



---

**A/D Flash MCU with EEPROM**

**HT66F31A5**

Revision: V1.40 Date: September 07, 2023

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

## Features

### CPU Features

- Operating Voltage
  - ♦  $f_{SYS}=8\text{MHz}$ : 1.8V~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
- Oscillator types
  - ♦ External High Speed Crystal – HXT
  - ♦ Internal High Speed 8/12/16 MHz RC – HIRC
  - ♦ External Low Speed 32.768kHz Crystal – LXT
  - ♦ Internal Low Speed 32kHz RC – LIRC
- Multi-mode operation: FAST, SLOW, IDLE and SLEEP
- Fully integrated internal oscillators require no external components
- All instructions executed in 1~3 instruction cycles
- Table read instructions
- 115 powerful instructions
- 12-level subroutine nesting
- Bit manipulation instruction

### Peripheral Features

- Flash Program Memory: 16K $\times$ 16
- RAM Data Memory: 1K $\times$ 8
- True EEPROM Memory: 1K $\times$ 8
- Watchdog Timer function
- In Application Programming – IAP
- 30 bidirectional I/O lines
- Three external interrupt lines shared with I/O pins
- Programmable I/O port source current for LED applications
- Multiple Timer Modules for time measure, compare match output, PWM output function or single pulse output function
- Serial Interface Module – SIM for SPI or I<sup>2</sup>C communication
- Two Fully-duplex / Half-duplex Asynchronous Receiver and Transmitter Interfaces – UARTs
- Software controlled 30-SCOM/SSEG lines LCD driver with 1/3 bias
- 12 external channel 12-bit resolution A/D converter with Programmable Internal Reference Voltage  $V_{VR}$
- Single comparator function
- Dual Time Base functions for generation of fixed time interrupt signals
- Low voltage reset function
- Low voltage detect function
- Package types: 24-pin SSOP/QFN, 28-pin SSOP/QFN, 32-pin QFN

## Development Tools

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

[https://www.holtek.com/washing\\_machine\\_workshop](https://www.holtek.com/washing_machine_workshop)

## General Description

The device is a Flash Memory type 8-bit high performance RISC architecture microcontroller.

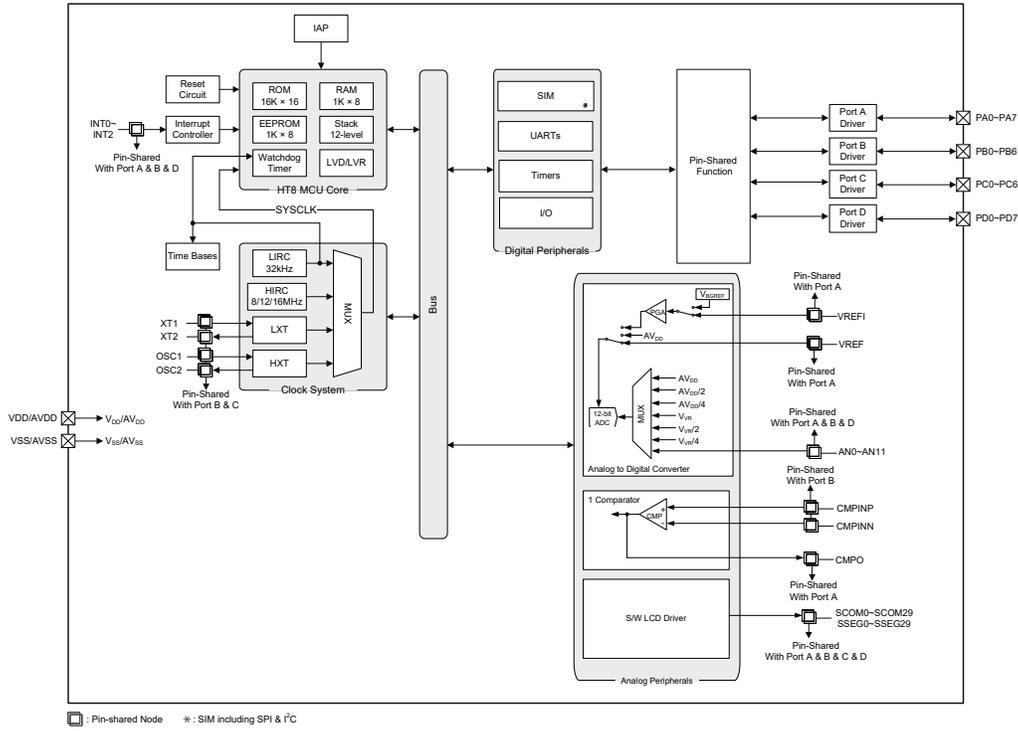
For memory features, the Flash Memory offers users the convenience of Flash Memory multi-programming features. Other memory includes an area of RAM Data Memory as well as an area of true EEPROM memory for storage of non-volatile data such as serial numbers, calibration data etc. By using the In Application Programming technology, user have a convenient means to directly store their measured data in the Flash Program Memory as well as having the ability to easily update their application programs.

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 output functions. Communication with the outside world is catered for by including fully integrated SPI, I<sup>2</sup>C and UART interface functions, three 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 low and high oscillator functions are provided, the fully integrated system oscillators require no external components for their implementation. The ability to operate and switch dynamically between a range of operating modes using different clock sources gives users the ability to optimise microcontroller operation and minimise power consumption.

The inclusion of flexible I/O programming features, Time Base functions 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.

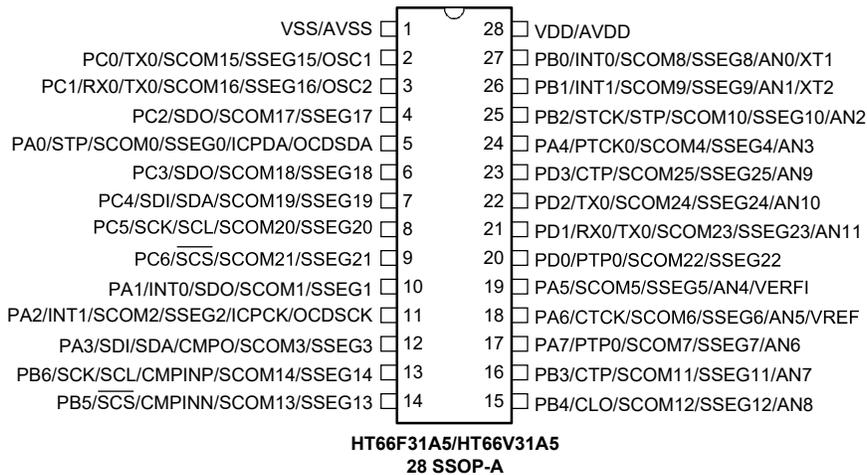
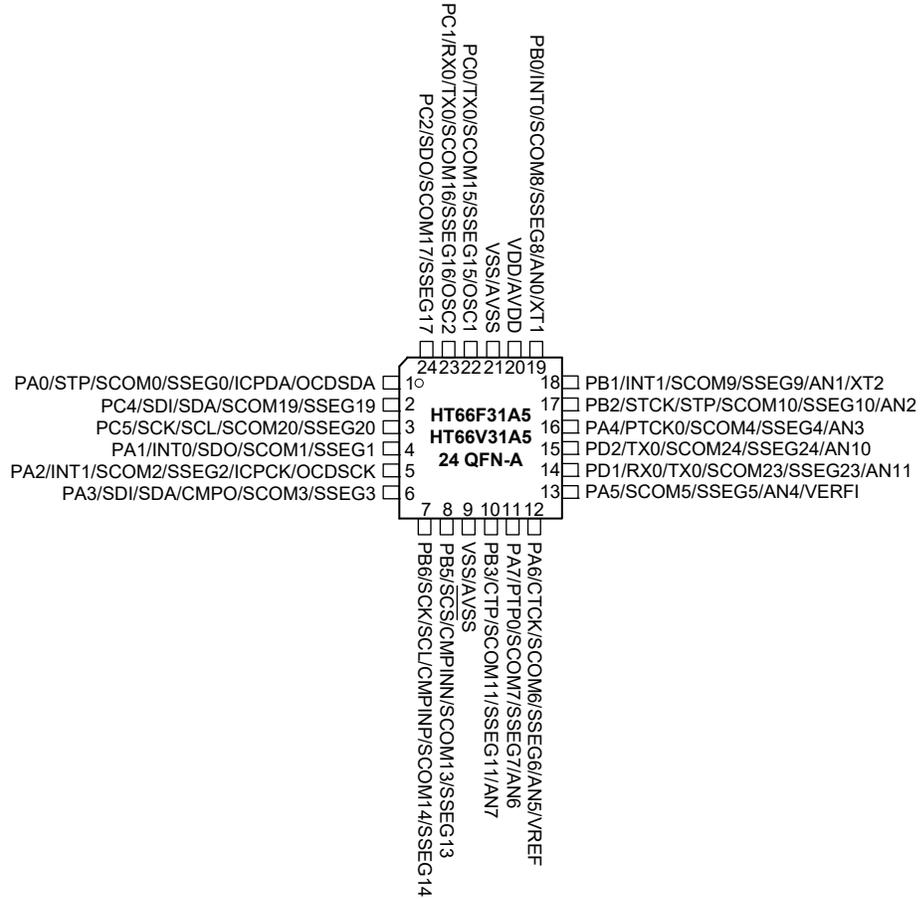
## Block Diagram

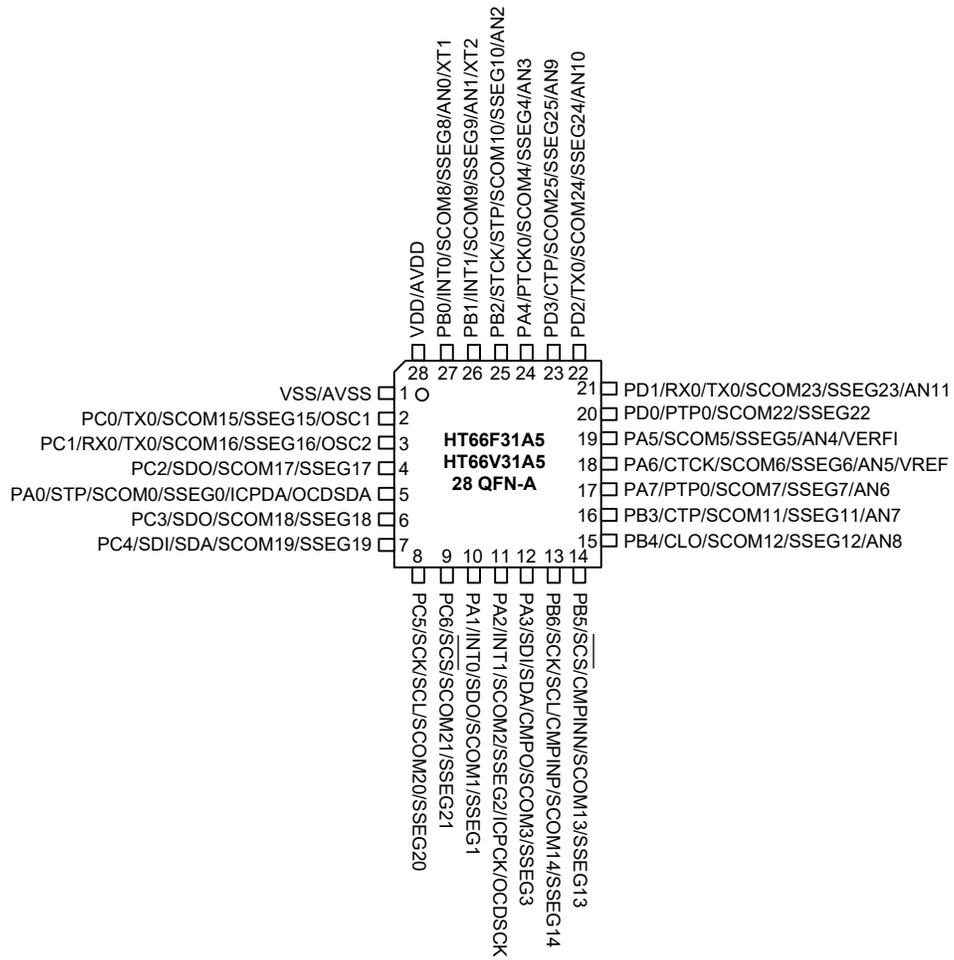


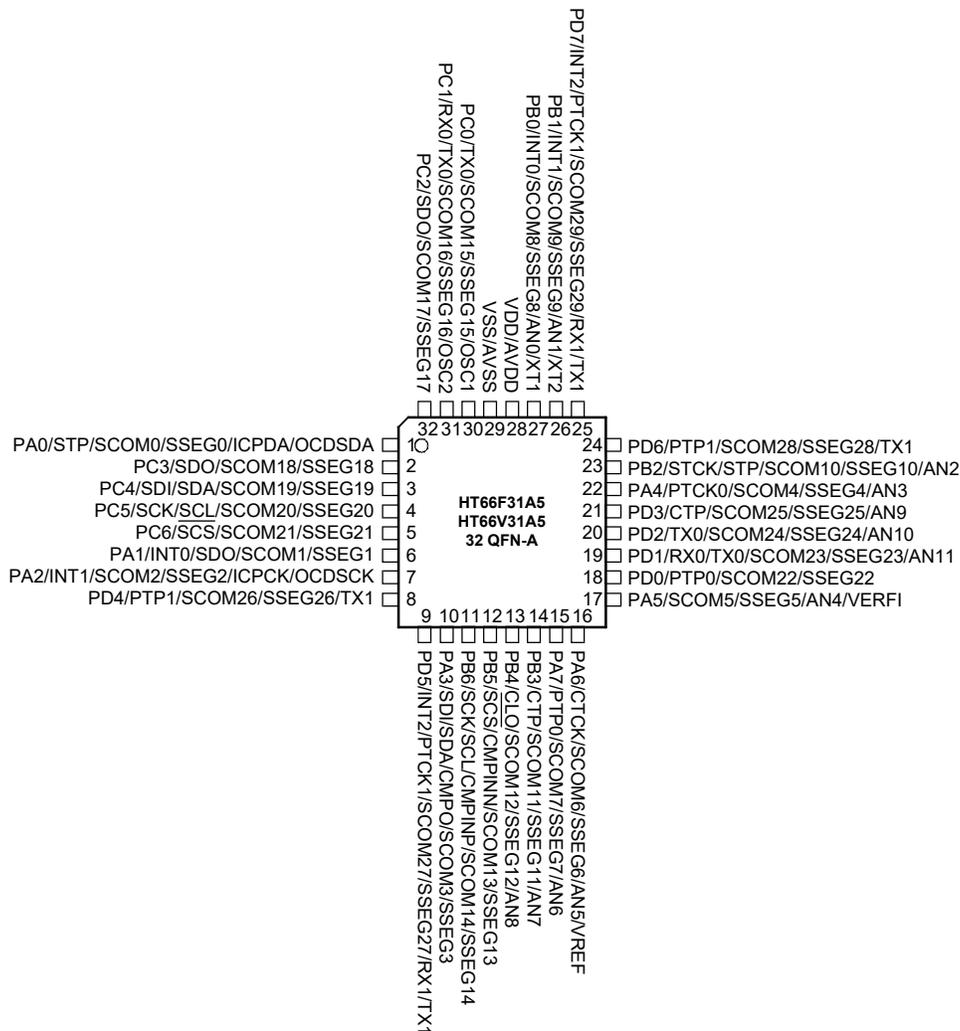
## Pin Assignment

|                                   |    |    |                                |
|-----------------------------------|----|----|--------------------------------|
| VSS/AVSS                          | 1  | 24 | VDD/AVDD                       |
| PC0/TX0/SCOM15/SSEG15/OSC1        | 2  | 23 | PB0/INT0/SCOM8/SSEG8/AN0/XT1   |
| PC1/RX0/TX0/SCOM16/SSEG16/OSC2    | 3  | 22 | PB1/INT1/SCOM9/SSEG9/AN1/XT2   |
| PC2/SDO/SCOM17/SSEG17             | 4  | 21 | PB2/STCK/STP/SCOM10/SSEG10/AN2 |
| PA0/STP/SCOM0/SSEG0/ICPDA/OCSDA   | 5  | 20 | PA4/PTCK0/SCOM4/SSEG4/AN3      |
| PC4/SDI/SDA/SCOM19/SSEG19         | 6  | 19 | PD2/TX0/SCOM24/SSEG24/AN10     |
| PC5/SCK/SCL/SCOM20/SSEG20         | 7  | 18 | PD1/RX0/TX0/SCOM23/SSEG23/AN11 |
| PA1/INT0/SDO/SCOM1/SSEG1          | 8  | 17 | PA5/SCOM5/SSEG5/AN4/VREFI      |
| PA2/INT1/SCOM2/SSEG2/ICPCK/OCDSCK | 9  | 16 | PA6/CTCK/SCOM6/SSEG6/AN5/VREF  |
| PA3/SDI/SDA/CMPO/SCOM3/SSEG3      | 10 | 15 | PA7/PTP0/SCOM7/SSEG7/AN6       |
| PB6/SCK/SCL/CMPINP/SCOM14/SSEG14  | 11 | 14 | PB3/CTP/SCOM11/SSEG11/AN7      |
| PB5/SCS/CMPINN/SCOM13/SSEG13      | 12 | 13 | PB4/CLO/SCOM12/SSEG12/AN8      |

**HT66F31A5/HT66V31A5**  
**24 SSOP-A**







- Note: 1. If the pin-shared pin functions have multiple outputs, the desired pin-shared function is determined by the corresponding software control bits.
2. The OCSDA and OCDSCK pins are used as the OCDS dedicated pins and only available for the HT66V31A5 device which is the OCDS EV chip of the HT66F31A5.
3. For less pin-count package types there will be unbonded pins which should be properly configured to avoid unwanted current consumption resulting from floating input conditions. Refer to the “Standby Current Considerations” and “Input/Output Ports” sections.

## Pin Description

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

| Pin Name                              | Function | OPT                            | I/T | O/T  | Description   |
|---------------------------------------|----------|--------------------------------|-----|------|---|
| PA0/STP/SCOM0/SSEG0/<br>ICPDA/OCDSDA  | PA0      | PAS0<br>PAWU<br>PAPU           | ST  | CMOS | General purpose I/O. Register enabled pull-high and wake-up |
|                                       | STP      | PAS0                           | —   | CMOS | STM output  |
|                                       | SCOM0    | PAS0                           | —   | SCOM | Software controlled LCD common output                       |
|                                       | SSEG0    | PAS0                           | —   | SSEG | Software controlled LCD segment output                      |
|                                       | ICPDA    | —                              | ST  | CMOS | ICP data/address pin  |
|                                       | OCDSDA   | —                              | ST  | CMOS | OCDS data/address pin, for EV chip only                     |
| PA1/INT0/SDO/SCOM1/<br>SSEG1          | PA1      | PAS0<br>PAWU<br>PAPU           | ST  | CMOS | General purpose I/O. Register enabled pull-high and wake-up |
|                                       | INT0     | PAS0<br>INTEG<br>INTC0<br>IFS0 | ST  | —    | External interrupt 0  |
|                                       | SDO      | PAS0                           | —   | CMOS | SPI serial data output                                      |
|                                       | SCOM1    | PAS0                           | —   | SCOM | Software controlled LCD common output                       |
|                                       | SSEG1    | PAS0                           | —   | SSEG | Software controlled LCD segment output                      |
| PA2/INT1/SCOM2/SSEG2/<br>ICPCK/OCDSCK | PA2      | PAS0<br>PAWU<br>PAPU           | ST  | CMOS | General purpose I/O. Register enabled pull-high and wake-up |
|                                       | INT1     | PAS0<br>INTEG<br>INTC2<br>IFS0 | ST  | —    | External interrupt 1  |
|                                       | SCOM2    | PAS0                           | —   | SCOM | Software controlled LCD common output                       |
|                                       | SSEG2    | PAS0                           | —   | SSEG | Software controlled LCD segment output                      |
|                                       | ICPCK    | —                              | ST  | —    | ICP clock pin   |
|                                       | OCDSCK   | —                              | ST  | —    | OCDS clock pin, for EV chip only                            |
| PA3/SDI/SDA/CMPO/<br>SCOM3/SSEG3      | PA3      | PAS0<br>PAWU<br>PAPU           | ST  | CMOS | General purpose I/O. Register enabled pull-high and wake-up |
|                                       | SDI      | PAS0<br>IFS0                   | ST  | —    | SPI serial data input                                       |
|                                       | SDA      | PAS0<br>IFS0                   | ST  | NMOS | I <sup>2</sup> C data line                                  |
|                                       | CMPO     | PAS0                           | —   | CMOS | Comparator output   |
|                                       | SCOM3    | PAS0                           | —   | SCOM | Software controlled LCD common output                       |
|                                       | SSEG3    | PAS0                           | —   | SSEG | Software controlled LCD segment output                      |
| PA4/PTCK0/SCOM4/<br>SSEG4/AN3         | PA4      | PAS1<br>PAWU<br>PAPU           | ST  | CMOS | General purpose I/O. Register enabled pull-high and wake-up |
|                                       | PTCK0    | PAS1                           | ST  | —    | PTM0 clock input  |
|                                       | SCOM4    | PAS1                           | —   | SCOM | Software controlled LCD common output                       |
|                                       | SSEG4    | PAS1                           | —   | SSEG | Software controlled LCD segment output                      |
|                                       | AN3      | PAS1                           | AN  | —    | A/D Converter analog input channel 3                        |

| Pin Name                       | Function | OPT                            | I/T | O/T  | Description   |
|--------------------------------|----------|--------------------------------|-----|------|---|
| PA5/SCOM5/SSEG5/AN4/VREF1      | PA5      | PAS1<br>PAWU<br>PAPU           | ST  | CMOS | General purpose I/O. Register enabled pull-high and wake-up |
|                                | SCOM5    | PAS1                           | —   | SCOM | Software controlled LCD common output                       |
|                                | SSEG5    | PAS1                           | —   | SSEG | Software controlled LCD segment output                      |
|                                | AN4      | PAS1                           | AN  | —    | A/D Converter analog input channel 4                        |
|                                | VREF1    | PAS1                           | AN  | —    | A/D Converter PGA input                                     |
| PA6/CTCK/SCOM6/SSEG6/AN5/VREF  | PA6      | PAS1<br>PAWU<br>PAPU           | ST  | CMOS | General purpose I/O. Register enabled pull-high and wake-up |
|                                | CTCK     | PAS1                           | ST  | —    | CTM clock input   |
|                                | SCOM6    | PAS1                           | —   | SCOM | Software controlled LCD common output                       |
|                                | SSEG6    | PAS1                           | —   | SSEG | Software controlled LCD segment output                      |
|                                | AN5      | PAS1                           | AN  | —    | A/D Converter analog input channel 5                        |
| PA7/PTP0/SCOM7/SSEG7/AN6       | PA7      | PAS1<br>PAWU<br>PAPU           | ST  | CMOS | General purpose I/O. Register enabled pull-high and wake-up |
|                                | PTP0     | PAS1                           | —   | CMOS | PTM0 output   |
|                                | SCOM7    | PAS1                           | —   | SCOM | Software controlled LCD common output                       |
|                                | SSEG7    | PAS1                           | —   | SSEG | Software controlled LCD segment output                      |
|                                | AN6      | PAS1                           | AN  | —    | A/D Converter analog input channel 6                        |
| PB0/INT0/SCOM8/SSEG8/AN0/XT1   | PB0      | PBS0<br>PBPU                   | ST  | CMOS | General purpose I/O. Register enabled pull-high             |
|                                | INT0     | PBS0<br>INTEG<br>INTC0<br>IFS0 | ST  | —    | External interrupt 0  |
|                                | SCOM8    | PBS0                           | —   | SCOM | Software controlled LCD common output                       |
|                                | SSEG8    | PBS0                           | —   | SSEG | Software controlled LCD segment output                      |
|                                | AN0      | PBS0                           | AN  | —    | A/D Converter analog input channel 0                        |
|                                | XT1      | PBS0                           | AN  | —    | LXT oscillator pin  |
| PB1/INT1/SCOM9/SSEG9/AN1/XT2   | PB1      | PBS0<br>PBPU                   | ST  | CMOS | General purpose I/O. Register enabled pull-high             |
|                                | INT1     | PBS0<br>INTEG<br>INTC2<br>IFS0 | ST  | —    | External Interrupt 1  |
|                                | SCOM9    | PBS0                           | —   | SCOM | Software controlled LCD common output                       |
|                                | SSEG9    | PBS0                           | —   | SSEG | Software controlled LCD segment output                      |
|                                | AN1      | PBS0                           | AN  | —    | A/D Converter analog input channel 1                        |
|                                | XT2      | PBS0                           | —   | AN   | LXT oscillator pin  |
| PB2/STCK/STP/SCOM10/SSEG10/AN2 | PB2      | PBS0<br>PBPU                   | ST  | CMOS | General purpose I/O. Register enabled pull-high             |
|                                | STCK     | PBS0                           | ST  | —    | STM clock input   |
|                                | STP      | PBS0                           | —   | CMOS | STM output  |
|                                | SCOM10   | PBS0                           | —   | SCOM | Software controlled LCD common output                       |
|                                | SSEG10   | PBS0                           | —   | SSEG | Software controlled LCD segment output                      |
|                                | AN2      | PBS0                           | AN  | —    | A/D Converter analog input channel 2                        |

| Pin Name   | Function                | OPT          | I/T | O/T  | Description  |
|--|-------------------------|--------------|-----|------|--|
| PB3/CTP/SCOM11/<br>SSEG11/AN7                          | PB3                     | PBS0<br>PBPU | ST  | CMOS | General purpose I/O. Register enabled pull-high  |
|  | CTP                     | PBS0         | —   | CMOS | CTM output   |
|  | SCOM11                  | PBS0         | —   | SCOM | Software controlled LCD common output  |
|  | SSEG11                  | PBS0         | —   | SSEG | Software controlled LCD segment output   |
|  | AN7                     | PBS0         | AN  | —    | A/D Converter analog input channel 7   |
| PB4/CLO/SCOM12/<br>SSEG12/AN8                          | PB4                     | PBS1<br>PBPU | ST  | CMOS | General purpose I/O. Register enabled pull-high  |
|  | CLO                     | PBS1         | —   | CMOS | System clock output  |
|  | SCOM12                  | PBS1         | —   | SCOM | Software controlled LCD common output  |
|  | SSEG12                  | PBS1         | —   | SSEG | Software controlled LCD segment output   |
|  | AN8                     | PBS1         | AN  | —    | A/D Converter analog input channel 8   |
| PB5/ $\overline{\text{SCS}}$ /CMPINN/<br>SCOM13/SSEG13 | PB5                     | PBS1<br>PBPU | ST  | CMOS | General purpose I/O. Register enabled pull-high  |
|  | $\overline{\text{SCS}}$ | PBS1<br>IFS0 | ST  | CMOS | SPI slave select pin   |
|  | CMPINN                  | PBS1         | AN  | —    | Comparator negative input  |
|  | SCOM13                  | PBS1         | —   | SCOM | Software controlled LCD common output  |
|  | SSEG13                  | PBS1         | —   | SSEG | Software controlled LCD segment output   |
| PB6/SCK/SCL/CMPINP/<br>SCOM14/SSEG14                   | PB6                     | PBS1<br>PBPU | ST  | CMOS | General purpose I/O. Register enabled pull- high   |
|  | SCK                     | PBS1<br>IFS0 | ST  | CMOS | SPI serial clock   |
|  | SCL                     | PBS1<br>IFS0 | ST  | NMOS | I <sup>2</sup> C clock line  |
|  | CMPINP                  | PBS1         | AN  | —    | Comparator positive input  |
|  | SCOM14                  | PBS1         | —   | SCOM | Software controlled LCD common output  |
|  | SSEG14                  | PBS1         | —   | SSEG | Software controlled LCD segment output   |
| PC0/TX0/SCOM15/<br>SSEG15/OSC1                         | PC0                     | PCS0<br>PCPU | ST  | CMOS | General purpose I/O. Register enabled pull- high   |
|  | TX0                     | PCS0         | —   | CMOS | UART0 serial data output   |
|  | SCOM15                  | PCS0         | —   | SCOM | Software controlled LCD common output  |
|  | SSEG15                  | PCS0         | —   | SSEG | Software controlled LCD segment output   |
|  | OSC1                    | PCS0         | AN  | —    | HXT oscillator pin   |
| PC1/RX0/TX0/SCOM16/<br>SSEG16/OSC2                     | PC1                     | PCS0<br>PCPU | ST  | CMOS | General purpose I/O. Register enabled pull-high  |
|  | RX0/TX0                 | PCS0<br>IFS0 | ST  | CMOS | UART0 serial data input in full-duplex communication or UART0 serial data input/output in single wire mode communication |
|  | SCOM16                  | PCS0         | —   | SCOM | Software controlled LCD common output  |
|  | SSEG16                  | PCS0         | —   | SSEG | Software controlled LCD segment output   |
|  | OSC2                    | PCS0         | —   | AN   | HXT oscillator pin   |
| PC2/SDO/SCOM17/<br>SSEG17                              | PC2                     | PCS0<br>PCPU | ST  | CMOS | General purpose I/O. Register enabled pull-high  |
|  | SDO                     | PCS0         | —   | CMOS | SPI serial data output   |
|  | SCOM17                  | PCS0         | —   | SCOM | Software controlled LCD common output  |
|  | SSEG17                  | PCS0         | —   | SSEG | Software controlled LCD segment output   |

| Pin Name                                 | Function         | OPT          | I/T | O/T  | Description  |
|--|------------------|--------------|-----|------|--|
| PC3/SDO/SCOM18/<br>SSEG18                | PC3              | PCS0<br>PCPU | ST  | CMOS | General purpose I/O. Register enabled pull-high  |
|  | SDO              | PCS0         | —   | CMOS | SPI serial data output   |
|  | SCOM18           | PCS0         | —   | SCOM | Software controlled LCD segment output   |
|  | SSEG18           | PCS0         | —   | SSEG | Software controlled LCD common output  |
| PC4/SDI/SDA/SCOM19/<br>SSEG19            | PC4              | PCS1<br>PCPU | ST  | CMOS | General purpose I/O. Register enabled pull-high  |
|  | SDI              | PCS1<br>IFS0 | ST  | —    | SPI serial data input  |
|  | SDA              | PCS1<br>IFS0 | ST  | NMOS | I <sup>2</sup> C data line   |
|  | SCOM19           | PCS1         | —   | SCOM | Software controlled LCD common output  |
|  | SSEG19           | PCS1         | —   | SSEG | Software controlled LCD segment output   |
| PC5/SCK/SCL/SCOM20/<br>SSEG20            | PC5              | PCS1<br>PCPU | ST  | CMOS | General purpose I/O. Register enabled pull-high  |
|  | SCK              | PCS1<br>IFS0 | ST  | CMOS | SPI serial clock   |
|  | SCL              | PCS1<br>IFS0 | ST  | NMOS | I <sup>2</sup> C clock line  |
|  | SCOM20           | PCS1         | —   | SCOM | Software controlled LCD common output  |
|  | SSEG20           | PCS1         | —   | SSEG | Software controlled LCD segment output   |
| PC6/ $\overline{SCS}$ /SCOM21/<br>SSEG21 | PC6              | PCS1<br>PCPU | ST  | CMOS | General purpose I/O. Register enabled pull-high  |
|  | $\overline{SCS}$ | PCS1<br>IFS0 | ST  | CMOS | SPI slave select pin   |
|  | SCOM21           | PCS1         | —   | SCOM | Software controlled LCD common output  |
|  | SSEG21           | PCS1         | —   | SSEG | Software controlled LCD segment output   |
| PD0/PTP0/SCOM22/<br>SSEG22               | PD0              | PDS0<br>PDPU | ST  | CMOS | General purpose I/O. Register enabled pull-high  |
|  | PTP0             | PDS0         | —   | CMOS | PTM0 output  |
|  | SCOM22           | PDS0         | —   | SCOM | Software controlled LCD common output  |
|  | SSEG22           | PDS0         | —   | SSEG | Software controlled LCD segment output   |
| PD1/RX0/TX0/SCOM23/<br>SSEG23/AN11       | PD1              | PDS0<br>PDPU | ST  | CMOS | General purpose I/O. Register enabled pull-high  |
|  | RX0/TX0          | PDS0<br>IFS0 | ST  | CMOS | UART0 serial data input in full-duplex communication or UART0 serial data input/output in single wire mode communication |
|  | SCOM23           | PDS0         | —   | SCOM | Software controlled LCD common output  |
|  | SSEG23           | PDS0         | —   | SSEG | Software controlled LCD segment output   |
|  | AN11             | PDS0         | AN  | —    | A/D Converter analog input channel 11  |
| PD2/TX0/SCOM24/<br>SSEG24/AN10           | PD2              | PDS0<br>PDPU | ST  | CMOS | General purpose I/O. Register enabled pull-high  |
|  | TX0              | PDS0         | —   | CMOS | UART0 serial data output   |
|  | SCOM24           | PDS0         | —   | SCOM | Software controlled LCD common output  |
|  | SSEG24           | PDS0         | —   | SSEG | Software controlled LCD segment output   |
|  | AN10             | PDS0         | AN  | —    | A/D Converter analog input channel 10  |

| Pin Name                                 | Function | OPT                            | I/T | O/T  | Description  |
|--|----------|--------------------------------|-----|------|--|
| PD3/CTP/SCOM25/<br>SSEG25/AN9            | PD3      | PDS0<br>PDP1                   | ST  | CMOS | General purpose I/O. Register enabled pull-high  |
|  | CTP      | PDS0                           | —   | CMOS | CTM output   |
|  | SCOM25   | PDS0                           | —   | SCOM | Software controlled LCD common output  |
|  | SSEG25   | PDS0                           | —   | SSEG | Software controlled LCD segment output   |
|  | AN9      | PDS0                           | AN  | —    | A/D Converter analog input channel 9   |
| PD4/PTP1/SCOM26/<br>SSEG26/TX1           | PD4      | PDS1<br>PDP1                   | ST  | CMOS | General purpose I/O. Register enabled pull-high  |
|  | PTP1     | PDS1                           | —   | CMOS | PTM1 output  |
|  | SCOM26   | PDS1                           | —   | SCOM | Software controlled LCD common output  |
|  | SSEG26   | PDS1                           | —   | SSEG | Software controlled LCD segment output   |
|  | TX1      | PDS1                           | —   | CMOS | UART1 serial data output   |
| PD5/INT2/PTCK1/SCOM27/<br>SSEG27/RX1/TX1 | PD5      | PDS1                           | ST  | CMOS | General purpose I/O. Register enabled pull-high  |
|  | INT2     | PDS1<br>INTEG<br>INTC3<br>IFS0 | ST  | —    | External interrupt 2   |
|  | PTCK1    | PDS1<br>IFS1                   | ST  | —    | PTM1 clock input   |
|  | SCOM27   | PDS1                           | —   | SCOM | Software controlled LCD common output  |
|  | SSEG27   | PDS1                           | —   | SSEG | Software controlled LCD segment output   |
|  | RX1/TX1  | PDS1<br>IFS0                   | ST  | CMOS | UART1 serial data input in full-duplex communication or UART1 serial data input/output in single wire mode communication |
| PD6/PTP1/SCOM28/<br>SSEG28/TX1           | PD6      | PDS1                           | ST  | CMOS | General purpose I/O. Register enabled pull-high  |
|  | PTP1     | PDS1                           | —   | CMOS | PTM1 output  |
|  | SCOM28   | PDS1                           | —   | SCOM | Software controlled LCD common output  |
|  | SSEG28   | PDS1                           | —   | SSEG | Software controlled LCD segment output   |
|  | TX1      | PDS1                           | —   | CMOS | UART1 serial data output   |
| PD7/INT2/PTCK1/SCOM29/<br>SSEG29/RX1/TX1 | PD7      | PDS1                           | ST  | CMOS | General purpose I/O. Register enabled pull-high  |
|  | INT2     | PDS1<br>INTEG<br>INTC3<br>IFS0 | ST  | —    | External interrupt 2   |
|  | PTCK1    | PDS1<br>IFS1                   | ST  | —    | PTM1 clock input   |
|  | SCOM29   | PDS1                           | —   | SCOM | Software controlled LCD common output  |
|  | SSEG29   | PDS1                           | —   | SSEG | Software controlled LCD segment output   |
|  | RX1/TX1  | PDS1<br>IFS0                   | ST  | CMOS | UART1 serial data input in full-duplex communication or UART1 serial data input/output in single wire mode communication |
| VDD/AVDD                                 | VDD      | —                              | PWR | —    | Digital positive power supply  |
|  | AVDD     | —                              | PWR | —    | Analog positive power supply   |
| VSS/AVSS                                 | VSS      | —                              | PWR | —    | Digital negative power supply, ground  |
|  | AVSS     | —                              | PWR | —    | Analog negative power supply, ground   |

Legend: I/T: Input type;

OPT: Optional by register option;

PWR: Power;

ST: Schmitt Trigger input;

SSEG: Software controlled LCD SEG;

O/T: Output type;

AN: Analog signal;

CMOS: CMOS output;

NMOS: NMOS output;

SCOM: Software controlled LCD COM.

## Absolute Maximum Ratings

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

Note: These are stress ratings only. Stresses exceeding the range specified under “Absolute Maximum Ratings” may cause substantial damage to the device. Functional operation of 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

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

### Operating Voltage Characteristics

$T_a=-40^{\circ}C-85^{\circ}C$

| Symbol                   | Parameter                | Test Conditions     |                 | Min. | Typ. | Max. | Unit |
|--------------------------|--------------------------|---------------------|-----------------|------|------|------|------|
|                          |                          | $V_{DD}$            | Conditions      |      |      |      |      |
| $V_{DD}$                 | Operating Voltage – HXT  | —                   | $f_{SYS}=8MHz$  | 1.8  | —    | 5.5  | V    |
|                          |                          |                     | $f_{SYS}=12MHz$ | 2.7  | —    | 5.5  |      |
|                          |                          |                     | $f_{SYS}=16MHz$ | 3.3  | —    | 5.5  |      |
|                          | Operating Voltage – HIRC | —                   | $f_{SYS}=8MHz$  | 1.8  | —    | 5.5  | V    |
|                          |                          |                     | $f_{SYS}=12MHz$ | 2.7  | —    | 5.5  |      |
|                          |                          |                     | $f_{SYS}=16MHz$ | 3.3  | —    | 5.5  |      |
| Operating Voltage – LXT  | —                        | $f_{SYS}=32.768kHz$ | 1.8             | —    | 5.5  | V    |      |
| Operating Voltage – LIRC | —                        | $f_{SYS}=32kHz$     | 1.8             | —    | 5.5  | V    |      |

### Operating Current Characteristics

$T_a=-40^{\circ}C-85^{\circ}C$

| Symbol   | Operation Mode   | Test Conditions |                   | Min. | Typ. | Max. | Unit    |
|----------|------------------|-----------------|-------------------|------|------|------|---------|
|          |                  | $V_{DD}$        | Conditions        |      |      |      |         |
| $I_{DD}$ | SLOW Mode – LIRC | 1.8V            | $f_{SYS}=32kHz$   | —    | 8    | 16   | $\mu A$ |
|          |                  | 3V              |                   | —    | 10   | 20   |         |
|          |                  | 5V              |                   | —    | 30   | 50   |         |
|          | SLOW Mode – LXT  | 1.8V            | $f_{SYS}=32768Hz$ | —    | 8    | 16   | $\mu A$ |
|          |                  | 3V              |                   | —    | 10   | 20   |         |
|          |                  | 5V              |                   | —    | 30   | 50   |         |

| Symbol          | Operation Mode   | Test Conditions         |                         | Min.                   | Typ. | Max. | Unit |    |
|-----------------|------------------|-------------------------|-------------------------|------------------------|------|------|------|----|
|                 |                  | V <sub>DD</sub>         | Conditions              |                        |      |      |      |    |
| I <sub>DD</sub> | FAST Mode – HIRC | 1.8V                    | f <sub>sys</sub> =8MHz  | —                      | 0.8  | 1.2  | mA   |    |
|                 |                  | 3V                      |                         | —                      | 1.0  | 1.5  |      |    |
|                 |                  | 5V                      |                         | —                      | 2    | 3    |      |    |
|                 |                  | 2.7V                    | f <sub>sys</sub> =12MHz | —                      | 1.2  | 2.2  | mA   |    |
|                 |                  | 3V                      |                         | —                      | 1.50 | 2.75 |      |    |
|                 |                  | 5V                      |                         | —                      | 3.0  | 4.5  |      |    |
|                 | 3.3V             | f <sub>sys</sub> =16MHz | —                       | 3.2                    | 4.8  | mA   |      |    |
|                 | 5V               |                         | —                       | 4                      | 6    |      |      |    |
|                 | FAST Mode – HXT  | FAST Mode – HXT         | 1.8V                    | f <sub>sys</sub> =8MHz | —    | 0.8  | 1.2  | mA |
|                 |                  |                         | 3V                      |                        | —    | 1.0  | 1.5  |    |
|                 |                  |                         | 5V                      |                        | —    | 2    | 3    |    |
|                 |                  | 2.7V                    | f <sub>sys</sub> =12MHz | —                      | 1.2  | 2.2  | mA   |    |
|                 |                  | 3V                      |                         | —                      | 1.50 | 2.75 |      |    |
|                 |                  | 5V                      |                         | —                      | 3.0  | 4.5  |      |    |
| 3.3V            |                  | f <sub>sys</sub> =16MHz | —                       | 3.2                    | 4.8  | mA   |      |    |
| 5V              |                  |                         | —                       | 4                      | 6    |      |      |    |

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

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

### Standby Current Characteristics

T<sub>a</sub>=25°C, unless otherwise specified

| Symbol           | Operation Mode    | Test Conditions                              |  | Min.  | Typ. | Max. | Max.<br>@85°C | Unit |    |
|------------------|-------------------|--|--|---|------|------|---------------|------|----|
|                  |                   | V <sub>DD</sub>                              | Conditions                                   |   |      |      |               |      |    |
| I <sub>STB</sub> | SLEEP Mode        | 1.8V   | WDT off                                      | —   | 0.45 | 1.50 | 6.00          | μA   |    |
|                  |                   | 3V   |  | —   | 0.45 | 1.80 | 7.00          |      |    |
|                  |                   | 5V   |  | —   | 0.5  | 2.0  | 8.0           |      |    |
|                  |                   | 1.8V   | WDT on                                       | —   | 1.5  | 3.0  | 7.0           | μA   |    |
|                  |                   | 3V   |  | —   | 1.8  | 3.6  | 8.0           |      |    |
|                  |                   | 5V   |  | —   | 3    | 5    | 10            |      |    |
|                  | IDLE0 Mode – LIRC | IDLE0 Mode – LIRC                            | 1.8V   | f <sub>SUB</sub> on                         | —    | 2.4  | 4.0           | 8.0  | μA |
|                  |                   |  | 3V   |   | —    | 3    | 5             | 9    |    |
|                  |                   |  | 5V   |   | —    | 5    | 10            | 11   |    |
|                  | IDLE0 Mode – LXT  | IDLE0 Mode – LXT                             | 1.8V   | f <sub>SUB</sub> on                         | —    | 2.4  | 4.0           | 8.0  | μA |
|                  |                   |  | 3V   |   | —    | 3    | 5             | 9    |    |
|                  |                   |  | 5V   |   | —    | 5    | 10            | 11   |    |
|                  | IDLE1 Mode – HIRC | IDLE1 Mode – HIRC                            | 1.8V   | f <sub>SUB</sub> on, f <sub>sys</sub> =8MHz | —    | 288  | 400           | 480  | μA |
|                  |                   |  | 3V   |   | —    | 360  | 500           | 600  |    |
|                  |                   |  | 5V   |   | —    | 850  | 1000          | 1200 |    |
|                  |                   | 2.7V   | f <sub>SUB</sub> on, f <sub>sys</sub> =12MHz | —   | 550  | 700  | 800           | μA   |    |
|                  |                   | 3V   |  | —   | 650  | 800  | 900           |      |    |
|                  |                   | 5V   |  | —   | 1800 | 2000 | 2200          |      |    |
| 3.3V             |                   | f <sub>SUB</sub> on, f <sub>sys</sub> =16MHz | —  | 1.8   | 3.6  | 4.4  | mA            |      |    |
| 5V               |                   |  | —  | 2.0   | 4.0  | 4.8  |               |      |    |

| Symbol           | Operation Mode   | Test Conditions |  | Min. | Typ. | Max. | Max.<br>@85°C | Unit |
|------------------|------------------|-----------------|--|------|------|------|---------------|------|
|                  |                  | V <sub>DD</sub> | Conditions                                   |      |      |      |               |      |
| I <sub>STB</sub> | IDLE1 Mode – HXT | 1.8V            | f <sub>SUB</sub> on, f <sub>SYS</sub> =8MHz  | —    | 288  | 400  | 480           | μA   |
|                  |                  | 3V              |  | —    | 360  | 500  | 600           |      |
|                  |                  | 5V              |  | —    | 600  | 800  | 960           |      |
|                  |                  | 2.7V            | f <sub>SUB</sub> on, f <sub>SYS</sub> =12MHz | —    | 432  | 600  | 720           | μA   |
|                  |                  | 3V              |  | —    | 540  | 750  | 900           |      |
|                  |                  | 5V              |  | —    | 800  | 1200 | 1440          |      |
|                  |                  | 3.3V            | f <sub>SUB</sub> on, f <sub>SYS</sub> =16MHz | —    | 1.8  | 3.6  | 4.4           | mA   |
| 5V               | —                | 2.0             |  | 4.0  | 4.8  |      |               |      |

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

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

## A.C. Characteristics

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

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

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

| Symbol                              | Parameter                           | Test Conditions |            | Min.  | Typ.  | Max.  | Unit |
|-------------------------------------|-------------------------------------|-----------------|------------|-------|-------|-------|------|
|                                     |                                     | V <sub>DD</sub> | Temp.      |       |       |       |      |
| f <sub>HIRC</sub>                   | 8MHz Writer Trimmed HIRC Frequency  | 3V/5V           | 25°C       | -1%   | 8     | +1%   | MHz  |
|                                     |                                     |                 | -40°C~85°C | -2%   | 8     | +2%   |      |
|                                     |                                     | 2.7V~5.5V       | 25°C       | -2.5% | 8     | +2.5% |      |
|                                     |                                     |                 | -40°C~85°C | -3%   | 8     | +3%   |      |
|                                     |                                     | 2.2V~5.5V       | 25°C       | -3.5% | 8     | +3.5% |      |
|                                     |                                     |                 | -40°C~85°C | -5%   | 8     | +5%   |      |
|                                     |                                     | 1.8V~5.5V       | 25°C       | -5%   | 8     | +5%   |      |
|                                     |                                     |                 | -40°C~85°C | -10%  | 8     | +10%  |      |
|                                     | 12MHz Writer Trimmed HIRC Frequency | 3V/5V           | 25°C       | -1%   | 12    | +1%   | MHz  |
|                                     |                                     |                 | -40°C~85°C | -2%   | 12    | +2%   |      |
|                                     |                                     | 2.7V~5.5V       | 25°C       | -2.5% | 12    | +2.5% |      |
|                                     |                                     |                 | -40°C~85°C | -3%   | 12    | +3%   |      |
| 16MHz Writer Trimmed HIRC Frequency | 5V                                  | 25°C            | -1%        | 16    | +1%   | MHz   |      |
|                                     |                                     | -40°C~85°C      | -2%        | 16    | +2%   |       |      |
|                                     | 3.3V~5.5V                           | 25°C            | -2.5%      | 16    | +2.5% |       |      |
|                                     |                                     | -40°C~85°C      | -3%        | 16    | +3%   |       |      |

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

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

3. The minimum and maximum tolerance values provided in the table are for the frequency at which the writer trims the HIRC oscillator. After trimming at this chosen specific frequency any change in HIRC oscillator frequency using the oscillator register control bits by the application program will give a frequency tolerance to within  $\pm 20\%$ .

**External High Speed Crystal/Ceramic Oscillator Characteristics – HXT**

| Symbol           | Parameter          | Test Conditions |            | Min. | Typ. | Max. | Unit |
|------------------|--------------------|-----------------|------------|------|------|------|------|
|                  |                    | V <sub>DD</sub> | Temp.      |      |      |      |      |
| f <sub>HXT</sub> | System Clock – HXT | 1.8V~5.5V       | -40°C~85°C | —    | 8    | —    | MHz  |
|                  |                    | 2.7V~5.5V       | -40°C~85°C | —    | 12   | —    |      |
|                  |                    | 3.3V~5.5V       | -40°C~85°C | —    | 16   | —    |      |

**Low Speed Internal Oscillator Characteristics – LIRC**

| Symbol             | Parameter          | Test Conditions |            | Min. | Typ. | Max. | Unit |
|--------------------|--------------------|-----------------|------------|------|------|------|------|
|                    |                    | V <sub>DD</sub> | Temp.      |      |      |      |      |
| f <sub>LIRC</sub>  | LIRC Frequency     | 3V              | 25°C       | -2%  | 32   | +2%  | kHz  |
|                    |                    | 1.8V~5.5V       | -40°C~85°C | -10% | 32   | +10% |      |
| t <sub>START</sub> | LIRC Start-up Time | —               | -40°C~85°C | —    | —    | 100  | μs   |

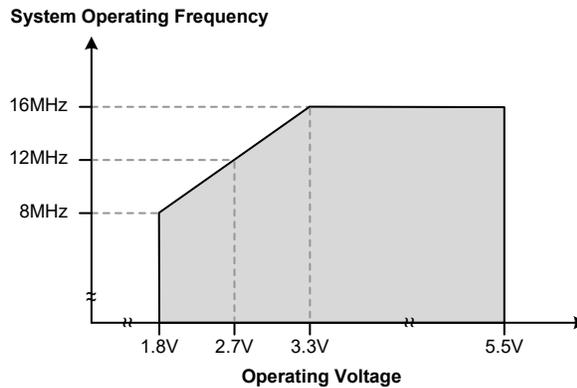
**External Low Speed Crystal Oscillator Characteristics – LXT**

T<sub>a</sub>=25°C

| Symbol             | Parameter           | Test Conditions |            | Min.  | Typ.  | Max. | Unit |
|--------------------|---------------------|-----------------|------------|-------|-------|------|------|
|                    |                     | V <sub>DD</sub> | Conditions |       |       |      |      |
| f <sub>LXT</sub>   | System Clock – LXT  | 1.8V~5.5V       | —          | —     | 32768 | —    | Hz   |
| t <sub>START</sub> | LXT Start Up Time   | 3V              | —          | —     | —     | 1000 | ms   |
|                    |                     | 5V              | —          | —     | —     | 1000 | ms   |
| Duty Cycle         | Duty Cycle          | —               | —          | 40    | —     | 60   | %    |
| R <sub>NEG</sub>   | Negative Resistance | 1.8V            | —          | 3×ESR | —     | —    | Ω    |

Note: C<sub>1</sub>, C<sub>2</sub> and R<sub>P</sub> are external components. C<sub>1</sub>=C<sub>2</sub>=10pF. R<sub>P</sub>=10MΩ. C<sub>L</sub>=7pF, ESR=30kΩ.

**Operating Frequency Characteristic Curves**



### System Start Up Time Characteristics

Ta=-40°C~85°C

| Symbol   | Parameter   | Test Conditions                          |   | Min. | Typ. | Max.              | Unit              |
|--|---|--|---|------|------|-------------------|-------------------|
|  |   | V <sub>DD</sub>                          | Conditions  |      |      |                   |                   |
| t <sub>SST</sub>   | System Start-up Time<br>(Wake-up from Condition where f <sub>SYS</sub> is off)      | —  | f <sub>SYS</sub> =f <sub>H</sub> ~f <sub>H</sub> /64, f <sub>H</sub> =f <sub>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>LXT</sub>  | —    | 1024 | —                 | t <sub>LXT</sub>  |
|  |   | —  | f <sub>SYS</sub> =f <sub>SUB</sub> =f <sub>LIRC</sub>   | —    | 2    | —                 | t <sub>LIRC</sub> |
|  | System Start-up Time<br>(Wake-up from Condition where f <sub>SYS</sub> is on)       | —  | f <sub>SYS</sub> =f <sub>H</sub> ~f <sub>H</sub> /64, f <sub>H</sub> =f <sub>HXT</sub> or f <sub>HIRC</sub> | —    | 2    | —                 | t <sub>H</sub>    |
|  |   | —  | f <sub>SYS</sub> =f <sub>SUB</sub> =f <sub>LXT</sub> or f <sub>LIRC</sub>                                   | —    | 2    | —                 | t <sub>SUB</sub>  |
| System Speed Switch Time<br>(FAST to SLOW Mode or SLOW to FAST Mode) | —   | f <sub>HXT</sub> switches from off → on  | —   | 1024 | —    | t <sub>HXT</sub>  |                   |
|  | —   | f <sub>HIRC</sub> switches from off → on | —   | 16   | —    | t <sub>HIRC</sub> |                   |
|  | —   | f <sub>LXT</sub> switches from off → on  | —   | 1024 | —    | t <sub>LXT</sub>  |                   |
| t <sub>RSTD</sub>  | System Reset Delay Time<br>(Reset source from Power-on Reset or LVR Hardware Reset) | —  | RR <sub>POR</sub> =5V/ms  | 14   | 16   | 18                | ms                |
|  | System Reset Delay Time<br>(LVRC/WDT/ RSTC Register Software Reset)                 | —  | —   |      |      |                   |                   |
|  | System Reset Delay Time<br>(Reset Source from WDT Overflow Reset)                   | —  | —   |      |      |                   |                   |
| t <sub>SRESET</sub>  | Minimum Software Reset Width to Reset   | —  | —   | 45   | 90   | 120               | μs                |

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

### Input/Output Characteristics

Ta=-40°C~85°C

| Symbol          | Parameter                        | Test Conditions |                                     | Min.               | Typ. | Max.               | Unit |
|-----------------|----------------------------------|-----------------|-------------------------------------|--------------------|------|--------------------|------|
|                 |                                  | V <sub>DD</sub> | Conditions                          |                    |      |                    |      |
| V <sub>IL</sub> | Input Low Voltage for I/O Ports  | 5V              | —                                   | 0                  | —    | 1.5                | V    |
|                 |                                  | —               |                                     | 0                  | —    | 0.2V <sub>DD</sub> |      |
| V <sub>IH</sub> | Input High Voltage for I/O Ports | 5V              | —                                   | 3.5                | —    | 5.0                | V    |
|                 |                                  | —               |                                     | 0.8V <sub>DD</sub> | —    | V <sub>DD</sub>    |      |
| I <sub>OL</sub> | Sink Current for I/O Ports       | 3V              | V <sub>OL</sub> =0.1V <sub>DD</sub> | 16                 | 32   | —                  | mA   |
|                 |                                  | 5V              |                                     | 32                 | 65   | —                  |      |

| Symbol            | Parameter  | Test Conditions |  | Min. | Typ.  | Max.  | Unit |
|-------------------|--|-----------------|--|------|-------|-------|------|
|                   |  | V <sub>DD</sub> | Conditions   |      |       |       |      |
| I <sub>OH</sub>   | Source Current for I/O Ports                             | 3V              | V <sub>OH</sub> =0.9V <sub>DD</sub> ,<br>SLEDCn[m+1:m]=00B<br>(n=0, 1; m=0, 2, 4, 6) | -0.7 | -1.5  | —     | mA   |
|                   |  | 5V              |  | -1.5 | -2.9  | —     |      |
|                   |  | 3V              | V <sub>OH</sub> =0.9V <sub>DD</sub> ,<br>SLEDCn[m+1:m]=01B<br>(n=0, 1; m=0, 2, 4, 6) | -1.3 | -2.5  | —     |      |
|                   |  | 5V              |  | -2.5 | -5.1  | —     |      |
|                   |  | 3V              | V <sub>OH</sub> =0.9V <sub>DD</sub> ,<br>SLEDCn[m+1:m]=10B<br>(n=0, 1; m=0, 2, 4, 6) | -1.8 | -3.6  | —     |      |
|                   |  | 5V              |  | -3.6 | -7.3  | —     |      |
|                   |  | 3V              | V <sub>OH</sub> =0.9V <sub>DD</sub> ,<br>SLEDCn[m+1:m]=11B<br>(n=0, 1; m=0, 2, 4, 6) | -4   | -8    | —     |      |
|                   |  | 5V              |  | -8   | -16   | —     |      |
| R <sub>PH</sub>   | Pull-high Resistance for I/O Ports <small>(Note)</small> | 3V              | LVPU=0, PxPU=FFH (Px: PA, PB, PC, PD)  | 20   | 60    | 100   | kΩ   |
|                   |  | 5V              |  | 10   | 30    | 50    |      |
|                   |  | 3V              | LVPU=1, PxPU=FFH (Px: PA, PB, PC, PD)  | 6.67 | 15.00 | 23.00 | kΩ   |
|                   |  | 5V              |  | 3.5  | 7.5   | 12.0  |      |
| I <sub>LEAK</sub> | Input Leakage Current                                    | 5V              | V <sub>IN</sub> =V <sub>DD</sub> or V <sub>IN</sub> =V <sub>SS</sub>                 | —    | —     | ±1    | μA   |
| t <sub>TCK</sub>  | TM Clock Input Pin Minimum Pulse Width                   | —               | —  | 0.3  | —     | —     | μs   |
| t <sub>INT</sub>  | External Interrupt Minimum Pulse Width                   | —               | —  | 10   | —     | —     | μs   |

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

## Memory Characteristics

T<sub>a</sub>=-40°C~85°C, unless otherwise specified

| Symbol                      | Parameter  | Test Conditions |                      | Min.  | Typ.  | Max.  | Unit             |
|-----------------------------|--|-----------------|----------------------|-------|-------|-------|------------------|
|                             |  | V <sub>DD</sub> | Conditions           |       |       |       |                  |
| <b>Flash Program Memory</b> |  |                 |                      |       |       |       |                  |
| t <sub>FER</sub>            | ROM Erase Time                                     | —               | —                    | 2.273 | 2.500 | 2.778 | ms               |
|                             | IAP Erase Time                                     | —               | FWERTS=0             | —     | 3.2   | 3.9   | ms               |
|                             |  | —               | FWERTS=1             | —     | 3.7   | 4.5   |                  |
| t <sub>FWR</sub>            | ROM Write Time                                     | —               | —                    | 1.364 | 1.500 | 1.667 | ms               |
|                             | IAP Write Time                                     | —               | FWERTS=0             | —     | 2.2   | 2.7   | ms               |
|                             |  | —               | FWERTS=1             | —     | 3.0   | 3.6   |                  |
| E <sub>p</sub>              | Cell Endurance                                     | —               | —                    | 100K  | —     | —     | E/W              |
| t <sub>RETD</sub>           | ROM Data Retention Time                            | —               | T <sub>a</sub> =25°C | —     | 40    | —     | Year             |
| t <sub>ACTV</sub>           | ROM Activation Time – Wake-up from IDLE/SLEEP Mode | —               | —                    | 32    | —     | 64    | μs               |
| <b>Data EEPROM Memory</b>   |  |                 |                      |       |       |       |                  |
| t <sub>EERD</sub>           | EEPROM Read Time                                   | —               | —                    | —     | —     | 4     | t <sub>sys</sub> |
| t <sub>EEWR</sub>           | EEPROM Write Time (Byte Mode)                      | —               | EWERTS=0             | —     | 5.4   | 6.6   | ms               |
|                             |  | —               | EWERTS=1             | —     | 6.7   | 8.1   | ms               |
|                             | EEPROM Write Time (Page Mode)                      | —               | EWERTS=0             | —     | 2.2   | 2.7   | ms               |
|                             |  | —               | EWERTS=1             | —     | 3.0   | 3.6   | ms               |
| t <sub>EEER</sub>           | EEPROM Erase Time                                  | —               | EWERTS=0             | —     | 3.2   | 3.9   | ms               |
|                             |  | —               | EWERTS=1             | —     | 3.7   | 4.5   | ms               |

| Symbol                 | Parameter                  | Test Conditions |            | Min. | Typ. | Max. | Unit |
|------------------------|----------------------------|-----------------|------------|------|------|------|------|
|                        |                            | V <sub>DD</sub> | Conditions |      |      |      |      |
| E <sub>P</sub>         | Cell Endurance             | —               | —          | 100K | —    | —    | E/W  |
| t <sub>RET</sub>       | Data Retention Time        | —               | Ta=25°C    | —    | 40   | —    | Year |
| <b>RAM Data Memory</b> |                            |                 |            |      |      |      |      |
| V <sub>DR</sub>        | RAM Data Retention Voltage | —               | —          | 1    | —    | —    | V    |

Note: 1. “E/W” means Erase/Write times.

2. The ROM activation time t<sub>ACTV</sub> should be added when calculating the total system start-up time of a wake-up from the IDLE/SLEEP mode.

## A/D Converter Electrical Characteristics

Ta=-40°C~85°C

| Symbol           | Parameter                  | Test Conditions |   | Min. | Typ. | Max.             | Unit |
|------------------|----------------------------|-----------------|---|------|------|------------------|------|
|                  |                            | V <sub>DD</sub> | Conditions  |      |      |                  |      |
| AV <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  | —    | AV <sub>DD</sub> | V    |
| N <sub>R</sub>   | Resolution                 | —               | —   | —    | —    | 12               | Bit  |
| DNL              | Differential Non-linearity | 1.8V            | SAINS[3:0]=0000B,<br>SAVRS[1:0]=01B, V <sub>REF</sub> =AV <sub>DD</sub> ,<br>t <sub>ADCK</sub> =2μs   | -4   | —    | +4               | LSB  |
|                  |                            | 2V              | SAINS[3:0]=0000B,<br>SAVRS[1:0]=01B, V <sub>REF</sub> =AV <sub>DD</sub> ,<br>t <sub>ADCK</sub> =1μs   |      |      |                  |      |
|                  |                            | 1.8V            | SAINS[3:0]=0000B,<br>SAVRS[1:0]=01B, V <sub>REF</sub> =AV <sub>DD</sub> ,<br>t <sub>ADCK</sub> =10μs  | -3   | —    | +3               | LSB  |
|                  |                            | 3V              | SAINS[3:0]=0000B,<br>SAVRS[1:0]=01B, V <sub>REF</sub> =AV <sub>DD</sub> ,<br>t <sub>ADCK</sub> =0.5μs |      |      |                  |      |
|                  |                            | 5V              | SAINS[3:0]=0000B,<br>SAVRS[1:0]=01B, V <sub>REF</sub> =AV <sub>DD</sub> ,<br>t <sub>ADCK</sub> =10μs  |      |      |                  |      |
|                  |                            |                 | SAINS[3:0]=0000B,<br>SAVRS[1:0]=01B, V <sub>REF</sub> =AV <sub>DD</sub> ,<br>t <sub>ADCK</sub> =10μs  |      |      |                  |      |

| Symbol   | Parameter   | Test Conditions |   | Min.                  | Typ. | Max.                  | Unit              |
|--|---|-----------------|---|-----------------------|------|-----------------------|-------------------|
|  |   | V <sub>DD</sub> | Conditions  |                       |      |                       |                   |
| INL  | Integral Non-linearity                              | 1.8V            | SAINS[3:0]=0000B,<br>SAVRS[1:0]=01B, V <sub>REF</sub> =AV <sub>DD</sub> ,<br>t <sub>ADCK</sub> =2.0μs | -4                    | —    | +4                    | LSB               |
|  |   |                 | SAINS[3:0]=0000B,<br>SAVRS[1:0]=01B, V <sub>REF</sub> =AV <sub>DD</sub> ,<br>t <sub>ADCK</sub> =10μs  |                       |      |                       |                   |
|  |   | 2V              | SAINS[3:0]=0000B,<br>SAVRS[1:0]=01B, V <sub>REF</sub> =AV <sub>DD</sub> ,<br>t <sub>ADCK</sub> =1μs   |                       |      |                       |                   |
|  |   | 3V              | SAINS[3:0]=0000B,<br>SAVRS[1:0]=01B, V <sub>REF</sub> =AV <sub>DD</sub> ,<br>t <sub>ADCK</sub> =0.5μs |                       |      |                       |                   |
|  |   |                 | SAINS[3:0]=0000B,<br>SAVRS[1:0]=01B, V <sub>REF</sub> =AV <sub>DD</sub> ,<br>t <sub>ADCK</sub> =10μs  |                       |      |                       |                   |
|  |   | 5V              | SAINS[3:0]=0000B,<br>SAVRS[1:0]=01B, V <sub>REF</sub> =AV <sub>DD</sub> ,<br>t <sub>ADCK</sub> =0.5μs |                       |      |                       |                   |
| SAINS[3:0]=0000B,<br>SAVRS[1:0]=01B, V <sub>REF</sub> =AV <sub>DD</sub> ,<br>t <sub>ADCK</sub> =10μs |   |                 |   |                       |      |                       |                   |
| I <sub>ADC</sub>   | Additional Current for A/D Converter Enable         | 1.8V            | No load, t <sub>ADCK</sub> =2.0μs   | —                     | 280  | 400                   | μA                |
|  |   | 3V              | No load, t <sub>ADCK</sub> =0.5μs   | —                     | 450  | 600                   |                   |
|  |   | 5V              | No load, t <sub>ADCK</sub> =0.5μs   | —                     | 850  | 1000                  |                   |
| t <sub>ADCK</sub>  | Clock Period  | —               | 1.8V ≤ AV <sub>DD</sub> < 2.0V  | 2.0                   | —    | 10.0                  | μs                |
|  |   |                 | 2.0V ≤ AV <sub>DD</sub> ≤ 5.5V  | 0.5                   | —    | 10.0                  |                   |
| t <sub>ADS</sub>   | Sampling Time                                       | —               | —   | —                     | 4    | —                     | t <sub>ADCK</sub> |
| t <sub>ADC</sub>   | Conversion Time (Includes A/D Sample and Hold Time) | —               | —   | —                     | 16   | —                     | t <sub>ADCK</sub> |
| t <sub>ON2ST</sub>   | A/D Converter On-to-Start Time                      | —               | —   | 4                     | —    | —                     | μs                |
| I <sub>PGA</sub>   | Additional Current for PGA Enable                   | 2.2V            | No load, PGAIS=1,<br>PGAGS[1:0]=01B   | —                     | 250  | 500                   | μA                |
|  |   | 3V              | No load, PGAIS=1,<br>PGAGS[1:0]=01B   | —                     | 300  | 600                   | μA                |
|  |   | 5V              | No load, PGAIS=1,<br>PGAGS[1:0]=01B   | —                     | 400  | 700                   | μA                |
| V <sub>OR</sub>  | PGA Maximum Output Voltage Range                    | 2.2V            | —   | AV <sub>SS</sub> +0.1 | —    | AV <sub>DD</sub> -0.1 | V                 |
|  |   | 3V              | —   | AV <sub>SS</sub> +0.1 | —    | AV <sub>DD</sub> -0.1 | V                 |
|  |   | 5V              | —   | AV <sub>SS</sub> +0.1 | —    | AV <sub>DD</sub> -0.1 | V                 |
| V <sub>VR</sub>  | Fix Voltage Output of PGA                           | 2.2V~5.5V       | V <sub>RI</sub> =V <sub>BGREF</sub> (PGAIS=1)   | -1%                   | 2    | +1%                   | V                 |
|  |   | 3.2V~5.5V       | V <sub>RI</sub> =V <sub>BGREF</sub> (PGAIS=1)   | -1%                   | 3    | +1%                   | V                 |
|  |   | 4.2V~5.5V       | V <sub>RI</sub> =V <sub>BGREF</sub> (PGAIS=1)   | -1%                   | 4    | +1%                   | V                 |
| V <sub>IR</sub>  | PGA Input Voltage Range                             | 3V              | Gain=1, PGAIS=0,<br>Relative gain,<br>Gain error <±5%   | AV <sub>SS</sub> +0.1 | —    | AV <sub>DD</sub> -1.4 | V                 |
|  |   | 5V              | —   | AV <sub>SS</sub> +0.1 | —    | AV <sub>DD</sub> -1.4 | V                 |
| V <sub>OS_PGA</sub>  | PGA Input Offset Voltage                            | 3V              | —   | -15                   | —    | +15                   | mV                |
|  |   | 5V              | —   | -15                   | —    | +15                   |                   |

## 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 Detector Voltage           | —               | LVD enable, voltage select 1.8V                                      | -5%  | 1.8  | +5%  | V    |
|                      |  |                 | LVD enable, voltage select 2.0V                                      |      | 2.0  |      |      |
|                      |  |                 | LVD enable, voltage select 2.4V                                      |      | 2.4  |      |      |
|                      |  |                 | LVD enable, voltage select 2.7V                                      |      | 2.7  |      |      |
|                      |  |                 | LVD enable, voltage select 3.0V                                      |      | 3.0  |      |      |
|                      |  |                 | LVD enable, voltage select 3.3V                                      |      | 3.3  |      |      |
|                      |  |                 | LVD enable, voltage select 3.6V                                      |      | 3.6  |      |      |
| I <sub>LVR/LVD</sub> | Operating Current                      | 3V              | LVD enable, LVR enable, V <sub>LVR</sub> =1.9V, V <sub>LVD</sub> =2V | —    | —    | 10   | μA   |
|                      |  | 5V              | LVD enable, LVR enable, V <sub>LVR</sub> =1.9V, V <sub>LVD</sub> =2V | —    | 10   | 15   |      |
| t <sub>LVDS</sub>    | LVDO Stable Time                       | —               | For LVR enable, VBGEN=0, LVD off → on                                | —    | —    | 18   | μs   |
|                      |  |                 | For LVR disable, VBGEN=0, LVD off → on                               | —    | —    | 150  |      |
| 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  | —    | —    | 14   | μA   |
| I <sub>LVD</sub>     | Additional Current for LVD Enable      | 5V              | LVR disable  | —    | —    | 14   | μA   |

## Reference Voltage Electrical Characteristics

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

| Symbol             | Parameter                    | Test Conditions |  | Min. | Typ. | Max. | Unit              |
|--------------------|------------------------------|-----------------|--|------|------|------|-------------------|
|                    |                              | V <sub>DD</sub> | Conditions   |      |      |      |                   |
| V <sub>DD</sub>    | Operating Voltage            | —               | —  | 1.8  | —    | 5.5  | V                 |
| V <sub>BGREF</sub> | Bandgap Reference Voltage    | 1.8V~<2.2V      | —  | -10% | 1.2  | +10% | V                 |
|                    |                              | 2.2V~5.5V       | —  | -1%  | 1.2  | +1%  |                   |
| I <sub>BGREF</sub> | Operating Current            | 5.5V            | —  | —    | 25   | 35   | μA                |
| PSRR               | Power Supply Rejection Ratio | —               | Ta=25°C, V <sub>RIPPLE</sub> =1V <sub>P-P</sub> , f <sub>RIPPLE</sub> =100Hz | 75   | —    | —    | dB                |
| En                 | Output Noise                 | —               | Ta=25°C, No load current, f=0.1Hz~10Hz                                       | —    | 300  | —    | μV <sub>RMS</sub> |
| I <sub>SD</sub>    | Shutdown Current             | —               | VBGREN=0   | —    | —    | 0.1  | μA                |
| t <sub>START</sub> | Startup Time                 | 1.8V~5.5V       | Ta=25°C  | —    | —    | 400  | μs                |

 Note: The V<sub>BGREF</sub> voltage is used as the A/D converter PGA input signal.

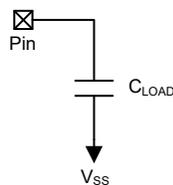
## Comparator Electrical Characteristics

Ta=25°C

| Symbol            | Parameter                                | Test Conditions |  | Min.            | Typ. | Max.               | Unit |
|-------------------|--|-----------------|--|-----------------|------|--------------------|------|
|                   |  | V <sub>DD</sub> | Conditions                             |                 |      |                    |      |
| I <sub>CMP</sub>  | Additional Current for Comparator Enable | 3V              | —                                      | —               | 70   | 90                 | μA   |
|                   |  | 5V              | —                                      | —               | 80   | 100                |      |
| I <sub>CSTB</sub> | Power Down Current                       | 3V              | Comparator disable                     | —               | —    | 0.1                | μA   |
|                   |  | 5V              |  | —               | —    | 0.1                |      |
| V <sub>CM</sub>   | Common Mode Voltage Range                | 3V              | —                                      | V <sub>SS</sub> | —    | V <sub>DD</sub> -1 | V    |
|                   |  | 5V              | —                                      | V <sub>SS</sub> | —    | V <sub>DD</sub> -1 | V    |
| A <sub>OL</sub>   | Open Loop Gain                           | 3V              | —                                      | 60              | 80   | —                  | dB   |
|                   |  | 5V              | —                                      | 60              | 80   | —                  |      |
| V <sub>HYS</sub>  | Hysteresis                               | 3V              | CMPHYEN=0                              | 0               | 0    | 5                  | mV   |
|                   |  | 5V              |  | 0               | 0    | 5                  |      |
|                   |  | 3V              | CMPHYEN=1                              | 5               | 24   | 30                 | mV   |
|                   |  | 5V              |  | 5               | 24   | 30                 |      |
| t <sub>RP</sub>   | Response Time                            | 3V              | With 100mV overdrive <sup>(1)(2)</sup> | —               | 200  | 400                | ns   |
|                   |  | 5V              |  | —               | 200  | 400                |      |

- Note: 1. All measurement is under the condition where the comparator positive input voltage is equal to (V<sub>DD</sub>-1)/2 and remains constant.  
 2. Measured with comparator one input pin at V<sub>CM</sub>=(V<sub>DD</sub>-1.0)/2 while the other pin input transition from V<sub>SS</sub> to (V<sub>CM</sub>+100mV) or from (V<sub>DD</sub>-1V) to (V<sub>CM</sub>-100mV).  
 3. Load Condition: C<sub>LOAD</sub>=50pF.

Load Condition



## Software Controlled LCD Driver Electrical Characteristics

Ta=-40°C~85°C

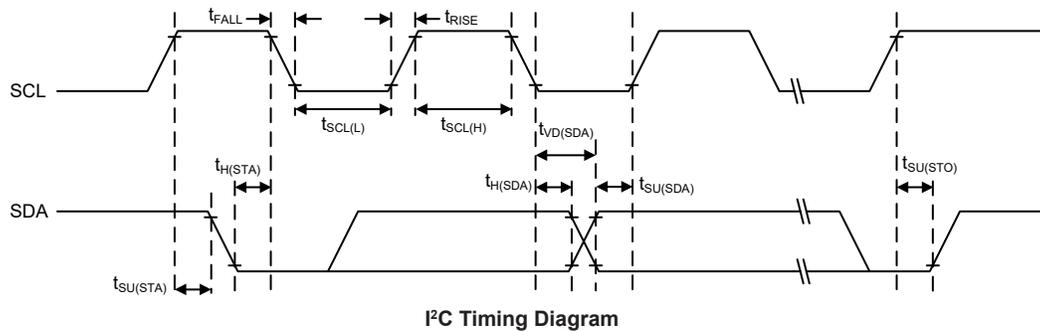
| Symbol            | Parameter   | Test Conditions |               | Min.                 | Typ.                 | Max.                 | Unit |
|-------------------|---|-----------------|---------------|----------------------|----------------------|----------------------|------|
|                   |   | V <sub>DD</sub> | Conditions    |                      |                      |                      |      |
| I <sub>BIAS</sub> | Bias Current  | 5V              | ISEL[1:0]=00B | 5.81                 | 8.30                 | 10.79                | μA   |
|                   |   |                 | ISEL[1:0]=01B | 11.62                | 16.60                | 21.58                |      |
|                   |   |                 | ISEL[1:0]=10B | 35                   | 50                   | 65                   |      |
|                   |   |                 | ISEL[1:0]=11B | 70                   | 100                  | 130                  |      |
| V <sub>SCOM</sub> | V <sub>DD</sub> ×2/3 Voltage for LCD SCOM/SSEG Output | 2.2V~5.5V       | No load       | 0.305V <sub>DD</sub> | 0.330V <sub>DD</sub> | 0.355V <sub>DD</sub> | V    |
|                   | V <sub>DD</sub> ×1/3 Voltage for LCD SCOM/SSEG Output | 2.2V~5.5V       | No load       | 0.31V <sub>DD</sub>  | 0.33V <sub>DD</sub>  | 0.35V <sub>DD</sub>  | V    |

## I<sup>2</sup>C Electrical Characteristics

Ta=25°C

| Symbol               | Parameter  | Test Conditions |                         | Min. | Typ. | Max. | Unit |
|----------------------|--|-----------------|-------------------------|------|------|------|------|
|                      |  | V <sub>DD</sub> | Conditions              |      |      |      |      |
| f <sub>I2C</sub>     | I <sup>2</sup> C Standard Mode (100kHz) f <sub>sys</sub> Frequency <small>(Note)</small> | —               | No clock debounce       | 2    | —    | —    | MHz  |
|                      |  |                 | 2 system clock debounce | 4    | —    | —    |      |
|                      |  |                 | 4 system clock debounce | 4    | —    | —    |      |
|                      | I <sup>2</sup> C Fast Mode (400kHz) f <sub>sys</sub> Frequency <small>(Note)</small>     | —               | No clock debounce       | 4    | —    | —    | MHz  |
|                      |  |                 | 2 system clock debounce | 8    | —    | —    |      |
|                      |  |                 | 4 system clock debounce | 8    | —    | —    |      |
| f <sub>SCL</sub>     | SCL Clock Frequency  | 3V/5V           | Standard mode           | —    | —    | 100  | kHz  |
|                      |  |                 | Fast mode               | —    | —    | 400  |      |
| t <sub>SCL(H)</sub>  | SCL Clock High Time  | 3V/5V           | Standard mode           | 3.5  | —    | —    | μs   |
|                      |  |                 | Fast mode               | 0.9  | —    | —    |      |
| t <sub>SCL(L)</sub>  | SCL Clock Low Time   | 3V/5V           | Standard mode           | 3.5  | —    | —    | μs   |
|                      |  |                 | Fast mode               | 0.9  | —    | —    |      |
| t <sub>FALL</sub>    | SCL and SDA Fall Time  | 3V/5V           | Standard mode           | —    | —    | 1.3  | μs   |
|                      |  |                 | Fast mode               | —    | —    | 0.34 |      |
| t <sub>RISE</sub>    | SCL and SDA Rise Time  | 3V/5V           | Standard mode           | —    | —    | 1.3  | μs   |
|                      |  |                 | Fast mode               | —    | —    | 0.34 |      |
| t <sub>SU(SDA)</sub> | SDA Data Setup Time  | 3V/5V           | Standard mode           | 0.25 | —    | —    | μs   |
|                      |  |                 | Fast mode               | 0.1  | —    | —    |      |
| t <sub>H(SDA)</sub>  | SDA Data Hold Time   | 3V/5V           | —                       | 0.1  | —    | —    | μs   |
| t <sub>VD(SDA)</sub> | SDA Data Valid Time  | 3V/5V           | —                       | —    | —    | 0.6  | μs   |
| t <sub>SU(STA)</sub> | Start Condition Setup Time   | 3V/5V           | Standard mode           | 3.5  | —    | —    | μs   |
|                      |  |                 | Fast mode               | 0.6  | —    | —    |      |
| t <sub>H(STA)</sub>  | Start Condition Hold Time  | 3V/5V           | —                       | 0.6  | —    | —    | μs   |
| t <sub>SU(STO)</sub> | Stop Condition Setup Time  | 3V/5V           | Standard mode           | 3.5  | —    | —    | μs   |
|                      |  |                 | Fast mode               | 0.6  | —    | —    |      |

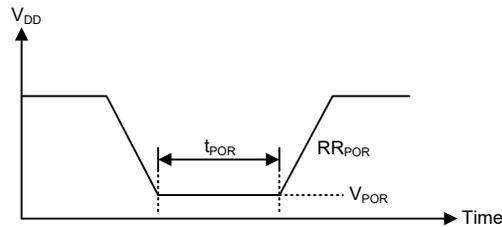
Note: Using the debounce function can make the transmission more stable and reduce the probability of communication failure due to interference.



## Power-on Reset Characteristics

Ta=-40°C~85°C

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



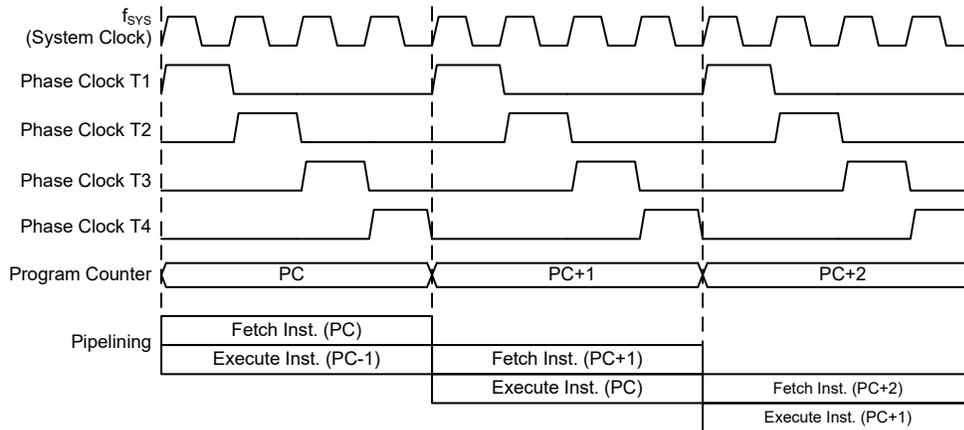
## System Architecture

A key factor in the high-performance features of the Holtek range of microcontrollers is attributed to their internal system architecture. The device takes advantage of the usual features found within RISC microcontrollers providing increased speed of operation and enhanced performance. The pipelining scheme is implemented in such a way that instruction fetching and instruction execution are overlapped, hence instructions are effectively executed in one or two cycles for most of the standard or extended instructions respectively. The exceptions to these 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.

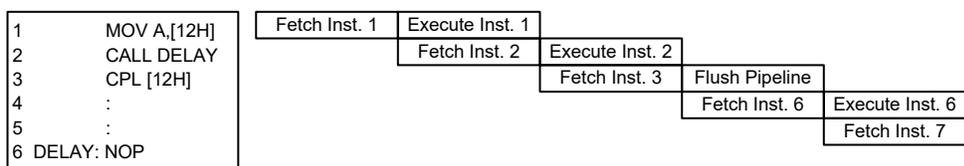
### Clocking and Pipelining

The main system clock, derived from either an 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.

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



**System Clocking and Pipelining**



**Instruction Fetching**

### Program Counter

During program execution, the Program Counter is used to keep track of the address of the next instruction to be executed. It is automatically incremented by one each time an instruction is executed except for instructions, such as “JMP” or “CALL” that demands a jump to a non-consecutive Program Memory address. For the device with a Program Memory capacity in excess of 8K words, the Program Memory high byte address must be setup by selecting a certain program memory bank which is implemented using the program memory bank pointer bit, PBP0. Only the lower 8 bits, known as the Program Counter Low Register, are directly addressable by the application program.

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

| Program Counter |                |
|-----------------|----------------|
| High Byte       | Low Byte (PCL) |
| PBP0, PC12~PC8  | PCL7~PCL0      |

**Program Counter**

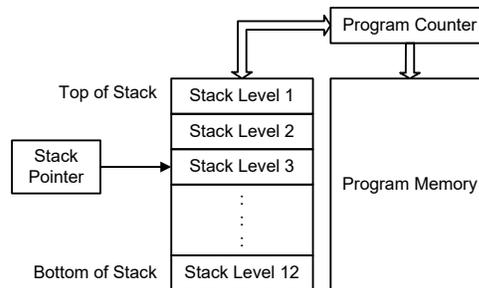
The lower byte of the Program Counter, known as the Program Counter Low register or PCL, is available for program control and is a readable and writable register. By transferring data directly into this register, a short program jump can be executed directly. However, as only this low byte is available 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 12 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:

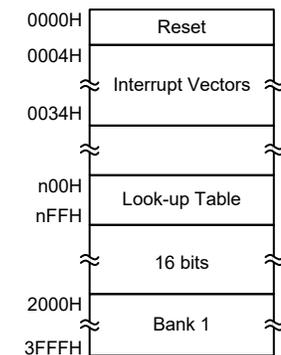
- Arithmetic operations:  
 ADD, ADDM, ADC, ADCM, SUB, SUBM, SBC, SBCM, DAA,  
 LADD, LADDM, LADC, LADCM, LSUB, LSUBM, LSBC, LSBCM, LDAA
- Logic operations:  
 AND, OR, XOR, ANDM, ORM, XORM, CPL, CPLA,  
 LAND, LOR, LXOR, LANDM, LORM, LXORM, LCPL, LCPLA
- Rotation:  
 RRA, RR, RRCA, RRC, RLA, RL, RLCA, RLC,  
 LRR, LRRCA, LRRCA, LRRCA, LRRCA, LRRCA, LRRCA, LRRCA
- Increment and Decrement:  
 INCA, INC, DECA, DEC,  
 LINCA, LINC, LDECA, LDEC
- Branch decision:  
 JMP, SZ, SZA, SNZ, SIZ, SDZ, SIZA, SDZA, CALL, RET, RETI,  
 LSZ, LSZA, LSNZ, LSIZ, LSDZ, LSIZA, 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 16K×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 arranged in any location within the Program Memory, is addressed by a separate table pointer register.



### Special Vectors

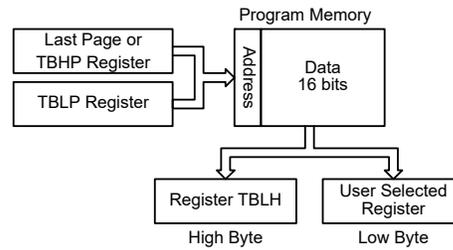
Within the Program Memory, certain locations are reserved for the reset and interrupts. The location 0000H is reserved for use by the device reset for program initialisation. After a device reset is initiated, the program will jump to this location and begin execution.

### 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 configured by placing the address of the look up data to be retrieved in the table pointer registers, TBLP and TBHP. These registers define the total address of the look-up table.

After setting up the table pointer, the table data can be retrieved from the Program Memory using the 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 except Sector 0, 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. Any unused bits in this transferred higher order byte will be read as “0”.

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 is located in ROM Bank 1 and refers to the start address of the last page within the 16K words Program Memory of the device. The table pointer low byte register is set 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 “3F06H” or 6 locations after the start of the last page. Note that the value for the table pointer is referenced to the specific address pointed by the TBLP and TBHP registers if the “TABRD [m]” or “LTABRD [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]” or “LTABRD [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

```

rombank 1 code1
ds .section 'data'
tempreg1 db ? ; temporary register #1
tempreg2 db ? ; temporary register #2
:
:
mov a,06h      ; initialise low table pointer - note that this address is referenced
mov tblp,a    ; to the last page or the page that tbhp pointed
mov a,3Fh     ; initialise high table pointer
mov tbhp,a
:
:
tabrd tempreg1 ; transfers value in table referenced by table pointer,
               ; data at program memory address "3F06H" 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 "3F05H" transferred to tempreg2 and TBLH
               ; in this example the data "1AH" is transferred to tempreg1 and data "0FH"
               ; to register tempreg2
               ; the value "00H" will be transferred to the high byte register TBLH
:
:

```

```
code1 .section 'code'
org 1F00h      ; sets initial address of program memory
dc 00Ah, 00Bh, 00Ch, 00Dh, 00Eh, 00Fh, 01Ah, 01Bh
:
```

### In Circuit Programming – ICP

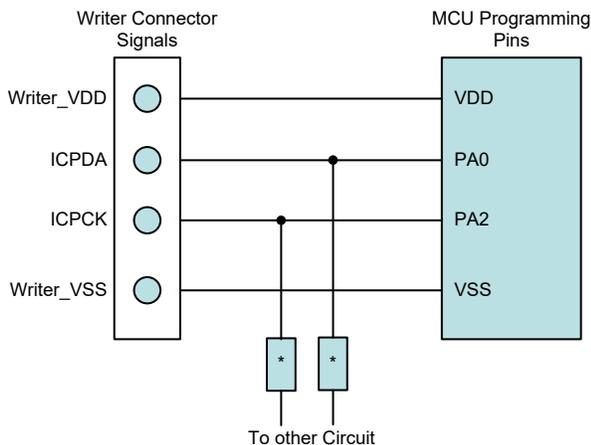
The provision of Flash type Program Memory provides the user with a means of convenient and easy upgrades and modifications to their programs on the same device.

As an additional convenience, a means of programming the microcontroller in-circuit has provided using a 4-pin interface. This provides manufacturers with the possibility of manufacturing their circuit boards complete with a programmed or un-programmed microcontroller, and then programming or upgrading the program at a later stage. This enables product manufacturers to easily keep their manufactured products supplied with the latest program releases without removal and re-insertion of the device.

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

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

During the programming process, the user can take care 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 HT66V31A5 which is used to emulate the real MCU device named HT66F31A5. The EV chip device also provides an “On-Chip Debug” function to debug the real MCU device during the development process. The EV chip and the real MCU device are almost functionally compatible except for “On-Chip Debug” function. Users can use the EV chip device to emulate the real chip device behavior by connecting the OCDSDA and OCDSCK pins to the Holtek HT-IDE development tools. The OCDSDA pin is the OCDS Data/Address input/output pin while the OCDSCK pin is the OCDS clock input pin. When users use the EV chip for debugging, other functions which are shared with the OCDSDA and OCDSCK pins in the device will have no effect in the EV chip. However, the two OCDS pins which are pin-shared with the ICP programming pins are still used as the Flash Memory programming pins for ICP. For more detailed OCDS information, refer to the corresponding document named “Holtek e-Link for 8-bit MCU OCDS User’s Guide”.

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

### In Application Programming – IAP

Flash type Program Memory provides the user with a means of convenient and easy upgrades and modifications to their programs on the same device. The provision of the IAP function offers users the convenience of Flash Memory multi-programming features. The convenience of the IAP function is that it can execute the updated program procedure using its internal firmware, without requiring an external Program Writer or PC. In addition, the IAP interface can also be any type of communication protocol, such as UART, using I/O pins. Regarding the internal firmware, the user can select versions provided by Holtek or create their own. The following section illustrates the procedures regarding how to implement the IAP firmware.

#### Flash Memory Read/Write Size

The Flash memory Erase and Write operations are carried out in a page format while the Read operation is carried out in a word format. The page size and write buffer size are both assigned with a capacity of 32 words. Note that the Erase operation should be executed before the Write operation is executed.

When the Flash Memory Erase/Write Function is successfully enabled, the CFWEN bit will be set high. When the CFWEN bit is set high, the data can be written into the write buffer. The FWT bit is used to initiate the write process and then indicate the write operation status. This bit is set high by application programs to initiate a write process and will be cleared by hardware if the write process is finished.

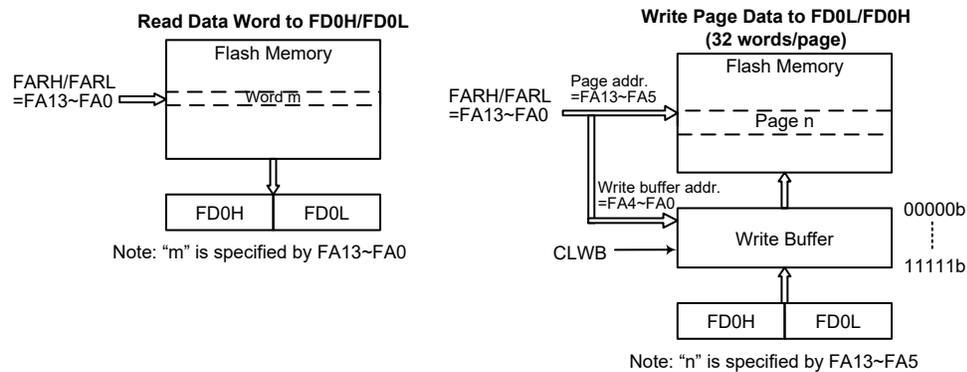
The Read operation can be carried out by executing a specific read procedure. The FRDEN bit is used to enable the read function and the FRD bit is used to initiate the read process by application programs and then indicate the read operation status. When the read process is finished, this bit will be cleared by hardware.

| Operations                                  | Format        |
|---|---------------|
| Erase                                       | 32 words/page |
| Write                                       | 32 words/time |
| Read  | 1 word/time   |
| Note: Page size=Write buffer size=32 words. |               |

**IAP Operation Format**

| Page | FARH      | FARL[7:5] | FARL[4:0]   |
|------|-----------|-----------|-------------|
| 0    | 0000 0000 | 000       | Tag Address |
| 1    | 0000 0000 | 001       |             |
| 2    | 0000 0000 | 010       |             |
| 3    | 0000 0000 | 011       |             |
| 4    | 0000 0000 | 100       |             |
| 5    | 0000 0000 | 101       |             |
| 6    | 0000 0000 | 110       |             |
| 7    | 0000 0000 | 111       |             |
| 8    | 0000 0001 | 000       |             |
| ⋮    | ⋮         | ⋮         |             |
| 510  | 0011 1111 | 110       |             |
| 511  | 0011 1111 | 111       |             |

**Page Number and Address Selection**



**Flash Memory IAP Read/Write Structure**

**Write Buffer**

The write buffer is used to store the written data temporarily when executing the write operation. The Write Buffer can be filled with written data after the Flash Memory Erase/Write Function has been successfully enabled by executing the Flash Memory Erase/Write Function Enable procedure. The write buffer can be cleared by configuring the CLWB bit in the FC2 register. The CLWB bit can be set high to enable the Clear Write Buffer procedure. When the procedure is finished this bit will be cleared to low by the hardware. It is recommended that the write buffer should be cleared by setting the CLWB bit high before the write buffer is used for the first time or when the data in the write buffer is updated.

The write buffer size is 32 words corresponding to a page. The write buffer address is mapped to a specific flash memory page specified by the memory address bits, FA13~FA5. The data written into the FD0L and FD0H registers will be loaded into the write buffer. When data is written into the high byte data register, FD0H, it will result in the data stored in the high and low byte data registers both being written into the write buffer. It will also cause the flash memory address to be incremented by one, after which the new address will be loaded into the FARH and FARL address registers. When the flash memory address reaches the page boundary, 11111b of a page with 32 words, the address will now not be incremented but will stop at the last address of the page. At this point a new page address should be specified for any other erase/write operations.

After a write process is finished, the write buffer will automatically be cleared by the hardware. Note that the write buffer should be cleared manually by the application program when the data written into the flash memory is incorrect in the data verification step. The data should again be written into the write buffer after the write buffer has been cleared when the data is found to be incorrect during the data verification step.

### IAP Flash Program Memory Registers

There are two address registers, four 16-bit data registers and three control registers. All the registers are located in Sector 0. Read and Write operations to the Flash memory are carried out by 16-bit data operations using the address and data registers and the control register. Several registers control the overall operation of the internal Flash Program Memory. The address registers are named FARL and FARH, the data registers are named FDnL and FDnH and the control registers are named FC0, FC1 and FC2.

| Register Name | Bit   |       |       |       |       |      |        |      |
|---------------|-------|-------|-------|-------|-------|------|--------|------|
|               | 7     | 6     | 5     | 4     | 3     | 2    | 1      | 0    |
| FC0           | CFWEN | FMOD2 | FMOD1 | FMOD0 | FWPEN | FWT  | FRDEN  | FRD  |
| FC1           | D7    | D6    | D5    | D4    | D3    | D2   | D1     | D0   |
| FC2           | —     | —     | —     | —     | —     | —    | FWERTS | CLWB |
| FARL          | FA7   | FA6   | FA5   | FA4   | FA3   | FA2  | FA1    | FA0  |
| FARH          | —     | —     | FA13  | FA12  | FA11  | FA10 | FA9    | FA8  |
| FD0L          | D7    | D6    | D5    | D4    | D3    | D2   | D1     | D0   |
| FD0H          | D15   | D14   | D13   | D12   | D11   | D10  | D9     | D8   |
| FD1L          | D7    | D6    | D5    | D4    | D3    | D2   | D1     | D0   |
| FD1H          | D15   | D14   | D13   | D12   | D11   | D10  | D9     | D8   |
| FD2L          | D7    | D6    | D5    | D4    | D3    | D2   | D1     | D0   |
| FD2H          | D15   | D14   | D13   | D12   | D11   | D10  | D9     | D8   |
| FD3L          | D7    | D6    | D5    | D4    | D3    | D2   | D1     | D0   |
| FD3H          | D15   | D14   | D13   | D12   | D11   | D10  | D9     | D8   |

IAP Register List

#### • FARL Register

| Bit  | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   |
|------|-----|-----|-----|-----|-----|-----|-----|-----|
| Name | FA7 | FA6 | FA5 | FA4 | FA3 | FA2 | FA1 | FA0 |
| 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      **FA7~FA0**: Flash Memory Address bit 7 ~ bit 0

#### • FARH Register

| Bit  | 7 | 6 | 5    | 4    | 3    | 2    | 1   | 0   |
|------|---|---|------|------|------|------|-----|-----|
| Name | — | — | FA13 | FA12 | FA11 | FA10 | FA9 | FA8 |
| 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      **FA13~FA8**: Flash Memory Address bit 13 ~ bit 8

• **FD0L Register**

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

Bit 7~0      **D7~D0:** The first Flash Memory data bit 7 ~ bit 0

Note that data written into the low byte data register FD0L will only be stored in the FD0L register and not loaded into the lower 8-bit write buffer.

• **FD0H 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      **D15~D8:** The first Flash Memory data bit 15 ~ bit 8

Note that when the 8-bit data is written into the high byte data register FD0H, the whole 16 bits of data stored in the FD0H and FD0L registers will simultaneously be loaded into the 16-bit write buffer after which the contents of the Flash memory address register pair, FARH and FARL, will be incremented by one.

• **FD1L Register**

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

Bit 7~0      **D7~D0:** The second Flash Memory data bit 7 ~ bit 0

• **FD1H 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      **D15~D8:** The second Flash Memory data bit 15 ~ bit 8

• **FD2L Register**

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

Bit 7~0      **D7~D0:** The third Flash Memory data bit 7 ~ bit 0

• **FD2H 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      **D7~D0:** The third Flash Memory data bit 15 ~ bit 8

• **FD3L Register**

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

Bit 7~0      **D7~D0**: The fourth Flash Memory data bit 7 ~ bit 0

• **FD3H 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      **D15~D8**: The fourth Flash Memory data bit 15 ~ bit 8

• **FC0 Register**

| Bit  | 7     | 6     | 5     | 4     | 3     | 2   | 1     | 0   |
|------|-------|-------|-------|-------|-------|-----|-------|-----|
| Name | CFWEN | FMOD2 | FMOD1 | FMOD0 | FWPEN | FWT | FRDEN | FRD |
| 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      **CFWEN**: Flash Memory Erase/Write function enable control

0: Flash Memory erase/write function is disabled

1: Flash Memory erase/write function has been successfully enabled

When this bit is cleared to zero by application program, the Flash Memory erase/write function is disabled. Note that this bit cannot be set high by application programs. Writing “1” into this bit results in no action. This bit is used to indicate that the Flash Memory erase/write function status. When this bit is set high by hardware, it means that the Flash Memory erase/write function is enabled successfully. Otherwise, the Flash Memory erase/write function is disabled as the bit content is zero.

Bit 6~4      **FMOD2~FMOD0**: Flash Memory Mode selection

000: Write Mode

001: Page Erase Mode

010: Reserved

011: Read Mode

100: Reserved

101: Reserved

110: Flash Memory Erase/Write function Enable Mode

111: Reserved

These bits are used to select the Flash Memory operation modes. Note that the “Flash memory Erase/Write function Enable Mode” should first be successfully enabled before the Erase or Write Flash memory operation is executed.

Bit 3      **FWPEN**: Flash Memory Erase/Write function enable procedure trigger

0: Erase/Write function enable procedure is not triggered or procedure timer times out

1: Erase/Write function enable procedure is triggered and procedure timer starts to count

This bit is used to activate the flash memory Erase/Write function enable procedure and an internal timer. It is set by the application programs and then cleared to zero by the hardware when the internal timer times out. The correct patterns must be written into the FD1L/FD1H, FD2L/FD2H and FD3L/FD3H register pairs respectively as soon as possible after the FWPEN bit is set high.

- Bit 2      **FWT**: Flash Memory write initiate control  
             0: Do not initiate Flash Memory write or indicating that a Flash Memory write process has completed  
             1: Initiate a Flash Memory write process  
 This bit is set by software and cleared to zero by the hardware when the Flash memory write process has completed.
- Bit 1      **FRDEN**: Flash Memory read enabled bit  
             0: Flash Memory read disable  
             1: Flash Memory read enable  
 This is the Flash memory Read Enable bit which must be set high before any Flash memory read operations are carried out. Clearing this bit to zero will inhibit Flash memory read operations.
- Bit 0      **FRD**: Flash Memory read control bit  
             0: Do not initiate Flash Memory read or indicating that a Flash Memory read process has completed  
             1: Initiate a Flash Memory read process  
 This bit is set by software and cleared to zero by the hardware when the Flash memory read process has completed.

- Note: 1. The FWT, FRDEN and FRD bits cannot be set to “1” at the same time with a single instruction.  
 2. Ensure that the  $f_{SUB}$  clock is stable before executing the erase or write operation.  
 3. Note that the CPU will be stopped when a read, erase or write operation is successfully activated.  
 4. Ensure that the read, erase or write operation is totally complete before executing other operations.

• **FC1 Register**

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

- Bit 7~0      **D7~D0**: Chip Reset Pattern  
 When a specific value of “55H” is written into this register, a reset signal will be generated to reset the whole chip.

• **FC2 Register**

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

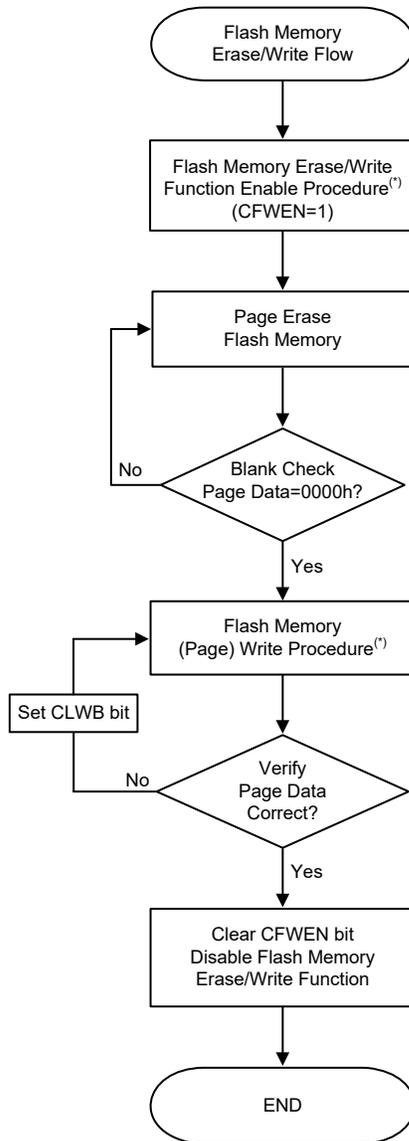
- Bit 7~2      Unimplemented, read as “0”
- Bit 1      **FWERTS**: Erase time and Write time selection  
             0: Erase time is 3.2ms ( $t_{FER}$ ) / Write time is 2.2ms ( $t_{FWR}$ )  
             1: Erase time is 3.7ms ( $t_{FER}$ ) / Write time is 3.0ms ( $t_{FWR}$ )
- Bit 0      **CLWB**: Flash Memory Write buffer clear control  
             0: Do not initiate a Write Buffer Clear process or indicating that a Write Buffer Clear process has completed  
             1: Initiate a Write Buffer Clear process  
 This bit is set by software and cleared to zero by hardware when the Write Buffer Clear process has completed.

### **Flash Memory Erase/Write Flow**

It is important to understand the Flash memory Erase/Write flow before the Flash memory contents are updated. Users can refer to the corresponding operation procedures when developing their IAP program to ensure that the flash memory contents are correctly updated.

### **Flash Memory Erase/Write Flow Descriptions**

1. Activate the “Flash Memory Erase/Write function enable procedure” first. When the Flash Memory Erase/Write function is successfully enabled, the CFWEN bit in the FC0 register will automatically be set high by hardware. After this, Erase or Write operations can be executed on the Flash memory. Refer to the “Flash Memory Erase/Write Function Enable Procedure” for details.
2. Configure the flash memory address to select the desired erase page, tag address and then erase this page. For a page erase operation, set the FARL and FARH registers to specify the start address of the erase page, then write dummy data into the FD0H register to tag address. The current address will be internally incremented by one after each dummy data is written into the FD0H register. When the address reaches the page boundary, 11111b, the address will not be further incremented but stop at the last address of the page. Note that the write operation to the FD0H register is used to tag address, it must be implemented to determine which addresses to be erased.
3. Execute a Blank Check operation to ensure whether the page erase operation is successful or not. The “TABRD” instruction should be executed to read the flash memory contents and to check if the contents is 0000h or not. If the flash memory page erase operation fails, users should go back to Step 2 and execute the page erase operation again.
4. Write data into the specific page. Refer to the “Flash Memory Write Procedure” for details.
5. Execute the “TABRD” instruction to read the flash memory contents and check if the written data is correct or not. If the data read from the flash memory is different from the written data, it means that the page write operation has failed. The CLWB bit should be set high to clear the write buffer and then write the data into the specific page again if the write operation has failed.
6. Clear the CFWEN bit to disable the Flash Memory Erase/Write function enable mode if the current page Erase and Write operations are completed and no more pages need to be erased or written.



**Flash Memory Erase/Write Flow**

Note: \* The Flash Memory Erase/Write Function Enable procedure and Flash Memory Write procedure will be described in the following sections.

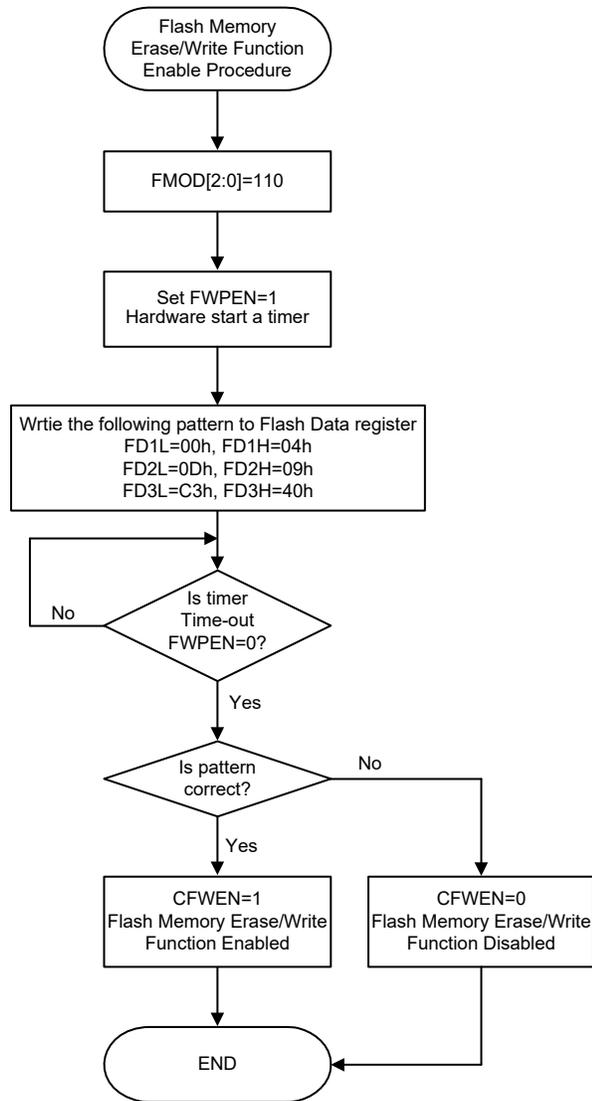
### **Flash Memory Erase/Write Function Enable Procedure**

The Flash Memory Erase/Write Function Enable Mode is specially designed to prevent the flash memory contents from being wrongly modified. In order to allow users to change the Flash memory data using the IAP control registers, users must first enable the Flash memory Erase/Write function.

### **Flash Memory Erase/Write Function Enable Procedure Description**

1. Write data “110” to the FMOD [2:0] bits in the FC0 register to select the Flash Memory Erase/Write Function Enable Mode.
2. Set the FWPEN bit in the FC0 register to “1” to activate the Flash Memory Erase/Write Enable Function. This will also activate an internal timer.
3. Write the correct data pattern into the Flash data registers, FD1L~FD3L and FD1H~FD3H, as soon as possible after the FWPEN bit is set high. The data pattern to enable Flash memory erase/write function is 00H, 0DH, C3H, 04H, 09H and 40H corresponding to the FD1L~FD3L and FD1H~FD3H registers respectively.
4. Once the timer has timed out, the FWPEN bit will automatically be cleared to zero by hardware regardless of the input data pattern.
5. If the written data pattern is incorrect, the Flash memory erase/write function will not be enabled successfully and the above steps should be repeated. If the written data pattern is correct, the Flash memory erase/write function will be enabled successfully.
6. Once the Flash memory erase/write function is enabled, the Flash memory contents can be updated by executing the page erase and write operations using the IAP control registers.

To disable the Flash memory erase/write function, the CFWEN bit in the FC0 register can be cleared. There is no need to execute the above procedure.



Flash Memory Erase/Write Function Enable Procedure

### **Flash Memory Write Procedure**

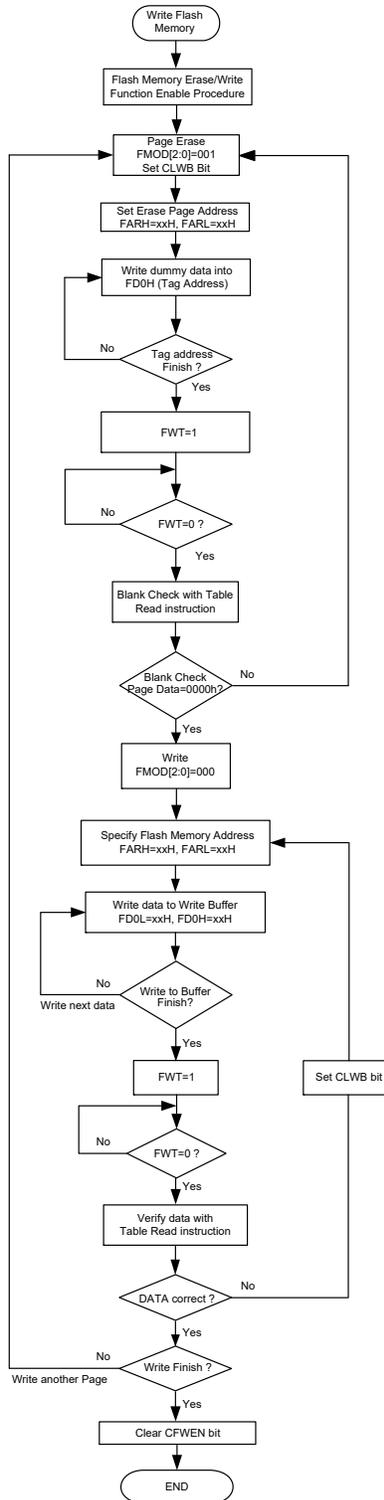
After the Flash memory erase/write function has been successfully enabled as the CFWEN bit is set high, the data to be written into the flash memory can be loaded into the write buffer. The selected flash memory page data should be erased by properly configuring the IAP control registers before the data write procedure is executed.

The write buffer size is 32 words, known as a page, whose address is mapped to a specific flash memory page specified by the memory address bits, FA13~FA5. It is important to ensure that the page where the write buffer data is located is the same one which the memory address bits, FA13~FA5, specify.

### **Flash Memory Consecutive Write Description**

The maximum amount of write data is 32 words for each write operation. The write buffer address will be automatically incremented by one when consecutive write operations are executed. The start address of a specific page should first be written into the FARL and FARH registers. Then the data word should first be written into the FD0L register and then the FD0H register. At the same time the write buffer address will be incremented by one and then the next data word can be written into the FD0L and FD0H registers for the next address without modifying the address register pair, FARH and FARL. When the write buffer address reaches the page boundary the address will not be further incremented but will stop at the last address of the page.

1. Activate the “Flash Memory Erase/Write function enable procedure”. Check the CFWEN bit value and then execute the erase/write operations if the CFWEN bit is set high. Refer to the “Flash Memory Erase/Write function enable procedure” for more details.
2. Set the FMOD2~FMOD0 to “001” to select the erase operation and set the CLWB bit high to clear the write buffer. Set the FWT bit high to erase the desired page which is specified by the FARH and FARL registers and has been tagged address. Wait until the FWT bit goes low.
3. Execute a Blank Check operation using the table read instruction to ensure that the erase operation has successfully completed.  
Go to step 2 if the erase operation is not successful.  
Go to step 4 if the erase operation is successful.
4. Set the FMOD2~FMOD0 to “000” to select the write operation.
5. Setup the desired start address in the FARH and FARL registers. Write the desired data words consecutively into the FD0L and FD0H registers within a page as specified by their consecutive addresses. The maximum written data number is 32 words.
6. Set the FWT bit high to write the data words from the write buffer to the flash memory. Wait until the FWT bit goes low.
7. Verify the data using the table read instruction to ensure that the write operation has successfully completed.  
If the write operation has not successfully completed, set the CLWB bit high to clear the write buffer and then go to step 5.  
Go to step 8 if the write operation is successful.
8. Clear the CFWEN bit low to disable the Flash memory erase/write function.



**Flash Memory Consecutive Write Procedure**

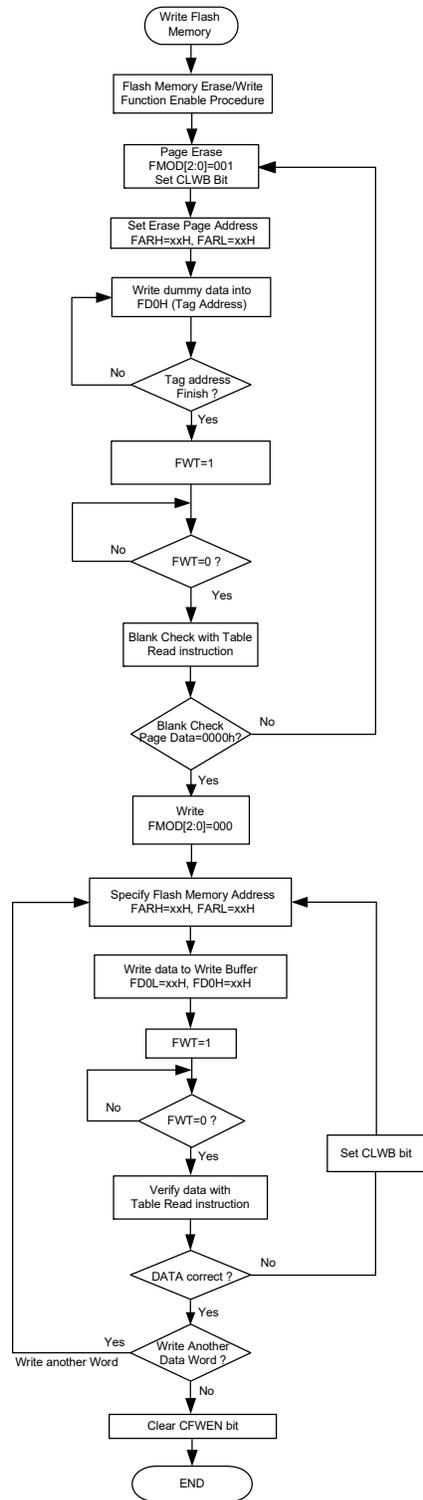
- Note: 1. When the erase or write operation is successfully activated, all CPU operations will temporarily cease.  
 2. It will take certain time for the FWT bit state changing from high to low in the erase or write operation, which can be selected by the FWERTS bit in the FC2 register.

### **Flash Memory Non-Consecutive Write Description**

The main difference between Flash Memory Consecutive and Non-Consecutive Write operations is whether the data words to be written are located in consecutive addresses or not. If the data to be written is not located in consecutive addresses the desired address should be re-assigned after a data word is successfully written into the Flash Memory.

A two data word non-consecutive write operation is taken as an example here and described as follows:

1. Activate the “Flash Memory Erase/Write function enable procedure”. Check the CFWEN bit value and then execute the erase/write operation if the CFWEN bit is set high. Refer to the “Flash Memory Erase/Write function enable procedure” for more details.
2. Set the FMOD2~FMOD0 to “001” to select the erase operation and set the CLWB bit high to clear the write buffer. Set the FWT bit high to erase the desired page which is specified by the FARH and FARL registers and has been tagged address. Wait until the FWT bit goes low.
3. Execute a Blank Check operation using the table read instruction to ensure that the erase operation has successfully completed.  
Go to step 2 if the erase operation is not successful.  
Go to step 4 if the erase operation is successful.
4. Set the FMOD2~FMOD0 to “000” to select the write operation.
5. Setup the desired address ADDR1 in the FARH and FARL registers. Write the desired data word DATA1 first into the FD0L register and then into the FD0H register.
6. Set the FWT bit high to transfer the data word from the write buffer to the flash memory. Wait until the FWT bit goes low.
7. Verify the data using the table read instruction to ensure that the write operation has successfully completed.  
If the write operation has not successfully completed, set the CLWB bit high to clear the write buffer and then go to step 5.  
Go to step 8 if the write operation is successful.
8. Setup the desired address ADDR2 in the FARH and FARL registers. Write the desired data word DATA2 first into the FD0L register and then into the FD0H register.
9. Set the FWT bit high to transfer the data word from the write buffer to the flash memory. Wait until the FWT bit goes low.
10. Verify the data using the table read instruction to ensure that the write operation has successfully completed.  
If the write operation has not successfully completed, set the CLWB bit high to clear the write buffer and then go to step 8.  
Go to step 11 if the write operation is successful.
11. Clear the CFWEN bit low to disable the Flash memory erase/write function.



**Flash Memory Non-Consecutive Write Procedure**

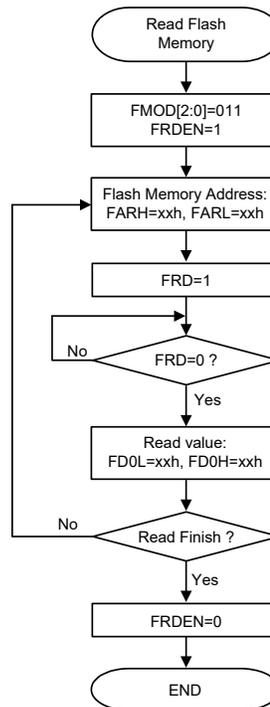
- Note: 1. When the erase or write operation is successfully activated, all CPU operations will temporarily cease.  
 2. It will take certain time for the FWT bit state changing from high to low in the erase or write operation, which can be selected by the FWERTS bit in the FC2 register.

**Important Points to Note for Flash Memory Write Operations**

1. The “Flash Memory Erase/Write Function Enable Procedure” must be successfully activated before the Flash Memory erase/write operation is executed.
2. The Flash Memory erase operation is executed to erase a whole page.
3. The whole write buffer data will be written into the flash memory in a page format. The corresponding address cannot exceed the page boundary.
4. After the data is written into the flash memory the flash memory contents must be read out using the table read instruction, TABRD, and checked if it is correct or not. If the data written into the flash memory is incorrect, the write buffer should be cleared by setting the CLWB bit high and then write the data again into the write buffer. Then activate a write operation on the same flash memory page without erasing it. The data check, buffer clear and data re-write steps should be repeatedly executed until the data written into the flash memory is correct.
5. The system frequency should be setup to the maximum application frequency when data write and data check operations are executed using the IAP function.

**Flash Memory Read Procedure**

To activate the Flash Memory Read procedure, the FMOD field should be set to “011” to select the flash memory read mode and the FRDEN bit should be set high to enable the read function. The desired flash memory address should be written into the FARH and FARL registers and then the FRD bit should be set high. After this the flash memory read operation will be activated. The data stored in the specified address can be read from the data registers, FD0H and FD0L, when the FRD bit goes low. There is no need to first activate the Flash Memory Erase/Write Function Enable Procedure before the flash memory read operation is executed.



**Flash Memory Read Procedure**

- Note: 1. When the read operation is successfully activated, all CPU operations will temporarily cease.
2. It will take a typical time of three instruction cycles for the FRD bit state changing from high to low.

## 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.

Categorised into two types, the first of these is an area of RAM, known as the Special Function Data Memory. These registers have fixed locations and are necessary for correct operation of the device. Many of these registers can be read from and written to directly under program control, however, some remain protected from user manipulation. The second area of Data Memory is 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.

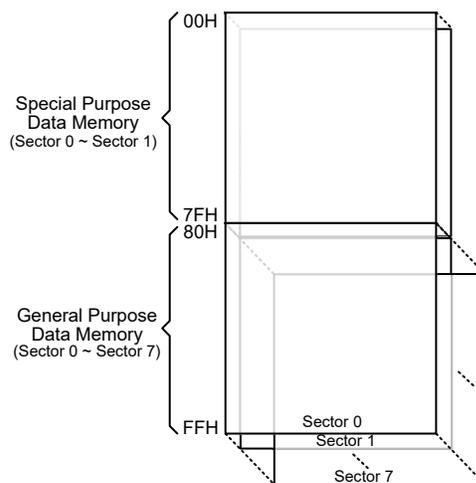
Switching between the different Data Memory sectors is achieved by properly setting the Memory Pointers to correct value when using the indirectly accessing method.

### Structure

The Data Memory is subdivided into several sectors, all of which are implemented in 8-bit wide RAM. Each of the Data Memory sector is categorized into two types, the Special Purpose Data Memory and the General Purpose Data Memory. The address range of the Special Purpose Data Memory for the device is from 00H to 7FH while the General Purpose Data Memory address range is from 80H to FFH.

| Special Purpose Data Memory | General Purpose Data Memory |   |
|-----------------------------|-----------------------------|---|
| Located Sectors             | Capacity                    | Sector: Address   |
| 0, 1                        | 1K×8                        | 0: 80H~FFH<br>1: 80H~FFH<br>2: 80H~FFH<br>⋮<br>7: 80H~FFH |

**Data Memory Summary**



**Data Memory Structure**

## **Data Memory Addressing**

For the device that supports the extended instructions, there is no Bank Pointer for Data Memory addressing. The Bank Pointer, PBB, is only available for Program 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 11 valid bits, the high byte indicates a sector and the low byte indicates a specific address within the sector.

## **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      | PC         |          |
| 02H      | IAR1    |          | 42H      | PCC        |          |
| 03H      | MP1L    |          | 43H      | PCPU       |          |
| 04H      | MP1H    |          | 44H      | LXTC       |          |
| 05H      | ACC     |          | 45H      | SIMC0      |          |
| 06H      | PCL     |          | 46H      | SIMC1      |          |
| 07H      | TBLP    |          | 47H      | SIMD       |          |
| 08H      | TBLH    |          | 48H      | SIMC2/SIMA |          |
| 09H      | TBHP    |          | 49H      | SIMTOC     |          |
| 0AH      | STATUS  |          | 4AH      | VBGRC      |          |
| 0BH      | PBP     |          | 4BH      | TB1C       |          |
| 0CH      | IAR2    |          | 4CH      | MFIO       | U0SR     |
| 0DH      | MP2L    |          | 4DH      | MF11       | UOCR1    |
| 0EH      | MP2H    |          | 4EH      | MF12       | UOCR2    |
| 0FH      | RSTFC   |          | 4FH      | SLEDC0     | UOCR3    |
| 10H      | INTEG   |          | 50H      | SLEDC1     | BRDH0    |
| 11H      | INTC0   |          | 51H      | IFS0       | BRDL0    |
| 12H      | INTC1   |          | 52H      | PD         | UFCR0    |
| 13H      | INTC2   |          | 53H      | PDC        | TXR_RXR0 |
| 14H      | PA      |          | 54H      | PDPU       | RxCNT0   |
| 15H      | PAC     |          | 55H      | ORMC       | U1SR     |
| 16H      | PAPU    |          | 56H      | SLCDS3     | U1CR1    |
| 17H      | PAWU    |          | 57H      | IFS1       | U1CR2    |
| 18H      | LVDC    |          | 58H      | PDS1       | U1CR3    |
| 19H      | SCC     |          | 59H      |            | BRDH1    |
| 1AH      | WDTC    |          | 5AH      | LVPUC      | BRDL1    |
| 1BH      | TB0C    |          | 5BH      | PAS0       | UFCR1    |
| 1CH      | HIRCC   |          | 5CH      | PAS1       | TXR_RXR1 |
| 1DH      | LVRC    |          | 5DH      | PBS0       | RxCNT1   |
| 1EH      | INTC3   |          | 5EH      | PBS1       |          |
| 1FH      |         |          | 5FH      | PCS0       |          |
| 20H      | SADOL   |          | 60H      | PCS1       |          |
| 21H      | SADOH   |          | 61H      | PDS0       |          |
| 22H      | SADC0   |          | 62H      | SLCDC0     |          |
| 23H      | SADC1   |          | 63H      | SLCDS0     |          |
| 24H      | SADC2   |          | 64H      | SLCDS1     |          |
| 25H      | PB      |          | 65H      | SLCDS2     |          |
| 26H      | PBC     |          | 66H      | PSC0R      |          |
| 27H      | PBPU    |          | 67H      | PSC1R      |          |
| 28H      | CTMC0   |          | 68H      | RSTC       |          |
| 29H      | CTMC1   |          | 69H      | FC0        |          |
| 2AH      | CTMDL   |          | 6AH      | FC1        |          |
| 2BH      | CTMDH   |          | 6BH      | FC2        |          |
| 2CH      | CTMAL   |          | 6CH      | FARL       |          |
| 2DH      | CTMAH   |          | 6DH      | FARH       |          |
| 2EH      | CTMRP   |          | 6EH      | FD0L       |          |
| 2FH      | STMC0   |          | 6FH      | FD0H       |          |
| 30H      | STMC1   |          | 70H      | FD1L       |          |
| 31H      | STMDL   |          | 71H      | FD1H       |          |
| 32H      | STMDH   |          | 72H      | FD2L       |          |
| 33H      | STMAL   |          | 73H      | FD2H       |          |
| 34H      | STMAH   |          | 74H      | FD3L       |          |
| 35H      | STMRP   |          | 75H      | FD3H       |          |
| 36H      | HXTC    |          | 76H      | EEAL       |          |
| 37H      | PTM0C0  | PTM1C0   | 77H      | EEAH       |          |
| 38H      | PTM0C1  | PTM1C1   | 78H      | EED        |          |
| 39H      | PTM0DL  | PTM1DL   | 79H      |            |          |
| 3AH      | PTM0DH  | PTM1DH   | 7AH      |            |          |
| 3BH      | PTM0AL  | PTM1AL   | 7BH      |            |          |
| 3CH      | PTM0AH  | PTM1AH   | 7CH      |            |          |
| 3DH      | PTM0RPL | PTM1RPL  | 7DH      |            |          |
| 3EH      | PTM0RPH | PTM1RPH  | 7EH      |            |          |
| 3FH      | CMPC    |          | 7FH      |            |          |

□ : Unused, read as 00H

**Special Purpose Data Memory Structure**

## Special Function Register Description

Most of the Special Function Register details will be described in the relevant functional 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 the MP1L/MP1H register pair and IAR2 register together with the MP2L/MP2H register pair can access data from any Data Memory sector. As the Indirect Addressing Registers are not physically implemented, reading the Indirect Addressing Registers will return a result of “00H” and writing to the registers will result in no operation.

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

Five Memory Pointers, known as MP0, MP1L, MP1H, MP2L, 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 extended 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.

### Program Memory Bank Pointer – PBP

For this device the Program Memory is divided into several banks. Selecting the required Program Memory area is achieved using the Program Memory Bank Pointer, PBP. The PBP register should be properly configured before the device executes the “Branch” operation using the “JMP” or “CALL” instruction. After that a jump to a non-consecutive Program Memory address which is located in a certain bank selected by the program memory bank pointer bits will occur.

#### • PBP Register

| Bit  | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0    |
|------|---|---|---|---|---|---|---|------|
| Name | — | — | — | — | — | — | — | PBP0 |
| R/W  | — | — | — | — | — | — | — | R/W  |
| POR  | — | — | — | — | — | — | — | 0    |

Bit 7~1 Unimplemented, read as “0”

Bit 0 **PBP0**: Program Memory Bank selection  
 0: Bank 0  
 1: Bank 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 Byte Register – PCL

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

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

These three special function registers are used to control operation of the look-up table which is stored in the Program Memory. TBLP and TBHP are the table pointers and indicate the location where the table data is located. Their value must be set 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.

### Option Memory Mapping Register – ORMC

The ORMC register is used to enable Option Memory Mapping function. The Option Memory capacity is 64 words. When a specific pattern of 55H and AAH is consecutively written into this register, the Option Memory Mapping function will be enabled and then the Option Memory code

can be read by using the table read instruction. The Option Memory addresses 00H~3FH will be mapped to Program Memory last page addresses C0H~FFH.

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

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

• **ORMC Register**

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

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

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

**Status Register – STATUS**

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

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

The Z, OV, AC, 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.
- CZ is the operational result of different flags for different instructions. Refer to register definitions for more details.
- SC is the result of the “XOR” operation which is performed by the OV flag and the MSB of the current instruction operation result.

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/W | R/W | 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 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  
 The C flag is also affected by a rotate through carry instruction.

## EEPROM Data Memory

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

### EEPROM Data Memory Structure

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

### EEPROM Registers

Four registers control the overall operation of the internal EEPROM Data Memory. These are the address registers, EEAL and EEAH, the data register, EED and a single control register, EEC. As both the EEAL, EEAH 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, can only be read from or written to indirectly using the MP1L/MP1H or MP2L/MP2H Memory Pointer pairs and Indirect Addressing Register, IAR1/IAR2. Because the EEC control register is located at address 40H in Sector 1, the MP1L or MP2L Memory Pointer must first be set to the value 40H and the MP1H or MP2H Memory Pointer high byte 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     |
| EEAL          | EEAL7  | EEAL6 | EEAL5 | EEAL4 | EEAL3 | EEAL2 | EEAL1 | EEAL0 |
| EEAH          | —      | —     | —     | —     | —     | —     | EEAH1 | EEAH0 |
| EED           | EED7   | EED6  | EED5  | EED4  | EED3  | EED2  | EED1  | EED0  |
| EEC           | EWERTS | EREN  | ER    | MODE  | WREN  | WR    | RDEN  | RD    |

EEPROM Register List

#### • EEAL Register

| Bit  | 7     | 6     | 5     | 4     | 3     | 2     | 1     | 0     |
|------|-------|-------|-------|-------|-------|-------|-------|-------|
| Name | EEAL7 | EEAL6 | EEAL5 | EEAL4 | EEAL3 | EEAL2 | EEAL1 | EEAL0 |
| 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      **EEAL7~EEAL0**: Data EEPROM low byte address bit 7 ~ bit 0

• **EEAH Register**

| Bit  | 7 | 6 | 5 | 4 | 3 | 2 | 1     | 0     |
|------|---|---|---|---|---|---|-------|-------|
| Name | — | — | — | — | — | — | EEAH1 | EEAH0 |
| R/W  | — | — | — | — | — | — | R/W   | R/W   |
| POR  | — | — | — | — | — | — | 0     | 0     |

Bit 7~2 Unimplemented, read as “0”

Bit 1~0 **EEAH1~EEAH0**: Data EEPROM high byte address bit 1 ~ bit0

• **EED Register**

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

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

• **EEC Register**

| Bit  | 7      | 6    | 5   | 4    | 3    | 2   | 1    | 0   |
|------|--------|------|-----|------|------|-----|------|-----|
| Name | EWERTS | EREN | ER  | MODE | WREN | WR  | RDEN | RD  |
| 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 **EWERTS**: EEPROM Erase time and Write time selection  
 0: Erase time is 3.2ms ( $t_{EEER}$ ) / Write time is 2.2ms ( $t_{EEWR}$ )  
 1: Erase time is 3.7ms ( $t_{EEER}$ ) / Write time is 3.0ms ( $t_{EEWR}$ )

Bit 6 **EREN**: Data EEPROM Erase Enable  
 0: Disable  
 1: Enable

This bit is used to enable data EEPROM erase function and must be set high before erase operations are carried out. This bit will be automatically reset to zero by the hardware after the erase cycle has finished. Clearing this bit to zero will inhibit data EEPROM erase operations.

Bit 5 **ER**: Data EEPROM Erase Control  
 0: Erase cycle has finished  
 1: Activate a erase cycle

When this bit is set high by the application program, an erase cycle will be activated. This bit will be automatically reset to zero by the hardware after the erase cycle has finished. Setting this bit high will have no effect if the EREN has not first been set high.

Bit 4 **MODE**: Data EEPROM Operation mode selection  
 0: Byte operation mode  
 1: Page operation mode

This is the EEPROM operation mode selection bit. When the bit is set high by the application program, the Page write, erase or read function will be selected. Otherwise, the byte write or read function will be selected. The EEPROM page buffer size is 16 bytes.

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. Note that the WREN bit will automatically be cleared to zero after the write operation is finished.

- Bit 2      **WR**: Data EEPROM Write Control  
            0: Write cycle has finished  
            1: Activate a write cycle  
            This is the Data EEPROM Write Control Bit and when set high by the application program will activate a write cycle. This bit will be automatically reset to zero by the hardware after the write cycle has finished. Setting this bit high will have no effect if the WREN has not first been set high.
- Bit 1      **RDEN**: Data EEPROM Read Enable  
            0: Disable  
            1: Enable  
            This is the Data EEPROM Read Enable Bit which must be set high before Data EEPROM read operations are carried out. Clearing this bit to zero will inhibit Data EEPROM read operations.
- Bit 0      **RD**: EEPROM Read Control  
            0: Read cycle has finished  
            1: Activate a read cycle  
            This is the Data EEPROM Read Control Bit and when set high by the application program will activate a read cycle. This bit will be automatically reset to zero by the hardware after the read cycle has finished. Setting this bit high will have no effect if the RDEN has not first been set high.

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

## Read Operation from the EEPROM

Reading data from the EEPROM can be implemented by two modes for this device, byte read mode or page read mode, which is controlled by the EEPROM operation mode selection bit, MODE, in the EEC register.

### Byte Read Mode

The EEPROM byte read operation can be executed when the mode selection bit, MODE, is cleared to zero. For a byte read operation the desired EEPROM address should first be placed in the EEAH and EEAL registers, as well as the read enable bit, RDEN, in the EEC register should be set high to enable the read function. Then setting the RD bit high will initiate the EEPROM byte read operation. Note that setting the RD bit high only will not initiate a read operation if the RDEN bit is not set high. When the read cycle terminates, the RD bit will automatically be cleared to zero and the EEPROM 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.

### Page Read Mode

The EEPROM page read operation can be executed when the mode selection bit, MODE, is set high. The page size can be up to 16 bytes for the page read operation. For a page read operation the start address of the desired EEPROM page should first be placed in the EEAH and EEAL registers, as well as the read enable bit, RDEN, in the EEC register should be set high to enable the read function. Then setting the RD bit high will initiate the EEPROM page read operation. Note that setting the RD bit high only will not initiate a read operation if the RDEN bit is not set high. When the current byte read cycle terminates, the RD bit will automatically be cleared indicating that the EEPROM data can be read from the EED register, and the current address will be incremented by one by hardware. The data which is stored in the next EEPROM address can continuously be read from the EED register

when the RD bit is again set high without reconfiguring the EEPROM address and RDEN control bit. The application program can poll the RD bit to determine when the data is valid for reading.

The EEPROM address higher 6 bits are used to specify the desired page location while the lower 4 bits are used to point to the actual address. In the page operation mode the lower 4-bit address value will automatically be incremented by one. However, the higher 6-bit address value will not be incremented by hardware. When the EEPROM address lower 4-bit value which is internally incremented by one in the page mode reaches the page boundary, known as 0FH, the EEPROM address lower 4-bit value will stop at 0FH. The EEPROM address will not “roll over”.

### **Page Erase Operation to the EEPROM**

The EEPROM page erase operation can be executed when the mode selection bit, MODE, is set high. The EEPROM is capable of a 16-byte page erase. The internal page buffer will be cleared by hardware after power-on reset. When the EEPROM erase enable control bit, namely EREN, is changed from “1” to “0”, the internal page buffer will also be cleared. Note that when the EREN bit is changed from “0” to “1”, the internal page buffer will not be cleared. The EEPROM address higher 6 bits are used to specify the desired page location while the lower 4 bits are used to point to the actual address. In the page erase operation mode the lower 4-bit address value will automatically be incremented by one after each dummy data byte is written into the EED register. However, the higher 6-bit address value will not be incremented by hardware. When the EEPROM address lower 4-bit value which is internally incremented by one in the page mode reaches the page boundary, namely 0FH, the EEPROM address lower 4-bit value will stop at 0FH. The EEPROM address will not “roll over”.

For page erase operations the start address of the desired EEPROM page should first be placed in the EEAH and EEAL registers and the dummy data to be written is placed in the EED register. The maximum data length for a page is 16 bytes. Note that the write operation to the EED register is used to tag address, it must be implemented to determine which addresses to be erased. When the page dummy data is completely written then the EREN bit in the EEC register should first be set high to enable erase operations and the ER bit must be immediately set high to initiate the EEPROM erase process. These two instructions must be executed in two consecutive instruction cycles to activate an erase operation successfully. The global interrupt enable bit EMI should also first be cleared before implementing an erase operation and then set again after a valid erase activation procedure has completed.

As the EEPROM erase 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 erased from the EEPROM. Detecting when the erase cycle has finished can be implemented either by polling the ER bit in the EEC register or by using the EEPROM interrupt. When the erase cycle terminates, the ER bit will be automatically cleared to zero by the microcontroller, informing the user that the page data has been erased. The application program can therefore poll the ER bit to determine when the erase cycle has ended. After the erase operation is finished, the EREN bit will be set low by hardware. The Data EEPROM erased page content will all be zero after a page erase operation.

### **Write Operation to the EEPROM**

Writing data to the EEPROM can be implemented by two modes for this device, byte write mode or page write mode, which is controlled by the EEPROM operation mode selection bit, MODE, in the EEC register.

### Byte Write Mode

The EEPROM byte write operation can be executed when the mode selection bit, MODE, is cleared to zero. For byte write operations the desired EEPROM address should first be placed in the EEAL and EEAH registers and the data to be written should be placed in the EED register. To write data to the EEPROM, the write enable bit, WREN, in the EEC register must first be set high to enable the write function. After this, the WR bit in the EEC register must be immediately set high to initiate a write cycle. These two instructions must be executed in two consecutive instruction cycles to activate a write operation successfully. The global interrupt bit EMI should also first be cleared before implementing any write operations, and then set high again after a valid write activation procedure has completed. 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. Note that a byte erase operation will automatically be executed before a byte write operation is successfully activated.

### Page Write Mode

Before a page write operation is executed, it is important to ensure that a relevant page erase operation has been successfully executed. The EEPROM page write operation can be executed when the mode selection bit, MODE, is set high. The EEPROM is capable of a 16-byte page write. The internal page buffer will be cleared by hardware after power on reset. When the EEPROM write enable control bit, namely WREN, is changed from “1” to “0”, the internal page buffer will also be cleared. Note that when the WREN bit is changed from “0” to “1”, the internal page buffer will not be cleared. A page write is initiated in the same way as a byte write initiation except that the EEPROM data can be written up to 16 bytes. The EEPROM address higher 6 bits are used to specify the desired page location while the lower 4 bits are used to point to the actual address. In the page write operation mode the lower 4-bit address value will automatically be incremented by one after each data byte is written into the EED register. However, the higher 6-bit address value will not be incremented by hardware. When the EEPROM address lower 4-bit value which is internally incremented by one in the page mode reaches the page boundary, namely 0FH, the EEPROM address lower 4-bit value will stop at 0FH. The EEPROM address will not “roll over”. At this point any data write operations to the EED register will be invalid.

For page write operations the the start address of the desired EEPROM page must first be placed in the EEAH and EEAL registers and the data placed in the EED register. The maximum data length for a page is 16 bytes. Note that when a data byte is written into the EED register, then the data in the EED register will be loaded into the internal page buffer and the current address value will automatically be incremented by one. When the page data is completely written into the page buffer, then the write enable bit, WREN, in the EEC register should first be set high to enable the write function. After this, the WR bit in the EEC register must be immediately set high to initiate a write cycle. These two instructions must be executed in two consecutive instruction cycles. The global interrupt bit EMI should also first be cleared before implementing any write operations, and then set again after a valid write activation procedure has completed. 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. After the write operation is finished, the WREN bit will be set low by hardware.

### **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 interrupt is generated when an EEPROM erase or 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 interrupt is contained within a Multi-function Interrupt, the associated multi-function interrupt enable bit must also be set. When an EEPROM erase or write cycle ends, the DEF request flag and its associated multi-function interrupt request flag will both be set high. 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. The EMI bit will also be automatically cleared to disable other interrupts. More details can be obtained in the Interrupts section.

### **Programming Considerations**

Care must be taken that data is not inadvertently written to the EEPROM. Protection can be enhanced by ensuring that the Write Enable bit is normally cleared to zero when not writing. Also the Memory Pointer high byte register, MP1H or MP2H, could be normally cleared to zero as this would inhibit access to Sector 1 where the EEPROM control register exists. Although certainly not necessary, consideration might be given in the application program to the checking of the validity of new write data by a simple read back process.

When writing or erasing data the WR/ER bit must be set high immediately after the WREN/EREN bit has been set high, to ensure the write cycle or erase executes correctly. The global interrupt bit EMI should also be cleared before a write or erase cycle is executed and then set again after a valid write or erase activation procedure has completed. Note that the device should not enter the IDLE or SLEEP mode until the EEPROM read or write operation is totally complete. Otherwise, the EEPROM read, erase or write operation will fail.

### Programming Examples

#### Reading a Data Byte from the EEPROM – polling method

```

MOV A, 40H                ; setup memory mointer lower byte MP1L
MOV MP1L, A              ; MP1L points to EEC register
MOV A, 01H               ; setup memory pointer high byte MP1H
MOV MP1H, A
CLR IAR1.4               ; clear MODE bit, select byte operation mode
MOV A, EEPROM_ADRES_H    ; user defined high byte address
MOV EEAH, A
MOV A, EEPROM_ADRES_L    ; user defined low byte address
MOV EEAL, 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 function
CLR MP1H
MOV A, EED                ; move read data to register
MOV READ_DATA, A

```

#### Reading a Data Page from the EEPROM – polling method

```

MOV A, 40H                ; set memory pointer low byte MP1L
MOV MP1L, A              ; MP1L points to EEC register
MOV A, 01H               ; set memory pointer high byte MP1H
MOV MP1H, A
SET IAR1.4               ; set MODE bit, select page operation mode
MOV A, EEPROM_ADRES_H    ; user defined high byte address
MOV EEAH, A
MOV A, EEPROM_ADRES_L    ; user defined low byte address
MOV EEAL, A
SET IAR1.1               ; set RDEN bit, enable read operations
; ~~~~ The data length can be up to 16 bytes (Start) ~~~~
CALL READ
CALL READ
:
:
JMP PAGE_READ_FINISH
; ~~~~ The data length can be up to 16 bytes (End) ~~~~
READ:
SET IAR1.0               ; start Read Cycle - set RD bit
BACK:
SZ IAR1.0                ; check for read cycle end
JMP BACK
MOV A, EED                ; move read data to register
MOV READ_DATA, A
RET
:
PAGE_READ_FINISH:
CLR IAR1                  ; disable EEPROM read function
CLR MP1H

```

**Erasing a Data Page to the EEPROM – polling method**

```
MOV A, 40H ; set memory pointer low byte MP1L
MOV MP1L, A ; MP1L points to EEC register
MOV A, 01H ; set memory pointer high byte MP1H
MOV MP1H, A
SET IAR1.4 ; set MODE bit, select page operation mode
MOV A, EEPROM_ADRES_H ; user defined high byte address
MOV EEAH, A
MOV A, EEPROM_ADRES_L ; user defined low byte address
MOV EEAL, A
; ~~~~ The data length can be up to 16 bytes (Start) ~~~~
CALL WRITE_BUF
CALL WRITE_BUF
:
:
JMP Erase_START
; ~~~~ The data length can be up to 16 bytes (End) ~~~~
WRITE_BUF:
MOV A, EEPROM_DATA ; user defined data, erase mode don't care data value
MOV EED, A
RET
:
Erase_START:
CLR EMI
SET IAR1.6 ; set EREN bit, enable write operations
SET IAR1.5 ; start Write Cycle - set ER bit - executed immediately
; after setting EREN bit

SET EMI
BACK:
SZ IAR1.5 ; check for write cycle end
JMP BACK
CLR MP1H
```

**Writing a Data Byte to the EEPROM – polling method**

```
MOV A, 40H ; set memory pointer low byte MP1L
MOV MP1L, A ; MP1L points to EEC register
MOV A, 01H ; set memory pointer high byte MP1H
MOV MP1H, A
CLR IAR1.4 ; clear MODE bit, select byte write mode
MOV A, EEPROM_ADRES_H ; user defined high byte address
MOV EEAH, A
MOV A, EEPROM_ADRES_L ; user defined low byte address
MOV EEAL, A
MOV A, EEPROM_DATA ; user defined data
MOV EED, A
CLR EMI
SET IAR1.3 ; set WREN bit, enable write operations
SET IAR1.2 ; start Write Cycle - set WR bit - executed immediately
; after setting WREN bit

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

**Writing a Data Page to the EEPROM – polling method**

```
MOV A, 40H ; set memory pointer low byte MP1L
MOV MP1L, A ; MP1L points to EEC register
MOV A, 01H ; set memory pointer high byte MP1H
MOV MP1H, A
SET IAR1.4 ; set MODE bit, select page operation mode
MOV A, EEPROM_ADRES_H ; user defined high byte address
MOV EEAH, A
MOV A, EEPROM_ADRES_L ; user defined low byte address
MOV EEAL, A
; ~~~~ The data length can be up to 16 bytes (Start) ~~~~
CALL WRITE_BUF
CALL WRITE_BUF
:
:
JMP WRITE_START
; ~~~~ The data length can be up to 16 bytes (End) ~~~~
WRITE_BUF:
MOV A, EEPROM_DATA ; user define data
MOV EED, A
RET
:
WRITE_START:
CLR EMI
SET IAR1.3 ; set WREN bit, enable write operations
SET IAR1.2 ; start Write Cycle - set WR bit - executed immediately
; after setting WREN bit

SET EMI
BACK:
SZ IAR1.2 ; check for write cycle end
JMP BACK
CLR 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 option 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 range of both fast and slow system oscillators. 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.

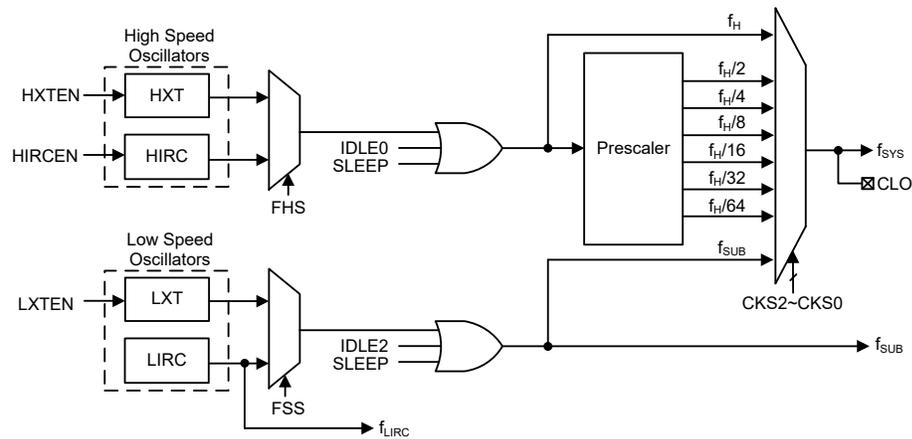
| Type                                   | Name | Frequency    | Pins      |
|--|------|--------------|-----------|
| External High Speed Crystal Oscillator | HXT  | 400kHz~16MHz | OSC1/OSC2 |
| Internal High Speed RC Oscillator      | HIRC | 8/12/16MHz   | —         |
| External Low Speed Crystal Oscillator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

**Oscillator Types**

### System Clock Configurations

There are several oscillator sources, two high speed oscillators and two low speed oscillators for the device. The high speed oscillators are the external crystal/ceramic oscillator, HXT, and the internal 8/12/16MHz RC oscillator, HIRC. The 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 the system clock can be dynamically selected.

The actual source clock used for the low speed oscillator is chosen via 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 also 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.

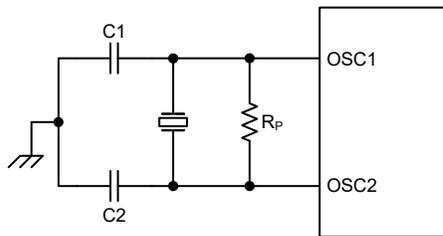


**System Clock Configurations**

### 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/Ceramic Oscillator – HXT**

| HXT Oscillator C1 and C2 Values |       |       |
|---------------------------------|-------|-------|
| Crystal Frequency               | C1    | C2    |
| 16MHz                           | 0pF   | 0pF   |
| 12MHz                           | 0pF   | 0pF   |
| 8MHz                            | 0pF   | 0pF   |
| 6MHz                            | 0pF   | 0pF   |
| 4MHz                            | 0pF   | 0pF   |
| 1MHz                            | 100pF | 100pF |

Note: C1 and C2 values are for guidance only.

**Crystal Recommended Capacitor Values**

**Internal High Speed RC Oscillator – HIRC**

The internal RC oscillator is one of the high frequency oscillator choices, which is selected via a software control bit, FHS. It is a fully integrated system oscillator requiring no external components. The internal RC oscillator has three fixed frequencies of 8MHz, 12MHz and 16MHz, which are selected by HIRC1~HIRC0 bits in the HIRCC register. These bits must be setup to match the selected configuration option frequency to ensure that the HIRC frequency accuracy specified in the A.C. Characteristics is achieved. 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. Note that if this internal system clock option requires no external pins for its operation.

**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.

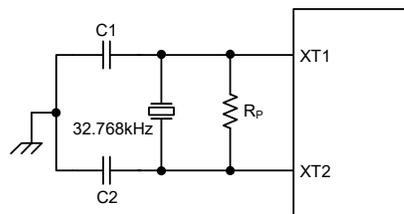
When the microcontroller enters the SLEEP or IDLE Mode, the system clock is switched off to stop microcontroller activity and to conserve power. However, in many microcontroller applications it may be necessary to keep the internal timers operational even when the microcontroller is in the SLEEP or IDLE 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's specification. The external parallel feedback resistor, R<sub>p</sub>, is required.

The pin-shared software control bits determine if the XT1/XT2 pins are used for the LXT oscillator or as 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<sub>p</sub>, 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.<br>2. R <sub>p</sub> =5MΩ~10MΩ 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 register.

| LXTSP | LXT 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 stabilise 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 CKS2~CKS0 bits and FSS bit in the SCC register, the LXT oscillator operating mode cannot be changed.

It should be noted that, no matter what condition the LXTSP bit is set to, the LXT oscillator will be always function normally, the only difference is that it will take more time to start up if it is in the low power mode.

### Internal 32kHz Oscillator – LIRC

The Internal 32kHz Oscillator is one of the low frequency oscillator choices, which is selected by the FSS bit in the SCC register. It is a fully integrated RC oscillator with a typical frequency of 32kHz, requiring no external components for its implementation. Device trimming during the manufacturing process and the inclusion of internal frequency compensation circuits are used to ensure that the influence of the power supply voltage, temperature and process variations on the oscillation frequency are minimised.

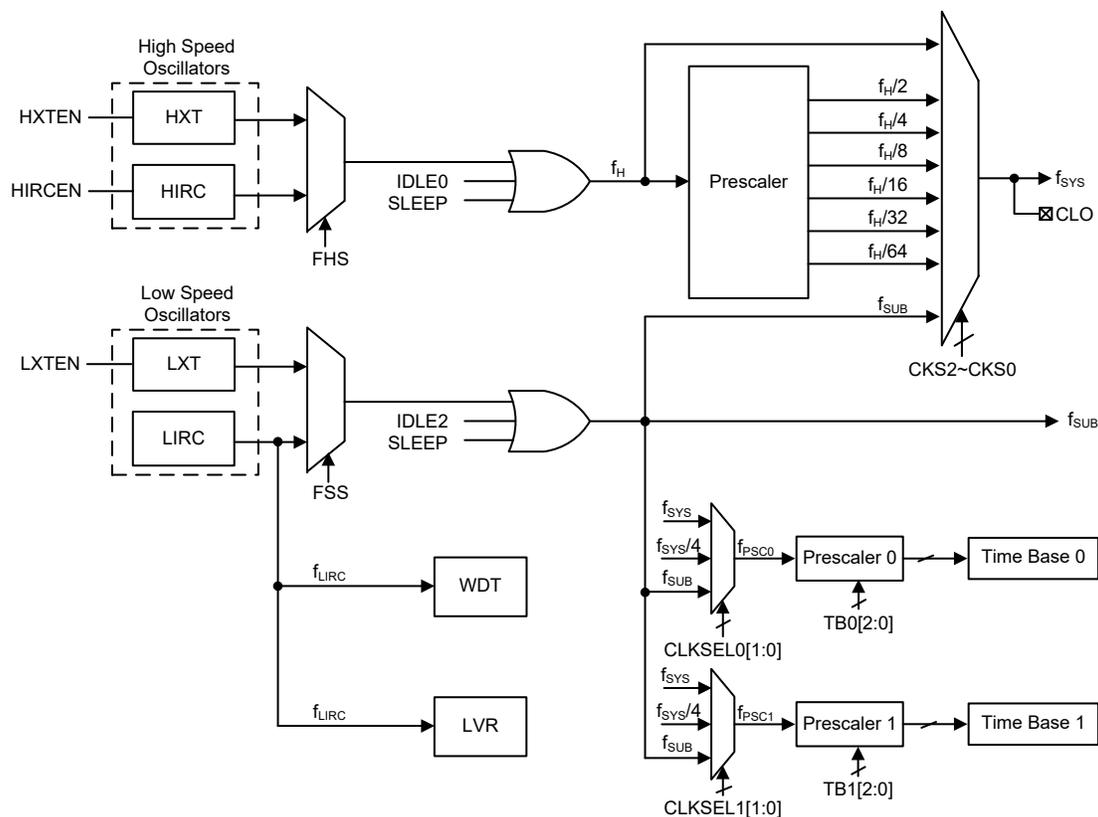
## Operating Modes and System Clocks

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

### System Clocks

The device has many different clock sources for both the CPU and peripheral function operation. By providing the user with a wide range of clock 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 can be sourced from either 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 internal clock  $f_{SUB}$ . If  $f_{SUB}$  is selected then it can be sourced by either the LXT or LIRC oscillator, 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 will stop to conserve the power or continue to oscillate to provide the clock source,  $f_H \sim f_H/64$ , for peripheral circuit to use, which is determined by configuring the corresponding high speed oscillator enable control bit.

### System Operation Modes

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

| Operation Mode | CPU | Register Setting |        |           | $f_{SYS}$         | $f_H$                 | $f_{SUB}$ | $f_{LIRC}$            |
|----------------|-----|------------------|--------|-----------|-------------------|-----------------------|-----------|-----------------------|
|                |     | FHIDEN           | FSIDEN | CKS2~CKS0 |                   |                       |           |                       |
| FAST           | On  | x                | x      | 000~110   | $f_H \sim f_H/64$ | On                    | On        | On                    |
| SLOW           | On  | x                | x      | 111       | $f_{SUB}$         | On/Off <sup>(1)</sup> | On        | On                    |
| IDLE0          | Off | 0                | 1      | 000~110   | Off               | Off                   | On        | On                    |
|                |     |                  |        | 111       | On                |                       |           |                       |
| IDLE1          | Off | 1                | 1      | xxx       | On                | On                    | On        | On                    |
| IDLE2          | Off | 1                | 0      | 000~110   | On                | On                    | Off       | On                    |
|                |     |                  |        | 111       | Off               |                       |           |                       |
| SLEEP          | Off | 0                | 0      | xxx       | Off               | Off                   | Off       | On/Off <sup>(2)</sup> |

"x": Don't care

Note: 1. The  $f_H$  clock will be switched on or off by configuring the corresponding oscillator enable bit in the SLOW mode.

2. The  $f_{LIRC}$  clock can be switched on or off which is controlled by the WDT function being enabled or disabled in the SLEEP mode.

#### FAST Mode

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

#### SLOW Mode

This is also a mode where the microcontroller operates normally although now with a slower speed clock source. The clock source used will be from  $f_{SUB}$ . The  $f_{SUB}$  clock is derived from either the LIRC or LXT oscillator determined by the FSS bit in the SCC register.

#### SLEEP Mode

The SLEEP Mode is entered when a HALT instruction is executed and when the FHIDEN and FSIDEN bit are low. In the SLEEP mode the CPU will be stopped. The  $f_{SUB}$  clock provided to the peripheral function will also be stopped, too. However the  $f_{LIRC}$  clock can continue to operate if the WDT function is enabled by the WDTC register.

#### IDLE0 Mode

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

#### IDLE1 Mode

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

#### IDLE2 Mode

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

### Control Registers

The 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 Register 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 oscillator selection

- 0: HIRC
- 1: HXT

Bit 2 **FSS**: Low Frequency oscillator 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.

Note: A certain delay is required before the relevant clock is successfully switched to the target clock source after any clock switching setup using the CKS2~CKS0 bits, FHS bit or FSS bit. A proper delay time must be arranged before executing the following operations which require immediate reaction with the target clock source.

Clock switching delay time= $4 \times t_{SYS} + [0 \sim (1.5 \times t_{CURR} + 0.5 \times t_{TAR})]$ , where  $t_{CURR}$  indicates the current clock period,  $t_{TAR}$  indicates the target clock period and  $t_{SYS}$  indicates the current system clock period.

• 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: 8MHz
- 01: 12MHz
- 10: 16MHz
- 11: 8MHz

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 high.

It is recommended that the HIRC frequency selected by these two bits should be same with the frequency determined by the configuration option to achieve the HIRC frequency accuracy specified in the A.C. Characteristics.

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 high to enable the HIRC oscillator or the HIRC frequency selection is changed by application program, the HIRCF bit will first be cleared to zero and then set high 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  $\leq$  10MHz (sink/source current is smaller)  
           1: HXT frequency  $>$  10MHz (sink/source current is largler)

Note that this bit should be configured correctly according to the used HXT frequency. If HXTM=0 while the HXT frequency is larger than 10MHz, the oscillation performance at a low voltage condition may be not well. If HXTM=1 while the HXT frequency is less than 10MHz, the oscillator frequency and current may be obnormal.

This bit must be properly configured before the HXT is enabled. When the OSC1 and OSC2 pin functions have been enabled using relevant pin-shared control bits and the HXTEN bit is set to 1 to enable the HXT oscillator, it is invalid to change the value of this bit. When the OSC1 and OSC2 pin functions are disabled, then the HXTM bit can be changed by software, regardless of the HXTEN bit value.

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 high to enable the HXT oscillator, the HXTF bit will first be cleared to zero and then set high 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  | — | — | — | — | — | R/W   | R    | R/W   |
| POR  | — | — | — | — | — | 0     | 0    | 0     |

Bit 7~3 Unimplemented, read as “0”

Bit 2 **LXTSP**: LXT 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 high, the LXT oscillator will oscillate quickly but consume more power. If the LXTSP bit is cleared to zero, the LXT oscillator will consume less power but take longer time to stabilise. It is important to note that this bit cannot 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

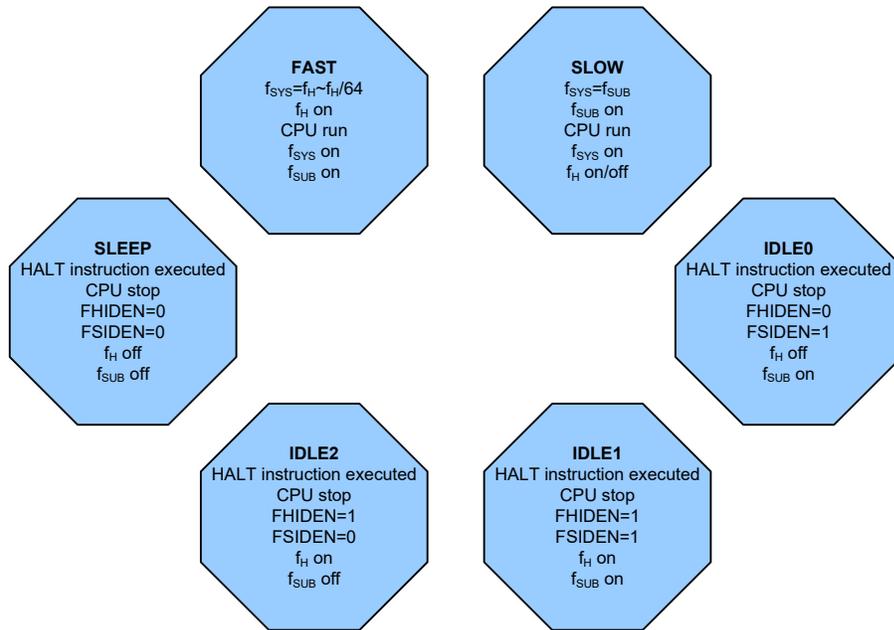
This bit is used to indicate whether the LXT oscillator is stable or not. When the LXTEN bit is set high to enable the LXT oscillator, the LXTF bit will first be cleared to zero and then set high after the LXT oscillator is stable.

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.

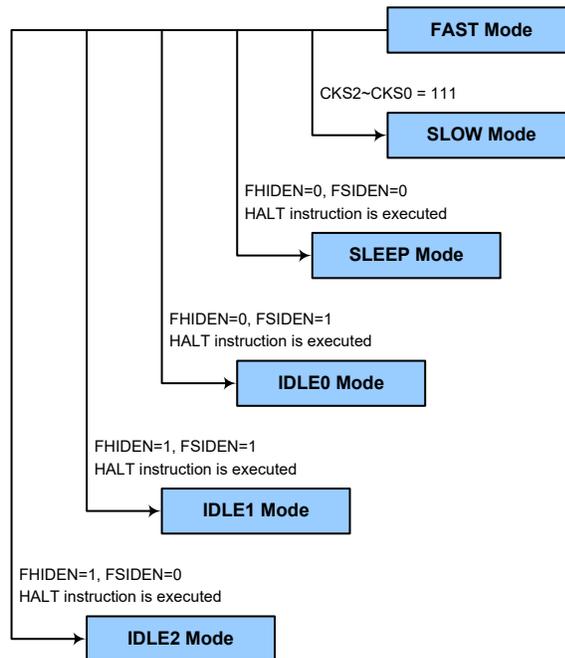
In simple terms, mode switching between the FAST Mode and SLOW Mode is executed using the CKS2~CKS0 bits in the SCC register while mode switching from the FAST/SLOW Modes to the SLEEP/IDLE Modes is executed via the HALT instruction. When a HALT instruction is executed, whether the device enters the IDLE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



**FAST Mode to SLOW Mode Switching**

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

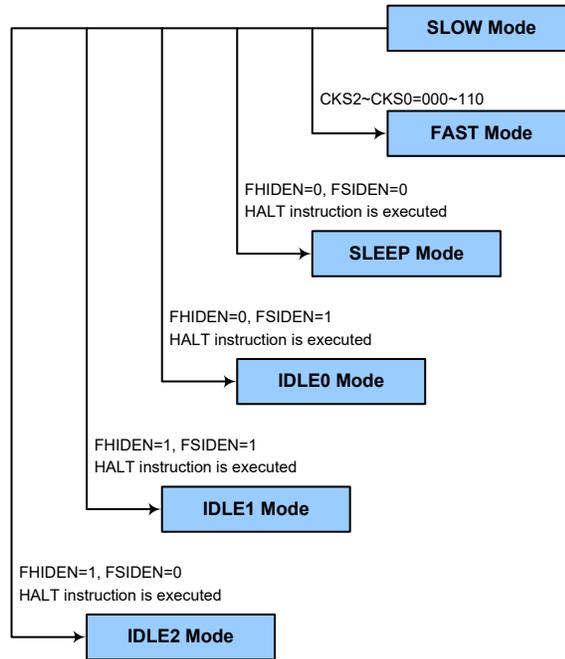
The SLOW Mode system clock is sourced from the LXT or LIRC oscillator determined by the FSS bit in the SCC register and therefore requires the selected oscillator to be stable before full mode switching occurs.



### SLOW Mode to FAST Mode Switching

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

However, if  $f_H$  is not used in SLOW mode and thus switched off, it will take some time to re-oscillate and stabilise when switching to the FAST mode from the SLOW Mode. This is monitored using the 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 System Start Up Time Characteristics.



### Entering the SLEEP Mode

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

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

### Entering the IDLE0 Mode

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

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

### Entering the IDLE1 Mode

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

- The  $f_H$  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 if the WDT function is enabled. If the WDT function is disabled, the WDT will be cleared and then stopped.

### Entering the IDLE2 Mode

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

- The  $f_H$  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 the Watchdog time-out flag, TO, will be cleared.
- The WDT will be cleared and resume counting if the WDT function is enabled. If the WDT function is disabled, the WDT will be cleared and then stopped.

### Standby Current Considerations

As the main reason for entering the SLEEP or IDLE Mode is to keep the current consumption of the device to as low a value as possible, perhaps only in the order of several micro-amps except in the IDLE1 and IDLE2 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. This also applies to the device which has different package types, as there may be unbonded pins. These must either be set as outputs or if set 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 set 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 via configuration option.

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

## **Wake-up**

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

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

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

When the device executes the “HALT” instruction, it will enter the IDLE or SLEEP mode and the PDF flag will be set 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 set 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.

## Watchdog Timer

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

### Watchdog Timer Clock Source

The Watchdog Timer clock source is provided by the internal clock,  $f_{LIRC}$ , which is sourced from the LIRC oscillator. The LIRC internal oscillator has an approximate frequency of 32kHz and this specified internal clock period can vary with  $V_{DD}$ , temperature and process variations. The Watchdog Timer source clock is then subdivided by a ratio of  $2^8$  to  $2^{18}$  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 time-out period as well as the Watchdog Timer the enable/disable and the MCU reset operation.

#### • WDTC Register

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

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

10101: Disable

01010: Enable

Other values: Reset MCU

When these bits are changed to any other values due to environmental noise the microcontroller will be reset; this reset operation will be activated after a delay time,  $t_{SRESET}$ , and the WRF bit in the RSTFC register will be set high.

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

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

001:  $2^{10}/f_{LIRC}$

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  
Refer to Internal Reset Control section.

Bit 2 **LVRF**: LVR function reset flag  
Refer to the Low Voltage Reset section.

- Bit 1      **LRF**: LVR control register software reset flag  
Refer to the Low Voltage Reset section.
- Bit 0      **WRF**: WDT control register software reset flag  
0: Not occurred  
1: Occurred  
This bit is set high by the WDT Control register software reset and cleared to zero by the application program. Note that this bit can only be cleared to zero by the application program.

**Watchdog Timer Operation**

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

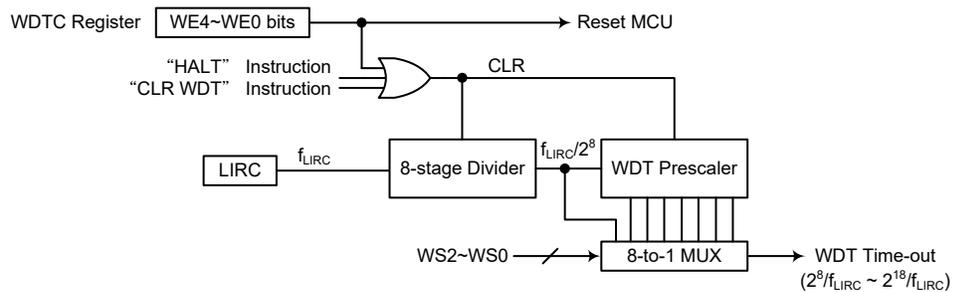
| WE4~WE0 Bits    | WDT Function |
|-----------------|--------------|
| 10101B          | Disable      |
| 01010B          | Enable       |
| Any other value | Reset MCU    |

**Watchdog Timer Function 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 WDTC software reset, which means a certain value except 01010B and 10101B written into the WE4~WE0 bit, the second is using the Watchdog Timer software clear instruction and the third is via a HALT instruction.

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

The maximum time-out period is when the  $2^{18}$  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 8ms for the  $2^8$  division ratio.



**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.

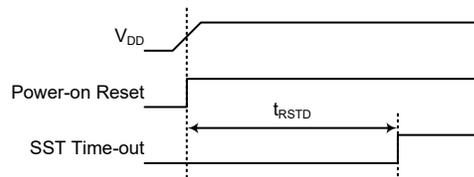
In addition to the power-on reset, another reset exists in the form of a Low Voltage Reset, LVR, where a full reset is implemented in situations where the power supply voltage falls below a certain threshold. Another type of reset is when the Watchdog Timer overflows and resets the microcontroller. All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

### Reset Functions

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

#### Power-on Reset

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



**Power-On Reset Timing Chart**

#### 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 a delay time, t<sub>SRESET</sub>. 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 a delay time,  $t_{SRESET}$  and the RSTF bit in the RSTFC register will be set to 1. All resets will reset this register to POR value except the WDT time-out reset during SLEEP/IDLE mode.

• **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 high by the RSTC control register software reset and cleared to zero by the application program. Note that this bit can only be cleared to 0 by the application program.

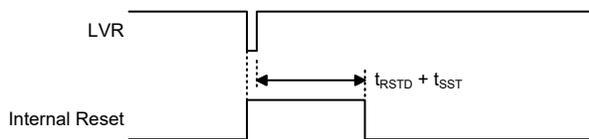
Bit 2 **LVRF**: LVR function reset flag  
 Refer to the Low Voltage Reset section.

Bit 1 **LRF**: LVR control register software reset flag  
 Refer to the Low Voltage Reset section.

Bit 0 **WRF**: WDT control register software reset flag  
 Refer to the Watchdog Timer Control Register section.

**Low Voltage Reset – LVR**

The microcontroller contains a low voltage reset circuit in order to monitor the supply voltage of the device and provides an MCU reset should the value fall below a certain predefined level. The LVR function can be enabled or disabled by the LVRC control register. If the LVRC control register is configured to enable the LVR function, the LVR function will be always enabled except in the SLEEP or IDLE mode. If the supply voltage of the device drops to within a range of  $0.9V \sim V_{LVR}$  such as might occur when changing the battery in battery powered applications, the LVR will automatically reset the device internally and the LVRF bit in the RSTFC register will also be set high. 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 LVS7~LVS0 bits in the LVRC register. If the LVS7~LVS0 bits are changed to some different values by environmental noise, the LVR will reset the device after a delay time,  $t_{SRESET}$ . When this happens, the LRF bit in the RSTFC register will be set high. After power-on the register will have the value of 01100110B. Note that the LVR function will be automatically disabled when the device enters the IDLE or SLEEP mode.



**Low Voltage Reset Timing Chart**

• **LVRC Register**

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

Bit 7~0 **LVS7~LVS0**: LVR voltage selection

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

Any register value, other than the five defined LVR values and 11110000B above, will also result in the generation of an MCU reset. The reset operation will be activated after a delay time,  $t_{SRESET}$ . 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  
Refer to the Internal Reset Control section.

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

This bit is set high when a specific low voltage reset situation condition occurs. This bit can only be cleared to zero by the application program.

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

This bit is set high if the LVRC register containing any non-defined LVR voltage register values. This in effect acts like a software reset function. This bit can only be cleared to zero by the application program.

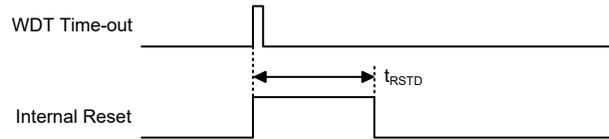
Bit 0 **WRF**: WDT control register software reset flag  
Refer to the Watchdog Timer Control Register section.

### IAP Reset

When a specific value of “55H” is written into the FC1 register, a reset signal will be generated to reset the whole device. Refer to the In Application Programming section for more associated details.

### Watchdog Time-out Reset during Normal Operation

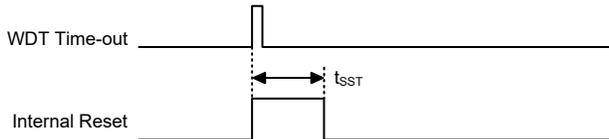
When the Watchdog time-out Reset during normal operations in the FAST or SLOW mode occurs, the Watchdog time-out flag TO will be set to “1”.



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

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

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



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

### Reset Initial Conditions

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

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

“u” stands for unchanged

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

| Item               | Condition after Reset                            |
|--------------------|--|
| Program Counter    | Reset to zero                                    |
| Interrupts         | All interrupts will be disabled                  |
| WDT, Time Bases    | Cleared after reset, WDT begins counting         |
| Timer Modules      | Timer Modules will be turned off                 |
| Input/Output Ports | I/O ports will be set 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.

| Name   | Power-On Reset | WDT Time-out<br>(Normal Operation) | WDT Time-out<br>(IDLE/SLEEP) |
|--------|----------------|------------------------------------|------------------------------|
| IAR0   | 0000 0000      | 0000 0000                          | uuuu uuuu                    |
| MP0    | 0000 0000      | 0000 0000                          | uuuu uuuu                    |
| IAR1   | 0000 0000      | 0000 0000                          | uuuu uuuu                    |
| MP1L   | 0000 0000      | 0000 0000                          | uuuu uuuu                    |
| MP1H   | 0000 0000      | 0000 0000                          | uuuu uuuu                    |
| ACC    | xxxx xxxx      | uuuu uuuu                          | uuuu uuuu                    |
| PCL    | 0000 0000      | 0000 0000                          | 0000 0000                    |
| TBLP   | xxxx xxxx      | uuuu uuuu                          | uuuu uuuu                    |
| TBLH   | xxxx xxxx      | uuuu uuuu                          | uuuu uuuu                    |
| TBHP   | --xx xxxx      | --uu uuuu                          | --uu uuuu                    |
| STATUS | xx00 xxxx      | uu1u uuuu                          | uu11 uuuu                    |
| PBP    | ---- --0       | ---- --0                           | ---- --u                     |
| IAR2   | 0000 0000      | 0000 0000                          | uuuu uuuu                    |
| MP2L   | 0000 0000      | 0000 0000                          | uuuu uuuu                    |
| MP2H   | 0000 0000      | 0000 0000                          | uuuu uuuu                    |
| RSTFC  | ---- 0x00      | ---- uuuu                          | ---- uuuu                    |
| INTEG  | --00 0000      | --00 0000                          | --uu uuuu                    |
| INTC0  | -000 0000      | -000 0000                          | -uuu uuuu                    |
| INTC1  | 0000 0000      | 0000 0000                          | uuuu uuuu                    |
| INTC2  | 0000 0000      | 0000 0000                          | uuuu uuuu                    |
| PA     | 1111 1111      | 1111 1111                          | uuuu uuuu                    |
| PAC    | 1111 1111      | 1111 1111                          | uuuu uuuu                    |
| PAPU   | 0000 0000      | 0000 0000                          | uuuu uuuu                    |
| PAWU   | 0000 0000      | 0000 0000                          | uuuu uuuu                    |
| LVDC   | --00 -000      | --00 -000                          | --uu -uuu                    |
| SCC    | 000- 0000      | 000- 0000                          | uuu- uuuu                    |
| WDTC   | 0101 0011      | 0101 0011                          | uuuu uuuu                    |
| TB0C   | 0--- -000      | 0--- -000                          | u--- -uuu                    |
| HIRCC  | ---- 0001      | ---- 0001                          | ---- uuuu                    |
| LVRC   | 0110 0110      | 0110 0110                          | uuuu uuuu                    |
| INTC3  | --00 --00      | --00 --00                          | --uu --uu                    |
| SADOL  | xxxx ----      | xxxx ----                          | uuuu ----<br>(ADRF5=0)       |
|        |                |                                    | uuuu uuuu<br>(ADRF5=1)       |
| SADOH  | xxxx xxxx      | xxxx xxxx                          | uuuu uuuu<br>(ADRF5=0)       |
|        |                |                                    | ---- uuuu<br>(ADRF5=1)       |
| SADC0  | 0000 0000      | 0000 0000                          | uuuu uuuu                    |
| SADC1  | 0000 -000      | 0000 -000                          | uuuu -uuu                    |
| SADC2  | 0--0 0000      | 0--0 0000                          | u--u uuuu                    |
| PB     | -111 1111      | -111 1111                          | -uuu uuuu                    |
| PBC    | -111 1111      | -111 1111                          | -uuu uuuu                    |

| Name       | Power-On Reset | WDT Time-out<br>(Normal Operation) | WDT Time-out<br>(IDLE/SLEEP) |
|------------|----------------|------------------------------------|------------------------------|
| PBPU       | -000 0000      | -000 0000                          | -uuu uuuu                    |
| CTMC0      | 0000 0---      | 0000 0---                          | uuuu u---                    |
| CTMC1      | 0000 0000      | 0000 0000                          | uuuu uuuu                    |
| CTMDL      | 0000 0000      | 0000 0000                          | uuuu uuuu                    |
| CTMDH      | 0000 0000      | 0000 0000                          | uuuu uuuu                    |
| CTMAL      | 0000 0000      | 0000 0000                          | uuuu uuuu                    |
| CTMAH      | 0000 0000      | 0000 0000                          | uuuu uuuu                    |
| CTMRP      | 0000 0000      | 0000 0000                          | uuuu uuuu                    |
| STMC0      | 0000 0---      | 0000 0---                          | uuuu u---                    |
| STMC1      | 0000 0000      | 0000 0000                          | uuuu uuuu                    |
| STMDL      | 0000 0000      | 0000 0000                          | uuuu uuuu                    |
| STMDH      | 0000 0000      | 0000 0000                          | uuuu uuuu                    |
| STMAL      | 0000 0000      | 0000 0000                          | uuuu uuuu                    |
| STMAH      | 0000 0000      | 0000 0000                          | uuuu uuuu                    |
| STMRP      | 0000 0000      | 0000 0000                          | uuuu uuuu                    |
| HXTC       | ---- -000      | ---- -000                          | ---- -uuu                    |
| PTM0C0     | 0000 0---      | 0000 0---                          | uuuu u---                    |
| PTM0C1     | 0000 0000      | 0000 0000                          | uuuu uuuu                    |
| PTM0DL     | 0000 0000      | 0000 0000                          | uuuu uuuu                    |
| PTM0DH     | ---- --00      | ---- --00                          | ---- --uu                    |
| PTM0AL     | 0000 0000      | 0000 0000                          | uuuu uuuu                    |
| PTM0AH     | ---- --00      | ---- --00                          | ---- --uu                    |
| PTM0RPL    | 0000 0000      | 0000 0000                          | uuuu uuuu                    |
| PTM0RPH    | ---- --00      | ---- --00                          | ---- --uu                    |
| CMPC       | -000 ---1      | -000 ---1                          | -uuu ---u                    |
| PC         | -111 1111      | -111 1111                          | -uuu uuuu                    |
| PCC        | -111 1111      | -111 1111                          | -uuu uuuu                    |
| PCPU       | -000 0000      | -000 0000                          | -uuu uuuu                    |
| LXTC       | ---- -000      | ---- -000                          | ---- -uuu                    |
| SIMC0      | 111- 0000      | 111- 0000                          | uuu- uuuu                    |
| SIMC1      | 1000 0001      | 1000 0001                          | uuuu uuuu                    |
| SIMD       | xxxx xxxx      | xxxx xxxx                          | uuuu uuuu                    |
| SIMA/SIMC2 | 0000 0000      | 0000 0000                          | uuuu uuuu                    |
| SIMTOC     | 0000 0000      | 0000 0000                          | uuuu uuuu                    |
| VBGRC      | ---- ---0      | ---- ---0                          | ---- ---u                    |
| TB1C       | 0--- -000      | 0--- -000                          | u--- -uuu                    |
| MF10       | 0000 0000      | 0000 0000                          | uuuu uuuu                    |
| MF11       | 0000 0000      | 0000 0000                          | uuuu uuuu                    |
| MF12       | --00 --00      | --00 --00                          | --uu --uu                    |
| SLEDC0     | 0000 0000      | 0000 0000                          | uuuu uuuu                    |
| SLEDC1     | 0000 0000      | 0000 0000                          | uuuu uuuu                    |
| IFS0       | 0000 0000      | 0000 0000                          | uuuu uuuu                    |
| PD         | 1111 1111      | 1111 1111                          | uuuu uuuu                    |
| PDC        | 1111 1111      | 1111 1111                          | uuuu uuuu                    |
| PDPU       | 0000 0000      | 0000 0000                          | uuuu uuuu                    |
| ORMC       | 0000 0000      | 0000 0000                          | 0000 0000                    |
| SLCDS3     | --00 0000      | --00 0000                          | --uu uuuu                    |

| Name    | Power-On Reset | WDT Time-out<br>(Normal Operation) | WDT Time-out<br>(IDLE/SLEEP) |
|---------|----------------|------------------------------------|------------------------------|
| IFS1    | ---- ---0      | ---- ---0                          | ---- ---u                    |
| PSD1    | 0000 0000      | 0000 0000                          | uuuu uuuu                    |
| LVPUC   | ---- ---0      | ---- ---0                          | ---- ---u                    |
| PAS0    | 0000 0000      | 0000 0000                          | uuuu uuuu                    |
| PAS1    | 0000 0000      | 0000 0000                          | uuuu uuuu                    |
| PBS0    | 0000 0000      | 0000 0000                          | uuuu uuuu                    |
| PBS1    | --00 0000      | --00 0000                          | --uu uuuu                    |
| PCS0    | 0000 0000      | 0000 0000                          | uuuu uuuu                    |
| PCS1    | --00 0000      | --00 0000                          | --uu uuuu                    |
| PDS0    | 0000 0000      | 0000 0000                          | uuuu uuuu                    |
| SLCDC0  | 0000 ----      | 0000 ----                          | uuuu ----                    |
| SLCDS0  | 0000 0000      | 0000 0000                          | uuuu uuuu                    |
| SLCDS1  | 0000 0000      | 0000 0000                          | uuuu uuuu                    |
| SLCDS2  | 0000 0000      | 0000 0000                          | uuuu uuuu                    |
| PSC0R   | ---- --00      | ---- --00                          | ---- --uu                    |
| PSC1R   | ---- --00      | ---- --00                          | ---- --uu                    |
| RSTC    | 0101 0101      | 0101 0101                          | uuuu uuuu                    |
| FC0     | 0000 0000      | 0000 0000                          | uuuu uuuu                    |
| FC1     | 0000 0000      | 0000 0000                          | uuuu uuuu                    |
| FC2     | ---- --00      | ---- --00                          | ---- --uu                    |
| FARL    | 0000 0000      | 0000 0000                          | uuuu uuuu                    |
| FARH    | --00 0000      | --00 0000                          | --uu uuuu                    |
| FD0L    | 0000 0000      | 0000 0000                          | uuuu uuuu                    |
| FD0H    | 0000 0000      | 0000 0000                          | uuuu uuuu                    |
| FD1L    | 0000 0000      | 0000 0000                          | uuuu uuuu                    |
| FD1H    | 0000 0000      | 0000 0000                          | uuuu uuuu                    |
| FD2L    | 0000 0000      | 0000 0000                          | uuuu uuuu                    |
| FD2H    | 0000 0000      | 0000 0000                          | uuuu uuuu                    |
| FD3L    | 0000 0000      | 0000 0000                          | uuuu uuuu                    |
| FD3H    | 0000 0000      | 0000 0000                          | uuuu uuuu                    |
| EEAL    | 0000 0000      | 0000 0000                          | uuuu uuuu                    |
| EEAH    | ---- --00      | ---- --00                          | ---- --uu                    |
| EED     | 0000 0000      | 0000 0000                          | uuuu uuuu                    |
| PTM1C0  | 0000 0---      | 0000 0---                          | uuuu u---                    |
| PTM1C1  | 0000 0000      | 0000 0000                          | uuuu uuuu                    |
| PTM1DL  | 0000 0000      | 0000 0000                          | uuuu uuuu                    |
| PTM1DH  | ---- --00      | ---- --00                          | ---- --uu                    |
| PTM1AL  | 0000 0000      | 0000 0000                          | uuuu uuuu                    |
| PTM1AH  | ---- --00      | ---- --00                          | ---- --uu                    |
| PTM1RPL | 0000 0000      | 0000 0000                          | uuuu uuuu                    |
| PTM1RPH | ---- --00      | ---- --00                          | ---- --uu                    |
| EEC     | 0000 0000      | 0000 0000                          | uuuu uuuu                    |
| U0SR    | 0000 1011      | 0000 1011                          | uuuu uuuu                    |
| U0CR1   | 0000 00x0      | 0000 00x0                          | uuuu uuuu                    |
| U0CR2   | 0000 0000      | 0000 0000                          | uuuu uuuu                    |
| U0CR3   | ---- ---0      | ---- ---0                          | ---- ---u                    |
| BRDH0   | 0000 0000      | 0000 0000                          | uuuu uuuu                    |

| Name     | Power-On Reset | WDT Time-out<br>(Normal Operation) | WDT Time-out<br>(IDLE/SLEEP) |
|----------|----------------|------------------------------------|------------------------------|
| BRDL0    | 0000 0000      | 0000 0000                          | uuuu uuuu                    |
| UFCR0    | --00 0000      | --00 0000                          | --uu uuuu                    |
| TXR_RXR0 | xxxx xxxx      | xxxx xxxx                          | uuuu uuuu                    |
| RxCNT0   | ---- -000      | ---- -000                          | ---- -uuu                    |
| U1SR     | 0000 1011      | 0000 1011                          | uuuu uuuu                    |
| U1CR1    | 0000 00x0      | 0000 00x0                          | uuuu uuuu                    |
| U1CR2    | 0000 0000      | 0000 0000                          | uuuu uuuu                    |
| U1CR3    | ---- - -0      | ---- - -0                          | ---- - -u                    |
| BRDH1    | 0000 0000      | 0000 0000                          | uuuu uuuu                    |
| BRDL1    | 0000 0000      | 0000 0000                          | uuuu uuuu                    |
| UFCR1    | --00 0000      | --00 0000                          | --uu uuuu                    |
| TXR_RXR1 | xxxx xxxx      | xxxx xxxx                          | uuuu uuuu                    |
| RxCNT1   | ---- -000      | ---- -000                          | ---- -uuu                    |

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 name PA~PD. 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            | —     | PB6   | PB5   | PB4   | PB3   | PB2   | PB1   | PB0   |
| PBC           | —     | PBC6  | PBC5  | PBC4  | PBC3  | PBC2  | PBC1  | PBC0  |
| PBPU          | —     | PBPU6 | PBPU5 | PBPU4 | PBPU3 | PBPU2 | PBPU1 | PBPU0 |
| PC            | —     | PC6   | PC5   | PC4   | PC3   | PC2   | PC1   | PC0   |
| PCC           | —     | PCC6  | PCC5  | PCC4  | PCC3  | PCC2  | PCC1  | PCC0  |
| PCPU          | —     | PCPU6 | PCPU5 | PCPU4 | PCPU3 | PCPU2 | PCPU1 | PCPU0 |
| PD            | PD7   | PD6   | PD5   | PD4   | PD3   | PD2   | PD1   | PD0   |
| PDC           | PDC7  | PDC6  | PDC5  | PDC4  | PDC3  | PDC2  | PDC1  | PDC0  |
| PDPU          | PDPU7 | PDPU6 | PDPU5 | PDPU4 | PDPU3 | PDPU2 | PDPU1 | PDPU0 |

“—”: Unimplemented, read as “0”

### I/O Logic Function Register List

## Pull-high Resistors

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

Note that the pull-high resistor can be controlled by the relevant pull-high control register only when the pin-shared functional pin is selected as a digital input or NMOS output. Otherwise, the pull-high resistors 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 or D. 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:** Pull-high resistor selection when low voltage power supply

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

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

This bit is used to select the pull-high resistor value for low voltage power supply applications. The LVPU bit is only available when the corresponding pin pull-high function is enabled by setting the relevant pull-high control bit high. This bit will have no effect when the pull-high function is disabled.

## Port A Wake-up

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

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

• PAWU Register

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

Bit 7~0 PAWU7~PAWU0: PA7~PA0 wake-up function control  
0: Disable  
1: Enable

**I/O Port Control Registers**

Each I/O port has its own control register known as PAC~PDC, 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 set as a CMOS output. If the pin is currently set 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  |
| 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 or D. However, the actual available bits for each I/O Port may be different.

**I/O Port Source Current Selection**

The device supports different output source current driving capability for each I/O port. With the selection register, SLEDCn, specific I/O port can support four levels of the source current driving capability. These source current selection bits are available only when the corresponding pin is configured as a CMOS output. Otherwise, these select bits have no effect. Users should refer to the Input/Output Characteristics section to select the desired output 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 |

I/O Port Source Current Control Register 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**: PB6~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**: PD7~PD4 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 **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**: PC6~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.)

## 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, labeled as IFS<sub>i</sub>, which can select the desired functions of the multi-function pin-shared pins.

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

| Register Name | Bit   |        |        |        |            |            |        |         |
|---------------|-------|--------|--------|--------|------------|------------|--------|---------|
|               | 7     | 6      | 5      | 4      | 3          | 2          | 1      | 0       |
| PAS0          | PAS07 | PAS06  | PAS05  | PAS04  | PAS03      | PAS02      | PAS01  | PAS00   |
| PAS1          | PAS17 | PAS16  | PAS15  | PAS14  | PAS13      | PAS12      | PAS11  | PAS10   |
| PBS0          | PBS07 | PBS06  | PBS05  | PBS04  | PBS03      | PBS02      | PBS01  | PBS00   |
| PBS1          | —     | —      | PBS15  | PBS14  | PBS13      | PBS12      | PBS11  | PBS10   |
| PCS0          | PCS07 | PCS06  | PCS05  | PCS04  | PCS03      | PCS02      | PCS01  | PCS00   |
| PCS1          | —     | —      | PCS15  | PCS14  | PCS13      | PCS12      | PCS11  | PCS10   |
| PDS0          | PDS07 | PDS06  | PDS05  | PDS04  | PDS03      | PDS02      | PDS01  | PDS00   |
| PDS1          | PDS17 | PDS16  | PDS15  | PDS14  | PDS13      | PDS12      | PDS11  | PDS10   |
| IFS0          | RX1PS | INT2PS | INT1PS | INT0PS | SDI_ SDAPS | SCK_ SCLPS | SCSBPS | RX0PS   |
| IFS1          | —     | —      | —      | —      | —          | —          | —      | PTCK1PS |

**Pin-shared Function Selection Register List**

• **PAS0 Register**

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

- Bit 7~6     **PAS07~PAS06:** PA3 Pin-Shared Function Selection  
             00: PA3  
             01: SDI/SDA  
             10: CMPO  
             11: SCOM3/SSEG3
- Bit 5~4     **PAS05~PAS04:** PA2 Pin-Shared Function Selection  
             00: PA2/INT1  
             01: PA2/INT1  
             10: PA2/INT1  
             11: SCOM2/SSEG2
- Bit 3~2     **PAS03~PAS02:** PA1 Pin-Shared Function Selection  
             00: PA1/INT0  
             01: PA1/INT0  
             10: SDO  
             11: SCOM1/SSEG1
- Bit 1~0     **PAS01~PAS00:** PA0 Pin-Shared Function Selection  
             00: PA0  
             01: PA0  
             10: STP  
             11: SCOM0/SSEG0

• **PAS1 Register**

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

- Bit 7~6     **PAS17~PAS16:** PA7 Pin-Shared Function Selection  
             00: PA7  
             01: PTP0  
             10: SCOM7/SSEG7  
             11: AN6
- Bit 5~4     **PAS15~PAS14:** PA6 Pin-Shared Function Selection  
             00: PA6/CTCK  
             01: SCOM6/SSEG6  
             10: AN5  
             11: VREF
- Bit 3~2     **PAS13~PAS12:** PA5 Pin-Shared Function Selection  
             00: PA5  
             01: SCOM5/SSEG5  
             10: AN4  
             11: VREFI
- Bit 1~0     **PAS11~PAS10:** PA4 Pin-Shared Function Selection  
             00: PA4/PTCK0  
             01: PA4/PTCK0  
             10: SCOM4/SSEG4  
             11: AN3

• **PBS0 Register**

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

- Bit 7~6     **PBS07~PBS06:** PB3 Pin-Shared Function Selection  
00: PB3  
01: CTP  
10: SCOM11/SSEG11  
11: AN7
- Bit 5~4     **PBS05~PBS04:** PB2 Pin-Shared Function Selection  
00: PB2/STCK  
01: STP  
10: SCOM10/SSEG10  
11: AN2
- Bit 3~2     **PBS03~PBS02:** PB1 Pin-Shared Function Selection  
00: PB1/INT1  
01: SCOM9/SSEG9  
10: AN1  
11: XT2
- Bit 1~0     **PBS01~PBS00:** PB0 Pin-Shared Function Selection  
00: PB0/INT0  
01: SCOM8/SSEG8  
10: AN0  
11: XT1

• **PBS1 Register**

| Bit  | 7 | 6 | 5     | 4     | 3     | 2     | 1     | 0     |
|------|---|---|-------|-------|-------|-------|-------|-------|
| Name | — | — | PBS15 | PBS14 | PBS13 | PBS12 | PBS11 | PBS10 |
| 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     **PBS15~PBS14:** PB6 Pin-Shared function selection  
00: PB6  
01: SCK/SCL  
10: CMPINP  
11: SCOM14/SSEG14
- Bit 3~2     **PBS13~PBS12:** PB5 Pin-Shared function selection  
00: PB5  
01: SCS  
10: CMPINN  
11: SCOM13/SSEG13
- Bit 1~0     **PBS11~PBS10:** PB4 Pin-Shared function selection  
00: PB4  
01: CLO  
10: SCOM12/SSEG12  
11: AN8

• **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-Shared Function Selection  
             00: PC3  
             01: PC3  
             10: SDO  
             11: SCOM18/SSEG18
- Bit 5~4     **PCS05~PCS04:** PC2 Pin-Shared Function Selection  
             00: PC2  
             01: PC2  
             10: SDO  
             11: SCOM17/SSEG17
- Bit 3~2     **PCS05~PCS04:** PC1 Pin-Shared Function Selection  
             00: PC1  
             01: RX0/TX0  
             10: SCOM16/SSEG16  
             11: OSC2
- Bit 1~0     **PCS01~PCS00:** PC0 Pin-Shared Function Selection  
             00: PC0  
             01: TX0  
             10: SCOM15/SSEG15  
             11: OSC1

• **PCS1 Register**

| Bit  | 7 | 6 | 5     | 4     | 3     | 2     | 1     | 0     |
|------|---|---|-------|-------|-------|-------|-------|-------|
| Name | — | — | PCS15 | PCS14 | PCS13 | PCS12 | PCS11 | PCS10 |
| 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     **PCS15~PCS14:** PC6 Pin-Shared Function Selection  
             00: PC6  
             01: PC6  
             10: SCS  
             11: SCOM21/SSEG21
- Bit 3~2     **PCS13~PCS12:** PC5 Pin-Shared Function Selection  
             00: PC5  
             01: PC5  
             10: SCK/SCL  
             11: SCOM20/SSEG20
- Bit 1~0     **PCS11~PCS10:** PC4 Pin-Shared Function Selection  
             00: PC4  
             01: PC4  
             10: SDI/SDA  
             11: SCOM19/SSEG19

• **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-Shared Function Selection  
00: PD3  
01: CTP  
10: SCOM25/SSEG25  
11: AN9
- Bit 5~4     **PDS05~PDS04:** PD2 Pin-Shared Function Selection  
00: PD2  
01: TX0  
10: SCOM24/SSEG24  
11: AN10
- Bit 3~2     **PDS03~PCS02:** PD1 Pin-Shared Function Selection  
00: PD1  
01: RX0/TX0  
10: SCOM23/SSEG23  
11: AN11
- Bit 1~0     **PCS01~PCS00:** PD0 Pin-Shared Function Selection  
00: PD0  
01: PD0  
10: PTP0  
11: SCOM22/SSEG22

• **PDS1 Register**

| Bit  | 7     | 6     | 5     | 4     | 3     | 2     | 1     | 0     |
|------|-------|-------|-------|-------|-------|-------|-------|-------|
| Name | PDS17 | PDS16 | PDS15 | PDS14 | PDS13 | PDS12 | PDS11 | PDS10 |
| 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     **PDS17~PDS16:** PD7 Pin-Shared Function Selection  
00: PD7/INT2/PTCK1  
01: PD7/INT2/PTCK1  
10: RX1/TX1  
11: SCOM29/SSEG29
- Bit 5~4     **PDS15~PDS14:** PD6 Pin-Shared Function Selection  
00: PD6  
01: PTP1  
10: TX1  
11: SCOM28/SSEG28
- Bit 3~2     **PDS13~PCS12:** PD5 Pin-Shared Function Selection  
00: PD5/INT2/PTCK1  
01: PD5/INT2/PTCK1  
10: RX1/TX1  
11: SCOM27/SSEG27
- Bit 1~0     **PCS11~PCS10:** PD4 Pin-Shared Function Selection  
00: PD4  
01: PTP1  
10: TX1  
11: SCOM26/SSEG26

• **IFS0 Register**

| Bit  | 7     | 6      | 5      | 4      | 3         | 2         | 1      | 0     |
|------|-------|--------|--------|--------|-----------|-----------|--------|-------|
| Name | RX1PS | INT2PS | INT1PS | INT0PS | SDI_SDAPS | SCK_SCLPS | SCSBPS | RX0PS |
| 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      **RX1PS**: RX1/TX1 input source pin selection  
0: PD5  
1: PD7
- Bit 6      **INT2PS**: INT2 input source pin selection  
0: PD7  
1: PD5
- Bit 5      **INT1PS**: INT1 input source pin selection  
0: PB1  
1: PA2
- Bit 4      **INT0PS**: INT0 input source pin selection  
0: PB0  
1: PA1
- Bit 3      **SDI\_SDAPS**: SDI/SDA input source pin selection  
0: PC4  
1: PA3
- Bit 2      **SCK\_SCLPS**: SCK/SCL input source pin selection  
0: PC5  
1: PB6
- Bit 1      **SCSBPS**:  $\overline{\text{SCS}}$  input source pin selection  
0: PC6  
1: PB5
- Bit 0      **RX0PS**: RX0/TX0 input source pin selection  
0: PD1  
1: PC1

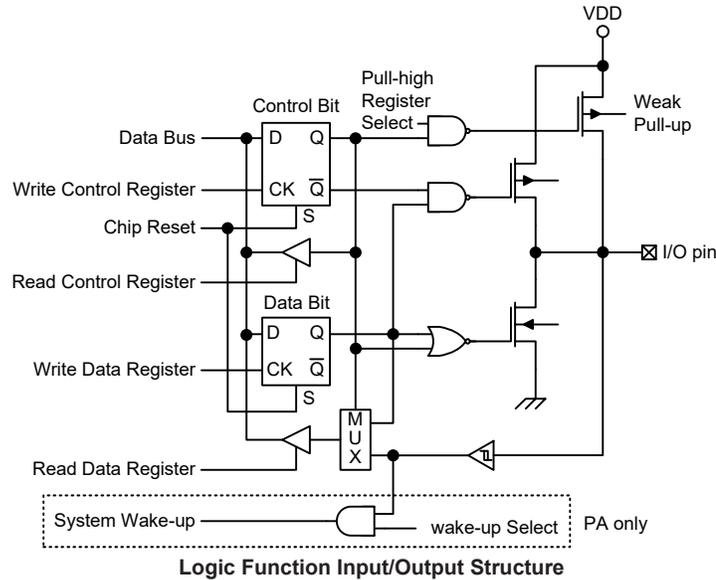
• **IFS1 Register**

| Bit  | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0       |
|------|---|---|---|---|---|---|---|---------|
| Name | — | — | — | — | — | — | — | PTCK1PS |
| R/W  | — | — | — | — | — | — | — | R/W     |
| POR  | — | — | — | — | — | — | — | 0       |

- Bit 7~1      Unimplemented, read as “0”
- Bit 0      **PTCK1PS**: PTCK1 input source pin selection  
0: PD5  
1: PD7

## 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 drawing, it is supplied as a guide only to assist with the functional understanding of the I/O logic function. 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 are then programmed to set some pins as outputs, these output pins will have an initial high output value unless the associated port data registers are first programmed. Selecting which pins are inputs and which are outputs can be achieved byte-wide by loading the correct values into the appropriate port control register or by programming individual bits in the port control register using the "SET [m].i" and "CLR [m].i" instructions. Note that when using these bit control instructions, a read-modify-write operation takes place. The microcontroller must first read in the data on the entire port, modify it to the required new bit values and then rewrite this data back to the output ports.

Port A has the additional capability of providing wake-up function. 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 set to have this function.

## Timer Modules – TM

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

### Introduction

The device contains several 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                | √              | √              | √              |
| Compare Match Output         | √              | √              | √              |
| PWM Output                   | √              | √              | √              |
| Single Pulse Output          | —              | √              | √              |
| 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 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 xTnCK2~xTnCK0 bits in the xTMn control registers, where “x” stands for C, S or P type TM and “n” stands for the specific TM serial number. For the CTM and STM there are no serial number “n” in the relevant pins, registers and control bits since there is only one CTM and one STM in the device. 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 xTCKn pin. The xTCKn 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

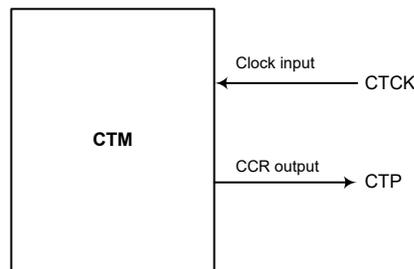
Each of the TMs, irrespective of what type, has one TM input pin, with the label xTCKn. The xTMn input pin, xTCKn, is essentially a clock source for the xTMn and is selected using the xTnCK2~xTnCK0 bits in the xTMnC0 register. This external TM input pin allows an external clock source to drive the internal TM. The xTCKn input pin can be chosen to have either a rising or falling active edge. The STCK and PTCKn pins are also used as the external trigger input pin in single pulse output mode for the STM and PTMn respectively.

The TMs each has one output pin, xTPn. When the TM is in the Compare Match Output Mode, the pin can be controlled by the xTMn to switch to a high or low level or to toggle when a compare match situation occurs. The output pin is also the pin where the TM generates the PWM output waveform.

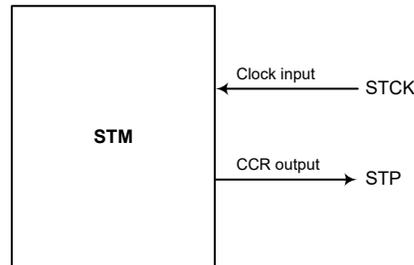
As the TM input and output pins are pin-shared with other functions, the TM input and output functions must first be setup using the relevant pin-shared function selection bits described in the Pin-shared Function section. The details of the pin-shared function selection are described in the pin-shared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

| CTM   |        | STM   |        | PTM0  |        | PTM1  |        |
|-------|--------|-------|--------|-------|--------|-------|--------|
| Input | Output | Input | Output | Input | Output | Input | Output |
| CTCK  | CTP    | STCK  | STP    | PTCK0 | PTP0   | PTCK1 | PTP1   |

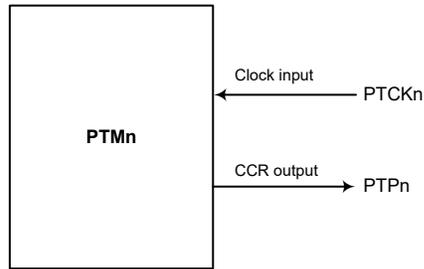
**TM External Pins**



**CTM Function Pin Block Diagram**



**STM Function Pin Block Diagram**

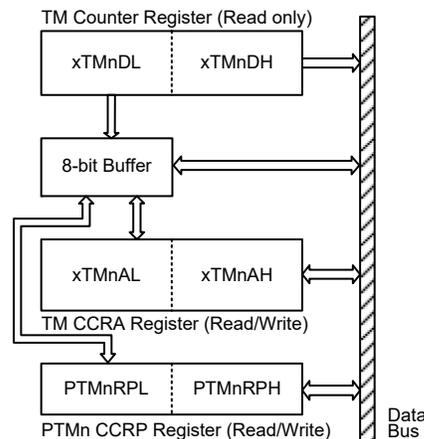


PTMn Function Pin Block Diagram (n=0~1)

### Programming Considerations

The TM Counter Registers and the 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 xTMnAL and PTMnRPL, using the following access procedures. Accessing the CCRA or CCRP low byte registers without following these access procedures will result in unpredictable values.

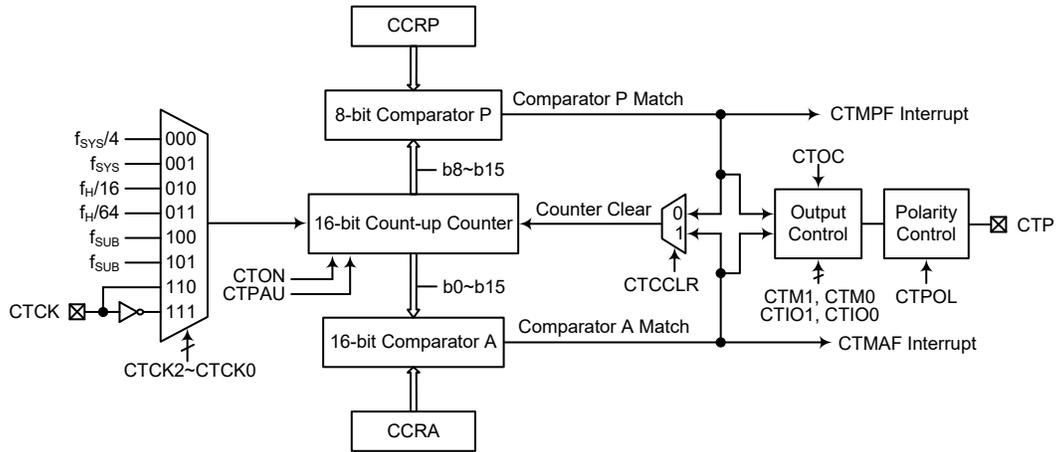


The following steps show the read and write procedures:

- Writing Data to CCRA or CCRP
  - ♦ Step 1. Write data to Low Byte xTMnAL or PTMnRPL
    - Note that here data is only written to the 8-bit buffer.
  - ♦ Step 2. Write data to High Byte xTMnAH or PTMnRPH
    - 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, xTMnAH or PTMnRPH
    - 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, xTMnAL or PTMnRPL
    - 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 one external output pin.



Note: The CTM external pins are pin-shared with other functions, so before using the CTM function, ensure that the relevant pin-shared function registers have been set properly to enable the CTM pin function. The CTCK pin, if used, must also be set as an input by setting the corresponding bit in the port control register.

**16-bit Compact Type TM Block Diagram**

### Compact Type 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 one output pin. All operating setup conditions are selected using relevant internal registers.

### 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 Register 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:** CTM counter clock selection  
 000:  $f_{SYS}/4$   
 001:  $f_{SYS}$   
 010:  $f_H/16$   
 011:  $f_H/64$   
 100:  $f_{SUB}$   
 101:  $f_{SUB}$   
 110: 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 or the PWM Output Mode then the CTM output pin will be reset to its initial condition, as specified by the CTOC bit, when the CTON bit changes from low to high.

**Bit 2~0**    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**: CTM operating mode selection

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

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

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

Bit 3 **CTOC**: CTM CTP output control

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

Output Mode. It has no effect if the CTM is in the Timer/Counter Mode. In the Compare Match Output Mode it determines the logic level of the CTM output pin before a compare match occurs. In the PWM Output Mode it determines if the PWM signal is active high or active low.

Bit 2     **CTPOL**: CTM CTP output polarity control  
           0: Non-invert  
           1: Invert

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

Bit 1     **CTDPX**: CTM PWM 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     **D7~D0**: 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 7~2     **D15~D8**: 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     **D7~D0**: 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      **D15~D8**: 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      **D7~D0**: CTM CCRP 8-bit register, compared with the CTM Counter bit 15 ~ bit 8  
 Comparator P Match Period=  
 0: 65536 CTM clocks  
 1~255:  $256 \times (1\sim255)$  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 Output Mode or Timer/Counter Mode. The operating mode is selected using the CTM1 and CTM0 bits in the CTMC1 register.

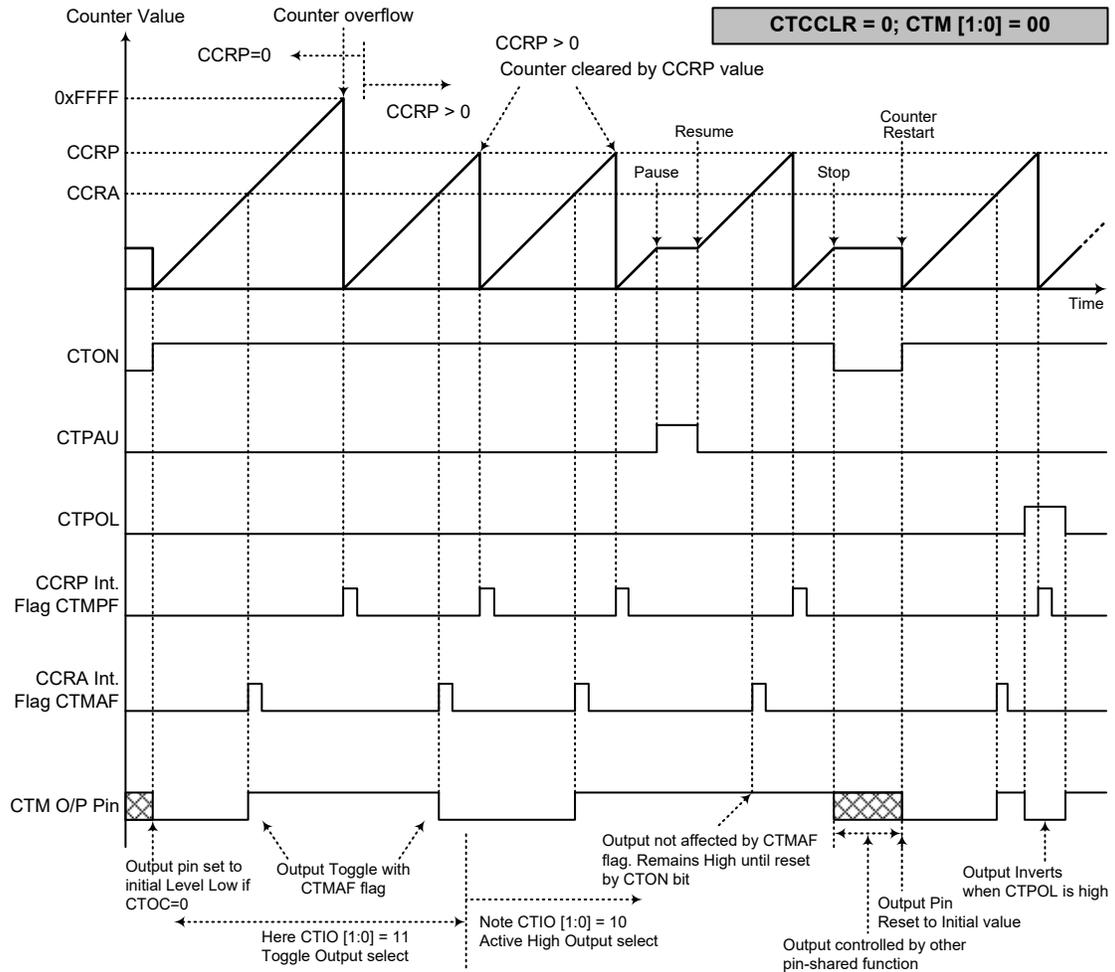
**Compare Match Output Mode**

To select this mode, bits CTM1 and CTM0 in the CTMC1 register, should be set to "00" respectively. In this mode once the counter is enabled and running it can be cleared by three methods. These are a counter overflow, a compare match from Comparator A and a compare match from Comparator P. When the CTCCLR bit is low, there are two ways in which the counter can be cleared. One is when a compare match 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.

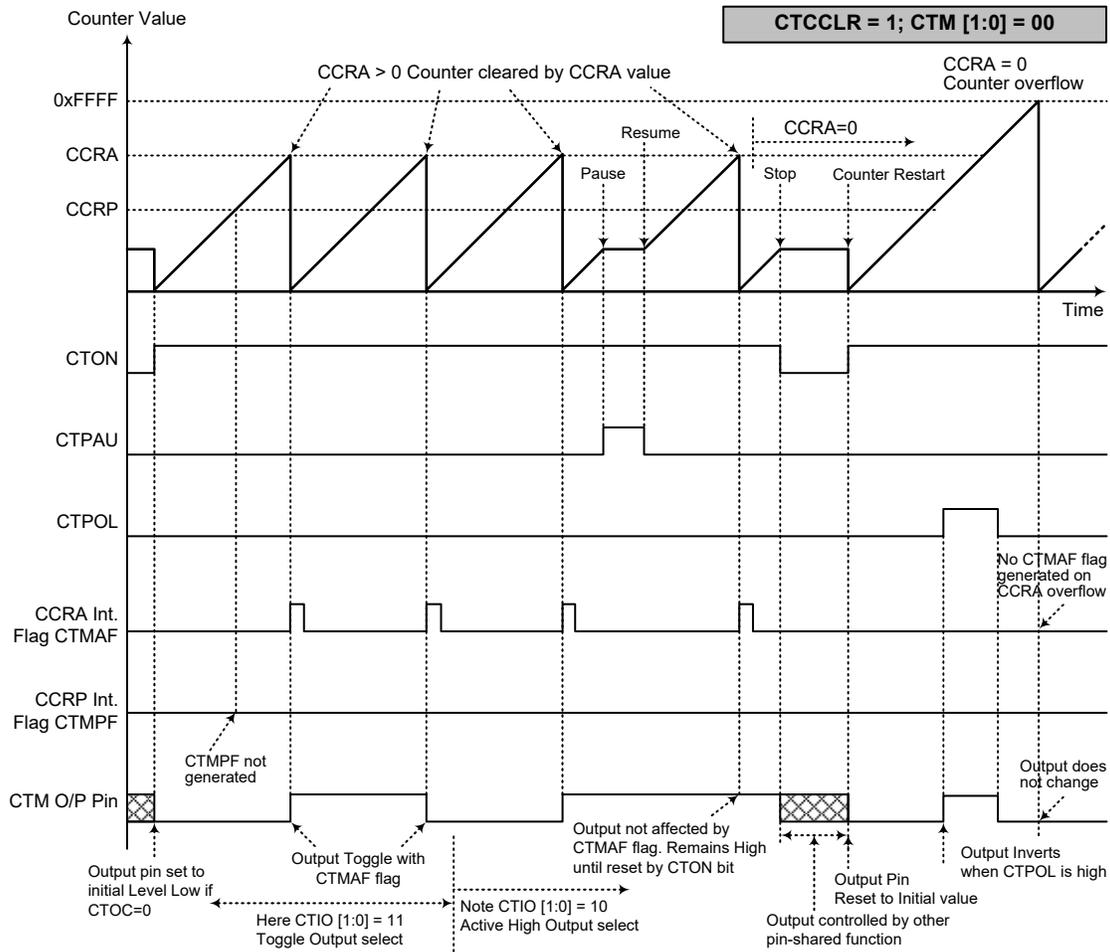
If the CTCCLR bit in the CTMC1 register is high then the counter will be cleared when a compare match occurs from Comparator A. However, here only the CTMAF interrupt request flag will be generated even if the value of the CCRP bits is less than that of the CCRA registers. Therefore when CTCCLR is high 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.



- 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 functions.

**PWM Output Mode**

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

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

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

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

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

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

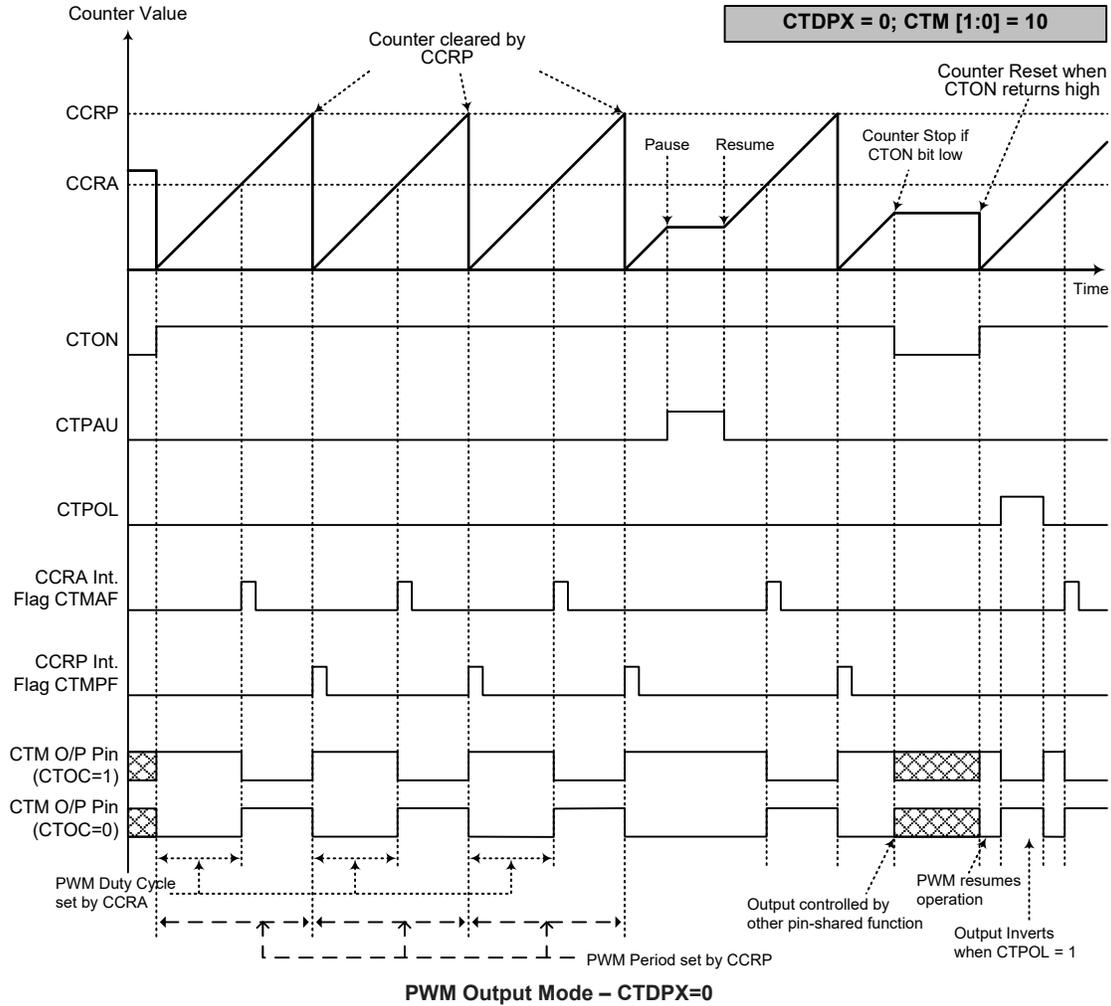
The CTM PWM output frequency= $(f_{SYS}/4)/512=f_{SYS}/2048=7.8125\text{kHz}$ , duty=128/512=25%,

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%.

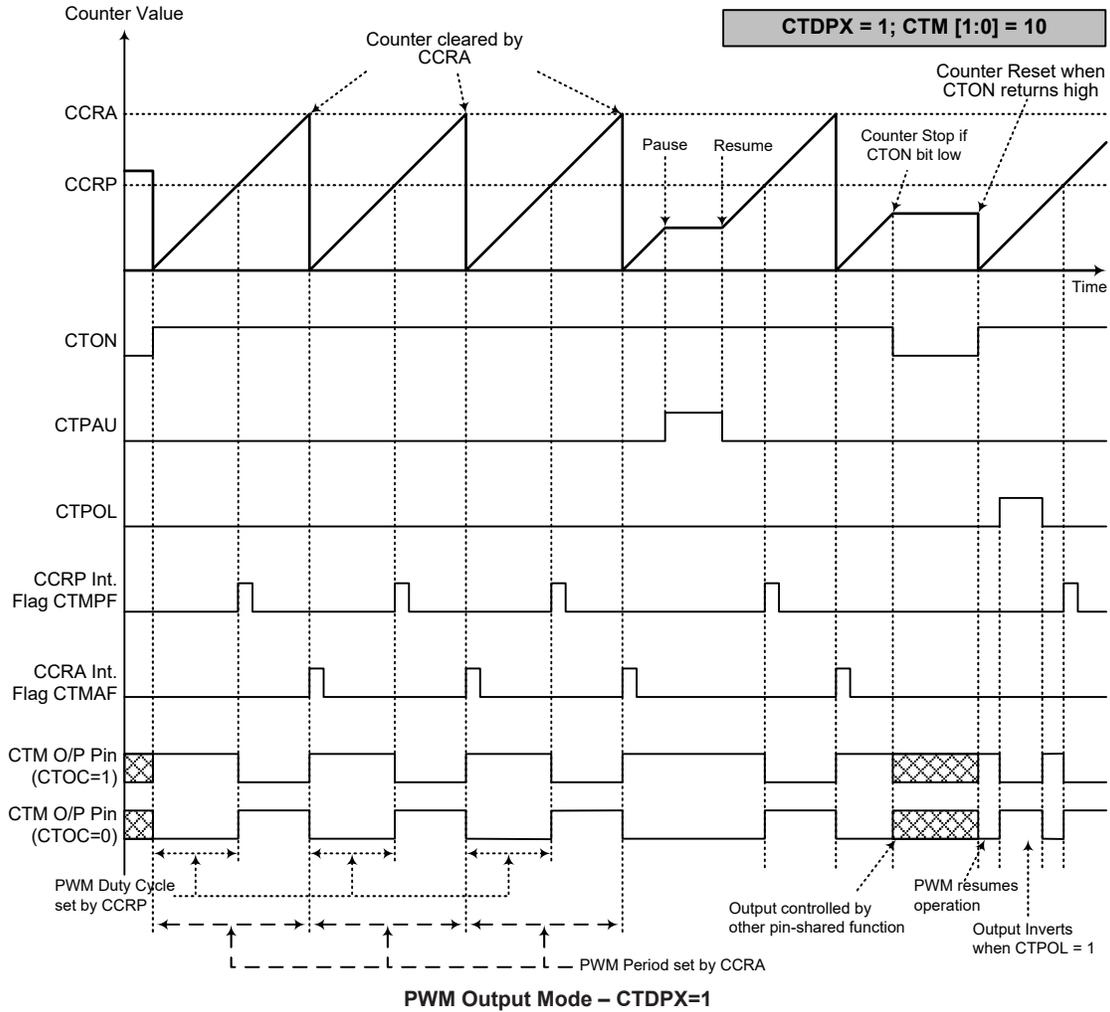
• **16-bit CTM, PWM Output Mode, Edge-aligned Mode, CTD PX=1**

| 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.



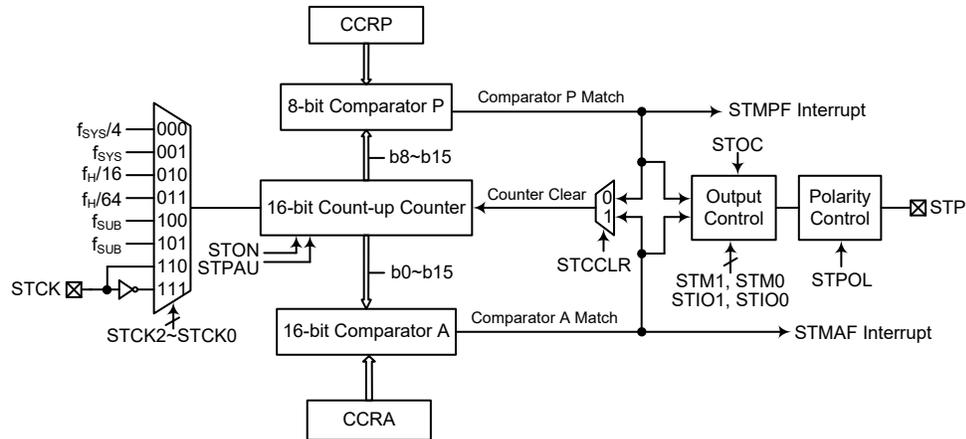
- Note: 1. Here CTDPX=0 – Counter cleared by CCRP  
 2. A counter clear sets the PWM Period  
 3. The internal PWM function continues even when CTIO[1:0]=00 or 01  
 4. The CTCCLR bit has no influence on PWM operation



- 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 CTCCLR bit has no influence on PWM operation

## Standard Type TM – STM

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



Note: The STM external pins are pin-shared with other functions, so before using the STM function, ensure that the relevant pin-shared function registers have been set properly to enable the STM pin function. The STCK pin, if used, must also be set as an input by setting the corresponding bits in the port control register.

16-bit Standard Type TM Block Diagram

### Standard Type 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 with the 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, an STM interrupt signal will also usually be generated. The Standard Type TM can operate in a number of different operational modes, can be driven by different clock sources including an input pin and can also control the output 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 set 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 Register 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**: STM counter clock selection

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

These three bits are used to select the clock source for the STM. The external pin clock source can be chosen to be active on the rising or falling edge. The clock source  $f_{SYS}$  is the system clock, while  $f_H$  and  $f_{SUB}$  are other internal clocks, the details of which can be found in the 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, 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**: STM operating mode selection  
 00: Compare Match Output Mode  
 01: Undefined  
 10: PWM Output Mode or Single Pulse Output Mode  
 11: Timer/Counter Mode

These bits 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**: STM external pin function selection

- Compare Match Output Mode  
 00: No change  
 01: Output low  
 10: Output high  
 11: Toggle output
- PWM Output Mode/Single Pulse Output Mode  
 00: PWM output inactive state  
 01: PWM output active state  
 10: PWM output  
 11: Single Pulse Output

Timer/Counter Mode  
 Unused

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

- Bit 3     **STOC**: STM STP output control

- Compare Match Output Mode  
 0: Initial low  
 1: Initial high
- PWM Output Mode/Single Pulse Output Mode  
 0: Active low  
 1: Active high

This is the output control bit for the STM output pin. Its operation depends upon whether STM is being used in the Compare Match Output Mode or in the PWM Output Mode/Single Pulse Output Mode. It has no effect if the STM is in the Timer/

Counter Mode. In the Compare Match Output Mode it determines the logic level of the STM output pin before a compare match occurs. In the PWM Output Mode it determines if the PWM signal is active high or active low. In the Single Pulse Output Mode it determines the logic level of the STM output pin when the STON bit changes from low to high.

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

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

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

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

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

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

• **STMDL Register**

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

Bit 7~0 **D7~D0**: STM Counter Low Byte Register bit 7 ~ bit 0  
STM 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 **D15~D8**: 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 **D7~D0**: 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      **D15~D8:** STM CCRA High Byte Register bit 7 ~ bit 0  
STM 16-bit CCRA bit 15 ~ bit 8

• **STMRP Register**

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

Bit 7~0      **D7~D0:** STM 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 four operating modes, Compare Match Output Mode, PWM Output Mode, Single Pulse Output Mode or Timer/Counter Mode. The operating mode is selected using the STM1 and STM0 bits in the STMC1 register.

### Compare Match Output Mode

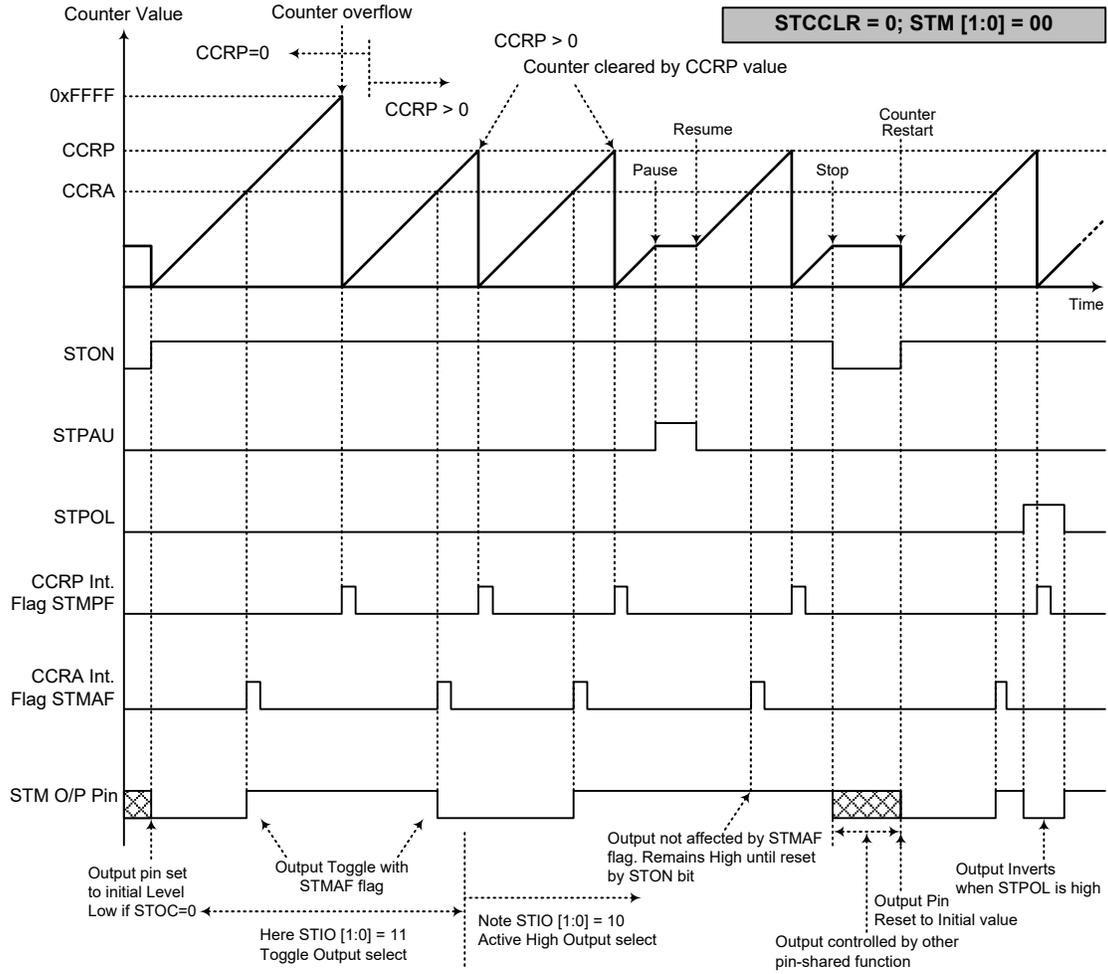
To select this mode, bits STM1 and STM0 in the STMC1 register, should be set to "00" respectively. In this mode once the counter is enabled and running it can be cleared by three methods. These are a counter overflow, a compare match from Comparator A and a compare match from Comparator P. When the STCCLR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

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

If the CCRA bits are all zero, the counter will overflow when it reaches its maximum 16-bit, FFFF Hex, value, however here the STMAF interrupt request flag will not be generated.

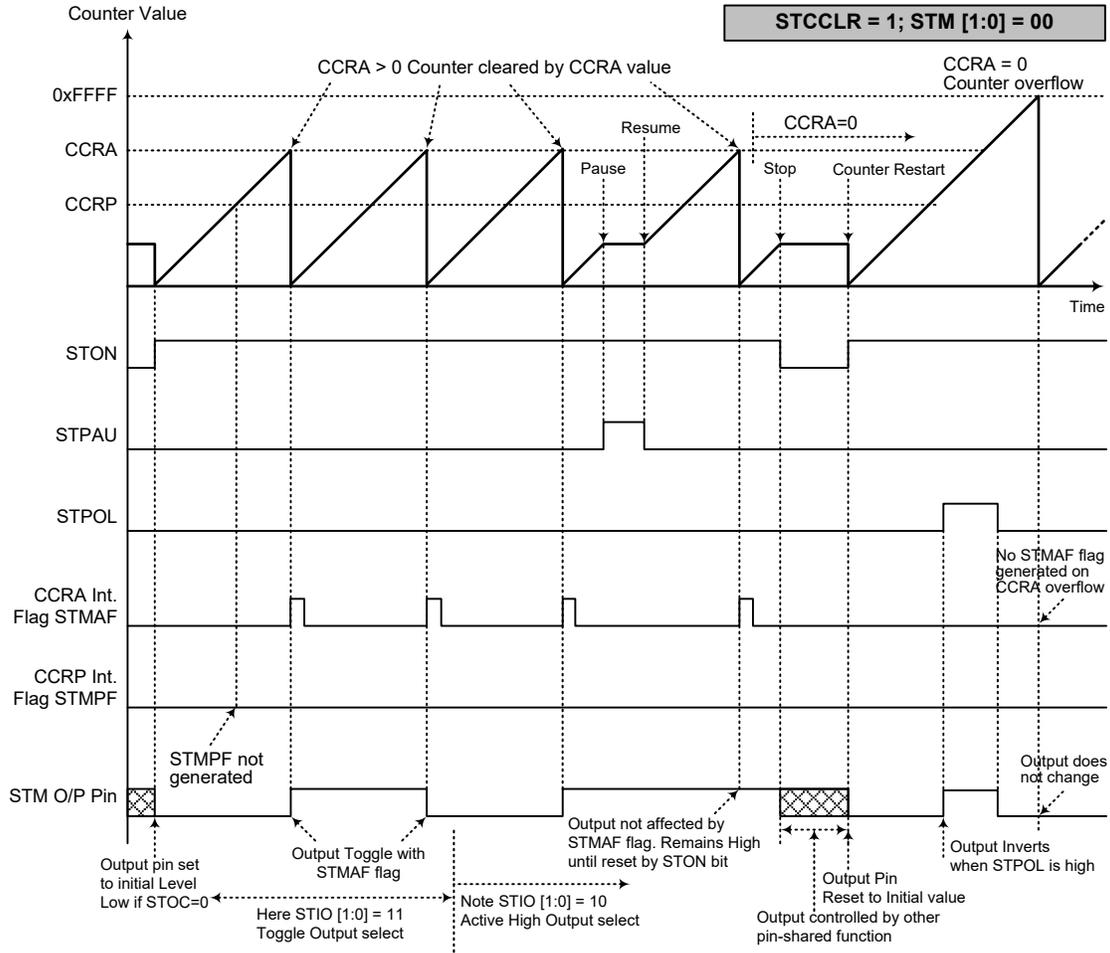
As the name of the mode suggests, after a comparison is made, the STM output pin, will change state. The STM output pin condition however only changes state when an STMAF interrupt request flag is generated after a compare match occurs from Comparator A. The STMPF interrupt request

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 an 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 an STON bit rising edge
4. The STMPF flag is not generated when STCCLR=1

**Timer/Counter Mode**

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

**PWM Output Mode**

To select this mode, bits STM1 and STM0 in the STMC1 register should be set to “10” respectively 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 Output Mode, the STCCLR bit has no effect on 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 Output 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,

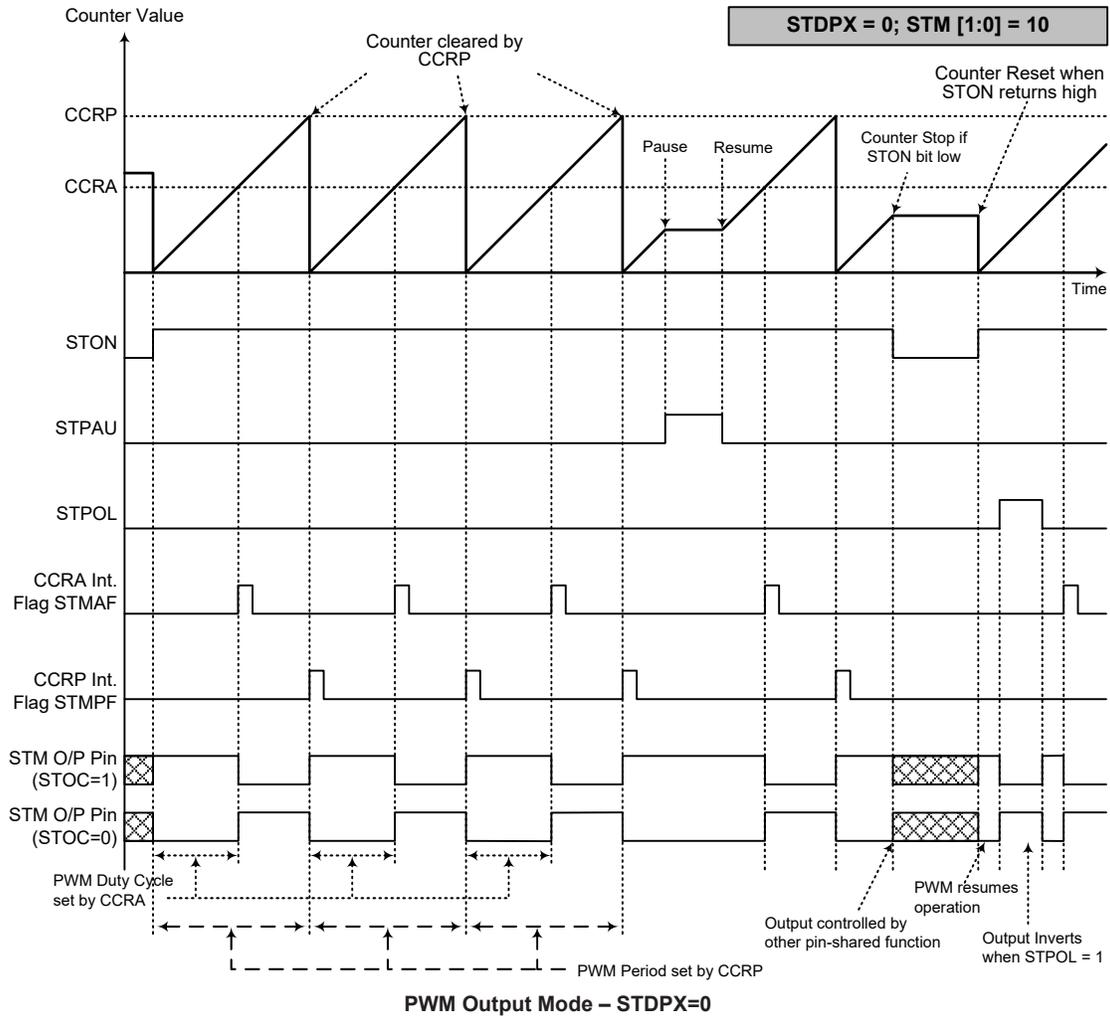
The STM PWM output frequency= $(f_{SYS}/4)/(2\times 256)=f_{SYS}/2048=7.8125\text{kHz}$ , duty= $128/(2\times 256)=25\%$ ,

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%.

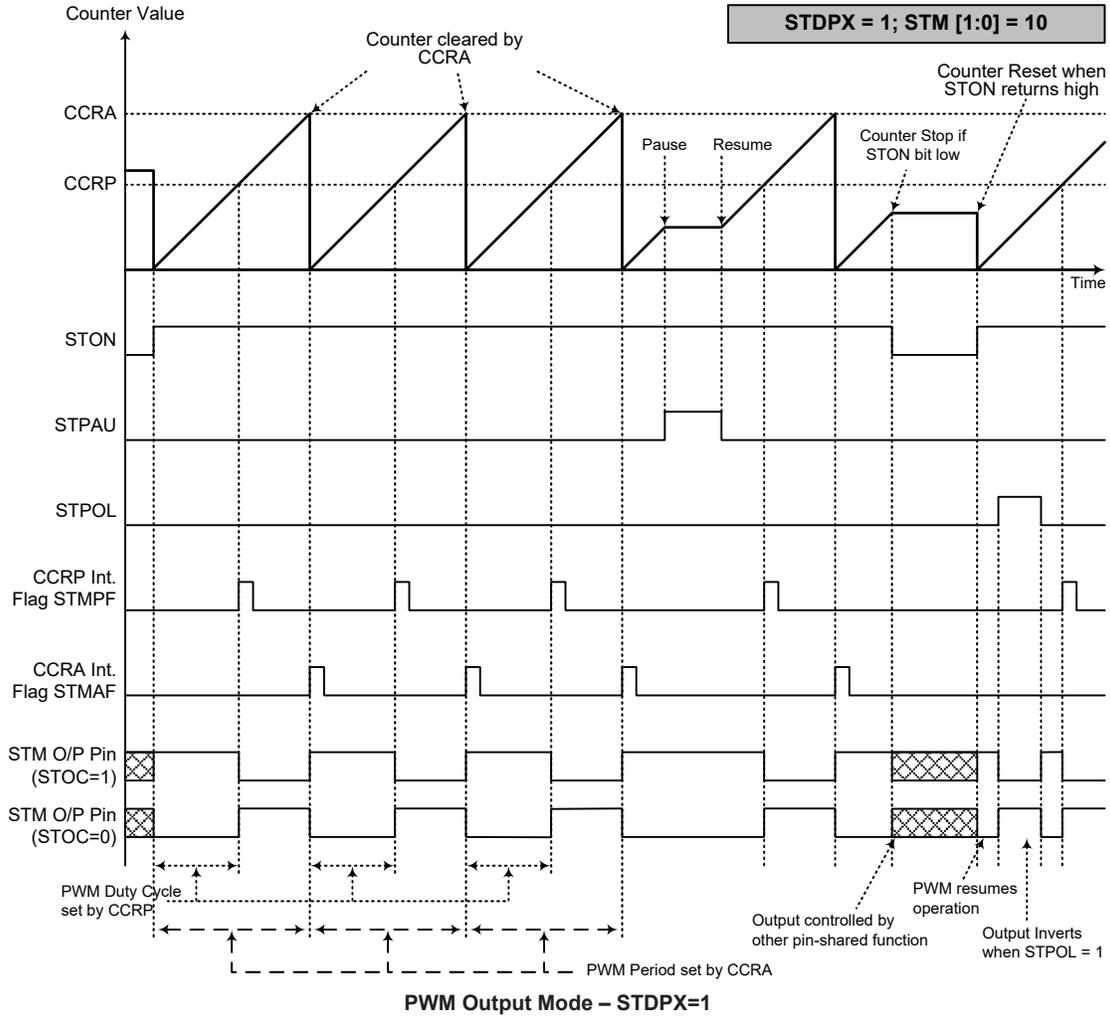
• **16-bit STM, PWM Output Mode, Edge-aligned Mode, STDPX=1**

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

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



- Note: 1. Here STDPX=0 – Counter cleared by CCRP  
 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



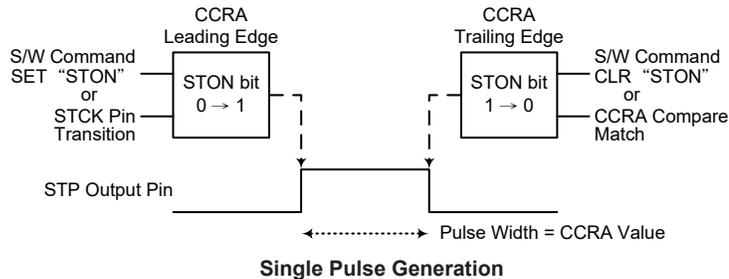
- Note: 1. Here STDPX=1 – Counter cleared by CCRA  
 2. A counter clear sets the PWM Period  
 3. The internal PWM function continues even when STIO[1:0]=00 or 01  
 4. The STCCLR bit has no influence on PWM operation

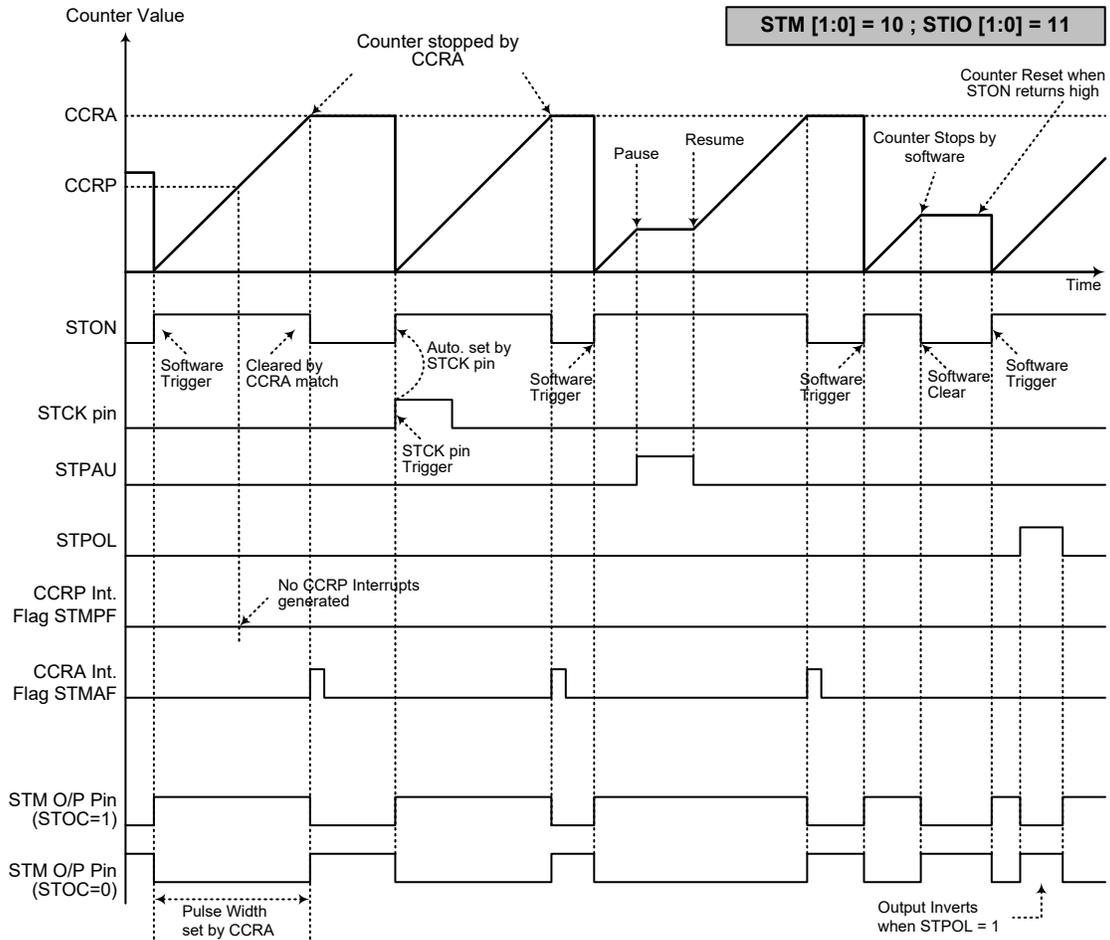
**Single Pulse Output Mode**

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

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

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



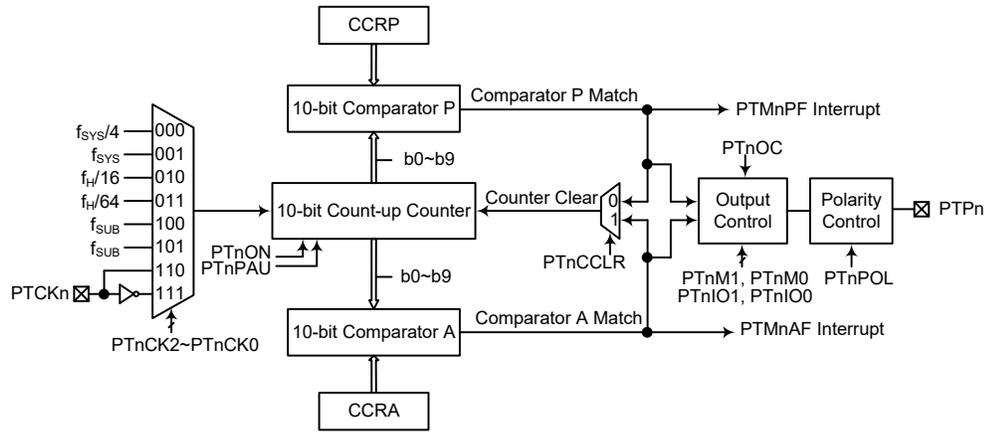


**Single Pulse Output Mode**

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

## Periodic Type TM – PTM

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



Note: The PTMn external pins are pin-shared with other functions, so before using the PTMn function, ensure that the relevant pin-shared function registers have be set properly to enable the PTMn pin function. The PTCKn pin, if used, must also be set as an input by setting the corresponding bits in the port control register.

**10-bit Periodic Type TM Block Diagram (n=0~1)**

### Periodic Type 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 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 PTnON 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 PTMn 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 an output pin. 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       |
| PTMnCO        | PTnPAU | PTnCK2 | PTnCK1 | PTnCK0 | PTnON  | —      | —      | —       |
| PTMnC1        | PTnM1  | PTnM0  | PTnIO1 | PTnIO0 | PTnOC  | PTnPOL | D1     | PTnCCLR |
| PTMnDL        | D7     | D6     | D5     | D4     | D3     | D2     | D1     | D0      |
| PTMnDH        | —      | —      | —      | —      | —      | —      | D9     | D8      |
| PTMnAL        | D7     | D6     | D5     | D4     | D3     | D2     | D1     | D0      |
| PTMnAH        | —      | —      | —      | —      | —      | —      | D9     | D8      |
| PTMnRPL       | PTnRP7 | PTnRP6 | PTnRP5 | PTnRP4 | PTnRP3 | PTnRP2 | PTnRP1 | PTnRP0  |
| PTMnRPH       | —      | —      | —      | —      | —      | —      | PTnRP9 | PTnRP8  |

**10-bit Periodic TM Register List (n=0~1)**
**• PTMnCO Register**

| Bit  | 7      | 6      | 5      | 4      | 3     | 2 | 1 | 0 |
|------|--------|--------|--------|--------|-------|---|---|---|
| Name | PTnPAU | PTnCK2 | PTnCK1 | PTnCK0 | PTnON | — | — | — |
| R/W  | R/W    | R/W    | R/W    | R/W    | R/W   | — | — | — |
| POR  | 0      | 0      | 0      | 0      | 0     | — | — | — |

Bit 7 **PTnPAU**: PTMn 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 PTMn 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 **PTnCK2~PTnCK0**: PTMn counter clock selection

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

These three bits are used to select the clock source for the PTMn. 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 **PTnON**: PTMn counter on/off control

0: Off  
1: On

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

Bit 2~0 Unimplemented, read as “0”

• PTMnC1 Register

| Bit  | 7     | 6     | 5      | 4      | 3     | 2      | 1   | 0       |
|------|-------|-------|--------|--------|-------|--------|-----|---------|
| Name | PTnM1 | PTnM0 | PTnIO1 | PTnIO0 | PTnOC | PTnPOL | D1  | PTnCCLR |
| 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 **PTnM1~PTnM0**: PTMn operating mode selection  
 00: Compare Match Output Mode  
 01: Undefined  
 10: PWM Output Mode or Single Pulse Output Mode  
 11: Timer/Counter Mode

These bits setup the required operating mode for the PTMn. To ensure reliable operation the PTMn should be switched off before any changes are made to the PTnM1 and PTnM0 bits. In the Timer/Counter Mode, the PTMn output pin state is undefined.

Bit 5~4 **PTnIO1~PTnIO0**: PTMn external pin function selection

Compare Match Output Mode

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

PWM Output Mode/Single Pulse Output Mode

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

Timer/Counter Mode

Unused

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

In the Compare Match Output Mode, the PTnIO1 and PTnIO0 bits determine how the PTMn 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 PTMn output pin should be setup using the PTnOC bit in the PTMnC1 register. Note that the output level requested by the PTnIO1 and PTnIO0 bits must be different from the initial value setup using the PTnOC bit otherwise no change will occur on the PTMn output pin when a compare match occurs. After the PTMn output pin changes state, it can be reset to its initial level by changing the level of the PTnON bit from low to high.

In the PWM Output Mode, the PTnIO1 and PTnIO0 bits determine how the PTMn 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 change the values of the PTnIO1 and PTnIO0 bits only after the PTMn has been switched off. Unpredictable PWM outputs will occur if the PTnIO1 and PTnIO0 bits are changed when the PTMn is running.

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

Bit 2 **PTnPOL**: PTMn PTPn output polarity control

0: Non-invert

1: Invert

This bit controls the polarity of the PTPn output pin. When the bit is set high the PTMn 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 **D1**: Reserved, must be fixed at “0”

Bit 0 **PTnCCLR**: PTMn 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 PTnCCLR 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 PTnCCLR bit is not used in the PWM Output or Single Pulse Output Mode.

• **PTMnDL Register**

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

Bit 7~0 **D7~D0**: PTMn Counter Low Byte Register bit 7 ~ bit 0

PTMn 10-bit Counter bit 7 ~ bit 0

• **PTMnDH Register**

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

Bit 7~2 Unimplemented, read as “0”

Bit 1~0 **D9~D8**: PTMn Counter High Byte Register bit 1 ~ bit 0

PTMn 10-bit Counter bit 9 ~ bit 8

• **PTMnAL Register**

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

Bit 7~0 **D7~D0**: PTMn CCRA Low Byte Register bit 7 ~ bit 0

PTMn 10-bit CCRA bit 7 ~ bit 0

• **PTMnAH Register**

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

Bit 7~2 Unimplemented, read as “0”

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

• **PTMnRPL Register**

| Bit  | 7      | 6      | 5      | 4      | 3      | 2      | 1      | 0      |
|------|--------|--------|--------|--------|--------|--------|--------|--------|
| Name | PTnRP7 | PTnRP6 | PTnRP5 | PTnRP4 | PTnRP3 | PTnRP2 | PTnRP1 | PTnRP0 |
| 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 **PTnRP7~PTnRP0**: PTMn CCRP Low Byte Register bit 7 ~ bit 0  
PTMn 10-bit CCRP bit 7 ~ bit 0

• **PTMnRPH Register**

| Bit  | 7 | 6 | 5 | 4 | 3 | 2 | 1      | 0      |
|------|---|---|---|---|---|---|--------|--------|
| Name | — | — | — | — | — | — | PTnRP9 | PTnRP8 |
| R/W  | — | — | — | — | — | — | R/W    | R/W    |
| POR  | — | — | — | — | — | — | 0      | 0      |

Bit 7~2 Unimplemented, read as “0”

Bit 1~0 **PTnRP9~PTnRP8**: PTMn CCRP High Byte Register bit 1 ~ bit 0  
PTMn 10-bit CCRP bit 9 ~ bit 8

## Periodic Type TM Operation Modes

The Periodic Type TM can operate in one of four operating modes, Compare Match Output Mode, PWM Output Mode, Single Pulse Output Mode or Timer/Counter Mode. The operating mode is selected using the PTnM1 and PTnM0 bits in the PTMnC1 register.

### Compare Match Output Mode

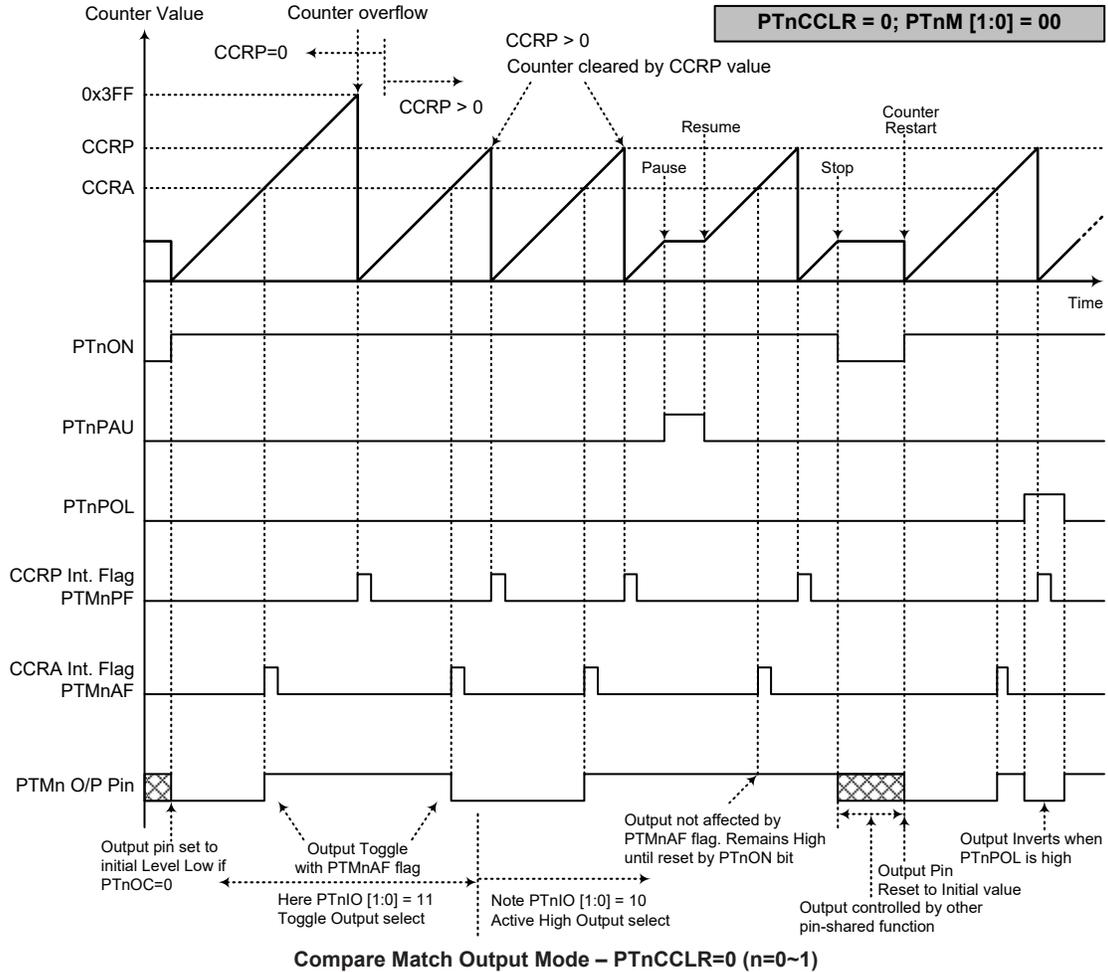
To select this mode, bits PTnM1 and PTnM0 in the PTMnC1 register, should be set to “00” respectively. In this mode once the counter is enabled and running it can be cleared by three methods. These are a counter overflow, a compare match from Comparator A and a compare match from Comparator P. When the PTnCCLR 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 PTMnAF and PTMnPF interrupt request flags for Comparator A and Comparator P respectively, will both be generated.

If the PTnCCLR bit in the PTMnC1 register is high then the counter will be cleared when a compare match occurs from Comparator A. However, here only the PTMnAF interrupt request flag will be generated even if the value of the CCRP bits is less than that of the CCRA registers. Therefore when PTnCCLR is high no PTMnPF interrupt request flag will be generated. In the Compare Match Output Mode, the CCRA can not be set to “0”.

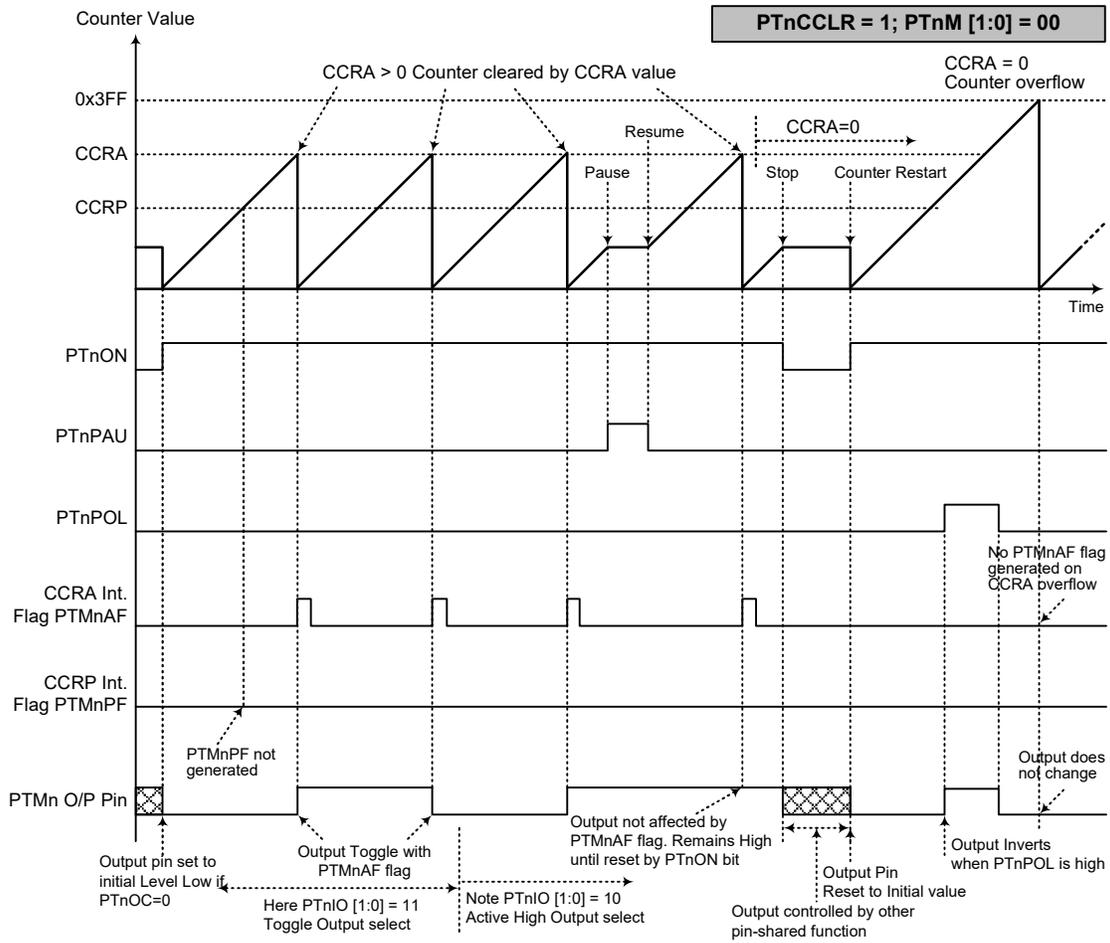
If the CCRA bits are all zero, the counter will overflow when its reaches its maximum 10-bit, 3FF Hex, value, however here the PTMnAF interrupt request flag will not be generated.

As the name of the mode suggests, after a comparison is made, the PTMn output pin will change

state. The PTMn output pin condition however only changes state when a PTMnAF interrupt request flag is generated after a compare match occurs from Comparator A. The PTMnPF interrupt request flag, generated from a compare match occurs from Comparator P, will have no effect on the PTMn output pin. The way in which the PTMn output pin changes state are determined by the condition of the PTnIO1 and PTnIO0 bits in the PTMnC1 register. The PTMn output pin can be selected using the PTnIO1 and PTnIO0 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 PTMn output pin, which is setup after the PTnON bit changes from low to high, is setup using the PTnOC bit. Note that if the PTnIO1 and PTnIO0 bits are zero then no pin change will take place.



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



**Compare Match Output Mode – PTnCCR=1 (n=0~1)**

- Note: 1. With PTnCCR=1, a Comparator A match will clear the counter  
 2. The PTMn output pin is controlled only by the PTMnAF flag  
 3. The output pin is reset to its initial state by a PTnON bit rising edge  
 4. A PTMnPF flag is not generated when PTnCCR=1

**Timer/Counter Mode**

To select this mode, bits PTnM1 and PTnM0 in the PTMnC1 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 PTMn 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 PTMn output pin is not used in this mode, the pin can be used as a normal I/O pin or other pin-shared functions.

**PWM Output Mode**

To select this mode, bits PTnM1 and PTnM0 in the PTMnC1 register should be set to “10” respectively and also the PTnIO1 and PTnIO0 bits should be set to “10” respectively. The PWM function within the PTMn 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 PTMn output pin, a square wave AC waveform can be generated with varying equivalent DC RMS values.

As both the period and duty cycle of the PWM waveform can be controlled, the choice of generated waveform is extremely flexible. In the PWM Output Mode, the PTnCCLR 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 PTnOC bit in the PTMnC1 register is used to select the required polarity of the PWM waveform while the two PTnIO1 and PTnIO0 bits are used to enable the PWM output or to force the PTMn output pin to a fixed high or low level. The PTnPOL bit is used to reverse the polarity of the PWM output waveform.

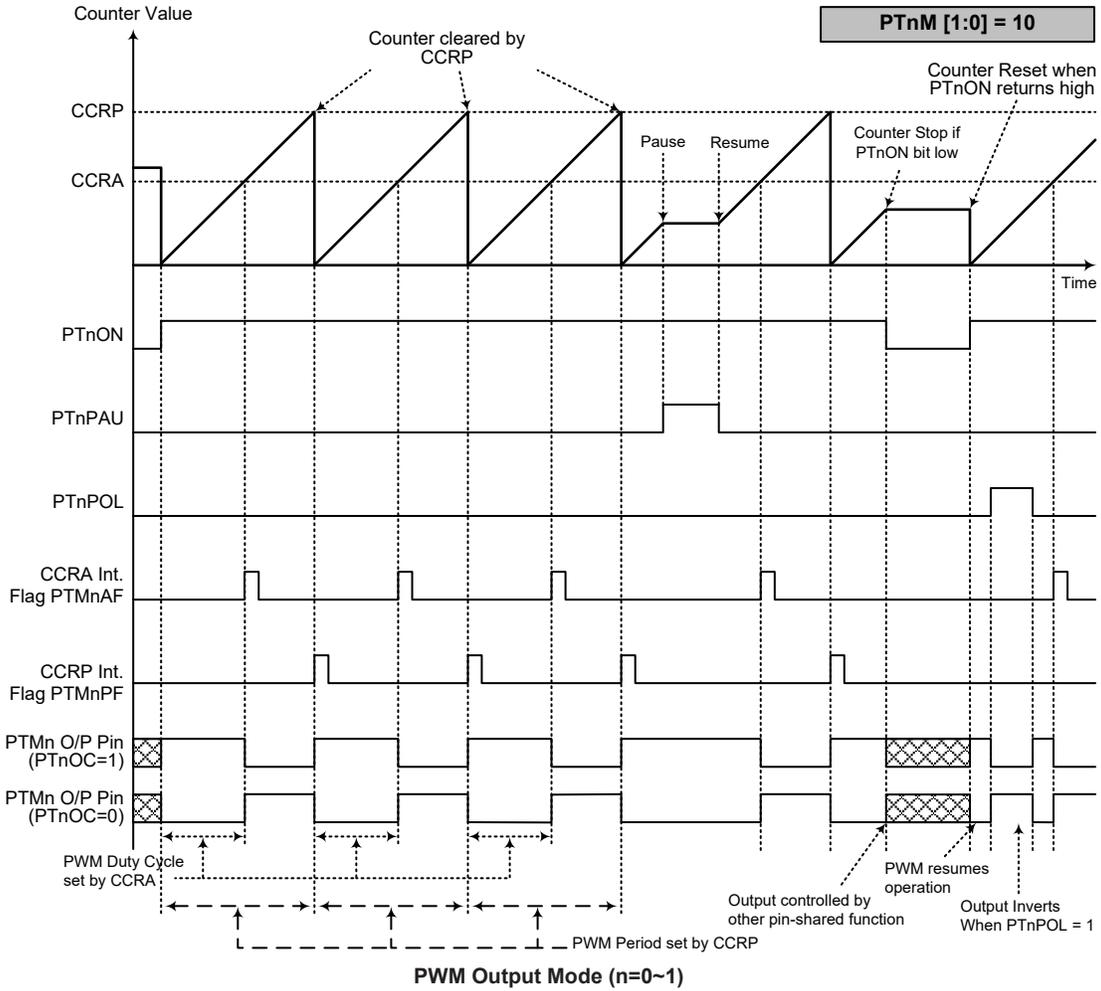
• **10-bit PWM Output Mode, Edge-aligned Mode**

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

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

The PTMn 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%.



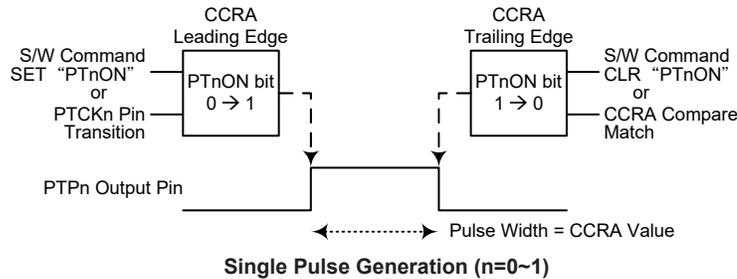
- Note: 1. The counter is cleared by CCRP  
 2. A counter clear sets the PWM Period  
 3. The internal PWM function continues running even when PTnIO[1:0]=00 or 01  
 4. The PTnCCLR bit has no influence on PWM operation

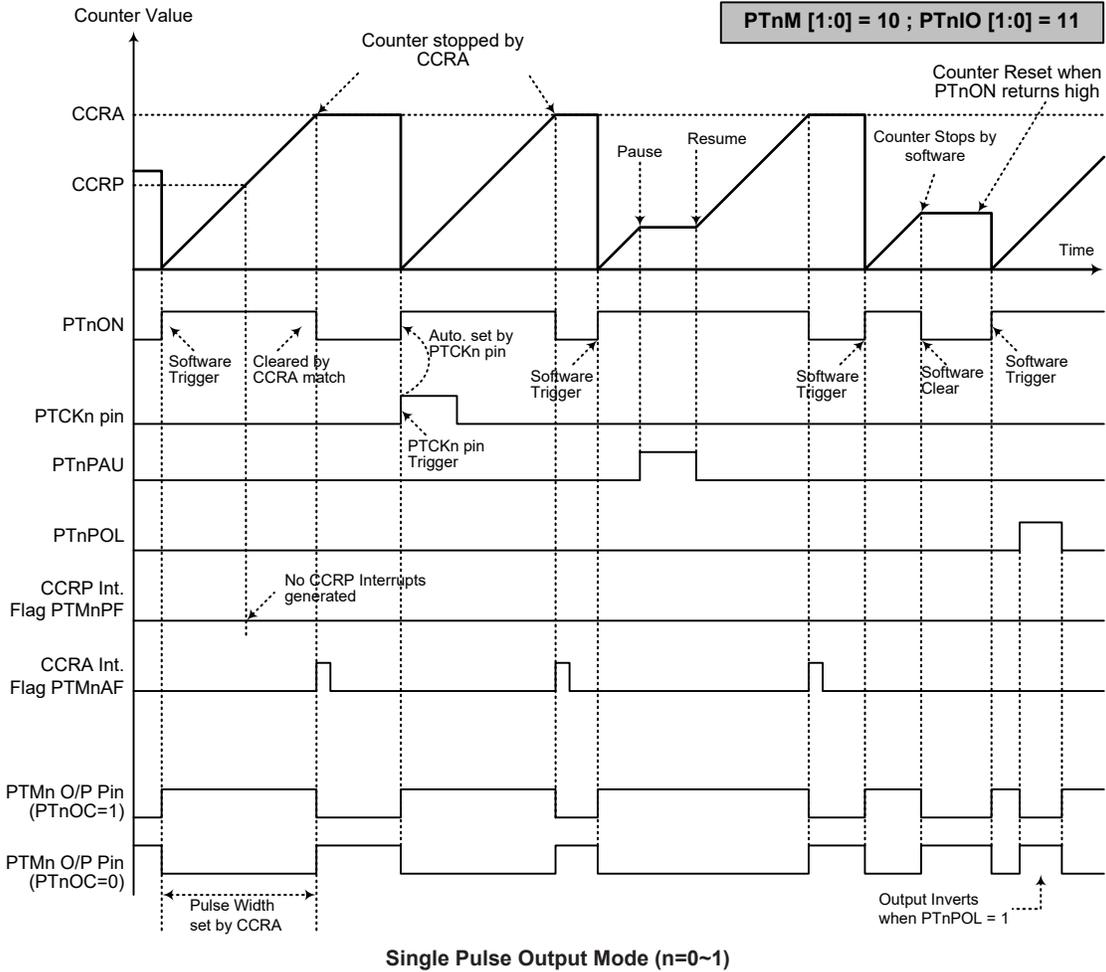
**Single Pulse Output Mode**

To select this mode, bits PTnM1 and PTnM0 in the PTMnC1 register should be set to “10” respectively and also the PTnIO1 and PTnIO0 bits should be set to “11” respectively. The Single Pulse Output Mode, as the name suggests, will generate a single shot pulse on the PTMn output pin.

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





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

## Analog to Digital Converter

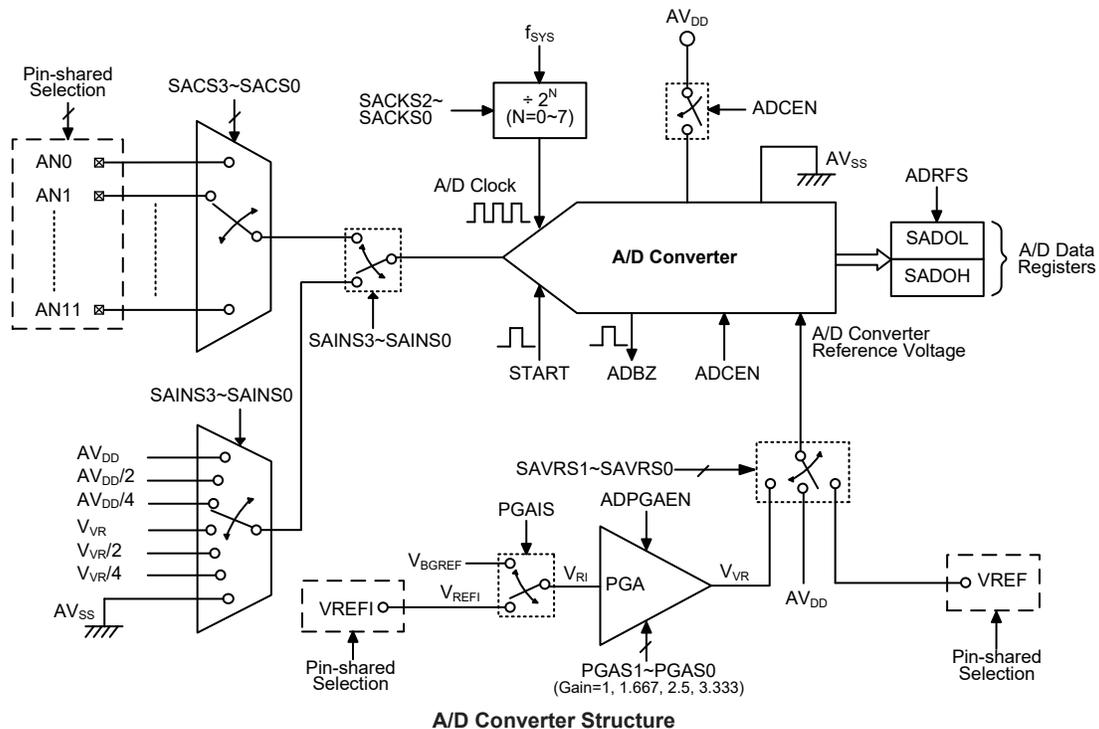
The need to interface to real world analog signals is a common requirement for many electronic systems. However, to properly process these signals by a microcontroller, they must first be converted into digital signals by A/D converters. By integrating the A/D conversion electronic circuitry into the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

### A/D Converter Overview

The device contains a multi-channel analog to digital converter which can directly interface to external analog signals, such as that from sensors or other control signals and convert these signals directly into a 12-bit digital value. It also can convert the internal signals, such as the internal A/D converter power supply and the internal PGA output voltage, and convert these signals directly into a 12-bit digital value. The external or internal analog signal to be converted is determined by the SAINS3~SAINS0 bits together with the SACS3~SACS0 bits. When the external analog signal is to be converted, the corresponding pin-shared control bits should first be properly configured and then desired external channel input should be selected using the SAINS3~SAINS0 and SACS3~SACS0 bits. Note that when the internal analog signal is to be converted using the SAINS3~SAINS0 bits, the external channel analog input will be automatically be switched off. More detailed information about the A/D converter input signal is described in the “A/D Converter Control Registers” and “A/D Converter Input Signals” sections respectively.

| External Input Channels | Internal Analog Signals   | A/D Signal Select          |
|-------------------------|---|----------------------------|
| AN0~AN11                | AV <sub>DD</sub> , AV <sub>DD</sub> /2, AV <sub>DD</sub> /4, V <sub>VR</sub> , V <sub>VR</sub> /2, V <sub>VR</sub> /4, AV <sub>SS</sub> | 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 Register Description

Overall operation of the A/D converter is controlled using a series of registers. A read only register pair exists to store the A/D converter data 12-bit value. Three registers, SADC0, SADC1 and SADC2, are the control registers which setup the operating conditions and control function of the A/D converter. The VBGRC register contains the VBGREN bit to control the bandgap reference voltage.

| 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 A/D converter also contains a programmable gain amplifier, PGA, to generate the A/D converter internal reference voltage. The overall operation of the PGA is controlled using the SADC2 register. 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  
 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 cleared to zero, 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  
           1100~1111: Non-existed channel, the input will be floating

• 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: External signal – External analog channel input , ANn  
 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  $AV_{DD}/4$   
 0100: External signal – External analog channel input , ANn  
 0101: Internal signal – Internal signal derived from PGA output  $V_{VR}$   
 0110: Internal signal – Internal signal derived from PGA output  $V_{VR}/2$   
 0111: Internal signal – Internal signal derived from PGA output  $V_{VR}/4$   
 10xx: Unused, connected to ground  
 1100~1111: External signal – External analog channel input , ANn  
 Care must be taken if the SAINS3~SAINS0 bits are set to “0001”~“0011”, “0101”~“0111” to select the internal analog signal to be converted. 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 bits value. It will prevent the external channel input from being connected together with the internal analog signal.

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:  $f_{SYS}/128$

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

• SADC2 Register

| Bit  | 7       | 6 | 5 | 4     | 3      | 2      | 1      | 0      |
|------|---------|---|---|-------|--------|--------|--------|--------|
| Name | ADPGAEN | — | — | 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 is used to control the A/D converter internal PGA function. When the PGA output voltage is selected as A/D input or A/D reference voltage, the PGA needs to be enabled by setting this bit high. Otherwise the PGA needs to be disabled by clearing the ADPGAEN bit to zero to conserve power.

Bit 6~5 Unimplemented, read as “0”

Bit 4 **PGAIS**: PGA input voltage ( $V_{RI}$ ) selection  
 0: From VREFI pin  
 1: From internal reference voltage  $V_{BREF}$

This bit is used to select the PGA input voltage source. When the internal reference voltage  $V_{BGREF}$  is selected as the PGA input voltage, the external reference voltage on the VREFI pin will be automatically switched off. When this bit is set high to select  $V_{BGREF}$  as PGA input, the internal bandgap reference  $V_{BGREF}$  should be enabled by setting the VBGREN bit in the VBGRC register to “1”.

- Bit 3~2     **SAVRS1~SAVRS0**: A/D converter reference voltage selection  
 00: Internal A/D converter power,  $AV_{DD}$   
 01: External VREF pin  
 1x: Internal PGA output voltage,  $V_{VR}$

These bits are used to select the A/D converter reference voltage source. When the internal A/D converter power supply or PGA output is set as the reference voltage, the reference voltage derived from the external VREF pin will be automatically switched off.

- Bit 1~0     **PGAGS1~PGAGS0**: PGA gain select  
 00: Gain=1  
 01: Gain=1.667 –  $V_{VR}=2V$  as  $V_{RI}=1.2V$   
 10: Gain=2.5 –  $V_{VR}=3V$  as  $V_{RI}=1.2V$   
 11: Gain=3.333 –  $V_{VR}=4V$  as  $V_{RI}=1.2V$

These bits are used to select the PGA gain. Note that here the gain is guaranteed only when the PGA input voltage is equal to 1.2V.

**Bandgap Referenc Voltage Control Register – VBGRC**

A high performance bandgap voltage reference is included in the device. It has an accurate voltage reference output,  $V_{BGREF}$ , when input supply voltage changes or temperature variates. The VBGRC register is used to control the bandgap reference voltage circuit enable or disable.

• **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 reference voltage control  
 0: Disable  
 1: Enable

This bit is used to enable the internal Bandgap reference circuit. The internal Bandgap reference circuit should first be enabled before the  $V_{BGREF}$  voltage is selected to be used. A specific start-up time is necessary for the Bandgap circuit to become stable and accurate. When this bit is cleared to 0, the Bandgap voltage output  $V_{BGREF}$  is in a low state.

**A/D Converter Reference Voltage**

The actual reference voltage supply to the A/D Converter can be supplied from the internal A/D converter power,  $AV_{DD}$ , an external reference source supplied on pin VREF or an internal reference voltage  $V_{VR}$  determined by the SAVRS1~SAVRS0 bits in the SADC2 register. The internal reference voltage  $V_{VR}$  is derived from a programmable gain amplifier, PGA, which is controlled by the ADPGAEN bit in the SADC2 register. The PGA gain can be equal to 1, 1.667, 2.5 or 3.333 and selected using the PGAGS1~PGAGS0 bits in the SADC2 register. The PGA input can come from the external reference input pin, VREFI, or an internal Bandgap reference voltage,  $V_{BGREF}$ , selected by the PGAIS bit in the SADC2 register. Note that the internal Bandgap reference circuit should first be enabled before the  $V_{BGREF}$  is selected to be used.

As the VREFI and VREF pin both are pin-shared with other functions, when the VREFI or VREF pin is selected as the reference voltage pin, the VREFI or VREF pin-shared function selection bits should first be properly configured to disable other pin-shared functions. However, if the internal reference signal is selected as the reference source, the external reference voltage input from the VREF or VREFI pin will automatically be switched off by hardware.

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

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

**A/D Converter Reference Voltage Selection**

### A/D Converter Input Signals

All of the external A/D 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 PxS1 and PxS0 registers, determine whether the external input pins are set 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 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 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” or “1100~1111”, the external analog channel input is selected to be converted and the SACS3~SACS0 bits can determine which actual external channel is selected to be converted. If the SAINS3~SAINS0 bits are set to other values, the internal analog signal will be selected. If the internal analog signal is selected to be converted, the external input channel will automatically be switched off regardless of the SACS3~SACS0 bits value. It will prevent the external channel input from being connected together with the internal analog signal.

| SAINS[3:0]               | SACS[3:0] | Input Signals       | Description   |
|--------------------------|-----------|---------------------|---|
| 0000, 0100,<br>1100~1111 | 0000~1011 | AN0~AN11            | External channel analog input ANn                               |
|                          | 1100~1111 | —                   | Floating, no external channel is selected                       |
| 0001                     | xxxx      | AV <sub>DD</sub>    | Internal A/D converter power supply voltage AV <sub>DD</sub>    |
| 0010                     | xxxx      | AV <sub>DD</sub> /2 | Internal A/D converter power supply voltage AV <sub>DD</sub> /2 |
| 0011                     | xxxx      | AV <sub>DD</sub> /4 | Internal A/D converter power supply voltage AV <sub>DD</sub> /4 |
| 0101                     | xxxx      | V <sub>VR</sub>     | Internal A/D converter PGA output V <sub>VR</sub>               |
| 0110                     | xxxx      | V <sub>VR</sub> /2  | Internal A/D converter PGA output V <sub>VR</sub> /2            |
| 0111                     | xxxx      | V <sub>VR</sub> /4  | Internal A/D converter PGA output V <sub>VR</sub> /4            |
| 10xx                     | xxxx      | AV <sub>SS</sub>    | Connected to the ground   |

“x”: Don't care

**A/D Converter Input Signal Selection**

## A/D Converter Operation

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

The ADBZ bit in the SADC0 register is used to indicate whether the analog to digital conversion process is in process or not. This bit will be automatically set to “1” by the microcontroller after an A/D conversion is successfully initiated. When the A/D conversion is complete, the ADBZ will be cleared to “0”. In addition, the corresponding A/D 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 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 8MHz, the SACKS2~SACKS0 bits should not be set to “000”, “001” 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 \* special care must be taken, as the values may be less or larger than the specified A/D clock period range.

| $f_{SYS}$ | A/D Conversion Clock Period ( $t_{ADCK}$ ) |                                       |                                       |                                       |  |  |  |   |
|-----------|--|---------------------------------------|---------------------------------------|---------------------------------------|--|--|--|---|
|           | SACKS[2:0]<br>=000<br>( $f_{SYS}$ )        | SACKS[2:0]<br>=001<br>( $f_{SYS}/2$ ) | SACKS[2:0]<br>=010<br>( $f_{SYS}/4$ ) | SACKS[2:0]<br>=011<br>( $f_{SYS}/8$ ) | SACKS[2:0]<br>=100<br>( $f_{SYS}/16$ ) | SACKS[2:0]<br>=101<br>( $f_{SYS}/32$ ) | SACKS[2:0]<br>=110<br>( $f_{SYS}/64$ ) | SACKS[2:0]<br>=111<br>( $f_{SYS}/128$ ) |
| 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**

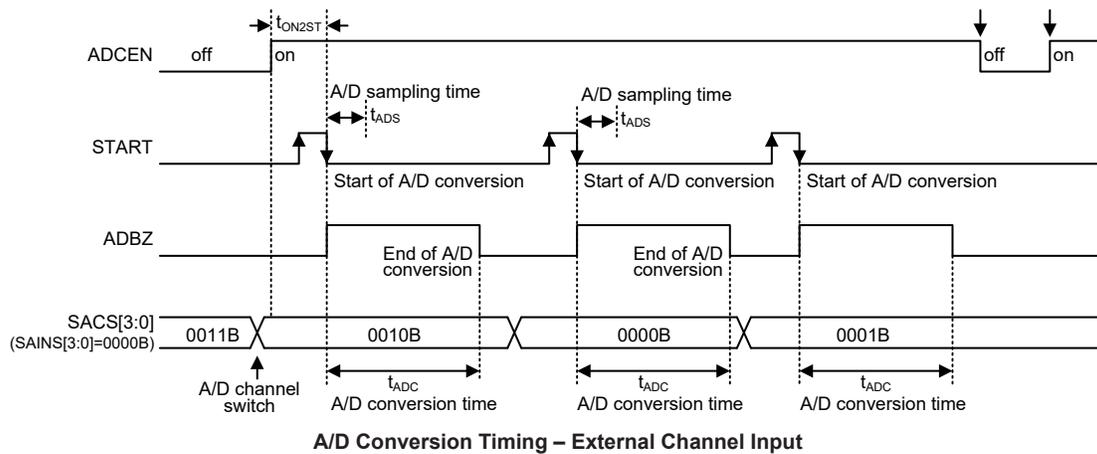
Controlling the power on/off function of the A/D conversion circuitry is implemented using the ADCEN bit in the SADC0 register. This bit must be set high to power on the A/D converter. When the ADCEN bit is set high to power on the A/D conversion internal circuitry, a certain delay as indicated in the 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 Conversion Rate and Timing Diagram

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

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

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



### 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 input signal comes from the external channel input selected by configuring the SAINS3~SAINS0, the corresponding pin should be configured as an A/D input function by configuring the relevant pin-shared function control bits. The desired external channel then should be selected by configuring the SACS3~SACS0. After this step, go to Step 6.

- Step 5  
If the A/D input signal is selected to come from the internal analog signal by configuring the SAINS3~SAINS0 and the external channel analog signal input will be automatically switched off regardless of the SACS3~SACS0 bits value. After this step, go to Step 6.
- Step 6  
Select the reference voltage source by configuring the SAVRS1~SAVRS0 bits in the SADC2 register. Select the PGA input signal and the desired PGA gain if the PGA output voltage,  $V_{VR}$ , is selected as the A/D converter reference voltage.
- Step 7  
Select A/D converter output data format by configuring the ADRFS bit in the SADC0 register.
- 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 bit, 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.

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 setting the ADCEN bit low 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 Conversion Function

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

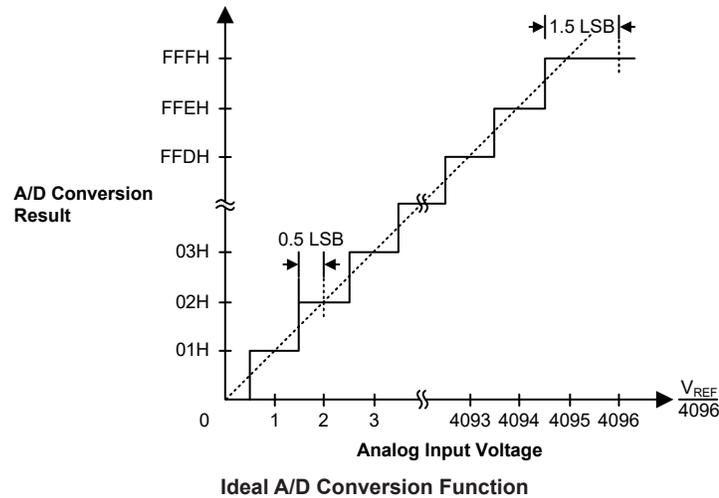
$$1 \text{ LSB} = V_{REF} \div 4096$$

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

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

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

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



### 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 input signal from external channel input, fsys/8 as A/D
                  ; clock

mov a,00H
mov SADC2,a      ; select reference voltage from AVDD
mov a,02h        ; setup PBS0 to configure pin AN0
mov PBS0,a
mov a,20h
mov SADC0,a      ; enable A/D converter and connect AN0 channel to A/D converter
:
start_conversion:
clr START        ; high pulse on start bit to initiate conversion
set START        ; reset A/D 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

```

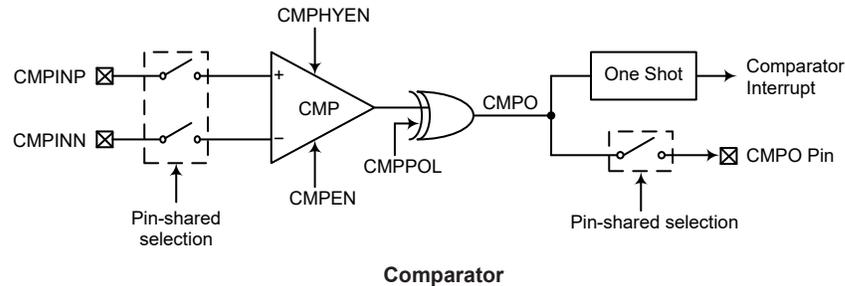
**Example: using the interrupt method to detect the end of conversion**

```
clr ADE          ; disable ADC interrupt
mov a,03H
mov SADC1,a      ; select input signal from external channel input, fsys/8 as A/D
                  ; clock

mov a,00H
mov SADC2,a      ; select reference voltage from AVDD
mov a,02h        ; setup PBS0 to configure pin AN0
mov PBS0,a
mov a,20h
mov SADC0,a      ; enable A/D converter and connect AN0 channel to A/D converter
Start_conversion:
clr START        ; high pulse on START bit to initiate conversion
set START        ; reset A/D 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 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 which will apply a small amount of positive feedback to the comparator. 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 the control register, CMPC. 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.

#### • CMPC Register

| Bit  | 7 | 6     | 5      | 4    | 3 | 2 | 1 | 0       |
|------|---|-------|--------|------|---|---|---|---------|
| Name | — | CMPEN | CMPPOL | CMPO | — | — | — | CMPHYEN |
| R/W  | — | R/W   | R/W    | R    | — | — | — | R/W     |
| POR  | — | 0     | 0      | 0    | — | — | — | 1       |

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 device enters the SLEEP or IDLE mode. Note that the comparator output will be set low when this bit is cleared to zero.

|         |   |
|---------|---|
| Bit 5   | <b>CMPPOL:</b> Comparator output polarity Control<br>0: Output is not inverted<br>1: Output is inverted<br><br>This is the Comparator polarity control bit. If the bit is zero then the comparator output bit, CMPO, will reflect the non-inverted output condition of the comparator. If the bit is high the comparator output bit will be inverted. |
| Bit 4   | <b>CMPO:</b> Comparator output bit<br>If CMPPOL=0,<br>0: CMPINP < CMPINN<br>1: CMPINP > CMPINN<br>If CMPPOL=1,<br>0: CMPINP > CMPINN<br>1: CMPINP < CMPINN<br><br>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 CMPPOL bit.                   |
| Bit 3~1 | Unimplemented, read as “0”  |
| Bit 0   | <b>CMPHYEN:</b> Comparator hysteresis voltage control<br>0: Comparator hysteresis disable<br>1: Comparator hysteresis enable<br><br>Refer to Comparator Characteristics section.  |

### Comparator Interrupt

The comparator has its own interrupt function. When the comparator output bit changes state, its relevant interrupt flag will be set, and if the corresponding interrupt enable bit is set, then a jump to its relevant interrupt vector will be executed. If the microcontroller is in the SLEEP or IDLE Mode and the comparator is enabled, then if the external input lines cause the comparator output bit to change state, the resulting generated interrupt flag will also generate a wake-up. If it is required to disable a wake-up from occurring, then the interrupt flag should be first set high before entering the SLEEP or IDLE Mode.

### Programming Considerations

If the comparator is enabled, it will remain active when the microcontroller enters the SLEEP or IDLE Mode, however as it will consume a certain amount of power, the user may wish to consider disabling it before the SLEEP or IDLE Mode is entered.

As comparator pins are shared with normal I/O pins the I/O registers for these pins will be read as zero (port control register is “1”) or read as port data register value (port control register is “0”) if the comparator function is enabled.

## Serial Interface Module – SIM

The device contains a Serial Interface Module, which includes 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.

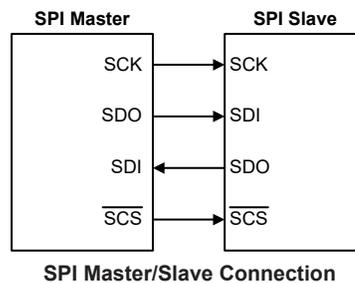
### SPI Interface

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 device can be either master or slave. Although the SPI interface specification can control multiple slave devices from a single master, the device provides only one  $\overline{SCS}$  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  $\overline{SCS}$ . Pins SDI and SDO are the Serial Data Input and Serial Data Output lines, the SCK pin is the Serial Clock line and  $\overline{SCS}$  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 setting 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  $\overline{SCS}$  pin only one slave device can be utilized. The  $\overline{SCS}$  pin is controlled by software, set CSEN bit to 1 to enable  $\overline{SCS}$  pin function, set CSEN bit to 0 the  $\overline{SCS}$  pin will be floating state.

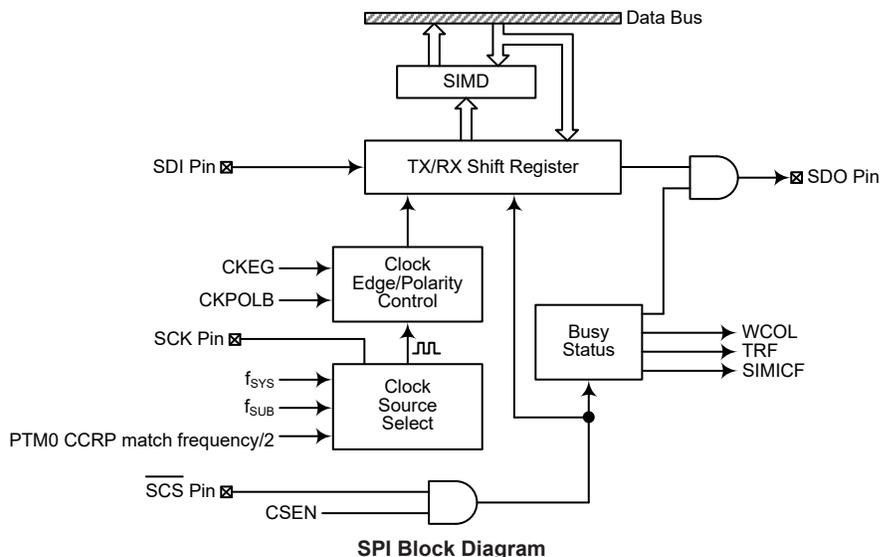


The SPI function in this device offers the following features:

- 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 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. 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 Register List

### SPI Data 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.

#### • 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

Bit 7~0      **D7~D0**: SIM data register bit 7 ~ bit 0

### SPI Control Registers

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. The SIMC0 register is used to control the enable/disable function and to set the data transmission clock frequency. The SIMC2 register 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 PTM0 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 PTM0 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

These bits are only used in the I<sup>2</sup>C mode and the detailed definition is described in the I<sup>2</sup>C function.

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 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. 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 is not occurred
- 1: SIM SPI slave mode incomplete condition is 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 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 set high. 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**

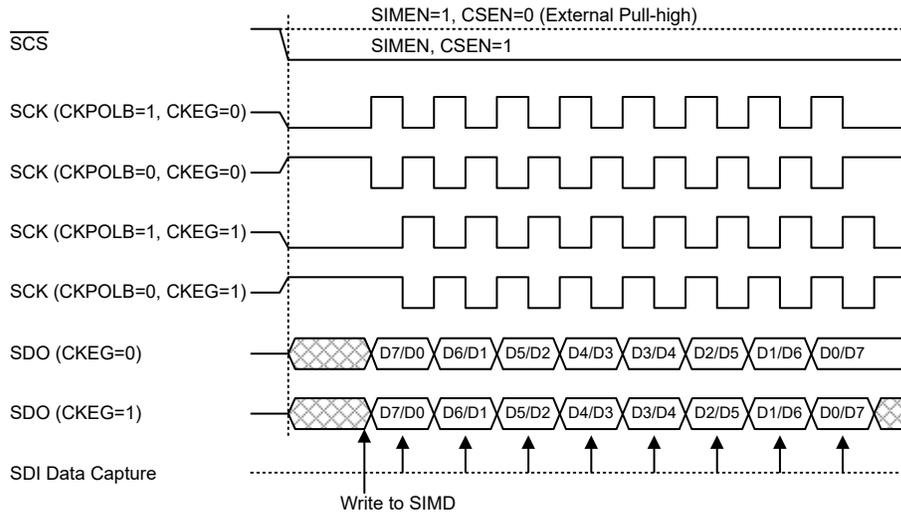
| Bit  | 7   | 6   | 5      | 4    | 3   | 2    | 1    | 0   |
|------|-----|-----|--------|------|-----|------|------|-----|
| Name | D7  | D6  | CKPOLB | CKEG | MLS | CSEN | WCOL | TRF |
| R/W  | R/W | R/W | R/W    | R/W  | R/W | R/W  | R/W  | R/W |
| POR  | 0   | 0   | 0      | 0    | 0   | 0    |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

- Bit 7~6     **D7~D6:** Undefined bits  
These bits can be read or written by 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 when the clock is inactive and data capture at SCK rising edge  
1: SCK is high base level when the clock is inactive and data capture at SCK falling edge  
CKPOLB=1  
0: SCK is low base level when the clock is inactive and data capture at SCK falling edge  
1: SCK is low base level when the clock is inactive 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  $\overline{SCS}$  pin control  
0: Disable  
1: Enable  
The CSEN bit is used as an enable/disable for the  $\overline{SCS}$  pin. If this bit is low, then the  $\overline{SCS}$  pin will be disabled and placed into a floating condition. If the bit is high the  $\overline{SCS}$  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 if a data collision has occurred. 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. The 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 transmission is completed  
The TRF bit is the Transmit/Receive Complete flag and is set “1” automatically when an SPI data transmission is completed, but must set to “0” by the application program. It can be used to generate an interrupt.

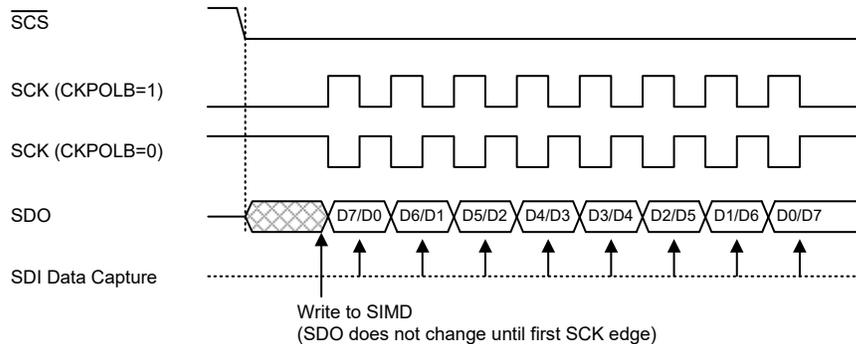
**SPI Communication**

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 an  $\overline{SCS}$  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 SCK signal depending upon the configurations of the CKPOLB bit and CKEG bit. The accompanying timing diagrams show the relationship between the slave data and SCK signal for various configurations of the CKPOLB and CKEG bits.

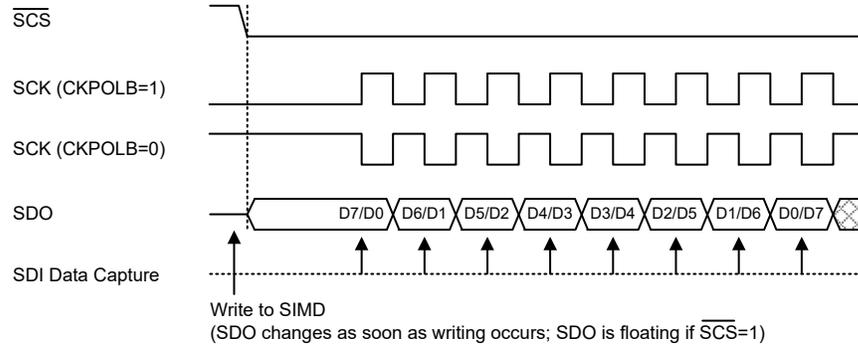
The SPI Master mode will continue to function if the SPI clock is running.



**SPI Master Mode Timing**

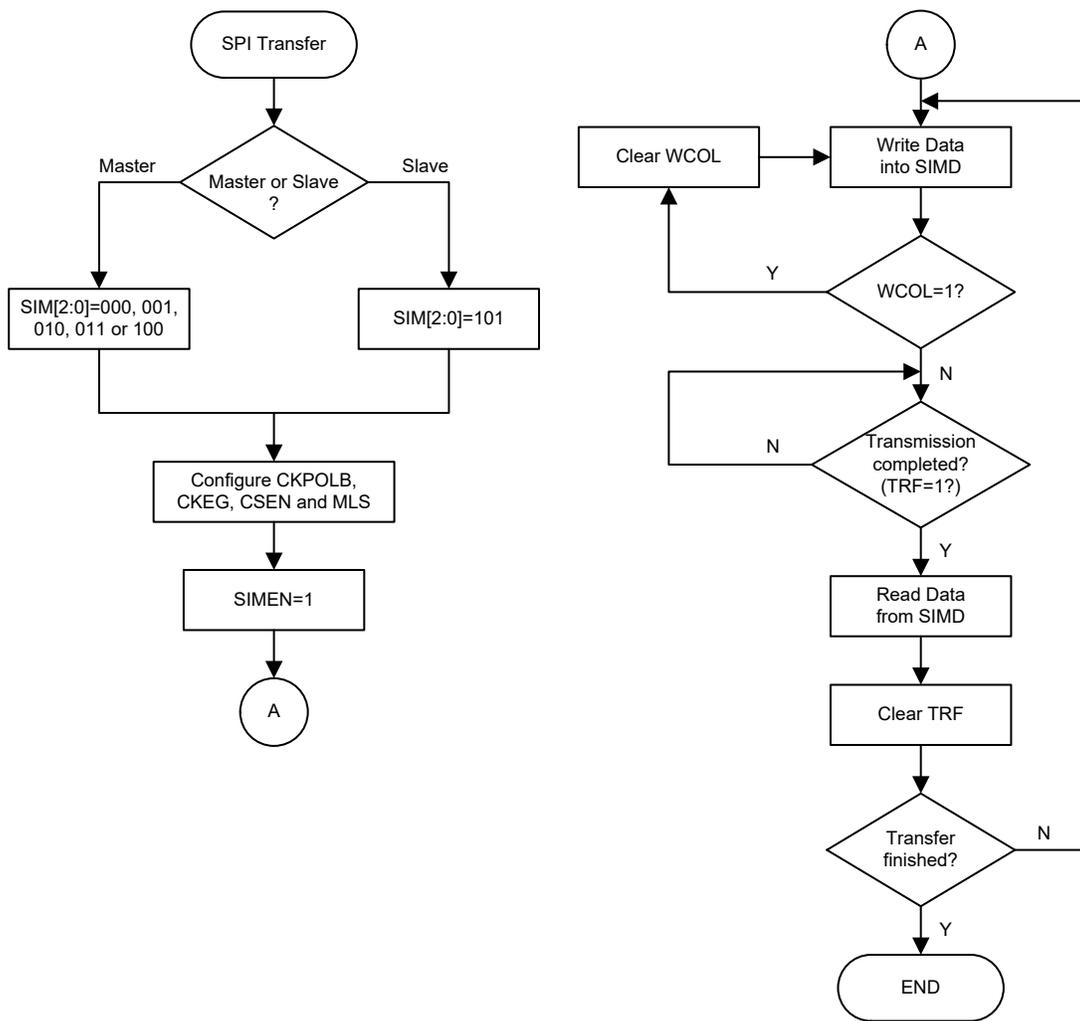


**SPI Slave Mode Timing – CKEG=0**



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

**SPI Slave Mode Timing – CKEG=1**



**SPI Transfer Control Flow Chart**

### **SPI Bus Enable/Disable**

To enable the SPI bus, set CSEN=1 and  $\overline{SCS}$ =0, then wait for data to be written into the SIMD (TXRX buffer) register. For the Master Mode, after data has been written to the SIMD (TXRX buffer) register, then transmission or reception will start automatically. When all the data has been transferred, the TRF bit should be set. For the Slave Mode, when clock pulses are received on SCK, data in the TXRX buffer will be shifted out or data on SDI will be shifted in.

When the SPI bus is disabled, SCK, SDI, SDO and  $\overline{SCS}$  can become I/O pins or other pin-shared functions using the corresponding control bits.

### **SPI Operation Steps**

All communication is carried out using the 4-line interface for either Master or Slave Mode.

The CSEN bit in the SIMC2 register controls the overall function of the SPI interface. Setting this bit high will enable the SPI interface by allowing the  $\overline{SCS}$  line to be active, which can then be used to control the SPI interface. If the CSEN bit is low, the SPI interface will be disabled and the  $\overline{SCS}$  line will be in a floating condition and can therefore not be used for control of the SPI interface. If the CSEN bit and the SIMEN bit in the SIMC0 are set high, this will place the SDI line in a floating condition and the SDO line high. If in Master Mode the SCK line will be either high or low depending upon the clock polarity selection bit CKPOLB in the SIMC2 register. If in Slave Mode the SCK line will be in a floating condition. If the SIMEN bit is low, then the bus will be disabled and  $\overline{SCS}$ , SDI, SDO and SCK will all become I/O pins or the other functions using the corresponding pin-shared control bits. In the Master Mode the Master will always generate the clock signal. The clock and data transmission will be initiated after data has been written into the SIMD register. In the Slave Mode, the clock signal will be received from an external master device for both data transmission and reception. The following sequences show the order to be followed for data transfer in both Master and Slave Mode.

#### **Master Mode**

- Step 1  
Select the SPI Master mode and clock source using the SIM2~SIM0 bits in the SIMC0 control register.
- Step 2  
Setup the CSEN bit and setup the MLS bit to choose if the data is MSB or LSB first, this setting must be the same with the Slave devices.
- Step 3  
Setup the SIMEN bit in the SIMC0 control register to enable the SPI interface.
- Step 4  
For write operations: write the data to the SIMD register, which will actually place the data into the TXRX buffer. Then use the SCK and SDO lines to output the data. After this, go to step5.  
For read operations: the data transferred in on the SDI line will be stored in the TXRX buffer until all the data has been received at which point it will be latched into the SIMD register.
- Step 5  
Check the WCOL bit if set high then a collision error has occurred so return to step 4. If equal to zero then go to the following step.
- Step 6  
Check the TRF bit or wait for an SIM SPI serial bus interrupt.
- Step 7  
Read data from the SIMD register.

- Step 8  
Clear TRF.
- Step 9  
Go to step 4.

**Slave Mode**

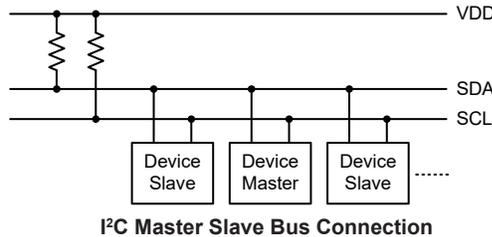
- Step 1  
Select the SPI Slave mode using the SIM2~SIM0 bits in the SIMC0 control register
- Step 2  
Setup the CSEN bit and setup the MLS bit to choose if the data is MSB or LSB first, this setting must be the same with the Master devices.
- Step 3  
Setup the SIMEN bit in the SIMC0 control register to enable the SPI interface.
- Step 4  
For write operations: write the data to the SIMD register, which will actually place the data into the TXRX buffer. Then wait for the master clock SCK and  $\overline{\text{SCS}}$  signal. After this, go to step5.  
For read operations: the data transferred in on the SDI line will be stored in the TXRX buffer until all the data has been received at which point it will be latched into the SIMD register.
- Step 5  
Check the WCOL bit if set high then a collision error has occurred so return to step 4. If equal to zero then go to the following step.
- Step 6  
Check the TRF bit or wait for an SIM SPI serial bus interrupt.
- Step 7  
Read data from the SIMD register.
- Step 8  
Clear TRF.
- Step 9  
Go to step 4.

**Error Detection**

The WCOL bit in the SIMC2 register is provided to indicate errors during data transfer. The bit is set by the SPI serial Interface but must be cleared by the application program. This bit indicates a data collision has occurred which happens if a write to the SIMD register takes place during a data transfer operation and will prevent the write operation from continuing.

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

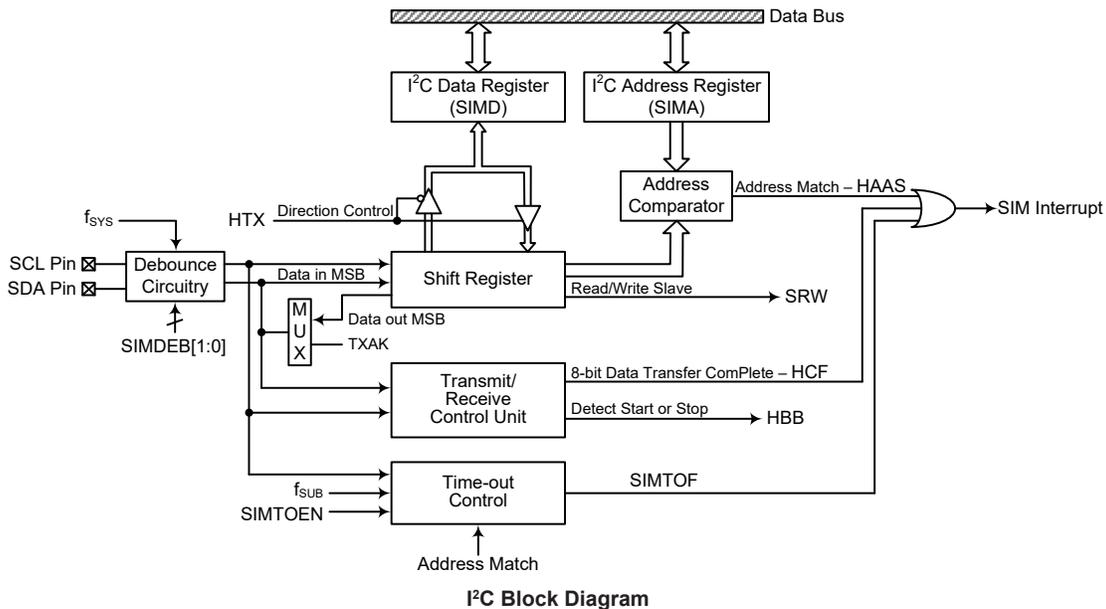
The I<sup>2</sup>C interface is used to communicate with external peripheral devices such as sensors, EEPROM memory etc. Originally developed by Philips, it is a two-line low speed serial interface for synchronous serial data transfer. The advantage of only two lines for communication, relatively simple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

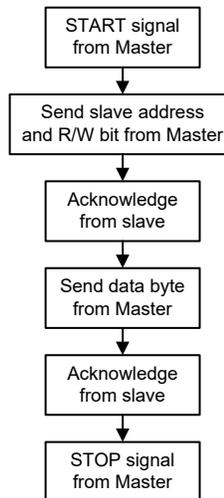


### I<sup>2</sup>C interface Operation

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 operates 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 pull-high control function pin-shared with SCL/SDA pin is still applicable even if I<sup>2</sup>C device is activated and the related internal pull-high function could be controlled by its corresponding pull-high control register.





**I<sup>2</sup>C Interface Operation**

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                              | $f_{SYS} > 2\text{MHz}$                 | $f_{SYS} > 4\text{MHz}$             |
| 2 system clock debounce                  | $f_{SYS} > 4\text{MHz}$                 | $f_{SYS} > 8\text{MHz}$             |
| 4 system clock debounce                  | $f_{SYS} > 4\text{MHz}$                 | $f_{SYS} > 8\text{MHz}$             |

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

**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 slave address register, SIMA, and one data register, SIMD.

| 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    |
| SIMD          | D7      | D6     | D5      | D4      | D3      | D2      | D1      | D0      |
| SIMA          | SIMA6   | SIMA5  | SIMA4   | SIMA3   | SIMA2   | SIMA1   | SIMA0   | D0      |
| SIMTOC        | SIMTOEN | SIMTOF | SIMTOS5 | SIMTOS4 | SIMTOS3 | SIMTOS2 | SIMTOS1 | SIMTOS0 |

**I<sup>2</sup>C Register List**

**I<sup>2</sup>C Data 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.

• **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

Bit 7~0 **D7~D0**: SIM data register bit 7 ~ bit 0

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

The SIMA register is also used by the SPI interface but has the name of SIMC2. The SIMA register is the location where the 7-bit slave address of the slave device is stored. Bit 7~bit 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 locates at the same register address as SIMC2 which is used by the SPI interface.

• **SIMA Register**

| Bit  | 7     | 6     | 5     | 4     | 3     | 2     | 1     | 0   |
|------|-------|-------|-------|-------|-------|-------|-------|-----|
| Name | SIMA6 | SIMA5 | SIMA4 | SIMA3 | SIMA2 | SIMA1 | SIMA0 | 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 **SIMA6~SIMA0**: I<sup>2</sup>C slave address  
 SIMA6~SIMA0 is the I<sup>2</sup>C slave address bit 6 ~ bit 0.

Bit 0 **D0**: Reserved bit, can be read or written

**I<sup>2</sup>C Control Registers**

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 select the I<sup>2</sup>C slave mode and debounce time. The SIMC1 register contains the relevant flags which are used to indicate the I<sup>2</sup>C communication status. The SIMTOC register is used to control the I<sup>2</sup>C bus time-out function which is 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 PTM0 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 PTM0 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  
 These bits are used to select the I<sup>2</sup>C debounce time when the SIM is configured as the I<sup>2</sup>C interface function by setting the SIM2~SIM0 bits to “110”.
- 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 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 incomplete flag  
 This bit is only available when the SIM is configured to operate in an SPI slave mode. Refer to the SPI register section.

• **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/W | 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 address match 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

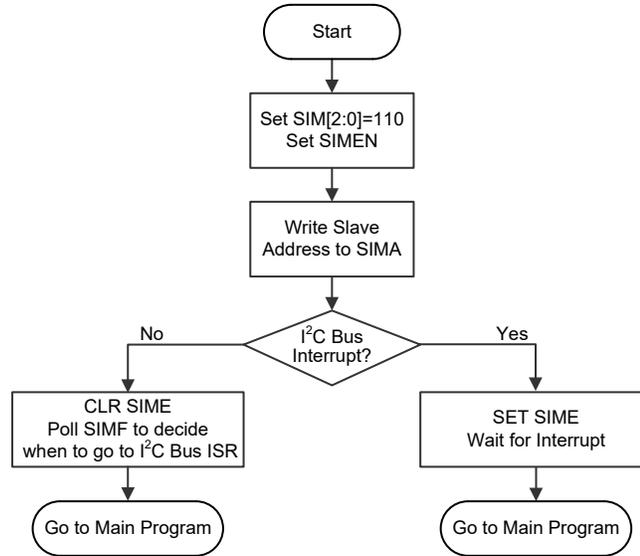
- Bit 3      **TXAK:** I<sup>2</sup>C bus transmit acknowledge flag  
            0: Slave send acknowledge flag  
            1: Slave does not send acknowledge flag  
The TXAK bit is the transmit acknowledge flag. After the slave device receipt of 8 bits of data, this bit will be transmitted to the bus on the 9th clock from the slave device. The slave device must always clear the 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 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  
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 to zero 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 9th 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 SIM 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 or from the completion of an 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 8th 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 and SIMEN bits in the SIMC0 register to “110” and “1” respectively 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 SIM I<sup>2</sup>C bus interrupt signal will be generated. The next bit following the address, which is the 8th 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 9th bit. The slave device will also set the status flag HAAS when the addresses match.

As an SIM I<sup>2</sup>C bus interrupt signal 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 from 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 master 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.

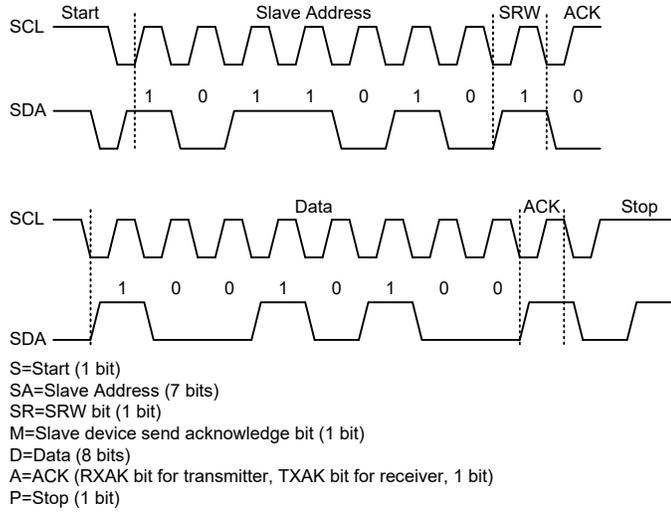
### **I<sup>2</sup>C Bus Slave Address Acknowledge Signal**

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**

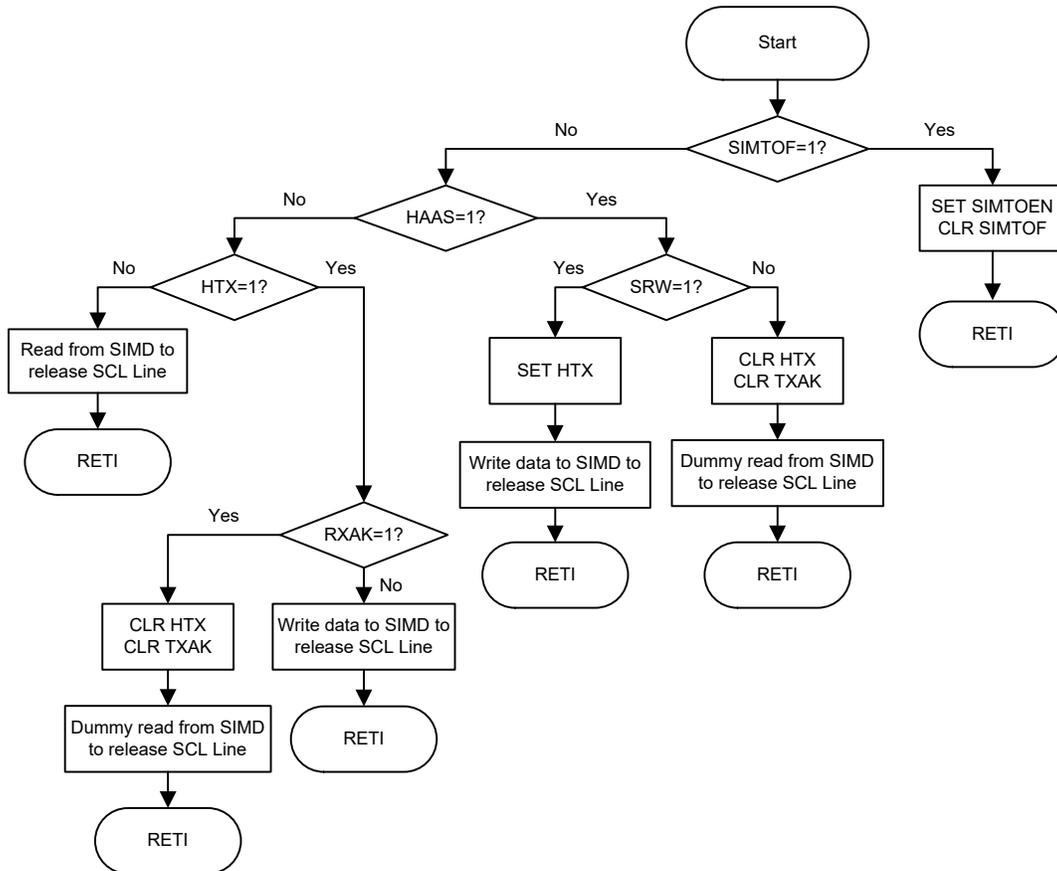
The transmitted data is 8-bits wide and is transmitted after the slave device has acknowledged receipt of its slave address. The order of serial bit transmission is the MSB first and the LSB last. After receipt of 8-bits of data, the receiver must transmit an acknowledge signal, level “0”, before it 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 9th 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.



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

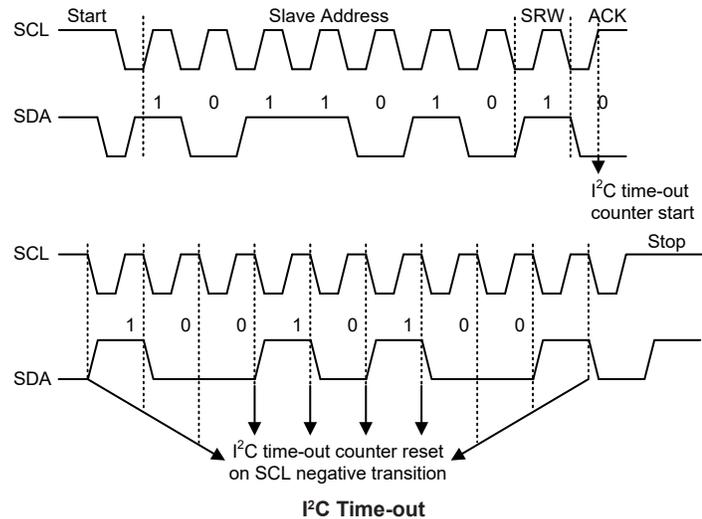
Note: When a slave address is matched, the device must be placed in either the transmit mode and then write data to the SIMD register, or in the receive mode where it must implement a dummy read from the SIMD register to release the SCL line.



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

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

In order to reduce the problem of I<sup>2</sup>C lockup due to reception of erroneous clock sources, a time-out function is provided. If the clock source 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 counting 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 setup 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 SIM 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:

| Registers         | 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 bit field in the SIMTOC register. The time-out duration is calculated by the formula:  $((1\sim64)\times32)/f_{SUB}$ . 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 function 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

This bit is set high when time-out occurs and can only be cleared to zero by application program.

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  $f_{SUB}/32$ .

I<sup>2</sup>C time-out time is equal to  $(SIMTOS[5:0]+1) \times (32/f_{SUB})$ .

## UART Interfaces

The device contains two integrated full-duplex or half-duplex asynchronous serial communications UART interfaces that enable communication with external devices that contain a serial interface. Each 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. Each UART function possesses its own internal interrupt which can be used to indicate when a reception occurs or when a transmission terminates.

The integrated UARTn function contains the following features:

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



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

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

### UART Data Transfer Scheme

The UARTn Data Transfer Block Diagram shows the overall data transfer structure arrangement for the UARTn interface. The actual data to be transmitted from the MCU is first transferred to the TXR\_RXRn 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 TXn pin at a rate controlled by the Baud Rate Generator. Only the TXR\_RXRn 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 UARTn is accepted on the external RXn/TXn 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\_RXRn register, where it is buffered and can be manipulated by the application program. Only the TXR\_RXRn register is mapped onto the MCU Data Memory, the Receiver Shift Register is not mapped and is therefore inaccessible to the application program.

It should be noted that the actual register for data transmission and reception only exists as a single shared register in the Data Memory. This shared register known as the TXR\_RXRn register is used for both data transmission and data reception.

### UART Status and Control Registers

There are nine control registers associated with the UARTn function. The SWMn bit in the UnCR3 register is used to enable/disable the UARTn Single Wire Mode. The UnSR, UnCR1, UnCR2, UFCRn and RxCNTn registers control the overall function of the UARTn, while the BRDHn and BRDLn registers control the Baud rate. The actual data to be transmitted and received on the serial interface is managed through the TXR\_RXRn data register.

| Register Name | Bit     |        |        |        |        |        |         |         |
|---------------|---------|--------|--------|--------|--------|--------|---------|---------|
|               | 7       | 6      | 5      | 4      | 3      | 2      | 1       | 0       |
| UnSR          | PERRn   | NFn    | FERRn  | OERRn  | RIDLEn | RXIFn  | TIDLEn  | TXIFn   |
| UnCR1         | UARTENn | BNO n  | PRENn  | PRTn1  | PRTn0  | TXBRKn | RX8n    | TX8n    |
| UnCR2         | TXENn   | RXENn  | STOPSn | ADDENn | WAKEn  | RIEn   | TIIE n  | TEIE n  |
| UnCR3         | —       | —      | —      | —      | —      | —      | —       | SWMn    |
| TXR_RXRn      | TXRX7n  | TXRX6n | TXRX5n | TXRX4n | TXRX3n | TXRX2n | TXRX1n  | TXRX0n  |
| BRDHn         | D7      | D6     | D5     | D4     | D3     | D2     | D1      | D0      |
| BRDLn         | D7      | D6     | D5     | D4     | D3     | D2     | D1      | D0      |
| UFCRn         | —       | —      | UMODn2 | UMODn1 | UMODn0 | BRDSn  | RxFTRn1 | RxFTRn0 |
| RxCNTn        | —       | —      | —      | —      | —      | D2     | D1      | D0      |

**UARTn Register List (n=0~1)**

• **UnSR Register**

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

| Bit  | 7     | 6   | 5     | 4     | 3      | 2     | 1      | 0     |
|------|-------|-----|-------|-------|--------|-------|--------|-------|
| Name | PERRn | NFn | FERRn | OERRn | RIDLEn | RXIFn | TIDLEn | TXIFn |
| R/W  | R     | R   | R     | R     | R      | R     | R      | R     |
| POR  | 0     | 0   | 0     | 0     | 1      | 0     | 1      | 1     |

- Bit 7 PERRn:** Parity error flag  
 0: No parity error is detected  
 1: Parity error is detected  
 The PERRn flag is the parity error flag. When this read only flag is “0”, it indicates a parity error has not been detected. When the flag is “1”, it indicates that the parity of the received word is incorrect. This error flag is applicable only if the parity is enabled and the parity type (odd, even, mark or space) is selected. The flag can also be cleared to zero by a software sequence which involves a read to the status register UnSR followed by an access to the TXR\_RXRn data register.
- Bit 6 NFn:** Noise flag  
 0: No noise is detected  
 1: Noise is detected  
 The NFn 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 UARTn has detected noise on the receiver input. The NFn flag is set during the same cycle as the RXIFn flag but will not be set in the case of an overrun. The NFn flag can be cleared to zero by a software sequence which will involve a read to the status register UnSR followed by an access to the TXR\_RXRn data register.
- Bit 5 FERRn:** Framing error flag  
 0: No framing error is detected  
 1: Framing error is detected  
 The FERRn flag is the framing error flag. When this read only flag is “0”, it indicates that there is no framing error. When the flag is “1”, it indicates that a framing error has been detected for the current character. The flag can also be cleared to zero by a software sequence which will involve a read to the status register UnSR followed by an access to the TXR\_RXRn data register.
- Bit 4 OERRn:** Overrun error flag  
 0: No overrun error is detected  
 1: Overrun error is detected  
 The OERRn 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\_RXRn receive data register. The flag is cleared to zero by a software sequence, which is a read to the status register UnSR followed by an access to the TXR\_RXRn data register.
- Bit 3 RIDLEn:** Receiver status  
 0: Data reception is in progress (Data being received)  
 1: No data reception is in progress (Receiver is idle)  
 The RIDLEn 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 RIDLEn bit is “1” indicating that the UARTn receiver is idle and the RXn/TXn pin stays in logic high condition.

- Bit 2**     **RXIFn:** Receive TXR\_RXRn data register status  
           0: TXR\_RXRn data register is empty  
           1: TXR\_RXRn data register has available data and Receiver FIFO trigger level is reached
- The RXIFn flag is the receive data register status flag. When this read only flag is “0”, it indicates that the TXR\_RXRn read data register is empty. When and Receiver FIFO trigger level is reached the flag is “1”, it indicates that the TXR\_RXRn read data register contains new data. When the contents of the shift register are transferred to the TXR\_RXRn register and Receiver FIFO trigger level is reached, an interrupt is generated if RIEn=1 in the UnCR2 register. If one or more errors are detected in the received word, the appropriate receive-related flags NFn, FERRn, and/or PERRn are set within the same clock cycle. The RXIFn flag will eventually be cleared to zero when the UnSR register is read with RXIFn set, followed by a read from the TXR\_RXRn register, and if the TXR\_RXRn register has no more new data available.
- Bit 1**     **TIDLEn:** Transmission idle  
           0: Data transmission is in progress (Data being transmitted)  
           1: No data transmission is in progress (Transmitter is idle)
- The TIDLEn flag is known as the transmission complete flag. When this read only flag is “0”, it indicates that a transmission is in progress. This flag will be set high when the TXIFn flag is “1” and when there is no transmit data or break character being transmitted. When TIDLEn is equal to “1”, the TXn pin becomes idle with the pin state in logic high condition. The TIDLEn flag is cleared to zero by reading the UnSR register with TIDLEn set and then writing to the TXR\_RXRn register. The flag is not generated when a data character or a break is queued and ready to be sent.
- Bit 0**     **TXIFn:** Transmit TXR\_RXRn data register status  
           0: Character is not transferred to the transmit shift register  
           1: Character has transferred to the transmit shift register (TXR\_RXRn data register is empty)
- The TXIFn 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\_RXRn data register. The TXIFn flag is cleared to zero by reading the UARTn status register (UnSR) with TXIFn set and then writing to the TXR\_RXRn data register. Note that when the TXENn bit is set, the TXIFn flag bit will also be set since the transmit data register is not yet full.

• **UnCR1 Register**

The UnCR1 register together with the UnCR2 and UnCR3 registers are the three UARTn control registers that are used to set the various options for the UARTn function, such as overall on/off control, parity control, data transfer bit length, single wire mode communication etc. Further explanation on each of the bits is given below:

| Bit  | 7       | 6     | 5     | 4     | 3     | 2      | 1    | 0    |
|------|---------|-------|-------|-------|-------|--------|------|------|
| Name | UARTENn | BNO n | PRENn | PRTn1 | PRTn0 | TXBRKn | RX8n | TX8n |
| 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**     **UARTENn:** UARTn function enable control  
           0: Disable UARTn. TXn and RXn/TXn pins are in a floating state  
           1: Enable UARTn. TXn and RXn/TXn pins can function as UARTn pins
- The UARTENn bit is the UARTn enable bit. When this bit is equal to “0”, the UARTn will be disabled and the RXn/TXn pin as well as the TXn pin will be set in a floating state. When the bit is equal to “1”, the UARTn will be enabled and the TXn and RXn/TXn pins will function as defined by the SWMn mode selection bit together with the TXENn and RXENn enable control bits.

When the UARTn 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 UARTn is disabled, all error and status flags will be reset. Also the TXENn, RXENn, TXBRKn, RXIFn, OERRn, FERRn, PERRn and NFn bits as well as the RxCNTn register will be cleared to zero, while the TIDLEn, TXIFn and RIDLEn bits will be set high. Other control bits in UnCR1, UnCR2, UnCR3, UFCRn, BRDHn and BRDLn registers will remain unaffected. If the UARTn is active and the UARTENn bit is cleared to zero, all pending transmissions and receptions will be terminated and the module will be reset as defined above. When the UARTn is re-enabled, it will restart in the same configuration.

- Bit 6     **BNO<sub>n</sub>**: 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 RX8n and TX8n will be used to store the 9th bit of the received and transmitted data respectively.
- Note that the 9th bit of data if BNO<sub>n</sub>=1, or the 8th bit of data if BNO<sub>n</sub>=0, which is used as the parity bit, does not transfer to RX8n or TX8n respectively when the parity function is enabled.
- Bit 5     **PRE<sub>n</sub>**: Parity function enable control  
           0: Parity function is disabled  
           1: Parity function is enabled
- This is the parity enable bit. When this bit is equal to “1”, the parity function will be enabled. If the bit is equal to “0”, then the parity function will be disabled.
- Bit 4~3   **PRT<sub>n1</sub>~PRT<sub>n0</sub>**: Parity type selection bits  
           00: Even parity for parity generator  
           01: Odd parity for parity generator  
           10: Mark parity for parity generator  
           11: Space parity for parity generator
- These bits are the parity type selection bits. When these bits are equal to 00b, even parity type will be selected. If these bits are equal to 01b, then odd parity type will be selected. If these bits are equal to 10b, then a 1 (Mark) in the parity bit location will be selected. If these bits are equal to 11b, then a 0 (Space) in the parity bit location will be selected.
- Bit 2     **TXBRK<sub>n</sub>**: Transmit break character  
           0: No break character is transmitted  
           1: Break characters transmit
- The TXBRK<sub>n</sub> bit is the Transmit Break Character bit. When this bit is “0”, there are no break characters and the TXn pin operates normally. When the bit is “1”, there are transmit break characters and the transmitter will send logic zeros. When this bit is equal to “1”, after the buffered data has been transmitted, the transmitter output is held low for a minimum of a 13-bit length and until the TXBRK<sub>n</sub> bit is reset.
- Bit 1     **RX8<sub>n</sub>**: Receive data bit 8 for 9-bit data transfer format (read only)
- This bit is only used if 9-bit data transfers are used, in which case this bit location will store the 9th bit of the received data known as RX8n. The BNO<sub>n</sub> bit is used to determine whether data transfers are in 8-bit or 9-bit format.
- Bit 0     **TX8<sub>n</sub>**: Transmit data bit 8 for 9-bit data transfer format (write only)
- This bit is only used if 9-bit data transfers are used, in which case this bit location will store the 9th bit of the transmitted data known as TX8n. The BNO<sub>n</sub> bit is used to determine whether data transfers are in 8-bit or 9-bit format.

• **UnCR2 Register**

The UnCR2 register is the second of the UARTn control registers and serves several purposes. One of its main functions is to control the basic enable/disable operation of the UARTn Transmitter and Receiver as well as enabling the various UARTn interrupts sources. The register also serves to control the STOP bit number selection, receiver wake-up enable and the address detect enable. Further explanation on each of the bits is given below:

| Bit  | 7     | 6     | 5      | 4      | 3     | 2    | 1      | 0      |
|------|-------|-------|--------|--------|-------|------|--------|--------|
| Name | TXENn | RXENn | STOPSn | ADDENn | WAKEn | RIEn | TIIEEn | TEIEEn |
| 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      **TXENn**: UARTn transmitter enabled control

- 0: UARTn transmitter is disabled
- 1: UARTn transmitter is enabled

The bit named TXENn 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 TXn pin will be set in a floating state.

If the TXENn bit is equal to “1” and the UARTEEn bit are also equal to “1”, the transmitter will be enabled and the TXn pin will be controlled by the UARTn. Clearing the TXENn bit during a transmission will cause the data transmission to be aborted and will reset the transmitter. If this situation occurs, the TXn pin will be set in a floating state.

Bit 6      **RXENn**: UARTn Receiver enabled control

- 0: UARTn receiver is disabled
- 1: UARTn receiver is enabled

The bit named RXENn is the Receiver Enable Bit. When this bit is equal to “0”, the receiver will be disabled with any pending data receptions being aborted. In addition the receive buffers will be reset. In this situation the RXn/TXn pin will be set in a floating state. If the RXENn bit is equal to “1” and the UARTEEn bit is also equal to “1”, the receiver will be enabled and the RXn/TXn pin will be controlled by the UARTn. Clearing the RXENn bit during a reception will cause the data reception to be aborted and will reset the receiver. If this situation occurs, the RXn/TXn pin will be set in a floating state.

Bit 5      **STOPSn**: Number of stop bits selection for transmitter

- 0: One stop bit format is used
- 1: Two stop bits format is used

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

Bit 4      **ADDENn**: Address detect function enable control

- 0: Address detect function is disabled
- 1: Address detect function is enabled

The bit named ADDENn is the address detect function enable control bit. When this bit is equal to “1”, the address detect function is enabled. When it occurs, if the 8th bit, which corresponds to TXRX7n if BNO=0 or the 9th bit, which corresponds to RX8n if BNO=1, has a value of “1”, then the received word will be identified as an address, rather than data. If the corresponding interrupt is enabled, an interrupt request will be generated each time the received word has the address bit set, which is the 8th or 9th bit depending on the value of BNO. If the address bit known as the 8th or 9th bit of the received word is “0” with the address detect function being enabled, an interrupt will not be generated and the received data will be discarded.

- Bit 3     **WAKEn**: RXn/TXn pin wake-up UARTn function enable control  
           0: RXn/TXn pin wake-up UARTn function is disabled  
           1: RXn/TXn pin wake-up UARTn function is enabled  
 This bit is used to control the wake-up UARTn function when a falling edge on the RXn/TXn pin occurs. Note that this bit is only available when the UARTn clock ( $f_{Hn}$ ) is switched off. There will be no RXn/TXn pin wake-up UARTn function if the UARTn clock ( $f_{Hn}$ ) exists. If the WAKEn bit is set to 1 as the UARTn clock ( $f_{Hn}$ ) is switched off, a UARTn wake-up request will be initiated when a falling edge on the RXn/TXn pin occurs. When this request happens and the corresponding interrupt is enabled, an RXn/TXn pin wake-up UARTn interrupt will be generated to inform the MCU to wake up the UARTn function by switching on the UARTn clock ( $f_{Hn}$ ) via the application program. Otherwise, the UARTn function cannot resume even if there is a falling edge on the RXn/TXn pin when the WAKEn bit is cleared to 0.
- Bit 2     **RIEn**: Receiver interrupt enable control  
           0: Receiver related interrupt is disabled  
           1: Receiver related interrupt is enabled  
 This bit enables or disables the receiver interrupt. If this bit is equal to “1” and when the receiver overrun flag OERRn or receive data available flag RXIFn is set, the UARTn interrupt request flag will be set. If this bit is equal to “0”, the UARTn interrupt request flag will not be influenced by the condition of the OERRn or RXIFn flags.
- Bit 1     **TIEn**: Transmitter Idle interrupt enable control  
           0: Transmitter idle interrupt is disabled  
           1: Transmitter idle interrupt is enabled  
 This bit enables or disables the transmitter idle interrupt. If this bit is equal to “1” and when the transmitter idle flag TIDLEn is set, due to a transmitter idle condition, the UARTn interrupt request flag will be set. If this bit is equal to “0”, the UARTn interrupt request flag will not be influenced by the condition of the TIDLEn flag.
- Bit 0     **TEIEn**: Transmitter Empty interrupt enable control  
           0: Transmitter empty interrupt is disabled  
           1: Transmitter empty interrupt is enabled  
 This bit enables or disables the transmitter empty interrupt. If this bit is equal to “1” and when the transmitter empty flag TXIFn is set, due to a transmitter empty condition, the UARTn interrupt request flag will be set. If this bit is equal to “0”, the UARTn interrupt request flag will not be influenced by the condition of the TXIFn flag.

• **UnCR3 Register**

The UnCR3 register is used to enable the UARTn Single Wire Mode communication. As the name suggests in the single wire mode the UARTn communication can be implemented in one single line, RXn/TXn, together with the control of the RXENn and TXENn bits in the UnCR2 register.

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

- Bit 7~1   Unimplemented, read as “0”
- Bit 0     **SWMn**: Single Wire Mode enable control  
           0: Disable, the RXn/TXn pin is used as UARTn receiver function only  
           1: Enable, the RXn/TXn pin can be used as UARTn receiver or transmitter function controlled by the RXENn and TXENn bits  
 Note that when the Single Wire Mode is enabled, if both the RXENn and TXENn bits are high, the RXn/TXn pin will just be used as UARTn receiver input.

**• TXR\_RXRn Register**

The TXR\_RXRn register is the data register which is used to store the data to be transmitted on the TXn pin or being received from the RXn/TXn pin.

| Bit  | 7      | 6      | 5      | 4      | 3      | 2      | 1      | 0      |
|------|--------|--------|--------|--------|--------|--------|--------|--------|
| Name | TXRX7n | TXRX6n | TXRX5n | TXRX4n | TXRX3n | TXRX2n | TXRX1n | TXRX0n |
| 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 **TXRX7n~TXRX0n**: UARTn Transmit/Receive Data bit 7 ~ bit 0

**• BRDHn Register**

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

Bit 7~0 **D7~D0**: Baud Rate divider high byte

The baud rate divider BRDn (BRDHn/BRDLn) defines the UARTn clock divider ratio.  
 $\text{Baud Rate} = f_{\text{H}} / (\text{BRDn} + \text{UMODn} / 8)$

$\text{BRDn} = 16 \sim 65535$  or  $8 \sim 65535$  depending on BRDSn

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

2. The BRDLn must be written first and then BRDHn, otherwise errors may occur.

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

**• BRDLn Register**

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

Bit 7~0 **D7~D0**: Baud Rate divider low byte

The baud rate divider BRDn (BRDHn/BRDLn) defines the UARTn clock divider ratio.  
 $\text{Baud Rate} = f_{\text{H}} / (\text{BRDn} + \text{UMODn} / 8)$

$\text{BRDn} = 16 \sim 65535$  or  $8 \sim 65535$  depending on BRDSn

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

2. The BRDLn must be written first and then BRDHn, otherwise errors may occur.

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

• **UFCRn Register**

The UFCRn register is the FIFO control register which is used for UARTn modulation control, BRDn range selection and trigger level selection for RXIFn and interrupt.

| Bit  | 7 | 6 | 5      | 4      | 3      | 2     | 1       | 0       |
|------|---|---|--------|--------|--------|-------|---------|---------|
| Name | — | — | UMODn2 | UMODn1 | UMODn0 | BRDSn | RxFTRn1 | RxFTRn0 |
| R/W  | — | — | R/W    | R/W    | R/W    | R/W   | R/W     | R/W     |
| POR  | — | — | 0      | 0      | 0      | 0     | 0       | 0       |

Bit 7~6 Unimplemented, read as “0”

Bit 5~3 **UMODn2~UMODn0**: UARTn Modulation Control bits

The modulation control bits are used to correct the baud rate of the received or transmitted UARTn signal. These bits determine if the extra UARTn clock cycle should be added in a UARTn bit time. The UMODn2~UMODn0 will be added to internal accumulator for every UARTn bit time. Until a carry to bit 3, the corresponding UARTn bit time increases a UARTn clock cycle.

Bit 2 **BRDSn**: BRDn range selection

0: BRDn range is from 16 to 65535

1: BRDn range is from 8 to 65535

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

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

Bit 1~0 **RxFTRn1~RxFTRn0**: Receiver FIFO trigger level (bytes)

00: 4 bytes in Receiver FIFO

01: 1 or more bytes in Receiver FIFO

10: 2 or more bytes in Receiver FIFO

11: 3 or more bytes in Receiver FIFO

For the receiver these bits define the number of received data bytes in the Receiver FIFO that will trigger the RXIFn bit being set high, an interrupt will also be generated if the RIEN bit is enabled. After the reset the receiver FIFO is empty.

• **RxCNTn Register**

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

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

Bit 7~3 Unimplemented, read as “0”

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

The RxCNTn register is the counter used to indicate the number of receiver data bytes in Receiver FIFO which is not read by MCU. When Receiver FIFO receives one byte data, the RxCNTn will increase by one; when the MCU reads one byte data from Receiver FIFO, the RxCNTn will decrease by one. If there are 4 bytes of data in the Receiver FIFO, the 5<sup>th</sup> data will be saved in the shift register. If there is 6<sup>th</sup> data, the 6<sup>th</sup> data will be saved in the shift register. But the RxCNTn remains the value of 4. The RxCNTn will be cleared when reset occurs or UARTENn=1. This register is read only.

### Baud Rate Generator

To setup the speed of the serial data communication, the UARTn function contains its own dedicated baud rate generator. The baud rate is controlled by its own internal free running 16-bit timer, the period of which is determined by two factors. The first of these is the value placed in BRDHn/BRDLn register and the second is the UARTn modulation control bits (UMODn2~UMODn0). If a baud rate BR is required with UARTn clock  $f_H$ .

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

The integer part is loaded into BRDn (BRDHn/BRDLn). The fractional part is multiplied by 8 and rounded, then loaded into UMODn bit field as following:

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

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

Therefore, the actual baud rate is as following:

$$\text{Baud rate} = f_H / [BRDn + (UMODn/8)]$$

### Calculating the Baud Rate and Error Values

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

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

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

The actual Baud Rate =  $f_H / [BRDn + (UMODn/8)] = 230215.83$

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

### Modulation Control Example

To get the best-fitting bit sequence for UARTn modulation control bits UMODn2~UMODn0, the following algorithm can be used: Firstly, the fractional part of the theoretical division factor is multiplied by 8. Then the product will be rounded and UMODn2~UMODn0 bits will be filled with the rounded value. The UMODn2~UMODn0 bits will be added to internal accumulator for every UARTn bit time. Until a carry to bit 3, the corresponding UARTn bit time increases a UARTn clock cycle. The following is an example using the fraction 0.36111 previously calculated:  $UMODn[2:0] = \text{ROUND}(0.36111 \times 8) = 011b$ .

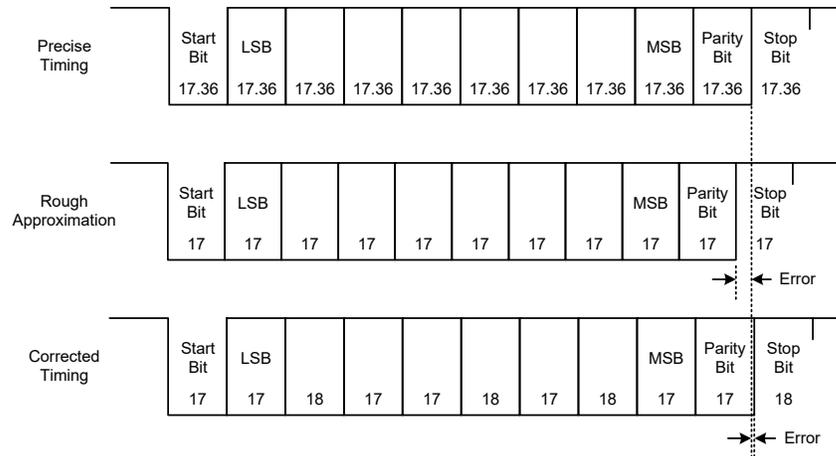
| Fraction Addition | Carry to Bit 3 | UARTn Bit Time Sequence | Extra UARTn Clock Cycle |
|-------------------|----------------|-------------------------|-------------------------|
| 0000b+0011b=0011b | No             | Start bit               | No                      |
| 0011b+0011b=0110b | No   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

### Baud Rate Correction Example

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

The following figure shows three different frames:

- The upper frame is the correct one, with a bit-length of 17.36  $f_H$  cycles ( $4000000/230400=17.36$ ).
- The middle frame uses a rough estimate, with 17  $f_H$  cycles for the bit length.
- The lower frame shows a corrected frame using the best fit for the UARTn modulation control bits UMODn2~UMODn0.



### UART Setup and Control

For data transfer, the UARTn 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 UARTn hardware, and can be setup to be even, odd, mark, space or no parity. For the most common data format, 8 data bits along with no parity and one stop bit, denoted as 8, N, 1, is used as the default setting, which is the setting at power-on. The number of data bits and stop bits, along with the parity, are setup by programming the corresponding BNO<sub>n</sub>, PRTn1~PRTn0, PREN<sub>n</sub>, and STOPS<sub>n</sub> bits. The baud rate used to transmit and receive data is setup using the internal 16-bit baud rate generator, while the data is transmitted and received LSB first. Although the UARTn transmitter and receiver are functionally independent, they both use the same data format and baud rate. In all cases stop bits will be used for data transmission.

#### Enabling/Disabling the UARTn Interface

The basic on/off function of the internal UARTn function is controlled using the UARTEN<sub>n</sub> bit in the UnCR1 register. If the UARTEN<sub>n</sub>, TXEN<sub>n</sub> and RXEN<sub>n</sub> bits are set, then these two UARTn pins will act as normal TX<sub>n</sub> output pin and RX<sub>n</sub>/TX<sub>n</sub> input pin respectively. If no data is being transmitted on the TX<sub>n</sub> pin, then it will default to a logic high value.

Clearing the UARTEN<sub>n</sub> bit will disable the TX<sub>n</sub> and RX<sub>n</sub>/TX<sub>n</sub> pins and allow these two pins to be used as normal I/O or other pin-shared functional pins by configuring the corresponding pin-shared control bits. When the UARTn function is disabled the buffer will be reset to an empty condition, at the same time discarding any remaining residual data. Disabling the UARTn will also reset the error and status flags with bits TXEN<sub>n</sub>, RXEN<sub>n</sub>, TXBRK<sub>n</sub>, RXIF<sub>n</sub>, OERR<sub>n</sub>, FERR<sub>n</sub>, PERR<sub>n</sub> and NFN as well as register RxCNT<sub>n</sub> being cleared while bits TIDLE<sub>n</sub>, TXIF<sub>n</sub> and RIDLE<sub>n</sub> will be set. The

remaining control bits in the UnCR1, UnCR2, UnCR3, UFCRn, BRDHn and BRDLn registers will remain unaffected. If the UARTENn bit in the UnCR1 register is cleared while the UARTn is active, then all pending transmissions and receptions will be immediately suspended and the UARTn will be reset to a condition as defined above. If the UARTn is then subsequently re-enabled, it will restart again in the same configuration.

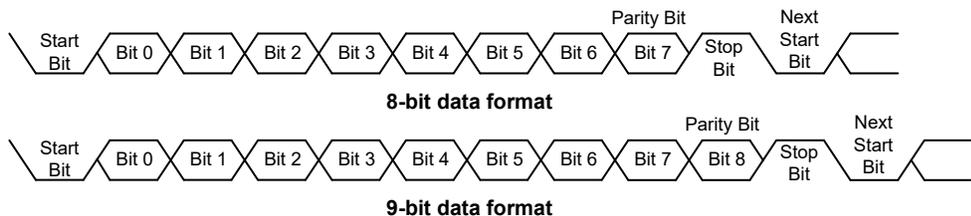
**Data, Parity and Stop Bit Selection**

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 UnCR1 and UnCR2 registers. The BNO n bit controls the number of data bits which can be set to either 8 or 9, the PRTn1~PRTn0 bits control the choice of odd, even, mark or space parity, the PRENn bit controls the parity on/off function and the STOPSn 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 is only used for the transmitter. There is only one stop bit for the receiver.

| Start Bit                            | Data Bits | Address Bit | 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 n bit in the UnCR1 register. When BNO n bit is set, the word length will be set to 9 bits. In this case the 9th bit, which is the MSB, needs to be stored in the TX8n bit in the UnCR1 register. At the transmitter core lies the Transmitter Shift Register, more commonly known as the TSRn, whose data is obtained from the transmit data register, which is known as the TXR\_RXRn register. The data to be transmitted is loaded into this TXR\_RXRn register by the application program. The TSRn 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 TSRn can then be loaded with new data from the TXR\_RXRn register, if it is available. It should be noted that the TSRn 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 TXENn bit is set, but the data will not be transmitted until the TXR\_RXRn 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\_RXRn register, after which the TXENn bit can be set. When a transmission of data begins, the TSRn is normally empty, in which case a transfer to the TXR\_RXRn register will result in an immediate transfer to the TSRn. If during a transmission the TXENn bit is cleared, the transmission will immediately cease and the transmitter will be reset. The TXn output pin can then be configured as the I/O or other pin-shared function by configuring the corresponding pin-shared control bits.

### **Transmitting Data**

When the UARTn is transmitting data, the data is shifted on the TXn pin from the shift register, with the least significant bit first. In the transmit mode, the TXR\_RXRn 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 TX8n bit in the UnCR1 register. The steps to initiate a data transfer can be summarized as follows:

- Make the correct selection of the BNO<sub>n</sub>, PRTn1~PRTn0, PREN<sub>n</sub> and STOPS<sub>n</sub> bits to define the required word length, parity type and number of stop bits.
- Setup the BRDH<sub>n</sub>, BRDL<sub>n</sub> registers and UMODn2~UMODn0 bits to select the desired baud rate.
- Set the TXENn bit to ensure that the TXn pin is used as a UARTn transmitter pin.
- Access the UnSR register and write the data that is to be transmitted into the TXR\_RXRn register. Note that this step will clear the TXIFn bit.

This sequence of events can now be repeated to send additional data. It should be noted that when TXIFn=0, data will be inhibited from being written to the TXR\_RXRn register. Clearing the TXIFn flag is always achieved using the following software sequence:

1. A UnSR register access
2. A TXR\_RXRn register write execution

The read-only TXIFn flag is set by the UARTn hardware and if set indicates that the TXR\_RXRn register is empty and that other data can now be written into the TXR\_RXRn register without overwriting the previous data. If the TEIE<sub>n</sub> bit is set then the TXIFn flag will generate an interrupt.

During a data transmission, a write instruction to the TXR\_RXRn register will place the data into the TXR\_RXRn 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\_RXRn register will place the data directly into the shift register, resulting in the commencement of data transmission, and the TXIFn bit being immediately set. When a frame transmission is complete, which happens after stop bits are sent or after the break frame, the TIDLE<sub>n</sub> bit will be set. To clear the TIDLE<sub>n</sub> bit the following software sequence is used:

1. A UnSR register access
2. A TXR\_RXRn register write execution

Note that both the TXIFn and TIDLE<sub>n</sub> bits are cleared by the same software sequence.

### Transmitting Break

If the TXBRK<sub>n</sub> bit is set high and the state keeps for a time greater than  $[(BRD_{n+1}) \times t_{H}]$  while TIDLE<sub>n</sub>=1, then break characters will be sent on the next transmission. Break character transmission consists of a start bit, followed by  $13 \times N$  '0' bits and stop bits, where  $N=1, 2, \text{etc.}$  If a break character is to be transmitted then the TXBRK<sub>n</sub> 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<sub>n</sub> 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<sub>n</sub> bit, the transmitter will finish transmitting the last break character and subsequently send out one or two stop bits. The automatic logic highs at the end of the last break character will ensure that the start bit of the next frame is recognized.

### UART Receiver

The UART<sub>n</sub> is capable of receiving word lengths of either 8 or 9 bits. If the BNO<sub>n</sub> bit is set, the word length will be set to 9 bits with the MSB being stored in the RX8<sub>n</sub> bit of the UnCR1 register. At the receiver core lies the Receive Serial Shift Register, commonly known as the RSR<sub>n</sub>. The data which is received on the RX<sub>n</sub>/TX<sub>n</sub> 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<sub>n</sub>/TX<sub>n</sub> pin is sampled for the stop bit, the received data in RSR<sub>n</sub> is transferred to the receive data register, if the register is empty. The data which is received on the external RX<sub>n</sub>/TX<sub>n</sub> input pin is sampled three times by a majority detect circuit to determine the logic level that has been placed onto the RX<sub>n</sub>/TX<sub>n</sub> pin. It should be noted that the RSR<sub>n</sub> 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<sub>n</sub> receiver is receiving data, the data is serially shifted in on the external RX<sub>n</sub>/TX<sub>n</sub> input pin, LSB first. In the read mode, the TXR\_RXR<sub>n</sub> register forms a buffer between the internal bus and the receiver shift register. The TXR\_RXR<sub>n</sub> register is a four-byte deep FIFO data buffer, where four bytes can be held in the FIFO while a fifth byte can continue to be received. Note that the application program must ensure that the data is read from TXR\_RXR<sub>n</sub> before the fifth byte has been completely shifted in, otherwise this fifth byte will be discarded and an overrun error OERR<sub>n</sub> will be subsequently indicated. The steps to initiate a data transfer can be summarized as follows:

- Make the correct selection of BNO<sub>n</sub>, PRT<sub>n1</sub>~PRT<sub>n0</sub> and PREN<sub>n</sub> bits to define the word length, parity type.
- Setup the BRDH<sub>n</sub>, BRDL<sub>n</sub> registers and the UMOD<sub>n2</sub>~UMOD<sub>n0</sub> to select the desired baud rate.
- Set the RXEN<sub>n</sub> bit to ensure that the RX<sub>n</sub>/TX<sub>n</sub> pin is used as a UART<sub>n</sub> 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<sub>n</sub> bit in the UnSR register will be set when the TXR\_RXR<sub>n</sub> register has data available, the number of the available data bytes can be checked by polling the RxCNT<sub>n</sub> register content.
- When the contents of the shift register have been transferred to the TXR\_RXR<sub>n</sub> register and Receiver FIFO trigger level is reached, if the RIEN 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<sub>n</sub> bit can be cleared using the following software sequence:

1. A UnSR register access
2. A TXR\_RXR<sub>n</sub> register read execution

### Receiving Break

Any break character received by the UARTn 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<sub>n</sub> bit plus 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<sub>n</sub> plus one stop bit. The RXIF<sub>n</sub> bit is set, FERR<sub>n</sub> is set, zeros are loaded into the receive data register, interrupts are generated if appropriate and the RIDLE<sub>n</sub> bit is set. A break is regarded as a character that contains only zeros with the FERR<sub>n</sub> 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<sub>n</sub> 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<sub>n</sub> read only flag will go high when the stop bits have not yet been received. The reception of a break character on the UARTn registers will result in the following:

- The framing error flag, FERR<sub>n</sub>, will be set.
- The receive data register, TXR\_RXR<sub>n</sub>, will be cleared.
- The OERR<sub>n</sub>, NF<sub>n</sub>, PERR<sub>n</sub>, RIDLE<sub>n</sub> or RXIF<sub>n</sub> 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 UnSR register, otherwise known as the RIDLE<sub>n</sub> flag, will have a zero value. In between the reception of a stop bit and the detection of the next start bit, the RIDLE<sub>n</sub> flag will have a high value, which indicates the receiver is in an idle condition.

### Receiver Interrupt

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

## Managing Receiver Errors

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

### Overrun Error – OERR<sub>n</sub>

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

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

- The OERR<sub>n</sub> flag in the UnSR register will be set.
- The TXR\_RXR<sub>n</sub> contents will not be lost.
- The shift register will be overwritten.
- An interrupt will be generated if the RIEN bit is set.

The OERR<sub>n</sub> flag can be cleared by an access to the UnSR register followed by a read to the TXR\_RXR<sub>n</sub> register.

### Noise Error – NF<sub>n</sub>

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<sub>n</sub>, in the UnSR register will be set on the rising edge of the RXIF<sub>n</sub> bit.
- Data will be transferred from the shift register to the TXR\_RXR<sub>n</sub> register.
- No interrupt will be generated. However this bit rises at the same time as the RXIF<sub>n</sub> bit which itself generates an interrupt.

Note that the NF<sub>n</sub> flag is reset by a UnSR register read operation followed by a TXR\_RXR<sub>n</sub> register read operation.

### Framing Error – FERR<sub>n</sub>

The read only framing error flag, FERR<sub>n</sub>, in the UnSR 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<sub>n</sub> flag will be set. The FERR<sub>n</sub> flag and the received data will be recorded in the UnSR and TXR\_RXR<sub>n</sub> registers respectively, and the flag is cleared in any reset.

### Parity Error – PERR<sub>n</sub>

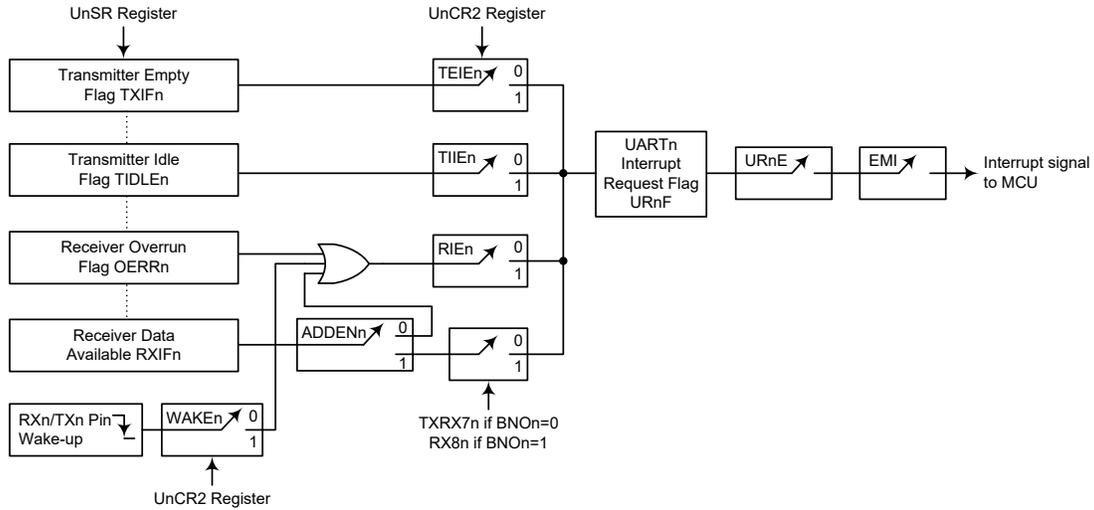
The read only parity error flag, PERR<sub>n</sub>, in the UnSR register, is set if the parity of the received word is incorrect. This error flag is only applicable if the parity is enabled, PREN<sub>n</sub>=1, and if the parity type, odd or even is selected. The read only PERR<sub>n</sub> flag and the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

## UART Interrupt Structure

Several individual UART<sub>n</sub> conditions can generate a UART<sub>n</sub> interrupt. When these conditions exist, a low pulse will be generated to get the attention of the microcontroller. These conditions are a transmitter data register empty, transmitter idle, receiver reaching FIFO trigger level, receiver overrun, address detect and an RX<sub>n</sub>/TX<sub>n</sub> pin wake-up. When any of these conditions are created, if the global interrupt enable bit and its corresponding interrupt control bit are enabled and the stack is not full, the program will jump to its corresponding interrupt vector where it can be serviced before returning to the main program. Four of these conditions have the corresponding UnSR register flags which will generate a UART<sub>n</sub> interrupt if its associated interrupt enable control bit in the UnCR2 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<sub>n</sub> interrupt sources.

The address detect condition, which is also a UART<sub>n</sub> interrupt source, does not have an associated flag, but will generate a UART<sub>n</sub> interrupt when an address detect condition occurs if its function is enabled by setting the ADDEN<sub>n</sub> bit in the UnCR2 register. An RX<sub>n</sub>/TX<sub>n</sub> pin wake-up, which is also a UART<sub>n</sub> interrupt source, does not have an associated flag, but will generate a UART<sub>n</sub> interrupt if the UART<sub>n</sub> clock ( $f_{H}$ ) source is switched off and the WAKEN and RIEN bits in the UnCR2 register are set when a falling edge on the RX<sub>n</sub>/TX<sub>n</sub> pin occurs.

Note that the UnSR 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<sub>n</sub>, the details of which are given in the UART<sub>n</sub> register section. The overall UART<sub>n</sub> 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<sub>n</sub> module is masked out or allowed.



UARTn Interrupt Structure (n=0~1)

### Address Detect Mode

Setting the Address Detect Mode bit, ADDENn, in the UnCR2 register, enables this special mode. If this bit is enabled then an additional qualifier will be placed on the generation of a Receiver Data Available interrupt, which is requested by the RXIFn flag. If the ADDENn bit is enabled, then when data is available, an interrupt will only be generated if the highest received bit has a high value. Note that the URnE and EMI interrupt enable bits must also be enabled for correct interrupt generation. This highest address bit is the 9th bit if BNO=1 or the 8th bit if BNO=0. If this bit is high, then the received word will be defined as an address rather than data. A Data Available interrupt will be generated every time the last bit of the received word is set. If the ADDENn bit is not enabled, then a Receiver Data Available interrupt will be generated each time the RXIFn flag is set, irrespective of the data last bit status. The address detect mode and parity enable are mutually exclusive functions. Therefore if the address detect mode is enabled, then to ensure correct operation, the parity function should be disabled by resetting the parity enable bit PRENn to zero.

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

ADDENn Bit Function (n=0~1)

### UART Power Down and Wake-up

When the UARTn clock, f<sub>H</sub>, is switched off, the UARTn will cease to function, all clock sources to the module are shutdown. If the UARTn clock, f<sub>H</sub>, is off while a transmission is still in progress, then the transmission will be paused until the UARTn clock source derived from the microcontroller is activated. In a similar way, if the MCU enters the IDLE or SLEEP mode while receiving data, then the reception of data will likewise be paused. When the MCU enters the IDLE or SLEEP mode, note that the UnSR, UnCR1, UnCR2, UnCR3, UFRCn, RxCNTn and TXR\_RXRn registers, as well as the BRDHn/BRDLn register will not be affected. It is recommended to make sure first that the UARTn data transmission or reception has been finished before the microcontroller enters the IDLE or SLEEP mode.

The UARTn function contains a receiver RXn/TXn pin wake-up function, which is enabled or disabled by the WAKEn bit in the UnCR2 register. If this bit, along with the UARTn enable bit, UARTENn, the receiver enable bit, RXENn and the receiver interrupt bit, RInE, are all set when the UARTn clock (f<sub>rt</sub>) is off, then a falling edge on the RXn/TXn pin will trigger an RXn/TXn pin wake-up UARTn interrupt. Note that as it takes certain system clock cycles after a wake-up, before normal microcontroller operation resumes, any data received during this time on the RXn/TXn pin will be ignored.

For a UARTn wake-up interrupt to occur, in addition to the bits for the wake-up being set, the global interrupt enable bit, EMI, and the UARTn interrupt enable bit, URnE, must be set. If the EMI and URnE 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 UARTn interrupt will not be generated until after this time has elapsed.

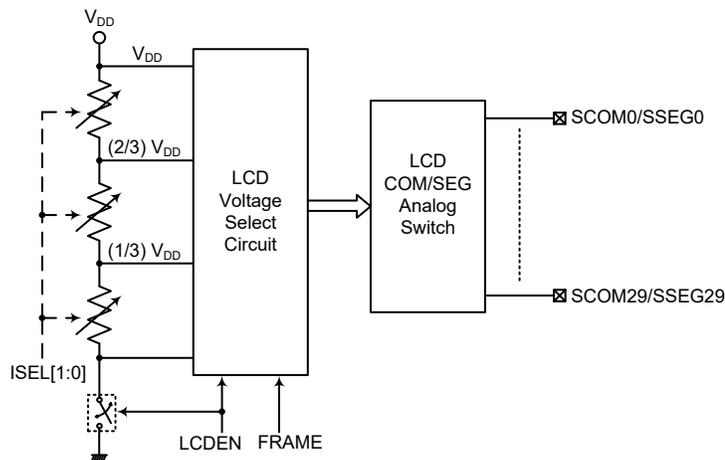
### SCOM/SSEG Function for LCD

The device has the capability of driving external LCD panels. The common and segment pins for LCD driving, SCOM0~SCOM29 and SSEG0~SSEG29, are pin-shared with certain pins on the I/O ports. The LCD signals, COM and SEG, are generated using the application program.

#### LCD Operation

An external LCD panel can be driven using the device by configuring the I/O pins as common pins and segment pins. The LCD driver function is controlled using the LCD control registers which in addition to controlling the overall on/off function also controls the bias voltage setup function. This enables the LCD COM and SEG driver to generate the necessary V<sub>SS</sub>, (1/3) V<sub>DD</sub>, (2/3) V<sub>DD</sub> and V<sub>DD</sub> voltage levels for LCD 1/3 bias operation.

The LCDEN bit in the SLCDC0 register is the overall master control for the LCD driver. This bit is used in conjunction with the corresponding pin-shared function selection bits to select which I/O pins are used for LCD driving. Note that the corresponding Port Control register does not need to first setup the pins as outputs to enable the LCD driver operation.



**Software Controlled LCD Driver Structure**

**LCD Frames**

A cyclic LCD waveform includes two frames known as Frame 0 and Frame 1 for which the following offers a functional explanation.

**Frame 0**

To select Frame 0, clear the FRAME bit in the SLCDC0 register to 0.

In frame 0, the COM signal output can have a value of  $V_{DD}$  or a  $V_{BIAS}$  value of  $(1/3) V_{DD}$ . The SEG signal output can have a value of  $V_{SS}$  or a  $V_{BIAS}$  value of  $(2/3) V_{DD}$ .

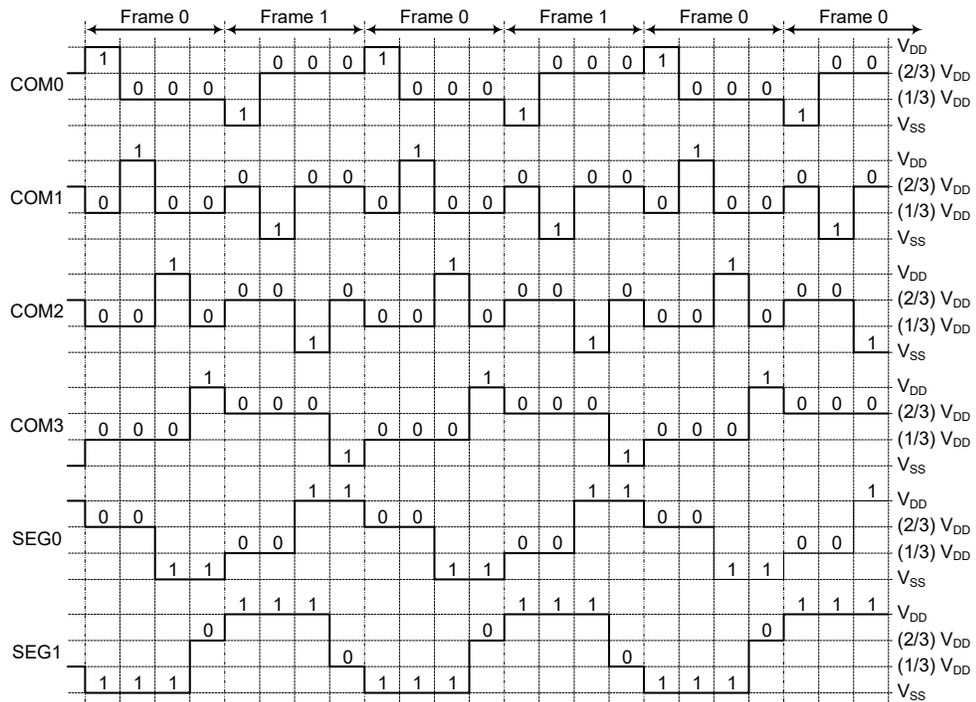
**Frame 1**

To select Frame 1, set the FRAME bit in the SLCDC0 register to 1.

In frame 1, the COM signal output can have a value of  $V_{SS}$  or a  $V_{BIAS}$  value of  $(2/3) V_{DD}$ . The SEG signal output can have a value of  $V_{DD}$  or a  $V_{BIAS}$  value of  $(1/3) V_{DD}$ .

The SCOMm waveform is controlled by the application program using the FRAME bit in the SLCDC0 register and the corresponding pin-shared I/O data bit for the respective SCOMm pin to determine whether the SCOMm output has a value of  $V_{DD}$ ,  $V_{SS}$  or  $V_{BIAS}$ . The SSEGn waveform is controlled in a similar way using the FRAME bit and the corresponding pin-shared I/O data bit for the respective SSEGn pin to determine whether the SSEGn output has a value of  $V_{DD}$ ,  $V_{SS}$  or  $V_{BIAS}$ .

The accompanying waveform diagram shows a typical 1/3 bias LCD waveform generated using the application program together with the LCD voltage select circuit. Note that the depiction of a “1” in the diagram illustrates an illuminated LCD pixel. The COM signal polarity generated on pins SCOM0~SCOM29, whether “0” or “1”, are generated using the corresponding pin-shared I/O data register bit.



Note: The logical values shown in the above diagram are the corresponding pin-shared I/O data bit value.

**1/3 Bias LCD Waveform – 4-COM & 2-SEG Application**

## LCD Control Registers

The LCD SCOM and SSEG 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 SLCDC0 register. All SCOM and SSEG pins are pin-shared with I/O pins and selected as SCOM and SSEG pins using the corresponding pin function selection bits in the SLCDS0~SLCDS3 registers respectively.

| Register Name | Bit       |           |           |           |           |           |           |           |
|---------------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|
|               | 7         | 6         | 5         | 4         | 3         | 2         | 1         | 0         |
| SLCDC0        | FRAME     | ISEL1     | ISEL0     | LCDEN     | —         | —         | —         | —         |
| SLCDS0        | COMSEGS7  | COMSEGS6  | COMSEGS5  | COMSEGS4  | COMSEGS3  | COMSEGS2  | COMSEGS1  | COMSEGS0  |
| SLCDS1        | COMSEGS15 | COMSEGS14 | COMSEGS13 | COMSEGS12 | COMSEGS11 | COMSEGS10 | COMSEGS9  | COMSEGS8  |
| SLCDS2        | COMSEGS23 | COMSEGS22 | COMSEGS21 | COMSEGS20 | COMSEGS19 | COMSEGS18 | COMSEGS17 | COMSEGS16 |
| SLCDS3        | —         | —         | COMSEGS29 | COMSEGS28 | COMSEGS27 | COMSEGS26 | COMSEGS25 | COMSEGS24 |

LCD Driver Control Register List

### • SLCDC0 Register

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

Bit 7 **FRAME**: SCOMn/SSEGN Output Frame selection  
 0: Frame 0  
 1: Frame 1

Bit 6~5 **ISEL1~ISEL0**: Select resistor for R type LCD bias current (@ $V_{DD}=5V$ )  
 00:  $3 \times 200k\Omega$  (1/3 Bias),  $I_{BIAS}=8.3\mu A$   
 01:  $3 \times 100k\Omega$  (1/3 Bias),  $I_{BIAS}=16.6\mu A$   
 10:  $3 \times 33.3k\Omega$  (1/3 Bias),  $I_{BIAS}=50\mu A$   
 11:  $3 \times 16.6k\Omega$  (1/3 Bias),  $I_{BIAS}=100\mu A$

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

When the LCDEN bit is cleared to 0, then the SCOMm and SSEGN outputs will be fixed at a  $V_{SS}$  level.

Bit 3~0 Unimplemented, read as “0”

### • SLCDS0 Register

| Bit  | 7        | 6        | 5        | 4        | 3        | 2        | 1        | 0        |
|------|----------|----------|----------|----------|----------|----------|----------|----------|
| Name | COMSEGS7 | COMSEGS6 | COMSEGS5 | COMSEGS4 | COMSEGS3 | COMSEGS2 | COMSEGS1 | COMSEGS0 |
| 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 **COMSEGS7**: SCOM7/SSEG7 pin function selection  
 0: SCOM7  
 1: SSEG7

Bit 6 **COMSEGS6**: SCOM6/SSEG6 pin function selection  
 0: SCOM6  
 1: SSEG6

Bit 5 **COMSEGS5**: SCOM5/SSEG5 pin function selection  
 0: SCOM5  
 1: SSEG5

- Bit 4      **COMSEGS4:** SCOM4/SSEG4 pin function selection  
             0: SCOM4  
             1: SSEG4
- Bit 3      **COMSEGS3:** SCOM3/SSEG3 pin function selection  
             0: SCOM3  
             1: SSEG3
- Bit 2      **COMSEGS2:** SCOM2/SSEG2 pin function selection  
             0: SCOM2  
             1: SSEG2
- Bit 1      **COMSEGS1:** SCOM1/SSEG1 pin function selection  
             0: SCOM1  
             1: SSEG1
- Bit 0      **COMSEGS0:** SCOM0/SSEG0 pin function selection  
             0: SCOM0  
             1: SSEG0

• **SLCDS1 Register**

| Bit  | 7         | 6         | 5         | 4         | 3         | 2         | 1        | 0        |
|------|-----------|-----------|-----------|-----------|-----------|-----------|----------|----------|
| Name | COMSEGS15 | COMSEGS14 | COMSEGS13 | COMSEGS12 | COMSEGS11 | COMSEGS10 | COMSEGS9 | COMSEGS8 |
| 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      **COMSEGS15:** SCOM15/SSEG15 pin function selection  
             0: SCOM15  
             1: SSEG15
- Bit 6      **COMSEGS14:** SCOM14/SSEG14 pin function selection  
             0: SCOM14  
             1: SSEG14
- Bit 5      **COMSEGS13:** SCOM13/SSEG13 pin function selection  
             0: SCOM13  
             1: SSEG13
- Bit 4      **COMSEGS12:** SCOM12/SSEG12 pin function selection  
             0: SCOM12  
             1: SSEG12
- Bit 3      **COMSEGS11:** SCOM11/SSEG11 pin function selection  
             0: SCOM11  
             1: SSEG11
- Bit 2      **COMSEGS10:** SCOM10/SSEG10 pin function selection  
             0: SCOM10  
             1: SSEG10
- Bit 1      **COMSEGS9:** SCOM9/SSEG9 pin function selection  
             0: SCOM9  
             1: SSEG9
- Bit 0      **COMSEGS8:** SCOM8/SSEG8 pin function selection  
             0: SCOM8  
             1: SSEG8

• **SLCDS2 Register**

| Bit  | 7         | 6         | 5         | 4         | 3         | 2         | 1         | 0         |
|------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|
| Name | COMSEGS23 | COMSEGS22 | COMSEGS21 | COMSEGS20 | COMSEGS19 | COMSEGS18 | COMSEGS17 | COMSEGS16 |
| 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      **COMSEGS23**: SCOM23/SSEG23 pin function selection  
0: SCOM23  
1: SSEG23
- Bit 6      **COMSEGS22**: SCOM22/SSEG22 pin function selection  
0: SCOM22  
1: SSEG22
- Bit 5      **COMSEGS21**: SCOM21/SSEG21 pin function selection  
0: SCOM21  
1: SSEG21
- Bit 4      **COMSEGS20**: SCOM20/SSEG20 pin function selection  
0: SCOM20  
1: SSEG20
- Bit 3      **COMSEGS19**: SCOM19/SSEG19 pin function selection  
0: SCOM19  
1: SSEG19
- Bit 2      **COMSEGS18**: SCOM18/SSEG18 pin function selection  
0: SCOM18  
1: SSEG18
- Bit 1      **COMSEGS17**: SCOM17/SSEG17 pin function selection  
0: SCOM17  
1: SSEG17
- Bit 0      **COMSEGS16**: SCOM16/SSEG16 pin function selection  
0: SCOM16  
1: SSEG16

• **SLCDS3 Register**

| Bit  | 7 | 6 | 5         | 4         | 3         | 2         | 1         | 0         |
|------|---|---|-----------|-----------|-----------|-----------|-----------|-----------|
| Name | — | — | COMSEGS29 | COMSEGS28 | COMSEGS27 | COMSEGS26 | COMSEGS25 | COMSEGS24 |
| 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      **COMSEGS29**: SCOM29/SSEG29 pin function selection  
0: SCOM29  
1: SSEG29
- Bit 4      **COMSEGS28**: SCOM28/SSEG28 pin function selection  
0: SCOM28  
1: SSEG28
- Bit 3      **COMSEGS27**: SCOM27/SSEG27 pin function selection  
0: SCOM27  
1: SSEG27
- Bit 2      **COMSEGS26**: SCOM26/SSEG26 pin function selection  
0: SCOM26  
1: SSEG26
- Bit 1      **COMSEGS25**: SCOM25/SSEG25 pin function selection  
0: SCOM25  
1: SSEG25

Bit 0      **COMSEGS24**: SCOM24/SSEG24 pin function selection  
 0: SCOM24  
 1: SSEG24

## Low Voltage Detector – LVD

The device has a Low Voltage Detector function, also known as LVD. This enables 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 | — | VLVD2 | VLVD1 | VLVD0 |
| R/W  | — | — | R    | R/W   | — | R/W   | R/W   | R/W   |
| POR  | — | — | 0    | 0     | — | 0     | 0     | 0     |

Bit 7~6      Unimplemented, read as “0”

Bit 5      **LVDO**: LVD output flag  
 0: No Low Voltage detected  
 1: Low Voltage detected

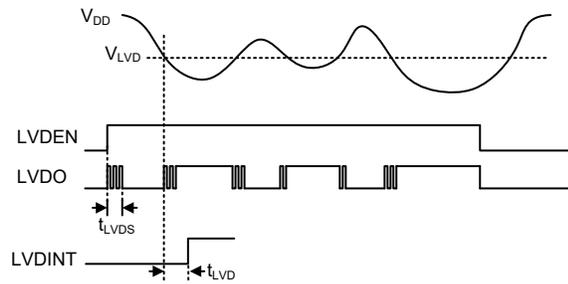
Bit 4      **LVDEN**: Low Voltage Detector Enable control  
 0: Disable  
 1: Enable

Bit 3      Unimplemented, read as “0”

Bit 2~0      **VLVD2~VLVD0**: LVD Voltage selection  
 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. When the device is in the SLEEP mode, the low voltage detector will be disabled 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.



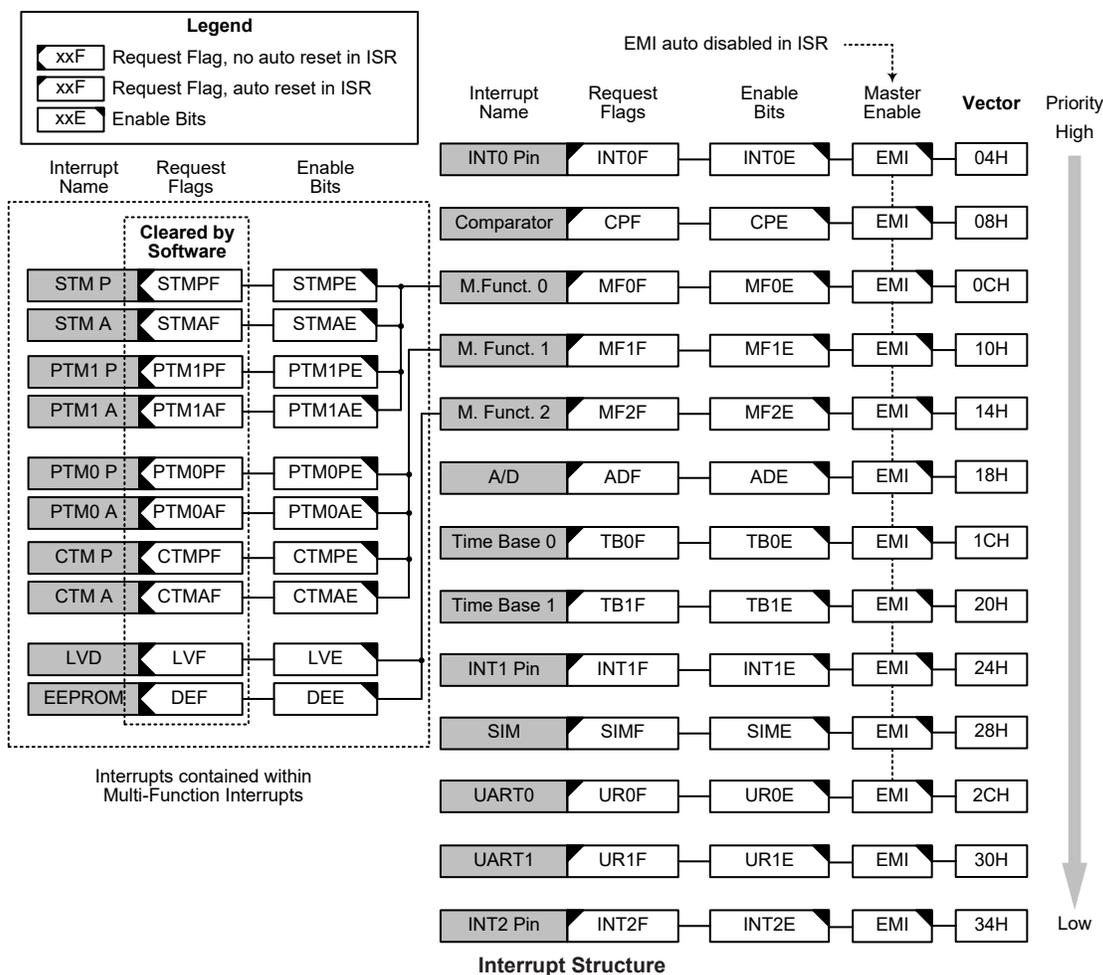
**LVD Operation**

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, i.e.,  $V_{DD}$  falls below the preset LVD voltage. In this case, the LVF interrupt request flag will be set, causing an interrupt to be generated. This will cause the device to wake up from the 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 IDLE Mode.

### Interrupts

Interrupts are an important part of any microcontroller system. When an external event or an internal function such as a Timer Module or an A/D converter requires microcontroller attention, their corresponding interrupt will enforce a temporary suspension of the main program allowing the microcontroller to direct attention to their respective needs. The device contains several external interrupt and internal interrupt functions. The external interrupts are generated by the action of the external INT0, INT1 and INT2 pins, while the internal interrupts are generated by various internal functions such as TMs, Time Base, LVD, EEPROM, SIM, UART and the A/D converter, etc.

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.



### Interrupt Registers

Overall interrupt control, which basically means the setting of request flags when certain microcontroller conditions occur and the setting of interrupt enable bits by the application program, is controlled by a series of registers, located in the Special Purpose Data Memory. The registers fall into three categories. The first is the INTC0~INTC3 registers which setup the primary interrupts, the second is the MFI0~MFI2 registers which setup the Multi-function interrupts. Finally there is an INTEG register which setup the external interrupt trigger edge type.

Each register contains a number of enable bits to enable or disable individual registers as well as interrupt flags to indicate the presence of an interrupt request. The naming convention of these follows a specific pattern. First is listed an abbreviated interrupt type, then the (optional) number of that interrupt followed by either an “E” for enable/disable bit or “F” for request flag.

| Function       | Enable Bit | Request Flag | Notes |
|----------------|------------|--------------|-------|
| Global         | EMI        | —            | —     |
| INTn Pins      | INTnE      | INTnF        | n=0~2 |
| Comparator     | CPE        | CPF          | —     |
| Multi-function | MFnE       | MFnF         | n=0~2 |
| A/D Converter  | ADE        | ADF          | —     |

| Function   | Enable Bit | Request Flag | Notes |
|------------|------------|--------------|-------|
| Time Bases | TBnE       | TBnF         | n=0~1 |
| SIM        | SIME       | SIMF         | —     |
| UART       | URnE       | URnF         | n=0~1 |
| LVD        | LVE        | LVF          | —     |
| EEPROM     | DEE        | DEF          | —     |
| CTM        | CTMPE      | CTMPF        | —     |
|            | CTMAE      | CTMAF        | —     |
| STM        | STMPE      | STMPF        | —     |
|            | STMAE      | STMAF        | —     |
| PTM        | PTMnPE     | PTMnPF       | n=0~1 |
|            | PTMnAE     | PTMnAF       |       |

**Interrupt Register Bit Naming Conventions**

| Register Name | Bit    |        |        |        |        |        |        |        |
|---------------|--------|--------|--------|--------|--------|--------|--------|--------|
|               | 7      | 6      | 5      | 4      | 3      | 2      | 1      | 0      |
| INTEG         | —      | —      | INT2S1 | INT2S0 | INT1S1 | INT1S0 | INT0S1 | INT0S0 |
| INTC0         | —      | MF0F   | CPF    | INT0F  | MF0E   | CPE    | INT0E  | EMI    |
| INTC1         | TB0F   | ADF    | MF2F   | MF1F   | TB0E   | ADE    | MF2E   | MF1E   |
| INTC2         | UR0F   | SIMF   | INT1F  | TB1F   | UR0E   | SIME   | INT1E  | TB1E   |
| INTC3         | —      | —      | INT2F  | UR1F   | —      | —      | INT2E  | UR1E   |
| MF10          | PTM1AF | PTM1PF | STMAF  | STMPF  | PTM1AE | PTM1PE | STMAE  | STMPE  |
| MF11          | CTMAF  | CTMPF  | PTM0AF | PTM0PF | CTMAE  | CTMPE  | PTM0AE | PTM0PE |
| MF12          | —      | —      | DEF    | LVF    | —      | —      | DEE    | LVE    |

**Interrupt Register List**

• **INTEG Register**

| Bit  | 7 | 6 | 5      | 4      | 3      | 2      | 1      | 0      |
|------|---|---|--------|--------|--------|--------|--------|--------|
| Name | — | — | INT2S1 | INT2S0 | INT1S1 | INT1S0 | INT0S1 | INT0S0 |
| 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 **INT2S1~INT2S0**: Interrupt edge control for INT2 pin
  - 00: Disable
  - 01: Rising edge
  - 10: Falling edge
  - 11: Rising and falling edges
- Bit 3~2 **INT1S1~INT1S0**: Interrupt edge control for INT1 pin
  - 00: Disable
  - 01: Rising edge
  - 10: Falling edge
  - 11: Rising and falling edges
- Bit 1~0 **INT0S1~INT0S0**: Interrupt edge control for INT0 pin
  - 00: Disable
  - 01: Rising edge
  - 10: Falling edge
  - 11: Rising and falling edges

• **INTC0 Register**

| Bit  | 7 | 6    | 5   | 4     | 3    | 2   | 1     | 0   |
|------|---|------|-----|-------|------|-----|-------|-----|
| Name | — | 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 | UR0F | SIMF | INT1F | TB1F | UR0E | 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      **UR0F**: UART0 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 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      **UR0E**: UART0 interrupt control  
0: Disable  
1: Enable
- Bit 2      **SIME**: SIM interrupt control  
0: Disable  
1: Enable
- Bit 1      **INT1E**: INT1 interrupt control  
0: Disable  
1: Enable
- Bit 0      **TB1E**: Time Base 1 interrupt control  
0: Disable  
1: Enable

• **INTC3 Register**

| Bit  | 7 | 6 | 5     | 4    | 3 | 2 | 1     | 0    |
|------|---|---|-------|------|---|---|-------|------|
| Name | — | — | INT2F | UR1F | — | — | INT2E | UR1E |
| R/W  | — | — | R/W   | R/W  | — | — | R/W   | R/W  |
| POR  | — | — | 0     | 0    | — | — | 0     | 0    |

- Bit 7~6      Unimplemented, read as “0”
- Bit 5      **INT2F**: INT2 interrupt request flag  
0: No request  
1: Interrupt request
- Bit 4      **UR1F**: UART1 interrupt request flag  
0: No request  
1: Interrupt request
- Bit 3~2      Unimplemented, read as “0”

- Bit 1      **INT2E**: INT2 interrupt control  
             0: Disable  
             1: Enable
- Bit 0      **UR1E**: UART1 interrupt control  
             0: Disable  
             1: Enable

• **MFIO Register**

| Bit  | 7      | 6      | 5     | 4     | 3      | 2      | 1     | 0     |
|------|--------|--------|-------|-------|--------|--------|-------|-------|
| Name | PTM1AF | PTM1PF | STMAF | STMPF | PTM1AE | PTM1PE | STMAE | STMPE |
| R/W  | R/W    | R/W    | R/W   | R/W   | R/W    | R/W    | R/W   | R/W   |
| POR  | 0      | 0      | 0     | 0     | 0      | 0      | 0     | 0     |

- Bit 7      **PTM1AF**: PTM1 Comparator A match interrupt request flag  
             0: No request  
             1: Interrupt request  
             Note that this bit must be cleared to zero by the application program when the interrupt is serviced.
- Bit 6      **PTM1PF**: PTM1 Comparator P match interrupt request flag  
             0: No request  
             1: Interrupt request  
             Note that this bit must be cleared to zero by the application program when the interrupt is serviced.
- Bit 5      **STMAF**: STM Comparator A match interrupt request flag  
             0: No request  
             1: Interrupt request  
             Note that this bit must be cleared to zero by the application program when the interrupt is serviced.
- Bit 4      **STMPF**: STM Comparator P match interrupt request flag  
             0: No request  
             1: Interrupt request  
             Note that this bit must be cleared to zero by the application program when the interrupt is serviced.
- Bit 3      **PTM1AE**: PTM1 Comparator A match interrupt control  
             0: Disable  
             1: Enable
- Bit 2      **PTM1PE**: PTM1 Comparator P match interrupt control  
             0: Disable  
             1: Enable
- Bit 1      **STMAE**: STM Comparator A match interrupt control  
             0: Disable  
             1: Enable
- Bit 0      **STMPE**: STM Comparator P match interrupt control  
             0: Disable  
             1: Enable

• **MF1 Register**

| Bit  | 7     | 6     | 5      | 4      | 3     | 2     | 1      | 0      |
|------|-------|-------|--------|--------|-------|-------|--------|--------|
| Name | CTMAF | CTMPF | PTM0AF | PTM0PF | CTMAE | CTMPE | PTM0AE | PTM0PE |
| 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  
Note that this bit must be cleared to zero by the application program when the interrupt is serviced.
- Bit 6      **CTMPF**: CTM Comparator P match interrupt request flag  
0: No request  
1: Interrupt request  
Note that this bit must be cleared to zero by the application program when the interrupt is serviced.
- Bit 5      **PTM0AF**: PTM0 Comparator A match interrupt request flag  
0: No request  
1: Interrupt request  
Note that this bit must be cleared to zero by the application program when the interrupt is serviced.
- Bit 4      **PTM0PF**: PTM0 Comparator P match interrupt request flag  
0: No request  
1: Interrupt request  
Note that this bit must be cleared to zero by the application program when the interrupt is serviced.
- 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      **PTM0AE**: PTM0 Comparator A match interrupt control  
0: Disable  
1: Enable
- Bit 0      **PTM0PE**: PTM0 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  
Note that this bit must be cleared to zero by the application program when the interrupt is serviced.

|         |   |
|---------|---|
| Bit 4   | <b>LVF</b> : LVD interrupt request flag<br>0: No request<br>1: Interrupt request<br>Note that this bit must be cleared to zero by the application program when the interrupt is serviced. |
| Bit 3~2 | Unimplemented, read as “0”  |
| Bit 1   | <b>DEE</b> : Data EEPROM interrupt control<br>0: Disable<br>1: Enable   |
| Bit 0   | <b>LVE</b> : LVD interrupt control<br>0: Disable<br>1: Enable   |

### Interrupt Operation

When the conditions for an interrupt event occur, such as a TM Comparator P or 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.

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 interrupt structure 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 Interrupts

The external interrupts are controlled by signal transitions on the pins INT0~INT2. An external interrupt request will take place when the external interrupt request flags, INT0F~INT2F, 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 their respective interrupt vector addresses, the global interrupt enable bit, EMI, and respective external interrupt enable bit, INT0E~INT2E, 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~INT2F, 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 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 Interrupts

Within the device there are 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 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 flag will be set when any of their included functions generate an interrupt request flag. To allow the program to branch to their respective interrupt vector addresses, when the Multi-function interrupt is enabled and the stack is not full, and any one of the interrupts contained within each of the Multi-function interrupt occurs, a subroutine call to the related Multi-function interrupt vector 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 interrupt will not be automatically reset and must be manually reset by the application program.

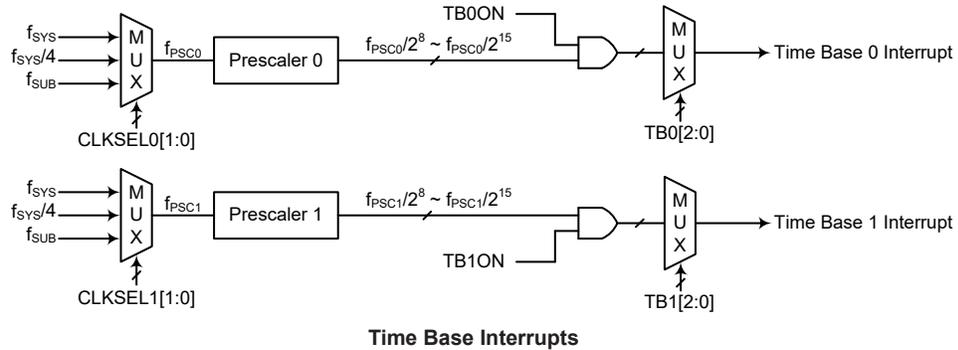
### A/D Converter Interrupt

The A/D Converter Interrupt is controlled by the termination of an A/D conversion process. An A/D Converter Interrupt request will take place when the A/D Converter Interrupt request flag, ADF, is set, which occurs when the A/D conversion process finishes. To allow the program to branch to its interrupt vector address, the global interrupt enable bit, EMI, and A/D Interrupt enable bit, ADE, must first be set. When the interrupt is enabled, the stack is not full and the A/D conversion process has ended, a subroutine call to the A/D Converter Interrupt vector, will take place. When the interrupt is serviced, the A/D Converter Interrupt flag, ADF, will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts.

### Time Base Interrupts

The function of the Time Base Interrupt is to provide regular time signal in the form of an internal interrupt. They are controlled by the overflow signals from their respective timer functions. When these happen their respective interrupt request flags, TB0F or TB1F, will be set. To allow the program to branch to their respective interrupt vector addresses, the global interrupt enable bit, EMI and Time Base enable bit, TB0E or TB1E, must first be set. When the interrupt is enabled, the stack is not full and the Time Base overflows, a subroutine call to their respective vector locations will take place. When the interrupt is serviced, the respective interrupt request flag, TB0F or TB1F, will be automatically reset and the EMI bit will be cleared to disable other interrupts.

The purpose of the Time Base Interrupt is to provide an interrupt signal at fixed time periods. Their clock sources  $f_{PSC0}$  or  $f_{PSC1}$ , originate from the internal clock source  $f_{SYS}$ ,  $f_{SYS}/4$  or  $f_{SUB}$  and then passes through a divider, the division ratio of which is selected by programming the appropriate bits in the TB0C or TB1C register to obtain longer interrupt periods whose value ranges. The clock source 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.



#### • 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:  $2^{15}/f_{PSC1}$

## **TM Interrupts**

The Compact, Standard and Periodic TMs each 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 their respective interrupt vector addresses, 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 location, 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.

## **Serial Interface Module Interrupt**

The Serial Interface Module Interrupt, also known as the SIM interrupt. An 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 matches or I<sup>2</sup>C bus time-out occurs. To allow the program to branch to its 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, a subroutine call to the SIM Interrupt vector will take place. When the Serial Interface Interrupt is serviced, the SIM Interrupt flag, SIMF, will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts.

## **UART Interrupts**

Each UART 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 reaching FIFO trigger level, receiver overrun, address detect and an RXn/TXn pin wake-up. To allow the program to branch to their corresponding interrupt vector addresses, the global interrupt enable bit, EMI, and UARTn Interrupt enable bit, URnE, 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 UARTn Interrupt vector, will take place. When the interrupt is serviced, the UARTn Interrupt flag, URnF, will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts. However, the UnSR register flags will only be cleared when certain actions are taken by the UARTn, the details of which are given in the UART Interfaces section.

## **LVD Interrupt**

The Low Voltage Detector Interrupt is contained within the Multi-function Interrupt. An 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 interrupt vector address, the global interrupt enable bit, EMI, Low Voltage Interrupt enable bit, LVE, and associated Multi-function interrupt enable bit, MFnE, 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.

### **EEPROM Interrupt**

The EEPROM Interrupt is contained within the Multi-function Interrupt. An EEPROM Interrupt request will take place when the EEPROM Interrupt request flag, DEF, is set, which occurs when an EEPROM Erase or Write cycle ends. To allow the program to branch to its interrupt vector address, the global interrupt enable bit, EMI, EEPROM Interrupt enable bit, DEE, and associated Multi-function interrupt enable bit, MFnE, must first be set. When the interrupt is enabled, the stack is not full and an EEPROM Erase or Write cycle ends, a subroutine call to the respective Multi-function Interrupt vector will take place. When the EEPROM 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.

### **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, MFnF, 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.

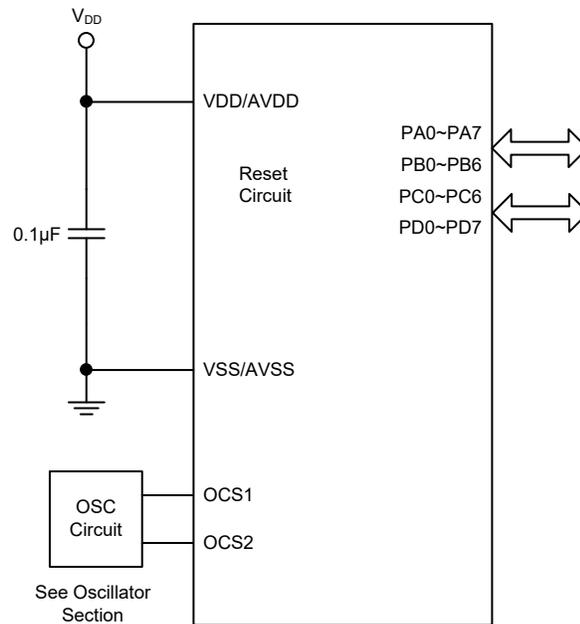
## Configuration Options

Configuration options refer to certain options within the MCU that are programmed into the device during the programming process. During the development process, these options are selected using the HT-IDE software development tools. The option must be defined for proper system function, the details of which are shown in the table.

| No.                      | Options   |
|--------------------------|---|
| <b>Oscillator Option</b> |   |
| 1                        | HIRC frequency selection – $f_{HIRC}$ :<br>8MHz, 12MHz or 16MHz |

Note: When the HIRC has been configured at a frequency shown in this table, the HIRC1 and HIRC0 bits should also be set to select the same frequency to achieve the HIRC frequency accuracy specified in the A.C. Characteristics.

## 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 several 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 such as 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 setup 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 (specific page) 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 two cycles are required, if no skip takes place only one cycle is required.

2. Any instruction which changes the contents of the PCL will also require 2 cycles for execution.

## 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 sector except sector 0, the extended instruction can be used to directly access the data memory instead of using the indirect addressing access. This can not only reduce the use of Flash memory space but also improve the CPU execution efficiency.

| 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 (specific page) 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 (specific page) 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 three cycles are required, if no skip takes place two cycles is required.
2. 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        | $ACC \leftarrow ACC \text{ "OR" } [m]$   |
| 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        | $ACC \leftarrow ACC \text{ "OR" } x$   |
| 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        | $[m] \leftarrow ACC \text{ "OR" } [m]$   |
| 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 $\leftarrow$ 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 $\leftarrow$ Stack<br>$ACC \leftarrow x$   |
| 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 $\leftarrow$ Stack<br>$EMI \leftarrow 1$   |
| 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        | $[m].(i+1) \leftarrow [m].i; (i=0\sim 6)$<br>$[m].0 \leftarrow [m].7$  |
| 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) ← [m].i; (i=0~6)<br>ACC.0 ← [m].7   |
| Affected flag(s) | None  |
| <br>             |   |
| <b>RLC [m]</b>   | Rotate Data Memory left through Carry   |
| Description      | The contents of the specified Data Memory and the carry flag are rotated left by 1 bit. Bit 7 replaces the Carry bit and the original carry flag is rotated into bit 0.   |
| Operation        | [m].(i+1) ← [m].i; (i=0~6)<br>[m].0 ← C<br>C ← [m].7  |
| Affected flag(s) | C   |
| <br>             |   |
| <b>RLCA [m]</b>  | Rotate Data Memory left through Carry with result in ACC  |
| Description      | Data in the specified Data Memory and the carry flag are rotated left by 1 bit. Bit 7 replaces the Carry bit and the original carry flag is rotated into the bit 0. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged. |
| Operation        | ACC.(i+1) ← [m].i; (i=0~6)<br>ACC.0 ← C<br>C ← [m].7  |
| Affected flag(s) | C   |
| <br>             |   |
| <b>RR [m]</b>    | Rotate Data Memory right  |
| Description      | The contents of the specified Data Memory are rotated right by 1 bit with bit 0 rotated into bit 7.   |
| Operation        | [m].i ← [m].(i+1); (i=0~6)<br>[m].7 ← [m].0   |
| Affected flag(s) | None  |
| <br>             |   |
| <b>RRA [m]</b>   | Rotate Data Memory right with result in ACC   |
| Description      | Data in the specified Data Memory is rotated right by 1 bit with bit 0 rotated into bit 7. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.  |
| Operation        | ACC.i ← [m].(i+1); (i=0~6)<br>ACC.7 ← [m].0   |
| Affected flag(s) | None  |
| <br>             |   |
| <b>RRC [m]</b>   | Rotate Data Memory right through Carry  |
| Description      | The contents of the specified Data Memory and the carry flag are rotated right by 1 bit. Bit 0 replaces the Carry bit and the original carry flag is rotated into bit 7.  |
| Operation        | [m].i ← [m].(i+1); (i=0~6)<br>[m].7 ← C<br>C ← [m].0  |
| Affected flag(s) | C   |

|                   |   |
|-------------------|---|
| <b>RRCA [m]</b>   | Rotate Data Memory right through Carry with result in ACC   |
| Description       | Data in the specified Data Memory and the carry flag are rotated right by 1 bit. Bit 0 replaces the Carry bit and the original carry flag is rotated into bit 7. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.  |
| Operation         | ACC.i ← [m].(i+1); (i=0~6)<br>ACC.7 ← C<br>C ← [m].0  |
| Affected flag(s)  | C   |
| <b>SBC A,[m]</b>  | Subtract Data Memory from ACC with Carry  |
| Description       | The contents of the specified Data Memory and the complement of the carry flag are subtracted from the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.   |
| Operation         | ACC ← ACC – [m] – $\bar{C}$   |
| Affected flag(s)  | OV, Z, AC, C, 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] – $\bar{C}$   |
| 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 bit i of Data Memory is not 0  |
| Description      | If bit i of the specified Data Memory is not 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is 0 the program proceeds with the following instruction.  |
| Operation        | Skip if $[m].i \neq 0$   |
| Affected flag(s) | None   |
| <br>             |  |
| <b>SNZ [m]</b>   | Skip if Data Memory is not 0   |
| Description      | The contents of the specified Data Memory are read out and then written back to the specified Data Memory again. 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  |
| <br>              |   |
| <b>SUB A,x</b>    | Subtract immediate data from ACC  |
| Description       | The immediate data specified by the code is subtracted from the contents of the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.  |
| Operation         | $ACC \leftarrow ACC - x$  |
| Affected flag(s)  | OV, Z, AC, C, SC, CZ  |
| <br>              |   |
| <b>SWAP [m]</b>   | Swap nibbles of Data Memory   |
| Description       | The low-order and high-order nibbles of the specified Data Memory are interchanged.   |
| Operation         | $[m].3 \sim [m].0 \leftrightarrow [m].7 \sim [m].4$   |
| Affected flag(s)  | None  |
| <br>              |   |
| <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  |
| <br>              |   |
| <b>SZ [m]</b>     | Skip if Data Memory is 0  |
| Description       | The contents of the specified Data Memory are read out and then written back to the specified Data Memory again. If the contents of the specified Data Memory is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction. |
| Operation         | Skip if $[m]=0$   |
| Affected flag(s)  | None  |
| <br>              |   |
| <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  |
| <br>              |   |
| <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 (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   |
|                    |  |
| <b>TABRDL [m]</b>  | Read table (last page) to TBLH and Data Memory   |
| Description        | The low byte of the program code (last page) addressed by the table pointer (TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.  |
| Operation          | [m] ← program code (low byte)<br>TBLH ← program code (high byte)   |
| Affected flag(s)   | None   |
|                    |  |
| <b>ITABRD [m]</b>  | Increment table pointer low byte first and read table (specific page) to TBLH and Data Memory  |
| Description        | Increment table pointer low byte, TBLP, first and then the program code (specific page) 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   |
|                    |  |
| <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   |
|                    |  |
| <b>XOR A,[m]</b>   | Logical XOR Data Memory to ACC   |
| Description        | Data in the Accumulator and the specified Data Memory perform a bitwise logical XOR operation. The result is stored in the Accumulator.  |
| Operation          | ACC ← ACC "XOR" [m]  |
| Affected flag(s)   | Z  |
|                    |  |
| <b>XORM A,[m]</b>  | Logical XOR ACC to Data Memory   |
| Description        | Data in the specified Data Memory and the Accumulator perform a bitwise logical XOR operation. The result is stored in the Data Memory.  |
| Operation          | [m] ← ACC "XOR" [m]  |
| Affected flag(s)   | Z  |
|                    |  |
| <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          | $[m].i \leftarrow [m].(i+1); (i=0\sim 6)$<br>$[m].7 \leftarrow [m].0$   |
| Affected flag(s)   | None  |
|                    |   |
| <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          | $ACC.i \leftarrow [m].(i+1); (i=0\sim 6)$<br>$ACC.7 \leftarrow [m].0$   |
| Affected flag(s)   | None  |
|                    |   |
| <b>LRRRC [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          | $[m].i \leftarrow [m].(i+1); (i=0\sim 6)$<br>$[m].7 \leftarrow C$<br>$C \leftarrow [m].0$   |
| Affected flag(s)   | C   |
|                    |   |
| <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          | $ACC.i \leftarrow [m].(i+1); (i=0\sim 6)$<br>$ACC.7 \leftarrow C$<br>$C \leftarrow [m].0$   |
| Affected flag(s)   | C   |
|                    |   |
| <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          | $ACC \leftarrow ACC - [m] - \bar{C}$  |
| Affected flag(s)   | OV, Z, AC, C, SC, CZ  |
|                    |   |
| <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          | $[m] \leftarrow ACC - [m] - \bar{C}$  |
| 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 three 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 three 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 three 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 three 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 bit i of Data Memory is not 0   |
| Description       | If bit i of the specified Data Memory is not 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a three 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        | The contents of the specified Data Memory are read out and then written to the specified Data Memory again. 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 three 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        | The contents of the specified Data Memory are read out and then written to the specified Data Memory again. If the contents of the specified Data Memory is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a three 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 three 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 three 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 (specific page) to TBLH and Data Memory   |
| Description         | The low byte of the program code (specific page) 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>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 (specific page) to TBLH and Data Memory  |
| Description         | Increment table pointer low byte, TBLP, first and then the program code (specific page) 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>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  |

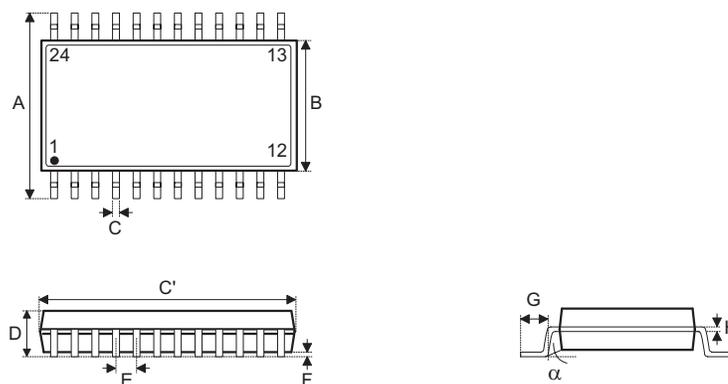
## Package Information

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 Information](#).

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

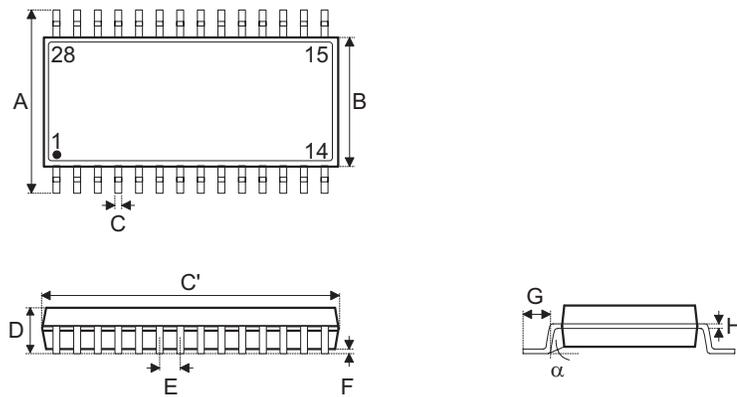
- [Package Information \(include Outline Dimensions, Product Tape and Reel Specifications\)](#)
- [The Operation Instruction of Packing Materials](#)
- [Carton information](#)

24-pin SSOP (150mil) Outline Dimensions



| Symbol   | Dimensions in inch |      |       |
|----------|--------------------|------|-------|
|          | Min.               | Nom. | Max.  |
| A        | 0.236 BSC          |      |       |
| B        | 0.154 BSC          |      |       |
| C        | 0.008              | —    | 0.012 |
| C'       | 0.341 BSC          |      |       |
| D        | —                  | —    | 0.069 |
| E        | 0.025 BSC          |      |       |
| F        | 0.004              | —    | 0.010 |
| G        | 0.016              | —    | 0.050 |
| H        | 0.004              | —    | 0.010 |
| $\alpha$ | 0°                 | —    | 8°    |

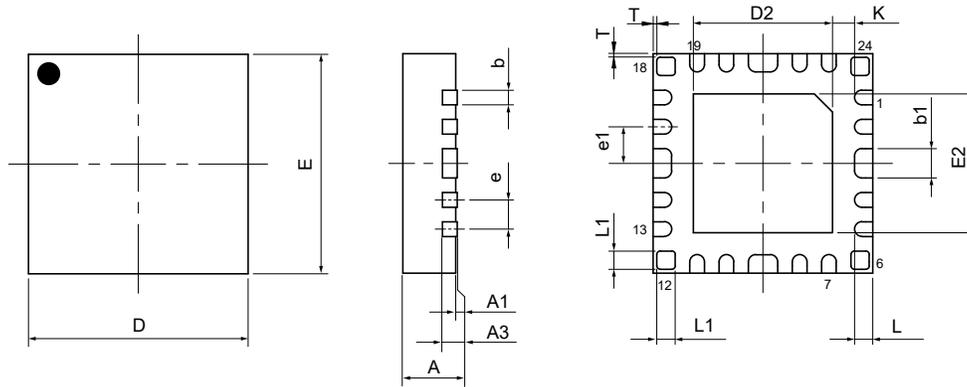
| Symbol   | Dimensions in mm |      |      |
|----------|------------------|------|------|
|          | Min.             | Nom. | Max. |
| A        | 6.00 BSC         |      |      |
| B        | 3.90 BSC         |      |      |
| C        | 0.20             | —    | 0.30 |
| C'       | 8.66 BSC         |      |      |
| D        | —                | —    | 1.75 |
| E        | 0.635 BSC        |      |      |
| F        | 0.10             | —    | 0.25 |
| G        | 0.41             | —    | 1.27 |
| H        | 0.10             | —    | 0.25 |
| $\alpha$ | 0°               | —    | 8°   |

**28-pin SSOP (150mil) Outline Dimensions**


| Symbol   | Dimensions in inch |      |       |
|----------|--------------------|------|-------|
|          | Min.               | Nom. | Max.  |
| A        | 0.236 BSC          |      |       |
| B        | 0.154 BSC          |      |       |
| C        | 0.008              | —    | 0.012 |
| C'       | 0.390 BSC          |      |       |
| D        | —                  | —    | 0.069 |
| E        | 0.025 BSC          |      |       |
| F        | 0.004              | —    | 0.010 |
| G        | 0.016              | —    | 0.050 |
| H        | 0.004              | —    | 0.010 |
| $\alpha$ | 0°                 | —    | 8°    |

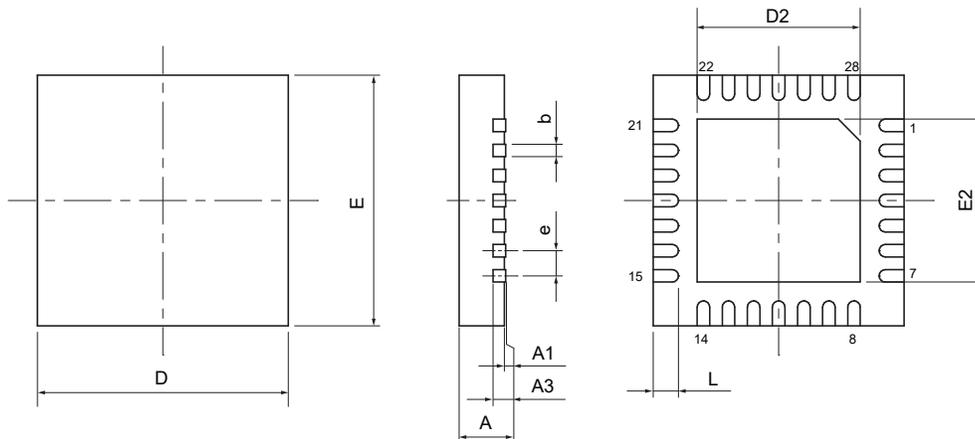
| Symbol   | Dimensions in mm |      |      |
|----------|------------------|------|------|
|          | Min.             | Nom. | Max. |
| A        | 6.00 BSC         |      |      |
| B        | 3.90 BSC         |      |      |
| C        | 0.20             | —    | 0.30 |
| C'       | 9.90 BSC         |      |      |
| D        | —                | —    | 1.75 |
| E        | 0.635 BSC        |      |      |
| F        | 0.10             | —    | 0.25 |
| G        | 0.41             | —    | 1.27 |
| H        | 0.10             | —    | 0.25 |
| $\alpha$ | 0°               | —    | 8°   |

SAW Type 24-pin QFN (3mm×3mm×0.55mm) Outline Dimensions



| Symbol | Dimensions in inch |       |       |
|--------|--------------------|-------|-------|
|        | Min.               | Nom.  | Max.  |
| A      | 0.020              | 0.022 | 0.024 |
| A1     | 0.000              | 0.001 | 0.002 |
| A3     | 0.006 REF          |       |       |
| b      | 0.006              | 0.008 | 0.010 |
| b1     | 0.014              | 0.016 | 0.018 |
| D      | 0.118 BSC          |       |       |
| E      | 0.118 BSC          |       |       |
| e      | 0.016 BSC          |       |       |
| e1     | 0.020 BSC          |       |       |
| D2     | 0.073              | —     | 0.077 |
| E2     | 0.073              | —     | 0.077 |
| L      | 0.006              | 0.010 | 0.014 |
| L1     | 0.008              | 0.010 | 0.012 |
| K      | 0.008              | —     | —     |
| T      | 0.000              | 0.002 | 0.004 |

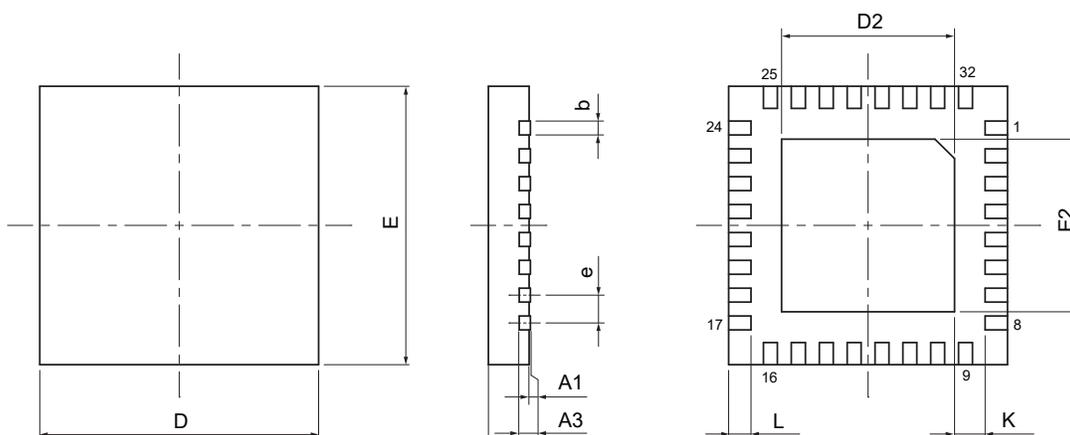
| Symbol | Dimensions in mm |      |      |
|--------|------------------|------|------|
|        | Min.             | Nom. | Max. |
| A      | 0.50             | 0.55 | 0.60 |
| A1     | 0.00             | 0.02 | 0.05 |
| A3     | 0.15 REF         |      |      |
| b      | 0.15             | 0.20 | 0.25 |
| b1     | 0.35             | 0.40 | 0.45 |
| D      | 3.00 BSC         |      |      |
| E      | 3.00 BSC         |      |      |
| e      | 0.40 BSC         |      |      |
| e1     | 0.50 BSC         |      |      |
| D2     | 1.85             | —    | 1.95 |
| E2     | 1.85             | —    | 1.95 |
| L      | 0.15             | 0.25 | 0.35 |
| L1     | 0.20             | 0.25 | 0.30 |
| K      | 0.20             | —    | —    |
| T      | 0.00             | 0.05 | 0.10 |

**SAW Type 28-pin QFN (4mm×4mm×0.75mm) Outline Dimensions**


| Symbol | Dimensions in inch |       |       |
|--------|--------------------|-------|-------|
|        | Min.               | Nom.  | Max.  |
| A      | 0.028              | 0.030 | 0.031 |
| A1     | 0.000              | 0.001 | 0.002 |
| A3     | 0.008 REF          |       |       |
| b      | 0.006              | 0.008 | 0.010 |
| D      | 0.157 BSC          |       |       |
| E      | 0.157 BSC          |       |       |
| e      | 0.016 BSC          |       |       |
| D2     | 0.091              | —     | 0.104 |
| E2     | 0.091              | —     | 0.104 |
| L      | 0.012              | 0.016 | 0.020 |

| Symbol | Dimensions in mm |      |      |
|--------|------------------|------|------|
|        | Min.             | Nom. | Max. |
| A      | 0.70             | 0.75 | 0.80 |
| A1     | 0.00             | 0.02 | 0.05 |
| A3     | 0.203 REF        |      |      |
| b      | 0.15             | 0.20 | 0.25 |
| D      | 4.00 BSC         |      |      |
| E      | 4.00 BSC         |      |      |
| e      | 0.40 BSC         |      |      |
| D2     | 2.30             | —    | 2.65 |
| E2     | 2.30             | —    | 2.65 |
| L      | 0.30             | 0.40 | 0.50 |

**SAW Type 32-pin QFN (4mm×4mm×0.75mm) Outline Dimensions**



| Symbol | Dimensions in inch |       |       |
|--------|--------------------|-------|-------|
|        | Min.               | Nom.  | Max.  |
| A      | 0.028              | 0.030 | 0.031 |
| A1     | 0.000              | 0.001 | 0.002 |
| A3     | 0.008 REF          |       |       |
| b      | 0.006              | 0.008 | 0.010 |
| D      | 0.157 BSC          |       |       |
| E      | 0.157 BSC          |       |       |
| e      | 0.016 BSC          |       |       |
| D2     | 0.100              | —     | 0.108 |
| E2     | 0.100              | —     | 0.108 |
| L      | 0.014              | 0.016 | 0.018 |
| K      | 0.008              | —     | —     |

| Symbol | Dimensions in mm |      |      |
|--------|------------------|------|------|
|        | Min.             | Nom. | Max. |
| A      | 0.70             | 0.75 | 0.80 |
| A1     | 0.00             | 0.02 | 0.05 |
| A3     | 0.203 REF        |      |      |
| b      | 0.15             | 0.20 | 0.25 |
| D      | 4.00 BSC         |      |      |
| E      | 4.00 BSC         |      |      |
| e      | 0.40 BSC         |      |      |
| D2     | 2.55             | —    | 2.75 |
| E2     | 2.55             | —    | 2.75 |
| L      | 0.35             | 0.40 | 0.45 |
| K      | 0.20             | —    | —    |

Copyright© 2023 by HOLTEK SEMICONDUCTOR INC. All Rights Reserved.

The information provided in this document has been produced with reasonable care and attention before publication, however, HOLTEK does not guarantee that the information is completely accurate. The information contained in this publication is provided for reference only and may be superseded by updates. HOLTEK disclaims any expressed, implied or statutory warranties, including but not limited to suitability for commercialization, satisfactory quality, specifications, characteristics, functions, fitness for a particular purpose, and non-infringement of any third-party's rights. HOLTEK disclaims all liability arising from the information and its application. In addition, HOLTEK does not recommend the use of HOLTEK's products where there is a risk of personal hazard due to malfunction or other reasons. HOLTEK hereby declares that it does not authorize the use of these products in life-saving, life-sustaining or safety critical components. Any use of HOLTEK's products in life-saving/sustaining or safety applications is entirely at the buyer's risk, and the buyer agrees to defend, indemnify and hold HOLTEK harmless from any damages, claims, suits, or expenses resulting from such use. The information provided in this document, including but not limited to the content, data, examples, materials, graphs, and trademarks, is the intellectual property of HOLTEK (and its licensors, where applicable) and is protected by copyright law and other intellectual property laws. No license, express or implied, to any intellectual property right, is granted by HOLTEK herein. HOLTEK reserves the right to revise the information described in the document at any time without prior notice. For the latest information, please contact us.