Touch A/D Flash MCU

# BS84B08C/BS84C12C

Revision: V1.70 Date: June 17, 2026

www.holtek.com

# Table of Contents

**Features** ..... 6
CPU Features ..... 6
Peripheral Features ..... 6
**Development Tools** ..... 7
**General Description** ..... 7
**Selection Table** ..... 7
**Block Diagram** ..... 8
**Pin Assignment** ..... 8
**Pin Descriptions** ..... 10
**Absolute Maximum Ratings** ..... 13
**D.C. Electrical Characteristics** ..... 14
Operating Voltage Characteristics ..... 14
Operating Current Characteristics ..... 14
Standby Current Characteristics ..... 14
**A.C. Characteristics** ..... 15
High Speed Internal Oscillator – HIRC – Frequency Accuracy ..... 15
Low Speed Oscillators Characteristics – LIRC ..... 16
Operating Frequency Characteristic Curves ..... 16
System Start Up Time Characteristics ..... 16
**Input/Output Characteristics** ..... 17
**Memory Characteristics** ..... 17
**LVR Electrical Characteristics** ..... 18
**A/D Converter Electrical Characteristics** ..... 18
**Internal Reference Voltage Characteristics** ..... 18
**Power-on Reset Characteristics** ..... 19
**System Architecture** ..... 19
Clocking and Pipelining ..... 19
Program Counter ..... 20
Stack ..... 21
Arithmetic and Logic Unit – ALU ..... 21
**Flash Program Memory** ..... 22
Structure ..... 22
Special Vectors ..... 22
Look-up Table ..... 22
Table Program Example ..... 23
In Circuit Programming – ICP ..... 24
On-Chip Debug Support – OCDS ..... 25

**Data Memory ..... 25**
Structure ..... 25
General Purpose Data Memory ..... 26
Special Purpose Data Memory ..... 26
**Special Function Register Description ..... 29**
Indirect Addressing Registers – IAR0, IAR1 ..... 29
Memory Pointers – MP0, MP1 ..... 29
Bank Pointer – BP ..... 30
Accumulator – ACC ..... 30
Program Counter Low Register – PCL ..... 30
Look-up Table Registers – TBLP, TBHP, TBLH ..... 31
Status Register – STATUS ..... 31
**EEPROM Data Memory ..... 33**
EEPROM Data Memory Structure ..... 33
EEPROM Registers ..... 33
Reading Data from the EEPROM ..... 35
Writing Data to the EEPROM ..... 35
Write Protection ..... 35
EEPROM Interrupt ..... 35
Programming Considerations ..... 36
**Oscillators ..... 37**
Oscillator Overview ..... 37
System Clock Configurations ..... 37
Internal High Speed RC Oscillator – HIRC ..... 38
Internal 32kHz Oscillator – LIRC ..... 38
**Operating Modes and System Clocks ..... 38**
System Clocks ..... 38
System Operation Modes ..... 39
Control Registers ..... 40
Operating Mode Switching ..... 42
Standby Current Considerations ..... 44
Wake-up ..... 45
Programming Considerations ..... 45
**Watchdog Timer ..... 46**
Watchdog Timer Clock Source ..... 46
Watchdog Timer Control Register ..... 46
Watchdog Timer Operation ..... 47
**Reset and Initialisation ..... 48**
Reset Functions ..... 48
Reset Initial Conditions ..... 51
**Input/Output Ports ..... 54**
Pull-high Resistors ..... 54
Port A Wake-up ..... 55

I/O Port Control Registers ..... 55
I/O Port Source Current Selection ..... 56
Pin-remapping Function ..... 58
I/O Pin Structures ..... 59
Programming Considerations ..... 59

**Timer Modules – TM ..... 60**
Introduction ..... 60
TM Operation ..... 60
TM Clock Source ..... 61
TM Interrupts ..... 61
TM External Pins ..... 61
TM Pin Control Register ..... 62
Programming Considerations ..... 63

**Compact Type TM – CTM ..... 65**
Compact Type TM Operation ..... 65
Compact Type TM Register Description ..... 65
Compact Type TM Operation Modes ..... 69

**Periodic Type TM – PTM ..... 75**
Periodic Type TM Operation ..... 75
Periodic Type TM Register Description ..... 75
Periodic Type TM Operation Modes ..... 79

**Analog to Digital Converter ..... 88**
A/D Overview ..... 88
A/D Converter Register Description ..... 88
A/D Converter Data Registers – ADRL, ADRH ..... 89
A/D Converter Control Registers – ADCR0, ADCR1, ACERL ..... 89
A/D Operation ..... 92
A/D Input Pins ..... 93
Summary of A/D Conversion Steps ..... 93
Programming Considerations ..... 94
A/D Transfer Function ..... 95
A/D Programming Example ..... 95

**Touch Key Function ..... 97**
Touch Key Structure ..... 97
Touch Key Register Definition ..... 98
Touch Key Operation ..... 103
Touch Key Interrupt ..... 104
Programming Considerations ..... 104

**Serial Interface Module – SIM ..... 105**
SPI Interface ..... 105
$I^2C$ Interface ..... 113

**Interrupts** ..... **123**
Interrupt Registers ..... 123
Interrupt Operation ..... 127
External Interrupt ..... 129
Touch Key Module Interrupt ..... 129
Serial Interface Module Interrupt ..... 129
Time Base Interrupt ..... 130
EEPROM Interrupt ..... 131
A/D Converter Interrupt ..... 131
Multi-function Interrupts ..... 131
TM Interrupts ..... 131
Interrupt Wake-up Function ..... 132
Programming Considerations ..... 132
**Configuration Options** ..... **133**
**Application Circuits** ..... **133**
**Instruction Set** ..... **134**
Introduction ..... 134
Instruction Timing ..... 134
Moving and Transferring Data ..... 134
Arithmetic Operations ..... 134
Logical and Rotate Operation ..... 135
Branches and Control Transfer ..... 135
Bit Operations ..... 135
Table Read Operations ..... 135
Other Operations ..... 135
**Instruction Set Summary** ..... **136**
Table Conventions ..... 136
**Instruction Definition** ..... **138**
**Package Information** ..... **147**
16-pin NSOP (150mil) Outline Dimensions ..... 148
20-pin SOP (300mil) Outline Dimensions ..... 149
20-pin SSOP (150mil) Outline Dimensions ..... 150
24-pin SOP (300mil) Outline Dimensions ..... 151
24-pin SSOP (150mil) Outline Dimensions ..... 152
28-pin SOP (300mil) Outline Dimensions ..... 153
28-pin SSOP (150mil) Outline Dimensions ..... 154

# Features

## CPU Features

- Operating Voltage:
  - $f_{SYS}$ = 8MHz: 2.2V~5.5V
  - $f_{SYS}$ = 12MHz: 2.7V~5.5V
  - $f_{SYS}$ = 16MHz: 3.3V~5.5V
- Up to 0.25μs instruction cycle with 16MHz system clock at $V_{DD}$=5V
- 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
- Multi-mode operation: FAST, SLOW, IDLE and SLEEP
- Fully integrated internal oscillators require no external components
- All instructions executed in one or two instruction cycles
- Table read instructions
- 61 powerful instructions
- 6-level subroutine nesting
- Bit manipulation instruction

## Peripheral Features

- Flash Program Memory: 3K×16 ~ 4K×16
- Data Memory: 288×8 ~ 512×8
- True EEPROM Memory: 64×8 ~ 128×8
- Up to 12 touch key functions – fully integrated without requiring external components
- Watchdog Timer function
- Up to 26 bidirectional I/O lines
- Programmable I/O port source current for LED driver
- Single external interrupt line shared with I/O pin
- Multiple Timer Modules for time measurement, input capture, compare match output or PWM output or single pulse output function
- Single Time-Base function for generation of fixed time interrupt signals
- 8-channel 12-bit resolution A/D converter
- Serial Interface Module includes SPI and $I^2C$ interfaces
- Low voltage reset function
- Wide range of available package types

## 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/tool-detail/dev_plat/touch/Touch_Workshop

## General Description

The series of devices are Flash Memory type 8-bit high performance RISC architecture microcontrollers with fully integrated touch key functions. With all touch key functions provided internally and with the convenience of Flash Memory multi-programming features, each device has all the features to offer designers a reliable and easy means of implementing Touch Keyes within their products applications.

The touch key functions are fully integrated completely eliminating the need for external components. In addition to the flash program memory, 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 feature includes a multi-channel 12-bit A/D converter. Protective features such as an internal Watchdog Timer and Low Voltage Reset coupled with excellent noise immunity and ESD protection ensure that reliable operation is maintained in hostile electrical environments.

A full choice of internal high and low speed oscillators are provided including fully integrated system oscillators which require no external components for its implementation. The ability to operate and switch dynamically between a range of operating modes using different clock sources gives users the ability to optimise microcontroller operation and minimise power consumption. Easy communication with the outside world is provided using the internal SPI and $I^2C$ interfaces, while the inclusion of flexible I/O programming features, Time-Base function, Timer Modules and many other features further enhance device functionality and flexibility.

The touch key devices will find excellent use in a huge range of modern Touch Key product applications such as instrumentation, household appliances, electronically controlled tools to name but a few.

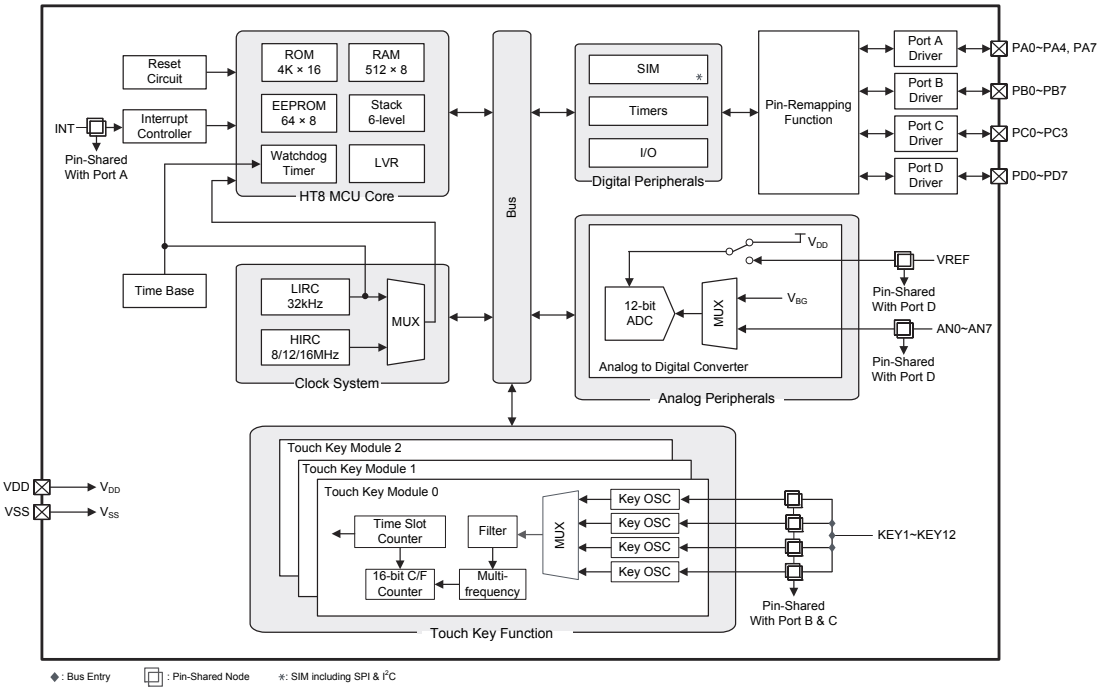
## Selection Table

Most features are common to all devices and the main features distinguishing them are Memory capacity, I/O count, Touch key count, Time Base, Timer Module number and package types. The following table summarises the main features of each device.

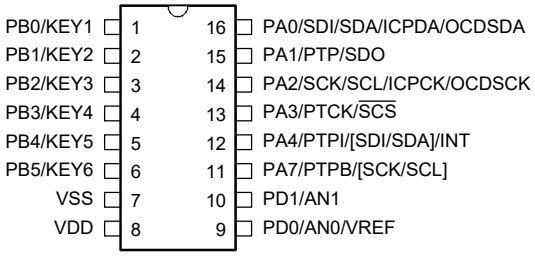
| Part No. | Program Memory | Data Memory | Data EEPROM | I/O | External Interrupt | A/D | Time Base | Timer Module | Touch Key | SIM | Stack | Package |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BS84B08C | 3K×16 | 288×8 | 64×8 | 22 | 1 | 12-bit ×8 | 1 | 10-bit PTM×1 | 8 | √ | 6 | 16NSOP<br>20SOP<br>24SOP |
| BS84C12C | 4K×16 | 512×8 | 128×8 | 26 | 1 | 12-bit ×8 | 1 | 10-bit CTM×1<br>10-bit PTM×1 | 12 | √ | 6 | 20SOP/SSOP<br>24SOP/SSOP<br>28SOP/SSOP |

Note: As devices exist in more than one package format, the table reflects the situation for the package with the most pins.

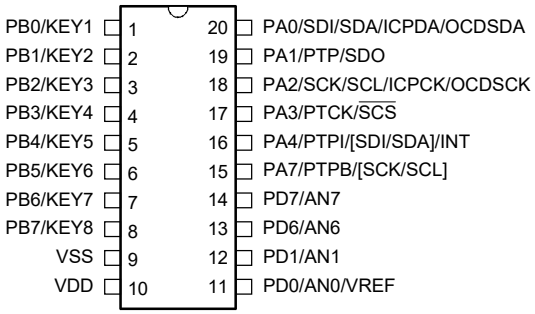
## Block Diagram



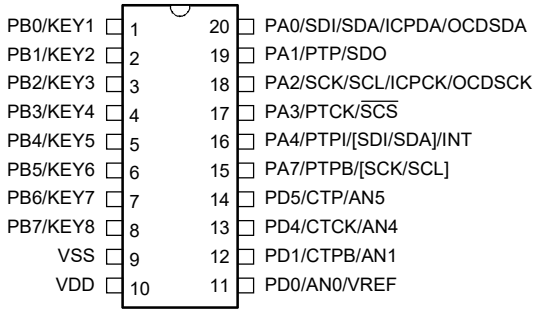

## Pin Assignment



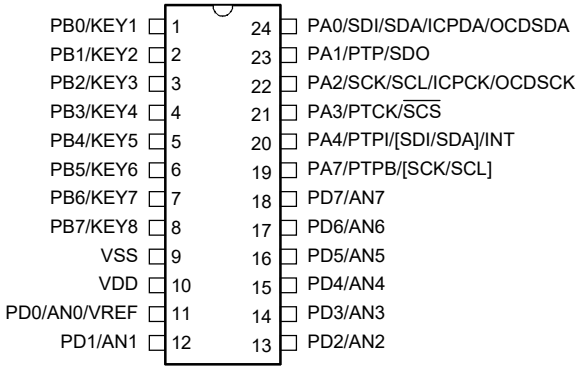

**BS84B08C/BS84BV08C**
**16 NSOP-A**



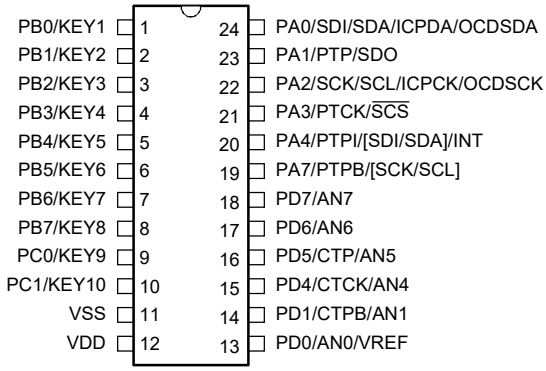

**BS84B08C/BS84BV08C**
**20 SOP-A**



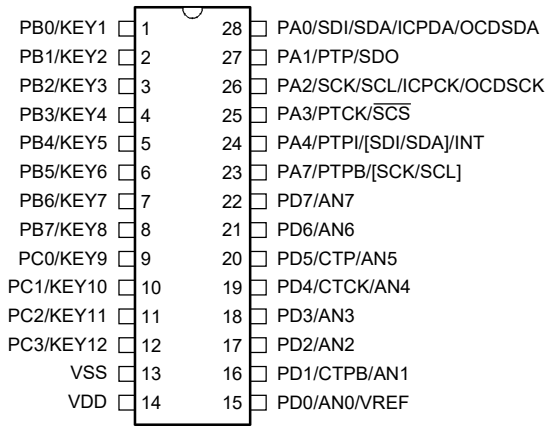

**BS84C12C/BS84CV12C**
**20 SOP-A/SSOP-A**




**BS84B08C/BS84BV08C**
**24 SOP-A**




**BS84C12C/BS84CV12C**
**24 SOP-A/SSOP-A**




**BS84C12C/BS84CV12C**
**28 SOP-A/SSOP-A**

Note: 1. Bracketed pin names indicate non-default pinout remapping locations.

2. If the pin-shared pin functions have multiple outputs simultaneously, its pin names at the right side of the "/" sign can be used for higher priority.

3. The OCDSDA and OCDSCK pins are supplied for the OCDS dedicated pins and as such only available for the BS84BV08C/BS84CV12C devices which are the OCDS EV chips for the BS84B08C/BS84C12C devices respectively.

4. For less pin-count package types there will be unbonded pins which should be properly configured to avoid unwanted current consumption resulting from floating input conditions. Refer to the "Standby Current Considerations" and "Input/Output Ports" sections.

## Pin Descriptions

With the exception of the power pins, all pins on the devices can be referenced by their Port name, e.g. PA0, PA1 etc., which refer to the digital I/O function of the pins. However these Port pins are also shared with other function such as the Touch Key function, Timer Module pins etc. The function of each pin is listed in the following table, however the details behind how each pin is configured is contained in other sections of the datasheet.

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.

**BS84B08C**

| Pin Name | Function | OPT | I/T | O/T | Description |
|---|---|---|---|---|---|
| PA0/SDI/SDA/ ICPDA/OCDSDA | PA0 | PAPU PAWU | ST | CMOS | General purpose I/O. Register enabled pull-up and wake-up |
| | SDI | SIMC0 PXRM | ST | — | SPI serial data input |
| | SDA | SIMC0 PXRM | ST | NMOS | $I^2C$ data line |
| | ICPDA | — | ST | CMOS | ICP data/address pin |
| | OCDSDA | — | ST | CMOS | OCDS data/address pin, for EV chip only |
| PA1/PTP/SDO | PA1 | PAPU PAWU | ST | CMOS | General purpose I/O. Register enabled pull-up and wake-up |
| | PTP | PXRM | — | CMOS | PTM output |
| | SDO | SIMC0 | — | CMOS | SPI serial data output |

| Pin Name | Function | OPT | I/T | O/T | Description |
|---|---|---|---|---|---|
| PA2/SCK/SCL/ ICPCK/OCDSCK | PA2 | PAPU PAWU | ST | CMOS | General purpose I/O. Register enabled pull-up and wake-up |
| | SCK | SIMC0 PXRM | ST | CMOS | SPI serial clock |
| | SCL | SIMC0 PXRM | ST | NMOS | $I^2C$ clock line |
| | ICPCK | — | ST | — | ICP clock pin |
| | OCDSCK | — | ST | — | OCDS clock pin, for EV chip only |
| PA3/PTCK/$\overline{SCS}$ | PA3 | PAPU PAWU | ST | CMOS | General purpose I/O. Register enabled pull-up and wake-up |
| | PTCK | PTMC0 | ST | — | PTM clock input |
| | $\overline{SCS}$ | SIMC0 SIMC2 | ST | CMOS | SPI slave chip select |
| PA4/PTPI/ [SDI/SDA]/INT | PA4 | PAPU PAWU | ST | CMOS | General purpose I/O. Register enabled pull-up and wake-up |
| | PTPI | PTMC0 PTMC1 | ST | — | PTM capture input |
| | SDI | SIMC0 PXRM | ST | — | SPI serial data input |
| | SDA | SIMC0 PXRM | ST | NMOS | $I^2C$ data line |
| | INT | INTC0 | ST | — | External interrupt input |
| PA7/PTPB/ [SCK/SCL] | PA7 | PAPU PAWU | ST | CMOS | General purpose I/O. Register enabled pull-up and wake-up |
| | PTPB | PXRM | — | CMOS | PTM inverted output |
| | SCK | SIMC0 PXRM | ST | CMOS | SPI serial clock |
| | SCL | SIMC0 PXRM | ST | NMOS | $I^2C$ clock line |
| PB0/KEY1~ PB3/KEY4 | PB0~PB3 | PBPU | ST | CMOS | General purpose I/O. Register enabled pull-up |
| | KEY1~KEY4 | TKM0C1 | NSI | — | Touch key input |
| PB4/KEY5~ PB7/KEY8 | PB4~PB7 | PBPU | ST | CMOS | General purpose I/O. Register enabled pull-up |
| | KEY5~KEY8 | TKM1C1 | NSI | — | Touch key input |
| PD0/AN0/VREF | PD0 | PDPU | ST | CMOS | General purpose I/O. Register enabled pull-up |
| | AN0 | ACERL | AN | — | A/D Converter external input channel |
| | VREF | ADCR1 | AN | — | A/D Converter external reference voltage input |
| PD1/AN1~ PD7/AN7 | PD1~PD7 | PDPU | ST | CMOS | General purpose I/O. Register enabled pull-up |
| | AN1~AN7 | ACERL | AN | — | A/D Converter external input channel |
| VDD | VDD | — | PWR | — | Positive power supply |
| VSS | VSS | — | PWR | — | Negative power supply, ground |

Legend: I/T: Input type; O/T: Output type;
OPT: Optional by register selection;
PWR: Power; ST: Schmitt Trigger input;
CMOS: CMOS output; NMOS: NMOS output;
AN: Analog signal; NSI: Non-standard input.

**BS84C12C**

| Pin Name | Function | OPT | I/T | O/T | Description |
|---|---|---|---|---|---|
| PA0/SDI/SDA/ ICPDA/OCDSDA | PA0 | PAPU PAWU | ST | CMOS | General purpose I/O. Register enabled pull-up and wake-up |
| | SDI | SIMC0 PXRM | ST | — | SPI serial data input |
| | SDA | SIMC0 PXRM | ST | NMOS | $I^2C$ data line |
| | ICPDA | — | ST | CMOS | ICP data/address pin |
| | OCDSDA | — | ST | CMOS | OCDS data/address pin, for EV chip only |
| PA1/PTP/SDO | PA1 | PAPU PAWU | ST | CMOS | General purpose I/O. Register enabled pull-up and wake-up |
| | PTP | PXRM | — | CMOS | PTM output |
| | SDO | SIMC0 | — | CMOS | SPI serial data output |
| PA2/SCK/SCL/ ICPCK/OCDSCK | PA2 | PAPU PAWU | ST | CMOS | General purpose I/O. Register enabled pull-up and wake-up |
| | SCK | SIMC0 PXRM | ST | CMOS | SPI serial clock |
| | SCL | SIMC0 PXRM | ST | NMOS | $I^2C$ clock line |
| | ICPCK | — | ST | — | ICP clock pin |
| | OCDSCK | — | ST | — | OCDS clock pin, for EV chip only |
| PA3/PTCK/$\overline{SCS}$ | PA3 | PAPU PAWU | ST | CMOS | General purpose I/O. Register enabled pull-up and wake-up |
| | PTCK | PTMC0 | ST | — | PTM clock input |
| | $\overline{SCS}$ | SIMC0 SIMC2 | ST | CMOS | SPI slave chip select |
| PA4/PTPI/ [SDI/SDA]/INT | PA4 | PAPU PAWU | ST | CMOS | General purpose I/O. Register enabled pull-up and wake-up |
| | PTPI | PTMC0 PTMC1 | ST | — | PTM capture input |
| | SDI | SIMC0 PXRM | ST | — | SPI serial data input |
| | SDA | SIMC0 PXRM | ST | NMOS | $I^2C$ data line |
| | INT | INTC0 | ST | — | External interrupt input |
| PA7/PTPB/ [SCK/SCL] | PA7 | PAPU PAWU | ST | CMOS | General purpose I/O. Register enabled pull-up and wake-up |
| | PTPB | PXRM | — | CMOS | PTM inverted output |
| | SCK | SIMC0 PXRM | ST | CMOS | SPI serial clock |
| | SCL | SIMC0 PXRM | ST | NMOS | $I^2C$ clock line |
| PB0/KEY1~ PB3/KEY4 | PB0~PB3 | PBPU | ST | CMOS | General purpose I/O. Register enabled pull-up |
| | KEY1~KEY4 | TKM0C1 | NSI | — | Touch key input |
| PB4/KEY5~ PB7/KEY8 | PB4~PB7 | PBPU | ST | CMOS | General purpose I/O. Register enabled pull-up |
| | KEY5~KEY8 | TKM1C1 | NSI | — | Touch key input |
| PC0/KEY9~ PC3/KEY12 | PC0~PC3 | PCPU | ST | CMOS | General purpose I/O. Register enabled pull-up |
| | KEY9~KEY12 | TKM2C1 | NSI | — | Touch key input |
| PD0/AN0/VREF | PD0 | PDPU | ST | CMOS | General purpose I/O. Register enabled pull-up |
| | AN0 | ACERL | AN | — | A/D Converter external input channel |
| | VREF | ADCR1 | AN | — | A/D Converter external reference voltage input |

| Pin Name | Function | OPT | I/T | O/T | Description |
|---|---|---|---|---|---|
| PD1/CTPB/AN1 | PD1 | PDPU | ST | CMOS | General purpose I/O. Register enabled pull-up |
| | CTPB | PXRM | — | CMOS | CTM inverted output |
| | AN1 | ACERL | AN | — | A/D Converter external input channel |
| PD2/AN2 | PD2 | PDPU | ST | CMOS | General purpose I/O. Register enabled pull-up |
| | AN2 | ACERL | AN | — | A/D Converter external input channel |
| PD3/AN3 | PD3 | PDPU | ST | CMOS | General purpose I/O. Register enabled pull-up |
| | AN3 | ACERL | AN | — | A/D Converter external input channel |
| PD4/CTCK/AN4 | PD4 | PDPU | ST | CMOS | General purpose I/O. Register enabled pull-up |
| | CTCK | CTMC0 | ST | — | CTM clock input |
| | AN4 | ACERL | AN | — | A/D Converter external input channel |
| PD5/CTP/AN5 | PD5 | PDPU | ST | CMOS | General purpose I/O. Register enabled pull-up |
| | CTP | PXRM | — | CMOS | CTM output |
| | AN5 | ACERL | AN | — | A/D Converter external input channel |
| PD6/AN6 | PD6 | PDPU | ST | CMOS | General purpose I/O. Register enabled pull-up |
| | AN6 | ACERL | AN | — | A/D Converter external input channel |
| PD7/AN7 | PD7 | PDPU | ST | CMOS | General purpose I/O. Register enabled pull-up |
| | AN7 | ACERL | AN | — | A/D Converter external input channel |
| VDD | VDD | — | PWR | — | Positive power supply |
| VSS | VSS | — | PWR | — | Negative power supply, ground |

Legend: I/T: Input type;
OPT: Optional by register option;
ST: Schmitt Trigger input;
NMOS: NMOS output;
NSI: Non Standard input.
O/T: Output type;
PWR: Power;
CMOS: CMOS output;
AN: Analog signal;

## Absolute Maximum Ratings

Supply Voltage ..... $V_{SS}$−0.3V to $V_{SS}$+6.0V

Input Voltage ..... $V_{SS}$−0.3V to $V_{DD}$+0.3V

Storage Temperature ..... -60°C to 150°C

Operating Temperature ..... -40°C to 85°C

$I_{OH}$ Total ..... -80mA

$I_{OL}$ Total ..... 80mA

Total Power Dissipation ..... 500mW

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

## D.C. Electrical 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, can all exert an influence on the measured values.

### Operating Voltage Characteristics

Ta=-40°C ~ 85°C

| Symbol | Parameter | Test Conditions | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|---|
| $V_{DD}$ | Operating Voltage – HIRC | $f_{SYS}=f_{HIRC}$=8MHz | 2.2 | — | 5.5 | V |
| | | $f_{SYS}=f_{HIRC}$=12MHz | 2.7 | — | 5.5 | |
| | | $f_{SYS}=f_{HIRC}$=16MHz | 3.3 | — | 5.5 | |
| | Operating Voltage – LIRC | $f_{SYS}=f_{HIRC}$=32kHz | 2.2 | — | 5.5 | |

### Operating Current Characteristics

Ta=25°C

| Symbol | Parameter | Test Conditions | | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|---|---|
| | | $V_{DD}$ | Conditions | | | | |
| $I_{DD}$ | SLOW Mode – LIRC | 2.2V | $f_{SYS}=f_{LIRC}$=32kHz, LVR enable | — | 25 | 50 | μA |
| | | 3V | | — | 28 | 56 | |
| | | 5V | | — | 36 | 72 | |
| | FAST Mode – HIRC | 2.2V | $f_{SYS}=f_{HIRC}$=8MHz | — | 0.6 | 1.0 | mA |
| | | 3V | | — | 0.8 | 1.2 | |
| | | 5V | | — | 1.6 | 2.4 | |
| | | 2.7V | $f_{SYS}=f_{HIRC}$=12MHz | — | 1.0 | 1.4 | mA |
| | | 3V | | — | 1.2 | 1.8 | |
| | | 5V | | — | 2.4 | 3.6 | |
| | | 3.3V | $f_{SYS}=f_{HIRC}$=16MHz | — | 3.0 | 4.5 | mA |
| | | 5V | | — | 4.0 | 6.0 | |

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

1. Any digital input is 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

| Symbol | Parameter | Test Conditions | | Min. | Typ. | Max. | Max. @85°C | Unit |
|---|---|---|---|---|---|---|---|---|
| | | $V_{DD}$ | Conditions | | | | | |
| $I_{STB}$ | SLEEP Mode | 2.2V | WDT on | — | 1.2 | 2.4 | 2.9 | μA |
| | | 3V | | — | 1.5 | 3.0 | 3.6 | |
| | | 5V | | — | 3.0 | 5.0 | 6.0 | |
| | IDLE0 Mode – LIRC | 2.2V | $f_{SUB}$ on | — | 2.4 | 4.0 | 4.8 | μA |
| | | 3V | | — | 3.0 | 5.0 | 6.0 | |
| | | 5V | | — | 5.0 | 10 | 12 | |

| Symbol | Parameter | Test Conditions | | Min. | Typ. | Max. | Max. @85°C | Unit |
|---|---|---|---|---|---|---|---|---|
| | | $V_{DD}$ | Conditions | | | | | |
| $I_{STB}$ | IDLE1 Mode – HIRC | 2.2V | $f_{SUB}$ on, $f_{SYS}$=8MHz | — | 288 | 400 | 480 | μA |
| | | 3V | | — | 360 | 500 | 600 | |
| | | 5V | | — | 600 | 800 | 960 | |
| | | 2.7V | $f_{SUB}$ on, $f_{SYS}$=12MHz | — | 432 | 600 | 720 | μA |
| | | 3V | | — | 540 | 750 | 900 | |
| | | 5V | | — | 800 | 1200 | 1440 | |
| | | 3.3V | $f_{SUB}$ on, $f_{SYS}$=16MHz | — | 1.1 | 1.6 | 1.9 | mA |
| | | 5V | | — | 1.4 | 2.0 | 2.4 | |

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

1. Any digital input is 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 executed thus stopping all instruction execution.

## A.C. Characteristics

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

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

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

| Symbol | Parameter | Test Conditions | | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|---|---|
| | | $V_{DD}$ | Temp. | | | | |
| $f_{HIRC}$ | 8MHz Writer Trimmed HIRC Frequency | 3V/5V | 25°C | -1% | 8 | +1% | MHz |
| | | | -40°C~85°C | -2% | 8 | +2% | |
| | | 2.2V~5.5V | 25°C | -2.5% | 8 | +2.5% | |
| | | | -40°C~85°C | -3% | 8 | +3% | |
| | 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 VDD range operating voltage. It is recommended that the trim voltage is fixed at 3V for application voltage ranges from 2.2V to 3.6V and fixed at 5V for application voltage ranges from 3.3V to 5.5V.

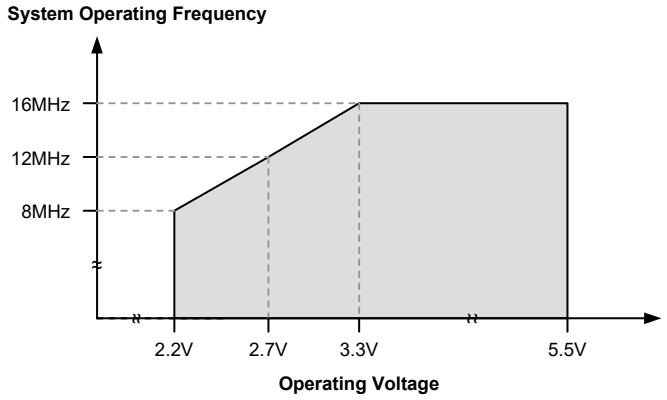
3. The minimum and maximum tolerance values provided in the table are only for the frequency at which the writer trims the HIRC oscillator. After trimming at this chosen specific frequency any change in HIRC oscillator frequency using the oscillator register control bits by the application program will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

## Low Speed Oscillators Characteristics – LIRC

Ta=25°C, unless otherwise specified

| Symbol | Parameter | Test Conditions | | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|---|---|
| | | $V_{DD}$ | Temp. | | | | |
| $f_{LIRC}$ | LIRC Frequency | 2.2V~5.5V | 25°C | -10% | 32 | +10% | kHz |
| | | | -40°C~85°C | -50% | 32 | +60% | |
| $t_{START}$ | LIRC Start Up Time | — | — | — | — | 500 | μs |

## 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<br>Wake-up from condition where $f_{SYS}$ is off | — | $f_{SYS}=f_H$~$f_H/64$, $f_H=f_{HIRC}$ | — | 16 | — | $t_{HIRC}$ |
| | | — | $f_{SYS}=f_{SUB}=f_{LIRC}$ | — | 2 | — | $t_{LIRC}$ |
| | System Start-up Time<br>Wake-up from condition where $f_{SYS}$ is on | — | $f_{SYS}=f_H$~$f_H/64$, $f_H=f_{HIRC}$ | — | 2 | — | $t_H$ |
| | | — | $f_{SYS}=f_{SUB}=f_{LIRC}$ | — | 2 | — | $t_{SUB}$ |
| | System Speed Switch Time<br>FAST to SLOW Mode or<br>SLOW to FAST Mode | — | $f_{HIRC}$ switches from off → on | — | 16 | — | $t_{HIRC}$ |
| $t_{RSTD}$ | System Reset Delay Time<br>Reset source from Power-on reset or<br>LVR hardware reset | — | $RR_{POR}$=5V/ms | 42 | 48 | 54 | ms |
| | System Reset Delay Time<br>LVRC/WDTC software reset | — | — | | | | |
| | System Reset Delay Time<br>Reset source from WDT overflow | — | — | 14 | 16 | 18 | ms |

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

Ta=25°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 I/O Ports | 3V | $V_{OL}=0.1V_{DD}$ | 16 | 32 | — | mA |
| | | 5V | | 32 | 65 | — | |
| $I_{OH}$ | Source Current for I/O Ports | 3V | $V_{OH}=0.9V_{DD}$, SLEDCn[m+1:m]=00B (n=0, 1; m=0, 2, 4 or 6) | -0.7 | -1.5 | — | mA |
| | | 5V | | -1.5 | -2.9 | — | |
| | | 3V | $V_{OH}=0.9V_{DD}$, SLEDCn[m+1:m]=01B (n=0, 1; m=0, 2, 4 or 6) | -1.3 | -2.5 | — | |
| | | 5V | | -2.5 | -5.1 | — | |
| | | 3V | $V_{OH}=0.9V_{DD}$, SLEDCn[m+1:m]=10B (n=0, 1; m=0, 2, 4 or 6) | -1.8 | -3.6 | — | |
| | | 5V | | -3.6 | -7.3 | — | |
| | | 3V | $V_{OH}=0.9V_{DD}$, SLEDCn[m+1:m]=11B (n=0, 1; m=0, 2, 4 or 6) | -4 | -8 | — | |
| | | 5V | | -8 | -16 | — | |
| $R_{PH}$ | Pull-high Resistance for I/O Ports (Note) | 3V | — | 20 | 60 | 100 | kΩ |
| | | 5V | | 10 | 30 | 50 | |
| $I_{LEAK}$ | Input Leakage Current | 5V | $V_{IN}=V_{DD}$ or $V_{IN}=V_{SS}$ | — | — | ±1 | μA |
| $t_{TCK}$ | TM Clock Input Pin Minimum Pulse Width | — | — | 0.3 | — | — | μs |
| $t_{TPI}$ | TM Capture Input Pin Minimum Pulse Width | — | — | 0.3 | — | — | μs |
| $t_{INT}$ | External Interrupt Minimum Pulse Width | — | — | 1 | 5 | 10 | μs |

Note: 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 input sink current at the specified supply voltage level. Dividing the voltage by this measured current provides the $R_{PH}$ value.

## Memory Characteristics

Ta=-40°C~85°C

| Symbol | Parameter | Test Conditions | | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|---|---|
| | | $V_{DD}$ | Conditions | | | | |
| $V_{RW}$ | $V_{DD}$ for Read/Write | — | — | $V_{DDmin}$ | — | $V_{DDmax}$ | V |
| **Flash Program/Data EEPROM Memory** | | | | | | | |
| $t_{DEW}$ | Erase/Write Cycle Time – Flash Program Memory | — | — | — | 2 | 3 | ms |
| | Write Cycle Time – Data EEPROM Memory | — | — | — | 4 | 6 | ms |
| $I_{DDPGM}$ | Programming/Erase Current on $V_{DD}$ | — | — | — | — | 5.0 | mA |
| $E_P$ | Cell Endurance – Program Flash Memory | — | — | 10K | — | — | E/W |
| | Cell Endurance – Data EEPROM Memory | — | — | 100K | — | — | E/W |
| $t_{RETD}$ | ROM Data Retention Time | — | Ta=25°C | — | 40 | — | Year |
| **RAM Data Memory** | | | | | | | |
| $V_{DR}$ | RAM Data Retention Voltage | — | Device in SLEEP Mode | 1.0 | — | — | V |

Note: "E/W" means Erase/Write times.

## LVR Electrical Characteristics

Ta=25°C

| Symbol | Parameter | Test Conditions | | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|---|---|
| | | $V_{DD}$ | Conditions | | | | |
| $V_{LVR}$ | Low Voltage Reset Voltage | — | LVR enable, voltage select 2.1V | -5% | 2.1 | +5% | V |
| | | — | LVR enable, voltage select 2.55V | | 2.55 | | |
| | | — | LVR enable, voltage select 3.15V | | 3.15 | | |
| | | — | LVR enable, voltage select 3.8V | | 3.8 | | |
| $t_{LVR}$ | Minimum Low Voltage Width to Reset | — | — | 120 | 240 | 480 | μs |

## A/D Converter Electrical Characteristics

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

| Symbol | Parameter | Test Conditions | | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|---|---|
| | | $V_{DD}$ | Conditions | | | | |
| $V_{DD}$ | Operating Voltage | — | — | 2.2 | — | 5.5 | V |
| $V_{ADI}$ | Input Voltage | — | — | 0 | — | $V_{REF}$ | V |
| $V_{REF}$ | Reference Voltage | — | — | 2 | — | $V_{DD}$ | V |
| DNL | Differential Non-linearity | 2.2V | $V_{REF}$=$V_{DD}$, $t_{ADCK}$=0.5μs | -3 | — | +3 | LSB |
| | | 3V | | | | | |
| | | 5V | | | | | |
| INL | Integral Non-linearity | 2.2V | $V_{REF}$=$V_{DD}$, $t_{ADCK}$=0.5μs | -4 | — | +4 | LSB |
| | | 3V | | | | | |
| | | 5V | | | | | |
| $I_{ADC}$ | Additional Current Consumption for A/D Converter Enable | 2.2V | No load, $t_{ADCK}$=0.5μs | — | 1 | 2 | mA |
| | | 3V | | — | 1 | 2 | mA |
| | | 5V | | — | 1.5 | 3 | mA |
| $t_{ADCK}$ | Clock Period | — | — | 0.5 | — | 10 | μs |
| $t_{ON2ST}$ | A/D Converter On-to-Start Time | — | — | 4 | — | — | μs |
| $t_{ADS}$ | Sampling time | — | — | — | 4 | — | $t_{ADCK}$ |
| $t_{ADC}$ | Conversion Time (Including A/D Sample and Hold Time) | — | — | — | 16 | — | $t_{ADCK}$ |

## Internal Reference Voltage Characteristics

Ta=25°C, unless otherwise specified

| Symbol | Parameter | Test Conditions | | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|---|---|
| | | $V_{DD}$ | Conditions | | | | |
| $V_{BG}$ | Bandgap Reference Voltage | 3.15V | — | -3% | 1.19 | +3% | V |

Note: 1. All the above parameters are measured under conditions of no load condition unless otherwise described.

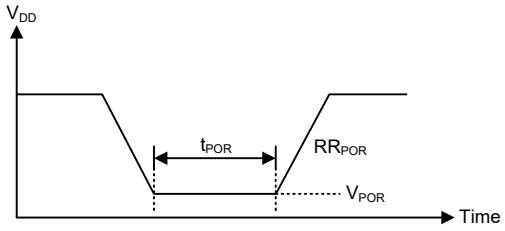
2. A 0.1μF ceramic capacitor should be connected between VDD and GND.

3. The $V_{BG}$ voltage is used as the A/D converter internal signal input.

## 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 |

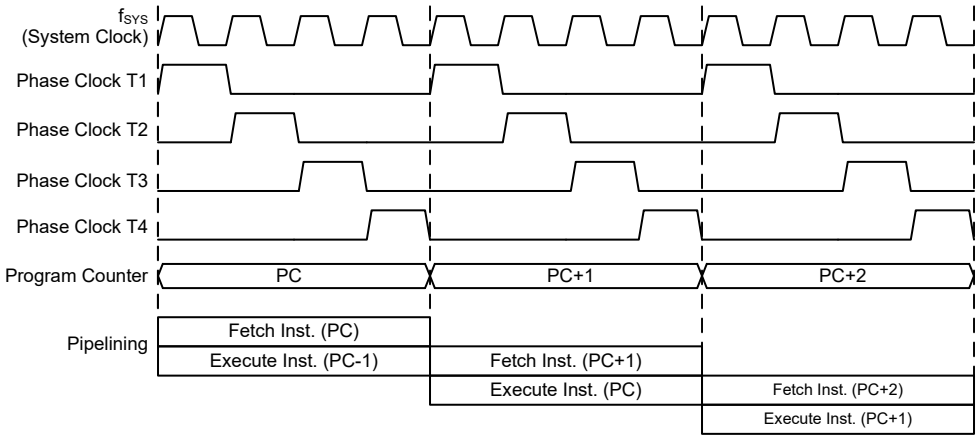



## System Architecture

A key factor in the high-performance features of the range of microcontrollers is attributed to their internal system architecture. These devices take advantage of the usual features found within RISC microcontrollers providing increased speed of operation and enhanced performance. The pipelining scheme is implemented in such a way that instruction fetching and instruction execution are overlapped, hence instructions are effectively executed in one cycle, with the exception of branch or call instructions. 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 these devices suitable for low-cost, high-volume production for controller applications.

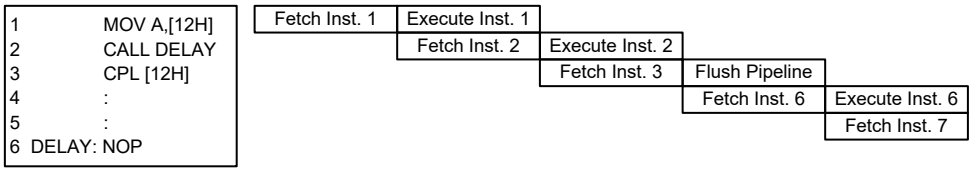
### Clocking and Pipelining

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




**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. 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.

| Device | Program Counter | |
|---|---|---|
| | High Byte | Low Byte (PCL) |
| BS84B08C | PC11~PC8 | PCL7~PCL0 |
| BS84C12C | PC11~PC8 | PCL7~PCL0 |

**Program Counter**

The lower byte of the Program Counter, known as the Program Counter Low register or PCL, is available for program control and is a readable and writeable register. By transferring data directly into this register, a short program jump can be executed directly; however, as only this low byte is available for manipulation, the jumps are limited to the present page of memory that is 256

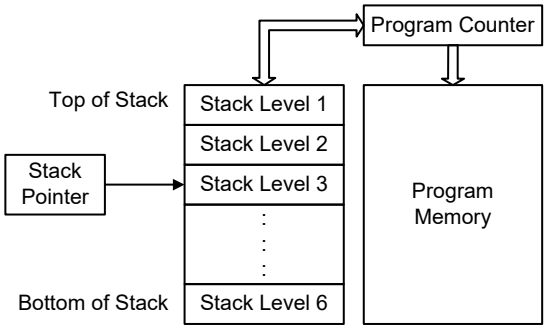
locations. When such program jumps are executed it should also be noted that a dummy cycle will be inserted. Manipulating the PCL register may cause program branching, so an extra cycle is needed to pre-fetch.

## Stack

This is a special part of the memory which is used to save the contents of the Program Counter only. The stack is organized into 6 levels and neither part of the data nor part of the program space, and is neither readable nor writeable. The activated level is indexed by the Stack Pointer, and is neither readable nor writeable. At a subroutine call or interrupt acknowledge signal, the contents of the Program Counter are pushed onto the stack. At the end of a subroutine or an interrupt routine, signaled by a return instruction, RET or RETI, the Program Counter is restored to its previous value from the stack. After a device reset, the Stack Pointer will point to the top of the stack.

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

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




## Arithmetic and Logic Unit – ALU

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

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

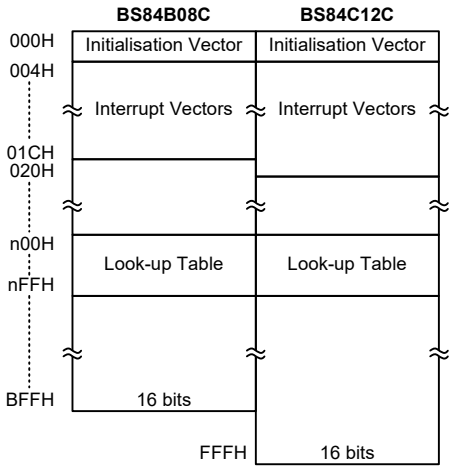
## Flash Program Memory

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

| Device | Capacity |
|---|---|
| BS84B08C | 3K×16 |
| BS84C12C | 4K×16 |

### Structure

The Program Memory has a capacity of 3K×16 to 4K×16 bits. The Program Memory is addressed by the Program Counter and also contains data, table information and interrupt entries. Table data, which can be setup in any location within the Program Memory, is addressed by a separate table pointer register.




**Program Memory Structure**

### Special Vectors

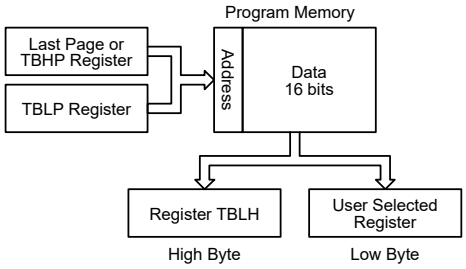
Within the Program Memory, certain locations are reserved for the reset and interrupts. The location 0000H is reserved for use by the device reset for program initialisation. After a device reset is initiated, the program will jump to this location and begin execution.

### 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 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 "0B00H" which refers to the start address of the last page within the 3K Program Memory of the BS84B08C device. The table pointer low byte register is setup here to have an initial value of "06H". This will ensure that the first data read from the data table will be at the Program Memory address "0B06H" or 6 locations after the start of the last page. Note that the value for the table pointer is referenced to the first address specified by TBLP and TBHP if the "TABRD [m]" instruction is being used. The high byte of the table data which in this case is equal to zero will be transferred to the TBLH register automatically when the "TABRD [m]" instruction is executed.

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

### Table Read Program Example

```
tempreg1 db ?      ; temporary register #1
tempreg2 db ?      ; temporary register #2
:
:
mov a,06h          ; initialise low table pointer - note that this address
                   ; is referenced
mov tblp,a         ; to the last page or the page that tbhp pointed
mov a,0Bh          ; initialise high table pointer
mov tbhp,a
:
:
tabrd tempreg1     ; transfers value in table referenced by table pointer data at program
                   ; memory address "0B06H" 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 "0B05H" transferred to
                   ; tempreg2 and TBLH in this example the data "1AH" is
                   ; transferred to tempreg1 and data "0FH" to register tempreg2
:
:
org 0B00h          ; sets initial address of program memory
dc 00Ah, 00Bh, 00Ch, 00Dh, 00Eh, 00Fh, 01Ah, 01Bh
:
:
```

## In Circuit Programming – ICP

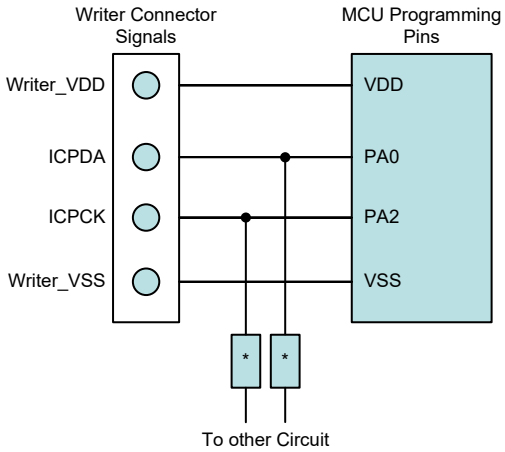
The provision of Flash type Program Memory provides the user with a means of convenient and easy upgrades and modifications to their programs on the same device.

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

| 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 devices is beyond the scope of this document and will be supplied in supplementary literature.

During the programming process, the user must take care of the PA0 and PA2 pins for data and clock programming purposes to ensure that no other outputs are connected to these two pins.




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

### On-Chip Debug Support – OCDS

There are EV chips named BS84BV08C and BS84CV12C which are used to emulate the real MCU devices named BS84B08C and BS84C12C respectively. The EV chip devices also provide an "On-Chip Debug" function to debug the real MCU devices during the development process. The EV chips and the real MCU devices are almost functionally compatible except for "On-Chip Debug" function. Users can use the EV chip devices to emulate the real chip devices behavior by connecting the OCDSDA and OCDSCK pins to the Holtek HT-IDE development tools. The OCDSDA pin is the OCDS Data/Address input/output pin while the OCDSCK pin is the OCDS clock input pin. When users use the EV chip devices for debugging, the corresponding pin functions shared with the OCDSDA and OCDSCK pins in the real MCU devices will have no effect in the EV chip. However, the two OCDS pins which are pin-shared with the ICP programming pins are still used as the Flash Memory programming pins for ICP. For more detailed OCDS information, refer to the corresponding document named "Holtek e-Link for 8-bit MCU OCDS User's Guide".

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

## Data Memory

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

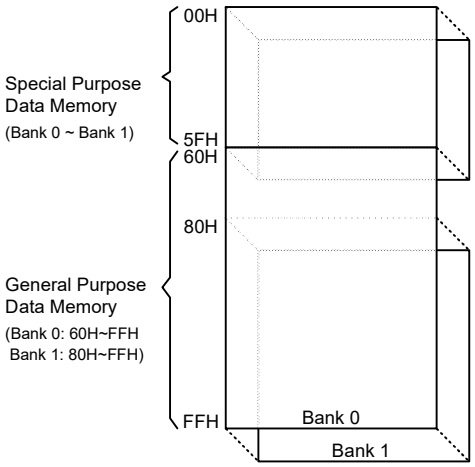
Categorized into two types, the first of these is an area of RAM where special function registers are located. These registers have fixed locations and are necessary for correct operation of the devices. Many of these registers can be read from and written to directly under program control, however, 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.

### Structure

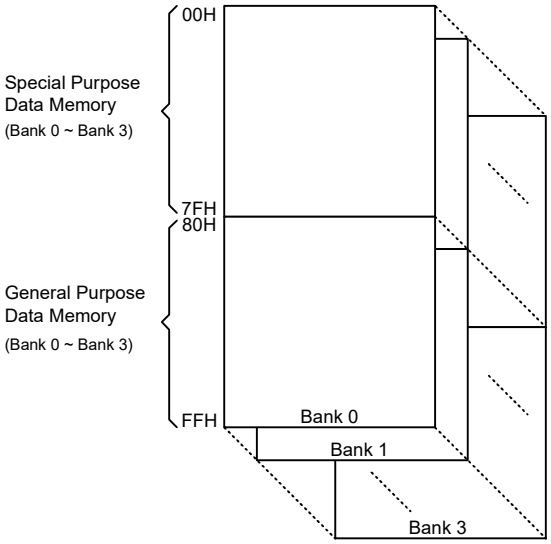
The overall Data Memory is subdivided into several banks. The Special Purpose Data Memory registers are accessible in all banks, with the exception of the EEC register at address 40H, which is only accessible in Bank 1. Switching between the different Data Memory banks is achieved by setting the Bank Pointer to the correct value. The start address of the Data Memory for the devices is the address 00H.

| Device | Special Purpose Data Memory | General Purpose Data Memory | |
|---|---|---|---|
| | Available Banks | Capacity | Bank: Address |
| BS84B08C | Bank 0 ~ Bank 1 | 288×8 | Bank 0: 60H~FFH<br>Bank 1: 80H~FFH |
| BS84C12C | Bank 0 ~ Bank 3 | 512×8 | Bank 0: 80H~FFH<br>Bank 1: 80H~FFH<br>Bank 2: 80H~FFH<br>Bank 3: 80H~FFH |

**Data Memory Summary**




**Data Memory Structure – BS84B08C**




**Data Memory Structure – BS84C12C**

## 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 programing for both reading and writing operations. By using the bit operation instructions individual bits can be set or reset under program control giving the user a large range of flexibility for bit manipulation in the Data Memory.

## Special Purpose Data Memory

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

| | Bank 0, 1 |
|---|---|
| 00H | IAR0 |
| 01H | MP0 |
| 02H | IAR1 |
| 03H | MP1 |
| 04H | BP |
| 05H | ACC |
| 06H | PCL |
| 07H | TBLP |
| 08H | TBLH |
| 09H | TBHP |
| 0AH | STATUS |
| 0BH | SMOD |
| 0CH | CTRL |
| 0DH | INTEG |
| 0EH | INTC0 |
| 0FH | INTC1 |
| 10H | |
| 11H | |
| 12H | MFI0 |
| 13H | LVRC |
| 14H | PA |
| 15H | PAC |
| 16H | PAPU |
| 17H | PAWU |
| 18H | PXRM |
| 19H | |
| 1AH | WDTC |
| 1BH | TBC |
| 1CH | PSCR |
| 1DH | |
| 1EH | EEA |
| 1FH | EED |
| 20H | PB |
| 21H | PBC |
| 22H | PBPU |
| 23H | SIMTOC |
| 24H | SIMC0 |
| 25H | SIMC1 |
| 26H | SIMD |
| 27H | SIMC2/SIMA |
| 28H | |
| 29H | |
| 2AH | |
| 2BH | |
| 2CH | |
| 2DH | |
| 2EH | ADRL |
| 2FH | ADRH |

| | Bank 0 | Bank 1 |
|---|---|---|
| 30H | ADCR0 | |
| 31H | ADCR1 | |
| 32H | ACERL | |
| 33H | SLEDC0 | |
| 34H | SLEDC1 | |
| 35H | PD | |
| 36H | PDC | |
| 37H | PDPU | |
| 38H | | |
| 39H | | |
| 3AH | | |
| 3BH | PTMC0 | |
| 3CH | PTMC1 | |
| 3DH | PTMDL | |
| 3EH | PTMDH | |
| 3FH | PTMAL | |
| 40H | PTMAH | EEC |
| 41H | PTMRPL | |
| 42H | PTMRPH | |
| 43H | TKTMR | |
| 44H | TKC0 | |
| 45H | TK16DL | |
| 46H | TK16DH | |
| 47H | TKC1 | |
| 48H | TKM016DL | |
| 49H | TKM016DH | |
| 4AH | TKM0ROL | |
| 4BH | TKM0ROH | |
| 4CH | TKM0C0 | |
| 4DH | TKM0C1 | |
| 4EH | TKM116DL | |
| 4FH | TKM116DH | |
| 50H | TKM1ROL | |
| 51H | TKM1ROH | |
| 52H | TKM1C0 | |
| 53H | TKM1C1 | |
| 54H | | |
| 55H | | |
| 56H | | |
| 57H | | |
| 58H | | |
| 59H | | |
| 5AH | | |
| 5BH | | |
| 5CH | | |
| 5DH | | |
| 5EH | | |
| 5FH | | |

: Unused, read as 00H

**Special Purpose Data Memory Structure – BS84B08C**

| | Bank 0, 1, 2, 3 |
|---|---|
| 00H | IAR0 |
| 01H | MP0 |
| 02H | IAR1 |
| 03H | MP1 |
| 04H | BP |
| 05H | ACC |
| 06H | PCL |
| 07H | TBLP |
| 08H | TBLH |
| 09H | TBHP |
| 0AH | STATUS |
| 0BH | SMOD |
| 0CH | CTRL |
| 0DH | INTEG |
| 0EH | INTC0 |
| 0FH | INTC1 |
| 10H | INTC2 |
| 11H | |
| 12H | MFI0 |
| 13H | LVRC |
| 14H | PA |
| 15H | PAC |
| 16H | PAPU |
| 17H | PAWU |
| 18H | PXRM |
| 19H | |
| 1AH | WDTC |
| 1BH | TBC |
| 1CH | PSCR |
| 1DH | MFI1 |
| 1EH | EEA |
| 1FH | EED |
| 20H | PB |
| 21H | PBC |
| 22H | PBPU |
| 23H | SIMTOC |
| 24H | SIMC0 |
| 25H | SIMC1 |
| 26H | SIMD |
| 27H | SIMC2/SIMA |
| 28H | CTMC0 |
| 29H | CTMC1 |
| 2AH | CTMDL |
| 2BH | CTMDH |
| 2CH | CTMAL |
| 2DH | CTMAH |
| 2EH | ADRL |
| 2FH | ADRH |
| 30H | ADCR0 |
| 31H | ADCR1 |
| 32H | ACERL |
| 33H | SLEDC0 |
| 34H | SLEDC1 |
| 35H | PD |
| 36H | PDC |
| 37H | PDPU |
| 38H | PC |
| 39H | PCC |
| 3AH | PCPU |
| 3BH | PTMC0 |
| 3CH | PTMC1 |
| 3DH | PTMDL |
| 3EH | PTMDH |
| 3FH | PTMAL |

| | Bank 0, 2, 3 | Bank 1 |
|---|---|---|
| 40H | PTMAH | EEC |
| 41H | PTMRPL | |
| 42H | PTMRPH | |
| 43H | TKTMR | |
| 44H | TKC0 | |
| 45H | TK16DL | |
| 46H | TK16DH | |
| 47H | TKC1 | |
| 48H | TKM016DL | |
| 49H | TKM016DH | |
| 4AH | TKM0ROL | |
| 4BH | TKM0ROH | |
| 4CH | TKM0C0 | |
| 4DH | TKM0C1 | |
| 4EH | TKM116DL | |
| 4FH | TKM116DH | |
| 50H | TKM1ROL | |
| 51H | TKM1ROH | |
| 52H | TKM1C0 | |
| 53H | TKM1C1 | |
| 54H | TKM216DL | |
| 55H | TKM216DH | |
| 56H | TKM2ROL | |
| 57H | TKM2ROH | |
| 58H | TKM2C0 | |
| 59H | TKM2C1 | |
| 5AH | | |
| 5BH | | |
| 5CH | | |
| 5DH | | |
| 5EH | | |
| 5FH | | |
| 60H | | |
| 61H | | |
| 62H | | |
| 63H | | |
| 64H | | |
| 65H | | |
| 66H | | |
| 67H | | |
| 68H | | |
| 69H | | |
| 6AH | | |
| 6BH | | |
| 6CH | | |
| 6DH | | |
| 6EH | | |
| 6FH | | |
| 70H | | |
| 71H | | |
| 72H | | |
| 73H | | |
| 74H | | |
| 75H | | |
| 76H | | |
| 77H | | |
| 78H | | |
| 79H | | |
| 7AH | | |
| 7BH | | |
| 7CH | | |
| 7DH | | |
| 7EH | | |
| 7FH | | |

: Unused, read as 00H

**Special Purpose Data Memory Structure – BS84C12C**

## Special Function Register Description

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

### Indirect Addressing Registers – IAR0, IAR1

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

### Memory Pointers – MP0, MP1

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

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

#### Indirect Addressing Program Example

```
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:
```

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

### Bank Pointer – BP

Depending upon which device is used, the Data Memory is divided into several banks. Selecting the required Data Memory area is achieved using the Bank Pointer.

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

- **BP Register – BS84B08C**

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

Bit 7~1 Unimplemented, read as “0”

Bit 0 **DMBP0**: Select Data Memory Banks
0: Bank 0
1: Bank 1

- **BP Register – BS84C12C**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | — | — | — | — | — | — | DMBP1 | DMBP0 |
| R/W | — | — | — | — | — | — | R/W | R/W |
| POR | — | — | — | — | — | — | 0 | 0 |

Bit 7~2 Unimplemented, read as “0”

Bit 1~0 **DMBP1~DMBP0**: Select Data Memory Banks
00: Bank 0
01: Bank 1
10: Bank 2
11: Bank 3

## Accumulator – ACC

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

## Program Counter Low Register – PCL

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

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

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

## Status Register – STATUS

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

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

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

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

In addition, on entering an interrupt sequence or executing a subroutine call, the status register will not be pushed onto the stack automatically. If the contents of the status 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 | — | — | TO | PDF | OV | Z | AC | C |
| R/W | — | — | R | R | R/W | R/W | R/W | R/W |
| POR | — | — | 0 | 0 | x | x | x | x |

"x": unknown

Bit 7~6 Unimplemented, read as "0"

Bit 5 **TO**: Watchdog Time-out flag
0: After power up or executing the "CLR WDT" or "HALT" instruction
1: A watchdog time-out occurred.

Bit 4 **PDF**: Power down flag
0: After power up or executing the "CLR WDT" instruction
1: By executing the "HALT" instruction

Bit 3 **OV**: Overflow flag
0: No overflow
1: An operation results in a carry into the highest-order bit but not a carry out of the highest-order bit or vice versa.

Bit 2 **Z**: Zero flag
0: The result of an arithmetic or logical operation is not zero
1: The result of an arithmetic or logical operation is zero

Bit 1 **AC**: Auxiliary flag
0: No auxiliary carry
1: An operation results in a carry out of the low nibbles in addition, or no borrow from the high nibble into the low nibble in subtraction

Bit 0 **C**: Carry flag
0: No carry-out
1: An operation results in a carry during an addition operation or if a borrow does not take place during a subtraction operation

The "C" flag is also affected by a rotate through carry instruction.

## EEPROM Data Memory

These devices contain 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.

| Device | Capacity | Address |
|---|---|---|
| BS84B08C | 64×8 | 00H~3FH |
| BS84C12C | 128×8 | 00H~7FH |

### EEPROM Data Memory Structure

The EEPROM Data Memory capacity varies from 64×8 to 128×8 bits, according to the device selected. Unlike the Program Memory and RAM Data Memory, the EEPROM Data Memory is not directly mapped into memory space and is therefore not directly addressable in the same way as the other types of memory. Read and Write operations to the EEPROM are carried out in single byte operations using an address and a data register in Bank 0 and a single control register in Bank 1.

### EEPROM Registers

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

| Register Name | Bit | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| EEA (BS84B08C) | — | — | EEA5 | EEA4 | EEA3 | EEA2 | EEA1 | EEA0 |
| EEA (BS84C12C) | — | EEA6 | EEA5 | EEA4 | EEA3 | EEA2 | EEA1 | EEA0 |
| EED | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| EEC | — | — | — | — | WREN | WR | RDEN | RD |

**EEPROM Register List**

- **EEA Register – BS84B08C**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | — | — | EEA5 | EEA4 | EEA3 | EEA2 | EEA1 | EEA0 |
| 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 **EEA5~EEA0**: Data EEPROM address bit 5 ~ bit 0

• **EEA Register – BS84C12C**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | — | EEA6 | EEA5 | EEA4 | EEA3 | EEA2 | EEA1 | EEA0 |
| 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 6~0 **EEA6~EEA0**: Data EEPROM address bit 6 ~ 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
Data EEPROM data bit 7 ~ bit 0

• **EEC Register**

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

Bit 7~4 Unimplemented, read as "0"

Bit 3 **WREN**: Data EEPROM Write Enable
0: Disable
1: Enable
This is the Data EEPROM Write Enable Bit which must be set high before Data EEPROM write operations are carried out. Clearing this bit to zero will inhibit Data EEPROM write operations.

Bit 2 **WR**: EEPROM Write Control
0: Write cycle has finished
1: Activate a write cycle
This is the Data EEPROM Write Control Bit and when set high by the application program will activate a write cycle. This bit will be automatically reset to zero by the hardware after the write cycle has finished. Setting this bit high will have no effect if the WREN has not first been set high.

Bit 1 **RDEN**: Data EEPROM Read Enable
0: Disable
1: Enable
This is the Data EEPROM Read Enable Bit which must be set high before Data EEPROM read operations are carried out. Clearing this bit to zero will inhibit Data EEPROM read operations.

Bit 0 **RD**: EEPROM Read Control
0: Read cycle has finished
1: Activate a read cycle
This is the Data EEPROM Read Control Bit and when set high by the application program will activate a read cycle. This bit will be automatically reset to zero by the hardware after the read cycle has finished. Setting this bit high will have no effect if the RDEN has not first been set high.

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

### Reading Data from the EEPROM

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

### Writing Data to the EEPROM

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

### Write Protection

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

### EEPROM Interrupt

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

## Programming Considerations

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

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

## Programming Examples

### Reading data from the EEPROM – polling method

```
MOV A, EEPROM_ADRES       ; user defined address
MOV EEA, A
MOV A, 040H               ; setup memory pointer MP1
MOV MP1, A                ; MP1 points to EEC register
MOV A, 01H                ; setup Bank Pointer
MOV BP, A
SET IAR1.1                ; set RDEN bit, enable read operations
SET IAR1.0                ; start Read Cycle - set RD bit
BACK:
SZ IAR1.0                 ; check for read cycle end
JMP BACK
CLR IAR1                  ; disable EEPROM read if no more read operations are required
CLR BP
MOV A, EED                ; move read data to register
MOV READ_DATA, A
```

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

### Writing Data to the EEPROM – polling method

```
MOV A, EEPROM_ADRES       ; user defined address
MOV EEA, A
MOV A, EEPROM_DATA        ; user defined data
MOV EED, A
MOV A, 040H               ; setup memory pointer MP1
MOV MP1, A                ; MP1 points to EEC register
MOV A, 01H                ; setup Bank Pointer
MOV BP, A
CLR EMI
SET IAR1.3                ; set WREN bit, enable write operations
SET IAR1.2                ; start Write Cycle - set WR bit - executed immediately
                          ; after set WREN bit
SET EMI
BACK:
SZ IAR1.2                 ; check for write cycle end
JMP BACK
CLR BP
```

# Oscillators

Various oscillator types 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. Two fully integrated internal oscillators, requiring no external components, are provided to form a wide range of both fast and slow system oscillators. The higher frequency oscillator provides higher performance but carry with it the disadvantage of higher power requirements, while the opposite is of course true for the lower frequency oscillator. With the capability of dynamically switching between fast and slow system clock, the devices have the flexibility to optimize the performance/power ratio, a feature especially important in power sensitive portable applications.

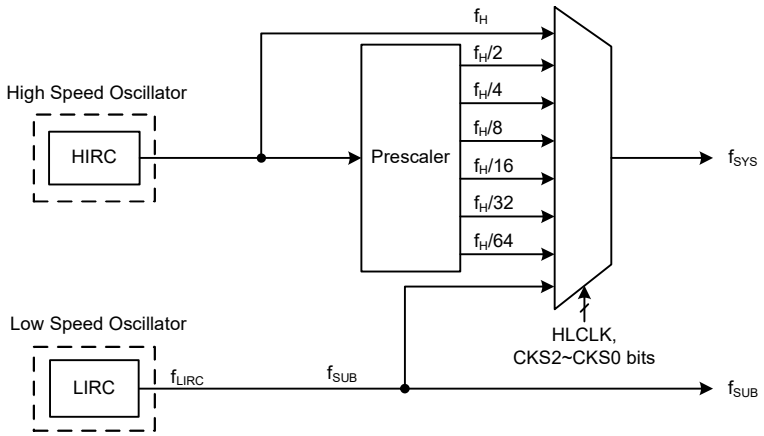
| Type | Name | Frequency |
|---|---|---|
| Internal High Speed RC | HIRC | 8/12/16MHz |
| Internal Low Speed RC | LIRC | 32kHz |

**Oscillator Types**

## System Clock Configurations

There are two methods of generating the system clock, one high speed oscillator and one low speed oscillator. The high speed oscillator is the internal 8/12/16MHz RC oscillator, HIRC. The low speed oscillator is the internal 32kHz RC oscillator, LIRC. Selecting whether the low or high speed oscillator is used as the system oscillator is implemented using the HLCLK and CKS2~CKS0 bits in the SMOD register and as the system clock can be dynamically selected.

The frequency of the slow speed or high speed system clock is determined using the HLCLK and CKS2~CKS0 bits in the SMOD register. Note that two oscillator selections must be made namely one high speed and one low speed system oscillator. It is not possible to choose a no-oscillator selection for either the high or low speed oscillator.




**System Clock Configurations**

### Internal High Speed RC Oscillator – HIRC

The internal RC oscillator is a fully integrated system oscillator requiring no external components. The internal RC oscillator has three fixed frequencies of 8MHz, 12MHz and 16MHz, which is selected using a configuration option. The HIRCS1~HIRCS0 bits in the CTRL register must also be setup to match the selected configuration option frequency. Setting up these bits is necessary 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.

### Internal 32kHz Oscillator – LIRC

The Internal 32kHz System Oscillator is the low frequency oscillator. It is a fully integrated RC oscillator with a typical frequency of 32kHz at a full voltage range, 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.

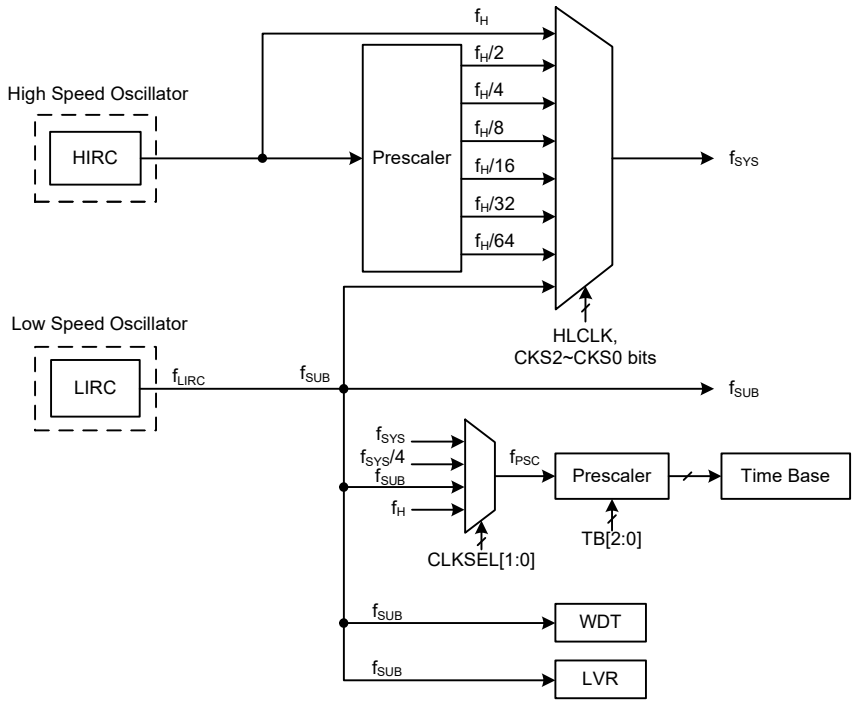
## Operating Modes and System Clocks

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 these 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

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

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




**Device Clock Configurations**

Note: When the system clock source $f_{SYS}$ is switched to $f_{SUB}$ from $f_H$, the high speed oscillation will stop to conserve the power. Thus there is no $f_H$~$f_H$/64 for peripheral circuit to use.

## System Operation Modes

There are five 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 three modes, the SLEEP, IDLE0 and IDLE1 Mode are used when the microcontroller CPU is switched off to conserve power.

| Operation Mode | Description | | |
|---|---|---|---|
| | CPU | $f_{SYS}$ | $f_{SUB}$ |
| FAST | On | $f_H$~$f_H$/64 | On |
| SLOW | On | $f_{SUB}$ | On |
| IDLE0 | Off | Off | On |
| IDLE1 | Off | On | On |
| SLEEP | Off | Off | On |

### 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 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 and HLCLK bits in the SMOD 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 the LIRC oscillator. Running the microcontroller in this mode allow it to run with much lower operating currents. In the SLOW Mode, the $f_H$ is off.

### SLEEP Mode

The SLEEP Mode is entered when an HALT instruction is executed and when the IDLEN bit in the SMOD register is low. In the SLEEP mode the CPU will be stopped and both the high and low speed oscillators will be switched off. However the $f_{SUB}$ clock will continue to operate since the WDT function is always enabled.

### IDLE0 Mode

The IDLE0 Mode is entered when an HALT instruction is executed and when the IDLEN bit in the SMOD register is high and the FSYSON bit in the CTRL register is low. In the IDLE0 Mode the system oscillator will be switched off and therefore will be inhibited from driving the CPU but some peripheral functions will remain operational.

### IDLE1 Mode

The IDLE1 Mode is entered when an HALT instruction is executed and when the IDLEN bit in the SMOD register is high and the FSYSON bit in the CTRL register is high. In the IDLE1 Mode the system oscillator will be inhibited from driving the CPU but may continue to provide a clock source to keep some peripheral functions operational. In the IDLE1 Mode, the system oscillator will continue to run and this system oscillator may be the high speed or low speed oscillator.

## Control Registers

The registers, SMOD and CTRL, are used to control the internal clocks within the devices.

| Register Name | Bit | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SMOD | CKS2 | CKS1 | CKS0 | — | LTO | HTO | IDLEN | HLCLK |
| CTRL | FSYSON | — | HIRCS1 | HIRCS0 | — | LVRF | LRF | WRF |

**System Operating Mode Control Register List**

- **SMOD Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | CKS2 | CKS1 | CKS0 | — | LTO | HTO | IDLEN | HLCLK |
| R/W | R/W | R/W | R/W | — | R | R | R/W | R/W |
| POR | 0 | 0 | 0 | — | 0 | 0 | 1 | 1 |

Bit 7~5 **CKS2~CKS0**: System clock selection when HLCLK is “0”
000: $f_{SUB}$ ($f_{LIRC}$)
001: $f_{SUB}$ ($f_{LIRC}$)
010: $f_H/64$
011: $f_H/32$
100: $f_H/16$
101: $f_H/8$
110: $f_H/4$
111: $f_H/2$

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 $f_{SUB}$, a divided version of the high speed system oscillator can also be chosen as the system clock source.

Bit 4 Unimplemented, read as "0"

Bit 3 **LTO**: Low speed system oscillator ready flag
0: Not ready
1: Ready
This is the low speed system oscillator ready flag which indicates when the low speed system oscillator is stable after power on reset or a wake-up has occurred. The flag will be low when in the SLEEP Mode but after a wake-up has occurred, the flag will change to a high level after 1~2 clock cycles if the LIRC oscillator is used.

Bit 2 **HTO**: High speed system oscillator ready flag
0: Not ready
1: Ready
This is the high speed system oscillator ready flag which indicates when the high speed system oscillator is stable after a wake-up has occurred. This flag is cleared to zero by hardware when the device is powered on and then changes to a high level after the high speed system oscillator is stable. Therefore, this flag will always be read as "1" by the application program after device power-on. The flag will be low when in the SLEEP or IDLE0 Mode but after power on reset or a wake-up has occurred, the flag will change to a high level after 15~16 clock cycles if the HIRC oscillator is used.

Bit 1 **IDLEN**: IDLE Mode Control
0: Disable
1: Enable
This bit is the IDLE Mode control bit and determines what happens when the HALT instruction is executed. If this bit is high, when a HALT instruction is executed the device will enter the IDLE Mode. In the IDLE1 Mode the CPU will stop running but the system clock will continue to keep the peripheral functions operational, if FSYSON bit is high. If the FSYSON bit is low, the CPU and the system clock will all stop in the IDLE0 Mode. If the bit is low the device will enter the SLEEP Mode when a HALT instruction is executed.

Bit 0 **HLCLK**: System clock selection
0: $f_H/2$~$f_H/64$ or $f_{SUB}$
1: $f_H$
This bit is used to select if the $f_H$ clock, the $f_H/2$~$f_H/64$ or $f_{SUB}$ clock is used as the system clock. When this bit is high the $f_H$ clock will be selected and if low the $f_H/2$~$f_H/64$ or $f_{SUB}$ clock will be selected. When the system clock is switched from the $f_H$ clock to the $f_{SUB}$ clock, the $f_H$ clock will automatically be switched off to conserve power.

- **CTRL Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | FSYSON | — | HIRCS1 | HIRCS0 | — | LVRF | LRF | WRF |
| R/W | R/W | — | R/W | R/W | — | R/W | R/W | R/W |
| POR | 0 | — | 0 | 0 | — | x | 0 | 0 |

"x": unknown

Bit 7 **FSYSON**: $f_{SYS}$ Control in IDLE Mode
0: Disable
1: Enable

Bit 6 Unimplemented, read as "0"

Bit 5~4 **HIRCS1~HIRCS0**: HIRC frequency selection
00: 8MHz
01: 16MHz
10: 12MHz
11: 8MHz

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

Bit 3 Unimplemented, read as "0"

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

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

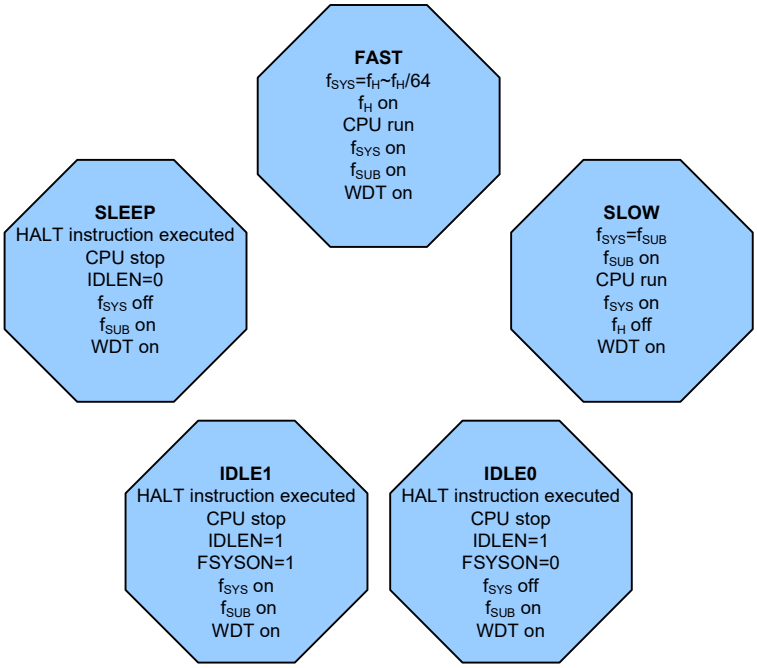
Bit 0 **WRF**: WDT control register software reset flag
Described elsewhere

## 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 HLCLK and CKS2~CKS0 bits in the SMOD register while Mode Switching from the FAST/SLOW Modes to the SLEEP/IDLE Modes is executed via the HALT instruction. When an HALT instruction is executed, whether the devices enter the IDLE Mode or the SLEEP Mode is determined by the condition of the IDLEN bit in the SMOD register and the FSYSON bit in the CTRL register.

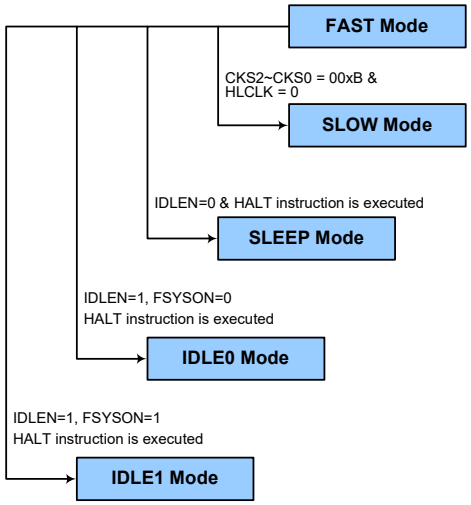
When the HLCLK bit switches to a low level, which implies that clock source is switched from the high speed clock source, $f_H$, to the clock source, $f_H/2$~$f_H/64$ or $f_{SUB}$. If the clock is from the $f_{SUB}$, the high speed clock source will stop running to conserve power. When this happens it must be noted that the $f_H/16$ and $f_H/64$ internal clock sources will also stop running. The accompanying flowchart shows what happens when the devices move between the various operating modes.



### 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 clearing the HLCLK bit to zero and setting the CKS2~CKS0 bits to "000" or "001" in the SMOD 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.

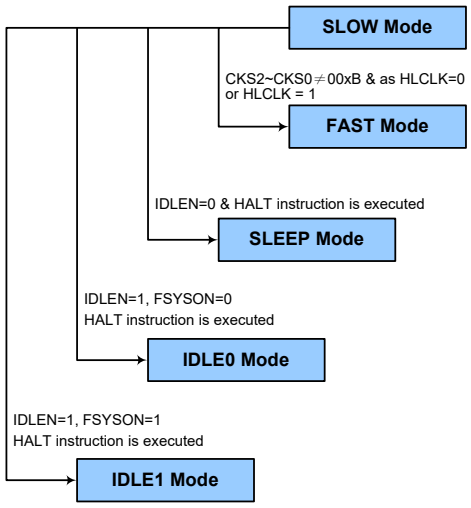
The SLOW Mode is sourced from the LIRC oscillator and therefore requires the specific oscillator to stable before full mode switching occurs.




### SLOW Mode to FAST Mode Switching

In SLOW mode the system uses the $f_{SUB}$ clock derived from the LIRC low speed oscillator as system clock. When system clock is switched back to the FAST mode from $f_{SUB}$, where the high speed system oscillator is used, the HLCLK bit should be set high or HLCLK bit is low but the CKS2~CKS0 bits are set to "010~111" and then the system clock will respectively be switched to $f_H$ ~$f_H$/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 back to the FAST mode from the SLOW Mode and the status of the HTO flag should be checked. The time duration required for the high speed system oscillator stabilization is specified in the relevant characteristics.



### Entering the SLEEP Mode

There is only one way for the devices to enter the SLEEP Mode and that is to execute the "HALT" instruction in the application program with the IDLEN bit in the SMOD register equal to "0". In this mode all the clocks and functions will be switched off except the WDT function. When this instruction is executed under the conditions described above, the following will occur:

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

### Entering the IDLE0 Mode

There is only one way for the devices to enter the IDLE0 Mode and that is to execute the "HALT" instruction in the application program with the IDLEN bit in the SMOD register equal to "1" and the FSYSON bit in the CTRL register equal to "0". When this instruction is executed under the conditions described above, the following will occur:

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

### Entering the IDLE1 Mode

There is only one way for the devices to enter the IDLE1 Mode and that is to execute the "HALT" instruction in the application program with the IDLEN bit in the SMOD register is equal to "1" and the FSYSON bit in the CTRL register equal to "1". When this instruction is executed under the conditions described above, the following will occur:

- The system and the low frequency $f_{SUB}$ clocks will be on but the application program will stop at the "HALT" instruction.
- The Data Memory contents and registers will maintain their present condition.
- The I/O ports will maintain their present conditions.
- In the status register, the Power Down flag PDF will be set, and WDT timeout flag TO will be cleared.
- The WDT will be cleared and resume counting since the WDT function is always enabled.

## Standby Current Considerations

As the main reason for entering the SLEEP or IDLE Mode is to keep the current consumption of the devices to as low a value as possible, perhaps only in the order of several micro-amps except in the IDLE1 Mode, there are other considerations which must also be taken into account by the circuit designer if the power consumption is to be minimised. Special attention must be made to the I/O pins on the 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 devices which have different package types, as there may be unbonded pins. These must either be setup as outputs or if setup as inputs must have pull-high resistors connected.

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

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

## Wake-up

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

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

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

When the devices execute the “HALT” instruction, the PDF flag will be set to 1. The PDF flag will be cleared to 0 if the devices experience 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 devices 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.

| System Oscillator | Wake-up Time (SLEEP Mode) | Wake-up Time (IDLE0 Mode) | Wake-up Time (IDLE1 Mode) |
|---|---|---|---|
| HIRC | 15~16 HIRC cycles | | 1~2 HIRC cycles |
| LIRC | 1~2 LIRC cycles | | 1~2 LIRC cycles |

**Wake-up Time**

## Programming Considerations

The high speed and low speed oscillators both use the same SST counter. For example, if the system is woken up from the SLEEP Mode the HIRC oscillator needs to start-up from an off state. If the devices are woken up from the SLEEP Mode to the FAST Mode, the high speed system oscillator needs an SST period. The devices will execute the first instruction after HTO is high.

# 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 $f_{SUB}$ clock which is in turn supplied by 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 enable and reset operations. This register controls the overall operation of the Watchdog Timer. The WDTC register is initiated to 01010011B at any reset except the WDT time-out hardware warm reset.

- **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 control
01010/10101: 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 2~3 LIRC clock cycles, and the WRF bit in the CTRL register will be set high.

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

- **CTRL Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | FSYSON | — | HIRCS1 | HIRCS0 | — | LVRF | LRF | WRF |
| R/W | R/W | — | R/W | R/W | — | R/W | R/W | R/W |
| POR | 0 | — | 0 | 0 | — | x | 0 | 0 |

"x": unknown

Bit 7 **FSYSON**: $f_{SYS}$ Control in IDLE Mode
Described elsewhere

Bit 6 Unimplemented, read as "0"

Bit 5~4 **HIRCS1~HIRCS0**: HIRC frequency selection
Described elsewhere

Bit 3 Unimplemented, read as "0"

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

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

Bit 0 **WRF**: WDT control register software reset flag
0: Not occur
1: Occurred
This bit is set high by the WDT control register software reset and cleared by the application program. Note that this bit can only be cleared to zero by the application program.

## Watchdog Timer Operation

The Watchdog Timer operates by providing a device reset when its timer overflows. This means that in the application program and during normal operation the user has to strategically clear the Watchdog Timer before it overflows to prevent the Watchdog Timer from executing a reset. This is done using the clear watchdog instruction. If the program malfunctions for whatever reason, jumps to an unknown location, or enters an endless loop, the clear instruction will not be executed in the correct manner, in which case the Watchdog Timer will overflow and reset the devices. With regard to the Watchdog Timer function control, there are five bits, WE4~WE0, in the WDTC register to offer the enable and reset control of the Watchdog Timer. The WDT function will be enabled when the WE4~WE0 bits are set to a value of 01010B or 10101B. If the WE4~WE0 bits are changed to any other values rather than 01010B and 10101B, which is caused by the environmental noise, it will reset the devices after 2~3 LIRC clock cycles. After power on these bits will have a value of 01010B.

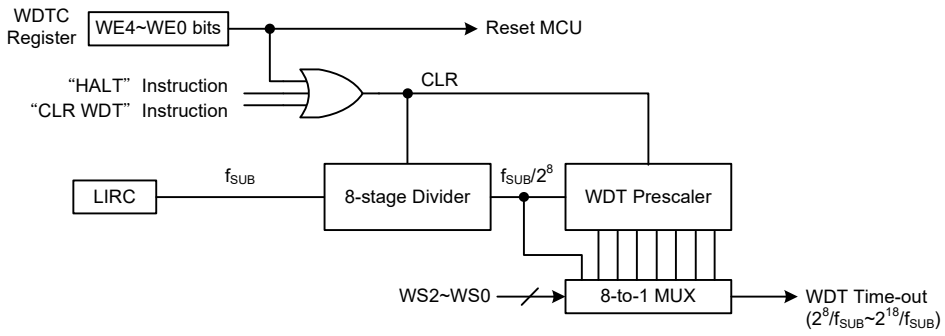
| WE4~WE0 Bits | WDT Function |
|---|---|
| 01010B or 10101B | Enable |
| Any other value | Reset MCU |

**Watchdog Timer Enable/Reset Control**

Under normal program operation, a Watchdog Timer time-out will initialise a device reset and set the status bit TO. However, if the system is in the SLEEP or IDLE Mode, when a Watchdog Timer time-out occurs, the TO bit in the status register will be set and only the Program Counter and Stack Pointer will be reset. Three methods can be adopted to clear the contents of the Watchdog Timer. The first is a WDT reset, which means a certain value except 01010B and 10101B written into the WE4~WE0 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 second for the $2^{18}$ division ratio and a minimum timeout of 8ms for the $2^{8}$ division ration.




**Watchdog Timer**

## Reset and Initialisation

A reset function is a fundamental part of any microcontroller ensuring that the devices 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.

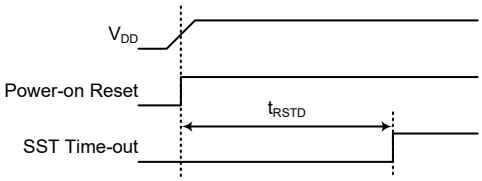
In addition to the power-on reset, another reset exists in the form of a Low Voltage Reset, LVR, where a full reset is implemented in situations where the power supply voltage falls below a certain threshold. Another type of reset is when the Watchdog Timer overflows and resets the microcontroller. All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

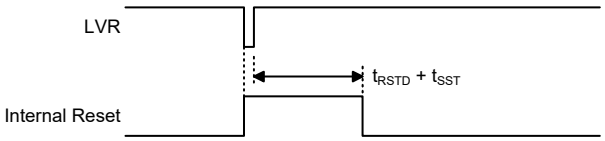



Note: $t_{RSTD}$ is power-on delay with typical time=48ms

**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 devices. The LVR function is always enabled with a specific LVR voltage, $V_{LVR}$. If the supply voltage of the devices drops to within a range of 0.9V~$V_{LVR}$ such as might occur when changing the battery, the LVR will automatically reset the devices internally and the LVRF bit in the CTRL register will also be set to 1. For a valid LVR signal, a low supply voltage, i.e., a voltage in the range between 0.9V~$V_{LVR}$ must exist for a time greater than that specified by $t_{LVR}$ in the LVR Electrical Characteristics. If the low supply voltage state does not exceed this value, the LVR will ignore the low supply voltage and will not perform a reset function. The actual $V_{LVR}$ value can be selected by the LVS7~LVS0 bits in the LVRC register. If the LVS7~LVS0 bits have any other value, which may perhaps occur due to adverse environmental conditions such as noise, the LVR will reset the devices after 2~3 LIRC clock cycles. When this happens, the LRF bit in the CTRL register will be set to 1. After power on the register will have the default value of 01010101B. Note that the LVR function will be automatically disabled when the devices enter the SLEEP/IDLE mode.




Note: $t_{RSTD}$ is power-on delay with typical time=48ms

**Low Voltage Reset Timing Chart**

- **LVRC Register**

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

Bit 7~0 **LVS7~LVS0**: LVR voltage select
01010101: 2.1V
00110011: 2.55V
10011001: 3.15V
10101010: 3.8V
Other values: Generates a MCU reset – register is reset to POR value

When an actual low voltage condition occurs, as specified by the LVR voltage value above, an MCU reset will be generated. The reset operation will be activated after the low voltage condition keeps for greater 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 four defined register values above, will also result in the generation of an MCU reset. The reset operation will be activated after 2~3 LIRC clock cycles. However in this situation the register contents will be reset to the POR value.

• **CTRL Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | FSYSON | — | HIRCS1 | HIRCS0 | — | LVRF | LRF | WRF |
| R/W | R/W | — | R/W | R/W | — | R/W | R/W | R/W |
| POR | 0 | — | 0 | 0 | — | x | 0 | 0 |

"x": unknown

Bit 7 **FSYSON**: $f_{SYS}$ Control in IDLE Mode
Described elsewhere

Bit 6 Unimplemented, read as "0"

Bit 5~4 **HIRCS1~HIRCS0**: HIRC frequency selection
Described elsewhere

Bit 3 Unimplemented, read as "0"

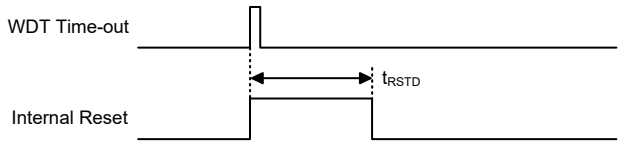
Bit 2 **LVRF**: LVR function reset flag
0: Not occurred
1: Occurred
This bit is set to 1 when a specific low voltage reset condition occurs. Note that this bit can only be cleared to 0 by the application program.

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

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

### Watchdog Time-out Reset during Normal Operation

The Watchdog time-out Reset during normal operations in the FAST or SLOW mode is the same as the hardware Low Voltage Reset except that the Watchdog time-out flag TO will be set to "1".

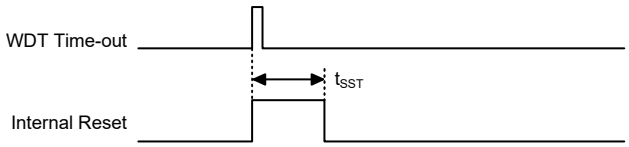



Note: $t_{RSTD}$ is power-on delay with typical time=16ms

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

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

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




**WDT Time-out Reset during SLEEP or IDLE 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" stands for unchanged

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

| Item | Condition after Reset |
|---|---|
| Program Counter | Reset to zero |
| Interrupts | All interrupts will be disabled |
| WDT, Time Base | 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 | BS84B08C | BS84C12C | Power On Reset | LVR Reset (Normal Operation) | WDT Time-out (Normal Operation) | WDT Time-out (IDLE/SLEEP) |
|---|---|---|---|---|---|---|
| IAR0 | ● | ● | xxxx xxxx | xxxx xxxx | xxxx xxxx | uuuu uuuu |
| MP0 | ● | ● | xxxx xxxx | xxxx xxxx | xxxx xxxx | uuuu uuuu |
| IAR1 | ● | ● | xxxx xxxx | xxxx xxxx | xxxx xxxx | uuuu uuuu |
| MP1 | ● | ● | xxxx xxxx | xxxx xxxx | xxxx xxxx | uuuu uuuu |
| BP | ● | | ---- ---0 | ---- ---0 | ---- ---0 | ---- ---u |
| | | ● | ---- --00 | ---- --00 | ---- --00 | ---- --uu |
| ACC | ● | ● | xxxx xxxx | uuuu uuuu | uuuu uuuu | uuuu uuuu |
| PCL | ● | ● | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 |
| TBLP | ● | ● | xxxx xxxx | uuuu uuuu | uuuu uuuu | uuuu uuuu |
| TBLH | ● | ● | xxxx xxxx | uuuu uuuu | uuuu uuuu | uuuu uuuu |
| TBHP | ● | ● | ---- xxxx | ---- uuuu | ---- uuuu | ---- uuuu |
| STATUS | ● | ● | --00 xxxx | --uu uuuu | --1u uuuu | --11 uuuu |
| SMOD | ● | ● | 000- 0011 | 000- 0011 | 000- 0011 | uuu- uuuu |
| CTRL | ● | ● | 0-00 -x00 | 0-00 -100 | 0-00 -x00 | u-uu --uu |
| INTEG | ● | ● | ---- --00 | ---- --00 | ---- --00 | ---- --uu |
| INTC0 | ● | ● | -000 0000 | -000 0000 | -000 0000 | -uuu uuuu |
| INTC1 | ● | ● | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| INTC2 | | ● | ---0 ---0 | ---0 ---0 | ---0 ---0 | ---- ---u |
| MFI0 | ● | ● | --00 --00 | --00 --00 | --00 --00 | --uu --uu |

| Register | BS84B08C | BS84C12C | Power On Reset | LVR Reset (Normal Operation) | WDT Time-out (Normal Operation) | WDT Time-out (IDLE/SLEEP) |
|---|---|---|---|---|---|---|
| LVRC | ● | ● | 0101 0101 | 0101 0101 | 0101 0101 | uuuu uuuu |
| PA | ● | ● | 1--1 1111 | 1--1 1111 | 1--1 1111 | u--u uuuu |
| PAC | ● | ● | 1--1 1111 | 1--1 1111 | 1--1 1111 | u--u uuuu |
| PAPU | ● | ● | 0--0 0000 | 0--0 0000 | 0--0 0000 | u--u uuuu |
| PAWU | ● | ● | 0--0 0000 | 0--0 0000 | 0--0 0000 | u--u uuuu |
| PXRM | ● | | 00-- ---00 | 00-- ---00 | 00-- ---00 | uu-- ---uu |
| | | ● | 0000 ---00 | 0000 ---00 | 0000 ---00 | uuuu ---uu |
| WDTC | ● | ● | 0101 0011 | 0101 0011 | 0101 0011 | uuuu uuuu |
| TBC | ● | ● | ---- 0000 | ---- 0000 | ---- 0000 | ---- uuuu |
| PSCR | ● | ● | ---- --00 | ---- --00 | ---- --00 | ---- --uu |
| MFI1 | | ● | --00 --00 | --00 --00 | --00 --00 | --uu --uu |
| EEA | ● | | --00 0000 | --00 0000 | --00 0000 | --uu uuuu |
| | | ● | -000 0000 | -000 0000 | -000 0000 | -uuu uuuu |
| EED | ● | ● | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| PB | ● | ● | 1111 1111 | 1111 1111 | 1111 1111 | uuuu uuuu |
| PBC | ● | ● | 1111 1111 | 1111 1111 | 1111 1111 | uuuu uuuu |
| PBPU | ● | ● | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| SIMTOC | ● | ● | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| SIMC0 | ● | ● | 111- 0000 | 111- 0000 | 111- 0000 | uuu- uuuu |
| SIMC1 | ● | ● | 1000 0001 | 1000 0001 | 1000 0001 | uuuu uuuu |
| SIMD | ● | ● | xxxx xxxx | xxxx xxxx | xxxx xxxx | uuuu uuuu |
| SIMC2 | ● | ● | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| SIMA | ● | ● | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| CTMC0 | | ● | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| CTMC1 | | ● | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| CTMDL | | ● | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| CTMDH | | ● | ---- --00 | ---- --00 | ---- --00 | ---- --uu |
| CTMAL | | ● | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| CTMAH | | ● | ---- --00 | ---- --00 | ---- --00 | ---- --uu |
| ADRL | ● | ● | xxxx ---- | xxxx ---- | xxxx ---- | xxxx ---- (ADRFS=0)<br>xxxx xxxx (ADRFS=1) |
| ADRH | ● | ● | xxxx xxxx | xxxx xxxx | xxxx xxxx | xxxx xxxx (ADRFS=0)<br>---- xxxx (ADRFS=1) |
| ADCR0 | ● | ● | 0110 -000 | 0110 -000 | 0110 -000 | uuuu -uuu |
| ADCR1 | ● | ● | 00-0 -000 | 00-0 -000 | 00-0 -000 | uu-u -uuu |
| ACERL | ● | ● | 1111 1111 | 1111 1111 | 1111 1111 | uuuu uuuu |
| SLEDC0 | ● | ● | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| SLEDC1 | ● | | ---- 0000 | ---- 0000 | ---- 0000 | ---- uuuu |
| | | ● | --00 0000 | --00 0000 | --00 0000 | --uu uuuu |
| PD | ● | ● | 1111 1111 | 1111 1111 | 1111 1111 | uuuu uuuu |
| PDC | ● | ● | 1111 1111 | 1111 1111 | 1111 1111 | uuuu uuuu |
| PDPU | ● | ● | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |

| Register | BS84B08C | BS84C12C | Power On Reset | LVR Reset (Normal Operation) | WDT Time-out (Normal Operation) | WDT Time-out (IDLE/SLEEP) |
|---|---|---|---|---|---|---|
| PC | | ● | ---- 1111 | ---- 1111 | ---- 1111 | ---- uuuu |
| PCC | | ● | ---- 1111 | ---- 1111 | ---- 1111 | ---- uuuu |
| PCPU | | ● | ---- 0000 | ---- 0000 | ---- 0000 | ---- uuuu |
| PTMC0 | ● | ● | 0000 0--- | 0000 0--- | 0000 0--- | uuuu u--- |
| PTMC1 | ● | ● | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| PTMDL | ● | ● | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| PTMDH | ● | ● | ---- --00 | ---- --00 | ---- --00 | ---- --uu |
| PTMAL | ● | ● | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| PTMAH | ● | ● | ---- --00 | ---- --00 | ---- --00 | ---- --uu |
| PTMRPL | ● | ● | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| PTMRPH | ● | ● | ---- --00 | ---- --00 | ---- --00 | ---- --uu |
| TKTMR | ● | ● | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKC0 | ● | ● | -000 0000 | -000 0000 | -000 0000 | -uuu uuuu |
| TK16DL | ● | ● | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TK16DH | ● | ● | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKC1 | ● | ● | ---- --11 | ---- --11 | ---- --11 | ---- --uu |
| TKM016DL | ● | ● | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM016DH | ● | ● | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM0ROL | ● | ● | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM0ROH | ● | ● | ---- --00 | ---- --00 | ---- --00 | ---- --uu |
| TKM0C0 | ● | ● | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM0C1 | ● | ● | 0-00 0000 | 0-00 0000 | 0-00 0000 | u-uu uuuu |
| TKM116DL | ● | ● | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM116DH | ● | ● | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM1ROL | ● | ● | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM1ROH | ● | ● | ---- --00 | ---- --00 | ---- --00 | ---- --uu |
| TKM1C0 | ● | ● | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM1C1 | ● | ● | 0-00 0000 | 0-00 0000 | 0-00 0000 | u-uu uuuu |
| TKM216DL | | ● | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM216DH | | ● | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM21ROL | | ● | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM2ROH | | ● | ---- --00 | ---- --00 | ---- --00 | ---- --uu |
| TKM2C0 | | ● | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM2C1 | | ● | 0-00 0000 | 0-00 0000 | 0-00 0000 | u-uu uuuu |
| EEC | ● | ● | ---- 0000 | ---- 0000 | ---- 0000 | ---- uuuu |

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

## Input/Output Ports

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

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

| Register Name | Bit | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| PA | PA7 | — | — | PA4 | PA3 | PA2 | PA1 | PA0 |
| PAC | PAC7 | — | — | PAC4 | PAC3 | PAC2 | PAC1 | PAC0 |
| PAPU | PAPU7 | — | — | PAPU4 | PAPU3 | PAPU2 | PAPU1 | PAPU0 |
| PAWU | PAWU7 | — | — | 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 (BS84C12C) | — | — | — | — | PC3 | PC2 | PC1 | PC0 |
| PCC (BS84C12C) | — | — | — | — | PCC3 | PCC2 | PCC1 | PCC0 |
| PCPU (BS84C12C) | — | — | — | — | PCPU3 | PCPU2 | PCPU1 | PCPU0 |
| PD | PD7 | PD6 | PD5 | PD4 | PD3 | PD2 | PD1 | PD0 |
| PDC | PDC7 | PDC6 | PDC5 | PDC4 | PDC3 | PDC2 | PDC1 | PDC0 |
| PDPU | PDPU7 | PDPU6 | PDPU5 | PDPU4 | PDPU3 | PDPU2 | PDPU1 | PDPU0 |

"—": Unimplemented, read as "0"

**I/O Logic Function Register List**

### Pull-high Resistors

Many product applications require pull-high resistors for their switch inputs usually requiring the use of an external resistor. To eliminate the need for these external resistors, all I/O pins, when configured as an input have the capability of being connected to an internal pull-high resistor. These pull-high resistors are selected using the relevant pull-high control registers and are implemented using weak PMOS transistors.

- **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 and D depending upon the selected device. However, the actual available bits for each I/O Port may be different.

## 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.

- **PAWU Register**

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

Bit 7, 4~0 **PAWU7, PAWU4~PAWU0**: Port A pin Wake-up function control
0: Disable
1: Enable

Bit 6~5 Unimplemented, read as "0"

## I/O Port Control Registers

Each I/O Port has its own control register which controls the input/output configuration. With this control register, each I/O pin with or without pull-high resistors can be reconfigured dynamically under software control. For the I/O pin to function as an input, the corresponding bit of the control register must be written as a "1". This will then allow the logic state of the input pin to be directly read by instructions. When the corresponding bit of the control register is written as a "0", the I/O pin will be setup as a CMOS output. If the pin is currently setup as an output, instructions can still be used to read the output register. However, it should be noted that the program will in fact only read the status of the output data latch and not the actual logic status of the output pin.

- **PxC Register**

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

**PxCn**: I/O Port x Pin type selection
0: Output
1: Input

The PxCn bit is used to control the pin type selection. Here the "x" is the Port name which can be A, B, C and D depending upon the selected device. However, the actual available bits for each I/O Port may be different.

## I/O Port Source Current Selection

These devices support different output source current driving capability for each I/O port. With the selection registers, SLEDC0~SLEDC1, specific I/O port can support four levels of the source current driving capability. Users should refer to the Input/Output Characteristics section to select the desired output source current for different applications.

| Register Name | Bit | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SLEDC0 | SLEDC07 | SLEDC06 | SLEDC05 | SLEDC04 | SLEDC03 | SLEDC02 | SLEDC01 | SLEDC00 |
| SLEDC1 (BS84B08C) | — | — | — | — | SLEDC13 | SLEDC12 | SLEDC11 | SLEDC10 |
| SLEDC1 (BS84C12C) | — | — | SLEDC15 | SLEDC14 | SLEDC13 | SLEDC12 | SLEDC11 | SLEDC10 |

**I/O Port Source Current Selection Register List**

### • SLEDC0 Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | SLEDC07 | SLEDC06 | SLEDC05 | SLEDC04 | SLEDC03 | SLEDC02 | SLEDC01 | SLEDC00 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| POR | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Bit 7~6 **SLEDC07~SLEDC06**: PB7~PB4 source current selection
00: Source current=Level 0 (Min.)
01: Source current=Level 1
10: Source current=Level 2
11: Source current=Level 3 (Max.)

Bit 5~4 **SLEDC05~SLEDC04**: PB3~PB0 source current selection
00: Source current=Level 0 (Min.)
01: Source current=Level 1
10: Source current=Level 2
11: Source current=Level 3 (Max.)

Bit 3~2 **SLEDC03~SLEDC02**: PA7 and PA4 source current selection
00: Source current=Level 0 (Min.)
01: Source current=Level 1
10: Source current=Level 2
11: Source current=Level 3 (Max.)

Bit 1~0 **SLEDC01~SLEDC00**: PA3~PA0 source current selection
00: Source current=Level 0 (Min.)
01: Source current=Level 1
10: Source current=Level 2
11: Source current=Level 3 (Max.)

• **SLEDC1 Register – BS84B08C**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | — | — | — | — | SLEDC13 | SLEDC12 | SLEDC11 | SLEDC10 |
| 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 **SLEDC13~ SLEDC12**: PD7~PD4 source current selection
00: Source current=Level 0 (Min.)
01: Source current=Level 1
10: Source current=Level 2
11: Source current=Level 3 (Max.)

Bit 1~0 **SLEDC11~SLEDC10**: PD3~PD0 source current selection
00: Source current=Level 0 (Min.)
01: Source current=Level 1
10: Source current=Level 2
11: Source current=Level 3 (Max.)

• **SLEDC1 Register – BS84C12C**

| 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 | 0 | 0 | 0 | 0 | 0 |

Bit 7~6 Unimplemented, read as "0"

Bit 5~4 **SLEDC15~SLEDC14**: PC3~PC0 source current selection
00: Source current=Level 0 (Min.)
01: Source current=Level 1
10: Source current=Level 2
11: Source current=Level 3 (Max.)

Bit 3~2 **SLEDC13~SLEDC12**: PD7~PD4 source current selection
00: Source current=Level 0 (Min.)
01: Source current=Level 1
10: Source current=Level 2
11: Source current=Level 3 (Max.)

Bit 1~0 **SLEDC11~SLEDC10**: PD3~PD0 source current selection
00: Source current=Level 0 (Min.)
01: Source current=Level 1
10: Source current=Level 2
11: Source current=Level 3 (Max.)

## Pin-remapping Function

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. The way in which the pin function of specific pins is selected is different for each function and a priority order is established where more than one pin function is selected simultaneously.

### Pin-remapping Selection Register

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.

- **PXRM Register – BS84B08C**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | PTMPC1 | PTMPC0 | — | — | — | — | PXRM1 | PXRM0 |
| R/W | R/W | R/W | — | — | — | — | R/W | R/W |
| POR | 0 | 0 | — | — | — | — | 0 | 0 |

Bit 7 **PTMPC1**: PTPB pin Control
Described elsewhere

Bit 6 **PTMPC0**: PTP pin Control
Described elsewhere

Bit 5~2 Unimplemented, read as “0”

Bit 1 **PXRM1**: SCK/SCL pin remapping function selection
0: PA2
1: PA7

Bit 0 **PXRM0**: SDI/SDA pin remapping function selection
0: PA0
1: PA4

- **PXRM Register – BS84C12C**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | PTMPC1 | PTMPC0 | CTMPC1 | CTMPC0 | — | — | PXRM1 | PXRM0 |
| R/W | R/W | R/W | R/W | R/W | — | — | R/W | R/W |
| POR | 0 | 0 | 0 | 0 | — | — | 0 | 0 |

Bit 7 **PTMPC1**: PTPB pin Control
Described elsewhere

Bit 6 **PTMPC0**: PTP pin Control
Described elsewhere

Bit 5 **CTMPC1**: CTPB pin Control
Described elsewhere

Bit 4 **CTMPC0**: CTP pin Control
Described elsewhere

Bit 3~2 Unimplemented, read as “0”

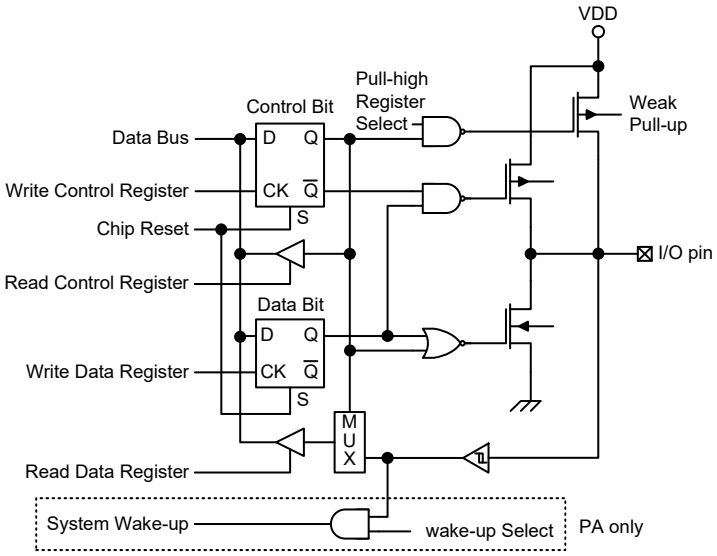
Bit 1 **PXRM1**: SCK/SCL pin remapping function selection
0: PA2
1: PA7

Bit 0 **PXRM0**: SDI/SDA pin remapping function selection
0: PA0

1: PA4

## 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.




**Logic Function Input/Output Structure**

## Programming Considerations

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

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

# Timer Modules – TM

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

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

## Introduction

These devices contain 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 type TMs will be described in this section and the detailed operation regarding each of the TM types will be described in separate sections. The main features and differences between the two types of TMs are summarised in the accompanying table.

| TM Function | CTM | PTM |
|---|---|---|
| Timer/Counter | √ | √ |
| Input Capture | — | √ |
| Compare Match Output | √ | √ |
| PWM Channels | 1 | 1 |
| Single Pulse Output | — | 1 |
| PWM Alignment | Edge | Edge |
| PWM Adjustment Period & Duty | Duty or Period | Duty or Period |

**TM Function Summary**

| Device | CTM | PTM |
|---|---|---|
| BS84B08C | — | PTM |
| BS84C12C | CTM | PTM |

**TM Name/Type Summary**

## TM Operation

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

## TM Clock Source

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

## TM Interrupts

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

## TM External Pins

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

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

The TMs each have two output pins, xTP and xTPB. When the TM is in the Compare Match Output Mode, these pins can be controlled by the TM to switch to a high or low level or to toggle when a compare match situation occurs. The external xTP and xTPB output pins are also the pins where the xTM generates the PWM output waveform. As the xTM output pins are pin-shared with other functions, the TM output function must first be setup using the relevant registers. A signal bit in the register determines if its associated pin is to be used as an external TM output pin or if it is to have another function. The number of external pins for each TM type is different, the details are provided in the accompanying table.

| Device | CTM | | PTM | |
|---|---|---|---|---|
| | Input | Output | Input | Output |
| BS84B08C | — | — | PTCK, PTPI | PTP, PTPB |
| BS84C12C | CTCK | CTP, CTPB | PTCK, PTPI | PTP, PTPB |

**TM External Pins**

### TM Pin Control Register

Selecting to have a TM output or whether to retain its other shared function is implemented using one register, with a single bit in the register corresponding to a TM output pin. Setting the bit high will setup the corresponding pin as a TM output, if reset to zero the pin will retain its original other function.

- **PXRM Register – BS84B08C**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | PTMPC1 | PTMPC0 | — | — | — | — | PXRM1 | PXRM0 |
| R/W | R/W | R/W | — | — | — | — | R/W | R/W |
| POR | 0 | 0 | — | — | — | — | 0 | 0 |

Bit 7 **PTMPC1**: PTPB pin Control
0: Disable
1: Enable

Bit 6 **PTMPC0**: PTP pin Control
0: Disable
1: Enable

Bit 5~2 Unimplemented, read as "0"

Bit 1 **PXRM1**: SCK/SCL pin remapping function selection
Described elsewhere

Bit 0 **PXRM0**: SDI/SDA pin remapping function selection
Described elsewhere

- **PXRM Register – BS84C12C**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | PTMPC1 | PTMPC0 | CTMPC1 | CTMPC0 | — | — | PXRM1 | PXRM0 |
| R/W | R/W | R/W | R/W | R/W | — | — | R/W | R/W |
| POR | 0 | 0 | 0 | 0 | — | — | 0 | 0 |

Bit 7 **PTMPC1**: PTPB pin Control
0: Disable
1: Enable

Bit 6 **PTMPC0**: PTP pin Control
0: Disable
1: Enable

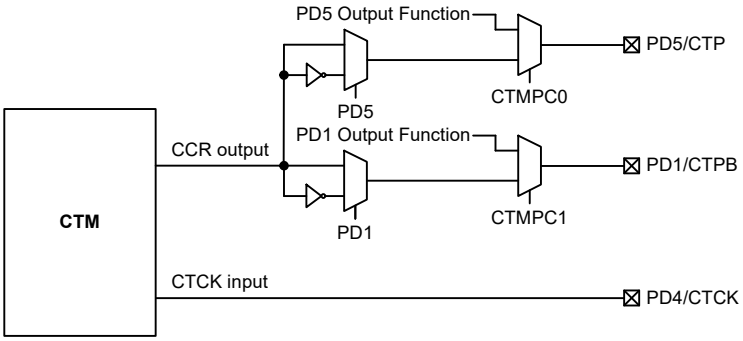
Bit 5 **CTMPC1**: CTPB pin Control
0: Disable
1: Enable

Bit 4 **CTMPC0**: CTP pin Control
0: Disable
1: Enable

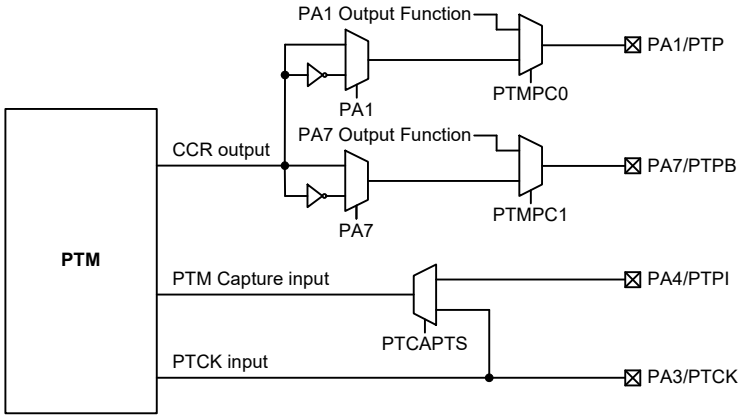
Bit 3~2 Unimplemented, read as "0"

Bit 1 **PXRM1**: SCK/SCL pin remapping function selection
Described elsewhere

Bit 0 **PXRM0**: SDI/SDA pin remapping function selection
Described elsewhere




**CTM Function Pin Control Block Diagram – BS84C12C**

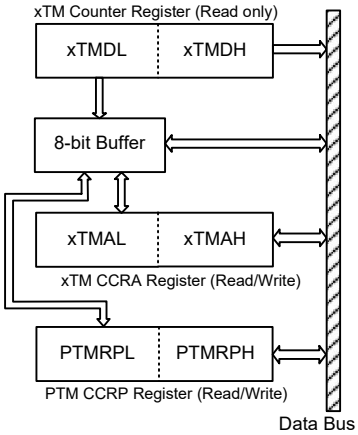



**PTM Function Pin Control Block Diagram**

## Programming Considerations

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

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



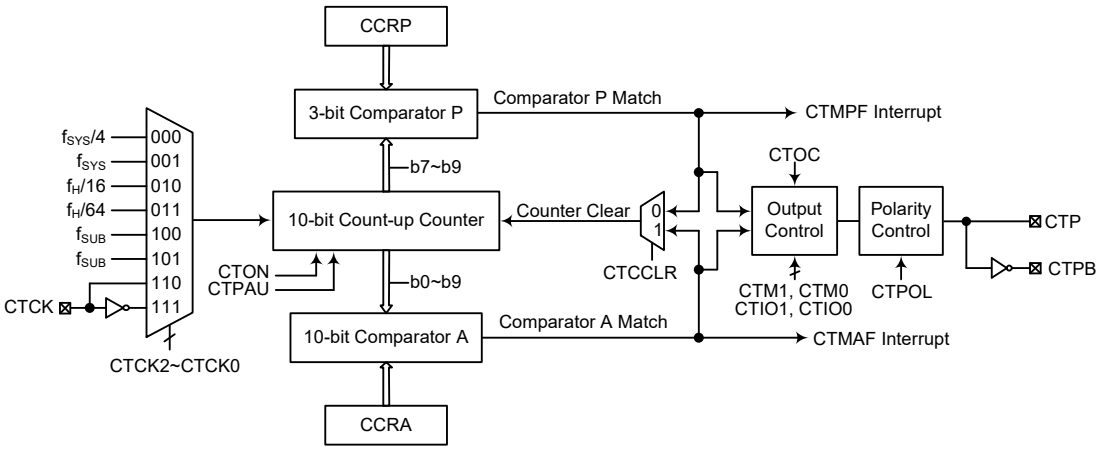

The following steps show the read and write procedures:

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

## Compact Type TM – CTM

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

| Device | CTM Core | CTM Input Pin | CTM Output Pin |
|---|---|---|---|
| BS84B08C | — | — | — |
| BS84C12C | 10-bit CTM | CTCK | CTP, CTPB |




**Compact Type TM Block Diagram**

### Compact Type TM Operation

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

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

### Compact Type TM Register Description

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

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

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

- **CTMC0 Register**

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

Bit 7 **CTPAU**: CTM Counter Pause control
0: Run
1: Pause

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

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

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

Bit 3 **CTON**: CTM Counter On/Off control
0: Off
1: On

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

Bit 2~0 **CTRP2~CTRP0**: CTM CCRP 3-bit register, compared with the CTM counter bit 9 ~ bit 7

Comparator P match period =
0: 1024 CTM clocks
1~7: (1~7)×128 CTM clocks

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

- **CTMC1 Register**

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

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

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

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

Compare Match Output Mode
00: No change
01: Output low
10: Output high
11: Toggle output

PWM Output Mode
00: PWM output inactive state
01: PWM output active state
10: PWM output
11: Undefined

Timer/Counter Mode
Unused

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

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

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

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

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

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

Bit 1 **CTDPX**: CTM PWM duty/period control
0: CCRP – period; CCRA – duty
1: CCRP – duty; CCRA – period

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

Bit 0 **CTCCLR**: CTM Counter Clear condition selection
0: Comparator P match
1: Comparator A match

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

- **CTMDL Register**

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

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

• **CTMDH Register**

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

Bit 7~2 Unimplemented, read as "0"

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

• **CTMAL Register**

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

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

• **CTMAH Register**

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

Bit 7~2 Unimplemented, read as "0"

Bit 1~0 **D9~D8**: CTM CCRA High Byte Register bit 7 ~ bit 0
CTM 10-bit CCRA bit 9 ~ bit 8

## Compact Type TM Operation Modes

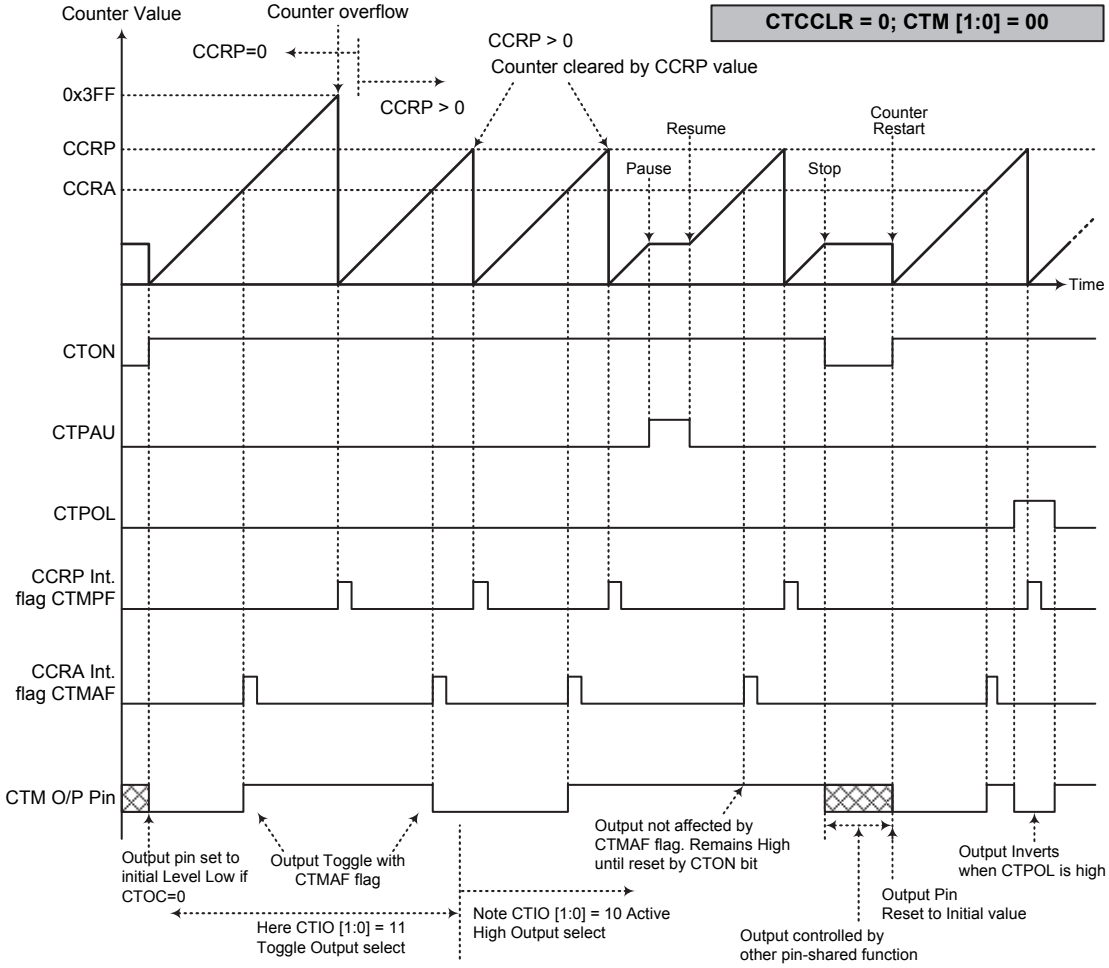
The Compact Type TM can operate in one of three operating modes, Compare Match Output Mode, PWM Output Mode or Timer/Counter Mode. The operating mode is selected using the CTM1 and CTM0 bits in the CTMC1 register.

### Compare Match Output Mode

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

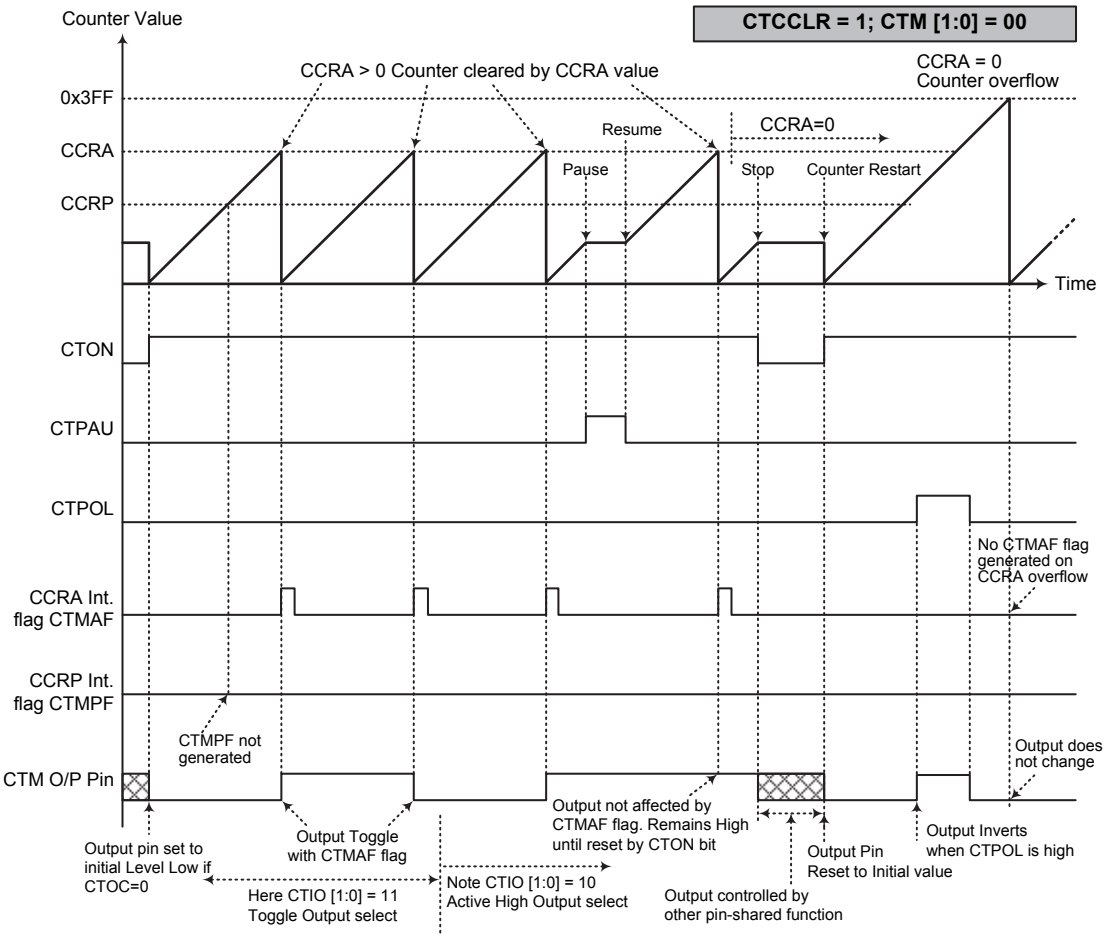
If the CTCCLR bit in the CTMC1 register is high then the counter will be cleared when a compare match occurs from Comparator A. However, here only the CTMAF interrupt request flag will be generated even if the value of the CCRP bits is less than that of the CCRA registers. Therefore when CTCCLR is high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

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




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

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




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

Note: 1. With CTCCLR=1 a Comparator A match will clear the counter

2. The CTM output pin is controlled only by the CTMAF flag

3. The output pin is reset to its initial state by a CTON bit rising edge

4. A CTMPF flag is not generated when CTCCLR=1

### Timer/Counter Mode

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

### PWM Output Mode

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

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

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

- **10-bit CTM, PWM Output Mode, Edge-aligned Mode, CTDPX=0**

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

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

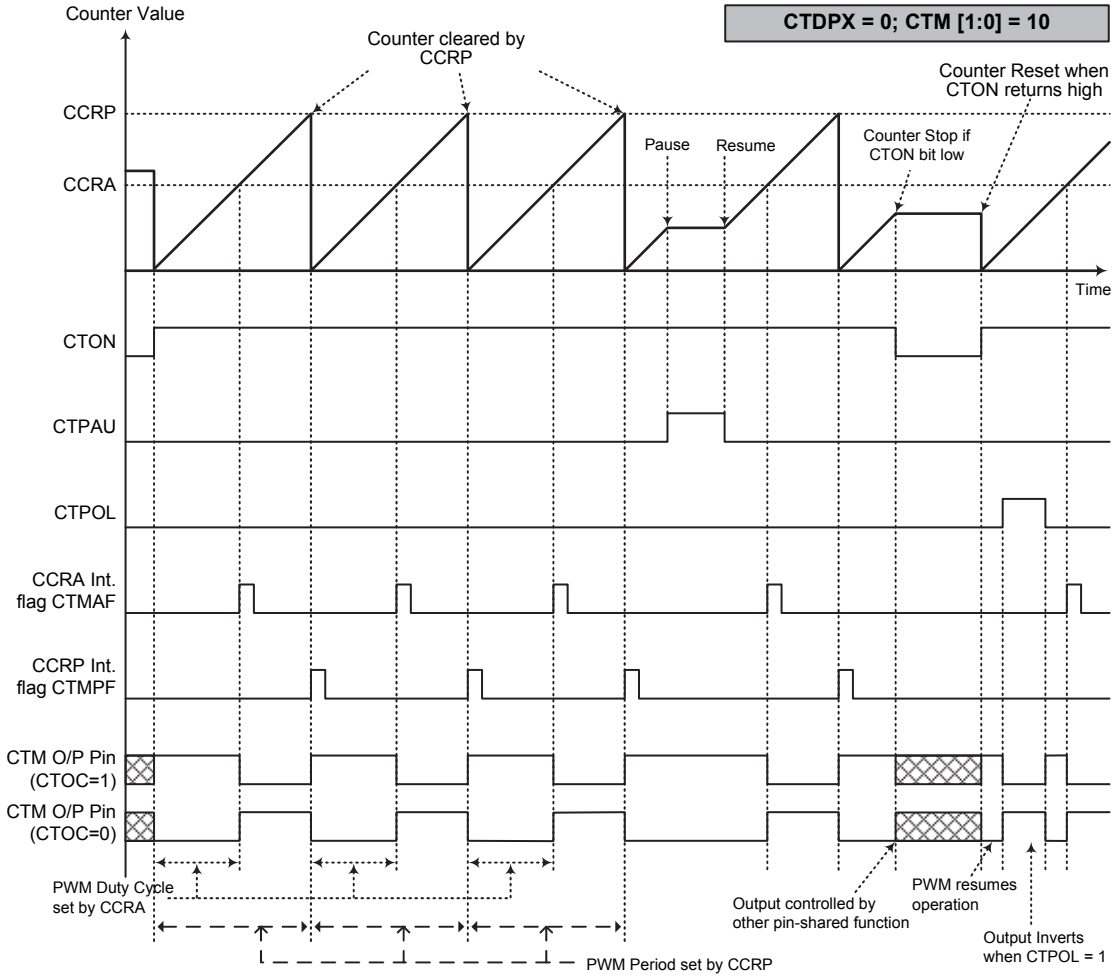
The CTM PWM output frequency=$(f_{SYS}/4)/(4\times128)=f_{SYS}/2048$=8kHz, duty=128/(4×128)=25%.

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

- **10-bit CTM, PWM Output Mode, Edge-aligned Mode, CTDPX=1**

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

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



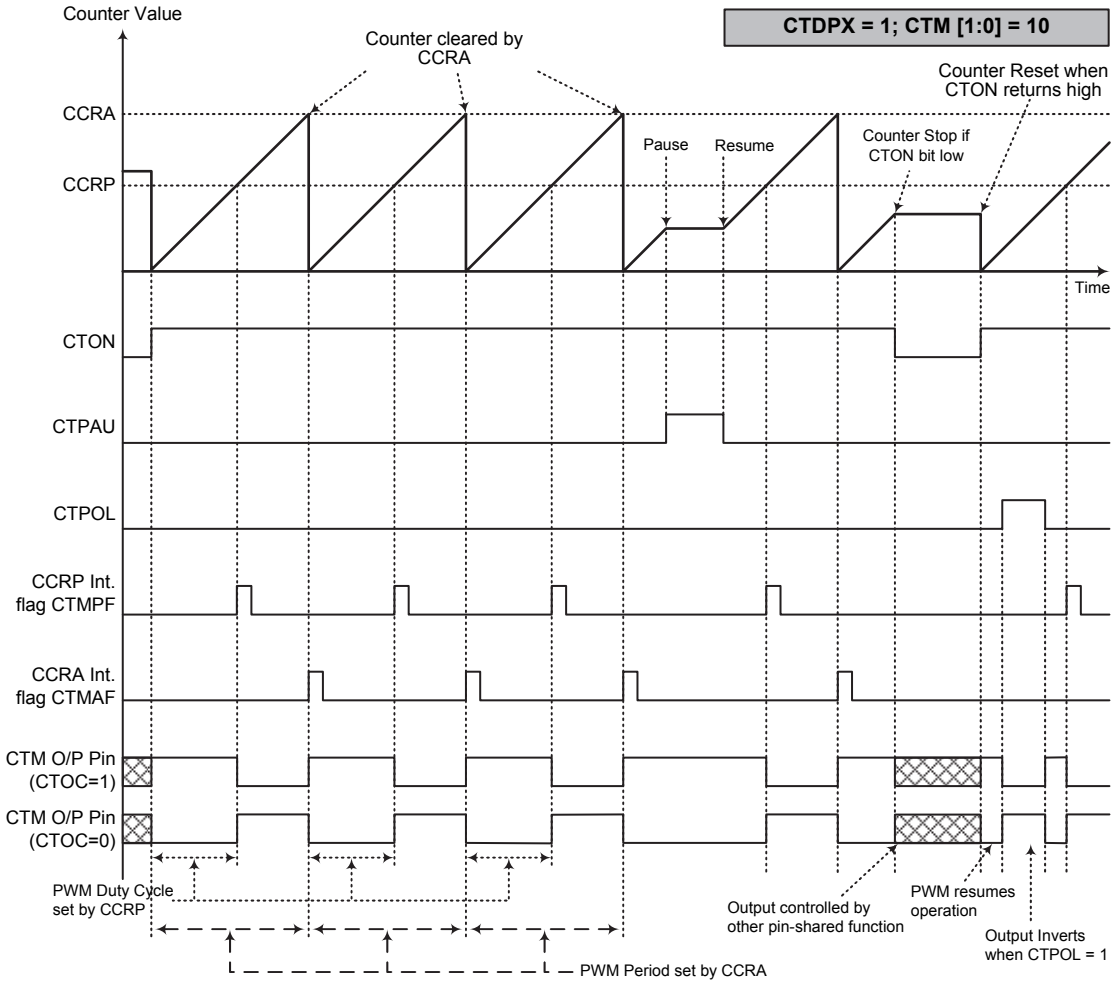

**PWM Output Mode – CTDPX=0**

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

2. A counter clear sets the PWM Period

3. The internal PWM function continues running even when CTIO[1:0]=00 or 01

4. The CTCCLR bit has no influence on PWM operation




**PWM Output Mode – CTDPX=1**

Note: 1. Here CTDPX=1 – Counter cleared by CCRA

2. A counter clear sets the PWM Period

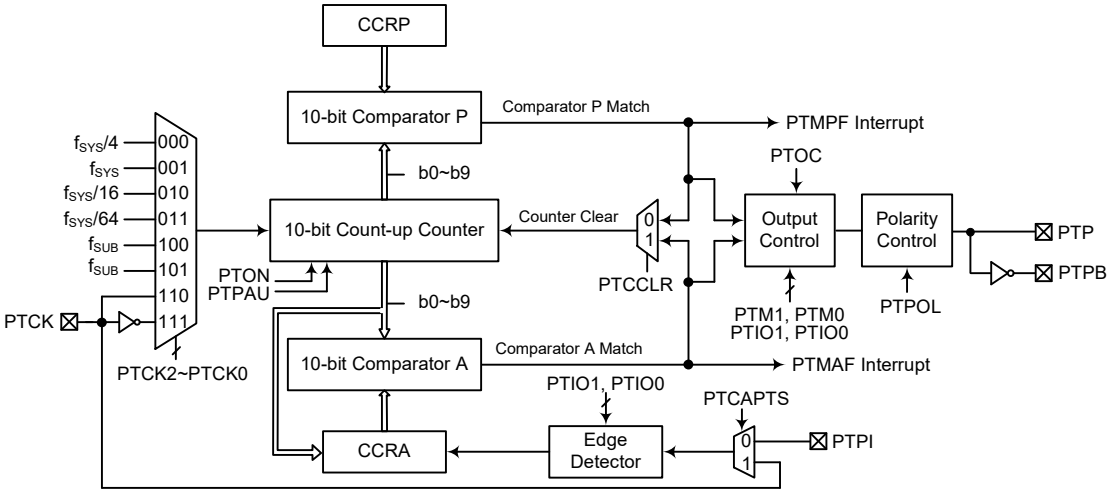
3. The internal PWM function continues even when CTIO[1:0]=00 or 01

4. The CTCCLR bit has no influence on PWM operation

## 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 Type TM can also be controlled with two external input pins and can drive two external output pins.

| Device | PTM Core | PTM Input Pin | PTM Output Pin |
|---|---|---|---|
| BS84B08C<br>BS84C12C | 10-bit PTM | PTCK, PTPI | PTP, PTPB |




**Periodic Type TM Block Diagram**

### Periodic Type TM Operation

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

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

### Periodic Type TM Register Description

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

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

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

- **PTMC0 Register**

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

Bit 7 **PTPAU**: PTM Counter Pause control
0: Run
1: Pause

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

Bit 6~4 **PTCK2~PTCK0**: Select PTM Counter clock
000: $f_{SYS}/4$
001: $f_{SYS}$
010: $f_H/16$
011: $f_H/64$
100: $f_{SUB}$
101: $f_{SUB}$
110: PTCK rising edge clock
111: PTCK falling edge clock

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

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

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

Bit 2~0 Unimplemented, read as “0”

• **PTMC1 Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | PTM1 | PTM0 | PTIO1 | PTIO0 | PTOC | PTPOL | PTCAPTS | PTCCLR |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| POR | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Bit 7~6 **PTM1~PTM0**: Select PTM Operating Mode
00: Compare Match Output Mode
01: Capture Input Mode
10: PWM Output Mode or Single Pulse Output Mode
11: Timer/Counter Mode

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

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

Compare Match Output Mode
00: No change
01: Output low
10: Output high
11: Toggle output

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

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

Timer/Counter Mode
Unused

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

In the Compare Match Output Mode, the PTIO1 and PTIO0 bits determine how the PTM output pin changes state when a compare match occurs from the Comparator A. The PTM output pin can be setup to switch high, switch low or to toggle its present state when a compare match occurs from the Comparator A. When the bits are both zero, then no change will take place on the output. The initial value of the PTM output pin should be setup using the PTOC bit in the PTMC1 register. Note that the output level requested by the PTIO1 and PTIO0 bits must be different from the initial value setup using the PTOC bit otherwise no change will occur on the PTM output pin when a compare match occurs. After the PTM output pin changes state, it can be reset to its initial level by changing the level of the PTON bit from low to high.

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

Bit 3 **PTOC**: PTM PTP Output control

Compare Match Output Mode
0: Initial low
1: Initial high

PWM Output Mode/Single Pulse Output Mode
0: Active low
1: Active high

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

Bit 2 **PTPOL**: PTM PTP Output polarity control
0: Non-invert
1: Invert

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

Bit 1 **PTCAPTS**: PTM Capture Trigger Source selection
0: From PTPI pin
1: From PTCK pin

Bit 0 **PTCCLR**: PTM Counter Clear condition selection
0: Comparator P match
1: Comparator A match

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

- **PTMDL Register**

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

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

- **PTMDH Register**

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

Bit 7~2 Unimplemented, read as “0”

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

• **PTMAL Register**

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

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

• **PTMAH Register**

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

Bit 7~2 Unimplemented, read as "0"

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

• **PTMRPL Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | PTRP7 | PTRP6 | PTRP5 | PTRP4 | PTRP3 | PTRP2 | PTRP1 | PTRP0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| POR | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Bit 7~0 **PTRP7~PTRP0**: PTM CCRP Low Byte Register bit 7 ~ bit 0
PTM 10-bit CCRP bit 7 ~ bit 0

• **PTMRPH Register**

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

Bit 7~2 Unimplemented, read as "0"

Bit 1~0 **PTRP9~PTRP8**: PTM CCRP High Byte Register bit 1 ~ bit 0
PTM 10-bit CCRP bit 9 ~ bit 8

## Periodic Type TM Operation Modes

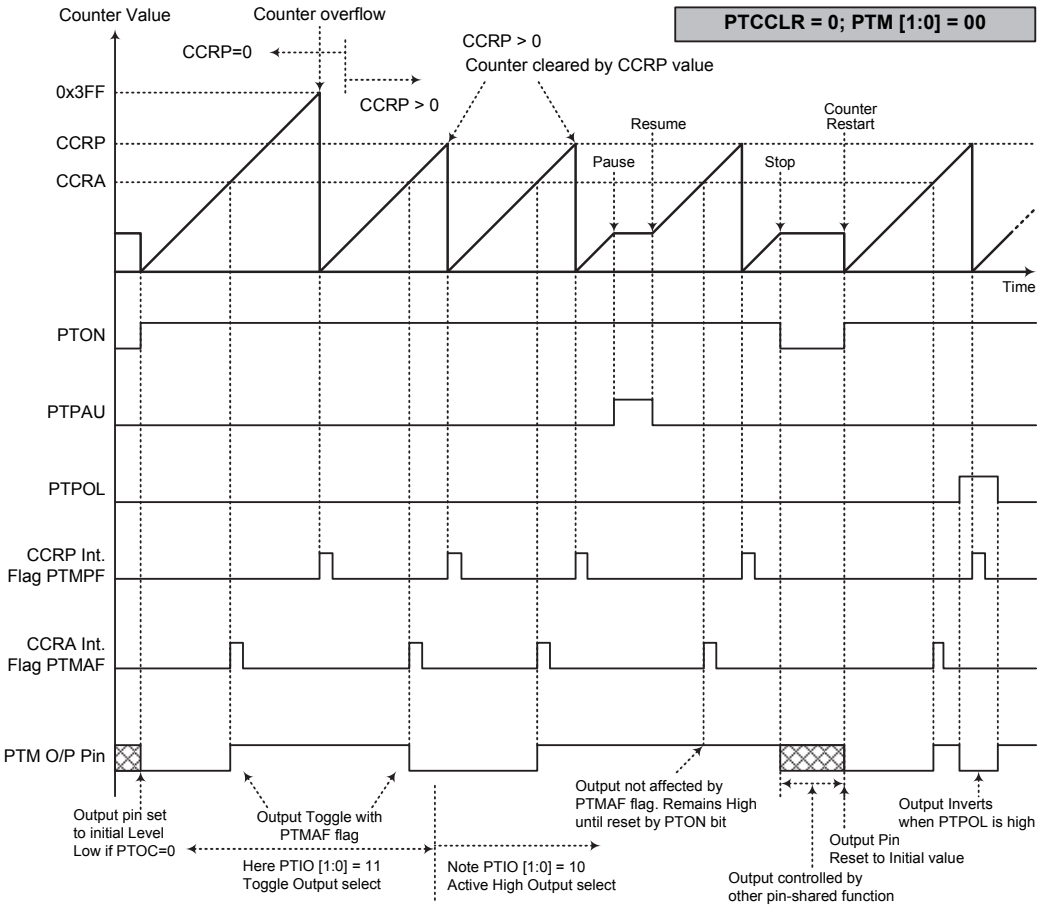
The Periodic Type TM can operate in one of five operating modes, Compare Match Output Mode, PWM Output Mode, Single Pulse Output Mode, Capture Input Mode or Timer/Counter Mode. The operating mode is selected using the PTM1 and PTM0 bits in the PTMC1 register.

### Compare Match Output Mode

To select this mode, bits PTM1 and PTM0 in the PTMC1 register, should be set to 00 respectively. In this mode once the counter is enabled and running it can be cleared by three methods. These are a counter overflow, a compare match from Comparator A and a compare match from Comparator P. When the PTCCLR bit is low, there are two ways in which the counter can be cleared. One is when a compare match from Comparator P, the other is when the CCRP bits are all zero which allows the counter to overflow. Here both PTMAF and PTMPF interrupt request flags for Comparator A and Comparator P respectively, will both be generated.

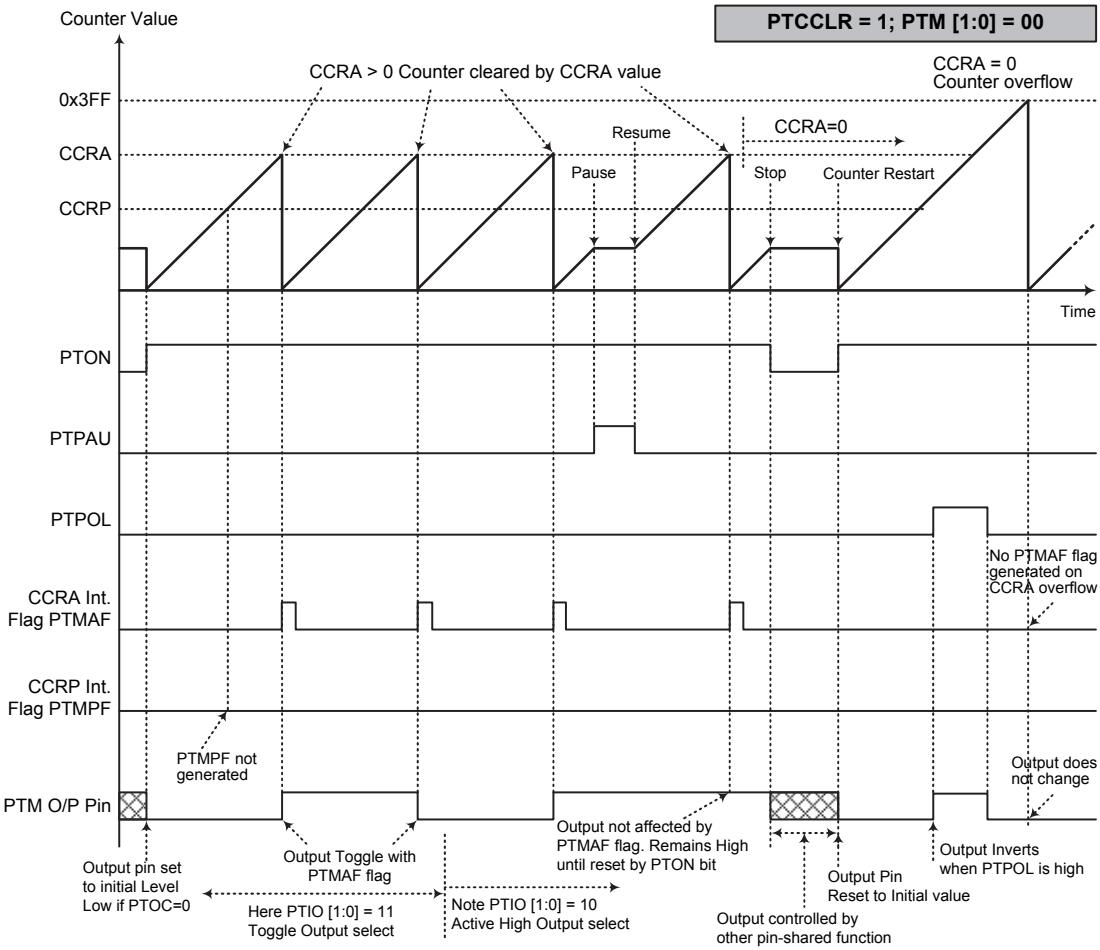
If the PTCCLR bit in the PTMC1 register is high then the counter will be cleared when a compare match occurs from Comparator A. However, here only the PTMAF interrupt request flag will be generated even if the value of the CCRP bits is less than that of the CCRA registers. Therefore when PTCCLR is high no PTMPF interrupt request flag will be generated. In the Compare Match Output Mode, the CCRA cannot be cleared to zero. If the CCRA bits are all zero, the counter will overflow when it reaches its maximum 10-bit, 3FF Hex, value, however here the PTMAF interrupt request flag will not be generated.

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




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

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




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

Note: 1. With PTCCLR=1, a Comparator A match will clear the counter

2. The PTM output pin is controlled only by the PTMAF flag

3. The output pin is reset to its initial state by a PTON bit rising edge

4. A PTMPF flag is not generated when PTCCLR =1

### Timer/Counter Mode

To select this mode, bits PTM1 and PTM0 in the PTMC1 register should be set to 11 respectively. The Timer/Counter Mode operates in an identical way to the Compare Match Output Mode generating the same interrupt flags. The exception is that in the Timer/Counter Mode the PTM output pin is not used. Therefore the above description and Timing Diagrams for the Compare Match Output Mode can be used to understand its function. As the PTM output pin is not used in this mode, the pin can be used as a normal I/O pin or other pin-shared function.

### PWM Output Mode

To select this mode, bits PTM1 and PTM0 in the PTMC1 register should be set to 10 respectively and also the PTIO1 and PTIO0 bits should be set to 10 respectively. The PWM function within the PTM is useful for applications which require functions such as motor control, heating control, illumination control, etc. By providing a signal of fixed frequency but of varying duty cycle on the PTM output pin, a square wave AC waveform can be generated with varying equivalent DC RMS values.

As both the period and duty cycle of the PWM waveform can be controlled, the choice of generated waveform is extremely flexible. In the PWM Output Mode, the PTCCLR bit has no effect as the PWM period. Both of the CCRP and CCRA registers are used to generate the PWM waveform, one register is used to clear the internal counter and thus control the PWM waveform frequency, while the other one is used to control the duty cycle. The PWM waveform frequency and duty cycle can therefore be controlled by the values in the CCRA and CCRP registers.

An interrupt flag, one for each of the CCRA and CCRP, will be generated when a compare match occurs from either Comparator A or Comparator P. The PTOC bit in the PTMC1 register is used to select the required polarity of the PWM waveform while the two PTIO1 and PTIO0 bits are used to enable the PWM output or to force the PTM output pin to a fixed high or low level. The PTPOL bit is used to reverse the polarity of the PWM output waveform.

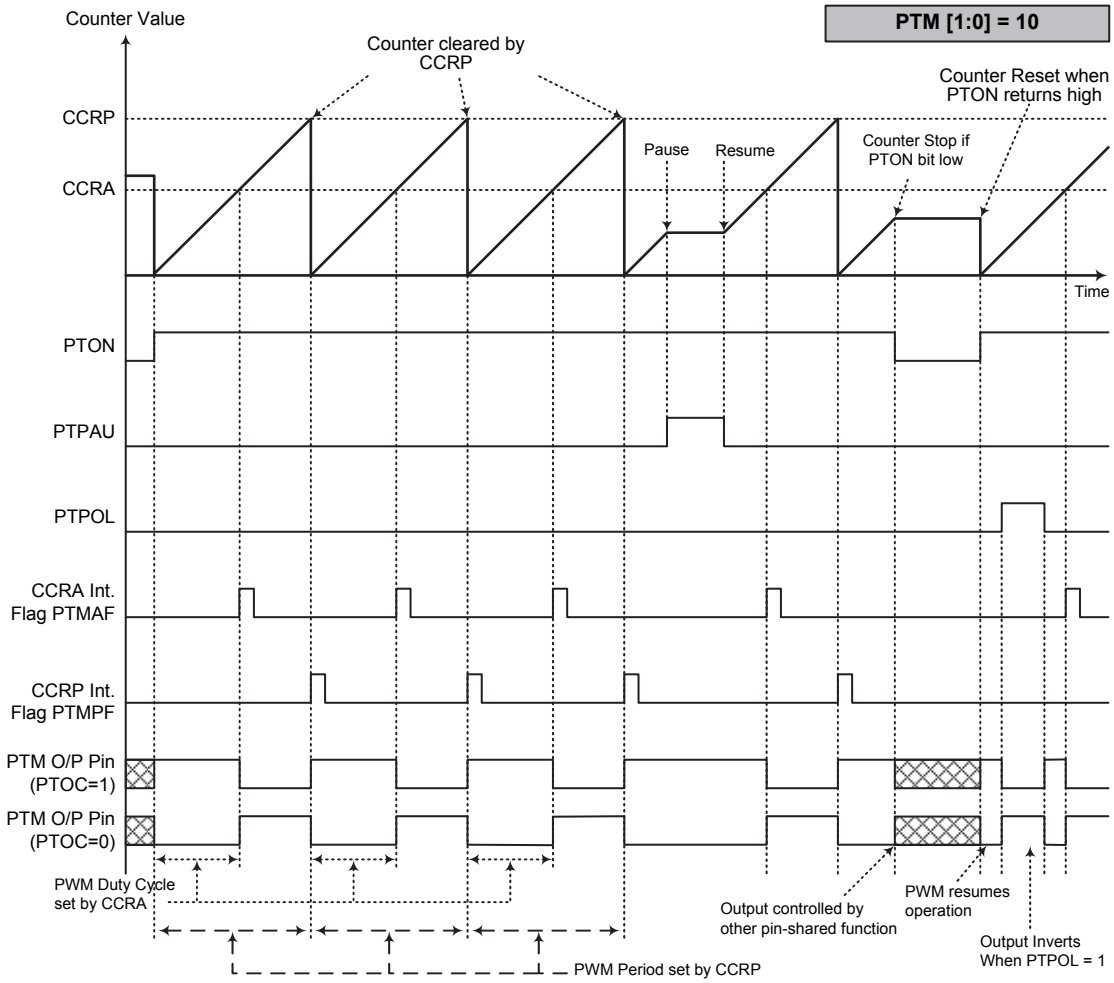
- **10-bit PTM, PWM Output Mode, Edge-aligned Mode**

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

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

The PTM PWM output frequency=($f_{SYS}/4$)/512=$f_{SYS}/2048$=8kHz, duty=128/512=25%,

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




**PWM Output Mode**

Note: 1. The counter is cleared by CCRP

2. A counter clear sets the PWM Period

3. The internal PWM function continues running even when PTIO[1:0]=00 or 01

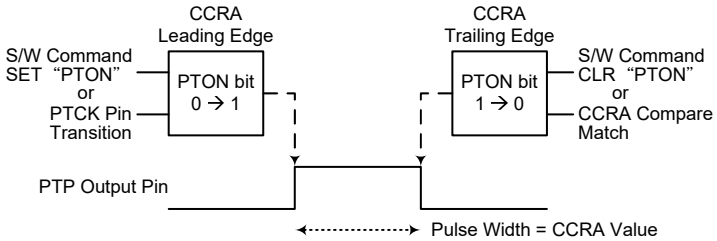
4. The PTCCLR bit has no influence on PWM operation

### Single Pulse Output Mode

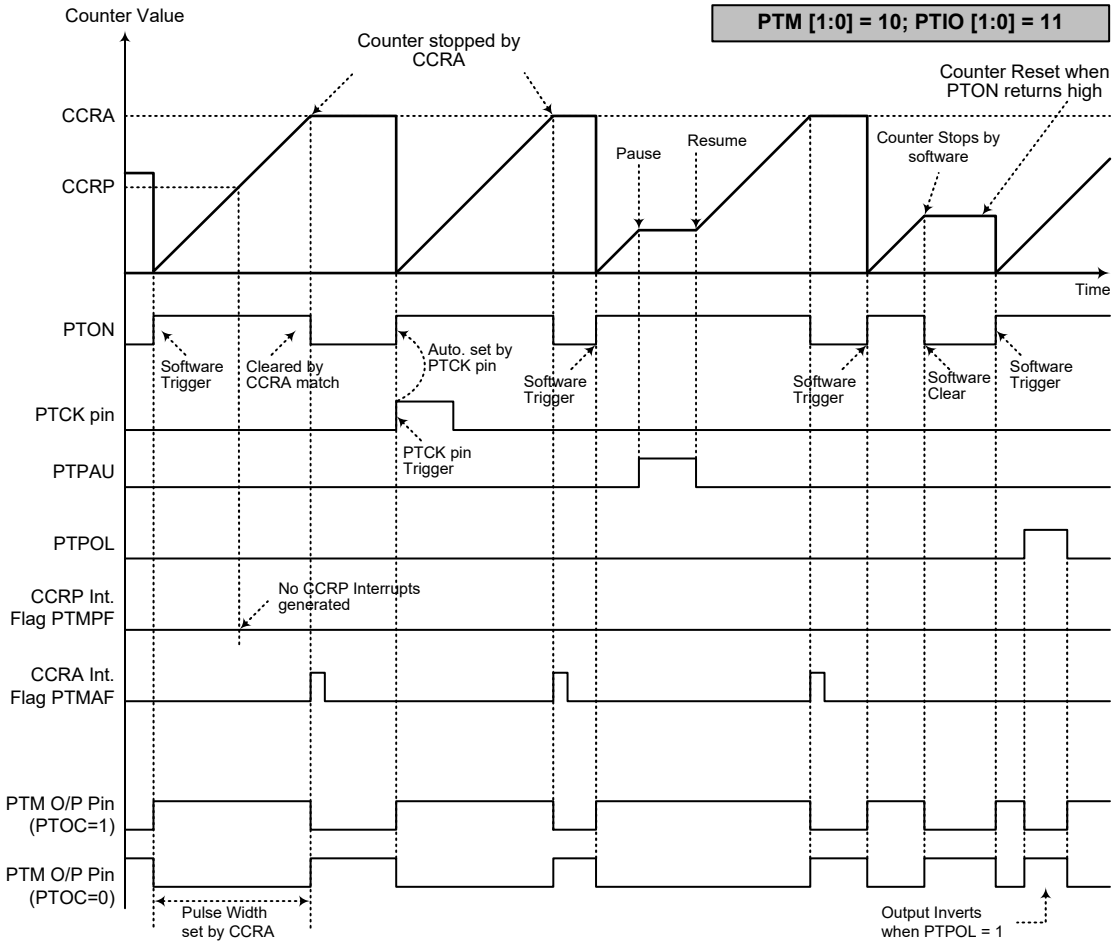
To select this mode, bits PTM1 and PTM0 in the PTMC1 register should be set to 10 respectively and also the PTIO1 and PTIO0 bits should be set to 11 respectively. The Single Pulse Output Mode, as the name suggests, will generate a single shot pulse on the PTM output pin.

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

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




**Single Pulse Generation**




**Single Pulse Output Mode**

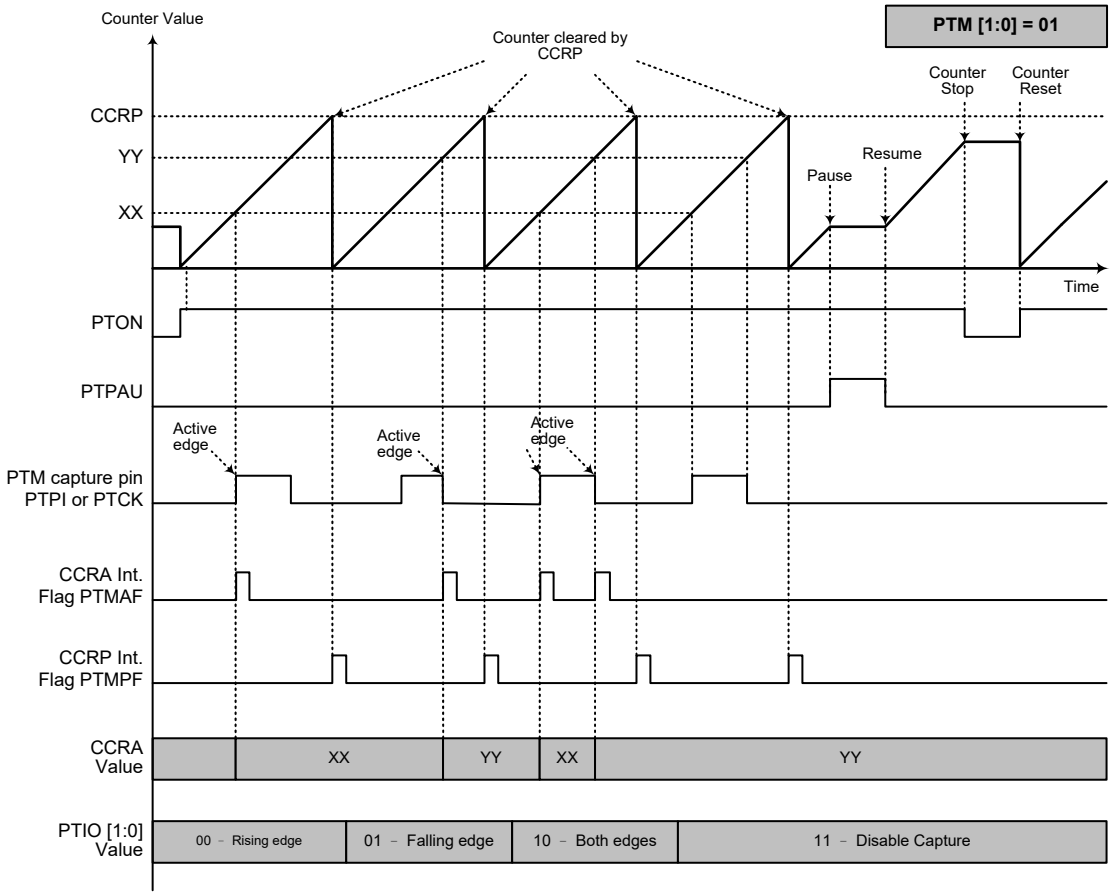
Note: 1. Counter stopped by CCRA
2. CCRP is not used
3. The pulse triggered by the PTCK pin or by setting the PTON bit high
4. A PTCK pin active edge will automatically set the PTON bit high
5. In the Single Pulse Output Mode, PTIO [1:0] must be set to “11” and cannot be changed

### Capture Input Mode

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

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

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




**Capture Input Mode**

Note: 1. PTM[1:0]=01 and active edge set by the PTIO[1:0] bits

2. A PTM Capture input pin active edge transfers the counter value to CCRA

3. PTCCLR bit not used

4. No output function – PTOC and PTPOL bits are not used

5. CCRP determines the counter value and the counter has a maximum count value when CCRP is equal to zero

## Analog to Digital Converter

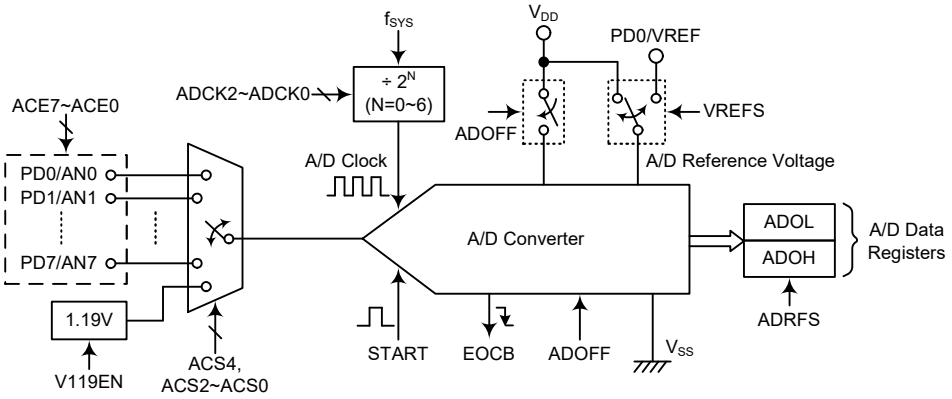
The need to interface to real world analog signals is a common requirement for many electronic systems. However, to properly process these signals by a microcontroller, they must first be converted into digital signals by A/D converters. By integrating the A/D conversion electronic circuitry into the 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 Overview

Each 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.

| Device | Input Channels | A/D Channel Select Bits | Input Pins |
|---|---|---|---|
| BS84B08C<br>BS84C12C | 8 | ACS4, ACS2~ACS0 | AN0~AN7 |

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




**A/D Converter Structure**

### A/D Converter Register Description

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

| Register Name | Bit | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| ADRL (ADRFS=0) | D3 | D2 | D1 | D0 | — | — | — | — |
| ADRL (ADRFS=1) | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| ADRH (ADRFS=0) | D11 | D10 | D9 | D8 | D7 | D6 | D5 | D4 |
| ADRH (ADRFS=1) | — | — | — | — | D11 | D10 | D9 | D8 |
| ADCR0 | START | EOCB | ADOFF | ADRFS | — | ACS2 | ACS1 | ACS0 |
| ADCR1 | ACS4 | V119EN | — | VREFS | — | ADCK2 | ADCK1 | ADCK0 |
| ACERL | ACE7 | ACE6 | ACE5 | ACE4 | ACE3 | ACE2 | ACE1 | ACE0 |

**A/D Converter Register List**

## A/D Converter Data Registers – ADRL, ADRH

As the devices contain an internal 12-bit A/D converter, they require two data registers to store the converted value.

These are a high byte register, known as ADRH, and a low byte register, known as ADRL. 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 ADRFS bit in the ADCR0 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.

| ADRFS | ADRH | | | | | | | | ADRL | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 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 Data Registers**

## A/D Converter Control Registers – ADCR0, ADCR1, ACERL

To control the function and operation of the A/D converter, three control registers known as ADCR0, ADCR1 and ACERL are provided. These 8-bit registers define functions such as the selection of which analog channel is connected to the internal A/D converter, the digitised data format, the A/D clock source as well as controlling the start function and monitoring the A/D converter end of conversion status. The ACS2~ACS0 bits in the ADCR0 register and ACS4 bit is the ADCR1 register define the A/D Converter input channel number. As the devices contain only one actual analog to digital converter hardware circuit, each of the individual 8 analog inputs must be routed to the converter. It is the function of the ACS4 and ACS2~ACS0 bits to determine which analog channel input pins or internal 1.19V is actually connected to the internal A/D converter.

The ACERL control register contain the ACE7~ACE0 bits which determine which pins on Port D are used as analog inputs for the A/D converter input and which pins are not to be used as the A/D converter input. Setting the corresponding bit high will select the A/D input function, clearing the bit to zero will select either the I/O or other pin-shared function. 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 resistors connected to these pins will be automatically removed if the pin is selected to be an A/D input.

### • ADCR0 Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | START | EOCB | ADOFF | ADRFS | — | ACS2 | ACS1 | ACS0 |
| R/W | R/W | R | R/W | R/W | — | R/W | R/W | R/W |
| POR | 0 | 1 | 1 | 0 | — | 0 | 0 | 0 |

Bit 7 **START**: Start the A/D conversion
0 → 1 → 0: Start
0 → 1: reset the A/D converter and set EOCB to “1”
This bit is used to initiate an A/D conversion process. The bit is normally low but if set high and then cleared low again, the A/D converter will initiate a conversion process. When the bit is set high the A/D converter will be reset.

Bit 6 **EOCB**: End of A/D conversion flag
0: A/D conversion ended
1: A/D conversion in progress
This read only flag is used to indicate when an A/D conversion process has completed. When the conversion process is running, the bit will be high.

Bit 5 **ADOFF**: A/D Converter module power on/off control bit
0: A/D Converter module power on
1: A/D Converter module power off

This bit controls the power to the A/D internal function. This bit should be cleared to zero to enable the A/D converter. If the bit is set high then the A/D converter will be switched off reducing the device power consumption. As the A/D converter will consume a limited amount of power, even when not executing a conversion, this may be an important consideration in power sensitive battery powered applications.

Note: 1. it is recommended to set ADOFF=1 before entering IDLE/SLEEP Mode for saving power.
2. ADOFF=1 will power down the A/D Converter module.

Bit 4 **ADRFS**: A/D conversion Data Format Control
0: A/D conversion Data MSB is ADRH bit 7, LSB is ADRL bit 4
1: A/D conversion Data MSB is ADRH bit 3, LSB is ADRL bit 0

This bit controls the format of the 12-bit converted A/D value in the two A/D data registers. Details are provided in the A/D data register section.

Bit 3 Unimplemented, read as “0”

Bit 2~0 **ACS2~ACS0**: Select A/D channel (when ACS4 is “0”)
000: AN0
001: AN1
010: AN2
011: AN3
100: AN4
101: AN5
110: AN6
111: AN7

These are the A/D channel select control bits. As there is only one internal hardware A/D converter each of the eight A/D inputs must be routed to the internal converter using these bits. If bit ACS4 in the ADCR1 register is set high then the internal 1.19V will be routed to the A/D Converter.

**• ADCR1 Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | ACS4 | V119EN | — | VREFS | — | ADCK2 | ADCK1 | ADCK0 |
| R/W | R/W | R/W | — | R/W | — | R/W | R/W | R/W |
| POR | 0 | 0 | — | 0 | — | 0 | 0 | 0 |

Bit 7 **ACS4**: Select Internal 1.19V bandgap voltage as A/D Converter input
0: Disable
1: Enable

This bit enables the 1.19V bandgap voltage to be connected to the A/D converter. The V119EN bit must first have been set to enable the bandgap circuit 1.19V voltage to be used by the A/D converter. When the ACS4 bit is set high, the bandgap 1.19V voltage will be routed to the A/D converter and the other A/D input channels disconnected.

Bit 6 **V119EN**: Internal 1.19V Control
0: Disable
1: Enable

This bit controls the internal Bandgap circuit on/off function to the A/D converter. When the bit is set high the 1.19V bandgap voltage can be used as an A/D converter input. If the 1.19V bandgap voltage is not used by the A/D converter and the LVR function is disabled then the bandgap reference circuit will be automatically switched off to conserve power. When the 1.19V bandgap voltage is switched on for use by the A/D converter, a time $t_{BG}$ should be allowed for the bandgap circuit to stabilise before implementing an A/D conversion.

Bit 5 Unimplemented, read as “0”

Bit 4 **VREFS**: Select A/D Converter reference voltage
0: Internal A/D Converter power
1: VREF pin
This bit is used to select the reference voltage for the A/D converter. If the bit is high, then the A/D converter reference voltage is supplied on the external VREF pin. If the pin is low, then the internal reference is used which is taken from the power supply pin, VDD.

Bit 3 Unimplemented, read as "0"

Bit 2~0 **ADCK2~ADCK0**: Select A/D Converter clock source
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: Undefined
These three bits are used to select the clock source for the A/D converter.

**• ACERL Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | ACE7 | ACE6 | ACE5 | ACE4 | ACE3 | ACE2 | ACE1 | ACE0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| POR | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

Bit 7 **ACE7**: Define PD7 is A/D input or not
0: Not A/D input
1: A/D input, AN7

Bit 6 **ACE6**: Define PD6 is A/D input or not
0: Not A/D input
1: A/D input, AN6

Bit 5 **ACE5**: Define PD5 is A/D input or not
0: Not A/D input
1: A/D input, AN5

Bit 4 **ACE4**: Define PD4 is A/D input or not
0: Not A/D input
1: A/D input, AN4

Bit 3 **ACE3**: Define PD3 is A/D input or not
0: Not A/D input
1: A/D input, AN3

Bit 2 **ACE2**: Define PD2 is A/D input or not
0: Not A/D input
1: A/D input, AN2

Bit 1 **ACE1**: Define PD1 is A/D input or not
0: Not A/D input
1: A/D input, AN1

Bit 0 **ACE0**: Define PD0 is A/D input or not
0: Not A/D input
1: A/D input, AN0

## A/D Operation

The START bit in the ADCR0 register is used to start and reset the A/D converter. When the microcontroller sets this bit from low to high and then low again, an analog to digital conversion cycle will be initiated. When the START bit is brought from low to high but not low again, the EOCB bit in the ADCR0 register will be set high and the analog to digital converter will be reset. It is the START bit that is used to control the overall start operation of the internal analog to digital converter.

The EOCB bit in the ADCR0 register is used to indicate when the analog to digital conversion process is complete. This bit will be automatically set to "0" by the microcontroller after a conversion cycle has ended. In addition, the corresponding A/D interrupt request flag will be set in the interrupt control register, and if the interrupts are enabled, an appropriate internal interrupt signal will be generated. This A/D internal interrupt signal will direct the program flow to the associated A/D internal interrupt address for processing. If the A/D internal interrupt is disabled, the microcontroller can be used to poll the EOCB bit in the ADCR0 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 ADCK2~ADCK0 bits in the ADCR1 register.

Although the A/D clock source is determined by the system clock, $f_{SYS}$, and by bits ADCK2~ADCK0, there are some limitations on the A/D clock source speed that can be selected. As the minimum value of permissible A/D clock period, $t_{ADCK}$, is from 0.5μs to 10μs, 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 * show where, special care must be taken, as the values may be beyond the specified A/D Clock Period range.

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

**A/D Clock Period Examples**

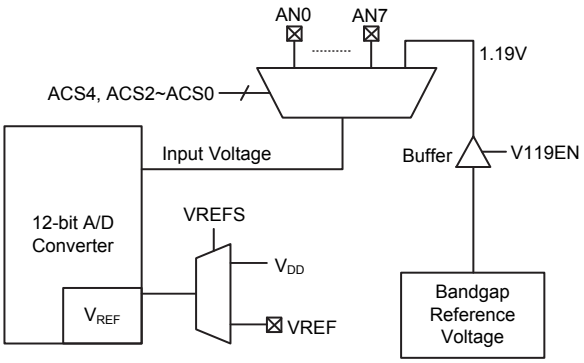
Controlling the power on/off function of the A/D converter circuitry is implemented using the ADOFF bit in the ADCR0 register. This bit must be zero to power on the A/D converter. Even if no pins are selected for use as A/D inputs by clearing the ACE7~ACE0 bits in the ACERL register, if the ADOFF bit is zero then some power will still be consumed. In power conscious applications it is therefore recommended that the ADOFF is set high to reduce power consumption when the A/D converter function is not being used.

The reference voltage supply to the A/D Converter can be supplied from either the positive power supply, VDD, or from an external reference sources supplied on pin VREF. The desired selection is made using the VREFS bit. As the VREF pin is pin-shared with other functions, when the VREFS bit is set high, the VREF pin function will be selected and the other pin functions will be disabled automatically.

## A/D Input Pins

All of the A/D analog input pins are pin-shared with the I/O pins on Port D as well as other functions. The ACE7~ACE0 bits in the ACERL register, determine whether the input pins are setup as A/D converter analog inputs or whether they have other functions. If the ACE7~ACE0 bits for its corresponding pin is set high then the pin will be setup to be an A/D converter input and the original pin functions 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 corresponding port control registers to enable the A/D input as when the ACE7~ACE0 bits enable an A/D input, the status of the port control register will be overridden.

The A/D converter has its own reference voltage pin, VREF, however the reference voltage can also be supplied from the power supply pin, a choice which is made through the VREFS bit in the ADCR1 register. The analog input values must not be allowed to exceed the value of the selected reference voltage.




**A/D Input Structure**

## Summary of A/D Conversion Steps

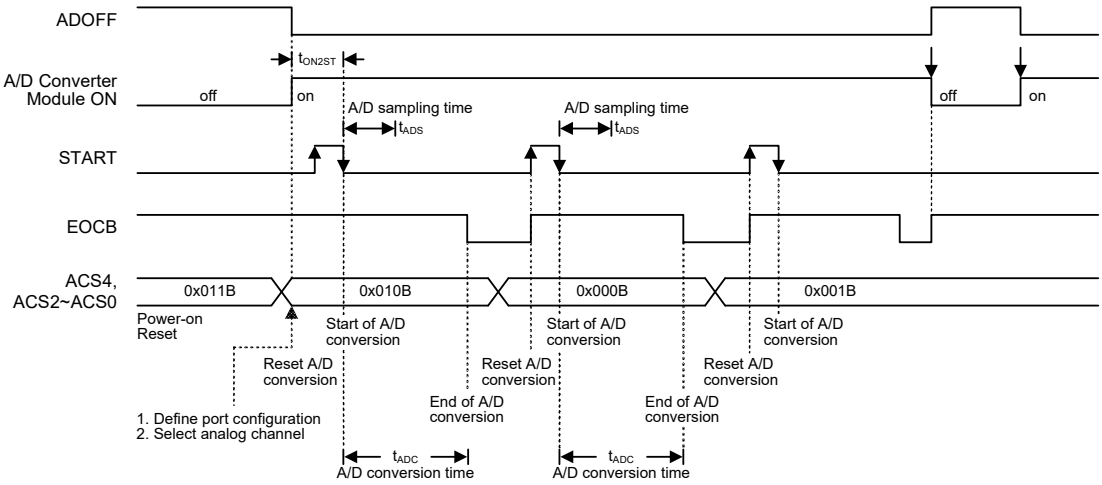
The following summarises the individual steps that should be executed in order to implement an A/D conversion process.

- Step 1
  Select the required A/D conversion clock by correctly programming bits ADCK2~ADCK0 in the ADCR1 register.
- Step 2
  Enable the A/D by clearing the ADOFF bit in the ADCR0 register to zero.
- Step 3
  Select which channel is to be connected to the internal A/D converter by correctly programming the ACS4, ACS2~ACS0 bits which are also contained in the ADCR1 and ADCR0 register.
- Step 4
  Select which pins are to be used as A/D inputs and configure them by correctly programming the ACE7~ACE0 bits in the ACERL register.
- Step 5
  If the interrupts are to be used, the interrupt control registers must be correctly configured to ensure the A/D converter interrupt function is active. The master interrupt control bit, EMI, and the A/D converter interrupt bit, ADE, must both be set high to do this.

- Step 6
  The analog to digital conversion process can now be initialised by setting the START bit in the ADCR0 register from low to high and then low again. Note that this bit should have been originally cleared to zero.
- Step 7
  To check when the analog to digital conversion process is complete, the EOCB bit in the ADCR0 register can be polled. The conversion process is complete when this bit goes low. When this occurs the A/D data registers ADRL and ADRH can be read to obtain the conversion value. As an alternative method, if the interrupts are enabled and the stack is not full, the program can wait for an A/D interrupt to occur.

  Note: When checking for the end of the conversion process, if the method of polling the EOCB bit in the ADCR0 register is used, the interrupt enable step above can be omitted.

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




**A/D Conversion Timing**

## 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 ADOFF high in the ADCR0 register. When this happens, the internal A/D converter circuits will not consume power irrespective of what analog voltage is applied to their input lines. If the A/D converter input lines are used as normal I/O pins, then care must be taken as if the input voltage is not at a valid logic level, then this may lead to some increase in power consumption.

## A/D Transfer Function

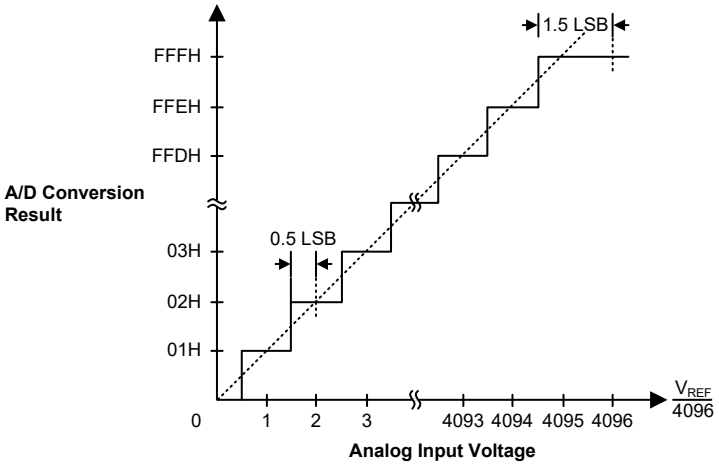
As the devices contain a 12-bit A/D converter, its full-scale converted digitised value is equal to 0FFFH. Since the full-scale analog input value is equal to the actual A/D converter reference voltage, $V_{REF}$, this gives a single bit analog input value of reference voltage value divided by 4096.

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

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

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

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




**Ideal A/D Transfer Function**

## A/D Programming Example

The following two programming examples illustrate how to setup and implement an A/D conversion. In the first example, the method of polling the EOCB bit in the ADCR0 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 EOCB polling method to detect the end of conversion**

```
clr ADE                 ; disable A/D Converter interrupt
mov a,03H
mov ADCR1,a             ; select fSYS/8 as A/D clock and switch off 1.19V
clr ADOFF
mov a,0Fh               ; setup ACERL to configure pins AN0~AN3
mov ACERL,a
mov a,00h
mov ADCR0,a             ; enable and connect AN0 channel to A/D converter
:
start_conversion:
clr START               ; high pulse on start bit to initiate conversion
set START               ; reset A/D
clr START               ; start A/D
polling_EOC:
sz EOCB                 ; poll the ADCR0 register EOCB bit to detect end of A/D conversion
```

```
jmp polling_EOC          ; continue polling
mov a,ADRL               ; read low byte conversion result value
mov adrl_buffer,a        ; save result to user defined register
mov a,ADRH               ; read high byte conversion result value
mov adrh_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 A/D Converter interrupt
mov a,03H
mov ADCR1,a              ; select fSYS/8 as A/D clock and switch off 1.19V
clr ADOFF
mov a,0Fh                ; setup ACERL to configure pins AN0~AN3
mov ACERL,a
mov a,00h
mov ADCR0,a              ; enable and connect AN0 channel to A/D converter
Start_conversion:
clr START                ; high pulse on START bit to initiate conversion
set START                ; reset A/D
clr START                ; start A/D
clr ADF                  ; clear A/D Converter interrupt request flag
set ADE                  ; enable A/D Converter interrupt
set EMI                  ; enable global interrupt
:
:
                         ; A/D Converter interrupt service routine
ADC_ISR:
mov acc_stack,a          ; save ACC to user defined memory
mov a,STATUS
mov status_stack,a       ; save STATUS to user defined memory
:
:
mov a,ADRL               ; read low byte conversion result value
mov adrl_buffer,a        ; save result to user defined register
mov a,ADRH               ; read high byte conversion result value
mov adrh_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
```

## Touch Key Function

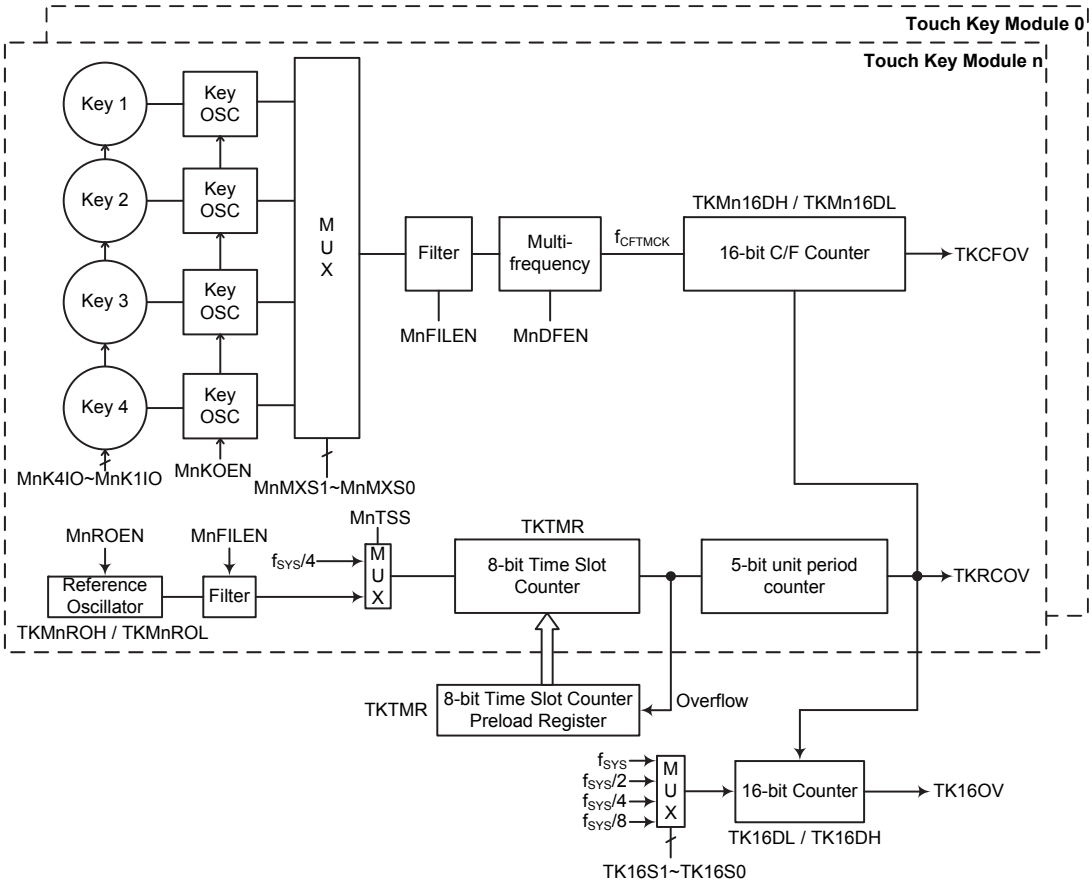
Each device provides multiple touch key functions. The touch key function is fully integrated and requires no external components, allowing touch key functions to be implemented by the simple manipulation of internal registers.

### 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 Mn. 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.

| Device | Total Key Number | Touch Key Module | | Touch Key | Shared I/O Pin |
|---|---|---|---|---|---|
| BS84B08C | 8 | Mn (n=0~1) | M0 | KEY1~KEY4 | PB0~PB3 |
| | | | M1 | KEY5~KEY8 | PB4~PB7 |
| BS84C12C | 12 | Mn (n=0~2) | M0 | KEY1~KEY4 | PB0~PB3 |
| | | | M1 | KEY5~KEY8 | PB4~PB7 |
| | | | M2 | KEY9~KEY12 | PC0~PC3 |

**Touch Key Structure**




Note: 1. The structure contained in the dash line is identical for each touch key module which contains four touch keys.
2. When MnTSS=0 and MnROEN=1 or when MnTSS=1, the touch key function 16-bit counter can operate normally.

**Touch Key Function Block Diagram (n=0~2)**

## 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 each touch key module. The Mn within the register name refers to the Touch Key module number. The series of devices has up to three Touch Key Modules dependent upon the selected device.

| 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 select low byte |
| TKMnROH | Touch key module n reference oscillator capacitor select high byte |
| TKMnC0 | Touch key module n Control register 0 |
| TKMnC1 | Touch key module n Control register 1 |

**Touch Key Function Register Definition (n=0~2)**

| Register Name | Bit | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| TKTMR | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| TKC0 | — | TKRCOV | TKST | TKCFOV | TK16OV | TSCS | TK16S1 | TK16S0 |
| TKC1 | — | — | — | — | — | — | 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 | MnMXS1 | MnMXS0 | MnDFEN | MnFILEN | MnSOFC | MnSOF2 | MnSOF1 | MnSOF0 |
| TKMnC1 | MnTSS | — | MnROEN | MnKOEN | MnK4IO | MnK3IO | MnK2IO | MnK1IO |

**Touch Key Function Register List (n=0~2)**

### • 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])×32 $t_{TSC}$, where $t_{TSC}$ is the time slot counter clock period.

• **TKC0 Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | — | TKRCOV | TKST | TKCFOV | TK16OV | TSCS | TK16S1 | TK16S0 |
| 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 Unimplemented, read as "0"

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 cannot be set by application program but must be cleared to 0 by application program.

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

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

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

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

Bit 2 **TSCS**: Touch key time slot counter select
0: Each touch key module uses its own time slot counter
1: All touch key modules use Module 0 time slot counter

Bit 1~0 **TK16S1~TK16S0**: Touch key function 16-bit counter clock source select
00: $f_{SYS}$
01: $f_{SYS}/2$
10: $f_{SYS}/4$
11: $f_{SYS}/8$

• **TKC1 Register**

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

Bit 7~2 Unimplemented, read as "0"

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

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

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

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

• **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. This register pair will be cleared to zero when the TKST bit is set low.

• **TKMnROH/TKMnROL – Touch Key Module n Reference Oscillator Capacitor Select 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 store the touch key module n reference oscillator capacitor value.

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

• **TKMnC0 Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | MnMXS1 | MnMXS0 | MnDFEN | MnFILEN | MnSOFC | MnSOF2 | MnSOF1 | MnSOF0 |
| 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 **MnMXS1~MnMXS0**: Multiplexer Key Select

| Bit | Touch Key Module Number | | |
|---|---|---|---|
| MnMXS[1:0] | M0 | M1 | M2 |
| 00 | KEY1 | KEY5 | KEY9 |
| 01 | KEY2 | KEY6 | KEY10 |
| 10 | KEY3 | KEY7 | KEY11 |
| 11 | KEY4 | KEY8 | KEY12 |
| BS84B08C | √ | √ | — |
| BS84C12C | √ | √ | √ |

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-to-F oscillator frequency hopping function control select
0: Controlled by the MnSOF2~MnSOF0
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 (MnSOFC=0)
000: 1.020MHz
001: 1.040MHz
010: 1.059MHz
011: 1.074MHz
100: 1.085MHz
101: 1.099MHz
110: 1.111MHz
111: 1.125MHz

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

The frequency mentioned here will be changed when the external or internal capacitor is with different values. If the touch key operates at 1MHz frequency, users can adjust the frequency in scale when any other frequency is selected.

• **TKMnC1 Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | MnTSS | — | MnROEN | MnKOEN | MnK4IO | MnK3IO | MnK2IO | MnK1IO |
| 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 select
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

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

Bit 3 **MnK4IO**: Touch key module n Key 4 enable control

| MnK4IO | Touch Key Module n – Mn | | |
|---|---|---|---|
| | M0 | M1 | M2 |
| 0: Disable | I/O or other functions | | |
| 1: Enable | KEY4 | KEY8 | KEY12 |
| BS84B08C | √ | √ | — |
| BS84C12C | √ | √ | √ |

Bit 2 **MnK3IO**: Touch key module n Key 3 enable control

| MnK3IO | Touch Key Module n – Mn | | |
|---|---|---|---|
| | M0 | M1 | M2 |
| 0: Disable | I/O or other functions | | |
| 1: Enable | KEY3 | KEY7 | KEY11 |
| BS84B08C | √ | √ | — |
| BS84C12C | √ | √ | √ |

Bit 1 **MnK2IO**: Touch key module n Key 2 enable control

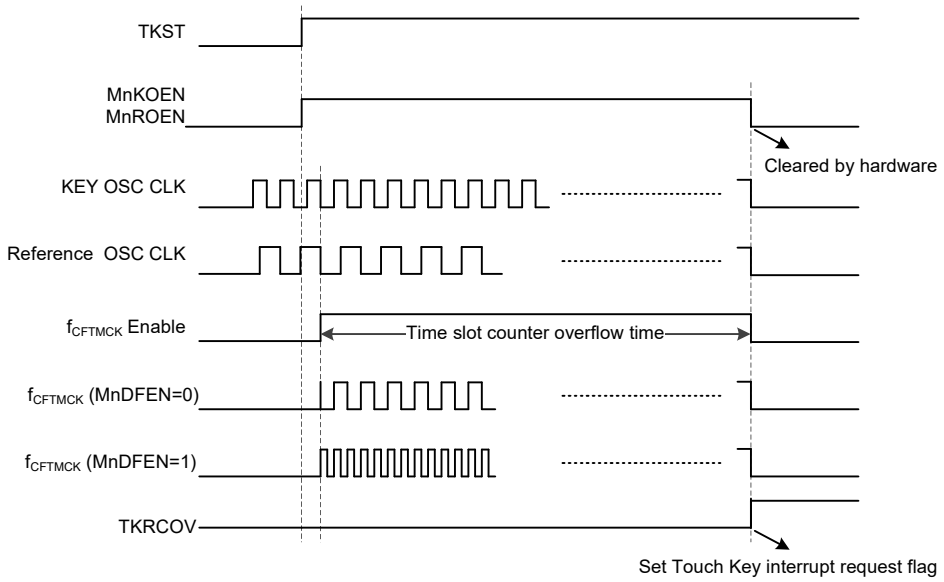
| MnK2IO | Touch Key Module n – Mn | | |
|---|---|---|---|
| | M0 | M1 | M2 |
| 0: Disable | I/O or other functions | | |
| 1: Enable | KEY2 | KEY6 | KEY10 |
| BS84B08C | √ | √ | — |
| BS84C12C | √ | √ | √ |

Bit 0 **MnK1IO**: Touch key module n Key 1 enable control

| MnK1IO | Touch Key Module n – Mn | | |
|---|---|---|---|
| | M0 | M1 | M2 |
| 0: Disable | I/O or other functions | | |
| 1: Enable | KEY1 | KEY5 | KEY9 |
| BS84B08C | √ | √ | — |
| BS84C12C | √ | √ | √ |

## 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 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 register bits. Each touch key has its own independent sense oscillator. Therefore, there are four sense oscillators within each 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.

Using the TSCS bit in the TKC0 register can select the module 0 time slot counter as the time slot counter for all modules. All modules use the same started 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 all modules 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 user. When the TKST bit changes from low to high, the 16-bit C/F counter, touch key function 16-bit counter, 5-bit time slot unit period counter and 8-bit time slot timer counter will be automatically switched on.

The key oscillator and reference oscillator in all modules will be automatically stopped and the 16-bit C/F counter, touch key function 16-bit counter, 5-bit time slot unit period counter and 8-bit time slot timer counter will be automatically switched off when the time slot counter overflows. The clock source for the time slot counter is sourced from the reference oscillator or $f_{SYS}/4$ which is selected using the 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, etc. Each touch key module has an identical structure.

## Touch Key Interrupt

The touch key only has single interrupt, 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. 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.

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. More details regarding the touch key interrupt is located in the interrupt section of the datasheet.

## Programming Considerations

After the relevant registers are setup, the touch key detection process is initiated by changing the TKST bit from low to high. This will enable and synchronise all relevant oscillators. The TKRCOV flag which is the time slot counter flag will go high when the counter overflows. When this happens an interrupt signal will be generated.

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

## Serial Interface Module – SIM

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

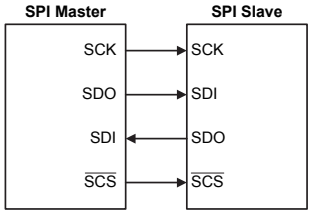
### SPI Interface

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

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

#### SPI Interface Operation

The SPI interface is a full duplex synchronous serial data link. It is a four line interface with pin names SDI, SDO, SCK and $\overline{SCS}$. Pins SDI and SDO are the Serial Data Input and Serial Data Output lines, the SCK pin is the Serial Clock line and $\overline{SCS}$ is the Slave Select line. As the SPI interface pins are pin-shared with normal I/O pins and with the $I^2C$ function pins, the SPI interface pins must first be selected by 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 $\overline{SCS}$ pin only one slave device can be utilized. The $\overline{SCS}$ pin is controlled by software, set CSEN bit to 1 to enable $\overline{SCS}$ pin function, set CSEN bit to 0 the $\overline{SCS}$ pin will be floating state.

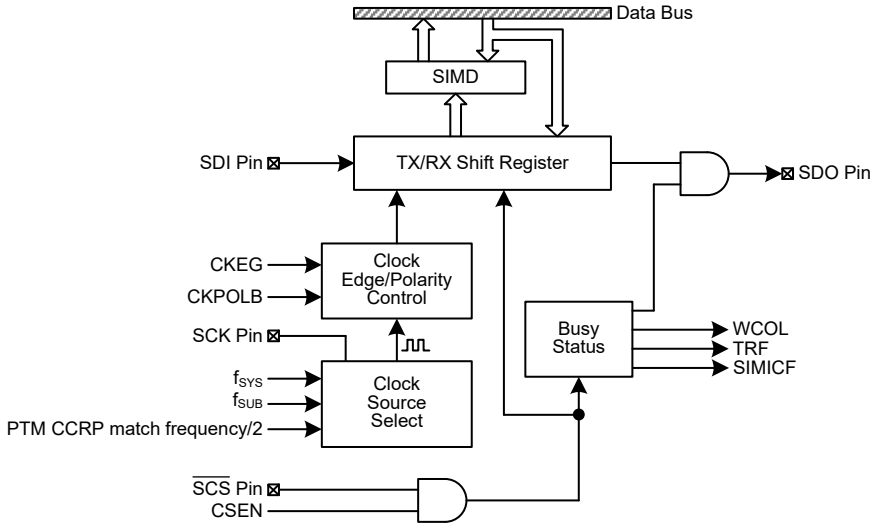



**SPI Master/Slave Connection**

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 Block Diagram**

## SPI Registers

There are three internal registers which control the overall operation of the SPI interface. These are the SIMD data register and two control registers, SIMC0 and SIMC2.

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

**SPI Register List**

### SPI Data Register

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

- **SIMD Register**

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

"x": unknown

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

### SPI Control Registers

There are also two control registers for the SPI interface, SIMC0 and SIMC2. Note that the SIMC2 register also has the name SIMA which is used by the I²C function. The SIMC0 register is used to control the enable/disable function and to set the data transmission clock frequency. The SIMC2 register is used for other control functions such as LSB/MSB selection, write collision flag etc.

- **SIMC0 Register**

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

Bit 7~5 **SIM2~SIM0**: SIM Operating Mode Control
000: SPI master mode; SPI clock is $f_{SYS}/4$
001: SPI master mode; SPI clock is $f_{SYS}/16$
010: SPI master mode; SPI clock is $f_{SYS}/64$
011: SPI master mode; SPI clock is $f_{SUB}$
100: SPI master mode; SPI clock is PTM CCRP match frequency/2
101: SPI slave mode
110: I²C slave mode
111: Unused mode

These bits setup the overall operating mode of the SIM 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 PTM and $f_{SUB}$. If the SPI Slave Mode is selected then the clock will be supplied by an external Master device.

Bit 4 Unimplemented, read as "0"

Bit 3~2 **SIMDEB1~SIMDEB0**: I²C Debounce Time Selection

These bits are only available when the SIM is configured to operate in the I²C mode. Refer to the I²C register section.

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

The bit is the overall on/off control for the SIM interface. When the SIMEN bit is cleared to zero to disable the SIM interface, the SDI, SDO, SCK and $\overline{SCS}$, or SDA and SCL lines will lose their SPI or I²C function and the SIM operating current will be reduced to a minimum value. When the bit is high the SIM interface is enabled. If the SIM is configured to operate as an SPI interface via the SIM2~SIM0 bits, the contents of the SPI control registers will remain at the previous settings when the SIMEN bit changes from low to high and should therefore be first initialised by the application program. If the SIM is configured to operate as an I²C interface via the 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**: SIM SPI Incomplete Flag
0: SIM SPI incomplete condition is not occurred
1: SIM SPI incomplete condition is occurred

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

• **SIMC2 Register**

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

Bit 7~6 **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 **MLS**: SPI data shift order
0: LSB first
1: MSB first
This is the data shift select bit and is used to select how the data is transferred, either MSB or LSB first. Setting the bit high will select MSB first and low for LSB first.

Bit 2 **CSEN**: SPI $\overline{\text{SCS}}$ pin control
0: Disable
1: Enable
The CSEN bit is used as an enable/disable for the $\overline{\text{SCS}}$ pin. If this bit is low, then the $\overline{\text{SCS}}$ pin will be disabled and placed into a floating condition. If the bit is high the $\overline{\text{SCS}}$ pin will be enabled and used as a select pin.

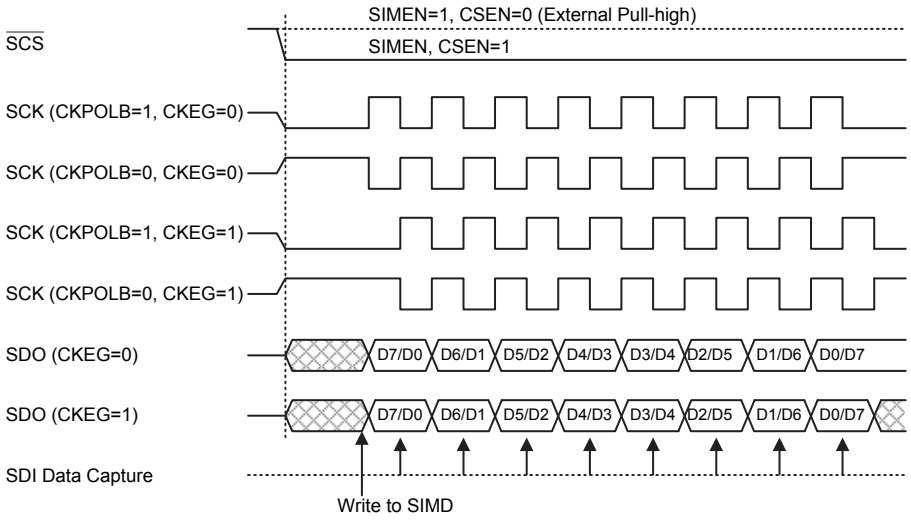
Bit 1 **WCOL**: SPI write collision flag
0: No collision
1: Collision
The WCOL flag is used to detect if a data collision has occurred. If this bit is high it means that data has been attempted to be written to the SIMD register during a data transfer operation. This writing operation will be ignored if data is being transferred. The bit can be cleared by the application program.

Bit 0 **TRF**: SPI Transmit/Receive complete flag
0: SPI data is being transferred
1: SPI data transmission is completed
The TRF bit is the Transmit/Receive Complete flag and is set “1” automatically when an SPI data transmission is completed, but must set to “0” by the application program. It can be used to generate an interrupt.

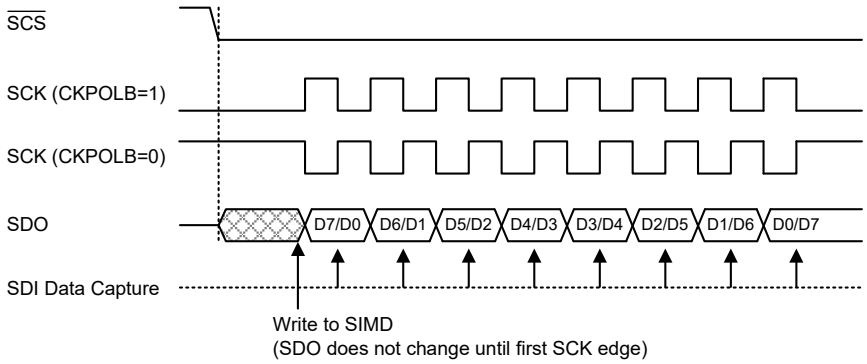
### SPI Communication

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

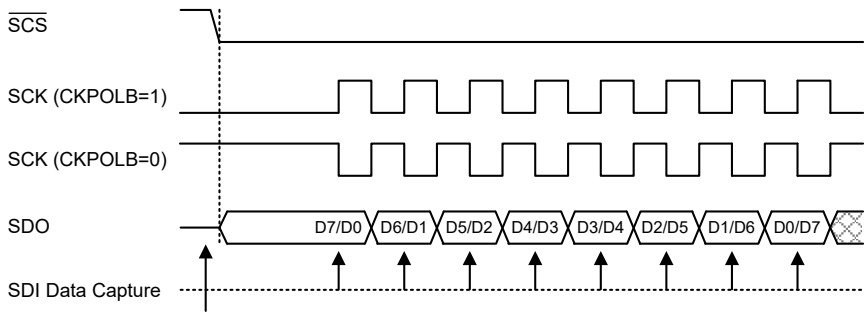
The SPI will continue to function in certain IDLE Modes if the clock source used by the SPI interface is still active.




**SPI Master Mode Timing**



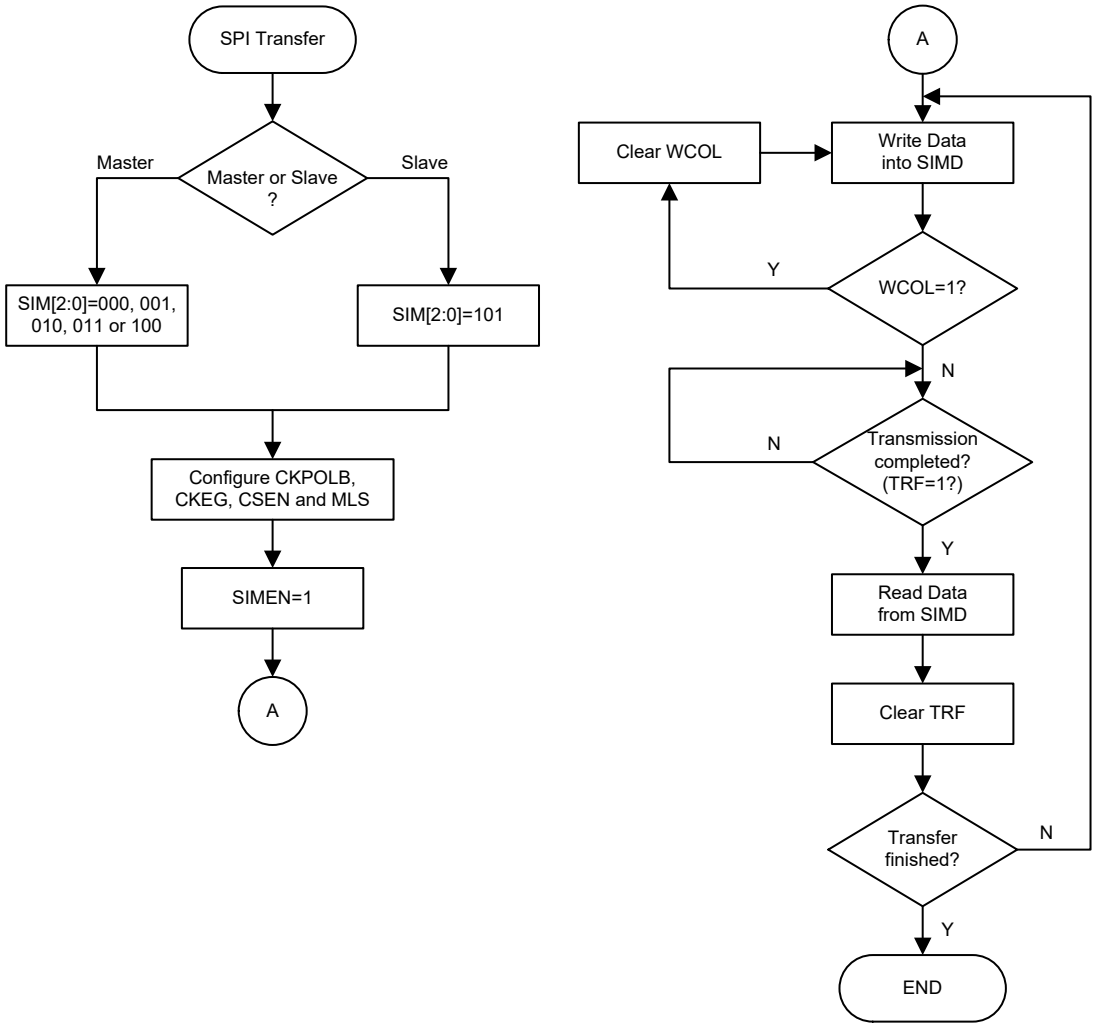

**SPI Slave Mode Timing – CKEG=0**




Write to SIMD
(SDO changes as soon as writing occurs; SDO is floating if $\overline{SCS}$=1)

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

**SPI Slave Mode Timing – CKEG=1**




**SPI Transfer Control 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 control bits.

### SPI Operation Steps

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

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

#### Master Mode

- Step 1
  Select the SPI Master mode and clock source using the SIM2~SIM0 bits in the SIMC0 control register.
- Step 2
  Setup the CSEN bit and setup the MLS bit to choose if the data is MSB or LSB first, this setting must be the same with the Slave devices.
- Step 3
  Setup the SIMEN bit in the SIMC0 control register to enable the SPI interface.
- Step 4
  For write operations: write the data to the SIMD register, which will actually place the data into the TXRX buffer. Then use the SCK and $\overline{SCS}$ 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 SPI serial bus interrupt.
- Step 7
  Read data from the SIMD register.

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

### Slave Mode

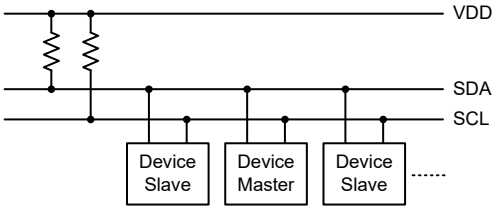
- Step 1
  Select the SPI Slave mode using the SIM2~SIM0 bits in the SIMC0 control register.
- Step 2
  Setup the CSEN bit and setup the MLS bit to choose if the data is MSB or LSB first, this setting must be the same with the Master devices.
- Step 3
  Setup the SIMEN bit in the SIMC0 control register to enable the SPI interface.
- Step 4
  For write operations: write the data to the SIMD register, which will actually place the data into the TXRX buffer. Then wait for the master clock SCK and $\overline{SCS}$ 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 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.

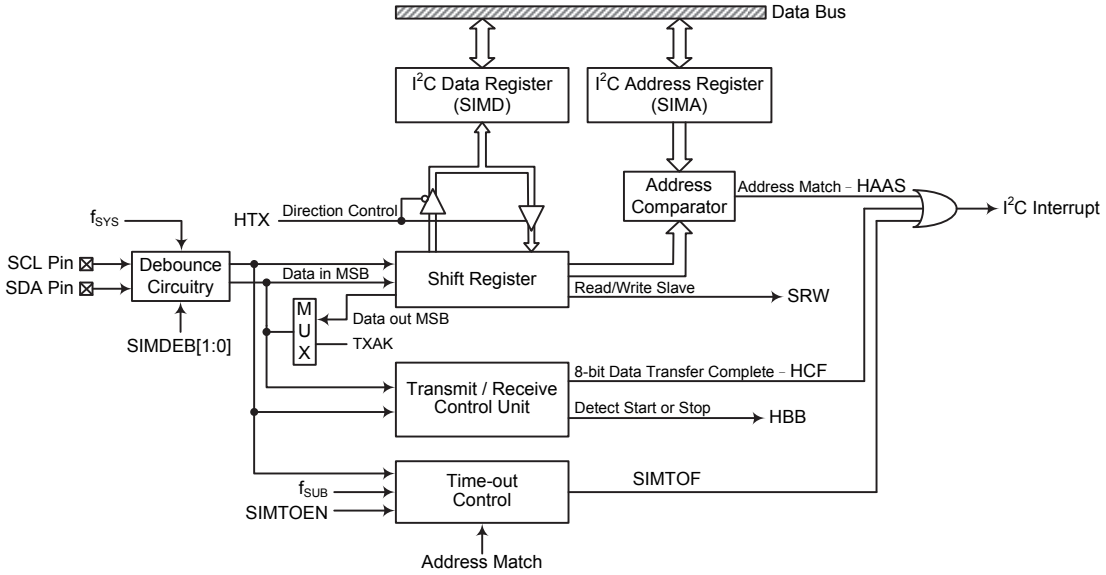



**I²C Master/Slave Bus Connection**

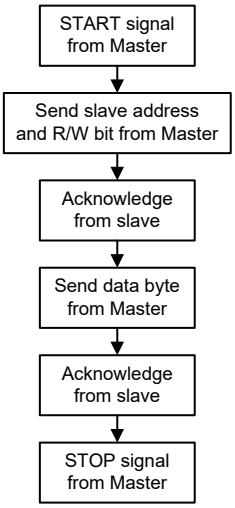
### I²C Interface Operation

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 function 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.

| I²C Debounce Time Selection | I²C Standard Mode (100kHz) | I²C Fast Mode (400kHz) |
|---|---|---|
| No Debounce | $f_{SYS}$ > 2MHz | $f_{SYS}$ > 5MHz |
| 2 system clock debounce | $f_{SYS}$ > 4MHz | $f_{SYS}$ > 10MHz |
| 4 system clock debounce | $f_{SYS}$ > 8MHz | $f_{SYS}$ > 20MHz |

**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.

| Register Name | Bit | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SIMC0 | SIM2 | SIM1 | SIM0 | — | SIMDEB1 | SIMDEB0 | SIMEN | SIMICF |
| SIMC1 | HCF | HAAS | HBB | HTX | TXAK | SRW | IAMWU | RXAK |
| SIMD | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| SIMA | SIMA6 | SIMA5 | SIMA4 | SIMA3 | SIMA2 | SIMA1 | SIMA0 | D0 |
| SIMTOC | SIMTOEN | SIMTOF | SIMTOS5 | SIMTOS4 | SIMTOS3 | SIMTOS2 | SIMTOS1 | SIMTOS0 |

**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**: SIM data register bit 7 ~ bit 0

### I²C Address Register

The SIMA register is also used by the SPI interface but has the name SIMC2. The SIMA register is the location where the 7-bit slave address of the slave device is stored. Bits 7~1 of the SIMA register define the device slave address. Bit 0 is not defined. When a master device, which is connected to the I²C bus, sends out an address, which matches the slave address in the SIMA register, the slave device will be selected. Note that the SIMA register is the same register address as SIMC2 which is used by the SPI interface.

- **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 by application program

### 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 | — | SIMDEB1 | SIMDEB0 | SIMEN | SIMICF |
| R/W | R/W | R/W | R/W | — | R/W | R/W | R/W | R/W |
| POR | 1 | 1 | 1 | — | 0 | 0 | 0 | 0 |

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

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

Bit 4 Unimplemented, read as "0"

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

These bits are used to select the $I^2C$ debounce time when the SIM is configured as the $I^2C$ interface function by setting the SIM2~SIM0 bits to "110".

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

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

Bit 0 **SIMICF**: SIM SPI Incomplete Flag

This bit is only available when the SIM is configured to operate in an SPI slave mode. Refer to the SPI register section.

• **SIMC1 Register**

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

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

The HCF flag is the data transfer flag. This flag will be zero when data is being transferred. Upon completion of an 8-bit data transfer the flag will go high and an interrupt will be generated.

Bit 6 **HAAS**: $I^2C$ Bus address match flag
0: Not address match
1: Address match

The HAAS flag is the address match flag. This flag is used to determine if the slave device address is the same as the master transmit address. If the addresses match then this bit will be high, if there is no match then the flag will be low.

Bit 5 **HBB**: $I^2C$ Bus busy flag
0: $I^2C$ Bus is not busy
1: $I^2C$ Bus is busy

The HBB flag is the $I^2C$ busy flag. This flag will be "1" when the $I^2C$ bus is busy which will occur when a START signal is detected. The flag will be set to "0" when the bus is free which will occur when a STOP signal is detected.

Bit 4 **HTX**: $I^2C$ slave device is transmitter or receiver selection
0: Slave device is the receiver
1: Slave device is the transmitter

Bit 3 **TXAK**: $I^2C$ 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**: $I^2C$ Slave Read/Write flag
0: Slave device should be in receive mode
1: Slave device should be in transmit mode

The SRW flag is the $I^2C$ Slave Read/Write flag. This flag determines whether the master device wishes to transmit or receive data from the $I^2C$ bus. When the transmitted address and slave address is match, that is when the HAAS flag is set high, the slave device will check the SRW flag to determine whether it should be in transmit mode or receive mode. If the SRW flag is high, the master is requesting to read data from the bus, so the slave device should be in transmit mode. When the SRW flag is zero, the master will write data to the bus, therefore the slave device should be in receive mode to read this data.

Bit 1 **IAMWU**: $I^2C$ Address Match Wake-up control
0: Disable
1: Enable

This bit should be set to 1 to enable the $I^2C$ address match wake up from the SLEEP or IDLE Mode. If the IAMWU bit has been set before entering either the SLEEP or IDLE mode to enable the $I^2C$ address match wake up, then this bit must be cleared by the application program after wake-up to ensure correction device operation.

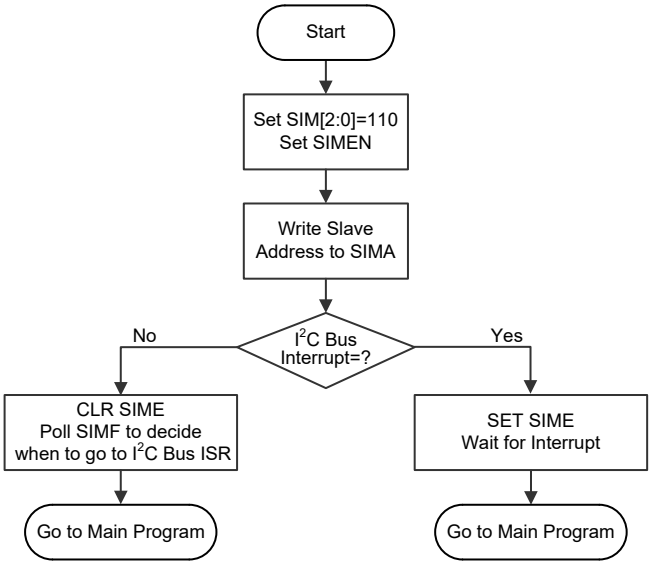
Bit 0 **RXAK**: $I^2C$ Bus Receive acknowledge flag
0: Slave receive acknowledge flag
1: Slave does not receive acknowledge flag

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 an I²C 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 SIM2~SIM0 and SIMEN bits in the SIMC0 register to "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 SIME interrupt enable bit of the interrupt control register to enable the SIM 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 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 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.

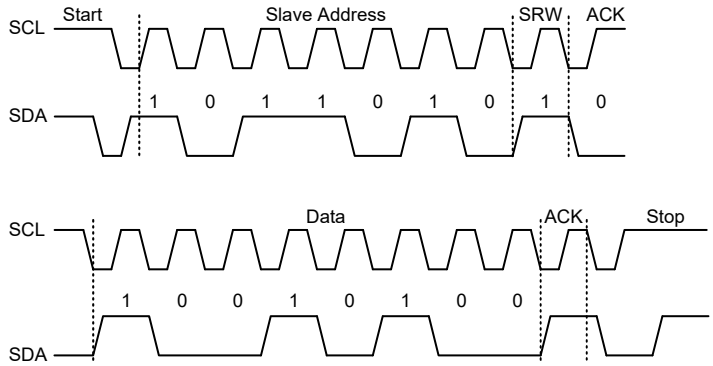
### I²C Bus Slave Address Acknowledge Signal

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.

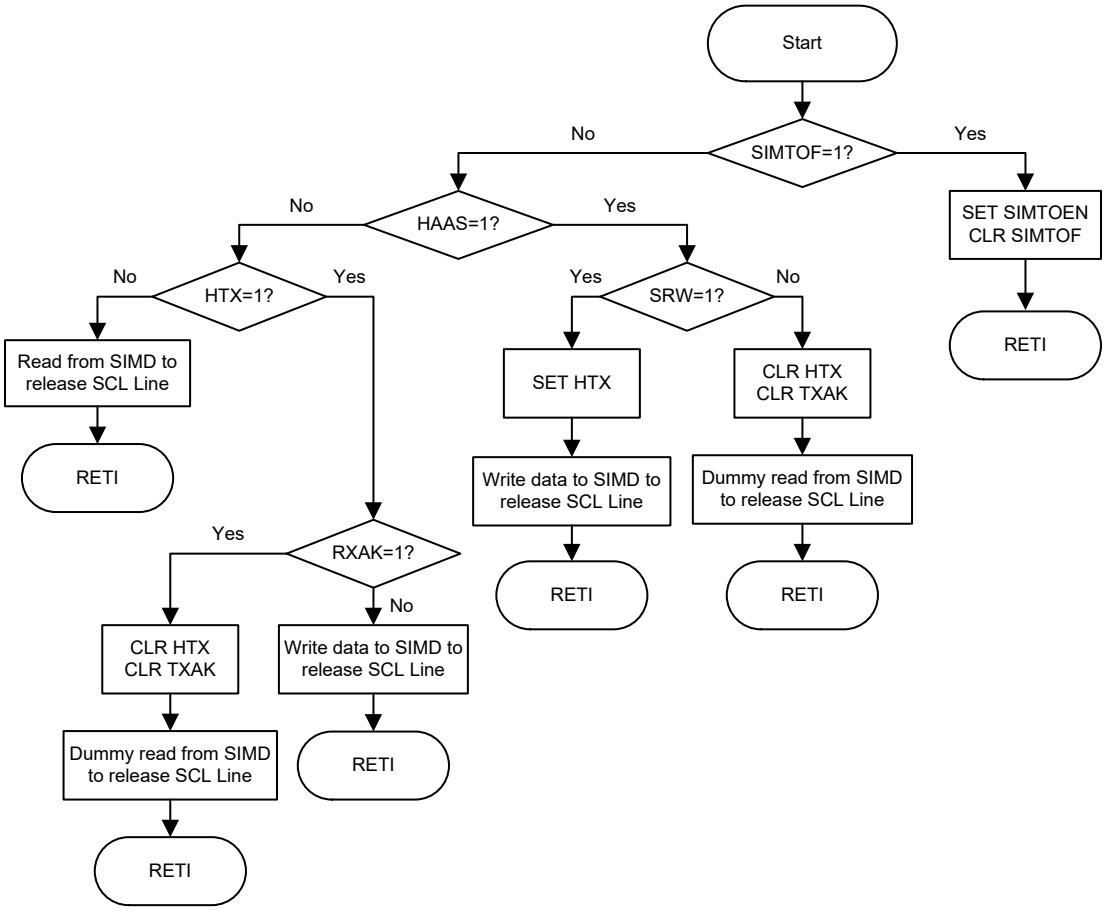



S=Start (1 bit)
SA=Slave Address (7 bits)
SR=SRW bit (1 bit)
M=Slave device send acknowledge bit (1 bit)
D=Data (8 bits)
A=ACK (RXAK bit for transmitter, TXAK bit for receiver, 1 bit)
P=Stop (1 bit)

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

**I²C Communication Timing Diagram**

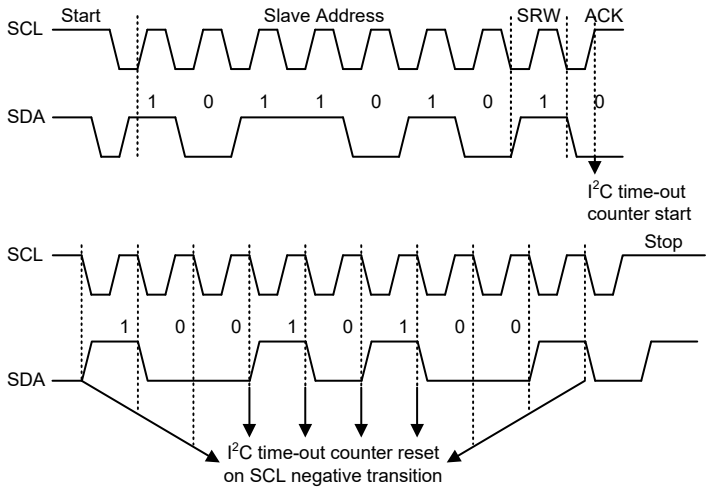
Note: When a slave address is matched, the device must be placed in either the transmit mode and then write data to the SIMD register, or in the receive mode where it must implement a dummy read from the SIMD register to release the SCL line.




**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.




**I²C Time-out**

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 I²C 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 I²C 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 SIMTOS bit field in the SIMTOC register. The time-out time is given by the formula: $((1\sim64)\times32)/f_{SUB}$. This gives a time-out period which ranges from about 1ms to 64ms.

- **SIMTOC Register**

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

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

Bit 6 **SIMTOF**: SIM I²C Time-out flag
0: No time-out occurred
1: Time-out occurred

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

# 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. These devices contain an external interrupt and several internal interrupt functions. The external interrupt is generated by the action of the external INT pin, while the internal interrupts are generated by various internal functions such as the TMs, Time Base, EEPROM and the A/D converter, etc.

## 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 number of registers depends upon the device chosen but fall into three categories. The first is the INTC0~INTC2 registers which setup the primary interrupts. The second is the MFI0~MFI1 register which setups the Multi-function interrupt. 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 interrupts as well as interrupt flags to indicate the presence of an interrupt request. The naming convention of these follows a specific pattern. First is listed an abbreviated interrupt type, then the (optional) number of that interrupt followed by either an "E" for enable/disable bit or "F" for request flag.

| Function | Enable Bit | Request Flag | Notes |
|---|---|---|---|
| Global | EMI | — | — |
| INT Pin | INTE | INTF | — |
| Touch Key Module | TKME | TKMF | — |
| SIM | SIME | SIMF | — |
| Time Base | TBE | TBF | — |
| EEPROM write operation | DEE | DEF | — |
| A/D Converter | ADE | ADF | — |
| Multi-function | MFnE | MFnF | n=0 for BS84B08C<br>n=0~1 for BS84C12C |
| CTM | CTMPE | CTMPF | For BS84C12C only |
| | CTMAE | CTMAF | |
| PTM | PTMPE | PTMPF | — |
| | PTMAE | PTMAF | |

**Interrupt Register Bit Naming Conventions**

| Register Name | Bit | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| INTEG | — | — | — | — | — | — | INTS1 | INTS0 |
| INTC0 | — | MF0F | TKMF | INTF | MF0E | TKME | INTE | EMI |
| INTC1 | ADF | DEF | TBF | SIMF | ADE | DEE | TBE | SIME |
| INTC2 (BS84C12C) | — | — | — | MF1F | — | — | — | MF1E |
| MFI0 | — | — | PTMAF | PTMPF | — | — | PTMAE | PTMPE |
| MFI1 (BS84C12C) | — | — | CTMAF | CTMPF | — | — | CTMAE | CTMPE |

**Interrupt Register List**

• **INTEG Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | — | — | — | — | — | — | INTS1 | INTS0 |
| R/W | — | — | — | — | — | — | R/W | R/W |
| POR | — | — | — | — | — | — | 0 | 0 |

Bit 7~4 Unimplemented, read as "0"

Bit 1~0 **INTS1~INTS0**: Interrupt edge control for INT pin
00: Disable
01: Rising edge
10: Falling edge
11: Rising and falling edges

• **INTC0 Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | — | MF0F | TKMF | INTF | MF0E | TKME | INTE | EMI |
| R/W | — | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| POR | — | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Bit 7 Unimplemented, read as "0"

Bit 6 **MF0F**: Multi-function interrupt 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 **INTF**: INT interrupt request flag
0: No request
1: Interrupt request

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

Bit 2 **TKME**: Touch key module interrupt control
0: Disable
1: Enable

Bit 1 **INTE**: INT interrupt control
0: Disable
1: Enable

Bit 0 **EMI**: Global interrupt control
0: Disable
1: Enable

• **INTC1 Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | ADF | DEF | TBF | SIMF | ADE | DEE | TBE | SIME |
| 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 **ADF**: A/D Converter interrupt request flag
0: No request
1: Interrupt request

Bit 6 **DEF**: Data EEPROM Interrupt request flag
0: No request
1: Interrupt request

Bit 5 **TBF**: Time Base interrupt request flag
0: No request
1: Interrupt request

Bit 4 **SIMF**: SIM interrupt request flag
0: No request
1: Interrupt request

Bit 3 **ADE**: A/D Converter interrupt control
0: Disable
1: Enable

Bit 2 **DEE**: Data EEPROM Interrupt control
0: Disable
1: Enable

Bit 1 **TBE**: Time Base interrupt control
0: Disable
1: Enable

Bit 0 **SIME**: SIM interrupt control
0: Disable
1: Enable

• **INTC2 Register – BS84C12C**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | — | — | — | MF1F | — | — | — | MF1E |
| R/W | — | — | — | R/W | — | — | — | R/W |
| POR | — | — | — | 0 | — | — | — | 0 |

Bit 7~5 Unimplemented, read as “0”

Bit 4 **MF1F**: Multi-function interrupt 1 request flag
0: No request
1: Interrupt request

Bit 3~1 Unimplemented, read as “0”

Bit 0 **MF1E**: Multi-function interrupt 1 control
0: Disable
1: Enable

• **MFI0 Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | — | — | PTMAF | PTMPF | — | — | PTMAE | PTMPE |
| R/W | — | — | R/W | R/W | — | — | R/W | R/W |
| POR | — | — | 0 | 0 | — | — | 0 | 0 |

Bit 7~6 Unimplemented, read as "0"

Bit 5 **PTMAF**: PTM Comparator A match interrupt request flag
0: No request
1: Interrupt request

Bit 4 **PTMPF**: PTM Comparator P match interrupt request flag
0: No request
1: Interrupt request

Bit 3~2 Unimplemented, read as "0"

Bit 1 **PTMAE**: PTM Comparator A match interrupt control
0: Disable
1: Enable

Bit 0 **PTMPE**: PTM Comparator P match interrupt control
0: Disable
1: Enable

• **MFI1 Register – BS84C12C**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | — | — | CTMAF | CTMPF | — | — | CTMAE | CTMPE |
| R/W | — | — | R/W | R/W | — | — | R/W | R/W |
| POR | — | — | 0 | 0 | — | — | 0 | 0 |

Bit 7~6 Unimplemented, read as "0"

Bit 5 **CTMAF**: CTM Comparator A match interrupt request flag
0: No request
1: Interrupt request

Bit 4 **CTMPF**: CTM Comparator P match interrupt request flag
0: No request
1: Interrupt request

Bit 3~2 Unimplemented, read as "0"

Bit 1 **CTMAE**: CTM Comparator A match interrupt control
0: Disable
1: Enable

Bit 0 **CTMPE**: CTM Comparator P match interrupt control
0: Disable
1: Enable

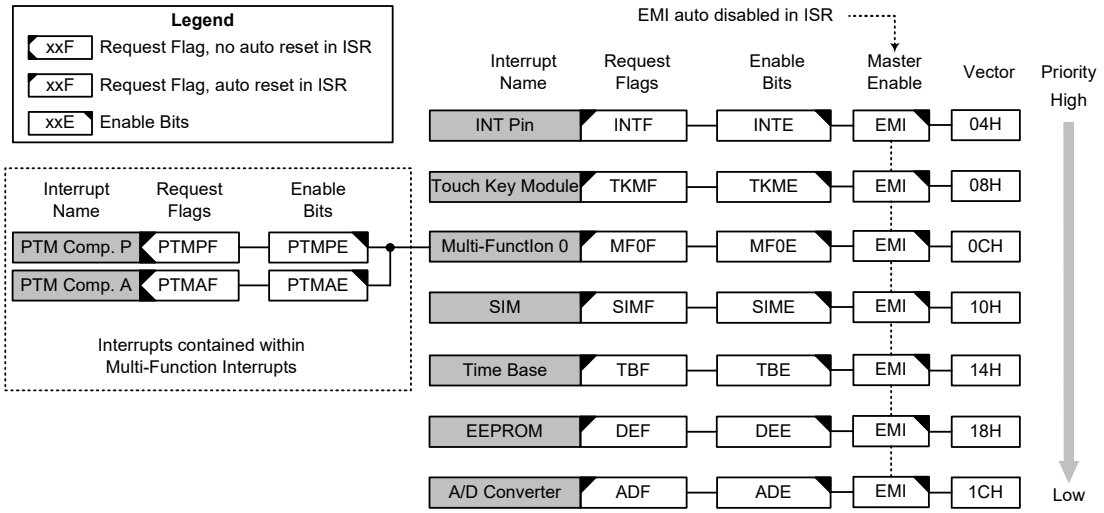
## Interrupt Operation

When the conditions for an interrupt event occur, such as a Touch Key Counter overflow, 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.

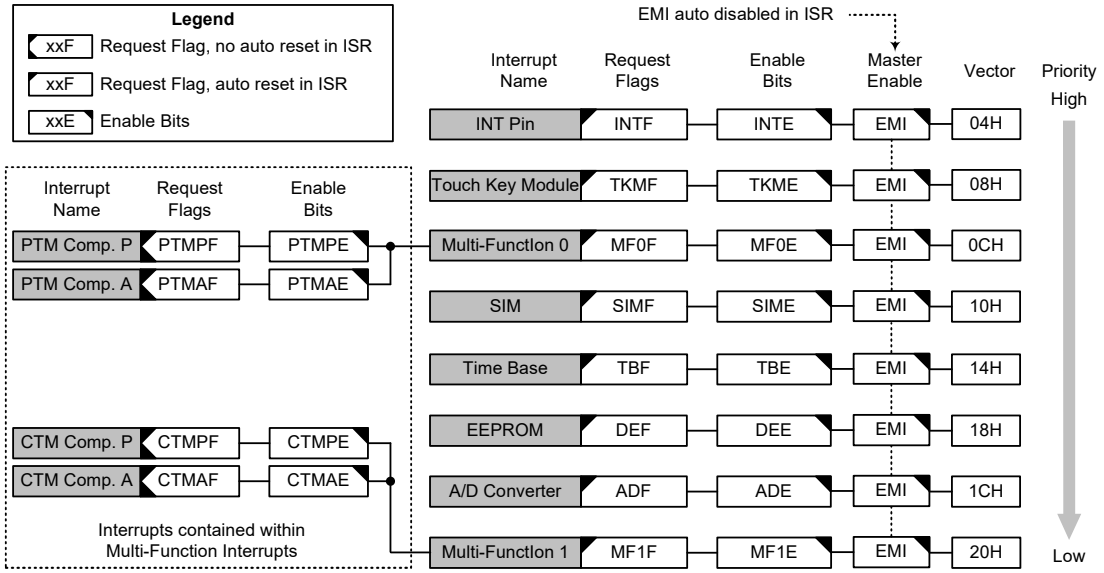
When an interrupt is generated, the Program Counter, which stores the address of the next instruction to be executed, will be transferred onto the stack. The Program Counter will then be loaded with a new address which will be the value of the corresponding interrupt vector. The microcontroller will 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.

The various interrupt enable bits, together with their associated request flags, are shown in the accompanying diagrams with their order of priority. Some interrupt sources have their own individual vector while others share the same multi-function interrupt vector. Once an interrupt subroutine is serviced, all other interrupts will be blocked, as the global interrupt enable bit, EMI bit will be cleared automatically. This will prevent any further interrupt nesting from occurring. However, if other interrupt requests occur during this interval, although the interrupt will not be immediately serviced, the request flag will still be recorded.

If an interrupt requires immediate servicing while the program is already in another interrupt service routine, the EMI bit should be set after entering the routine to allow interrupt nesting. If the stack is full, the interrupt request will not be acknowledged, even if the related interrupt is enabled, until the Stack Pointer is decremented. If immediate service is desired, the stack must be prevented from becoming full. In case of simultaneous requests, the accompanying diagram shows the priority that is applied. All of the interrupt request flags when set will wake-up the devices if it is in SLEEP or IDLE Mode, however to prevent a wake-up from occurring the corresponding flag should be set before the devices are in SLEEP or IDLE Mode.




**Interrupt Structure – BS84B08C**




**Interrupt Structure – BS84C12C**

### External Interrupt

The external interrupt is controlled by signal transitions on the INT pin. An external interrupt request will take place when the external interrupt request flag, INTF, is set, which will occur when a transition, whose type is chosen by the edge select bits, appears on the external interrupt pin. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and respective external interrupt enable bit, INTE, must first be set. Additionally the correct interrupt edge type must be selected using the INTEG register to enable the external interrupt function and to choose the trigger edge type. As the external interrupt pin is pin-shared with I/O pins, it can only be configured as an external interrupt pin if the external interrupt enable bit in the corresponding interrupt register has been set. The pin must also be setup as an input by setting the corresponding bit in the port control register. When the interrupt is enabled, the stack is not full and the correct transition type appears on the external interrupt pin, a subroutine call to the external interrupt vector, will take place. When the interrupt is serviced, the external interrupt request flags, INTF, 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 Module Interrupt

For a Touch Key interrupt to occur, the global interrupt enable bit, EMI, and the Touch Key interrupt enable bit, TKME, must be first set. An actual 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. When the interrupt is enabled, the stack is not full and the Touch Key time slot counter overflow occurs, a subroutine call to the relevant interrupt vector, will take place. When the interrupt is serviced, the Touch Key interrupt request flag will be automatically reset and the EMI bit will also be automatically cleared to disable other interrupts.

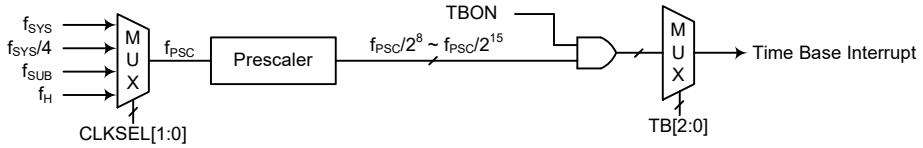
### Serial Interface Module Interrupt

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

## Time Base Interrupt

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

The purpose of the Time Base Interrupt is to provide an interrupt signal at fixed time periods. Its clock source, $f_{PSC}$, originates from the internal clock source $f_{SYS}$, $f_{SYS}/4$, $f_{SUB}$ or $f_H$ and then passes through a divider, the division ratio of which is selected by programming the appropriate bits in the TBC register to obtain longer interrupt periods whose value ranges. The clock source which in turn controls the Time Base interrupt period is selected using the CLKSEL[1:0] bits in the PSCR register.




**Time Base Interrupt**

- **PSCR Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | — | — | — | — | — | — | CLKSEL1 | CLKSEL0 |
| R/W | — | — | — | — | — | — | R/W | R/W |
| POR | — | — | — | — | — | — | 0 | 0 |

Bit 7~2 Unimplemented, read as "0"

Bit 1~0 **CLKSEL1~CLKSEL0**: Prescaler clock source $f_{PSC}$ selection
00: $f_{SYS}$
01: $f_{SYS}/4$
10: $f_{SUB}$
11: $f_H$

- **TBC Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | — | — | — | — | TBON | TB2 | TB1 | TB0 |
| R/W | — | — | — | — | R/W | R/W | R/W | R/W |
| POR | — | — | — | — | 0 | 0 | 0 | 0 |

Bit 7~4 Unimplemented, read as "0"

Bit 3 **TBON**: Time Base Enable Control
0: Disable
1: Enable

Bit 2~0 **TB2~TB0**: Time Base time-out period selection
000: $2^8/f_{PSC}$
001: $2^9/f_{PSC}$
010: $2^{10}/f_{PSC}$
011: $2^{11}/f_{PSC}$
100: $2^{12}/f_{PSC}$
101: $2^{13}/f_{PSC}$
110: $2^{14}/f_{PSC}$
111: $2^{15}/f_{PSC}$

### EEPROM Interrupt

The EEPROM Write Interrupt is an individual interrupt source with its own interrupt vector. An EEPROM Write Interrupt request will take place when the EEPROM Write Interrupt request flag, DEF, is set, which occurs when an EEPROM Write cycle ends. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and EEPROM Write Interrupt enable bit, DEE, must first be set. When the interrupt is enabled, the stack is not full and an EEPROM Write cycle ends, a subroutine call to the respective interrupt vector will take place. When the EEPROM Write Interrupt is serviced, the DEF flag will be automatically cleared and the EMI bit will also be automatically cleared to disable other interrupts.

### A/D Converter Interrupt

The A/D Converter Interrupt is controlled by the termination of an A/D conversion process. An A/D Converter Interrupt request will take place when the A/D Converter Interrupt request flag, ADF, is set, which occurs when the A/D conversion process finishes. To allow the program to branch to its respective 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 devices there is up to two Multi-function interrupts. Unlike the other independent interrupts, this interrupt has no independent source, but rather are formed from other existing interrupt sources, namely the TM interrupts.

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

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

### TM Interrupts

The Compact and Periodic Type TMs have two interrupts, one comes from the comparator A match situation and the other comes from the comparator P match situation. All of the TM interrupts are contained within the Multi-function Interrupts. For each of the Compact and Periodic Type TMs there are two interrupt request flags ,xTMPF and xTMAF, and two enable control bits, xTMPE and xTMAE. A TM interrupt request will take place when any of the TM request flags are set, a situation which occurs when a TM comparator P or A match situation happens.

To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, respective TM Interrupt enable bit and the relevant Multi-function Interrupt enable bit, MFnE, must first be set. When the interrupt is enabled, the stack is not full and a TM comparator match situation occurs, a subroutine call to the relevant TM Interrupt vector locations will take place. When the TM interrupt is serviced, the EMI bit will be automatically cleared to disable other interrupts, however only the related MFnF flag will be automatically cleared. As the TM interrupt request flags will not be automatically cleared, they have to be cleared by the application program.

## Interrupt Wake-up Function

Each of the interrupt functions has the capability of waking up the microcontroller when in the SLEEP or IDLE Mode. A wake-up is generated when an interrupt request flag changes from low to high and is independent of whether the interrupt is enabled or not. Therefore, even though these devices are in the SLEEP or IDLE Mode and its system oscillator stopped, situations such as external edge transitions on the external interrupt pin or a low power supply voltage 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 devices enter the SLEEP or IDLE Mode. The interrupt enable bits have no effect on the interrupt wake-up function.

## Programming Considerations

By disabling the relevant interrupt enable bits, a requested interrupt can be prevented from being serviced, however, once an interrupt request flag is set, it will remain in this condition in the interrupt register until the corresponding interrupt is serviced or until the request flag is cleared by the application program.

Where a certain interrupt is contained within a Multi-function interrupt, then when the interrupt service routine is executed, as only the Multi-function interrupt request flags, MFnF, will be automatically cleared, the individual request flag for the function needs to be cleared by the application program.

It is recommended that programs do not use the “CALL” instruction within the interrupt service subroutine. Interrupts often occur in an unpredictable manner or need to be serviced immediately. If only one stack is left and the interrupt is not well controlled, the original control sequence will be damaged once a CALL subroutine is executed in the interrupt subroutine.

Every interrupt has the capability of waking up the microcontroller when it is in 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.

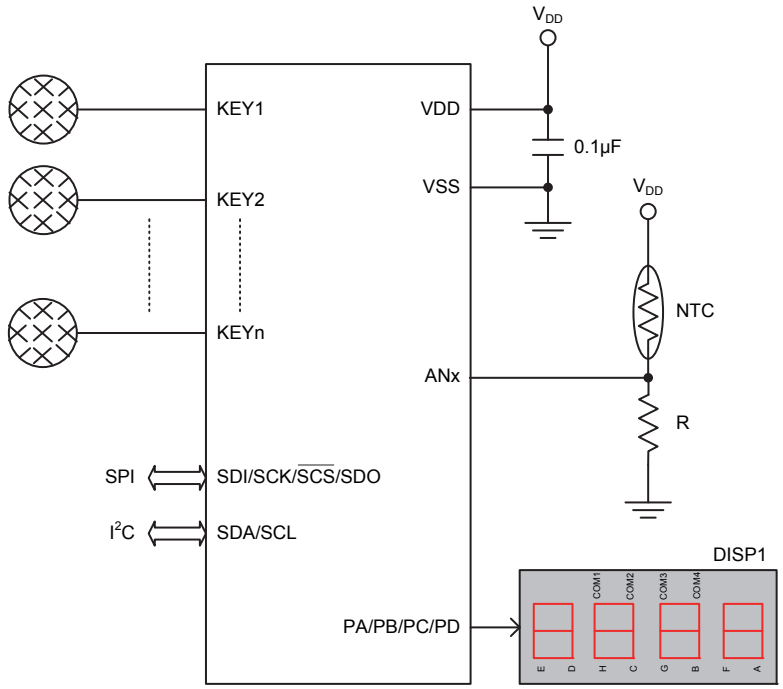
## Configuration Options

Configuration options refer to certain options within the MCU that are programmed into the devices during the programming process. During the development process, these options are selected using the HT-IDE software development tools. As these options are programmed into the devices using the hardware programming tools, once they are selected they cannot be changed later using the application program. All options must be defined for proper system function, the details of which are shown in the table.

| No. | Options |
|---|---|
| **Oscillator Option** | |
| 1 | HIRC frequency selection: 8MHz, 12MHz or 16MHz |

Note: When the HIRC has been configured at a frequency shown in this table, the HIRCS1 and HIRCS0 bits should also be setup 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 0.5μs and branch or call instructions would be implemented within 1μs. Although instructions which require one more cycle to implement are generally limited to the JMP, CALL, RET, RETI and table read instructions, it is important to realize that any other instructions which involve manipulation of the Program Counter Low register or PCL will also take one more cycle to implement. As instructions which change the contents of the PCL will imply a direct jump to that new address, one more cycle will be required. Examples of such instructions would be "CLR PCL" or "MOV PCL, A". For the case of skip instructions, it must be noted that if the result of the comparison involves a skip operation then this will also take one more cycle, if no skip is involved then only one cycle is required.

## Moving and Transferring Data

The transfer of data within the microcontroller program is one of the most frequently used operations. Making use of three kinds of MOV instructions, data can be transferred from registers to the Accumulator and vice-versa as well as being able to move specific immediate data directly into the Accumulator. One of the most important data transfer applications is to receive data from the input ports and transfer data to the output ports.

## Arithmetic Operations

The ability to perform certain arithmetic operations and data manipulation is a necessary feature of most microcontroller applications. Within the Holtek microcontroller instruction set are a range of add and subtract instruction mnemonics to enable the necessary arithmetic to be carried out. Care must be taken to ensure correct handling of carry and borrow data when results exceed 255 for addition and less than 0 for subtraction. The increment and decrement instructions INC, INCA, DEC and DECA provide a simple means of increasing or decreasing by a value of one of the values in the destination specified.

## Logical and Rotate Operation

The standard logical operations such as AND, OR, XOR and CPL all have their own instruction within the Holtek microcontroller instruction set. As with the case of most instructions involving data manipulation, data must pass through the Accumulator which may involve additional programming steps. In all logical data operations, the zero flag may be set if the result of the operation is zero. Another form of logical data manipulation comes from the rotate instructions such as RR, RL, RRC and RLC which provide a simple means of rotating one bit right or left. Different rotate instructions exist depending on program requirements. Rotate instructions are useful for serial port programming applications where data can be rotated from an internal register into the Carry bit from where it can be examined and the necessary serial bit set high or low. Another application which rotate data operations are used is to implement multiplication and division calculations.

## Branches and Control Transfer

Program branching takes the form of either jumps to specified locations using the JMP instruction or to a subroutine using the CALL instruction. They differ in the sense that in the case of a subroutine call, the program must return to the instruction immediately when the subroutine has been carried out. This is done by placing a return instruction "RET" in the subroutine which will cause the program to jump back to the address right after the CALL instruction. In the case of a JMP instruction, the program simply jumps to the desired location. There is no requirement to jump back to the original jumping off point as in the case of the CALL instruction. One special and extremely useful set of branch instructions are the conditional branches. Here a decision is first made regarding the condition of a certain data memory or individual bits. Depending upon the conditions, the program will continue with the next instruction or skip over it and jump to the following instruction. These instructions are the key to decision making and branching within the program perhaps determined by the condition of certain input switches or by the condition of internal data bits.

## Bit Operations

The ability to provide single bit operations on Data Memory is an extremely flexible feature of all Holtek microcontrollers. This feature is especially useful for output port bit programming where individual bits or port pins can be directly set high or low using either the "SET [m].i" or "CLR [m].i" instructions respectively. The feature removes the need for programmers to first read the 8-bit output port, manipulate the input data to ensure that other bits are not changed and then output the port with the correct new data. This read-modify-write process is taken care of automatically when these bit operation instructions are used.

## Table Read Operations

Data storage is normally implemented by using registers. However, when working with large amounts of fixed data, the volume involved often makes it inconvenient to store the fixed data in the Data Memory. To overcome this problem, Holtek microcontrollers allow an area of Program Memory to be set as a table where data can be directly stored. A set of easy to use instructions provides the means by which this fixed data can be referenced and retrieved from the Program Memory.

## Other Operations

In addition to the above functional instructions, a range of other instructions also exist such as the "HALT" instruction for Power-down operations and instructions to control the operation of the Watchdog Timer for reliable program operations under extreme electric or electromagnetic environments. For their relevant operations, refer to the functional related sections.

## Instruction Set Summary

The following table depicts a summary of the instruction set categorised according to function and can be consulted as a basic instruction reference using the following listed conventions.

### Table Conventions

x: Bits immediate data
m: Data Memory address
A: Accumulator
i: 0~7 number of bits
addr: Program memory address

| Mnemonic | Description | Cycles | Flag Affected |
|---|---|---|---|
| **Arithmetic** | | | |
| ADD A,[m] | Add Data Memory to ACC | 1 | Z, C, AC, OV |
| ADDM A,[m] | Add ACC to Data Memory | 1 Note | Z, C, AC, OV |
| ADD A,x | Add immediate data to ACC | 1 | Z, C, AC, OV |
| ADC A,[m] | Add Data Memory to ACC with Carry | 1 | Z, C, AC, OV |
| ADCM A,[m] | Add ACC to Data memory with Carry | 1 Note | Z, C, AC, OV |
| SUB A,x | Subtract immediate data from the ACC | 1 | Z, C, AC, OV |
| SUB A,[m] | Subtract Data Memory from ACC | 1 | Z, C, AC, OV |
| SUBM A,[m] | Subtract Data Memory from ACC with result in Data Memory | 1 Note | Z, C, AC, OV |
| SBC A,[m] | Subtract Data Memory from ACC with Carry | 1 | Z, C, AC, OV |
| SBCM A,[m] | Subtract Data Memory from ACC with Carry, result in Data Memory | 1 Note | Z, C, AC, OV |
| DAA [m] | Decimal adjust ACC for Addition with result in Data Memory | 1 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 | 1 Note | Z |
| ORM A,[m] | Logical OR ACC to Data Memory | 1 Note | Z |
| XORM A,[m] | Logical XOR ACC to Data Memory | 1 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 | 1 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 | 1 Note | Z |
| DECA [m] | Decrement Data Memory with result in ACC | 1 | Z |
| DEC [m] | Decrement Data Memory | 1 Note | Z |
| **Rotate** | | | |
| RRA [m] | Rotate Data Memory right with result in ACC | 1 | None |
| RR [m] | Rotate Data Memory right | 1 Note | None |
| RRCA [m] | Rotate Data Memory right through Carry with result in ACC | 1 | C |
| RRC [m] | Rotate Data Memory right through Carry | 1 Note | C |
| RLA [m] | Rotate Data Memory left with result in ACC | 1 | None |
| RL [m] | Rotate Data Memory left | 1 Note | None |
| RLCA [m] | Rotate Data Memory left through Carry with result in ACC | 1 | C |
| RLC [m] | Rotate Data Memory left through Carry | 1 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].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 or current page) to TBLH and Data Memory | 2[Note] | None |
| TABRDL [m] | 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.

## 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 ← ACC + [m] + C

Affected flag(s): OV, Z, AC, C

**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] ← ACC + [m] + C

Affected flag(s): OV, Z, AC, C

**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 ← ACC + [m]

Affected flag(s): OV, Z, AC, C

**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 ← ACC + x

Affected flag(s): OV, Z, AC, C

**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] ← ACC + [m]

Affected flag(s): OV, Z, AC, C

**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 ← ACC ″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 ← ACC ″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] ← ACC ″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] ← [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 ← [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 ← 0<br>PDF ← 1 |
| Affected flag(s) | TO, PDF |

| | |
|---|---|
| **INC [m]** | Increment Data Memory |
| Description | Data in the specified Data Memory is incremented by 1. |
| Operation | [m] ← [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 ← [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 ← 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 ← [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 ← 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] ← 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<br>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<br>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)<br>[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: $ACC \leftarrow ACC - [m] - \overline{C}$

Affected flag(s): OV, Z, AC, C

**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: $[m] \leftarrow ACC - [m] - \overline{C}$

Affected flag(s): OV, Z, AC, C

**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] ← 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 ← 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] ← [m] + 1<br>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 ← [m] + 1<br>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 ≠ 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 ← ACC − [m] |
| Affected flag(s) | OV, Z, AC, C |

| | |
|---|---|
| **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] ← ACC − [m] |
| Affected flag(s) | OV, Z, AC, C |

| | |
|---|---|
| **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 ← ACC − x |
| Affected flag(s) | OV, Z, AC, C |

| | |
|---|---|
| **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~[m].0 ↔ [m].7~[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~ACC.0 ← [m].7~[m].4
ACC.7~ACC.4 ← [m].3~[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 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 ← [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 or current page) to TBLH and Data Memory

Description: The low byte of the program code addressed by the table pointer (TBHP and TBLP or only TBLP if no TBHP) is moved to the specified Data Memory and the high byte moved to TBLH.

Operation: [m] ← program code (low byte)
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

**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 |

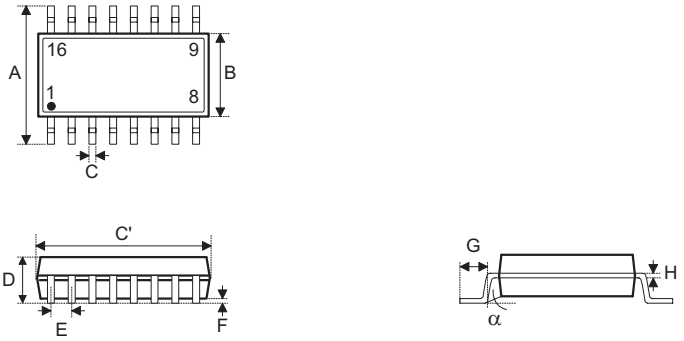
## Package Information

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

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

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

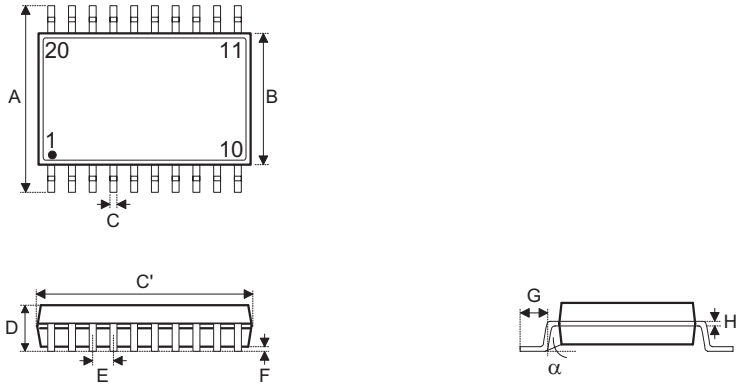
## 16-pin NSOP (150mil) Outline Dimensions




| Symbol | Dimensions in inch | | |
|---|---|---|---|
| | Min. | Nom. | Max. |
| A | | 0.236 BSC | |
| B | | 0.154 BSC | |
| C | 0.012 | — | 0.020 |
| C' | | 0.390 BSC | |
| D | — | — | 0.069 |
| E | | 0.050 BSC | |
| F | 0.004 | — | 0.010 |
| G | 0.016 | — | 0.050 |
| H | 0.004 | — | 0.010 |
| α | 0° | — | 8° |

| Symbol | Dimensions in mm | | |
|---|---|---|---|
| | Min. | Nom. | Max. |
| A | | 6.00 BSC | |
| B | | 3.90 BSC | |
| C | 0.31 | — | 0.51 |
| C' | | 9.90 BSC | |
| D | — | — | 1.75 |
| E | | 1.27 BSC | |
| F | 0.10 | — | 0.25 |
| G | 0.40 | — | 1.27 |
| H | 0.10 | — | 0.25 |
| α | 0° | — | 8° |

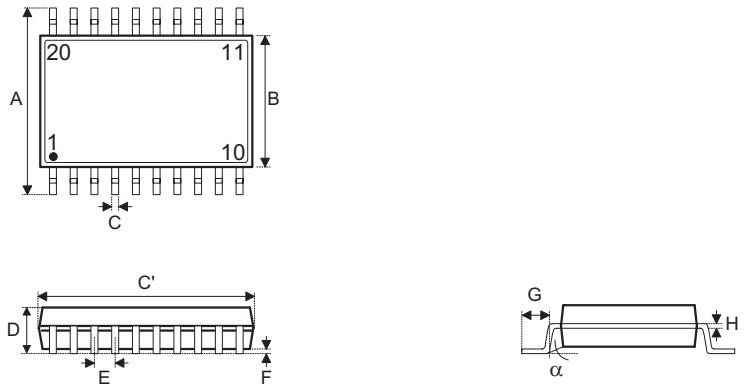
## 20-pin SOP (300mil) Outline Dimensions




| Symbol | Dimensions in inch | | |
|---|---|---|---|
| | Min. | Nom. | Max. |
| A | | 0.406 BSC | |
| B | | 0.295 BSC | |
| C | 0.012 | — | 0.020 |
| C’ | | 0.504 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’ | | 12.80 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° |

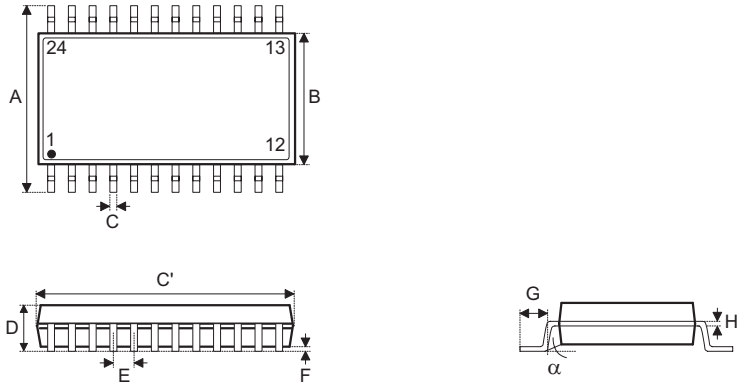
## 20-pin SSOP (150mil) Outline Dimensions




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

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

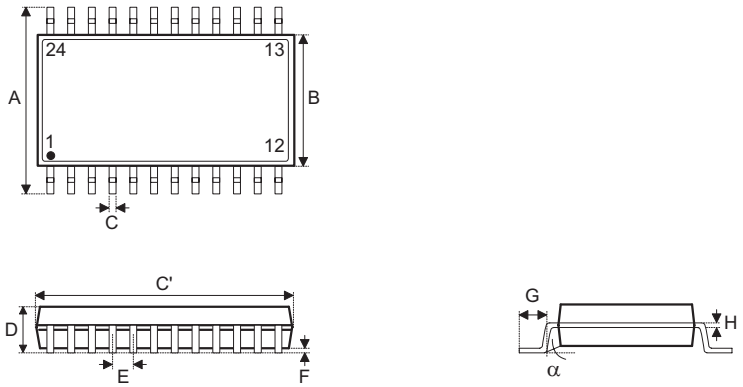
## 24-pin SOP (300mil) Outline Dimensions




| Symbol | Dimensions in inch | | |
|---|---|---|---|
| | Min. | Nom. | Max. |
| A | | 0.406 BSC | |
| B | | 0.295 BSC | |
| C | 0.012 | — | 0.020 |
| C’ | | 0.606 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’ | | 15.40 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° |

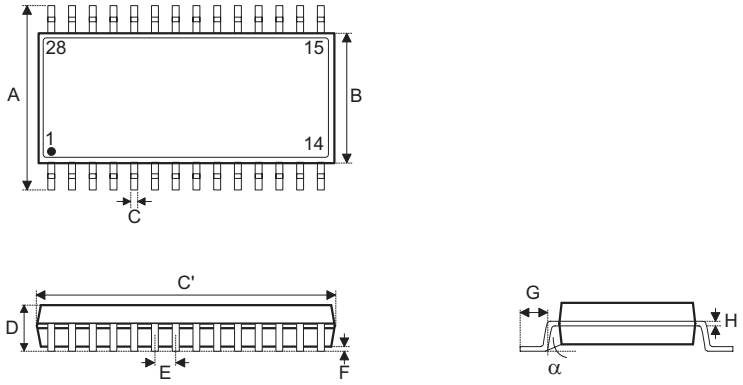
## 24-pin SSOP (150mil) Outline Dimensions




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

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

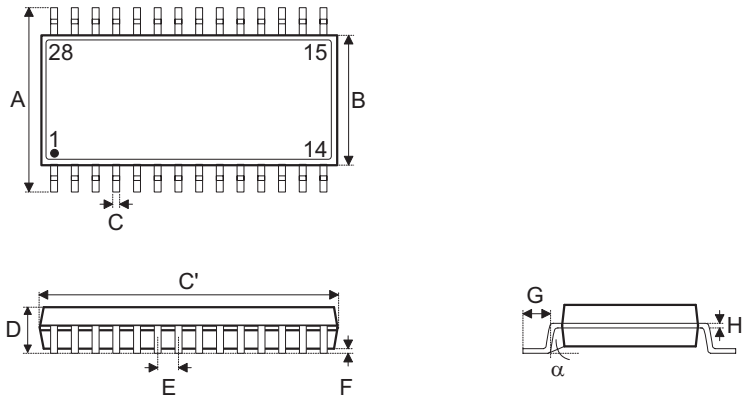
## 28-pin SOP (300mil) Outline Dimensions




| 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° |

Copyright© 2026 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 authorise 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.