HOLTEK

8-Bit Touch Key Flash MCU

# BS83B08-3/BS83B12-3
# BS83B16-3/BS83B16G-3
# BS83C24-3

Revision: 1.30 Date: September 22, 2011

www.holtek.com

# Table of Contents

## Technical Document

- Application Note
  HA0075E MCU Reset and Oscillator Circuits Application Note

## Features

### CPU Features

- Operating Voltage:
  $f_{SYS}$= 8MHz: $V_{LVR}$~5.5V
  $f_{SYS}$= 12MHz: 2.7V~5.5V
  $f_{SYS}$= 16MHz: 4.5V~5.5V
- Fully integrated 8/12/16/24 touch key functions -- require no external components
- Power down and wake-up functions to reduce power consumption
- Fully integrated low and high speed internal oscillators
  Low Speed -- 32kHz
  High speed -- 8MHz, 12MHz, 16MHz
- Multi-mode operation: NORMAL, SLOW, IDLE and SLEEP
- All instructions executed in one or two instruction cycles
- Table read instructions
- 63 powerful instructions
- Up to 8 subroutine nesting levels
- Bit manipulation instruction

### Peripheral Features

- Flash Program Memory: up to 4K×16
- RAM Data Memory: 160×8 ~ 512×8
- EEPROM Memory: up to 128×8
- Watchdog Timer function
- Up to 41 bidirectional I/O lines
- External interrupt line shared with I/O pin
- Single 8-bit Timer/Event Counter
- Single 16-bit Timer/Event Counter for BS83C24-3
- Single Time-Base functions for generation of fixed time interrupt signals
- $I^2C$ and SPI interfaces
- Low voltage reset function
- 8/12/16/24 touch key functions

## General Description

These devices are a series of 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, this device range 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 EEPROM memory for storage of non-volatile data such as serial numbers, calibration data etc. Protective features such as an internal Watchdog Timer and Low Voltage Reset functions coupled with excellent noise immunity and ESD protection ensure that reliable operation is maintained in hostile electrical environments.

All devices include fully integrated low and high speed oscillators which require no external components for their implementation. The ability to operate and switch dynamically between a range of operating modes using different clock sources gives users the ability to optimise microcontroller operation and minimise power consumption. Easy communication with the outside world is provided using the internal $I^2C$ and SPI interfaces, while the inclusion of flexible I/O programming features, Timer/Event Counters and many other features further enhance device functionality and flexibility.

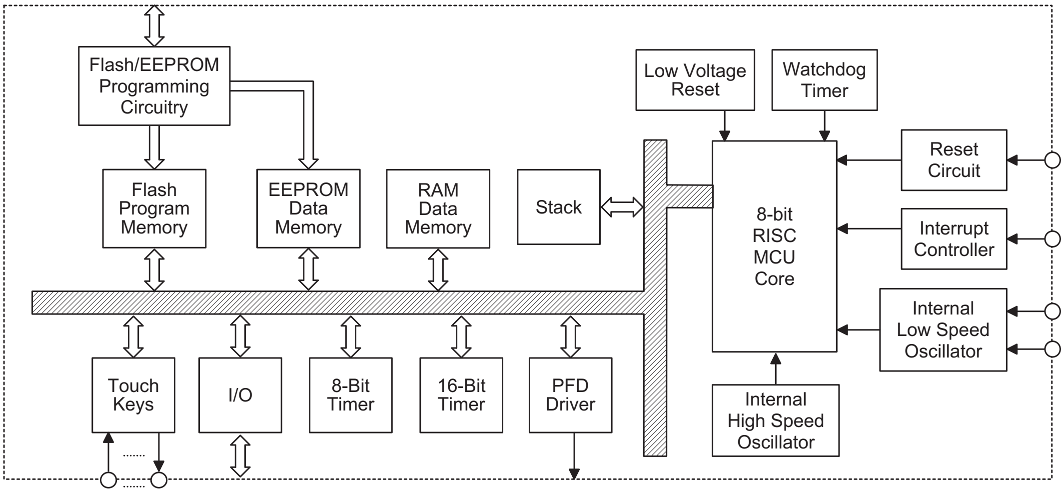
These touch key devices will find excellent use in a huge range of modern Touch Key product applications such as instrumentation, household appliances, electronically controlled tools to name but a few.

## Selection Table

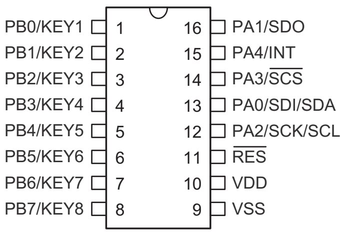
Most features are common to all devices, the main distinguishing feature is the number of I/Os and Touch Keys. The following table summarises the main features of each device.

| Part No. | Internal Clock | VDD | System Clock | Program Memory | Data Memory | Data EEPROM | I/O | 8-bit Timer | 16-bit Timer | Touch Key | SPI/ $I^2C$ | PFD | Stack | Package |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BS83B08-3 | 8MHz<br>12MHz<br>16MHz | $V_{LVR}$~ 5.5V | 8MHz~ 16MHz | 2K×15 | 160×8 | 64×8 | 13 | 1 | – | 8 | 1 | – | 4 | 16NSOP<br>16SSOP |
| BS83B12-3 | 8MHz<br>12MHz<br>16MHz | $V_{LVR}$~ 5.5V | 8MHz~ 16MHz | 2K×15 | 288×8 | 64×8 | 17 | 1 | – | 12 | 1 | – | 4 | 20SOP<br>20SSOP |
| BS83B16-3<br>BS83B16G-3 | 8MHz<br>12MHz<br>16MHz | $V_{LVR}$~ 5.5V | 8MHz~ 16MHz | 2K×15 | 288×8 | 64×8 | 21 | 1 | – | 16 | 1 | – | 4 | 24SOP<br>24SSOP<br>COG |
| BS83C24-3 | 8MHz<br>12MHz<br>16MHz | $V_{LVR}$~ 5.5V | 8MHz~ 16MHz | 4K×16 | 512×8 | 128×8 | 41 | 1 | 1 | 24 | 1 | 1 | 8 | 28SOP<br>28SSOP<br>44QFP |

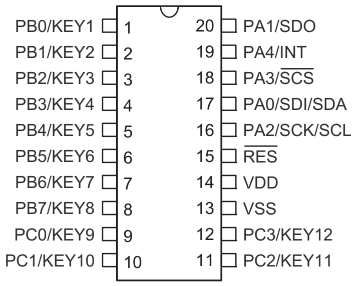
## Block Diagram



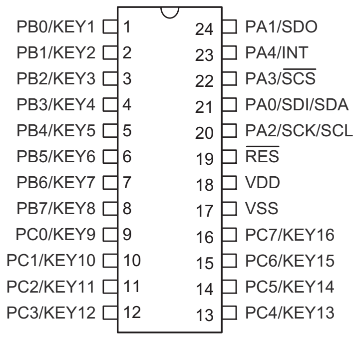

## Pin Assignment



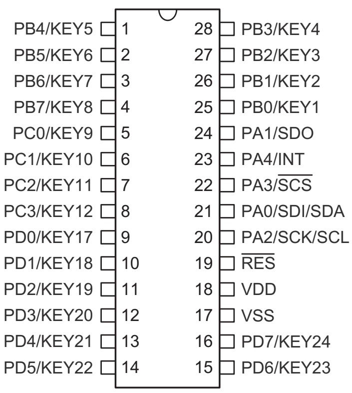

**BS83B08-3**
**16 NSOP-A/SSOP-A**



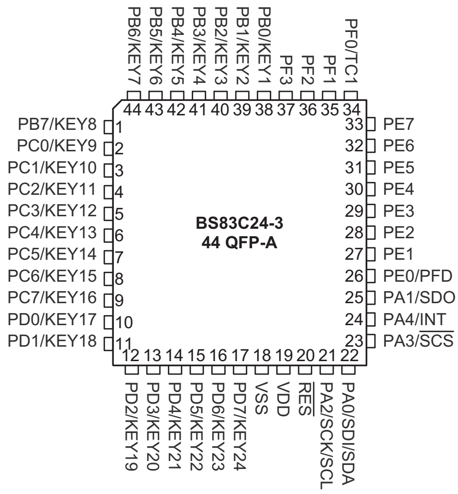

**BS83B12-3**
**20 SOP-A/SSOP-A**




**BS83B16-3**
**24 SOP-A/SSOP-A**




**BS83C24-3**
**28 SOP-A/SSOP-A**



## Pin Description

The function of each pin is listed in the following table, however the details behind how each pin is configured is contained in other sections of the datasheet.

### BS83B08-3/B12-3/B16-3/B16G-3

| Pin Name | Function | OPT | I/T | O/T | Description |
|---|---|---|---|---|---|
| PA0/SDI/SDA | PA0 | PAWU<br>PAPU | ST | CMOS | General purpose I/O. Register enabled pull-up and wake-up. |
| | SDI | SIMC0 | ST | — | SPI data input |
| | SDA | SIMC0 | ST | NMOS | I$^2$C data |
| PA1/SDO | PA1 | PAWU<br>PAPU | ST | CMOS | General purpose I/O. Register enabled pull-up and wake-up. |
| | SDO | SIMC0 | — | CMOS | SPI data output |
| PA2/SCK/SCL | PA2 | PAWU<br>PAPU | ST | CMOS | General purpose I/O. Register enabled pull-up and wake-up. |
| | SCK | SIMC0 | ST | CMOS | SPI serial clock |
| | SCL | SIMC0 | ST | NMOS | I$^2$C clock |
| PA3/$\overline{SCS}$ | PA3 | PAWU<br>PAPU | ST | CMOS | General purpose I/O. Register enabled pull-up and wake-up. |
| | $\overline{SCS}$ | SIMC0 | ST | CMOS | SPI slave select |
| PA4/INT | PA4 | PAWU<br>PAPU | ST | CMOS | General purpose I/O. Register enabled pull-up and wake-up. |
| | INT | INTEG | ST | — | External interrupt |
| PB0/KEY1~<br>PB3/KEY4 | PB0~PB3 | PBPU | ST | CMOS | General purpose I/O. Register enabled pull-up. |
| | KEY1~KEY4 | TKM0C1 | NSI | — | Touch key inputs |
| PB4/KEY5~<br>PB7/KEY8 | PB4~PB7 | PBPU | ST | CMOS | General purpose I/O. Register enabled pull-up. |
| | KEY5~KEY8 | TKM1C1 | NSI | — | Touch key inputs |
| PC0/KEY9~<br>PC3/KEY12 | PC0~PC3 | PCPU | ST | CMOS | General purpose I/O. Register enabled pull-up. |
| | KEY9~KEY12 | TKM2C1 | NSI | — | Touch key inputs |
| PC4/KEY13~<br>PC7/KEY16 | PC4~PC7 | PCPU | ST | CMOS | General purpose I/O. Register enabled pull-up. |
| | KEY13~KEY16 | TKM3C1 | NSI | — | Touch key inputs |
| $\overline{RES}$ | Reset pin | — | ST | — | — |
| VDD | Power supply * | — | PWR | — | — |
| VSS | Ground ** | — | PWR | — | — |

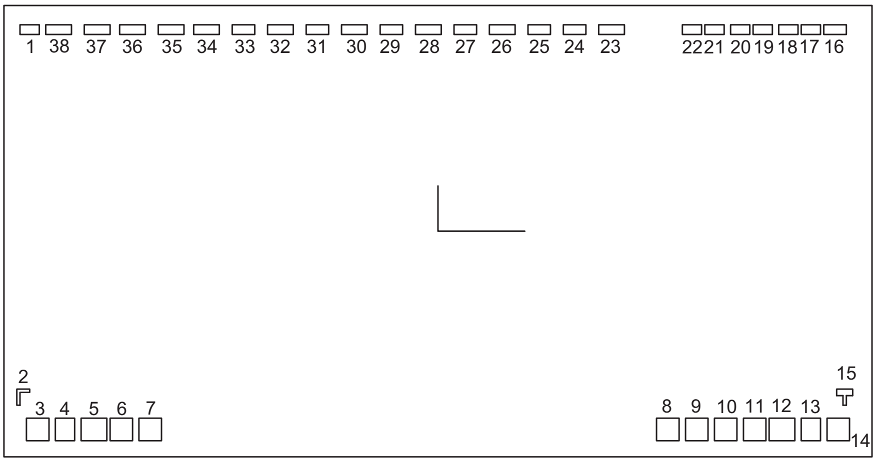
Note: I/T: Input type
O/T: Output type
OPT: Optional by register selection
PWR: Power
ST: Schmitt Trigger input
CMOS: CMOS output
NMOS: NMOS output
NSI; Non-standard input

### BS83C24-3

| Pin Name | Function | OPT | I/T | O/T | Description |
|---|---|---|---|---|---|
| PA0/SDI/SDA | PA0 | PAWU<br>PAPU | ST | CMOS | General purpose I/O. Register enabled pull-up and wake-up. |
| | SDI | SIMC0 | ST | — | SPI data input |
| | SDA | SIMC0 | ST | NMOS | $I^2C$ data |
| PA1/SDO | PA1 | PAWU<br>PAPU | ST | CMOS | General purpose I/O. Register enabled pull-up and wake-up. |
| | SDO | SIMC0 | — | CMOS | SPI data output |
| PA2/SCK/SCL | PA2 | PAWU<br>PAPU | ST | CMOS | General purpose I/O. Register enabled pull-up and wake-up. |
| | SCK | SIMC0 | ST | CMOS | SPI serial clock |
| | SCL | SIMC0 | ST | NMOS | $I^2C$ clock |
| PA3/$\overline{\text{SCS}}$ | PA3 | PAWU<br>PAPU | ST | CMOS | General purpose I/O. Register enabled pull-up and wake-up. |
| | $\overline{\text{SCS}}$ | SIMC0 | ST | CMOS | SPI slave select |
| PA4/INT | PA4 | PAWU<br>PAPU | ST | CMOS | General purpose I/O. Register enabled pull-up and wake-up. |
| | INT | INTEG | ST | — | External interrupt |
| $\overline{\text{RES}}$ | Reset pin | — | ST | — | — |
| PB0/KEY1~<br>PB3/KEY4 | PB0~PB3 | PBPU | ST | CMOS | General purpose I/O. Register enabled pull-up. |
| | KEY1~KEY4 | TKM0C1 | NSI | — | Touch key inputs |
| PB4/KEY5~<br>PB7/KEY8 | PB4~PB7 | PBPU | ST | CMOS | General purpose I/O. Register enabled pull-up. |
| | KEY5~KEY8 | TKM1C1 | NSI | — | Touch key inputs |
| PC0/KEY9~<br>PC3/KEY12 | PC0~PC3 | PCPU | ST | CMOS | General purpose I/O. Register enabled pull-up. |
| | KEY9~KEY12 | TKM2C1 | NSI | — | Touch key inputs |
| PC4/KEY13~<br>PC7/KEY16 | PC4~PC7 | PCPU | ST | CMOS | General purpose I/O. Register enabled pull-up. |
| | KEY13~KEY16 | TKM3C1 | NSI | — | Touch key inputs |
| PD0/KEY17~<br>PD3/KEY20 | PD0~PD3 | PDPU | ST | CMOS | General purpose I/O. Register enabled pull-up. |
| | KEY17~KEY20 | TKM4C1 | NSI | — | Touch key inputs |
| PD4/KEY21~<br>PD7/KEY24 | PD4~PD7 | PDPU | ST | CMOS | General purpose I/O. Register enabled pull-up. |
| | KEY21~KEY24 | TKM5C1 | NSI | — | Touch key inputs |
| PE0/PFD | PE0 | PEPU | ST | CMOS | General purpose I/O. Register enabled pull-up. |
| | PFD | TMR1C | — | CMOS | PFD output |
| PE1~PE7 | PE1~PE7 | PEPU | ST | CMOS | General purpose I/O. Register enabled pull-up. |
| PF0/TC1 | PF0 | PFPU | ST | CMOS | General purpose I/O. Register enabled pull-up. |
| | TC1 | — | ST | — | External Timer 1 clock input |
| PF1~PF3 | PF1~PF3 | PFPU | ST | CMOS | General purpose I/O. Register enabled pull-up. |
| VDD | VDD | — | PWR | — | Power supply |
| VSS | VSS | — | PWR | — | Ground |

Note: I/T: Input type; O/T: Output type
OPT: Optional by register selection
PWR: Power; ST: Schmitt Trigger input
SP: Special input; CMOS: CMOS output; NMOS: NMOS output
NSI; Non-standard input

## Pad Assignment for BS83B16G-3




## Pad Coordinates for BS83B16G-3

| Pad No. | Pad Name | X | Y | Pad No. | Pad Name | X | Y |
|---|---|---|---|---|---|---|---|
| 1 | DUMMY | -1361.480 | 677.500 | 20 | Dummy | 1007.340 | 677.500 |
| 2 | Align1 | -1379.250 | -557.780 | 21 | Dummy | 927.340 | 677.500 |
| 3 | Dummy2020 | -1335.090 | -658.000 | 22 | Dummy | 847.340 | 677.500 |
| 4 | VSS | -1240.090 | -658.000 | 23 | PB0/KEY1 | 585.120 | 677.500 |
| 5 | VSS | -1145.090 | -658.000 | 24 | PB1/KEY2 | 462.1220 | 677.500 |
| 6 | VDD | -1050.090 | -658.000 | 25 | PB2/KEY3 | 339.120 | 677.500 |
| 7 | VDD | -955.090 | -658.000 | 26 | PB3/KEY4 | 216.120 | 677.500 |
| 8 | $\overline{\text{RES}}$ | 771.020 | -658.000 | 27 | PB4/KEY5 | 93.120 | 677.500 |
| 9 | PA2/SCK/SCL | 866.020 | -658.000 | 28 | PB5/KEY6 | -29.880 | 677.500 |
| 10 | PA0/SDI/SDA | 961.020 | -658.000 | 29 | PB6/KEY7 | -152.880 | 677.500 |
| 11 | PA3/$\overline{\text{SCS}}$ | 1056.020 | -658.000 | 30 | PB7/KEY8 | -275.880 | 677.500 |
| 12 | PA4/INT | 1151.020 | -658.000 | 31 | PC0/KEY9 | -398.880 | 677.500 |
| 13 | PA1/SDO | 1246.020 | -658.000 | 32 | PC1/KEY10 | -521.880 | 677.500 |
| 14 | Dummy | 1341.020 | -658.000 | 33 | PC2/KEY11 | -644.880 | 677.500 |
| 15 | Align2 | 1361.400 | -559.990 | 34 | PC3/KEY12 | -767.880 | 677.500 |
| 16 | Dummy | 1327.340 | 677.500 | 35 | PC4/KEY13 | -890.880 | 677.500 |
| 17 | Dummy | 1247.340 | 677.500 | 36 | PC5/KEY14 | -1013.880 | 677.500 |
| 18 | Dummy | 1167.340 | 677.500 | 37 | PC6/KEY15 | -1136.880 | 677.500 |
| 19 | Dummy | 1087.340 | 677.500 | 38 | PC7/KEY16 | -1259.880 | 677.500 |

## Absolute Maximum Ratings

Supply Voltage ........ $V_{SS}$–0.3V to $V_{SS}$+6.0V
Storage Temperature ........ –50°C to 125°C
Input Voltage ........ $V_{SS}$–0.3V to $V_{DD}$+0.3V
Operating Temperature ........ –40°C to 85°
$CI_{OL}$ Total ........ 80mA
$I_{OH}$ Total ........ –80mA
Total Power Dissipation ........ 500mW

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

## D.C. Characteristics

Ta=25°C

| Symbol | Parameter | Test Conditions | | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|---|---|
| | | $V_{DD}$ | Conditions | | | | |
| $V_{DD}$ | Operating Voltage (HIRC) | — | $f_{SYS}$=8MHz | $V_{LVR}$ | — | 5.5 | V |
| | | | $f_{SYS}$=12MHz | 2.7 | — | 5.5 | V |
| | | | $f_{SYS}$=16MHz | 4.5 | — | 5.5 | V |
| $I_{DD1}$ | Operating Current (HIRC), ($f_{SYS}$=$f_H$) | 3V | No load, $f_H$=8MHz, WDT enable | — | 1.2 | 1.8 | mA |
| | | 5V | | — | 2.2 | 3.3 | mA |
| | | 3V | No load, $f_H$=12MHz, WDT enable | — | 1.6 | 2.4 | mA |
| | | 5V | | — | 3.3 | 5.0 | mA |
| | | 5V | No load, $f_H$=16MHz, WDT enable | — | 4.0 | 6.0 | mA |
| $I_{DD2}$ | Operating Current (LIRC), ($f_{SYS}$=$f_L$) for BS83B08-3/B12-3/B16-3 | 3V | No load, $f_L$=32kHz, WDT enable | — | 50 | 100 | μA |
| | | 5V | | — | 70 | 150 | μA |
| $I_{DD3}$ | Operating Current (LIRC), ($f_{SYS}$=$f_L$) for BS83C24-3 | 3V | No load, $f_L$=32kHz, WDT enable | — | 15 | 30 | μA |
| | | 5V | | — | 30 | 60 | μA |
| $I_{IDLE0}$ | IDLE0 Mode Standby Current | 3V | No load, LVR disable | — | 1.5 | 3.0 | μA |
| | | 5V | | — | 3.0 | 6.0 | μA |
| $I_{IDLE1}$ | IDLE1 Mode Standby Current | 3V | No load, LVR disable, $f_{SYS}$=12MHz on | — | 0.9 | 1.4 | mA |
| | | 5V | | — | 1.6 | 2.4 | mA |
| $I_{SLEEP}$ | SLEEP1 Mode Standby Current | 3V | No load, LVR disable | — | 1.5 | 3.0 | μA |
| | | 5V | | — | 2.5 | 5.0 | μA |
| $V_{IL1}$ | Input Low Voltage for I/O Ports or Input Pins except $\overline{RES}$ pin | 5V | — | 0 | — | 1.5 | V |
| | | — | | 0 | — | 0.2$V_{DD}$ | V |
| $V_{IH1}$ | Input High Voltage for I/O Ports or Input Pins except $\overline{RES}$ pin | 5V | — | 3.5 | — | 5.0 | V |
| | | — | | 0.8$V_{DD}$ | — | $V_{DD}$ | V |
| $V_{IL2}$ | Input Low Voltage ($\overline{RES}$) | — | — | 0 | — | 0.4$V_{DD}$ | V |
| $V_{IH2}$ | Input High Voltage ($\overline{RES}$) | — | — | 0.9$V_{DD}$ | — | $V_{DD}$ | V |

Ta=25°C

| Symbol | Parameter | Test Conditions | | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|---|---|
| | | $V_{DD}$ | Conditions | | | | |
| $V_{LVR}$ | LVR Voltage Level | — | LVR Enable | –5% | 2.55 | +5% | V |
| $V_{OL}$ | Output Low Voltage I/O Port | 3V | $I_{OL}$=9mA | — | — | 0.3 | V |
| | | 5V | $I_{OL}$=20mA | — | — | 0.5 | V |
| $V_{OH}$ | Output High Voltage I/O Port | 3V | $I_{OH}$=–3.2mA | 2.7 | — | — | V |
| | | 5V | $I_{OH}$=–7.4mA | 4.5 | — | — | V |
| $R_{PH}$ | Pull-high Resistance for I/O Ports | 3V | — | 20 | 60 | 100 | kΩ |
| | | 5V | | 10 | 30 | 50 | kΩ |

## A.C. Characteristics

Ta=25°C

| Symbol | Parameter | Test Conditions | | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|---|---|
| | | $V_{DD}$ | Conditions | | | | |
| $f_{CPU}$ | Operating Clock | — | $V_{LVR}$~5.5V | DC | — | 8 | MHz |
| | | | 2.7V~5.5V | DC | — | 12 | MHz |
| | | | 4.5V~5.5V | DC | — | 16 | MHz |
| $f_{HIRC}$ | System Clock (HIRC) | 3V/5V | Ta=25°C | –2% | 8 | +2% | MHz |
| | | 3V/5V | Ta=25°C | –2% | 12 | +2% | MHz |
| | | 5V | Ta=25°C | –2% | 16 | +2% | MHz |
| | | 3V/5V | Ta=0~70°C | –4% | 8 | +3% | MHz |
| | | 3V/5V | Ta=0~70°C | –4% | 12 | +3% | MHz |
| | | 5V | Ta=0~70°C | –4% | 16 | +3% | MHz |
| | | 2.5V~4.0V | Ta=0~70°C | –9% | 8 | +6% | MHz |
| | | 3.0V~5.5V | Ta=0~70°C | –5% | 8 | +12% | MHz |
| | | 2.7V~4.0V | Ta=0~70°C | –9% | 12 | +5% | MHz |
| | | 3.0V~5.5V | Ta=0~70°C | –5% | 12 | +11% | MHz |
| | | 4.5V~5.5V | Ta=0~70°C | –5% | 16 | +5% | MHz |
| | | 2.5V~4.0V | Ta= –40°C~85°C | –12% | 8 | +6% | MHz |
| | | 3.0V~5.5V | Ta= –40°C~85°C | –8% | 8 | +12% | MHz |
| | | 2.7V~4.0V | Ta= –40°C~85°C | –13% | 12 | +5% | MHz |
| | | 3.0V~5.5V | Ta= –40°C~85°C | –8% | 12 | +11% | MHz |
| | | 4.5V~5.5V | Ta= –40°C~85°C | –7% | 16 | +5% | MHz |

| Symbol | Parameter | Test Conditions | | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|---|---|
| | | $V_{DD}$ | Conditions | | | | |
| $f_{LIRC}$ | System Clock (LIRC) | 5V | — | –10% | 32 | +10% | kHz |
| | | 2.2V~ 5.5V | Ta=–40°C~+85°C | –50% | 32 | +60% | kHz |
| $f_{TIMER}$ | Timer Input Pin Frequency | — | — | — | — | 1 | $f_{SYS}$ |
| $t_{RES}$ | External Reset Low Pulse Width | — | — | 1 | — | — | μs |
| $t_{INT}$ | Interrupt Pulse Width | — | — | 1 | — | — | μs |
| $t_{LVR}$ | Low Voltage Width to Reset | — | — | 60 | 120 | 240 | μs |
| $t_{EERD}$ | EEPROM Read Time | — | — | — | 2 | 4 | $t_{SYS}$ |
| $t_{EEWR}$ | EEPROM Write Time | — | — | — | 2 | 4 | ms |
| $t_{SST}$ | System Start-up Timer Period (Wake-up from HALT) | — | $f_{SYS}$=HIRC | — | 15~16 | — | $t_{SYS}$ |
| | | | $f_{SYS}$=LIRC | — | 1~2 | — | |

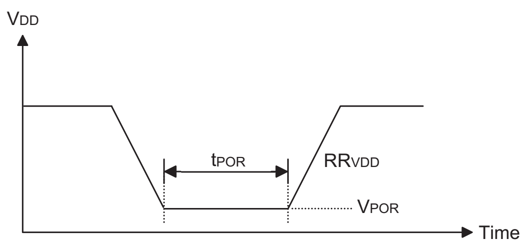
Note: 1. $t_{SYS}=1/f_{SYS}$

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

## Power-on Reset Characteristics

Ta=25°C

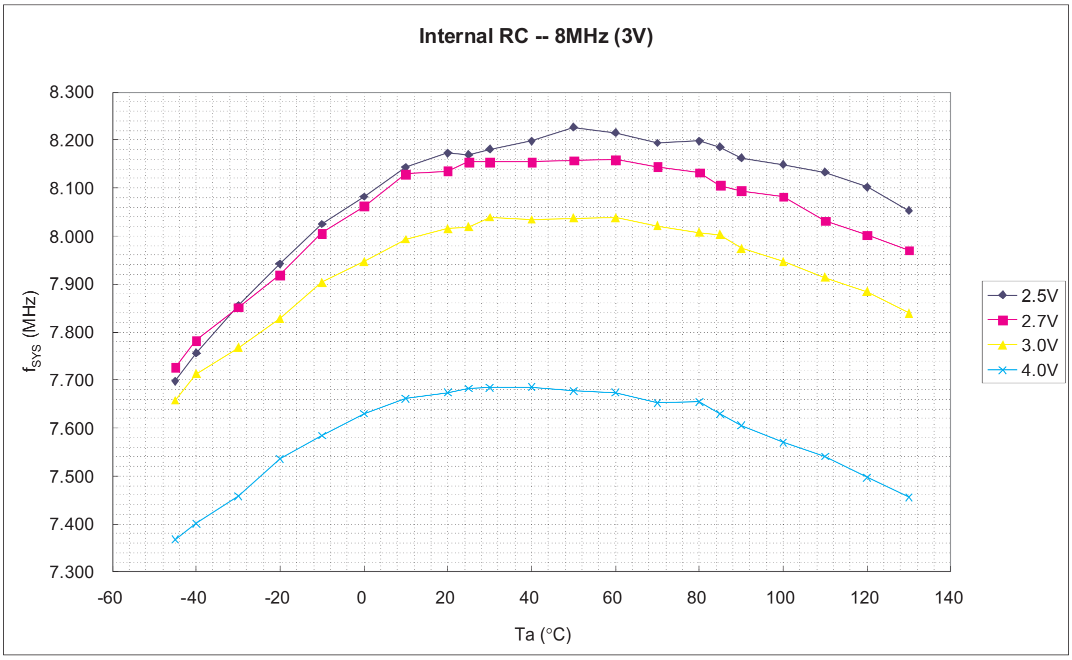
| Symbol | Parameter | Test Conditions | | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|---|---|
| | | $V_{DD}$ | Conditions | | | | |
| $V_{POR}$ | VDD Start Voltage to Ensure Power-on Reset | — | — | — | — | 100 | mV |
| $R_{POR\ AC}$ | VDD Raising Rate to Ensure Power-on Reset | — | — | 0.035 | — | — | V/ms |
| $t_{POR}$ | Minimum Time for VDD Stays at $V_{POR}$ to Ensure Power-on Reset | — | — | 1 | — | — | ms |

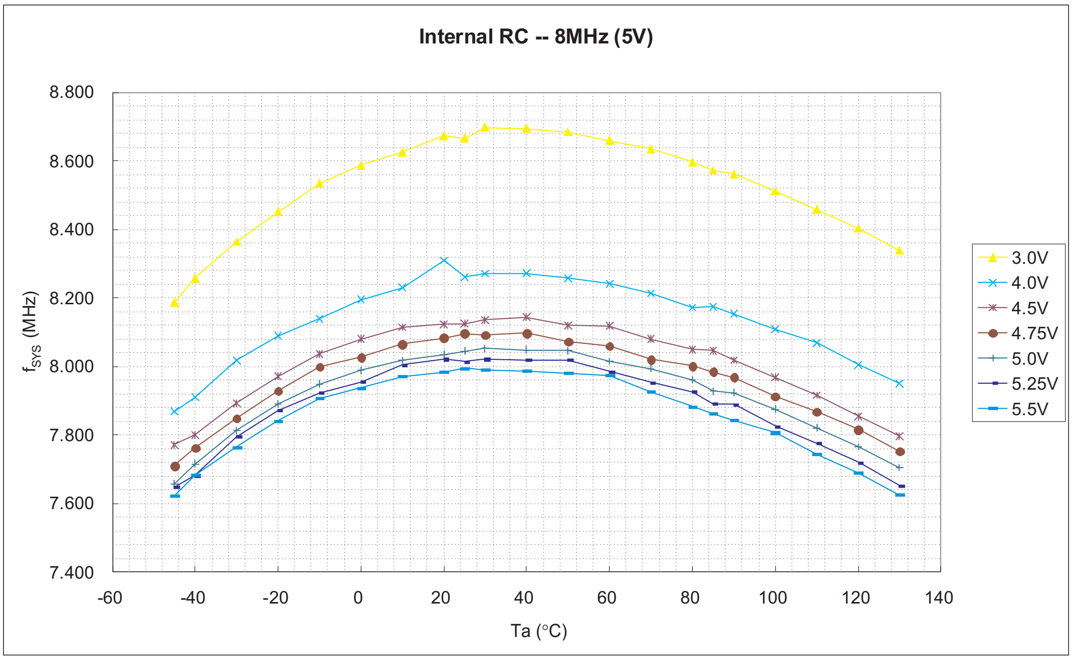


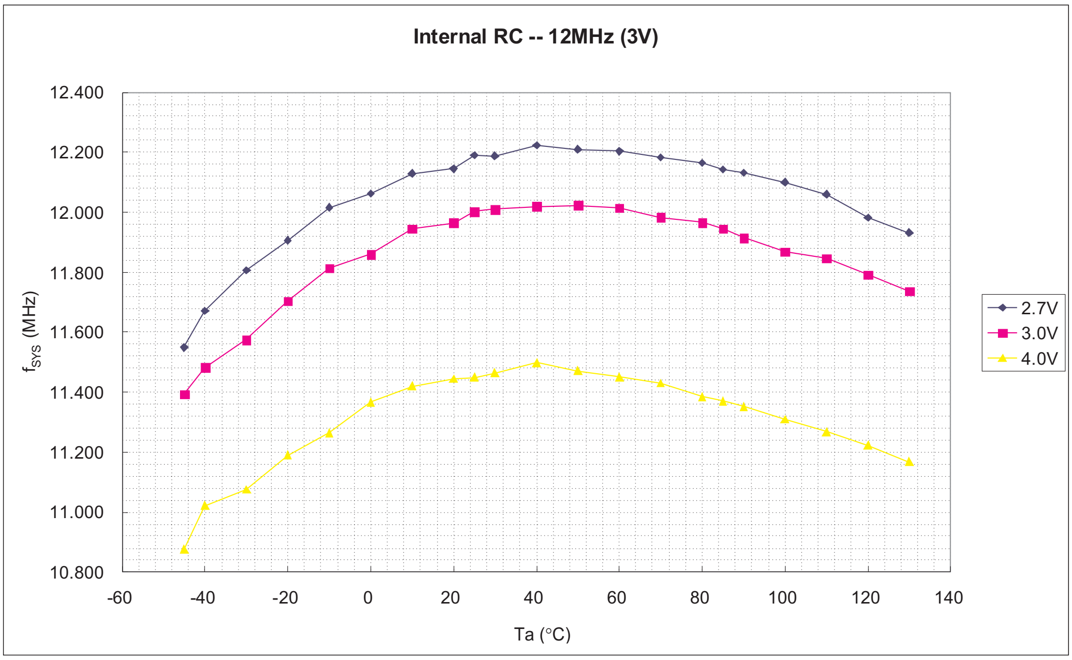
## Oscillator Temperature/Frequency Characteristics

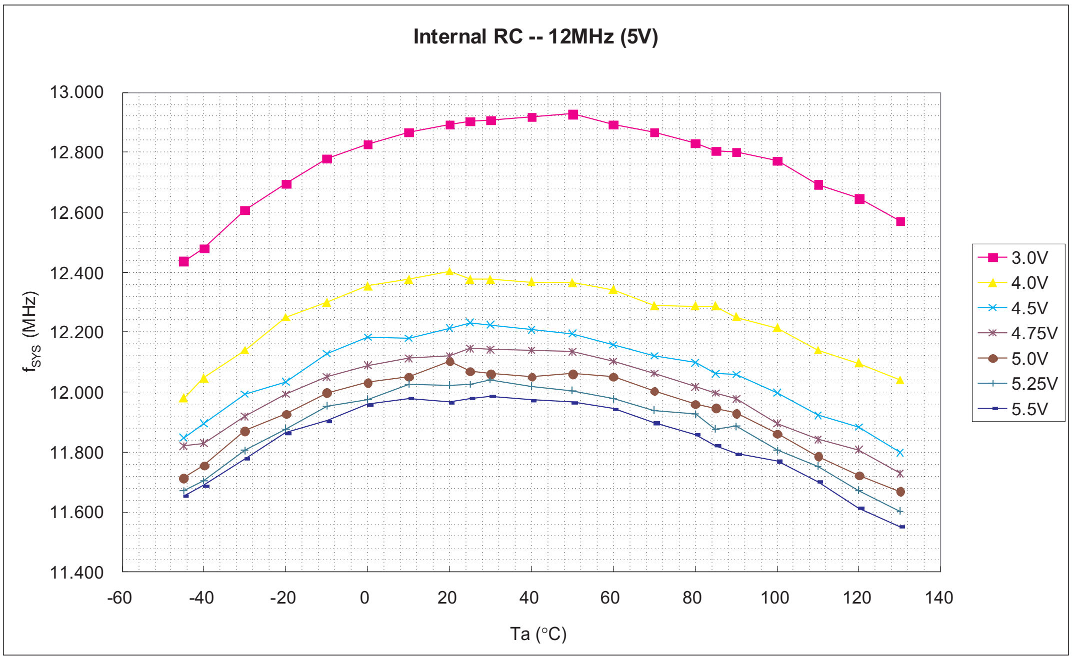
The following characteristic graphics depicts typical oscillator behavior. The data presented here is a statistical summary of data gathered on units from different lots over a period of time. This is for information only and the figures were not tested during manufacturing.

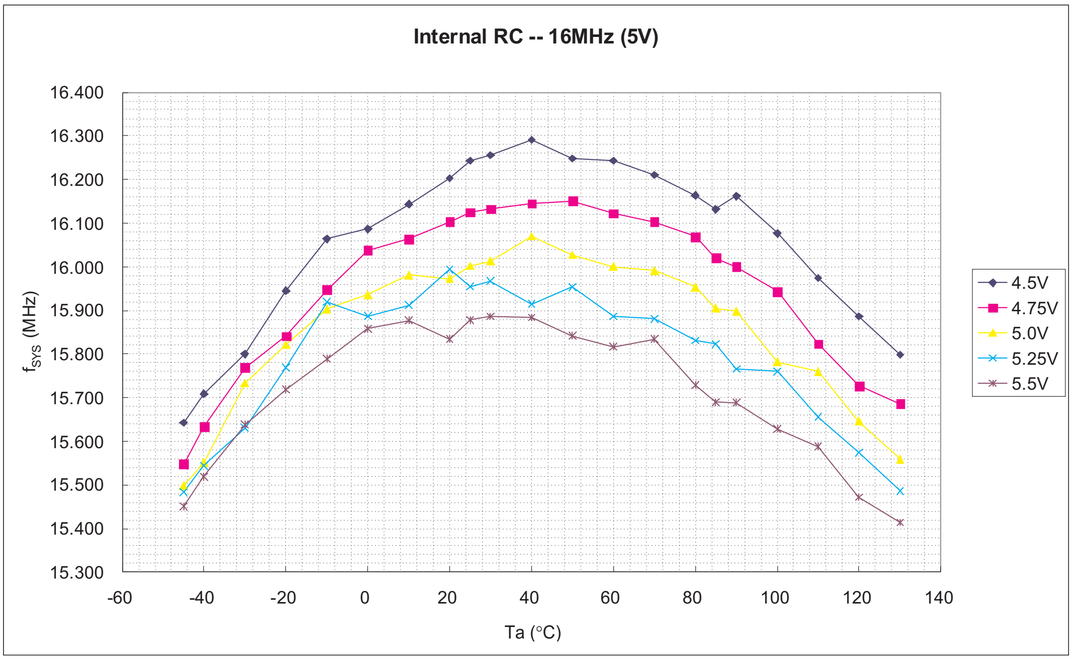
In some of the graphs, the data exceeding the specified operating range are shown for information purposes only. The device will operate properly only within the specified range.




Internal RC -- 12MHz (3V)
12.400
12.200
12.000
11.800
11.600
11.400
11.200
11.000
10.800
$f_{SYS}$ (MHz)
-60
-40
-20
0
20
40
60
80
100
120
140
Ta (°C)
2.7V
3.0V
4.0V


Internal RC -- 12MHz (5V)
13.000
12.800
12.600
12.400
12.200
12.000
11.800
11.600
11.400
$f_{SYS}$ (MHz)
-60
-40
-20
0
20
40
60
80
100
120
140
Ta (°C)
3.0V
4.0V
4.5V
4.75V
5.0V
5.25V
5.5V

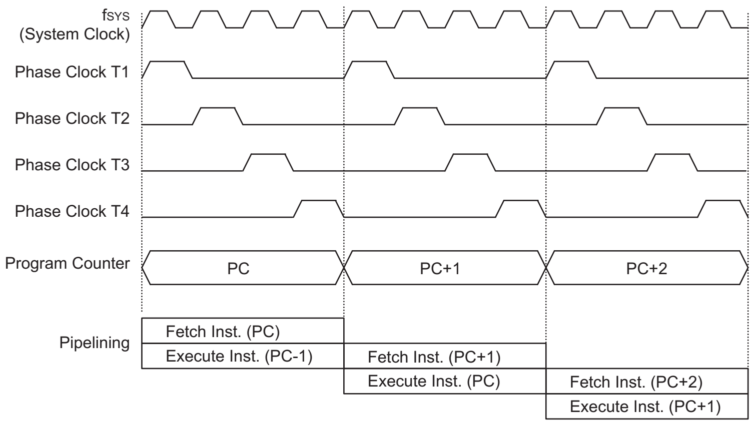

Internal RC -- 16MHz (5V)
16.400
16.300
16.200
16.100
16.000
15.900
15.800
15.700
15.600
15.500
15.400
15.300
$f_{SYS}$ (MHz)
-60
-40
-20
0
20
40
60
80
100
120
140
Ta (°C)
4.5V
4.75V
5.0V
5.25V
5.5V

## System Architecture

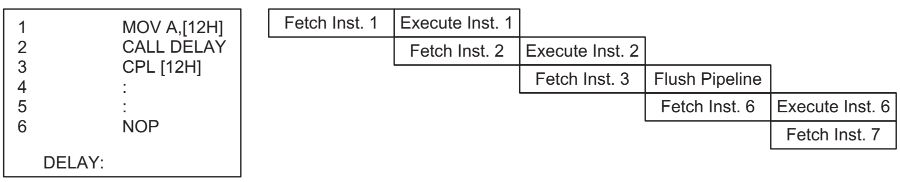
A key factor in the high-performance features of the Holtek range of microcontroller is attributed to their internal system architecture. The range of devices take advantage of the usual features found within RISC microcontroller providing increased speed of operation and enhanced performance. The pipelining 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 control system with maximum reliability and flexibility. This makes the device suitable for low-cost, high-volume production for controller applications.

### Clocking and Pipelining

The main system clock, derived from either a high or low speed 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**




**Instruction Fetching**

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.

## 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.

<table>
<tr><th rowspan="2">Device</th><th colspan="2">Program Counter</th></tr>
<tr><th>Program Counter High Byte</th><th>PCL Register</th></tr>
<tr><td>BS83B08-3<br>BS83B12-3<br>BS83B16-3<br>BS83B16G-3</td><td>PC10~PC8</td><td rowspan="2">PCL7~PCL0</td></tr>
<tr><td>BS83C24-3</td><td>PC11~PC8</td></tr>
</table>

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.

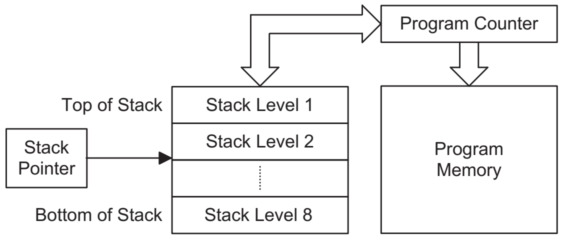
## Stack

This is a special part of the memory which is used to save the contents of the Program Counter  only. The stack has multiple levels depending upon the device and is 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.

| Device | Stack Levels |
|---|---|
| BS83B08-3<br>BS83B12-3<br>BS83B16-3<br>BS83B16G-3 | 4 |
| BS83C24-3 | 8 |




### 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

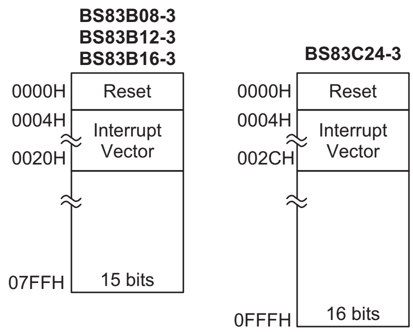
## Flash Program Memory

The Program Memory is the location where the user code or program is stored. For this device series 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, these Flash devices offer users the flexibility to conveniently debug and develop their applications while also offering a means of field programming and updating.

### Structure

The Program Memory has a capacity of 2K×15 or 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.

| Device | Capacity |
|---|---|
| BS83B08-3<br>BS83B12-3<br>BS83B16-3<br>BS83B16G-3 | 2K×15 |
| BS83C24-3 | 4K×16 |




**Flash Program Memory Structure**

## Special Vectors

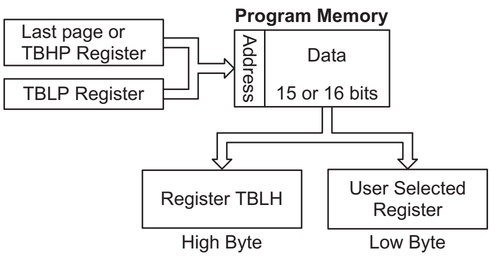
Within the Program Memory, certain locations are reserved for the reset and interrupts. The location 000H is reserved for use by the device reset for program initialisation. After a device reset is initiated, the program will jump to this location and begin execution.

## Look-up Table

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




| Instruction | Table Location Bits | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | b11 | b10 | b9 | b8 | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
| TABRDC [m] | PC11 | PC10 | PC9 | PC8 | @7 | @6 | @5 | @4 | @3 | @2 | @1 | @0 |
| TABRDL [m] | 1 | 1 | 1 | 1 | @7 | @6 | @5 | @4 | @3 | @2 | @1 | @0 |

**Table Location**

Note: PC11~PC8: Current Program Counter bits
@7~@0: Table Pointer TBLP bits
For the BS83B08-3, BS83B12-3 and BS83B16-3/BS83B16G-3, the Table address location is 11 bits, i.e. from b10~b0.
For the BS83C24-3, the Table address location is 12 bits, i.e. from b11~b0

### 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 "700H" which refers to the start address of the last page within the 2K words Program Memory of the device. The table pointer is setup here to have an initial value of "06H". This will ensure that the first data read from the data table will be at the Program Memory address "706H" or 6 locations after the start of the last page. Note that the value for the table pointer is referenced to the first address of the present page if the "TABRD [m]" instruction is being used. The high byte of the table data which in this case is equal to zero will be transferred to the TBLH register automatically when the "TABRD [m]" instruction is executed.

Because the TBLH register is a read-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.

```
Tempreg1  db ?      ; temporary register #1
tempreg2  db ?      ; temporary register #2
:
:
mov a,06h           ; initialise low table pointer - note that this address
mov tblp,a          ; is referenced
mov a,07h           ; initialise high table pointer
mov tbhp,a
:
:
tabrd tempreg1      ; transfers value in table referenced by table pointer data at
                    ; program memory address "706H" 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 "705H" transferred to tempreg2 and TBLH in
                    ; this example the data "1AH" is transferred to tempreg1 and data
                    ; "0FH" to register tempreg2
:
:
org 700h            ; sets initial address of program memory

dc 00Ah, 00Bh, 00Ch, 00Dh, 00Eh, 00Fh, 01Ah, 01Bh
:
:
```

## In Circuit Programming

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.

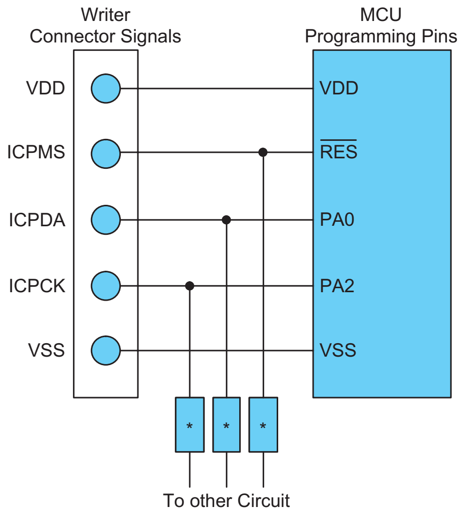
As an additional convenience, Holtek has provided a means of programming the microcontroller in-circuit using a 5-pin interface. This provides manufacturers with the possibility of manufacturing their circuit boards complete with a programmed or un-programmed microcontroller, and then programming or upgrading the program at a later stage. This enables product manufacturers to easily keep their manufactured products supplied with the latest program releases without removal and re-insertion of the device.

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

| Holtek Writer | Pin Name | Pin Description |
|---|---|---|
| ICPDA | PA0 | Serial Address and data -- read/write |
| ICPCK | PA2 | Address and data serial clock input |
| ICPMS | $\overline{\text{RES}}$ | Programming Mode Select |
| VDD | VDD | Power Supply (5.0V) |
| VSS | VSS | Ground |

The Program Memory and EEPROM data memory can both be programmed serially in-circuit using this 5-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 and one line for the reset. The technical details regarding the in-circuit programming of the devices are beyond the scope of this document and will be supplied in supplementary literature.

During the programming process the $\overline{\text{RES}}$ pin will be held low by the programmer disabling the normal operation of the microcontroller and taking control of the PA0 and PA2 I/O pins for data and clock programming purposes. The user must there take care to ensure that no other outputs are connected to these two pins.




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

## RAM Data Memory

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

### Structure

Divided into two sections, the first of these is an area of RAM, known as the Special Function Data Memory. Here are located registers which are necessary for correct operation of the device. Many of these registers can be read from and written to directly under program control, however, some remain protected from user manipulation.

| Device | Capacity | Bank 0 | Bank 1 | Bank 2 | Bank 3 |
|---|---|---|---|---|---|
| BS83B08-3 | 160×8 | 60H~FFH | — | — | — |
| BS83B12-3 | 288×8 | 60H~FFH | 80H~FFH | — | — |
| BS83B16-3<br>BS83B16G-3 | 288×8 | 60H~FFH | 80H~FFH | — | — |
| BS83C24-3 | 512×8 | 80H~FFH | 80H~FFH | 80H~FFH | 80H~FFH |

**General Purpose Data Memory**

The second area of Data Memory is known as the General Purpose Data Memory, which is reserved for general purpose use. All locations within this area are read and write accessible under program control.

The overall Data Memory is subdivided into several banks for the devices. 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 all devices is the address 00H.

## 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.

**BS83B08-3**

| | Bank 0, Bank 1 | | Bank 0 | Bank 1 |
|---|---|---|---|---|
| 00H | IAR0 | 30H | TKM116DH | |
| 01H | MP0 | 31H | TKM116DL | |
| 02H | IAR1 | 32H | Reserved | |
| 03H | MP1 | 33H | Reserved | |
| 04H | BP | 34H | TKM1C0 | |
| 05H | ACC | 35H | TKM1C1 | |
| 06H | PCL | 36H | TKM1C2 | |
| 07H | TBLP | 37H | TKM1C3 | |
| 08H | TBLH | 38H | Unused | |
| 09H | TBHP | 39H | Unused | |
| 0AH | STATUS | 3AH | Unused | |
| 0BH | SMOD | 3BH | Unused | |
| 0CH | Unused | 3CH | Unused | |
| 0DH | INTEG | 3DH | CTRL | |
| 0EH | INTC0 | 3EH | Unused | |
| 0FH | INTC1 | 3FH | Unused | |
| 10H | INTC2 | 40H | Unused | EEC |
| 11H | MFI0 | 41H | Unused | |
| 12H | Unused | 42H | Unused | |
| 13H | Unused | 43H | Unused | |
| 14H | PA | 44H | Unused | |
| 15H | PAC | 45H | Unused | |
| 16H | PAPU | 46H | Unused | |
| 17H | PAWU | 47H | Unused | |
| 18H | Unused | 48H | Unused | |
| 19H | Unused | 49H | Unused | |
| 1AH | WDTC | 4AH | Unused | |
| 1BH | TBC | 4BH | Unused | |
| 1CH | TMR | 4CH | Unused | |
| 1DH | TMRC | 4DH | Unused | |
| 1EH | EEA | 4EH | Unused | |
| 1FH | EED | 4FH | Unused | |
| 20H | PB | 50H | Unused | |
| 21H | PBC | 51H | Unused | |
| 22H | PBPU | 52H | Unused | |
| 23H | I2CTOC | 53H | Unused | |
| 24H | SIMC0 | 54H | Unused | |
| 25H | SIMC1 | 55H | Unused | |
| 26H | SIMD | 56H | Unused | |
| 27H | SIMA/SIMC2 | 57H | Unused | |
| 28H | TKM016DH | 58H | Unused | |
| 29H | TKM016DL | 59H | Unused | |
| 2AH | Reserved | 5AH | Unused | |
| 2BH | Reserved | 5BH | Unused | |
| 2CH | TKM0C0 | 5CH | Unused | |
| 2DH | TKM0C1 | 5DH | Unused | |
| 2EH | TKM0C2 | 5EH | Unused | |
| 2FH | TKM0C3 | 5FH | Unused | |

**BS83B12-3**

| | Bank 0, Bank 1 | | Bank 0 | Bank 1 |
|---|---|---|---|---|
| 00H | IAR0 | 30H | TKM116DH | |
| 01H | MP0 | 31H | TKM116DL | |
| 02H | IAR1 | 32H | Reserved | |
| 03H | MP1 | 33H | Reserved | |
| 04H | BP | 34H | TKM1C0 | |
| 05H | ACC | 35H | TKM1C1 | |
| 06H | PCL | 36H | TKM1C2 | |
| 07H | TBLP | 37H | TKM1C3 | |
| 08H | TBLH | 38H | PC | |
| 09H | TBHP | 39H | PCC | |
| 0AH | STATUS | 3AH | PCPU | |
| 0BH | SMOD | 3BH | Unused | |
| 0CH | Unused | 3CH | Unused | |
| 0DH | INTEG | 3DH | CTRL | |
| 0EH | INTC0 | 3EH | Unused | |
| 0FH | INTC1 | 3FH | Unused | |
| 10H | INTC2 | 40H | Unused | EEC |
| 11H | MFI0 | 41H | Unused | |
| 12H | MFI1 | 42H | Unused | |
| 13H | Unused | 43H | Unused | |
| 14H | PA | 44H | Unused | |
| 15H | PAC | 45H | Unused | |
| 16H | PAPU | 46H | Unused | |
| 17H | PAWU | 47H | Unused | |
| 18H | Unused | 48H | TKM216DH | |
| 19H | Unused | 49H | TKM216DL | |
| 1AH | WDTC | 4AH | Reserved | |
| 1BH | TBC | 4BH | Reserved | |
| 1CH | TMR | 4CH | TKM2C0 | |
| 1DH | TMRC | 4DH | TKM2C1 | |
| 1EH | EEA | 4EH | TKM2C2 | |
| 1FH | EED | 4FH | TKM2C3 | |
| 20H | PB | 50H | Unused | |
| 21H | PBC | 51H | Unused | |
| 22H | PBPU | 52H | Unused | |
| 23H | I2CTOC | 53H | Unused | |
| 24H | SIMC0 | 54H | Unused | |
| 25H | SIMC1 | 55H | Unused | |
| 26H | SIMD | 56H | Unused | |
| 27H | SIMA/SIMC2 | 57H | Unused | |
| 28H | TKM016DH | 58H | Unused | |
| 29H | TKM016DL | 59H | Unused | |
| 2AH | Reserved | 5AH | Unused | |
| 2BH | Reserved | 5BH | Unused | |
| 2CH | TKM0C0 | 5CH | Unused | |
| 2DH | TKM0C1 | 5DH | Unused | |
| 2EH | TKM0C2 | 5EH | Unused | |
| 2FH | TKM0C3 | 5FH | Unused | |

**Special Purpose Data Memory – BS83B08-3/BS83B12-3**

Note: The "Reserved" bytes shown in the table must not be modified by the user.

**BS83B16-3**

| Address | Bank 0, Bank 1 | Address | Bank 0 | Bank 1 |
|---|---|---|---|---|
| 00H | IAR0 | 30H | TKM116DH | |
| 01H | MP0 | 31H | TKM116DL | |
| 02H | IAR1 | 32H | Reserved | |
| 03H | MP1 | 33H | Reserved | |
| 04H | BP | 34H | TKM1C0 | |
| 05H | ACC | 35H | TKM1C1 | |
| 06H | PCL | 36H | TKM1C2 | |
| 07H | TBLP | 37H | TKM1C3 | |
| 08H | TBLH | 38H | PC | |
| 09H | TBHP | 39H | PCC | |
| 0AH | STATUS | 3AH | PCPU | |
| 0BH | SMOD | 3BH | Unused | |
| 0CH | Unused | 3CH | Unused | |
| 0DH | INTEG | 3DH | CTRL | |
| 0EH | INTC0 | 3EH | Unused | |
| 0FH | INTC1 | 3FH | Unused | |
| 10H | INTC2 | 40H | Unused | EEC |
| 11H | MFI0 | 41H | Unused | |
| 12H | MFI1 | 42H | Unused | |
| 13H | Unused | 43H | Unused | |
| 14H | PA | 44H | Unused | |
| 15H | PAC | 45H | Unused | |
| 16H | PAPU | 46H | Unused | |
| 17H | PAWU | 47H | Unused | |
| 18H | Unused | 48H | TKM216DH | |
| 19H | Unused | 49H | TKM216DL | |
| 1AH | WDTC | 4AH | Reserved | |
| 1BH | TBC | 4BH | Reserved | |
| 1CH | TMR | 4CH | TKM2C0 | |
| 1DH | TMRC | 4DH | TKM2C1 | |
| 1EH | EEA | 4EH | TKM2C2 | |
| 1FH | EED | 4FH | TKM2C3 | |
| 20H | PB | 50H | TKM316DH | |
| 21H | PBC | 51H | TKM316DL | |
| 22H | PBPU | 52H | Reserved | |
| 23H | I2CTOC | 53H | Reserved | |
| 24H | SIMC0 | 54H | TKM3C0 | |
| 25H | SIMC1 | 55H | TKM3C1 | |
| 26H | SIMD | 56H | TKM3C2 | |
| 27H | SIMA/SIMC2 | 57H | TKM3C3 | |
| 28H | TKM016DH | 58H | Unused | |
| 29H | TKM016DL | 59H | Unused | |
| 2AH | Reserved | 5AH | Unused | |
| 2BH | Reserved | 5BH | Unused | |
| 2CH | TKM0C0 | 5CH | Unused | |
| 2DH | TKM0C1 | 5DH | Unused | |
| 2EH | TKM0C2 | 5EH | Unused | |
| 2FH | TKM0C3 | 5FH | Unused | |

**Special Purpose Data Memory – BS83B16-3/BS83B16G-3**

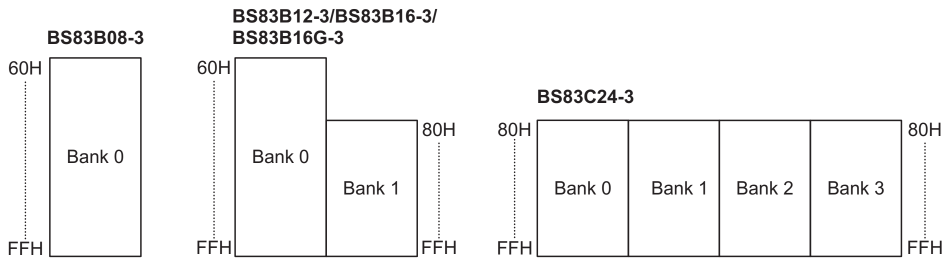
Note: The ″Reserved″ bytes shown in the table must not be modified by the user.

**BS83C24-3**

| Address | Bank 0, 1, 2, 3 | Address | Bank 0, 2, 3 / Bank 1 | Address | Bank 0, 1, 2, 3 |
|---|---|---|---|---|---|
| 00H | IAR0 | 30H | TKM116DH | 60H | TKM516DH |
| 01H | MP0 | 31H | TKM116DL | 61H | TKM516DL |
| 02H | IAR1 | 32H | Reserved | 62H | Reserved |
| 03H | MP1 | 33H | Reserved | 63H | Reserved |
| 04H | BP | 34H | TKM1C0 | 64H | TKM5C0 |
| 05H | ACC | 35H | TKM1C1 | 65H | TKM5C1 |
| 06H | PCL | 36H | TKM1C2 | 66H | TKM5C2 |
| 07H | TBLP | 37H | TKM1C3 | 67H | TKM5C3 |
| 08H | TBLH | 38H | PC | 68H | PF |
| 09H | TBHP | 39H | PCC | 69H | PFC |
| 0AH | STATUS | 3AH | PCPU | 6AH | PFPU |
| 0BH | SMOD | 3BH | Unused | 6BH | TMR1H |
| 0CH | INTC3 | 3CH | Unused | 6CH | TMR1L |
| 0DH | INTEG | 3DH | CTRL | 6DH | TMR1C |
| 0EH | INTC0 | 3EH | Unused | 6EH | Unused |
| 0FH | INTC1 | 3FH | Unused | 6FH | Unused |
| 10H | INTC2 | 40H | Unused / EEC | 70H | Unused |
| 11H | MFI0 | 41H | PD | 71H | Unused |
| 12H | MFI1 | 42H | PDC | 72H | Unused |
| 13H | MFI2 | 43H | PDPU | 73H | Unused |
| 14H | PA | 44H | PE | 74H | Unused |
| 15H | PAC | 45H | PEC | 75H | Unused |
| 16H | PAPU | 46H | PEPU | 76H | Unused |
| 17H | PAWU | 47H | Unused | 77H | Unused |
| 18H | Unused | 48H | TKM216DH | 78H | Unused |
| 19H | Unused | 49H | TKM216DL | 79H | Unused |
| 1AH | WDTC | 4AH | Reserved | 7AH | Unused |
| 1BH | TBC | 4BH | Reserved | 7BH | Unused |
| 1CH | TMR0 | 4CH | TKM2C0 | 7CH | Unused |
| 1DH | TMR0C | 4DH | TKM2C1 | 7DH | Unused |
| 1EH | EEA | 4EH | TKM2C2 | 7EH | Unused |
| 1FH | EED | 4FH | TKM2C3 | 7FH | Unused |
| 20H | PB | 50H | TKM316DH | | |
| 21H | PBC | 51H | TKM316DL | | |
| 22H | PBPU | 52H | Reserved | | |
| 23H | I2CTOC | 53H | Reserved | | |
| 24H | SIMC0 | 54H | TKM3C0 | | |
| 25H | SIMC1 | 55H | TKM3C1 | | |
| 26H | SIMD | 56H | TKM3C2 | | |
| 27H | SIMA/SIMC2 | 57H | TKM3C3 | | |
| 28H | TKM016DH | 58H | TKM416DH | | |
| 29H | TKM016DL | 59H | TKM416DL | | |
| 2AH | Reserved | 5AH | Reserved | | |
| 2BH | Reserved | 5BH | Reserved | | |
| 2CH | TKM0C0 | 5CH | TKM4C0 | | |
| 2DH | TKM0C1 | 5DH | TKM4C1 | | |
| 2EH | TKM0C2 | 5EH | TKM4C2 | | |
| 2FH | TKM0C3 | 5FH | TKM4C3 | | |

**Special Purpose Data Memory – BS83C24-3**

Note: The "Reserved" bytes shown in the table must not be modified by the user.




**General Purpose Data Memory**

## 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. Note that for this series of devices, the Memory Pointers, MP0 and MP1, are both 8-bit registers and used to access the Data Memory together with their corresponding indirect addressing registers IAR0 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 example shown above, no reference is made to specific RAM addresses.

## Bank Pointer – BP

For this series of devices, the Data Memory is divided into several banks. Selecting the required Data Memory area is achieved using the Bank Pointer. In the BS83B08-3, BS83B12-3, BS83B16-3 and BS83B16G-3, the data memory is divided into two banks .The Bit 0 is used to select Data Memory Banks 0~1. In the BS83C24-3, the data memory is divided into four banks. The Bit 0 and Bit 1 are used to select Data Memory Banks 0~3.

The Data Memory is initialised to Bank 0 after a reset, except for a WDT time-out reset in the Power Down Mode, in which case, the Data Memory bank remains unaffected. It should be noted that the Special Function Data Memory is not affected by the bank selection, which means that the Special Function Registers can be accessed from within any bank. 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.

**Bank Pointer Register -- BS83B08-3, BS83B12-3, BS83B16-3 and BS83B16G-3**

| 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

**Bank Pointer Register -- BS83C24-3**

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

Bit 7 ~ 1 unimplemented, read as ″0″

Bit 0 **DMBP1~DMBP0**: select data memory banks
0: bank 0
1: bank 1
2: bank 2
3: 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 pointer and indicates 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
**C** is also affected by a rotate through carry instruction.

## EEPROM Data Memory

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

### EEPROM Data Memory Structure

The EEPROM Data Memory capacity is 64×8 or 128×8 bits for this series of devices. 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 data register in Bank 0 and a single control register in Bank 1.

| Device | Capacity | Address |
|---|---|---|
| BS83B08-3/B12-3/B16-3/B16G-3 | 64×8 | 00H ~ 3FH |
| BS83C24-3 | 128×8 | 00H ~ 7FH |

**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 was as any other Special Function Register. The EEC register however, being located in Bank1, cannot be addressed directly and can only be read from or written to indirectly using the MP1 Memory Pointer and Indirect Addressing Register, IAR1. Because the EEC control register is located at address 40H in Bank 1, the MP1 Memory Pointer must first be set to the value 40H and the Bank Pointer register, BP, set to the value, 01H, before any operations on the EEC register are executed.

**BS83B08-3/B12-3/B16-3/B16G-3**

• EEPROM Register List

| Name | Bit | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| EEA | — | — | D5 | D4 | D3 | D2 | D1 | D0 |
| EED | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| EEC | — | — | — | — | WREN | WR | RDEN | RD |

• EEA Register

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

"x" unknown

Bit 7~6 unimplemented, read as "0"

Bit 5~0 Data EEPROM address
Data EEPROM address bit 5~bit 0

**BS83C24-3**

• EEPROM Register List

| Name | Bit | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| EEA | — | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| EED | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| EEC | — | — | — | — | WREN | WR | RDEN | RD |

• **EEA Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | — | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| 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" unknown

Bit 7 unimplemented, read as "0"

Bit 6~0 Data EEPROM address
Data EEPROM address bit 6~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: The WREN, WR, RDEN and RD can not be set to "1" at the same time in one instruction. The WR and RD can not be set to "1" at the same time.

**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 | x | x | x | x | x | x | x | x |

"x" unknown

Bit 7~0 Data EEPROM data
Data EEPROM data bit 7~bit 0

### Reading Data from the EEPROM

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

### Writing Data to the EEPROM

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

### Write Protection

Protection against inadvertent write operation is provided in several ways. After the device is powered-on the Write Enable bit in the control register will be cleared preventing any write operations. Also at power-on the 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. However as the EEPROM is contained within a Multi-function Interrupt, the associated multi-function interrupt enable bit must also be set. When an EEPROM write cycle ends, the DEF request flag and its associated multi-function interrupt request flag will both be set. If the global, EEPROM and Multi-function interrupts are enabled and the stack is not full, a jump to the associated Multi-function Interrupt vector will take place. When the interrupt is serviced only the Multi-function interrupt flag will be automatically reset, the EEPROM interrupt flag must be manually reset by the application program. 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.

## 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/write
CLR BP
MOV A, EEDATA             ; move read data to register
MOV READ_DATA, A
```

### Writing Data to the EEPROM – Polling Method

```
CLR EMI
MOV A, EEPROM_ADRES       ; user defined address
MOV , A
MOV A, EEPROM_DATA        ; user defined data
MOV , 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 IAR1.3                ; set WREN bit, enable write operations
SET IAR1.2                ; Start Write Cycle - set WR bit - executed immediately
                          ; after set WREN bit
SET  EMI
BACK:
SZ  IAR1.2                ; check for write cycle end
JMP BACK
CLR IAR1                  ; disable EEPROM read/write
    BP
```

# Oscillator

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

## Oscillator Overview

The devices include two internal oscillators, a low speed oscillator and high speed oscillator. Both can be chosen as the clock source for the main system clock however the slow speed oscillator is also used as a clock source for other functions such as the Watchdog Timer, Time Base and Timer/Event Counter. Both oscillators require no external components for their implementation. All oscillator options are selected using registers. The high speed oscillator provides higher performance but carries with it the disadvantage of higher power requirements, while the opposite is of course true for the low speed oscillator. With the capability of dynamically switching between fast and slow system clock, the device has the flexibility to optimise the performance/power ratio, a feature especially important in power sensitive portable applications.

| Type | Name | Freq. |
|---|---|---|
| Internal High Speed | HIRC | 8, 12 or 16MHz |
| Internal Low Speed | LIRC | 32kHz |

**Oscillator Types**

## System Clock Configurations

There are two methods of generating the system clock, a high speed internal clock source and low speed internal clock source. The high speed oscillator is an internal 8MHz, 12MHz or 16MHz RC oscillator while the low speed oscillator is an internal 32kHz RC oscillator. Both oscillators are fully integrated and do not require external components. Selecting whether the low or high speed oscillator is used as the system oscillator is implemented using the HLCLK bit and CKS2 ~ CKS0 bits in the SMOD register allowing the system clock to be dynamically selected.

## Internal High Speed RC Oscillator – HIRC

The internal High Speed RC oscillator is a fully integrated system oscillator requiring no external components. The internal RC oscillator has a power on default frequency of 8 MHz but can be selected to be either 8MHz, 12MHz or 16MHz using the HIRCS1 and HIRCS0 bits in the CTRL register. 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.

**CTRL Register**

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

"x" unknown

Bit 7 **RESBF**: Reset pin reset flag
described elsewhere

Bit 6 unimplemented, read as "0"

Bits 5,4 **HIRCS1**, **HIRCS0**: High frequency clock select
00: 8MHz
01: 16MHz
10: 12MHz
11: 8MHz

Bits 3,2 unimplemented, read as "0"

Bits 1,0 Reserved bits, must not be modified.

## Internal Low Speed RC Oscillator – LIRC

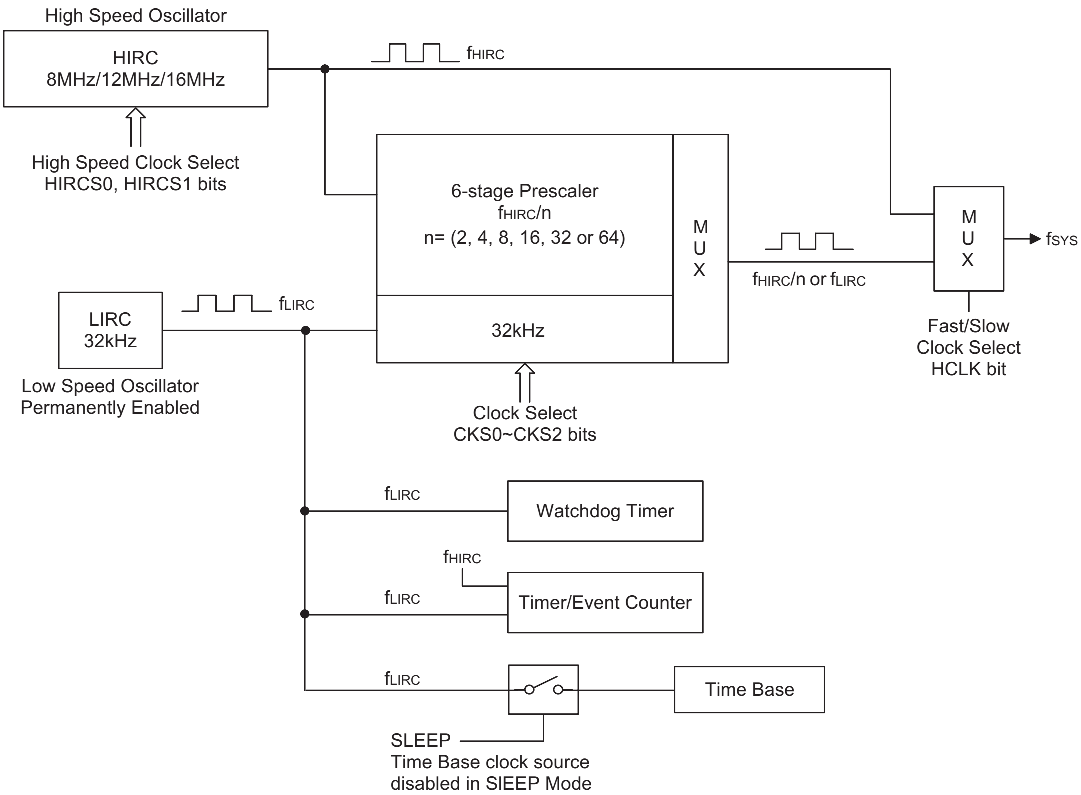
The Internal 32kHz System Oscillator is the low frequency oscillator. It is a fully integrated RC oscillator with a typical frequency of 32kHz at 5V, requiring no external components for its implementation. Device trimming during the manufacturing process and the inclusion of internal frequency compensation circuits are used to ensure that the influence of the power supply voltage, temperature and process variations on the oscillation frequency are minimised. After power on this LIRC oscillator will be permanently enabled; there is no provision to disable the oscillator using register bits.

## 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

The main system clock, can come from either a high frequency, $f_H$, or low frequency, $f_L$, source, and is selected using the HLCLK bit and CKS2~CKS0 bits in the SMOD register. Both the high and low speed system clocks are sourced from internal RC oscillators.




**System Clock Configurations**

### Control Register

A single register, SMOD, is used for overall control of the internal clocks within the device.

**SMOD Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | CKS2 | CKS1 | CKS0 | D4 | LTO | HTO | IDLEN | HLCLK |
| R/W | R/W | R/W | R/W | R/W | R | R | R/W | R/W |
| POR | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |

Bit 7~5 **CKS2~CKS0**: The system clock selection when HLCLK is ″0″
000: $f_L$ ($f_{LIRC}$)
001: $f_L$ ($f_{LIRC}$)
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, which is LIRC, a divided version of the high speed system oscillator can also be chosen as the system clock source.

Bit 4 Undefined bit
This bit can be read or written by user software program.

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.

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. This flag is cleared to ″0″ 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 a wake-up has occurred, the flag will change to a high level after 15~16 clock cycles.

Bit 1 **IDLEN**: IDLE Mode control
0: disable
1: enable
This 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 FSYSON bit is low, the CPU and the system clock will all stop in 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 $f_L$
1: $f_H$
This bit is used to select if the $f_H$ clock or the $f_H$/2 ~ $f_H$/64 or $f_L$ clock is used as the system clock. When the bit is high the $f_H$ clock will be selected and if low the $f_H$/2 ~ $f_H$/64 or $f_L$ clock will be selected. When system clock switches from the $f_H$ clock to the $f_L$ clock and the $f_H$ clock will be automatically switched off to conserve power.

## 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 NORMAL 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 | $f_{SYS}$ | $f_{LIRC}$ | $f_{TBC}$ |
| NORMAL Mode | On | $f_H$~ $f_H$/64 | On | On |
| SLOW Mode | On | $f_L$ | On | On |
| IDLE0 Mode | Off | Off | On | On |
| IDLE1 Mode | Off | On | On | On |
| SLEEP Mode | Off | Off | On | Off |

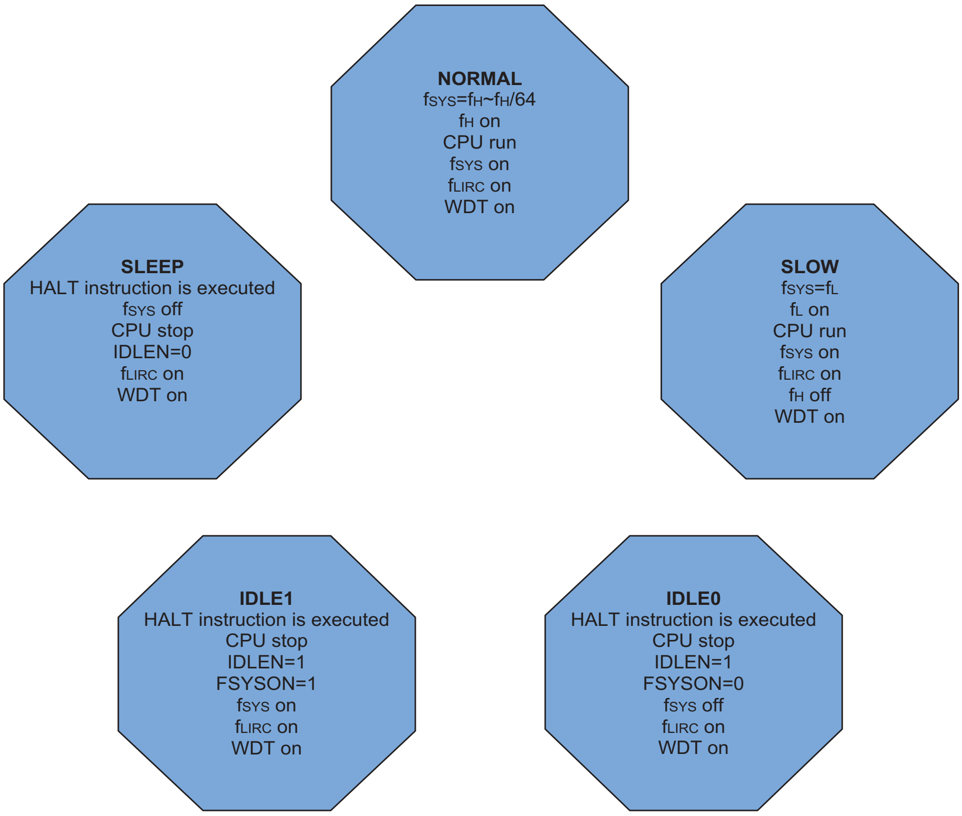
- NORMAL Mode
  As the name suggests this is one of the main operating modes where the microcontroller has all of its functions operational and where the system clock is provided by the high speed oscillator. The high speed oscillator will however first be divided by a ratio ranging from 1 to 64, the 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 the slow speed clock source. Running the microcontroller in this mode allows it to run with much lower operating currents. In the SLOW Mode, the high speed clock is off.

- SLEEP Mode
  The SLEEP Mode is entered when a HALT instruction is executed and when the IDLEN bit in the SMOD register is low. In the SLEEP mode the CPU will be stopped however as the $f_{LIRC}$ oscillator continues to run the Watchdog Timer will continue to operate.

- IDLE0 Mode
  The IDLE0 Mode is entered when a HALT instruction is executed and when the IDLEN bit in the SMOD register is high and the FSYSON bit in the WDTC register is low. In the IDLE0 Mode the system oscillator the system oscillator will be stopped and will therefore be inhibited from driving the CPU.

- IDLE1 Mode
  The IDLE1 Mode is entered when a HALT instruction is executed and when the IDLEN bit in the SMOD register is high and the FSYSON bit in the WDTC 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 system oscillator.

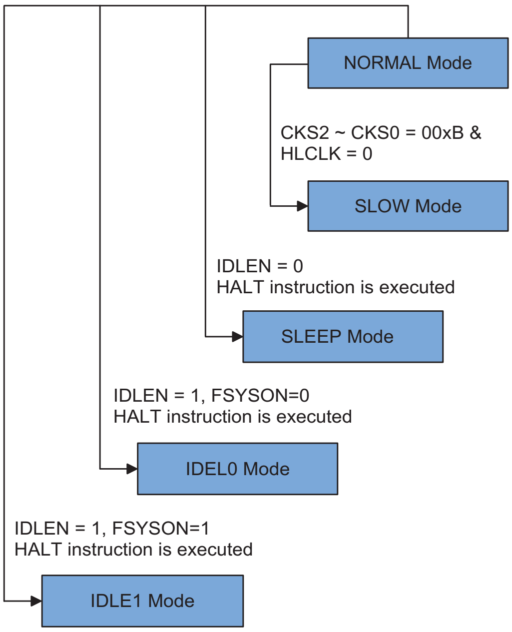


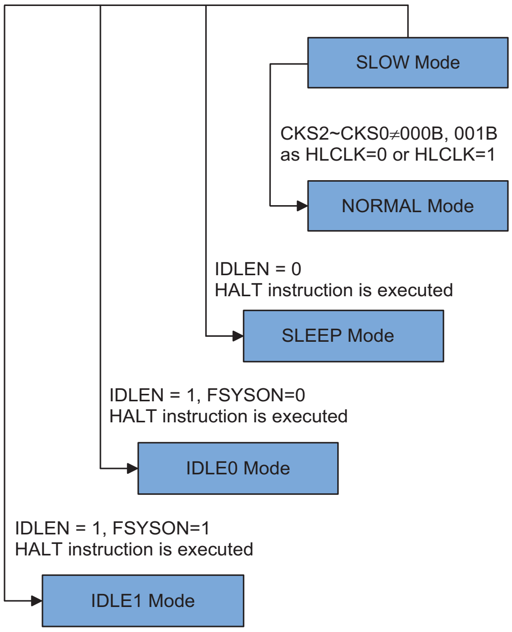

## Operating Mode Switching

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

In simple terms, Mode Switching between the NORMAL Mode and SLOW Mode is executed using the HLCLK bit and CKS2~CKS0 bits in the SMOD register while Mode Switching from the NORMAL/SLOW Modes to the SLEEP/IDLE Modes is executed via the HALT instruction. When a HALT instruction is executed, whether the device enters the IDLE Mode or the SLEEP Mode is determined by the condition of the IDLEN bit in the SMOD register and FSYSON in the WDTC register.

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




### NORMAL Mode to SLOW Mode Switching

When running in the NORMAL Mode, which uses the high speed system oscillator, and therefore consumes more power, the system clock can switch to run in the SLOW Mode by set the HLCLK bit to "0" and set 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 clock is sourced from the LIRC oscillator.

### SLOW Mode to NORMAL Mode Switching

In SLOW Mode the system uses the LIRC low speed system oscillator. To switch back to the NORMAL Mode, where the high speed system oscillator is used, the HLCLK bit should be set to "1" or HLCLK bit is "0", but CKS2~CKS0 is set to "010", "011", "100", "101", "110" or "111". As a certain amount of time will be required for the high frequency clock to stabilise, the status of the HTO bit is checked. The amount of time required for high speed system oscillator stabilization depends upon which high speed system oscillator type is used.

### Entering the SLEEP Mode

There is only one way for the device to enter the SLEEP Mode and that is to execute the "HALT" instruction in the application program with the IDLEN bit in SMOD 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 $f_{LIRC}$ clock will be on.
- The Data Memory contents and registers will maintain their present condition.
- The WDT will be cleared and resume counting.
- The I/O ports will maintain their present conditions.
- In the status register, the Power Down flag, PDF, will be set and the Watchdog time-out flag, TO, will be cleared.

### Entering the IDLE0 Mode

There is only one way for the device to enter the IDLE0 Mode and that is to execute the ″HALT″ instruction in the application program with the IDLEN bit in SMOD register equal to ″1″ and the FSYSON bit in WDTC 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 Time Base clock and $f_{LIRC}$ clock will be on.
- The Data Memory contents and registers will maintain their present condition.
- The WDT will be cleared and resume counting.
- The I/O ports will maintain their present conditions.
- In the status register, the Power Down flag, PDF, will be set and the Watchdog time-out flag, TO, will be cleared.

### Entering the IDLE1 Mode

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

- The system clock and $f_{LIRC}$ clock will be on and the application program will stop at the ″HALT″ instruction.
- The Data Memory contents and registers will maintain their present condition.
- The WDT will be cleared and resume counting.
- The I/O ports will maintain their present conditions.
- In the status register, the Power Down flag, PDF, will be set and the Watchdog time-out flag, TO, will be cleared.

### Standby Current Considerations

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

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

### Wake-up

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

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

If the system is woken up by an external reset, the device will experience a full system reset, however, if the device is woken up by a WDT overflow, a Watchdog Timer reset will be initiated. Although both of these wake-up methods will initiate a reset operation, the actual source of the wake-up can be determined by examining the TO and PDF flags. The PDF flag is cleared by a system power-up or executing the clear Watchdog Timer instructions and is set when executing the "HALT" instruction. 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, the other flags remain in their original status.

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

| 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 Times**

### 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 device is woken up from the SLEEP Mode to the NORMAL Mode, the high speed system oscillator needs an SST period. The device will execute 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 low speed oscillator, $f_{LIRC}$. The Watchdog Timer source clock is then subdivided by a ratio of $2^8$ to $2^{15}$ to give longer timeouts, the actual value being chosen using the WS2~WS0 bits in the WDTC register. The LIRC internal oscillator has an approximate period of 32kHz at a supply voltage of 5V.

However, it should be noted that this specified internal clock period can vary with VDD, temperature and process variations.

### Watchdog Timer Control Register

A single register, WDTC, controls the required timeout period.

**WDTC Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | FSYSON | WS2 | WS1 | WS0 | D3 | D2 | D1 | D0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| POR | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 |

Bit 7 **FSYSON**: $f_{SYS}$ control in IDLE Mode
0: disable
1: enable

Bit 6~4 **WS2, WS1, WS0** : WDT time-out period selection
000: $256/f_{LIRC}$
001: $512/f_{LIRC}$
010: $1024/f_{LIRC}$
011: $2048/f_{LIRC}$
100: $4096/f_{LIRC}$
101: $8192/f_{LIRC}$
110: $16384/f_{LIRC}$
111: $32768/f_{LIRC}$
These three bits determine the division ratio of the Watchdog Timer source clock, which in turn determines the timeout period.

Bit 3~0 Undefined bit
These bits can be read or written by user software program.

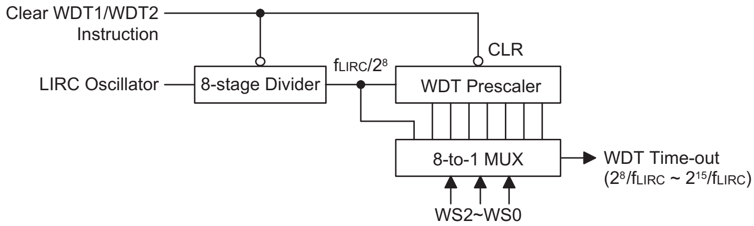
### Watchdog Timer Operation

In these devices the Watchdog Timer supplied by the $f_{LIRC}$ oscillator and is therefore always on. The Watchdog Timer operates by providing a device reset when its timer overflows. This means that in the application program and during normal operation the user has to strategically clear the Watchdog Timer before it overflows to prevent the Watchdog Timer from executing a reset. This is done using the clear watchdog instructions. If the program malfunctions for whatever reason, jumps to an unkown location, or enters an endless loop, these clear instructions will not be executed in the correct manner, in which case the Watchdog Timer will overflow and reset the device.

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 an external hardware reset, which means a low level on the $\overline{RES}$ pin, the second is using the Watchdog Timer software clear instructions and the third is via a HALT instruction.

There Watchdog Timer is cleared using two instructions, CLR WDT1 and CLR WDT2. These instructions must be executed alternately to successfully clear the Watchdog Timer. Note that if CLR WDT1 is used to clear the Watchdog Timer, successive executions of this instruction will have no effect, only the execution of a CLR WDT2 instruction will clear the Watchdog Timer. Similarly after the CLR WDT2 instruction has been executed, only a successive CLR WDT1 instruction can clear the Watchdog Timer. For these devices the single CLR WDT instruction will have no effect so care must be taken not to use this instruction.

The maximum time out period is when the $2^{15}$ division ratio is selected. As an example, with the LIRC oscillator as its source clock, this will give a maximum watchdog period of around 1 second for the $2^{15}$ division ratio, and a minimum timeout of 7.8ms for the $2^8$ division ration.




**Watchdog Timer**

## Reset and Initialisation

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

In addition to the power-on reset, situations may arise where it is necessary to forcefully apply a reset condition when the microcontroller is running. One example of this is where after power has been applied and the microcontroller is already running, the $\overline{\text{RES}}$ line is forcefully pulled low. In such a case, known as a normal operation reset, some of the microcontroller registers remain unchanged allowing the microcontroller to proceed with normal operation after the reset line is allowed to return high.

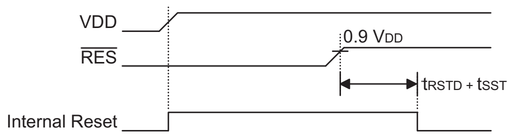
Another type of reset is when the Watchdog Timer overflows and resets the microcontroller. All types of reset operations result in different register conditions being setup. Another reset exists in the form of a Low Voltage Reset, LVR, where a full reset, similar to the $\overline{\text{RES}}$ reset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

### Reset Functions

There are five ways in which a microcontroller reset can occur, through events occurring both internally and externally:

#### 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, typical time=50ms for BS83C24-3, =100ms except BS83C24-3.

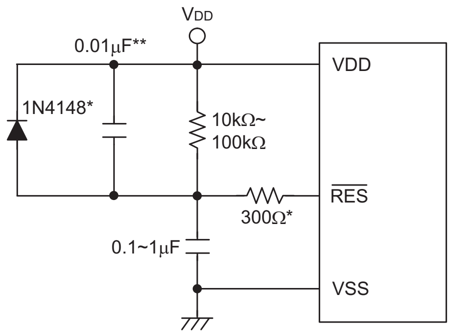
**Power-On Reset Timing Chart**

### $\overline{RES}$ Pin

Although the microcontroller has an internal RC reset function, if the VDD power supply rise time is not fast enough or does not stabilise quickly at power-on, the internal reset function may be incapable of providing proper reset operation. For this reason it is recommended that an external RC network is connected to the $\overline{RES}$ pin, whose additional time delay will ensure that the $\overline{RES}$ pin remains low for an extended period to allow the power supply to stabilise. During this time delay, normal operation of the microcontroller will be inhibited. After the $\overline{RES}$ line reaches a certain voltage value, the reset delay time $t_{RSTD}$ is invoked to provide an extra delay time after which the microcontroller will begin normal operation. The abbreviation SST in the figures stands for System Start-up Timer.

For most applications a resistor connected between VDD and the $\overline{RES}$ pin and a capacitor connected between VSS and the $\overline{RES}$ pin will provide a suitable external reset circuit. Any wiring connected to the $\overline{RES}$ pin should be kept as short as possible to minimise any stray noise interference.

For applications that operate within an environment where more noise is present the Enhanced Reset Circuit shown is recommended.

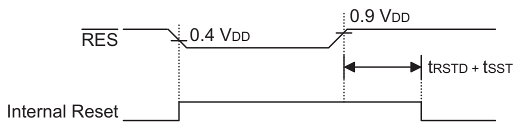



Note: "*" It is recommended that this component is added for added ESD protection
"**" It is recommended that this component is added in environments where power line noise is significant

**External $\overline{RES}$ Circuit**

More information regarding external reset circuits is located in Application Note HA0075E on the Holtek website.

Pulling the $\overline{RES}$ Pin low using external hardware will also execute a device reset. In this case, as in the case of other resets, the Program Counter will reset to zero and program execution initiated from this point.




Note: $t_{RSTD}$ is power-on delay, typical time=50ms for BS83C24-3, =100ms except BS83C24-3.

**$\overline{RES}$ Reset Timing Chart**

The $\overline{RES}$ bit in the CTRL register indicates what kind of reset has occurred. This bit can only be set high by the external reset pin. Any other software reset type will clear the bit to zero. If the application reads this bit and it is high then this indicates that a hardware reset has occurred. After reading the bit it should be cleared to zero by the application program. Note however that after a power-on reset this pin will be in an unknown condition.

**CTRL Register**

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

"x" unknown

Bit 7 **RESBF**: Reset Pin reset flag -- BS83B08-3/B12-3/B16-3
0: no hardware reset occurred
1: hardware reset occured, this bit is cleared to zero by software.

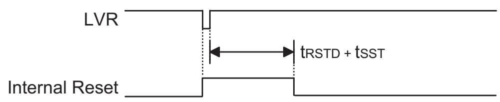
Bit 6 unimplemented, read as "0"

Bits 5,4 **HIRCS1**, **HIRCS0**: High frequency clock select
Described elsewhere

Bits 3~0 unimplemented, read as "0"

### Low Voltage Reset – LVR

The microcontroller contains a low voltage reset circuit in order to monitor the supply voltage of the device, which is selected via a configuration option. If the supply voltage of the device drops to within a range of 0.9V~$V_{LVR}$ such as might occur when changing the battery, the LVR will automatically reset the device internally. The LVR includes the following specifications: For a valid LVR signal, a low voltage, i.e., a voltage in the range between 0.9V~$V_{LVR}$ must exist for greater than the value $t_{LVR}$ specified in the A.C. characteristics. If the low voltage state does not exceed $t_{LVR}$, the LVR will ignore it and will not perform a reset function. One of a range of specified voltage values for $V_{LVR}$ can be selected using configuration options.

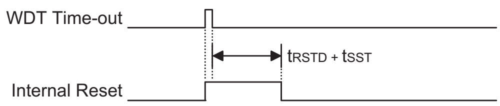



Note: $t_{RSTD}$ is power-on delay, typical time=50ms for BS83C24-3, =100ms except BS83C24-3.

**Low Voltage Reset Timing Chart**

### Watchdog Time-out Reset during Normal Operation

The Watchdog time-out Reset during normal operation is the same as a hardware $\overline{RES}$ pin reset except that the Watchdog time-out flag TO will be set to "1" and RESBF is unchange.



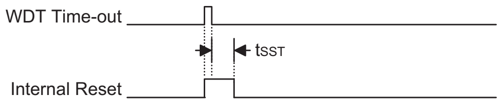

Note: $t_{RSTD}$ is power-on delay, typical time=50ms for BS83C24-3, =100ms except BS83C24-3.

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

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

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

**Note:** The $t_{SST}$ is 15~16 clock cycles if the system clock source is provided by HIRC. The $t_{SST}$ is 1~2 clock for LIRC.




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

## Reset Initial Conditions

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

| TO | PDF | RESET Conditions |
|---|---|---|
| 0 | 0 | Power-on reset |
| u | u | $\overline{\text{RES}}$ or LVR reset during NORMAL or SLOW Mode operation |
| 1 | u | WDT time-out reset during NORMAL or SLOW Mode operation |
| 1 | 1 | WDT time-out reset during IDLE or SLEEP Mode operation |

Note: "u" stands for unchanged

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

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

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

**BS83B08-3 Register**

| Register | Reset (Power-on) | $\overline{RES}$ Reset (Normal Operation) | $\overline{RES}$ Reset (HALT) | WDT Time-out (Normal Operation) | WDT Time-out (IDLE or SLEEP) |
|---|---|---|---|---|---|
| MP0 | xxxx xxxx | uuuu uuuu | uuuu uuuu | uuuu uuuu | uuuu uuuu |
| MP1 | xxxx xxxx | uuuu uuuu | uuuu uuuu | uuuu uuuu | uuuu uuuu |
| BP | ---- ---1 | ---- ---1 | ---- ---1 | ---- ---1 | ---- ---u |
| ACC | xxxx xxxx | uuuu uuuu | uuuu uuuu | uuuu uuuu | uuuu uuuu |
| PCL | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 |
| TBLP | xxxx xxxx | uuuu uuuu | uuuu uuuu | uuuu uuuu | uuuu uuuu |
| TBLH | -xxx xxxx | -uuu uuuu | -uuu uuuu | -uuu uuuu | -uuu uuuu |
| TBHP | ---- -xxx | ---- -uuu | ---- -uuu | ---- -uuu | ---- -uuu |
| STATUS | --00 xxxx | --uu uuuu | --01 uuuu | --1u uuuu | --11 uuuu |
| SMOD | 0000 0011 | 0000 0011 | 0000 0011 | 0000 0011 | uuuu uuuu |
| INTEG | ---- --00 | ---- --00 | ---- --00 | ---- --00 | ---- --uu |
| INTC0 | -000 0000 | -000 0000 | -000 0000 | -000 0000 | -uuu uuuu |
| INTC1 | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| INTC2 | ---0 ---0 | ---0 ---0 | ---0 ---0 | ---0 ---0 | ---u ---u |
| MFI0 | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| PA | ---1 1111 | ---1 1111 | ---1 1111 | ---1 1111 | ---u uuuu |
| PAC | ---1 1111 | ---1 1111 | ---1 1111 | ---1 1111 | ---u uuuu |
| PAPU | ---0 0000 | ---0 0000 | ---0 0000 | ---0 0000 | ---u uuuu |
| PAWU | ---0 0000 | ---0 0000 | ---0 0000 | ---0 0000 | ---u uuuu |
| WDTC | 0111 1010 | 0111 1010 | 0111 1010 | 0111 1010 | uuuu uuuu |
| TBC | --00 ---- | --00 ---- | --00 ---- | --00 ---- | --uu ---- |
| TMR | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TMRC | --00 -000 | --00 -000 | --00 -000 | --00 -000 | --uu -uuu |
| EEA | --00 0000 | --00 0000 | --00 0000 | --00 0000 | --uu uuuu |
| EED | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| PB | 1111 1111 | 1111 1111 | 1111 1111 | 1111 1111 | uuuu uuuu |
| PBC | 1111 1111 | 1111 1111 | 1111 1111 | 1111 1111 | uuuu uuuu |
| PBPU | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| I2CTOC | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| SIMC0 | 1110 000- | 1110 000- | 1110 000- | 1110 000- | uuuu uuu- |
| SIMC1 | 1000 0001 | 1000 0001 | 1000 0001 | 1000 0001 | uuuu uuuu |
| SIMD | xxxx xxxx | xxxx xxxx | xxxx xxxx | xxxx xxxx | uuuu uuuu |

| Register | Reset (Power-on) | $\overline{RES}$ Reset (Normal Operation) | $\overline{RES}$ Reset (HALT) | WDT Time-out (Normal Operation) | WDT Time-out (IDLE or SLEEP) |
|---|---|---|---|---|---|
| SIMA/SIMC2 | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM016DH | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM016DL | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM0C0 | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM0C1 | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM0C2 | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM0C3 | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM116DH | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM116DL | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM1C0 | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM1C1 | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM1C2 | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM1C3 | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| CTRL | x–00 ––00 | 1–00 ––00 | 1–00 ––00 | u–00 ––00 | u–uu ––uu |
| EEC | –––– 0000 | –––– 0000 | –––– 0000 | –––– 0000 | –––– uuuu |

Note: "u" stands for unchanged
"x" stands for unknown
"–" stands for unimplemented

**BS83B12-3 Register**

| Register | Reset (Power-on) | $\overline{RES}$ Reset (Normal Operation) | $\overline{RES}$ Reset (HALT) | WDT Time-out (Normal Operation) | WDT Time-out (IDLE or SLEEP) |
|---|---|---|---|---|---|
| MP0 | xxxx xxxx | uuuu uuuu | uuuu uuuu | uuuu uuuu | uuuu uuuu |
| MP1 | xxxx xxxx | uuuu uuuu | uuuu uuuu | uuuu uuuu | uuuu uuuu |
| BP | – – – – – – – 1 | – – – – – – – 1 | – – – – – – – 1 | – – – – – – – 1 | – – – – – – – u |
| ACC | xxxx xxxx | uuuu uuuu | uuuu uuuu | uuuu uuuu | uuuu uuuu |
| PCL | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 |
| TBLP | xxxx xxxx | uuuu uuuu | uuuu uuuu | uuuu uuuu | uuuu uuuu |
| TBLH | – xxx xxxx | – uuu uuuu | – uuu uuuu | – uuu uuuu | – uuu uuuu |
| TBHP | – – – – – xxx | – – – – – uuu | – – – – – uuu | – – – – – uuu | – – – – – uuu |
| STATUS | – – 00 xxxx | – – uu uuuu | – – 01 uuuu | – – 1u uuuu | – – 11 uuuu |
| SMOD | 0000 0011 | 0000 0011 | 0000 0011 | 0000 0011 | uuuu uuuu |
| INTEG | – – – – – – 00 | – – – – – – 00 | – – – – – – 00 | – – – – – – 00 | – – – – – – uu |
| INTC0 | – 000 0000 | – 000 0000 | – 000 0000 | – 000 0000 | – uuu uuuu |
| INTC1 | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| INTC2 | – 000 – 000 | – 000 – 000 | – 000 – 000 | – 000 – 000 | – uuu – uuu |
| MFI0 | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| MFI1 | – – 00 – – 00 | – – 00 – – 00 | – – 00 – – 00 | – – 00 – – 00 | – – uu – – uu |
| PA | – – – 1 1111 | – – – 1 1111 | – – – 1 1111 | – – – 1 1111 | – – – u uuuu |
| PAC | – – – 1 1111 | – – – 1 1111 | – – – 1 1111 | – – – 1 1111 | – – – u uuuu |
| PAPU | – – – 0 0000 | – – – 0 0000 | – – – 0 0000 | – – – 0 0000 | – – – u uuuu |
| PAWU | – – – 0 0000 | – – – 0 0000 | – – – 0 0000 | – – – 0 0000 | – – – u uuuu |
| WDTC | 0111 1010 | 0111 1010 | 0111 1010 | 0111 1010 | uuuu uuuu |
| TBC | – – 00 – – – – | – – 00 – – – – | – – 00 – – – – | – – 00 – – – – | – – uu – – – – |
| TMR | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TMRC | – – 00 – 000 | – – 00 – 000 | – – 00 – 000 | – – 00 – 000 | – – uu – uuu |
| EEA | – – 00 0000 | – – 00 0000 | – – 00 0000 | – – 00 0000 | – – uu uuuu |
| EED | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| PB | 1111 1111 | 1111 1111 | 1111 1111 | 1111 1111 | uuuu uuuu |
| PBC | 1111 1111 | 1111 1111 | 1111 1111 | 1111 1111 | uuuu uuuu |
| PBPU | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| I2CTOC | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| SIMC0 | 1110 000 – | 1110 000 – | 1110 000 – | 1110 000 – | uuuu uuu – |
| SIMC1 | 1000 0001 | 1000 0001 | 1000 0001 | 1000 0001 | uuuu uuuu |
| SIMD | xxxx xxxx | xxxx xxxx | xxxx xxxx | xxxx xxxx | uuuu uuuu |
| SIMA/SIMC2 | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM016DH | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM016DL | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |

| Register | Reset (Power-on) | $\overline{RES}$ Reset (Normal Operation) | $\overline{RES}$ Reset (HALT) | WDT Time-out (Normal Operation) | WDT Time-out (IDLE or SLEEP) |
|---|---|---|---|---|---|
| TKM0C0 | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM0C1 | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM0C2 | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM0C3 | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM116DH | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM116DL | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM1C0 | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM1C1 | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM1C2 | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM1C3 | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| PC | – – – – 1111 | – – – – 1111 | – – – – 1111 | – – – – 1111 | – – – – uuuu |
| PCC | – – – – 1111 | – – – – 1111 | – – – – 1111 | – – – – 1111 | – – – – uuuu |
| PCPU | – – – – 0000 | – – – – 0000 | – – – – 0000 | – – – – 0000 | – – – – uuuu |
| CTRL | x – 0 0 – – 0 0 | 1 – 0 0 – – 0 0 | 1 – 0 0 – – 0 0 | u – 0 0 – – 0 0 | u – u u – – u u |
| EEC | – – – – 0000 | – – – – 0000 | – – – – 0000 | – – – – 0000 | – – – – uuuu |
| TKM216DH | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM216DL | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM2C0 | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM2C1 | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM2C2 | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM2C3 | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |

Note: "u" stands for unchanged
"x" stands for unknown
"–" stands for unimplemented

**BS83B16-3/BS83B16G-3 Register**

| Register | Reset (Power-on) | $\overline{\text{RES}}$ Reset (Normal Operation) | $\overline{\text{RES}}$ Reset (HALT) | WDT Time-out (Normal Operation) | WDT Time-out (IDLE or SLEEP) |
|---|---|---|---|---|---|
| MP0 | xxxx xxxx | uuuu uuuu | uuuu uuuu | uuuu uuuu | uuuu uuuu |
| MP1 | xxxx xxxx | uuuu uuuu | uuuu uuuu | uuuu uuuu | uuuu uuuu |
| BP | – – – – – – – 1 | – – – – – – – 1 | – – – – – – – 1 | – – – – – – – 1 | – – – – – – – u |
| ACC | xxxx xxxx | uuuu uuuu | uuuu uuuu | uuuu uuuu | uuuu uuuu |
| PCL | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 |
| TBLP | xxxx xxxx | uuuu uuuu | uuuu uuuu | uuuu uuuu | uuuu uuuu |
| TBLH | – xxx xxxx | – uuu uuuu | – uuu uuuu | – uuu uuuu | – uuu uuuu |
| TBHP | – – – – – xxx | – – – – – uuu | – – – – – uuu | – – – – – uuu | – – – – – uuu |
| STATUS | – – 00 xxxx | – – uu uuuu | – – 01 uuuu | – – 1u uuuu | – – 11 uuuu |
| SMOD | 0000 0011 | 0000 0011 | 0000 0011 | 0000 0011 | uuuu uuuu |
| INTEG | – – – – – – 00 | – – – – – – 00 | – – – – – – 00 | – – – – – – 00 | – – – – – – uu |
| INTC0 | – 000 0000 | – 000 0000 | – 000 0000 | – 000 0000 | – uuu uuuu |
| INTC1 | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| INTC2 | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| MFI0 | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| MFI1 | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| PA | – – – 1 1111 | – – – 1 1111 | – – – 1 1111 | – – – 1 1111 | – – – u uuuu |
| PAC | – – – 1 1111 | – – – 1 1111 | – – – 1 1111 | – – – 1 1111 | – – – u uuuu |
| PAPU | – – – 0 0000 | – – – 0 0000 | – – – 0 0000 | – – – 0 0000 | – – – u uuuu |
| PAWU | – – – 0 0000 | – – – 0 0000 | – – – 0 0000 | – – – 0 0000 | – – – u uuuu |
| WDTC | 0111 1010 | 0111 1010 | 0111 1010 | 0111 1010 | uuuu uuuu |
| TBC | – – 00 – – – – | – – 00 – – – – | – – 00 – – – – | – – 00 – – – – | – – uu – – – – |
| TMR | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TMRC | – – 00 – 000 | – – 00 – 000 | – – 00 – 000 | – – 00 – 000 | – – uu – uuu |
| EEA | – – 00 0000 | – – 00 0000 | – – 00 0000 | – – 00 0000 | – – uu uuuu |
| EED | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| PB | 1111 1111 | 1111 1111 | 1111 1111 | 1111 1111 | uuuu uuuu |
| PBC | 1111 1111 | 1111 1111 | 1111 1111 | 1111 1111 | uuuu uuuu |
| PBPU | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| I2CTOC | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| SIMC0 | 1110 000 – | 1110 000 – | 1110 000 – | 1110 000 – | uuuu uuu – |
| SIMC1 | 1000 0001 | 1000 0001 | 1000 0001 | 1000 0001 | uuuu uuuu |
| SIMD | xxxx xxxx | xxxx xxxx | xxxx xxxx | xxxx xxxx | uuuu uuuu |
| SIMA/SIMC2 | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM016DH | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM016DL | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |

| Register | Reset (Power-on) | $\overline{RES}$ Reset (Normal Operation) | $\overline{RES}$ Reset (HALT) | WDT Time-out (Normal Operation) | WDT Time-out (IDLE or SLEEP) |
|---|---|---|---|---|---|
| TKM0C0 | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM0C1 | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM0C2 | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM0C3 | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM116DH | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM116DL | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM1C0 | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM1C1 | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM1C2 | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM1C3 | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| PC | 1111 1111 | 1111 1111 | 1111 1111 | 1111 1111 | uuuu uuuu |
| PCC | 1111 1111 | 1111 1111 | 1111 1111 | 1111 1111 | uuuu uuuu |
| PCPU | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| CTRL | x–00 ––00 | 1–00 ––00 | 1–00 ––00 | u–00 ––00 | u–uu ––uu |
| EEC | –––– 0000 | –––– 0000 | –––– 0000 | –––– 0000 | –––– uuuu |
| TKM216DH | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM216DL | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM2C0 | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM2C1 | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM2C2 | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM2C3 | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM316DH | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM316DL | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM3C0 | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM3C1 | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM3C2 | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM3C3 | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |

Note: "u" stands for unchanged
"x" stands for unknown
"–" stands for unimplemented

**BS83C24-3 Register**

| Register | Reset (Power-on) | WDT Time-out (Normal Operation) | WDT Time-out (HALT) * |
|---|---|---|---|
| IAR0 | 0000 0000 | 0000 0000 | uuuu uuuu |
| MP0 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| IAR1 | 0000 0000 | 0000 0000 | uuuu uuuu |
| MP1 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| BP | – – – – – – 0 0 | – – – – – – 0 0 | – – – – – – u u |
| ACC | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| PCL | 0000 0000 | 0000 0000 | 0000 0000 |
| TBLP | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TBLH | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TBHP | – – – – xxxx | – – – – uuuu | – – – – uuuu |
| STATUS | – – 0 0 xxxx | – – 1 u uuuu | – – 1 1 uuuu |
| SMOD | 0000 0011 | 0000 0011 | uuuu uuuu |
| INTC3 | 0000 0000 | 0000 0000 | uuuu uuuu |
| INTEG | – – – – – – 0 0 | – – – – – – 0 0 | – – – – – – u u |
| INTC0 | – 000 0000 | – 000 0000 | – uuu uuuu |
| INTC1 | 0000 0000 | 0000 0000 | uuuu uuuu |
| INTC2 | 0000 0000 | 0000 0000 | uuuu uuuu |
| MFI0 | 0000 0000 | 0000 0000 | uuuu uuuu |
| MFI1 | 0000 0000 | 0000 0000 | uuuu uuuu |
| MFI2 | 0000 0000 | 0000 0000 | uuuu uuuu |
| PA | – – – 1 1111 | – – – 1 1111 | – – – u uuuu |
| PAC | – – – 1 1111 | – – – 1 1111 | – – – u uuuu |
| PAPU | – – – 0 0000 | – – – 0 0000 | – – – u uuuu |
| PAWU | – – – 0 0000 | – – – 0 0000 | – – – u uuuu |
| WDTC | 0111 1010 | 0111 1010 | uuuu uuuu |
| TBC | – – 0 0 – – – – | – – 0 0 – – – – | – – u u – – – – |
| TMR0 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TMR0C | – – 0 0 – 000 | – – 0 0 – 000 | – – u u – uuu |
| EEA | – 000 0000 | – 000 0000 | – uuu uuuu |
| EED | 0000 0000 | 0000 0000 | uuuu uuuu |
| PB | 1111 1111 | 1111 1111 | uuuu uuuu |
| PBC | 1111 1111 | 1111 1111 | uuuu uuuu |
| PBPU | 0000 0000 | 0000 0000 | uuuu uuuu |
| I2CTOC | 0000 0000 | 0000 0000 | uuuu uuuu |
| SIMC0 | 1110 000 – | 1110 000 – | uuuu uuu – |
| SIMC1 | 1000 0001 | 1000 0001 | uuuu uuuu |

| Register | Reset (Power-on) | WDT Time-out (Normal Operation) | WDT Time-out (HALT) * |
|---|---|---|---|
| SIMD | xxxx xxxx | xxxx xxxx | uuuu uuuu |
| SIMA/SIMC2 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM016DH | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM016DL | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM0C0 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM0C1 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM0C2 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM0C3 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM116DH | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM116DL | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM1C0 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM1C1 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM1C2 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM1C3 | 0000 0000 | 0000 0000 | uuuu uuuu |
| PC | 1111 1111 | 1111 1111 | uuuu uuuu |
| PCC | 1111 1111 | 1111 1111 | uuuu uuuu |
| PCPU | 0000 0000 | 0000 0000 | uuuu uuuu |
| CTRL | x–00 ––00 | u–00 ––00 | u–uu ––uu |
| PD | 1111 1111 | 1111 1111 | uuuu uuuu |
| PDC | 1111 1111 | 1111 1111 | uuuu uuuu |
| PDPU | 0000 0000 | 0000 0000 | uuuu uuuu |
| PE | 1111 1111 | 1111 1111 | uuuu uuuu |
| PEC | 1111 1111 | 1111 1111 | uuuu uuuu |
| PEPU | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM216DH | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM216DL | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM2C0 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM2C1 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM2C2 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM2C3 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM316DH | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM316DL | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM3C0 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM3C1 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM3C2 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM3C3 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM416DH | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM416DL | 0000 0000 | 0000 0000 | uuuu uuuu |

| Register | Reset (Power-on) | WDT Time-out (Normal Operation) | WDT Time-out (HALT) * |
|---|---|---|---|
| TKM4C0 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM4C1 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM4C2 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM4C3 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM516DH | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM516DL | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM5C0 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM5C1 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM5C2 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TKM5C3 | 0000 0000 | 0000 0000 | uuuu uuuu |
| PF | ---- 1111 | ---- 1111 | ---- uuuu |
| PFC | ---- 1111 | ---- 1111 | ---- uuuu |
| PFPU | ---- 0000 | ---- 0000 | ---- uuuu |
| TMR1H | 0000 0000 | 0000 0000 | uuuu uuuu |
| TMR1L | 0000 0000 | 0000 0000 | uuuu uuuu |
| TMR1C | 0000 10-- | 0000 10-- | uuuu uu-- |
| EEC | ---- 0000 | ---- 0000 | ---- uuuu |

Note: "*" stands for warm reset
"u" stands for unchanged
"x" stands for unknown
"–" stands for unimplemented

## Input/Output Ports

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

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

### I/O Register List

**BS83B08-3**

| Register Name | Bit | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| PAWU | — | — | — | D4 | D3 | D2 | D1 | D0 |
| PAPU | — | — | — | D4 | D3 | D2 | D1 | D0 |
| PA | — | — | — | D4 | D3 | D2 | D1 | D0 |
| PAC | — | — | — | D4 | D3 | D2 | D1 | D0 |
| PBPU | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| PB | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| PBC | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |

**BS83B12-3**

| Register Name | Bit | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| PAWU | — | — | — | D4 | D3 | D2 | D1 | D0 |
| PAPU | — | — | — | D4 | D3 | D2 | D1 | D0 |
| PA | — | — | — | D4 | D3 | D2 | D1 | D0 |
| PAC | — | — | — | D4 | D3 | D2 | D1 | D0 |
| PBPU | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| PB | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| PBC | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| PCPU | — | — | — | — | D3 | D2 | D1 | D0 |
| PC | — | — | — | — | D3 | D2 | D1 | D0 |
| PCC | — | — | — | — | D3 | D2 | D1 | D0 |

**BS83B16-3/BS83B16G-3**

| Register Name | Bit | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| PAWU | — | — | — | D4 | D3 | D2 | D1 | D0 |
| PAPU | — | — | — | D4 | D3 | D2 | D1 | D0 |
| PA | — | — | — | D4 | D3 | D2 | D1 | D0 |
| PAC | — | — | — | D4 | D3 | D2 | D1 | D0 |
| PBPU | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| PB | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| PBC | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| PCPU | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| PC | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| PCC | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |

**BS83C24-3**

| Register Name | Bit | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| PAWU | — | — | — | D4 | D3 | D2 | D1 | D0 |
| PAPU | — | — | — | D4 | D3 | D2 | D1 | D0 |
| PA | — | — | — | D4 | D3 | D2 | D1 | D0 |
| PAC | — | — | — | D4 | D3 | D2 | D1 | D0 |
| PBPU | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| PB | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| PBC | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| PCPU | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| PC | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| PCC | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| PDPU | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| PD | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| PDC | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| PEPU | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| PE | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| PEC | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| PFPU | — | — | — | — | D3 | D2 | D1 | D0 |
| PF | — | — | — | — | D3 | D2 | D1 | D0 |
| PFC | — | — | — | — | D3 | D2 | D1 | D0 |

## 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 register PAPU~PFPU, and are implemented using weak PMOS transistors.

### PAPU Register

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

Bit 7~5 unimplemented, read as "0"

Bit 4~0 **PAPU**: Port A bit 4~bit 0 pull-high control
0: disable
1: enable

### PBPU 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 PBPU Port B bit 7~bit 0 pull-high control
0: disable
1: enable

### PCPU Register

- BS83B12-3

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

Bit 7~4 unimplemented, read as "0"

Bit 3~0 **PCPU**: Port C bit 3~bit 0 pull-high control
0: disable
1: enable

- BS83B16-3/BS83B16G-3/BS83C24-3

| 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 **PCPU**: Port C bit 7~bit 0 pull-high control
0: disable
1: enable

**PDPU, PEPU Register**

- BS83C24-3

| 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 **PDPU, PEPU**: Port D~Port E bit 7~bit 0 pull-high control
0: disable
1: enable

**PFPU Register**

- BS83C24-3

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

Bit 7~4 unimplemented, read as "0"
Bit 3~0 **PFPU**: Port F bit 3~bit 0 pull-high control
0: disable
1: enable

## 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 | — | — | — | D4 | D3 | D2 | D1 | D0 |
| R/W | — | — | — | R/W | R/W | R/W | R/W | R/W |
| POR | — | — | — | 0 | 0 | 0 | 0 | 0 |

Bit 7~5 unimplemented, read as "0"
Bit 4~0 PAWU: Port A bit 4~bit 0 wake-up control
0: disable
1: enable

## I/O Port Control Register

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

**PAC Register**

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

Bit 7~5 unimplemented, read as ″0″

Bit 4~0 I/O port bit 4~bit 0 input/output control
0: output
1: input

**PBC 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 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

Bit 7~0 I/O port bit 7 ~ bit 0 input/output control
0: output
1: input

**PCC Register**

- BS83B12-3

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

Bit 7~4 unimplemented, read as "0"

Bit 3~0 **PCC**: Port C bit 3~bit 0 input/output control
0: output
1: input

- BS83B16-3/BS83B16G-3/BS83C24-3

| 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 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

Bit 7~0 **PCC**: Port C bit 7~bit 0 input/output control
0: output
1: input

**PDC Register**

- BS83C24-3

| 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 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

Bit 7~0 **PDC**: Port D bit 7~bit 0 input/output control
0: output
1: input

**PEC Register**

- BS83C24-3

| 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 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

Bit 7~0 **PEC**: Port E bit 7~bit 0 input/output control
0: output
1: input

**PFC Register**

- BS83C24-3

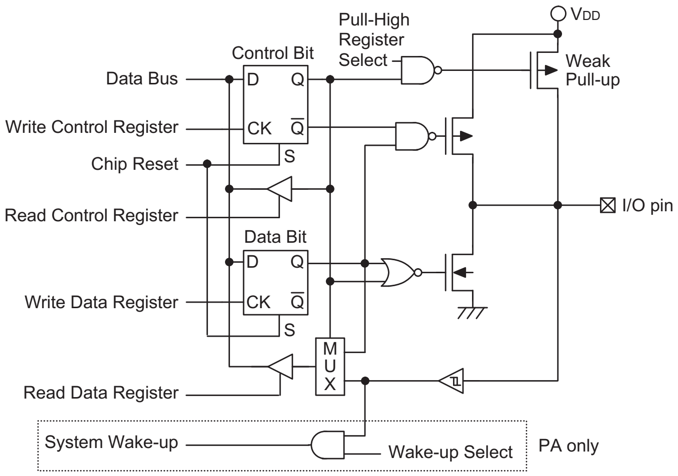
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | — | — | — | — | D3 | D2 | D1 | D0 |
| R/W | — | — | — | — | R/W | R/W | R/W | R/W |
| POR | — | — | — | — | 1 | 1 | 1 | 1 |

Bit 7~4 unimplemented, read as ″0″

Bit 3~0 **PFC**: Port F bit 3~bit 0 input/output control
0: output
1: input

## I/O Pin Structures

The accompanying diagrams illustrate the internal structures of some generic I/O pin types. As the exact logical construction of the I/O pin will differ from these drawings, they are supplied as a guide only to assist with the functional understanding of the I/O pins. The wide range of pin-shared structures does not permit all types to be shown.




**Generic Input/Output Structure**

## Programming Considerations

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

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

## Timer/Event Counters

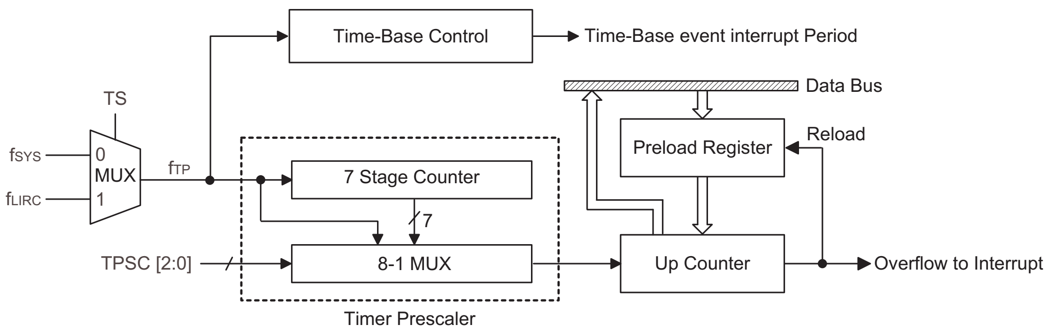
The provision of timers form an important part of any microcontroller, giving the designer a means of carrying out time related functions. The devices contain one 8-bit and one 16-bit timers. The 8-bit timer is a general timer. As the 16-bit timer has three different operating modes, it can be configured to operate as a general timer, an external event counter or as a pulse width capture device. The provision of an internal prescaler to the clock circuitry on gives added range to the timers.

There are two types of registers related to the Timer/Event Counters. The first is the register that contains the actual value of the timer and into which an initial value can be preloaded. Reading from this register retrieves the contents of the Timer/Event Counter. The second type of associated register is the Timer Control Register which defines the timer options and determines how the timer is to be used. The device can have the timer clock configured to come from the internal clock source. In addition, the timer clock source can also be configured to come from an external timer pin.

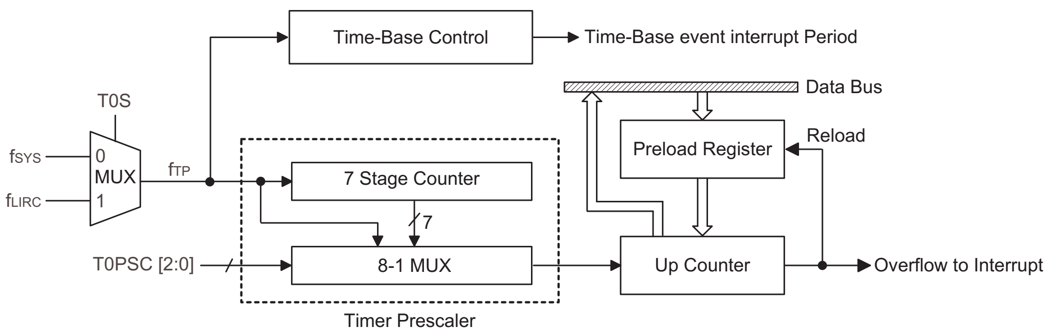
The accompanying table illustrates the Timer Type list for the devices.

| Device | Timer Type | Timer Register Name | Timer Control Register Name | Time Operating Modes |
|---|---|---|---|---|
| BS83B08-3<br>BS83B12-3<br>BS83B16-3<br>BS83B16G-3 | 8-bit | TMR | TMRC | Timer Mode |
| BS83C24-3 | 8-bit | TMR0 | TMRC0 | Timer Mode |
| | 16-bit | TMR1L/TMR1H | TMRC1 | Timer Mode<br>Event Counter Mode<br>Pulse Width Capture Mode |

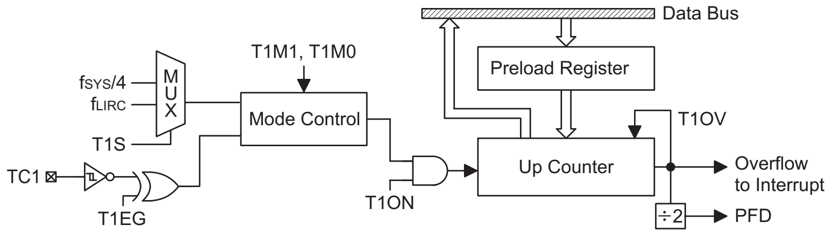
**Timer Type Summary Table**




**8-bit Timer/Event Counter**




**8-bit Timer/Event Counter 0**




**16-bit Timer/Event Counter 1**

## Configuring the Timer/Event Counter Input Clock Source

The Timer/Event Counter clock source can originate from various sources, an internal clock or an external pin. The internal clock source is used when the timer is in the timer mode or in the pulse width capture mode. For some Timer/Event Counters, this internal clock source may be first divided by a prescaler, the division ratio of which is conditioned by the Timer Control Register bits. An external clock source is used when the timer is in the event counting mode, the clock source being provided on an external timer pin TC1. Depending upon the condition of the T1EG bit, each high to low, or low to high transition on the external timer pin will increment the counter by one.

## Timer Register – TMR, TMR0, TMR1L, TMR1H

The timer registers are special function registers located in the Special Purpose Data Memory and is the place where the actual timer value is stored. These registers are known as TMR, TMR0, TMR1L and TMR1H. The value in the timer registers increases by one each time an internal clock pulse is received or an external transition occurs on the external timer pin. The timer will count from the initial value loaded by the preload register to the full count of FFH for the 8-bit Timer/Event Counter or FFFFH for the 16-bit Timer/Event Counters, at which point the timer overflows and an internal interrupt signal is generated. The timer value will then be reset with the initial preload register value and continue counting. Note that to achieve a maximum full range count of FFH or FFFFH, the preload register must first be cleared to all zeros. It should be noted that after power-on, the preload registers will be in an unknown condition.

Note that if the Timer/Event Counter is in an OFF condition and data is written to its preload register, this data will be immediately written into the actual counter. However, if the counter is enabled and counting, any new data written into the preload data register during this period will remain in the preload register and will only be written into the actual counter the next time an overflow occurs.

## Timer Control Register – TMRC, TMR0C, TMR1C

The flexible features of the Holtek microcontroller Timer/Event Counters enable them to operate in three different modes, the options of which are determined by the contents of their respective control register.

The Timer Control Register is known as TMRC, TMR0C and TMR1C. The TMRC or TMR0C is used to control the 8-bit Timer while the TMR1C are used for 16-bit Timer. It is the Timer Control Register together with its corresponding timer register that control the full operation of the Timer/Event Counter. Before the timer can be used, it is essential that the Timer Control Register is fully programmed with the right data to ensure its correct operation, a process that is normally carried out during program initialisation.

The timer-on bit, which is bit 4 of the Timer Control Register and known as TON, T0ON or T1ON bit, provides the basic on/off control of the respective timer. Setting the bit high allows the counter to run, clearing the bit stops the counter. Bits 0~2 of the TMRC or TMR0C registers determine the division ratio of the input clock prescaler. In addition, the TS, T0S and T1S bits select the internal clock source.

The 16-bit timer, TMR1, can operate in three different modes. To choose which of the three modes the timer is to operate in, either in the timer mode, the event counting mode or the pulse width capture mode, bits 7 and 6 of the Timer Control Register, which are known as the bit pair T1M1/T1M0, must be set to the required logic levels. If the TMR1 is in the event count or pulse width capture mode, the active transition edge level type is selected by the logic level of bit 3 of the Timer Control Register which is known as T1EG.

**TMR0C Register**

- BS83B08-3/B12-3/B16-3/B16G-3

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | — | — | TS | TON | — | TPSC2 | TPSC1 | TPSC0 |
| R/W | — | — | R/W | R/W | — | R/W | R/W | R/W |
| POR | — | — | 0 | 0 | — | 0 | 0 | 0 |

Bits 7, 6 unimplemented, read as "0"

Bit 5 **TS**: Timer/Event Counter Clock Source
0: $f_{SYS}$
1: $f_{LIRC}$

Bit 4 **TON**: Timer/Event Counter Counting Enable
0: disable
1: enable

Bit 3 unimplemented, read as "0"

Bits 2~0 **TPSC2~TPSC0**: Timer prescaler rate selection
Timer internal clock=
000: $f_{TP}$
001: $f_{TP}/2$
010: $f_{TP}/4$
011: $f_{TP}/8$
100: $f_{TP}/16$
101: $f_{TP}/32$
110: $f_{TP}/64$
111: $f_{TP}/128$

- BS83C24-3

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | — | — | T0S | T0ON | — | T0PSC2 | T0PSC1 | T0PSC0 |
| R/W | — | — | R/W | R/W | — | R/W | R/W | R/W |
| POR | — | — | 0 | 0 | — | 0 | 0 | 0 |

Bits 7, 6 unimplemented, read as "0"

Bit 5 **T0S**: Timer/Event Counter Clock Source
0: $f_{SYS}$
1: $f_{LIRC}$

Bit 4 **T0ON**: Timer/Event Counter Counting Enable
0: disable
1: enable

Bit 3 unimplemented, read as "0"

Bits 2~0 **T0PSC2~T0PSC0**: Timer prescaler rate selection
Timer internal clock=
000: $f_{TP}$
001: $f_{TP}/2$
010: $f_{TP}/4$
011: $f_{TP}/8$
100: $f_{TP}/16$
101: $f_{TP}/32$
110: $f_{TP}/64$
111: $f_{TP}/128$

### TMR1C Register

• BS83C24-3

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | T1M1 | T1M0 | T1S | T1ON | T1EG | PFDC | — | — |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | — | — |
| POR | 0 | 0 | 0 | 0 | 1 | 0 | — | — |

Bits 7, 6 T1M1, T1M0: Timer 1 operation mode selection
00: no mode available
01: event counter mode
10: timer mode
11: pulse width capture mode

Bit 5 **T1S**: timer clock source
0: $f_{SYS}/4$
1: LIRC oscillator

Bit 4 **T1ON**: timer/event counter counting enable
0: disable
1: enable

Bit 3 T1EG:
Event counter active edge selection
0: count on rising edge
1: count on falling edge
Pulse Width Capture active edge selection
0: start counting on falling edge, stop on rising edge
1: start counting on raising edge, stop on falling edge

Bit 2 PFDC: I/O or PFD selection Bit
0: I/O
1: PFD

Bits 1, 0 unimplemented, read as ″0″

## 8-Bit Timer/Event Counter Operating Mode

The Timer/Event Counter can be utilised to measure fixed time intervals, providing an internal interrupt signal each time the Timer/Event Counter overflows. The internal clock is used as the timer clock. The timer input clock source is either the $f_{SYS}/4$ or the LIRC oscillator. However, this timer clock source is further divided by a prescaler, the value of which is determined by the bits TPSC2~TPSC0 or T0PSC2~T0PSC0 in the Timer Control Register. The timer-on bit, TON or T0ON, must be set high to enable the timer to run. Each time an internal clock high to low transition occurs, the timer increments by one; when the timer is full and overflows, an interrupt signal is generated and the timer will reload the value already loaded into the preload register and continue counting. A timer overflow condition and corresponding internal interrupt is one of the wake-up sources, however, the internal interrupts can be disabled by ensuring that the ET0I bits of the INTC1 register are reset to zero.

## 16-Bit Timer/Event Counter 1 Operating Modes -- BS83C24-3

The 16-bit timer has three different operating modes, it can be configured to operate as a general timer, an external event counter or as a pulse width capture device via the T1M1 and T1M0 bits in the TMR1C register.

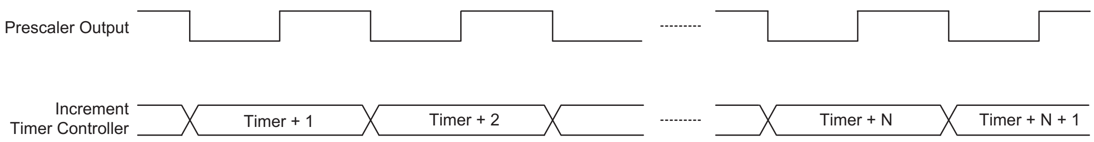
### Timer Mode

In this mode, the Timer/Event Counter can be utilised to measure fixed time intervals, providing an internal interrupt signal each time the Timer/Event Counter overflows. To operate in this mode, the Operating Mode Select bit pair, T1M1/T1M0, in the Timer Control Register must be set to the correct value as shown.

Control Register Operating Mode
Select Bits for the Timer Mode

| Bit7 | Bit6 |
|---|---|
| 1 | 0 |

In this mode the internal clock is used as the timer clock. The timer input clock source is either the $f_{SYS}/4$ or the LIRC oscillator. The timer-on bit, T1ON must be set high to enable the timer to run. Each time an internal clock high to low transition occurs, the timer increments by one; when the timer is full and overflows, an interrupt signal is generated and the timer will reload the value already loaded into the preload register and continue counting. A timer overflow condition and corresponding internal interrupt is one of the wake-up sources, however, the internal interrupts can be disabled by ensuring that the ET1I bits of the INTC3 register are reset to zero.




**Time Mode Timing Chart**

### Event Counter Mode

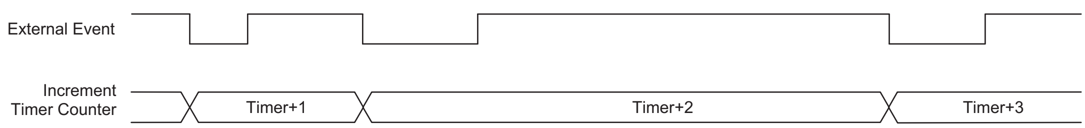
In this mode, a number of externally changing logic events, occurring on the external timer TC1 pin, can be recorded by the Timer/Event Counter. To operate in this mode, the Operating Mode Select bit pair, T1M1/T1M0, in the Timer Control Register must be set to the correct value as shown.

Control Register Operating Mode
Select Bits for the Event Counter Mode

| Bit7 | Bit6 |
|---|---|
| 0 | 1 |

In this mode, the external timer TC1 pin is used as the Timer/Event Counter clock source. After the other bits in the Timer Control Register have been setup, the enable bit T1ON, which is bit 4 of the Timer Control Register, can be set high to enable the Timer/Event Counter to run. If the Active Edge Select bit, T1EG, which is bit 3 of the Timer Control Register, is low, the Timer/Event Counter will increment each time the external timer pin receives a low to high transition. If the T1EG is high, the counter will increment each time the external timer pin receives a high to low transition. When it is full and overflows, an interrupt signal is generated and the Timer/Event Counter will reload the value already loaded into the preload register and continue counting. The interrupt can be disabled by ensuring that the Timer/Event Counter Interrupt Enable bit in the corresponding Interrupt Control Register is reset to zero.

As the external timer pin is shared with an I/O pin, to ensure that the pin is configured to operate as an event counter input pin, two things have to happen. The first is to ensure that the Operating Mode Select bits in the Timer Control Register place the Timer/Event Counter in the Event Counting Mode, the second is to ensure that the port control register configures the pin as an input. It should be noted that in the event counting mode, even if the microcontroller is in the Idle/Sleep Mode, the Timer/Event Counter will continue to record externally changing logic events on the timer input TC1 pin. As a result when the timer overflows it will generate a timer interrupt and corresponding wake-up source.




**Event Counter Mode Timing Chart (T1EG=1)**

#### Pulse Width Capture Mode

In this mode, the Timer/Event Counter can be utilised to measure the width of external pulses applied to the external timer pin. To operate in this mode, the Operating Mode Select bit pair, T1M1/T1M0, in the Timer Control Register must be set to the correct value as shown.

Control Register Operating Mode
Select Bits for the Pulse Width Capture Mode

| Bit7 | Bit6 |
|---|---|
| 1 | 1 |

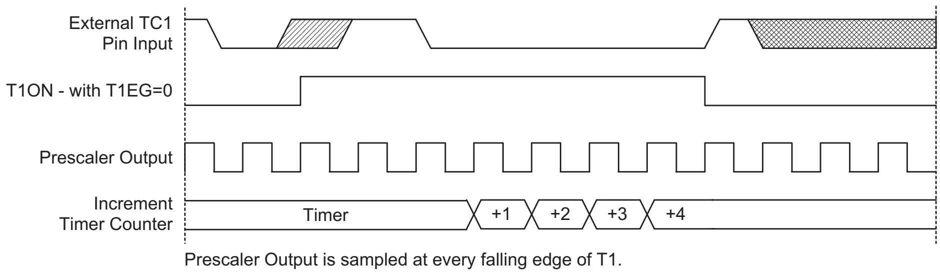
In this mode the internal clock, $f_{SYS}/4$ or the LIRC, is used as the internal clock for the 16-bit Timer/Event Counter. After the other bits in the Timer Control Register have been setup, the enable bit T1ON, which is bit 4 of the Timer Control Register, can be set high to enable the Timer/Event Counter, however it will not actually start counting until an active edge is received on the external timer pin.

If the Active Edge Select bit T1EG, which is bit 3 of the Timer Control Register, is low, once a high to low transition has been received on the external timer pin, the Timer/Event Counter will start counting until the external timer pin returns to its original high level. At this point the enable bit will be automatically reset to zero and the Timer/Event Counter will stop counting. If the Active Edge Select bit is high, the Timer/Event Counter will begin counting once a low to high transition has been received on the external timer pin and stop counting when the external timer pin returns to its original low level. As before, the enable bit will be automatically reset to zero and the Timer/Event Counter will stop counting. It is important to note that in the pulse width capture Mode, the enable bit is automatically reset to zero when the external control signal on the external timer pin returns to its original level, whereas in the other two modes the enable bit can only be reset to zero under program control.

The residual value in the Timer/Event Counter, which can now be read by the program, therefore represents the length of the pulse received on the TC1 pin. As the enable bit has now been reset, any further transitions on the external timer pin will be ignored. The timer cannot begin further pulse width capture until the enable bit is set high again by the program. In this way, single shot pulse measurements can be easily made.

It should be noted that in this mode the Timer/Event Counter is controlled by logical transitions on the external timer pin and not by the logic level. When the Timer/Event Counter is full and overflows, an interrupt signal is generated and the Timer/Event Counter will reload the value already loaded into the preload register and continue counting. The interrupt can be disabled by ensuring that the Timer/Event Counter Interrupt Enable bit in the corresponding Interrupt Control Register is reset to zero.

As the TC1 pin is shared with an I/O pin, to ensure that the pin is configured to operate as a pulse width capture pin, two things have to happen. The first is to ensure that the Operating Mode Select bits in the Timer Control Register place the Timer/Event Counter in the pulse width capture mode, the second is to ensure that the port control register configures the pin as an input.




**Pulse Width Capture Mode Timing Chart (T1EG=0)**

### Prescaler

Bits T0PSC0~T0PSC2 of the TMR0C or TPSC0~TPSC2 of the TMRC register can be used to define a division ratio for the internal clock source of the 8-bit Timer/Event Counter enabling longer time out periods to be setup.

### PFD Function

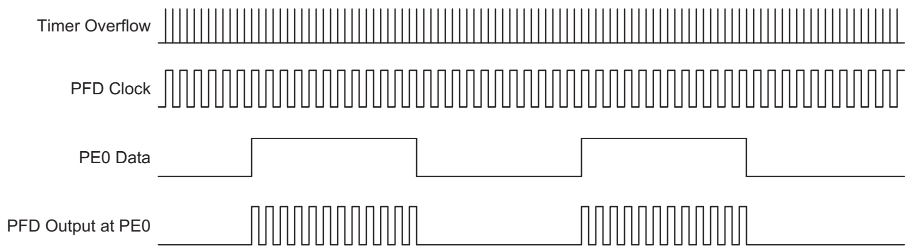
The Programmable Frequency Divider provides a means of producing a variable frequency output suitable for applications, such as piezo-buzzer driving or other interfaces requiring a precise frequency generator.

As the pins are shared with I/O pins, the function is selected using the TMR1C register.

The Timer/Event Counter 1 overflow signal is the clock source for the PFD function. The output frequency is controlled by loading the required values into the timer registers to give the required division ratio. The counter will begin to count-up from this preload register value until full, at which point an overflow signal is generated, causing the PFD outputs to change state. The counter will then be automatically reloaded with the preload register value and continue counting-up.

If the TMR1C register has selected the PFD function, it is essential for the Port E control register PEC, to setup the PFD pin as output. The bit PE0 must be set high to activate the PFD. The output data bit can be used as the on/off control bit for the PFD outputs. Note that the PFD output will all be low if the output data bit is cleared to zero.

Using this method of frequency generation, and if a crystal oscillator is used for the system clock, very precise values of frequency can be generated.




**PFD Function**

### I/O Interfacing

The Timer/Event Counter, when configured to run in the event counter or pulse width capture mode, requires the use of an external timer pin for its operation. As this pin is a shared pin it must be configured correctly to ensure that it is setup for use as a Timer/Event Counter input pin. This is achieved by ensuring that the mode select bits in the Timer/Event Counter control register select either the event counter or pulse width capture mode. Additionally the corresponding Port Control Register bit must be set high to ensure that the pin is setup as an input. Any pull-high resistor connected to this pin will remain valid even if the pin is used as a Timer/Event Counter input.

### Programming Considerations

When configured to run in the timer mode, the internal system clock is used as the timer clock source and is therefore synchronised with the overall operation of the microcontroller. In this mode when the appropriate timer register is full, the microcontroller will generate an internal interrupt signal directing the program flow to the respective internal interrupt vector. For the pulse width capture mode, the internal system clock is also used as the timer clock source but the timer will only run when the correct logic condition appears on the external timer input pin. As this is an external event and not synchronised with the internal timer clock, the microcontroller will only see this external event when the next timer clock pulse arrives. As a result, there may be small differences in measured values requiring programmers to take this into account during programming. The same applies if the timer is configured to be in the event counting mode, which again is an external event and not synchronised with the internal system or timer clock.

When the Timer/Event Counter is read, or if data is written to the preload register, the clock is inhibited to avoid errors, however as this may result in a counting error, this should be taken into account by the programmer. Care must be taken to ensure that the timers are properly initialized before using them for the first time. The associated timer enable bits in the interrupt control register must be properly set otherwise the internal interrupt associated with the timer will remain inactive. The edge select, timer mode and clock source control bits in timer control register must also be correctly set to ensure the timer is properly configured for the required application. It is also important to ensure that an initial value is first loaded into the timer registers before the timer is switched on; this is because after power-on the initial values of the timer registers are unknown. After the timer has been initialized the timer can be turned on and off by controlling the enable bit in the timer control register.

When the Timer/Event Counter overflows, its corresponding interrupt request flag in the interrupt control register will be set. If the Timer/Event Counter interrupt is enabled this will in turn generate an interrupt signal. However irrespective of whether the interrupts are enabled or not, a Timer/Event Counter overflow will also generate a wake-up signal if the device is in a Power-down condition. This situation may occur if the Timer/Event Counter is in the Event Counting Mode and if the external signal continues to change state. In such a case, the Timer/Event Counter will continue to count these external events and if an overflow occurs the device will be woken up from its Power-down condition. To prevent such a wake-up from occurring, the timer interrupt request flag should first be set high before issuing the ″HALT″ instruction to enter the Idle/Sleep Mode.

## Timer Program Example-Timer/Event Counter 0

The program shows how the Timer/Event Counter 0 registers are setup along with how the interrupts are enabled and managed. Note how the Timer/Event Counter is turned on, by setting bit 4 of the Timer Control Register. The Timer/Event Counter can be turned off in a similar way by clearing the same bit. This example program sets the Timer/Event Counters to be in the timer mode, which uses the internal system clock as their clock source.

**PFD Programming Example**

```
org      04h             ; external interrupt vector
org      08h             ; Timer Counter 0 interrupt vector
jmp      tmr0int         ; jump here when Timer 0 overflows
  :
  :
org      20h             ; main program
  :
  :
  ;internal Timer 0 interrupt routine
tmr0int:
:
  ; Timer 0 main program placed here
:
:
begin:
;setup Timer 0 registers
mov      a,09bh          ; setup Timer 0 preload value
mov      tmr,a
mov      a,001h          ; setup Timer 0 control register
mov      tmrc,a          ; timer mode and prescaler set to /2
;setup interrupt register
mov      a,00dh          ; enable master interrupt and both timer interrupts
mov      intc0,a
:
:
set      tmrc.4          ; start Timer 0
```

## 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 PB, PC and PD logic I/O pins, with the desired function chosen via register bits. Keys are organised into groups of four, with each group known as a module and having a module number, M0 to M5.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 | Keys - n | Touch Key Module | Touch Key | Shared I/O Pin |
|---|---|---|---|---|
| BS83B08-3 | 8 | M0 | K1~K4 | PB0~PB3 |
| | | M1 | K5~K8 | PB4~PB7 |
| BS83B12-3 | 12 | M0 | K1~K4 | PB0~PB3 |
| | | M1 | K5~K8 | PB4~PB7 |
| | | M2 | K9~K12 | PC0~PC3 |
| BS83B16-3<br>BS83B16G-3 | 16 | M0 | K1~K4 | PB0~PB3 |
| | | M1 | K5~K8 | PB4~PB7 |
| | | M2 | K9~K12 | PC0~PC3 |
| | | M3 | K13~K16 | PC4~PC7 |
| BS83C24-3 | 16 | M0 | K1~K4 | PB0~PB3 |
| | | M1 | K5~K8 | PB4~PB7 |
| | | M2 | K9~K12 | PC0~PC3 |
| | | M3 | K13~K16 | PC4~PC7 |
| | | M4 | K17~K20 | PD0~PD3 |
| | | M5 | K21~K24 | PD4~PD7 |

**General Purpose Data Memory**

### Touch Key Register Definition

Each touch key module, which contains four touch key functions, has its own suite of six 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 and has a range of M0 to M5.

| Name | Usage |
|---|---|
| TKMn16DH | 16-bit C/F counter high byte |
| TKMn16DL | 16-bit C/F counter low byte |
| TKMnC0 | Control Register 0<br>Key Select/X2 freq/filter control/frequency select |
| TKMnC1 | Control Register 1<br>Sensor Oscillator Control/Touch key or I/O select. |
| TKMnC2 | Control Register 2<br>Counter on-off and clear control/reference clock control/Start bit |
| TKMnC3 | Control Register 3<br>Counter overflow bits/Reference Oscillator Overflow Time Select |

**Register Listing**

| Register Name | Bit | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| TKMn16DH | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| TKMn16DL | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| TKMnC0 | MnMXS1 | MnMXS0 | D5 | D4 | D3 | D2 | D1 | D0 |
| TKMnC1 | MnK4OEN | MnK3OEN | MnK2OEN | MnK1OEN | MnK4IO | MnK3IO | MnK2IO | MnK1IO |
| TKMnC2 | Mn16CTON | D6 | MnST | MnROEN | MnRCCLR | Mn16CTCLR | D1 | MnROS |
| TKMnC3 | D9 | D8 | MnRCOV | Mn16CTOV | D3 | MnROVS2 | MnROVS1 | MnROVS0 |

**Touch Key Module**

**TKMn16DH 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 Module n 16-bit counter high byte contents

**TKMn16DL 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 Module n 16-bit counter low byte contents

**TKMnC0 Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | MnMXS1 | MnMXS0 | 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 |

Bits 7~6 **MnMXS1**, **MnMXS0**: Multiplexer Key Select

| Bit | | Module Number | | | | | |
|---|---|---|---|---|---|---|---|
| MnMXS1 | MnMXS0 | M0 | M1 | M2 | M3 | M4 | M5 |
| 0 | 0 | Key 1 | Key 5 | Key 9 | Key 13 | Key 17 | Key 21 |
| 0 | 1 | Key 2 | Key 6 | Key 10 | Key 14 | Key 18 | Key 22 |
| 1 | 0 | Key 3 | Key 7 | Key 11 | Key 15 | Key 19 | Key 23 |
| 1 | 1 | Key 4 | Key 8 | Key 12 | Key 16 | Key 20 | Key 24 |

Bit 5~0 **D5~D0**: These bits must be set to the binary value ″011000″

**TKMnC1 Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | MnK4OEN | MnK3OEN | MnK2OEN | MnK1OEN | MnK4IO | MnK3IO | MnK2IO | MnK1IO |
| 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 |

Bits 7~4 MnK4OEN~ MnK1OEN: key selector control

| MnK4OEN | M0 | M1 | M2 | M3 | M4 | M5 |
|---|---|---|---|---|---|---|
| | Key 4 | Key 8 | Key 12 | Key 16 | Key 20 | Key 24 |
| 0 | Disable | | | | | |
| 1 | Enable | | | | | |

| MnK3OEN | M0 | M1 | M2 | M3 | M4 | M5 |
|---|---|---|---|---|---|---|
| | Key 3 | Key 7 | Key 11 | Key 15 | Key 19 | Key 23 |
| 0 | Disable | | | | | |
| 1 | Enable | | | | | |

| MnK2OEN | M0 | M1 | M2 | M3 | M4 | M5 |
|---|---|---|---|---|---|---|
| | Key 2 | Key 6 | Key 10 | Key 14 | Key 18 | Key 22 |
| 0 | Disable | | | | | |
| 1 | Enable | | | | | |

| MnK1OEN | M0 | M1 | M2 | M3 | M4 | M5 |
|---|---|---|---|---|---|---|
| | Key 1 | Key 5 | Key 9 | Key 13 | Key 17 | Key 21 |
| 0 | Disable | | | | | |
| 1 | Enable | | | | | |

Bits 3~0 I/O Pin or Touch Key Function Select

| MnK4IO | M0 | M1 | M2 | M3 | M4 | M5 |
|---|---|---|---|---|---|---|
| | PB3/Key 4 | PB7/Key 8 | PC3/Key 12 | PC7/Key 16 | PD3/Key 20 | PD7/Key 24 |
| 0 | I/O pin | | | | | |
| 1 | Touch Key | | | | | |

| MnK3IO | M0 | M1 | M2 | M3 | M4 | M5 |
|---|---|---|---|---|---|---|
| | PB2/Key 3 | PB6/Key 7 | PC2/Key 11 | PC6/Key 15 | PD2/Key 19 | PD6/Key 23 |
| 0 | I/O pin | | | | | |
| 1 | Touch Key | | | | | |

| MnK2IO | M0 | M1 | M2 | M3 | M4 | M5 |
|---|---|---|---|---|---|---|
| | PB1/Key 2 | PB5/Key 6 | PC1/Key 10 | PC5/Key 14 | PD1/Key 18 | PD5/Key 22 |
| 0 | I/O pin | | | | | |
| 1 | Touch Key | | | | | |

| MnK1IO | M0 | M1 | M2 | M3 | M4 | M5 |
|---|---|---|---|---|---|---|
| | PB0/Key 1 | PB4/Key 5 | PC0/Key 9 | PC4/Key 13 | PD0/Key 17 | PD4/Key 21 |
| 0 | I/O pin | | | | | |
| 1 | Touch Key | | | | | |

**TKMnC2 Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | Mn16CTON | — | MnST | MnROEN | MnRCCLR | Mn16CTCLR | — | MnROS |
| R/W | R/W | — | R/W | R/W | R/W | R/W | — | R/W |
| POR | 0 | — | 0 | 0 | 0 | 0 | — | 0 |

Bit 7 **Mn16CTON**: 16-bit C/F counter control
0: disable
1: enable

Bit 6 Reserved bit, must not be modified.

Bit 5 **MnST**: Time slot counter start control
0: time slot counter stopped
0 → 1: enable time slot counter.
When this bit changes from low to high the time slot counter will be enabled and the touch sense procedure started. When the time slot counter has completed its counting an interrupt will be generated.

Bit 4 **MnROEN**: Reference clock control
0: disable
1: enable

Bit 3 **MnRCCLR**: Time slot counter clear control
0: no change
1: clear counter
This bit must be first set to 1 and then to 0.

Bit 2 **Mn16CTCLR**: 16-bit C/F counter clear control
0: no change
1: clear counter
This bit must be first set to 1 and then to 0.

Bit 1 Reserved bit, must not be modified.

Bit 0 **MnROS**: Time slot counter clock source
0: reference clock
1: sense key oscillator
M0: Key 4, M1: Key 8, M2: Key12, M3: Key 16, M4: Key 20, M5: Key 24

**TKMnC3 Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | D7 | D6 | MnRCOV | Mn16CTOV | — | MnROVS2 | MnROVS1 | MnROVS0 |
| R/W | R | R | R | R/W | — | R/W | R/W | R/W |
| POR | 0 | 0 | 0 | 0 | — | 0 | 0 | 0 |

Bit 7~6 **D7, D6**: Read only bits -- unknown values

Bit 5 **MnRCOV**: Time slot counter overflow flag
0: no overflow
1: overflow

Bit 4 **Mn16CTOV**: 16-bit C/F counter overflow flag
0: no overflow
1: overflow

Bit 3 Reserved bit, must not be modified.

Bits 2~1 **MnROVS2~MnROVS0**: Time slot counter overflow time setup
000: 64 count
001: 128 count
010: 256 count
011: 512 count
100: 1024 count
101: 2048 count
110: 4096 count
111: 8192 count

## 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.

The device contains four touch key inputs which are shared with logical I/O pins, with the desired function selected using register bits. The Touch Key module also has its own interrupt vectors and set of interrupts flags.

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.

## Touch Key Interrupt

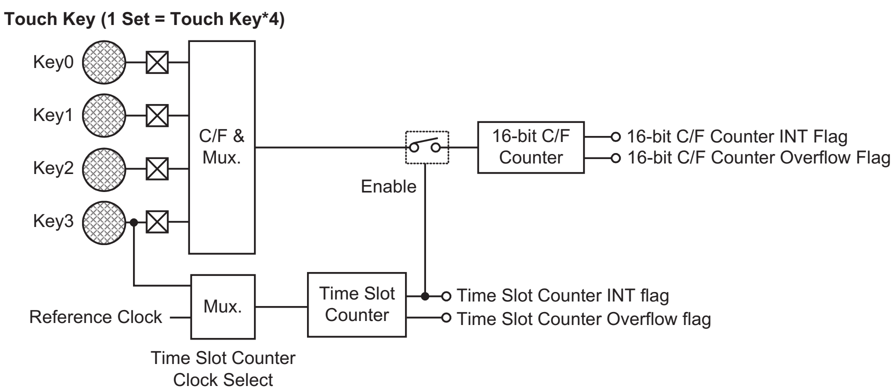
Each touch key module, which consists of four touch keys, has two independent interrupts, one for each of the, 16-bit C/F counter and time slot counter.

The time slot counter interrupt has its own interrupt vector while the 16-bit C/F counter interrupts are contained within the Multi-function interrupts and therefore do not have their own vector. Care must be taken during programming as the 16-bit C/F counter interrupt flags contained within the Multi-function interrupts will not be automatically reset upon entry into the interrupt service routine but rather must be reset manually by the application program. More details regarding the touch key interrupts are located in the interrupt section of the datasheet.

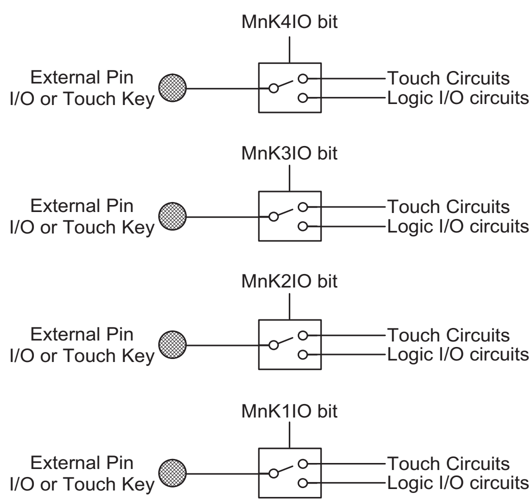
## Programming Considerations

After the relevant registers are setup, the touch key detection process is initiated the changing the MnST bit from low to high. This will enable and synchronise all relevant oscillators. The MnRCOV flag, which is the time slot counter flag will go high and remain high until 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.




**Touch Switch Module Block Diagram**




**Touch Key or I/O Function Select**

## Serial Interface Module − SIM

These devices contain a Serial Interface Module, which includes both the four line SPI interface or 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 pins are pin shared with other I/O pins and must 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.

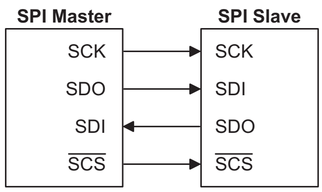
### SPI Interface

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 this device provided 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, SCK 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 must first be enabled 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 as I/O function.

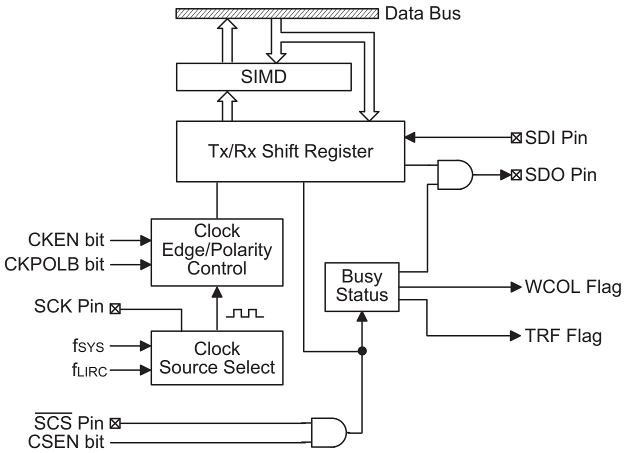



**SPI Master/Slave Connection**

The SPI function in this device offers the following features:

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

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




SPI Block Diagram

### SPI Registers

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

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

SPI Registers List

The SIMD register is used to store the data being transmitted and received. The same register is used by both the SPI and $I^2C$ 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

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

**SIMC0 Register**

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

Bit 7~5 **SIM2, SIM1, 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 $f_{LIRC}$
100: Unused
101: SPI slave mode
110: $I^2C$ slave mode
111: Unused
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 the TM0. If the SPI Slave Mode is selected then the clock will be supplied by an external Master device.

Bit 4~2 unimplemented, read as "0"

Bit 1 **SIMEN**: SIM Control
0: disable
1: enable
The bit is the overall on/off control for the SIM interface. When the SIMEN bit is cleared, the SDI, SDO, SCK and $\overline{SCS}$, or SDA and SCL lines will be as I/O 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 unimplemented, read as "0"

**SIMC2 Register**

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

Bit 7~6 Undefined bit
This bit can be read or written by user software program.

Bit 5 **CKPOLB**: Determines the base condition of the clock line
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**: Determines SPI SCK active clock edge type
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
1: MSB
This is the data shift select bit and is used to select how the data is transferred, either MSB or LSB first. Setting the bit high will select MSB first and low for LSB first.

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

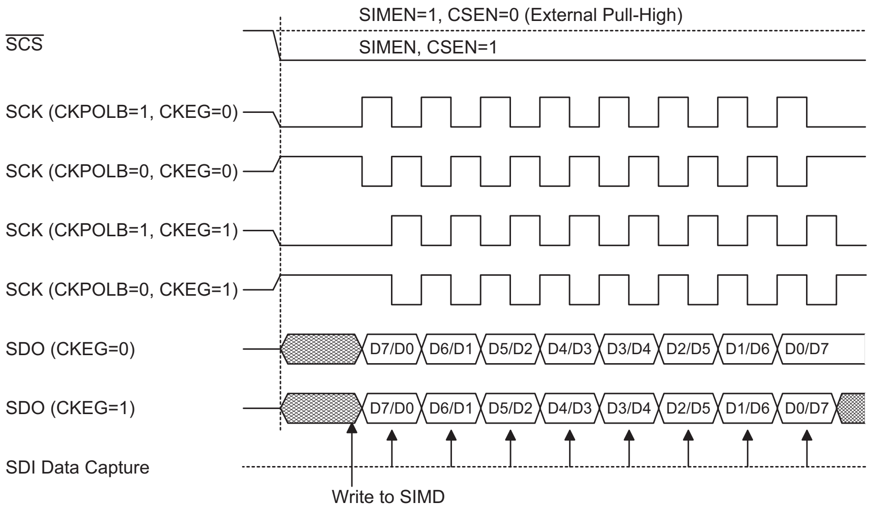
Bit 1 **WCOL**: SPI Write Collision flag
0: No collision
1: Collision
The WCOL flag is used to detect if a data collision has occurred. If this bit is high it means that data has been attempted to be written to the SIMD register during a data transfer operation. This writing operation will be ignored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

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

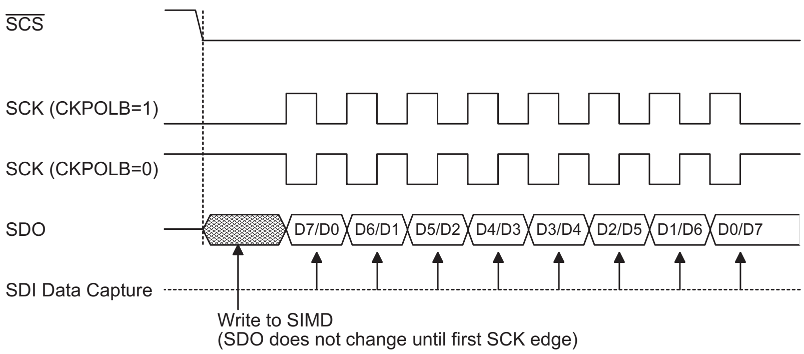
### SPI Communication

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

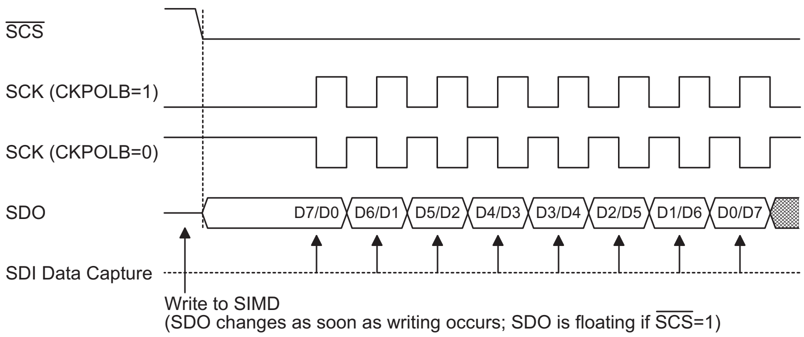
The SPI will continue to function even in the IDLE Mode.




**SPI Master Mode Timing**

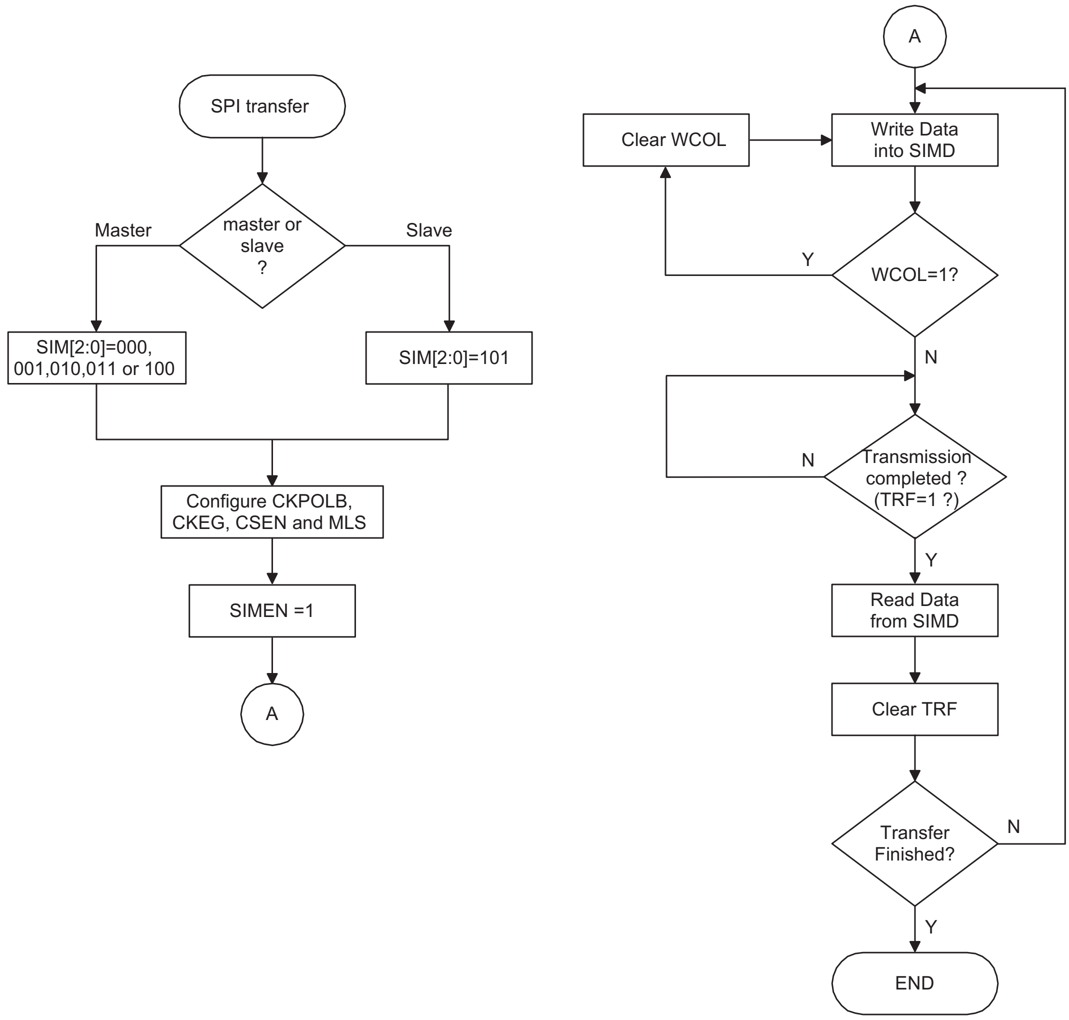



**SPI Slave Mode Timing – CKEG=0**




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

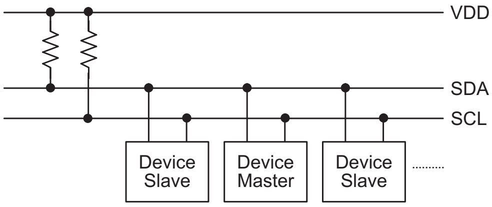
**SPI Slave Mode Timing – CKEG=1**




**SPI Transfer Control Flowchart**

## 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.

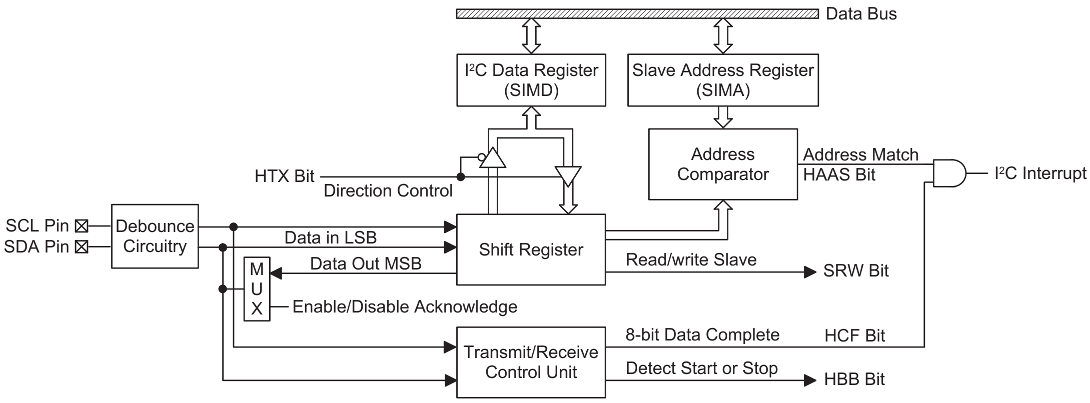



**I²C Master Slave Bus Connection**

### 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 these devices, 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.

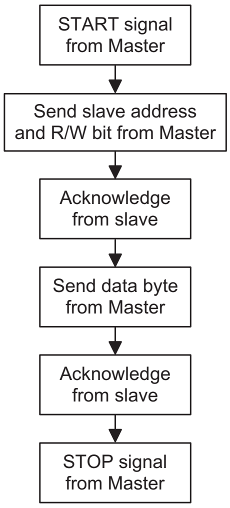



**I²C Block Diagram**

The debounce time of the $I^2C$ interface uses the system clock to in effect add a debounce time to the external clock to reduce the possibility of glitches on the clock line causing erroneous operation. The debounce time, is 2 system clocks. To achieve the required $I^2C$ data transfer speed, there exists a relationship between the system clock, $f_{SYS}$, and the $I^2C$ debounce time. For either the $I^2C$ Standard or Fast mode operation, users must take care of the selected system clock frequency and the configured debounce time to match the criterion shown in the following table.

| $I^2C$ Debounce Time Selection | $I^2C$ Standard Mode (100kHz) | $I^2C$ Fast Mode (400kHz) |
|---|---|---|
| 2 system clock debounce | $f_{SYS}$ > 4MHz | $f_{SYS}$ > 10MHz |

**$I^2C$ Minimum $f_{SYS}$ Frequency**




### $I^2C$ Registers

There are four control registers associated with the $I^2C$ bus, SIMC0, SIMC1, SIMA and I2CTOC and one data register, SIMD. The SIMD register, which is shown in the above SPI section, is used to store the data being transmitted and received on the $I^2C$ bus. Before the microcontroller writes data to the $I^2C$ bus, the actual data to be transmitted must be placed in the SIMD register. After the data is received from the $I^2C$ bus, the microcontroller can read it from the SIMD register. Any transmission or reception of data from the $I^2C$ bus must be made via the SIMD register. The SIM pins are pin shared with other I/O pins and must be selected using the SIMEN bit in the SIMC0 register.

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

| Register Name | Bit | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SIMC0 | SIM2 | SIM1 | SIM0 | — | — | — | SIMEN | — |
| SIMC1 | HCF | HAAS | HBB | HTX | TXAK | SRW | IAMWU | RXAK |
| SIMD | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| SIMA | IICA6 | IICA5 | IICA4 | IICA3 | IICA2 | IICA1 | IICA0 | D0 |
| I2CTOC | I2CTOEN | I2CTOF | I2CTOS5 | I2CTOS4 | I2CTOS3 | I2CTOS2 | I2CTOS1 | I2CTOS0 |

**$I^2C$ Register List**

**SIMC0 Register**

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

Bit 7~5 **SIM2, SIM1, 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 $f_{LIRC}$
100: Unused
101: SPI slave mode
110: $I^2C$ slave mode
111: Unused
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 the TM0. If the SPI Slave Mode is selected then the clock will be supplied by an external Master device.

Bit 4~2 unimplemented, read as "0"

Bit 1 **SIMEN**: SIM Control
0: disable
1: enable
The bit is the overall on/off control for the SIM interface. When the SIMEN bit is cleared, the SDI, SDO, SCK and $\overline{SCS}$, or SDA and SCL lines will be as I/O 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 unimplemented, read as "0"

**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 HASS 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**: Select $I^2C$ slave device is transmitter or receiver
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 $I^2C$ address match wake-up from 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 do not receive acknowledge flag
The RXAK flag is the receiver acknowledge flag. When the RXAK flag is ″0″, it means that a acknowledge signal has been received at the 9th clock, after 8 bits of data have been transmitted. When the slave device in the transmit mode, the slave device checks the RXAK flag to determine if the master receiver wishes to receive the next byte. The slave transmitter will therefore continue sending out data until the RXAK flag is ″1″. When this occurs, the slave transmitter will release the SDA line to allow the master to send a STOP signal to release the $I^2C$ Bus.

**I2CTOC Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | I2CTOEN | I2CTOF | I2CTOS5 | I2CTOS4 | I2CTOS3 | I2CTOS2 | I2CTOS1 | I2CTOS0 |
| 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 **I2CTOEN**: $I^2C$ Time-out Control
0: disable
1: enable

Bit 6 **I2CTOF**: Time-out flag
0: no time-out
1: time-out occurred

Bit 5~0 **I2CTOS5~I2CTOS0**: Time-Out Time Definition
$I^2C$ time-out clock source is $f_{LIRC}/32$.
$I^2C$ Time-Out time is given by: [I2CTOS5 : I2CTOS0]+1) x ($32/f_{LIRC}$)

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

**SIMA Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | IICA6 | IICA5 | IICA4 | IICA3 | IICA2 | IICA1 | IICA0 | 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 7~1 **IICA6~ IICA0**: $I^2C$ slave address
IICA6~ IICA0 is the $I^2C$ slave address bit 6~bit 0.
The SIMA register is also used by the SPI interface but has the name SIMC2. The SIMA register is the location where the 7-bit slave address of the slave device is stored. Bits 7~1 of the SIMA register define the device slave address. Bit 0 is not defined.
When a master device, which is connected to the $I^2C$ bus, sends out an address, which matches the slave address in the SIMA register, the slave device will be selected. Note that the SIMA register is the same register address as SIMC2 which is used by the SPI interface.

Bit 0 Undefined bit
This bit can be read or written by user software program.

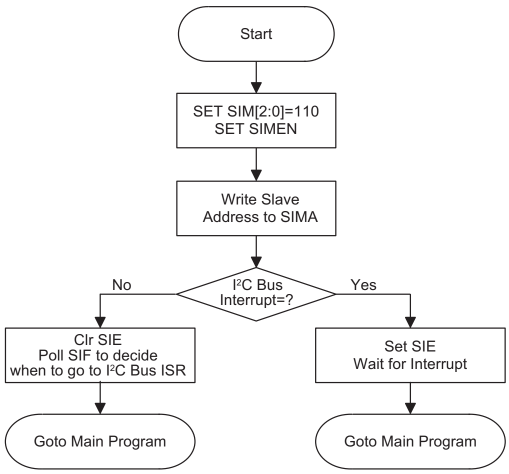
### I²C Bus Communication

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 bit to determine whether the interrupt source originates from an address match or from the completion of an 8-bit data transfer. 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 "1" to enable the I²C bus.

Step 2
Write the slave address of the device to the I²C bus address register SIMA.

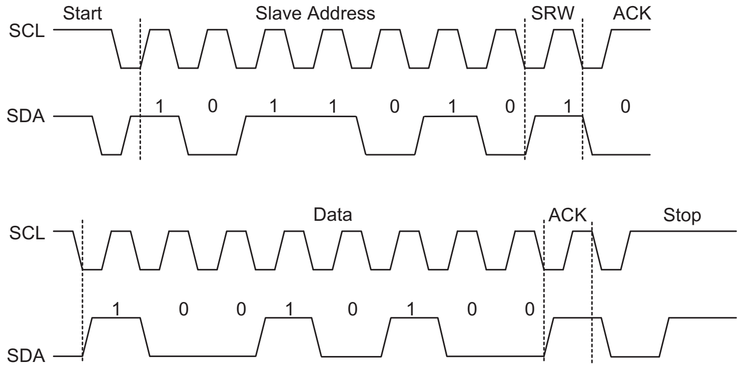
Step 3
Set the SIME and SIM Muti-Function interrupt enable bit of the interrupt control register to enable the SIM interrupt and Multi-function interrupt.




**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.




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

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

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

**$I^2C$ Communication Timing Diagram**

#### Slave Address

The transmission of a START signal by the master will be detected by all devices on the $I^2C$ bus. To determine which slave device the master wishes to communicate with, the address of the slave device will be sent out immediately following the START signal. All slave devices, after receiving this 7-bit address data, will compare it with their own 7-bit slave address. If the address sent out by the master matches the internal address of the microcontroller slave device, then an internal $I^2C$ 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 $I^2C$ bus interrupt can come from two sources, when the program enters the interrupt subroutine, the HAAS bit 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. When a slave address is matched, the device must be placed in either the transmit mode and then write data to the SIMD register, or in the receive mode where it must implement a dummy read from the SIMD register to release the SCL line.

#### $I^2C$ Bus Read/Write Signal

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

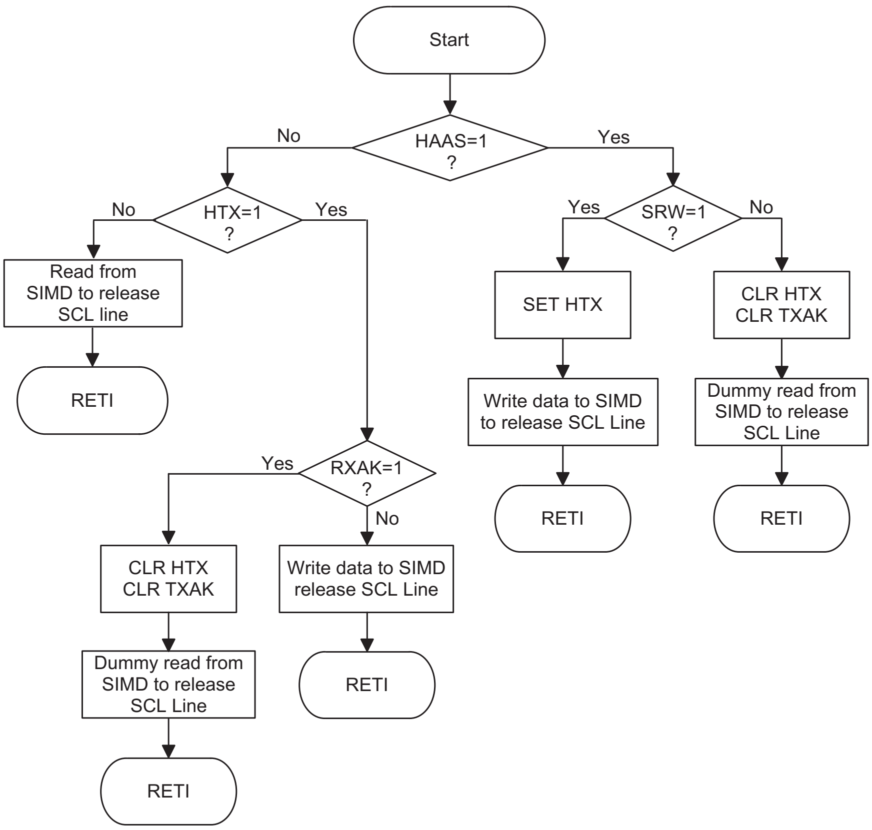
#### $I^2C$ Bus Slave Address Acknowledge Signal

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

#### $I^2C$ Bus Data and Acknowledge Signal

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

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

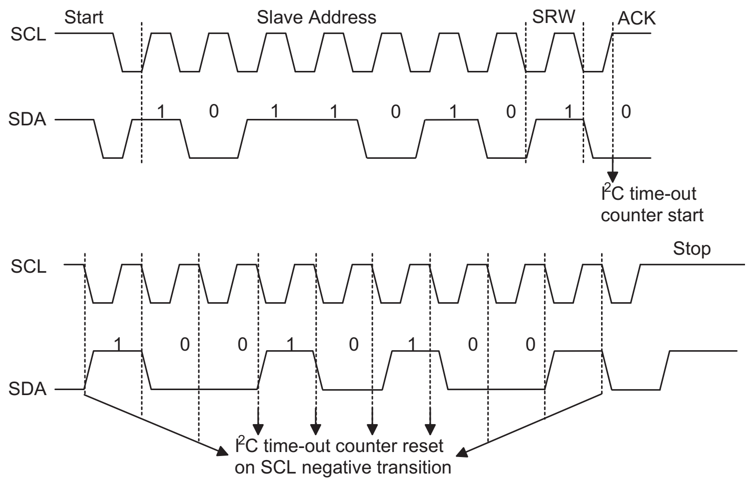



$I^2C$ Bus ISR Flow Chart

### I²C Time-out Control

In order to reduce the problem of I²C lockup due to reception of erroneous clock sources, clock, a time-out function is provided. If the clock source to the I²C is not received then after a fixed time period, the I²C circuitry and registers will be reset.

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 I2CTOC 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 I2CTOEN bit will be cleared to zero and the I2CTF bit will be set high to indicate that a time-out condition as occurred. The time-out condition will also generate an interrupt which uses the I²C interrrupt vector. When an I²C time-out occurs the I²C internal circuitry will be reset and the registers will be reset into the following condition:

| Register | After I²C Time-out |
|---|---|
| SIMDR, SIMAR, SIMC0 | No change |
| SIMC1 | Reset to POR condition |

**I²C Registers After Time-out**

The I2CTOF flag can be cleared by the application program. There are 64 time-out periods which can be selected using bits in the I2CTOC register. The time-out time is given by the formula:

$((1\sim64) \times 32) / f_{LIRC}$. This gives a range of about 1ms to 64ms. Note also that the LIRC oscillator is continuously enabled.

## Interrupts

Interrupts are an important part of any microcontroller system. When an external event or an internal function such as a Touch Action or Timer/Event Counter overflow requires microcontroller attention, their corresponding interrupt will enforce a temporary suspension of the main program allowing the microcontroller to direct attention to their respective needs. The devices contains several external interrupt and internal interrupts 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 Touch Keys, Timer/Event Counter, Time Base, SIM 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~INTC3 registers which setup the primary interrupts, the second is the MFI0~MFI2 registers which setup the Multi-function interrupts. Finally there is an INTEG register to setup the external interrupt trigger edge type.

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

| Function | Enable Bit | Request Flag | Notes |
|---|---|---|---|
| Global | EMI | — | — |
| INT Pin | INTE | INTF | — |
| Touch Key Module | TKMnE | TKMnF | n=0~5 |
| SIM | SIME | SIMF | |
| EEPROM | DEE | DEF | |
| Multi-function | MFnE | MFnF | n=0~2 |
| Time Base | TBE | TBF | — |
| Timer/Event Counter | TnE, TE | TnF, TF | n=0~1 |
| Touch Key Module 16-bit Counter | Mn16CTE | Mn16CTF | n=0~5 |

**Interrupt Register Bit Naming Conventions**

### Interrupt Register Contents

**BS83B08-3**

| Name | Bit | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| INTEG | — | — | — | — | — | — | INTS1 | INTS0 |
| INTC0 | — | TKM1F | TKM0F | INTF | TKM1E | TKM0E | INTE | EMI |
| INTC1 | TF | MF0F | DEF | SIMF | TE | MF0E | DEE | SIME |
| INTC2 | — | — | — | TBF | — | — | — | TBE |
| MFI0 | M116CTF | D6 | M016CTF | D4 | M116CTE | D2 | M016CTE | D0 |

**BS83B12-3**

| Name | Bit | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| INTEG | — | — | — | — | — | — | INTS1 | INTS0 |
| INTC0 | — | TKM1F | TKM0F | INTF | TKM1E | TKM0E | INTE | EMI |
| INTC1 | TF | MF0F | DEF | SIMF | TE | MF0E | DEE | SIME |
| INTC2 | — | TKM2F | MF1F | TBF | — | TKM2E | MF1E | TBE |
| MFI0 | M116CTF | D6 | M016CTF | D4 | M116CTE | D2 | M016CTE | D0 |
| MFI1 | — | — | M216CTF | D4 | — | — | M216CTE | D0 |

**BS83B16-3/BS83B16G-3**

| Name | Bit | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| INTEG | — | — | — | — | — | — | INTS1 | INTS0 |
| INTC0 | — | TKM1F | TKM0F | INTF | TKM1E | TKM0E | INTE | EMI |
| INTC1 | TF | MF0F | DEF | SIMF | TE | MF0E | DEE | SIME |
| INTC2 | TKM3F | TKM2F | MF1F | TBF | TKM3E | TKM2E | MF1E | TBE |
| MFI0 | M116CTF | D6 | M016CTF | D4 | M116CTE | D2 | M016CTE | D0 |
| MFI1 | M316CTF | D6 | M216CTF | D4 | M316CTE | D2 | M216CTE | D0 |

**BS83C24-3**

| Name | Bit | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| INTEG | — | — | — | — | — | — | INTS1 | INTS0 |
| INTC0 | — | TKM1F | TKM0F | INTF | TKM1E | TKM0E | INTE | EMI |
| INTC1 | T0F | MF0F | DEF | SIMF | T0E | MF0E | DEE | SIME |
| INTC2 | TKM3F | TKM2F | MF1F | TBF | TKM3E | TKM2E | MF1E | TBE |
| INTC3 | TKM5F | TKM4F | MF2F | T1F | TKM5E | TKM4E | MF2E | T1E |
| MFI0 | M116CTF | D6 | M016CTF | D4 | M116CTE | D2 | M016CTE | D0 |
| MFI1 | M316CTF | D6 | M216CTF | D4 | M316CTE | D2 | M216CTE | D0 |
| MFI2 | M516CTF | D6 | M416CTF | D4 | M516CTE | D2 | M416CTE | D0 |

**INTEG Register -- All devices**

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

Bit 7~2 unimplemented, read as "0"

Bit 1~0 **INTS1, INTS0**: interrupt edge control for INT pin
- 00: disable
- 01: rising edge
- 10: falling edge
- 11: rising and falling edges

**INTC0 Register -- All devices**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | — | TKM1F | TKM0F | INTF | TKM1E | TKM0E | 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 **TKM1F**: Touch key module 1 interrupt request flag
- 0: No request
- 1: interrupt request

Bit 5 **TKM0F**: Touch Key module 0 interrupt request flag
- 0: No request
- 1: Interrupt request

Bit 4 **INTF**: INT pin interrupt request flag
- 0: No request
- 1: Interrupt request

Bit 3 **TKM1E**: Touch key module 1 interrupt control
- 0: disable
- 1: enable

Bit 2 **TKM0E**: Touch key module 0 interrupt control
- 0: disable
- 1: enable

Bit 1 **INTE**: INT pin interrupt control
- 0: disable
- 1: enable

Bit 0 **EMI**: Global interrupt control
- 0: disable
- 1: enable

**INTC1 Register -- BS83B08-3/B12-3/B16-3/B16G-3**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | TF | MF0F | DEF | SIMF | TE | MF0E | DEE | SIME |
| R/W | R/W | R/W | R/W | — | R/W | R/W | R/W | — |
| POR | 0 | 0 | 0 | — | 0 | 0 | 0 | — |

Bit 7 **TF**: Timer/Event Counter interrupt request flag
0: no request
1: interrupt request

Bit 6 **MF0F**: Multi-function interrupt 0 request flag
0: no request
1: interrupt request

Bit 5 **DEF**: Data EEPROM interrupt request flag
0: no request
1: interrupt request

Bit 4 **SIMF**: SIM interrupt reqeust flag
0: no request
1: interrupt request

Bit 3 **TE**: Timer/Event Counter interrupt control
0: disable
1: enable

Bit 2 **MF0E**: Multi-function interrupt 0 control
0: disable
1: enable

Bit 1 **DEE**: Data EEPROM interrupt control
0: disable
1: enable

Bit 0 **SIME**: SIM interrupt control
0: disable
1: enable

**INTC1 Register -- BS83C24-3**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | T0F | MF0F | DEF | SIMF | T0E | MF0E | DEE | SIME |
| R/W | R/W | R/W | R/W | — | R/W | R/W | R/W | — |
| POR | 0 | 0 | 0 | — | 0 | 0 | 0 | — |

Bit 7 **T0F**: Timer/Event Counter interrupt 0 request flag
0: no request
1: interrupt request

Bit 6 **MF0F**: Multi-function interrupt 0 request flag
0: no request
1: interrupt request

Bit 5 **DEF**: Data EEPROM interrupt request flag
0: no request
1: interrupt request

Bit 4 **SIMF**: SIM interrupt reqeust flag
0: no request
1: interrupt request

Bit 3 **T0E**: Timer/Event Counter 0 interrupt control
0: disable
1: enable

Bit 2 **MF0E**: Multi-function interrupt 0 control
0: disable
1: enable

Bit 1 **DEE**: Data EEPROM interrupt control
0: disable
1: enable

Bit 0 **SIME**: SIM interrupt control
0: disable
1: enable

**INTC2 Register -- BS83B08-3**

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

Bit 7~5 unimplemented, read as ″0″

Bit 4 **TBF**: Time Base interrupt request flag
0: no request
1: interrupt request

Bit 3~1 unimplemented, read as ″0″

Bit 0 **TBE**: Time Base interrupt control
0: disable
1: enable

**INTC2 Register -- BS83B12-3**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | — | TKM2F | MF1F | TBF | — | TKM2E | MF1E | TBE |
| R/W | — | R/W | R/W | R/W | — | R/W | R/W | R/W |
| POR | — | 0 | 0 | 0 | — | 0 | 0 | 0 |

Bit 7 unimplemented, read as "0"

Bit 6 **TKM2F**: Touch key Module 2 interrupt request flag
0: no request
1: interrupt request

Bit 5 **MF1F**: Multi-function interrupt 1 request flag
0: no request
1: interrupt request

Bit 4 **TBF**: Time Base interrupt request flag
0: no request
1: interrupt request

Bit 3 unimplemented, read as "0"

Bit 2 **TKM2E**: Touch key module 2 interrupt control
0: disable
1: enable

Bit 1 **MF1E**: Multi-function interrupt 1 control
0: disable
1: enable

Bit 0 **TBE**: Time Base interrupt control
0: disable
1: enable

**INTC2 Register -- BS83B16-3/BS83B16G-3/BS83C24-3**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | TKM3F | TKM2F | MF1F | TBF | TKM3E | TKM2E | MF1E | TBE |
| 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 **TKM3F**: Touch key module 3 interrupt request flag
0: No request
1: Interrupt request

Bit 6 **TKM2F**: Touch key module 2 interrupt request flag
0: No request
1: Interrupt request

Bit 5 **MF1F**: Multi-function interrupt 1 request flag
0: No request
1: Interrupt request

Bit 4 **TBF**: Time Base interrupt request flag
0: No request
1: Interrupt request

Bit 3 **TKM3E**: Touch key module 3 interrupt control
0: disable
1: enable

Bit 2 **TKM2E**: Touch key module 2 interrupt control
0: disable
1: enable

Bit 1 **MF1E**: Multi-function interrupt 1 control
0: disable
1: enable

Bit 0 **TBE**: Time Base interrupt control
0: disable
1: enable

**INTC3 Register -- BS83C24-3**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | TKM5F | TKM4F | MF2F | T1F | TKM5E | TKM4E | MF2E | T1E |
| 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 **TKM5F**: Touch key module 5 interrupt request flag
0: No request
1: Interrupt request

Bit 6 **TKM4F**: Touch key module 4 interrupt request flag
0: No request
1: Interrupt request

Bit 5 **MF2F**: Multi-function interrupt 2 request flag
0: No request
1: Interrupt request

Bit 4 **T1F**: Timer/Event Counter 1 interrupt request flag
0: No request
1: Interrupt request

Bit 3 **TKM5E**: Touch key module 5 interrupt control
0: disable
1: enable

Bit 2 **TKM4E**: Touch key module 4 interrupt control
0: disable
1: enable

Bit 1 **MF2E**: Multi-function interrupt 2 control
0: disable
1: enable

Bit 0 **T1E**: Timer/Event Counter 1 Interrupt Control
0: disable
1: enable

**MFI0 Register -- All devices**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | M116CTF | D6 | M016CTF | D4 | M116CTE | D2 | M016CTE | 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 **M116CTF**: Touch key module 1 16-bit counter interrupt request flag
0: no request
1: interrupt request

Bit 6 **D6**: Reserved bit, must not be modified.

Bit 5 **M016CTF**: Touch key module 0 16-bit counter interrupt request flag
0: no request
1: interrupt request

Bit 4 **D4**: Reserved bit, must not be modified.

Bit 3 **M116CTE**: Touch key module 1 16-bit timer interrupt control
0: disable
1: enable

Bit 2 **D2**: Reserved bit, must not be modified.

Bit 1 **M016CTE**: Touch key module 0 16-bit timer interrupt control
0: disable
1: enable

Bit 0 **D0**: Reserved bit, must not be modified.

**MFI1 Register -- BS83B12-3**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | — | — | M216CTF | D4 | — | — | M216CTE | D0 |
| R/W | — | — | R/W | R/W | — | — | R/W | R/W |
| POR | — | — | 0 | 0 | — | — | 0 | 0 |

Bit 7~6 unimplemented, read as "0"

Bit 5 **M216CTF**: Touch key module 2 16-bit counter interrupt request flag
0: no request
1: interrupt request

Bit 4 **D4**: Reserved bit, must not be modified.
0: no request
1: interrupt request

Bit 3~2 unimplemented, read as "0"

Bit 1 **M216CTE**: Touch key module 2 16-bit timer interrupt control
0: disable
1: enable

Bit 0 **D0**: Reserved bit, must not be modified.
0: disable
1: enable

**MFI1 Register -- BS83B16-3/BS83B16G-3/BS83C24-3**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | M316CTF | D6 | M216CTF | D4 | M316CTE | D2 | M216CTE | 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 **M316CTF**: Touch key module 3 16-bit counter interrupt request flag
0: no request
1: interrupt request

Bit 6 **D6**: Reserved bit, must not be modified.

Bit 5 **M216CTF**: Touch key module 2 16-bit counter interrupt request flag
0: no request
1: interrupt request

Bit 4 **D4**: Reserved bit, must not be modified.

Bit 3 **M316CTE**: Touch key module 3 16-bit timer interrupt control
0: disable
1: enable

Bit 2 **D2**: Reserved bit, must not be modified.

Bit 1 **M216CTE**: Touch key module 2 16-bit timer interrupt control
0: disable
1: enable

Bit 0 **D0**: Reserved bit, must not be modified.

**MFI2 Register -- BS83C24-3**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | M516CTF | D6 | M416CTF | D4 | M516CTE | D2 | M416CTE | 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 **M516CTF**: Touch key module 5 16-bit counter interrupt request flag
0: no request
1: interrupt request

Bit 6 **D6**: Reserved bit, must not be modified.

Bit 5 **M416CTF**: Touch key module 4 16-bit counter interrupt request flag
0: no request
1: interrupt request

Bit 4 **D4**: Reserved bit, must not be modified.

Bit 3 **M516CTE**: Touch key module 5 16-bit timer interrupt control
0: disable
1: enable

Bit 2 **D2**: Reserved bit, must not be modified.

Bit 1 **M416CTE**: Touch key module 4 16-bit timer interrupt control
0: disable
1: enable

Bit 0 **D0**: Reserved bit, must not be modified.

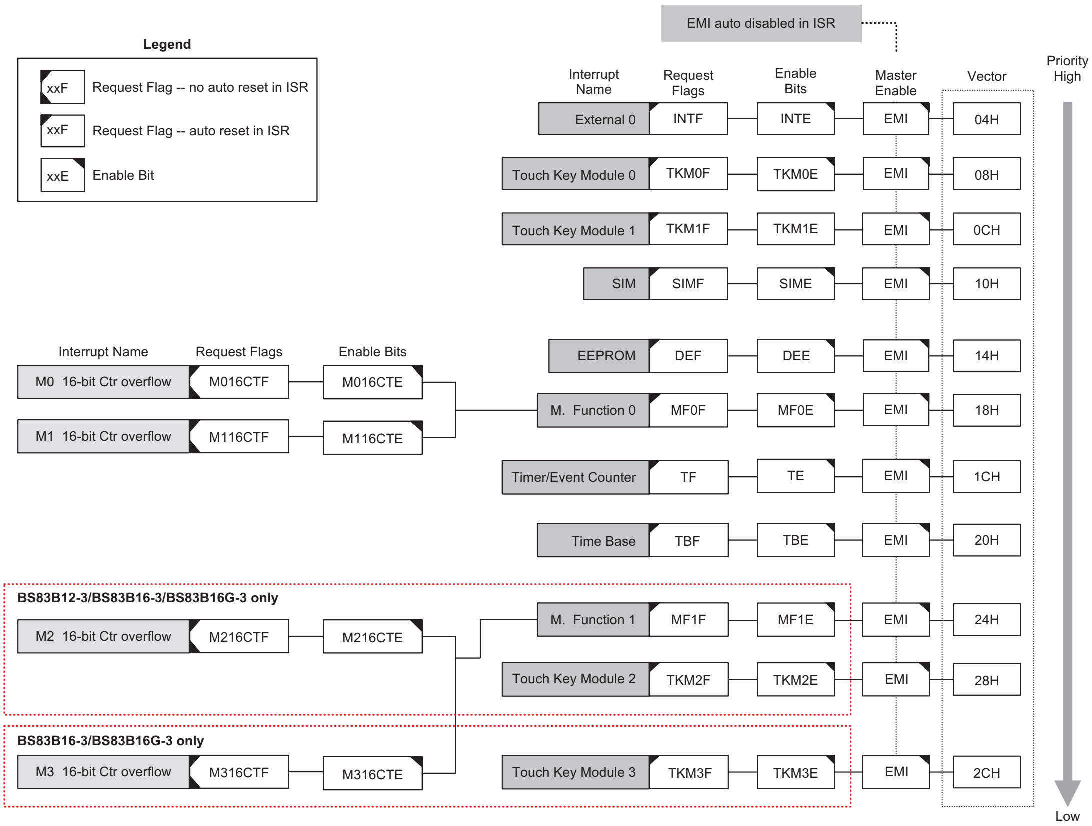
## Interrupt Operation

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

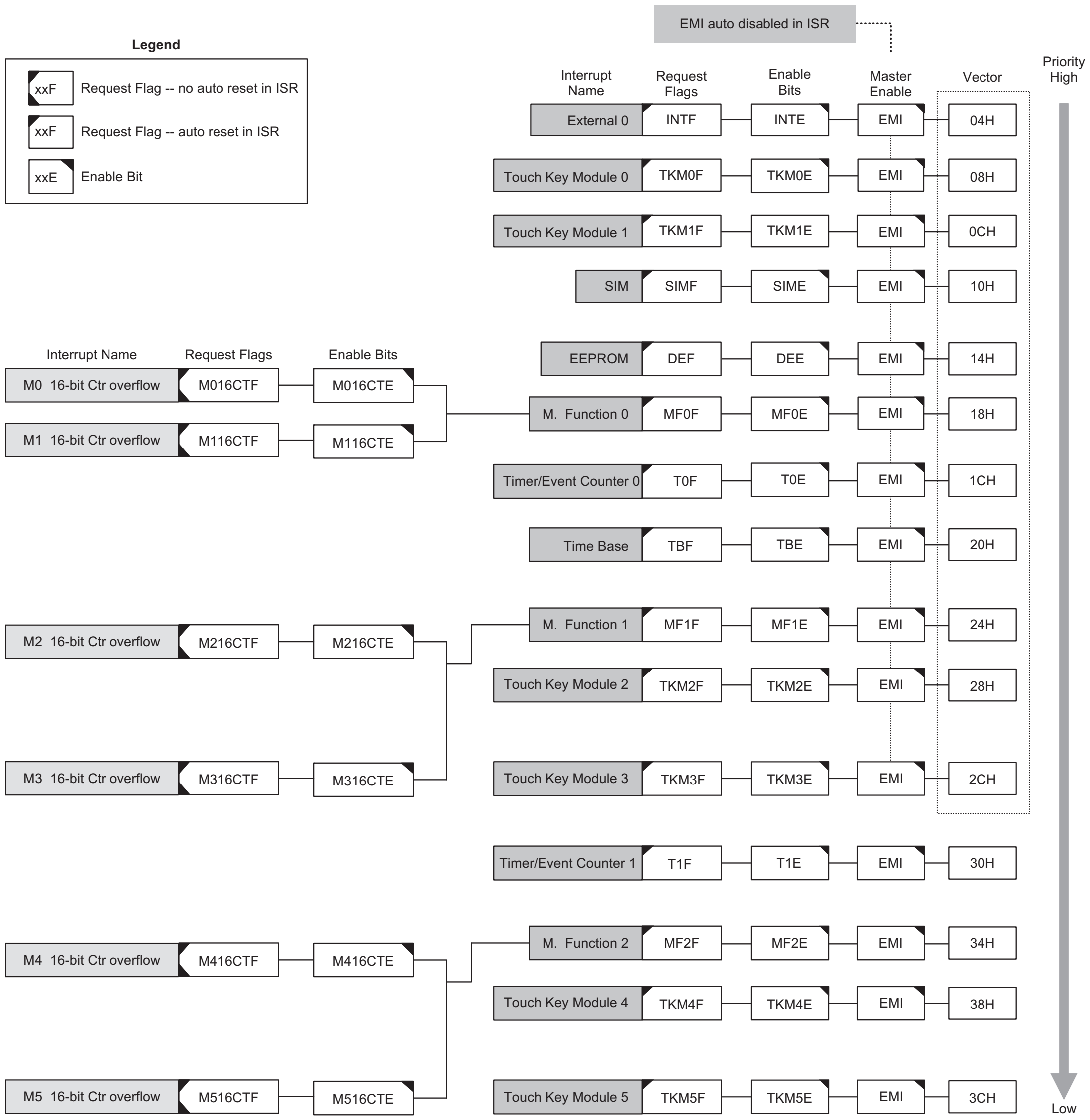
When an interrupt is generated, the Program Counter, which stores the address of the next instruction to be executed, will be transferred onto the stack. The Program Counter will then be loaded with a new address which will be the value of the corresponding interrupt vector. The microcontroller will then fetch its next instruction from this interrupt vector. The instruction at this vector will usually be a JMP instruction 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 instruction, which retrieves the original Program Counter address from the stack and allows the microcontroller to continue with normal execution at the point where the interrupt occurred.

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

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




**Interrupt Structure -- BS83B08-3/B12-3/B16-3/B16G-3**




**Interrupt Structure -- BS83C24-3**

### 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 pin, it can only be configured as 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 flag, 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 pin will remain valid even if the pin is used as an external interrupt input.

The INTEG register is used to select the type of active edge that will trigger the external interrupt. A choice of either rising or falling or both edge types can be chosen to trigger an external interrupt. Note that the INTEG register can also be used to disable the external interrupt function.

### Multi-function Interrupt

Within these devices there are one or two Multi-function interrupts. Unlike the other independent interrupts, these interrupts have no independent source, but rather are formed from the Touch Key module timer interrupt sources.

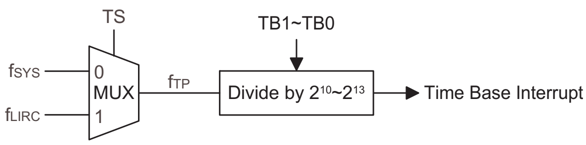
A Multi-function interrupt request will take place when any of the Multi-function interrupt request flags, MFnF are set. The Multi-function interrupt flags will be set when any of their included functions generate an interrupt request flag. To allow the program to branch to its respective interrupt vector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

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

### Time Base Interrupts

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 respective timer function. When this happens its respective 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 respective 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 with a fixed time period. Its clock source originates from the internal clock source $f_{SYS}$ or $f_{LIRC}$. This clock 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.




**Time Base Structure**

**TBC Register**

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

Bit 7~6 unimplemented, read as ″0″

Bit 5~4 **TB1~TB0**: Select Time Base Time-out Period
00: $1024/f_{TP}$
01: $2048/f_{TP}$
10: $4096/f_{TP}$
11: $8192/f_{TP}$

Bit 3~0 unimplemented, read as ″0″

### Timer/Event Counter Interrupt

For a Timer/Event Counter interrupt to occur, the global interrupt enable bit, EMI, and the corresponding timer interrupt enable bit, TE, T0E or T1E, must first be set. An actual Timer/Event Counter interrupt will take place when the Timer/Event Counter request flag, TF, T0F or T1F, is set, a situation that will occur when the relevant Timer/Event Counter overflows. When the interrupt is enabled, the stack is not full and a Timer/Event Counter n overflow occurs, a subroutine call to the relevant timer interrupt vector, will take place. When the interrupt is serviced, the timer interrupt request flag, TF, T0F or T1F, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts.

### EEPROM Interrupt

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

### Touch Key Interrupts

For a Touch Key interrupt to occur, the global interrupt enable bit, EMI, and the corresponding Touch Key interrupt enable TKMnE must be first set. An actual Touch Key interrupt will take place when the Touch Key request flag. TKMnF, is set, a situation that will occur when the 13-bit time slot counter in the relevant Touch Key module 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 timer interrupt vector, will take place. When the interrupt is serviced, the Touch Key interrupt request flag, TKMnF, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts.

### 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. 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 a byte of data has been transmitted or received by the SIM interface, a subroutine call to the respective interrupt vector, will take place. When the Serial Interface Interrupt is serviced, the SIM interrupt request flag, SIF, will be automatically cleared and the EMI bit will be automatically cleared to disable other interrupts.

## Interrupt Wake-up Function

Each of the interrupt functions has the capability of waking up the microcontroller when in the SLEEP or IDLE Mode. A wake-up is generated when an interrupt request flag changes from low to high and is independent of whether the interrupt is enabled or not. Therefore, even though the device is in the SLEEP or IDLE Mode and its system oscillator stopped, situations such as external edge transitions on the external interrupt pins, a low power supply voltage or comparator input change may cause their respective interrupt flag to be set high and consequently generate an interrupt. Care must therefore be taken if spurious wake-up situations are to be avoided. If an interrupt wake-up function is to be disabled then the corresponding interrupt request flag should be set high before the device enters the SLEEP or IDLE Mode. The interrupt enable bits have no effect on the interrupt wake-up function.

## Programming Considerations

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

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

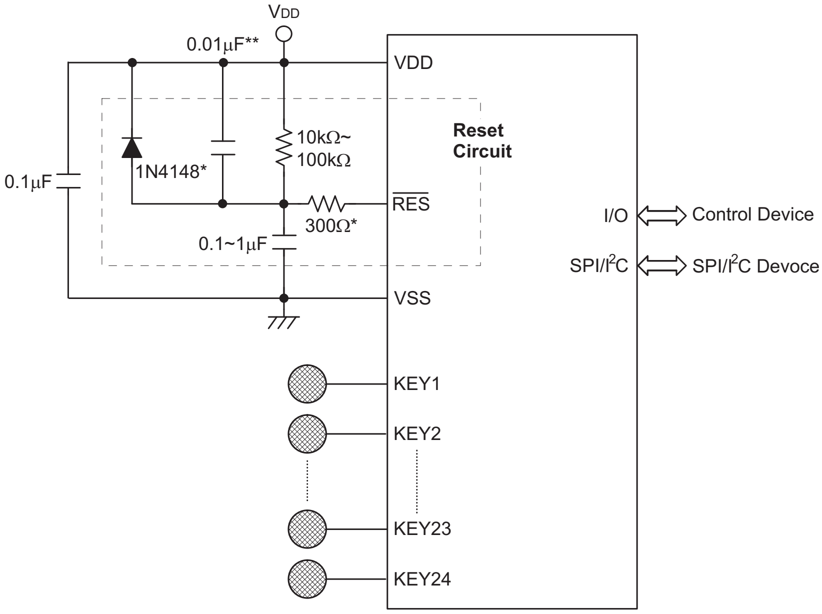
It is recommended that programs do not use the "CALL" instruction within the interrupt service subroutine. Interrupts often occur in an unpredictable manner or need to be serviced immediately. If only one stack is left and the interrupt is not well controlled, the original control sequence will be damaged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

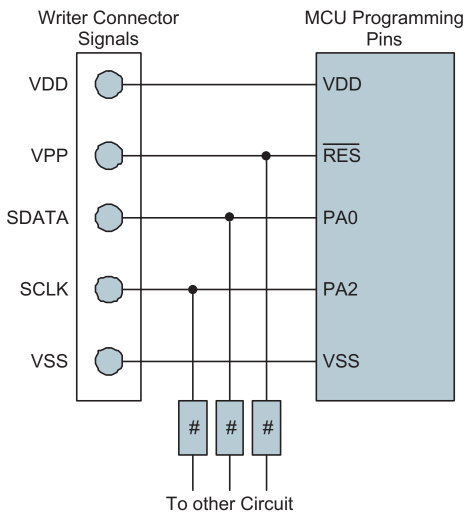
Every interrupt has the capability of waking up the microcontroller when it is in SLEEP or IDLE Mode, the wake up being generated when the interrupt request flag changes from low to high. If it is required to prevent a certain interrupt from waking up the microcontroller then its respective request flag should be first set high before enter SLEEP or IDLE Mode.

As only the Program Counter is pushed onto the stack, then when the interrupt is serviced, if the contents of the accumulator, status register or other registers are altered by the interrupt service program, their contents should be saved to the memory at the beginning of the interrupt service routine.

To return from an interrupt subroutine, either a RET or RETI instruction may be executed. The RETI instruction in addition to executing a return to the main program also automatically sets the EMI bit high to allow further interrupts. The RET instruction however only executes a return to the main program leaving the EMI bit in its present zero state and therefore disabling the execution of further interrupts.

## Application Circuits




Note: ″*″ It is recommended that this component is added for added ESD protection.

″**″ It is recommended that this component is added in environments where power line noise is significant.

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

# 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 microcontrollers, 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 Operations

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 where rotate data operations are used is to implement multiplication and division calculations.

### Branches and Control Transfer

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

### Bit Operations

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

### Table Read Operations

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

### Other Operations

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

## Instruction Set Summary

The 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 | 1Note | 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 | 1Note | 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 | 1Note | 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 | 1Note | Z, C, AC, OV |
| DAA [m] | Decimal adjust ACC for Addition with result in Data Memory | 1Note | 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 | 1Note | Z |
| ORM A,[m] | Logical OR ACC to Data Memory | 1Note | Z |
| XORM A,[m] | Logical XOR ACC to Data Memory | 1Note | 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 | 1Note | 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 | 1Note | Z |
| DECA [m] | Decrement Data Memory with result in ACC | 1 | Z |
| DEC [m] | Decrement Data Memory | 1Note | Z |
| **Rotate** | | | |
| RRA [m] | Rotate Data Memory right with result in ACC | 1 | None |
| RR [m] | Rotate Data Memory right | 1Note | None |
| RRCA [m] | Rotate Data Memory right through Carry with result in ACC | 1 | C |
| RRC [m] | Rotate Data Memory right through Carry | 1Note | C |
| RLA [m] | Rotate Data Memory left with result in ACC | 1 | None |
| RL [m] | Rotate Data Memory left | 1Note | None |
| RLCA [m] | Rotate Data Memory left through Carry with result in ACC | 1 | C |
| RLC [m] | Rotate Data Memory left through Carry | 1Note | C |
| **Data Move** | | | |
| MOV A,[m] | Move Data Memory to ACC | 1 | None |
| MOV [m],A | Move ACC to Data Memory | 1Note | None |
| MOV A,x | Move immediate data to ACC | 1 | None |

| Mnemonic | Description | Cycles | Flag Affected |
|---|---|---|---|
| **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** | | | |
| 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** | | | |
| TABRDC [m] | Read table (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 |
| CLR WDT1 | Pre-clear Watchdog Timer | 1 | TO, PDF |
| CLR WDT2 | Pre-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.

3. For the ″CLR WDT1″ and ″CLR WDT2″ instructions the TO and PDF flags may be affected by the execution status. The TO and PDF flags are cleared after both ″CLR WDT1″ and ″CLR WDT2″ instructions are consecutively executed. Otherwise the TO and PDF flags remain unchanged.

## 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 bitwise 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

**CLR WDT1** Pre-clear Watchdog Timer

Description: The TO, PDF flags and the WDT are all cleared. Note that this instruction works in conjunction with CLR WDT2 and must be executed alternately with CLR WDT2 to have effect. Repetitively executing this instruction without alternately executing CLR WDT2 will have no effect.

Operation: WDT cleared
TO ← 0
PDF ← 0

Affected flag(s): TO, PDF

**CLR WDT2** Pre-clear Watchdog Timer

Description: The TO, PDF flags and the WDT are all cleared. Note that this instruction works in conjunction with CLR WDT1 and must be executed alternately with CLR WDT1 to have effect. Repetitively executing this instruction without alternately executing CLR WDT1 will have no effect.

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: $[m] \leftarrow \overline{[m]}$

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: $ACC \leftarrow \overline{[m]}$

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:
$[m] \leftarrow ACC + 00H$ or
$[m] \leftarrow ACC + 06H$ or
$[m] \leftarrow ACC + 60H$ or
$[m] \leftarrow ACC + 66H$

Affected flag(s): C

**DEC [m]** Decrement Data Memory

Description: Data in the specified Data Memory is decremented by 1.

Operation: $[m] \leftarrow [m] - 1$

Affected flag(s): Z

**DECA [m]** Decrement Data Memory with result in ACC

Description: Data in the specified Data Memory is decremented by 1. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.

Operation: $ACC \leftarrow [m] - 1$

Affected flag(s): Z

**HALT** Enter power down mode

Description: This instruction stops the program execution and turns off the system clock. The contents of the Data Memory and registers are retained. The WDT and prescaler are cleared. The power down flag PDF is set and the WDT time-out flag TO is cleared.

Operation:
$TO \leftarrow 0$
$PDF \leftarrow 1$

Affected flag(s): TO, PDF

| | |
|---|---|
| **INC [m]** | Increment Data Memory |
| Description | Data in the specified Data Memory is incremented by 1. |
| Operation | [m] ← [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)<br>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 and the carry flag are rotated right by 1 bit with bit 0 rotated into bit 7. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

Operation:
ACC.i ← [m].(i+1); (i = 0~6)
ACC.7 ← [m].0

Affected flag(s): None

**RRC [m]** Rotate Data Memory right through Carry

Description: The contents of the specified Data Memory and the carry flag are rotated right by 1 bit. Bit 0 replaces the Carry bit and the original carry flag is rotated into bit 7.

Operation:
[m].i ← [m].(i+1); (i = 0~6)
[m].7 ← C
C ← [m].0

Affected flag(s): C

**RRCA [m]** Rotate Data Memory right through Carry with result in ACC

Description: Data in the specified Data Memory and the carry flag are rotated right by 1 bit. Bit 0 replaces the Carry bit and the original carry flag is rotated into bit 7. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

Operation:
ACC.i ← [m].(i+1); (i = 0~6)
ACC.7 ← C
C ← [m].0

Affected flag(s): C

| | |
|---|---|
| **SBC A,[m]** | Subtract Data Memory from ACC with Carry |
| Description | The contents of the specified Data Memory and the complement of the carry flag are subtracted from the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1. |
| Operation | ACC ← ACC − [m] − $\overline{C}$ |
| 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] ← 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<br>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<br>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
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
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<br>ACC.7 ~ ACC.4 ← [m].3 ~ [m].0 |
| Affected flag(s) | None |

| | |
|---|---|
| **SZ [m]** | Skip if Data Memory is 0 |
| Description | If the contents of the specified Data Memory is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction. |
| Operation | Skip if [m] = 0 |
| Affected flag(s) | None |

| | |
|---|---|
| **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]<br>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 |

| | |
|---|---|
| **TABRDC [m]** | Read table (current page) to TBLH and Data Memory |
| Description | The low byte of the program code (current page) addressed by the table pointer (TBLP) is moved to the specified Data Memory and the high byte moved to TBLH. |
| Operation | [m] ← program code (low byte)<br>TBLH ← program code (high byte) |
| Affected flag(s) | None |

**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.

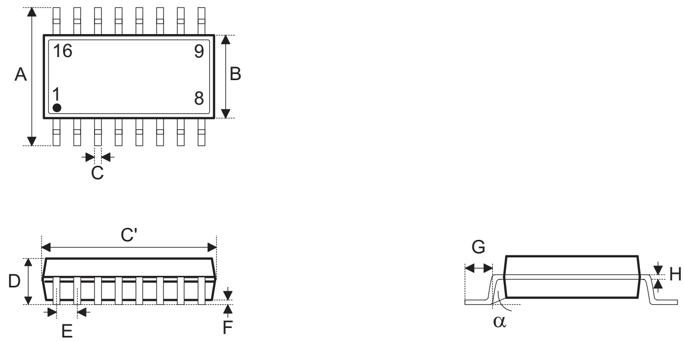
Operation: ACC ← ACC ″XOR″ x

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 (http://www.holtek.com.tw/english/literature/package.pdf) for the latest version of the package information.

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

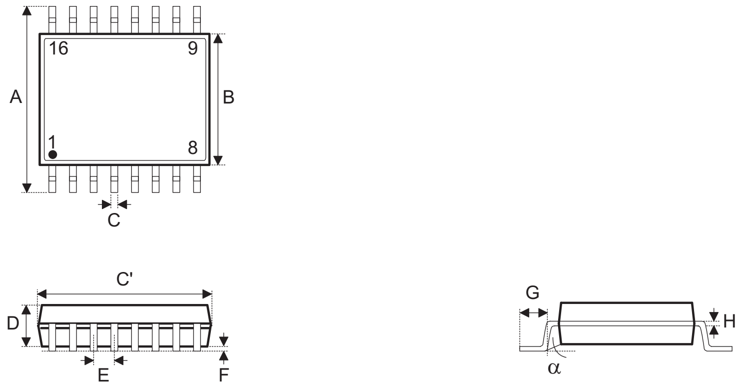



**MS-012**

| Symbol | Dimensions in inch | | |
|---|---|---|---|
| | Min. | Nom. | Max. |
| A | 0.228 | — | 0.244 |
| B | 0.150 | — | 0.157 |
| C | 0.012 | — | 0.020 |
| C′ | 0.386 | — | 0.402 |
| D | — | — | 0.069 |
| E | — | 0.050 | — |
| F | 0.004 | — | 0.010 |
| G | 0.016 | — | 0.050 |
| H | 0.007 | — | 0.010 |
| α | 0° | — | 8° |

| Symbol | Dimensions in mm | | |
|---|---|---|---|
| | Min. | Nom. | Max. |
| A | 5.79 | — | 6.20 |
| B | 3.81 | — | 3.99 |
| C | 0.30 | — | 0.51 |
| C′ | 9.80 | — | 10.21 |
| D | — | — | 1.75 |
| E | — | 1.27 | — |
| F | 0.10 | — | 0.25 |
| G | 0.41 | — | 1.27 |
| H | 0.18 | — | 0.25 |
| α | 0° | — | 8° |

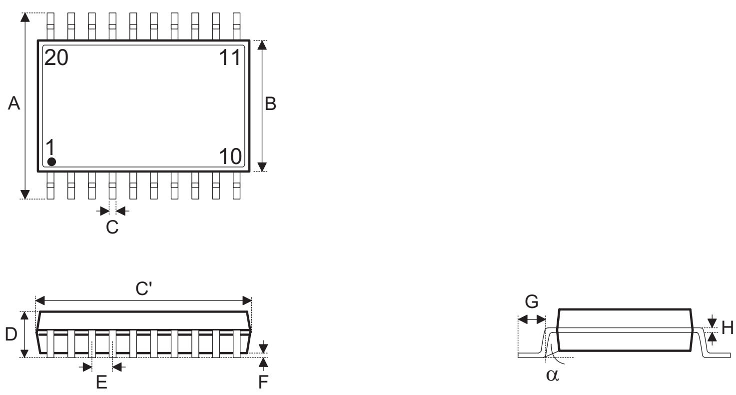
## 16-pin SSOP (150mil) Outline Dimensions




| Symbol | Dimensions in inch | | |
|---|---|---|---|
| | Min. | Nom. | Max. |
| A | 0.228 | — | 0.244 |
| B | 0.150 | — | 0.157 |
| C | 0.008 | — | 0.012 |
| C′ | 0.189 | — | 0.197 |
| D | 0.054 | — | 0.060 |
| E | — | 0.025 | — |
| F | 0.004 | — | 0.010 |
| G | 0.022 | — | 0.028 |
| H | 0.007 | — | 0.010 |
| α | 0° | — | 8° |

| Symbol | Dimensions in mm | | |
|---|---|---|---|
| | Min. | Nom. | Max. |
| A | 5.79 | — | 6.20 |
| B | 3.81 | — | 3.99 |
| C | 0.20 | — | 0.30 |
| C′ | 4.80 | — | 5.00 |
| D | 1.37 | — | 1.52 |
| E | — | 0.64 | — |
| F | 0.10 | — | 0.25 |
| G | 0.56 | — | 0.71 |
| H | 0.18 | — | 0.25 |
| α | 0° | — | 8° |

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

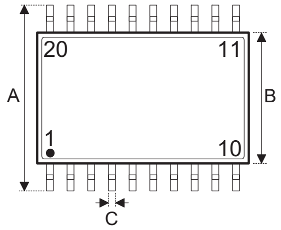


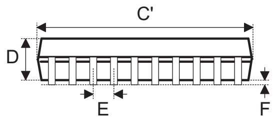

**MS-013**

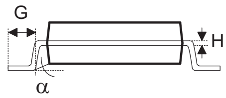
| Symbol | Dimensions in inch | | |
|---|---|---|---|
| | Min. | Nom. | Max. |
| A | 0.393 | — | 0.419 |
| B | 0.256 | — | 0.300 |
| C | 0.012 | — | 0.020 |
| C′ | 0.496 | — | 0.512 |
| D | — | — | 0.104 |
| E | — | 0.050 | — |
| F | 0.004 | — | 0.012 |
| G | 0.016 | — | 0.050 |
| H | 0.008 | — | 0.013 |
| α | 0° | — | 8° |

| Symbol | Dimensions in mm | | |
|---|---|---|---|
| | Min. | Nom. | Max. |
| A | 9.98 | — | 10.64 |
| B | 6.50 | — | 7.62 |
| C | 0.30 | — | 0.51 |
| C′ | 12.60 | — | 13.00 |
| D | — | — | 2.64 |
| E | — | 1.27 | — |
| F | 0.10 | — | 0.30 |
| G | 0.41 | — | 1.27 |
| H | 0.20 | — | 0.33 |
| α | 0° | — | 8° |

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



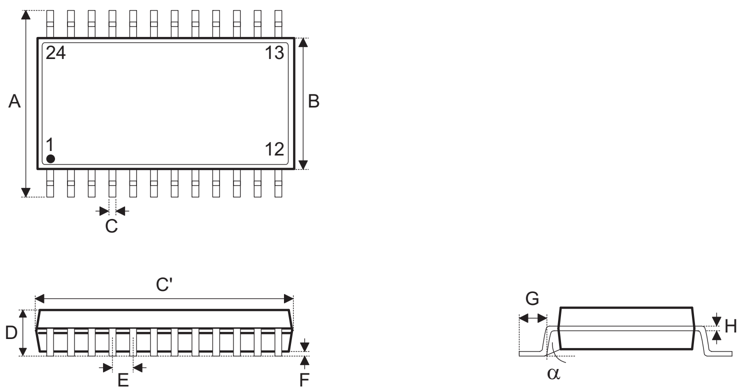




| Symbol | Dimensions in inch | | |
|---|---|---|---|
| | Min. | Nom. | Max. |
| A | 0.228 | — | 0.244 |
| B | 0.150 | — | 0.158 |
| C | 0.008 | — | 0.012 |
| C′ | 0.335 | — | 0.347 |
| D | 0.049 | — | 0.065 |
| E | — | 0.025 | — |
| F | 0.004 | — | 0.010 |
| G | 0.015 | — | 0.050 |
| H | 0.007 | — | 0.010 |
| α | 0° | — | 8° |

| Symbol | Dimensions in mm | | |
|---|---|---|---|
| | Min. | Nom. | Max. |
| A | 5.79 | — | 6.20 |
| B | 3.81 | — | 4.01 |
| C | 0.20 | — | 0.30 |
| C′ | 8.51 | — | 8.81 |
| D | 1.24 | — | 1.65 |
| E | — | 0.64 | — |
| F | 0.10 | — | 0.25 |
| G | 0.38 | — | 1.27 |
| H | 0.18 | — | 0.25 |
| α | 0° | — | 8° |

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

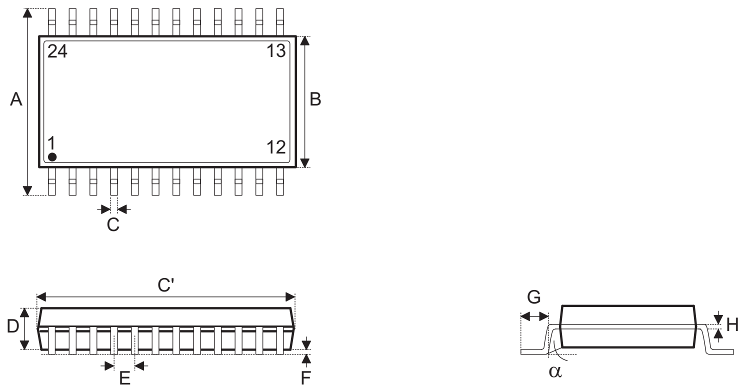



**MS-013**

| Symbol | Dimensions in inch | | |
|---|---|---|---|
| | Min. | Nom. | Max. |
| A | 0.393 | — | 0.419 |
| B | 0.256 | — | 0.300 |
| C | 0.012 | — | 0.020 |
| C′ | 0.598 | — | 0.613 |
| D | — | — | 0.104 |
| E | — | 0.050 | — |
| 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 | 9.98 | — | 10.64 |
| B | 6.50 | — | 7.62 |
| C | 0.30 | — | 0.51 |
| C′ | 15.19 | — | 15.57 |
| D | — | — | 2.64 |
| E | — | 1.27 | — |
| F | 0.10 | — | 0.30 |
| G | 0.41 | — | 1.27 |
| H | 0.20 | — | 0.33 |
| α | 0° | — | 8° |

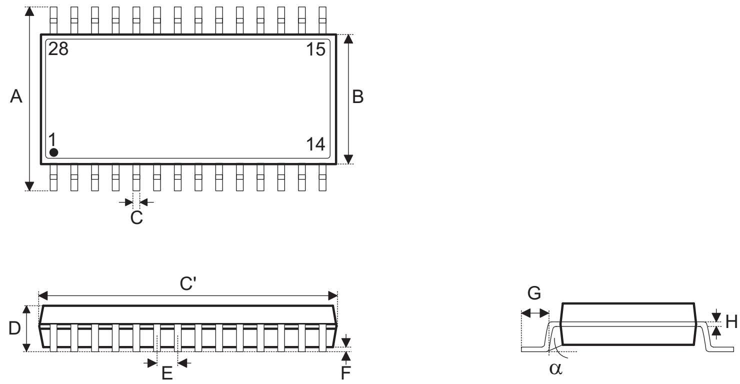
### 24-pin SSOP (150mil) Outline Dimensions




| Symbol | Dimensions in inch | | |
|---|---|---|---|
| | Min. | Nom. | Max. |
| A | 0.228 | — | 0.244 |
| B | 0.150 | — | 0.157 |
| C | 0.008 | — | 0.012 |
| C′ | 0.335 | — | 0.346 |
| D | 0.054 | — | 0.060 |
| E | — | 0.025 | — |
| F | 0.004 | — | 0.010 |
| G | 0.022 | — | 0.028 |
| H | 0.007 | — | 0.010 |
| α | 0° | — | 8° |

| Symbol | Dimensions in mm | | |
|---|---|---|---|
| | Min. | Nom. | Max. |
| A | 5.79 | — | 6.20 |
| B | 3.81 | — | 3.99 |
| C | 0.20 | — | 0.30 |
| C′ | 8.51 | — | 8.79 |
| D | 1.37 | — | 1.52 |
| E | — | 0.64 | — |
| F | 0.10 | — | 0.25 |
| G | 0.56 | — | 0.71 |
| H | 0.18 | — | 0.25 |
| α | 0° | — | 8° |

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

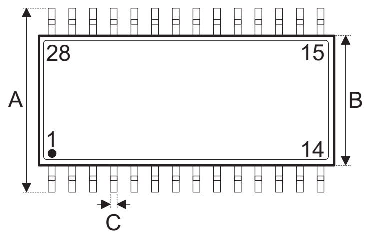


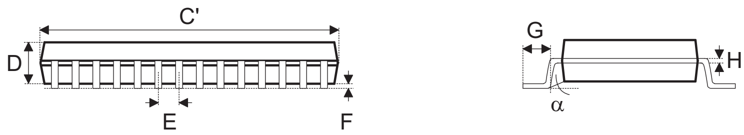

- MS-013

| Symbol | Dimensions in inch | | |
|---|---|---|---|
| | Min. | Nom. | Max. |
| A | 0.393 | — | 0.419 |
| B | 0.256 | — | 0.300 |
| C | 0.012 | — | 0.020 |
| C′ | 0.697 | — | 0.713 |
| D | — | — | 0.104 |
| E | — | 0.050 | — |
| 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 | 9.98 | — | 10.64 |
| B | 6.50 | — | 7.62 |
| C | 0.30 | — | 0.51 |
| C′ | 17.70 | — | 18.11 |
| D | — | — | 2.64 |
| E | — | 1.27 | — |
| F | 0.10 | — | 0.30 |
| G | 0.41 | — | 1.27 |
| H | 0.20 | — | 0.33 |
| α | 0° | — | 8° |

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

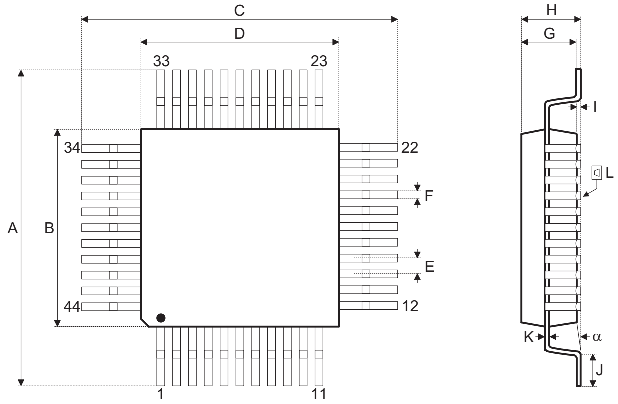



| Symbol | Dimensions in inch | | |
|---|---|---|---|
| | Min. | Nom. | Max. |
| A | 0.228 | — | 0.244 |
| B | 0.150 | — | 0.157 |
| C | 0.008 | — | 0.012 |
| C′ | 0.386 | — | 0.394 |
| D | 0.054 | — | 0.060 |
| E | — | 0.025 | — |
| F | 0.004 | — | 0.010 |
| G | 0.022 | — | 0.028 |
| H | 0.007 | — | 0.010 |
| α | 0° | — | 8° |

| Symbol | Dimensions in mm | | |
|---|---|---|---|
| | Min. | Nom. | Max. |
| A | 5.79 | — | 6.20 |
| B | 3.81 | — | 3.99 |
| C | 0.20 | — | 0.30 |
| C′ | 9.80 | — | 10.01 |
| D | 1.37 | — | 1.52 |
| E | — | 0.64 | — |
| F | 0.10 | — | 0.25 |
| G | 0.56 | — | 0.71 |
| H | 0.18 | — | 0.25 |
| α | 0° | — | 8° |

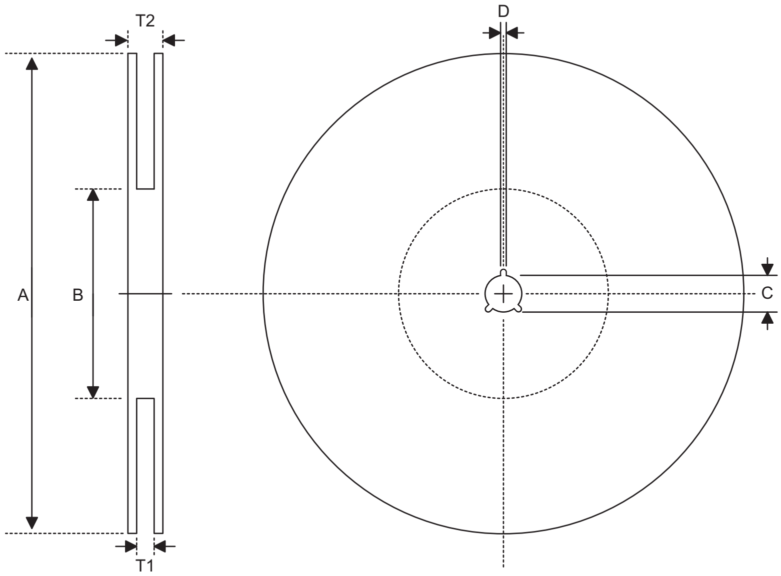
### 44-pin QFP (10mm×10mm) Outline Dimensions




| Symbol | Dimensions in inch | | |
|---|---|---|---|
| | Min. | Nom. | Max. |
| A | 0.512 | — | 0.528 |
| B | 0.390 | — | 0.398 |
| C | 0.512 | — | 0.528 |
| D | 0.390 | — | 0.398 |
| E | — | 0.031 | — |
| F | — | 0.012 | — |
| G | 0.075 | — | 0.087 |
| H | — | — | 0.106 |
| I | 0.010 | — | 0.020 |
| J | 0.029 | — | 0.037 |
| K | 0.004 | — | 0.008 |
| L | — | 0.004 | — |
| α | 0° | — | 7° |

| Symbol | Dimensions in mm | | |
|---|---|---|---|
| | Min. | Nom. | Max. |
| A | 13.00 | — | 13.40 |
| B | 9.90 | — | 10.10 |
| C | 13.00 | — | 13.40 |
| D | 9.90 | — | 10.10 |
| E | — | 0.80 | — |
| F | — | 0.30 | — |
| G | 1.90 | — | 2.20 |
| H | — | — | 2.70 |
| I | 0.25 | — | 0.50 |
| J | 0.73 | — | 0.93 |
| K | 0.10 | — | 0.20 |
| L | — | 0.10 | — |
| α | 0° | — | 7° |

## Reel Dimensions




**SSOP 16S (150mil)**

| Symbol | Description | Dimensions in mm |
|---|---|---|
| A | Reel Outer Diameter | 330.0±1.0 |
| B | Reel Inner Diameter | 100.0±1.5 |
| C | Spindle Hole Diameter | $13.0^{+0.5/-0.2}$ |
| D | Key Slit Width | 2.0±0.5 |
| T1 | Space Between Flange | $12.8^{+0.3/-0.2}$ |
| T2 | Reel Thickness | 18.2±0.2 |

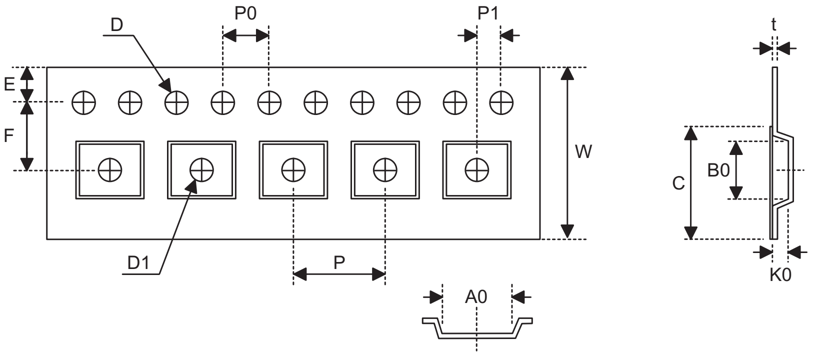
**SOP 20W, SOP 24W, SOP 28W (300mil)**

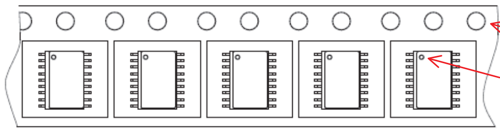
| Symbol | Description | Dimensions in mm |
|---|---|---|
| A | Reel Outer Diameter | 330.0±1.0 |
| B | Reel Inner Diameter | 100.0±1.5 |
| C | Spindle Hole Diameter | $13.0^{+0.5/-0.2}$ |
| D | Key Slit Width | 2.0±0.5 |
| T1 | Space Between Flange | $24.8^{+0.3/-0.2}$ |
| T2 | Reel Thickness | 30.2±0.2 |

**SOP 16N (150mil), SSOP 20S (150mil), SSOP 24S (150mil), SSOP 28S (150mil)**

| Symbol | Description | Dimensions in mm |
|---|---|---|
| A | Reel Outer Diameter | 330.0±1.0 |
| B | Reel Inner Diameter | 100.0±1.5 |
| C | Spindle Hole Diameter | $13.0^{+0.5/-0.2}$ |
| D | Key Slit Width | 2.0±0.5 |
| T1 | Space Between Flange | $16.8^{+0.3/-0.2}$ |
| T2 | Reel Thickness | 22.2±0.2 |

### Carrier Tape Dimensions




**SOP 16N (150mil)**

| Symbol | Description | Dimensions in mm |
|---|---|---|
| W | Carrier Tape Width | 16.0±0.3 |
| P | Cavity Pitch | 8.0±0.1 |
| E | Perforation Position | 1.75±0.1 |
| F | Cavity to Perforation (Width Direction) | 7.5±0.1 |
| D | Perforation Diameter | $1.55^{+0.10/-0.00}$ |
| D1 | Cavity Hole Diameter | $1.50^{+0.25/-0.00}$ |
| P0 | Perforation Pitch | 4.0±0.1 |
| P1 | Cavity to Perforation (Length Direction) | 2.0±0.1 |
| A0 | Cavity Length | 6.5±0.1 |
| B0 | Cavity Width | 10.3±0.1 |
| K0 | Cavity Depth | 2.1±0.1 |
| t | Carrier Tape Thickness | 0.30±0.05 |
| C | Cover Tape Width | 13.3±0.1 |

**SSOP 16S**

| Symbol | Description | Dimensions in mm |
|---|---|---|
| W | Carrier Tape Width | $12.0^{+0.3/-0.1}$ |
| P | Cavity Pitch | 8.0±0.1 |
| E | Perforation Position | 1.75±0.10 |
| F | Cavity to Perforation (Width Direction) | 5.5±0.1 |
| D | Perforation Diameter | 1.55±0.10 |
| D1 | Cavity Hole Diameter | $1.50^{+0.25/-0.00}$ |
| P0 | Perforation Pitch | 4.0±0.1 |
| P1 | Cavity to Perforation (Length Direction) | 2.0±0.1 |
| A0 | Cavity Length | 6.4±0.1 |
| B0 | Cavity Width | 5.2±0.1 |
| K0 | Cavity Depth | 2.1±0.1 |
| t | Carrier Tape Thickness | 0.30±0.05 |
| C | Cover Tape Width | 9.3±0.1 |

**SOP 20W**

| Symbol | Description | Dimensions in mm |
|---|---|---|
| W | Carrier Tape Width | $24.0^{+0.3/-0.1}$ |
| P | Cavity Pitch | 12.0±0.1 |
| E | Perforation Position | 1.75±0.10 |
| F | Cavity to Perforation (Width Direction) | 11.5±0.1 |
| D | Perforation Diameter | $1.5^{+0.1/-0.0}$ |
| D1 | Cavity Hole Diameter | $1.50^{+0.25/-0.00}$ |
| P0 | Perforation Pitch | 4.0±0.1 |
| P1 | Cavity to Perforation (Length Direction) | 2.0±0.1 |
| A0 | Cavity Length | 10.8±0.1 |
| B0 | Cavity Width | 13.3±0.1 |
| K0 | Cavity Depth | 3.2±0.1 |
| t | Carrier Tape Thickness | 0.30±0.05 |
| C | Cover Tape Width | 21.3±0.1 |

**SSOP 20S (150mil)**

| Symbol | Description | Dimensions in mm |
|---|---|---|
| W | Carrier Tape Width | $16.0^{+0.3/-0.1}$ |
| P | Cavity Pitch | 8.0±0.1 |
| E | Perforation Position | 1.75±0.10 |
| F | Cavity to Perforation (Width Direction) | 7.5±0.1 |
| D | Perforation Diameter | $1.5^{+0.1/-0.0}$ |
| D1 | Cavity Hole Diameter | $1.50^{+0.25/-0.00}$ |
| P0 | Perforation Pitch | 4.0±0.1 |
| P1 | Cavity to Perforation (Length Direction) | 2.0±0.1 |
| A0 | Cavity Length | 6.5±0.1 |
| B0 | Cavity Width | 9.0±0.1 |
| K0 | Cavity Depth | 2.3±0.1 |
| t | Carrier Tape Thickness | 0.30±0.05 |
| C | Cover Tape Width | 13.3±0.1 |

**SOP 24W**

| Symbol | Description | Dimensions in mm |
|---|---|---|
| W | Carrier Tape Width | 24.0±0.3 |
| P | Cavity Pitch | 12.0±0.1 |
| E | Perforation Position | 1.75±0.1 |
| F | Cavity to Perforation (Width Direction) | 11.5±0.1 |
| D | Perforation Diameter | $1.55^{+0.10/-0.00}$ |
| D1 | Cavity Hole Diameter | $1.50^{+0.25/-0.00}$ |
| P0 | Perforation Pitch | 4.0±0.1 |
| P1 | Cavity to Perforation (Length Direction) | 2.0±0.1 |
| A0 | Cavity Length | 10.9±0.1 |
| B0 | Cavity Width | 15.9±0.1 |
| K0 | Cavity Depth | 3.1±0.1 |
| t | Carrier Tape Thickness | 0.35±0.05 |
| C | Cover Tape Width | 21.3±0.1 |

SSOP 24S (150mil)

| Symbol | Description | Dimensions in mm |
|---|---|---|
| W | Carrier Tape Width | 16.0 $^{+0.3/-0.1}$ |
| P | Cavity Pitch | 8.0±0.1 |
| E | Perforation Position | 1.75±0.10 |
| F | Cavity to Perforation (Width Direction) | 7.5±0.1 |
| D | Perforation Diameter | 1.5 $^{+0.1/-0.0}$ |
| D1 | Cavity Hole Diameter | 1.50 $^{+0.25/-0.00}$ |
| P0 | Perforation Pitch | 4.0±0.1 |
| P1 | Cavity to Perforation (Length Direction) | 2.0±0.1 |
| A0 | Cavity Length | 6.5±0.1 |
| B0 | Cavity Width | 9.5±0.1 |
| K0 | Cavity Depth | 2.1±0.1 |
| t | Carrier Tape Thickness | 0.30±0.05 |
| C | Cover Tape Width | 13.3±0.1 |

SOP 28W (300mil)

| Symbol | Description | Dimensions in mm |
|---|---|---|
| W | Carrier Tape Width | 24.0±0.3 |
| P | Cavity Pitch | 12.0±0.1 |
| E | Perforation Position | 1.75±0.10 |
| F | Cavity to Perforation (Width Direction) | 11.5±0.1 |
| D | Perforation Diameter | 1.5 $^{+0.1/-0.0}$ |
| D1 | Cavity Hole Diameter | 1.50 $^{+0.25/-0.00}$ |
| P0 | Perforation Pitch | 4.0±0.1 |
| P1 | Cavity to Perforation (Length Direction) | 2.0±0.1 |
| A0 | Cavity Length | 10.85±0.10 |
| B0 | Cavity Width | 18.34±0.10 |
| K0 | Cavity Depth | 2.97±0.10 |
| t | Carrier Tape Thickness | 0.35±0.01 |
| C | Cover Tape Width | 21.3±0.1 |

**SSOP 28S (150mil)**

| Symbol | Description | Dimensions in mm |
|---|---|---|
| W | Carrier Tape Width | 16.0±0.3 |
| P | Cavity Pitch | 8.0±0.1 |
| E | Perforation Position | 1.75±0.1 |
| F | Cavity to Perforation (Width Direction) | 7.5±0.1 |
| D | Perforation Diameter | 1.55 $^{+0.10/-0.00}$ |
| D1 | Cavity Hole Diameter | 1.50 $^{+0.25/-0.00}$ |
| P0 | Perforation Pitch | 4.0±0.1 |
| P1 | Cavity to Perforation (Length Direction) | 2.0±0.1 |
| A0 | Cavity Length | 6.5±0.1 |
| B0 | Cavity Width | 10.3±0.1 |
| K0 | Cavity Depth | 2.1±0.1 |
| t | Carrier Tape Thickness | 0.30±0.05 |
| C | Cover Tape Width | 13.3±0.1 |

**Holtek Semiconductor Inc. (Headquarters)**
No.3, Creation Rd. II, Science Park, Hsinchu, Taiwan
Tel: 886-3-563-1999
Fax: 886-3-563-1189
http://www.holtek.com.tw

**Holtek Semiconductor Inc. (Taipei Sales Office)**
4F-2, No. 3-2, YuanQu St., Nankang Software Park, Taipei 115, Taiwan
Tel: 886-2-2655-7070
Fax: 886-2-2655-7373
Fax: 886-2-2655-7383 (International sales hotline)

**Holtek Semiconductor Inc. (Shenzhen Sales Office)**
5F, Unit A, Productivity Building, No.5 Gaoxin M 2nd Road, Nanshan District, Shenzhen, China 518057
Tel: 86-755-8616-9908, 86-755-8616-9308
Fax: 86-755-8616-9722

**Holtek Semiconductor (USA), Inc. (North America Sales Office)**
46729 Fremont Blvd., Fremont, CA 94538, USA
Tel: 1-510-252-9880
Fax: 1-510-252-9885
http://www.holtek.com