



Touch A/D Flash MCU with LED Driver

BS86E20CA

Revision: V1.10 Date: June 04, 2024

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 μs instruction cycle with 16MHz system clock at $V_{DD}=5\text{V}$
- Power down and wake-up functions to reduce power consumption
- Oscillator types
 - ♦ Internal High Speed 8/12/16MHz RC – HIRC
 - ♦ Internal Low Speed 32kHz RC – LIRC
 - ♦ External Low Speed 32.768kHz Crystal – LXT
- 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
- Data Memory: 1024 \times 8
- True EEPROM Memory: 1024 \times 8
- 20 touch key functions – fully integrated without requiring external components
- Watchdog Timer function
- In Application Programming – IAP
- Up to 46 bidirectional I/O lines
- Programmable I/O source current for LED applications
- Two external interrupt lines shared with I/O pins
- Multiple Timer Modules for time measure, input capture, compare match output, PWM output function or single pulse output function
- Universal Serial Interface Module – USIM for SPI, I²C or UART communication
- Fully-duplex/Half-duplex Universal Asynchronous Receiver and Transmitter Interface – UART
- Dual Time-Base functions for generation of fixed time interrupt signals
- Software controlled 26-SCOM/SSEG lines LCD driver with 1/3 bias
- 12 external channel 12-bit resolution A/D converter with internal reference voltage V_{VR}
- Over Current Protection function – OCP
- Integrated 16-bit Cyclic Redundancy Check function – CRC
- Low voltage reset function
- Low voltage detect function
- Package types: 28-pin SOP/SSOP, 44/48-pin LQFP

Development Tools

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

https://www.holtek.com/page/detail/dev_plat/Touch_Workshop

General Description

The device is a Flash Memory type 8-bit high performance RISC architecture microcontrollers with fully integrated touch key functions. With all touch key functions provided internally, completely eliminating the need for external components, the device has all the features to offer designers a reliable and easy means of implementing touch keys within their products applications.

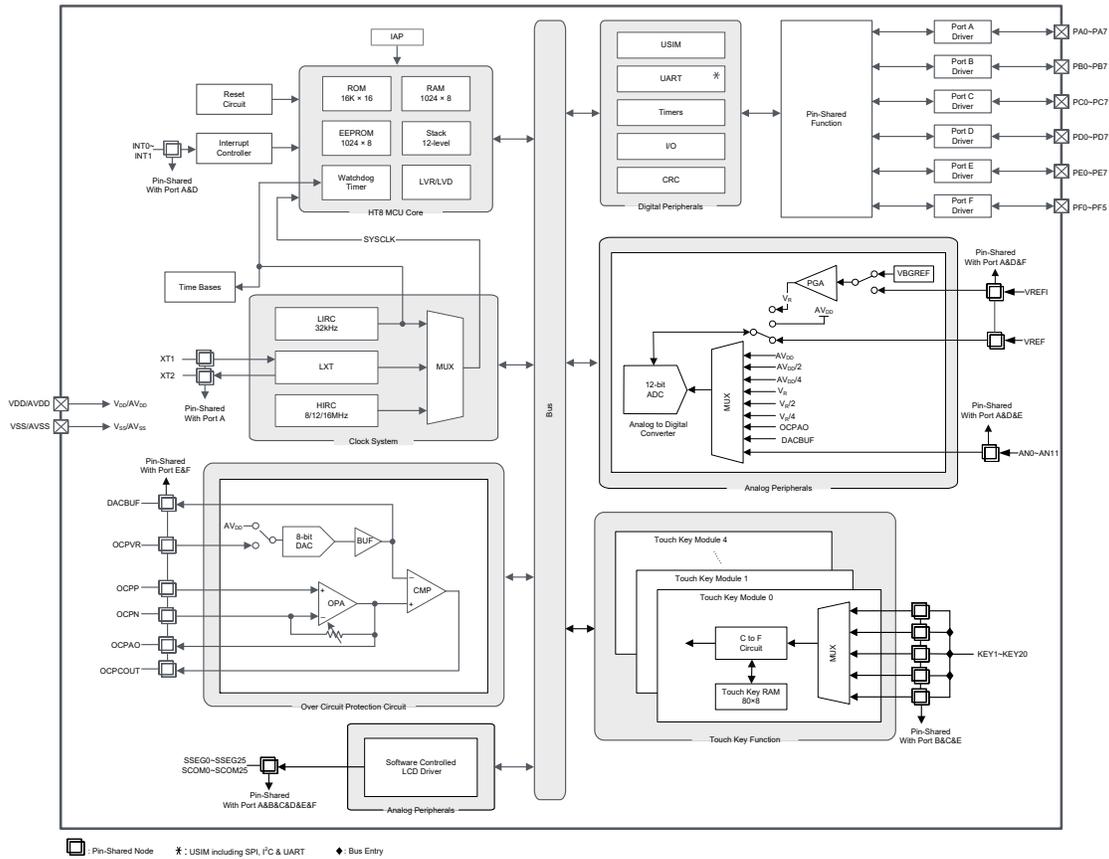
For memory features, the Flash Memory offers users the convenience of multi-programming features. Other memory includes an area of RAM Data Memory as well as an area of true EEPROM memory for storage of non-volatile data such as serial numbers, calibration data etc.

Analog features include multi-channel 12-bit A/D converter, an Over Current Protection and a software controlled LCD function. Multiple and extremely flexible Timer Modules provide timing, pulse generation and PWM generation functions. Communication with the outside world is catered for by including fully integrated SPI, I²C and UART interface functions, three types of 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 low, internal high and low oscillators is provided including two fully integrated system oscillators which require no external components for their implementation. The ability to operate and switch dynamically between a range of operating modes using different clock sources gives users the ability to optimize microcontroller operation and minimize 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 sensor signal processing, motor driving, industrial control, consumer products and subsystem controllers, etc.

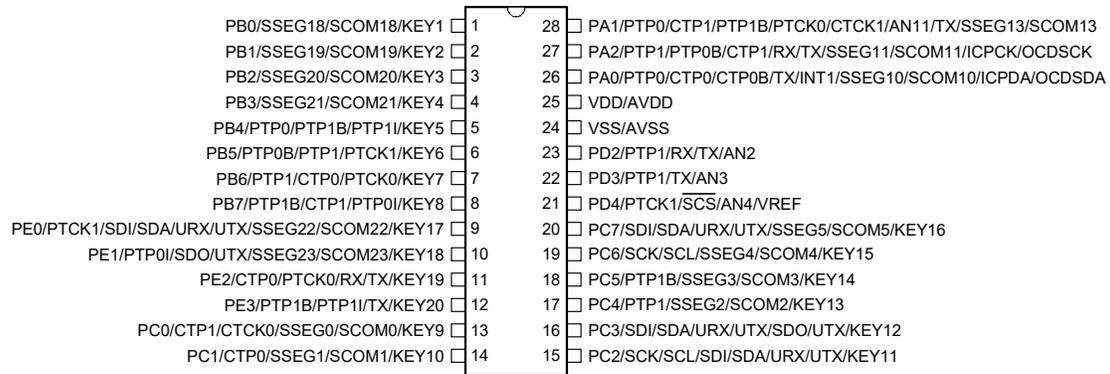
Block Diagram



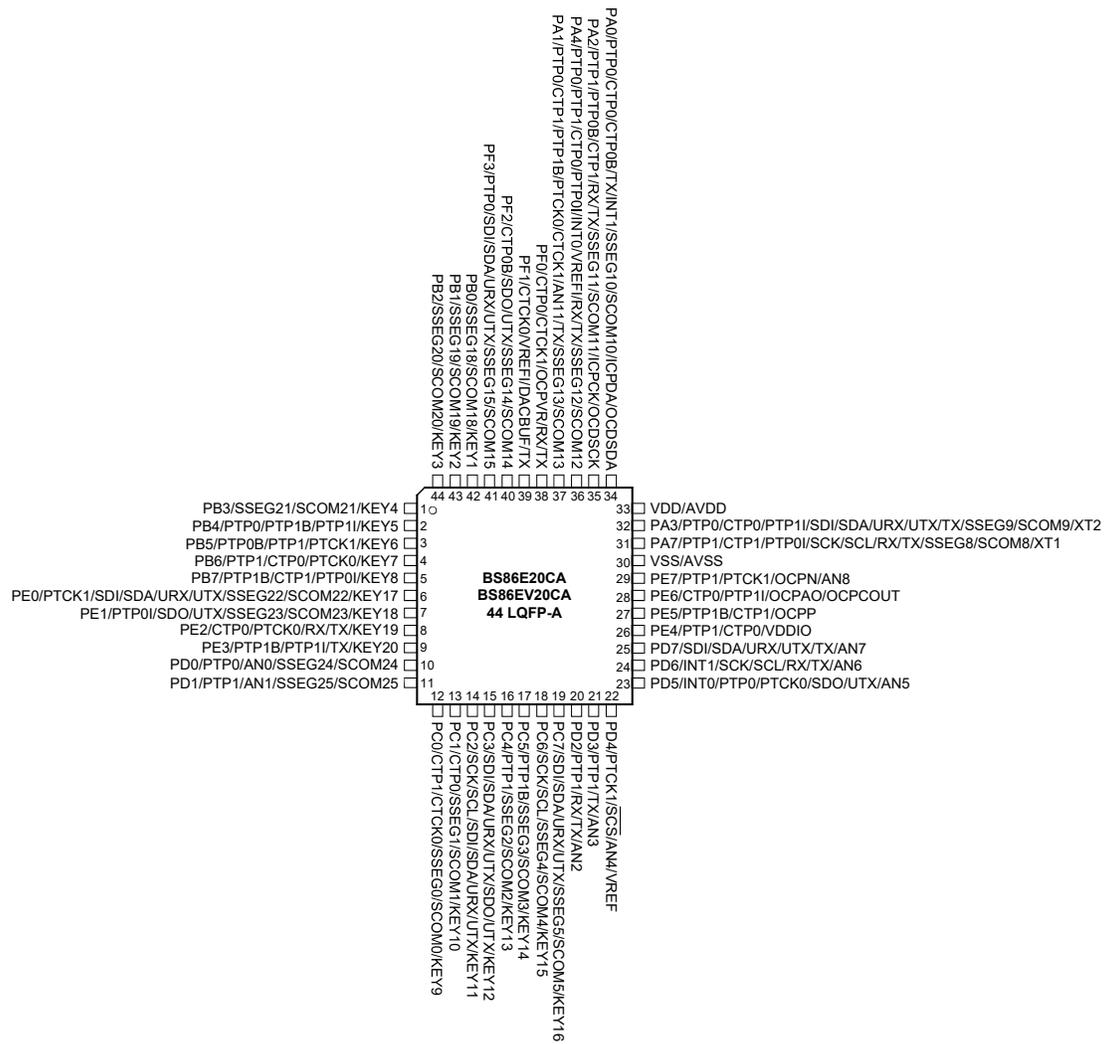
Pin Assignment

| | | | |
|-----------------------------------|----|----|---|
| PB0/SSEG18/SCOM18/KEY1 | 1 | 28 | PA1/PTP0/CTP1/PTP1B/PTCK0/CTCK1/AN11/TX/SSEG13/SCOM13 |
| PB1/SSEG19/SCOM19/KEY2 | 2 | 27 | PA4/PTP0/PTP1/CTP0/PTP0I/INT0/VREF1/RX/TX/SSEG12/SCOM12 |
| PB2/SSEG20/SCOM20/KEY3 | 3 | 26 | PA2/PTP1/PTP0B/CTP1/RX/TX/SSEG11/SCOM11/ICPK/OCDSCK |
| PB3/SSEG21/SCOM21/KEY4 | 4 | 25 | PA0/PTP0/CTP0/CTP0B/TX/INT1/SSEG10/SCOM10/ICPDA/OCDSCK |
| PB4/PTP0/PTP1B/PTP1I/KEY5 | 5 | 24 | VDD/AVDD |
| PB5/PTP0B/PTP1/PTCK1/KEY6 | 6 | 23 | PA3/PTP0/CTP0/PTP1I/SDI/SDA/URX/UTX/TX/SSEG9/SCOM9/XT2 |
| PB6/PTP1/CTP0/PTCK0/KEY7 | 7 | 22 | PA7/PTP1/CTP1/PTP0I/SCK/SCL/RX/TX/SSEG8/SCOM8/XT1 |
| PB7/PTP1B/CTP1/PTP0I/KEY8 | 8 | 21 | VSS/AVSS |
| PD0/PTP0/AN0/SSEG24/SCOM24 | 9 | 20 | PD7/SDI/SDA/URX/UTX/TX/AN7 |
| PD1/PTP1/AN1/SSEG25/SCOM25 | 10 | 19 | PD6/INT1/SCK/SCL/RX/TX/AN6 |
| PC0/CTP1/CTCK0/SSEG0/SCOM0/KEY9 | 11 | 18 | PD5/INT0/PTP0/PTCK0/SDO/UTX/AN5 |
| PC1/CTP0/SSEG1/SCOM1/KEY10 | 12 | 17 | PD4/PTCK1/SCS/AN4/VREF |
| PC2/SCK/SCL/SDI/SDA/URX/UTX/KEY11 | 13 | 16 | PD3/PTP1/TX/AN3 |
| PC3/SDI/SDA/URX/UTX/SDO/UTX/KEY12 | 14 | 15 | PD2/PTP1/RX/TX/AN2 |

BS86E20CA/BS86EV20CA
28 SOP-A/SSOP-A



BS86E20CA/BS86EV20CA
28 SOP-B/SSOP-B



BS86E20CA
BS86EV20CA
44 LQFP-A

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/PTP0/CTP0/ CTP0B/TX/INT1/ SSEG10/SCOM10/ ICPDA/OCSDA | PA0 | PAPU PAWU PAS0 | ST | CMOS | General purpose I/O. Register enabled pull-up and wake-up |
| | PTP0 | PAS0 | — | CMOS | PTM0 output |
| | CTP0 | PAS0 | — | CMOS | CTM0 output |
| | CTP0B | PAS0 | — | CMOS | CTM0 inverted output |
| | TX | PAS0 | — | CMOS | UART serial data output |
| | INT1 | PAS0 IFS1 INTEG INTC0 | ST | — | External interrupt 1 input |
| | SSEG10 | PAS0 | — | AN | Software controlled LCD segment output |
| | SCOM10 | PAS0 | — | AN | Software controlled LCD common output |
| | ICPDA | — | ST | CMOS | ICP data/address pin |
| | OCSDA | — | ST | CMOS | OCDS data/address pin, for EV chip only |
| PA1/PTP0/CTP1/ PTP1B/PTCK0/CTCK1/ AN11/TX/SSEG13/ SCOM13 | PA1 | PAWU PAPU PAS3 | ST | CMOS | General purpose I/O. Register enabled pull-up and wake-up |
| | PTP0 | PAS3 | — | CMOS | PTM0 output |
| | CTP1 | PAS3 | — | CMOS | CTM1 output |
| | PTP1B | PAS3 | — | CMOS | PTM1 inverted output |
| | PTCK0 | PAS3 IFS0 | ST | — | PTM0 clock input or capture input |
| | CTCK1 | PAS3 IFS0 | ST | — | CTM1 clock input |
| | AN11 | PAS3 | AN | — | A/D converter external input channel |
| | TX | PAS3 | — | CMOS | UART serial data output |
| | SSEG13 | PAS3 | — | AN | Software controlled LCD segment output |
| | SCOM13 | PAS3 | — | AN | Software controlled LCD common output |
| PA2/PTP1/PTP0B/ CTP1/RX/TX/SSEG11/ SCOM11/ICPCK/ OCDSCK | PA2 | PAPU PAWU PAS1 | ST | CMOS | General purpose I/O. Register enabled pull-up and wake-up |
| | PTP1 | PAS1 | — | CMOS | PTM1 output |
| | PTP0B | PAS1 | — | CMOS | PTM0 inverted output |
| | CTP1 | PAS1 | — | CMOS | CTM1 output |
| | RX/TX | PAS1 IFS1 | ST | CMOS | UART serial data input in full-duplex communication or UART serial data input/output in Single Wire Mode communication |
| | SSEG11 | PAS1 | — | AN | Software controlled LCD segment output |
| | SCOM11 | PAS1 | — | AN | Software controlled LCD common output |
| | ICPCK | — | ST | — | ICP clock pin |
| OCDSCK | — | ST | — | OCDS clock pin, for EV chip only | |

| Pin Name | Function | OPT | I/T | O/T | Description |
|---|----------|--------------------------------|-----|--------------------|---|
| PA3/PTP0/CTP0/PTP1/ SDI/SDA/URX/UTX/TX/ SSEG9/SCOM9/XT2 | PA3 | PAPU PAWU PAS1 | ST | CMOS | General purpose I/O. Register enabled pull-up and wake-up |
| | PTP0 | PAS1 | — | CMOS | PTM0 output |
| | CTP0 | PAS1 | — | CMOS | CTM0 output |
| | PTP1 | PAS1 IFS1 | ST | — | PTM1 capture input |
| | SDI | PAS1 IFS2 | ST | — | SPI data input |
| | SDA | PAS1 IFS2 | ST | NMOS | I ² C data line |
| | URX/UTX | PAS1 IFS2 | ST | CMOS | USIM UART serial data input in full-duplex communication or UART serial data input/output in Single Wire Mode communication |
| | TX | PAS1 | — | CMOS | UART serial data output |
| | SSEG9 | PAS1 | — | AN | Software controlled LCD segment output |
| | SCOM9 | PAS1 | — | AN | Software controlled LCD common output |
| XT2 | PAS1 | — | AN | LXT oscillator pin | |
| PA4/PTP0/PTP1/CTP0/ PTP0I/INT0/VREFI/RX/ TX/SSEG12/SCOM12 | PA4 | PAWU PAPU PAS2 | ST | CMOS | General purpose I/O. Register enabled pull-up and wake-up |
| | PTP0 | PAS2 | — | CMOS | PTM0 output |
| | PTP1 | PAS2 | — | CMOS | PTM1 output |
| | CTP0 | PAS2 | — | CMOS | CTM0 output |
| | PTP0I | PAS2 IFS0 | ST | — | PTM0 capture input |
| | INT0 | PAS2 IFS1 INTEG INTC0 | ST | — | External interrupt 0 input |
| | VREFI | PAS2 | AN | — | A/D converter PGA input |
| | RX/TX | PAS2 IFS1 | ST | CMOS | UART serial data input in full-duplex communication or UART serial data input/output in Single Wire Mode communication |
| | SSEG12 | PAS2 | — | AN | Software controlled LCD segment output |
| | SCOM12 | PAS2 | — | AN | Software controlled LCD common output |
| PA5/PTP0/CTP1/ PTP1B/CTP0B/AN9/ SSEG6/SCOM6 | PA5 | PAWU PAPU PAS2 | ST | CMOS | General purpose I/O. Register enabled pull-up and wake-up |
| | PTP0 | PAS2 | — | CMOS | PTM0 output |
| | CTP1 | PAS2 | — | CMOS | CTM1 output |
| | PTP1B | PAS2 | — | CMOS | PTM1 inverted output |
| | CTP0B | PAS2 | — | CMOS | CTM0 inverted output |
| | AN9 | PAS2 | AN | — | A/D converter external input channel |
| | SSEG6 | PAS2 | — | AN | Software controlled LCD segment output |
| | SCOM6 | PAS2 | — | AN | Software controlled LCD common output |

| Pin Name | Function | OPT | I/T | O/T | Description |
|---|----------|----------------------|-----|--------------------|--|
| PA6/PTP1/CTP0/ PTP0B/CTP1B/AN10/ SSEG7/SCOM7 | PA6 | PAWU PAPU PAS3 | ST | CMOS | General purpose I/O. Register enabled pull-up and wake-up |
| | PTP1 | PAS3 | — | CMOS | PTM1 output |
| | CTP0 | PAS3 | — | CMOS | CTM0 output |
| | PTP0B | PAS3 | — | CMOS | PTM0 inverted output |
| | CTP1B | PAS3 | — | CMOS | CTM1 inverted output |
| | AN10 | PAS3 | AN | — | A/D converter external input channel |
| | SSEG7 | PAS3 | — | AN | Software controlled LCD segment output |
| | SCOM7 | PAS3 | — | AN | Software controlled LCD common output |
| | VREF | PAS3 | AN | — | A/D converter external reference voltage input |
| PA7/PTP1/CTP1/PTP0I/ SCK/SCL/RX/TX/ SSEG8/SCOM8/XT1 | PA7 | PAPU PAWU PAS0 | ST | CMOS | General purpose I/O. Register enabled pull-up and wake-up |
| | PTP1 | PAS0 | — | CMOS | PTM1 output |
| | CTP1 | PAS0 | — | CMOS | CTM1 output |
| | PTP0I | PAS0 IFS0 | ST | — | PTM0 capture input |
| | SCK | PAS0 IFS2 | ST | CMOS | SPI serial clock |
| | SCL | PAS0 IFS2 | ST | NMOS | I ² C clock line |
| | RX/TX | PAS0 IFS1 | ST | CMOS | UART serial data input in full-duplex communication or UART serial data input/output in Single Wire Mode communication |
| | SSEG8 | PAS0 | — | AN | Software controlled LCD segment output |
| | SCOM8 | PAS0 | — | AN | Software controlled LCD common output |
| XT1 | PAS0 | AN | — | LXT oscillator pin | |
| PB0/SSEG18/SCOM18/ KEY1 | PB0 | PBPU PBS0 | ST | CMOS | General purpose I/O. Register enabled pull-up |
| | SSEG18 | PBS0 | — | AN | Software controlled LCD segment output |
| | SCOM18 | PBS0 | — | AN | Software controlled LCD common output |
| | KEY1 | PBS0 | AN | — | Touch key input |
| PB1/SSEG19/SCOM19/ KEY2 | PB1 | PBPU PBS0 | ST | CMOS | General purpose I/O. Register enabled pull-up |
| | SSEG19 | PBS0 | — | AN | Software controlled LCD segment output |
| | SCOM19 | PBS0 | — | AN | Software controlled LCD common output |
| | KEY2 | PBS0 | AN | — | Touch key input |
| PB2/SSEG20/SCOM20/ KEY3 | PB2 | PBPU PBS0 | ST | CMOS | General purpose I/O. Register enabled pull-up |
| | SSEG20 | PBS0 | — | AN | Software controlled LCD segment output |
| | SCOM20 | PBS0 | — | AN | Software controlled LCD common output |
| | KEY3 | PBS0 | AN | — | Touch key input |
| PB3/SSEG21/SCOM21/ KEY4 | PB3 | PBPU PBS0 | ST | CMOS | General purpose I/O. Register enabled pull-up |
| | SSEG21 | PBS0 | — | AN | Software controlled LCD segment output |
| | SCOM21 | PBS0 | — | AN | Software controlled LCD common output |
| | KEY4 | PBS0 | AN | — | Touch key input |

| Pin Name | Function | OPT | I/T | O/T | Description |
|-------------------------------------|----------|--------------|-----|------|---|
| PB4/PTP0/PTP1B/ PTP1I/KEY5 | PB4 | PBPU PBS1 | ST | CMOS | General purpose I/O. Register enabled pull-up |
| | PTP0 | PBS1 | — | CMOS | PTM0 output |
| | PTP1B | PBS1 | — | CMOS | PTM1 inverted output |
| | PTP1I | PBS1 IFS1 | ST | — | PTM1 capture input |
| | KEY5 | PBS1 | AN | — | Touch key input |
| PB5/PTP0B/PTP1/ PTCK1/KEY6 | PB5 | PBPU PBS1 | ST | CMOS | General purpose I/O. Register enabled pull-up |
| | PTP0B | PBS1 | — | CMOS | PTM0 inverted output |
| | PTP1 | PBS1 | — | CMOS | PTM1 output |
| | PTCK1 | PBS1 IFS0 | ST | — | PTM1 clock input or capture input |
| | KEY6 | PBS1 | AN | — | Touch key input |
| PB6/PTP1/CTP0/ PTCK0/KEY7 | PB6 | PBPU PBS1 | ST | CMOS | General purpose I/O. Register enabled pull-up |
| | PTP1 | PBS1 | — | CMOS | PTM1 output |
| | CTP0 | PBS1 | — | CMOS | CTM0 output |
| | PTCK0 | PBS1 IFS0 | ST | — | PTM0 clock input or capture input |
| | KEY7 | PBS1 | AN | — | Touch key input |
| PB7/PTP1B/CTP1/ PTP0I/KEY8 | PB7 | PBPU PBS1 | ST | CMOS | General purpose I/O. Register enabled pull-up |
| | PTP1B | PBS1 | — | CMOS | PTM1 inverted output |
| | CTP1 | PBS1 | — | CMOS | CTM1 output |
| | PTP0I | PBS1 IFS0 | ST | — | PTM0 capture input |
| | KEY8 | PBS1 | AN | — | Touch key input |
| PC0/CTP1/CTCK0/ SSEG0/SCOM0/KEY9 | PC0 | PCPU PCS0 | ST | CMOS | General purpose I/O. Register enabled pull-up |
| | CTP1 | PCS0 | — | CMOS | CTM1 output |
| | CTCK0 | PCS0 IFS0 | ST | — | CTM0 clock input |
| | SSEG0 | PCS0 | — | AN | Software controlled LCD segment output |
| | SCOM0 | PCS0 | — | AN | Software controlled LCD common output |
| | KEY9 | PCS0 | AN | — | Touch key input |
| PC1/CTP0/SSEG1/ SCOM1/KEY10 | PC1 | PCPU PCS0 | ST | CMOS | General purpose I/O. Register enabled pull-up |
| | CTP0 | PCS0 | — | CMOS | CTM0 output |
| | SSEG1 | PCS0 | — | AN | Software controlled LCD segment output |
| | SCOM1 | PCS0 | — | AN | Software controlled LCD common output |
| | KEY10 | PCS0 | AN | — | Touch key input |

| Pin Name | Function | OPT | I/T | O/T | Description |
|-----------------------------------|----------|--------------|-----|------|---|
| PC2/SCK/SCL/SDI/SDA/URX/UTX/KEY11 | PC2 | PCPU PCS0 | ST | CMOS | General purpose I/O. Register enabled pull-up |
| | SCK | PCS0 IFS2 | ST | CMOS | SPI serial clock |
| | SCL | PCS0 IFS2 | ST | NMOS | I ² C clock line |
| | SDI | PCS0 IFS2 | ST | — | SPI data input |
| | SDA | PCS0 IFS2 | ST | NMOS | I ² C data line |
| | URX/UTX | PCS0 IFS2 | ST | CMOS | USIM UART serial data input in full-duplex communication or UART serial data input/output in Single Wire Mode communication |
| | KEY11 | PCS0 | AN | — | Touch key input |
| PC3/SDI/SDA/URX/UTX/SDO/UTX/KEY12 | PC3 | PCPU PCS0 | ST | CMOS | General purpose I/O. Register enabled pull-up |
| | SDI | PCS0 IFS2 | ST | — | SPI data input |
| | SDA | PCS0 IFS2 | ST | NMOS | I ² C data line |
| | URX/UTX | PCS0 IFS2 | ST | CMOS | USIM UART serial data input in full-duplex communication or UART serial data input/output in Single Wire Mode communication |
| | SDO | PCS0 | — | CMOS | SPI serial data output |
| | UTX | PCS0 | — | CMOS | USIM UART serial data output |
| | KEY12 | PCS0 | AN | — | Touch key input |
| PC4/PTP1/SSEG2/SCOM2/KEY13 | PC4 | PCPU PCS1 | ST | CMOS | General purpose I/O. Register enabled pull-up |
| | PTP1 | PCS1 | — | CMOS | PTM1 output |
| | SSEG2 | PCS1 | — | AN | Software controlled LCD segment output |
| | SCOM2 | PCS1 | — | AN | Software controlled LCD common output |
| | KEY13 | PCS1 | AN | — | Touch key input |
| PC5/PTP1B/SSEG3/SCOM3/KEY14 | PC5 | PCPU PCS1 | ST | CMOS | General purpose I/O. Register enabled pull-up |
| | PTP1B | PCS1 | — | CMOS | PTM1 inverted output |
| | SSEG3 | PCS1 | — | AN | Software controlled LCD segment output |
| | SCOM3 | PCS1 | — | AN | Software controlled LCD common output |
| | KEY14 | PCS1 | AN | — | Touch key input |
| PC6/SCK/SCL/SSEG4/SCOM4/KEY15 | PC6 | PCPU PCS1 | ST | CMOS | General purpose I/O. Register enabled pull-up |
| | SCK | PCS1 IFS2 | ST | CMOS | SPI serial clock |
| | SCL | PCS1 IFS2 | ST | NMOS | I ² C clock line |
| | SSEG4 | PCS1 | — | AN | Software controlled LCD segment output |
| | SCOM4 | PCS1 | — | AN | Software controlled LCD common output |
| | KEY15 | PCS1 | AN | — | Touch key input |

| Pin Name | Function | OPT | I/T | O/T | Description |
|---|------------------|--------------|-----|------|---|
| PC7/SDI/SDA/URX/ UTX/SSEG5/SCOM5/ KEY16 | PC7 | PCPU PCS1 | ST | CMOS | General purpose I/O. Register enabled pull-up |
| | SDI | PCS1 IFS2 | ST | — | SPI data input |
| | SDA | PCS1 IFS2 | ST | NMOS | I ² C data line |
| | URX/UTX | PCS1 IFS2 | ST | CMOS | USIM UART serial data input in full-duplex communication or UART serial data input/output in Single Wire Mode communication |
| | SSEG5 | PCS1 | — | AN | Software controlled LCD segment output |
| | SCOM5 | PCS1 | — | AN | Software controlled LCD common output |
| | KEY16 | PCS1 | AN | — | Touch key input |
| PD0/PTP0/AN0/ SSEG24/SCOM24 | PD0 | PDP PDS0 | ST | CMOS | General purpose I/O. Register enabled pull-up |
| | PTP0 | PDS0 | — | CMOS | PTM0 output |
| | AN0 | PDS0 | AN | — | A/D converter external input channel |
| | SSEG24 | PDS0 | — | AN | Software controlled LCD segment output |
| | SCOM24 | PDS0 | — | AN | Software controlled LCD common output |
| PD1/PTP1/AN1/ SSEG25/SCOM25 | PD1 | PDP PDS0 | ST | CMOS | General purpose I/O. Register enabled pull-up |
| | PTP1 | PDS0 | — | CMOS | PTM1 output |
| | AN1 | PDS0 | AN | — | A/D converter external input channel |
| | SSEG25 | PDS0 | — | AN | Software controlled LCD segment output |
| | SCOM25 | PDS0 | — | AN | Software controlled LCD common output |
| PD2/PTP1/RX/TX/AN2 | PD2 | PDP PDS0 | ST | CMOS | General purpose I/O. Register enabled pull-up |
| | PTP1 | PDS0 | — | CMOS | PTM1 output |
| | RX/TX | PDS0 IFS1 | ST | CMOS | UART serial data input in full-duplex communication or UART serial data input/output in Single Wire Mode communication |
| | AN2 | PDS0 | AN | — | A/D converter external input channel |
| PD3/PTP1/TX/AN3 | PD3 | PDP PDS0 | ST | CMOS | General purpose I/O. Register enabled pull-up |
| | PTP1 | PDS0 | — | CMOS | PTM1 output |
| | TX | PDS0 | — | CMOS | UART serial data output |
| | AN3 | PDS0 | AN | — | A/D converter external input channel |
| PD4/PTCK1/ \overline{SCS} /AN4/ VREF | PD4 | PDP PDS1 | ST | CMOS | General purpose I/O. Register enabled pull-up |
| | PTCK1 | PDS1 IFS0 | ST | — | PTM1 clock input or capture input |
| | \overline{SCS} | PDS1 IFS2 | ST | CMOS | SPI slave select |
| | AN4 | PDS1 | AN | — | A/D converter external input channel |
| | VREF | PDS1 | AN | — | A/D converter external reference voltage input |

| Pin Name | Function | OPT | I/T | O/T | Description |
|---|----------|--------------------------------|-----|------|---|
| PD5/INT0/PTP0/PTCK0/ SDO/UTX/AN5 | PD5 | PDP PDS1 | ST | CMOS | General purpose I/O. Register enabled pull-up |
| | INT0 | PDS1 IFS1 INTEG INTC0 | ST | — | External interrupt 0 input |
| | PTP0 | PDS1 | — | CMOS | PTM0 output |
| | PTCK0 | PDS1 IFS0 | ST | — | PTM0 clock input or capture input |
| | SDO | PDS1 | — | CMOS | SPI data output |
| | UTX | PDS1 | — | CMOS | USIM UART serial data output |
| | AN5 | PDS1 | AN | — | A/D converter external input channel |
| PD6/INT1/SCK/SCL/ RX/TX/AN6 | PD6 | PDP PDS1 | ST | CMOS | General purpose I/O. Register enabled pull-up |
| | INT1 | PDS1 IFS1 INTEG INTC0 | ST | — | External interrupt 1 input |
| | SCK | PDS1 IFS2 | ST | CMOS | SPI serial clock |
| | SCL | PDS1 IFS2 | ST | NMOS | I ² C clock line |
| | RX/TX | PDS1 IFS1 | ST | CMOS | UART serial data input in full-duplex communication or UART serial data input/output in Single Wire Mode communication |
| | AN6 | PDS1 | AN | — | A/D converter external input channel |
| PD7/SDI/SDA/URX/UTX/ TX/AN7 | PD7 | PDP PDS1 | ST | CMOS | General purpose I/O. Register enabled pull-up |
| | SDI | PDS1 IFS2 | ST | — | SPI data input |
| | SDA | PDS1 IFS2 | ST | NMOS | I ² C data line |
| | URX/UTX | PDS1 IFS2 | ST | CMOS | USIM UART serial data input in full-duplex communication or UART serial data input/output in Single Wire Mode communication |
| | TX | PDS1 | — | CMOS | UART serial data output |
| | AN7 | PDS1 | AN | — | A/D converter external input channel |
| PE0/PTCK1/SDI/SDA/ URX/UTX/SSEG22/ SCOM22/KEY17 | PE0 | PEPU PES0 | ST | CMOS | General purpose I/O. Register enabled pull-up |
| | PTCK1 | PES0 IFS0 | ST | — | PTM1 clock input or capture input |
| | SDI | PES0 IFS2 | ST | — | SPI data input |
| | SDA | PES0 IFS2 | ST | NMOS | I ² C data line |
| | URX/UTX | PES0 IFS2 | ST | CMOS | USIM UART serial data input in full-duplex communication or UART serial data input/output in Single Wire Mode communication |
| | SSEG22 | PES0 | — | AN | Software controlled LCD segment output |
| | SCOM22 | PES0 | — | AN | Software controlled LCD common output |
| | KEY17 | PES0 | AN | — | Touch key input |

| Pin Name | Function | OPT | I/T | O/T | Description |
|---|----------|--------------|-----|------|--|
| PE1/PTP0I/SDO/UTX/ SSEG23/SCOM23/ KEY18 | PE1 | PEPU PES0 | ST | CMOS | General purpose I/O. Register enabled pull-up |
| | PTP0I | PES0 IFS0 | ST | — | PTM0 capture input |
| | SDO | PES0 | — | CMOS | SPI data output |
| | UTX | PES0 | — | CMOS | USIM UART serial data output |
| | SSEG23 | PES0 | — | AN | Software controlled LCD segment output |
| | SCOM23 | PES0 | — | AN | Software controlled LCD common output |
| | KEY18 | PES0 | AN | — | Touch key input |
| PE2/CTP0/PTCK0/RX/ TX/KEY19 | PE2 | PEPU PES0 | ST | CMOS | General purpose I/O. Register enabled pull-up |
| | CTP0 | PES0 | — | CMOS | CTM0 output |
| | PTCK0 | PES0 IFS0 | ST | — | PTM0 clock input or capture input |
| | RX/TX | PES0 IFS1 | ST | CMOS | UART serial data input in full-duplex communication or UART serial data input/output in Single Wire Mode communication |
| | KEY19 | PES0 | AN | — | Touch key input |
| PE3/PTP1B/PTP1I/TX/ KEY20 | PE3 | PEPU PES0 | ST | CMOS | General purpose I/O. Register enabled pull-up |
| | PTP1B | PES0 | — | CMOS | PTM1 inverted output |
| | PTP1I | PES0 IFS1 | ST | — | PTM1 capture input |
| | TX | PES0 | — | CMOS | UART serial data output |
| | KEY20 | PES0 | AN | — | Touch key input |
| PE4/PTP1/CTP0/VDDIO | PE4 | PEPU PES1 | ST | CMOS | General purpose I/O. Register enabled pull-up |
| | PTP1 | PES1 | — | CMOS | PTM1 output |
| | CTP0 | PES1 | — | CMOS | CTM0 output |
| | VDDIO | PES1 PMPS | PWR | — | Power for PD4~PD7 pins |
| PE5/PTP1B/CTP1/ OCPP | PE5 | PEPU PES1 | ST | CMOS | General purpose I/O. Register enabled pull-up |
| | PTP1B | PES1 | — | CMOS | PTM1 inverted output |
| | CTP1 | PES1 | — | CMOS | CTM1 output |
| | OCPP | PES1 | AN | — | OCP positive input |
| PE6/CTP0/PTP1I/ OCPAO/OCPCOUT | PE6 | PEPU PES1 | ST | CMOS | General purpose I/O. Register enabled pull-up |
| | CTP0 | PES1 | — | CMOS | CTM0 output |
| | PTP1I | PES1 IFS1 | ST | — | PTM1 capture input |
| | OCPAO | PES1 | — | AN | OCP operational amplifier output |
| | OCPCOUT | PES1 | — | CMOS | OCP comparator output |
| PE7/PTP1/PTCK1/ OCPN/AN8 | PE7 | PEPU PES1 | ST | CMOS | General purpose I/O. Register enabled pull-up |
| | PTP1 | PES1 | — | CMOS | PTM1 output |
| | PTCK1 | PES1 IFS0 | ST | — | PTM1 clock input or capture input |
| | OCPN | PES1 | AN | — | OCP negative input |
| | AN8 | PES1 | AN | — | A/D converter external input channel |

| Pin Name | Function | OPT | I/T | O/T | Description |
|--|----------|--------------|-----|------|---|
| PF0/CTP0/CTCK1/ OCPVR/RX/TX | PF0 | PFPU PFS0 | ST | CMOS | General purpose I/O. Register enabled pull-up |
| | CTP0 | PFS0 | — | CMOS | CTM0 output |
| | CTCK1 | PFS0 IFS0 | ST | — | CTM1 clock input |
| | OCPVR | PFS0 | AN | — | OCP DAC reference voltage input |
| | RX/TX | PFS0 IFS1 | ST | CMOS | UART serial data input in full-duplex communication or UART serial data input/output in Single Wire Mode communication |
| PF1/CTCK0/VREFI/ DACBUF/TX | PF1 | PFPU PFS0 | ST | CMOS | General purpose I/O. Register enabled pull-up |
| | CTCK0 | PFS0 IFS0 | ST | — | CTM0 clock input |
| | VREFI | PFS0 | AN | — | A/D converter PGA input |
| | DACBUF | PFS0 | — | AN | OCP DAC output with buffer |
| | TX | PFS0 | — | CMOS | UART serial data output |
| PF2/CTP0B/SDO/UTX/ SSEG14/SCOM14 | PF2 | PFPU PFS0 | ST | CMOS | General purpose I/O. Register enabled pull-up |
| | CTP0B | PFS0 | — | CMOS | CTM0 inverted output |
| | SDO | PFS0 | — | CMOS | SPI data output |
| | UTX | PFS0 | — | CMOS | USIM UART serial data output |
| | SSEG14 | PFS0 | — | AN | Software controlled LCD segment output |
| | SCOM14 | PFS0 | — | AN | Software controlled LCD common output |
| PF3/PTP0/SDI/SDA/ URX/UTX/SSEG15/ SCOM15 | PF3 | PFPU PFS0 | ST | CMOS | General purpose I/O. Register enabled pull-up |
| | PTP0 | PFS0 | — | CMOS | PTM0 output |
| | SDI | PFS0 IFS2 | ST | — | SPI data input |
| | SDA | PFS0 IFS2 | ST | NMOS | I ² C data line |
| | URX/UTX | PFS0 IFS2 | ST | CMOS | USIM UART serial data input in full-duplex communication or UART serial data input/output in Single Wire Mode communication |
| | SSEG15 | PFS0 | — | AN | Software controlled LCD segment output |
| | SCOM15 | PFS0 | — | AN | Software controlled LCD common output |
| PF4/PTP1/SCK/SCL/ SSEG16/SCOM16 | PF4 | PFPU PFS1 | ST | CMOS | General purpose I/O. Register enabled pull-up |
| | PTP1 | PFS1 | — | CMOS | PTM1 output |
| | SCK | PFS1 IFS2 | ST | CMOS | SPI serial clock |
| | SCL | PFS1 IFS2 | ST | NMOS | I ² C clock line |
| | SSEG16 | PFS1 | — | AN | Software controlled LCD segment output |
| | SCOM16 | PFS1 | — | AN | Software controlled LCD common output |
| PF5/PTP0/SCS/ SSEG17/SCOM17 | PF5 | PFPU PFS1 | ST | CMOS | General purpose I/O. Register enabled pull-up |
| | PTP0 | PFS1 | — | CMOS | PTM0 output |
| | SCS | PFS1 IFS2 | ST | CMOS | SPI slave select |
| | SSEG17 | PFS1 | — | AN | Software controlled LCD segment output |
| | SCOM17 | PFS1 | — | AN | Software controlled LCD common output |

| Pin Name | Function | OPT | I/T | O/T | Description |
|----------|----------|-----|-----|-----|--------------------------------------|
| VDD/AVDD | VDD | — | PWR | — | Positive power supply |
| | AVDD | — | PWR | — | Analog positive power supply |
| VSS/AVSS | VSS | — | PWR | — | Negative power supply, ground |
| | AVSS | — | PWR | — | Analog negative power supply, ground |

Legend: I/T: Input type;

OPT: Optional by register option;

ST: Schmitt Trigger input;

NMOS: NMOS output;

O/T: Output type;

PWR: Power;

CMOS: CMOS output;

AN: Analog signal

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°C to 150°C |
| Operating Temperature..... | -40°C to 85°C |
| I _{OH} Total | -80mA |
| I _{OL} Total | 100mA |
| Total Power Dissipation | 500mW |

Note: These are stress ratings only. Stresses exceeding the range specified under “Absolute Maximum Ratings” may cause substantial damage to the device. Functional operation of the device at other conditions beyond those listed in the specification is not implied and prolonged exposure to extreme conditions may affect 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°C~85°C

| Symbol | Parameter | Test Conditions | Min. | Typ. | Max. | Unit |
|-----------------|--------------------------|---------------------------|------|------|------|------|
| V _{DD} | 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} =32768Hz | 1.8 | — | 5.5 | V |
| | Operating Voltage – LIRC | f _{sys} =32kHz | 1.8 | — | 5.5 | V |

Operating Current Characteristics

Ta=-40°C~85°C

| Symbol | Operating Mode | Test Conditions | | Min. | Typ. | Max. | Unit |
|-----------------|------------------|-----------------|---------------------------|-------------------------|------|------|------|
| | | V _{DD} | Conditions | | | | |
| I _{DD} | SLOW Mode – LIRC | 1.8V | f _{sys} =32kHz | — | 10 | 20 | μA |
| | | 3V | | — | 12.5 | 25.0 | μA |
| | | 5V | | — | 30 | 50 | μA |
| | SLOW Mode – LXT | 1.8V | f _{sys} =32768Hz | — | 12.5 | 25.0 | μA |
| | | 3V | | — | 15 | 30 | μA |
| | | 5V | | — | 30 | 50 | μA |
| | FAST Mode – HIRC | 1.8V | f _{sys} =8MHz | — | 0.6 | 1.0 | mA |
| | | 3V | | — | 0.8 | 1.2 | mA |
| | | 5V | | — | 1.6 | 2.4 | mA |
| | | 2.7V | f _{sys} =12MHz | — | 1.0 | 1.4 | mA |
| | | 3V | | — | 1.2 | 1.8 | mA |
| | | 5V | | — | 2.4 | 3.6 | mA |
| | | 3.3V | | f _{sys} =16MHz | — | 1.5 | 3.0 |
| | 5V | — | 2.5 | | 5.0 | mA | |

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

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

Standby Current Characteristics

Ta=25°C, unless otherwise specified

| Symbol | Standby Mode | Test Conditions | | Min. | Typ. | Max. | Max. @85°C | Unit |
|------------------|-------------------|--|--|------|------|------|---------------|------|
| | | V _{DD} | Conditions | | | | | |
| I _{STB} | SLEEP Mode | 1.8V | WDT off | — | 0.45 | 0.80 | 7.00 | μA |
| | | 3V | | — | 0.45 | 0.90 | 8.00 | μA |
| | | 5V | | — | 0.5 | 2.0 | 10.0 | μA |
| | | 1.8V | WDT on | — | 1.5 | 3.0 | 9.2 | μA |
| | | 3V | | — | 2.00 | 4.00 | 11.25 | μA |
| | | 5V | | — | 3.2 | 6.4 | 16.3 | μA |
| | IDLE0 Mode – LIRC | 1.8V | f _{SUB} on | — | 2.4 | 4.0 | 8.1 | μA |
| | | 3V | | — | 3.00 | 5.00 | 9.75 | μA |
| | | 5V | | — | 5.0 | 10.0 | 13.3 | μA |
| | IDLE0 Mode – LXT | 1.8V | f _{SUB} on | — | 2.4 | 4.0 | 9.2 | μA |
| | | 3V | | — | 3.0 | 5.0 | 11.2 | μA |
| | | 5V | | — | 5.0 | 10.0 | 15.4 | μA |
| | IDLE1 Mode – HIRC | 1.8V | f _{SUB} on, f _{sys} =8MHz | — | 288 | 400 | 480 | μA |
| | | 3V | | — | 360 | 500 | 600 | μA |
| | | 5V | | — | 600 | 800 | 960 | μA |
| | | 2.7V | f _{SUB} on, f _{sys} =12MHz | — | 432 | 600 | 720 | μA |
| 3V | | — | | 540 | 750 | 900 | μA | |
| 5V | | — | | 800 | 1200 | 1440 | μA | |
| 3.3V | | f _{SUB} on, f _{sys} =16MHz | | — | 0.80 | 1.20 | 1.44 | mA |
| 5V | — | | 1.4 | 2.0 | 2.4 | mA | | |

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.

LXT Current Characteristics

Ta=-40°C~85°C

| Symbol | Parameter | Test Conditions | | Min. | Typ. | Max. | Unit |
|------------------|-----------------------------------|-----------------|---------------------|------|------|------|------|
| | | V _{DD} | Conditions | | | | |
| I _{LXT} | Additional current for LXT enable | 3V | LXTSP[1:0]=00 | — | — | 2 | μA |
| | | | LXTSP[1:0]=01 | — | — | 3 | |
| | | | LXTSP[1:0]=10 or 11 | — | — | 5 | |
| | | 5V | LXTSP[1:0]=00 | — | — | 2 | μA |
| | | | LXTSP[1:0]=01 | — | — | 3 | |
| | | | LXTSP[1:0]=10 or 11 | — | — | 5 | |

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.

Internal High Speed 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 _{DD} | Temp. | | | | |
| f _{HIRC} | 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 | -10% | 8 | +5% | | |
| | | -40°C~85°C | -15% | 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_{DD} 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_{DD} 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 only 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 $\pm 20\%$.

Low Speed Internal Oscillator Characteristics – LIRC

| Symbol | Parameter | Test Conditions | | Min. | Typ. | Max. | Unit |
|--------------------|--------------------|-----------------|------------|------|------|------|------|
| | | V _{DD} | Temp. | | | | |
| f _{LIRC} | LIRC Frequency | 3V | 25°C | -2% | 32 | +2% | kHz |
| | | 2.2V~5.5V | -40°C~85°C | -7% | 32 | +7% | |
| | | 1.8V~5.5V | -40°C~85°C | -10% | 32 | +10% | |
| t _{START} | LIRC Start Up Time | — | — | — | — | 100 | μs |

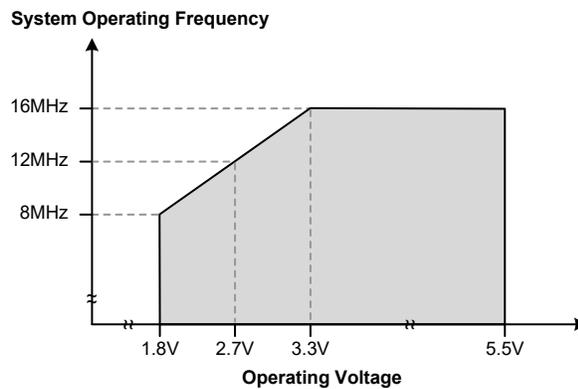
External Low Speed Crystal Oscillator Characteristics – LXT

Ta=25°C

| Symbol | Parameter | Test Conditions | | Min. | Typ. | Max. | Unit |
|--------------------|---------------------------------------|-----------------|------------|-------|-------|------|------|
| | | V _{DD} | Conditions | | | | |
| f _{LXT} | Oscillator Frequency | 1.8V~5.5V | — | — | 32768 | — | Hz |
| t _{START} | LXT Start Up Time | 3V | — | — | — | 1000 | ms |
| | | 5V | — | — | — | 1000 | ms |
| Duty Cycle | Duty Cycle | — | — | 40 | — | 60 | % |
| R _{NEG} | Negative Resistance ^(Note) | 1.8V | — | 3×ESR | — | — | Ω |

Note: C₁, C₂ and R_p are external components. When the LXTSP[1:0] bits is 00 or 01, C₁=C₂=10pF, R_p=10MΩ, C_L=6pF, ESR=30kΩ. When the LXTSP[1:0] bits is 10 or 11, C₁=C₂=12pF, R_p=10MΩ, C_L=7pF, ESR=90kΩ.

Operating Frequency Characteristic Curves



System Start Up Time Characteristics

Ta=-40°C~85°C

| Symbol | Parameter | Test Conditions | | Min. | Typ. | Max. | Unit |
|---------------------|--|---|---|------|------|------------------|-------------------|
| | | V _{DD} | Conditions | | | | |
| t _{SST} | System Start-up Time (Wake-up from Condition where f _{sys} is off) | — | f _{sys} =f _H ~f _H /64, f _H =f _{HIRC} | — | 16 | — | t _{HIRC} |
| | | — | f _{sys} =f _{SUB} =f _{LXT} | — | 1024 | — | t _{LXT} |
| | | — | f _{sys} =f _{SUB} =f _{LIRC} | — | 2 | — | t _{LIRC} |
| | System Start-up Time (Wake-up from Condition where f _{sys} is on) | — | f _{sys} =f _H ~f _H /64, f _H =f _{HIRC} | — | 2 | — | t _H |
| | | — | f _{sys} =f _{SUB} =f _{LXT} or f _{LIRC} | — | 2 | — | t _{SUB} |
| | System Speed Switch Time (FAST to SLOW Mode or SLOW to FAST Mode) | — | f _{HIRC} switches from off → on | — | 16 | — | t _{HIRC} |
| — | | f _{LXT} switches from off → on | — | 1024 | — | t _{LXT} | |
| t _{RSTD} | System Reset Delay Time (Reset source from Power-on Reset or LVR Hardware Reset) | — | RR _{POR} =5 V/ms | 10 | 16 | 24 | ms |
| | System Reset Delay Time (LVRC/WDT Software Reset) | — | — | | | | |
| | System Reset Delay Time (Reset source from WDT Overflow) | — | — | | | | |
| t _{SRESET} | Minimum Software Reset Width to Reset | — | — | 45 | 90 | 180 | μs |

- Note: 1. For the System Start-up time values, whether f_{sys} is on or off depends upon the mode type and the chosen f_{sys} system oscillator. Details are provided in the System Operating Modes section.
2. The time units, shown by the symbols t_{HIRC}, t_{sys} etc. are the inverse of the corresponding frequency values as provided in the frequency tables. For example, t_{HIRC}=1/f_{HIRC}, t_{sys}=1/f_{sys} 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_{START}, as provided in the LIRC frequency table, must be added to the t_{SST} 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

Input/Output (without Multi-power) D.C. Characteristics

For PA0~PA7, PB0~PB7, PC0~PC7, PD0~PD3, PE0~PE7 and PF0~PF5 Pins.

Ta=-40°C~85°C

| Symbol | Parameter | Test Conditions | | Min. | Typ. | Max. | Unit |
|-----------------|---|-----------------|--|--------------------|------|--------------------|------|
| | | V _{DD} | Conditions | | | | |
| V _{IL} | Input Low Voltage for I/O Ports | 5V | — | 0 | — | 1.5 | V |
| | | — | — | 0 | — | 0.2V _{DD} | |
| V _{IH} | Input High Voltage for I/O Ports | 5V | — | 3.5 | — | 5.0 | V |
| | | — | — | 0.8V _{DD} | — | V _{DD} | |
| I _{OL} | Sink Current for PC0~PC7, PD0~PD3 and PF0~PF5 Pins | 3V | V _{OL} =0.1V _{DD} , P _{xNSn} =0 | 16 | 32 | — | mA |
| | | | V _{OL} =0.1V _{DD} , P _{xNSn} =1 | 25 | 50 | — | |
| | | 5V | V _{OL} =0.1V _{DD} , P _{xNSn} =0 | 32 | 65 | — | mA |
| | | | V _{OL} =0.1V _{DD} , P _{xNSn} =1 | 50 | 100 | — | |
| | Sink Current for PA0~PA7, PB0~PB7 and PE0~PE7 Pins | 3V | V _{OL} =0.1V _{DD} | 16 | 32 | — | mA |
| | | 5V | | 32 | 65 | — | |

| Symbol | Parameter | Test Conditions | | Min. | Typ. | Max. | Unit |
|--------------------|---|-----------------|--|---------------------------------|-------|------|------------------|
| | | V _{DD} | Conditions | | | | |
| I _{OH} | Source Current for I/O Ports | 3V | V _{OH} =0.9V _{DD} , SLEDCn[m+2:m+1:m]=000B, (n=0, 1, 2, 3, 4 or 5, m=0 or 3) | -0.5 | -1.0 | — | mA |
| | | 5V | (n=0, 1, 2, 3, 4 or 5, m=0 or 3) | -1.0 | -2.0 | — | mA |
| | | 3V | V _{OH} =0.9V _{DD} , SLEDCn[m+2:m+1:m]=001B, (n=0, 1, 2, 3, 4 or 5, m=0 or 3) | -1.2 | -2.5 | — | mA |
| | | 5V | (n=0, 1, 2, 3, 4 or 5, m=0 or 3) | -2.5 | -5.0 | — | mA |
| | | 3V | V _{OH} =0.9V _{DD} , SLEDCn[m+2:m+1:m]=010B, (n=0, 1, 2, 3, 4 or 5, m=0 or 3) | -2.0 | -4.0 | — | mA |
| | | 5V | (n=0, 1, 2, 3, 4 or 5, m=0 or 3) | -4.0 | -8.0 | — | mA |
| | | 3V | V _{OH} =0.9V _{DD} , SLEDCn[m+2:m+1:m]=011B, (n=0, 1, 2, 3, 4 or 5, m=0 or 3) | -2.5 | -5.0 | — | mA |
| | | 5V | (n=0, 1, 2, 3, 4 or 5, m=0 or 3) | -5.5 | -11.0 | — | mA |
| | | 3V | V _{OH} =0.9V _{DD} , SLEDCn[m+2:m+1:m]=100B, (n=0, 1, 2, 3, 4 or 5, m=0 or 3) | -3.5 | -7.0 | — | mA |
| | | 5V | (n=0, 1, 2, 3, 4 or 5, m=0 or 3) | -7 | -14 | — | mA |
| | | 3V | V _{OH} =0.9V _{DD} , SLEDCn[m+2:m+1:m]=101B, (n=0, 1, 2, 3, 4 or 5, m=0 or 3) | -4.0 | -8.5 | — | mA |
| | | 5V | (n=0, 1, 2, 3, 4 or 5, m=0 or 3) | -8.5 | -17.0 | — | mA |
| | | 3V | V _{OH} =0.9V _{DD} , SLEDCn[m+2:m+1:m]=110B, (n=0, 1, 2, 3, 4 or 5, m=0 or 3) | -5 | -10 | — | mA |
| | | 5V | (n=0, 1, 2, 3, 4 or 5, m=0 or 3) | -10 | -20 | — | mA |
| R _{PH} | Pull-high Resistance for I/O Ports ⁽¹⁾ | 3V | LVPU=0 PxPU=FFH (Px: PA, PB, PC, PD, PE, PF) | 20 | 60 | 100 | kΩ |
| | | 5V | | 10 | 30 | 50 | |
| | | 3V | LVPU=1 PxPU=FFH (Px: PA, PB, PC, PD, PE, PF) | 6.67 | 15 | 23 | |
| | | 5V | | 3.5 | 7.5 | 12 | |
| I _{LEAK} | Input Leakage Current | 5V | V _{IN} =V _{DD} or V _{IN} =V _{SS} | — | — | ±1 | μA |
| t _{INT} | External Interrupt Input Minimum Pulse Width | — | — | 10 | — | — | μs |
| t _{TCK} | CTM/PTM Clock Input Minimum Pulse Width | — | — | 0.3 | — | — | μs |
| t _{TPI} | PTMn Capture Input Pin Minimum Pulse Width | — | — | 0.3 | — | — | μs |
| f _{TMCLK} | PTMn Maximum Timer Clock Source Frequency | 5V | — | — | — | 1 | f _{sys} |
| t _{CPW} | PTMn Minimum Capture Pulse Width | — | — | t _{CPW} ⁽²⁾ | — | — | μs |

Note: 1. The R_{PH} 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_{PH} value.

2. If PTnCAPTS=0, then t_{CPW}=max(2×t_{TMCLK}, t_{TPI})

If PTnCAPTS=1, then t_{CPW}=max(2×t_{TMCLK}, t_{TCK})

Ex1: If PTnCAPTS=0, f_{TMCLK}=16MHz, t_{TPI}=0.3μs, then t_{CPW}=max(0.125μs, 0.3μs)=0.3μs

Ex2: If PTnCAPTS=1, f_{TMCLK}=16MHz, t_{TCK}=0.3μs, then t_{CPW}=max(0.125μs, 0.3μs)=0.3μs

Ex3: If PTnCAPTS=0, f_{TMCLK}=8MHz, t_{TPI}=0.3μs, then t_{CPW}=max(0.25μs, 0.3μs)=0.3μs

Where t_{TMCLK}=1/f_{TMCLK}

Input/Output (with Multi-power) D.C. Characteristics

For PD4~PD7 Pins.

Ta=-40°C~85°C

| Symbol | Parameter | Test Conditions | | Min. | Typ. | Max. | Unit |
|--|---|-----------------|---|---|------|---|------|
| | | V _{DD} | Conditions | | | | |
| V _{DD} | Power Supply – V _{DD} | — | — | 1.8 | 5.0 | 5.5 | V |
| V _{DDIO} | Power Supply – V _{DD1} | — | — | 1.8 | — | V _{DD} | V |
| V _{IL} | Input Low Voltage for I/O Ports | 5V | Pin power=V _{DDn} , V _{DDn} =V _{DD} , n=0~1 | 0 | — | 1.5 | V |
| | | — | Pin power=V _{DDn} , n=0~1 | 0 | — | 0.2 (V _{DD} /V _{DDIO}) | |
| V _{IH} | Input High Voltage for I/O Ports | 5V | Pin power=V _{DDn} , V _{DDn} =V _{DD} , n=0~1 | 3.5 | — | 5.0 | V |
| | | — | Pin power=V _{DDn} , n=0~1 | 0.8 (V _{DD} /V _{DDIO}) | — | V _{DD} /V _{DDIO} | |
| I _{OL} | Sink Current for I/O Ports | 3V | V _{OL} =0.1V _{DDn} , V _{DDn} =V _{DD} , n=0~1, P _{XNSn} =0 | 16 | 32 | — | mA |
| | | | V _{OL} =0.1V _{DDn} , V _{DDn} =V _{DD} , n=0~1, P _{XNSn} =1 | 25 | 50 | — | |
| | | 5V | V _{OL} =0.1V _{DDn} , V _{DDn} =V _{DD} , n=0~1, P _{XNSn} =0 | 32 | 65 | — | mA |
| | | | V _{OL} =0.1V _{DDn} , V _{DDn} =V _{DD} , n=0~1, P _{XNSn} =1 | 50 | 100 | — | |
| | | | V _{OL} =0.1V _{DDn} , V _{DDn} =3V, n=0~1, P _{XNSn} =0 | 20 | 40 | — | |
| V _{OL} =0.1V _{DDn} , V _{DDn} =3V, n=0~1, P _{XNSn} =1 | 25 | 50 | — | | | | |
| I _{OH} | Source Current for I/O Ports | 3V | V _{OH} =0.9V _{DDn} , V _{DDn} =V _{DD} , n=0~1 SLEDC _{n[m+2:m+1:m]} =000B, (n=0, 1, 2, 3, 4 or 5, m=0 or 3) | -0.5 | -1.0 | — | mA |
| | | 5V | (n=0, 1, 2, 3, 4 or 5, m=0 or 3) | -1.0 | -2.0 | — | |
| | | 5V | V _{OH} =0.9V _{DDn} , V _{DDn} =3V, n=0~1 SLEDC _{n[m+2:m+1:m]} =000B, (n=0, 1, 2, 3, 4 or 5, m=0 or 3) | -0.3 | -0.6 | — | mA |
| | | 3V | V _{OH} =0.9V _{DDn} , V _{DDn} =V _{DD} , n=0~1 SLEDC _{n[m+2:m+1:m]} =001B, (n=0, 1, 2, 3, 4 or 5, m=0 or 3) | -1.2 | -2.5 | — | |
| | | 5V | (n=0, 1, 2, 3, 4 or 5, m=0 or 3) | -2.5 | -5.0 | — | mA |
| | | 5V | V _{OH} =0.9V _{DDn} , V _{DDn} =3V, n=0~1 SLEDC _{n[m+2:m+1:m]} =001B, (n=0, 1, 2, 3, 4 or 5, m=0 or 3) | -0.6 | -1.3 | — | |
| | | 3V | V _{OH} =0.9V _{DDn} , V _{DDn} =V _{DD} , n=0~1 SLEDC _{n[m+2:m+1:m]} =010B, (n=0, 1, 2, 3, 4 or 5, m=0 or 3) | -2.0 | -4.0 | — | mA |
| | | 5V | (n=0, 1, 2, 3, 4 or 5, m=0 or 3) | -4.0 | -8.0 | — | |
| 5V | V _{OH} =0.9V _{DDn} , V _{DDn} =3V, n=0~1 SLEDC _{n[m+2:m+1:m]} =010B, (n=0, 1, 2, 3, 4 or 5, m=0 or 3) | -1.0 | -2.0 | — | | | |

| Symbol | Parameter | Test Conditions | | Min. | Typ. | Max. | Unit |
|-------------------|---|-----------------|---|------|-------|------|------|
| | | V _{DD} | Conditions | | | | |
| I _{OH} | Source Current for I/O Ports | 3V | V _{OH} =0.9V _{DDn} , V _{DDn} =V _{DD} , n=0~1 SLEDCn[m+2:m+1:m]=011B, (n=0, 1, 2, 3, 4 or 5, m=0 or 3) | -2.5 | -5.0 | — | mA |
| | | 5V | (n=0, 1, 2, 3, 4 or 5, m=0 or 3) | -5.5 | -11.0 | — | |
| | | 5V | V _{OH} =0.9V _{DDn} , V _{DDn} =3V, n=0~1 SLEDCn[m+2:m+1:m]=011B, (n=0, 1, 2, 3, 4 or 5, m=0 or 3) | -1.2 | -2.5 | — | mA |
| | | 3V | V _{OH} =0.9V _{DDn} , V _{DDn} =V _{DD} , n=0~1 SLEDCn[m+2:m+1:m]=100B, (n=0, 1, 2, 3, 4 or 5, m=0 or 3) | -3.5 | -7.0 | — | |
| | | 5V | (n=0, 1, 2, 3, 4 or 5, m=0 or 3) | -7.0 | -14.0 | — | mA |
| | | 5V | V _{OH} =0.9V _{DDn} , V _{DDn} =3V, n=0~1 SLEDCn[m+2:m+1:m]=100B, (n=0, 1, 2, 3, 4 or 5, m=0 or 3) | -1.8 | -3.5 | — | |
| | | 3V | V _{OH} =0.9V _{DDn} , V _{DDn} =V _{DD} , n=0~1 SLEDCn[m+2:m+1:m]=101B, (n=0, 1, 2, 3, 4 or 5, m=0 or 3) | -4.0 | -8.5 | — | mA |
| | | 5V | (n=0, 1, 2, 3, 4 or 5, m=0 or 3) | -8.5 | -17.0 | — | |
| | | 5V | V _{OH} =0.9V _{DDn} , V _{DDn} =3V, n=0~1 SLEDCn[m+2:m+1:m]=101B, (n=0, 1, 2, 3, 4 or 5, m=0 or 3) | -2.0 | -4.0 | — | mA |
| | | 3V | V _{OH} =0.9V _{DDn} , V _{DDn} =V _{DD} , n=0~1 SLEDCn[m+2:m+1:m]=110B, (n=0, 1, 2, 3, 4 or 5, m=0 or 3) | -5 | -10 | — | |
| | | 5V | (n=0, 1, 2, 3, 4 or 5, m=0 or 3) | -10 | -20 | — | mA |
| | | 5V | V _{OH} =0.9V _{DDn} , V _{DDn} =3V, n=0~1 SLEDCn[m+2:m+1:m]=110B, (n=0, 1, 2, 3, 4 or 5, m=0 or 3) | -2.5 | -5.0 | — | |
| | | 3V | V _{OH} =0.9V _{DDn} , V _{DDn} =V _{DD} , n=0~1 SLEDCn[m+2:m+1:m]=111B, (n=0, 1, 2, 3, 4 or 5, m=0 or 3) | -5.5 | -11.0 | — | mA |
| | | 5V | (n=0, 1, 2, 3, 4 or 5, m=0 or 3) | -11 | -23 | — | |
| 5V | V _{OH} =0.9V _{DDn} , V _{DDn} =3V, n=0~1 SLEDCn[m+2:m+1:m]=111B, (n=0, 1, 2, 3, 4 or 5, m=0 or 3) | -2.8 | -5.5 | — | | | |
| R _{PH} | Pull-high Resistance for I/O Ports ⁽¹⁾ | 3V | V _{DDn} =V _{DD} , n=0~1, LVPU=0 | 20 | 60 | 100 | kΩ |
| | | 5V | V _{DDn} =V _{DD} , n=0~1, LVPU=0 | 10 | 30 | 50 | |
| | | 5V | V _{DDn} =3V, n=0~1, LVPU=0 | 36 | 110 | 180 | |
| | | 3V | V _{DDn} =V _{DD} , n=0~1, LVPU=0 | 6.67 | 15 | 23 | |
| | | 5V | V _{DDn} =V _{DD} , n=0~1, LVPU=0 | 3.5 | 7.5 | 12 | |
| | | 5V | V _{DDn} =3V, n=0~1, LVPU=0 | 9.0 | 27.5 | 45.0 | |
| I _{LEAK} | Input Leakage Current for I/O Ports | 5V | V _{IN} =V _{SS} or V _{IN} =V _{DDn} , n=0~1 | — | — | ±1 | μA |

Note: 1. The R_{PH} 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_{PH} value.

2. The actual V_{DDn} in the “Conditions” column, which can be V_{DD} or V_{DDIO}, is determined by the “V_{DD}” column value and the individual V_{DDn} voltage range.

Memory Electrical Characteristics

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

| Symbol | Parameter | Test Conditions | | Min. | Typ. | Max. | Unit |
|-----------------------------|--|-----------------|------------|------|------|------|------------------|
| | | V _{DD} | Conditions | | | | |
| Flash Program Memory | | | | | | | |
| V _{DD} | V _{DD} for Read | — | — | 1.8 | — | 5.5 | V |
| | V _{DD} for Erase and Write | — | — | 2.2 | — | 5.5 | V |
| t _{FER} | Erase Time | — | FWERTS=0 | — | 3.2 | 3.9 | ms |
| | | — | FWERTS=1 | — | 3.7 | 4.5 | ms |
| t _{FWR} | Write Time | — | FWERTS=0 | — | 2.2 | 2.7 | ms |
| | | — | FWERTS=1 | — | 3.0 | 3.6 | ms |
| I _{DDPGM} | Programming/Erase Current on V _{DD} | — | — | — | — | 5.0 | mA |
| E _P | Cell Endurance | — | — | 100K | — | — | E/W |
| t _{RETD} | ROM Data Retention Time | — | Ta=25°C | — | 40 | — | Year |
| t _{ACTV} | ROM Activation Time – Wake-up from IDLE/SLEEP Mode | — | — | 32 | — | 64 | µs |
| Data EEPROM Memory | | | | | | | |
| V _{DD} | V _{DD} for Read and Write | — | — | 1.8 | — | 5.5 | V |
| t _{EERD} | Read Time | — | — | — | — | 4 | t _{sys} |
| t _{EEWR} | Write Time (Byte Mode) | — | EWERTS=0 | — | 5.4 | 6.6 | ms |
| | | — | EWERTS=1 | — | 6.7 | 8.1 | |
| | Write Time (Page Mode) | — | EWERTS=0 | — | 2.2 | 2.7 | |
| | | — | EWERTS=1 | — | 3.0 | 3.6 | |
| t _{EEER} | Erase Time | — | EWERTS=0 | — | 3.2 | 3.9 | ms |
| | | — | EWERTS=1 | — | 3.7 | 4.5 | |
| E _P | Cell Endurance | — | — | 100K | — | — | E/W |
| t _{RETD} | Data Retention Time | — | Ta=25°C | — | 40 | — | Year |
| RAM Data Memory | | | | | | | |
| V _{DR} | RAM Data Retention Voltage | — | — | 1.0 | — | — | V |

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

2. The ROM activation time t_{ACTV} should be added when calculating the total system start-up time of a wake-up from the IDLE/SLEEP mode.

LVR/LVD Electrical Characteristics

Ta=-40°C~85°C

| Symbol | Parameter | Test Conditions | | Min. | Typ. | Max. | Unit |
|----------------------|--|-----------------|--|------|------|------|------|
| | | V _{DD} | Conditions | | | | |
| V _{LVR} | 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 _{LVD} | Low Voltage Detection 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 | | |
| | | — | LVD enable, voltage select 4.0V | | 4.0 | | |
| I _{LVR/LVD} | Operating Current | 3V | LVD enable, LVR enable, | — | — | 10 | μA |
| | | 5V | V _{LVR} =1.9V, V _{LVD} =2V | — | 10 | 15 | μA |
| t _{LVDS} | LVDO Stable Time | — | For LVR enable LVD off → on | — | — | 18 | μs |
| t _{LVR} | Minimum Low Voltage Width to Reset | — | TLVR[1:0]=00B | 120 | 240 | 480 | μs |
| | | | TLVR[1:0]=01B | 0.5 | 1 | 2 | ms |
| | | | TLVR[1:0]=10B | 1 | 2 | 4 | |
| | | | TLVR[1:0]=11B | 2 | 4 | 8 | |
| t _{LVD} | Minimum Low Voltage Width to Interrupt | — | TLVD[1:0]=00B/11B | 60 | 140 | 220 | μs |
| | | | TLVD[1:0]=01B | 90 | 200 | 340 | |
| | | | TLVD[1:0]=10B | 150 | 320 | 580 | |
| I _{LVR} | Additional Current for LVR Enable | 5V | LVD disable | — | — | 14 | μA |

Software Controlled LCD Driver Electrical Characteristics

Ta=-40°C~85°C

| Symbol | Parameter | Test Conditions | | Min. | Typ. | Max. | Unit |
|-------------------|--|-----------------|---------------|--------------------------|-------------------------|--------------------------|------|
| | | V _{DD} | Conditions | | | | |
| I _{BIAS} | LCD 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 _{SCOM} | V _{DD} ×2/3 Voltage for LCD SCOM Output | 2.2V~5.5V | No load | 0.305 V _{DD} | 0.33 V _{DD} | 0.355 V _{DD} | V |
| | V _{DD} ×1/3 Voltage for LCD SCOM Output | 2.2V~5.5V | No load | 0.31 V _{DD} | 0.33 V _{DD} | 0.35 V _{DD} | V |

A/D Converter Electrical Characteristics

Ta=-40°C~85°C

| Symbol | Parameter | Test Conditions | | Min. | Typ. | Max. | Unit |
|--------------------|---|-----------------|--|-------------------------|------|-------------------------|-------------------|
| | | V _{DD} | Conditions | | | | |
| V _{DD} | Operating Voltage | — | — | 1.8 | — | 5.5 | V |
| V _{ADI} | Input Voltage | — | — | 0 | — | V _{REF} | V |
| V _{REF} | Reference Voltage | — | — | 1.8 | — | V _{DD} | V |
| N _R | Resolution | — | — | — | — | 12 | Bit |
| DNL | Differential Nonlinearity | 1.8V | SAINS[3:0]=0000B, SAVRS[1:0]=01B, V _{REF} =V _{DD} , t _{ADCK} =2.0μs | -3 | — | 3 | LSB |
| | | 2V | SAINS[3:0]=0000B, SAVRS[1:0]=01B, V _{REF} =V _{DD} , t _{ADCK} =0.5μs | | | | |
| | | 3V | SAINS[3:0]=0000B, SAVRS[1:0]=01B, V _{REF} =V _{DD} , t _{ADCK} =0.5μs | | | | |
| | | 5V | SAINS[3:0]=0000B, SAVRS[1:0]=01B, V _{REF} =V _{DD} , t _{ADCK} =0.5μs | | | | |
| | | 1.8V | SAINS[3:0]=0000B, SAVRS[1:0]=01B, V _{REF} =V _{DD} , t _{ADCK} =10μs | | | | |
| | | 3V | SAINS[3:0]=0000B, SAVRS[1:0]=01B, V _{REF} =V _{DD} , t _{ADCK} =10μs | | | | |
| INL | Integral Nonlinearity | 1.8V | SAINS[3:0]=0000B, SAVRS[1:0]=01B, V _{REF} =V _{DD} , t _{ADCK} =2.0μs | -4 | — | 4 | LSB |
| | | 2V | SAINS[3:0]=0000B, SAVRS[1:0]=01B, V _{REF} =V _{DD} , t _{ADCK} =0.5μs | | | | |
| | | 3V | SAINS[3:0]=0000B, SAVRS[1:0]=01B, V _{REF} =V _{DD} , t _{ADCK} =0.5μs | | | | |
| | | 5V | SAINS[3:0]=0000B, SAVRS[1:0]=01B, V _{REF} =V _{DD} , t _{ADCK} =0.5μs | | | | |
| | | 1.8V | SAINS[3:0]=0000B, SAVRS[1:0]=01B, V _{REF} =V _{DD} , t _{ADCK} =10μs | | | | |
| | | 3V | SAINS[3:0]=0000B, SAVRS[1:0]=01B, V _{REF} =V _{DD} , t _{ADCK} =10μs | | | | |
| I _{ADC} | Additional Current for ADC Enable | 1.8V | No load, t _{ADCK} =2.0μs | — | 280 | 400 | μA |
| | | 3V | No load, t _{ADCK} =0.5μs | — | 450 | 600 | μA |
| | | 5V | No load, t _{ADCK} =0.5μs | — | 850 | 1000 | μA |
| t _{ADCK} | Clock Period | — | 1.8V≤V _{DD} <2.0V | 2.0 | — | 10.0 | μs |
| | | — | 2.0V≤V _{DD} ≤5.5V | 0.5 | — | 10.0 | μs |
| t _{ON2ST} | ADC On to ADC Start | — | — | 4 | — | — | μs |
| t _{ADS} | Sampling Time | — | — | — | 4 | — | t _{ADCK} |
| t _{ADC} | Conversion Time (Include ADC Sample and Hold Time) | — | — | — | 16 | — | t _{ADCK} |
| GERR | A/D Conversion Gain Error | 1.8V | SAINS[3:0]=0000B, SAVRS[1:0]=01B, V _{REF} =V _{DD} | -4 | — | 4 | LSB |
| | | 3V | SAINS[3:0]=0000B, SAVRS[1:0]=01B, V _{REF} =V _{DD} | | | | |
| | | 5V | SAINS[3:0]=0000B, SAVRS[1:0]=01B, V _{REF} =V _{DD} | | | | |
| OSRR | A/D Conversion Offset Error | 1.8V | SAINS[3:0]=0000B, SAVRS[1:0]=01B, V _{REF} =V _{DD} | -4 | — | 4 | LSB |
| | | 3V | SAINS[3:0]=0000B, SAVRS[1:0]=01B, V _{REF} =V _{DD} | | | | |
| | | 5V | SAINS[3:0]=0000B, SAVRS[1:0]=01B, V _{REF} =V _{DD} | | | | |
| I _{PGA} | Additional Current for PGA Enable | 2.2V | — | — | 500 | 1000 | μA |
| | | 3V | No load, PGAIS=1, PGAGS[1:0]=01 | — | 600 | 1200 | μA |
| | | 5V | No load, PGAIS=1, PGAGS[1:0]=01 | — | 700 | 1400 | μA |
| V _{OR} | PGA Maximum Output Voltage Range | 2.2V | — | V _{SS} +0.1 | — | V _{DD} -0.1 | V |
| | | 3V | — | | | | |
| | | 5V | — | | | | |

| Symbol | Parameter | Test Conditions | | Min. | Typ. | Max. | Unit |
|-----------------|---------------------------|-----------------|---|-------------------------|------|-------------------------|------|
| | | V _{DD} | Conditions | | | | |
| V _{VR} | Fix Voltage Output of PGA | 2.2V ~5.5V | V _{RI} =V _{BGREF} , PGAIS=1 | -1% | 2 | +1% | V |
| | | 3.2V ~5.5V | | -1% | 3 | +1% | V |
| | | 4.2V ~5.5V | | -1% | 4 | +1% | V |
| V _{IR} | PGA Input Voltage Range | 3V | Gain=1, PGAIS=0 Relative gain, Gain error<±5% | V _{SS} +0.1 | — | V _{DD} -1.4 | V |
| | | 5V | | V | | | |

Internal Reference Voltage Electrical Characteristics

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

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

- Note: 1. All the above parameters are measured under conditions of no load condition unless otherwise described.
 2. A 0.1μF ceramic capacitor should be connected between V_{DD} and GND.
 3. The V_{BGREF} voltage is used as the A/D converter PGA reference voltage input.

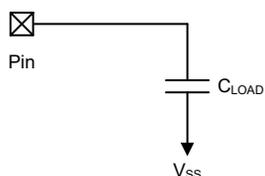
OCP Electrical Characteristics

T_a=-40°C~85°C

| Symbol | Parameter | Test Conditions | | Min. | Typ. | Max. | Unit |
|---------------------|---------------------------------|-----------------|--|------|------|-----------------|------|
| | | V _{DD} | Conditions | | | | |
| V _{DD} | Operating Voltage | — | — | 1.8 | — | 5.5 | V |
| I _{OCP} | Operating Current | 3V | OCPEN=1, DACEN=1, OCPVRS=0, G[3:0]=1000 | — | 0.8 | 1.2 | mA |
| | | 5V | — | — | 1.15 | 1.50 | |
| V _{REF} | DAC Reference Voltage | 3V | OCPVRS=0 | 1.8 | — | V _{DD} | V |
| | | 5V | | 1.8 | — | V _{DD} | |
| V _{OS_CMP} | Comparator Input Offset Voltage | 3V | Without calibration (OCPCOF[4:0]=10000B) | -15 | — | 15 | mV |
| | | 5V | | -15 | — | 15 | |
| | | 3V | With calibration | -2 | — | 2 | mV |
| | | 5V | | -2 | — | 2 | |
| V _{HYS} | Hysteresis | 3V | — | 10 | — | 45 | mV |
| | | 5V | — | 10 | 25 | 45 | |

| Symbol | Parameter | Test Conditions | | Min. | Typ. | Max. | Unit |
|---------------------|--------------------------------------|-----------------|--|----------------------|------|----------------------|------|
| | | V _{DD} | Conditions | | | | |
| V _{CM_CMP} | Comparator Common Mode Voltage Range | 3V | — | V _{SS} | — | V _{DD} -1.0 | V |
| | | 5V | — | V _{SS} | — | V _{DD} -1.0 | V |
| t _{CMP_RP} | CMP Response Time | 3V | With 100mV overdrive (Note) | — | 0.3 | 1 | μs |
| | | 5V | | — | 0.3 | 1 | μs |
| V _{OS_OPA} | OPA Input Offset Voltage | 3V | Without calibration (OCPOOF [5:0]=100000B) | -15 | — | 15 | mV |
| | | 5V | | -15 | — | 15 | mV |
| | | 3V | With calibration | -2 | — | 2 | mV |
| | | 5V | | -2 | — | 2 | mV |
| V _{CM_OPA} | OPA Common Mode Voltage Range | 3V | — | V _{SS} | — | V _{DD} -1.4 | V |
| | | 5V | — | V _{SS} | — | V _{DD} -1.4 | V |
| V _{OR} | OPA Maximum Output Voltage Range | 3V | — | V _{SS} +0.1 | — | V _{DD} -0.1 | V |
| | | 5V | — | V _{SS} +0.1 | — | V _{DD} -0.1 | V |
| Ga | PGA Gain Accuracy | 3V | S0=S3=S4=ON, S1=S2=OFF, S5=S6=OFF, All gain | -15 | — | 15 | % |
| | | 5V | S0=S3=S4=ON, S1=S2=OFF, S5=S6=OFF, All gain | -15 | — | 15 | % |
| Ga2 | PGDA Gain Accuracy | 3V | S0=S1=OFF, S2=S3=S4=ON, S5=S6=OFF, DAC OUT=1/2V _{DD} , All gain | -15 | — | 15 | % |
| | | 5V | S0=S1=OFF, S2=S3=S4=ON, S5=S6=OFF, DAC OUT=1/2V _{DD} , All gain | -15 | — | 15 | % |
| DNL | Differential Nonlinearity | 3V | DAC V _{REF} =AV _{DD} | — | — | ±1.5 | LSB |
| | | 5V | | — | — | ±1 | LSB |
| INL | Integral Nonlinearity | 3V | DAC V _{REF} =AV _{DD} | — | — | ±2 | LSB |
| | | 5V | DAC V _{REF} =AV _{DD} | — | — | ±1.5 | LSB |
| I _{DACOL} | Output Sink Current | 3V | Code=00H, V _{DACO} =0.1V _{REF} | 1 | — | — | mA |
| | | 5V | Code=00H, V _{DACO} =0.1V _{REF} | 2 | — | — | mA |
| I _{DACOH} | Output Source Current | 3V | Code=FFH, V _{DACO} =0.9V _{REF} | 0.4 | — | — | mA |
| | | 5V | Code=FFH, V _{DACO} =0.9V _{REF} | 1 | — | — | mA |
| I _{SC_DAC} | Output Short-circuit Current | 3V | Code=FFH | 2 | — | — | mA |
| | | 5V | Code=FFH | 6 | — | — | mA |

Note: Load Condition: C_{LOAD}=50pF.

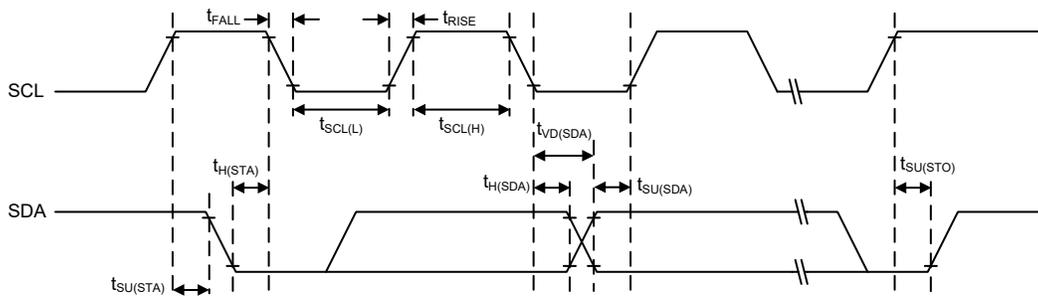


I²C Electrical Characteristics

Ta=25°C

| Symbol | Parameter | Test Conditions | | Min. | Typ. | Max. | Unit |
|----------------------|--|-----------------|-------------------------|------|------|------|------|
| | | V _{DD} | Conditions | | | | |
| f _{I2C} | I ² C Standard Mode (100kHz) f _{sys} Frequency <small>(Note)</small> | — | No clock debounce | 2 | — | — | MHz |
| | | | 2 system clock debounce | 4 | — | — | |
| | | | 4 system clock debounce | 4 | — | — | |
| | I ² C Fast Mode (400kHz) f _{sys} Frequency <small>(Note)</small> | — | No clock debounce | 4 | — | — | MHz |
| | | | 2 system clock debounce | 8 | — | — | |
| | | | 4 system clock debounce | 8 | — | — | |
| f _{SCL} | SCL Clock Frequency | 3V/5V | Standard mode | — | — | 100 | kHz |
| | | | Fast mode | — | — | 400 | |
| t _{SCL(H)} | SCL Clock High Time | 3V/5V | Standard mode | 3.5 | — | — | μs |
| | | | Fast mode | 0.9 | — | — | |
| t _{SCL(L)} | SCL Clock Low Time | 3V/5V | Standard mode | 3.5 | — | — | μs |
| | | | Fast mode | 0.9 | — | — | |
| t _{FALL} | SCL and SDA Fall Time | 3V/5V | Standard mode | — | — | 1.3 | μs |
| | | | Fast mode | — | — | 0.34 | |
| t _{RISE} | SCL and SDA Rise Time | 3V/5V | Standard mode | — | — | 1.3 | μs |
| | | | Fast mode | — | — | 0.34 | |
| t _{SU(SDA)} | SDA Data Setup Time | 3V/5V | Standard mode | 0.25 | — | — | μs |
| | | | Fast mode | 0.1 | — | — | |
| t _{H(SDA)} | SDA Data Hold Time | 3V/5V | — | 0.1 | — | — | μs |
| t _{VD(SDA)} | SDA Data Valid Time | 3V/5V | — | — | — | 0.6 | μs |
| t _{SU(STA)} | Start Condition Setup Time | 3V/5V | Standard mode | 3.5 | — | — | μs |
| | | | Fast mode | 0.6 | — | — | |
| t _{H(STA)} | Start Condition Hold Time | 3V/5V | Standard mode | 4.0 | — | — | μs |
| | | | Fast mode | 0.6 | — | — | |
| t _{SU(STO)} | 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.

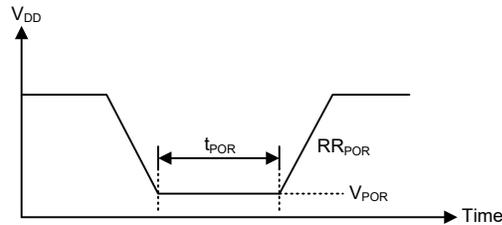


I²C Timing Diagram

Power-on Reset Characteristics

Ta=25°C

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

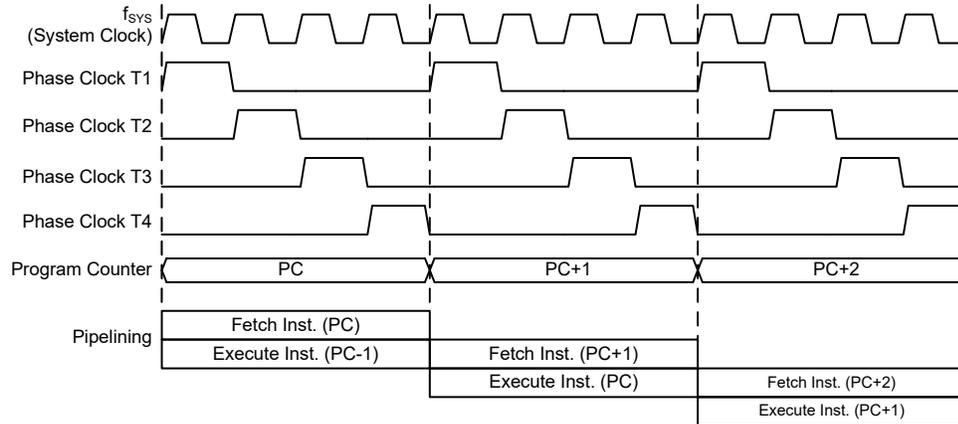


System Architecture

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, with the exception of branch or call instructions which need one more cycle. An 8-bit wide ALU is used in practically all instruction set operations, which carries out arithmetic operations, logic operations, rotation, increment, decrement, branch decisions, etc. The internal data path is simplified by moving data through the Accumulator and the ALU. Certain internal registers are implemented in the Data Memory and can be directly or indirectly addressed. The simple addressing methods of these registers along with additional architectural features ensure that a minimum of external components is required to provide a functional I/O and A/D control system with maximum reliability and flexibility. This makes the device suitable for low-cost, high-volume production for controller applications.

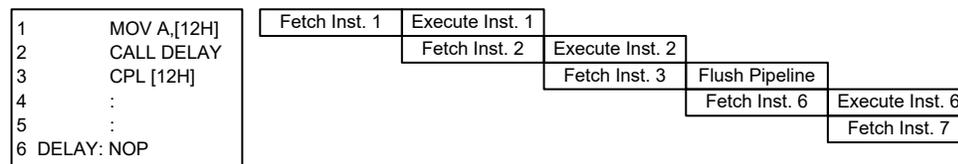
Clocking and Pipelining

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



System Clocking and Pipelining

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



Instruction Fetching

Program Counter

During program execution, the Program Counter is used to keep track of the address of the next instruction to be executed. It is automatically incremented by one each time an instruction is executed except for instructions, such as “JMP” or “CALL” that demand 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 | |
|---------------------------|--------------|
| Program Counter High Byte | PCL Register |
| PBP0, PC12~PC8 | PCL7~PCL0 |

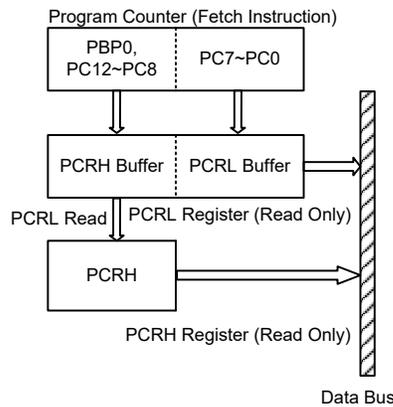
Program Counter

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

Program Counter Read Registers

The Program Counter Read registers are a read only register pair for reading the program counter value which indicates the current program execution address. Read the low byte register first then the high byte register. Reading the low byte register, PCRL, will read the low byte data of the current program execution address, and place the high byte data of the program counter into the 8-bit PCRH buffer. Then reading the PCRH register will read the corresponding data from the 8-bit PCRH buffer. The following example shows how to read the current program execution address. When the current program execution address is 123H, the steps to execute the instructions are as follows:

- (1) MOV A, PCRL → the ACC value is 23H, and the PCRH value is 01H;
 MOV A, PCRH → the ACC value is 01H.
- (2) LMOV A, PCRL → the ACC value is 23H, and the PCRH value is 01H;
 LMOV A, PCRH → the ACC value is 01H.



• PCRL 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:** Program Counter Read Low byte register bit 7 ~ bit 0

• PCRH Register

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

Bit 7~6 Unimplemented, read as “0”

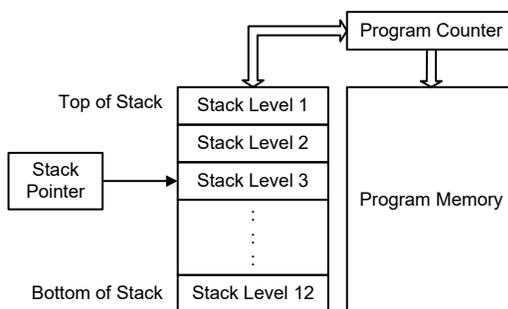
Bit 5~0 **D13~D8:** Program Counter Read High byte register bit 5 ~ bit 0

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, STKPTR, which is a read only register. 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.



• STKPTR Register

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

Bit 7 **OSF**: Stack overflow flag
The OSF bit will be set high, when the stack is full and a CALL subroutine instruction is executed, or when the stack is empty and a RET instruction is executed. The OSF bit can only be cleared to zero by the software.

Bit 6~4 Unimplemented, read as “0”

Bit 3~0 **D3~D0**: Stack pointer register bit 3 ~ bit 0

The following example shows how the Stack Pointer and Stack Overflow Flag change when program branching conditions occur.

- (1) When the CALL subroutine instruction is executed 13 times continuously and the RET instruction is not executed during the period, the corresponding changes of the STKPTR[3:0] and OSF bits are as follow:

| CALL Execution Times | 0 | 1 | 2 | 3 | 4 | 5 | 6 | ... | 10 | 11 | 12 | 13 |
|-----------------------|---|---|---|---|---|---|---|-----|----|----|----|----|
| STKPTR[3:0] Bit Value | 0 | 1 | 2 | 3 | 4 | 5 | 6 | ... | 10 | 11 | 0 | 1 |
| OSF Bit Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

- (2) When the OSF bit is set high and not cleared, it will remain high no matter how many times the RET instruction is executed.

(3) When the stack is empty, the RET instruction is executed 12 times continuously, the corresponding changes of the STKPTR[3:0] and OSF bits are as follow:

| RET Execution Times | 0 | 1 | 2 | 3 | 4 | 5 | 6 | ... | 10 | 11 | 12 |
|-----------------------|---|----|----|---|---|---|---|-----|----|----|----|
| STKPTR[3:0] Bit Value | 0 | 11 | 10 | 9 | 8 | 7 | 6 | ... | 2 | 1 | 0 |
| OSF Bit Value | 0 | 1 | 1 | 1 | 1 | 1 | 1 | | 1 | 1 | 1 |

Arithmetic and Logic Unit – ALU

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

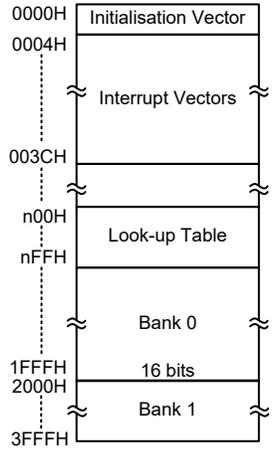
- Arithmetic operations:
 ADD, ADDM, ADC, ADCM, SUB, SUBM, SBC, SBCM, DAA,
 LADD, LADDM, LADC, LADCM, LSUB, LSUBM, LSBC, LSBCM, LDAA
- Logic operations:
 AND, OR, XOR, ANDM, ORM, XORM, CPL, CPLA,
 LAND, LANDM, LOR, LORM, LXOR, LXORM, LCPL, LCPLA
- Rotation:
 RRA, RR, RRCA, RRC, RLA, RL, RLCA, RLC,
 LRR,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

Flash Program Memory

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

Structure

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



Program Memory Structure

Special Vectors

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

Look-up Table

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

After setting up the table pointer, the table data can be retrieved from the Program Memory using the “TABRD [m]” or “TABRDL [m]” instructions 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.

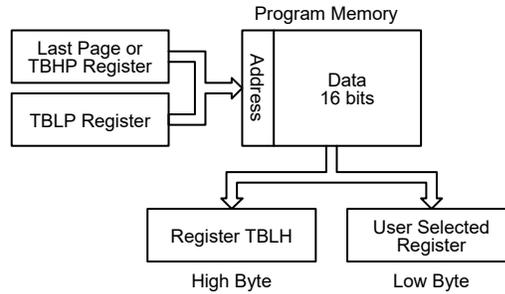


Table Program Example

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. The table pointer low byte register is setup here to have an initial value of “06H”. This will ensure that the first data read from the data table will be at the Program Memory address “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 TBHP and TBLP 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
code0 .section 'code'
mov a,06h         ; initialise 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        ; it is not necessary to set tbhp if executing tabrdl or ltabrdl
:
:
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 tempreg2 the value "00H" will be transferred to the high
                  ; byte register TBLH
:
:
code1 .section 'code'
org 1F00h         ; sets initial address of last page
dc 00Ah,00Bh,00Ch,00Dh,00Eh,00Fh,01Ah,01Bh
```

In Circuit Programming – ICP

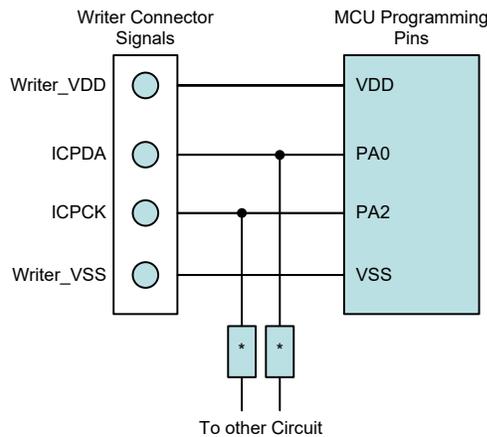
The provision of Flash type Program Memory provides the user with a means of convenient and easy upgrades and modifications to their programs on the same device.

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

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

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

During the programming process, the user must 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

An EV chip exists for the purposes of device emulation. This EV chip device also provides an “On-Chip Debug” function to debug the real MCU device during the development process. The EV chip and the real MCU device are almost functionally compatible except for “On-Chip Debug” function. Users can use the EV chip device to emulate the real chip device behavior by connecting the OCSDA and OCDSCK pins to the Holtek HT-IDE development tools. The OCSDA 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 OCSDA and OCDSCK pins in the device will have no effect in the EV chip. 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 |
|--------------------|--------------|---|
| OCSDA | OCSDA | On-Chip Debug Support Data/Address input/output |
| OCDSCK | OCDSCK | On-Chip Debug Support Clock input |
| VDD | VDD | Power Supply |
| VSS | VSS | 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.

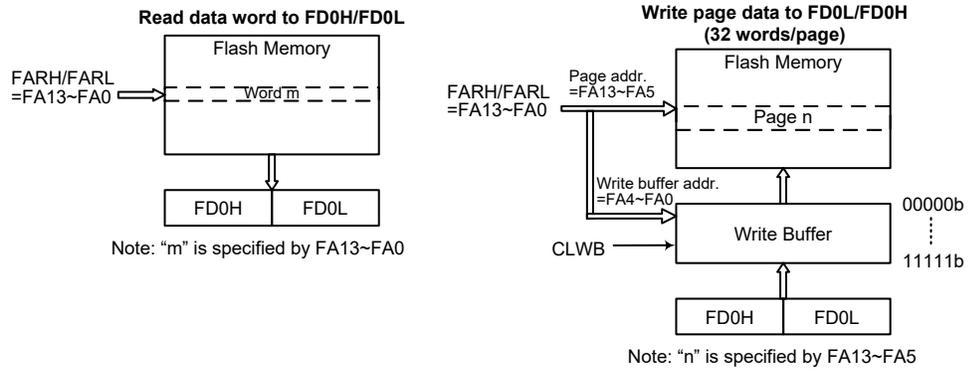
| Operations | Format |
|------------|---------------|
| Erase | 32 words/page |
| Write | 32 words/time |
| Read | 1 word/time |

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 | |
| : | : | : | |
| : | : | : | |
| 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 zero by 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 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 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, which are all located in Sector 1. Read and Write operations to the Flash memory are carried out using 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. As all the registers are located in Sector 1, they can be addressed directly using the corresponding extended instructions or can be read from or written to indirectly using the MP1H/MP1L or MP2H/MP2L Memory Pointer pairs and Indirect Addressing Register, IAR1 or IAR2.

| 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

• **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 0 by application program, the Flash memory erase/write function is disabled. Note that this bit cannot be set high by application programs. Writing a “1” into this bit results in no action. This bit is used to indicate the Flash memory erase/write function status. When this bit is set to 1 by hardware, it means that the Flash memory erase/write function is enabled successfully. Otherwise, the Flash memory erase/write function is disabled if the bit 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 by 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 Flash memory write process
 This bit is set by software and cleared by hardware when the Flash memory write process has completed.
- Bit 1 **FRDEN**: Flash memory read enable control
 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 initiate control
 0: Do not initiate Flash memory read or indicating that a Flash memory read process has completed
 1: Initiate Flash memory read process
 This bit is set by software and cleared by 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, write or erase 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 Write Buffer Clear process
 This bit is set by software and cleared by hardware when the Write Buffer Clear process has completed.

• **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 word 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 word bit 15 ~ bit 8

Note that when 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 word 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 word 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 word 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 **D15~D8**: The third Flash Memory data word 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 word 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 word bit 15 ~ bit 8

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.

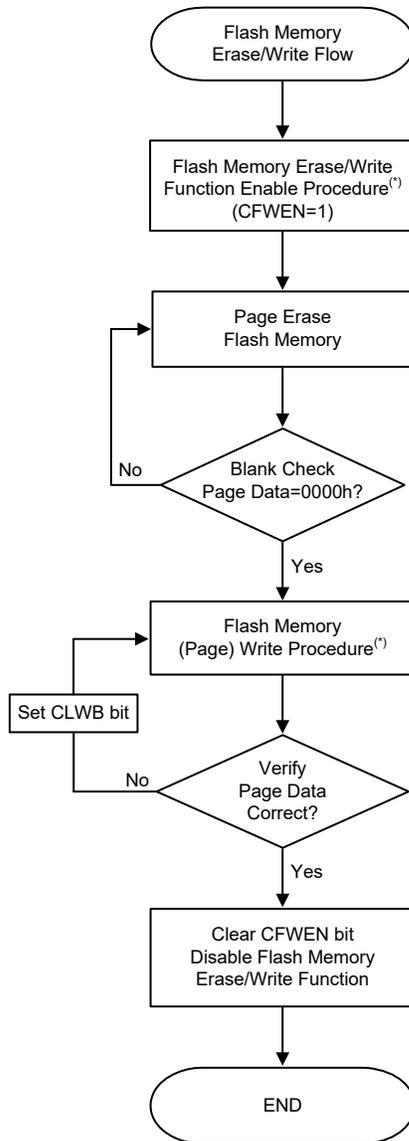
Flash Memory Erase/Write Flow Descriptions

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 complete if 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.

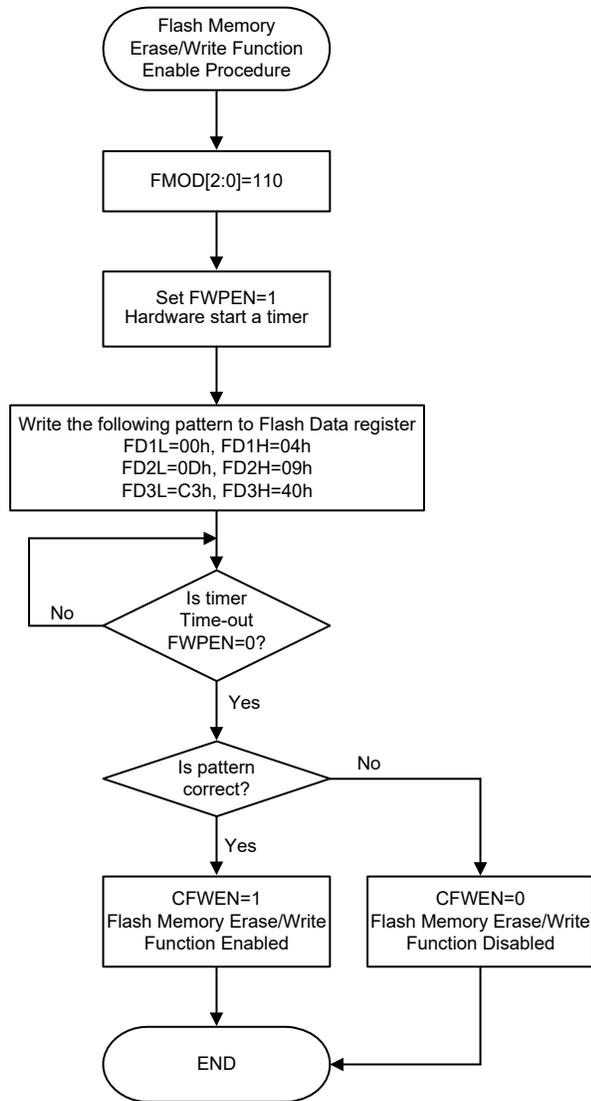
Flash Memory Erase/Write Function Enable Procedure

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 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 enable Flash memory erase/write function data pattern 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 0 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

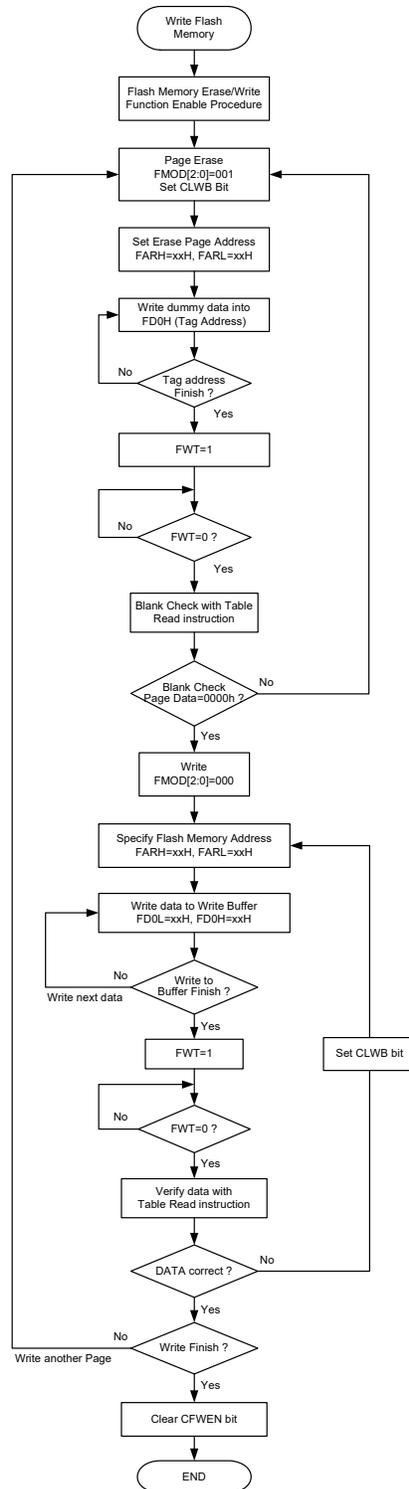
After the Flash memory erase/write function has been successfully enabled as the CFWEN bit is set high, the data to be written into the Flash memory can be loaded into the write buffer. The selected Flash memory page data should be erased by properly configuring the IAP control registers before the data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

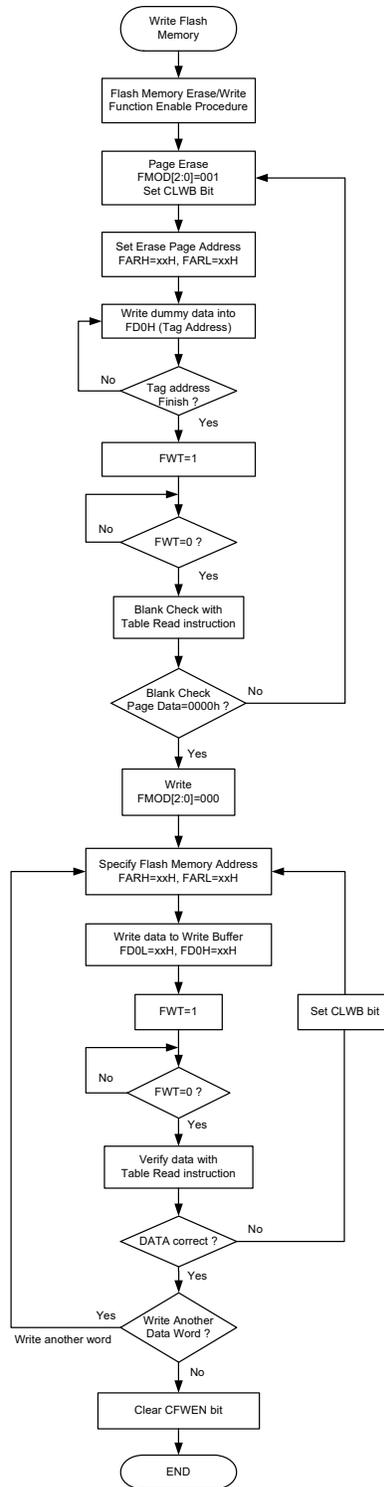
- Note: 1. When the erase or write operation is successfully activated, all CPU operations will temporarily cease.
 2. It will take certain time for the FWT bit state changing from high to low in the erase or write operation, which can be selected by the FWERTS bit in the FC2 register.

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 FMOD field 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 FMOD field to “000” to select the write operation.
5. Setup the desired address ADDR1 in the FARH and FRARL 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 FRARL 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

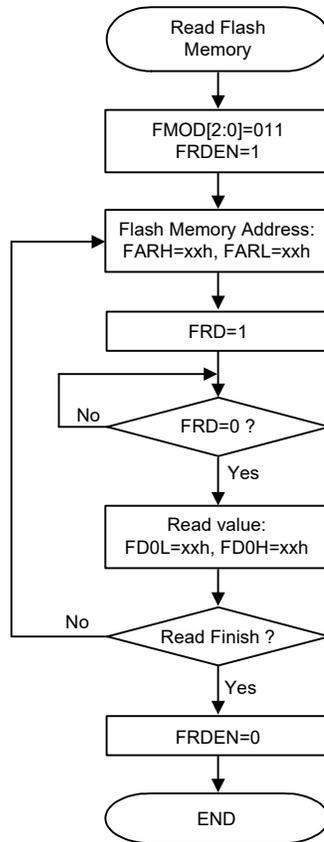
- Note: 1. When the erase or write operation is successfully activated, all CPU operations will temporarily cease.
 2. It will take certain time for the FWT bit state changing from high to low in the erase or write operation, which can be selected by the FWERTS bit in the FC2 register.

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 writing 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, FDOH and FDOL, 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.

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

The device also provides dedicated memory areas for the Touch Key data storage. In this chapter, only the General Purpose Data Memory and the Special Function Register Data Memory are introduced. More information about the Touch Key Data Memory can be obtained in its corresponding chapter.

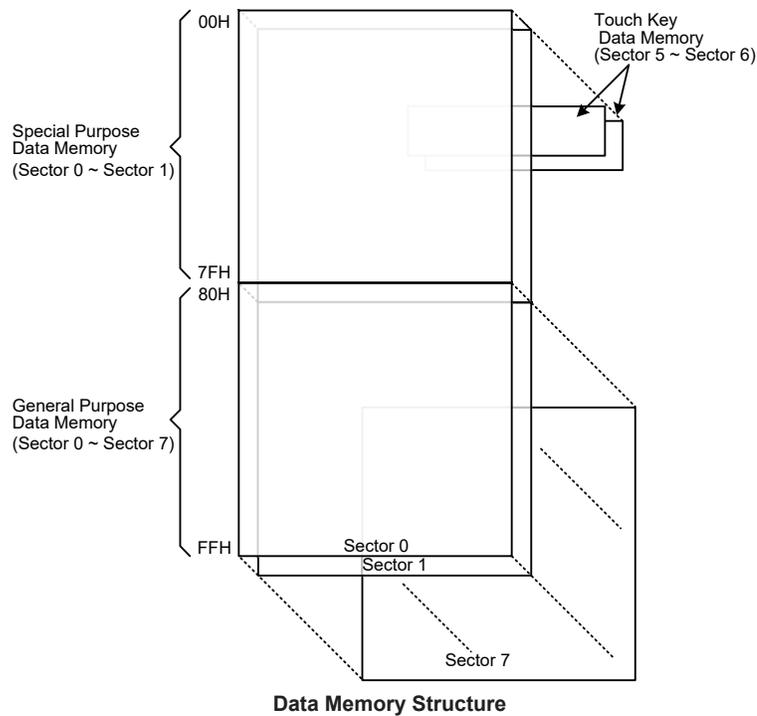
Structure

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.

Switching between the different Data Memory sectors is achieved by properly setting the Memory Pointers to correct value if using the indirect addressing method.

| Special Purpose Data Memory | General Purpose Data Memory | | Touch Key Data Memory | |
|-----------------------------|-----------------------------|--|-----------------------|--|
| Located Sectors | Capacity | Sector: Address | Capacity | Sector: Address |
| 0, 1 | 1024×8 | Sector 0: 80H~FFH Sector 1: 80H~FFH : Sector 7: 80H~FFH | 80×8 | Sector 5: 00H~27H Sector 6: 00H~27H |

Data Memory Summary



Data Memory Addressing

For the device that supports the extended instructions, there is no Bank Pointer for Data Memory addressing. 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 extended instructions 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 for this device, the high byte indicates a sector and the low byte indicates a specific address.

General Purpose Data Memory

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

Special Purpose Data Memory

This area of Data Memory is where registers, necessary for the correct operation of the microcontroller, are stored. Most of the registers are both readable and writable 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 | CTM0C0 | 40H | WDTC | EEC |
| 01H | MP0 | CTM0C1 | 41H | EEAL | |
| 02H | IAR1 | CTM0DL | 42H | EEAH | FC0 |
| 03H | MP1L | CTM0DH | 43H | EED | FC1 |
| 04H | MP1H | CTM0AL | 44H | TKTMR | FC2 |
| 05H | ACC | CTM0AH | 45H | TKC0 | FARL |
| 06H | PCL | CTM1C0 | 46H | TKC1 | FARH |
| 07H | TBLP | CTM1C1 | 47H | TK16DL | FD0L |
| 08H | TBLH | CTM1DL | 48H | TK16DH | FD0H |
| 09H | TBHP | CTM1DH | 49H | TKM0C0 | FD1L |
| 0AH | STATUS | CTM1AL | 4AH | TKM0C1 | FD1H |
| 0BH | PBP | CTM1AH | 4BH | TKM0C2 | FD2L |
| 0CH | IAR2 | PTM0C0 | 4CH | TKM016DL | FD2H |
| 0DH | MP2L | PTM0C1 | 4DH | TKM016DH | FD3L |
| 0EH | MP2H | PTM0DL | 4EH | TKM0ROL | FD3H |
| 0FH | RSTFC | PTM0DH | 4FH | TKM0ROH | IECC |
| 10H | INTC0 | PTM0AL | 50H | TKM1C0 | PCRL |
| 11H | INTC1 | PTM0AH | 51H | TKM1C1 | PCRH |
| 12H | INTC2 | PTM0RPL | 52H | TKM1C2 | OCPC0 |
| 13H | INTC3 | PTM0RPH | 53H | TKM116DL | OCPC1 |
| 14H | PA | | 54H | TKM116DH | OCPDA |
| 15H | PAC | PAS0 | 55H | TKM1ROL | OCPOCAL |
| 16H | PAPU | PAS1 | 56H | TKM1ROH | OCPCCAL |
| 17H | PAWU | PAS2 | 57H | TKM2C0 | |
| 18H | | PAS3 | 58H | TKM2C1 | |
| 19H | MFI0 | PBS0 | 59H | TKM2C2 | |
| 1AH | MFI1 | PBS1 | 5AH | TKM216DL | |
| 1BH | TB0C | PCS0 | 5BH | TKM216DH | |
| 1CH | PSC0R | PCS1 | 5CH | TKM2ROL | |
| 1DH | LVRC | PDS0 | 5DH | TKM2ROH | |
| 1EH | LVPUC | PDS1 | 5EH | TKM3C0 | |
| 1FH | LXTC | PES0 | 5FH | TKM3C1 | |
| 20H | | PES1 | 60H | TKM3C2 | |
| 21H | PSC1R | PFS0 | 61H | TKM316DL | |
| 22H | TB1C | PFS1 | 62H | TKM316DH | |
| 23H | SADC0 | PCNS | 63H | TKM3ROL | |
| 24H | SADC1 | PDNS | 64H | TKM3ROH | |
| 25H | SADC2 | PFNS | 65H | TKM4C0 | |
| 26H | SADOL | PMPS | 66H | TKM4C1 | |
| 27H | SADOH | PTM1C0 | 67H | TKM4C2 | |
| 28H | SCC | PTM1C1 | 68H | TKM416DL | |
| 29H | HIRCC | PTM1DL | 69H | TKM416DH | |
| 2AH | USR | PTM1DH | 6AH | TKM4ROL | |
| 2BH | UCR1 | PTM1AL | 6BH | TKM4ROH | |
| 2CH | UCR2 | PTM1AH | 6CH | PB | |
| 2DH | UCR3 | PTM1RPL | 6DH | PBC | |
| 2EH | BRDH | PTM1RPH | 6EH | PBPU | |
| 2FH | BRDL | SLCDC0 | 6FH | PC | |
| 30H | UFCR | SLCDS0 | 70H | PCC | |
| 31H | TXR_RXR | SLCDS1 | 71H | PCPU | |
| 32H | RxCNT | SLCDS2 | 72H | PD | |
| 33H | IFS0 | SLCDS3 | 73H | PDC | |
| 34H | IFS1 | CRCCR | 74H | PDPU | |
| 35H | IFS2 | CRCIN | 75H | PE | |
| 36H | LVDC | CRCDL | 76H | PEC | |
| 37H | VBGRC | CRCDH | 77H | PEPU | |
| 38H | SIMC0 | SLEDC0 | 78H | PF | |
| 39H | SIMC1/UUCR1 | SLEDC1 | 79H | PFC | |
| 3AH | SIMD/UTXR_RXR | SLEDC2 | 7AH | PFPU | |
| 3BH | SIMA/SIMC2/UUCR2 | SLEDC3 | 7BH | | |
| 3CH | UUCR3 | SLEDC4 | 7CH | | |
| 3DH | SIMTOC/UBRG | SLEDC5 | 7DH | | |
| 3EH | UUSR | TLVRC | 7EH | | |
| 3FH | INTEG | STKPTR | 7FH | ORMC | |

 : Unused, read as 00H

Special Purpose Data Memory Structure

Special Function Register Description

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

Indirect Addressing Registers – IAR0, IAR1, 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 only 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 and MP2H, are provided. These Memory Pointers are physically implemented in the Data Memory and can be manipulated in the same way as normal registers providing a convenient way with which to address and track data. When any operation to the relevant Indirect Addressing Registers is carried out, the actual address that the microcontrollers 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 data sectors according to the corresponding MP1H or MP2H register. Direct Addressing can be used in all data 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 mplh,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 RAM 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 the 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 bit 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 Point bit 0
 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. The TBLP and TBHP registers are the table pointer pair and indicates the location where the table data is located. Their value must be setup before any table read instructions 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 zero flag (Z), carry flag (C), auxiliary carry flag (AC), overflow flag (OV), SC flag, CZ flag, power down flag (PDF), and watchdog time-out flag (TO). These arithmetic/logical operation and system management flags are used to record the status and operation of the microcontrollers.

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

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

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

- OV is set if an operation results in a carry into the highest-order bit but not a carry out of the highest-order bit, or vice versa; otherwise OV is cleared.
- PDF is cleared by a system power-up or executing the “CLR WDT” instruction. PDF is set by executing the “HALT” instruction.
- TO is cleared by a system power-up or executing the “CLR WDT” or “HALT” instruction. TO is set by a WDT time-out.
- SC is the result of the “XOR” operation which is performed by the OV flag and the MSB of the current instruction operation result.
- CZ is the operational result of different flags for different instructions. Refer to register definitions for more details.

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

• **STATUS Register**

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

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

EEPROM Data Memory Structure

The EEPROM Data Memory capacity is 1024×8 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 either the byte mode or page mode determined by the mode selection bit, MODE, in the control register, EEC.

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 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 MP1H/MP1L or MP2H/MP2L Memory Pointer pair and Indirect Addressing Register, IAR1 or IAR2. Because the EEC control register is located at address 40H in Sector 1, the Memory Pointer low byte register, MP1L or MP2L, must first be set to the value 40H and the Memory Pointer high byte register, MP1H or MP2H, set to the value, 01H, before any operations on the EEC register are executed.

| Register Name | Bit | | | | | | | |
|---------------|--------|-------|-------|-------|-------|-------|-------|-------|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| EEAL | EEAL7 | EEAL6 | EEAL5 | EEAL4 | EEAL3 | EEAL2 | EEAL1 | EEAL0 |
| EEAH | — | — | — | — | — | — | EEAH1 | EEAH0 |
| EED | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| 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 ~ bit 0

• **EED Register**

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

Bit 7~0 **D7~D0**: 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**: Data 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 Data EEPROM erase operations are carried out. This bit will be automatically reset to zero by 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 an erase cycle

This is the Data EEPROM Erase Control Bit. 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 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. When this bit is set high by the application program, a write cycle will be activated. This bit will be automatically reset to zero by 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**: Data EEPROM read control
0: Read cycle has finished
1: Activate a read cycle

This is the Data EEPROM Read Control Bit. When this bit is set high by the application program, a read cycle will be activated. This bit will be automatically reset to zero by 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 to “1” at the same time in one instruction. The WR and RD cannot be set to “1” at the same time.

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 to zero indicating that the EEPROM data can be read from the EED register and then the current address will be incremented by one by hardware. The data which is stored in the next EEPROM address can continuously be read 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 read 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, known as 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 should be 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 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 cleared to zero 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 EEAH and EEAL 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 only will not initiate a write cycle if the WREN bit is not 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 cleared to zero by hardware. 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, known as 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 start address of the desired EEPROM page should first be placed in the EEAH and EEAL registers and the data to be written should be 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 WREN bit in the EEC register should be set high to enable write operations and the WR bit must be immediately set high to initiate the EEPROM write process. These two instructions must be executed in two consecutive instruction cycles to activate a write operation successfully. The global interrupt enable 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 only will not initiate a write cycle if the WREN bit is not 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 cleared to zero 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. When an EEPROM erase or write cycle ends, the DEF request flag will be set high. If the global and EEPROM interrupts are enabled and the stack is not full, a jump to the EEPROM Interrupt vector will take place. When the interrupt is serviced, the EEPROM interrupt flag will be automatically reset. More details can be obtained in the Interrupt section.

Programming Considerations

Care must be taken that data is not inadvertently written to the EEPROM. Protection can be enhanced by ensuring that the Write Enable bit is normally cleared to zero when not writing. Also the 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 data the WR bit must be set high immediately after the WREN bit has been set high, to ensure the write cycle executes correctly. The global interrupt bit EMI should also be cleared before a write 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, erase 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, 040H           ; setup memory pointer low byte MP1L
MOV MP1L, A          ; MP1 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, 040H           ; setup memory pointer low byte MP1L
MOV MP1L, A           ; MP1 points to EEC register
MOV A, 01H            ; setup 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, 040H           ; setup memory pointer low byte MP1L
MOV MP1L, A           ; MP1 points to EEC register
MOV A, 01H            ; setup 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 erase operations
SET IAR1.5            ; start Erase Cycle - set ER bit - executed immediately
; after setting EREN bit

SET EMI
```

```

BACK:
SZ IAR1.5           ; check for erase cycle end
JMP BACK
CLR MP1H

```

Writing a Data Byte to the EEPROM – polling method

```

MOV A, 040H         ; setup memory pointer low byte MP1L
MOV MP1L, A         ; MP1 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
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, 040H         ; setup memory pointer low byte MP1L
MOV MP1L, A         ; MP1 points to EEC register
MOV A, 01H          ; setup 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 defined 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 options and relevant control registers.

Oscillator Overview

In addition to being the source of the main system clock, the oscillators also provide clock sources for the Watchdog Timer and Time Base Interrupts. External oscillator requiring some external components as well as fully integrated internal oscillators, requiring no external components, are provided to form a wide range of both fast and slow system oscillators. All oscillator options are selected through the relevant control registers. The higher frequency oscillator provides higher performance but carry with it the disadvantage of higher power requirements, while the opposite is of course true for the lower frequency 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.

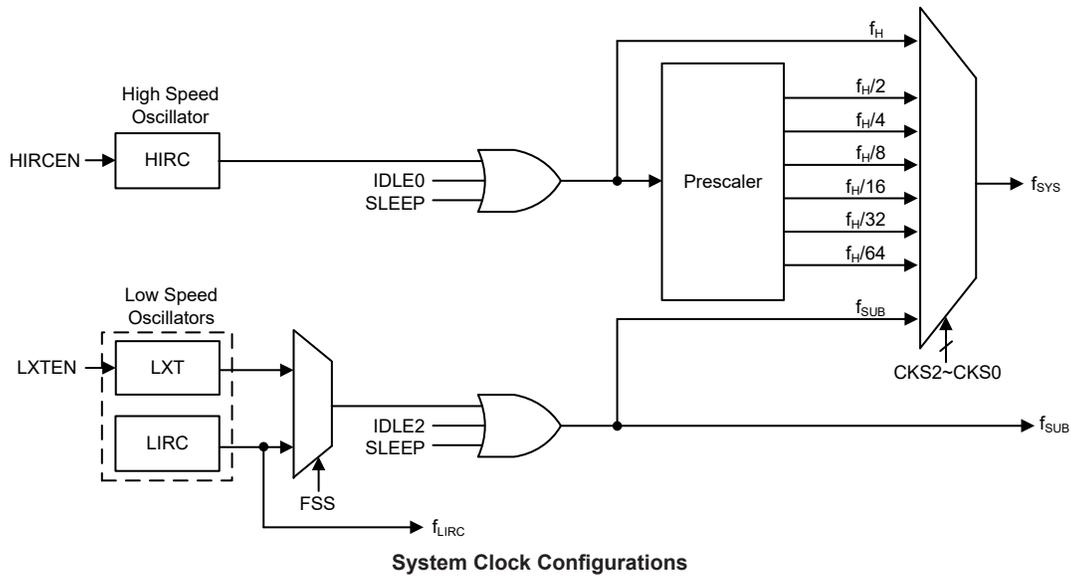
| Type | Name | Frequency | Pins |
|----------------------------|------|------------|---------|
| Internal High Speed RC | HIRC | 8/12/16MHz | — |
| External Low Speed Crystal | LXT | 32.768kHz | XT1/XT2 |
| Internal Low Speed RC | LIRC | 32kHz | — |

Oscillator Types

System Clock Configurations

There are several oscillator sources, one high speed oscillator and two low speed oscillators. The high speed system clock is sourced from the internal 8/12/16MHz RC oscillator, HIRC. The low speed oscillators are the external 32.768kHz crystal oscillator, LXT, and the internal 32kHz RC oscillator, LIRC. Selecting whether the low or high speed oscillator is used as the system oscillator is implemented using the CKS2~CKS0 bits in the SCC register and as the system clock can be dynamically selected.

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



Internal High Speed RC Oscillator – HIRC

The internal high speed RC oscillator is a fully integrated system oscillator requiring no external components. The internal RC oscillator has a fixed frequency 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 is selected, it requires no external pins for its operation.

External 32.768kHz Crystal Oscillator – LXT

The External 32.768 kHz 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.

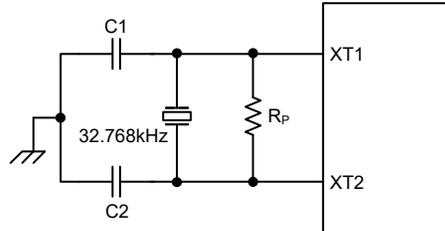
When the microcontroller enters the SLEEP or IDLE Mode, the system clock is switched off to stop microcontroller activity and to conserve power. However, in many microcontroller applications it may be necessary to keep the internal timers operational even when the microcontroller is in the SLEEP or IDLE 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_p, 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_p , C1 and C2 are required.
 2. Although not shown XT1/XT2 pins have a parasitic capacitance of around 7pF.

External LXT Oscillator

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

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

32.768kHz Crystal Recommended Capacitor Values

LXT Oscillator Low Power Function

The LXT oscillator can function in one of three modes, the Quick Start mode, Low Power mode and the High ESR mode. The mode selection is executed using the LXTSP[1:0] bits in the LXTC register.

| LXTSP[1:0] | | LXT Mode |
|------------|---|--|
| 0 | 1 | Quick Start, $C_L=6pF$, $ESR=30k\Omega$ |
| 0 | 0 | Low Power, $C_L=6pF$, $ESR=30k\Omega$ |
| 1 | 0 | High ESR, $C_L=7pF$, $ESR=90k\Omega$ |
| 1 | 1 | (No Low Power/Quick Start distinction) |

When the LXTSP[1:0] bits are set to 01, the LXT Quick Start mode will be enabled. In the Quick Start 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[1:0] bits to zero. 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[1:0] bits are set to, the LXT oscillator will 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 System Oscillator is one of the low frequency oscillator choices, which is selected via a software control bit, FSS. It is a fully integrated RC oscillator with a typical frequency of 32kHz at 5V, requiring no external components for its implementation. Device trimming during the manufacturing process and the inclusion of internal frequency compensation circuits are used to ensure that the influence of the power supply voltage, temperature and process variations on the oscillation frequency are minimised.

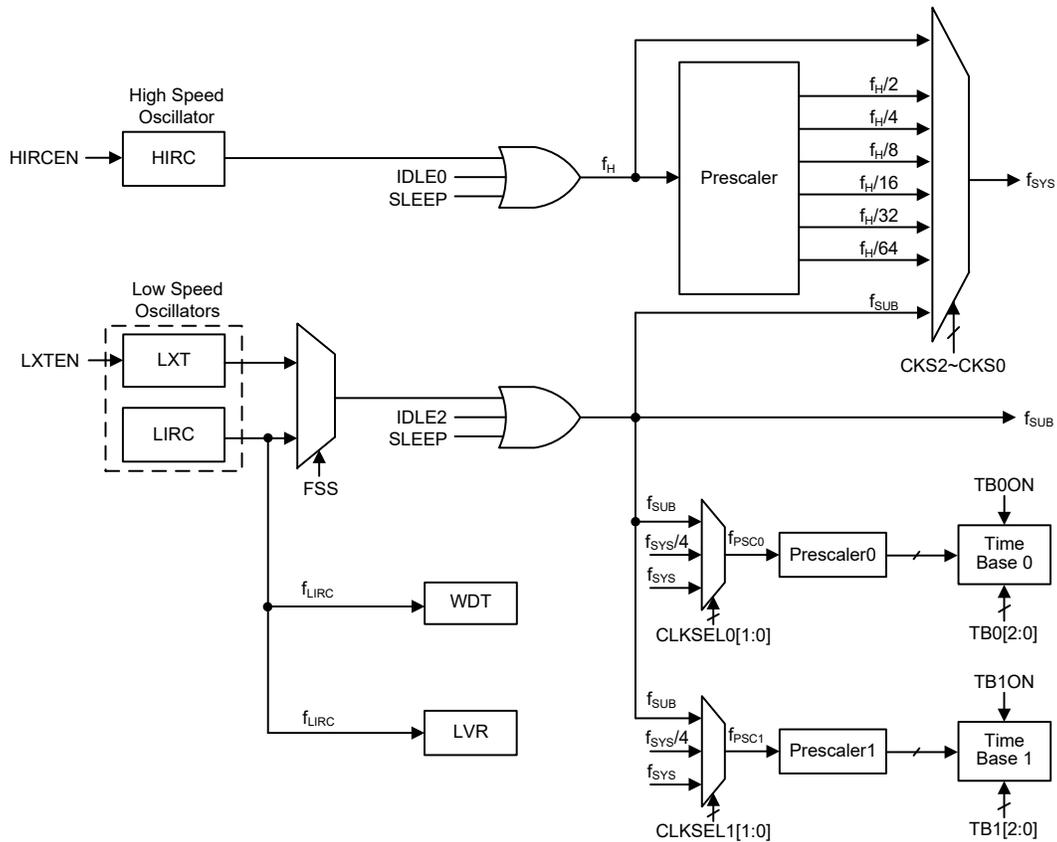
Operating Modes and System Clocks

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 devices with both high and low speed clock sources and the means to switch between them dynamically, the user can optimise the operation of their microcontroller to achieve the best performance/power ratio.

System Clocks

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

The main system clock can come from a high frequency, f_H , or low frequency, f_{SUB} , source, and is selected using the CKS2~CKS0 bits in the SCC register. The high speed system clock is sourced from the HIRC oscillator. The low speed system clock source can be sourced from the internal clock f_{SUB} . If f_{SUB} is selected then it can be sourced by either the LXT or LIRC 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 ⁽¹⁾ | 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 ⁽²⁾ |

"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 the high speed oscillator. This mode operates allowing the microcontroller to operate normally with a clock source which will come from the high speed oscillator, HIRC. The high speed oscillator will however first be divided by a ratio ranging from 1 to 64, the actual ratio being selected by the CKS2~CKS0 bits in the SCC register. Although a high speed oscillator is used, running the microcontroller at a divided clock ratio reduces the operating current.

SLOW Mode

This is also a mode where the microcontroller operates normally although now with a slower speed clock source. The clock source used will be from f_{SUB} . 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 bits 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 will 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 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 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 | — | — | FSS | FHIDEN | FSIDEN |
| HIRCC | — | — | — | — | HIRC1 | HIRC0 | HIRCF | HIRCEN |
| LXTC | — | — | — | — | LXTSP1 | LXTSP0 | LXTF | LXTEN |

System Operating Mode Control Register List

• **SCC Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------|------|------|------|---|---|-----|--------|--------|
| Name | CKS2 | CKS1 | CKS0 | — | — | FSS | FHIDEN | FSIDEN |
| R/W | R/W | R/W | R/W | — | — | R/W | R/W | R/W |
| POR | 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 4~3 Unimplemented, read as “0”

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 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 to 1 to enable the HIRC oscillator, the HIRCF bit will first be cleared to 0 and then set to 1 after the HIRC oscillator is stable.

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

• **LXTC Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------|---|---|---|---|--------|--------|------|-------|
| Name | — | — | — | — | LXTSP1 | LXTSP0 | LXTF | LXTEN |
| R/W | — | — | — | — | R/W | R/W | R | R/W |
| POR | — | — | — | — | 0 | 0 | 0 | 0 |

Bit 7~4 Unimplemented, read as “0”

Bit 3~2 **LXTSP1~LXTSP0**: LXT mode selection
 00: Low Power mode
 01: Quick Start mode
 1x: High ESR mode

These bits are used to control whether the LXT oscillator is operating in the Low Power, Quick Start or High ESR mode. It is important to note that these bits cannot be changed after the LXT oscillator is selected as the system clock source and the LXTF bit is set high.

Bit 1 **LXTF**: LXT oscillator stable flag
 0: LXT unstable
 1: LXT stable

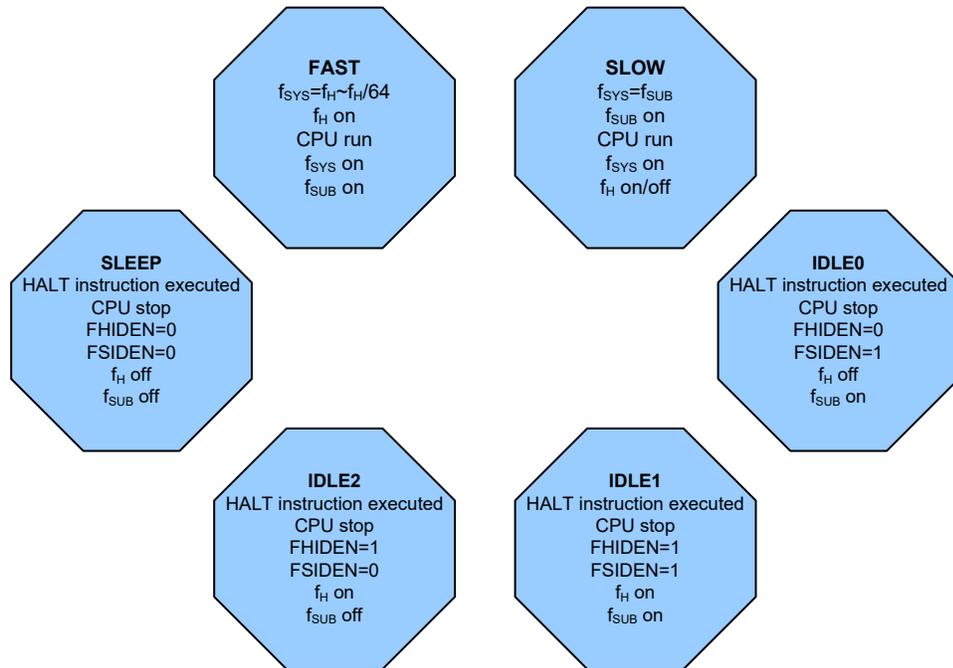
This bit is used to indicate whether the LXT oscillator is stable or not. When the LXTEN bit is set to 1 to enable the LXT oscillator, the LXTF bit will first be cleared to 0 and then set to 1 after the LXT oscillator is stable.

Bit 0 **LXTEN**: LXT oscillator enable control
 0: Disable
 1: Enable

Operating Mode Switching

These devices 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.

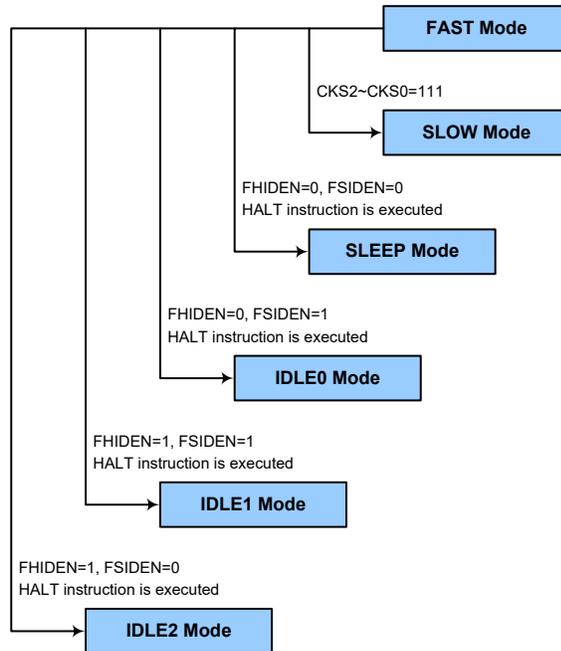
In simple terms, mode switching between the FAST Mode and SLOW Mode is executed using the CKS2~CKS0 bits in the SCC register while mode switching from the FAST/SLOW Modes to the SLEEP/IDLE Modes is executed via the HALT instruction. When a HALT instruction is executed, whether the device enters the IDLE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

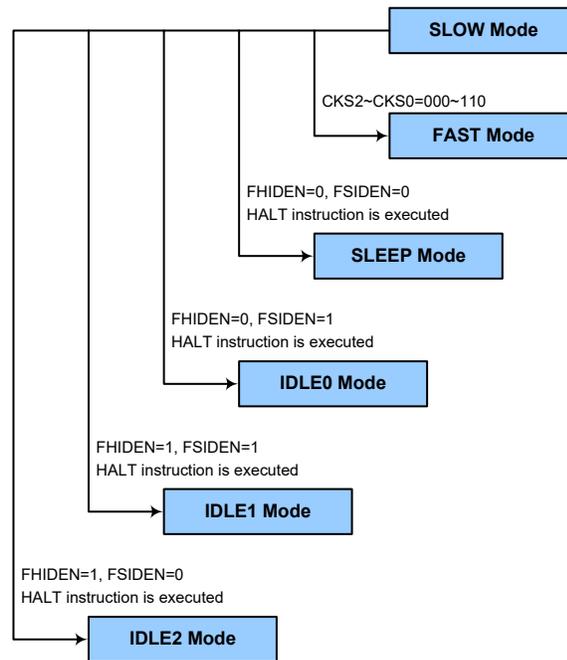
The SLOW Mode is sourced from the LXT or LIRC oscillator determined by the FSS bit in the SCC register and therefore requires the selected oscillator to be stable before full mode switching occurs.



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 HIRCF bit in the HIRCC register. The time duration required for the high speed system oscillator stabilization is specified in the System Start Up Time Characteristics.



Entering the SLEEP Mode

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

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

Entering the IDLE0 Mode

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

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

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 these devices 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 these devices. 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.

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

Wake-up

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

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

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

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

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 timeout period as well as the WDT enable/disable and software reset MCU operation. This register controls the overall operation of the Watchdog Timer.

• WDTC Register

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

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

10101: Disable

01010: Enable

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 WRF bit in the RSTFC register will be set to 1.

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

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

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

010: $2^{12}/f_{LIRC}$

011: $2^{14}/f_{LIRC}$

100: $2^{15}/f_{LIRC}$

101: $2^{16}/f_{LIRC}$

110: $2^{17}/f_{LIRC}$

111: $2^{18}/f_{LIRC}$

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

• RSTFC Register

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

“x”: unknown

Bit 7~3 Unimplemented, read as “0”

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

Refer to the Low Voltage Reset section.

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

Refer to the Low Voltage Reset section.

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

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

Watchdog Timer Operation

The Watchdog Timer operates by providing a device reset when its timer overflows. This means that in the application program and during normal operation the user has to strategically clear the Watchdog Timer before it overflows to prevent the Watchdog Timer from executing a reset. This is done using the clear watchdog 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. With regard to the Watchdog Timer enable/disable function, there are five bits, WE4~WE0, in the WDTC register to offer the enable/disable control and the MCU reset. The WDT function will be enabled when the WE4~WE0 bits are set to a value of 01010B while the WDT function will be disabled if the WE4~WE0 bits are equal to 10101B. If the WE4~WE0 bits are set to any other values rather than 01010B and 10101B, it will reset the device after a delay time, t_{SRESET} . After power on these bits will have a value of 01010B.

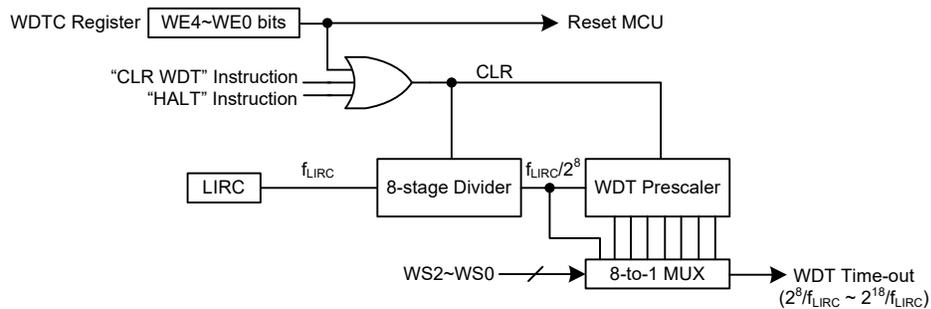
| WE4~WE0 Bits | WDT Function |
|-----------------|--------------|
| 10101B | Disable |
| 01010B | Enable |
| Any other value | Reset MCU |

Watchdog Timer Enable/Disable Control

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

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.

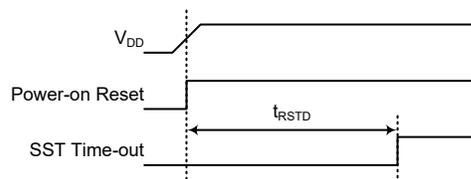
In addition to the power-on reset, another reset exists in the form of a Low Voltage Reset, LVR, where a full reset is implemented in situations where the power supply voltage falls below a certain threshold. Another type of reset is when the Watchdog Timer overflows and resets the microcontroller. All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

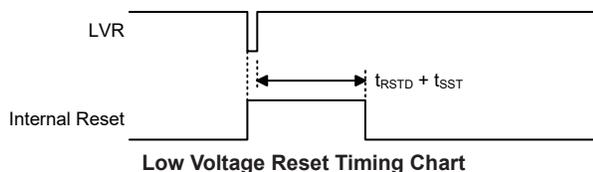
The most fundamental and unavoidable reset is the one that occurs after power is first applied to the microcontroller. As well as ensuring that the Program Memory begins execution from the first memory address, a power-on reset also ensures that certain other registers are preset to known conditions. 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

Low Voltage Reset – LVR

The microcontrollers contain a low voltage reset circuit in order to monitor the supply voltage of the device and provide 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 LVR/LVD 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.


• 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

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 defined register values above, will also result in the generation of an MCU reset. The reset operation will be activated after a delay time, t_{RESET} . However in this situation the register contents will be reset to the POR value.

• TLVRC Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------|---|---|---|---|---|---|-------|-------|
| Name | — | — | — | — | — | — | TLVR1 | TLVR0 |
| R/W | — | — | — | — | — | — | R/W | R/W |
| POR | — | — | — | — | — | — | 0 | 1 |

Bit 7~2 Unimplemented, read as “0”

Bit 1~0 **TLVR1~TLVR0**: Minimum low voltage width to reset time (t_{LVR}) selection

00: $(7\sim8) \times t_{LIRC}$
 01: $(31\sim32) \times t_{LIRC}$
 10: $(63\sim64) \times t_{LIRC}$
 11: $(127\sim128) \times t_{LIRC}$

• RSTFC Register

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

“x”: unknown

Bit 7~3 Unimplemented, read as “0”

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

0: Not occurred
 1: Occurred

This bit is set high when a specific Low Voltage Reset situation occurs. This bit can only be cleared to zero by the application program.

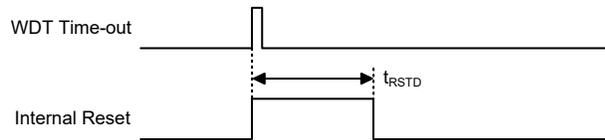
- Bit 1 **LRF**: LVR control register software reset flag
 0: Not occurred
 1: Occurred
 This bit is set high if the LVRC register contains any non-defined LVR voltage register values. This in effect acts like a software-reset function. This bit can only be cleared to zero by the application program.
- 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 IAP section for more associated details.

Watchdog Time-out Reset during Normal Operation

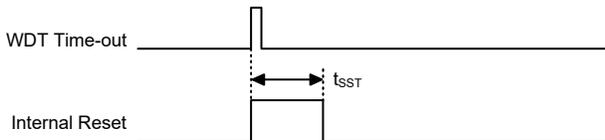
When the Watchdog time-out Reset during normal operations in the FAST or SLOW mode occurs, the Watchdog time-out flag TO will be set to “1”.



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 Mode 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”: 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 | Clear after reset, WDT begins counting |
| Timer Modules | Timer Modules will be turned off |
| Input/Output Ports | I/O ports will be setup as inputs |
| Stack Pointer | Stack Pointer will point to the top of the stack |

The different kinds of resets all affect the internal registers of the microcontroller in different ways. To ensure reliable continuation of normal program execution after a reset occurs, it is important to know what condition the microcontroller is in after a particular reset occurs. The following table describes how each type of reset affects the microcontroller internal registers.

| Register | Power-On Reset | WDT Time-out (Normal Operation) | WDT Time-out (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 | x xxx x xxx | uuuu uuuu | uuuu uuuu |
| PCL | 0000 0000 | 0000 0000 | 0000 0000 |
| TBLP | x xxx x xxx | uuuu uuuu | uuuu uuuu |
| TBLH | x xxx x xxx | uuuu uuuu | uuuu uuuu |
| TBHP | --xx x xxx | --uu uuuu | --uu uuuu |
| STATUS | xx00 x xxx | 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 | ---- -x00 | ---- -uuu | ---- -uuu |
| INTC0 | -000 0000 | -000 0000 | -uuu uuuu |
| INTC1 | 0000 0000 | 0000 0000 | uuuu uuuu |
| INTC2 | 0000 0000 | 0000 0000 | uuuu uuuu |
| INTC3 | 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 |
| MF10 | -000 -000 | -000 -000 | -uuu -uuu |
| MF11 | -000 -000 | -000 -000 | -uuu -uuu |
| TB0C | 0--- -000 | 0--- -000 | u--- -uuu |
| PSC0R | ---- --00 | ---- --00 | ---- --uu |
| LVRC | 0110 0110 | 0110 0110 | uuuu uuuu |
| LVPUC | ---- ---0 | ---- ---0 | ---- ---u |
| LXTC | ---- 0000 | ---- 0000 | ---- uuuu |
| PSC1R | ---- --00 | ---- --00 | ---- --uu |
| TB1C | 0--- -000 | 0--- -000 | u--- -uuu |
| SADC0 | 0000 0000 | 0000 0000 | uuuu uuuu |

| Register | Power-On Reset | WDT Time-out (Normal Operation) | WDT Time-out (IDLE/SLEEP) |
|----------|----------------|------------------------------------|------------------------------|
| SADC1 | 0000 -000 | 0000 -000 | uuuu -uuu |
| SADC2 | 0--0 0000 | 0--0 0000 | u--u uuuu |
| SADOL | xxxx ---- | xxxx ---- | uuuu ---- (ADRF5=0) |
| | | | uuuu uuuu (ADRF5=1) |
| SADOH | xxxx xxxx | xxxx xxxx | uuuu uuuu (ADRF5=0) |
| | | | ---- uuuu (ADRF5=1) |
| SCC | 000- -000 | 000- -000 | uuu- -uuu |
| HIRCC | ---- 0001 | ---- 0001 | ---- uuuu |
| USR | 0000 1011 | 0000 1011 | uuuu uuuu |
| UCR1 | 0000 00x0 | 0000 00x0 | uuuu uuuu |
| UCR2 | 0000 0000 | 0000 0000 | uuuu uuuu |
| UCR3 | ---- ---0 | ---- ---0 | ---- ---u |
| BRDH | 0000 0000 | 0000 0000 | uuuu uuuu |
| BRDL | 0000 0000 | 0000 0000 | uuuu uuuu |
| UFCR | --00 0000 | --00 0000 | --uu uuuu |
| TXR_RXR | xxxx xxxx | xxxx xxxx | uuuu uuuu |
| RxCNT | ---- -000 | ---- -000 | ---- -uuu |
| IFS0 | 0000 0000 | 0000 0000 | uuuu uuuu |
| IFS1 | -000 0000 | -000 0000 | -uuu uuuu |
| IFS2 | -000 0000 | -000 0000 | -uuu uuuu |
| LVDC | 0000 -000 | 0000 -000 | uuuu -uuu |
| VBGRC | ---- ---0 | ---- ---0 | ---- ---u |
| SIMC0 | 1110 0000 | 1110 0000 | uuuu uuuu |
| SIMC1 | 1000 0001 | 1000 0001 | uuuu uuuu |
| UUCR1 | 0000 00x0 | 0000 00x0 | uuuu uuuu |
| SIMD | xxxx xxxx | xxxx xxxx | uuuu uuuu |
| UTXR_RXR | xxxx xxxx | xxxx xxxx | uuuu uuuu |
| SIMA | 0000 0000 | 0000 0000 | uuuu uuuu |
| SIMC2 | 0000 0000 | 0000 0000 | uuuu uuuu |
| UUCR2 | 0000 0000 | 0000 0000 | uuuu uuuu |
| UUCR3 | ---- ---0 | ---- ---0 | ---- ---u |
| SIMTOC | 0000 0000 | 0000 0000 | uuuu uuuu |
| UBRG | xxxx xxxx | xxxx xxxx | uuuu uuuu |
| UUSR | 0000 1011 | 0000 1011 | uuuu uuuu |
| INTEG | ---- 0000 | ---- 0000 | ---- uuuu |
| WDTC | 0101 0011 | 0101 0011 | uuuu uuuu |
| EEAL | 0000 0000 | 0000 0000 | uuuu uuuu |
| EEAH | ---- --00 | ---- --00 | ---- --uu |
| EED | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKTMR | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKC0 | 0000 0-00 | 0000 0-00 | uuuu u-uu |
| TKC1 | 0000 0011 | 0000 0011 | uuuu uuuu |
| TK16DL | 0000 0000 | 0000 0000 | uuuu uuuu |
| TK16DH | 0000 0000 | 0000 0000 | uuuu uuuu |

| Register | Power-On Reset | WDT Time-out (Normal Operation) | WDT Time-out (IDLE/SLEEP) |
|----------|----------------|------------------------------------|------------------------------|
| TKM0C0 | --00 0000 | --00 0000 | --uu uuuu |
| TKM0C1 | 0-00 0000 | 0-00 0000 | u-uu uuuu |
| TKM0C2 | 1110 0100 | 1110 0100 | uuuu uuuu |
| TKM016DL | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM016DH | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM0ROL | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM0ROH | ---- --00 | ---- --00 | ---- --uu |
| TKM1C0 | --00 0000 | --00 0000 | --uu uuuu |
| TKM1C1 | 0-00 0000 | 0-00 0000 | u-uu uuuu |
| TKM1C2 | 1110 0100 | 1110 0100 | uuuu uuuu |
| TKM116DL | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM116DH | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM1ROL | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM1ROH | ---- --00 | ---- --00 | ---- --uu |
| TKM2C0 | --00 0000 | --00 0000 | --uu uuuu |
| TKM2C1 | 0-00 0000 | 0-00 0000 | u-uu uuuu |
| TKM2C2 | 1110 0100 | 1110 0100 | uuuu uuuu |
| TKM216DL | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM216DH | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM2ROL | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM2ROH | ---- --00 | ---- --00 | ---- --uu |
| TKM3C0 | --00 0000 | --00 0000 | --uu uuuu |
| TKM3C1 | 0-00 0000 | 0-00 0000 | u-uu uuuu |
| TKM3C2 | 1110 0100 | 1110 0100 | uuuu uuuu |
| TKM316DL | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM316DH | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM3ROL | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM3ROH | ---- --00 | ---- --00 | ---- --uu |
| TKM4C0 | --00 0000 | --00 0000 | --uu uuuu |
| TKM4C1 | 0-00 0000 | 0-00 0000 | u-uu uuuu |
| TKM4C2 | 1110 0100 | 1110 0100 | uuuu uuuu |
| TKM416DL | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM416DH | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM4ROL | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM4ROH | ---- --00 | ---- --00 | ---- --uu |
| PB | 1111 1111 | 1111 1111 | uuuu uuuu |
| PBC | 1111 1111 | 1111 1111 | uuuu uuuu |
| PBPU | 0000 0000 | 0000 0000 | uuuu uuuu |
| PC | 1111 1111 | 1111 1111 | uuuu uuuu |
| PCC | 1111 1111 | 1111 1111 | uuuu uuuu |
| PCPU | 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 |
| PE | 1111 1111 | 1111 1111 | uuuu uuuu |
| PEC | 1111 1111 | 1111 1111 | uuuu uuuu |
| PEPU | 0000 0000 | 0000 0000 | uuuu uuuu |

| Register | Power-On Reset | WDT Time-out (Normal Operation) | WDT Time-out (IDLE/SLEEP) |
|----------|----------------|------------------------------------|------------------------------|
| PF | --11 1111 | --11 1111 | --uu uuuu |
| PFC | --11 1111 | --11 1111 | --uu uuuu |
| PFPU | --00 0000 | --00 0000 | --uu uuuu |
| ORMC | 0000 0000 | 0000 0000 | 0000 0000 |
| CTM0C0 | 0000 0000 | 0000 0000 | uuuu uuuu |
| CTM0C1 | 0000 0000 | 0000 0000 | uuuu uuuu |
| CTM0DL | 0000 0000 | 0000 0000 | uuuu uuuu |
| CTM0DH | ---- --00 | ---- --00 | ---- --uu |
| CTM0AL | 0000 0000 | 0000 0000 | uuuu uuuu |
| CTM0AH | ---- --00 | ---- --00 | ---- --uu |
| CTM1C0 | 0000 0000 | 0000 0000 | uuuu uuuu |
| CTM1C1 | 0000 0000 | 0000 0000 | uuuu uuuu |
| CTM1DL | 0000 0000 | 0000 0000 | uuuu uuuu |
| CTM1DH | ---- --00 | ---- --00 | ---- --uu |
| CTM1AL | 0000 0000 | 0000 0000 | uuuu uuuu |
| CTM1AH | ---- --00 | ---- --00 | ---- --uu |
| 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 |
| PAS0 | -000 -000 | -000 -000 | -uuu -uuu |
| PAS1 | -000 -000 | -000 -000 | -uuu -uuu |
| PAS2 | -000 -000 | -000 -000 | -uuu -uuu |
| PAS3 | -000 -000 | -000 -000 | -uuu -uuu |
| PBS0 | 0000 0000 | 0000 0000 | uuuu uuuu |
| PBS1 | 0000 0000 | 0000 0000 | uuuu uuuu |
| PCS0 | 0000 0000 | 0000 0000 | uuuu uuuu |
| PCS1 | 0000 0000 | 0000 0000 | uuuu uuuu |
| PDS0 | 0000 0000 | 0000 0000 | uuuu uuuu |
| PDS1 | 0000 0000 | 0000 0000 | uuuu uuuu |
| PES0 | 0000 0000 | 0000 0000 | uuuu uuuu |
| PES1 | 0000 0000 | 0000 0000 | uuuu uuuu |
| PFS0 | 0000 0000 | 0000 0000 | uuuu uuuu |
| PFS1 | ---- 0000 | ---- 0000 | ---- uuuu |
| PCNS | 0000 0000 | 0000 0000 | uuuu uuuu |
| PDNS | 0000 0000 | 0000 0000 | uuuu uuuu |
| PFNS | --00 0000 | --00 0000 | --uu uuuu |
| PMPS | 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 |

| Register | Power-On Reset | WDT Time-out (Normal Operation) | WDT Time-out (IDLE/SLEEP) |
|----------|----------------|------------------------------------|------------------------------|
| PTM1AH | ---- --00 | ---- --00 | ---- --uu |
| PTM1RPL | 0000 0000 | 0000 0000 | uuuu uuuu |
| PTM1RPH | ---- --00 | ---- --00 | ---- --uu |
| 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 |
| SLCDS3 | ---- --00 | ---- --00 | ---- --uu |
| CRCCR | ---- ---0 | ---- ---0 | ---- ---u |
| CRCIN | 0000 0000 | 0000 0000 | uuuu uuuu |
| CRCDL | 0000 0000 | 0000 0000 | uuuu uuuu |
| CRCDH | 0000 0000 | 0000 0000 | uuuu uuuu |
| SLEDC0 | --01 0010 | --01 0010 | --uu uuuu |
| SLEDC1 | --01 0010 | --01 0010 | --uu uuuu |
| SLEDC2 | --01 0010 | --01 0010 | --uu uuuu |
| SLEDC3 | --01 0010 | --01 0010 | --uu uuuu |
| SLEDC4 | --01 0010 | --01 0010 | --uu uuuu |
| SLEDC5 | --01 0010 | --01 0010 | --uu uuuu |
| TLVRC | ---- --01 | ---- --01 | ---- --uu |
| STKPTR | 0--- 0000 | 0--- 0000 | u--- uuuu |
| EEC | 0000 0000 | 0000 0000 | 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 |
| IECC | 0000 0000 | 0000 0000 | uuuu uuuu |
| PCRL | 0000 0000 | 0000 0000 | uuuu uuuu |
| PCRH | --00 0000 | --00 0000 | --uu uuuu |
| OCPC0 | 0000 0000 | 0000 0000 | uuuu uuuu |
| OCPC1 | 0000 0000 | 0000 0000 | uuuu uuuu |
| OCPDA | 0000 0000 | 0000 0000 | uuuu uuuu |
| OCPOCAL | 0010 0000 | 0010 0000 | uuuu uuuu |
| OCPCCAL | 0001 0000 | 0001 0000 | uuuu uuuu |

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

Input/Output Ports

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

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

| Register Name | Bit | | | | | | | |
|---------------|-------|-------|-------|-------|-------|-------|-------|-------|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| PA | PA7 | PA6 | PA5 | PA4 | PA3 | PA2 | PA1 | PA0 |
| PAC | PAC7 | PAC6 | PAC5 | PAC4 | PAC3 | PAC2 | PAC1 | PAC0 |
| PAPU | PAPU7 | PAPU6 | PAPU5 | PAPU4 | PAPU3 | PAPU2 | PAPU1 | PAPU0 |
| PAWU | PAWU7 | PAWU6 | PAWU5 | PAWU4 | PAWU3 | PAWU2 | PAWU1 | PAWU0 |
| PB | PB7 | PB6 | PB5 | PB4 | PB3 | PB2 | PB1 | PB0 |
| PBC | PBC7 | PBC6 | PBC5 | PBC4 | PBC3 | PBC2 | PBC1 | PBC0 |
| PBPU | PBPU7 | PBPU6 | PBPU5 | PBPU4 | PBPU3 | PBPU2 | PBPU1 | PBPU0 |
| PC | PC7 | PC6 | PC5 | PC4 | PC3 | PC2 | PC1 | PC0 |
| PCC | PCC7 | PCC6 | PCC5 | PCC4 | PCC3 | PCC2 | PCC1 | PCC0 |
| PCPU | PCPU7 | PCPU6 | PCPU5 | PCPU4 | PCPU3 | PCPU2 | PCPU1 | PCPU0 |
| PD | 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 |
| PE | PE7 | PE6 | PE5 | PE4 | PE3 | PE2 | PE1 | PE0 |
| PEC | PEC7 | PEC6 | PEC5 | PEC4 | PEC3 | PEC2 | PEC1 | PEC0 |
| PEPU | PEPU7 | PEPU6 | PEPU5 | PEPU4 | PEPU3 | PEPU2 | PEPU1 | PEPU0 |
| PF | — | — | PF5 | PF4 | PF3 | PF2 | PF1 | PF0 |
| PFC | — | — | PFC5 | PFC4 | PFC3 | PFC2 | PFC1 | PFC0 |
| PFPU | — | — | PFPU5 | PFPU4 | PFPU3 | PFPU2 | PFPU1 | PFPU0 |
| LVPUC | — | — | — | — | — | — | — | LVPUC |

“—”: Unimplemented, read as “0”

I/O Logic Function Register List

Pull-high Resistors

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

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

• **PxPU Register**

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

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” is the Port name which can be A, B, C, D, E or F. However, the actual available bits for each I/O Port may be different.

• **LVPUC Register**

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

Bit 7~1 Unimplemented, read as “0”

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

0: All pin pull-high resistors are 60kΩ (typ.) @ 3V

1: All pin pull-high resistors are 15kΩ (typ.) @ 3V

The LVPUC register is used to select the pull-high resistor value for low voltage power supply applications. Note that the LVPU bit is only available when the corresponding pin pull-high function is enabled. This bit will have no effect on selecting the pull-high resistor value 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 registers 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 Port has its own control register which controls the input/output configuration. With this control register, each CMOS output or input can be reconfigured dynamically under software control. Each pin of the I/O ports is directly mapped to a bit in its associated port control register. For the I/O pin to

function as an input, the corresponding bit of the control register must be written as a “1”. This will then allow the logic state of the input pin to be directly read by instructions. When the corresponding bit of the control register is written as a “0”, the I/O pin will be setup as a CMOS output. If the pin is currently setup as an output, instructions can still be used to read the output register.

However, it should be noted that the program will in fact only read the status of the output data latch and not the actual logic status of the output pin when the IECM is cleared to “0”.

• **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” is the Port name which can be A, B, C, D, E or F. 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 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 | — | — | SLEDC05 | SLEDC04 | SLEDC03 | SLEDC02 | SLEDC01 | SLEDC00 |
| SLEDC1 | — | — | SLEDC15 | SLEDC14 | SLEDC13 | SLEDC12 | SLEDC11 | SLEDC10 |
| SLEDC2 | — | — | SLEDC25 | SLEDC24 | SLEDC23 | SLEDC22 | SLEDC21 | SLEDC20 |
| SLEDC3 | — | — | SLEDC35 | SLEDC34 | SLEDC33 | SLEDC32 | SLEDC31 | SLEDC30 |
| SLEDC4 | — | — | SLEDC45 | SLEDC44 | SLEDC43 | SLEDC42 | SLEDC41 | SLEDC40 |
| SLEDC5 | — | — | SLEDC55 | SLEDC54 | SLEDC53 | SLEDC52 | SLEDC51 | SLEDC50 |

I/O Port Source Current Selection Register List

• **SLEDC0 Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------|---|---|---------|---------|---------|---------|---------|---------|
| Name | — | — | SLEDC05 | SLEDC04 | SLEDC03 | SLEDC02 | SLEDC01 | SLEDC00 |
| R/W | — | — | R/W | R/W | R/W | R/W | R/W | R/W |
| POR | — | — | 0 | 1 | 0 | 0 | 1 | 0 |

Bit 7~6 Unimplemented, read as “0”

Bit 5~3 **SLEDC05~SLEDC03:** PA7~PA4 source current selection

000: source current=Level 0 (Min.)

001: source current=Level 1

010: source current=Level 2

011: source current=Level 3

100: source current=Level 4

101: source current=Level 5

110: source current=Level 6

111: source current=Level 7 (Max.)

- Bit 2~0 **SLEDC02~SLEDC00**: PA3~PA0 source current selection
 000: source current=Level 0 (Min.)
 001: source current=Level 1
 010: source current=Level 2
 011: source current=Level 3
 100: source current=Level 4
 101: source current=Level 5
 110: source current=Level 6
 111: source current=Level 7 (Max.)

• **SLEDC1 Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------|---|---|---------|---------|---------|---------|---------|---------|
| Name | — | — | SLEDC15 | SLEDC14 | SLEDC13 | SLEDC12 | SLEDC11 | SLEDC10 |
| R/W | — | — | R/W | R/W | R/W | R/W | R/W | R/W |
| POR | — | — | 0 | 1 | 0 | 0 | 1 | 0 |

- Bit 7~6 Unimplemented, read as “0”
- Bit 5~3 **SLEDC15~SLEDC13**: PB7~PB4 source current selection
 000: source current=Level 0 (Min.)
 001: source current=Level 1
 010: source current=Level 2
 011: source current=Level 3
 100: source current=Level 4
 101: source current=Level 5
 110: source current=Level 6
 111: source current=Level 7 (Max.)
- Bit 2~0 **SLEDC12~SLEDC10**: PB3~PB0 source current selection
 000: source current=Level 0 (Min.)
 001: source current=Level 1
 010: source current=Level 2
 011: source current=Level 3
 100: source current=Level 4
 101: source current=Level 5
 110: source current=Level 6
 111: source current=Level 7 (Max.)

• **SLEDC2 Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------|---|---|---------|---------|---------|---------|---------|---------|
| Name | — | — | SLEDC25 | SLEDC24 | SLEDC23 | SLEDC22 | SLEDC21 | SLEDC20 |
| R/W | — | — | R/W | R/W | R/W | R/W | R/W | R/W |
| POR | — | — | 0 | 1 | 0 | 0 | 1 | 0 |

- Bit 7~6 Unimplemented, read as “0”
- Bit 5~3 **SLEDC25~SLEDC23**: PC7~PC4 source current selection
 000: source current=Level 0 (Min.)
 001: source current=Level 1
 010: source current=Level 2
 011: source current=Level 3
 100: source current=Level 4
 101: source current=Level 5
 110: source current=Level 6
 111: source current=Level 7 (Max.)

- Bit 2~0 **SLEDC22~SLEDC20**: PC3~PC0 source current selection
 000: source current=Level 0 (Min.)
 001: source current=Level 1
 010: source current=Level 2
 011: source current=Level 3
 100: source current=Level 4
 101: source current=Level 5
 110: source current=Level 6
 111: source current=Level 7 (Max.)

• **SLEDC3 Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------|---|---|---------|---------|---------|---------|---------|---------|
| Name | — | — | SLEDC35 | SLEDC34 | SLEDC33 | SLEDC32 | SLEDC31 | SLEDC30 |
| R/W | — | — | R/W | R/W | R/W | R/W | R/W | R/W |
| POR | — | — | 0 | 1 | 0 | 0 | 1 | 0 |

- Bit 7~6 Unimplemented, read as “0”
- Bit 5~3 **SLEDC35~SLEDC33**: PD7~PD4 source current selection
 000: source current=Level 0 (Min.)
 001: source current=Level 1
 010: source current=Level 2
 011: source current=Level 3
 100: source current=Level 4
 101: source current=Level 5
 110: source current=Level 6
 111: source current=Level 7 (Max.)
- Bit 2~0 **SLEDC32~SLEDC30**: PD3~PD0 source current selection
 000: source current=Level 0 (Min.)
 001: source current=Level 1
 010: source current=Level 2
 011: source current=Level 3
 100: source current=Level 4
 101: source current=Level 5
 110: source current=Level 6
 111: source current=Level 7 (Max.)

• **SLEDC4 Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------|---|---|---------|---------|---------|---------|---------|---------|
| Name | — | — | SLEDC45 | SLEDC44 | SLEDC43 | SLEDC42 | SLEDC41 | SLEDC40 |
| R/W | — | — | R/W | R/W | R/W | R/W | R/W | R/W |
| POR | — | — | 0 | 1 | 0 | 0 | 1 | 0 |

- Bit 7~6 Unimplemented, read as “0”
- Bit 5~3 **SLEDC45~SLEDC43**: PE7~PE4 source current selection
 000: source current=Level 0 (Min.)
 001: source current=Level 1
 010: source current=Level 2
 011: source current=Level 3
 100: source current=Level 4
 101: source current=Level 5
 110: source current=Level 6
 111: source current=Level 7 (Max.)

- Bit 2~0 **SLEDC42~SLEDC40**: PE3~PE0 source current selection
 000: source current=Level 0 (Min.)
 001: source current=Level 1
 010: source current=Level 2
 011: source current=Level 3
 100: source current=Level 4
 101: source current=Level 5
 110: source current=Level 6
 111: source current=Level 7 (Max.)

• **SLEDC5 Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------|---|---|---------|---------|---------|---------|---------|---------|
| Name | — | — | SLEDC55 | SLEDC54 | SLEDC53 | SLEDC52 | SLEDC51 | SLEDC50 |
| R/W | — | — | R/W | R/W | R/W | R/W | R/W | R/W |
| POR | — | — | 0 | 1 | 0 | 0 | 1 | 0 |

- Bit 7~6 Unimplemented, read as “0”
- Bit 5~3 **SLEDC55~SLEDC53**: PF5~PF4 source current selection
 000: source current=Level 0 (Min.)
 001: source current=Level 1
 010: source current=Level 2
 011: source current=Level 3
 100: source current=Level 4
 101: source current=Level 5
 110: source current=Level 6
 111: source current=Level 7 (Max.)
- Bit 2~0 **SLEDC52~SLEDC50**: PF3~PF0 source current selection
 000: source current=Level 0 (Min.)
 001: source current=Level 1
 010: source current=Level 2
 011: source current=Level 3
 100: source current=Level 4
 101: source current=Level 5
 110: source current=Level 6
 111: source current=Level 7 (Max.)

I/O Port Sink Current Selection

The device supports different output sink current driving capability for PC, PD and PF ports. With the selection register, PxNS, specific I/O port can support two levels of the sink current driving capability. These sink current selection bits are available 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 sink current for different applications.

| Register Name | Bit | | | | | | | |
|---------------|-------|-------|-------|-------|-------|-------|-------|-------|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| PCNS | PCNS7 | PCNS6 | PCNS5 | PCNS4 | PCNS3 | PCNS2 | PCNS1 | PCNS0 |
| PDNS | PDNS7 | PDNS6 | PDNS5 | PDNS4 | PDNS3 | PDNS2 | PDNS1 | PDNS0 |
| PFNS | — | — | PFNS5 | PFNS4 | PFNS3 | PFNS2 | PFNS1 | PFNS0 |

I/O Port Sink Current Selection Register List

• **PCNS Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------|-------|-------|-------|-------|-------|-------|-------|-------|
| Name | PCNS7 | PCNS6 | PCNS5 | PCNS4 | PCNS3 | PCNS2 | PCNS1 | PCNS0 |
| 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 **PCNS7:** PC7 sink current selection (NMOS adjust)
 0: Sink current=Level 0 (Min.)
 1: Sink current=Level 1 (Max.)
- Bit 6 **PCNS6:** PC6 sink current selection (NMOS adjust)
 0: Sink current=Level 0 (Min.)
 1: Sink current=Level 1 (Max.)
- Bit 5 **PCNS5:** PC5 sink current selection (NMOS adjust)
 0: Sink current=Level 0 (Min.)
 1: Sink current=Level 1 (Max.)
- Bit 4 **PCNS4:** PC4 sink current selection (NMOS adjust)
 0: Sink current=Level 0 (Min.)
 1: Sink current=Level 1 (Max.)
- Bit 3 **PCNS3:** PC3 sink current selection (NMOS adjust)
 0: Sink current=Level 0 (Min.)
 1: Sink current=Level 1 (Max.)
- Bit 2 **PCNS2:** PC2 sink current selection (NMOS adjust)
 0: Sink current=Level 0 (Min.)
 1: Sink current=Level 1 (Max.)
- Bit 1 **PCNS1:** PC1 sink current selection (NMOS adjust)
 0: Sink current=Level 0 (Min.)
 1: Sink current=Level 1 (Max.)
- Bit 0 **PCNS0:** PC0 sink current selection (NMOS adjust)
 0: Sink current=Level 0 (Min.)
 1: Sink current=Level 1 (Max.)

• **PDNS Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------|-------|-------|-------|-------|-------|-------|-------|-------|
| Name | PDNS7 | PDNS6 | PDNS5 | PDNS4 | PDNS3 | PDNS2 | PDNS1 | PDNS0 |
| 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 **PDNS7:** PD7 sink current selection (NMOS adjust)
 0: Sink current=Level 0 (Min.)
 1: Sink current=Level 1 (Max.)
- Bit 6 **PDNS6:** PD6 sink current selection (NMOS adjust)
 0: Sink current=Level 0 (Min.)
 1: Sink current=Level 1 (Max.)
- Bit 5 **PDNS5:** PD5 sink current selection (NMOS adjust)
 0: Sink current=Level 0 (Min.)
 1: Sink current=Level 1 (Max.)
- Bit 4 **PDNS4:** PD4 sink current selection (NMOS adjust)
 0: Sink current=Level 0 (Min.)
 1: Sink current=Level 1 (Max.)
- Bit 3 **PDNS3:** PD3 sink current selection (NMOS adjust)
 0: Sink current=Level 0 (Min.)
 1: Sink current=Level 1 (Max.)

- Bit 2 **PDNS2:** PD2 sink current selection (NMOS adjust)
0: Sink current=Level 0 (Min.)
1: Sink current=Level 1 (Max.)
- Bit 1 **PDNS1:** PD1 sink current selection (NMOS adjust)
0: Sink current=Level 0 (Min.)
1: Sink current=Level 1 (Max.)
- Bit 0 **PDNS0:** PD0 sink current selection (NMOS adjust)
0: Sink current=Level 0 (Min.)
1: Sink current=Level 1 (Max.)

• **PFNS Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------|---|---|-------|-------|-------|-------|-------|-------|
| Name | — | — | PFNS5 | PFNS4 | PFNS3 | PFNS2 | PFNS1 | PFNS0 |
| 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 **PFNS5:** PF5 sink current selection (NMOS adjust)
0: Sink current=Level 0 (Min.)
1: Sink current=Level 1 (Max.)
- Bit 4 **PFNS4:** PF4 sink current selection (NMOS adjust)
0: Sink current=Level 0 (Min.)
1: Sink current=Level 1 (Max.)
- Bit 3 **PFNS3:** PF3 sink current selection (NMOS adjust)
0: Sink current=Level 0 (Min.)
1: Sink current=Level 1 (Max.)
- Bit 2 **PFNS2:** PF2 sink current selection (NMOS adjust)
0: Sink current=Level 0 (Min.)
1: Sink current=Level 1 (Max.)
- Bit 1 **PFNS1:** PF1 sink current selection (NMOS adjust)
0: Sink current=Level 0 (Min.)
1: Sink current=Level 1 (Max.)
- Bit 0 **PFNS0:** PF0 sink current selection (NMOS adjust)
0: Sink current=Level 0 (Min.)
1: Sink current=Level 1 (Max.)

I/O Port Power Source Control

The device supports different I/O port power source selections for PD4~PD7 pins. With the exception of RES/OCDS, the multi-power function is only effective when the pin is set to have a digital input or output function.

The port power can come from either the power pin VDD or VDDIO, which is determined using the PMPS register. The VDDIO power pin function should first be selected using the corresponding pin-shared function selection bits if the port power is supposed to come from the VDDIO pin.

An important point to know is that the input power voltage on the VDDIO pin should be equal to or less than the device supply power voltage V_{DD} when the VDDIO pin is selected as the port power supply pin.

• **PMPS Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------|-------|-------|-------|-------|-------|-------|-------|-------|
| Name | PMPS7 | PMPS6 | PMPS5 | PMPS4 | PMPS3 | PMPS2 | PMPS1 | PMPS0 |
| 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 **PMPS7~PMPS6**: PD7 pin power supply selection

0x: V_{DD}
 1x: V_{DDIO}

Bit 5~4 **PMPS5~PMPS4**: PD6 pin power supply selection

0x: V_{DD}
 1x: V_{DDIO}

Bit 3~2 **PMPS3~PMPS2**: PD5 pin power supply selection

0x: V_{DD}
 1x: V_{DDIO}

Bit 1~0 **PMPS1~PMPS0**: PD4 pin power supply selection

0x: V_{DD}
 1x: V_{DDIO}

If the PE4 pin is switched to the VDDIO function, and the PMPS[m+1, m] (m=0, 2, 4 or 6) bits are set to “1x”, the VDDIO pin input voltage can be used for PD4~PD7 pin power.

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 PxSn, and Input Function Selection register “i”, labeled as IFSi, 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 digital input pins, such as INTn, xTCKn, PTPnI, etc, which share the same pin-shared control configuration with their corresponding general purpose I/O functions when setting the relevant pin-shared control bits. To select these pin functions, in addition to the necessary pin-shared control and peripheral functional setup aforementioned, they must also be set as an input by setting the corresponding bit in the I/O port control register. To correctly deselect the pin-shared function, the peripheral function should first be disabled and then the corresponding pin-shared function control register can be modified to select other pin-shared functions.

| Register Name | Bit | | | | | | | |
|---------------|----------|----------|----------|----------|----------|----------|----------|----------|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| PAS0 | — | PAS06 | PAS05 | PAS04 | — | PAS02 | PAS01 | PAS00 |
| PAS1 | — | PAS16 | PAS15 | PAS14 | — | PAS12 | PAS11 | PAS10 |
| PAS2 | — | PAS26 | PAS25 | PAS24 | — | PAS22 | PAS21 | PAS20 |
| PAS3 | — | PAS36 | PAS35 | PAS34 | — | PAS32 | PAS31 | PAS30 |
| PBS0 | PBS07 | PBS06 | PBS05 | PBS04 | PBS03 | PBS02 | PBS01 | PBS00 |
| PBS1 | PBS17 | PBS16 | PBS15 | PBS14 | PBS13 | PBS12 | PBS11 | PBS10 |
| PCS0 | PCS07 | PCS06 | PCS05 | PCS04 | PCS03 | PCS02 | PCS01 | PCS00 |
| PCS1 | PCS17 | PCS16 | PCS15 | PCS14 | PCS13 | PCS12 | PCS11 | PCS10 |
| PDS0 | PDS07 | PDS06 | PDS05 | PDS04 | PDS03 | PDS02 | PDS01 | PDS00 |
| PDS1 | PDS17 | PDS16 | PDS15 | PDS14 | PDS13 | PDS12 | PDS11 | PDS10 |
| PES0 | PES07 | PES06 | PES05 | PES04 | PES03 | PES02 | PES01 | PES00 |
| PES1 | PES17 | PES16 | PES15 | PES14 | PES13 | PES12 | PES11 | PES10 |
| PFS0 | PFS07 | PFS06 | PFS05 | PFS04 | PFS03 | PFS02 | PFS01 | PFS00 |
| PFS1 | — | — | — | — | PFS13 | PFS12 | PFS11 | PFS10 |
| IFS0 | PTP0IPS1 | PTP0IPS0 | PTCK1PS1 | PTCK1PS0 | PTCK0PS1 | PTCK0PS0 | CTCK1PS | CTCK0PS |
| IFS1 | — | RXPS2 | RXPS1 | RXPS0 | INT1PS | INT0PS | PTP1IPS1 | PTP1IPS0 |
| IFS2 | — | SCSBPS | SCKPS2 | SCKPS1 | SCKPS0 | URXPS2 | URXPS1 | URXPS0 |

Pin-shared Function Selection Register List

• **PAS0 Register**

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

Bit 7 Unimplemented, read as “0”

Bit 6~4 **PAS06~PAS04**: PA1 pin-shared function selection
 000: PA1/PTCK0/CTCK1
 001: PTP0
 010: CTP1
 011: PTP1B
 100: AN11
 101: SSEG13/SCOM13
 110: TX
 111: PA1/PTCK0/CTCK1

Bit 3 Unimplemented, read as “0”

Bit 2~0 **PAS02~PAS00**: PA0 pin-shared function selection
 000: PA0/INT1
 001: PTP0
 010: CTP0
 011: CTP0B
 100: TX
 101: SSEG10/SCOM10
 110: PA0/INT1
 111: PA0/INT1

• **PAS1 Register**

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

- Bit 7 Unimplemented, read as “0”
- Bit 6~4 **PAS16~PAS14**: PA3 pin-shared function selection
 000: PA3/PTP1I
 001: PTP0
 010: CTP0
 011: SDI/SDA/URX/UTX
 100: TX
 101: SSEG9/SCOM9
 110: XT2
 111: PA3/PTP1I
- Bit 3 Unimplemented, read as “0”
- Bit 2~0 **PAS12~PAS10**: PA2 pin-shared function selection
 000: PA2
 001: PTP1
 010: PTP0B
 011: CTP1
 100: RX/TX
 101: SSEG11/SCOM11
 110: PA2
 111: PA2

• **PAS2 Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------|---|-------|-------|-------|---|-------|-------|-------|
| Name | — | PAS26 | PAS25 | PAS24 | — | PAS22 | PAS21 | PAS20 |
| R/W | — | R/W | R/W | R/W | — | R/W | R/W | R/W |
| POR | — | 0 | 0 | 0 | — | 0 | 0 | 0 |

- Bit 7 Unimplemented, read as “0”
- Bit 6~4 **PAS26~PAS24**: PA5 pin-shared function selection
 000: PA5
 001: PTP0
 010: CTP1
 011: PTP1B
 100: CTP0B
 101: AN9
 110: SSEG6/SCOM6
 111: PA5
- Bit 3 Unimplemented, read as “0”
- Bit 2~0 **PAS22~PAS20**: PA4 pin-shared function selection
 000: PA4/PTP0I/INT0
 001: PTP0
 010: PTP1
 011: CTP0
 100: RX/TX
 101: SSEG12/SCOM12
 110: VREFI
 111: PA4/PTP0I/INT0

• **PAS3 Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------|---|-------|-------|-------|---|-------|-------|-------|
| Name | — | PAS36 | PAS35 | PAS34 | — | PAS32 | PAS31 | PAS30 |
| R/W | — | R/W | R/W | R/W | — | R/W | R/W | R/W |
| POR | — | 0 | 0 | 0 | — | 0 | 0 | 0 |

- Bit 7 Unimplemented, read as “0”
- Bit 6~4 **PAS36~PAS34**: PA7 pin-shared function selection
 000: PA7/PTP0I
 001: PTP1
 010: CTP1
 011: SCK/SCL
 100: RX/TX
 101: SSEG8/SCOM8
 110: XT1
 111: PA7/PTP0I
- Bit 3 Unimplemented, read as “0”
- Bit 2~0 **PAS32~PAS30**: PA6 pin-shared function selection
 000: PA6
 001: PTP1
 010: CTP0
 011: PTP0B
 100: CTP1B
 101: AN10
 110: SSEG7/SCOM7
 111: PA6

• **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: PB3
 10: SSEG21/SCOM21
 11: KEY4
- Bit 5~4 **PBS05~PBS04**: PB2 pin-shared function selection
 00: PB2
 01: PB2
 10: SSEG20/SCOM20
 11: KEY3
- Bit 3~2 **PBS03~PBS02**: PB1 pin-shared function selection
 00: PB1
 01: PB1
 10: SSEG19/SCOM19
 11: KEY2
- Bit 1~0 **PBS01~PBS00**: PB0 pin-shared function selection
 00: PB0
 01: PB0
 10: SSEG18/SCOM18
 11: KEY1

• **PBS1 Register**

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

- Bit 7~6 **PBS17~PBS16:** PB7 pin-shared function selection
00: PB7/PTP0I
01: PTP1B
10: CTP1
11: KEY8
- Bit 5~4 **PBS15~PBS14:** PB6 pin-shared function selection
00: PB6/PTCK0
01: PTP1
10: CTP0
11: KEY7
- Bit 3~2 **PBS13~PBS12:** PB5 pin-shared function selection
00: PB5/PTCK1
01: PTP0B
10: PTP1
11: KEY6
- Bit 1~0 **PBS11~PBS10:** PB4 pin-shared function selection
00: PB4/PTP1I
01: PTP0
10: PTP1B
11: KEY5

• **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: SDI/SDA/URX/UTX
10: SDO/UTX
11: KEY12
- Bit 5~4 **PCS05~PCS04:** PC2 pin-shared function selection
00: PC2
01: SCK/SCL
10: SDI/SDA/URX/UTX
11: KEY11
- Bit 3~2 **PCS03~PCS02:** PC1 pin-shared function selection
00: PC1
01: CTP0
10: SSEG1/SCOM1
11: KEY10
- Bit 1~0 **PCS01~PCS00:** PC0 pin-shared function selection
00: PC0/CTCK0
01: CTP1
10: SSEG0/SCOM0
11: KEY9

• **PCS1 Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------|-------|-------|-------|-------|-------|-------|-------|-------|
| Name | PCS17 | PCS16 | PCS15 | PCS14 | PCS13 | PCS12 | PCS11 | PCS10 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| POR | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

- Bit 7~6 **PCS17~PCS16:** PC7 pin-shared function selection
 00: PC7
 01: SDI/SDA/URX/UTX
 10: SSEG5/SCOM5
 11: KEY16
- Bit 5~4 **PCS15~PCS14:** PC6 pin-shared function selection
 00: PC6
 01: SCK/SCL
 10: SSEG4/SCOM4
 11: KEY15
- Bit 3~2 **PCS13~PCS12:** PC5 pin-shared function selection
 00: PC5
 01: PTP1B
 10: SSEG3/SCOM3
 11: KEY14
- Bit 1~0 **PCS11~PCS10:** PC4 pin-shared function selection
 00: PC4
 01: PTP1
 10: SSEG2/SCOM2
 11: KEY13

• **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: PTP1
 10: TX
 11: AN3
- Bit 5~4 **PDS05~PDS04:** PD2 pin-shared function selection
 00: PD2
 01: PTP1
 10: RX/TX
 11: AN2
- Bit 3~2 **PDS03~PDS02:** PD1 pin-shared function selection
 00: PD1
 01: PTP1
 10: SSEG25/SCOM25
 11: AN1
- Bit 1~0 **PDS01~PDS00:** PD0 pin-shared function selection
 00: PD0
 01: PTP0
 10: SSEG24/SCOM24
 11: AN0

• **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
 01: SDI/SDA/URX/UTX
 10: TX
 11: AN7
- Bit 5~4 **PDS15~PDS14:** PD6 pin-shared function selection
 00: PD6/INT1
 01: SCK/SCL
 10: RX/TX
 11: AN6
- Bit 3~2 **PDS13~PDS12:** PD5 pin-shared function selection
 00: PD5/INT0/PTCK0
 01: PTP0
 10: SDO/UTX
 11: AN5
- Bit 1~0 **PDS11~PDS10:** PD4 pin-shared function selection
 00: PD4/PTCK1
 01: SCS
 10: VREF
 11: AN4

• **PES0 Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------|-------|-------|-------|-------|-------|-------|-------|-------|
| Name | PES07 | PES06 | PES05 | PES04 | PES03 | PES02 | PES01 | PES00 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| POR | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

- Bit 7~6 **PES07~PES06:** PE3 Pin-Shared function selection
 00: PE3/PTP1I
 01: PTP1B
 10: TX
 11: KEY20
- Bit 5~4 **PES05~PES04:** PE2 Pin-Shared function selection
 00: PE2/PTCK0
 01: CTP0
 10: RX/TX
 11: KEY19
- Bit 3~2 **PES03~PES02:** PE1 Pin-Shared function selection
 00: PE1/PTP0I
 01: SDO/UTX
 10: SSEG23/SCOM23
 11: KEY18
- Bit 1~0 **PES01~PES00:** PE0 Pin-Shared function selection
 00: PE0/PTCK1
 01: SDI/SDA/URX/UTX
 10: SSEG22/SCOM22
 11: KEY17

• **PES1 Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------|-------|-------|-------|-------|-------|-------|-------|-------|
| Name | PES17 | PES16 | PES15 | PES14 | PES13 | PES12 | PES11 | PES10 |
| 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 **PES17~PES16:** PE7 Pin-Shared function selection
 00: PE7/PTCK1
 01: PTP1
 10: OCPN
 11: AN8
- Bit 5~4 **PES15~PES14:** PE6 Pin-Shared function selection
 00: PE6/PTP1I
 01: CTP0
 10: OCPAO
 11: OCPCOUT
- Bit 3~2 **PES13~PES12:** PE5 Pin-Shared function selection
 00: PE5
 01: PTP1B
 10: OCPP
 11: CTP1
- Bit 1~0 **PES11~PES10:** PE4 Pin-Shared function selection
 00: PE4
 01: PTP1
 10: CTP0
 11: VDDIO

• **PFS0 Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------|-------|-------|-------|-------|-------|-------|-------|-------|
| Name | PFS07 | PFS06 | PFS05 | PFS04 | PFS03 | PFS02 | PFS01 | PFS00 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| POR | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

- Bit 7~6 **PFS07~PFS06:** PF3 Pin-Shared function selection
 00: PF3
 01: PTP0
 10: SSEG15/SCOM15
 11: SDI/SDA/URX/UTX
- Bit 5~4 **PFS05~PFS04:** PF2 Pin-Shared function selection
 00: PF2
 01: CTP0B
 10: SSEG14/SCOM14
 11: SDO/UTX
- Bit 3~2 **PFS03~PFS02:** PF1 Pin-Shared function selection
 00: PF1/CTCK0
 01: VREFI
 10: DACBUF
 11: TX
- Bit 1~0 **PFS01~PFS00:** PF0 Pin-Shared function selection
 00: PF0/CTCK1
 01: CTP0
 10: OCPVR
 11: RX/TX

• **PFS1 Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------|---|---|---|---|-------|-------|-------|-------|
| Name | — | — | — | — | PFS13 | PFS12 | PFS11 | PFS10 |
| R/W | — | — | — | — | R/W | R/W | R/W | R/W |
| POR | — | — | — | — | 0 | 0 | 0 | 0 |

- Bit 7~4 Unimplemented, read as “0”
- Bit 3~2 **PFS13~PFS12**: PF5 Pin-Shared function selection
 00: PF5
 01: PTP0
 10: SSEG17/SCOM17
 11: SCS
- Bit 1~0 **PFS11~PFS10**: PF4 Pin-Shared function selection
 00: PF4
 01: PTP1
 10: SSEG16/SCOM16
 11: SCK/SCL

• **IFS0 Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------|----------|----------|----------|----------|----------|----------|---------|---------|
| Name | PTP0IPS1 | PTP0IPS0 | PTCK1PS1 | PTCK1PS0 | PTCK0PS1 | PTCK0PS0 | CTCK1PS | CTCK0PS |
| 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 **PTP0IPS1~PTP0IPS0**: PTP0I input source pin selection
 00: PE1
 01: PB7
 10: PA4
 11: PA7
- Bit 5~4 **PTCK1PS1~PTCK1PS0**: PTCK1 input source pin selection
 00: PE7
 01: PE0
 10: PD4
 11: PB5
- Bit 3~2 **PTCK0PS1~PTCK0PS0**: PTCK0 input source pin selection
 00: PE2
 01: PD5
 10: PB6
 11: PA1
- Bit 1 **CTCK1PS**: CTCK1 input source pin selection
 0: PF0
 1: PA1
- Bit 0 **CTCK0PS**: CTCK0 input source pin selection
 0: PF1
 1: PC0

• IFS1 Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------|---|-------|-------|-------|--------|--------|----------|----------|
| Name | — | RXPS2 | RXPS1 | RXPS0 | INT1PS | INT0PS | PTP1IPS1 | PTP1IPS0 |
| 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 6~4 **RXPS2~RXPS0**: RX/TX input source pin selection
 000: PD6
 001: PA7
 010: PA2
 011: PA4
 100: PD2
 101: PE2
 110: PF0
 111: PD6
- Bit 3 **INT1PS**: INT1 input source pin selection
 0: PA0
 1: PD6
- Bit 2 **INT0PS**: INT0 input source pin selection
 0: PA4
 1: PD5
- Bit 1~0 **PTP1IPS1~PTP1IPS0**: PTP1I input source pin selection
 00: PE6
 01: PE3
 10: PB4
 11: PA3

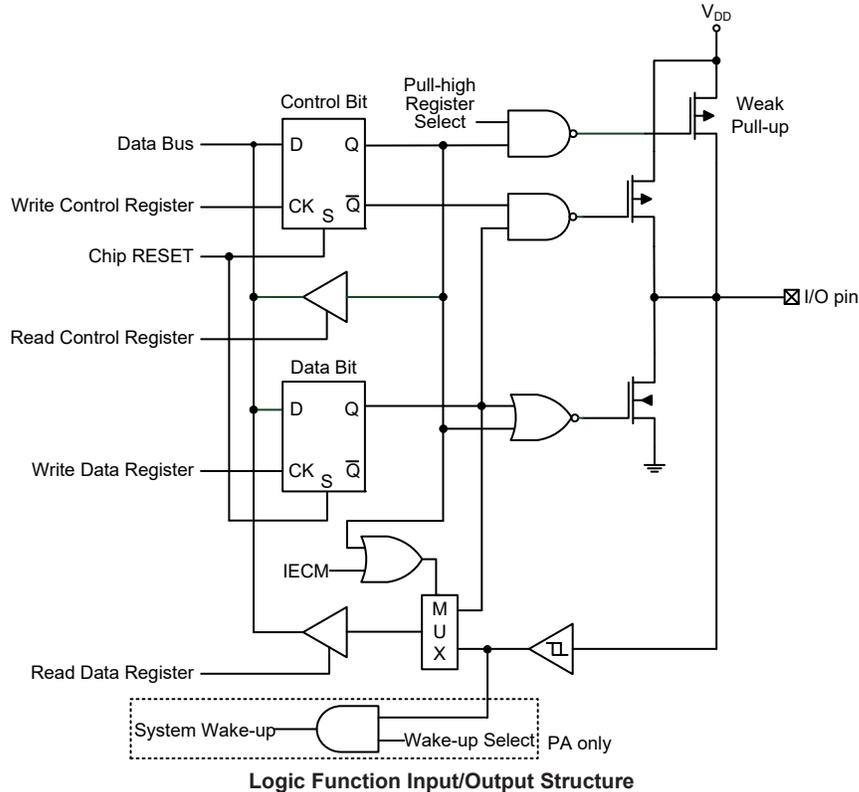
• IFS2 Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------|---|--------|--------|--------|--------|--------|--------|--------|
| Name | — | SCSBPS | SCKPS2 | SCKPS1 | SCKPS0 | URXPS2 | URXPS1 | URXPS0 |
| R/W | — | R/W |
| POR | — | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

- Bit 7 Unimplemented, read as “0”
- Bit 6 **SCSBPS**: SCS input source pin selection
 0: PD4
 1: PF5
- Bit 5~3 **SCKPS2~SCKPS0**: SCK/SCL input source pin selection
 000: PC6
 001: PA7
 010: PC2
 011: PD6
 100: PF4
 101: PC6
 110: PC6
 111: PC6
- Bit 2~0 **URXPS2~URXPS0**: SDI/SDA/URX/UTX input source pin selection
 000: PD7
 001: PA3
 010: PC2
 011: PC3
 100: PC7
 101: PE0
 110: PF3
 111: 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.



READ PORT Function

The READ PORT function is used to manage the reading of the output data from the data latch or I/O pin, which is specially designed for the IEC 60730 self-diagnostic test on the I/O function and A/D paths. There is a register, IECC, which is used to control the READ PORT function. If the READ PORT function is disabled, the pin function will operate as the selected pin-shared function. When a specific data pattern, "11001010", is written into the IECC register, the internal signal named IECM will be set high to enable the READ PORT function. If the READ PORT function is enabled, the value on the corresponding pins will be passed to the accumulator ACC when the read port instruction "mov a, Px" is executed where the "x" stands for the corresponding I/O port name.

Note that the READ PORT mode can only control the input path and will not affect the pin-shared function assignment and the current MCU operation. However, when the IECC register content is set to any other values rather than "11001010", the IECM internal signal will be cleared to 0 to disable the READ PORT function, and the reading path will be from the data latch. If the READ PORT function is disabled, the pin function will operate as the selected pin-shared function.

• **IECC Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------|-------|-------|-------|-------|-------|-------|-------|-------|
| Name | IECS7 | IECS6 | IECS5 | IECS4 | IECS3 | IECS2 | IECS1 | IECS0 |
| 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 |

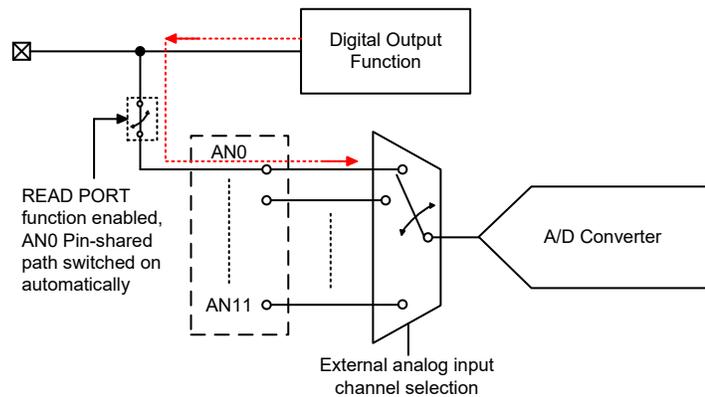
Bit 7~0 **IECS7~IECS0**: READ PORT function enable control bit 7~ bit 0
 11001010: IECM=1 – READ PORT function is enabled
 Others: IECM=0 – READ PORT function is disabled

| READ PORT Function Port Control Register Bit – PxC.n | Disabled | | Enabled | |
|---|-----------|------------------|-----------|---|
| | 1 | 0 | 1 | 0 |
| I/O Function | Pin value | Data latch value | Pin value | |
| Digital Input Function | | | | |
| Digital Output Function (except USIM and UART) | | | | |
| USIM: SCK/SCL, SDI/SDA, URX/UTX, UTX UART: RX/TX | | | | |
| Analog Function | 0 | | | |

Note: The value on the above table is the content of the ACC register after “mov a, Px” instruction is executed where “x” means the relevant port name.

The additional function of the READ PORT mode is to check the A/D path. When the READ PORT function is disabled, the A/D path from the external pin to the internal analog input will be switched off if the A/D input pin function is not selected by the corresponding selection bits. For the MCU with A/D converter channels, such as A/D AN0~AN11, the desired A/D channel can be switched on by properly configuring the external analog input channel selection bits in the A/D Control Register together with the corresponding analog input pin function is selected. However, the additional function of the READ PORT mode is to force the A/D path to be switched on. For example, when the AN0 is selected as the analog input channel as the READ PORT function is enabled, the AN0 analog input path will be switched on even if the AN0 analog input pin function is not selected. In this way, the AN0 analog input path can be examined by internally connecting the digital output on this shared pin with the AN0 analog input pin switch and then converting the corresponding digital data without any external analog input voltage connected.

Note that the A/D converter reference voltage should be equal to the I/O power supply voltage when examining the A/D path using the READ PORT function.



A/D Channel Input Path Internally Connection

Programming Considerations

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

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

Timer Modules – TM

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

The common features of the different TM types are described here with more detailed information provided in the individual Compact and 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 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 and Periodic TMs will be described in this section. The detailed operation regarding each of the TM types will be described in separate sections. The main features and differences between the two types of TMs are summarised in the accompanying table.

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

TM Function Summary

TM Operation

The different types of TM offer a diverse range of functions, from simple timing operations to PWM signal generation. The key to understanding how the TM operates is to see it in terms of a free running counter whose value is then compared with the value of pre-programmed internal comparators. When the free running 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 or P type TM and “n” stands for the specific TM serial number. The clock source can be a ratio of the system clock f_{SYS} or the internal high clock f_{IH} , 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 or for event counting.

TM Interrupts

Each Compact or Periodic type TM has two internal interrupts, one for each of the internal comparator A or comparator P, which generates 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 input pin with the label xTCKn while the Periodic TMs have another input pin with the label PTPnI. 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 PTCKn pin is also used as the external trigger input pin in single pulse output mode for the PTMn.

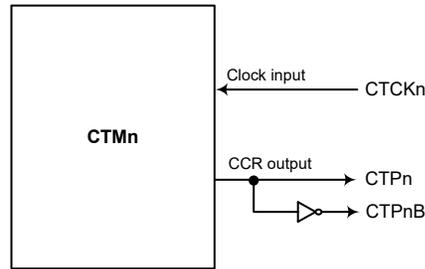
The Periodic type TM has another input pin, PTPnI, which is the capture input whose active edge can be a rising edge, a falling edge or both rising and falling edges and the active edge transition type is selected using the PTnIO1~PTnIO0 bits in the PTMnC1 register. There is another capture input, PTCKn, for PTMn capture input mode, which can be used as the external trigger input source except the PTPnI pin.

The TMs each has two output pins with label xTPn and the xTPnB. When the TM is in the Compare Match Output Mode, the xTPn 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 xTPnB pin output is the inverted signal of the xTPn. The external output pins are also the pins where the TM generates the PWM output waveform.

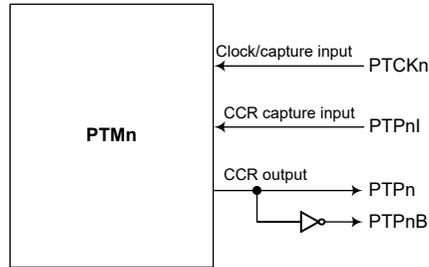
As the TM input and output pins are pin-shared with other functions, the TM input and output functions must first be selected using the relevant pin-shared function selection bits described in the Pin-shared Function section.

| CTM | | PTM | |
|-----------------|-----------------------------|-------------------------------|-----------------------------|
| Input | Output | Input | Output |
| CTCK0; CTCK1 | CTP0, CTP0B; CTP1, CTP1B | PTCK0, PTP0I; PTCK1, PTP1I | PTP0, PTP0B; PTP1, PTP1B |

TM External Pins



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

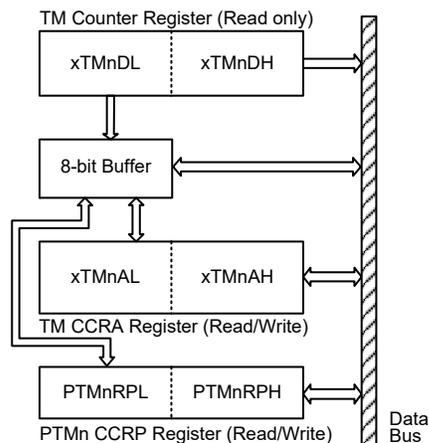


PTMn Function Pin Block Diagram (n=0~1)

Programming Considerations

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

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

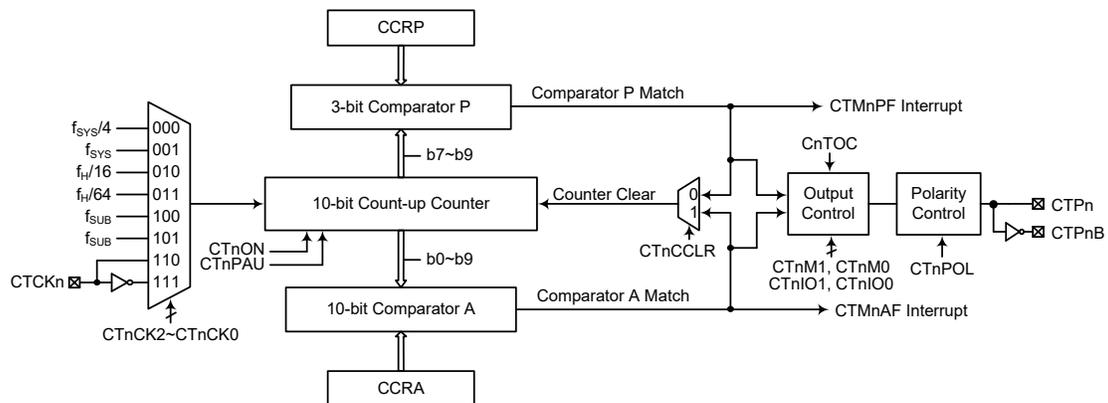


The following steps show the read and write procedures:

- 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 PTMnRPL
 - 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, 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

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



- Note: 1. As the CTMn external pins are pin-shared with other functions, so before using the CTMn function the relevant pin-shared function registers must be set properly to enable the CTMn pin function. The CTCKn pins, if used, must also be set as an input by setting the corresponding bits in the port control register.
2. The CTPnB is the inverted signal of the CTPn.

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

Compact Type TM Operation

The size of Compact 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 is three bits wide whose value is compared with the highest 3 bits in the counter while the CCRA is the 10 bits and therefore compares with all counter bits.

The only way of changing the value of the 10-bit counter using the application program, is to clear the counter by changing the CTnON 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 CTMn interrupt signal will also usually be generated. The Compact Type TM can operate in a number of different operational modes, can be driven by different clock sources including an input pin and can also control two output pins. All operating setup conditions are selected using relevant internal registers.

Compact Type TM Register Description

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

| Register Name | Bit | | | | | | | |
|---------------|--------|--------|--------|--------|-------|--------|--------|---------|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CTMnC0 | CTnPAU | CTnCK2 | CTnCK1 | CTnCK0 | CTnON | CTnRP2 | CTnRP1 | CTnRP0 |
| CTMnC1 | CTnM1 | CTnM0 | CTnIO1 | CTnIO0 | CTnOC | CTnPOL | CTnDPX | CTnCCLR |
| CTMnDL | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| CTMnDH | — | — | — | — | — | — | D9 | D8 |
| CTMnAL | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| CTMnAH | — | — | — | — | — | — | D9 | D8 |

10-bit Compact TM Register List (n=0~1)

• CTMnC0 Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------|--------|--------|--------|--------|-------|--------|--------|--------|
| Name | CTnPAU | CTnCK2 | CTnCK1 | CTnCK0 | CTnON | CTnRP2 | CTnRP1 | CTnRP0 |
| 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 **CTnPAU**: CTMn 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 CTMn 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 **CTnCK2~CTnCK0**: Select CTMn Counter clock
 000: $f_{SYS}/4$
 001: f_{SYS}
 010: $f_H/16$
 011: $f_H/64$
 100: f_{SUB}
 101: f_{SUB}
 110: CTCKn rising edge clock
 111: CTCKn falling edge clock

These three bits are used to select the clock source for the CTMn. 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 **CTnON**: CTMn Counter On/Off Control
 0: Off
 1: On
- This bit controls the overall on/off function of the CTMn. Setting the bit high enables the counter to run, clearing the bit disables the CTMn. Clearing this bit to zero will stop the counter from counting and turn off the CTMn 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 CTMn is in the Compare Match Output Mode or the PWM Output Mode then the CTMn output pin will be reset to its initial condition, as specified by the CTnOC bit, when the CTnON bit changes from low to high.
- Bit 2~0 **CTnRP2~CTnRP0**: CTMn CCRP 3-bit register, compared with the CTMn Counter bit 9~bit 7 Comparator P Match Period=
 000: 1024 CTMn clocks
 001: 128 CTMn clocks
 010: 256 CTMn clocks
 011: 384 CTMn clocks
 100: 512 CTMn clocks
 101: 640 CTMn clocks
 110: 768 CTMn clocks
 111: 896 CTMn clocks
- These three bits are used to setup the value on the internal CCRP 3-bit register, which are then compared with the internal counter's highest three bits. The result of this comparison can be selected to clear the internal counter if the CTnCCLR bit is set to zero. Setting the CTnCCLR bit to zero ensures that a compare match with the CCRP values will reset the internal counter. As the CCRP bits are only compared with the highest three counter bits, the compare values exist in 128 clock cycle multiples. Clearing all three bits to zero is in effect allowing the counter to overflow at its maximum value.

• **CTMnC1 Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------|-------|-------|--------|--------|-------|--------|--------|---------|
| Name | CTnM1 | CTnM0 | CTnIO1 | CTnIO0 | CTnOC | CTnPOL | CTnDPX | CTnCCLR |
| 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 **CTnM1~CTnM0**: Select CTMn operating mode
 00: Compare Match Output Mode
 01: Undefined
 10: PWM Output Mode
 11: Timer/Counter Mode
- These bits setup the required operating mode for the CTMn. To ensure reliable operation the CTMn should be switched off before any changes are made to the CTnM1 and CTnM0 bits. In the Timer/Counter Mode, the CTMn output pin state is undefined.
- Bit 5~4 **CTnIO1~CTnIO0**: Select CTMn 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 CTMn output pin changes state when a certain condition is reached. The function that these bits select depends upon in which mode the CTMn is running.

In the Compare Match Output Mode, the CTnIO1 and CTnIO0 bits determine how the CTMn output pin changes state when a compare match occurs from the Comparator A. The CTMn 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 CTMn output pin should be setup using the CTnOC bit in the CTnMC1 register. Note that the output level requested by the CTnIO1 and CTnIO0 bits must be different from the initial value setup using the CTnOC bit otherwise no change will occur on the CTMn output pin when a compare match occurs. After the CTMn output pin changes state it can be reset to its initial level by changing the level of the CTnON bit from low to high.

In the PWM Output Mode, the CTnIO1 and CTnIO0 bits determine how the CTMn 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 CTnIO1 and CTnIO0 bits only after the CTMn has been switched off. Unpredictable PWM outputs will occur if the CTnIO1 and CTnIO0 bits are changed when The CTMn is running.

Bit 3 **CTnOC**: CTPn Output control bit

Compare Match Output Mode

0: Initial low

1: Initial high

PWM Output Mode

0: Active low

1: Active high

This is the output control bit for the CTMn output pin. Its operation depends upon whether CTMn is being used in the Compare Match Output Mode or in the PWM Output Mode. It has no effect if the CTMn is in the Timer/Counter Mode. In the Compare Match Output Mode it determines the logic level of the CTMn 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 **CTnPOL**: CTPn Output polarity Control

0: Non-invert

1: Invert

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

Bit 1 **CTnDPX**: CTMn PWM period/duty Control

0: CCRP – period, CCRA – duty

1: CCRP – duty; CCRA – period

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

Bit 0 **CTnCCLR**: Select CTMn Counter clear condition

0: CTMn Comparatror P match

1: CTMn Comparatror A match

This bit is used to select the method which clears the counter. Remember that the CTMn contains two comparators, Comparator A and Comparator P, either of which can be selected to clear the internal counter. With the CTnCCLR 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 CTnCCLR bit is not used in the PWM Output Mode.

• **CTMnDL 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**: CTMn Counter Low Byte Register bit 7 ~ bit 0
CTMn 10-bit Counter bit 7 ~ bit 0

• **CTMnDH 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**: CTMn Counter High Byte Register bit 1 ~ bit 0
CTMn 10-bit Counter bit 9 ~ bit 8

• **CTMnAL 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**: CTMn CCRA Low Byte Register bit 7 ~ bit 0
CTMn 10-bit CCRA bit 7 ~ bit 0

• **CTMnAH 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**: CTMn CCRA High Byte Register bit 1 ~ bit 0
CTMn 10-bit CCRA bit 9 ~ bit 8

Compact Type TM Operating Modes

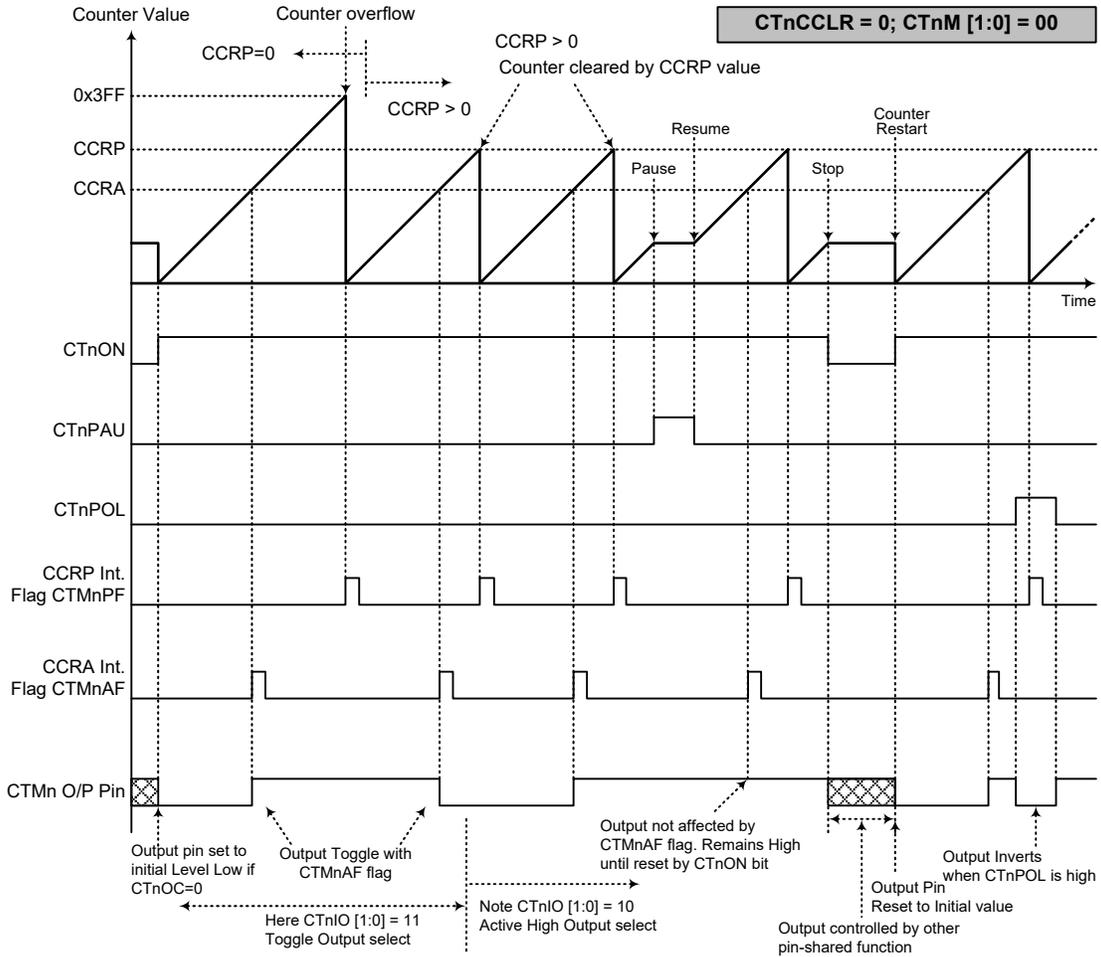
The Compact Type TM can operate in one of three operating modes, Compare Match Output Mode, PWM Output Mode or Timer/Counter Mode. The operating mode is selected using the CTnM1 and CTnM0 bits in the CTMnC1 register.

Compare Match Output Mode

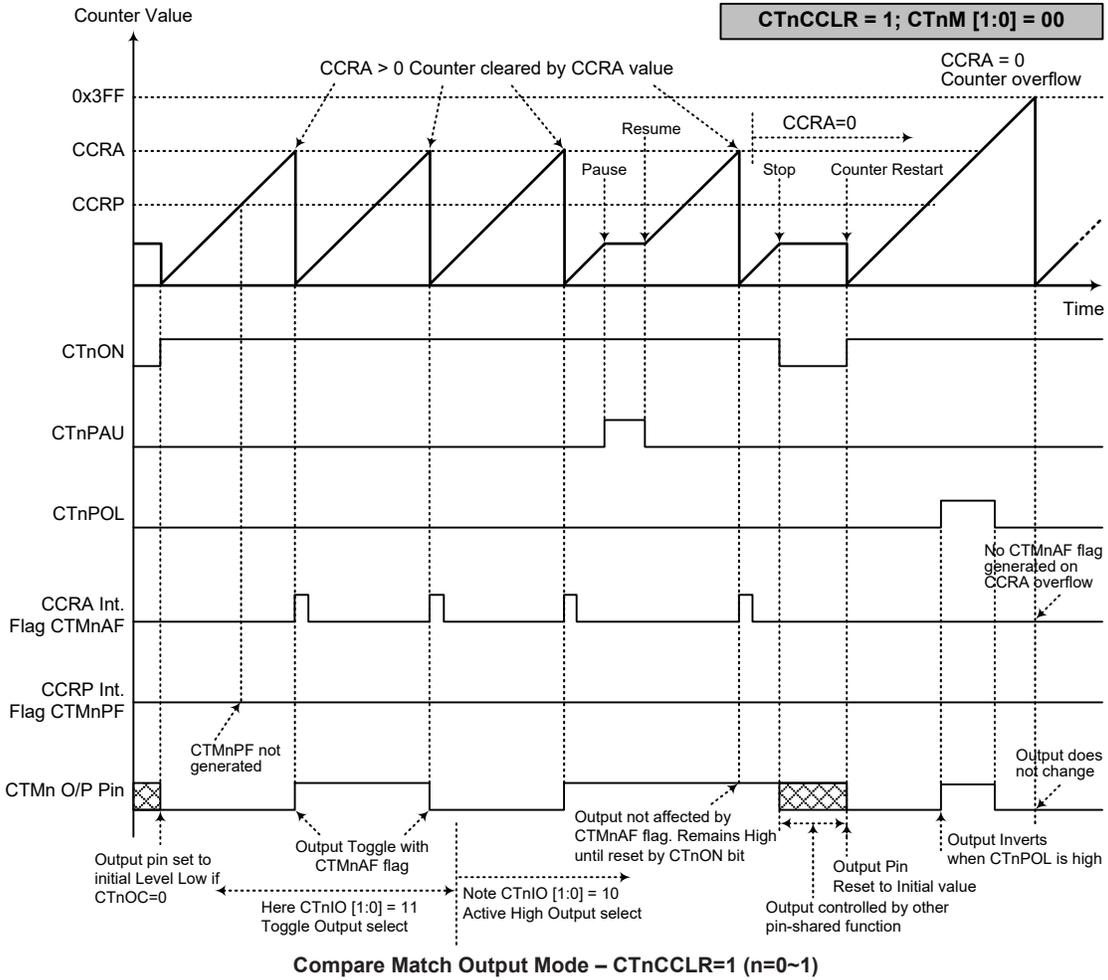
To select this mode, bits CTnM1 and CTnM0 in the CTnMC1 register, should be set to 00 respectively. In this mode once the counter is enabled and running it can be cleared by three methods. These are a counter overflow, a compare match from Comparator A and a compare match from Comparator P. When the CTnCCLR 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 CTMnAF and CTMnPF interrupt request flags for Comparator A and Comparator P respectively, will both be generated.

If the CTnCCLR bit in the CTMnC1 register is high then the counter will be cleared when a compare match occurs from Comparator A. However, here only the CTMnAF interrupt request flag will be generated even if the value of the CCRP bits is less than that of the CCRA registers. Therefore when CTnCCLR is high no CTMnPF interrupt request flag will be generated. If the CCRA bits are all zero, the counter will overflow when it reaches its maximum 10-bit, 3FF Hex, value, however here the CTMnAF interrupt request flag will not be generated.

As the name of the mode suggests, after a comparison is made, the CTMn output pin will change state. The CTMn output pin condition however only changes state when a CTMnAF interrupt request flag is generated after a compare match occurs from Comparator A. The CTMnPF interrupt request flag, generated from a compare match occurs from Comparator P, will have no effect on the CTMn output pin. The way in which the CTMn output pin changes state are determined by the condition of the CTnIO1 and CTnIO0 bits in the CTMnC1 register. The CTMn output pin can be selected using the CTnIO1 and CTnIO0 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 CTMn output pin, which is setup after the CTnON bit changes from low to high, is setup using the CTnOC bit. Note that if the CTnIO1 and CTnIO0 bits are zero then no pin change will take place.



- Note: 1. With $CTnCCR=0$, a Comparator P match will clear the counter
 2. The $CTMn$ output pin controlled only by the $CTMnAF$ flag
 3. The output pin reset to initial state by a $CTnON$ bit rising edge



- Note: 1. With CTnCCR=1, a Comparator A match will clear the counter
2. The CTMn output pin controlled only by the CTMnAF flag
3. The output pin reset to initial state by a CTnON rising edge
4. The CTMnPF flags is not generated when CTnCCR=1

Timer/Counter Mode

To select this mode, bits CTnM1 and CTnM0 in the CTMnC1 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 CTMn 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 CTMn output pin is not used in this mode, the pin can be used as a normal I/O pin or other pin-shared function.

PWM Output Mode

To select this mode, bits CTnM1 and CTnM0 in the CTMnC1 register should be set to 10 respectively. The PWM function within the CTMn 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 CTMn 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 CTnCCLR 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 CTnDPX bit in the CTMnC1 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 CTnOC bit in the CTMnC1 register is used to select the required polarity of the PWM waveform while the two CTnIO1 and CTnIO0 bits are used to enable the PWM output or to force the CTMn output pin to a fixed high or low level. The CTnPOL bit is used to reverse the polarity of the PWM output waveform.

• CTMn, PWM Output Mode, Edge-aligned Mode, CTnDPX=0

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

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

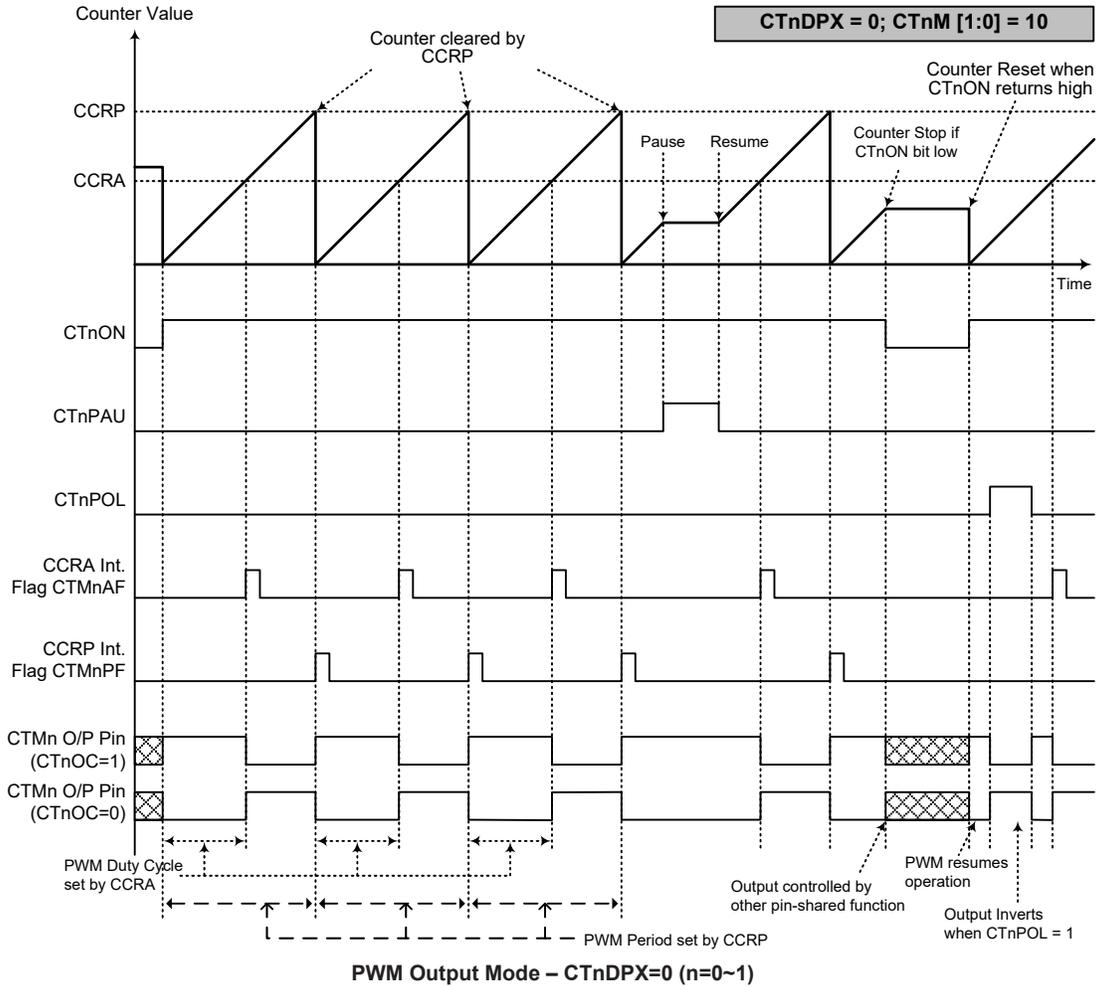
The CTMn PWM output frequency= $(f_{SYS}/4)/512=f_{SYS}/2048=3.9063\text{kHz}$, duty=128/512=25%.

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%.

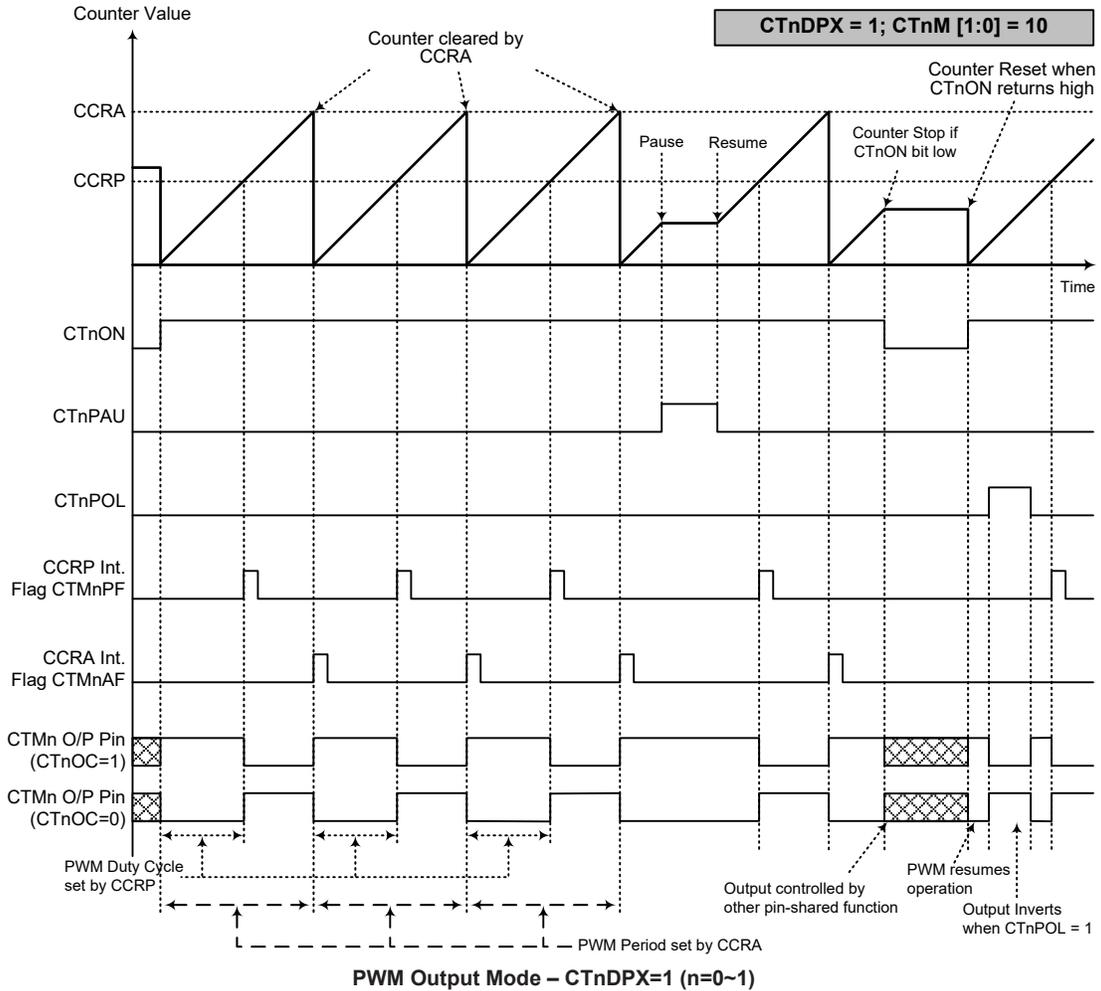
• CTMn, PWM Output Mode, Edge-aligned Mode, CTnDPX=1

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

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



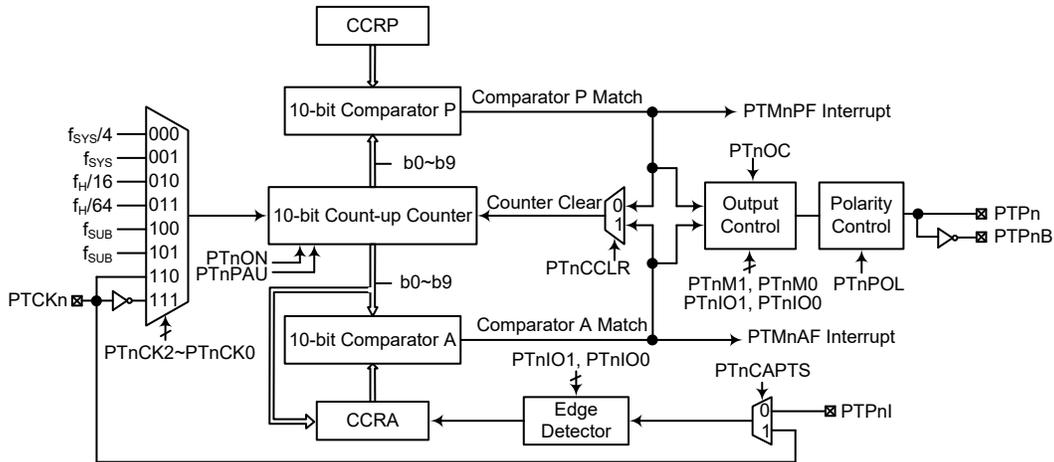
- Note: 1. Here CTnDPX=0 – Counter cleared by CCRP
 2. A counter clear sets PWM Period
 3. The internal PWM function continues running even when CTnIO[1:0]=00 or 01
 4. The CTnCCLR bit has no influence on PWM operation



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

Periodic Type TM – PTM

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



- Note: 1. The PTMn external pins are pin-shared with other functions, so before using the PTMn function, ensure that the pin-shared function registers have been set properly to enable the PTMn pin function. The PTCKn and PTPnI pins, if used, must also be set as an input by setting the corresponding bits in the port control register.
2. The PTPnB is the inverted signal of the PTPn.

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 two input pins and can also control two output pins. All operating setup conditions are selected using relevant internal registers.

Periodic Type TM Register Description

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

| Register Name | Bit | | | | | | | |
|---------------|--------|--------|--------|--------|--------|--------|----------|---------|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| PTMnC0 | PTnPAU | PTnCK2 | PTnCK1 | PTnCK0 | PTnON | — | — | — |
| PTMnC1 | PTnM1 | PTnM0 | PTnIO1 | PTnIO0 | PTnOC | PTnPOL | PTnCAPTS | 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)

• **PTMnC0 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 | PTnCAPTS | 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: Capture Input Mode
 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

Capture Input Mode

- 00: Input capture at rising edge of PTPnI or PTCKn
- 01: Input capture at falling edge of PTPnI or PTCKn
- 10: Input capture at both rising/falling edge of PTPnI or PTCKn
- 11: Input capture disabled

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 PTMn 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 PTMn 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 PTMn is in the Timer/Counter Mode.
- Bit 1 **PTnCAPTS**: PTMn Capture Trigger Source selection
 0: From PTPnI pin
 1: From PTCKn pin
- 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, Single Pulse Output or Capture Input 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 five operating modes, Compare Match Output Mode, PWM Output Mode, Single Pulse Output Mode, Capture Input Mode or Timer/Counter Mode. The operating mode is selected using the PTnM1 and PTnM0 bits in the PTMnC1 register.

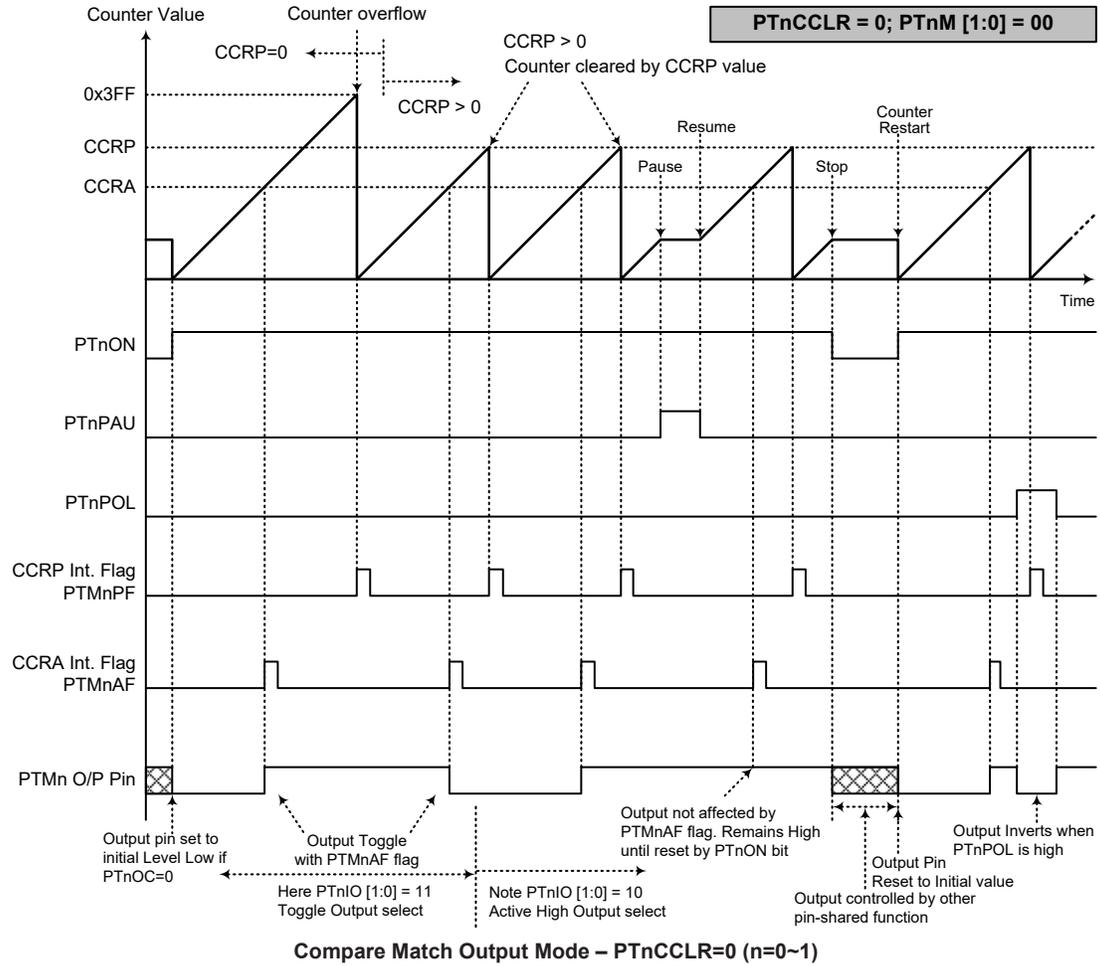
Compare Match Output Mode

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.

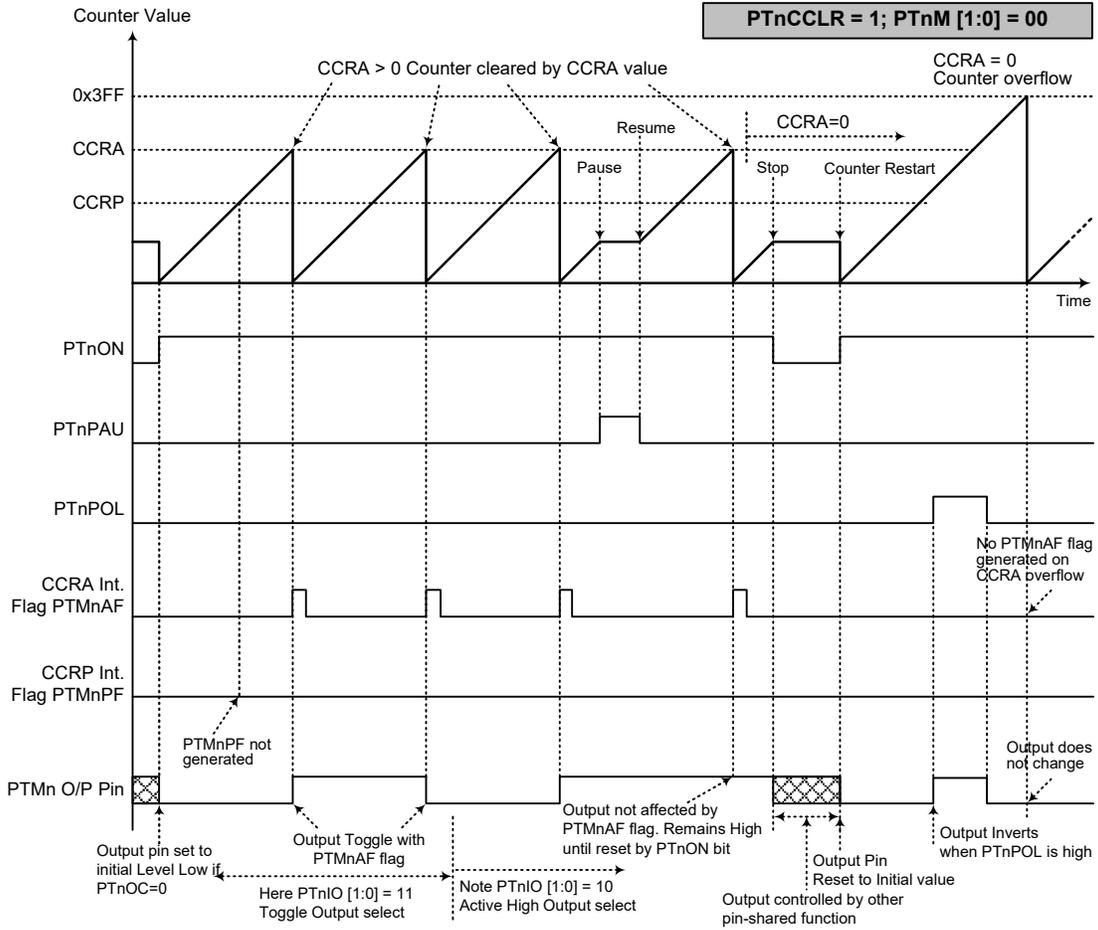
If the PTnCCLR bit in the PTMnC1 register is high then the counter will be cleared when a compare match occurs from Comparator A. However, here only the PTMnAF interrupt request flag will be generated even if the value of the CCRP bits is less than that of the CCRA registers. Therefore when PTnCCLR is 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 it 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 on 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.

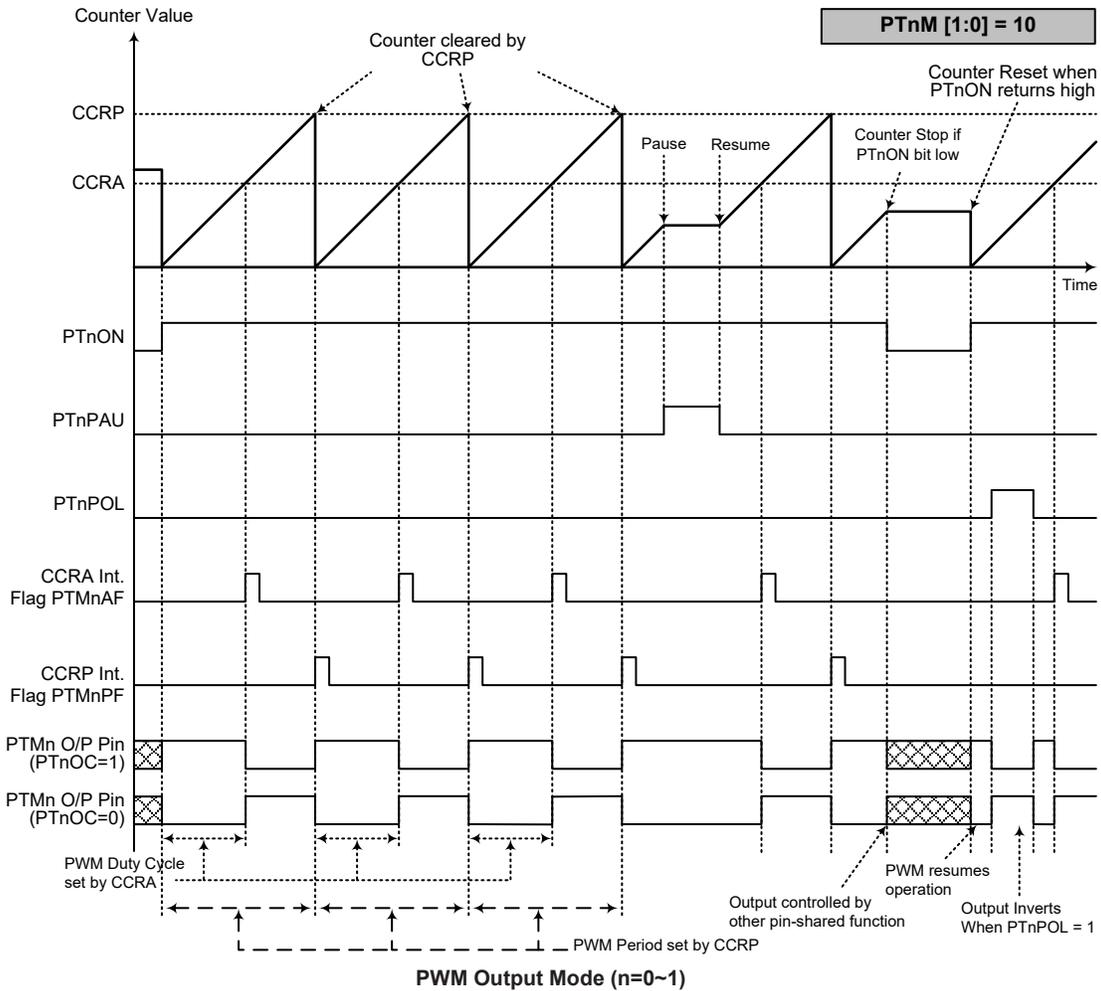
• **10-bit PWM Output Mode, Edge-aligned Mode**

| 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%.



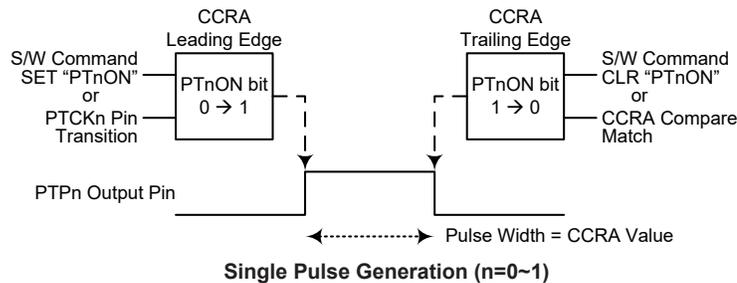
- Note:
1. The counter is cleared by CCRP
 2. A counter clear sets the PWM Period
 3. The internal PWM function continues running even when PTnIO[1:0]=00 or 01
 4. The PTnCCLR bit has no influence on PWM operation

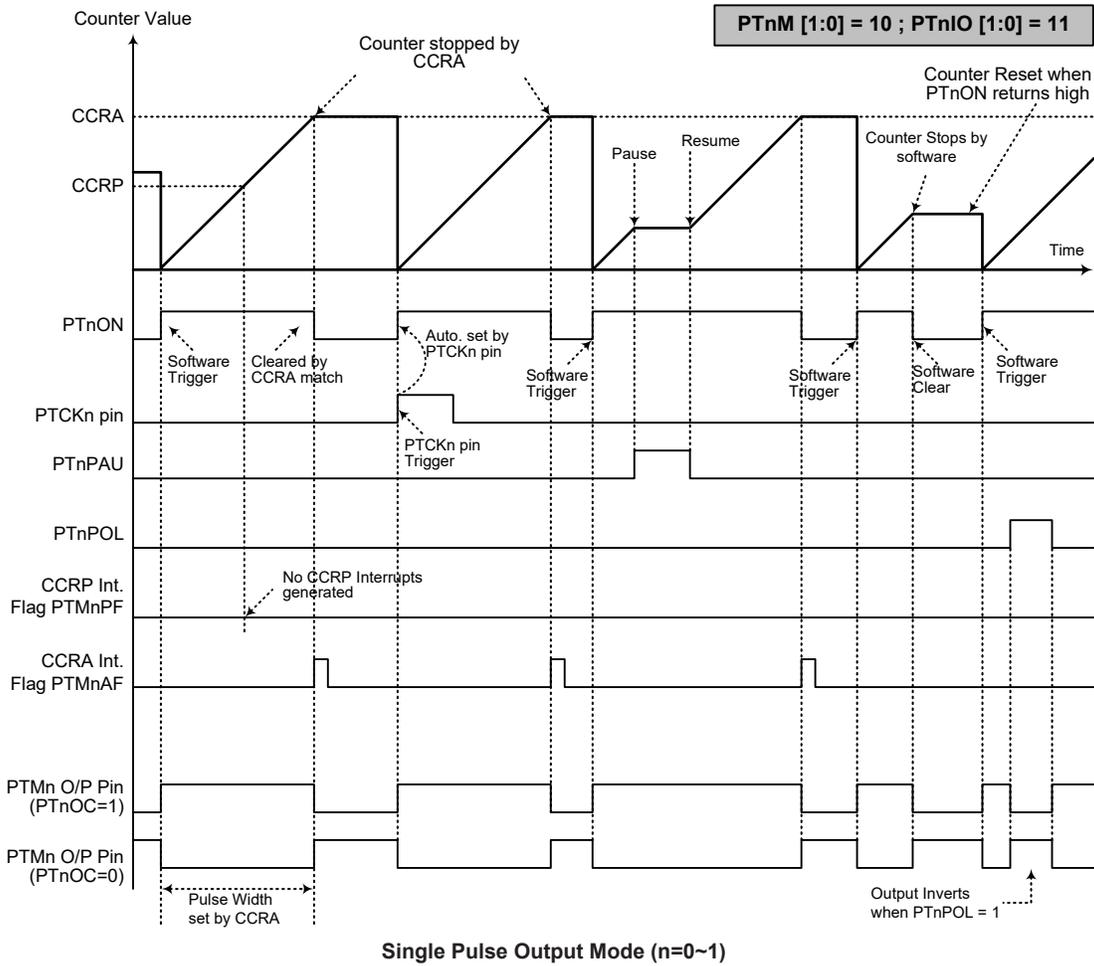
Single Pulse Output Mode

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.





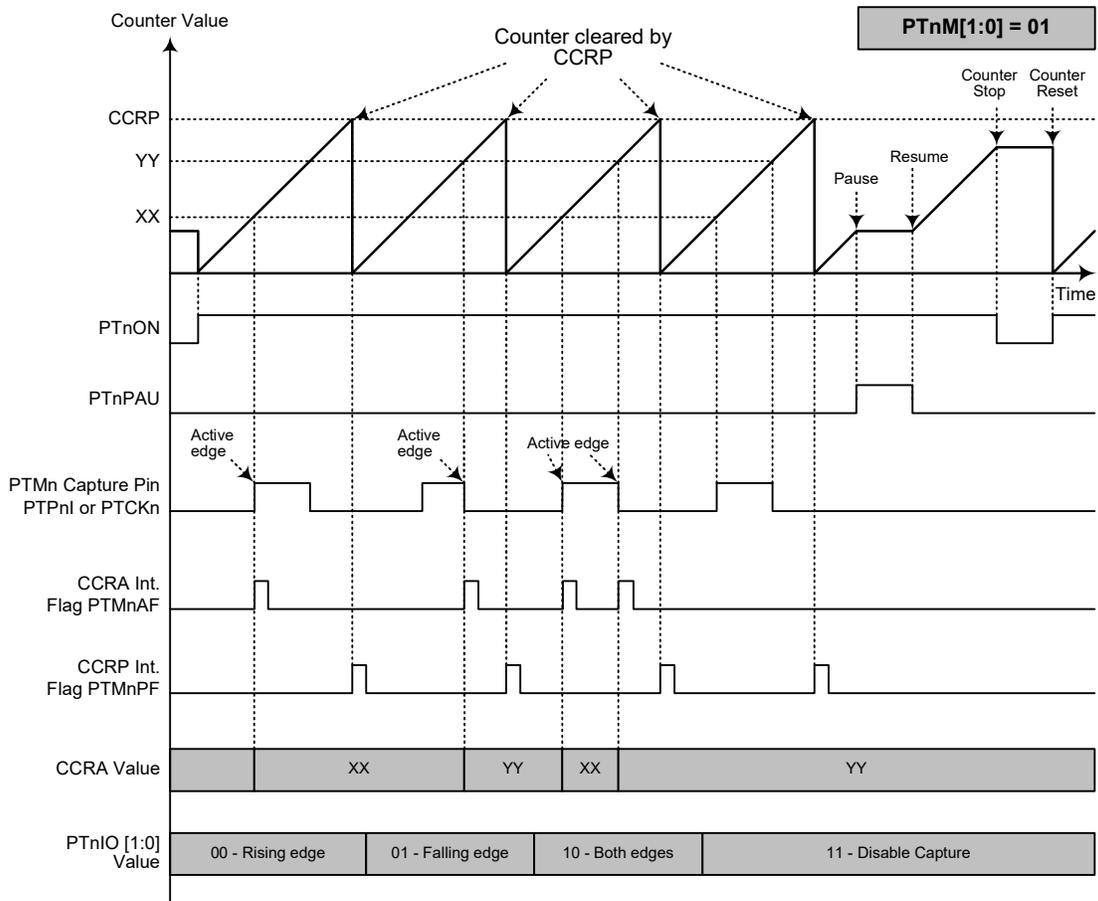
- Note:
1. Counter stopped by CCRA
 2. CCRP is not used
 3. The pulse triggered by the PTCKn pin or by setting the PTnON bit high
 4. A PTCKn pin active edge will automatically set the PTnON bit high.
 5. In the Single Pulse Output Mode, PTnIO[1:0] must be set to "11" and can not be changed

Capture Input Mode

To select this mode, bits PTnM1 and PTnM0 in the PTMnC1 register should be set to “01” respectively. This mode enables external signals to capture and store the present value of the internal counter and can therefore be used for applications such as pulse width measurements. The external signal is supplied on the PTPnI or PTCKn pin which is selected using the PTnCPTS bit in the PTMnC1 register. The input pin active edge can be either a rising edge, a falling edge or both rising and falling edges; the active edge transition type is selected using the PTnIO1 and PTnIO0 bits in the PTMnC1 register. The counter is started when the PTnON bit changes from low to high which is initiated using the application program.

When the required edge transition appears on the PTPnI or PTCKn pin the present value in the counter will be latched into the CCRA registers and a PTMn interrupt generated. Irrespective of what events occur on the PTPnI or PTCKn pin, the counter will continue to free run until the PTnON bit changes from high to low. When a CCRP compare match occurs the counter will reset back to zero; in this way the CCRP value can be used to control the maximum counter value. When a CCRP compare match occurs from Comparator P, a PTMn interrupt will also be generated. Counting the number of overflow interrupt signals from the CCRP can be a useful method in measuring long pulse widths. The PTnIO1 and PTnIO0 bits can select the active trigger edge on the PTPnI or PTCKn pin to be a rising edge, falling edge or both edge types. If the PTnIO1 and PTnIO0 bits are both set high, then no capture operation will take place irrespective of what happens on the PTPnI or PTCKn pin, however it must be noted that the counter will continue to run. The PTnCCLR, PTnOC and PTnPOL bits are not used in this mode.

There are some considerations that should be noted. If PTCKn is used as the capture input source, then it cannot be selected as the PTMn clock source. If the captured pulse width is less than 2 timer clock periods, it may be ignored by hardware. After the counter value is latched to the CCRA registers by an active capture edge, the PTMnAF flag will be set high after 0.5 timer clock period. The delay time from the active capture edge received to the action of latching counter value to CCRA registers is less than 1.5 timer clock periods.



- Note: 1. PTnM[1:0]=01 and active edge set by the PTnIO[1:0] bits
 2. A PTMn Capture input pin active edge transfers the counter value to CCRA
 3. PTnCCLR bit not used
 4. No output function – PTnOC and PTnPOL bits are not used
 5. CCRP determines the counter value and the counter has a maximum count value when CCRP is equal to zero
 6. The capture input mode cannot be used if the selected PTM counter clock is not available

Analog to Digital Converter

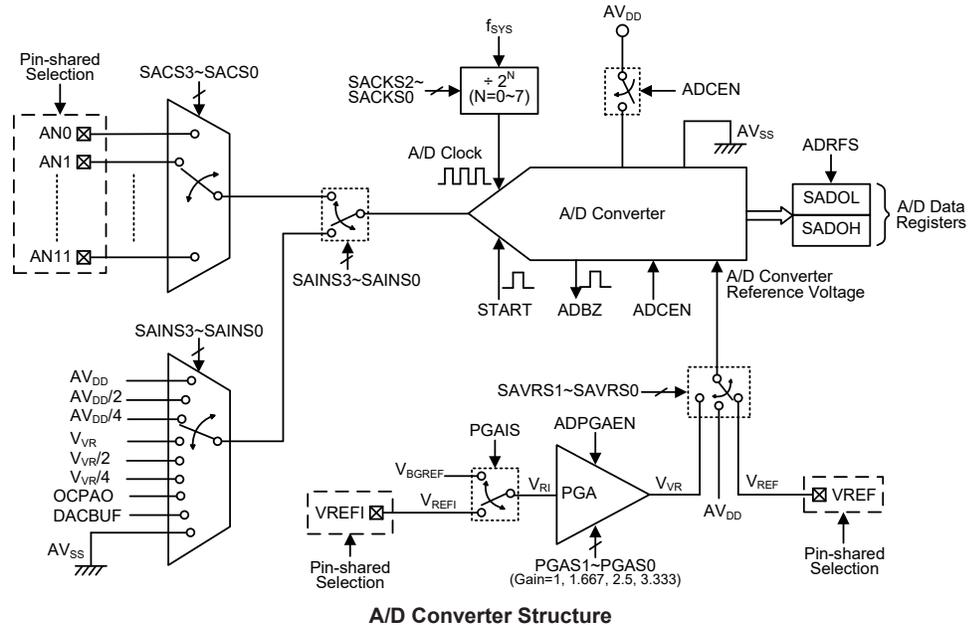
The need to interface to real world analog signals is a common requirement for many electronic systems. However, to properly process these signals by a microcontroller, they must first be converted into digital signals by A/D converters. By integrating the A/D conversion electronic circuitry into the 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 reference voltage, into a 12-bit digital value. The external or internal analog signal to be converted is determined by the SAINS3~SAINS0 and SACS3~SACS0 bits. Note that when the internal analog signal is selected to be converted using the SAINS3~SAINS0 bits, the external channel analog input will automatically be switched off. More detailed information about the A/D input signal selection will be described in the “A/D Converter Input Signals” section.

| External Input Channels | Internal Signal | A/D Signal Select |
|-------------------------|--|------------------------------|
| AN0~AN11 | AV _{DD} , AV _{DD} /2, AV _{DD} /4, V _{VR} , V _{VR} /2, V _{VR} /4, OCPAO, DACBUF, AV _{SS} | SAINS3~SAINS0 SACS3~SACS0 |

The accompanying block diagram shows the internal structure of the A/D converter with its associated registers and control bits.



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 |
| SAD0H (ADRF5=0) | D11 | D10 | D9 | D8 | D7 | D6 | D5 | D4 |
| SAD0H (ADRF5=1) | — | — | — | — | D11 | D10 | D9 | D8 |
| SADC0 | START | ADBZ | ADCEN | ADRF5 | SACS3 | SACS2 | SACS1 | SACS0 |
| SADC1 | 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, SAD0H

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

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

A/D Converter Data Registers

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

To control the function and operation of the A/D converter, three control registers known as SADC0, SADC1 and SADC2 are provided. These 8-bit registers define functions such as the selection of which analog signal is connected to the internal A/D converter, the digitised data format, the A/D clock source as well as controlling the start function and monitoring the A/D converter busy status. 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 SAINS3~SAINS0 bits in the SADC1 register and the SACS3~SACS0 bits in the SADC0 register are used to determine which analog signal derived from the external or internal signals will be connected to the A/D converter. 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. When the pin is selected to be an A/D input, its original function whether it is an I/O or other pin-shared function will be removed. In addition, any internal pull-high resistor connected to the pin will be automatically removed if the pin is selected to be an A/D converter input.

• **SADC0 Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------|-------|------|-------|-------|-------|-------|-------|-------|
| Name | START | ADBZ | ADCEN | 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
This bit is used to initiate an A/D conversion process. The bit is normally low but if set high and then cleared low again, the A/D converter will initiate a conversion process.
- Bit 6** **ADBZ:** A/D Converter busy flag
0: No A/D conversion is in progress
1: A/D conversion is in progress
This read only flag is used to indicate whether the A/D conversion is in progress or not. When the START bit is set from low to high and then to low again, the ADBZ flag will be set to 1 to indicate that the A/D conversion is initiated. The ADBZ flag will be cleared to 0 after the A/D conversion is complete.
- Bit 5** **ADCEN:** A/D Converter function enable control
0: Disable
1: Enable
This bit controls the A/D internal function. This bit should be set to one to enable the A/D converter. If the bit is 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 data register pair known as SADOH and SADOL will be unchanged.
- Bit 4** **ADRF5:** A/D conversion data format selection
0: A/D converter data format → SADOH=D[11:4]; SADOL=D[3:0]
1: A/D converter data format → SADOH=D[11:8]; SADOL=D[7:0]
This bit controls the format of the 12-bit converted A/D value in the two A/D data registers. Details are provided in the A/D 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: Undefined, input 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 source – External analog channel input, ANn
 0001: Internal source – Internal A/D converter power supply voltage AV_{DD}
 0010: Internal source – Internal A/D converter power supply voltage $AV_{DD}/2$
 0011: Internal source – Internal A/D converter power supply voltage $AV_{DD}/4$
 0100: External source – External analog channel input, ANn
 0101: Internal source – Internal A/D converter PGA output voltage V_{VR}
 0110: Internal source – Internal A/D converter PGA output voltage $V_{VR}/2$
 0111: Internal source – Internal A/D converter PGA output voltage $V_{VR}/4$
 1000: Internal source – OCP operational amplifier output, OCPAO
 1001: Internal source – OCP D/A converter output with buffer, DACBUF
 1010~1011: Internal source – Ground
 1100~1111: External source – External analog channel input, ANn

When the internal analog signal is selected to be converted, the external channel signal input will automatically be switched off regardless of the SACS3~SACS0 bit values. 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: $f_{SYS}/128$

• **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 selection
 0: From VREFI pin
 1: From internal reference voltage V_{BREF}

When the internal independent reference voltage V_{BREF} is selected as the PGA input, the external reference voltage on the VREFI pin will be automatically switched off. In addition, the internal bandgap reference V_{BREF} 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 reference voltage source is selected, the reference voltage derived from the external VREF pin will automatically be switched off.
- Bit 1~0 **PGAGS1~PGAGS0**: PGA gain selection
 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 source derived from the PGA output V_{VR} . The desired selection is made using the SAVRS1~SAVRS0 bits in the SADC2 register. The internal reference voltage is amplified through 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. The internal Bandgap reference circuit should first be enabled before the V_{BGREF} is selected to be used. A specific start-up time is necessary for the Bandgap circuit to become stable and accurate.

As the VREFI and VREF pins 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 input from the VREFI or VREF pin will automatically be switched off by hardware.

Note that the analog input values must not be allowed to exceed the value of the selected reference voltage.

| SAVRS[1:0] | Reference Source | Description |
|------------|------------------|---|
| 00 | AV _{DD} | Internal A/D converter power supply |
| 01 | VREF Pin | External A/D converter reference pin |
| 10 or 11 | V _{VR} | 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 setup as A/D converter analog channel inputs or whether they have other functions. If the corresponding pin is setup to be an A/D converter analog channel input, the original pin function will be disabled. In this way, pins can be changed under program control to change their function between A/D inputs and other functions. All pull-high resistors, which are setup through register programming, will be automatically disconnected if the pins are setup as A/D inputs. Note that it is not necessary to first setup the A/D pin as an input in the port control register to enable the A/D input as when the relevant A/D input function selection bits enable an A/D input, the status of the port control register will be overridden.

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 SAINS3~SAINS0 bits in the SADC1 register are used to determine that the analog signal to be converted comes from the external channel input or internal analog signal. The SACS3~SACS0 bits in the SADC0 register are used to determine which external channel input is selected to be converted. If the SAINS3~SAINS0 bits are set to “0000”, “0100”, or “1100~1111”, the external channel input will be selected to be converted and the SACS3~SACS0 bits can determine which external channel is selected.

When the SAINS3~SAINS0 bits is set to the value of “0001~0011”, “0101~1011”, the internal analog signal will be selected. If the internal analog signal is selected to be converted, the external channel signal input will automatically be switched off regardless of the SACS3~SACS0 bit values. 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, 1100~1111 | 0000~0111 | AN0~AN11 | External channel analog input ANn |
| | 1000~1111 | — | Floating |
| 0001 | xxxx | AV _{DD} | Internal A/D converter power supply voltage AV _{DD} |
| 0010 | xxxx | AV _{DD} /2 | Internal A/D converter power supply voltage AV _{DD} /2 |
| 0011 | xxxx | AV _{DD} /4 | Internal A/D converter power supply voltage AV _{DD} /4 |
| 0101 | xxxx | V _{VR} | Internal A/D converter PGA output V _{VR} |
| 0110 | xxxx | V _{VR} /2 | Internal A/D converter PGA output V _{VR} /2 |
| 0111 | xxxx | V _{VR} /4 | Internal A/D converter PGA output V _{VR} /4 |
| 1000 | xxxx | OCPAO | OCP operational amplifier output |
| 1001 | xxxx | DACBUF | OCP DAC output with buffer |
| 1010~1011 | xxxx | AV _{SS} | Connected to the ground |

“x”: Don't care

A/D Converter Input Signal Selection

A/D Conversion Operation

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

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

The clock source for the A/D converter, which originates from the system clock f_{SYS} , can be chosen to be either f_{SYS} or a subdivided version of f_{SYS} . The division ratio value is determined by the SACKS2~SACKS0 bits in the SADC1 register. Although the A/D clock source is determined by the system clock f_{SYS} and by bits SACKS2~SACKS0, there are some limitations on the maximum A/D clock source speed that can be selected. As the recommended range of permissible A/D clock period, t_{ADCK} , is from 0.5 μ s to 10 μ 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 clock periods that are less than the minimum A/D clock period or greater than the maximum A/D clock period which may result in inaccurate A/D conversion values. Refer to the following table for examples, where values marked with an asterisk * special care must be taken.

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

A/D Clock Period Examples

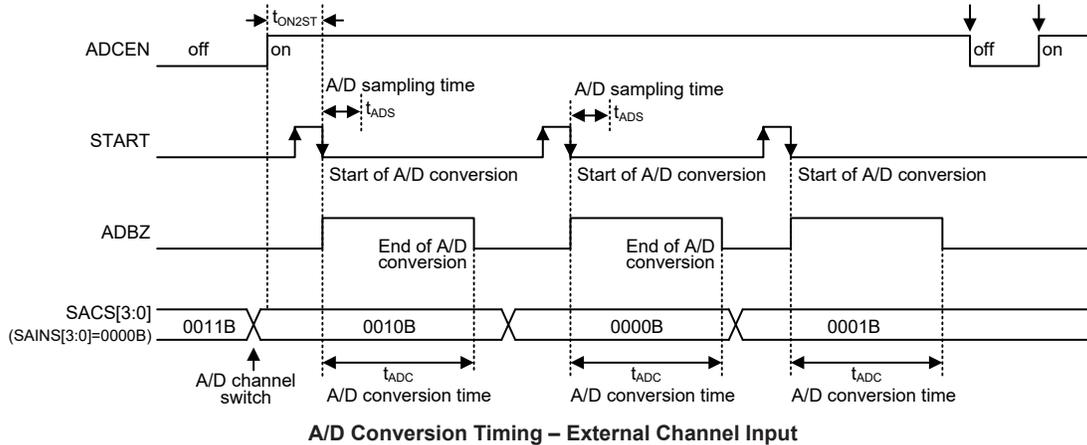
Controlling the power on/off function of the A/D converter circuitry is implemented using the ADCEN bit in the SADC0 register. This bit must be set high to power on the A/D converter. When the ADCEN bit is set high to power on the A/D converter internal circuitry, a certain delay as indicated in the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

Conversion Rate and Timing Diagram

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

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

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



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 one.
- Step 3
Select which signal is to be connected to the internal A/D converter by correctly configuring the SACS3~SACS0 and SAINS3~SAINS0 bits
Selecting the external channel input to be converted, go to Step 4.
Selecting the internal analog signal to be converted, go to Step 5.
- Step 4
If the SAINS3~SAINS0 bits are “0000”, “0100”, or “1100~1111”, the external channel input can be selected. The desired external channel input is selected by configuring the SACS3~SACS0 bits. When the A/D input signal comes from the external channel input, the corresponding pin should be configured as an A/D input function by selecting the relevant pin-shared function control bits. Then go to Step 6.
- Step 5
If the SAINS3~SAINS0 bits are set to “0001~0011”, “0101~1011”, the relevant internal analog signal will be selected. When the internal analog signal is selected to be converted, the external channel analog input will automatically be disconnected. Then go to Step 6.
- Step 6
Select the A/D converter output data format by configuring the ADRFS bit.

- Step 7
Select the A/D converter reference voltage source by configuring the SAVRS1~SAVRS0 bits.
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 8
If A/D conversion interrupt is used, the interrupt control registers must be correctly configured to ensure the A/D interrupt function is active. The master interrupt control bit, EMI, and the A/D conversion interrupt control bit, ADE, must both be set high in advance.
- Step 9
The A/D conversion procedure can now be 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 complete, the ADBZ flag will go low and then the output data can be read from SADOH and SADOL registers.

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

Programming Considerations

During microcontroller operations where the A/D converter is not being used, the A/D internal circuitry can be switched off to reduce power consumption, by setting bit ADCEN low in the SADC0 register. When this happens, the internal A/D converter circuits will not consume power irrespective of what analog voltage is applied to their input lines. If the A/D converter input lines are used as normal I/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 Transfer Function

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

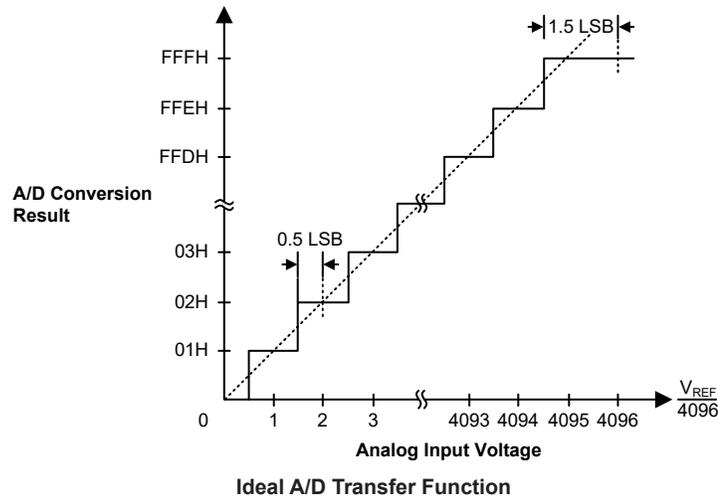
$$1 \text{ LSB} = V_{REF}/4096$$

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

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

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

Note that here the V_{REF} voltage is the actual A/D converter reference voltage determined by the SAVRS1~SAVRS0 bits.



A/D Programming Examples

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

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

```

clr ADE           ; disable ADC interrupt
mov a,03H        ; select fsys/8 as A/D clock and A/D input
mov SADC1,a      ; signal comes from external channel
mov a,00H        ; select AVDD as the A/D reference voltage source
mov SADC2,a
mov a,03H        ; setup PDS0 to configure pin AN0
mov PDS0,a
mov a,20H        ; enable A/D converter and select AN0 as
mov SADC0,a      ; the A/D external channel input
:
start_conversion:
clr START        ; high pulse on start bit to initiate conversion
set START        ; reset A/D
clr START        ; start A/D
:
polling_EOC:
sz ADBZ          ; poll the SADC0 register ADBZ bit to detect end of A/D conversion
jmp polling_EOC  ; continue polling
:
mov a,SADOL      ; read low byte conversion result value
mov SADOL_buffer,a ; save result to user defined register
mov a,SADOH      ; read high byte conversion result value
mov SADOH_buffer,a ; save result to user defined register
:
jmp start_conversion ; start next A/D conversion

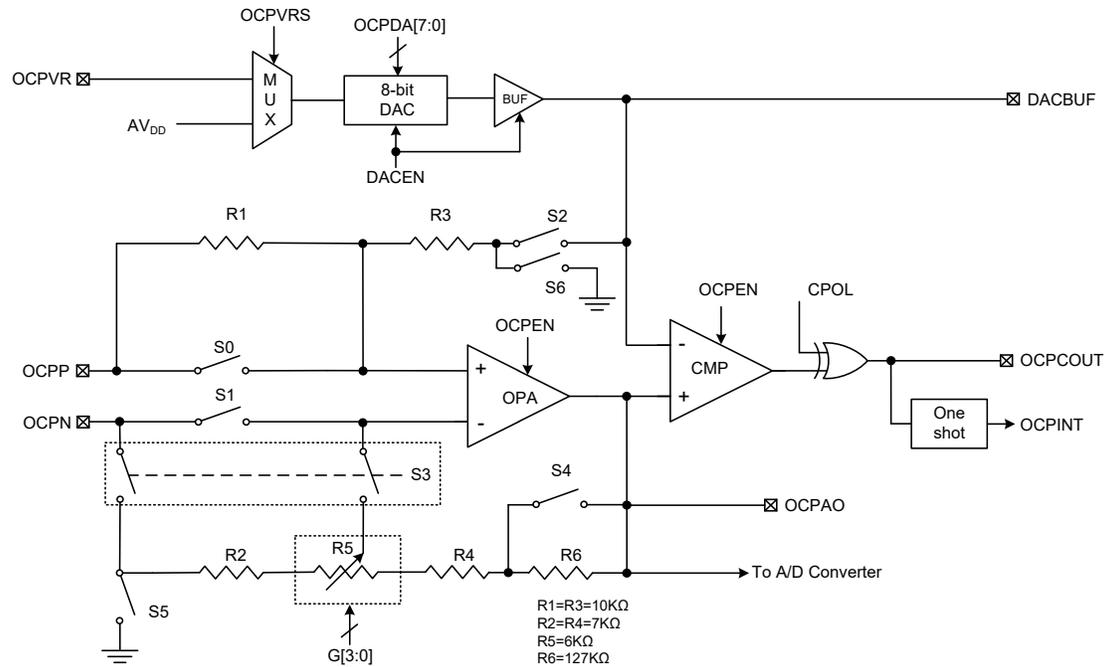
```

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

```
clr ADE ; disable ADC interrupt
mov a,03H ; select fsys/8 as A/D clock and A/D input
mov SADC1,a ; signal comes from external channel
mov a,00H ; select AVDD as the A/D reference voltage source
mov SADC2,a
mov a,03H ; setup PDS0 to configure pin AN0
mov PDS0,a
mov a,20H ; enable A/D converter and select AN0 as
mov SADC0,a ; the A/D external channel input
:
start_conversion:
clr START ; high pulse on START bit to initiate conversion
set START ; reset A/D
clr START ; start A/D
clr ADF ; clear ADC interrupt request flag
set ADE ; enable ADC interrupt
set EMI ; enable global interrupt
:
:
ADC_ISR: ; ADC interrupt service routine
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,SAD0H ; read high byte conversion result value
mov SAD0H_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
```

Over Current Protection – OCP

The device includes an over current protection function which provides a current protection mechanism for applications. The OCP function consists of one fully integrated operational amplifier and one comparator as well as one 8-bit D/A converter with buffer. The operational amplifier can be used for signal amplification according to specific user requirements. The 8-bit D/A Converter can provide an accurate reference voltage to the negative input of Comparator. The Comparator is used to compare two analog voltages and provide an output based on their difference and generate an interrupt to indicate a specific current condition has occurred if the corresponding interrupt control is enabled.



- Note: 1. As the OCP function relevant external pins are pin-shared with general I/O or other functions, ensure that the corresponding pin-shared function registers has been configured properly before using the OCP function.
2. In applications, the DACBUF function should be first selected by properly configuring the corresponding pin-shared function register before the DACEN bit is set high.

Over Current Protection Circuit

The above circuit can be combined to associative functions via switches S0–S6 by controlling the S0–S6 bits. The following table illustrates all implemented functions.

| S0 | S1 | S2 | S3 | S4 | S5 | S6 | Function |
|-----|-----|-----|-----|--------|-----|-----|--|
| ON | ON | OFF | OFF | ON/OFF | OFF | OFF | OCP function for external gain selection |
| ON | OFF | OFF | ON | OFF | OFF | OFF | A complete OCP function |
| OFF | OFF | ON | ON | ON | OFF | OFF | Differential amplifier |
| OFF | OFF | OFF | ON | OFF | OFF | ON | Inverting amplifier |
| OFF | OFF | OFF | ON | OFF | ON | OFF | Non-inverting amplifier |

Implemented Function Summary

Over Current Protection Operation

The OCP circuit is used to prevent the input current from exceeding a specific level. The current on the OCPP or OCPN pin is converted to a voltage and then amplified by the OCP Programmable Gain Amplifier (PGA). The PGA consists of an operational amplifier and three resistors R2, R4 and R5; the PGA gain can be positive or negative determined by input voltage connected to OCPP pin or OCPN pin. The Programmable gain differential amplifier, PGDA consists of an operational amplifier and five resistors R1, R2, R3, R4 and R5; the PGDA gain can be controlled by the R5 resistor value. After the current is converted and amplified to a specific voltage level, it will be compared with a reference voltage provided by an 8-bit D/A converter. The 8-bit D/A converter reference voltage can be supplied by AV_{DD} or the external OCPVR pin, which is selected by the OCPVRS bit in the OCPC0 register.

The OCP operational amplifier and the comparator can be configured to operate in the normal operating mode or input offset calibration mode determined by the OCPOOFM bit in the OCPOCAL register and the OCPCOFM bit in the OCPCCAL register respectively. The OCP operational amplifier output signal can also be directly output on the OCPAO pin, or internally connected to the A/D converter input. A buffer within the D/A converter can enhance output driving capability.

The comparator output bit, OCPCOUT, is obtained to indicate whether an over current condition occurs or not. If the OCPCOUT bit changes state from low to high, made by the specific voltages on OCP comparator inputs and by the corresponding condition of the CPOL bit, a one shot signal will be generated to trigger an OCP interrupt request. It is important to note that, only the rising edge of OCPCOUT bit can trigger an OCP interrupt request, so the CPOL bit must be properly configured according to user's application requirements.

Note that the filter clock is from the system clock f_{sys} . The amplified input voltage also can be read out by the A/D converter through the OCPAO pin. The D/A converter output voltage is controlled by the OCPDA register.

Over Current Protection Registers

Overall operation of the over current protection is controlled using several registers. The OCPDA register is used to provide the reference voltage for the over current protection circuit. The OCPOCAL and OCPCCAL registers are used to cancel out the operational amplifier and comparator input offset. The OCPC0 and OCPC1 registers are control registers which control the OCP function, D/A converter reference voltage selection, PGA gain selection together with the comparator hysteresis function.

| Register Name | Bit | | | | | | | |
|---------------|---------|---------|---------|---------|---------|---------|---------|---------|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| OCPC0 | OCPEN | DACEN | CPOL | OCPVRS | S3 | S2 | S1 | S0 |
| OCPC1 | S6 | G3 | G2 | G1 | G0 | S5 | S4 | OCPCHY |
| OCPDA | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| OCPOCAL | OCPOOFM | OCPORSP | OCPOOF5 | OCPOOF4 | OCPOOF3 | OCPOOF2 | OCPOOF1 | OCPOOF0 |
| OCPCCAL | OCPCOUT | OCPCOFM | OCPCRSR | OCPCOF4 | OCPCOF3 | OCPCOF2 | OCPCOF1 | OCPCOF0 |

Over Current Protection Register List

• **OCPC0 Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------|-------|-------|------|--------|-----|-----|-----|-----|
| Name | OCPEN | DACEN | CPOL | OCPVRS | S3 | S2 | S1 | S0 |
| 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 **OCPEN**: OCP function control

0: Disable

1: Enable

Bit 6 **DACEN**: DAC function control

0: Disable, DAC output floating.

1: Enable

Bit 5 **CPOL**: OCP comparator output polarity control

0: Non-invert

1: Invert

This is the comparator polarity bit. If the bit is zero, the OCPCOUT bit will reflect the non-inverted output condition of the comparator. If the bit is high, the comparator OCPCOUT bit will be inverted.

Bit 4 **OCPVRS**: OCP D/A converter reference voltage selection

0: From AV_{DD}

1: From OCPVR pin

Bit 3 **S3**: Switch S3 control

0: Off

1: On

Bit 2 **S2**: Switch S2 control

0: Off

1: On

Bit 1 **S1**: Switch S1 control

0: Off

1: On

Bit 0 **S0**: Switch S0 control

0: Off

1: On

• **OCPC1 Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------|-----|-----|-----|-----|-----|-----|-----|--------|
| Name | S6 | G3 | G2 | G1 | G0 | S5 | S4 | OCPCHY |
| 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 **S6**: Switch S6 control

0: Off

1: On

Bit 6~3 **G3~G0**: OCP PGA R5 resistor control

For example, when S4 is on:

$$R5 = R5' + R5''$$

$$R4 + R5' = 7k\Omega + 6k\Omega \times G[3:0]/16$$

$$R2 + R5'' = 7k\Omega + 6k\Omega \times (16 - G[3:0])/16$$

Bit 2 **S5**: Switch S5 control

0: Off

1: On

Bit 1 **S4**: Switch S4 control

0: Off

1: On

Bit 0 **OCPCHY**: OCP comparator hysteresis function control
 0: Disable
 1: Enable

• **OCPDA 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**: OCP D/A converter output voltage control bits
 $DAC\ Output = (D/A\ converter\ reference\ voltage / 256) \times OCPDA[7:0]$

• **OCPOCAL Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------|---------|---------|---------|---------|---------|---------|---------|---------|
| Name | OCPOOFM | OCPORSP | OCPOOF5 | OCPOOF4 | OCPOOF3 | OCPOOF2 | OCPOOF1 | OCPOOF0 |
| R/W | R/W | R/W3 | R/W | R/W | R/W | R/W | R/W | R/W |
| POR | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |

Bit 7 **OCPOOFM**: OCP Operational Amplifier Input Offset Calibration Mode enable control
 0: Input Offset Calibration Mode disable
 1: Input Offset Calibration Mode enable

Bit 6 **OCPORSP**: OCP Operational Amplifier Input Offset Calibration Reference selection
 0: Select negative input as the reference input
 1: Select positive input as the reference input

Bit 5~0 **OCPOOF5~OCPOOF0**: OCP Operational Amplifier Input Offset Calibration value
 This 6-bit field is used to perform the operational amplifier input offset Calibration operation and the value for the OCP operational amplifier input offset Calibration can be restored into this bit field. More detailed information is described in the “Operational Amplifier Input Offset Calibration” section.

• **OCPCCAL Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------|---------|---------|-----------|---------|---------|---------|---------|---------|
| Name | OCPCOUT | OCPCOFM | OCPCRSRSP | OCPCOF4 | OCPCOF3 | OCPCOF2 | OCPCOF1 | OCPCOF0 |
| R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| POR | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |

Bit 7 **OCPCOUT**: OCP comparator output, positive logic (read only)

Bit 6 **OCPCOFM**: OCP Comparator Input Offset Calibration Mode enable control
 0: Input Offset Calibration Mode disable
 1: Input Offset Calibration Mode enable

Bit 5 **OCPCRSRSP**: OCP Comparator Input Offset Calibration Reference selection
 0: Select negative input as the reference input
 1: Select positive input as the reference input

Bit 4~0 **OCPCOF4~OCPCOF0**: OCP Comparator Input Offset Calibration value
 This 5-bit field is used to perform the comparator input offset calibration operation and the value for the OCP comparator input offset calibration can be restored into this bit field. More detailed information is described in the “Comparator Input Offset Calibration” section.

Input Voltage Range

The PGA output for the positive or negative input voltage is calculated based on different formulas and described by the following.

- If the OCPP pin is connected to VSS and the input voltage comes from the OCPN pin, and S0=S3=S4=ON, S1=S2=OFF, R5'=6kΩ×G[3:0]/16, R5''=6kΩ×(16-G[3:0])/16, 0>V_{IN}>-0.1V. The PGA output is obtained using the formula below:

$$V_{OUT} = -(R4 + R5') / (R2 + R5'') \times V_{IN}$$

- If the OCPN pin is connected to VSS and the input voltage comes from the OCPP pin, and S0=S3=S4=ON, S1=S2=OFF, R5'=6kΩ×G[3:0]/16, R5''=6kΩ-6kΩ×G[3:0]/16, V_{IN}>0. The PGA output is obtained using the formula below:

$$V_{OUT} = (1 + (R4 + R5') / (R2 + R5'')) \times V_{IN}$$

- If R5'=6kΩ×G[3:0]/16, R5''=6kΩ×(16-G[3:0])/16, S0=S1=OFF, S2=S3=S4=ON, V_{INP}>0.2V, V_{INN}>0.2V.

$$V_{OUT} = ((V_{INP} - V_{DAC}) \times \frac{R3}{R1 + R3} + V_{DAC}) - V_{INN} \times (1 + \frac{R4 + R5'}{R2 + R5''}) + V_{INN}$$

Input Offset Calibration

Operational Amplifier Input Offset Calibration

Step 1. Set OCPEN=1, OCPOOFM=1, OCPCOFM=0 and OCPORSP=1, the OCP will operate in the operational amplifier input offset calibration mode.

Step 2. Set OCPOOF[5:0]=000000 and then read the OCPCOUT bit.

Step 3. Increase the OCPOOF[5:0] value by 1 and then read the OCPCOUT bit.

If the OCPCOUT bit state has not changed, then repeat Step 3 until the OCPCOUT bit state has changed.

If the OCPCOUT bit state has changed, record the OCPOOF value as V_{OOS1} and then go to Step 4.

Step 4. Set OCPOOF[5:0]=111111 and read the OCPCOUT bit.

Step 5. Decrease the OCPOOF[5:0] value by 1 and then read the OCPCOUT bit.

If the OCPCOUT bit state has not changed, then repeat Step 5 until the OCPCOUT bit state has changed.

If the OCPCOUT bit state has changed, record the OCPOOF value as V_{OOS2} and then go to Step 6.

Step 6. Restore the operational amplifier input offset calibration value V_{OOS} into the OCPOOF[5:0] bit field. The offset Calibration procedure is now finished.

$$\text{Where } V_{OOS} = (V_{OOS1} + V_{OOS2}) / 2$$

Note: In this mode, the operational amplifier outputs to OCPCOUT bypassing the comparator.

Comparator Input Offset Calibration

Step 1. Set OCPEN=1, OCPCOFM=1 and OCPOOFM=0, DACEN=1, OCPDA=Target voltage, the OCP will now operate in the comparator input offset calibration mode.

Step 2. Set OCPCOF[4:0]=00000 and read the OCPCOUT bit.

Step 3. Increase the OCPCOF[4:0] value by 1 and then read the OCPCOUT bit.

If the OCPCOUT bit state has not changed, then repeat Step 3 until the OCPCOUT bit state has changed.

If the OCPCOUT bit state has changed, record the OCPCOF value as V_{COS1} and then go to Step 4.

Step 4. Set OCPCOF[4:0]=11111 and then read the OCPCOUT bit.

Step 5. Decrease the OCPCOF[4:0] value by 1 and then read the OCPCOUT bit.

If the OCPCOUT bit state has not changed, then repeat Step 5 until the OCPCOUT bit state has changed.

If the OCPCOUT bit state has changed, record the OCPCOF value as V_{COS2} and then go to Step 6.

Step 6. Restore the comparator input offset calibration value V_{COS} into the OCPCOF[4:0] bit field.

The offset Calibration procedure is now finished.

$$\text{Where } V_{COS} = (V_{COS1} + V_{COS2}) / 2$$

Touch Key Function

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

Touch Key Structure

The touch keys are pin-shared with the I/O pins, with the desired function chosen via the corresponding selection register bits. Keys are organised into several groups, with each group known as a module and having a module number, M0 to M4. Each module is a fully independent set of four Touch Keys and each Touch Key has its own oscillator. Each module contains its own control logic circuits and register set. Examination of the register names will reveal the module number it is referring to.

| Total Key Number | Touch Key Module | Shared I/O Pin |
|------------------|------------------|----------------|
| 20 | M0 | KEY1~KEY4 |
| | M1 | KEY5~KEY8 |
| | M2 | KEY9~KEY12 |
| | M3 | KEY13~KEY16 |
| | M4 | KEY17~KEY20 |

Touch Key Structure

Touch Key Register Definition

Each touch key module, which contains four touch key functions, has its own suite registers. The following table shows the register set for the touch key module. The Mn within the register name refers to the Touch Key module number. The device has five Touch Key Modules.

| Register Name | Description |
|---------------|---|
| TKTMR | Touch key time slot 8-bit counter preload register |
| TKC0 | Touch key function control register 0 |
| TKC1 | Touch key function control register 1 |
| TK16DL | Touch key function 16-bit counter low byte |
| TK16DH | Touch key function 16-bit counter high byte |
| TKMn16DL | Touch key module n 16-bit C/F counter low byte |
| TKMn16DH | Touch key module n 16-bit C/F counter high byte |
| TKMnROL | Touch key module n reference oscillator capacitor selection low byte |
| TKMnROH | Touch key module n reference oscillator capacitor selection high byte |
| TKMnC0 | Touch key module n control register 0 |
| TKMnC1 | Touch key module n control register 1 |

| Register Name | Description |
|---------------|---------------------------------------|
| TKMnC2 | Touch key module n control register 2 |

Touch Key Function Register Definition (n=0~4)

| Register Name | Bit | | | | | | | |
|---------------|--------|--------|--------|---------|--------|--------|--------|--------|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| TKTMR | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| TKC0 | TKRAMC | TKRCOV | TKST | TKCFOV | TK16OV | — | TKMOD | TKBUSY |
| TKC1 | D7 | D6 | D5 | TSCS | TK16S1 | TK16S0 | TKFS1 | TKFS0 |
| TK16DL | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| TK16DH | D15 | D14 | D13 | D12 | D11 | D10 | D9 | D8 |
| TKMn16DL | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| TKMn16DH | D15 | D14 | D13 | D12 | D11 | D10 | D9 | D8 |
| TKMnROL | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| TKMnROH | — | — | — | — | — | — | D9 | D8 |
| TKMnC0 | — | — | MnDFEN | MnFILEN | MnSOFC | MnSOF2 | MnSOF1 | MnSOF0 |
| TKMnC1 | MnTSS | — | MnROEN | MnKOEN | MnK4EN | MnK3EN | MnK2EN | MnK1EN |
| TKMnC2 | MnSK31 | MnSK30 | MnSK21 | MnSK20 | MnSK11 | MnSK10 | MnSK01 | MnSK00 |

Touch Key Function Register List (n=0~4)

• **TKTMR Register**

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

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

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

Time slot counter overflow time=(256 - TKTMR[7:0])×32t_{TSC}, where t_{TSC} is the time slot counter clock period.

• **TKC0 Register**

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

Bit 7 **TKRAMC:** Touch key data memory access control

- 0: Accessed by MCU
- 1: Accessed by touch key module

This bit determines that the touch key data memory is used by the MCU or the touch key module. However, the touch key module will have the priority to access the touch key data memory when the touch key module operates in the auto scan mode, i.e., the TKST bit state is changed from 0 to 1 when the TKMOD bit is cleared to 0. After the touch key auto scan operation is completed, i.e., the TKBUSY bit state is changed from 1 to 0, the touch key data memory access will be controlled by the TKRAMC bit. Therefore, it is recommended to set the TKRAMC bit to 1 when the touch key module operates in the auto scan mode. Otherwise, the contents of the touch key data memory may be modified as this data memory space is configured by the touch key module

followed by the MCU access. When the TKRAMC bit is set to 1, MCU reading the touch key data memory will obtain uncertain values as the touch key data memory is controlled by the touch key circuit.

Bit 6 **TKRCOV**: Touch key time slot counter overflow flag
0: No overflow occurs
1: Overflow occurs

This bit can be accessed by application program. When this bit is set by touch key time slot counter overflow, the corresponding touch key interrupt request flag will be set. However, if this bit is set by application program, the touch key interrupt request flag will not be affected. Therefore, this bit can not be set by application program but must be cleared to 0 by application program.

In the auto scan mode, if module 0 or all module time slot counter, selected by TSCS bit, overflows but the touch key auto scan operation is not completed, the TKRCOV bit will not be set. At this time, all module touch key module 16-bit C/F counter, touch key function 16-bit counter and 5-bit time slot unit period counter will be automatically cleared but the 8-bit time slot counter will be reloaded from the 8-bit time slot counter preload register. When the touch key auto scan operation is completed, the TKRCOV bit and the Touch Key Module Interrupt request flag, TKMF, will be set and all module keys and reference oscillators will automatically stop. All touch key module 16-bit C/F counter, touch key function 16-bit counter, 5-bit time slot unit period counter and 8-bit time slot counter will be automatically switched off.

In the manual scan mode, if module 0 or all module time slot counter, selected by TSCS bit, overflows, the TKRCOV bit and the Touch Key Module Interrupt request flag, TKMF, will be set and all module keys and reference oscillators will automatically stop. All touch key module 16-bit C/F counter, touch key function 16-bit counter, 5-bit time slot unit period counter and 8-bit time slot counter will be automatically switched off.

Bit 5 **TKST**: Touch key detection start control
0: Stopped or no operation
0→1: Start detection

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

Bit 4 **TKCFOV**: Touch key module 16-bit C/F counter overflow flag
0: No overflow occurs
1: Overflow occurs

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

Bit 3 **TK16OV**: Touch key function 16-bit counter overflow flag
0: No overflow occurs
1: Overflow occurs

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

Bit 2 Unimplemented, read as “0”

Bit 1 **TKMOD**: Touch key scan mode selection
0: Auto scan mode
1: Manual mode

In the manual scan mode the reference oscillator capacitor value should be properly configured before the scan operation begins and the touch key module 16-bit C/F counter value should be read by application program after the scan operation finishes.

In the auto scan mode the data movement which is described above is implemented by

hardware. The individual reference oscillator capacitor value and 16-bit C/F counter content for all scanned keys will be read from and written into a dedicated Touch Key Data Memory area. In the auto scan mode the keys to be scanned can be arranged in a specific sequence which is determined by the MnSK3[1:0]~MnSK0[1:0] bits in the TKMnC2 register. The scan operation will not be stopped until all arranged keys are scanned.

- Bit 0 **TKBUSY**: Touch key scan operation busy flag
 0: Not busy – No scan operation is executed or scan operation is completed
 1: Busy – Scan operation is executing

This bit indicates whether the touch key scan operation is executing or not. It is set to 1 when the TKST bit is set high to start the scan operation.

In the manual scan mode this bit is cleared to 0 automatically when module 0 or all module time slot counter, selected by TSCS bit, overflows. In the auto scan mode this bit is cleared to 0 automatically when the touch key scan operation is completed.

• **TKC1 Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------|-----|-----|-----|------|--------|--------|-------|-------|
| Name | D7 | D6 | D5 | TSCS | TK16S1 | TK16S0 | TKFS1 | TKFS0 |
| 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 | 1 | 1 |

- Bit 7~5 **D7~D5**: Data bits for test only
 These bits are used for test purpose only and must be kept as “000” for normal operations.

- Bit 4 **TSCS**: Touch Key modules time slot counter selection
 0: Each module use its own time slot counter
 1: All touch key modules use module 0 time slot counter

- Bit 3~2 **TK16S1~TK16S0**: Touch key function 16-bit counter clock source selection
 00: f_{sys}
 01: $f_{sys}/2$
 10: $f_{sys}/4$
 11: $f_{sys}/8$

- Bit 1~0 **TKFS1~TKFS0**: Touch key oscillator and Reference oscillator frequency selection
 00: 1MHz
 01: 2.5MHz
 10: 7.5MHz
 11: 11MHz

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

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

This register pair is used to store the touch key function 16-bit counter value. This 16-bit counter can be used to calibrate the reference or key oscillator frequency. When the touch key time slot counter overflows in the manual scan mode, this 16-bit counter will be stopped and the counter content will be unchanged. However, this 16-bit counter content will be cleared to zero at the end of the time slot 0, slot 1 and slot 2 but kept unchanged at the end of the time slot 3 in the auto scan mode. This register pair will be cleared to zero when the TKST bit is cleared to zero.

• **TKMn16DH/TKMn16DL – Touch Key Module n 16-bit C/F Counter Register Pair**

| Register | TKMn16DH | | | | | | | | TKMn16DL | | | | | | | |
|----------|----------|-----|-----|-----|-----|-----|----|----|----------|----|----|----|----|----|----|----|
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Name | D15 | D14 | D13 | D12 | D11 | D10 | D9 | D8 | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| R/W | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R | R |
| POR | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

This register pair is used to store the touch key module n 16-bit C/F counter value. This 16-bit C/F counter will be stopped and the counter content will be kept unchanged when the touch key time slot counter overflows in the manual scan mode. However, this 16-bit C/F counter content will be cleared to zero at the end of the time slot 0, slot 1 and slot 2 after it is written to the touch key data memory but kept unchanged at the end of the time slot 3 in the auto scan mode. This register pair will be cleared to zero when the TKST bit is cleared to zero.

• **TKMnROH/TKMnROL – Touch Key Module n Reference Oscillator Capacitor Selection Register Pair**

| Register | TKMnROH | | | | | | | | TKMnROL | | | | | | | |
|----------|---------|---|---|---|---|---|-----|-----|---------|-----|-----|-----|-----|-----|-----|-----|
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Name | — | — | — | — | — | — | D9 | D8 | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| R/W | — | — | — | — | — | — | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| POR | — | — | — | — | — | — | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

This register pair is used to setup the touch key module n reference oscillator capacitor value. This register pair will be loaded with the corresponding next time slot capacitor value from the dedicated touch key data memory at the end of the current time slot when the auto scan mode is selected.

The reference oscillator internal capacitor value=(TKMnRO[9:0]×50pF)/1024

• **TKMnC0 Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------|---|---|--------|---------|--------|--------|--------|--------|
| Name | — | — | MnDFEN | MnFILEN | MnSOFC | MnSOF2 | MnSOF1 | MnSOF0 |
| 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 **MnDFEN**: Touch key module n multi-frequency control
 0: Disable
 1: Enable

This bit is used to control the touch key oscillator frequency doubling function. When this bit is set to 1, the key oscillator frequency will be doubled.

Bit 4 **MnFILEN**: Touch key module n filter function control
 0: Disable
 1: Enable

Bit 3 **MnSOFC**: Touch key module n C/F oscillator frequency hopping function control selection
 0: Controlled by the MnSOF2~MnSOF0 bits
 1: Controlled by hardware circuit

This bit is used to select the touch key oscillator frequency hopping function control method. When this bit is set to 1, the key oscillator frequency hopping function is controlled by the hardware circuit regardless of the MnSOF2~MnSOF0 bits value.

Bit 2~0 **MnSOF2~MnSOF0**: Touch key module n Reference and Key oscillators hopping frequency select
 000: 1.020MHz
 001: 1.040MHz
 010: 1.059MHz
 011: 1.074MHz
 100: 1.085MHz
 101: 1.099MHz
 110: 1.111MHz
 111: 1.125MHz

These bits are used to select the touch key oscillator frequency for the hopping function. Note that these bits are only available when the MnSOFC bit is cleared to 0.

The frequencies mentioned here are only for the condition where the key and reference oscillator frequency is selected to be 1MHz, these values will be changed when the external or internal capacitor has different values. Users can adjust the key and reference oscillator frequency in scale when any other frequency is selected.

• **TKMnC1 Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------|-------|---|--------|--------|--------|--------|--------|--------|
| Name | MnTSS | — | MnROEN | MnKOEN | MnK4EN | MnK3EN | MnK2EN | MnK1EN |
| 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 **MnTSS**: Touch key module n time slot counter clock source selection
 0: Touch key module n reference oscillator
 1: $f_{sys}/4$

Bit 6 Unimplemented, read as “0”

Bit 5 **MnROEN**: Touch key module n Reference oscillator enable control
 0: Disable
 1: Enable

In the manual scan mode, this bit is used to enable/disable the touch key module n reference oscillator. The module n reference oscillator should first be enabled by setting the MnROEN bit high before setting the TKST bit from low to high if the reference oscillator is selected to be used and will be automatically disabled when the TKBUSY bit is changed from high to low.

In the auto scan mode, this bit is controlled by hardware automatically. For the module 0, when the TKST bit changes from low to high, the M0ROEN bit will be set high automatically. For the other module n ($n \neq 0$), if the condition that $MnK4EN \sim MnK1EN \neq 0000B$, $MnTSS=0$ and $TSCS=0$ is satisfied, the MnROEN bit will be set high automatically when the TKST bit changes from low to high, while in other conditions the MnROEN bit will not be affected by the TKST bit setting. When the TKBUSY bit is changed from high to low, the MnROEN bit will automatically be cleared to zero to disable the reference oscillator.

Bit 4 **MnKOEN**: Touch key module n Key oscillator enable control
 0: Disable
 1: Enable

In the manual scan mode, this bit is used to enable/disable the touch key module n key oscillator. The touch key module n key oscillator should first be enabled by setting the MnKOEN bit high before setting the TKST bit from low to high if the relevant key is enabled to be scanned and will be disabled automatically when the TKBUSY bit is changed from high to low.

In the auto scan mode, the MnKOEN bit will be set high automatically to enable the key oscillator when the TKST bit changes from low to high. When the TKBUSY bit is changed from high to low, the MnKOEN bit will automatically be cleared to zero to disable the key oscillator.

Bit 3 **MnK4EN**: Touch key module n Key4 enable control

| MnK4EN | Touch Key Module n – Mn | | | | |
|------------|-------------------------|------|-------|-------|-------|
| | M0 | M1 | M2 | M3 | M4 |
| 0: Disable | I/O or other function | | | | |
| 1: Enable | KEY4 | KEY8 | KEY12 | KEY16 | KEY20 |

Bit 2 **MnK3EN**: Touch key module n Key3 enable control

| MnK3EN | Touch Key Module n – Mn | | | | |
|------------|-------------------------|------|-------|-------|-------|
| | M0 | M1 | M2 | M3 | M4 |
| 0: Disable | I/O or other function | | | | |
| 1: Enable | KEY3 | KEY7 | KEY11 | KEY15 | KEY19 |

Bit 1 **MnK2EN**: Touch key module n Key2 enable control

| MnK2EN | Touch Key Module n – Mn | | | | |
|------------|-------------------------|------|-------|-------|-------|
| | M0 | M1 | M2 | M3 | M4 |
| 0: Disable | I/O or other function | | | | |
| 1: Enable | KEY2 | KEY6 | KEY10 | KEY14 | KEY18 |

Bit 0 **MnK1EN**: Touch key module n Key1 enable control

| MnK1EN | Touch Key Module n – Mn | | | | |
|------------|-------------------------|------|------|-------|-------|
| | M0 | M1 | M2 | M3 | M4 |
| 0: Disable | I/O or other functions | | | | |
| 1: Enable | KEY1 | KEY5 | KEY9 | KEY13 | KEY17 |

• **TKMnC2 Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------|--------|--------|--------|--------|--------|--------|--------|--------|
| Name | MnSK31 | MnSK30 | MnSK21 | MnSK20 | MnSK11 | MnSK10 | MnSK01 | MnSK00 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| POR | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 |

Bit 7~6 **MnSK31~MnSK30**: Touch key module n time slot 3 key scan selection

- 00: Key1
- 01: Key2
- 10: Key3
- 11: Key4

These bits are used to select the desired scan key in time slot 3 and only available in the auto scan mode.

Bit 5~4 **MnSK21~MnSK20**: Touch key module n time slot 2 key scan selection

- 00: Key1
- 01: Key2
- 10: Key3
- 11: Key4

These bits are used to select the desired scan key in time slot 2 and only available in the auto scan mode.

Bit 3~2 **MnSK11~MnSK10**: Touch key module n time slot 1 key scan selection

- 00: Key1
- 01: Key2
- 10: Key3
- 11: Key4

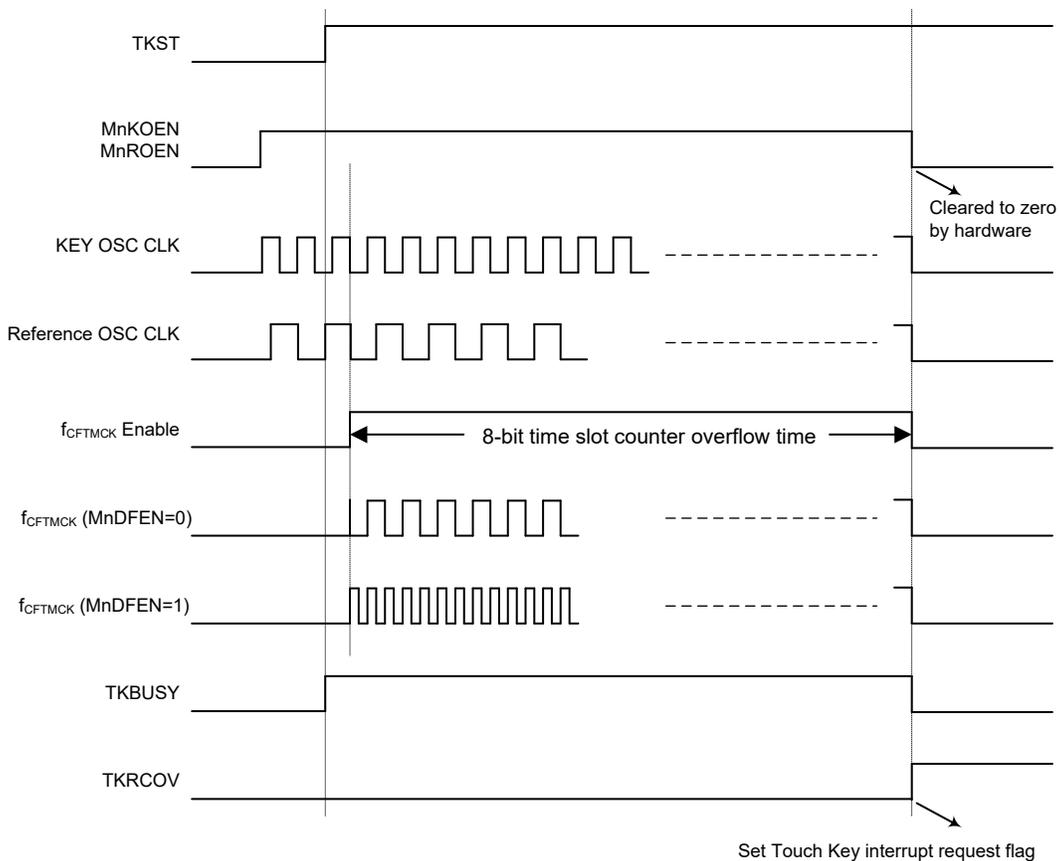
These bits are used to select the desired scan key in time slot 1 and only available in the auto scan mode.

Bit 1~0 **MnSK01~MnSK00**: Touch key module n time slot 0 key scan selection
 00: Key1
 01: Key2
 10: Key3
 11: Key4

These bits are used to select the desired scan key in time slot 0 in the auto scan mode or used as the multiplexer for scan key selection in the manual mode.

Touch Key Operation

When a finger touches or is in proximity to a touch pad, the capacitance of the pad will increase. By using this capacitance variation to change slightly the frequency of the internal sense oscillator, touch actions can be sensed by measuring these frequency changes. Using an internal programmable divider the reference clock is used to generate a fixed time period. By counting a number of generated clock cycles from the sense oscillator during this fixed time period touch key actions can be determined.



Touch Key Manual Scan Mode Timing Diagram

Each touch key module contains four touch key inputs, which are shared with logical I/O pins, and the desired function is selected using the corresponding pin-shared control register bits. The touch key has its own independent sense oscillator. There are therefore four sense oscillators within a touch key module.

During this reference clock fixed interval, the number of clock cycles generated by the sense oscillator is measured, and it is this value that is used to determine if a touch action has been made or not. At the end of the fixed reference clock time interval a Touch Key interrupt signal will be generated in the manual scan mode.

Using the TSCS bit in the TKC1 register can select the module 0 time slot counter as the time slot counter for all modules. All modules use the same start signal, TKST, in the TKC0 register. The touch key module 16-bit C/F counter, touch key function 16-bit counter, 5-bit time slot unit period counter in the module will be automatically cleared when the TKST bit is cleared to zero, but the 8-bit programmable time slot counter will not be cleared. The overflow time is setup by users. When the TKST bit changes from low to high, the 16-bit C/F counter, 16-bit counter, 5-bit time slot unit period counter and 8-bit time slot timer counter will be automatically switched on.

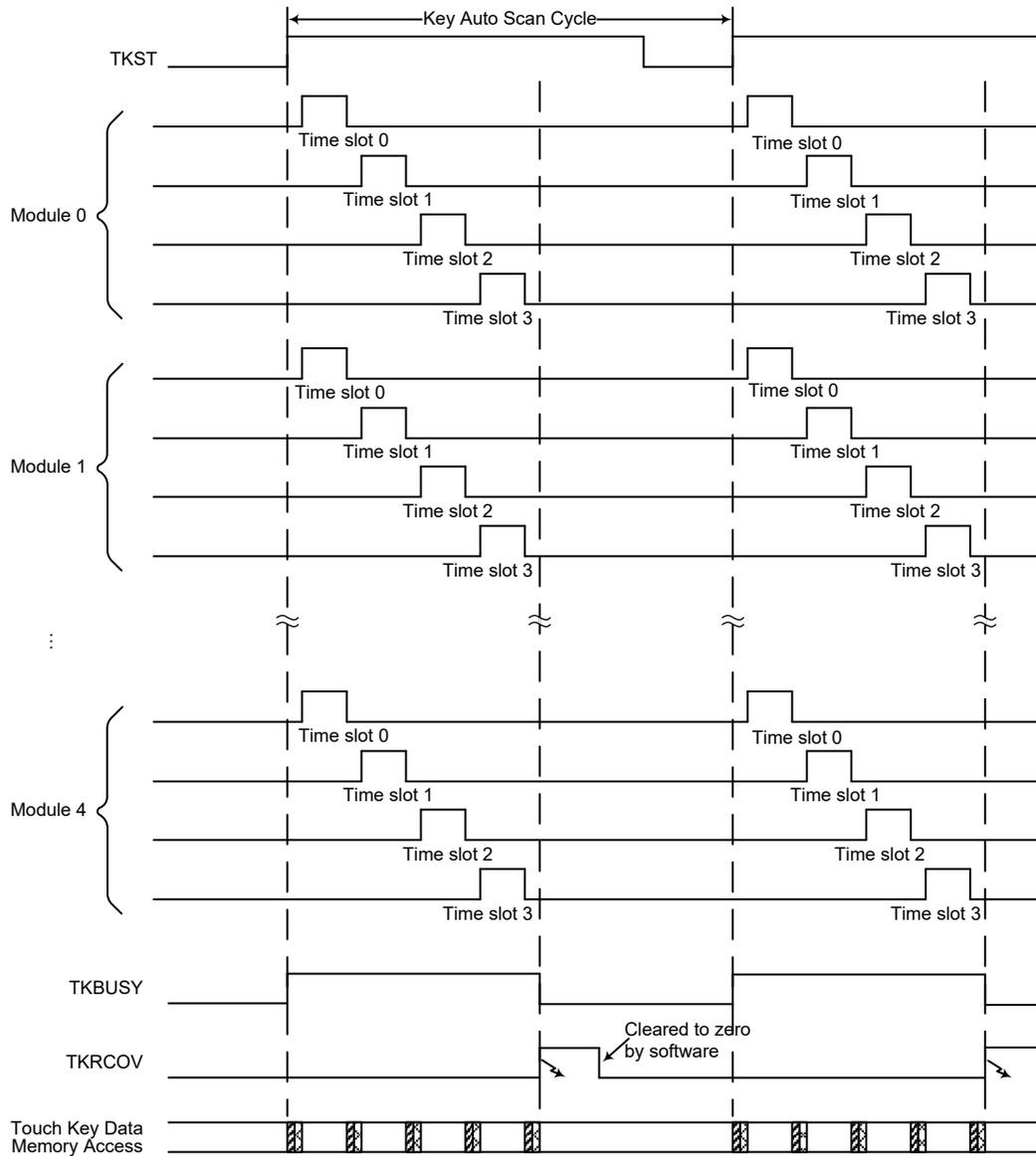
The key oscillator and reference oscillator in all modules will be automatically stopped and the 16-bit C/F counter, 16-bit counter, 5-bit time slot unit period counter and 8-bit time slot timer counter will be automatically switched off when the time slot counter overflows. The clock source for the time slot counter is sourced from the module n reference oscillator or $f_{SYS}/4$ which is selected using the MnTSS bit in the TKMnC1 register. The reference oscillator and key oscillator will be enabled by setting the MnROEN bit and MnKOEN bits in the TKMnC1 register.

When the time slot counter in all the touch key modules or in the touch key module 0 overflows, an actual Touch Key interrupt will take place. The touch keys mentioned here are the keys which are enabled.

Each touch key module consists of four touch keys, KEY1~KEY4 are contained in module 0, KEY5~KEY8 are contained in module 1, KEY9~KEY12 are contained in module 2, KEY13~KEY16 are contained in module 3, KEY17~KEY20 are contained in module 4. Each touch key module has an identical structure.

Auto Scan Mode

There are two scan modes contained for the touch key function. The auto scan mode and the manual scan mode are selected using the TKMOD bit in the TKC0 register. The auto scan mode can minimize the load of the application program and improve the touch key scan operation performance. When the TKMOD bit is set to 0, the auto scan mode is selected to scan the module keys in a specific sequence determined by the MnSK3[1:0]~MnSK0[1:0] bits in the TKMnC2 register.



-  : Set Touch Key interrupt request flag
 -  : Read 2N bytes from Touch Key Data Memory to TKMnROH/TKMnROL registers
 -  : Write 2N bytes from TKMn16DH/TKMn16DL registers to Touch Key Data Memory
- N = Touch Key Module Number; n = Module Serial Number

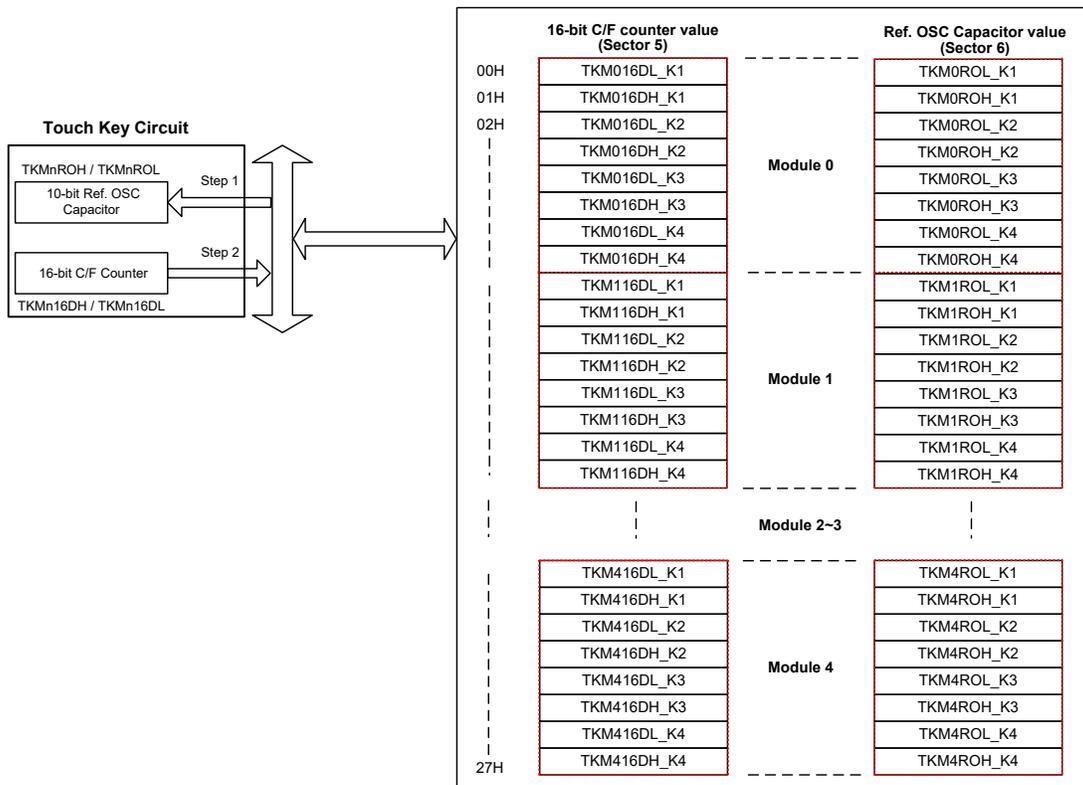
Touch Key Auto Scan Mode Timing Diagram

In the auto scan mode the module n key oscillator and reference oscillator which are required to be used will be enabled by hardware automatically when the TKST bit is set from low to high and disabled automatically when the TKBUSY bit changes from high to low. When the TKST bit is set from low to high in the auto scan mode, the internal capacitor value of the reference oscillator for the selected key to be scanned in the time slot 0 will first be read from a specific location of the dedicated touch key data memory and loaded into the corresponding TKMnROH/TKMnROL registers. Then the 16-bit C/F counter value will be written into the corresponding location of the

last time slot 3 scanned key in the touch key data memory. After this, the selected key will start to be scanned in time slot 0. At the end of the time slot 0 key scan operation, the reference oscillator internal capacitor value for the next selected key will be read from the touch key data memory and loaded into the next TKMnROH/TKMnROL registers. Then the 16-bit C/F counter value of the current scanned key will be written into the corresponding touch key data memory. The whole auto scan operation will sequentially be carried out in the above specific way from time slot 0 to time slot 3. At the end of the time slot 3 key scan operation, the reference oscillator internal capacitor value for the time slot 0 selected key will again be read from the touch key data memory and loaded into the corresponding TKMnROH/TKMnROL registers. Then the 16-bit C/F counter value will be written into the relevant location of the time slot 3 scanned key in the touch key data memory. After all selected keys are scanned, the TKRCOV bit will be set high and the TKBUSY bit will be cleared to zero as well as an auto scan mode operation is completed.

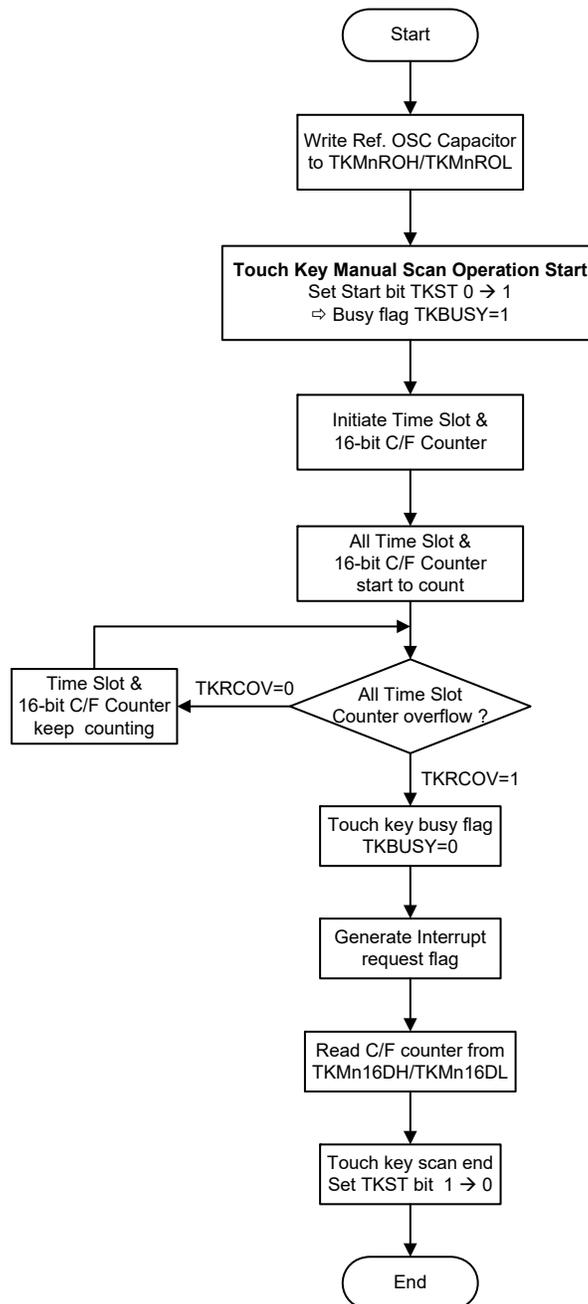
Touch Key Module Data Memory

The device provides two dedicated Data Memory area for the touch key auto scan mode. The addresses 00H~27H of both Sector 5 and Sector 6 are used by the touch key function, as shown in the following figure. One area is used to store the 16-bit C/F counter values of the touch key module 0~4 and located in Data Memory Sector 5. The other area is used to store the reference oscillator internal capacitor values of the touch key modules and located in Data Memory Sector 6.

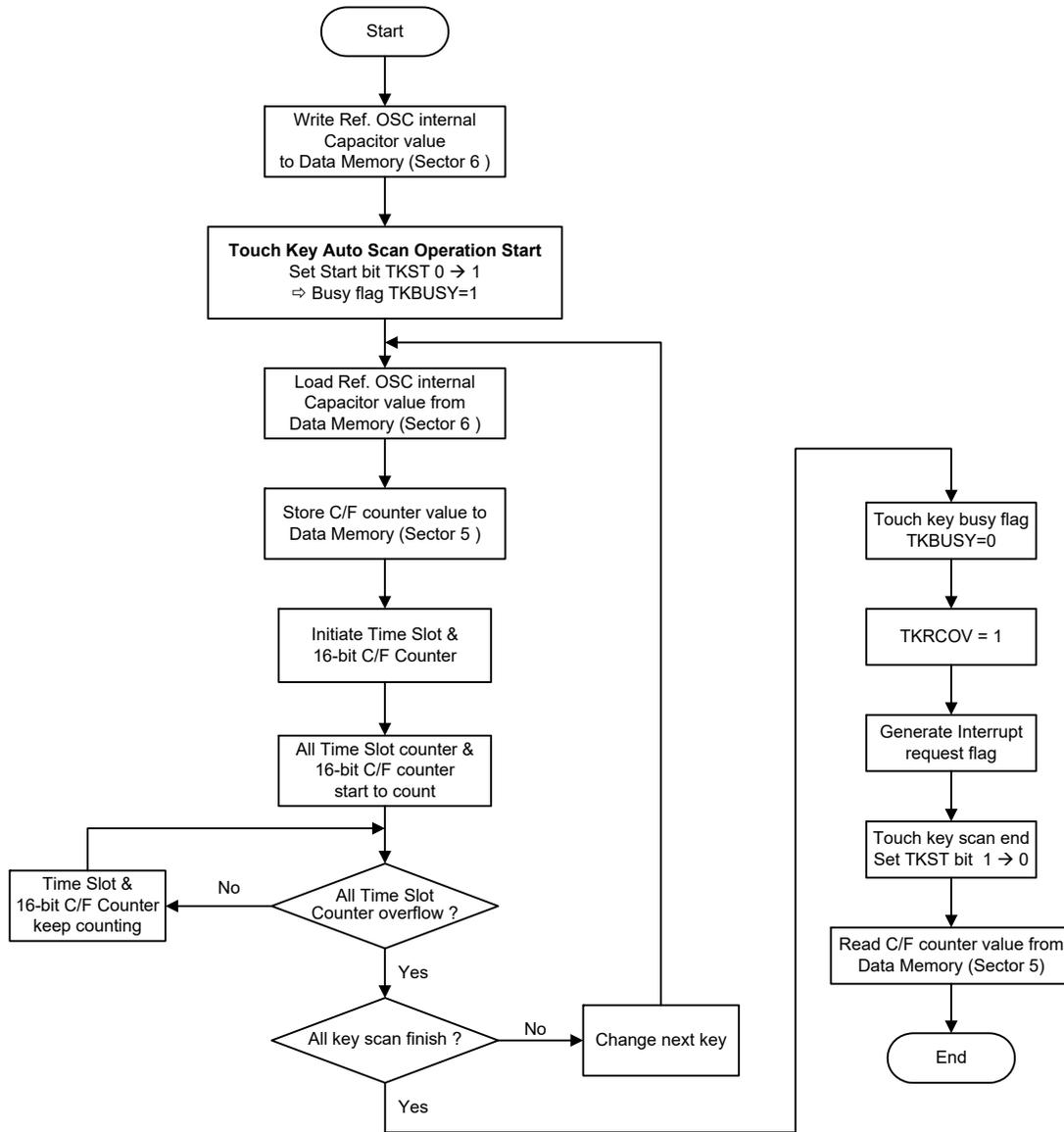


Touch Key Module Data Memory Map

Touch Key Scan Operation Flowchart



Touch Key Manual Scan Mode Flowchart – TKMOD=1, TSCS=0



Touch Key Auto Scan Mode Flowchart – TKMOD=0, TSCS=0

Touch Key Interrupt

The touch key only has a single interrupt, when the time slot counter in all the touch key modules or in the touch key module 0 overflows in manual mode or when all the touch key scan operation is complete in auto scan mode, an actual touch key interrupt will take place. The touch keys mentioned here are the keys which are enabled. The 16-bit C/F counter, 16-bit counter, 5-bit time slot unit period counter and 8-bit time slot counter in all modules will be automatically cleared. More details regarding the touch key interrupt is located in the interrupt section of the datasheet.

Programming Considerations

After the relevant registers are set, the touch key detection process is initiated by changing the TKST bit from low to high. This will enable and synchronise all relevant oscillators. The TKRCOV flag which is the time slot counter overflow flag will go high when the counter overflows. When

this happens an interrupt signal will be generated. As the TKRCOV flag will not be automatically cleared, it has to be cleared by the application program.

The TKCFOV flag which is the 16-bit C/F counter overflow flag will go high when any of the Touch Key Module 16-bit C/F counter overflows. As this flag will not be automatically cleared, it has to be cleared by the application program. The TK16OV flag which is the 16-bit counter overflow flag will go high when the 16-bit counter overflows. As this flag will not be automatically cleared, it has to be cleared by the application program.

When the external touch key size and layout are defined, their related capacitances will then determine the sensor oscillator frequency.

Universal Serial Interface Module – USIM

The device contains a Universal Serial Interface Module, which includes the four-line SPI interface, the two-line I²C interface and the two-wire/single-wire UART 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, I²C or UART based hardware such as sensors, Flash or EEPROM memory, etc. The USIM interface pins are pin-shared with other I/O pins therefore the USIM interface functional pins must first be selected using the corresponding pin-shared function selection bits. As all the interface types share the same pins and registers, the choice of whether the UART, SPI or I²C type is used is made using the UART mode selection bit, named UMD, and the SPI/I²C operating mode control bits, named SIM2~SIM0, in the SIMC0 register. These pull-high resistors of the USIM pin-shared I/O pins are selected using pull-high control registers when the USIM function is enabled and the corresponding pins are used as USIM input pins.

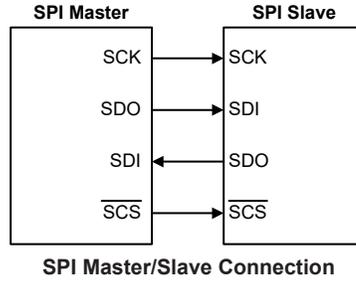
SPI Interface

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, but the device provides only one $\overline{\text{SCS}}$ pin. If the master needs to control multiple slave devices from a single master, the master can use I/O pin to select the slave devices.

SPI Interface Operation

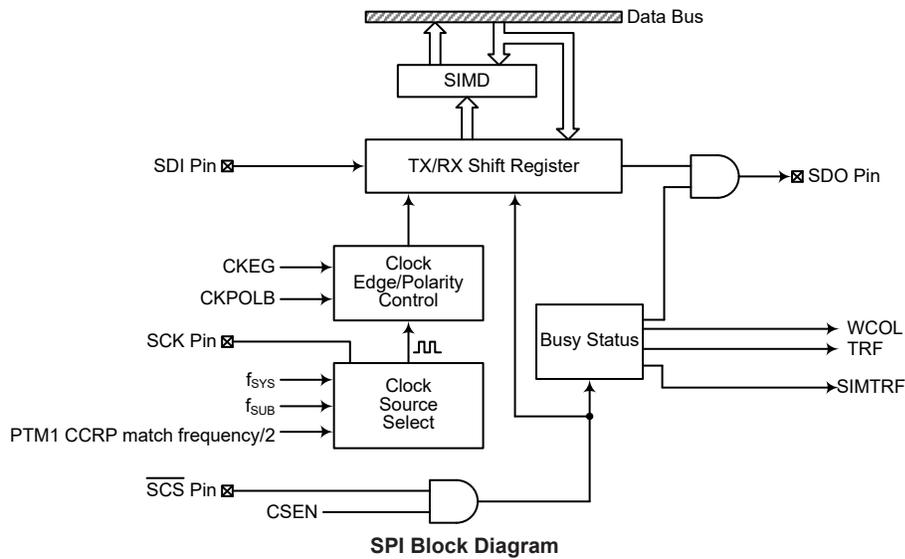
The SPI interface is a full duplex synchronous serial data link. It is a four-line interface with pin names SDI, SDO, SCK and $\overline{\text{SCS}}$. Pins SDI and SDO are the Serial Data Input and Serial Data Output lines, the SCK pin is the Serial Clock line and $\overline{\text{SCS}}$ is the Slave Select line. As the SPI interface pins are pin-shared with normal I/O pins and with the I²C/UART function pins, the SPI interface pins must first be selected by configuring the pin-shared function selection bits and setting the correct bits in the SIMC0 and SIMC2 registers. Communication between devices connected to the SPI interface is carried out in a slave/master mode with all data transfer initiations being implemented by the master. The Master also controls the clock signal. As the device only contains a single $\overline{\text{SCS}}$ pin only one slave device can be utilized. The $\overline{\text{SCS}}$ pin is controlled by software, set CSEN bit to 1 to enable $\overline{\text{SCS}}$ pin function, set CSEN bit to 0 the $\overline{\text{SCS}}$ pin will be floating state.



The SPI function in the 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 control registers, SIMC0 and SIMC2. Note that the SIMC2 and SIMD registers and their POR values are only available when the SPI mode is selected by properly configuring the UMD and SIM2~SIM0 bits in the SIMC0 register.

| Register Name | Bit | | | | | | | |
|---------------|------|------|--------|------|---------|---------|-------|--------|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SIMC0 | SIM2 | SIM1 | SIM0 | UMD | 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 I²C 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**: USIM SPI/I²C data register bit 7~bit 0

SPI Control Registers

There are also two control registers for the SPI interface, SIMC0 and SIMC2. 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 | UMD | SIMDEB1 | SIMDEB0 | SIMEN | SIMICF |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| POR | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |

Bit 7~5 **SIM2~SIM0**: USIM SPI/I²C 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 PTM1 CCRP match frequency/2
 101: SPI slave mode
 110: I²C slave mode
 111: Unused mode

When the UMD bit is cleared to zero, these bits setup the SPI or I²C operating mode of the USIM function. As well as selecting if the I²C 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 PTM1 and f_{SUB} . If the SPI Slave Mode is selected then the clock will be supplied by an external Master device.

Bit 4 **UMD**: UART mode selection bit
 0: SPI or I²C mode
 1: UART mode

This bit is used to select the UART mode. When this bit is cleared to zero, the actual SPI or I²C mode can be selected using the SIM2~SIM0 bits.

Bit 3~2 **SIMDEB1~SIMDEB0**: I²C Debounce Time Selection
 These bits are only available when the USIM is configured to operate in the I²C mode. Refer to the I²C register section.

- Bit 1** **SIMEN:** USIM SPI/I²C enable control
 0: Disable
 1: Enable
- The bit is the overall on/off control for the USIM SPI/I²C interface. When the SIMEN bit is cleared to zero to disable the USIM SPI/I²C interface, the SDI, SDO, SCK and SCS, or SDA and SCL lines will lose their SPI or I²C function and the USIM operating current will be reduced to a minimum value. When the bit is high the USIM SPI/I²C interface is enabled. If the USIM is configured to operate as an SPI interface via the UMD and 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 USIM is configured to operate as an I²C interface via the UMD and SIM2~SIM0 bits and the SIMEN bit changes from low to high, the contents of the I²C 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 I²C flags such as HCF, HAAS, HBB, SRW and RXAK will be set to their default states.
- Bit 0** **SIMICF:** USIM SPI incomplete flag
 0: USIM SPI incomplete condition is not occurred
 1: USIM SPI incomplete condition is occurred
- This bit is only available when the USIM 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 high but the \overline{SCS} line is pulled high by the external master device before the SPI data transfer is completely finished, the SIMICF bit will be set high together with the TRF bit. When this condition occurs, the corresponding interrupt will occur if the interrupt function is enabled. However, the TRF bit will not be set to 1 if the SIMICF bit is set to 1 by software application program.

• **SIMC2 Register**

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

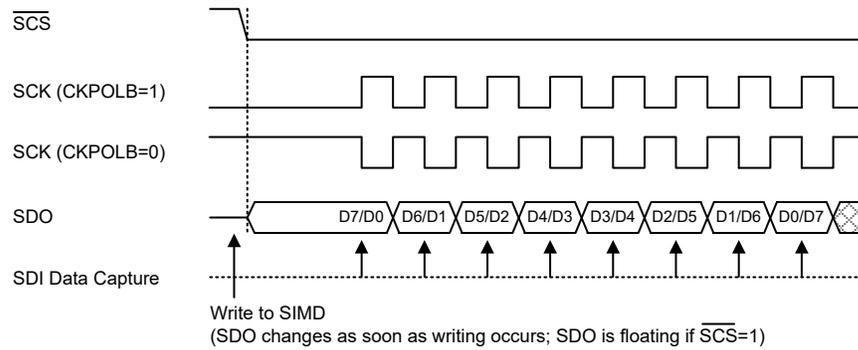
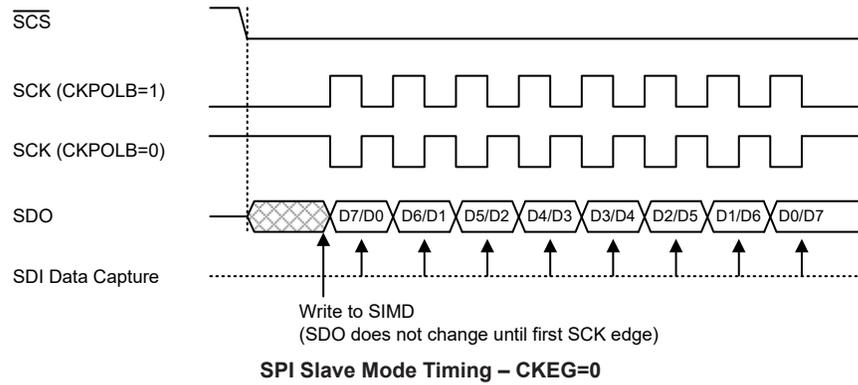
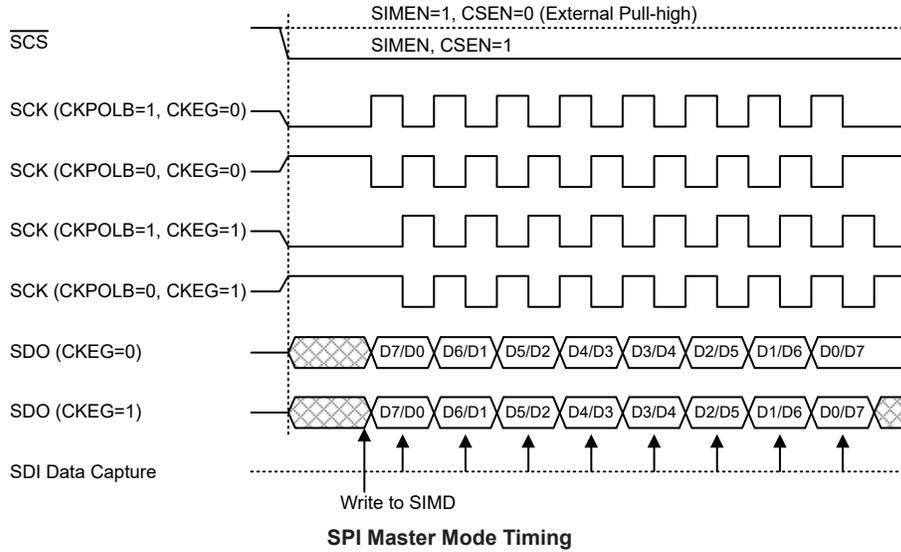
- Bit 7~6** **D7~D6:** Undefined bits
 These bits can be read or written by the application program.
- Bit 5** **CKPOLB:** SPI clock line base condition selection
 0: The SCK line will be high when the clock is inactive
 1: The SCK line will be low when the clock is inactive
- The CKPOLB bit determines the base condition of the clock line, if the bit is high, then the SCK line will be low when the clock is inactive. When the CKPOLB bit is low, then the SCK line will be high when the clock is inactive.
- Bit 4** **CKEG:** SPI SCK clock active edge type selection
 CKPOLB=0
 0: SCK is high base level and data capture at SCK rising edge
 1: SCK is high base level and data capture at SCK falling edge
 CKPOLB=1
 0: SCK is low base level and data capture at SCK falling edge
 1: SCK is low base level and data capture at SCK rising edge
- The CKEG and CKPOLB bits are used to setup the way that the clock signal outputs and inputs data on the SPI bus. These two bits must be configured before data transfer is executed otherwise an erroneous clock edge may be generated. The CKPOLB bit determines the base condition of the clock line, if the bit is high, then the SCK line will be low when the clock is inactive. When the CKPOLB bit is low, then the SCK line will be high when the clock is inactive. The CKEG bit determines active clock edge type which depends upon the condition of CKPOLB bit.

- | | |
|-------|---|
| Bit 3 | <p>MLS: SPI data shift order 0: LSB first 1: MSB first</p> <p>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.</p> |
| Bit 2 | <p>CSEN: SPI \overline{SCS} pin control 0: Disable 1: Enable</p> <p>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.</p> |
| Bit 1 | <p>WCOL: SPI write collision flag 0: No collision 1: Collision</p> <p>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.</p> |
| Bit 0 | <p>TRF: SPI Transmit/Receive complete flag 0: SPI data is being transferred 1: SPI data transmission is completed</p> <p>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.</p> |

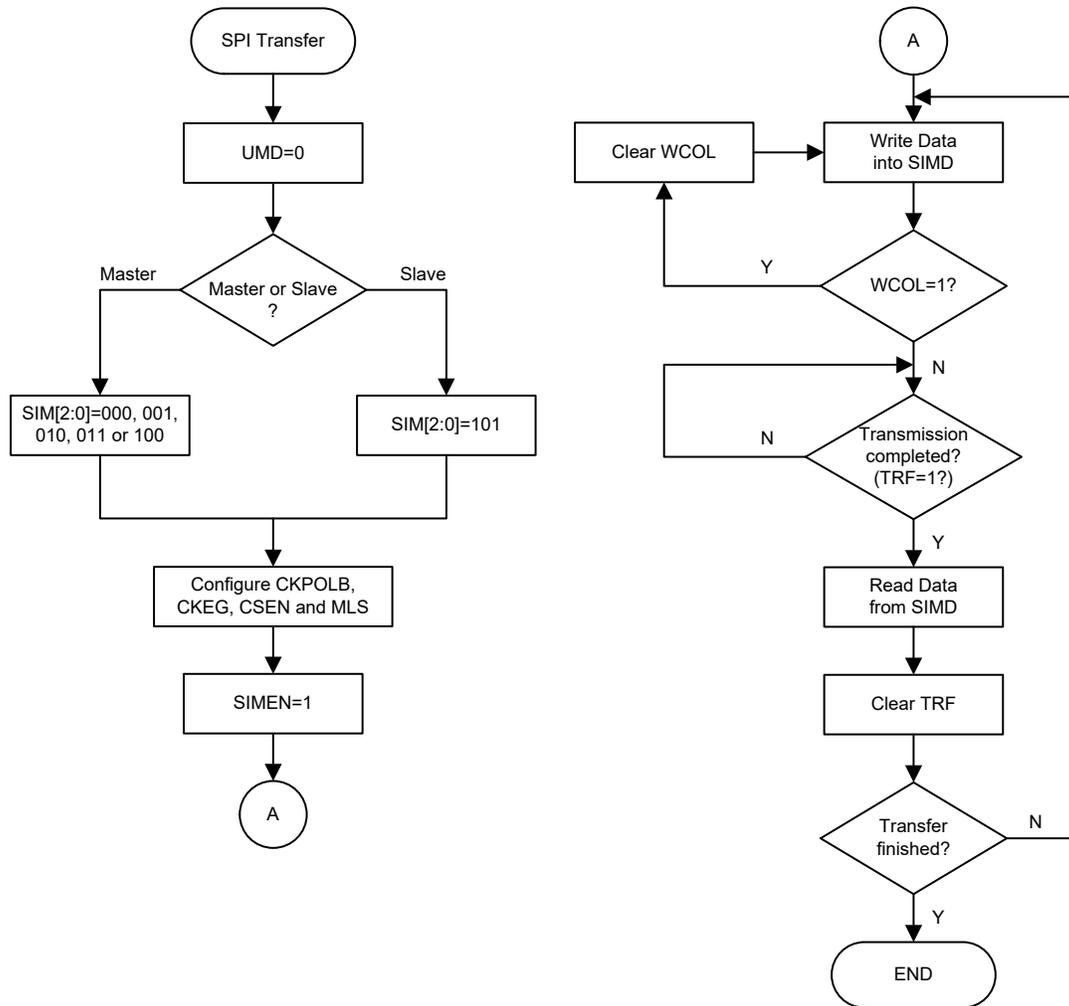
SPI Communication

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 completed, the TRF flag will be set high automatically, but must be cleared using the application program. In the Slave Mode, when the clock signal from the master has been received, any data in the SIMD register will be transmitted and any data on the SDI pin will be shifted into the SIMD register. The master should output a \overline{SCS} signal to enable the slave devices before a clock signal is provided. The slave data to be transferred should be well prepared at the appropriate moment relative to the 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 will continue to function in certain IDLE Modes if the clock source used by the SPI interface is still active.



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



SPI Transfer Control Flowchart

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 pin-shared 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 \overline{SCS} pin 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 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 UMD and 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 step 5.
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 a USIM 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 UMD and 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 SCS signal. After this, go to step 5.
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.

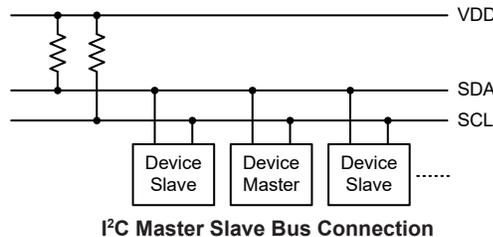
- Step 5
 Check the WCOL bit if set high then a collision error has occurred so return to step 4. If equal to zero then go to the following step.
- Step 6
 Check the TRF bit or wait for a USIM SPI serial bus interrupt.
- Step 7
 Read data from the SIMD register.
- Step 8
 Clear TRF.
- Step 9
 Go to step 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 that 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.

I²C Interface

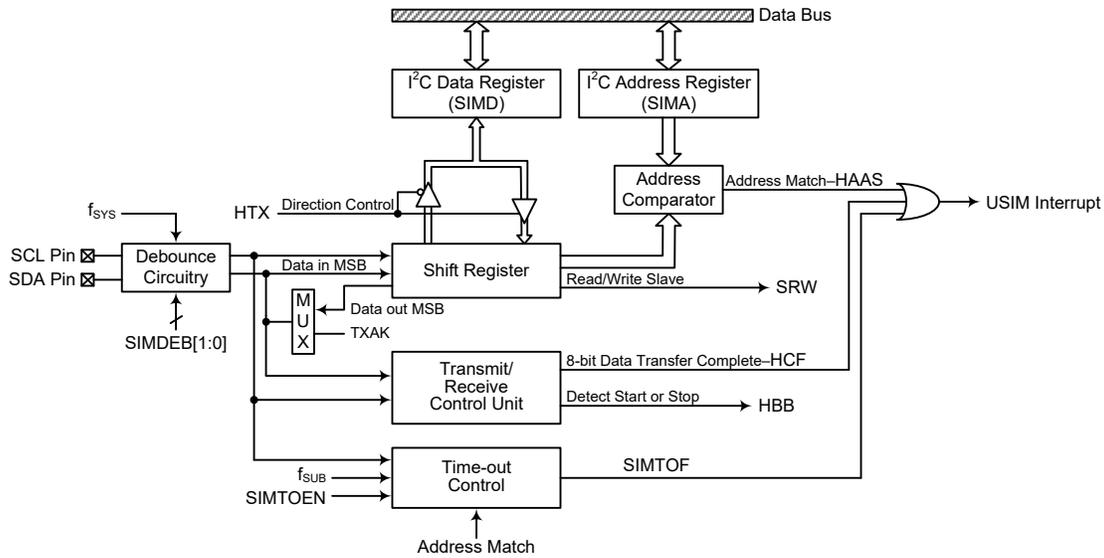
The I²C interface is used to communicate with external peripheral devices such as sensors, EEPROM memory etc. Originally developed by Philips, it is a two-line low speed serial interface for synchronous serial data transfer. The advantage of only two lines for communication, relatively simple communication protocol and the ability to accommodate multiple devices on the same bus has made it an extremely popular interface type for many applications.



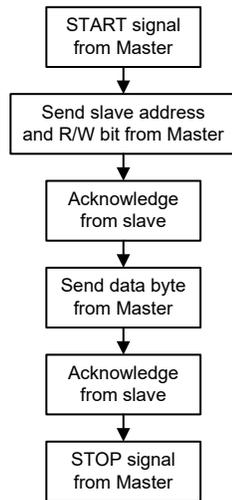
I²C Interface Operation

The I²C 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 I²C bus is identified by a unique address which will be transmitted and received on the I²C bus.

When two devices communicate with each other on the bidirectional I²C 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 I²C 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 I²C device is activated and the related internal pull-high register could be controlled by its corresponding pull-high control register.



I²C Block Diagram



I²C Interface Operation

The SIMDEB1 and SIMDEB0 bits determine the debounce time of the I²C interface. This uses the internal 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 I²C data transfer speed, there exists a relationship between the system clock, f_{SYS} , and the I²C debounce time. For either the I²C 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 ² C Debounce Time Selection | I ² C Standard Mode (100kHz) | I ² 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 | $f_{SYS} > 4\text{MHz}$ | $f_{SYS} > 8\text{MHz}$ |

I²C Minimum f_{SYS} Frequency Requirements

I²C Registers

There are three control registers associated with the I²C bus, SIMC0, SIMC1 and SIMTOC, one address register SIMA and one data register, SIMD. Note that the SIMC1, SIMD, SIMA and SIMTOC registers and their POR values are only available when the I²C mode is selected by properly configuring the UMD and SIM2~SIM0 bits in the SIMC0 register.

| Register Name | Bit | | | | | | | |
|---------------|---------|--------|---------|---------|---------|---------|---------|---------|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SIMC0 | SIM2 | SIM1 | SIM0 | UMD | 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 |

I²C Register List

I²C 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 I²C functions. Before the device writes data to the I²C bus, the actual data to be transmitted must be placed in the SIMD register. After the data is received from the I²C bus, the device can read it from the SIMD register. Any transmission or reception of data from the I²C 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**: USIM SPI/I²C data register bit 7~bit 0

I²C 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. Bits 7~1 of the SIMA register define the device slave address. Bit 0 is not defined. When a master device, which is connected to the I²C bus, sends out an address, which matches the slave address in the SIMA register, the slave device will be selected.

• 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**: I²C slave address
SIMA6~SIMA0 is the I²C slave address bit 6~bit 0.

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

I²C Control Registers

There are three control registers for the I²C interface, SIMC0, SIMC1 and SIMTOC. The SIMC0 register is used to control the enable/disable function and to set the data transmission clock frequency. The SIMC1 register contains the relevant flags which are used to indicate the I²C communication status. Another register, SIMTOC, is used to control the I²C time-out function and is described in the corresponding section.

• SIMC0 Register

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

Bit 7~5 **SIM2~SIM0**: USIM SPI/I²C 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 PTM1 CCRP match frequency/2
 101: SPI slave mode
 110: I²C slave mode
 111: Unused mode

When the UMD bit is cleared to zero, these bits setup the SPI or I²C operating mode of the USIM function. As well as selecting if the I²C 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 PTM1 and f_{SUB} . If the SPI Slave Mode is selected then the clock will be supplied by an external Master device.

Bit 4 **UMD**: UART mode selection bit
 0: SPI or I²C mode
 1: UART mode

This bit is used to select the UART mode. When this bit is cleared to zero, the actual SPI or I²C mode can be selected using the SIM2~SIM0 bits.

Bit 3~2 **SIMDEB1~SIMDEB0**: I²C Debounce Time Selection
 00: No debounce
 01: 2 system clock debounce
 1x: 4 system clock debounce

These bits are used to select the I²C debounce time when the USIM is configured as the I²C interface function by setting the UMD bit to “0” and the SIM2~SIM0 bits to “110”.

Bit 1 **SIMEN**: USIM SPI/I²C Enable Control
 0: Disable
 1: Enable

The bit is the overall on/off control for the USIM SPI/I²C interface. When the SIMEN bit is cleared to zero to disable the USIM SPI/I²C interface, the SDI, SDO, SCK and SCS, or SDA and SCL lines will lose their SPI or I²C function and the USIM operating current will be reduced to a minimum value. When the bit is high the USIM SPI/I²C interface is enabled. If the USIM is configured to operate as an SPI interface via the UMD and 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 USIM is configured to operate as an I²C interface via the UMD and SIM2~SIM0 bits and the SIMEN bit changes from low to high, the contents of the I²C 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 I²C flags such as HCF, HAAS, HBB, SRW and RXAK will be set to their default states.

Bit 0 **SIMICF**: USIM SPI Incomplete Flag
 This bit is only available when the USIM 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 | R/W | R |
| POR | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

Bit 7 **HCF**: I²C 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**: I²C 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**: I²C Bus busy flag
 0: I²C Bus is not busy
 1: I²C Bus is busy
 The HBB flag is the I²C busy flag. This flag will be “1” when the I²C 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**: I²C slave device is transmitter or receiver selection
 0: Slave device is the receiver
 1: Slave device is the transmitter

Bit 3 **TXAK**: I²C Bus transmit acknowledge flag
 0: Slave send acknowledge flag
 1: Slave do 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 set TXAK bit to “0” before further data is received.

Bit 2 **SRW**: I²C 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 I²C Slave Read/Write flag. This flag determines whether the master device wishes to transmit or receive data from the I²C 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**: I²C Address Match Wake-up control
 0: Disable
 1: Enable
 This bit should be set to 1 to enable the I²C 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 I²C address match wake-up, then this bit must be cleared by the application program after wake-up to ensure correction device operation.

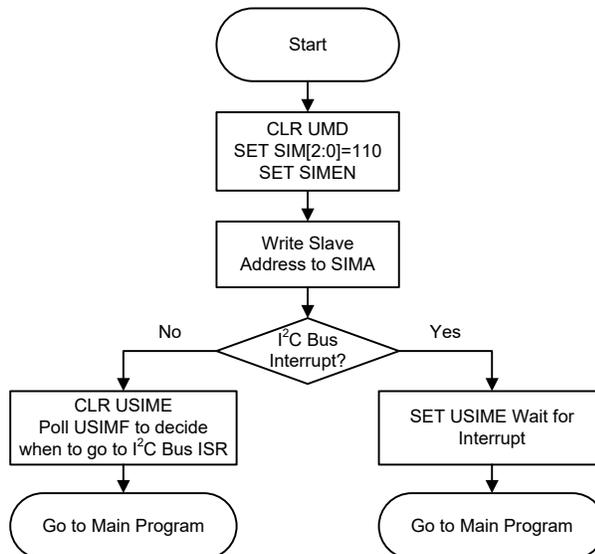
Bit 0 **RXAK:** I²C Bus Receive acknowledge flag
 0: Slave receive acknowledge flag
 1: Slave does not receive acknowledge flag

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 I²C Bus.

I²C Bus Communication

Communication on the I²C 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 I²C 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 a USIM 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 from the I²C 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 I²C bus, the microcontroller must initialise the bus, the following are steps to achieve this:

- Step 1
 Set the UMD, SIM2~SIM0 and SIMEN bits in the SIMC0 register to “0”, “110” and “1” respectively to enable the I²C bus.
- Step 2
 Write the slave address of the device to the I²C bus address register SIMA.
- Step 3
 Set the USIME interrupt enable bit of the interrupt control register to enable the USIM interrupt.



I²C Bus Initialisation Flow Chart

I²C Bus Start Signal

The START signal can only be generated by the master device connected to the I²C bus and not by the slave device. This START signal will be detected by all devices connected to the I²C bus. When detected, this indicates that the I²C 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.

I²C Slave Address

The transmission of a START signal by the master will be detected by all devices on the I²C 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 USIM I²C 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 USIM I²C bus interrupt can come from three sources, when the program enters the interrupt subroutine, the HAAS and SIMTOF bits should be examined to see whether the interrupt source has come from a matching slave address or from the completion of a data byte transfer or from the I²C 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.

I²C Bus Read/Write Signal

The SRW bit in the SIMC1 register defines whether the master device wishes to read data from the I²C bus or write data to the I²C 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 I²C bus, therefore the slave device must be setup to send data to the I²C bus as a transmitter. If the SRW flag is “0” then this indicates that the master wishes to send data to the I²C bus, therefore the slave device must be setup to read data from the I²C bus as a receiver.

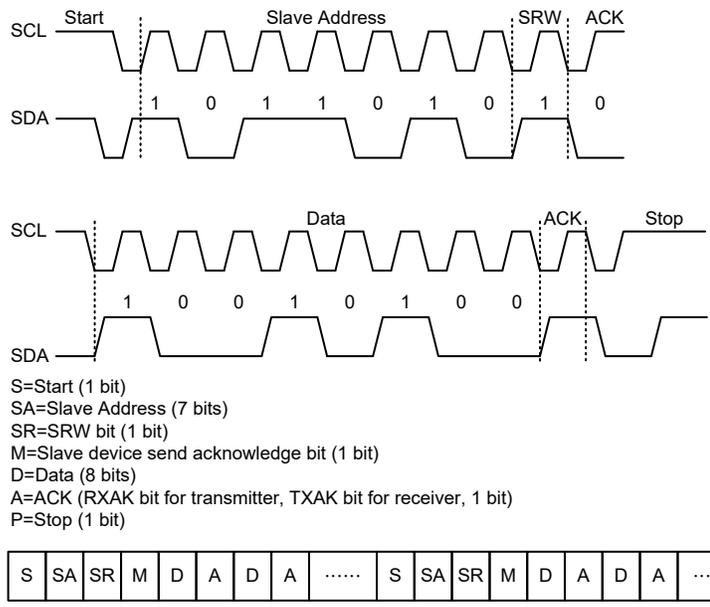
I²C Bus Slave Address Acknowledge Signal

After the master has transmitted a calling address, any slave device on the I²C 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”.

I²C Bus Data and Acknowledge Signal

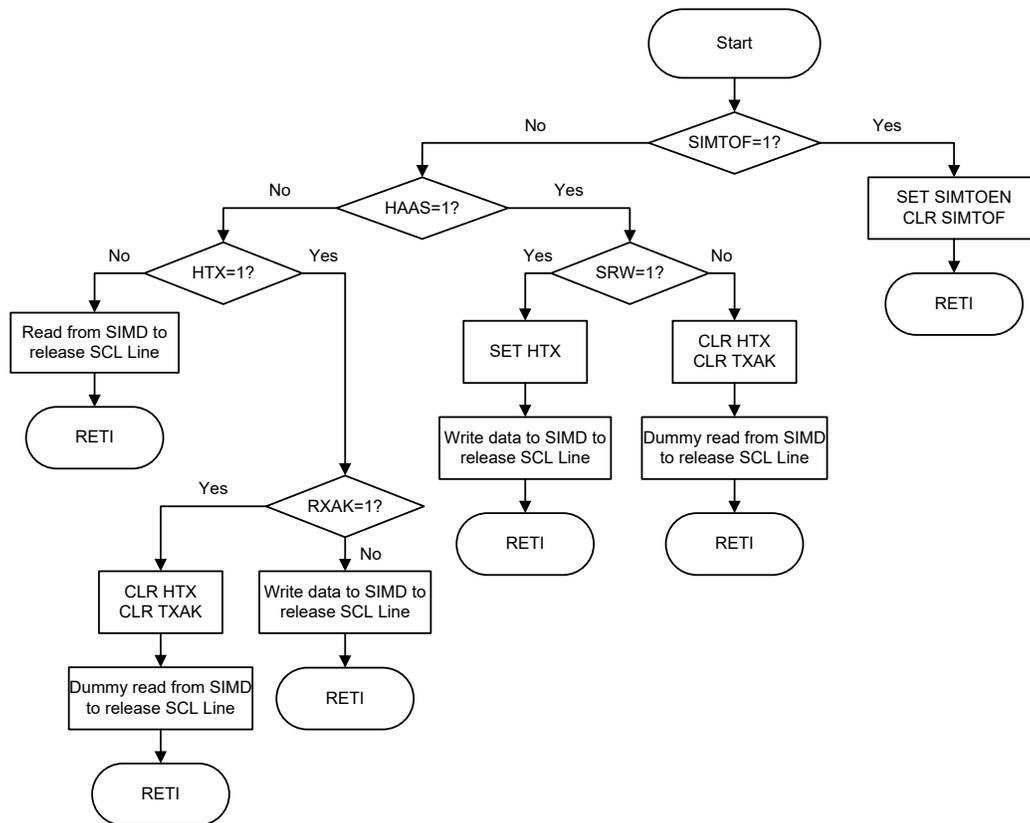
The transmitted data is 8-bit wide and is transmitted after the slave device has acknowledged receipt of its slave address. The order of serial bit transmission is the MSB first and the LSB last. After receipt of 8 bits of data, the receiver must transmit an acknowledge signal, level “0”, before it can 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 I²C 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.



I²C Communication Timing Diagram

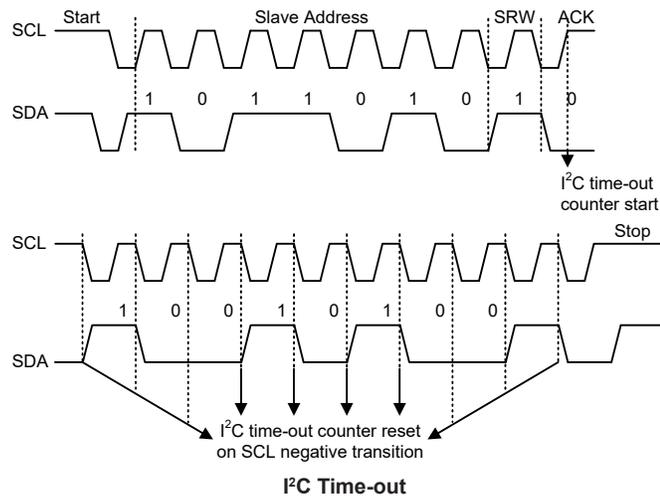
Note: When a slave address is matched, the device must be placed in either the transmit mode and then write data to the SIMD register, or in the receive mode where it must implement a dummy read from the SIMD register to release the SCL line.



I²C Bus ISR Flow Chart

I²C Time-out Control

In order to reduce the problem of I²C lockup due to reception of erroneous clock sources, a time-out function is provided. If the clock source to the I²C is not received for a while, then the I²C circuitry and registers will be reset after a certain time-out period. The time-out counter starts counting on an I²C 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 I²C “STOP” condition occurs.



When an I²C 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 USIM interrupt vector. When an I²C time-out occurs, the I²C internal circuitry will be reset and the registers will be reset into the following condition:

| Registers | After I ² C Time-out |
|-------------------|---------------------------------|
| SIMD, SIMA, SIMC0 | No change |
| SIMC1 | Reset to POR condition |

I²C Registers after Time-out

The SIMTOF flag can be cleared by the application program. There are 64 time-out periods which can be selected using SIMTOS5~SIMTOS0 bits in the SIMTOC register. The time-out time is given by the formula: $((1\sim64)\times 32)/f_{SUB}$. This gives a time-out period which ranges from about 1ms to 64ms.

• **SIMTOC Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------|---------|--------|---------|---------|---------|---------|---------|---------|
| Name | SIMTOEN | SIMTOF | SIMTOS5 | SIMTOS4 | SIMTOS3 | SIMTOS2 | SIMTOS1 | SIMTOS0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| POR | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Bit 7 **SIMTOEN**: USIM I²C Time-out control
 0: Disable
 1: Enable

Bit 6 **SIMTOF**: USIM I²C 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 by application program.

Bit 5~0 **SIMTOS5~SIMTOS0**: USIM I²C Time-out period selection
 I²C time-out clock source is $f_{SUB}/32$.
 I²C time-out time is equal to $(SIMTOS[5:0]+1)\times(32/f_{SUB})$.

UART Interface

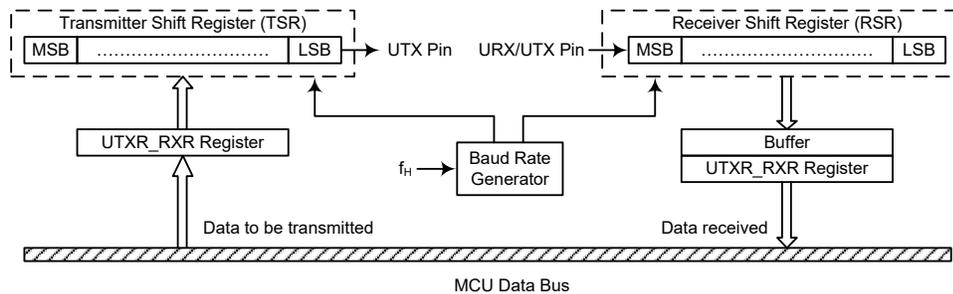
This UART interface function, which is part of the Universal Serial Interface Module, should not be confused with the other independent UART function, which is described in another section of this datasheet.

The device contains an integrated full-duplex or half-duplex asynchronous serial communications UART interface that enables communication with external devices that contain a serial interface. The UART function has many features and can transmit and receive data serially by transferring a frame of data with eight or nine data bits per transmission as well as being able to detect errors when the data is overwritten or incorrectly framed. The UART function shares the same internal interrupt vector with the SPI and I²C interfaces which can be used to indicate when a reception occurs or when a transmission terminates.

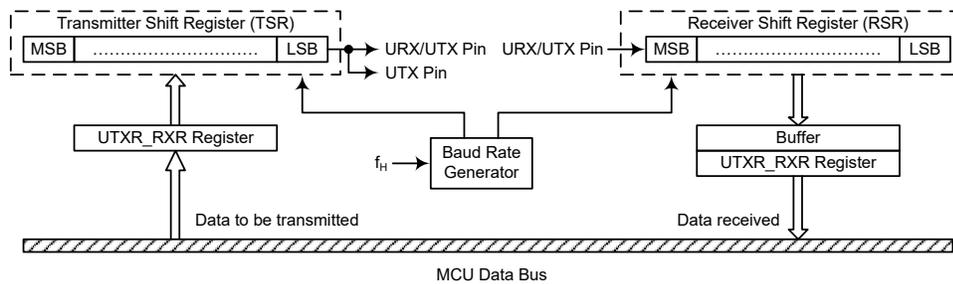
The integrated UART function contains the following features:

- Full-duplex or half-duplex (single wire mode) asynchronous communication
- 8 or 9 bits character length
- Even, odd or no parity options
- One or two stop bits

- Baud rate generator with 8-bit prescaler
- Parity, framing, noise and overrun error detection
- Support for interrupt on address detect (last character bit=1)
- Separately enabled transmitter and receiver
- 2-byte Deep FIFO Receive Data Buffer
- URX/UTX pin wake-up function
- Transmit and receive interrupts
- Interrupts can be triggered by the following conditions:
 - ♦ Transmitter Empty
 - ♦ Transmitter Idle
 - ♦ Receiver Full
 - ♦ Receiver Overrun
 - ♦ Address Mode Detect



UART Data Transfer Block Diagram – USWM=0



UART Data Transfer Block Diagram – USWM=1

UART External Pins

To communicate with an external serial interface, the internal UART has two external pins known as UTX pin and URX/UTX pin, which are pin-shared with I/O or other pin functions. The UTX and URX/UTX pin function should first be selected by the corresponding pin-shared function selection register before the UART function is used. Along with the UMD bit, the UREN bit, the UTXEN or URXEN bits, if set, will setup these pins to transmitter output and receiver input conditions. At this time the internal pull-high resistor related to the transmitter output pin will be disabled, while the internal pull-high resistor related to the receiver input pin is controlled by the corresponding I/O pull-high function control bit. When the UTX or URX/UTX pin function is disabled by clearing the UMD, UREN, UTXEN or URXEN bit, the UTX or URX/UTX pin will be set to a floating state. At this time whether the internal pull-high resistor is connected to the UTX or URX/UTX pin or not is determined by the corresponding I/O pull-high function control bit.

UART Single Wire Mode

The UART function also supports a Single Wire Mode communication which is selected using the USWM bit in the UUCR3 register. When the USWM bit is set high, the UART function will be in the single wire mode. In the single wire mode, a single URX/UTX pin can be used to transmit and receive data depending upon the corresponding control bits. When the URXEN bit is set high, the URX/UTX pin is used as a receiver pin. When the URXEN bit is cleared to zero and the UTXEN bit is set high, the URX/UTX pin will act as a transmitter pin.

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

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

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

UART Data Transfer Scheme

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

Data to be received by the UART is accepted on the external URX/UTX 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 UTXR_RXR register, where it is buffered and can be manipulated by the application program. Only the UTXR_RXR register is mapped onto the MCU Data Memory, the Receiver Shift Register is not mapped and is therefore inaccessible to the application program.

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

UART Status and Control Registers

There are seven control registers associated with the UART function. The UMD bit in the SIMC0 register can be used to select the UART mode. The USWM bit in the UUCR3 register is used to enable/disable the UART Single Wire Mode. The UUSR, UUCR1 and UUCR2 registers control the overall function of the UART, while the UBRG register controls the Baud rate. The actual data to be transmitted and received on the serial interface is managed through the UTXR_RXR data register. Note that UART related registers and their POR values are only available when the UART mode is selected by setting the UMD bit in the SIMC0 register to "1".

| Register Name | Bit | | | | | | | |
|---------------|-------|-------|-------|--------|---------|---------|--------|--------|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SIMC0 | SIM2 | SIM1 | SIM0 | UMD | SIMDEB1 | SIMDEB0 | SIMEN | SIMICF |
| UUSR | UPERR | UNF | UFERR | UOERR | URIDLE | URXIF | UTIDLE | UTXIF |
| UUCR1 | UREN | UBNO | UPREN | UPRT | USTOPS | UTXBRK | URX8 | UTX8 |
| UUCR2 | UTXEN | URXEN | UBRGH | UADDEN | UWAKE | URIE | UTIIE | UTEIE |
| UUCR3 | — | — | — | — | — | — | — | USWM |
| UTXR_RXR | UTXR7 | UTXR6 | UTXR5 | UTXR4 | UTXR3 | UTXR2 | UTXR1 | UTXR0 |
| UBRG | UBRG7 | UBRG6 | UBRG5 | UBRG4 | UBRG3 | UBRG2 | UBRG1 | UBRG0 |

UART Register List

• **SIMC0 Register**

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

- Bit 7~5 **SIM2~SIM0**: USIM SPI/I²C Operating Mode Control
When the UMD bit is cleared to zero, these bits setup the SPI or I²C operating mode of the USIM function. Refer to the SPI or I²C register section for more details.
- Bit 4 **UMD**: UART mode selection bit
0: SPI or I²C mode
1: UART mode
This bit is used to select the UART mode. When this bit is cleared to zero, the actual SPI or I²C mode can be selected using the SIM2~SIM0 bits.
- Bit 3~2 **SIMDEB1~SIMDEB0**: I²C Debounce Time Selection
Refer to the I²C register section.
- Bit 1 **SIMEN**: USIM SPI/I²C Enable Control
This bit is only available when the USIM is configured to operate in an SPI or I²C mode with the UMD bit set low. Refer to the SPI or I²C register section for more details.
- Bit 0 **SIMICF**: USIM SPI Incomplete Flag
Refer to the SPI register section.

• **UUSR Register**

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

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------|-------|-----|-------|-------|--------|-------|--------|-------|
| Name | UPERR | UNF | UFERR | UOERR | URIDLE | URXIF | UTIDLE | UTXIF |
| R/W | R | R | R | R | R | R | R | R |
| POR | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 |

- Bit 7 **UPERR**: Parity error flag
0: No parity error is detected
1: Parity error is detected
The UPERR flag is the parity error flag. When this read only flag is “0”, it indicates a parity error has not been detected. When the flag is “1”, it indicates that the parity of the received word is incorrect. This error flag is applicable only if Parity mode (odd or even) is selected. The flag can also be cleared by a software sequence which involves a read to the status register UUSR followed by an access to the UTXR_RXR data register.

- Bit 6 **UNF**: Noise flag
 0: No noise is detected
 1: Noise is detected
The UNF flag is the noise flag. When this read only flag is “0”, it indicates no noise condition. When the flag is “1”, it indicates that the UART has detected noise on the receiver input. The UNF flag is set during the same cycle as the URXIF flag but will not be set in the case of an overrun. The UNF flag can be cleared by a software sequence which will involve a read to the status register UUSR followed by an access to the UTXR_RXR data register.
- Bit 5 **UFERR**: Framing error flag
 0: No framing error is detected
 1: Framing error is detected
The UFERR flag is the framing error flag. When this read only flag is “0”, it indicates that there is no framing error. When the flag is “1”, it indicates that a framing error has been detected for the current character. The flag can also be cleared by a software sequence which will involve a read to the status register UUSR followed by an access to the UTXR_RXR data register.
- Bit 4 **UOERR**: Overrun error flag
 0: No overrun error is detected
 1: Overrun error is detected
The UOERR 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 UTXR_RXR receive data register. The flag is cleared by a software sequence, which is a read to the status register UUSR followed by an access to the UTXR_RXR data register.
- Bit 3 **URIDLE**: Receiver status
 0: Data reception is in progress (Data being received)
 1: No data reception is in progress (Receiver is idle)
The URIDLE 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 URIDLE bit is “1” indicating that the UART receiver is idle and the URX/UTX pin stays in logic high condition.
- Bit 2 **URXIF**: Receive UTXR_RXR data register status
 0: UTXR_RXR data register is empty
 1: UTXR_RXR data register has available data
The URXIF flag is the receive data register status flag. When this read only flag is “0”, it indicates that the UTXR_RXR read data register is empty. When the flag is “1”, it indicates that the UTXR_RXR read data register contains new data. When the contents of the shift register are transferred to the UTXR_RXR register, an interrupt is generated if URIE=1 in the UUCR2 register. If one or more errors are detected in the received word, the appropriate receive-related flags UNF, UFERR, and/or UPERR are set within the same clock cycle. The URXIF flag will eventually be cleared when the UUSR register is read with URXIF set, followed by a read from the UTXR_RXR register, and if the UTXR_RXR register has no more new data available.
- Bit 1 **UTIDLE**: Transmission idle
 0: Data transmission is in progress (Data being transmitted)
 1: No data transmission is in progress (Transmitter is idle)
The UTIDLE 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 URXIF flag is “1” and when there is no transmit data or break character being transmitted. When UTIDLE is equal to “1”, the UTX pin becomes idle with the pin state in logic high condition. The UTIDLE flag is cleared by reading the UUSR

register with UTIDLE set and then writing to the UTXR_RXR register. The flag is not generated when a data character or a break is queued and ready to be sent.

Bit 0 **UTXIF:** Transmit UTXR_RXR data register status
 0: Character is not transferred to the transmit shift register
 1: Character has transferred to the transmit shift register (UTXR_RXR data register is empty)

The UTXIF 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 UTXR_RXR data register. The UTXIF flag is cleared by reading the UART status register (UUSR) with UTXIF set and then writing to the UTXR_RXR data register. Note that when the UTXEN bit is set, the UTXIF flag bit will also be set since the transmit data register is not yet full.

• **UUCR1 Register**

The UUCR1 register together with the UUCR2 and UUCR3 registers are the three UART control registers that are used to set the various options for the UART function, such as overall on/off control, parity control, data transfer bit length, 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 | UREN | UBNO | UPREN | UPRT | USTOPS | UTXBRK | URX8 | UTX8 |
| 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 **UREN:** UART function enable control
 0: Disable UART. UTX and URX/UTX pins are in a floating state
 1: Enable UART. UTX and URX/UTX pins function as UART pins

The UREN bit is the UART enable bit. When this bit is equal to “0”, the UART will be disabled and the URX/UTX pin as well as the UTX pin will be set in a floating state. When the bit is equal to “1”, the UART will be enabled if the UMD bit is set and the UTX and URX/UTX pins will function as defined by the USWM mode selection bit together with the UTXEN and URXEN enable control bits.

When the UART is disabled, it will empty the buffer so any character remaining in the buffer will be discarded. In addition, the value of the baud rate counter will be reset. If the UART is disabled, all error and status flags will be reset. Also the UTXEN, URXEN, UTXBRK, URXIF, UOERR, UFERR, UPERR and UNF bits will be cleared, while the UTIDLE, UTXIF and URIDLE bits will be set. Other control bits in UUCR1, UUCR2, UUCR3 and UBRG registers will remain unaffected. If the UART is active and the UREN bit is cleared, all pending transmissions and receptions will be terminated and the module will be reset as defined above. When the UART is re-enabled, it will restart in the same configuration.

Bit 6 **UBNO:** 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 URX8 and UTX8 will be used to store the 9th bit of the received and transmitted data respectively.

Bit 5 **UPREN:** 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 **UPRT**: Parity type selection bit
 0: Even parity for parity generator
 1: Odd parity for parity generator
 This bit is the parity type selection bit. When this bit is equal to “1”, odd parity type will be selected. If the bit is equal to “0”, then even parity type will be selected.
- Bit 3 **USTOPS**: 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 2 **UTXBRK**: Transmit break character
 0: No break character is transmitted
 1: Break characters transmit
 The UTXBRK bit is the Transmit Break Character bit. When this bit is “0”, there are no break characters and the UTX 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 UTXBRK bit is reset.
- Bit 1 **URX8**: 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 URX8. The UBNO bit is used to determine whether data transfers are in 8-bit or 9-bit format.
- Bit 0 **UTX8**: 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 UTX8. The UBNO bit is used to determine whether data transfers are in 8-bit or 9-bit format.

• **UUCR2 Register**

The UUCR2 register is the second of the UART control registers and serves several purposes. One of its main functions is to control the basic enable/disable operation of the UART Transmitter and Receiver as well as enabling the various USIM UART mode interrupt sources. The register also serves to control the baud rate speed, 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 | UTXEN | URXEN | UBRGH | UADDEN | UWAKE | URIE | UTIIE | UTEIE |
| 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 **UTXEN**: UART Transmitter enabled control
 0: UART transmitter is disabled
 1: UART transmitter is enabled
 The bit named UTXEN 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 UTX pin will be set in a floating state.
 If the UTXEN bit is equal to “1” and the UMD and UREN bit are also equal to “1”, the transmitter will be enabled and the UTX pin will be controlled by the UART. Clearing the UTXEN bit during a transmission will cause the data transmission to be aborted and will reset the transmitter. If this situation occurs, the UTX pin will be set in a floating state.

- Bit 6 **URXEN:** UART Receiver enabled control
 0: UART receiver is disabled
 1: UART receiver is enabled
 The bit named URXEN 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 URX/UTX pin will be set in a floating state. If the URXEN bit is equal to “1” and the UMD and UREN bit are also equal to “1”, the receiver will be enabled and the URX/UTX pin will be controlled by the UART. Clearing the URXEN bit during a reception will cause the data reception to be aborted and will reset the receiver. If this situation occurs, the URX/UTX pin will be set in a floating state.
- Bit 5 **UBRGH:** Baud Rate speed selection
 0: Low speed baud rate
 1: High speed baud rate
 The bit named UBRGH selects the high or low speed mode of the Baud Rate Generator. This bit, together with the value placed in the baud rate register UBRG, controls the Baud Rate of the UART. If this bit is equal to “1”, the high speed mode is selected. If the bit is equal to “0”, the low speed mode is selected.
- Bit 4 **UADDEN:** Address detect function enable control
 0: Address detect function is disabled
 1: Address detect function is enabled
 The bit named UADDEN 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 UTXX7 if UBNO=0 or the 9th bit, which corresponds to URX8 if UBNO=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 UBNO. 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 **UWAKE:** URX/UTX pin wake-up UART function enable control
 0: URX/UTX pin wake-up UART function is disabled
 1: URX/UTX pin wake-up UART function is enabled
 This bit is used to control the wake-up UART function when a falling edge on the URX/UTX pin occurs. Note that this bit is only available when the UART clock (f_{H}) is switched off. There will be no URX/UTX pin wake-up UART function if the UART clock (f_{H}) exists. If the UWAKE bit is set to 1 as the UART clock (f_{H}) is switched off, a UART wake-up request will be initiated when a falling edge on the URX/UTX pin occurs. When this request happens and the corresponding interrupt is enabled, an URX/UTX pin wake-up UART interrupt will be generated to inform the MCU to wake up the UART function by switching on the UART clock (f_{H}) via the application program. Otherwise, the UART function cannot resume even if there is a falling edge on the URX/UTX pin when the UWAKE bit is cleared to 0.
- Bit 2 **URIE:** 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 UOERR or receive data available flag URXIF is set, the USIM interrupt request flag USIMF will be set. If this bit is equal to “0”, the USIM interrupt request flag USIMF will not be influenced by the condition of the UOERR or URXIF flags.
- Bit 1 **UTIE:** 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 UTIDLE is set, due to a transmitter idle condition, the USIM interrupt request flag USIMF will be set. If this bit is equal to “0”, the USIM interrupt request flag USIMF will not be influenced by the condition of the UTIDLE flag.

Bit 0 **UTEIE**: 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 UTXIF is set, due to a transmitter empty condition, the USIM interrupt request flag USIMF will be set. If this bit is equal to “0”, the USIM interrupt request flag USIMF will not be influenced by the condition of the UTXIF flag.

• **UUCR3 Register**

The UUCR3 register is used to enable the UART Single Wire Mode communication. As the name suggests in the single wire mode the UART communication can be implemented in one single line, URX/UTX, together with the control of the URXEN and UTXEN bits in the UUCR2 register.

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

Bit 7~1 Unimplemented, read as “0”

Bit 0 **USWM**: Single wire mode enable control
 0: Disable, the URX/UTX pin is used as UART receiver function only
 1: Enable, the URX/UTX pin can be used as UART receiver or transmitter function controlled by the URXEN and UTXEN bits

Note that when the Single Wire Mode is enabled, if both the URXEN and UTXEN bits are high, the URX/UTX pin will just be used as UART receiver input.

• **UTXR_RXR Register**

The UTXR_RXR register is the data register which is used to store the data to be transmitted on the UTX pin or being received from the URX/UTX pin.

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------|--------|--------|--------|--------|--------|--------|--------|--------|
| Name | UTXRX7 | UTXRX6 | UTXRX5 | UTXRX4 | UTXRX3 | UTXRX2 | UTXRX1 | UTXRX0 |
| 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 **UTXRX7~UTXRX0**: UART Transmit/Receive Data bit 7~bit 0

• **UBRG Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------|-------|-------|-------|-------|-------|-------|-------|-------|
| Name | UBRG7 | UBRG6 | UBRG5 | UBRG4 | UBRG3 | UBRG2 | UBRG1 | UBRG0 |
| 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 **UBRG7~UBRG0**: Baud Rate values

By programming the UBRGH bit in UUCR2 Register which allows selection of the related formula described above and programming the required value in the UBRG register, the required baud rate can be setup.

Note: Baud rate= $f_{IH}/[64 \times (N+1)]$ if UBRGH=0.

Baud rate= $f_{IH}/[16 \times (N+1)]$ if UBRGH=1.

Baud Rate Generator

To setup the speed of the serial data communication, the UART function contains its own dedicated baud rate generator. The baud rate is controlled by its own internal free running 8-bit timer, the period of which is determined by two factors. The first of these is the value placed in the baud rate register UBRG and the second is the value of the UBRGH bit with the control register UUCR2. The UBRGH bit decides if the baud rate generator is to be used in a high speed mode or low speed mode, which in turn determines the formula that is used to calculate the baud rate. The value N in the UBRG register which is used in the following baud rate calculation formula determines the division factor. Note that N is the decimal value placed in the UBRG register and has a range of between 0 and 255.

| UUCR2 UBRGH Bit | 0 | 1 |
|-----------------|--------------------|--------------------|
| Baud Rate (BR) | $f_{H}/[64 (N+1)]$ | $f_{H}/[16 (N+1)]$ |

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

Calculating the Baud Rate and Error Values

For a clock frequency of 4MHz, and with UBRGH cleared to zero determine the UBRG register value N, the actual baud rate and the error value for a desired baud rate of 4800.

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

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

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

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

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

UART Setup and Control

For data transfer, the UART function utilizes a non-return-to-zero, more commonly known as NRZ, format. This is composed of one start bit, eight or nine data bits, and one or two stop bits. Parity is supported by the UART hardware, and can be setup to be even, odd or no parity. For the most common data format, 8 data bits along with no parity and one stop bit, denoted as 8, N, 1, is used as the default setting, which is the setting at power-on. The number of data bits and stop bits, along with the parity, are setup by programming the corresponding UBNO, UPRT, UPREN and USTOPS bits in the UUCR1 register. The baud rate used to transmit and receive data is setup using the internal 8-bit baud rate generator, while the data is transmitted and received LSB first. Although the UART transmitter and receiver are functionally independent, they both use the same data format and baud rate. In all cases stop bits will be used for data transmission.

Enabling/Disabling the UART Interface

The basic on/off function of the internal UART function is controlled using the UREN bit in the UUCR1 register. When the UART mode is selected by setting the UMD bit in the SIMC0 register to "1", if the UREN, UTXEN and URXEN bits are set, then these two UART pins will act as normal UTX output pin and URX/UTX input pin respectively. If no data is being transmitted on the UTX pin, then it will default to a logic high value.

Clearing the UREN bit will disable the UTX and URX/UTX 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 UART function is disabled the buffer will be reset to an empty condition, at the same time discarding any remaining residual data. Disabling the UART will also reset the error and status flags with bits UTXEN, URXEN, UTXBRK, URXIF, UOERR, UFERR, UPERR and UNF being cleared while bits UTIDLE, UTXIF and URIDLE will be set. The remaining control bits in the UUCR1, UUCR2, UUCR3 and UBRG registers will remain unaffected. If the UREN bit in the UUCR1 register is cleared while the UART is active, then all pending transmissions and receptions will be immediately suspended and the UART will be reset to a condition as defined above. If the UART is then subsequently re-enabled, it will restart again in the same configuration.

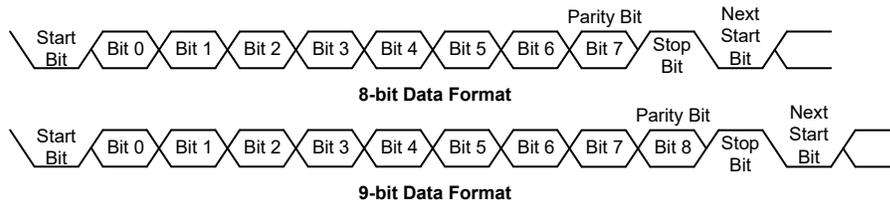
Data, Parity and Stop Bit Selection

The format of the data to be transferred is composed of various factors such as data bit length, parity on/off, parity type, address bits and the number of stop bits. These factors are determined by the setup of various bits within the UUCR1 register. The UBNO bit controls the number of data bits which can be set to either 8 or 9, the UPRT bit controls the choice of odd or even parity, the UPREN bit controls the parity on/off function and the USTOPS 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 |
|--------------------------------------|-----------|-------------|------------|----------|
| Example of 8-bit Data Formats | | | | |
| 1 | 8 | 0 | 0 | 1 |
| 1 | 7 | 0 | 1 | 1 |
| 1 | 7 | 1 | 0 | 1 |
| Example of 9-bit Data Formats | | | | |
| 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 UBNO bit in the UUCR1 register. When UBNO 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 UTX8 bit in the UUCR1 register. At the transmitter core lies the Transmitter Shift Register, more commonly known as the TSR, whose data is obtained from the transmit data register, which is known as the UTXR_RXR register. The data to be transmitted is loaded into this UTXR_RXR register by the application program. The TSR register is not written

to with new data until the stop bit from the previous transmission has been sent out. As soon as this stop bit has been transmitted, the TSR can then be loaded with new data from the UTXR_RXR register, if it is available. It should be noted that the TSR register, unlike many other registers, is not directly mapped into the Data Memory area and as such is not available to the application program for direct read/write operations. An actual transmission of data will normally be enabled when the UTXEN bit is set, but the data will not be transmitted until the UTXR_RXR register has been loaded with data and the baud rate generator has defined a shift clock source. However, the transmission can also be initiated by first loading data into the UTXR_RXR register, after which the UTXEN bit can be set. When a transmission of data begins, the TSR is normally empty, in which case a transfer to the UTXR_RXR register will result in an immediate transfer to the TSR. If during a transmission the UTXEN bit is cleared, the transmission will immediately cease and the transmitter will be reset. The UTX output pin can then be configured as the I/O or other pin-shared function by configuring the corresponding pin-shared control bits.

Transmitting Data

When the UART is transmitting data, the data is shifted on the UTX pin from the shift register, with the least significant bit first. In the transmit mode, the UTXR_RXR register forms a buffer between the internal bus and the transmitter shift register. It should be noted that if 9-bit data format has been selected, then the MSB will be taken from the UTX8 bit in the UUCR1 register. The steps to initiate a data transfer can be summarized as follows:

- Make the correct selection of the UBNO, UPRT, UPREN and USTOPS bits to define the required word length, parity type and number of stop bits.
- Setup the UBRG register to select the desired baud rate.
- Set the UTXEN bit to ensure that the UTX pin is used as a UART transmitter pin.
- Access the UUSR register and write the data that is to be transmitted into the UTXR_RXR register. Note that this step will clear the UTXIF bit.

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

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

1. A UUSR register access
2. A UTXR_RXR register write execution

The read-only UTXIF flag is set by the UART hardware and if set indicates that the UTXR_RXR register is empty and that other data can now be written into the UTXR_RXR register without overwriting the previous data. If the UTEIE bit is set then the UTXIF flag will generate an interrupt.

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

1. A UUSR register access
2. A UTXR_RXR register write execution

Note that both the UTXIF and UTIDLE bits are cleared by the same software sequence.

Transmitting Break

If the UTXBRK bit is set then and the state keeps for a time greater than $[(BRG+1) \times t_{H}]$ while UTIDLE=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 $N=1, 2$, etc. If a break character is to be transmitted then the UTXBRK 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 UTXBRK 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 UTXBRK 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 is capable of receiving word lengths of either 8 or 9 bits. If the UBNO bit is set, the word length will be set to 9 bits with the MSB being stored in the URX8 bit of the UUCR1 register. At the receiver core lies the Receive Serial Shift Register, commonly known as the RSR. The data which is received on the URX/UTX 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 URX/UTX pin is sampled for the stop bit, the received data in RSR is transferred to the receive data register, if the register is empty. The data which is received on the external URX/UTX input pin is sampled three times by a majority detect circuit to determine the logic level that has been placed onto the URX/UTX pin. It should be noted that the RSR register, unlike many other registers, is not directly mapped into the Data Memory area and as such is not available to the application program for direct read/write operations.

Receiving Data

When the UART receiver is receiving data, the data is serially shifted in on the external URX/UTX input pin, LSB first. In the read mode, the UTXR_RXR register forms a buffer between the internal bus and the receiver shift register. The UTXR_RXR register is a two-byte deep FIFO data buffer, where two bytes can be held in the FIFO while a third byte can continue to be received. Note that the application program must ensure that the data is read from UTXR_RXR before the third byte has been completely shifted in, otherwise this third byte will be discarded and an overrun error UOERR will be subsequently indicated. The steps to initiate a data transfer can be summarized as follows:

- Make the correct selection of UBNO, UPRT and UPREN bits to define the word length, parity type.
- Setup the UBRG register to select the desired baud rate.
- Set the URXEN bit to ensure that the URX/UTX pin is used as a UART receiver pin.

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

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

- The URXIF bit in the UUSR register will be set when the UTXR_RXR register has data available. There will be at most one more character available before an overrun error occurs.
- When the contents of the shift register have been transferred to the UTXR_RXR register, then if the URIF bit is set, an interrupt will be generated.
- If during reception, a frame error, noise error, parity error, or an overrun error has been detected, then the error flags can be set.

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

1. A UUSR register access
2. A UTXR_RXR register read execution

Receiving Break

Any break character received by the UART will be managed as a framing error. The receiver will count and expect a certain number of bit times as specified by the values programmed into the UBNO 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 UBNO plus one stop bit. The URXIF bit is set, UFERR is set, zeros are loaded into the receive data register, interrupts are generated if appropriate and the URIDLE bit is set. A break is regarded as a character that contains only zeros with the UFERR 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 UFERR 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 URIDLE read only flag will go high when the stop bits have not yet been received. The reception of a break character on the UART registers will result in the following:

- The framing error flag, UFERR, will be set.
- The receive data register, UTXR_RXR, will be cleared.
- The UOERR, UNF, UPERR, URIDLE or URXIF 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 UUSR register, otherwise known as the URIDLE flag, will have a zero value. In between the reception of a stop bit and the detection of the next start bit, the URIDLE flag will have a high value, which indicates the receiver is in an idle condition.

Receiver Interrupt

The read only receive interrupt flag URXIF in the UUSR register is set by an edge generated by the receiver. An interrupt is generated if URIE=1, when a word is transferred from the Receive Shift Register, RSR, to the Receive Data Register, UTXR_RXR. An overrun error can also generate an interrupt if URIE=1.

Managing Receiver Errors

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

Overrun Error – UOERR

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

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

- The UOERR flag in the UUSR register will be set.
- The UTXR_RXR contents will not be lost.
- The shift register will be overwritten.
- An interrupt will be generated if the URIE bit is set.

The UOERR flag can be cleared by an access to the UUSR register followed by a read to the UTXR_RXR register.

Noise Error – UNF

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, UNF, in the UUSR register will be set on the rising edge of the URXIF bit.
- Data will be transferred from the Shift register to the UTXR_RXR register.
- No interrupt will be generated. However this bit rises at the same time as the URXIF bit which itself generates an interrupt.

Note that the UNF flag is reset by a UUSR register read operation followed by a UTXR_RXR register read operation.

Framing Error – UFERR

The read only framing error flag, UFERR, in the UUSR 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 UFERR flag will be set. The UFERR flag and the received data will be recorded in the UUSR and UTXR_RXR registers respectively, and the flag is cleared in any reset.

Parity Error – UPERR

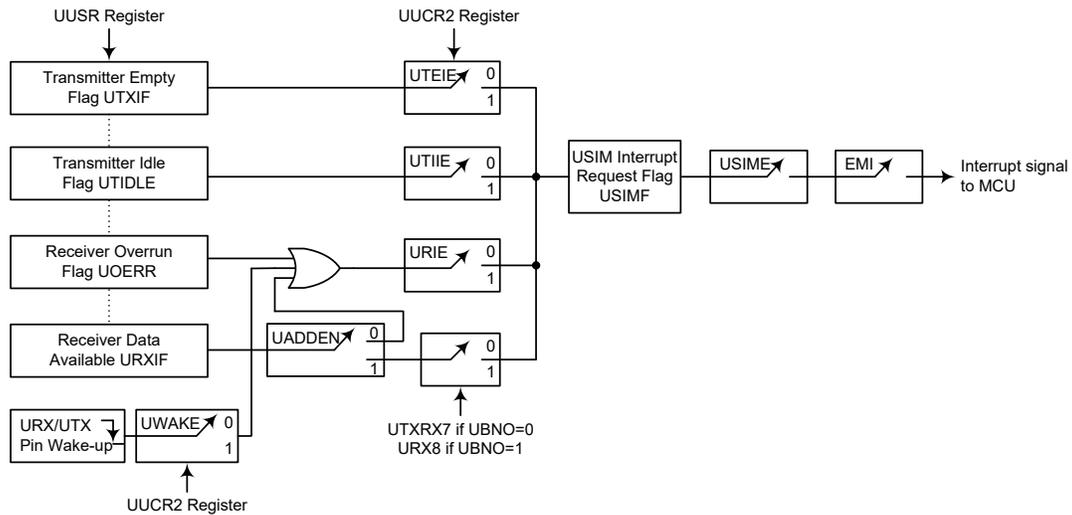
The read only parity error flag, UPERR, in the UUSR register, is set if the parity of the received word is incorrect. This error flag is only applicable if the parity is enabled, UPREN=1, and if the parity type, odd or even is selected. The read only UPERR flag and the received data will be recorded in the UUSR and UTXR_RXR registers respectively. It is cleared on any reset, it should be noted that the flags, UFERR and UPERR, in the UUSR register should first be read by the application program before reading the data word.

UART Interrupt Structure

Several individual UART conditions can trigger an USIM interrupt. When these conditions exist, a low pulse will be generated to get the attention of the microcontroller. These conditions are a transmitter data register empty, transmitter idle, receiver data available, receiver overrun, address detect and an URX/UTX pin wake-up. When any of these conditions are created, if the global interrupt enable bit and the USIM 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 UUSR register flags which will generate an USIM interrupt if its associated interrupt enable control bit in the UUCR2 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 USIM UART mode interrupt sources.

The address detect condition, which is also an USIM UART mode interrupt source, does not have an associated flag, but will generate an USIM interrupt when an address detect condition occurs if its function is enabled by setting the UADDEN bit in the UUCR2 register. An URX/UTX pin wake-up, which is also an USIM UART mode interrupt source, does not have an associated flag, but will generate an USIM interrupt if the UART clock (f_H) source is switched off and the UWAKE and URIE bits in the UUCR2 register are set when a falling edge on the URX/UTX pin occurs. Note that in the event of an URX/UTX wake-up interrupt occurring, there will be a certain period of delay, commonly known as the System Start-up Time, for the oscillator to restart and stabilize before the system resumes normal operation.

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



UART Interrupt Structure

Address Detect Mode

Setting the Address Detect Mode bit, UADDEEN, in the UUCR2 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 URXIF flag. If the UADDEEN 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 USIME and EMI interrupt enable bits must also be enabled for correct interrupt generation. This highest address bit is the 9th bit if UBNO=1 or the 8th bit if UBNO=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 UADDEEN bit is not enabled, then a Receiver Data Available interrupt will be generated each time the URXIF 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 UPREN to zero.

| UADDEEN | 9th bit if UBNO=1, 8th bit if UBNO=0 | USIM Interrupt Generated |
|---------|---|-----------------------------|
| 0 | 0 | √ |
| | 1 | √ |
| 1 | 0 | × |
| | 1 | √ |

UADDEEN Bit Function

UART Power Down and Wake-up

When the UART clock (f_{H}) is off, the UART will cease to function, all clock sources to the module are shutdown. If the UART clock (f_{H}) is off while a transmission is still in progress, then the transmission will be paused until the UART clock source derived from the microcontroller is activated. In a similar way, if the MCU 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 UUSR, UUCR1, UUCR2, UUCR3, transmit and receive registers, as well as the UBRG register will not be affected. It is recommended to make sure first that the UART data transmission or reception has been finished before the microcontroller enters the IDLE or SLEEP mode.

The UART function contains a receiver URX/UTX pin wake-up function, which is enabled or disabled by the UWAKE bit in the UUCR2 register. If this bit, along with the UART mode selection bit, UMD, the UART enable bit, UREN, the receiver enable bit, URXEN and the receiver interrupt bit, URIE, are all set when the UART clock (f_{H}) is off, then a falling edge on the URX/UTX pin will trigger an URX/UTX pin wake-up UART interrupt. Note that as it takes certain system clock cycles after a wake-up, before normal microcontroller operation resumes, any data received during this time on the URX/UTX pin will be ignored.

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

UART Interface

The device contains an independent UART function. It is important not to confuse this independent UART function with the additional one contained within the combined USIM function, which is described in another section of this datasheet.

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

The integrated UART function contains the following features:

- Full-duplex or half-duplex (single wire mode), asynchronous communication
- 8 or 9 bits character length
- Even, odd, mark, space or no parity options
- One or two stop bits configurable for receiver
- Two stop bits for transmitter
- Baud rate generator with 16-bit prescaler
- Parity, framing, noise and overrun error detection
- Support for interrupt on address detect (last character bit=1)
- Separately enabled transmitter and receiver
- 4-byte Deep FIFO Receive Data Buffer
- 1-byte Deep FIFO Transmit Data Buffer

UART Single Wire Mode

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

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

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

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

UART Data Transfer Scheme

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

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

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

UART Status and Control Registers

There are nine control registers associated with the UART function. The SWM bit in the UCR3 register is used to enable/disable the UART Single Wire Mode. The USR, UCR1, UCR2, UFCR and RxCNT registers control the overall function of the UART, while the BRDH and BRDL registers control the Baud rate. The actual data to be transmitted and received on the serial interface is managed through the TXR_RXR data register.

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

| Register Name | Bit | | | | | | | |
|---------------|-----|----|-------|-------|-------|------|--------|--------|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| BRDL | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| UFCR | — | — | UMOD2 | UMOD1 | UMOD0 | BRDS | RxFTR1 | RxFTR0 |
| RxCNT | — | — | — | — | — | D2 | D1 | D0 |

UART Register List

• **USR Register**

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

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

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

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

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

The NF flag is the noise flag. When this read only flag is “0”, it indicates no noise condition. When the flag is “1”, it indicates that the UART has detected noise on the receiver input. The NF flag is set during the same cycle as the RXIF flag but will not be set in the case of an overrun. The NF flag can be cleared by a software sequence which will involve a read to the status register USR followed by an access to the TXR_RXR data register.

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

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

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

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

- Bit 3 RIDLE:** Receiver status
 0: Data reception is in progress (Data being received)
 1: No data reception is in progress (Receiver is idle)
 The RIDLE flag is the receiver status flag. When this read only flag is “0”, it indicates that the receiver is between the initial detection of the start bit and the completion of the stop bit. When the flag is “1”, it indicates that the receiver is idle. Between the completion of the stop bit and the detection of the next start bit, the RIDLE bit is “1” indicating that the UART receiver is idle and the RX/TX pin stays in logic high condition.
- Bit 2 RXIF:** Receive TXR_RXR data register status
 0: TXR_RXR data register is empty
 1: TXR_RXR data register has available data and reach Receiver FIFO trigger level
 The RXIF flag is the receive data register status flag. When this read only flag is “0”, it indicates that the TXR_RXR read data register is empty. When the flag is “1”, it indicates that the TXR_RXR read data register contains new data. When the contents of the shift register are transferred to the TXR_RXR register and reach Receiver FIFO trigger level, an interrupt is generated if RIE=1 in the UCR2 register. If one or more errors are detected in the received word, the appropriate receive-related flags NF, FERR, and/or PERR are set within the same clock cycle. The RXIF flag is cleared when the USR register is read with RXIF set, followed by a read from the TXR_RXR register, and if the TXR_RXR register has no data available.
- Bit 1 TIDLE:** Transmission idle
 0: Data transmission is in progress (Data being transmitted)
 1: No data transmission is in progress (Transmitter is idle)
 The TIDLE flag is known as the transmission complete flag. When this read only flag is “0”, it indicates that a transmission is in progress. This flag will be set high when the TXIF flag is “1” and when there is no transmit data or break character being transmitted. When TIDLE is equal to “1”, the TX pin becomes idle with the pin state in logic high condition. The TIDLE flag is cleared by reading the USR register with TIDLE set and then writing to the TXR_RXR register. The flag is not generated when a data character or a break is queued and ready to be sent.
- Bit 0 TXIF:** Transmit TXR_RXR data register status
 0: Character is not transferred to the transmit shift register
 1: Character has transferred to the transmit shift register (TXR_RXR data register is empty)
 The TXIF flag is the transmit data register empty flag. When this read only flag is “0”, it indicates that the character is not transferred to the transmitter shift register. When the flag is “1”, it indicates that the transmitter shift register has received a character from the TXR_RXR data register. The TXIF flag is cleared by reading the UART status register (USR) with TXIF set and then writing to the TXR_RXR data register. Note that when the TXEN bit is set, the TXIF flag will also be set since the transmit data register is not yet full.

• **UCR1 Register**

The UCR1 register together with the UCR2 and UCR3 register are the three UART control registers that are used to set the various options for the UART function, such as overall on/off control, parity control, data transfer bit length, 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 | UARTEN | BNO | PREN | PRT1 | PRT0 | TXBRK | RX8 | TX8 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | W |
| POR | 0 | 0 | 0 | 0 | 0 | 0 | x | 0 |

“x”: unknown

- Bit 7 **UARTEN**: UART function enable control
 0: Disable UART. TX and RX/TX pins are in a floating state
 1: Enable UART. TX and RX/TX pins function as UART pins
 The UARTEN bit is the UART enable bit. When this bit is equal to “0”, the UART will be disabled and the RX/TX pin as well as the TX pin will be set in a floating state. When the bit is equal to “1”, the UART will be enabled and the TX and RX/TX pins will function as defined by the SWM mode selection bit together with the TXEN and RXEN enable control bits.
 When the UART is disabled, it will empty the buffer so any character remaining in the buffer will be discarded. In addition, the value of the baud rate counter will be reset. If the UART is disabled, all error and status flags will be reset. Also the TXEN, RXEN, TXBRK, RXIF, OERR, FERR, PERR and NF bits as well as the RxCNT register will be cleared, while the TIDLE, TXIF and RIDLE bits will be set. Other control bits in UCR1, UCR2, UCR3, UFCR, BRDH and BRDL registers will remain unaffected. If the UART is active and the UARTEN bit is cleared, all pending transmissions and receptions will be terminated and the module will be reset as defined above. When the UART is re-enabled, it will restart in the same configuration.
- Bit 6 **BNO**: Number of data transfer bits selection
 0: 8-bit data transfer
 1: 9-bit data transfer
 This bit is used to select the data length format, which can have a choice of either 8-bit or 9-bit format. When this bit is equal to “1”, a 9-bit data length format will be selected. If the bit is equal to “0”, then an 8-bit data length format will be selected. If 9-bit data length format is selected, then bits RX8 and TX8 will be used to store the 9th bit of the received and transmitted data respectively.
 Note that the 9th bit of data if BNO=1, or the 8th bit of data if BNO=0, which is used as the parity bit, does not transfer to RX8 or TXRX7 respectively when the parity function is enabled.
- Bit 5 **PREN**: Parity function enable control
 0: Parity function is disabled
 1: Parity function is enabled
 This is the parity enable bit. When this bit is equal to “1”, the parity function will be enabled. If the bit is equal to “0”, then the parity function will be disabled. Replace the most significant bit position with a parity bit.
- Bit 4~3 **PRT1~PRT0**: Parity type selection bits
 00: Even parity for parity generator
 01: Odd parity for parity generator
 10: Mark parity for parity generator
 11: Space parity for parity generator
 These bits are the parity type selection bits. When these bits are equal to 00b, even parity type will be selected. If these bits are equal to 01b, then odd parity type will be selected. If these bits are equal to 10b, then a 1 (Mark) in the parity bit location will be selected. If these bits are equal to 11b, then a 0 (Space) in the parity bit location will be selected.
- Bit 2 **TXBRK**: Transmit break character
 0: No break character is transmitted
 1: Break characters transmit
 The TXBRK bit is the Transmit Break Character bit. When this bit is “0”, there are no break characters and the TX pin operates normally. When the bit is “1”, there are transmit break characters and the transmitter will send logic zeros. When this bit is equal to “1”, after the buffered data has been transmitted, the transmitter output is held low for a minimum of a 13-bit length and until the TXBRK bit is reset.
- Bit 1 **RX8**: Receive data bit 8 for 9-bit data transfer format (read only)
 This bit is only used if 9-bit data transfers are used, in which case this bit location will store the 9th bit of the received data known as RX8. The BNO bit is used to determine whether data transfers are in 8-bit or 9-bit format.

Bit 0 **TX8**: Transmit data bit 8 for 9-bit data transfer format (write only)
 This bit is only used if 9-bit data transfers are used, in which case this bit location will store the 9th bit of the transmitted data known as TX8. The BNO bit is used to determine whether data transfers are in 8-bit or 9-bit format.

• **UCR2 Register**

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

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

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

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

Bit 5 **STOPS**: Number of Stop bits selection for receiver
 0: One stop bit format is used
 1: Two stop bits format is used
 This bit determines if one or two stop bits are to be used for receiver. When this bit is equal to “1”, two stop bits are used. If this bit is equal to “0”, then only one stop bit is used. Two stop bits are used for transmitter.

Bit 4 **ADDEN**: Address detect function enable control
 0: Address detect function is disabled
 1: Address detect function is enabled
 The bit named ADDEN is the address detect function enable control bit. When this bit is equal to “1”, the address detect function is enabled. When it occurs, if the 8th bit, which corresponds to TXRX7 if BNO=0 or the 9th bit, which corresponds to RX8 if BNO=1, has a value of “1”, then the received word will be identified as an address, rather than data. If the corresponding interrupt is enabled, an interrupt request will be generated each time the received word has the address bit set, which is the 8th or 9th bit depending on the value of BNO. If the address bit known as the 8th or 9th bit of the

received word is “0” with the address detect function being enabled, an interrupt will not be generated and the received data will be discarded.

- Bit 3 WAKE:** RX/TX pin wake-up UART function enable control
 0: RX/TX pin wake-up UART function is disabled
 1: RX/TX pin wake-up UART function is enabled
 This bit is used to control the wake-up UART function when a falling edge on the RX/TX pin occurs. Note that this bit is only available when the UART clock (f_H) is switched off. There will be no RX/TX pin wake-up UART function if the UART clock (f_H) exists. If the WAKE bit is set to 1 as the UART clock (f_H) is switched off, a UART wake-up request will be initiated when a falling edge on the RX/TX pin occurs. When this request happens and the corresponding interrupt is enabled, an RX/TX pin wake-up UART interrupt will be generated to inform the MCU to wake up the UART function by switching on the UART clock (f_H) via the application program. Otherwise, the UART function cannot resume even if there is a falling edge on the RX/TX pin when the WAKE bit is cleared to 0.
- Bit 2 RIE:** Receiver interrupt enable control
 0: Receiver related interrupt is disabled
 1: Receiver related interrupt is enabled
 This bit enables or disables the receiver interrupt. If this bit is equal to “1” and when the receiver overrun flag OERR or receive data available flag RXIF is set, the UART interrupt request flag will be set. If this bit is equal to “0”, the UART interrupt request flag will not be influenced by the condition of the OERR or RXIF flags.
- Bit 1 TIE:** Transmitter Idle interrupt enable control
 0: Transmitter idle interrupt is disabled
 1: Transmitter idle interrupt is enabled
 This bit enables or disables the transmitter idle interrupt. If this bit is equal to “1” and when the transmitter idle flag TIDLE is set, due to a transmitter idle condition, the UART interrupt request flag will be set. If this bit is equal to “0”, the UART interrupt request flag will not be influenced by the condition of the TIDLE flag.
- Bit 0 TEIE:** Transmitter Empty interrupt enable control
 0: Transmitter empty interrupt is disabled
 1: Transmitter empty interrupt is enabled
 This bit enables or disables the transmitter empty interrupt. If this bit is equal to “1” and when the transmitter empty flag TXIF is set, due to a transmitter empty condition, the UART interrupt request flag will be set. If this bit is equal to “0”, the UART interrupt request flag will not be influenced by the condition of the TXIF flag.

• **UCR3 Register**

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

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

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

• **TXR_RXR Register**

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

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

“x”: unknown

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

• **BRDH Register**

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

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

The baud rate divider BRD (BRDH/BRDL) defines the UART clock divider ratio.

$$\text{Baud Rate} = f_{ih} / (\text{BRD} + \text{UMOD} / 8)$$

$$\text{BRD} = 16 \sim 65535 \text{ or } 8 \sim 65535 \text{ depending on BRDS}$$

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

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

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

• **BRDL Register**

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

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

The baud rate divider BRD (BRDH/BRDL) defines the UART clock divider ratio.

$$\text{Baud Rate} = f_{ih} / (\text{BRD} + \text{UMOD} / 8)$$

$$\text{BRD} = 16 \sim 65535 \text{ or } 8 \sim 65535 \text{ depending on BRDS}$$

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

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

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

• **UFCR Register**

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

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

Bit 7~6 Unimplemented, read as “0”

- Bit 5~3 **UMOD2~UMOD0:** UART modulation control bits
 The modulation control bits are used to correct the baud rate of the received or transmitted UART signal. These bits determine if the extra UART clock cycle should be added in a UART bit time. The UMOD2~UMOD0 will be added to internal accumulator for every UART bit time. Until a carry to bit 3, the corresponding UART bit time increases a UART clock cycle.
- Bit 2 **BRDS:** BRD range selection
 0: BRD range is from 16 to 65535
 1: BRD range is from 8 to 65535
 The BRDS is used to control the sampling point in a UART bit time. If the BRDS bit is cleared to zero, the sampling point will be BRD/2, BRD/2+1×f_H, and BRD/2+2×f_H in a UART bit time. If the BRDS bit is set high, the sampling point will be BRD/2-1×f_H, BRD/2, and BRD/2+2×f_H in a UART bit time.
 Note that the BRDS bit should not be modified during data transmission process.
- Bit 1~0 **RxFTR1~RxFTR0:** Receiver FIFO trigger level (bytes)
 00: 4 bytes in 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 RXIF bit being set high, an interrupt will also be generated if the RIE bit is enabled. To prevent OERR from being set high, the receiver FIFO trigger level can be set to 2 bytes, avoiding an overrun state that cannot be processed by the program in time when more than 4 data bytes are received. After the reset the Receiver FIFO is empty.

• **RxCNT Register**

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

Baud Rate Generator

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

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

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

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

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

Therefore, the actual baud rate is calculated as follows:

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

Calculating the Baud Rate and Error Values

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

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

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

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

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

Modulation Control Example

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

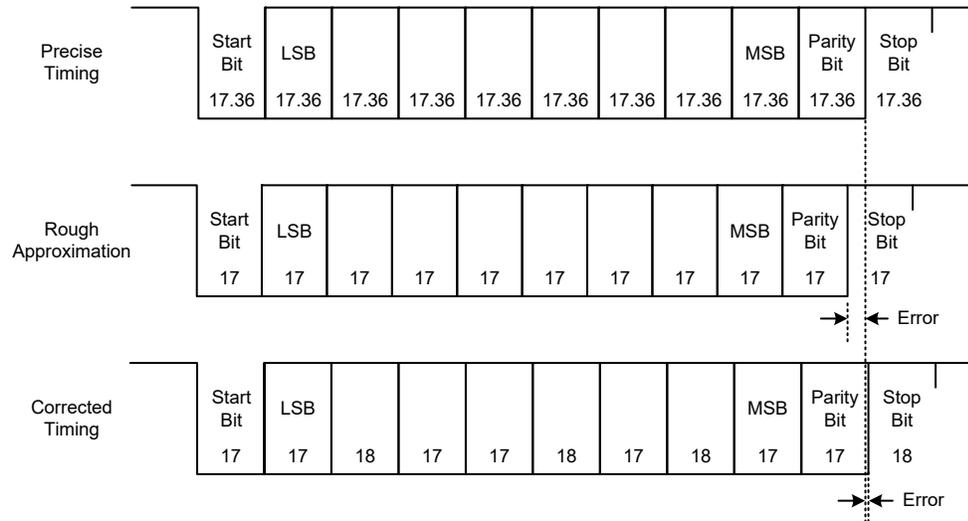
| Fraction Addition | Carry to Bit 3 | UART Bit Time Sequence | Extra UART Clock Cycle |
|-------------------|----------------|------------------------|------------------------|
| 0000b+0011b=0011b | No | Start bit | No |
| 0011b+0011b=0110b | No | D0 | No |
| 0110b+0011b=1001b | Yes | D1 | Yes |
| 1001b+0011b=1100b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

Baud Rate Correction Example

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

The following figure shows three different frames:

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



UART Setup and Control

For data transfer, the UART function utilizes a non-return-to-zero, more commonly known as NRZ, format. This is composed of one start bit, eight or nine data bits, and one or two stop bits. Parity is supported by the UART hardware, and can be setup to be even, odd, mark, space or no parity. For the most common data format, 8 data bits along with no parity and one stop bit, denoted as 8, N, 1, is used as the default setting, which is the setting at power-on. The number of data bits along with the parity are setup by programming the BNO, PRT1~PRT0 and PREN bits. The transmitter always uses two stop bits while the receiver uses one or two stop bits which is determined by the STOPS bit. The baud rate used to transmit and receive data is setup using the internal 16-bit baud rate generator, while the data is transmitted and received LSB first. Although the UART transmitter and receiver are functionally independent, they both use the same data format and baud rate. In all cases stop bits will be used for data transmission.

Enabling/Disabling the UART Interface

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

Clearing the UARTEN bit will disable the TX and RX/TX pins and allow these two pins to be used as normal I/O or other pin-shared functional pins by configuring the corresponding pin-shared control bits. When the UART function is disabled the buffer will be reset to an empty condition, at the same time discarding any remaining residual data. Disabling the UART will also reset the error and status flags with bits TXEN, RXEN, TXBRK, RXIF, OERR, FERR, PERR and NF as well as register RxCNT being cleared while bits TIDLE, TXIF and RIDLE will be set. The remaining control bits in the UCR1, UCR2, UCR3, UFCR, BRDH and BRDL registers will remain unaffected. If the UARTEN bit in the UCR1 register is cleared while the UART is active, then all pending transmissions and receptions will be immediately suspended and the UART will be reset to a condition as defined above. If the UART is then subsequently re-enabled, it will restart again in the same configuration.

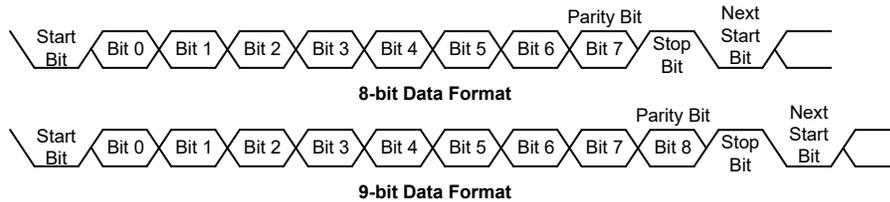
Data, Parity and Stop Bit Selection

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

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

Transmitter Receiver Data Format

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



UART Transmitter

Data word lengths of either 8 or 9 bits can be selected by programming the BNO bit in the UCR1 register. When BNO bit is set, the word length will be set to 9 bits. In this case the 9th bit, which is the MSB, needs to be stored in the TX8 bit in the UCR1 register. At the transmitter core lies the Transmitter Shift Register, more commonly known as the TSR, whose data is obtained from the transmit data register, which is known as the TXR_RXR register. The data to be transmitted is loaded into this TXR_RXR register by the application program. The TSR register is not written to with new data until the stop bit from the previous transmission has been sent out. As soon as this stop bit has been transmitted, the TSR can then be loaded with new data from the TXR_RXR register, if it is available. It should be noted that the TSR register, unlike many other registers, is not directly mapped into the Data Memory area and as such is not available to the application program for direct read/write operations. An actual transmission of data will normally be enabled when the TXEN bit is set, but the data will not be transmitted until the TXR_RXR register has been loaded with data and the baud rate generator has defined a shift clock source. However, the transmission can also be initiated by first loading data into the TXR_RXR register, after which the TXEN bit can be set. When a transmission of data begins, the TSR is normally empty, in which case a transfer to the TXR_RXR register will result in an immediate transfer to the TSR. If during a transmission the TXEN bit is cleared, the transmission will immediately cease and the transmitter will be reset. The

TX output pin can then be configured as the I/O or other pin-shared functions by configuring the corresponding pin-shared control bits.

Transmitting Data

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

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

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

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

1. A USR register access
2. A TXR_RXR register write execution

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

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

1. A USR register access
2. A TXR_RXR register write execution

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

Transmitting Break

If the TXBRK bit is set and the state keeps for a time greater than $(BRD+1) \times t_{TH}$ while TIDLE=1, then break characters will be sent on the next transmission. Break character transmission consists of a start bit, followed by $13 \times N$ '0' bits and stop bits, where $N=1, 2, \text{ etc.}$ If a break character is to be transmitted then the TXBRK bit must be first set by the application program, and then cleared to generate the stop bits. Transmitting a break character will not generate a transmit interrupt. Note that a break condition length is at least 13 bits long. If the TXBRK bit is continually kept at a logic high level then the transmitter circuitry will transmit continuous break characters. After the application program has cleared the TXBRK bit, the transmitter will finish transmitting the last break character and subsequently send out two stop bits. The automatic logic highs at the end of the last break character will ensure that the start bit of the next frame is recognized.

UART Receiver

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

Receiving Data

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

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

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

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

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

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

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

1. A USR register access
2. A TXR_RXR register read execution

Receiving Break

Any break character received by the UART will be managed as a framing error. The receiver will count and expect a certain number of bit times as specified by the values programmed into the BNO plus one or two stop bits. If the break is much longer than 13 bit times, the reception will be considered as complete after the number of bit times specified by BNO plus one or two stop

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

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

Idle Status

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

Receiver Interrupt

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

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

Managing Receiver Errors

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

Overrun Error – OERR

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

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

- The OERR flag in the USR register will be set.
- The TXR_RXR contents will not be lost.
- The shift register will be overwritten.
- An interrupt will be generated if the RIE bit is set.

When the OERR flag is set to “1”, it is necessary to read five data bytes from the four-byte deep receiver FIFO and the shift register immediately to avoid unexpected errors, such as the UART is

unable to receive data. If such an error occurs, clear the RXEN bit to “0” then set it to “1” again to continue data reception.

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

Noise Error – NF

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

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

Note that the NF flag is reset by a USR register read operation followed by a TXR_RXR register read operation.

Framing Error – FERR

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

Parity Error – PERR

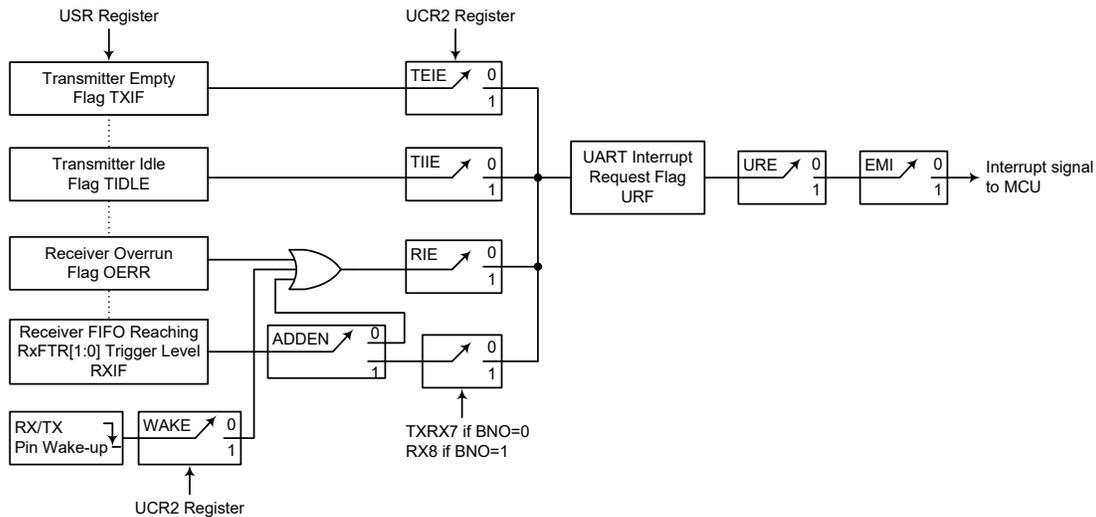
The read only parity error flag, PERR, in the USR register, is set if the parity of the received word is incorrect. This error flag is only applicable if the parity is enabled, PREN=1, and if the parity type, odd, even, mark or space, is selected. The read only PERR flag and the received data will be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

UART Interrupt Structure

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

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

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



UART Interrupt Structure

Address Detect Mode

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

| ADDEN | 9th Bit if BNO=1, 8th Bit if BNO=0 | UART Interrupt Generated |
|-------|---------------------------------------|-----------------------------|
| 0 | 0 | √ |
| | 1 | √ |
| 1 | 0 | × |
| | 1 | √ |

ADDEN Bit Function

UART Power Down and Wake-up

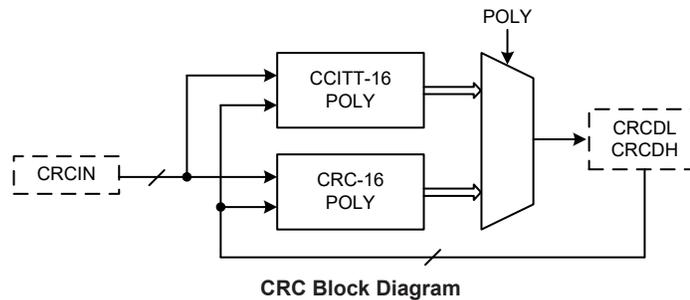
When the UART clock (f_{H}) is off, the UART will cease to function, all clock sources to the module are shutdown. If the UART clock (f_{H}) is off while a transmission is still in progress, then the transmission will be paused until the UART clock source derived from the microcontroller is activated. In a similar way, if the MCU 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 USR, UCR1, UCR2, UCR3, UFCR, RxCNT, TXR_RXR as well as the BRDH and BRDL registers will not be affected. It is recommended to make sure first that the UART data transmission or reception has been finished before the microcontroller enters the IDLE or SLEEP mode.

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

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

Cyclic Redundancy Check – CRC

The Cyclic Redundancy Check, CRC, calculation unit is an error detection technique test algorithm and uses to verify data transmission or storage data correctness. A CRC calculation takes a data stream or a block of data as input and generates a 16-bit output remainder. Ordinarily, a data stream is suffixed by a CRC code and used as a checksum when being sent or stored. Therefore, the received or restored data stream is calculated by the same generator polynomial as described in the following section.



CRC Registers

The CRC generator contains an 8-bit CRC data input register, CRCIN, and a CRC checksum register pair, CRCDH and CRCDL. The CRCIN register is used to input new data and the CRCDH and CRCDL registers are used to hold the previous CRC calculation result. A CRC control register, CRCCR, is used to select which CRC generating polynomial is used.

| Register Name | Bit | | | | | | | |
|---------------|-----|-----|-----|-----|-----|-----|----|------|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CRCCR | — | — | — | — | — | — | — | POLY |
| CRCIN | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| CRCDL | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| CRCDH | D15 | D14 | D13 | D12 | D11 | D10 | D9 | D8 |

CRC Register List

• **CRCCR Register**

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

Bit 7~1 Unimplemented, read as “0”
 Bit 0 **POLY**: 16-bit CRC generating polynomial selection
 0: CRC-CCITT: $X^{16}+X^{12}+X^5+1$
 1: CRC-16: $X^{16}+X^{15}+X^2+1$

• **CRCIN 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**: CRC input data register

• **CRCDL 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**: 16-bit CRC checksum low byte data register

• **CRCDH 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**: 16-bit CRC checksum high byte data register

CRC Operation

The CRC generator provides the 16-bit CRC result calculation based on the CRC16 and CCITT CRC16 polynomials. In this CRC generator, there are only these two polynomials available for the numeric values calculation. It cannot support the 16-bit CRC calculations based on any other polynomials.

The following two expressions can be used for the CRC generating polynomial which is determined using the POLY bit in the CRC control register, CRCCR. The CRC calculation result is called as the CRC checksum, CRCSUM, and stored in the CRC checksum register pair, CRCDH and CRCDL.

- CRC-CCITT: $X^{16}+X^{12}+X^5+1$.
- CRC-16: $X^{16}+X^{15}+X^2+1$.

CRC Computation

Each write operation to the CRCIN register creates a combination of the previous CRC value stored in the CRCDH and CRCDL registers and the new data input. The CRC unit calculates the CRC data register value is based on byte by byte. It will take one MCU instruction cycle to calculate the CRC checksum.

CRC Calculation Procedures:

1. Clear the checksum register pair, CRCDH and CRCDL.
2. Execute an “Exclusive OR” operation with the 8-bit input data byte and the 16-bit CRCSUM high byte. The result is called the temporary CRCSUM.
3. Shift the temporary CRCSUM value left by one bit and move a “0” into the LSB.
4. Check the shifted temporary CRCSUM value after procedure 3.

If the MSB is 0, then this shifted temporary CRCSUM will be considered as a new temporary CRCSUM.

Otherwise, execute an “Exclusive OR” operation with the shifted temporary CRCSUM in procedure 3 and a data “8005H”. Then the operation result will be regarded as the new temporary CRCSUM.

Note that the data to be perform an “Exclusive OR” operation is “8005H” for the CRC-16 polynomial while for the CRC-CCITT polynomial the data is “1021H”.

5. Repeat the procedure 3 ~ procedure 4 until all bits of the input data byte are completely calculated.
6. Repeat the procedure 2 ~ procedure 5 until all of the input data bytes are completely calculated. Then, the latest calculated result is the final CRC checksum, CRCSUM.

CRC Calculation Examples:

- Write 1 byte input data into the CRCIN register and the corresponding CRC checksum are individually calculated as the following table shown.

| CRC Data Input | 00H | 01H | 02H | 03H | 04H | 05H | 06H | 07H |
|-------------------------------------|-------|-------|-------|-------|-------|-------|-------|-------|
| CRC-CCITT ($X^{16}+X^{12}+X^5+1$) | 0000H | 1021H | 2042H | 3063H | 4084H | 50A5H | 60C6H | 70E7H |
| CRC-16 ($X^{16}+X^{15}+X^2+1$) | 0000H | 8005H | 800FH | 000AH | 801BH | 001EH | 0014H | 8011H |

Note: The initial value of the CRC checksum register pair, CRCDH and CRCDL, is zero before each CRC input data is written into the CRCIN register.

- Write 4 bytes input data into the CRCIN register sequentially and the CRC checksum are sequentially listed in the following table.

| CRC Data Input | CRCIN=78h→56h→34h→12h |
|-------------------------------------|--|
| CRC-CCITT ($X^{16}+X^{12}+X^5+1$) | (CRCDH, CRCDL)=FF9FH→BBC3H→A367H→D0FAH |
| CRC-16 ($X^{16}+X^{15}+X^2+1$) | (CRCDH, CRCDL)=0110h→91F1h→F2DEh→5C43h |

Note: The initial value of the CRC checksum register pair, CRCDH and CRCDL, is zero before the sequential CRC data input operation.

Program Memory CRC Checksum Calculation Example:

1. Clear the checksum register pair, CRCDH and CRCDL.
2. Select the CRC-CCITT or CRC-16 polynomial as the generating polynomial using the POLY bit in the CRCCR register.
3. Execute the table read instruction to read the program memory data value.
4. Write the table data low byte into the CRCIN register and execute the CRC calculation with the current CRCSUM value. Then a new CRCSUM result will be obtained and stored in the CRC checksum register pair, CRCDH and CRCDL.
5. Write the table data high byte into the CRCIN register and execute the CRC calculation with the current CRCSUM value. Then a new CRCSUM result will be obtained and stored in the CRC checksum register pair, CRCDH and CRCDL.
6. Repeat the procedure 3 ~ procedure 5 to read the next program memory data value and execute the CRC calculation until all program memory data are read followed by the sequential CRC calculation. Then the value in the CRC checksum register pair is the final CRC calculation result.

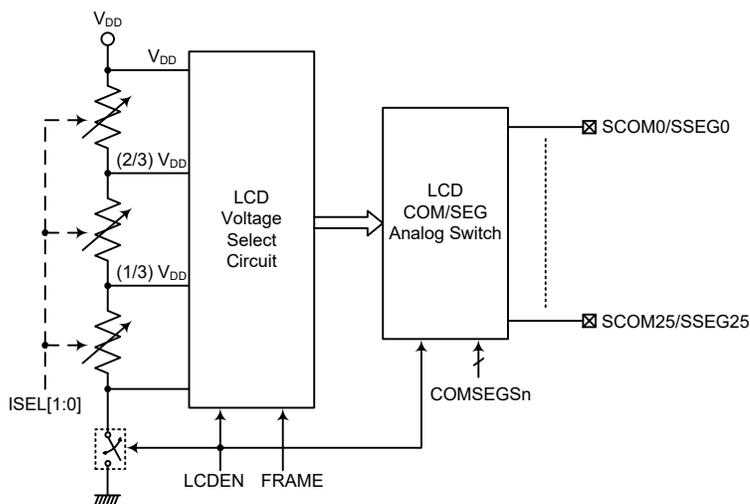
Software Controlled LCD Driver

The device has the capability of driving external LCD panels. The common pins, SCOM0~SCOM25 and segment pins, SSEG0~SSEG25 for LCD driving 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 these devices 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_{SS} , $(1/3) V_{DD}$, $(2/3) V_{DD}$ and V_{DD} 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 $(1/3) \times V_{DD}$. The SEG signal output can have a value of V_{SS} or a V_{BIAS} value of $(2/3) \times V_{DD}$.

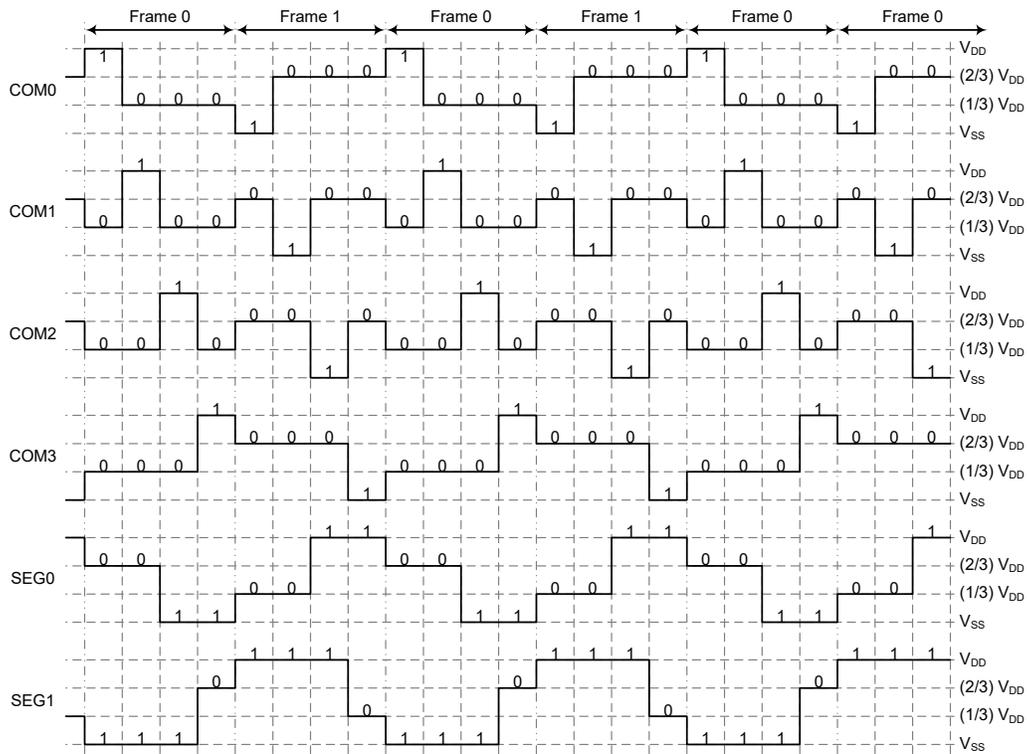
Frame 1

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 $(2/3) \times V_{DD}$. The SEG signal output can have a value of V_{DD} or a V_{BIAS} value of $(1/3) \times V_{DD}$.

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~SCOM25, 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 SCOMm and SSEGn pins using the corresponding pin-shared control bits and pin function selection bits.

| 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 | — | — | — | — | — | — | COMSEGS25 | COMSEGS24 |

LCD Register 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**: SCOMm/SSEGn Output Frame selection

- 0: Frame 0
- 1: Frame 1

Bit 6~5 **ISEL1~ISEL0**: Select resistor for R type LCD bias current

- 00: $3 \times 200k\Omega$ (1/3 Bias), $I_{BIAS}=8.3\mu A$ ($V_{DD}=5V$)
- 01: $3 \times 100k\Omega$ (1/3 Bias), $I_{BIAS}=16.6\mu A$ ($V_{DD}=5V$)
- 10: $3 \times 33.3k\Omega$ (1/3 Bias), $I_{BIAS}=50\mu A$ ($V_{DD}=5V$)
- 11: $3 \times 16.6k\Omega$ (1/3 Bias), $I_{BIAS}=100\mu A$ ($V_{DD}=5V$)

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 | — | — | — | — | — | — | COMSEGS25 | COMSEGS24 |
| R/W | — | — | — | — | — | — | R/W | R/W |
| POR | — | — | — | — | — | — | 0 | 0 |

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

Each 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 | TLVD1 | TLVD0 | LVDO | LVDEN | — | VLVD2 | VLVD1 | VLVD0 |
| R/W | R/W | R/W | R | R/W | — | R/W | R/W | R/W |
| POR | 0 | 0 | 0 | 0 | — | 0 | 0 | 0 |

Bit 7~6 **TLVD1~TLVD0**: Minimum low voltage width to interrupt time (t_{LVD}) selection

00: $(1\sim 2) \times t_{LIRC}$

01: $(3\sim 4) \times t_{LIRC}$

10: $(7\sim 8) \times t_{LIRC}$

11: $(1\sim 2) \times t_{LIRC}$

Bit 5 **LVDO**: LVD Output Flag

0: No Low Voltage Detected

1: Low Voltage Detected

Bit 4 **LVDEN**: Low Voltage Detector 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

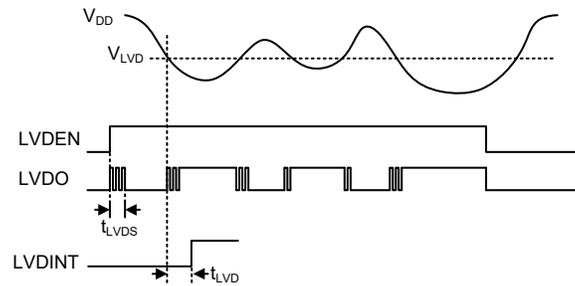
101: 3.3V

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.



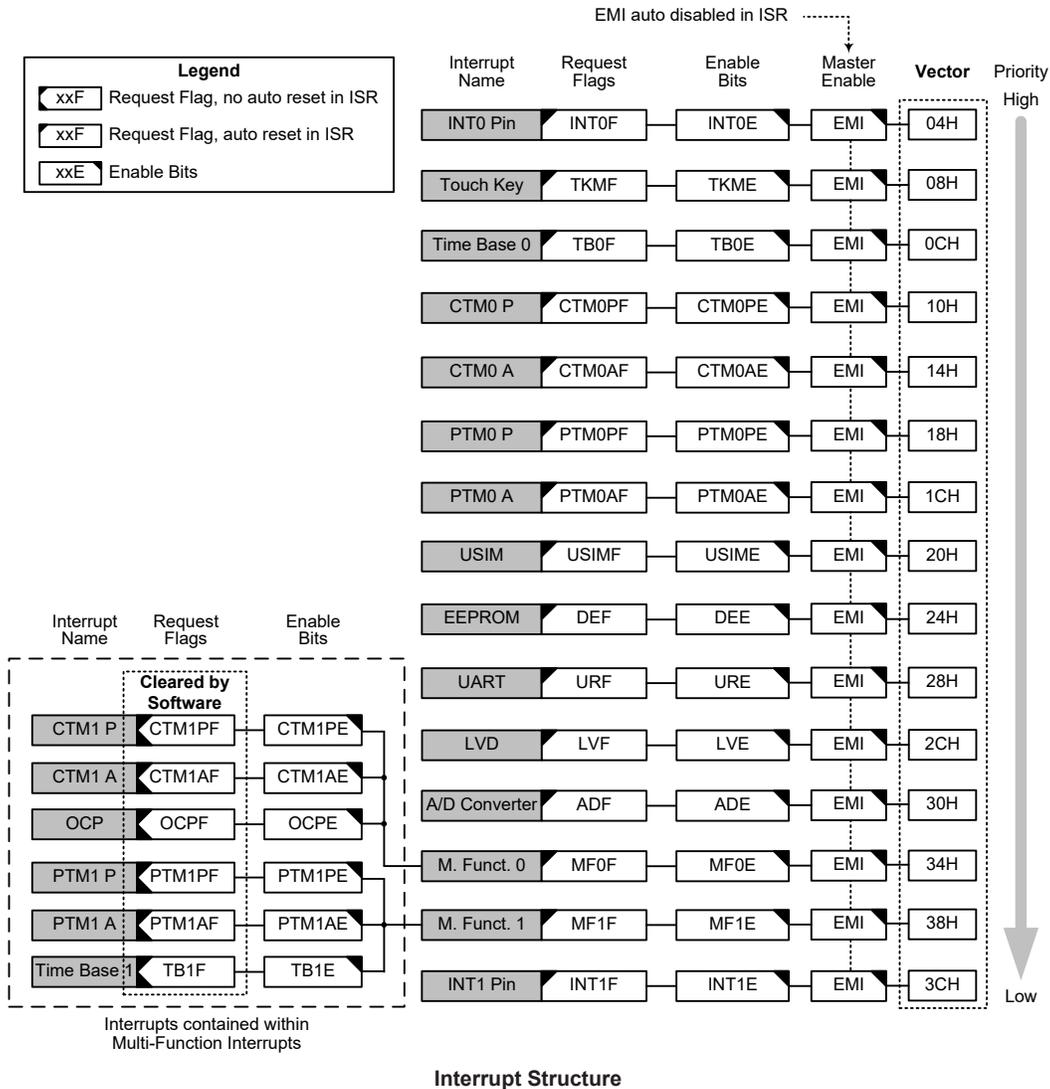
LVD Operation

The Low Voltage Detector also has its own interrupt, 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 LVDF 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 LVDF 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 pins, while the internal interrupts are generated by various internal functions such as the TMs, Time Bases, LVD, EEPROM, USIM, 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, as shown in the accompanying table. The registers fall into three categories. The first is the INTC0~INTC3 registers which setup the primary interrupts, the second is the MF10~MF11 registers which setup the Multi-function interrupts. Finally there is an INTEG register to 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 Pin | INTnE | INTnF | n=0~1 |
| Touch Key | TKME | TKMF | — |
| A/D Converter | ADE | ADF | — |
| Multi-function | MFnE | MFnF | n=0~1 |
| Over Circuit Protection | OCPE | OCPF | — |
| CTMn | CTMnPE | CTMnPF | n=0~1 |
| | CTMnAE | CTMnAF | |
| PTMn | PTMnPE | PTMnPF | n=0~1 |
| | PTMnAE | PTMnAF | |
| LVD | LVDE | LVDF | — |
| EEPROM erase or write operation | DEE | DEF | — |
| Time Base | TBnE | TBnF | n=0~1 |
| USIM | USIME | USIMF | — |
| UART | URE | URF | — |

Interrupt Register Bit Naming Conventions

| Register Name | Bit | | | | | | | |
|---------------|--------|--------|--------|--------|--------|--------|--------|--------|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| INTEG | — | — | — | — | INT1S1 | INT1S0 | INT0S1 | INT0S0 |
| INTC0 | — | TB0F | TKMF | INT0F | TB0E | TKME | INT0E | EMI |
| INTC1 | PTM0AF | PTM0PF | CTM0AF | CTM0PF | PTM0AE | PTM0PE | CTM0AE | CTM0PE |
| INTC2 | LVDF | URF | DEF | USIMF | LVDE | URE | DEE | USIME |
| INTC3 | INT1F | MF1F | MF0F | ADF | INT1E | MF1E | MF0E | ADE |
| MF10 | — | OCPF | CTM1AF | CTM1PF | — | OCPE | CTM1AE | CTM1PE |
| MF11 | — | TB1F | PTM1AF | PTM1PF | — | TB1E | PTM1AE | PTM1PE |

Interrupt Register List

• **INTEG Register**

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

Bit 7~4 Unimplemented, read as “0”

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

Bit 1~0 **INT0S1~INT0S0**: Interrupt trigger edge selection 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 | — | TB0F | TKMF | INT0F | TB0E | TKME | 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 **TB0F**: Time Base 0 request flag
0: No request
1: Interrupt request
- Bit 5 **TKMF**: Touch Key Module interrupt request flag
0: No request
1: Interrupt request
- Bit 4 **INT0F**: INT0 interrupt request flag
0: No request
1: Interrupt request
- Bit 3 **TB0E**: Time Base 0 interrupt control
0: Disable
1: Enable
- Bit 2 **TKME**: Touch Key Module 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 | PTM0AF | PTM0PF | CTM0AF | CTM0PF | PTM0AE | PTM0PE | CTM0AE | CTM0PE |
| 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 **PTM0AF**: PTM0 Comparator A match interrupt request flag
0: No request
1: Interrupt request
- Bit 6 **PTM0PF**: PTM0 Comparator P match interrupt request flag
0: No request
1: Interrupt request
- Bit 5 **CTM0AF**: CTM0 Comparator A match interrupt request flag
0: No request
1: Interrupt request
- Bit 4 **CTM0PF**: CTM0 Comparator P match interrupt request flag
0: No request
1: Interrupt request
- Bit 3 **PTM0AE**: PTM0 Comparator A match interrupt control
0: Disable
1: Enable
- Bit 2 **PTM0PE**: PTM0 Comparator P match interrupt control
0: Disable
1: Enable

- Bit 1 **CTM0AE**: CTM0 Comparator A match interrupt control
0: Disable
1: Enable
- Bit 0 **CTM0PE**: CTM0 Comparator P match interrupt control
0: Disable
1: Enable

• **INTC2 Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------|------|-----|-----|-------|------|-----|-----|-------|
| Name | LVDF | URF | DEF | USIMF | LVDE | URE | DEE | USIME |
| 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 **LVDF**: LVD interrupt request flag
0: No request
1: Interrupt request
- Bit 6 **URF**: UART interrupt request flag
0: No request
1: Interrupt request
- Bit 5 **DEF**: Data EEPROM interrupt request flag
0: No request
1: Interrupt request
- Bit 4 **USIMF**: USIM interrupt request flag
0: No request
1: Interrupt request
- Bit 3 **LVDE**: LVD interrupt control
0: Disable
1: Enable
- Bit 2 **URE**: UART interrupt control
0: Disable
1: Enable
- Bit 1 **DEE**: Data EEPROM interrupt control
0: Disable
1: Enable
- Bit 0 **USIME**: USIM interrupt control
0: Disable
1: Enable

• **INTC3 Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------|-------|------|------|-----|-------|------|------|-----|
| Name | INT1F | MF1F | MF0F | ADF | INT1E | MF1E | MF0E | ADE |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| POR | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

- Bit 7 **INT1F**: INT1 interrupt request flag
0: No request
1: Interrupt request
- Bit 6 **MF1F**: Multi-function interrupt 1 request flag
0: No request
1: Interrupt request
- Bit 5 **MF0F**: Multi-function interrupt 0 request flag
0: No request
1: Interrupt request

- Bit 4 **ADF**: A/D Converter interrupt request flag
 0: No request
 1: Interrupt request
- Bit 3 **INTIE**: INT1 interrupt control
 0: Disable
 1: Enable
- Bit 2 **MF1E**: Multi-function interrupt 1 control
 0: Disable
 1: Enable
- Bit 1 **MF0E**: Multi-function interrupt 0 control
 0: Disable
 1: Enable
- Bit 0 **ADE**: A/D Converter interrupt control
 0: Disable
 1: Enable

• **MF10 Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------|---|------|--------|--------|---|------|--------|--------|
| Name | — | OCPF | CTM1AF | CTM1PF | — | OCPE | CTM1AE | CTM1PE |
| R/W | — | R/W | R/W | R/W | — | R/W | R/W | R/W |
| POR | — | 0 | 0 | 0 | — | 0 | 0 | 0 |

- Bit 7 Unimplemented, read as “0”
- Bit 6 **OCPF**: OCP 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 **CTM1AF**: CTM1 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 **CTM1PF**: CTM1 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 Unimplemented, read as “0”
- Bit 2 **OCPE**: OCP interrupt control
 0: Disable
 1: Enable
- Bit 1 **CTM1AE**: CTM1 Comparator A match interrupt control
 0: Disable
 1: Enable
- Bit 0 **CTM1PE**: CTM1 Comparator P match interrupt control
 0: Disable
 1: Enable

• **MF1 Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------|---|------|--------|--------|---|------|--------|--------|
| Name | — | TB1F | PTM1AF | PTM1PF | — | TB1E | PTM1AE | PTM1PE |
| R/W | — | R/W | R/W | R/W | — | R/W | R/W | R/W |
| POR | — | 0 | 0 | 0 | — | 0 | 0 | 0 |

- Bit 7 Unimplemented, read as “0”
- Bit 6 **TB1F**: Time Base 1 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 **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 4 **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 3 Unimplemented, read as “0”
- Bit 2 **TB1E**: Time Base 1 interrupt control
 0: Disable
 1: Enable
- Bit 1 **PTM1AE**: PTM1 Comparator A match interrupt control
 0: Disable
 1: Enable
- Bit 0 **PTM1PE**: PTM1 Comparator P match interrupt control
 0: Disable
 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~INT1. An external interrupt request will take place when the external interrupt request flags, INT0F~INT1F, are set, which will occur when a transition, whose type is chosen by the edge select bits, appears on the external interrupt pins. To allow the program to branch to their respective interrupt vector addresses, the global interrupt enable bit, EMI, and respective external interrupt enable bits, INT0E~INT1E, must first be set. Additionally the correct interrupt edge type must be selected using the INTEG register to enable the external interrupt function and to choose the trigger edge type. As the external interrupt pins are pin-shared with I/O pins, they can only be configured as external interrupt pins if their external interrupt enable bit in the corresponding interrupt register has been set and the external interrupt pin is selected by the corresponding pin-shared function selection bits. The pin must also be setup as an input by setting the corresponding bit in the port control register. When the interrupt is enabled, the stack is not full and the correct transition type appears on the external interrupt pins, a subroutine call to the corresponding external interrupt vector, will take place. When the interrupt is serviced, the external interrupt request flags, INT0F~INT1F, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts. Note that any pull-high resistor selections on the external interrupt pins will remain valid even if the pin is used as an external interrupt input.

The INTEG register is used to select the type of active edge that will trigger the external interrupt. A choice of either rising or falling or both edge types can be chosen to trigger an external interrupt. Note that the INTEG register can also be used to disable the external interrupt function.

Touch Key Interrupt

A Touch Key interrupt will take place when the Touch Key interrupt request flag, TMKF, is set, a situation that will occur when the time slot counter overflows in manual mode or all the touch key auto scan operations finish. To allow the program to branch to its interrupt vector address, the global interrupt enable bit, EMI, and the Touch Key interrupt enable bit, TKME, must be first set. When the interrupt is enabled, the stack is not full and the touch key time slot counter overflow occurs or all the touch key auto scan operations finish, a subroutine call to the relevant interrupt vector, will take place. When the interrupt is serviced, the Touch Key interrupt request flag will be automatically reset and the EMI bit will also be automatically cleared to disable other interrupts.

Universal Serial Interface Module Interrupt

The Universal Serial Interface Module Interrupt, also known as the USIM interrupt, will take place when the USIM Interrupt request flag, USIMF, is set. As the USIM interface can operate in three

modes which are SPI mode, I²C mode and UART mode, the USIMF flag can be set by different conditions depending on the selected interface mode.

If the SPI or I²C mode is selected, the USIM interrupt can be triggered when a byte of data has been received or transmitted by the SPI/I²C interface, or an I²C slave address match occurs, or an I²C bus time-out occurs. If the UART mode is selected, several individual UART conditions including a transmitter data register empty, transmitter idle, receiver data available, receiver overrun, address detect and an URX/UTX pin wake-up, can generate a USIM interrupt with the USIMF flag bit set high.

To allow the program to branch to its interrupt vector address, the global interrupt enable bit, EMI, and the Serial Interface Interrupt enable bit, USIME, 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 USIM Interrupt vector, will take place. When the interrupt is serviced, the Universal Serial Interface Interrupt flag, USIMF, will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts.

Note that if the USIM interrupt is triggered by the UART interface, after the interrupt has been serviced, the UUSR register flags will only be cleared when certain actions are taken by the UART, the details of which are given in the UART section.

EEPROM 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, and EEPROM Interrupt enable bit, DEE, 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 EEPROM Interrupt vector will take place. When the EEPROM Interrupt is serviced, the EEPROM Interrupt flag, DEF, will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts.

UART Interrupt

Several individual UART conditions can generate a UART interrupt. When one of these conditions occurs, an interrupt pulse will be generated to get the attention of the microcontroller. These conditions are a transmitter data register empty, transmitter idle, receiver reaching FIFO trigger level, receiver overrun, address detect and an RX/TX pin wake-up. To allow the program to branch to the corresponding interrupt vector address, the global interrupt enable bit, EMI, and the UART interrupt enable bit, URE, must first be set. When the interrupt is enabled, the stack is not full and any of the conditions described above occurs, a subroutine call to the UART Interrupt vector, will take place. When the UART Interrupt is serviced, the UART Interrupt flag, URF, will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts. However, the USR register flags will only be cleared when certain actions are taken by the UART, the details of which are given in the UART section.

LVD Interrupt

An LVD Interrupt request will take place when the LVD Interrupt request flag, LVDF, 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, and Low Voltage Interrupt enable bit, LVDE, 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 LVD Interrupt vector, will take place. When the Low Voltage Interrupt is serviced, the LVD Interrupt flag, LVDF, will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts.

A/D Converter Interrupt

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

Multi-function Interrupts

Within the device there are two 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 CTM1 and PTM1 interrupts, OCP interrupt and Time Base 1 interrupt.

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

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

TM Interrupts

The Compact and Periodic Type TMs each have two interrupts, one comes from the comparator A match situation and the other comes from the comparator P match situation. The CTM0 and PTM0 interrupt sources have their own individual vector while the CTM1 and PTM1 interrupts are contained within the Multi-function Interrupts. For 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 the CTM0 or PTM0 interrupt vector address, the global interrupt enable bit, EMI, respective TM Interrupt enable bit must first be set. When the interrupt is enabled, the stack is not full and the CTM0 or PTM0 comparator match situation occurs, a subroutine call to the CTM0 or PTM0 Interrupt vector location, will take place. When the TM interrupt is serviced, the CTM0 or PTM0 interrupt request flag will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts.

To allow the program to branch to the CTM1 or PTM1 interrupt vector address, the global interrupt enable bit, EMI, respective TM Interrupt enable bit, and the relevant Multi-function Interrupt enable bit, MFnE, must first be set. When the interrupt is enabled, the stack is not full and the CTM1 or PTM1 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, the related MFnF flag will also be automatically cleared. As the CTM1 or PTM1 interrupt request flag will not be automatically cleared, they have to be cleared by the application program.

Over Current Protection Interrupt

The OCP Interrupt is controlled by detecting the OCPP or OCPN input current. An OCP Interrupt request will take place when the OCP Interrupt request flag, OCPF, is set, which occurs when the OCP circuit detects an over current condition. To allow the program to branch to the relevant interrupt vector address, the global interrupt enable bit, EMI, OCP Interrupt enable bit, and the relevant Multi-function Interrupt enable bit, MF0E, must first be set. When the interrupt is enabled, the stack is not full and an over current condition is detected, a subroutine call to the relevant Multi-function Interrupt vector location, will take place. When the OCP interrupt is serviced, the EMI bit will be automatically cleared to disable other interrupts, the related MF0F flag will also be automatically cleared. As the OCP interrupt request flag will not be automatically cleared, they have to be cleared by the application program.

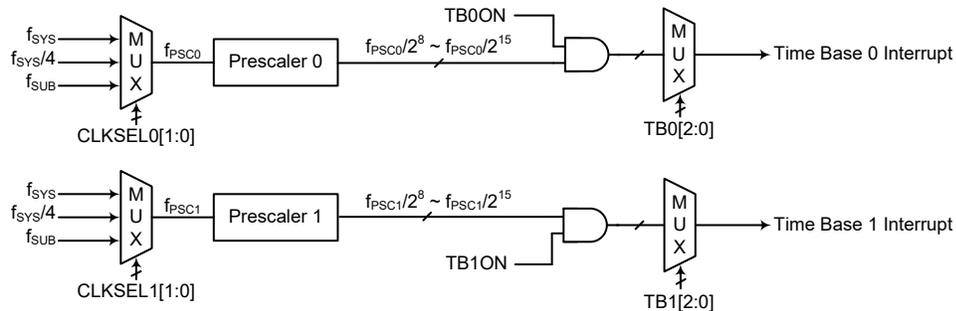
Time Base Interrupts

The function of the Time Base Interrupts is to provide regular time signal in the form of an internal interrupt. They are controlled by the overflow signals from their respective internal timer functions. When these happen their respective interrupt request flags, TB0F or TB0F, will be set.

To allow the program to branch to the Time Base 0 interrupt vector address, the global interrupt enable bit, EMI, and Time Base 0 enable bit, TB0E, must first be set. When the interrupt is enabled, the stack is not full and the Time Base 0 overflows, a subroutine call to its respective interrupt vector location will take place. When the Time Base 0 interrupt is serviced, the TB0F flag will be automatically reset and the EMI bit will automatically be cleared to disable other interrupts.

The Time Base 1 interrupt is contained within the Multi-function Interrupt. To allow the program to branch to the Time Base 1 interrupt vector address, the global interrupt enable bit, EMI, Time Base 1 Interrupt enable bit, and the relevant Multi-function Interrupt enable bit, MF1E, must first be set. When the interrupt is enabled, the stack is not full and the Time Base 1 overflows, a subroutine call to the relevant Multi-function Interrupt vector location, will take place. When the Time Base 1 interrupt is serviced, the EMI bit will be automatically cleared to disable other interrupts, the related MF1F flag will also be automatically cleared. As the Time Base 1 interrupt request flag will not be automatically cleared, they have to be cleared by the application program.

The purpose of the Time Base Interrupts is to provide an interrupt signal at fixed time periods. Their respective clock source, f_{PSC0} or f_{PSC1} , originates from the internal clock source f_{SYS} , $f_{SYS}/4$ or f_{SUB} and then passes through a divider, the division ratio of which is selected by programming the appropriate bits in the TB0C and TB1C registers to obtain longer interrupt periods whose value ranges. The clock source which in turn controls the Time Base interrupt period is selected using the CLKSEL0[1:0] and CLKSEL1[1:0] bits in the PSC0R and PSC1R register respectively.



Time Base Interrupts

• **PSC0R Register**

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

Bit 7~2 Unimplemented, read as “0”

Bit 1~0 **CLKSEL01~CLKSEL00**: Prescaler 0 clock source f_{PSC0} 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 f_{PSC1} 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: $2^{15}/f_{PSC1}$ |
|---------|---|

Interrupt Wake-up Function

Each of the interrupt functions has the capability of waking up the microcontroller when in the SLEEP or IDLE Mode. A wake-up is generated when an interrupt request flag changes from low to high and is independent of whether the interrupt is enabled or not. Therefore, even though the device is in the SLEEP or IDLE Mode and its system oscillator stopped, situations such as external edge transitions on the external interrupt pin may cause their respective interrupt flag to be set high and consequently generate an interrupt. Care must therefore be taken if spurious wake-up situations are to be avoided. If an interrupt wake-up function is to be disabled then the corresponding interrupt request flag should be set high before the device enters the SLEEP or IDLE Mode. The interrupt enable bits have no effect on the interrupt wake-up function.

Programming Considerations

By disabling the relevant interrupt enable bits, a requested interrupt can be prevented from being serviced, however, once an interrupt request flag is set, it will remain in this condition in the interrupt register until the corresponding interrupt is serviced or until the request flag is cleared by the application program.

Where a certain interrupt is contained within a Multi-function interrupt, then when the interrupt service routine is executed, as only the Multi-function interrupt request flags, MF_nF, 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 the 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.

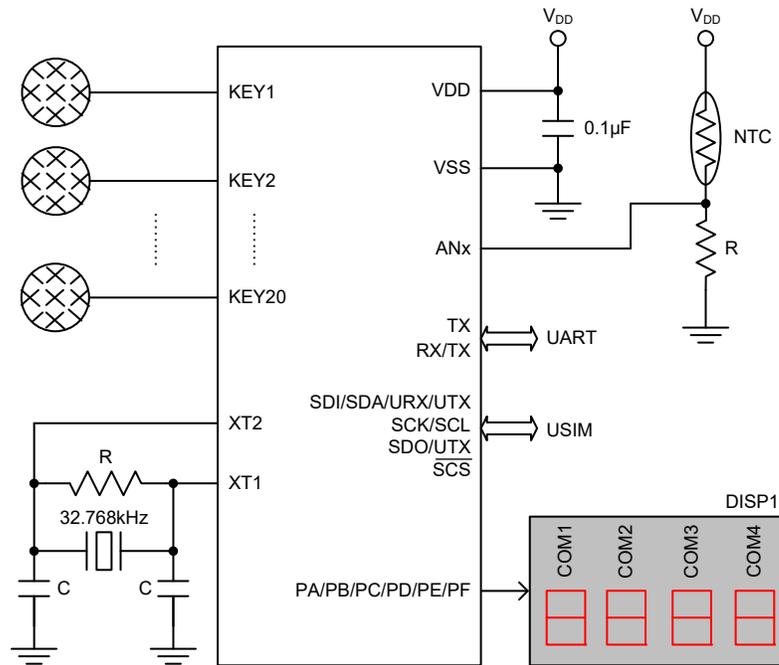
Configuration Options

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 |
|--------------------------|---|
| Oscillator Option | |
| 1 | HIRC frequency selection – f_{HIRC} : 8MHz, 12MHz, 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 μ s and branch or call instructions would be implemented within 1 μ 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 |
|----------------------------------|---|--------|----------------------|
| Arithmetic | | | |
| ADD A,[m] | Add Data Memory to ACC | 1 | Z, C, AC, OV, SC |
| ADDM A,[m] | Add ACC to Data Memory | ↑Note | 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 | ↑Note | 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 | ↑Note | 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 | ↑Note | Z, C, AC, OV, SC, CZ |
| DAA [m] | Decimal adjust ACC for Addition with result in Data Memory | ↑Note | C |
| Logic Operation | | | |
| 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 | ↑Note | Z |
| ORM A,[m] | Logical OR ACC to Data Memory | ↑Note | Z |
| XORM A,[m] | Logical XOR ACC to Data Memory | ↑Note | 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 | ↑Note | Z |
| CPLA [m] | Complement Data Memory with result in ACC | 1 | Z |
| Increment & Decrement | | | |
| INCA [m] | Increment Data Memory with result in ACC | 1 | Z |
| INC [m] | Increment Data Memory | ↑Note | Z |
| DECA [m] | Decrement Data Memory with result in ACC | 1 | Z |
| DEC [m] | Decrement Data Memory | ↑Note | Z |
| Rotate | | | |
| RRA [m] | Rotate Data Memory right with result in ACC | 1 | None |
| RR [m] | Rotate Data Memory right | ↑Note | None |
| RRCA [m] | Rotate Data Memory right through Carry with result in ACC | 1 | C |
| RRC [m] | Rotate Data Memory right through Carry | ↑Note | C |
| RLA [m] | Rotate Data Memory left with result in ACC | 1 | None |
| RL [m] | Rotate Data Memory left | ↑Note | None |
| RLCA [m] | Rotate Data Memory left through Carry with result in ACC | 1 | C |
| RLC [m] | Rotate Data Memory left through Carry | ↑Note | C |

| Mnemonic | Description | Cycles | Flag Affected |
|-----------------------------|---|-------------------|---------------|
| Data Move | | | |
| MOV A,[m] | Move Data Memory to ACC | 1 | None |
| MOV [m],A | Move ACC to Data Memory | 1 ^{Note} | None |
| MOV A,x | Move immediate data to ACC | 1 | None |
| Bit Operation | | | |
| CLR [m].i | Clear bit of Data Memory | 1 ^{Note} | None |
| SET [m].i | Set bit of Data Memory | 1 ^{Note} | None |
| Branch Operation | | | |
| JMP addr | Jump unconditionally | 2 | None |
| SZ [m] | Skip if Data Memory is zero | 1 ^{Note} | None |
| SZA [m] | Skip if Data Memory is zero with data movement to ACC | 1 ^{Note} | None |
| SZ [m].i | Skip if bit i of Data Memory is zero | 1 ^{Note} | None |
| SNZ [m] | Skip if Data Memory is not zero | 1 ^{Note} | None |
| SNZ [m].i | Skip if bit i of Data Memory is not zero | 1 ^{Note} | None |
| SIZ [m] | Skip if increment Data Memory is zero | 1 ^{Note} | None |
| SDZ [m] | Skip if decrement Data Memory is zero | 1 ^{Note} | None |
| SIZA [m] | Skip if increment Data Memory is zero with result in ACC | 1 ^{Note} | None |
| SDZA [m] | Skip if decrement Data Memory is zero with result in ACC | 1 ^{Note} | 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 |
| Table Read Operation | | | |
| TABRD [m] | Read table (specific page) to TBLH and Data Memory | 2 ^{Note} | None |
| TABRDL [m] | Read table (last page) to TBLH and Data Memory | 2 ^{Note} | None |
| ITABRD [m] | Increment table pointer TBLP first and Read table (specific page) to TBLH and Data Memory | 2 ^{Note} | None |
| ITABRDL [m] | Increment table pointer TBLP first and Read table (last page) to TBLH and Data Memory | 2 ^{Note} | None |
| Miscellaneous | | | |
| NOP | No operation | 1 | None |
| CLR [m] | Clear Data Memory | 1 ^{Note} | None |
| SET [m] | Set Data Memory | 1 ^{Note} | None |
| CLR WDT | Clear Watchdog Timer | 1 | TO, PDF |
| SWAP [m] | Swap nibbles of Data Memory | 1 ^{Note} | 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 |
|----------------------------------|---|-------------------|----------------------|
| Arithmetic | | | |
| LADD A,[m] | Add Data Memory to ACC | 2 | Z, C, AC, OV, SC |
| LADDM A,[m] | Add ACC to Data Memory | 2 ^{Note} | 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 ^{Note} | 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 ^{Note} | 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 ^{Note} | Z, C, AC, OV, SC, CZ |
| LDAA [m] | Decimal adjust ACC for Addition with result in Data Memory | 2 ^{Note} | C |
| Logic Operation | | | |
| 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 ^{Note} | Z |
| LORM A,[m] | Logical OR ACC to Data Memory | 2 ^{Note} | Z |
| LXORM A,[m] | Logical XOR ACC to Data Memory | 2 ^{Note} | Z |
| LCPL [m] | Complement Data Memory | 2 ^{Note} | Z |
| LCPLA [m] | Complement Data Memory with result in ACC | 2 | Z |
| Increment & Decrement | | | |
| LINCA [m] | Increment Data Memory with result in ACC | 2 | Z |
| LINC [m] | Increment Data Memory | 2 ^{Note} | Z |
| LDECA [m] | Decrement Data Memory with result in ACC | 2 | Z |
| LDEC [m] | Decrement Data Memory | 2 ^{Note} | Z |
| Rotate | | | |
| LRRRA [m] | Rotate Data Memory right with result in ACC | 2 | None |
| LRR [m] | Rotate Data Memory right | 2 ^{Note} | None |
| LRRCA [m] | Rotate Data Memory right through Carry with result in ACC | 2 | C |
| LRRC [m] | Rotate Data Memory right through Carry | 2 ^{Note} | C |
| LRLA [m] | Rotate Data Memory left with result in ACC | 2 | None |
| LRL [m] | Rotate Data Memory left | 2 ^{Note} | None |
| LRLCA [m] | Rotate Data Memory left through Carry with result in ACC | 2 | C |
| LRLC [m] | Rotate Data Memory left through Carry | 2 ^{Note} | C |
| Data Move | | | |
| LMOV A,[m] | Move Data Memory to ACC | 2 | None |
| LMOV [m],A | Move ACC to Data Memory | 2 ^{Note} | None |
| Bit Operation | | | |
| LCLR [m].i | Clear bit of Data Memory | 2 ^{Note} | None |
| LSET [m].i | Set bit of Data Memory | 2 ^{Note} | None |

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

| | |
|-------------------|---|
| ADC A,[m] | 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 |
| ADCM A,[m] | 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 |
| ADD A,[m] | 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 |
| ADD A,x | 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 |
| ADDM A,[m] | 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 |
| AND A,[m] | 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 |
| AND A,x | 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 |
| ANDM A,[m] | 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 |

| | |
|------------------|--|
| CALL addr | 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 Program Counter ← addr |
| Affected flag(s) | None |
| CLR [m] | Clear Data Memory |
| Description | Each bit of the specified Data Memory is cleared to 0. |
| Operation | [m] ← 00H |
| Affected flag(s) | None |
| CLR [m].i | 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 |
| CLR WDT | Clear Watchdog Timer |
| Description | The TO, PDF flags and the WDT are all cleared. |
| Operation | WDT cleared TO ← 0 PDF ← 0 |
| Affected flag(s) | TO, PDF |
| CPL [m] | 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 |
| CPLA [m] | 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 |
| DAA [m] | 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 [m] ← ACC + 06H or [m] ← ACC + 60H or [m] ← ACC + 66H |
| Affected flag(s) | C |

| | |
|------------------|--|
| DEC [m] | Decrement Data Memory |
| Description | Data in the specified Data Memory is decremented by 1. |
| Operation | $[m] \leftarrow [m] - 1$ |
| Affected flag(s) | Z |
| | |
| DECA [m] | 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 |
| | |
| HALT | 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 | $TO \leftarrow 0$ $PDF \leftarrow 1$ |
| Affected flag(s) | TO, PDF |
| | |
| INC [m] | Increment Data Memory |
| Description | Data in the specified Data Memory is incremented by 1. |
| Operation | $[m] \leftarrow [m] + 1$ |
| Affected flag(s) | Z |
| | |
| INCA [m] | 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 |
| | |
| JMP addr | 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 |
| | |
| MOV A,[m] | 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 |
| | |
| MOV A,x | Move immediate data to ACC |
| Description | The immediate data specified is loaded into the Accumulator. |
| Operation | $ACC \leftarrow x$ |
| Affected flag(s) | None |
| | |
| MOV [m],A | 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 |

| | |
|------------------|--|
| NOP | No operation |
| Description | No operation is performed. Execution continues with the next instruction. |
| Operation | No operation |
| Affected flag(s) | None |
| OR A,[m] | Logical OR Data Memory to ACC |
| Description | Data in the Accumulator and the specified Data Memory perform a bitwise logical OR operation. The result is stored in the Accumulator. |
| Operation | ACC ← ACC "OR" [m] |
| Affected flag(s) | Z |
| OR A,x | Logical OR immediate data to ACC |
| Description | Data in the Accumulator and the specified immediate data perform a bitwise logical OR operation. The result is stored in the Accumulator. |
| Operation | ACC ← ACC "OR" x |
| Affected flag(s) | Z |
| ORM A,[m] | Logical OR ACC to Data Memory |
| Description | Data in the specified Data Memory and the Accumulator perform a bitwise logical OR operation. The result is stored in the Data Memory. |
| Operation | [m] ← ACC "OR" [m] |
| Affected flag(s) | Z |
| RET | Return from subroutine |
| Description | The Program Counter is restored from the stack. Program execution continues at the restored address. |
| Operation | Program Counter ← Stack |
| Affected flag(s) | None |
| RET A,x | Return from subroutine and load immediate data to ACC |
| Description | The Program Counter is restored from the stack and the Accumulator loaded with the specified immediate data. Program execution continues at the restored address. |
| Operation | Program Counter ← Stack ACC ← x |
| Affected flag(s) | None |
| RETI | Return from interrupt |
| Description | The Program Counter is restored from the stack and the interrupts are re-enabled by setting the EMI bit. EMI is the master interrupt global enable bit. If an interrupt was pending when the RETI instruction is executed, the pending Interrupt routine will be processed before returning to the main program. |
| Operation | Program Counter ← Stack EMI ← 1 |
| Affected flag(s) | None |
| RL [m] | Rotate Data Memory left |
| Description | The contents of the specified Data Memory are rotated left by 1 bit with bit 7 rotated into bit 0. |
| Operation | [m].(i+1) ← [m].i; (i=0~6) [m].0 ← [m].7 |
| Affected flag(s) | None |

| | |
|------------------|---|
| RLA [m] | 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) ACC.0 ← [m].7 |
| Affected flag(s) | None |
| | |
| RLC [m] | 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) [m].0 ← C C ← [m].7 |
| Affected flag(s) | C |
| | |
| RLCA [m] | 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) ACC.0 ← C C ← [m].7 |
| Affected flag(s) | C |
| | |
| RR [m] | 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) [m].7 ← [m].0 |
| Affected flag(s) | None |
| | |
| RRA [m] | 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) ACC.7 ← [m].0 |
| Affected flag(s) | None |
| | |
| RRC [m] | 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) [m].7 ← C C ← [m].0 |
| Affected flag(s) | C |

| | |
|-------------------|---|
| RRCA [m] | 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) ACC.7 ← C C ← [m].0 |
| Affected flag(s) | C |
| SBC A,[m] | 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 |
| SBC A, x | 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 |
| SBCM A,[m] | 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 |
| SDZ [m] | 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 Skip if [m]=0 |
| Affected flag(s) | None |
| SDZA [m] | 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 Skip if ACC=0 |
| Affected flag(s) | None |

| | |
|------------------|--|
| SET [m] | Set Data Memory |
| Description | Each bit of the specified Data Memory is set to 1. |
| Operation | $[m] \leftarrow FFH$ |
| Affected flag(s) | None |
| | |
| SET [m].i | 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 |
| | |
| SIZ [m] | 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 | $[m] \leftarrow [m] + 1$ Skip if $[m]=0$ |
| Affected flag(s) | None |
| | |
| SIZA [m] | 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 | $ACC \leftarrow [m] + 1$ Skip if $ACC=0$ |
| Affected flag(s) | None |
| | |
| SNZ [m].i | 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 |
| | |
| SNZ [m] | 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 |
| | |
| SUB A,[m] | 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 |

| | |
|-------------------|---|
| SUBM A,[m] | 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 |
| | |
| SUB A,x | 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 |
| | |
| SWAP [m] | 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 |
| | |
| SWAPA [m] | 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 | $ACC.3 \sim ACC.0 \leftarrow [m].7 \sim [m].4$ $ACC.7 \sim ACC.4 \leftarrow [m].3 \sim [m].0$ |
| Affected flag(s) | None |
| | |
| SZ [m] | 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 |
| | |
| SZA [m] | 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 | $ACC \leftarrow [m]$ Skip if $[m]=0$ |
| Affected flag(s) | None |
| | |
| SZ [m].i | 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 |

| | |
|--------------------|--|
| TABRD [m] | 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) TBLH ← program code (high byte) |
| Affected flag(s) | None |
| | |
| TABRDL [m] | 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) TBLH ← program code (high byte) |
| Affected flag(s) | None |
| | |
| ITABRD [m] | 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) TBLH ← program code (high byte) |
| Affected flag(s) | None |
| | |
| ITABRDL [m] | 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) TBLH ← program code (high byte) |
| Affected flag(s) | None |
| | |
| XOR A,[m] | 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 |
| | |
| XORM A,[m] | 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 |
| | |
| XOR A,x | 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.

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

| | |
|------------------|--|
| LCPL [m] | 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 |
| | |
| LCPLA [m] | 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 |
| | |
| LDAA [m] | 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 $[m] \leftarrow ACC + 06H$ or $[m] \leftarrow ACC + 60H$ or $[m] \leftarrow ACC + 66H$ |
| Affected flag(s) | C |
| | |
| LDEC [m] | Decrement Data Memory |
| Description | Data in the specified Data Memory is decremented by 1. |
| Operation | $[m] \leftarrow [m] - 1$ |
| Affected flag(s) | Z |
| | |
| LDECA [m] | 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 |
| | |
| LINC [m] | Increment Data Memory |
| Description | Data in the specified Data Memory is incremented by 1. |
| Operation | $[m] \leftarrow [m] + 1$ |
| Affected flag(s) | Z |
| | |
| LINCA [m] | 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 |

| | |
|-------------------|---|
| LMOV A,[m] | 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 |
| LMOV [m],A | 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 |
| LOR A,[m] | 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 |
| LORM A,[m] | 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 |
| LRL [m] | 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 | $[m].(i+1) \leftarrow [m].i; (i=0\sim 6)$ $[m].0 \leftarrow [m].7$ |
| Affected flag(s) | None |
| LRLA [m] | 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 | $ACC.(i+1) \leftarrow [m].i; (i=0\sim 6)$ $ACC.0 \leftarrow [m].7$ |
| Affected flag(s) | None |
| LRLC [m] | 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 | $[m].(i+1) \leftarrow [m].i; (i=0\sim 6)$ $[m].0 \leftarrow C$ $C \leftarrow [m].7$ |
| Affected flag(s) | C |
| LRLCA [m] | 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 | $ACC.(i+1) \leftarrow [m].i; (i=0\sim 6)$ $ACC.0 \leftarrow C$ $C \leftarrow [m].7$ |
| Affected flag(s) | C |

| | |
|--------------------|---|
| LRR [m] | 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 | $[m].i \leftarrow [m].(i+1); (i=0\sim 6)$ $[m].7 \leftarrow [m].0$ |
| Affected flag(s) | None |
| | |
| LRRRA [m] | 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 | $ACC.i \leftarrow [m].(i+1); (i=0\sim 6)$ $ACC.7 \leftarrow [m].0$ |
| Affected flag(s) | None |
| | |
| LRRC [m] | 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 | $[m].i \leftarrow [m].(i+1); (i=0\sim 6)$ $[m].7 \leftarrow C$ $C \leftarrow [m].0$ |
| Affected flag(s) | C |
| | |
| LRRCA [m] | 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 | $ACC.i \leftarrow [m].(i+1); (i=0\sim 6)$ $ACC.7 \leftarrow C$ $C \leftarrow [m].0$ |
| Affected flag(s) | C |
| | |
| LSBC A,[m] | 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 |
| | |
| LSBCM A,[m] | 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 |

| | |
|-------------------|---|
| LSDZ [m] | 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 | $[m] \leftarrow [m] - 1$ Skip if $[m]=0$ |
| Affected flag(s) | None |
| LSDZA [m] | 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 | $ACC \leftarrow [m] - 1$ Skip if $ACC=0$ |
| Affected flag(s) | None |
| LSET [m] | Set Data Memory |
| Description | Each bit of the specified Data Memory is set to 1. |
| Operation | $[m] \leftarrow FFH$ |
| Affected flag(s) | None |
| LSET [m].i | 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 |
| LSIZ [m] | 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 | $[m] \leftarrow [m] + 1$ Skip if $[m]=0$ |
| Affected flag(s) | None |
| LSIZA [m] | 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 | $ACC \leftarrow [m] + 1$ Skip if $ACC=0$ |
| Affected flag(s) | None |
| LSNZ [m].i | 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 |

| | |
|--------------------|--|
| LSNZ [m] | 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 |
| | |
| LSUB A,[m] | 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 |
| | |
| LSUBM A,[m] | 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 |
| | |
| LSWAP [m] | 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 |
| | |
| LSWAPA [m] | 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 ACC.7~ACC.4 ← [m].3~[m].0 |
| Affected flag(s) | None |
| | |
| LSZ [m] | 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 |
| | |
| LSZA [m] | 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] Skip if [m]=0 |
| Affected flag(s) | None |

| | |
|---------------------|--|
| LSZ [m].i | 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 |
| | |
| LTABRD [m] | 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) TBLH ← program code (high byte) |
| Affected flag(s) | None |
| | |
| LTABRDL [m] | 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) TBLH ← program code (high byte) |
| Affected flag(s) | None |
| | |
| LITABRD [m] | 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) TBLH ← program code (high byte) |
| Affected flag(s) | None |
| | |
| LITABRDL [m] | 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) TBLH ← program code (high byte) |
| Affected flag(s) | None |
| | |
| LXOR A,[m] | 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 |
| | |
| LXORM A,[m] | 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 |

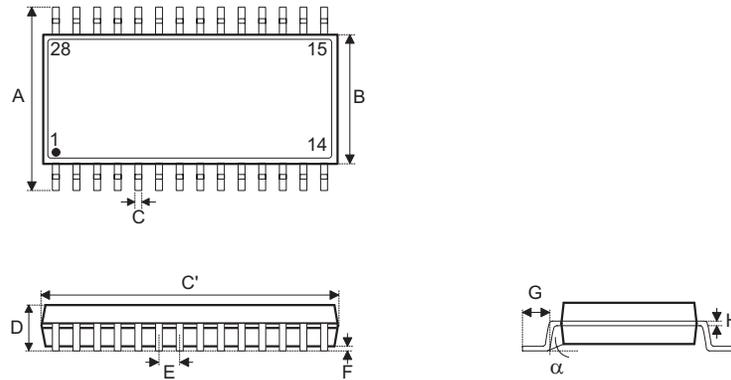
Package Information

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](#)

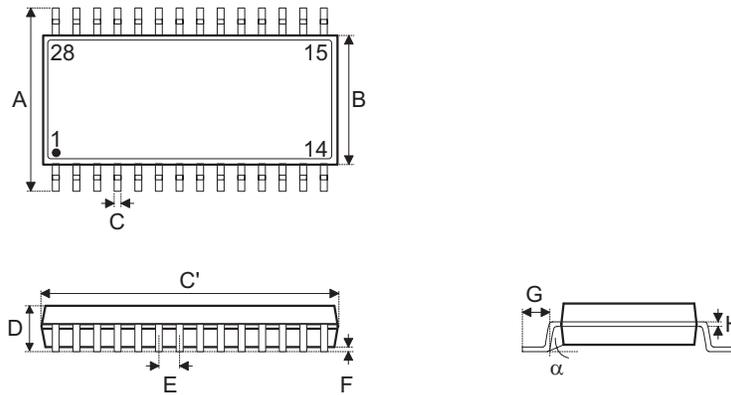
28-pin SOP (300mil) Outline Dimensions



| Symbol | Dimensions in inch | | |
|----------|--------------------|------|-------|
| | Min. | Nom. | Max. |
| A | 0.406 BSC | | |
| B | 0.295 BSC | | |
| C | 0.012 | — | 0.020 |
| C' | 0.705 BSC | | |
| D | — | — | 0.104 |
| E | 0.050 BSC | | |
| F | 0.004 | — | 0.012 |
| G | 0.016 | — | 0.050 |
| H | 0.008 | — | 0.013 |
| α | 0° | — | 8° |

| Symbol | Dimensions in mm | | |
|----------|------------------|------|------|
| | Min. | Nom. | Max. |
| A | 10.30 BSC | | |
| B | 7.50 BSC | | |
| C | 0.31 | — | 0.51 |
| C' | 17.90 BSC | | |
| D | — | — | 2.65 |
| E | 1.27 BSC | | |
| F | 0.10 | — | 0.30 |
| G | 0.40 | — | 1.27 |
| H | 0.20 | — | 0.33 |
| α | 0° | — | 8° |

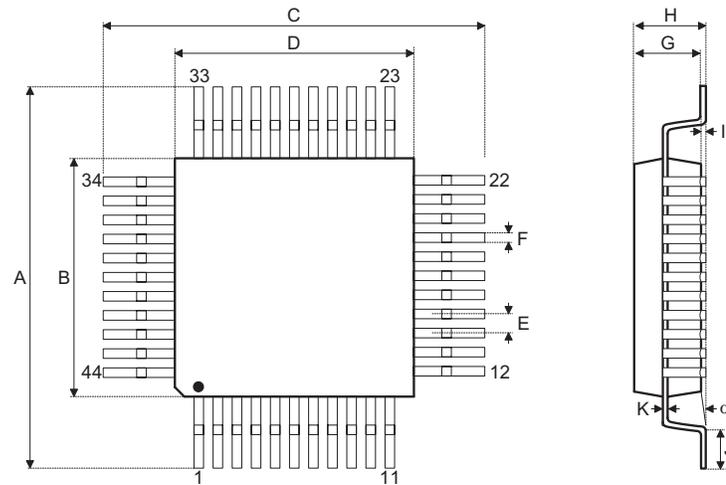
28-pin SSOP (150mil) Outline Dimensions



| 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 |
| α | 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 |
| α | 0° | — | 8° |

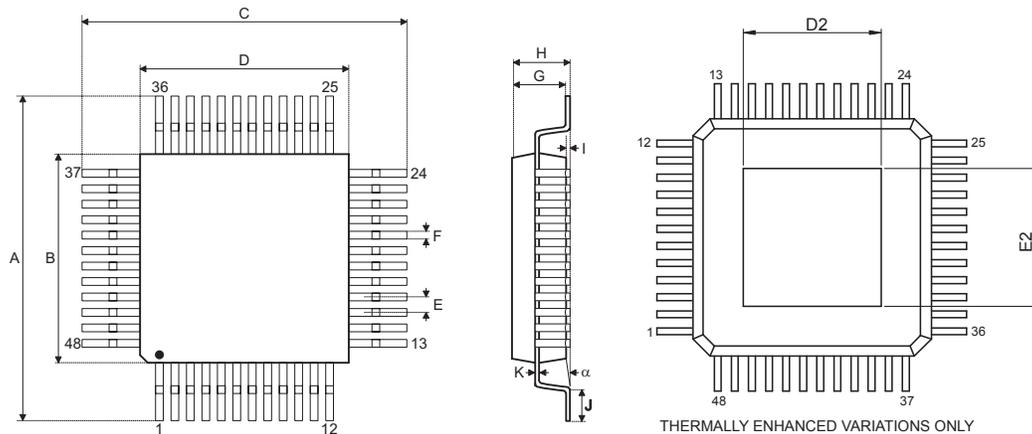
44-pin LQFP (10mm×10mm) (FP2.0mm) Outline Dimensions



| Symbol | Dimensions in inch | | |
|--------|--------------------|-----------|-------|
| | Min. | Nom. | Max. |
| A | | 0.472 BSC | |
| B | | 0.394 BSC | |
| C | | 0.472 BSC | |
| D | | 0.394 BSC | |
| E | | 0.032 BSC | |
| F | 0.012 | 0.015 | 0.018 |
| G | 0.053 | 0.055 | 0.057 |
| H | — | — | 0.063 |
| I | 0.002 | — | 0.006 |
| J | 0.018 | 0.024 | 0.030 |
| K | 0.004 | — | 0.008 |
| α | 0° | — | 7° |

| Symbol | Dimensions in mm | | |
|--------|------------------|-----------|------|
| | Min. | Nom. | Max. |
| A | | 12.00 BSC | |
| B | | 10.00 BSC | |
| C | | 12.00 BSC | |
| D | | 10.00 BSC | |
| E | | 0.80 BSC | |
| F | 0.30 | 0.37 | 0.45 |
| G | 1.35 | 1.40 | 1.45 |
| H | — | — | 1.60 |
| I | 0.05 | — | 0.15 |
| J | 0.45 | 0.60 | 0.75 |
| K | 0.09 | — | 0.20 |
| α | 0° | — | 7° |

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



THERMALLY ENHANCED VARIATIONS ONLY

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

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

Copyright© 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.