore the slave device must be setup to send data to the I<sup>2</sup>C bus as a transmitter. If the SRW flag is "0" then this indicates that the master wishes to send data to the I<sup>2</sup>C bus, therefore the slave device must be setup to read data from the I<sup>2</sup>C bus as a receiver.

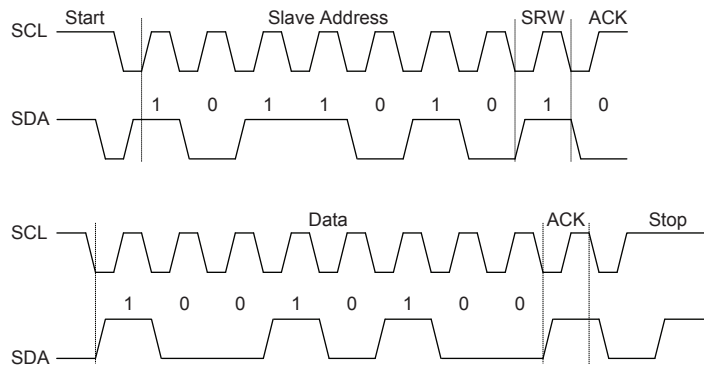
**I<sup>2</sup>C Bus Slave Address Acknowledge Signal**

After the master has transmitted a calling address, any slave device on the I<sup>2</sup>C bus, whose own internal address matches the calling address, must generate an acknowledge signal. The acknowledge signal will inform the master that a slave device has accepted its calling address. If no acknowledge signal is received by the master then a STOP signal must be transmitted by the master to end the communication. When the HAAS flag is high, the addresses have matched and the slave device must check the SRW flag to determine if it is to be a transmitter or a receiver. If the SRW flag is high, the slave device should be setup to be a transmitter so the HTX bit in the SIMC1 register should be set to "1". If the SRW flag is low, then the microcontroller slave device should be setup as a receiver and the HTX bit in the SIMC1 register should be set to "0".

**I<sup>2</sup>C Bus Data and Acknowledge Signal**

The transmitted data is 8-bits wide and is transmitted after the slave device has acknowledged receipt of its slave address. The order of serial bit transmission is the MSB first and the LSB last. After receipt of 8-bits of data, the receiver must transmit an acknowledge signal, level "0", before it can receive the next data byte. If the slave transmitter does not receive an acknowledge bit signal from the master receiver, then the slave transmitter will release the SDA line to allow the master to send a STOP signal to release the I<sup>2</sup>C Bus. The corresponding data will be stored in the SIMD register. If setup as a transmitter, the slave device must first write the data to be transmitted into the SIMD register. If setup as a receiver, the slave device must read the transmitted data from the SIMD register.

When the slave receiver receives the data byte, it must generate an acknowledge bit, known as TXAK, on the 9<sup>th</sup> clock. The slave device, which is setup as a transmitter will check the RXAK bit in the SIMC1 register to determine if it is to send another data byte, if not then it will release the SDA line and await the receipt of a STOP signal from the master.

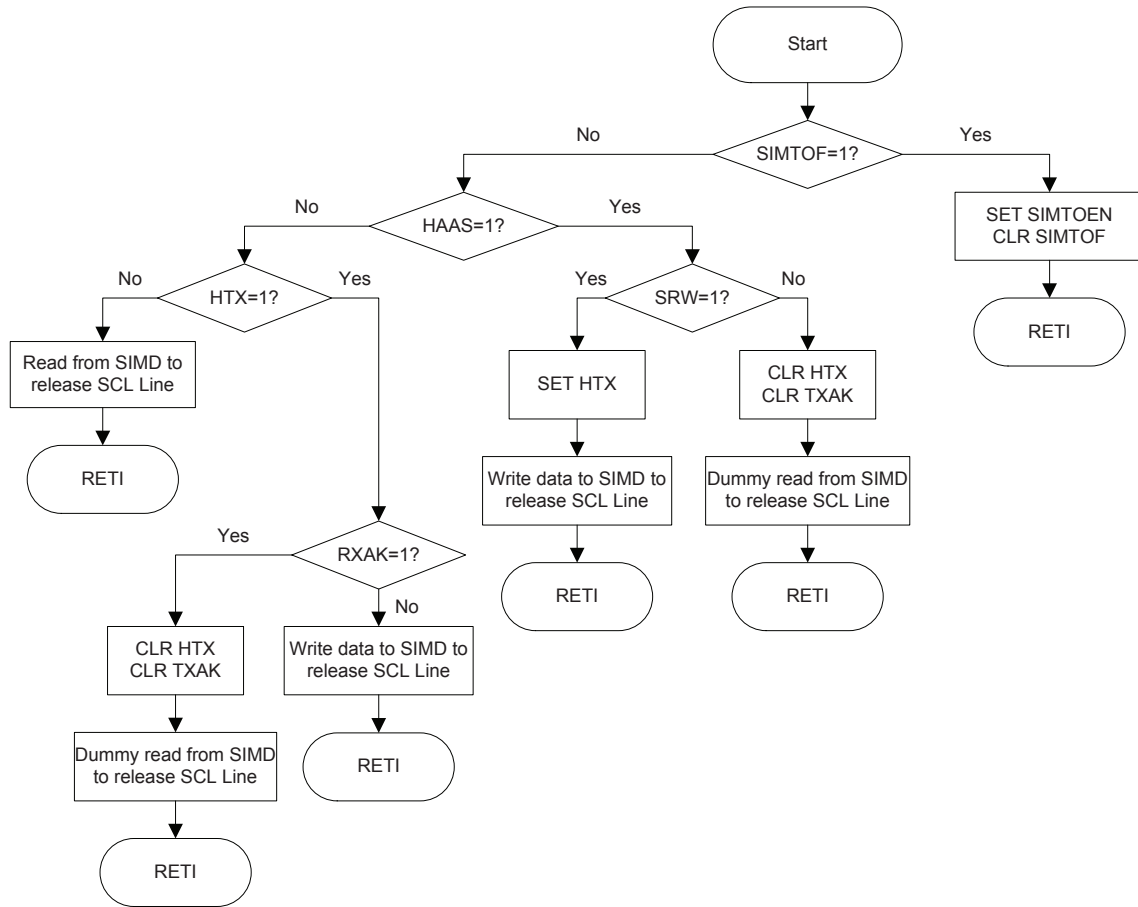


- S=Start (1 bit)
- SA=Slave Address (7 bits)
- SR=SRW bit (1 bit)
- M=Slave device send acknowledge bit (1 bit)
- D=Data (8 bits)
- A=ACK (RXAK bit for transmitter, TXAK bit for receiver, 1 bit)
- P=Stop (1 bit)

|   |    |    |   |   |   |   |   |       |   |    |    |   |   |   |   |   |       |   |
|---|----|----|---|---|---|---|---|-------|---|----|----|---|---|---|---|---|-------|---|
| S | SA | SR | M | D | A | D | A | ..... | S | SA | SR | M | D | A | D | A | ..... | P |
|---|----|----|---|---|---|---|---|-------|---|----|----|---|---|---|---|---|-------|---|

Note: \*When a slave address is matched, the device must be placed in either the transmit mode and then write data to the SIMD register, or in the receive mode where it must implement a dummy read from the SIMD register to release the SCL line.

**I<sup>2</sup>C Communication Timing Diagram**

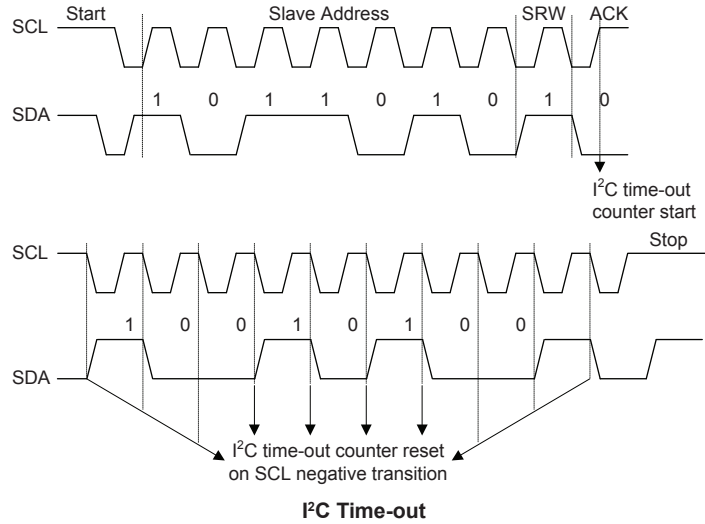


**I<sup>2</sup>C Bus ISR Flow Chart**

**I<sup>2</sup>C Time-out Control**

In order to reduce the I<sup>2</sup>C lockup problem due to reception of erroneous clock sources, a time-out function is provided. If the clock source connected to the I<sup>2</sup>C bus is not received for a while, then the I<sup>2</sup>C circuitry and registers will be reset after a certain time-out period. The time-out counter starts to count on an I<sup>2</sup>C bus "START" & "address match" condition, and is cleared by an SCL falling edge. Before the next SCL falling edge arrives, if the time elapsed is greater than the time-out period specified by the SIMTOC register, then a time-out condition will occur. The time-out function will stop when an I<sup>2</sup>C "STOP" condition occurs.





When an I<sup>2</sup>C time-out counter overflow occurs, the counter will stop and the SIMTOEN bit will be cleared to zero and the SIMTOF bit will be set high to indicate that a time-out condition has occurred. The time-out condition will also generate an interrupt which uses the I<sup>2</sup>C interrupt vector. When an I<sup>2</sup>C time-out occurs, the I<sup>2</sup>C internal circuitry will be reset and the registers will be reset into the following condition:

| Register          | After I <sup>2</sup> C Time-out |
|-------------------|---------------------------------|
| SIMD, SIMA, SIMC0 | No change                       |
| SIMC1             | Reset to POR condition          |

**I<sup>2</sup>C Register after Time-out**

The SIMTOF flag can be cleared by the application program. There are 64 time-out period selections which can be selected using the SIMTOS bits in the SIMTOC register. The time-out duration is calculated by the formula:  $((1 \sim 64) \times (32/f_{SUB}))$ . This gives a time-out period which ranges from about 1ms to 64ms.

• **SIMTOC Register**

| Bit  | 7       | 6      | 5       | 4       | 3       | 2       | 1       | 0       |
|------|---------|--------|---------|---------|---------|---------|---------|---------|
| Name | SIMTOEN | SIMTOF | SIMTOS5 | SIMTOS4 | SIMTOS3 | SIMTOS2 | SIMTOS1 | SIMTOS0 |
| 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 **SIMTOEN**: SIM I<sup>2</sup>C Time-out control  
 0: Disable  
 1: Enable

Bit 6 **SIMTOF**: SIM I<sup>2</sup>C Time-out flag  
 0: No time-out occurred  
 1: Time-out occurred

Bit 5~0 **SIMTOS5~SIMTOS0**: SIM I<sup>2</sup>C Time-out period selection  
 I<sup>2</sup>C Time-out clock source is  $f_{SUB}/32$

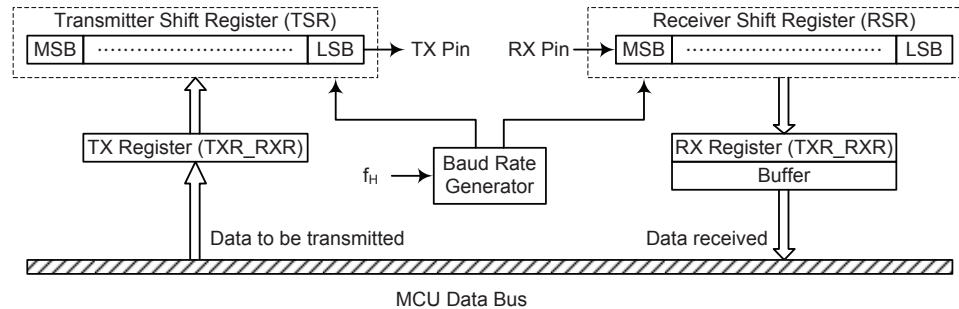
$$\text{I}^2\text{C Time-out period is equal to } (\text{SIMTOS}[5:0]+1) \times \frac{32}{f_{SUB}}$$

## UART Interface

The device contains an integrated full-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, asynchronous communication
- 8 or 9 bits character length
- Even, odd or no parity options
- One or two stop bits
- Baud rate generator with 8-bit prescaler
- Parity, framing, noise and overrun error detection
- Support for interrupt on address detect (last character bit=1)
- Separately enabled transmitter and receiver
- 2-byte Deep FIFO Receive Data Buffer
- Transmit and receive interrupts
- Interrupts can be initialized by the following conditions:
  - ♦ Transmitter Empty
  - ♦ Transmitter Idle
  - ♦ Receiver Full
  - ♦ Receiver Overrun
  - ♦ Address Mode Detect
  - ♦ RX pin wake-up



**UART Data Transfer Block Diagram**

## UART External Pin

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

## UART Data Transfer Scheme

The above diagram shows the overall data transfer structure arrangement for the UART interface. 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 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, although referred to in the text, and in application programs, as separate TXR\_RXR register, only exists as a single shared register in the Data Memory. This shared register known as the TXR\_RXR register is used for both data transmission and data reception.

## UART Status and Control Registers

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

### 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 pin.

| 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  | x   | x   | x   | x   | x   | x   | x   | x   |

"x": unknown

Bit 7~0      UART Transmit/Receive Data bits

### 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 and further explanations are 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 Parity mode (odd or even) is selected. The flag can also be cleared 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 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 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 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 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  
 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. When the contents of the shift register are transferred to the TXR\_RXR register, 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 is cleared 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 data available.
- Bit 1**     **TIDLE:** Transmission status  
           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 to "1" 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 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 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 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 | PRT | STOPS | 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 pins are in the state of high impedance.  
           1: Enable UART; TX and RX pins 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 pin as well as the TX pin will be in the state of high impedance. When the bit is equal to "1", the UART will be enabled and the TX and RX pins will function as defined by 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 will be cleared, while the TIDLE,

TXIF and RIDLE bits will be set. Other control bits in UCR1, UCR2 and BRG registers will remain unaffected. If the UART is active and the UARTEN bit is cleared, 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.
- Bit 5     **PREN**: Parity function enable control  
           0: Parity function is disabled  
           1: Parity function is enabled
- This bit is the parity function 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     **PRT**: Parity type selection bit  
           0: Even parity for parity generator  
           1: Odd parity for parity generator
- This bit is the parity type selection bit. When this bit is equal to 1, odd parity type will be selected. If the bit is equal to 0, then even parity type will be selected.
- Bit 3     **STOPS**: Number of stop bits selection  
           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. When this bit is equal to "1", two stop bits format are used. If the bit is equal to "0", then only one stop bit format is used.
- 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 equal to "0", there are no break characters and the TX pin operates normally. When the bit is equal to "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 9<sup>th</sup> 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 9<sup>th</sup> 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 baud rate speed, receiver wake-up function enable and the address detect function enable. Further explanation on each of the bits is given below.

| Bit  | 7    | 6    | 5    | 4     | 3    | 2   | 1    | 0    |
|------|------|------|------|-------|------|-----|------|------|
| Name | TXEN | RXEN | BRGH | 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 enable control  
 0: UART Transmitter is disabled  
 1: UART Transmitter is enabled

The TXEN bit 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 the state of high impedance. 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 the state of high impedance.

**Bit 6 RXEN:** UART Receiver enable control  
 0: UART Receiver is disabled  
 1: UART Receiver is enabled

The RXEN bit 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 receiver buffers will be reset. In this situation the RX pin will be in the state of high impedance. 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 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 pin will be in the state of high impedance.

**Bit 5 BRGH:** Baud Rate speed selection  
 0: Low speed baud rate  
 1: High speed baud rate

The bit named BRGH selects the high or low speed mode of the Baud Rate Generator. This bit, together with the value placed in the baud rate register, BRG, controls the baud rate of the UART. If the bit is equal to 0, the low speed mode is selected.

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

The bit named ADDEN is the address detection function enable control bit. When this bit is equal to 1, the address detection function is enabled. When it occurs, if the 8<sup>th</sup> bit, which corresponds to RX7 if BNO=0, or the 9<sup>th</sup> 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 8<sup>th</sup> or 9<sup>th</sup> bit depending on the value of the BNO bit. If the address bit known as the 8<sup>th</sup> or 9<sup>th</sup> bit of the received word is "0" with the address detection function being enabled, an interrupt will not be generated and the received data will be discarded.

- Bit 3     **WAKE:** RX pin falling edge wake-up function enable control  
           0: RX pin wake-up function is disabled  
           1: RX pin wake-up function is enabled  
 The bit enables or disables the receiver wake-up function. If this bit is equal to 1 and the device is in IDLE0 or SLEEP mode, a falling edge on the RX pin will wake up the device. If this bit is equal to 0 and the device is in IDLE or SLEEP mode, any edge transitions on the RX pin will not wake up the device.
- Bit 2     **RIE:** Receiver interrupt enable control  
           0: Receiver related interrupt is disabled  
           1: Receiver related interrupt is enabled  
 The bit enables or disables the receiver interrupt. If this bit is equal to 1 and when the receiver overrun flag OERR or received 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  
 The 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  
 The 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.

### 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 8-bit timer, the period of which is determined by two factors. The first of these is the value placed in the BRG register and the second is the value of the BRGH bit within the UCR2 control register. The BRGH bit decides, if the baud rate generator is to be used in a high speed mode or low speed mode, which in turn determines the formula that is used to calculate the baud rate. The value in the BRG register, N, which is used in the following baud rate calculation formula determines the division factor. Note that N is the decimal value placed in the BRG register and has a range of between 0 and 255.

| UCR2 BRGH Bit  | 0                           | 1                           |
|----------------|-----------------------------|-----------------------------|
| Baud Rate (BR) | $f_{\text{H}} / [64 (N+1)]$ | $f_{\text{H}} / [16 (N+1)]$ |

By programming the BRGH bit which allows selection of the related formula and programming the required value in the BRG register, the required baud rate can be setup. Note that because the actual baud rate is determined using a discrete value, N, placed in the BRG register, there will be an error associated between the actual and requested value. The following example shows how the BRG register value N and the error value can be calculated.



### BRG 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  | x   | x   | x   | x   | x   | x   | x   | x   |

"x": unknown

Bit 7~0 Baud Rate values

By programming the BRGH bit in the UCR2 register which allows selection of the related formula described above and programming the required value in the BRG register, the required baud rate can be setup.

### Calculating the Baud Rate and Error Values

For a clock frequency of 4MHz, and with BRGH set to 0 determine the BRG register value N, the actual baud rate and the error value for a desired baud rate of 4800.

From the above table the desired baud rate  $BR = f_{cl} / [64 (N+1)]$

Re-arranging this equation gives  $N = [f_{cl} / (BR \times 64)] - 1$

Giving a value for  $N = [4000000 / (4800 \times 64)] - 1 = 12.0208$

To obtain the closest value, a decimal value of 12 should be placed into the BRG register. This gives an actual or calculated baud rate value of  $BR = 4000000 / [64 \times (12 + 1)] = 4808$

Therefore the error is equal to  $(4808 - 4800) / 4800 = 0.16\%$

### 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 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 and stop bits, along with the parity, are setup by programming the corresponding BNO, PRT, PREN and STOPS bits in the UCR1 register. The baud rate used to transmit and receive data is setup using the internal 8-bit baud rate generator, while the data is transmitted and received LSB first. Although the transmitter and receiver of the UART 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 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 pins and these two pins will be used as an I/O or other pin-shared functional pin. 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 enable control, the error and status flags with bits TXEN, RXEN, TXBRK, RXIF, OERR, FERR, PERR and NF being cleared while bits TIDLE, TXIF and RIDLE will be set. The remaining control bits in the UCR1, UCR2 and BRG 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.

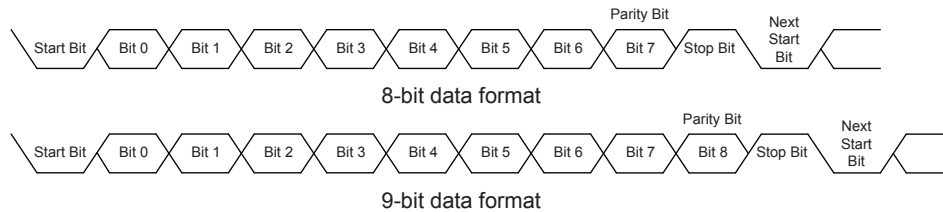
### Data, Parity and Stop Bit Selection

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 register. The BNO bit controls the number of data bits which can be set to either 8 or 9. The PRT bit controls the choice if odd or even parity. The PREN bit controls the parity on/off function. The STOPS bit decides whether one or two stop bits are to be used. 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 are only to be used for the Transmitter. There is only one stop bit for the Receiver.

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

**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 9<sup>th</sup> 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 will then return to the I/O or other pin-shared function.

### 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 LSB 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, PRT, PREN and STOPS bits to define the required word length, parity type and number of stop bits.
- Setup the BRG register to select the desired baud rate.
- Set the TXEN bit to ensure that the UART transmitter is enabled and 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, then the break characters will be sent on the next transmission. Break character transmission consists of a start bit, followed by  $13 \times N$  "0" 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 one or two stop bits. The automatic logic high 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 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 9<sup>th</sup> bit, which is the MSB, will be stored in the RX8 bit in the UCR1 register. At the receiver core lies the Receiver Shift Register more commonly known as the RSR. The data which is received on the RX external input 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 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 input pin is sampled three times by a majority detect circuit to determine the logic level that has been placed onto the RX 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 input pin to the shift register, with the least significant bit LSB first. The TXR\_RXR register is a two byte deep FIFO data buffer, where two bytes can be held in the FIFO while the third byte can continue to be received. Note that the application program must ensure that the data is read from TXR\_RXR before the third byte has been completely shifted in, otherwise the third byte will be discarded and an overrun error OERR will be subsequently indicated. The steps to initiate a data transfer can be summarized as follows:

- Make the correct selection of the BNO, PRT, PREN bits to define the required word length, parity type.
- Setup the BRG register to select the desired baud rate.
- Set the RXEN bit to ensure that the UART receiver is enabled and the RX 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. There will be at most one more character available before an overrun error occurs.
- When the contents of the shift register have been transferred to the TXR\_RXR register and if the RIE bit is set, then 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 plusing one stop bit. 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 and STOPS. 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 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.

### 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 two byte deep FIFO data buffer, where two bytes can be held in the FIFO register, while a third byte can continue to be received. Before the third 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.

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 an 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

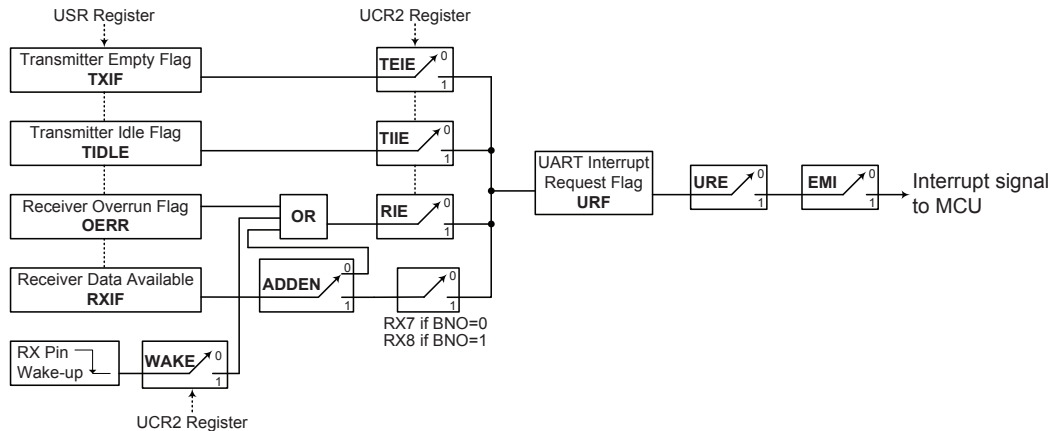
The read only parity error flag, PERR, in the USR register, is set if the parity of the received word is incorrect. This error flag is only applicable if the parity function is enabled, PREN=1, and if the parity type, odd or even, is selected. The read only PERR flag and the received data will be recorded 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 data available, receiver overrun, address detect and an RX pin wake-up. When any of these conditions are created, if its corresponding interrupt control is 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 pin wake-up, which is also a UART interrupt source, does not have an associated flag, but will generate a UART interrupt if the microcontroller is woken up from IDLE0 or SLEEP mode by a falling edge on the RX pin, if the WAKE and RIE bits in the UCR2 register are set. Note that in the event of an RX wake-up interrupt occurring, there will be a certain period of delay, commonly known as the System Start-up Time, for the oscillator to restart and stabilize before the system resumes normal operation.

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 function enable control bit, ADDEN, in the UCR2 register, enables this special function. If this bit is set to 1, 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 equal to 1, then when the data is available, an interrupt will only be generated, if the highest received bit has a high value. Note that the related interrupt enable control bit and the EMI bit of the microcontroller must also be enabled for correct interrupt generation. The highest address bit is the 9<sup>th</sup> bit if the bit BNO=1 or the 8<sup>th</sup> bit if the bit BNO=0. If the highest 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 equal to 0, then a Receive Data Available interrupt will be generated each time the RXIF flag is set, irrespective of the data last but status. The address detection and parity functions are mutually exclusive functions. Therefore, if the address detect function is enabled, then to ensure correct operation, the parity function should be disabled by resetting the parity function enable bit PREN to zero.

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

**ADDEN Bit Function**

### UART Power Down and Wake-up

When the MCU system clock is switched off, the UART will cease to function. If the MCU executes the "HALT" instruction and switches off the system clock 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 executes the "HALT" instruction and switches off the system clock 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, transmit and receive registers, as well as the BRG register 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 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 before the MCU enters the IDLE0 or SLEEP Mode, then a falling edge on the RX pin will wake up the MCU from the IDLE0 or SLEEP Mode. 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 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.

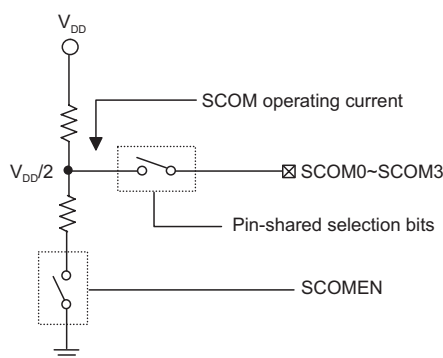


## SCOM Function for LCD

The device has the capability of driving external LCD panels. The common pins for LCD driving, SCOM0~SCOM3, are pin shared with the I/O pins. The LCD signals (COM and SEG) are generated using the application program.

### LCD Operation

An external LCD panel can be driven using this device by configuring the I/O pins as common pins and configuring the I/O pins as segment pins. The LCD driver function is controlled using the SCOMC register which in addition to controlling the overall on/off function also controls the bias voltage setup function. This enables the LCD COM driver to generate the necessary  $V_{DD}/2$  voltage levels for LCD 1/2 bias operation.



LCD COM Bias

The SCOMEN bit in the SCOMC register is the overall master control for the LCD driver. The LCD SCOMn pin is selected to be used for LCD driving by the corresponding pin-shared function selection bits. Note that the Port Control register does not need to first setup the pins as outputs to enable the LCD driver operation.

### LCD Bias Current Control

The LCD COM driver enables a range of selections to be provided to suit the requirement of the LCD panel which is being used. The bias resistor choice is implemented using the ISEL1 and ISEL0 bits in the SCOMC register.

#### SCOMC Register

| Bit  | 7 | 6     | 5     | 4      | 3 | 2 | 1 | 0 |
|------|---|-------|-------|--------|---|---|---|---|
| Name | — | ISEL1 | ISEL0 | SCOMEN | — | — | — | — |
| R/W  | — | R/W   | R/W   | R/W    | — | — | — | — |
| POR  | — | 0     | 0     | 0      | — | — | — | — |

Bit 7 Unimplemented, read as "0"

Bit 6~5 **ISEL1~ISEL0**: Select resistor for R type LCD bias current  
 00:  $2 \times 100\text{k}\Omega$  (1/2 Bias),  $I_{BIAS}=25\mu\text{A}@(\text{V}_{DD}=5\text{V})$   
 01:  $2 \times 50\text{k}\Omega$  (1/2 Bias),  $I_{BIAS}=50\mu\text{A}@(\text{V}_{DD}=5\text{V})$   
 10:  $2 \times 25\text{k}\Omega$  (1/2 Bias),  $I_{BIAS}=100\mu\text{A}@(\text{V}_{DD}=5\text{V})$   
 11:  $2 \times 12.5\text{k}\Omega$  (1/2 Bias),  $I_{BIAS}=200\mu\text{A}@(\text{V}_{DD}=5\text{V})$

Bit 4 **SCOMEN**: LCD control bit  
 0: Off  
 1: On

When SCOMEN is set, it will turn on the DC path of resistor to generate  $1/2 V_{DD}$  bias voltage.  
 Bit 3~0 Unimplemented, read as "0"

## 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 interrupts functions. The external interrupt is generated by the action of the external INTn pin, while the internal interrupts are generated by various internal functions such as TMs, Comparator, Time Bases, LVD, EEPROM, SIM, UART 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, as shown in the accompanying table. The first is the INTC0~INTC2 registers which setup the primary interrupts, the second is the MFI0~MFI2 registers which setup the Multi-function interrupts.

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        | —            | —        |
| INTn Pin       | INTnE      | INTnF        | n=0 or 1 |
| Comparator     | CPE        | CPF          | —        |
| Multi-function | MFnE       | MFnF         | n=0~2    |
| A/D Converter  | ADE        | ADF          | —        |
| Time Base      | TBnE       | TBnF         | n=0 or 1 |
| SIM            | SIME       | SIMF         | —        |
| UART           | URE        | URF          | —        |
| CTM            | CTMPE      | CTMPF        | —        |
|                | CTMAE      | CTMAF        |          |
| STM            | STMPE      | STMPF        | —        |
|                | STMAE      | STMAF        |          |
| PTM            | PTMPE      | PTMPF        | —        |
|                | PTMAE      | PTMAF        |          |
| LVD            | LVE        | LVF          | —        |
| EEPROM         | DEE        | DEF          | —        |

**Interrupt Register Bit Naming Conventions**

| Register Name | Bit   |       |       |       |        |        |        |        |
|---------------|-------|-------|-------|-------|--------|--------|--------|--------|
|               | 7     | 6     | 5     | 4     | 3      | 2      | 1      | 0      |
| INTEG         | —     | —     | —     | —     | INT1S1 | INT1S0 | INT0S1 | INT0S0 |
| INTC0         | —     | MF0F  | CPF   | INT0F | MF0E   | CPE    | INT0E  | EMI    |
| INTC1         | TB0F  | ADF   | MF2F  | MF1F  | TB0E   | ADE    | MF2E   | MF1E   |
| INTC2         | URF   | SIMF  | INT1F | TB1F  | URE    | SIME   | INT1E  | TB1E   |
| MF10          | —     | —     | STMAF | STMPF | —      | —      | STMAE  | STMPE  |
| MF11          | CTMAF | CTMPF | PTMAF | PTMPF | CTMAE  | CTMPE  | PTMAE  | PTMPE  |
| MF12          | —     | —     | DEF   | LVF   | —      | —      | DEE    | LVE    |

Interrupt Registers List

### INTEG Register

| Bit  | 7 | 6 | 5 | 4 | 3      | 2      | 1      | 0      |
|------|---|---|---|---|--------|--------|--------|--------|
| Name | — | — | — | — | INT1S1 | INT1S0 | INT0S1 | INT0S0 |
| R/W  | — | — | — | — | R/W    | R/W    | R/W    | R/W    |
| POR  | — | — | — | — | 0      | 0      | 0      | 0      |

Bit 7~4 Unimplemented, read as "0"

Bit 3~2 **INT1S1~INT1S0**: Interrupt edge control for INT1 pin  
 00: Disable  
 01: Rising edge  
 10: Falling edge  
 11: Both rising and falling edges

Bit 1~0 **INT0S1~INT0S0**: Interrupt edge control for INT0 pin  
 00: Disable  
 01: Rising edge  
 10: Falling edge  
 11: Both rising and falling edges

### INTC0 Register

| Bit  | 7 | 6    | 5   | 4     | 3    | 2   | 1     | 0   |
|------|---|------|-----|-------|------|-----|-------|-----|
| Name | — | MF0F | CPF | INT0F | MF0E | CPE | INT0E | 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 **MF0F**: Multi-function 0 interrupt request Flag  
 0: No request  
 1: Interrupt request

Bit 5 **CPF**: Comparator interrupt request flag  
 0: No request  
 1: Interrupt request

Bit 4 **INT0F**: INT0 interrupt request Flag  
 0: No request  
 1: Interrupt request

Bit 3 **MF0E**: Multi-function 0 interrupt control  
 0: Disable  
 1: Enable

Bit 2 **CPE**: Comparator interrupt control  
 0: Disable  
 1: Enable

- Bit 1     **INT0E**: INT0 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 | TB0F | ADF | MF2F | MF1F | TB0E | ADE | MF2E | MF1E |
| 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     **ADF**: A/D Converter interrupt request flag  
          0: No request  
          1: Interrupt request
- Bit 5     **MF2F**: Multi-function 2 interrupt request flag  
          0: No request  
          1: Interrupt request
- Bit 4     **MF1F**: Multi-function 1 interrupt request flag  
          0: No request  
          1: Interrupt request
- Bit 3     **TB0E**: Time Base 0 interrupt control  
          0: Disable  
          1: Enable
- Bit 2     **ADE**: A/D Converter interrupt control  
          0: Disable  
          1: Enable
- Bit 1     **MF2E**: Multi-function 2 interrupt control  
          0: Disable  
          1: Enable
- Bit 0     **MF1E**: Multi-function 1 interrupt control  
          0: Disable  
          1: Enable

**INTC2 Register**

| Bit  | 7   | 6    | 5     | 4    | 3   | 2    | 1     | 0    |
|------|-----|------|-------|------|-----|------|-------|------|
| Name | URF | SIMF | INT1F | TB1F | URE | SIME | INT1E | TB1E |
| 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     **URF**: UART interrupt request flag  
          0: No request  
          1: Interrupt request
- Bit 6     **SIMF**: SIM interrupt request flag  
          0: No request  
          1: Interrupt request
- Bit 5     **INT1F**: INT1 pin interrupt request flag  
          0: No request  
          1: Interrupt request

- Bit 4      **TB1F**: Time Base 1 interrupt request flag  
             0: No request  
             1: Interrupt request
- Bit 3      **URE**: UART interrupt control  
             0: Disable  
             1: Enable
- Bit 2      **SIME**: SIM interrupt control  
             0: Disable  
             1: Enable
- Bit 1      **INTIE**: INT1 pin interrupt control  
             0: Disable  
             1: Enable
- Bit 0      **TB1E**: Time Base 1 interrupt control  
             0: Disable  
             1: Enable

**MF10 Register**

| Bit  | 7 | 6 | 5     | 4     | 3 | 2 | 1     | 0     |
|------|---|---|-------|-------|---|---|-------|-------|
| Name | — | — | STMAF | STMPF | — | — | STMAE | STMPE |
| R/W  | — | — | R/W   | R/W   | — | — | R/W   | R/W   |
| POR  | — | — | 0     | 0     | — | — | 0     | 0     |

- Bit 7~6      Unimplemented, read as "0"
- 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 3~2      Unimplemented, read as "0"
- 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

**MF11 Register**

| Bit  | 7     | 6     | 5     | 4     | 3     | 2     | 1     | 0     |
|------|-------|-------|-------|-------|-------|-------|-------|-------|
| Name | CTMAF | CTMPF | PTMAF | PTMPF | CTMAE | CTMPE | PTMAE | PTMPE |
| 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      **PTMAF**: PTM Comparator A match interrupt request flag  
             0: No request  
             1: Interrupt request

- Bit 4      **PTMPF**: PTM Comparator P match 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      **PTMAE**: PTM Comparator A match interrupt control  
0: Disable  
1: Enable
- Bit 0      **PTMPE**: PTM Comparator P match interrupt control  
0: Disable  
1: Enable

**MF12 Register**

| Bit  | 7 | 6 | 5   | 4   | 3 | 2 | 1   | 0   |
|------|---|---|-----|-----|---|---|-----|-----|
| Name | — | — | DEF | LVF | — | — | DEE | LVE |
| R/W  | — | — | R/W | R/W | — | — | R/W | R/W |
| POR  | — | — | 0   | 0   | — | — | 0   | 0   |

- Bit 7~6      Unimplemented, read as "0"
- Bit 5      **DEF**: Data EEPROM interrupt request flag  
0: No request  
1: Interrupt request
- Bit 4      **LVF**: LVD interrupt request flag  
0: No request  
1: Interrupt request
- Bit 3~2      Unimplemented, read as "0"
- Bit 1      **DEE**: Data EEPROM interrupt control  
0: Disable  
1: Enable
- Bit 0      **LVE**: LVD interrupt control  
0: Disable  
1: Enable

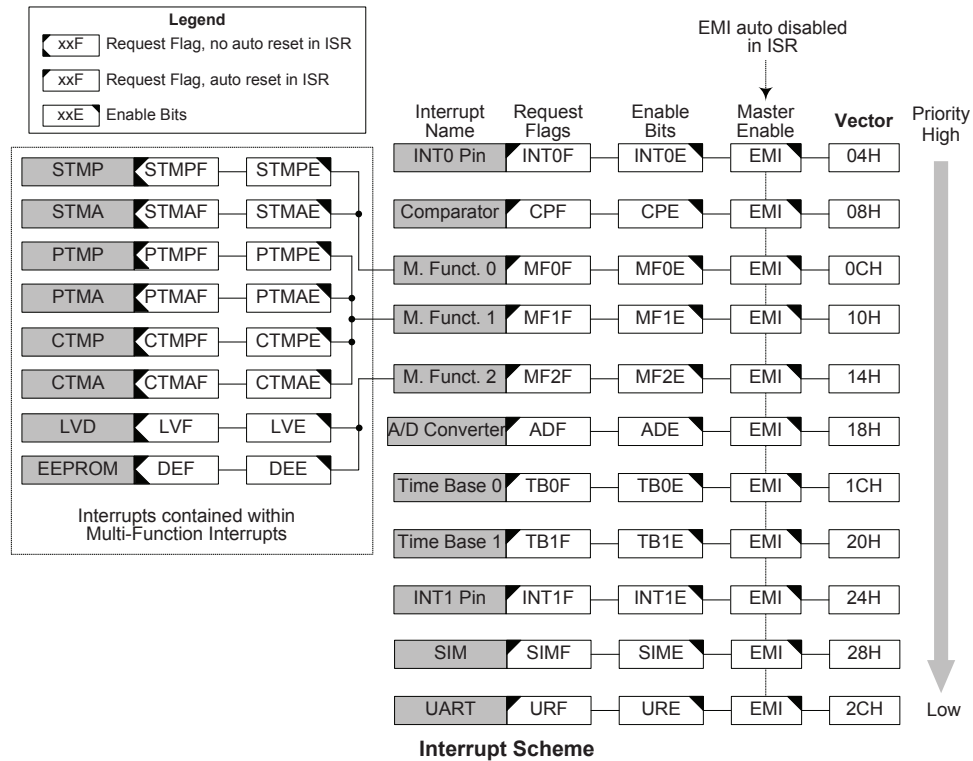
## **Interrupt Operation**

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. Some interrupt sources have their own individual vector while others share the same multi-function interrupt 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 is in SLEEP or IDLE Mode.



### External Interrupt

The external interrupts are controlled by signal transitions on the pins INT0~INT1. An external interrupt request will take place when the external interrupt request flags, INT0F~INT1F, are set, which will occur when a transition, whose type is chosen by the edge select bits, appears on the external interrupt pins. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and respective external interrupt enable bit, INT0E~INT1E, 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 pins are pin-shared with I/O pins, they can only be configured as external interrupt pins if their 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 flags, INT0F~INT1F, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts. Note that any pull-high resistor selections on the external interrupt pins 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.



### **Comparator Interrupt**

The comparator interrupt is controlled by the internal comparator. A comparator interrupt request will take place when the comparator interrupt request flag, CPF, is set, a situation that will occur when the comparator output bit changes state. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and comparator interrupt enable bit, CPE, must first be set. When the interrupt is enabled, the stack is not full and the comparator inputs generate a comparator output transition, a subroutine call to the comparator interrupt vector, will take place. When the interrupt is serviced, the comparator interrupt request flag will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts.

### **Multi-function Interrupt**

Within the device there are up to three Multi-function interrupts. Unlike the other independent interrupts, these interrupts have no independent source, but rather are formed from other existing interrupt sources, namely the TM interrupts, LVD interrupt, and EEPROM write operation interrupt.

A Multi-function interrupt request will take place when any of the Multi-function interrupt request flags MFnF are set. The Multi-function interrupt flags will be set when any of their included functions generate an interrupt request flag. To allow the program to branch to its respective interrupt vector address, when the Multi-function interrupt is enabled and the stack is not full, and either one of the interrupts contained within each of Multi-function interrupt occurs, a subroutine call to one of the Multi-function interrupt vectors will take place. When the interrupt is serviced, the related Multi-Function request flag will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts.

However, it must be noted that, although the Multi-function Interrupt request flags will be automatically reset when the interrupt is serviced, the request flags from the original source of the Multi-function interrupts will not be automatically reset and must be manually reset by the application program.

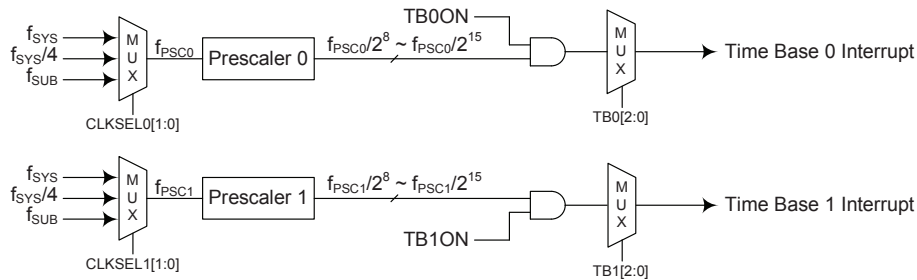
### **A/D Converter Interrupt**

The device contains an A/D converter which has its own independent 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 respective interrupt vector address, the global interrupt enable bit, EMI, and A/D converter 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.

### Time Base Interrupt

The function of the Time Base Interrupt is to provide regular time signal in the form of an internal interrupt. It is controlled by the overflow signal from its internal timer. When this happens its interrupt request flag, TBnF, will be set. To allow the program to branch to its respective interrupt vector addresses, the global interrupt enable bit, EMI and Time Base enable bit, TBnE, must first be set. When the interrupt is enabled, the stack is not full and the Time Base overflows, a subroutine call to its respective vector location will take place. When the interrupt is serviced, the interrupt request flag, TBnF, will be automatically reset and the EMI bit will be cleared to disable other interrupts.

The purpose of the Time Base Interrupt 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 and TB1C registers to obtain longer interrupt periods whose value ranges. The clock source 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**

#### PSC0R Register

| Bit  | 7 | 6 | 5 | 4 | 3 | 2 | 1        | 0        |
|------|---|---|---|---|---|---|----------|----------|
| Name | — | — | — | — | — | — | CLKSEL01 | CLKSEL00 |
| R/W  | — | — | — | — | — | — | R/W      | R/W      |
| POR  | — | — | — | — | — | — | 0        | 0        |

Bit 7~2 Unimplemented, read as "0"

Bit 1~0 **CLKSEL01~CLKSEL00:** Prescaler 0 clock source selection

00:  $f_{SYS}$

01:  $f_{SYS}/4$

1x:  $f_{SUB}$

#### PSC1R Register

| Bit  | 7 | 6 | 5 | 4 | 3 | 2 | 1        | 0        |
|------|---|---|---|---|---|---|----------|----------|
| Name | — | — | — | — | — | — | CLKSEL11 | CLKSEL10 |
| R/W  | — | — | — | — | — | — | R/W      | R/W      |
| POR  | — | — | — | — | — | — | 0        | 0        |

Bit 7~2 Unimplemented, read as "0"

Bit 1~0 **CLKSEL11~CLKSEL10:** Prescaler 1 clock source selection

00:  $f_{SYS}$

01:  $f_{SYS}/4$

1x:  $f_{SUB}$

### TB0C Register

| Bit  | 7     | 6 | 5 | 4 | 3 | 2    | 1    | 0    |
|------|-------|---|---|---|---|------|------|------|
| Name | TB0ON | — | — | — | — | TB02 | TB01 | TB00 |
| R/W  | R/W   | — | — | — | — | R/W  | R/W  | R/W  |
| POR  | 0     | — | — | — | — | 0    | 0    | 0    |

- Bit 7      **TB0ON**: Time Base 0 Enable Control  
0: Disable  
1: Enable
- Bit 6~3    Unimplemented, read as "0"
- Bit 2~0    **TB02~TB00**: Time Base 0 time-out period selection  
000:  $2^8/f_{PSC0}$   
001:  $2^9/f_{PSC0}$   
010:  $2^{10}/f_{PSC0}$   
011:  $2^{11}/f_{PSC0}$   
100:  $2^{12}/f_{PSC0}$   
101:  $2^{13}/f_{PSC0}$   
110:  $2^{14}/f_{PSC0}$   
111:  $2^{15}/f_{PSC0}$

### TB1C Register

| Bit  | 7     | 6 | 5 | 4 | 3 | 2    | 1    | 0    |
|------|-------|---|---|---|---|------|------|------|
| Name | TB1ON | — | — | — | — | TB12 | TB11 | TB10 |
| R/W  | R/W   | — | — | — | — | R/W  | R/W  | R/W  |
| POR  | 0     | — | — | — | — | 0    | 0    | 0    |

- Bit 7      **TB1ON**: Time Base 1 Enable Control  
0: Disable  
1: Enable
- Bit 6~3    Unimplemented, read as "0"
- Bit 2~0    **TB12~TB10**: Time Base 1 time-out period selection  
000:  $2^8/f_{PSC1}$   
001:  $2^9/f_{PSC1}$   
010:  $2^{10}/f_{PSC1}$   
011:  $2^{11}/f_{PSC1}$   
100:  $2^{12}/f_{PSC1}$   
101:  $2^{13}/f_{PSC1}$   
110:  $2^{14}/f_{PSC1}$   
111:  $2^{15}/f_{PSC1}$

### Serial Interface Module Interrupt

A SIM Interrupt request will take place when the SIM Interrupt request flag, SIMF, is set, which occurs when a byte of data has been received or transmitted by the SIM interface, an I<sup>2</sup>C slave address match or I<sup>2</sup>C bus time-out occurrence. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and the Serial Interface Interrupt enable bit, SIME, must first be set. When the interrupt is enabled, the stack is not full and any of the above described situations occurs, will take place. When the Serial Interface Module Interrupt is serviced, the interrupt request flag, SIMF, will be automatically reset and the EMI bit will be cleared to disable other interrupts.

### **UART Transfer Interrupt**

The UART Transfer Interrupt is controlled by several UART transfer conditions. When one of these conditions occurs, an interrupt pulse will be generated to get the attention of the microcontroller. These conditions are a transmitter data register empty, transmitter idle, receiver data available, receiver overrun, address detect and an RX pin wake-up. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and UART Interrupt enable bit, URE, must first be set. When the interrupt is enabled, the stack is not full and any of the conditions described above occurs, a subroutine call to the UART Interrupt vector, will take place. When the interrupt is serviced, the UART Interrupt flag, URF, will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts.

### **EEPROM Write Interrupt**

The EEPROM Write Interrupt is contained within the Multi-function Interrupt. An EEPROM Write Interrupt request will take place when the EEPROM Write Interrupt request flag, DEF, is set, which occurs when an EEPROM Write cycle ends. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, EEPROM Write Interrupt enable bit, DEE, and associated Multi-function interrupt enable bit 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 respective Multi-function Interrupt vector will take place. When the EEPROM Write Interrupt is serviced, the EMI bit will be automatically cleared to disable other interrupts. However, only the Multi-function interrupt request flag will be automatically cleared. As the DEF flag will not be automatically cleared, it has to be cleared by the application program.

### **LVD Interrupt**

The Low Voltage Detector Interrupt is contained within the Multi-function Interrupt. A LVD Interrupt request will take place when the LVD Interrupt request flag, LVF, is set, which occurs when the Low Voltage Detector function detects a low power supply voltage. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, Low Voltage Interrupt enable bit, LVE, and associated Multi-function interrupt enable bit, must first be set. When the interrupt is enabled, the stack is not full and a low voltage condition occurs, a subroutine call to the Multi-function Interrupt vector, will take place. When the Low Voltage Interrupt is serviced, the EMI bit will be automatically cleared to disable other interrupts, however only the Multi-function interrupt request flag will be also automatically cleared. As the LVF flag will not be automatically cleared, it has to be cleared by the application program.

### **TM Interrupts**

The Compact, Standard and Periodic TMs have two interrupts, one comes from the comparator A match situation and the other comes from the comparator P match situation. All of the TM interrupts are contained within the Multi-function Interrupts. For all of the TM types there are two interrupt request flags and two enable control bits. 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 its respective interrupt vector address, the global interrupt enable bit, EMI, respective TM Interrupt enable bit, and relevant Multi-function Interrupt enable bit, MFnE, 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 relevant Multi-function Interrupt vector locations, will take place. When the TM interrupt is serviced, the EMI bit will be automatically cleared to disable other interrupts. However, only the related MFnF flag will be automatically cleared. As the TM interrupt request flags will not be automatically cleared, they have to be cleared by the application program.

## **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 pins, a low power supply voltage or comparator input change 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 enters 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.

Where a certain interrupt is contained within a Multi-function interrupt, then when the interrupt service routine is executed, as only the Multi-function interrupt request flags, MF<sub>n</sub>F, will be automatically cleared, the individual request flag for the function needs to be 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.

## Low Voltage Detector – LVD

Each device has a Low Voltage Detector function, also known as LVD. This enabled the device to monitor the power supply voltage,  $V_{DD}$ , and provide a warning signal should it fall below a certain level. This function may be especially useful in battery applications where the supply voltage will gradually reduce as the battery ages, as it allows an early warning battery low signal to be generated. The Low Voltage Detector also has the capability of generating an interrupt signal.

### LVD Register

The Low Voltage Detector function is controlled using a single register with the name LVDC. Three bits in this register, VLVD2~VLVD0, are used to select one of eight fixed voltages below which a low voltage condition will be determined. A low voltage condition is indicated when the LVDO bit is set. If the LVDO bit is low, this indicates that the  $V_{DD}$  voltage is above the preset low voltage value. The LVDEN bit is used to control the overall on/off function of the low voltage detector. Setting the bit high will enable the low voltage detector. Clearing the bit to zero will switch off the internal low voltage detector circuits. As the low voltage detector will consume a certain amount of power, it may be desirable to switch off the circuit when not in use, an important consideration in power sensitive battery powered applications.

### LVDC Register

| Bit  | 7 | 6 | 5    | 4     | 3   | 2     | 1     | 0     |
|------|---|---|------|-------|-----|-------|-------|-------|
| Name | — | — | LVDO | LVDEN | D3  | VLVD2 | VLVD1 | VLVD0 |
| R/W  | — | — | R    | 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 **LVDO**: LVD Output Flag  
0: No Low Voltage Detect  
1: Low Voltage Detect

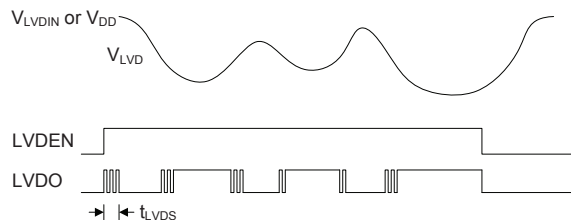
Bit 4 **LVDEN**: Low Voltage Detector Control  
0: Disable  
1: Enable

Bit 3 **D3**: it can only be fixed at zero. When the bit is written to 1, it will cause additional power consumption.

Bit 2~0 **VLVD2~VLVD0**: Select LVD Voltage  
000: 1.8V  
001: 2.0V  
010: 2.4V  
011: 2.7V  
100: 3.0V  
101: 3.3V  
110: 3.6V  
111: 4.0V

## LVD Operation

The Low Voltage Detector function operates by comparing the power supply voltage,  $V_{DD}$ , with a pre-specified voltage level stored in the LVDC register. This has a range of between 1.8V and 4.0V. When the power supply voltage,  $V_{DD}$ , falls below this pre-determined value, the LVDO bit will be set high indicating a low power supply voltage condition. The Low Voltage Detector function is supplied by a reference voltage which will be automatically enabled. When the device is SLEEP mode the Low Voltage Detector will disable, even if the LVDEN bit is high. After enabling the Low Voltage Detector, a time delay  $t_{LVDS}$  should be allowed for the circuitry to stabilise before reading the LVDO bit. Note also that as the  $V_{DD}$  voltage may rise and fall rather slowly, at the voltage nears that of  $V_{LVD}$ , there may be multiple bit LVDO transitions.



**LVD Operation**

The Low Voltage Detector also has its own interrupt which is contained within one of the Multi-function interrupts, providing an alternative means of low voltage detection, in addition to polling the LVDO bit. The interrupt will only be generated after a delay of  $t_{LVD}$  after the LVDO bit has been set high by a low voltage condition. In this case, the LVF interrupt request flag will be set, causing an interrupt to be generated if  $V_{DD}$  falls below the preset LVD voltage. This will cause the device to wake-up from the SLEEP or IDLE Mode, however if the Low Voltage Detector wake up function is not required then the LVF flag should be first set high before the device enters the SLEEP or IDLE Mode.

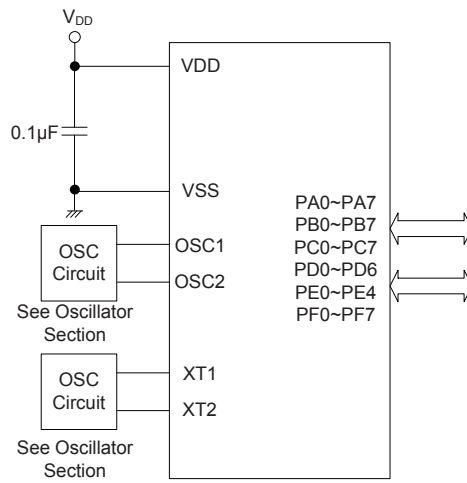
## Configuration Option

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. As these options are programmed into the device using the hardware programming tools, once they are selected they cannot be changed later using the application program. All options must be defined for proper system function, the details of which are shown in the table.

| No.                       | Options   |
|---------------------------|---|
| <b>Oscillator Options</b> |   |
| 1                         | HIRC frequency selection(for writer trim & EV trim code use only)<br>1. 4MHz<br>2. 8MHz<br>3. 12MHz |
| 2                         | HXT mode selection (for Low Voltage mode):<br>1. By S/W<br>2. HXT > 10MHz or ≤ 8MHz@1.8V            |



## 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 instructions related to the data memory access in the following table can be used when the desired data memory is located in Data Memory sector 0.

### 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, SC     |
| ADDM A,[m]                       | Add ACC to Data Memory  | 1 <sup>Note</sup> | Z, C, AC, OV, SC     |
| ADD A,x                          | Add immediate data to ACC                                       | 1                 | Z, C, AC, OV, SC     |
| ADC A,[m]                        | Add Data Memory to ACC with Carry                               | 1                 | Z, C, AC, OV, SC     |
| ADCM A,[m]                       | Add ACC to Data memory with Carry                               | 1 <sup>Note</sup> | Z, C, AC, OV, SC     |
| SUB A,x                          | Subtract immediate data from the ACC                            | 1                 | Z, C, AC, OV, SC, CZ |
| SUB A,[m]                        | Subtract Data Memory from ACC                                   | 1                 | Z, C, AC, OV, SC, CZ |
| SUBM A,[m]                       | Subtract Data Memory from ACC with result in Data Memory        | 1 <sup>Note</sup> | Z, C, AC, OV, SC, CZ |
| SBC A,x                          | Subtract immediate data from ACC with Carry                     | 1                 | Z, C, AC, OV, SC, CZ |
| SBC A,[m]                        | Subtract Data Memory from ACC with Carry                        | 1                 | Z, C, AC, OV, SC, CZ |
| SBCM A,[m]                       | Subtract Data Memory from ACC with Carry, result in Data Memory | 1 <sup>Note</sup> | Z, C, AC, OV, SC, CZ |
| 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]                     | Skip if Data Memory is not 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) 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          |
| ITABRD [m]                  | Increment table pointer TBLP first and Read table to TBLH and Data Memory             | 2 <sup>Note</sup> | None          |
| ITABRDL [m]                 | Increment table pointer TBLP first and 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 up to three 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.

3. For the "CLR WDT" instruction the TO and PDF flags may be affected by the execution status. The TO and PDF flags are cleared after the "CLR WDT" instructions is executed. Otherwise the TO and PDF flags remain unchanged.

### Extended Instruction Set

The extended instructions are used to support the full range address access for the data memory. When the accessed data memory is located in any data memory sections except sector 0, the extended instruction can be used to access the data memory instead of using the indirect addressing access to improve the CPU firmware performance.

| Mnemonic                         | Description   | Cycles            | Flag Affected        |
|----------------------------------|---|-------------------|----------------------|
| <b>Arithmetic</b>                |   |                   |                      |
| LADD A,[m]                       | Add Data Memory to ACC  | 2                 | Z, C, AC, OV, SC     |
| LADDM A,[m]                      | Add ACC to Data Memory  | 2 <sup>Note</sup> | Z, C, AC, OV, SC     |
| LADC A,[m]                       | Add Data Memory to ACC with Carry                               | 2                 | Z, C, AC, OV, SC     |
| LADCM A,[m]                      | Add ACC to Data memory with Carry                               | 2 <sup>Note</sup> | Z, C, AC, OV, SC     |
| LSUB A,[m]                       | Subtract Data Memory from ACC                                   | 2                 | Z, C, AC, OV, SC, CZ |
| LSUBM A,[m]                      | Subtract Data Memory from ACC with result in Data Memory        | 2 <sup>Note</sup> | Z, C, AC, OV, SC, CZ |
| LSBC A,[m]                       | Subtract Data Memory from ACC with Carry                        | 2                 | Z, C, AC, OV, SC, CZ |
| LSBCM A,[m]                      | Subtract Data Memory from ACC with Carry, result in Data Memory | 2 <sup>Note</sup> | Z, C, AC, OV, SC, CZ |
| LDAA [m]                         | Decimal adjust ACC for Addition with result in Data Memory      | 2 <sup>Note</sup> | C                    |
| <b>Logic Operation</b>           |   |                   |                      |
| LAND A,[m]                       | Logical AND Data Memory to ACC                                  | 2                 | Z                    |
| LOR A,[m]                        | Logical OR Data Memory to ACC                                   | 2                 | Z                    |
| LXOR A,[m]                       | Logical XOR Data Memory to ACC                                  | 2                 | Z                    |
| LANDM A,[m]                      | Logical AND ACC to Data Memory                                  | 2 <sup>Note</sup> | Z                    |
| LORM A,[m]                       | Logical OR ACC to Data Memory                                   | 2 <sup>Note</sup> | Z                    |
| LXORM A,[m]                      | Logical XOR ACC to Data Memory                                  | 2 <sup>Note</sup> | Z                    |
| LCPL [m]                         | Complement Data Memory  | 2 <sup>Note</sup> | Z                    |
| LCPLA [m]                        | Complement Data Memory with result in ACC                       | 2                 | Z                    |
| <b>Increment &amp; Decrement</b> |   |                   |                      |
| LINCA [m]                        | Increment Data Memory with result in ACC                        | 2                 | Z                    |
| LINC [m]                         | Increment Data Memory   | 2 <sup>Note</sup> | Z                    |
| LDECA [m]                        | Decrement Data Memory with result in ACC                        | 2                 | Z                    |
| LDEC [m]                         | Decrement Data Memory   | 2 <sup>Note</sup> | Z                    |
| <b>Rotate</b>                    |   |                   |                      |
| LRRRA [m]                        | Rotate Data Memory right with result in ACC                     | 2                 | None                 |
| LRR [m]                          | Rotate Data Memory right  | 2 <sup>Note</sup> | None                 |
| LRRCA [m]                        | Rotate Data Memory right through Carry with result in ACC       | 2                 | C                    |
| LRRC [m]                         | Rotate Data Memory right through Carry                          | 2 <sup>Note</sup> | C                    |
| LRLA [m]                         | Rotate Data Memory left with result in ACC                      | 2                 | None                 |
| LRL [m]                          | Rotate Data Memory left   | 2 <sup>Note</sup> | None                 |
| LRLCA [m]                        | Rotate Data Memory left through Carry with result in ACC        | 2                 | C                    |
| LRLC [m]                         | Rotate Data Memory left through Carry                           | 2 <sup>Note</sup> | C                    |
| <b>Data Move</b>                 |   |                   |                      |
| LMOV A,[m]                       | Move Data Memory to ACC   | 2                 | None                 |
| LMOV [m],A                       | Move ACC to Data Memory   | 2 <sup>Note</sup> | None                 |
| <b>Bit Operation</b>             |   |                   |                      |
| LCLR [m].i                       | Clear bit of Data Memory  | 2 <sup>Note</sup> | None                 |
| LSET [m].i                       | Set bit of Data Memory  | 2 <sup>Note</sup> | None                 |

| Mnemonic             | Description   | Cycles            | Flag Affected |
|----------------------|---|-------------------|---------------|
| <b>Branch</b>        |   |                   |               |
| LSZ [m]              | Skip if Data Memory is zero   | 2 <sup>Note</sup> | None          |
| LSZA [m]             | Skip if Data Memory is zero with data movement to ACC                                 | 2 <sup>Note</sup> | None          |
| LSNZ [m]             | Skip if Data Memory is not zero   | 2 <sup>Note</sup> | None          |
| LSZ [m].i            | Skip if bit i of Data Memory is zero  | 2 <sup>Note</sup> | None          |
| LSNZ [m].i           | Skip if bit i of Data Memory is not zero  | 2 <sup>Note</sup> | None          |
| LSIZ [m]             | Skip if increment Data Memory is zero   | 2 <sup>Note</sup> | None          |
| LSDZ [m]             | Skip if decrement Data Memory is zero   | 2 <sup>Note</sup> | None          |
| LSIZA [m]            | Skip if increment Data Memory is zero with result in ACC                              | 2 <sup>Note</sup> | None          |
| LSDZA [m]            | Skip if decrement Data Memory is zero with result in ACC                              | 2 <sup>Note</sup> | None          |
| <b>Table Read</b>    |   |                   |               |
| LTABRD [m]           | Read table to TBLH and Data Memory  | 3 <sup>Note</sup> | None          |
| LTABRDL [m]          | Read table (last page) to TBLH and Data Memory  | 3 <sup>Note</sup> | None          |
| LITABRD [m]          | Increment table pointer TBLP first and Read table to TBLH and Data Memory             | 3 <sup>Note</sup> | None          |
| LITABRDL [m]         | Increment table pointer TBLP first and Read table (last page) to TBLH and Data Memory | 3 <sup>Note</sup> | None          |
| <b>Miscellaneous</b> |   |                   |               |
| LCLR [m]             | Clear Data Memory   | 2 <sup>Note</sup> | None          |
| LSET [m]             | Set Data Memory   | 2 <sup>Note</sup> | None          |
| LSWAP [m]            | Swap nibbles of Data Memory   | 2 <sup>Note</sup> | None          |
| LSWAPA [m]           | Swap nibbles of Data Memory with result in ACC  | 2                 | None          |

Note: 1. For these extended skip instructions, if the result of the comparison involves a skip then up to four cycles are required, if no skip takes place two cycles is required.

- Any extended instruction which changes the contents of the PCL register will also require three 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, SC  |
| <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, SC  |
| <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, SC  |
| <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, SC  |
| <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, SC  |
| <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  |
| <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   |
| <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   |
| <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  |
| <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  |
| <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] - C   |
| Affected flag(s)  | OV, Z, AC, C, SC, CZ  |
| <b>SBC A, x</b>   | Subtract immediate data from ACC with Carry   |
| Description       | The immediate data 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, SC, CZ  |
| <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] - C   |
| Affected flag(s)  | OV, Z, AC, C, SC, CZ  |
| <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        | $[m] \leftarrow FFH$   |
| Affected flag(s) | None   |
| <br>             |  |
| <b>SET [m].i</b> | Set bit of Data Memory   |
| Description      | Bit i of the specified Data Memory is set to 1.  |
| Operation        | $[m].i \leftarrow 1$   |
| Affected flag(s) | None   |
| <br>             |  |
| <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 Data Memory is not 0   |
| Description      | If 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>SNZ [m]</b>   | Skip if Data Memory is not 0   |
| Description      | If 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 $[m] \neq 0$   |
| 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, SC, CZ   |

|                   |  |
|-------------------|--|
| <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, SC, CZ   |
| <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, SC, CZ   |
| <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         | $ACC.3 \sim ACC.0 \leftarrow [m].7 \sim [m].4$<br>$ACC.7 \sim ACC.4 \leftarrow [m].3 \sim [m].0$   |
| Affected flag(s)  | None   |
| <b>SZ [m]</b>     | Skip if Data Memory is 0   |
| Description       | 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         | $ACC \leftarrow [m]$<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) to TBLH and Data Memory   |
| Description        | The low byte of the program code (specific page) addressed by the table pointer pair (TBLP and 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   |
| <br>               |  |
| <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   |
| <br>               |  |
| <b>ITABRD [m]</b>  | Increment table pointer low byte first and read table to TBLH and Data Memory  |
| Description        | Increment table pointer low byte, TBLP, first and then the program code addressed by the table pointer (TBHP and 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   |
| <br>               |  |
| <b>ITABRDL [m]</b> | Increment table pointer low byte first and read table (last page) to TBLH and Data Memory  |
| Description        | Increment table pointer low byte, TBLP, first and then 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   |
| <br>               |  |
| <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  |
| <br>               |  |
| <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] ← ACC "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          | ACC ← ACC "XOR" x  |
| Affected flag(s)   | Z  |



### Extended Instruction Definition

The extended instructions are used to directly access the data stored in any data memory sections.

|                    |   |
|--------------------|---|
| <b>LADC 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, SC  |
| <b>LADCM 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, SC  |
| <b>LADD 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, SC  |
| <b>LADDM 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, SC  |
| <b>LAND 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>LANDM 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>LCLR [m]</b>    | Clear Data Memory   |
| Description        | Each bit of the specified Data Memory is cleared to 0.  |
| Operation          | $[m] \leftarrow 00H$  |
| Affected flag(s)   | None  |
| <b>LCLR [m].i</b>  | Clear bit of Data Memory  |
| Description        | Bit i of the specified Data Memory is cleared to 0.   |
| Operation          | $[m].i \leftarrow 0$  |
| Affected flag(s)   | None  |

|                  |  |
|------------------|--|
| <b>LCPL [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        | $[m] \leftarrow \overline{[m]}$  |
| Affected flag(s) | Z  |
|                  |  |
| <b>LCPLA [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        | $ACC \leftarrow \overline{[m]}$  |
| Affected flag(s) | Z  |
|                  |  |
| <b>LDAA [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        | $[m] \leftarrow ACC + 00H$ or<br>$[m] \leftarrow ACC + 06H$ or<br>$[m] \leftarrow ACC + 60H$ or<br>$[m] \leftarrow ACC + 66H$  |
| Affected flag(s) | C  |
|                  |  |
| <b>LDEC [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  |
|                  |  |
| <b>LDECA [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  |
|                  |  |
| <b>LINC [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  |
|                  |  |
| <b>LINCA [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  |

|                   |   |
|-------------------|---|
| <b>LMOV 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  |
| <b>LMOV [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>LOR 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         | $ACC \leftarrow ACC \text{ "OR" } [m]$  |
| Affected flag(s)  | Z   |
| <b>LORM 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         | $[m] \leftarrow ACC \text{ "OR" } [m]$  |
| Affected flag(s)  | Z   |
| <b>LRL [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         | $[m].(i+1) \leftarrow [m].i; (i=0\sim 6)$<br>$[m].0 \leftarrow [m].7$   |
| Affected flag(s)  | None  |
| <b>LRLA [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         | $ACC.(i+1) \leftarrow [m].i; (i=0\sim 6)$<br>$ACC.0 \leftarrow [m].7$   |
| Affected flag(s)  | None  |
| <b>LRLC [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         | $[m].(i+1) \leftarrow [m].i; (i=0\sim 6)$<br>$[m].0 \leftarrow C$<br>$C \leftarrow [m].7$   |
| Affected flag(s)  | C   |
| <b>LRLCA [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         | $ACC.(i+1) \leftarrow [m].i; (i=0\sim 6)$<br>$ACC.0 \leftarrow C$<br>$C \leftarrow [m].7$   |
| Affected flag(s)  | C   |

|                    |   |
|--------------------|---|
| <b>LRR [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>LRRRA [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>LRRC [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   |
| <br>               |   |
| <b>LRRCA [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          | $ACC.i \leftarrow [m].(i+1); (i=0\sim6)$<br>$ACC.7 \leftarrow C$<br>$C \leftarrow [m].0$  |
| Affected flag(s)   | C   |
| <br>               |   |
| <b>LSBC 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          | $ACC \leftarrow ACC - [m] - C$  |
| Affected flag(s)   | OV, Z, AC, C, SC, CZ  |
| <br>               |   |
| <b>LSBCM 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          | $[m] \leftarrow ACC - [m] - C$  |
| Affected flag(s)   | OV, Z, AC, C, SC, CZ  |

|                   |   |
|-------------------|---|
| <b>LSDZ [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         | $[m] \leftarrow [m] - 1$<br>Skip if $[m]=0$   |
| Affected flag(s)  | None  |
| <b>LSDZA [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         | $ACC \leftarrow [m] - 1$<br>Skip if $ACC=0$   |
| Affected flag(s)  | None  |
| <b>LSET [m]</b>   | Set Data Memory   |
| Description       | Each bit of the specified Data Memory is set to 1.  |
| Operation         | $[m] \leftarrow FFH$  |
| Affected flag(s)  | None  |
| <b>LSET [m].i</b> | Set bit of Data Memory  |
| Description       | Bit i of the specified Data Memory is set to 1.   |
| Operation         | $[m].i \leftarrow 1$  |
| Affected flag(s)  | None  |
| <b>LSIZ [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  |
| <b>LSIZA [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  |
| <b>LSNZ [m].i</b> | Skip if Data Memory is not 0  |
| Description       | If 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  |

|                    |  |
|--------------------|--|
| <b>LSNZ [m]</b>    | Skip if Data Memory is not 0   |
| Description        | If the content 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] ≠ 0  |
| Affected flag(s)   | None   |
| <b>LSUB 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 ← ACC – [m]  |
| Affected flag(s)   | OV, Z, AC, C, SC, CZ   |
| <b>LSUBM 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] ← ACC – [m]  |
| Affected flag(s)   | OV, Z, AC, C, SC, CZ   |
| <b>LSWAP [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~[m].0 ↔ [m].7~[m].4  |
| Affected flag(s)   | None   |
| <b>LSWAPA [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>LSZ [m]</b>     | Skip if Data Memory is 0   |
| Description        | 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>LSZA [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>LSZ [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   |
| <br>                |  |
| <b>LTABRD [m]</b>   | Read table (current page) to TBLH and Data Memory  |
| Description         | The low byte of the program code (current 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   |
| <br>                |  |
| <b>LTABRDL [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   |
| <br>                |  |
| <b>LITABRD [m]</b>  | Increment table pointer low byte first and read table to TBLH and Data Memory  |
| Description         | Increment table pointer low byte, TBLP, first and then the program code addressed by the table pointer (TBLP and TBLH) 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   |
| <br>                |  |
| <b>LITABRDL [m]</b> | Increment table pointer low byte first and read table (last page) to TBLH and Data Memory  |
| Description         | Increment table pointer low byte, TBLP, first and then 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   |
| <br>                |  |
| <b>LXOR 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  |
| <br>                |  |
| <b>LXORM 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] ← ACC "XOR" [m]  |
| Affected flag(s)    | Z  |

## Package Information

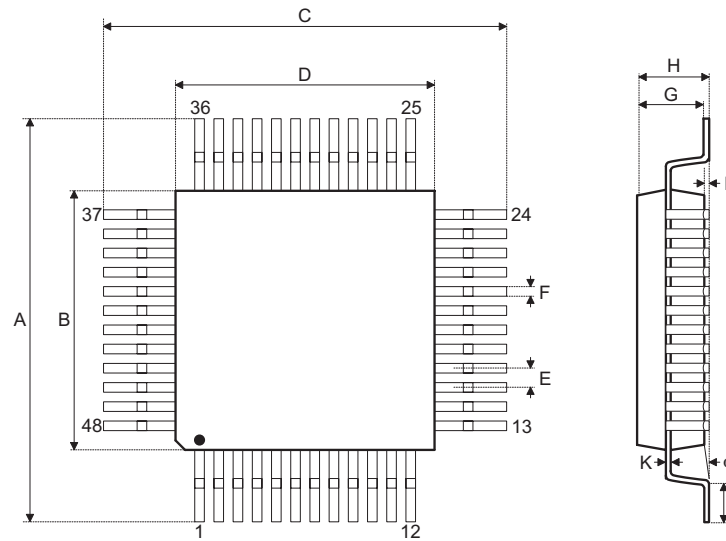
Note that the package information provided here is for consultation purposes only. As this information may be updated at regular intervals users are reminded to consult the [Holtek website](#) for the latest version of the [Package/Carton Information](#).

Additional supplementary information with regard to packaging is listed below. Click on the relevant section to be transferred to the relevant website page.

- Further Package Information (include Outline Dimensions, Product Tape and Reel Specifications)
- Packing Materials Information
- Carton information



**48-pin LQFP (7mm×7mm) Outline Dimensions**



| Symbol | Dimensions in inch |           |       |
|--------|--------------------|-----------|-------|
|        | Min.               | Nom.      | Max.  |
| A      | —                  | 0.354 BSC | —     |
| B      | —                  | 0.276 BSC | —     |
| C      | —                  | 0.354 BSC | —     |
| D      | —                  | 0.276 BSC | —     |
| E      | —                  | 0.020 BSC | —     |
| F      | 0.007              | 0.009     | 0.011 |
| G      | 0.053              | 0.055     | 0.057 |
| H      | —                  | —         | 0.063 |
| I      | 0.002              | —         | 0.006 |
| J      | 0.018              | 0.024     | 0.030 |
| K      | 0.004              | —         | 0.008 |
| α      | 0°                 | —         | 7°    |

| Symbol | Dimensions in mm |          |      |
|--------|------------------|----------|------|
|        | Min.             | Nom.     | Max. |
| A      | —                | 9.00 BSC | —    |
| B      | —                | 7.00 BSC | —    |
| C      | —                | 9.00 BSC | —    |
| D      | —                | 7.00 BSC | —    |
| E      | —                | 0.50 BSC | —    |
| F      | 0.17             | 0.22     | 0.27 |
| G      | 1.35             | 1.40     | 1.45 |
| H      | —                | —        | 1.60 |
| I      | 0.05             | —        | 0.15 |
| J      | 0.45             | 0.60     | 0.75 |
| K      | 0.09             | —        | 0.20 |
| α      | 0°               | —        | 7°   |

Copyright© 2016 by HOLTEK SEMICONDUCTOR INC.

The information appearing in this Data Sheet is believed to be accurate at the time of publication. However, Holtek assumes no responsibility arising from the use of the specifications described. The applications mentioned herein are used solely for the purpose of illustration and Holtek makes no warranty or representation that such applications will be suitable without further modification, nor recommends the use of its products for application that may present a risk to human life due to malfunction or otherwise. Holtek's products are not authorized for use as critical components in life support devices or systems. Holtek reserves the right to alter its products without prior notification. For the most up-to-date information, please visit our web site at <http://www.holtek.com.tw/en/home>.