HOLTEK

Touch Flash MCU

BS83B24C/BS83C40C

Revision: V1.00   Date: February 07, 2018

www.holtek.com

Table of Contents

Features 6
CPU Features 6
Peripheral Features 6
General Description 7
Selection Table 7
Block Diagram 8
Pin Assignment 8
Pin Description 9
Absolute Maximum Ratings 17
D.C. Characteristics 18
Operating Voltage Characteristics 18
Standby Current Characteristics 18
Operating Current Characteristics 19
A.C. Characteristics 20
High Speed Internal Oscillator – HIRC – Frequency Accuracy 20
Low Speed Internal Oscillator Characteristics – LIRC 20
Low Speed Crystal Oscillator Characteristics – LXT 20
Operating Frequency Characteristic Curves 21
System Start Up Time Characteristics 21
Input/Output Characteristics 22
Memory Characteristics 23
LVR Electrical Characteristics 23
Power-on Reset Characteristics 23
System Architecture 24
Clocking and Pipelining 24
Program Counter 25
Stack 25
Arithmetic and Logic Unit – ALU 26
Flash Program Memory 27
Structure 27
Special Vectors 27
Look-up Table 27
Table Program Example 28
In Circuit Programming – ICP 29
On-Chip Debug Support – OCDS 30
Data Memory 31
Structure 31
Data Memory Addressing 32
General Purpose Data Memory 32
Special Purpose Data Memory 32

Special Function Register Description 35
Indirect Addressing Registers – IAR0, IAR1, IAR2 35
Memory Pointers – MP0, MP1L, MP1H, MP2L, MP2H 35
Accumulator – ACC 36
Program Counter Low Register – PCL 37
Look-up Table Registers – TBLP, TBHP, TBLH 37
Status Register – STATUS 37

EEPROM Data Memory 39
EEPROM Data Memory Structure 39
EEPROM Registers 39
Reading Data from the EEPROM 40
Writing Data to the EEPROM 41
Write Protection 41
EEPROM Interrupt 41
Programming Considerations 41

Oscillators 43
Oscillator Overview 43
System Clock Configurations 43
Internal RC Oscillator – HIRC 44
External 32.768kHz Crystal Oscillator – LXT 44
Internal 32kHz Oscillator – LIRC 45

Operating Modes and System Clocks 46
System Clocks 46
System Operation Modes 47
Control Registers 48
Operating Mode Switching 50
Standby Current Considerations 54
Wake-up 54

Watchdog Timer 55
Watchdog Timer Clock Source 55
Watchdog Timer Control Register 55
Watchdog Timer Operation 56

Reset and Initialisation 57
Reset Functions 57
Reset Initial Conditions 60

Input/Output Ports 65
Pull-high Resistors 66
Port A Wake-up 66
I/O Port Control Registers 67
I/O Port Source Current Control 67
Pin-shared Functions 69
I/O Pin Structures 75
Programming Considerations 75

Timer Modules – TM ... 76
Introduction ... 76
TM Operation ... 76
TM Clock Source ... 76
TM Interrupts ... 77
TM External Pins ... 77
Programming Considerations ... 78

Compact Type TM – CTM ... 79
Compact TM Operation ... 79
Compact Type TM Register Description ... 79
Compact Type TM Operating Modes ... 83

Periodic Type TM – PTM ... 89
Periodic TM Operation ... 89
Periodic Type TM Register Description ... 90
Periodic Type TM Operation Modes ... 94

Universal Serial Interface Module – USIM ... 103
SPI Interface ... 103
I^{2}C Interface ... 111
UART Interface ... 121

Touch Key Function ... 136
Touch Key Structure ... 136
Touch Key Register Definition ... 138
Touch Key Operation ... 144
Touch Key Interrupt ... 150
Programming Considerations ... 150

Interrupts ... 151
Interrupt Registers ... 151
Interrupt Operation ... 154
External Interrupt ... 156
Multi-function Interrupt ... 156
Timer Module Interrupts ... 157
EEPROM Interrupt ... 157
USIM Interrupt ... 157
Touch Key Interrupt ... 158
Time Base Interrupts ... 158
Interrupt Wake-up Function ... 159
Programming Considerations ... 160

Configuration Options ... 160

Application Circuits ... 161

Instruction Set ... 162
Introduction ... 162
Instruction Timing ... 162
Moving and Transferring Data ... 162

Arithmetic Operations.. 162
Logical and Rotate Operation 163
Branches and Control Transfer 163
Bit Operations 163
Table Read Operations 163
Other Operations.. 163

Instruction Set Summary 164
Table Conventions.. 164
Extended Instruction Set 166

Instruction Definition.. 168
Extended Instruction Definition 177

Package Information 184
28-pin SOP (300mil) Outline Dimensions 185
28-pin SSOP (150mil) Outline Dimensions 186
44-pin LQFP (10mm×10mm) (FP2.0mm) Outline Dimensions 187

Features

CPU Features

- Operating Voltage
  - f_{SYS}=8MHz: 2.2V~5.5V
  - f_{SYS}=12MHz: 2.7V~5.5V
  - f_{SYS}=16MHz: 3.3V~5.5V
- Up to 0.25μs instruction cycle with 16MHz system clock at V_{DD}=5V
- Power down and wake-up functions to reduce power consumption
- Oscillator types:
  - Internal High Speed 8/12/16MHz RC Oscillator – HIRC
  - Internal Low Speed 32kHz RC Oscillator– LIRC
  - External Low Speed 32.768kHz Crystal – LXT
- Fully integrated internal oscillators require no external components
- Multi-mode operation: FAST, SLOW, IDLE and SLEEP
- All instructions executed in 1~3 instruction cycles
- Table read instructions
- 115 powerful instructions
- 6-level subroutine nesting
- Bit manipulation instruction

Peripheral Features

- Flash Program Memory: Up to 4K×16
- RAM Data Memory: Up to 768×8
- True EEPROM Memory: 128×8
- Up to 40 touch keys – fully integrated without requiring external components
- Watchdog Timer function
- Up to 42 bidirectional I/O lines
- Programmable I/O source current
- Single external interrupt line shared with I/O pin
- Multiple Timer Modules for time measurement, input capture, compare match output, PWM output function or single pulse output function
- Dual Time-Base functions for generation of fixed time interrupt signals
- Universial Serial Interface Module – USIM for SPI, I^2C or UART communication
- Low voltage reset function – LVR
- Wide range of package types

General Description

The series of devices are the Flash Memory 8-bit high performance RISC architecture microcontrollers with fully integrated touch key functions. With the touch key function provided internally and with the convenience of Flash Memory multi-programming features, this series of devices have all the features to offer designer a reliable and easy means of implementing touch keys with their product applications.

The Touch key function is completely integrated eliminating the need for external components. In addition to the Flash program memory, other memory includes an area of RAM Data Memory as well as an area of true EEPROM memory for storage of non-volatile data such as serial numbers, calibration data etc. Protective features such as an internal Watchdog Timer and Low Voltage Reset coupled with excellent noise immunity and ESD protection ensure that reliable operation is maintained in hostile electrical environments.

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

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

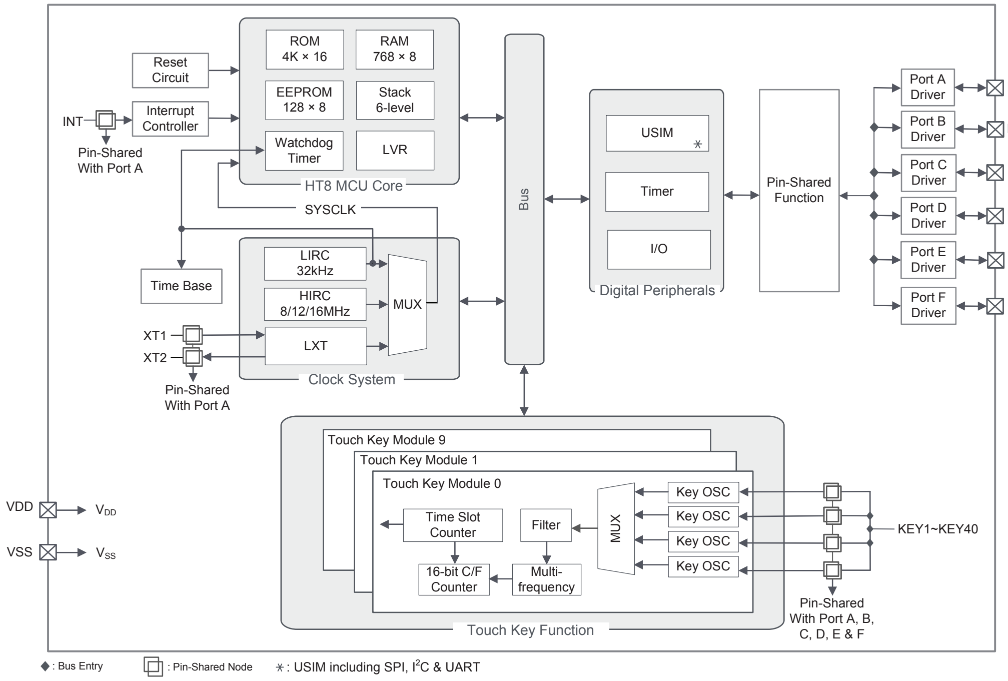
Selection Table

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

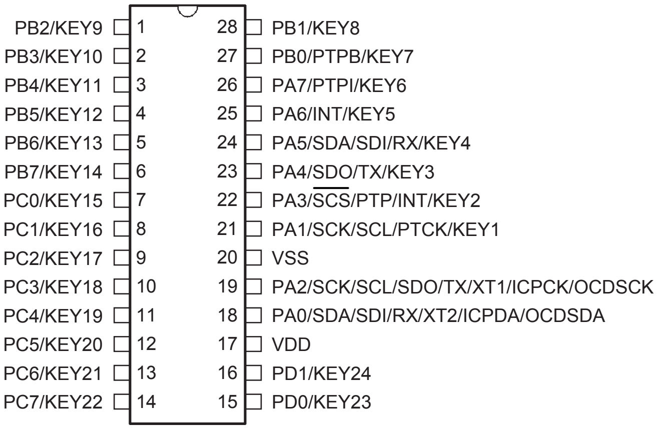
Part No.	Program Memory	Data Memory	Data EEPROM	I/O	External Interrupt
BS83B24C	3K × 16	512 × 8	128 × 8	26	1
BS83C40C	4K × 16	768 × 8	128 × 8	42	1

Part No.	Timer Module	Time Base	Touch key	USIM	Stacks	Package
BS83B24C	10-bit PTM × 1	2	24	√	6	28SOP 28SSOP
BS83C40C	10-bit CTM × 1 10-bit PTM × 1	2	40	√	6	44LQFP

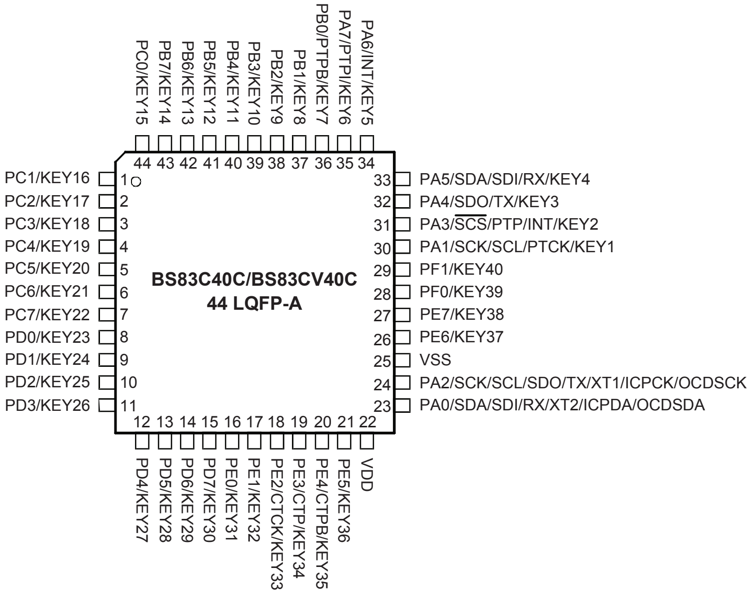
Block Diagram


ROM
4K × 16
RAM
768 × 8
Reset
Circuit
EEPROM
128 × 8
Stack
6-level
INT
Interrupt
Controller
Watchdog
Timer
LVR
Pin-Shared
With Port A
HT8 MCU Core
SYSCLK
Bus
USIM
Timer
I/O
Digital Peripherals
Pin-Shared
Function
Port A
Driver
PA0~PA7
Port B
Driver
PB0~PB7
Port C
Driver
PC0~PC7
Port D
Driver
PD0~PD7
Port E
Driver
PE0~PE7
Port F
Driver
PF0~PF1
LIRC
32kHz
HIRC
8/12/16MHz
MUX
Time Base
XT1
XT2
LXT
Pin-Shared
With Port A
Clock System
Touch Key Module 9
Touch Key Module 1
Touch Key Module 0
Time Slot
Counter
Filter
MUX
Key OSC
Key OSC
Key OSC
Key OSC
16-bit C/F
Counter
Multi-
frequency
KEY1~KEY40
Pin-Shared
With Port A, B,
C, D, E & F
Touch Key Function
VDD
V_{DD}
VSS
V_{SS}
◆ : Bus Entry
: Pin-Shared Node
*: USIM including SPI, I²C & UART

Pin Assignment


PB2/KEY9 1
PB3/KEY10 2
PB4/KEY11 3
PB5/KEY12 4
PB6/KEY13 5
PB7/KEY14 6
PC0/KEY15 7
PC1/KEY16 8
PC2/KEY17 9
PC3/KEY18 10
PC4/KEY19 11
PC5/KEY20 12
PC6/KEY21 13
PC7/KEY22 14
28 PB1/KEY8
27 PB0/PTPB/KEY7
26 PA7/PTPI/KEY6
25 PA6/INT/KEY5
24 PA5/SDA/SDI/RX/KEY4
23 PA4/SDO/TX/KEY3
22 PA3/$\overline{SCS}$/PTP/INT/KEY2
21 PA1/SCK/SCL/PTCK/KEY1
20 VSS
19 PA2/SCK/SCL/SDO/TX/XT1/ICPCK/OCDSCK
18 PA0/SDA/SDI/RX/XT2/ICPDA/OCDSDA
17 VDD
16 PD1/KEY24
15 PD0/KEY23

BS83B24C/BS83BV24C
28SOP-A/SSOP-A


PC0/KEY15
PB7/KEY14
PB6/KEY13
PB5/KEY12
PB4/KEY11
PB3/KEY10
PB2/KEY9
PB1/KEY8
PB0/PTPB/KEY7
PA7/PTPI/KEY6
PA6/INT/KEY5
44 43 42 41 40 39 38 37 36 35 34
PC1/KEY16 1
PC2/KEY17 2
PC3/KEY18 3
PC4/KEY19 4
PC5/KEY20 5
PC6/KEY21 6
PC7/KEY22 7
PD0/KEY23 8
PD1/KEY24 9
PD2/KEY25 10
PD3/KEY26 11
33 PA5/SDA/SDI/RX/KEY4
32 PA4/SDO/TX/KEY3
31 PA3/$\overline{SCS}$/PTP/INT/KEY2
30 PA1/SCK/SCL/PTCK/KEY1
29 PF1/KEY40
28 PF0/KEY39
27 PE7/KEY38
26 PE6/KEY37
25 VSS
24 PA2/SCK/SCL/SDO/TX/XT1/ICPCK/OCDSCK
23 PA0/SDA/SDI/RX/XT2/ICPDA/OCDSDA
BS83C40C/BS83CV40C
44 LQFP-A
12 13 14 15 16 17 18 19 20 21 22
PD4/KEY27
PD5/KEY28
PD6/KEY29
PD7/KEY30
PE0/KEY31
PE1/KEY32
PE2/CTCK/KEY33
PE3/CTP/KEY34
PE4/CTPB/KEY35
PE5/KEY36
VDD

Notes: 1. If the pin-shared pin functions have multiple outputs, the desired pin-shared function is determined by the corresponding software control bits.

2. The OCDSDA and OCDSCK pins are supplied as OCDS dedicated pins and as such only available for the BS83BV24C and the BS83CV40C devices which are the OCDS EV chips for the BS83B24C and the BS83C40C devices respectively.

Pin Description

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

BS83B24C

Pin Name	Function	OPT	I/T	O/T	Description
PA0/SDA/SDI/ RX/XT2/ICPDA/ OCDSDA	PA0	PAWU PAPU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	SDA	PAS0 IFS	ST	NMOS	I^{2}C data line
	SDI	PAS0 IFS	ST	—	SPI serial data input
	RX	PAS0 IFS	ST	—	UART serial data input
	XT2	PAS0	—	LXT	LXT oscillator pin
	ICPDA	—	ST	CMOS	ICP data/address
	OCDSDA	—	ST	CMOS	OCDS data/address, for EV chip only

Pin Name	Function	OPT	I/T	O/T	Description
PA1/SCK/SCL/ PTCK/KEY1	PA1	PAWU PAPU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	SCK	PAS0 IFS	ST	CMOS	SPI serial clock
	SCL	PAS0 IFS	ST	NMOS	I^{2}C clock line
	PTCK	PAS0	ST	—	PTM capture input
	KEY1	PAS0 TKM0C1	NSI	—	Touch key input 1
PA2/SCK/SCL/ SDO/TX/XT1/ ICPCK/OCDSCK	PA2	PAWU PAPU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	SCK	PAS0 IFS	ST	CMOS	SPI serial clock
	SCL	PAS0 IFS	ST	NMOS	I^{2}C clock line
	SDO	PAS0	—	CMOS	SPI serial data output
	TX	PAS0	—	CMOS	UART serial data output
	XT1	PAS0	LXT	—	LXT oscillator pin
	ICPCK	—	ST	—	ICP clock input
	OCDSCK	—	ST	—	OCDS clock input, for EV chip only
PA3/$\overline{SCS}$/PTP/ INT/KEY2	PA3	PAWU PAPU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	$\overline{SCS}$	PAS0	ST	—	SPI slave select pin
	PTP	PAS0	—	CMOS	PTM output
	INT	PAS0 INTEG INTC0 IFS	ST	—	External interrupt
	KEY2	PAS0 TKM0C1	NSI	—	Touch key input 2
PA4/SDO/TX/ KEY3	PA4	PAWU PAPU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	SDO	PAS1	—	CMOS	SPI serial data output
	TX	PAS1	—	CMOS	UART serial data output
	KEY3	PAS1 TKM0C1	NSI	—	Touch key input 3
PA5/SDA/SDI/RX/ KEY4	PA5	PAWU PAPU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	SDA	PAS1 IFS	ST	NMOS	I^{2}C data line
	SDI	PAS1 IFS	ST	—	SPI serial data input
	RX	PAS1 IFS	ST	—	UART serial data input
	KEY4	PAS1 TKM0C1	NSI	—	Touch key input 4

Pin Name	Function	OPT	I/T	O/T	Description
PA6/INT/KEY5	PA6	PAWU PAPU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	INT	PAS1 INTEG INTC0 IFS	ST	—	External interrupt
	KEY5	PAS1 TKM1C1	NSI	—	Touch key input 5
PA7/PTPI/KEY6	PA7	PAWU PAPU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	PTPI	PAS1	ST	—	PTM capture input
	KEY6	PAS1 TKM1C1	NSI	—	Touch key input 6
PB0/PTPB/KEY7	PB0	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-up.
	PTPB	PBS0	—	CMOS	PTM inverting output
	KEY7	PBS0 TKM1C1	NSI	—	Touch key input 7
PB1/KEY8	PB1	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-up.
	KEY8	PBS0 TKM1C1	NSI	—	Touch key input 8
PB2/KEY9	PB2	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-up.
	KEY9	PBS0 TKM2C1	NSI	—	Touch key input 9
PB3/KEY10	PB3	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-up.
	KEY10	PBS0 TKM2C1	NSI	—	Touch key input 10
PB4/KEY11	PB4	PBPU PBS1	ST	CMOS	General purpose I/O. Register enabled pull-up.
	KEY11	PBS1 TKM2C1	NSI	—	Touch key input 11
PB5/KEY12	PB5	PBPU PBS1	ST	CMOS	General purpose I/O. Register enabled pull-up.
	KEY12	PBS1 TKM2C1	NSI	—	Touch key input 12
PB6/KEY13	PB6	PBPU PBS1	ST	CMOS	General purpose I/O. Register enabled pull-up.
	KEY13	PBS1 TKM3C1	NSI	—	Touch key input 13
PB7/KEY14	PB7	PBPU PBS1	ST	CMOS	General purpose I/O. Register enabled pull-up.
	KEY14	PBS1 TKM3C1	NSI	—	Touch key input 14
PC0/KEY15	PC0	PCPU PCS0	ST	CMOS	General purpose I/O. Register enabled pull-up.
	KEY15	PCS0 TKM3C1	NSI	—	Touch key input 15

Pin Name	Function	OPT	I/T	O/T	Description
PC1/KEY16	PC1	PCPU PCS0	ST	CMOS	General purpose I/O. Register enabled pull-up.
	KEY16	PCS0 TKM3C1	NSI	—	Touch key input 16
PC2/KEY17	PC2	PCPU PCS0	ST	CMOS	General purpose I/O. Register enabled pull-up.
	KEY17	PCS0 TKM4C1	NSI	—	Touch key input 17
PC3/KEY18	PC3	PCPU PCS0	ST	CMOS	General purpose I/O. Register enabled pull-up.
	KEY18	PCS0 TKM4C1	NSI	—	Touch key input 18
PC4/KEY19	PC4	PCPU PCS1	ST	CMOS	General purpose I/O. Register enabled pull-up.
	KEY19	PCS1 TKM4C1	NSI	—	Touch key input 19
PC5/KEY20	PC5	PCPU PCS1	ST	CMOS	General purpose I/O. Register enabled pull-up.
	KEY20	PCS1 TKM4C1	NSI	—	Touch key input 20
PC6/KEY21	PC6	PCPU PCS1	ST	CMOS	General purpose I/O. Register enabled pull-up.
	KEY21	PCS1 TKM5C1	NSI	—	Touch key input 21
PC7/KEY22	PC7	PCPU PCS1	ST	CMOS	General purpose I/O. Register enabled pull-up.
	KEY22	PCS1 TKM5C1	NSI	—	Touch key input 22
PD0/KEY23	PD0	PDPU PDS0	ST	CMOS	General purpose I/O. Register enabled pull-up.
	KEY23	PDS0 TKM5C1	NSI	—	Touch key input 23
PD1/KEY24	PD1	PDPU PDS0	ST	CMOS	General purpose I/O. Register enabled pull-up.
	KEY24	PDS0 TKM5C1	NSI	—	Touch key input 24
VDD	VDD	—	PWR	—	Positive power supply
VSS	VSS	—	PWR	—	Negative power supply

Legend: I/T: Input type
OPT: Optional by register option
CMOS: CMOS output
AN: Analog signal
LXT: Low frequency crystal oscillator
O/T: Output type
ST: Schmitt Trigger input
NMOS: NMOS output
PWR: Power
NSI: Non-standard input

BS83C40C

Pin Name	Function	OPT	I/T	O/T	Description
PA0/SDA/SDI/RX/XT2/ICPDA/OCDSDA	PA0	PAWU PAPU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	SDA	PAS0 IFS	ST	NMOS	I^{2}C data line
	SDI	PAS0 IFS	ST	—	SPI serial data input
	RX	PAS0 IFS	ST	—	UART serial data input
	XT2	PAS0	—	LXT	LXT oscillator pin
	ICPDA	—	ST	CMOS	ICP data/address
	OCDSDA	—	ST	CMOS	OCDS data/address, for EV chip only
PA1/SCK/SCL/PTCK/KEY1	PA1	PAWU PAPU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	SCK	PAS0 IFS	ST	CMOS	SPI serial clock
	SCL	PAS0 IFS	ST	NMOS	I^{2}C clock line
	PTCK	PAS0	ST	—	PTM capture input
	KEY1	PAS0 TKM0C1	NSI	—	Touch key input 1
PA2/SCK/SCL/SDO/TX/XT1/ICPCK/OCDSCK	PA2	PAWU PAPU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	SCK	PAS0 IFS	ST	CMOS	SPI serial clock
	SCL	PAS0 IFS	ST	NMOS	I^{2}C clock line
	SDO	PAS0	—	CMOS	SPI serial data output
	TX	PAS0	—	CMOS	UART serial data output
	XT1	PAS0	LXT	—	LXT oscillator pin
	ICPCK	—	ST	—	ICP clock input
	OCDSCK	—	ST	—	OCDS clock input, for EV chip only
PA3/$\overline{\text{SCS}}$/PTP/INT/KEY2	PA3	PAWU PAPU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	$\overline{\text{SCS}}$	PAS0	ST	—	SPI slave select pin
	PTP	PAS0	—	CMOS	PTM output
	INT	PAS0 INTEG INTC0 IFS	ST	—	External interrupt
	KEY2	PAS0 TKM0C1	NSI	—	Touch key input 2
PA4/SDO/TX/KEY3	PA4	PAWU PAPU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	SDO	PAS1	—	CMOS	SPI serial data output
	TX	PAS1	—	CMOS	UART serial data output
	KEY3	PAS1 TKM0C1	NSI	—	Touch key input 3

Pin Name	Function	OPT	I/T	O/T	Description
PA5/SDA/SDI/RX/KEY4	PA5	PAWU PAPU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	SDA	PAS1 IFS	ST	NMOS	I^{2}C data line
	SDI	PAS1 IFS	ST	—	SPI serial data input
	RX	PAS1 IFS	ST	—	UART serial data input
	KEY4	PAS1 TKM0C1	NSI	—	Touch key input 4
PA6/INT/KEY5	PA6	PAWU PAPU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	INT	PAS1 INTEG INTC0 IFS	ST	—	External interrupt
	KEY5	PAS1 TKM1C1	NSI	—	Touch key input 5
PA7/PTPI/KEY6	PA7	PAWU PAPU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	PTPI	PAS1	ST	—	PTM capture input
	KEY6	PAS1 TKM1C1	NSI	—	Touch key input 6
PB0/PTPB/KEY7	PB0	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-up.
	PTPB	PBS0	—	CMOS	PTM inverting output
	KEY7	PBS0 TKM1C1	NSI	—	Touch key input 7
PB1/ KEY8	PB1	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-up.
	KEY8	PBS0 TKM1C1	NSI	—	Touch key input 8
PB2/KEY9	PB2	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-up.
	KEY9	PBS0 TKM2C1	NSI	—	Touch key input 9
PB3/KEY10	PB3	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-up.
	KEY10	PBS0 TKM2C1	NSI	—	Touch key input 10
PB4/KEY11	PB4	PBPU PBS1	ST	CMOS	General purpose I/O. Register enabled pull-up.
	KEY11	PBS1 TKM2C1	NSI	—	Touch key input 11
PB5/KEY12	PB5	PBPU PBS1	ST	CMOS	General purpose I/O. Register enabled pull-up.
	KEY12	PBS1 TKM2C1	NSI	—	Touch key input 12
PB6/KEY13	PB6	PBPU PBS1	ST	CMOS	General purpose I/O. Register enabled pull-up.
	KEY13	PBS1 TKM3C1	NSI	—	Touch key input 13

Pin Name	Function	OPT	I/T	O/T	Description
PB7/KEY14	PB7	PBPU PBS1	ST	CMOS	General purpose I/O. Register enabled pull-up.
	KEY14	PBS1 TKM3C1	NSI	—	Touch key input 14
PC0/KEY15	PC0	PCPU PCS0	ST	CMOS	General purpose I/O. Register enabled pull-up.
	KEY15	PCS0 TKM3C1	NSI	—	Touch key input 15
PC1/KEY16	PC1	PCPU PCS0	ST	CMOS	General purpose I/O. Register enabled pull-up.
	KEY16	PCS0 TKM3C1	NSI	—	Touch key input 16
PC2/KEY17	PC2	PCPU PCS0	ST	CMOS	General purpose I/O. Register enabled pull-up.
	KEY17	PCS0 TKM4C1	NSI	—	Touch key input 17
PC3/KEY18	PC3	PCPU PCS0	ST	CMOS	General purpose I/O. Register enabled pull-up.
	KEY18	PCS0 TKM4C1	NSI	—	Touch key input 18
PC4/KEY19	PC4	PCPU PCS1	ST	CMOS	General purpose I/O. Register enabled pull-up.
	KEY19	PCS1 TKM4C1	NSI	—	Touch key input 19
PC5/ KEY20	PC5	PCPU PCS1	ST	CMOS	General purpose I/O. Register enabled pull-up.
	KEY20	PCS1 TKM4C1	NSI	—	Touch key input 20
PC6/ KEY21	PC6	PCPU PCS1	ST	CMOS	General purpose I/O. Register enabled pull-up.
	KEY21	PCS1 TKM5C1	NSI	—	Touch key input 21
PC7/KEY22	PC7	PCPU PCS1	ST	CMOS	General purpose I/O. Register enabled pull-up.
	KEY22	PCS1 TKM5C1	NSI	—	Touch key input 22
PD0/KEY23	PD0	PDPU PDS0	ST	CMOS	General purpose I/O. Register enabled pull-up.
	KEY23	PDS0 TKM5C1	NSI	—	Touch key input 23
PD1/KEY24	PD1	PDPU PDS0	ST	CMOS	General purpose I/O. Register enabled pull-up.
	KEY24	PDS0 TKM5C1	NSI	—	Touch key input 24
PD2/KEY25	PD2	PDPU PDS0	ST	CMOS	General purpose I/O. Register enabled pull-up.
	KEY25	PDS0 TKM6C1	NSI	—	Touch key input 25
PD3/KEY26	PD3	PDPU PDS0	ST	CMOS	General purpose I/O. Register enabled pull-up.
	KEY26	PDS0 TKM6C1	NSI	—	Touch key input 26

Pin Name	Function	OPT	I/T	O/T	Description
PD4/KEY27	PD4	PDPU PDS1	ST	CMOS	General purpose I/O. Register enabled pull-up.
	KEY27	PDS1 TKM6C1	NSI	—	Touch key input 27
PD5/KEY28	PD5	PDPU PDS1	ST	CMOS	General purpose I/O. Register enabled pull-up.
	KEY28	PDS1 TKM6C1	NSI	—	Touch key input 28
PD6/KEY29	PD6	PDPU PDS1	ST	CMOS	General purpose I/O. Register enabled pull-up.
	KEY29	PDS1 TKM7C1	NSI	—	Touch key input 29
PD7/KEY30	PD7	PDPU PDS1	ST	CMOS	General purpose I/O. Register enabled pull-up.
	KEY30	PDS1 TKM7C1	NSI	—	Touch key input 30
PE0/KEY31	PE0	PEPU PES0	ST	CMOS	General purpose I/O. Register enabled pull-up.
	KEY31	PES0 TKM7C1	NSI	—	Touch key input 31
PE1/KEY32	PE1	PEPU PES0	ST	CMOS	General purpose I/O. Register enabled pull-up.
	KEY32	PES0 TKM7C1	NSI	—	Touch key input 32
PE2/CTCK/KEY33	PE2	PEPU PES0	ST	CMOS	General purpose I/O. Register enabled pull-up.
	CTCK	PES0	ST	—	CTM clock input
	KEY33	PES0 TKM8C1	NSI	—	Touch key input 33
PE3/CTP/KEY34	PE3	PEPU PES0	ST	CMOS	General purpose I/O. Register enabled pull-up.
	CTP	PES0	—	CMOS	CTM output
	KEY34	PES0 TKM8C1	NSI	—	Touch key input 34
PE4/CTPB/KEY35	PE4	PEPU PES1	ST	CMOS	General purpose I/O. Register enabled pull-up.
	CTPB	PES1	—	CMOS	CTM inverting output
	KEY35	PES1 TKM8C1	NSI	—	Touch key input 35
PE5/KEY36	PE5	PEPU PES1	ST	CMOS	General purpose I/O. Register enabled pull-up.
	KEY36	PES1 TKM8C1	NSI	—	Touch key input 36
PE6/KEY37	PE6	PEPU PES1	ST	CMOS	General purpose I/O. Register enabled pull-up.
	KEY37	PES1 TKM9C1	NSI	—	Touch key input 37
PE7/KEY38	PE7	PEPU PES1	ST	CMOS	General purpose I/O. Register enabled pull-up.
	KEY38	PES1 TKM9C1	NSI	—	Touch key input 38

Pin Name	Function	OPT	I/T	O/T	Description
PF0/KEY39	PF0	PFPU PFS0	ST	CMOS	General purpose I/O. Register enabled pull-up.
	KEY39	PFS0 TKM9C1	NSI	—	Touch key input 39
PF1/KEY40	PF1	PFPU PFS0	ST	CMOS	General purpose I/O. Register enabled pull-up.
	KEY40	PFS0 TKM9C1	NSI	—	Touch key input 40
VDD	VDD	—	PWR	—	Positive power supply
VSS	VSS	—	PWR	—	Negative power supply

Legend: I/T: Input type
OPT: Optional by register option
CMOS: CMOS output
AN: Analog signal
LXT: Low frequency crystal oscillator
O/T: Output type
ST: Schmitt Trigger input
NMOS: NMOS output
PWR: Power
NSI: Non-standard input

Absolute Maximum Ratings

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

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

D.C. Characteristics

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

Operating Voltage Characteristics

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
V_{DD}	Operating Voltage – HIRC	f_{SYS}=8MHz	2.2	—	5.5	V
		f_{SYS}=12MHz	2.7	—	5.5	V
		f_{SYS}=16MHz	3.3	—	5.5	V
	Operating Voltage – LXT	f_{SYS}=32768Hz	2.2	—	5.5	V
	Operating Voltage – LIRC	f_{SYS}=32kHz	2.2	—	5.5	V

Standby Current Characteristics

Ta=25°C

Symbol	Standby Mode	Test Conditions		Min.	Typ.	Max.	Max.	Unit
		V_{DD}	Conditions				85°C	
I_{STB}	SLEEP Mode	2.2V	WDT on	—	1.2	2.4	2.9	μA
		3V		—	1.5	3.0	3.6	
		5V		—	3.0	5.0	6.0	
	IDLE0 Mode – LIRC	2.2V	f_{SUB} on	—	2.4	4.0	4.8	μA
		3V		—	3.0	5.0	6.0	
		5V		—	5.0	10	12	
	IDLE0 Mode – LXT	2.2V	f_{SUB} on	—	2.4	4.0	4.8	μA
		3V		—	3.0	5.0	6.0	
		5V		—	5.0	10	12	
	IDLE1 Mode – HIRC	2.2V	f_{SUB} on, f_{SYS}=8MHz	—	288	400	480	μA
		3V		—	360	500	600	
		5V		—	600	800	960	
		2.7V	f_{SUB} on, f_{SYS}=12MHz	—	432	600	720	μA
		3V		—	540	750	900	
		5V		—	800	1200	1440	
		3.3V	f_{SUB} on, f_{SYS}=16MHz	—	1.1	1.6	1.9	mA
		5V		—	1.4	2.0	2.4	

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

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

Operating Current Characteristics

Ta=25°C

Symbol	Operating Mode	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{DD}	SLOW Mode – LIRC	2.2V	f_{SYS}=32kHz	—	8	16	μA
		3V		—	10	20	
		5V		—	30	50	
	SLOW Mode – LXT	2.2V	f_{SYS}=32768Hz	—	8	16	μA
		3V		—	10	20	
		5V		—	30	50	
	FAST Mode – HIRC	2.2V	f_{SYS}=8MHz	—	0.6	1.0	mA
		3V		—	0.8	1.2	
		5V		—	1.6	2.4	
		2.7V	f_{SYS}=12MHz	—	1.0	1.4	
		3V		—	1.2	1.8	
		5V		—	2.4	3.6	
		3.3V	f_{SYS}=16MHz	—	3.0	4.5	
		5V		—	4.0	6.0	

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

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

A.C. Characteristics

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

High Speed Internal Oscillator – HIRC – Frequency Accuracy

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

8/12/16MHz

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Temp.				
f_{HIRC}	8MHz writer trimmed HIRC frequency	3V/5V	25°C	-1%	8	+1%	MHz
			-40°C ~ 85°C	-2%	8	+2%	
		2.2V~5.5V	25°C	-2.5%	8	+2.5%	
			-40°C ~ 85°C	-3%	8	+3%	
	12MHz writer trimmed HIRC frequency	3V/5V	25°C	-1%	12	+1%	MHz
			-40°C ~ 85°C	-2%	12	+2%	
		2.7V~5.5V	25°C	-2.5%	12	+2.5%	
			-40°C ~ 85°C	-3%	12	+3%	
	16MHz writer trimmed HIRC frequency	5V	25°C	-1%	16	+1%	MHz
			-40°C ~ 85°C	-2%	16	+2%	
		3.3V~5.5V	25°C	-2.5%	16	+2.5%	
			-40°C ~ 85°C	-3%	16	+3%	

Notes: 1. The 3V/5V values for V_{DD} are provided as these are the two selectable fixed voltages at which the HIRC frequency is trimmed by the writer.

2. The row below the 3V/5V trim voltage row is provided to show the values for the full V_{DD} range operating voltage. It is recommended that the trim voltage is fixed at 3V for application voltage ranges from 2.2V to 3.6V and fixed at 5V for application voltage ranges from 3.3V to 5.5V.

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

Low Speed Internal Oscillator Characteristics – LIRC

Ta=25°C, unless otherwise specified

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

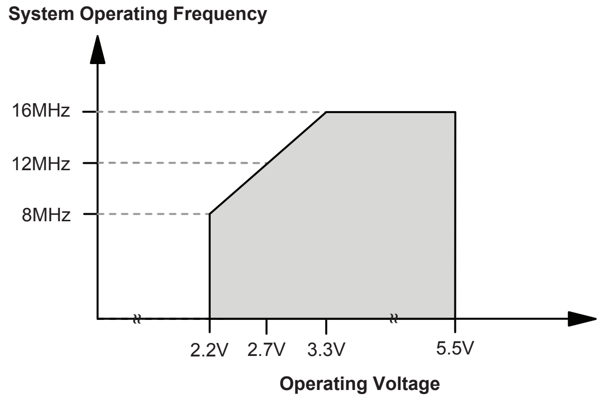
Low Speed Crystal Oscillator Characteristics – LXT

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Temp.				
f_{LXT}	Oscillator Frequency	2.2V~5.5V	-40°C~85°C	—	32768	—	Hz
t_{START}	Start-up Time	3V/5V	—	—	—	500	ms
Duty Cycle	Duty Cycle	—	—	45	50	55	%
R_{NEG}	Negative Resistance	2.2V	—	3×ESR	—	—	Ω

Note : C1, C2 and R_P are external components. C1=C2=10pF, R_P=R_U=10MΩ, C_L=7pF, ESR=30kΩ.

Operating Frequency Characteristic Curves



System Start Up Time Characteristics

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
t_{SST}	System Start-up Time Wake-up from Condition where f_{SYS} is Off	—	$f_{SYS}=f_H$ ~ $f_H/64$, $f_H=f_{HIRC}$	—	16	—	t_{HIRC}
		—	$f_{SYS}=f_{SUB}=f_{LXT}$	—	1024	—	t_{LXT}
		—	$f_{SYS}=f_{SUB}=f_{LIRC}$	—	2	—	t_{LIRC}
	System Start-up Time Wake-up from Condition where f_{SYS} is On	—	$f_{SYS}=f_H$ ~ $f_H/64$, $f_H=f_{HIRC}$	—	2	—	t_H
		—	$f_{SYS}=f_{SUB}=f_{LXT}$ or f_{LIRC}	—	2	—	t_{SUB}
	System Speed Switch Time FAST to SLOW Mode or SLOW to FAST Mode	—	f_{HIRC} switches from off → on	—	16	—	t_{HIRC}
		—	f_{LXT} switches from off → on	—	1024	—	t_{LXT}
t_{RSTD}	System Reset Delay Time Reset Source from Power-on Reset or LVR Hardware Reset	—	RR_{POR}=5V/ms	42	48	54	ms
	System Reset Delay Time LVRC/WDTC/RSTC Software Reset	—	—				
	System Reset Delay Time Reset Source from WDT Overflow	—	—	14	16	18	ms
t_{SRESET}	Minimum Software Reset Pulse Width to Reset	—	—	45	90	120	μs

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

Input/Output Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{IL}	Input Low Voltage for I/O Ports	5V	—	0	—	1.5	V
		—	—	0	—	$0.2V_{DD}$	
V_{IH}	Input High Voltage for I/O Ports	5V	—	3.5	—	5	V
		—	—	$0.8V_{DD}$	—	V_{DD}	
I_{OH}	Source Current for I/O Pins	3V	$V_{OH}=0.9V_{DD}$, SLEDCn[m+1, m]=00B (n=0,1...; m=0 or 2 or 4 or 6)	-0.7	-1.5	—	mA
		5V	$V_{OH}=0.9V_{DD}$, SLEDCn[m+1, m]=00B (n=0,1...; m=0 or 2 or 4 or 6)	-1.5	-2.9	—	
		3V	$V_{OH}=0.9V_{DD}$, SLEDCn[m+1, m]=01B (n=0,1...; m=0 or 2 or 4 or 6)	-1.3	-2.5	—	
		5V	$V_{OH}=0.9V_{DD}$, SLEDCn[m+1, m]=01B (n=0,1...; m=0 or 2 or 4 or 6)	-2.5	-5.1	—	
		3V	$V_{OH}=0.9V_{DD}$, SLEDCn[m+1, m]=10B (n=0,1...; m=0 or 2 or 4 or 6)	-1.8	-3.6	—	
		5V	$V_{OH}=0.9V_{DD}$, SLEDCn[m+1, m]=10B (n=0,1...; m=0 or 2 or 4 or 6)	-3.6	-7.3	—	
		3V	$V_{OH}=0.9V_{DD}$, SLEDCn[m+1, m]=11B (n=0,1...; m=0 or 2 or 4 or 6)	-4	-8	—	
		5V	$V_{OH}=0.9V_{DD}$, SLEDCn[m+1, m]=11B (n=0,1...; m=0 or 2 or 4 or 6)	-8	-16	—	
I_{OL}	Sink Current for I/O Pins	3V	$V_{OL}=0.1V_{DD}$	16	32	—	mA
		5V		32	65	—	
R_{PH}	Pull-high Resistance for I/O Ports^{Note}	3V	—	20	60	100	kΩ
		5V	—	10	30	50	
I_{LEAK}	Input Leakage Current	5V	$V_{IN}=V_{DD}$ or $V_{IN}=V_{SS}$	—	—	±1	μA
t_{TPI}	TM Capture Input Pin Minimum Pulse Width	—	—	0.3	—	—	μs
t_{TCK}	TM Clock Input Pin Minimum Pulse Width	—	—	0.3	—	—	μs
t_{INT}	External Interrupt Minimum Pulse Width	—	—	10	—	—	μs

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

Memory Characteristics

Ta=-40°C~85°C

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

LVR Electrical Characteristics

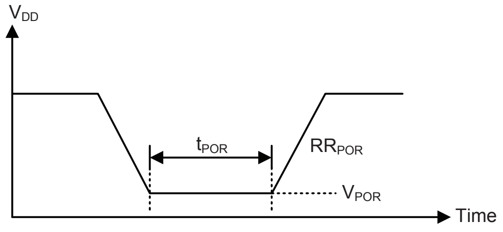
Ta=-40°C~85°C, unless otherwise specify

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{LVR}	Low Voltage Reset Voltage	—	LVR enable, voltage select 2.10V	- 5%	2.10	+ 5%	V
		—	LVR enable, voltage select 2.55V		2.55		
		—	LVR enable, voltage select 3.15V		3.15		
		—	LVR enable, voltage select 3.80V		3.80		
I_{LVRBG}	Operating Current	3V	LVR enable, VBGEN=0	—	—	18	μA
		5V		—	20	25	
		3V	LVR enable, VBGEN=1	—	—	150	
		5V		—	180	200	
t_{LVR}	Minimum Low Voltage Width to Reset	—	—	120	240	480	μs
I_{LVR}	Additional Current for LVR Enable	—	VBGEN=0	—	—	24	μA

Power-on Reset Characteristics

Ta=25°C

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



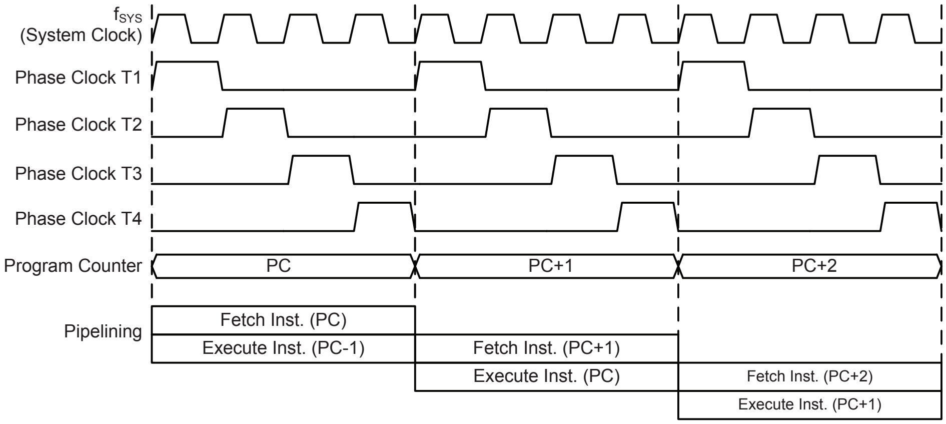
System Architecture

A key factor in the high-performance features of the Holtek range of microcontrollers is attributed to their internal system architecture. The series of devices take advantage of the usual features found within RISC microcontrollers providing increased speed of operation and enhanced performance. The pipelining scheme is implemented in such a way that instruction fetching and instruction execution are overlapped, hence instructions are effectively executed in one or two cycles for most of the standard or extended instructions respectively. The exceptions to this are branch or call instructions which need one more cycle. An 8-bit wide ALU is used in practically all instruction set operations, which carries out arithmetic operations, logic operations, rotation, increment, decrement, branch decisions, etc. The internal data path is simplified by moving data through the Accumulator and the ALU. Certain internal registers are implemented in the Data Memory and can be directly or indirectly addressed. The simple addressing methods of these registers along with additional architectural features ensure that a minimum of external components is required to provide a functional I/O control system with maximum reliability and flexibility. This makes the devices suitable for low-cost, high-volume production for controller applications.

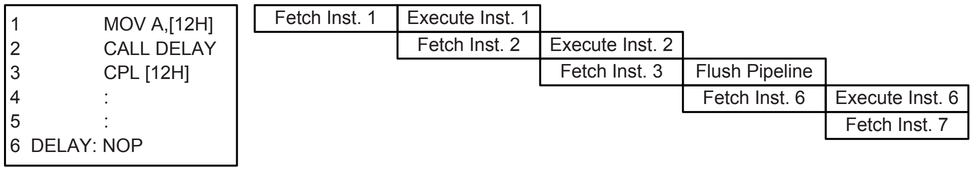
Clocking and Pipelining

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

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



System Clocking and Pipelining



```
1           MOV A,[12H]
2           CALL DELAY
3           CPL [12H]
4           :
5           :
6  DELAY: NOP
```

Instruction Fetching

Program Counter

During program execution, the Program Counter is used to keep track of the address of the next instruction to be executed. It is automatically incremented by one each time an instruction is executed except for instructions, such as “JMP” or “CALL” that demands a jump to a non-consecutive Program Memory address. Only the lower 8 bits, known as the Program Counter Low Register, are directly addressable by the application program.

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

Device	Program Counter	
	High Byte	PCL Register
BS83B24C	PC11~PC8	PCL7~PCL0
BS83C40C	PC11~PC8	PCL7~PCL0

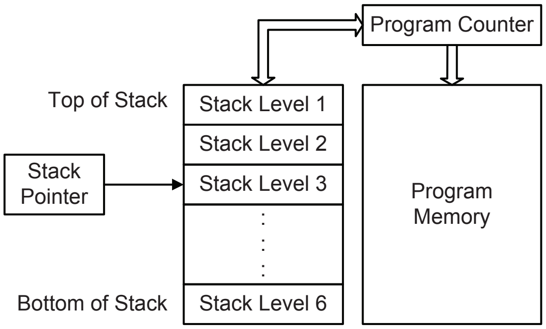
Program Counter

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

Stack

This is a special part of the memory which is used to save the contents of the Program Counter only. The stack is organised into 6 levels 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.


Program Counter
Top of Stack
Stack Level 1
Stack Level 2
Stack Pointer
Stack Level 3
Program Memory
Bottom of Stack
Stack Level 6

Arithmetic and Logic Unit – ALU

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

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

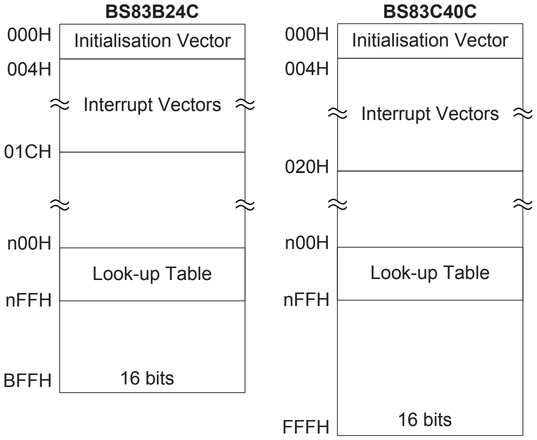
Flash Program Memory

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

Device	Capacity
BS83B24C	3K × 16
BS83C40C	4K × 16

Structure

The Program Memory has a capacity of 3K×16 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.



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 corresponding table read instruction such as "TABRD [m]" or "TABRDL [m]" respectively when the memory [m] is located in sector 0. If the memory [m] is located in other sectors, the data can be retrieved from the program memory using the corresponding extended table read instruction such as "LTABRD [m]" or "LTABRDL [m]" respectively. When the instruction is executed, the lower order table byte from the Program Memory will be transferred to the user defined Data Memory register [m] as specified in the instruction. The higher order table data byte from the Program Memory will be transferred to the TBLH special register.

The accompanying diagram illustrates the addressing data flow of the look-up table.

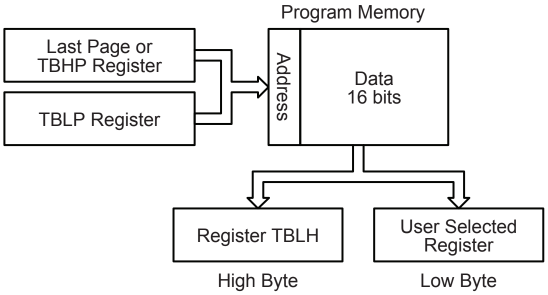


Table Program Example

The following example shows how the table pointer and table data is defined and retrieved from the microcontroller. This example uses raw table data located in the Program Memory which is stored there using the ORG statement. The value at this ORG statement is “F00H” refers to the start address of the last page within the 4K Program Memory of the BS83C40C. The table pointer low byte register is setup here to have an initial value of “06H”. This will ensure that the first data read from the data table will be at the Program Memory address “F06H” or 6 locations after the start of the last page. Note that the value for the table pointer is referenced to the first address specified by TBLP and TBHP if the “TABRD [m]” or “LTABRD [m]” instruction is being used. The high byte of the table data which in this case is equal to zero will be transferred to the TBLH register automatically when the “TABRD [m]” or “LTABRD [m]” instruction is executed.

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

Table Read Program Example

```
tempreg1 db ?   ; temporary register #1
tempreg2 db ?   ; temporary register #2
:
mov a,06h       ; initialise low table pointer - note that this address is referenced
mov tblp,a      ; to the last page or the page that tbhp pointed
mov a,0Fh       ; initialise high table pointer
mov tbhp,a      ; It is not necessary to set tbhp register if executing “tabrdl” instruction
:
tabrd tempreg1  ; transfers value in table referenced by table pointer data at program
                ; memory address “F06H” 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 “F05H” transferred to
                ; tempreg2 and TBLH in this example the data “1AH” is
                ; transferred to tempreg1 and data “0FH” to register tempreg2
:
org F00h        ; sets initial address of program memory
dc 00Ah, 00Bh, 00Ch, 00Dh, 00Eh, 00Fh, 01Ah, 01Bh
:
```

In Circuit Programming – ICP

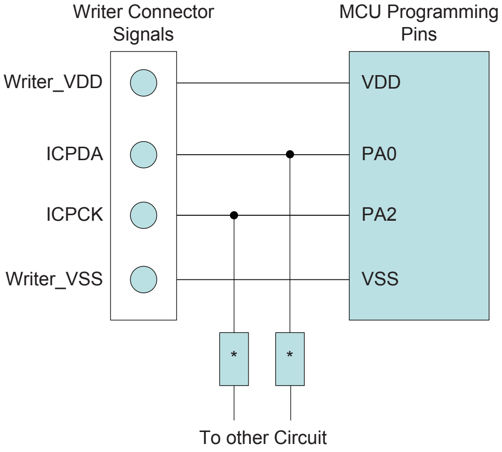
The provision of Flash type Program Memory provides the user with a means of convenient and easy upgrades and modifications to their programs on the same device. As an additional convenience, a means of programming the microcontroller in-circuit has provided using a 4-pin interface. This provides manufacturers with the possibility of manufacturing their circuit boards complete with a programmed or un-programmed microcontroller, and then programming or upgrading the program at a later stage. This enables product manufacturers to easily keep their manufactured products supplied with the latest program releases without removal and re-insertion of the device.

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

Writer Pins	MCU Programming Pins	Pin Description
ICPDA	PA0	Programming serial data/address
ICPCK	PA2	Programming clock
VDD	VDD	Power supply
VSS	VSS	Ground

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

During the programming process, taking control of the ICPDA and ICPCK 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.

On-Chip Debug Support – OCDS

There are EV chips named BS83BV24C and BS83CV40C which are used to emulate the real MCU devices named BS83B24C and BS83C40C respectively. The EV chip device also provides an "On-Chip Debug" function to debug the device during the development process. The EV chip and the actual MCU device are almost functionally compatible except for the "On-Chip Debug" function. Users can use the EV chip device to emulate the real chip device behavior by connecting the OCDSDA and OCDSCK pins to the HT-IDE development tools. The OCDSDA pin is the OCDS Data/Address input/output pin while the OCDSCK pin is the OCDS clock input pin. When users use the EV chip for debugging, other functions which are shared with the OCDSDA and OCDSCK pins in the actual MCU device will have no effect in the EV chip. However, the two OCDS pins which are pin-shared with the ICP programming pins are still used as the Flash Memory programming pins for ICP. For a more detailed OCDS description, refer to the corresponding document.

e-Link Pins	EV Chip Pins	Pin Description
OCDSDA	OCDSDA	On-chip debug support data/address input/output
OCDSCK	OCDSCK	On-chip debug support clock input
VDD	VDD	Power supply
VSS	VSS	Ground

Data Memory

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

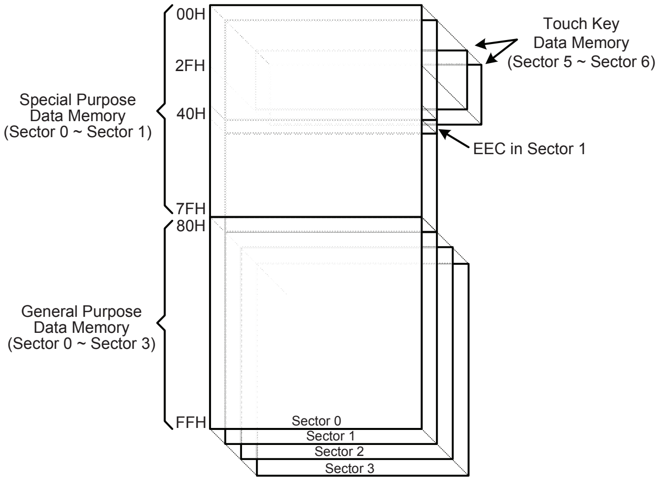
Structure

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

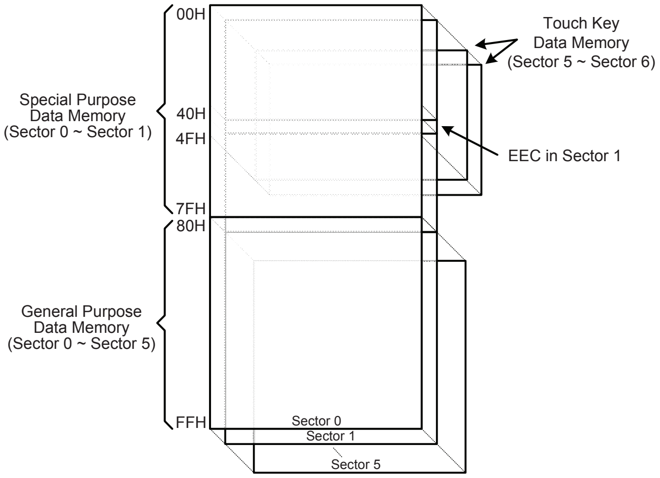
The overall Data Memory is subdivided into several sectors, all of which are implemented in 8-bit wide Memory. Each of the Data Memory sectors is categorized into two types, the Special Purpose Data Memory and the General Purpose Data Memory. The address range of the Special Purpose Data Memory for the device is from 00H to 7FH while the General Purpose Data Memory address range is from 80H to FFH. The Touch Key Data Memory is located in Sector 5 and Sector 6. Switching between the different Data Memory sectors is achieved by properly setting the Memory Pointers to correct value if using the indirect addressing method. The start address of the Data Memory for all devices is the address 00H.

Device	Special Purpose Data Memory	General Purpose Data Memory		Touch Key Data Memory	
	Available Sectors	Capacity	Sector: Address	Capacity	Sector: Address
BS83B24C	0, 1	512×8	0: 80H~FFH 1: 80H~FFH 2: 80H~FFH 3: 80H~FFH	96×8	5: 00H~2FH 6: 00H~2FH
BS83C40C	0, 1	768×8	0: 80H~FFH 1: 80H~FFH : 5: 80H~FFH	160×8	5: 00H~4FH 6: 00H~4FH

Data Memory Summary



Data Memory Structure – BS83B24C



Data Memory Structure – BS83C40C

Data Memory Addressing

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

Direct Addressing can be used in all sectors using the corresponding instruction which can address all available data memory space. For the accessed data memory which is located in any data memory sectors except sector 0, the extended instructions can be used to access the data memory instead of using the indirect addressing access. The main difference between standard instructions and extended instructions is that the data memory address “m” in the extended instructions can be up to 11 valid bits for the devices, the high byte indicates a sector and the low byte indicates a specific address.

General Purpose Data Memory

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

Special Purpose Data Memory

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

	Sector 0	Sector 1		Sector 0	Sector 1
00H	IAR0	PAS0	40H	▨	EEC
01H	MP0	PAS1	41H	EEA	▨
02H	IAR1	PBS0	42H	EED	▨
03H	MP1L	PBS1	43H	▨	▨
04H	MP1H	PCS0	44H	PTMC0	▨
05H	ACC	PCS1	45H	PTMC1	▨
06H	PCL	PDS0	46H	PTMDL	▨
07H	TBLP	▨	47H	PTMDH	▨
08H	TBLH	▨	48H	PTMAL	▨
09H	TBHP	▨	49H	PTMAH	▨
0AH	STATUS	▨	4AH	PTMRPL	▨
0BH	▨	▨	4BH	PTMRPH	▨
0CH	IAR2	TKTMR	4CH	▨	▨
0DH	MP2L	TKC0	4DH	▨	▨
0EH	MP2H	TK16DL	4EH	▨	▨
0FH	RSTFC	TK16DH	4FH	▨	▨
10H	INTEG	TKC1	50H	▨	▨
11H	INTC0	TKM016DL	51H	▨	▨
12H	INTC1	TKM016DH	52H	SIMC0	▨
13H	▨	TKM0ROL	53H	SIMC1/UUCR1	▨
14H	PA	TKM0ROH	54H	SIMC2/SIMA/UUCR2	▨
15H	PAC	TKM0C0	55H	SIMD/UTXR_RXR	▨
16H	PAPU	TKM0C1	56H	SIMTOC/UBRG	▨
17H	PAWU	TKM0C2	57H	UUSR	▨
18H	LVRC	TKM116DL	58H	▨	▨
19H	IFS	TKM116DH	59H	▨	▨
1AH	WDTC	TKM1ROL	5AH	▨	▨
1BH	TB0C	TKM1ROH	5BH	▨	▨
1CH	TB1C	TKM1C0	5CH	▨	▨
1DH	PSCR	TKM1C1	5DH	▨	▨
1EH	MFI0	TKM1C2	5EH	▨	▨
1FH	▨	TKM216DL	5FH	▨	▨
20H	PB	TKM216DH	60H	▨	▨
21H	PBC	TKM2ROL	61H	▨	▨
22H	PBPU	TKM2ROH	62H	▨	▨
23H	▨	TKM2C0	63H	▨	▨
24H	▨	TKM2C1	64H	▨	▨
25H	▨	TKM2C2	65H	▨	▨
26H	▨	TKM316DL	66H	▨	▨
27H	▨	TKM316DH	67H	▨	▨
28H	PC	TKM3ROL	68H	▨	▨
29H	PCC	TKM3ROH	69H	▨	▨
2AH	PCPU	TKM3C0	6AH	▨	▨
2BH	SCC	TKM3C1	6BH	▨	▨
2CH	HIRCC	TKM3C2	6CH	▨	▨
2DH	LXTC	TKM416DL	6DH	▨	▨
2EH	▨	TKM416DH	6EH	▨	▨
2FH	RSTC	TKM4ROL	6FH	▨	▨
30H	VBGC	TKM4ROH	70H	▨	▨
31H	▨	TKM4C0	71H	▨	▨
32H	▨	TKM4C1	72H	▨	▨
33H	▨	TKM4C2	73H	▨	▨
34H	▨	TKM516DL	74H	▨	▨
35H	PD	TKM516DH	75H	▨	▨
36H	PDC	TKM5ROL	76H	▨	▨
37H	PDPU	TKM5ROH	77H	▨	▨
38H	SLEDC0	TKM5C0	78H	▨	▨
39H	SLEDC1	TKM5C1	79H	▨	▨
3AH	▨	TKM5C2	7AH	▨	▨
3BH	▨	▨	7BH	▨	▨
3CH	▨	▨	7CH	▨	▨
3DH	▨	▨	7DH	▨	▨
3EH	▨	▨	7EH	▨	▨
3FH	▨	▨	7FH	▨	▨

▨ : Unused, read as 00H

Special Purpose Data Memory – BS83B24C

	Sector 0	Sector 1		Sector 0	Sector 1
00H	IAR0	PAS0	40H	CTMAH	EEC
01H	MP0	PAS1	41H	EEA	TKM616DL
02H	IAR1	PBS0	42H	EED	TKM616DH
03H	MP1L	PBS1	43H	▨	TKM6ROL
04H	MP1H	PCS0	44H	PTMC0	TKM6ROH
05H	ACC	PCS1	45H	PTMC1	TKM6C0
06H	PCL	PDS0	46H	PTMDL	TKM6C1
07H	TBLP	PDS1	47H	PTMDH	TKM6C2
08H	TBLH	PES0	48H	PTMAL	TKM716DL
09H	TBHP	PES1	49H	PTMAH	TKM716DH
0AH	STATUS	PFS0	4AH	PTMRPL	TKM7ROL
0BH	▨	▨	4BH	PTMRPH	TKM7ROH
0CH	IAR2	TKTMR	4CH	PE	TKM7C0
0DH	MP2L	TKC0	4DH	PEC	TKM7C1
0EH	MP2H	TK16DL	4EH	PEPU	TKM7C2
0FH	RSTFC	TK16DH	4FH	PF	TKM816DL
10H	INTEG	TKC1	50H	PFC	TKM816DL
11H	INTC0	TKM016DL	51H	PFPU	TKM8ROL
12H	INTC1	TKM016DH	52H	SIMC0	TKM8ROH
13H	INTC2	TKM0ROL	53H	SIMC1/UUCR1	TKM8C0
14H	PA	TKM0ROH	54H	SIMC2/SIMA/UUCR2	TKM8C1
15H	PAC	TKM0C0	55H	SIMD/UTXR_RXR	TKM8C2
16H	PAPU	TKM0C1	56H	SIMTOC/UBRG	TKM916DL
17H	PAWU	TKM0C2	57H	UUSR	TKM916DH
18H	LVRC	TKM116DL	58H	▨	TKM9ROL
19H	IFS	TKM116DH	59H	▨	TKM9ROH
1AH	WDTC	TKM1ROL	5AH	▨	TKM9C0
1BH	TB0C	TKM1ROH	5BH	▨	TKM9C1
1CH	TB1C	TKM1C0	5CH	▨	TKM9C2
1DH	PSCR	TKM1C1	5DH	▨	▨
1EH	MFI0	TKM1C2	5EH	▨	▨
1FH	MFI1	TKM216DL	5FH	▨	▨
20H	PB	TKM216DH	60H	▨	▨
21H	PBC	TKM2ROL	61H	▨	▨
22H	PBPU	TKM2ROH	62H	▨	▨
23H	▨	TKM2C0	63H	▨	▨
24H	▨	TKM2C1	64H	▨	▨
25H	▨	TKM2C2	65H	▨	▨
26H	▨	TKM316DL	66H	▨	▨
27H	▨	TKM316DH	67H	▨	▨
28H	PC	TKM3ROL	68H	▨	▨
29H	PCC	TKM3ROH	69H	▨	▨
2AH	PCPU	TKM3C0	6AH	▨	▨
2BH	SCC	TKM3C1	6BH	▨	▨
2CH	HIRCC	TKM3C2	6CH	▨	▨
2DH	LXTC	TKM416DL	6DH	▨	▨
2EH	▨	TKM416DH	6EH	▨	▨
2FH	RSTC	TKM4ROL	6FH	▨	▨
30H	VBGC	TKM4ROH	70H	▨	▨
31H	▨	TKM4C0	71H	▨	▨
32H	▨	TKM4C1	72H	▨	▨
33H	▨	TKM4C2	73H	▨	▨
34H	▨	TKM516DL	74H	▨	▨
35H	PD	TKM516DH	75H	▨	▨
36H	PDC	TKM5ROL	76H	▨	▨
37H	PDPU	TKM5ROH	77H	▨	▨
38H	SLEDC0	TKM5C0	78H	▨	▨
39H	SLEDC1	TKM5C1	79H	▨	▨
3AH	SLEDC2	TKM5C2	7AH	▨	▨
3BH	CTMC0	▨	7BH	▨	▨
3CH	CTMC1	▨	7CH	▨	▨
3DH	CTMDL	▨	7DH	▨	▨
3EH	CTMDH	▨	7EH	▨	▨
3FH	CTMAL	▨	7FH	▨	▨

▨ : Unused, read as 00H

Special Purpose Data Memory – BS83C40C

Special Function Register Description

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

Indirect Addressing Registers – IAR0, IAR1, IAR2

The Indirect Addressing Registers, IAR0, IAR1 and IAR2, although having their locations in normal RAM register space, do not actually physically exist as normal registers. The method of indirect addressing for RAM data manipulation uses these Indirect Addressing Registers and Memory Pointers, in contrast to direct memory addressing, where the actual memory address is specified. Actions on the IAR0, IAR1 and IAR2 registers will result in no actual read or write operation to these registers but rather to the memory location specified by their corresponding Memory Pointers, MP0, MP1L/MP1H or MP2L/MP2H. Acting as a pair, IAR0 and MP0 can together access data only from Sector 0 while the IAR1 register together with the MP1L/MP1H register pair and IAR2 register together with the MP2L/MP2H register pair can access data from any Data Memory Sector. As the Indirect Addressing Registers are not physically implemented, reading the Indirect Addressing Registers will return a result of “00H” and writing to the registers will result in no operation.

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

Five Memory Pointers, known as MP0, MP1L, MP1H, MP2L, MP2H, are provided. These Memory Pointers are physically implemented in the Data Memory and can be manipulated in the same way as normal registers providing a convenient way with which to address and track data. When any operation to the relevant Indirect Addressing Registers is carried out, the actual address that the microcontroller is directed to is the address specified by the related Memory Pointer. MP0, together with Indirect Addressing Register, IAR0, are used to access data from Sector 0, while MP1L/MP1H together with IAR1 and MP2L/MP2H together with IAR2 are used to access data from all sectors according to the corresponding MP1H or MP2H register. Direct Addressing can be used in all sectors using the corresponding instruction which can address all available data memory space.

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

Indirect Addressing Program Example 1

```
data .section ´data´
adres1   db ?
adres2   db ?
adres3   db ?
adres4   db ?
block    db ?
code .section at 0 ´code´
org 00h
start:
     mov a, 04h           ; setup size of block
     mov block, a
     mov a, offset adres1 ; Accumulator loaded with first RAM address
     mov mp0, a           ; setup memory pointer with first RAM address
loop:
     clr IAR0             ; clear the data at address defined by MP0
     inc mp0              ; increment memory pointer
     sdz block            ; check if last memory location has been cleared
     jmp loop
continue:
```

Indirect Addressing Program Example 2

```
data .section ´data´
adres1   db ?
adres2   db ?
adres3   db ?
adres4   db ?
block    db ?
code .section at 0 ´code´
org 00h
start:
     mov a, 04h           ; setup size of block
     mov block, a
     mov a, 01h           ; setup the memory sector
     mov mp1h, a
     mov a, offset adres1 ; Accumulator loaded with first RAM address
     mov mp1l, a          ; setup memory pointer with first RAM address
loop:
     clr IAR1             ; clear the data at address defined by MP1L
     inc mp1l             ; increment memory pointer MP1L
     sdz block            ; check if last memory location has been cleared
     jmp loop
continue:
```

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

Direct Addressing Program Example using extended instructions

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

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

Accumulator – ACC

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

Program Counter Low Register – PCL

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

Look-up Table Registers – TBLP, TBHP, TBLH

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

Status Register – STATUS

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

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

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

- C is set if an operation results in a carry during an addition operation or if a borrow does not take place during a subtraction operation; otherwise C is cleared. C is also affected by a rotate through carry instruction.
- AC is set if an operation results in a carry out of the low nibbles in addition, or no borrow from the high nibble into the low nibble in subtraction; otherwise AC is cleared.
- Z is set if the result of an arithmetic or logical operation is zero; otherwise Z is cleared.
- OV is set if an operation results in a carry into the highest-order bit but not a carry out of the highest-order bit, or vice versa; otherwise OV is cleared.
- PDF is cleared by a system power-up or executing the “CLR WDT” instruction. PDF is set by executing the “HALT” instruction.
- TO is cleared by a system power-up or executing the “CLR WDT” or “HALT” instruction. TO is set by a WDT time-out.
- CZ is the operational result of different flags for different instructions. Refer to register definitions for more details.
- SC is the result of the “XOR” operation which is performed by the OV flag and the MSB of the current instruction operation result.

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

- **STATUS Register**

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

"x": Unknown

Bit 7 **SC**: The result of the "XOR" operation which is performed by the OV flag and the MSB of the instruction operation result.

Bit 6 **CZ**: The operational result of different flags for different instructions.

For SUB/SUBM/LSUB/LSUBM instructions, the CZ flag is equal to the Z flag.

For SBC/SBCM/LSBC/LSBCM instructions, the CZ flag is the "AND" operation result which is performed by the previous operation CZ flag and current operation zero flag.

For other instructions, the CZ flag will not be affected.

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

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

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

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

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

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

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

EEPROM Data Memory

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

EEPROM Data Memory Structure

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

EEPROM Registers

Three registers control the overall operation of the internal EEPROM Data Memory. These are the address register, EEA, the data register, EED and a single control register, EEC. As both the EEA and EED registers are located in Sector 0, they can be directly accessed in the same was as any other Special Function Register. The EEC register however, being located in Sector 1, can be read from or written to indirectly using the MP1L/MP1H or MP2L/MP2H Memory Pointer and Indirect Addressing Register, IAR1/IAR2. Because the EEC control register is located at address 40H in Sector 1, the MP1L or MP2L Memory Pointer must first be set to the value 40H and the MP1H or MP2H Memory Pointer high byte set to the value, 01H, before any operations on the EEC register are executed.

Register Name	**Bit**							
	7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
EEA	—	EEA6	EEA5	EEA4	EEA3	EEA2	EEA1	EEA0
EED	EED7	EED6	EED5	EED4	EED3	EED2	EED1	EED0
EEC	—	—	—	—	WREN	WR	RDEN	RD

EEPROM Register List

- **EEA Register**

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	—	EEA6	EEA5	EEA4	EEA3	EEA2	EEA1	EEA0
R/W	—	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	0	0	0	0	0	0	0

Bit 7	Unimplemented, read as “0”
Bit 6~0	**EEA6~EEA0**: Data EEPROM address bit 6 ~ bit 0

- **EED Register**

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	EED7	EED6	EED5	EED4	EED3	EED2	EED1	EED0
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	**EED7~EED0**: Data EEPROM data bit 7 ~ bit 0

- **EEC Register**

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

Bit 7~4 Unimplemented, read as "0"

Bit 3 **WREN**: Data EEPROM Write Enable
0: Disable
1: Enable

This is the Data EEPROM Write Enable Bit which must be set high before Data EEPROM write operations are carried out. Clearing this bit to zero will inhibit Data EEPROM write operations.

Bit 2 **WR**: EEPROM Write Control
0: Write cycle has finished
1: Activate a write cycle

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

Bit 1 **RDEN**: Data EEPROM Read Enable
0: Disable
1: Enable

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

Bit 0 **RD**: EEPROM Read Control
0: Read cycle has finished
1: Activate a read cycle

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

Note: The WREN, WR, RDEN and RD cannot be set high at the same time in one instruction. The WR and RD cannot be set high at the same time.

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 Memory Pointer high byte register, MP1H or MP2H, will be reset to zero, which means that Data Memory Sector 0 will be selected. As the EEPROM control register is located in Sector 1, this adds a further measure of protection against spurious write operations. During normal program operation, ensuring that the Write Enable bit in the control register is cleared will safeguard against incorrect write operations.

EEPROM Interrupt

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

Programming Considerations

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

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

Programming Examples

Reading data from the EEPROM – polling method

```
MOV  A, EEPROM_ADRES          ; user defined address
MOV  EEA, A
MOV  A, 40H                   ; setup memory pointer MP1L
MOV  MP1L, A                  ; MP1L points to EEC register
MOV  A, 01H                   ; setup memory pointer MP1H
MOV  MP1H, A
SET  IAR1.1                   ; set RDEN bit, enable read operations
SET  IAR1.0                   ; start Read Cycle - set RD bit
BACK:
SZ   IAR1.0                   ; check for read cycle end
JMP  BACK
CLR  IAR1                     ; disable EEPROM read/write
CLR  MP1H
MOV  A, EED                   ; move read data to register
MOV  READ_DATA, A
```

Writing Data to the EEPROM – polling method

```
MOV  A, EEPROM_ADRES          ; user defined address
MOV  EEA, A
MOV  A, EEPROM_DATA           ; user defined data
MOV  EED, A
MOV  A, 040H                  ; setup memory pointer MP1L
MOV  MP1L, A                  ; MP1L points to EEC register
MOV  A, 01H                   ; setup memory pointer MP1H
MOV  MP1H, A
CLR  EMI
SET  IAR1.3                   ; set WREN bit, enable write operations
SET  IAR1.2                   ; start Write Cycle - set WR bit – executed immediately
                              ; after set WREN bit
SET  EMI
BACK:
SZ   IAR1.2                   ; check for write cycle end
JMP  BACK
CLR  IAR1                     ; disable EEPROM read/write
CLR  MP1H
```

Oscillators

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

Oscillator Overview

In addition to being the source of the main system clock the oscillators also provide clock sources for the Watchdog Timer and Time Base Interrupts. External oscillator requiring some external components as well as fully integrated internal oscillators, requiring no external components, are provided to form a wide range of both fast and slow system oscillators. All oscillator options are selected through the registers. The higher frequency oscillators provide higher performance but carry with it the disadvantage of higher power requirements, while the opposite is of course true for the lower frequency oscillators. With the capability of dynamically switching between fast and slow system clock, the device have the flexibility to optimize the performance/power ratio, a feature especially important in power sensitive portable applications.

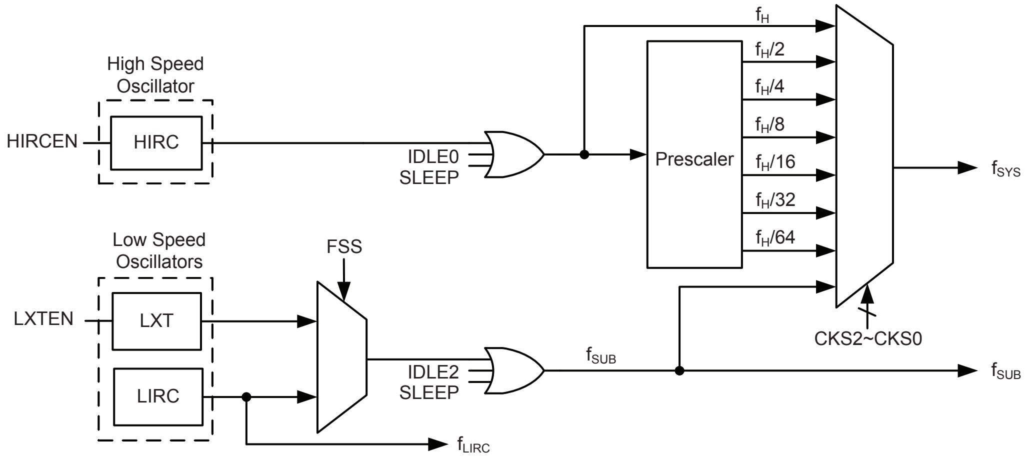
Type	Name	Freq.	Pins
Internal High Speed RC	HIRC	8/12/16MHz	—
Internal Low Speed RC	LIRC	32kHz	—
External Low Speed Crystal	LXT	32.768kHz	XT1/XT2

Oscillator Types

System Clock Configurations

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

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



System Clock Configurations

Internal RC Oscillator – HIRC

The internal RC oscillator is a fully integrated system oscillator requiring no external components. The internal RC oscillator has three fixed frequencies of 8/12/16 MHz, which is selected using a configuration option. The HIRC1~HIRC0 bits in the HIRCC register must also be setup to match the selected configuration option frequency. Setting up these bits is necessary to ensure that the HIRC frequency accuracy specified in the A.C. Characteristics is achieved. Device trimming during the manufacturing process and the inclusion of internal frequency compensation circuits are used to ensure that the influence of the power supply voltage, temperature and process variations on the oscillation frequency are minimised. As a result, at a power supply of 3V or 5V and at a temperature of 25°C degrees, the selected trimmed oscillation frequency will have a tolerance within 1%.

External 32.768kHz Crystal Oscillator – LXT

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

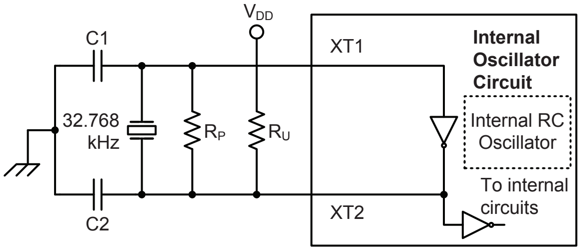
When the microcontroller enters the SLEEP or IDLE Mode, the system clock is switched off to stop microcontroller activity and to conserve power. However, in many microcontroller applications it may be necessary to keep the internal timers operational even when the microcontroller is in the SLEEP or IDLE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

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

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

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

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



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

External LXT Oscillator

LXT Oscillator C1 and C2 Values		
Crystal Frequency	**C1**	**C2**
32.768kHz	10pF	10pF
Note: 1. C1 and C2 values are for guidance only. 2. R_P=5M~10MΩ is recommended. 3. R_U=10MΩ is recommended.		

32.768kHz Crystal Recommended Capacitor Values

Internal 32kHz Oscillator – LIRC

The Internal 32kHz System Oscillator is one of the low frequency oscillator choices, which is selected via a software control bit, FSS. It is a fully integrated RC oscillator with a typical frequency of 32kHz at 5V, requiring no external components for its implementation. Device trimming during the manufacturing process and the inclusion of internal frequency compensation circuits are used to ensure that the influence of the power supply voltage, temperature and process variations on the oscillation frequency are minimised. As a result, at a power supply of 5V and at a temperature of 25˚C degrees, the fixed oscillation frequency of 32kHz will have a tolerance within 10%.

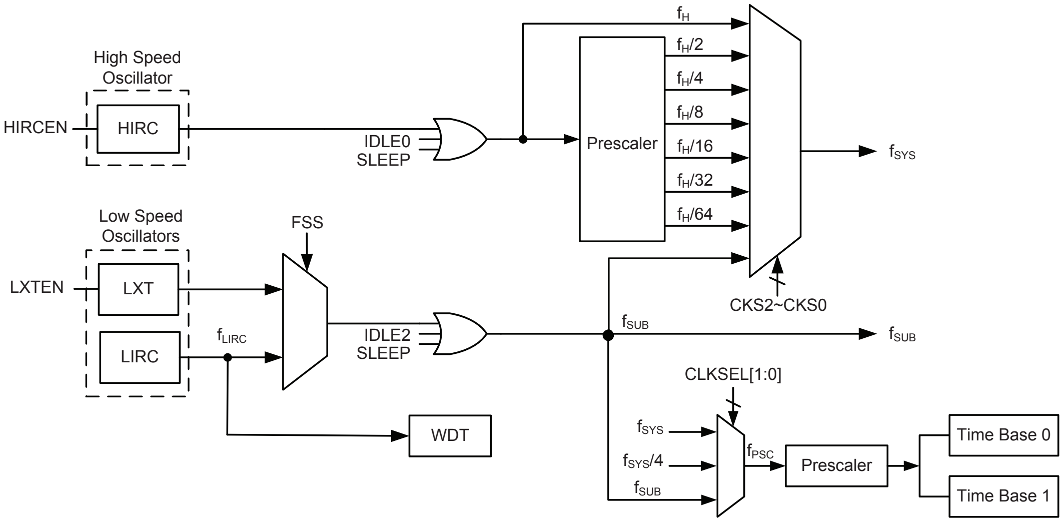
Operating Modes and System Clocks

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 both high and low speed clock sources are provided 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 devices have different clock sources for both the CPU and peripheral function operation. By providing the user with a wide range of clock selections using register programming, a clock system can be configured to obtain maximum application performance.

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


High Speed Oscillator
HIRCEN
HIRC
IDLE0
SLEEP
fH
fH/2
fH/4
fH/8
fH/16
fH/32
fH/64
Prescaler
fSYS
CKS2~CKS0
Low Speed Oscillators
LXTEN
LXT
LIRC
fLIRC
FSS
IDLE2
SLEEP
fSUB
fSUB
WDT
CLKSEL[1:0]
fSYS
fSYS/4
fSUB
fPSC
Prescaler
Time Base 0
Time Base 1

Device Clock Configurations

Note: When the system clock source f_{SYS} is switched to f_{SUB} from f_H, the high speed oscillator can be stopped to conserve the power or continue to oscillate to provide the clock source, f_H~$f_H/64$, for peripheral circuit to use, which is determined by configuring the corresponding high speed oscillator enable control bit.

System Operation Modes

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

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

"x": Don't care

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

2. The f_{LIRC} clock is always on as the WDT function is always enabled.

FAST 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. This mode operates allowing the microcontroller to operate normally with a clock source will come from the HIRC 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 bits in the SCC register. Although a high speed oscillator is used, running the microcontroller at a divided clock ratio reduces the operating current.

SLOW Mode

This is also a mode where the microcontroller operates normally although now with a slower speed clock source. The clock source used will be from f_{SUB}. The f_{SUB} clock is derived from the LXT or LIRC oscillator, which is selected via the software control bit FSS in the SCC register.

SLEEP Mode

The SLEEP Mode is entered when an HALT instruction is executed and when the FHIDEN and FSIDEN bit are low. In the SLEEP mode the CPU will be stopped, and the f_{SUB} clock to peripheral will be stopped too. However the f_{LIRC} clock will continue to operate as the WDT function is always enabled.

IDLE0 Mode

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

IDLE1 Mode

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

IDLE2 Mode

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

Control Registers

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

Register Name	Bit							
	7	6	5	4	3	2	1	0
SCC	CKS2	CKS1	CKS0	—	—	FSS	FHIDEN	FSIDEN
HIRCC	—	—	—	—	HIRC1	HIRC0	HIRCF	HIRCEN
LXTC	—	—	—	—	—	—	LXTF	LXTEN

System Operating Mode Control Register List

- **SCC Register**

Bit	7	6	5	4	3	2	1	0
Name	CKS2	CKS1	CKS0	—	—	FSS	FHIDEN	FSIDEN
R/W	R/W	R/W	R/W	—	—	R/W	R/W	R/W
POR	0	0	0	—	—	0	0	0

Bit 7~5 **CKS2~CKS0**: System clock selection
000: f_H
001: $f_H/2$
010: $f_H/4$
011: $f_H/8$
100: $f_H/16$
101: $f_H/32$
110: $f_H/64$
111: f_{SUB}

These three bits are used to select which clock is used as the system clock source. In addition to the system clock source directly derived from f_H or f_{SUB}, a divided version of the high speed system oscillator can also be chosen as the system clock source.

Bit 4~3 Unimplemented, read as “0”

Bit 2 **FSS**: Low Frequency clock selection
0: LIRC
1: LXT

Bit 1 **FHIDEN**: High Frequency oscillator control when CPU is switched off
0: Disable
1: Enable

This bit is used to control whether the high speed oscillator is activated or stopped when the CPU is switched off by executing an “HALT” instruction.

Bit 0 **FSIDEN**: Low Frequency oscillator control when CPU is switched off
0: Disable
1: Enable

This bit is used to control whether the low speed oscillator is activated or stopped when the CPU is switched off by executing an "HALT" instruction.

- **HIRCC Register**

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

Bit 7~4 Unimplemented, read as "0"

Bit 3~2 **HIRC1~HIRC0**: HIRC Frequency selection
00: 8MHz
01: 12MHz
10: 16MHz
11: 8MHz

When the HIRC oscillator is enabled or the HIRC frequency selection is changed by application program, the clock frequency will automatically be changed after the HIRCF flag is set to 1.

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

Bit 1 **HIRCF**: HIRC oscillator stable flag
0: HIRC unstable
1: HIRC stable

This bit is used to indicate whether the HIRC oscillator is stable or not. When the HIRCEN bit is set to 1 to enable the HIRC oscillator or the HIRC frequency selection is changed by application program, the HIRCF bit will first be cleared to 0 and then set to 1 after the HIRC oscillator is stable.

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

- **LXTC Register**

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

Bit 7~2 Unimplemented, read as 0.

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

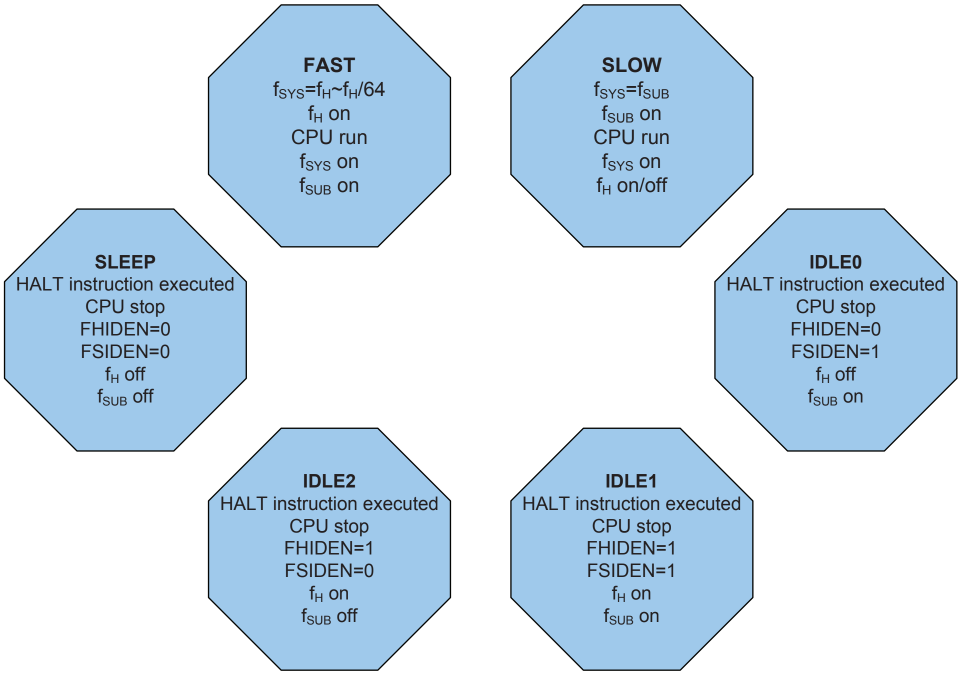
This bit is used to indicate whether the LXT oscillator is stable or not. When the LXTEN bit is set to 1 to enable the LXT oscillator, the LXTF bit will first be cleared to 0 and then set to 1 after the LXT oscillator is stable.

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

Operating Mode Switching

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

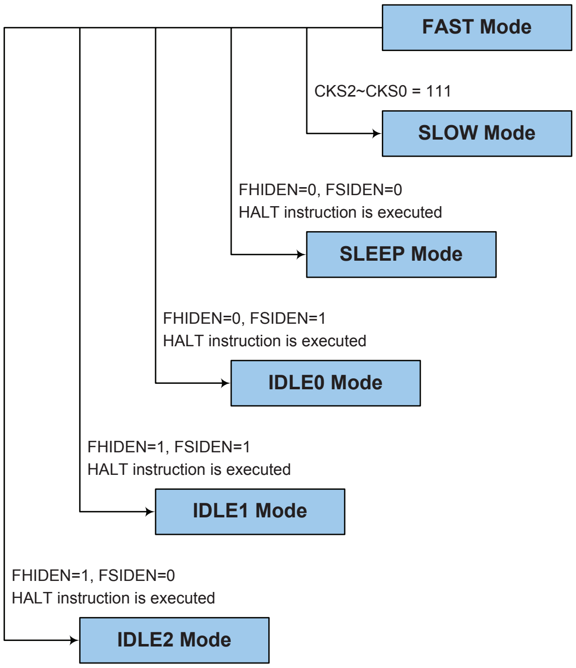
In simple terms, Mode Switching between the FAST Mode and SLOW Mode is executed using the CKS2~CKS0 bits in the SCC register while Mode Switching from the FAST/SLOW Modes to the SLEEP/IDLE Modes is executed via the HALT instruction. When an HALT instruction is executed, whether the device enters the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


FAST
$f_{SYS}=f_H\sim f_H/64$
f_H on
CPU run
f_{SYS} on
f_{SUB} on
SLOW
$f_{SYS}=f_{SUB}$
f_{SUB} on
CPU run
f_{SYS} on
f_H on/off
SLEEP
HALT instruction executed
CPU stop
FHIDEN=0
FSIDEN=0
f_H off
f_{SUB} off
IDLE0
HALT instruction executed
CPU stop
FHIDEN=0
FSIDEN=1
f_H off
f_{SUB} on
IDLE2
HALT instruction executed
CPU stop
FHIDEN=1
FSIDEN=0
f_H on
f_{SUB} off
IDLE1
HALT instruction executed
CPU stop
FHIDEN=1
FSIDEN=1
f_H on
f_{SUB} on

FAST Mode to SLOW Mode Switching

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

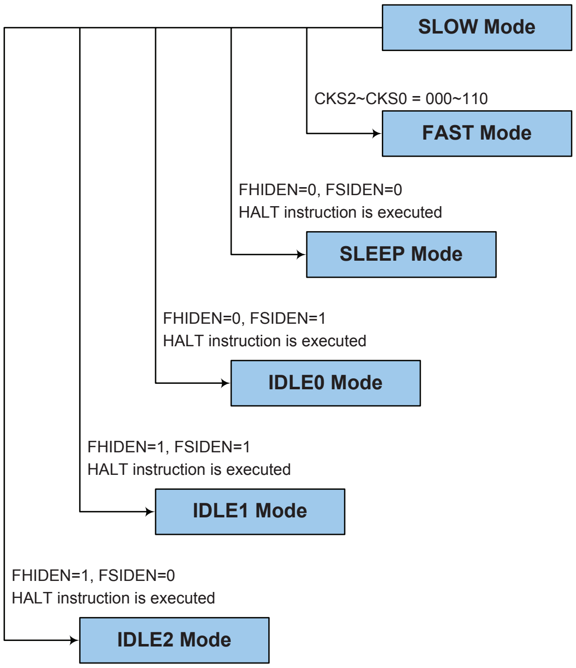
The SLOW Mode is sourced from the LXT or LIRC oscillator and therefore requires the oscillator to be stable before full mode switching occurs.


FAST Mode
CKS2~CKS0 = 111
SLOW Mode
FHIDEN=0, FSIDEN=0
HALT instruction is executed
SLEEP Mode
FHIDEN=0, FSIDEN=1
HALT instruction is executed
IDLE0 Mode
FHIDEN=1, FSIDEN=1
HALT instruction is executed
IDLE1 Mode
FHIDEN=1, FSIDEN=0
HALT instruction is executed
IDLE2 Mode

SLOW Mode to FAST Mode Switching

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

However, if f_H is not used in SLOW mode and thus switched off, it will take some time to re-oscillate and stabilise when switching to the FAST mode from the SLOW Mode. This is monitored using the HIRCF bit in the HIRCC register. The time duration required for the high speed system oscillator stabilization is specified in the relevant characteristics.


SLOW Mode
CKS2~CKS0 = 000~110
FAST Mode
FHIDEN=0, FSIDEN=0
HALT instruction is executed
SLEEP Mode
FHIDEN=0, FSIDEN=1
HALT instruction is executed
IDLE0 Mode
FHIDEN=1, FSIDEN=1
HALT instruction is executed
IDLE1 Mode
FHIDEN=1, FSIDEN=0
HALT instruction is executed
IDLE2 Mode

Entering the SLEEP Mode

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

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

Entering the IDLE0 Mode

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

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

Entering the IDLE1 Mode

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

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

Entering the IDLE2 Mode

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

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

Standby Current Considerations

As the main reason for entering the SLEEP or IDLE Mode is to keep the current consumption of the device to as low a value as possible, perhaps only in the order of several micro-amps except in the IDLE1 and IDLE2 Mode, there are other considerations which must also be taken into account by the circuit designer if the power consumption is to be minimised. Special attention must be made to the I/O pins on the device. All high-impedance input pins must be connected to either a fixed high or low level as any floating input pins could create internal oscillations and result in increased current consumption. This also applies to 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. Also note that additional standby current will also be required if the LXT or LIRC oscillator has been enabled.

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

Wake-up

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

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

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

When the device executes the “HALT” instruction, the PDF flag will be set to 1. The PDF flag will be cleared to 0 if the device experiences a system power-up or executes the clear Watchdog Timer instruction. If the system is woken up by a WDT overflow, a Watchdog Timer reset will be initiated and the TO flag will be set to 1. The TO flag is set if a WDT time-out occurs and causes a wake-up that only resets the Program Counter and Stack Pointer, other flags remain in their original status.

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

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 RC oscillator, f_{LIRC}. The LIRC internal oscillator has an approximate frequency of 32kHz and this specified internal clock period can vary with V_{DD}, temperature and process variations. The Watchdog Timer source clock is then subdivided by a ratio of 2^8 to 2^{18} to give longer timeouts, the actual value being chosen using the WS2~WS0 bits in the WDTC register.

Watchdog Timer Control Register

A single register, WDTC, controls the required timeout period as well as the enable and reset MCU operation.

- **WDTC Register**

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

Bit 7~3 **WE4~WE0**: WDT function software control
10101/01010: Enable
Others: Reset MCU
When these bits are changed by the environmental noise or software setting to reset the microcontroller, the reset operation will be activated after a delay time, t_{SRESET}, and the WRF bit in the RSTFC register will be set high.

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

- **RSTFC Register**

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

"x": Unknown

Bit 7~4 Unimplemented, read as "0"

Bit 3 **RSTF**: Reset Control Register Software Reset Flag
Described elsewhere

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

Bit 1 **LRF**: LVR Control Register Software Reset Flag
Described elsewhere

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

Watchdog Timer Operation

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

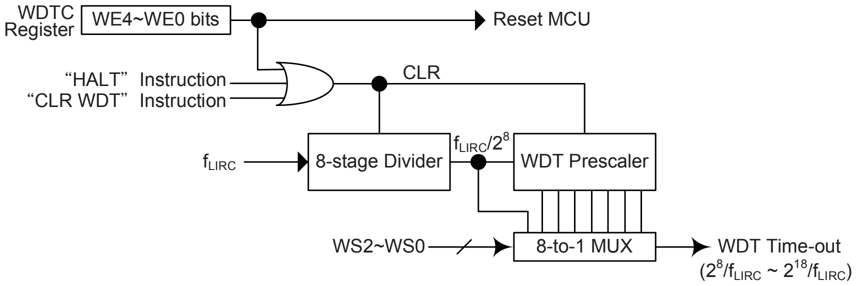
WE4 ~ WE0 Bits	WDT Function
10101B or 01010B	Enable
Any other values	Reset MCU

Watchdog Timer Function Control

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

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

The maximum time out period is when the 2^{18} division ratio is selected. As an example, with a 32kHz LIRC oscillator as its source clock, this will give a maximum watchdog period of around 8 seconds for the 2^{18} division ratio, and a minimum timeout of 8ms for the 2^{8} division 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.

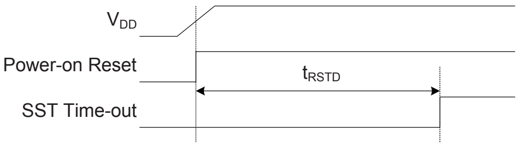
The Watchdog Timer overflow is one of many reset types and will reset the microcontroller. Another reset exists in the form of a Low Voltage Reset, LVR, where a full reset is implemented in situations where the power supply voltage falls below a certain threshold. All types of reset operations result in different register conditions being setup.

Reset Functions

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

Power-on Reset

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



Power-On Reset Timing Chart

Internal Reset Control

There is an internal reset control register, RSTC, which is used to provide a reset when the device operates abnormally due to the environmental noise interference. If the content of the RSTC register is set to any value other than 01010101B or 10101010B, it will reset the device after a delay time, t_{SRESET}. After power on the register will have a value of 01010101B.

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

Internal Reset Function Control

- **RSTC Register**

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

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

If these bits are changed due to adverse environmental conditions, the microcontroller will be reset. The reset operation will be activated after a delay time, t_{SRESET} and the RSTF bit in the RSTFC register will be set to 1.

- **RSTFC Register**

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

"x": Unknown

Bit 7~4 Unimplemented, read as "0"

Bit 3 **RSTF**: Reset Control Register Software Reset Flag
0: Not occurred
1: Occurred

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

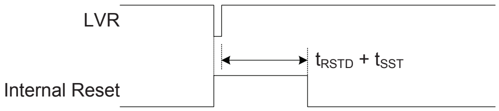
Bit 2 **LVRF**: LVR Function Reset Flag
Described elsewhere

Bit 1 **LRF**: LVR Control Register Software Reset Flag
Described elsewhere

Bit 0 **WRF**: WDT Control Register Software Reset Flag
Described elsewhere

Low Voltage Reset – LVR

The microcontrollers contain a low voltage reset circuit in order to monitor the supply voltage of the device. The LVR function is always enabled with a specific LVR voltage V_{LVR}. If the supply voltage of the device drops to within a range of 0.9V~V_{LVR} such as might occur when changing the battery, the LVR will automatically reset the device internally and the LVRF bit in the RSTFC register will also be set high. For a valid LVR signal, a low supply voltage, i.e., a voltage in the range between 0.9V~V_{LVR} must exist for a time greater than that specified by t_{LVR} in the LVR Electrical Characteristics. If the low supply voltage state does not exceed this value, the LVR will ignore the low supply voltage and will not perform a reset function. The actual V_{LVR} value can be selected by the LVS7~LVS0 bits in the LVRC register. If the LVS7~LVS0 bits are changed to some certain values by the environmental noise or software setting, the LVR will reset the device after a delay time, t_{SRESET}. When this happens, the LRF bit in the RSTFC register will be set high. After power on the register will have the value of 01010101B. Note that the LVR function will be automatically disabled when the device enters the IDLE/SLEEP mode.



Low Voltage Reset Timing Chart

- **LVRC Register**

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

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

When an actual low voltage condition occurs, as specified by one of the four defined LVR voltage values above, an MCU reset will be generated. The reset operation will be activated after the low voltage condition keeps more than a t_{LVR} time. In this situation the register contents will remain the same after such a reset occurs.

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

- **RSTFC Register**

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

"x": Unknown

Bit 7~4 Unimplemented, read as "0"

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

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

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

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

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

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

- **VBGC Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	VBGEN	—	—	—
R/W	—	—	—	—	R/W	—	—	—
POR	—	—	—	—	0	—	—	—

"x": Unknown

Bit 7~4 Unimplemented, read as "0"

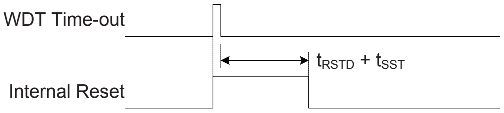
Bit 3 **VBGEN**: Bandgap Voltage Output Enable Control
0: Disable
1: Enable

Note that the Bandgap circuit is enabled when the LVR function is enabled or when the VBGEN bit is set to 1.

Bit 2~0 Unimplemented, read as "0"

Watchdog Time-out Reset during Normal Operation

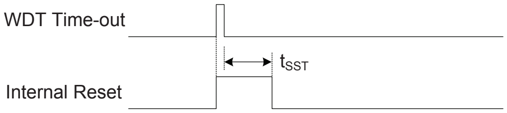
The Watchdog time-out Reset in the FAST mode or SLOW mode is the same as LVR reset except that the Watchdog time-out flag TO will be set high.



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 zero and the TO flag will be set high. Refer to the System Start Up Time Characteristics for t_{SST} details.



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

Reset Initial Conditions

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

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

"u": Unchanged

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

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

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

Register Name	BS83B24C	BS83C40C	Reset (Power On)	LVR Reset (Normal Operation)	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)
IAR0	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
MP0	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
IAR1	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
MP1L	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
MP1H	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
ACC	●	●	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
PCL	●	●	0000 0000	0000 0000	0000 0000	0000 0000
TBLP	●	●	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLH	●	●	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBHP	●	●	---- xxxx	----uuuu	---- uuuu	---- uuuu
STATUS	●	●	xx00 xxxx	uuuu uuuu	uu1u uuuu	uu11 uuuu
IAR2	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
MP2L	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
MP2H	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
RSTFC	●	●	---- 0x00	---- u1uu	---- uuuu	---- uuuu
INTEG	●	●	---- --00	---- --00	---- --00	---- --uu
INTC0	●	●	-000 0000	-000 0000	-000 0000	-uuu uuuu
INTC1	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
INTC2		●	---0 ---0	---0 ---0	---0 ---0	---u ---u
PA	●	●	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAC	●	●	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAPU	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PAWU	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
LVRC	●	●	0101 0101	0101 0101	0101 0101	uuuu uuuu
IFS	●	●	---- -000	---- -000	---- -000	---- -uuu
WDTC	●	●	0101 0011	0101 0011	0101 0011	uuuu uuuu
TB0C	●	●	0--- -000	0--- -000	0--- -000	u--- -uuu
TB1C	●	●	0--- -000	0--- -000	0--- -000	u--- -uuu
PSCR	●	●	---- --00	---- --00	---- --00	---- --uu
MFI0	●	●	--00 --00	--00 --00	--00 --00	--uu --uu
MFI1		●	--00 --00	--00 --00	--00 --00	--uu --uu
PB	●	●	1111 1111	1111 1111	1111 1111	uuuu uuuu
PBC	●	●	1111 1111	1111 1111	1111 1111	uuuu uuuu
PBPU	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PC	●	●	1111 1111	1111 1111	1111 1111	uuuu uuuu
PCC	●	●	1111 1111	1111 1111	1111 1111	uuuu uuuu
PCPU	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
SCC	●	●	000- -000	000- -000	000- -000	uuu- -uuu
HIRCC	●	●	---- 0001	---- 0001	---- 0001	---- uuuu
LXTC	●	●	---- --00	---- --00	---- --00	---- --uu
RSTC	●	●	0101 0101	0101 0101	0101 0101	uuuu uuuu
VBGC	●	●	---- 0---	---- 0---	---- 0---	---- u---
PD	●		---- --11	---- --11	---- --11	---- --uu
		●	1111 1111	1111 1111	1111 1111	uuuu uuuu

Register Name	BS83B24C	BS83C40C	Reset (Power On)	LVR Reset (Normal Operation)	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)
PDC	●		- - - - - - 1 1	- - - - - - 1 1	- - - - - - 1 1	- - - - - - u u
		●	1111 1111	1111 1111	1111 1111	uuuu uuuu
PDPU	●		- - - - - - 0 0	- - - - - - 0 0	- - - - - - 0 0	- - - - - - u u
		●	0000 0000	0000 0000	0000 0000	uuuu uuuu
SLEDC0	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
SLEDC1	●		- - 00 0000	- - 00 0000	- - 00 0000	- - uu uuuu
		●	0000 0000	0000 0000	0000 0000	uuuu uuuu
SLEDC2		●	- - 00 0000	- - 00 0000	- - 00 0000	- - uu uuuu
CTMC0		●	0000 0000	0000 0000	0000 0000	uuuu uuuu
CTMC1		●	0000 0000	0000 0000	0000 0000	uuuu uuuu
CTMDL		●	0000 0000	0000 0000	0000 0000	uuuu uuuu
CTMDH		●	- - - - - - 0 0	- - - - - - 0 0	- - - - - - 0 0	- - - - - - u u
CTMAL		●	0000 0000	0000 0000	0000 0000	uuuu uuuu
CTMAH		●	- - - - - - 0 0	- - - - - - 0 0	- - - - - - 0 0	- - - - - - u u
EEA	●	●	- 000 0000	- 000 0000	- 000 0000	- uuu uuuu
EED	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTMC0	●	●	0000 0 - - -	0000 0 - - -	0000 0 - - -	uuuu u - - -
PTMC1	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTMDL	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTMDH	●	●	- - - - - - 0 0	- - - - - - 0 0	- - - - - - 0 0	- - - - - - u u
PTMAL	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTMAH	●	●	- - - - - - 0 0	- - - - - - 0 0	- - - - - - 0 0	- - - - - - u u
PTMRPL	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTMRPH	●	●	- - - - - - 0 0	- - - - - - 0 0	- - - - - - 0 0	- - - - - - u u
PE		●	1111 1111	1111 1111	1111 1111	uuuu uuuu
PEC		●	1111 1111	1111 1111	1111 1111	uuuu uuuu
PEPU		●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PF		●	- - - - - - 1 1	- - - - - - 1 1	- - - - - - 1 1	- - - - - - u u
PFC		●	- - - - - - 1 1	- - - - - - 1 1	- - - - - - 1 1	- - - - - - u u
PFPU		●	- - - - - - 0 0	- - - - - - 0 0	- - - - - - 0 0	- - - - - - u u
SIMC0	●	●	1110 0000	1110 0000	1110 0000	uuuu uuuu
UUCR1* (UMD=1)	●	●	0000 00x0	0000 00x0	0000 00x0	uuuu uuuu
SIMC1 (UMD=0)	●	●	1000 0001	1000 0001	1000 0001	uuuu uuuu
SIMA/SIMC2/UUCR2	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
SIMD/UTXR_RXR	●	●	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
UBRG*	●	●	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
SIMTOC	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
UUSR	●	●	0000 1011	0000 1011	0000 1011	uuuu uuuu
PAS0	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PAS1	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PBS0	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PBS1	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PCS0	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PCS1	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu

Register Name	BS83B24C	BS83C40C	Reset (Power On)	LVR Reset (Normal Operation)	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)
PDS0	●		- - - - 0000	- - - - 0000	- - - - 0000	- - - - uuuu
		●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PDS1		●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PES0		●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PES1		●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PFS0		●	- - - - 0000	- - - - 0000	- - - - 0000	- - - - uuuu
TKTMR	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKC0	●	●	0000 0-00	0000 0-00	0000 0-00	uuuu u-uu
TK16DL	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TK16DH	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKC1	●	●	0000 0011	0000 0011	0000 0011	uuuu uuuu
TKM016DL	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM016DH	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM0ROL	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM0ROH	●	●	- - - - - - 00	- - - - - - 00	- - - - - - 00	- - - - - - uu
TKM0C0	●	●	- - 00 0000	- - 00 0000	- - 00 0000	- - uu uuuu
TKM0C1	●	●	0-00 0000	0-00 0000	0-00 0000	u-uu uuuu
TKM0C2	●	●	1110 0100	1110 0100	1110 0100	uuuu uuuu
TKM116DL	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM116DH	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM1ROL	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM1ROH	●	●	- - - - - - 00	- - - - - - 00	- - - - - - 00	- - - - - - uu
TKM1C0	●	●	- - 00 0000	- - 00 0000	- - 00 0000	- - uu uuuu
TKM1C1	●	●	0-00 0000	0-00 0000	0-00 0000	u-uu uuuu
TKM1C2	●	●	1110 0100	1110 0100	1110 0100	uuuu uuuu
TKM216DL	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM216DH	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM2ROL	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM2ROH	●	●	- - - - - - 00	- - - - - - 00	- - - - - - 00	- - - - - - uu
TKM2C0	●	●	- - 00 0000	- - 00 0000	- - 00 0000	- - uu uuuu
TKM2C1	●	●	0-00 0000	0-00 0000	0-00 0000	u-uu uuuu
TKM2C2	●	●	1110 0100	1110 0100	1110 0100	uuuu uuuu
TKM316DL	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM316DH	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM3ROL	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM3ROH	●	●	- - - - - - 00	- - - - - - 00	- - - - - - 00	- - - - - - uu
TKM3C0	●	●	- - 00 0000	- - 00 0000	- - 00 0000	- - uu uuuu
TKM3C1	●	●	0-00 0000	0-00 0000	0-00 0000	u-uu uuuu
TKM3C2	●	●	1110 0100	1110 0100	1110 0100	uuuu uuuu
TKM416DL	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM416DH	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM4ROL	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM4ROH	●	●	- - - - - - 00	- - - - - - 00	- - - - - - 00	- - - - - - uu
TKM4C0	●	●	- - 00 0000	- - 00 0000	- - 00 0000	- - uu uuuu
TKM4C1	●	●	0-00 0000	0-00 0000	0-00 0000	u-uu uuuu

Register Name	BS83B24C	BS83C40C	Reset (Power On)	LVR Reset (Normal Operation)	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)
TKM4C2	●	●	1110 0100	1110 0100	1110 0100	uuuu uuuu
TKM516DL	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM516DH	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM5ROL	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM5ROH	●	●	---- --00	---- --00	---- --00	---- --uu
TKM5C0	●	●	--00 0000	--00 0000	--00 0000	--uu uuuu
TKM5C1	●	●	0-00 0000	0-00 0000	0-00 0000	u-uu uuuu
TKM5C2	●	●	1110 0100	1110 0100	1110 0100	uuuu uuuu
EEC	●	●	---- 0000	---- 0000	---- 0000	---- uuuu
TKM616DL		●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM616DH		●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM6ROL		●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM6ROH		●	---- --00	---- --00	---- --00	---- --uu
TKM6C0		●	--00 0000	--00 0000	--00 0000	--uu uuuu
TKM6C1		●	0-00 0000	0-00 0000	0-00 0000	u-uu uuuu
TKM6C2		●	1110 0100	1110 0100	1110 0100	uuuu uuuu
TKM716DL		●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM716DH		●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM7ROL		●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM7ROH		●	---- --00	---- --00	---- --00	---- --uu
TKM7C0		●	--00 0000	--00 0000	--00 0000	--uu uuuu
TKM7C1		●	0-00 0000	0-00 0000	0-00 0000	u-uu uuuu
TKM816DL		●	1110 0100	1110 0100	1110 0100	uuuu uuuu
TKM7C2		●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM816DH		●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM8ROL		●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM8ROH		●	---- --00	---- --00	---- --00	---- --uu
TKM8C0		●	--00 0000	--00 0000	--00 0000	--uu uuuu
TKM8C1		●	0-00 0000	0-00 0000	0-00 0000	u-uu uuuu
TKM8C2		●	1110 0100	1110 0100	1110 0100	uuuu uuuu
TKM916DL		●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM916DH		●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM9ROL		●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM9ROH		●	---- --00	---- --00	---- --00	---- --uu
TKM9C0		●	--00 0000	--00 0000	--00 0000	--uu uuuu
TKM9C1		●	0-00 0000	0-00 0000	0-00 0000	u-uu uuuu
TKM9C2		●	1110 0100	1110 0100	1110 0100	uuuu uuuu

Note: “u” stands for unchanged
“x” stands for unknown
“-” stands for unimplemented
“*”: The UUCR1 and SIMC1 register share the same memory address while the UBRG and SIMTOC registers share the same memory address. The default value of the UUCR1 or UBRG register can be obtained when the UMD bit is set high by application program after a reset.

Input/Output Ports

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

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

Register Name	Bit							
	7	6	5	4	3	2	1	0
PA	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0
PAC	PAC7	PAC5	PAC5	PAC4	PAC3	PAC2	PAC1	PAC0
PAPU	PAPU7	PAPU4	PAPU5	PAPU4	PAPU3	PAPU2	PAPU1	PAPU0
PAWU	PAWU7	PAWU6	PAWU5	PAWU4	PAWU3	PAWU2	PAWU1	PAWU0
PB	PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0
PBC	PBC7	PBC6	PBC5	PBC4	PBC3	PBC2	PBC1	PBC0
PBPU	PBPU7	PBPU6	PBPU5	PBPU4	PBPU3	PBPU2	PBPU1	PBPU0
PC	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0
PCC	PCC7	PCC6	PCC5	PCC4	PCC3	PCC2	PCC1	PCC0
PCPU	PCPU7	PCPU6	PCPU5	PCPU4	PCPU3	PCPU2	PCPU1	PCPU0
PD	—	—	—	—	—	—	PD1	PD0
PDC	—	—	—	—	—	—	PDC1	PDC0
PDPU	—	—	—	—	—	—	PDPU1	PDPU0

“—”: Unimplemented

I/O Logic Function Register List – BS83B24C

Register Name	Bit							
	7	6	5	4	3	2	1	0
PA	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0
PAC	PAC7	PAC5	PAC5	PAC4	PAC3	PAC2	PAC1	PAC0
PAPU	PAPU7	PAPU4	PAPU5	PAPU4	PAPU3	PAPU2	PAPU1	PAPU0
PAWU	PAWU7	PAWU6	PAWU5	PAWU4	PAWU3	PAWU2	PAWU1	PAWU0
PB	PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0
PBC	PBC7	PBC6	PBC5	PBC4	PBC3	PBC2	PBC1	PBC0
PBPU	PBPU7	PBPU6	PBPU5	PBPU4	PBPU3	PBPU2	PBPU1	PBPU0
PC	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0
PCC	PCC7	PCC6	PCC5	PCC4	PCC3	PCC2	PCC1	PCC0
PCPU	PCPU7	PCPU6	PCPU5	PCPU4	PCPU3	PCPU2	PCPU1	PCPU0
PD	PD7	PD6	PD5	PD4	PD3	PD2	PD1	PD0
PDC	PDC7	PDC6	PDC5	PDC4	PDC3	PDC2	PDC1	PDC0
PDPU	PDPU7	PDPU6	PDPU5	PDPU4	PDPU3	PDPU2	PDPU1	PDPU0
PE	PE7	PE6	PE5	PE4	PE3	PE2	PE1	PE0
PEC	PEC7	PEC6	PEC5	PEC4	PEC3	PEC2	PEC1	PEC0
PEPU	PEPU7	PEPU6	PEPU5	PEPU4	PEPU3	PEPU2	PEPU1	PEPU0
PF	—	—	—	—	—	—	PF1	PF0
PFC	—	—	—	—	—	—	PFC1	PFC0
PFPU	—	—	—	—	—	—	PFPU1	PFPU0

“—”: Unimplemented

I/O Logic Function Register List – BS83C40C

Pull-high Resistors

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

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

- **PxPU Register**

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

PxPUn: I/O Port x Pin Pull-high Function Control
0: Disable
1: Enable

The PxPUn bit is used to control the pin pull-high function.Here the “x” can be A, B, C, D, E or F respectively depending upon the selected device. However, the actual available bits for each I/O port may be different.

Port A Wake-up

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

Note that the wake-up function can be controlled by the wake-up control registers only when the pin-shared functional pin is selected as general purpose input/output and the MCU enters the IDLE/SLEEP mode.

- **PAWU Register**

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

Bit 7~ 0 **PAWU7~PAWU0**: Port A Pin Wake-up Control
0: Disable
1: Enable

I/O Port Control Registers

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

- **PxC Register**

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

PxCn: I/O Port x Pin Type Selection
0: Output
1: Input

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

I/O Port Source Current Control

The devices support different source current driving capability for each I/O port. With the corresponding selection registers SLEDCn, each I/O port can support four levels of the source current driving capability. Users should refer to the Input/Output Characteristics section to select the desired source current for different applications.

- **SLEDC0 Register**

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

Bit 7~6 **SLEDC07~SLEDC06**: PB7~PB4 Source Current Selection
00: Source current=Level 0 (min.)
01: Source current=Level 1
10: Source current=Level 2
11: Source current=Level 3 (max.)

Bit 5~4 **SLEDC05~SLEDC04**: PB3~PB0 Source Current Selection
00: Source current=Level 0 (min.)
01: Source current=Level 1
10: Source current=Level 2
11: Source current=Level 3 (max.)

Bit 3~2 **SLEDC03~SLEDC02**: PA7~PA4 Source Current Selection
00: Source current=Level 0 (min.)
01: Source current=Level 1
10: Source current=Level 2
11: Source current=Level 3 (max.)

Bit 1~0 **SLEDC01~SLEDC00**: PA3~PA0 Source Current Selection
00: Source current=Level 0 (min.)
01: Source current=Level 1
10: Source current=Level 2
11: Source current=Level 3 (max.)

- **SLEDC1 Register – BS83B24C**

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

Bit 7~6 Unimplemented, read as "0"

Bit 5~4 **SLEDC15~SLEDC14**: PD1~PD0 Source Current Selection
00: Source current=Level 0 (min.)
01: Source current=Level 1
10: Source current=Level 2
11: Source current=Level 3 (max.)

Bit 3~2 **SLEDC13~SLEDC12**: PC7~PC4 Source Current Selection
00: Source current=Level 0 (min.)
01: Source current=Level 1
10: Source current=Level 2
11: Source current=Level 3 (max.)

Bit 1~0 **SLEDC11~SLEDC10**: PC3~PC0 Source Current Selection
00: Source current=Level 0 (min.)
01: Source current=Level 1
10: Source current=Level 2
11: Source current=Level 3 (max.)

- **SLEDC1 Register – BS83C40C**

Bit	7	6	5	4	3	2	1	0
Name	SLEDC17	SLEDC16	SLEDC15	SLEDC14	SLEDC13	SLEDC12	SLEDC11	SLEDC10
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 **SLEDC17~SLEDC16**: PD7~PD4 Source Current Selection
00: Source current=Level 0 (min.)
01: Source current=Level 1
10: Source current=Level 2
11: Source current=Level 3 (max.)

Bit 5~4 **SLEDC15~SLEDC14**: PD3~PD0 Source Current Selection
00: Source current=Level 0 (min.)
01: Source current=Level 1
10: Source current=Level 2
11: Source current=Level 3 (max.)

Bit 3~2 **SLEDC13~SLEDC12**: PC7~PC4 Source Current Selection
00: Source current=Level 0 (min.)
01: Source current=Level 1
10: Source current=Level 2
11: Source current=Level 3 (max.)

Bit 1~0 **SLEDC11~SLEDC10**: PC3~PC0 Source Current Selection
00: Source current=Level 0 (min.)
01: Source current=Level 1
10: Source current=Level 2
11: Source current=Level 3 (max.)

- **SLEDC2 Register – BS83C40C**

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

Bit 7~6 Unimplemented, read as "0"

Bit 5~4 **SLEDC25~SLEDC24**: PF1~PF0 Source Current Selection
00: Source current=Level 0 (min.)
01: Source current=Level 1
10: Source current=Level 2
11: Source current=Level 3 (max.)

Bit 3~2 **SLEDC23~SLEDC22**: PE7~PE4 Source Current Selection
00: Source current=Level 0 (min.)
01: Source current=Level 1
10: Source current=Level 2
11: Source current=Level 3 (max.)

Bit 1~0 **SLEDC21~SLEDC20**: PE3~PE0 Source Current Selection
00: Source current=Level 0 (min.)
01: Source current=Level 1
10: Source current=Level 2
11: Source current=Level 3 (max.)

Pin-shared Functions

The flexibility of the microcontroller range is greatly enhanced by the use of pins that have more than one function. Limited numbers of pins can force serious design constraints on designers but by supplying pins with multi-functions, many of these difficulties can be overcome. For these pins, the desired function of the multi-function I/O pins is selected by a series of registers via the application program control.

Pin-shared Function Selection Registers

The limited number of supplied pins in a package can impose restrictions on the amount of functions a certain device can contain. However by allowing the same pins to share several different functions and providing a means of function selection, a wide range of different functions can be incorporated into even relatively small package sizes. The devices include Port "x" output function Selection register "n", labeled as PxSn, and Input Function Selection register, labeled as IFS, which can select the desired functions of the multi-function pin-shared pins.

When the pin-shared input function is selected to be used, the corresponding input and output functions selection should be properly managed. For example, if the I^{2}C SDA line is used, the correnspinding pin-shared function should be configured as the SDA/SDI/RX function by configuring the PASn register and the SDA signal input should be properly selected using the IFS register. However, if the external interrupt function is selected to be used, the relevant output pin-shared function should be selected as an I/O function and the interrupt input signal should be selected.

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

Register Name	Bit							
	7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PAS0	PAS07	PAS06	PAS05	PAS04	PAS03	PAS02	PAS01	PAS00
PAS1	PAS17	PAS16	PAS15	PAS14	PAS13	PAS12	PAS11	PAS10
PBS0	PBS07	PBS06	PBS05	PBS04	PBS03	PBS02	PBS01	PBS00
PBS1	PBS17	PBS16	PBS15	PBS14	PBS13	PBS12	PBS11	PBS10
PCS0	PCS07	PCS06	PCS05	PCS04	PCS03	PCS02	PCS01	PCS00
PCS1	PCS17	PCS16	PCS15	PCS14	PCS13	PCS12	PCS11	PCS10
PDS0	—	—	—	—	PDS03	PDS02	PDS01	PDS00
IFS	—	—	—	—	—	IFS2	IFS1	IFS0

Pin-shared Function Selection Register List – BS83B24C

Register Name	Bit							
	7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PAS0	PAS07	PAS06	PAS05	PAS04	PAS03	PAS02	PAS01	PAS00
PAS1	PAS17	PAS16	PAS15	PAS14	PAS13	PAS12	PAS11	PAS10
PBS0	PBS07	PBS06	PBS05	PBS04	PBS03	PBS02	PBS01	PBS00
PBS1	PBS17	PBS16	PBS15	PBS14	PBS13	PBS12	PBS11	PBS10
PCS0	PCS07	PCS06	PCS05	PCS04	PCS03	PCS02	PCS01	PCS00
PCS1	PCS17	PCS16	PCS15	PCS14	PCS13	PCS12	PCS11	PCS10
PDS0	PDS07	PDS06	PDS05	PDS04	PDS03	PDS02	PDS01	PDS00
PDS1	PDS17	PDS16	PDS15	PDS14	PDS13	PDS12	PDS11	PDS10
PES0	PES07	PES06	PES05	PES04	PES03	PES02	PES01	PES00
PES1	PES17	PES16	PES15	PES14	PES13	PES12	PES11	PES10
PFS0	—	—	—	—	PFS03	PFS02	PFS01	PFS00
IFS	—	—	—	—	—	IFS2	IFS1	IFS0

Pin-shared Function Selection Register List – BS83C40C

- **PAS0 Register**

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

Bit 7~6 **PAS07~PAS06**: PA3 pin-shared function selection
00: PA3/$\overline{\text{INT}}$
01: $\overline{\text{SCS}}$
10: PTP
11: KEY2

Bit 5~4 **PAS05~PAS04**: PA2 pin-shared function selection
00: PA2
01: SCK/SCL
10: SDO/TX
11: XT1

Bit 3~2 **PAS03~PAS02**: PA1 pin-shared function selection
00/10: PA1/ PTCK
01: SCK/SCL
11: KEY1

Bit 1~0 **PAS01~PAS00**: PA0 pin-shared function selection
00/10: PA0
01: SDA/SDI/RX
11: XT2

- **PAS1 Register**

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

Bit 7~6 **PAS17~PAS16**: PA7 pin-shared function selection
00/01/10: PA7/ PTPI
11: KEY6

Bit 5~4 **PAS15~PAS14**: PA6 pin-shared function selection
00/01/10: PA6/INT
11: KEY5

Bit 3~2 **PAS13~PAS12**: PA5 pin-shared function selection
00/10: PA5
01: SDA/SDI/RX
11: KEY4

Bit 1~0 **PAS11~PAS10**: PA4 pin-shared function selection
00/10: PA4
01: SDO/TX
11: KEY3

- **PBS0 Register**

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

Bit 7~6 **PBS07~PBS06**: PB3 pin-shared function selection
00/01/10: PB3
11: KEY10

Bit 5~4 **PBS05~PBS04**: PB2 pin-shared function selection
00/01/10: PB2
11: KEY9

Bit 3~2 **PBS03~PBS02**: PB1 pin-shared function selection
00/01/10: PB1
11: KEY8

Bit 1~0 **PBS01~PBS00**: PB0 pin-shared function selection
00/01: PB0
10: PTPB
11: KEY7

- **PBS1 Register**

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

Bit 7~6 **PBS17~PBS16**: PB7 pin-shared function selection
00/01/10: PB7
11: KEY14

Bit 5~4 **PBS15~PBS14**: PB6 pin-shared function selection
00/01/10: PB6
11: KEY13

Bit 3~2 **PBS13~PBS12**: PB5 pin-shared function selection
00/01/10: PB5
11: KEY12

Bit 1~0 **PBS11~PBS10**: PB4 pin-shared function selection
00/01/10: PB4
11: KEY11

- **PCS0 Register**

Bit	7	6	5	4	3	2	1	0
Name	PCS07	PCS06	PCS05	PCS04	PCS03	PCS02	PCS01	PCS00
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 **PCS07~PCS06**: PC3 pin-shared function selection
00/01/10: PC3
11: KEY18

Bit 5~4 **PCS05~PCS04**: PC2 pin-shared function selection
00/01/10: PC2
11: KEY17

Bit 3~2 **PCS03~PCS02**: PC1 pin-shared function selection
00/01/10: PC1
11: KEY16

Bit 1~0 **PCS01~PCS00**: PC0 pin-shared function selection
00/01/10: PC0
11: KEY15

- **PCS1 Register**

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

Bit 7~6 **PCS17~PCS16**: PC7 pin-shared function selection
00/01/10: PC7
11: KEY22

Bit 5~4 **PCS15~PCS14**: PC6 pin-shared function selection
00/01/10: PC6
11: KEY21

Bit 3~2 **PCS13~PCS12**: PC5 pin-shared function selection
00/01/10: PC5
11: KEY20

Bit 1~0 **PCS11~PCS10**: PC4 pin-shared function selection
00/01/10: PC4
11: KEY19

- **PDS0 Register – BS83B24C**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	PDS03	PDS02	PDS01	PDS00
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0

Bit 7~4 Unimplemented, read as "0"

Bit 3~2 **PDS03~PDS02**: PD1 pin-shared function selection
00/01/10: PD1
11: KEY24

Bit 1~0 **PDS01~PDS00**: PD0 pin-shared function selection
00/01/10: PD0
11: KEY23

- **PDS0 Register – BS83C40C**

Bit	7	6	5	4	3	2	1	0
Name	PDS07	PDS06	PDS05	PDS04	PDS03	PDS02	PDS01	PDS00
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 **PDS07~PDS06**: PD3 pin-shared function selection
00/01/10: PD3
11: KEY26

Bit 5~4 **PDS05~PDS04**: PD2 pin-shared function selection
00/01/10: PD2
11: KEY25

Bit 3~2 **PDS03~PDS02**: PD1 pin-shared function selection
00/01/10: PD1
11: KEY24

Bit 1~0 **PDS01~PDS00**: PD0 pin-shared function selection
00/01/10: PD0
11: KEY23

- **PDS1 Register – BS83C40C**

Bit	7	6	5	4	3	2	1	0
Name	PDS17	PDS16	PDS15	PDS14	PDS13	PDS12	PDS11	PDS10
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 **PDS17~PDS16**: PD7 pin-shared function selection
00/01/10: PD7
11: KEY30

Bit 5~4 **PDS15~PDS14**: PD6 pin-shared function selection
00/01/10: PD6
11: KEY29

Bit 3~2 **PDS13~PDS12**: PD5 pin-shared function selection
00/01/10: PD5
11:KEY28

Bit 1~0 **PDS11~PDS10**: PD4 pin-shared function selection
00/01/10: PD4
11: KEY27

- **PES0 Register – BS83C40C**

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

Bit 7~6 **PES07~PES06**: PE3 pin-shared function selection
00/01: PE3
10: CTP
11: KEY34

Bit 5~4 **PES05~PES04**: PE2 pin-shared function selection
00/01/10: PE2/CTCK
11: KEY33

Bit 3~2 **PES03~PES02**: PE1 pin-shared function selection
00/01/10: PE1
11: KEY32

Bit 1~0 **PES01~PES00**: PE0 pin-shared function selection
00/01/10: PE0
11: KEY31

- **PES1 Register – BS83C40C**

Bit	7	6	5	4	3	2	1	0
Name	PES17	PES16	PES15	PES14	PES13	PES12	PES11	PES10
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 **PES17~PES16**: PE7 pin-shared function selection
00/01/10: PE7
11: KEY38

Bit 5~4 **PES15~PES14**: PE6 pin-shared function selection
00/01/10: PE6
11: KEY37

Bit 3~2 **PES13~PES12**: PE5 pin-shared function selection
00/01/10: PE5
11:KEY36

Bit 1~0 **PES11~PES10**: PE4 pin-shared function selection
00/01: PE4
10: CTPB
11: KEY35

- **PFS0 Register – BS83C40C**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	PFS03	PFS02	PFS01	PFS00
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0

Bit 7~4 Unimplemented, read as “0”

Bit 3~2 **PFS03~PFS02**: PF1 pin-shared function selection
00/01/10: PF1
11: KEY40

Bit 1~0 **PFS01~PFS00**: PF0 pin-shared function selection
00/01/10: PF0
11: KEY39

- **IFS Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	IFS2	IFS1	IFS0
R/W	—	—	—	—	—	R/W	R/W	R/W
POR	—	—	—	—	—	0	0	0

Bit 7~3 Unimplemented, read as “0”

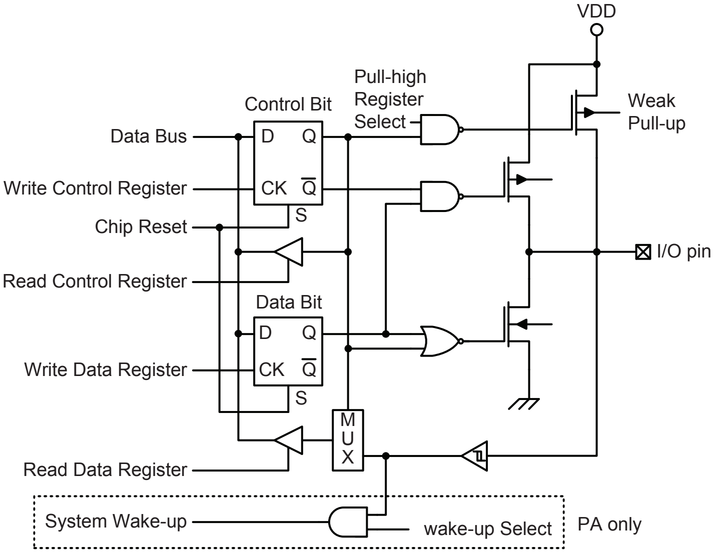
Bit 2 **IFS2**: INT input source pin selection
0: PA6
1: PA3

Bit 1 **IFS1**: USIM SCK/SCL input source pin selection
0: PA2
1: PA1

Bit 0 **IFS0**: USIM SDA/SDI/RX input source pin selection
0: PA0
1: PA5

I/O Pin Structures

The accompanying diagram illustrates the internal structures of the I/O logic function. As the exact logical construction of the I/O pin will differ from this diagram, it is supplied as a guide only to assist with the functional understanding of the logic function I/O pins. The wide range of pin-shared structures does not permit all types to be shown.


VDD
Control Bit
Pull-high Register Select
Weak Pull-up
Data Bus
D
Q
Write Control Register
CK
Q
S
Chip Reset
I/O pin
Read Control Register
Data Bit
D
Q
Write Data Register
CK
Q
S
M U X
Read Data Register
System Wake-up
wake-up Select
PA only

Logic Function Input/Output Structure

Programming Considerations

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

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

Timer Modules – TM

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

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

Introduction

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

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

TM Function Summary

Device	CTM	PTM
BS83B24C	—	10-bit PTM
BS83C40C	10-bit CTM	10-bit PTM

TM Name/Type Summary

TM Operation

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

TM Clock Source

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

TM Interrupts

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

TM External Pins

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

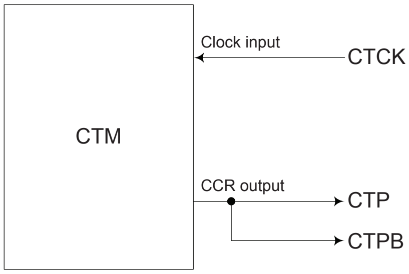
The Periodic type TM has another input pin, PTPI, which is the capture input whose active edge can be a rising edge, a falling edge or both rising and falling edges and the active edge transition type is selected using the PTIO1~PTIO0 bits in the PTMC1 register.

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

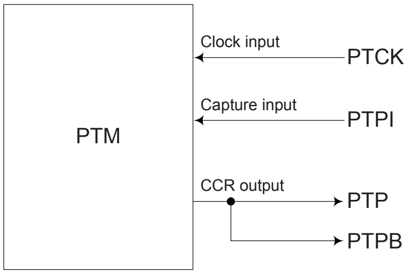
As the TM input/output pins are pin-shared with other functions, the TM input/output function must first be setup using relevant pin-shared function selection register. The details of the pin-shared function selection are described in the pin-shared function section.

Device	CTM		PTM	
	Input	Output	Input	Output
BS83B24C	—	—	PTCK, PTPI	PTP, PTPB
BS83C40C	CTCK	CTP, CTPB	PTCK, PTPI	PTP, PTPB

TM External Pins



CTM Function Pin Block Diagram

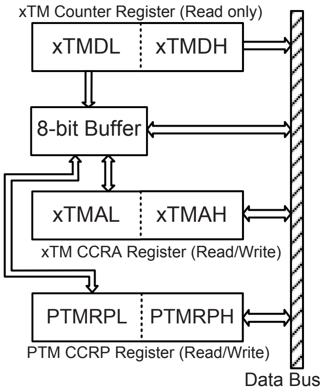


PTM Function Pin Block Diagram

Programming Considerations

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

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

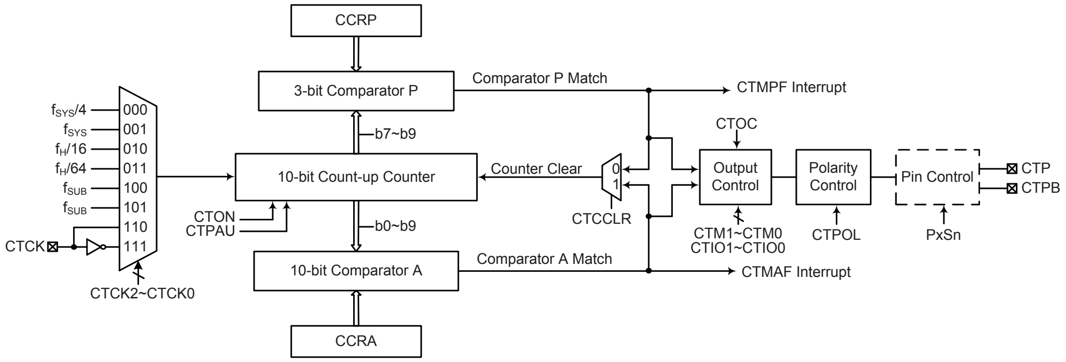

xTM Counter Register (Read only)
xTMDL
xTMDH
8-bit Buffer
xTMAL
xTMAH
xTM CCRA Register (Read/Write)
PTMRPL
PTMRPH
PTM CCRP Register (Read/Write)
Data Bus

The following steps show the read and write procedures:

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

Compact Type TM – CTM

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


CCRP
3-bit Comparator P
Comparator P Match
CTMPF Interrupt
CTOC
$f_{SYS}/4$
f_{SYS}
$f_H/16$
$f_H/64$
f_{SUB}
f_{SUB}
000
001
010
011
100
101
110
111
CTCK
b7~b9
10-bit Count-up Counter
Counter Clear
0
1
Output Control
Polarity Control
Pin Control
CTP
CTPB
CTON
CTPAU
CTCCLR
b0~b9
CTM1~CTM0
CTIO1~CTIO0
CTPOL
PxSn
10-bit Comparator A
Comparator A Match
CTMAF Interrupt
CTCK2~CTCK0
CCRA

Note: The CTPB is the inverted output of the CTP pin.

10-bit Compact Type TM Block Diagram

Compact TM Operation

At its core is a 10-bit count-up counter which is driven by a user selectable internal or external clock source. There are also two internal comparators with the names, Comparator A and Comparator P. These comparators will compare the value in the counter with CCRP and CCRA registers. The CCRP is three bits wide whose value is compared with the highest three bits in the counter while the CCRA is the ten bits and therefore compares with all counter bits.

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

Compact Type TM Register Description

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

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

10-bit Compact Type TM Register List

- **CTMC0 Register**

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

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

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

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

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

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

This bit controls the overall on/off function of the CTM. Setting the bit high enables the counter to run, clearing the bit to 0 disables the CTM. Clearing this bit to zero will stop the counter from counting and turn off the CTM which will reduce its power consumption. When the bit changes state from low to high the internal counter value will be reset to zero, however when the bit changes from high to low, the internal counter will retain its residual value until the bit returns high again.

If the CTM is in the Compare Match Output Mode or the PWM Output Mode then the CTM output pin will be reset to its initial condition, as specified by the CTOC bit, when the CTON bit changes from low to high.

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

Comparator P Match Period
000: 1024 CTM clocks
001: 128 CTM clocks
010: 256 CTM clocks
011: 384 CTM clocks
100: 512 CTM clocks
101: 640 CTM clocks
110: 768 CTM clocks
111: 896 CTM clocks

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

- **CTMC1 Register**

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

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

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

Bit 5~4 **CTIO1~CTIO0**: Select CTP output function

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

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

Timer/counter Mode
Unused

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

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

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

Bit 3 **CTOC**: CTP Output control bit

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

PWM Output Mode
0: Active low
1: Active high

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

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

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

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

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

Bit 0 **CTCCLR**: Select CTM Counter clear condition
0: CTM Comparatror P match
1: CTM Comparatror A match

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

- **CTMDL Register**

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

Bit 7~0 CTM Counter Low Byte Register bit 7 ~ bit 0
CTM 10-bit Counter bit 7 ~ bit 0

- **CTMDH Register**

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

Bit 7~2 Unimplemented, read as “0”

Bit 1~0 CTM Counter High Byte Register bit 1 ~ bit 0
CTM 10-bit Counter bit 9 ~ bit 8

- **CTMAL Register**

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

Bit 7~0	CTM CCRA Low Byte Register bit 7 ~ bit 0 CTM 10-bit CCRA bit 7 ~ bit 0

- **CTMAH Register**

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

Bit 7~2	Unimplemented, read as "0"
Bit 1~0	CTM CCRA High Byte Register bit 1 ~ bit 0 CTM 10-bit CCRA bit 9 ~ bit 8

Compact Type TM Operating Modes

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

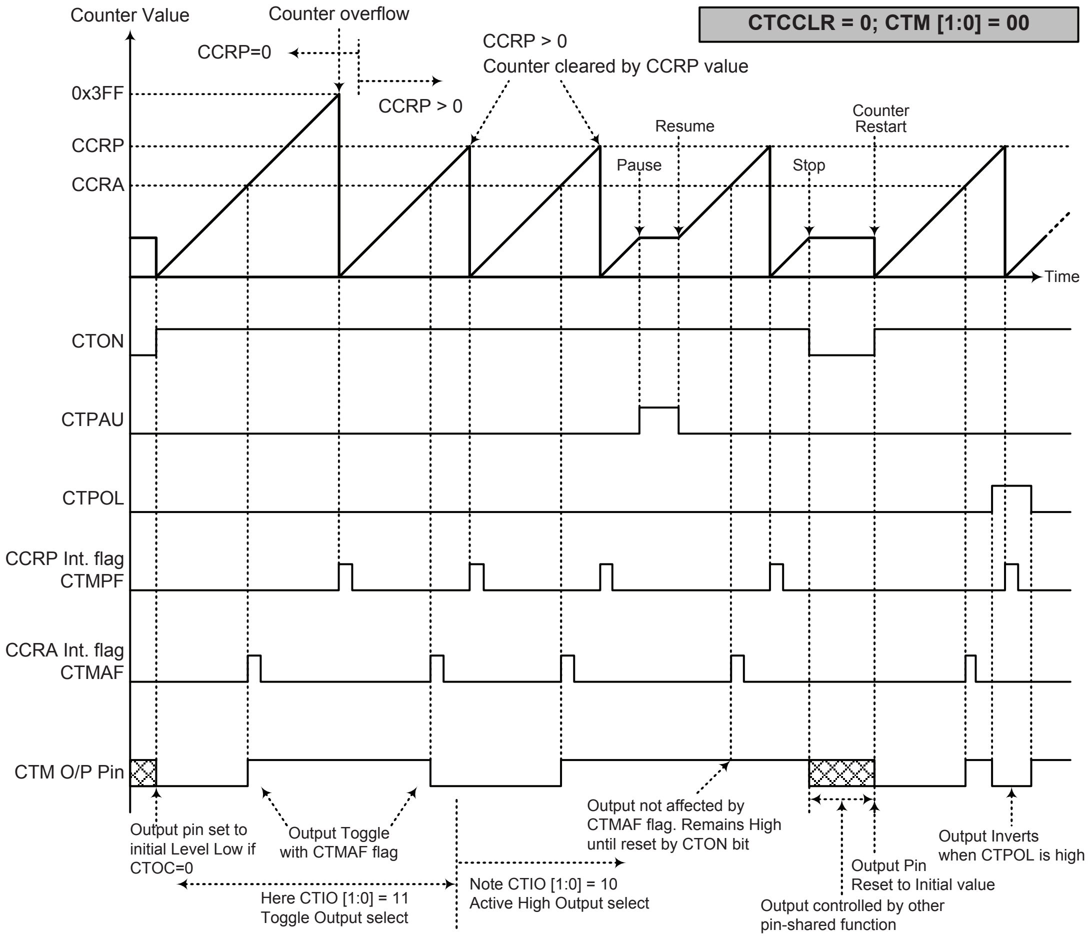
Compare Match Output Mode

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

If the CTCCLR bit in the CTMC1 register is high then the counter will be cleared when a compare match occurs from Comparator A. However, here only the CTMAF interrupt request flag will be generated even if the value of the CCRP bits is less than that of the CCRA registers. Therefore when CTCCLR is high no CTMPF interrupt request flag will be generated.

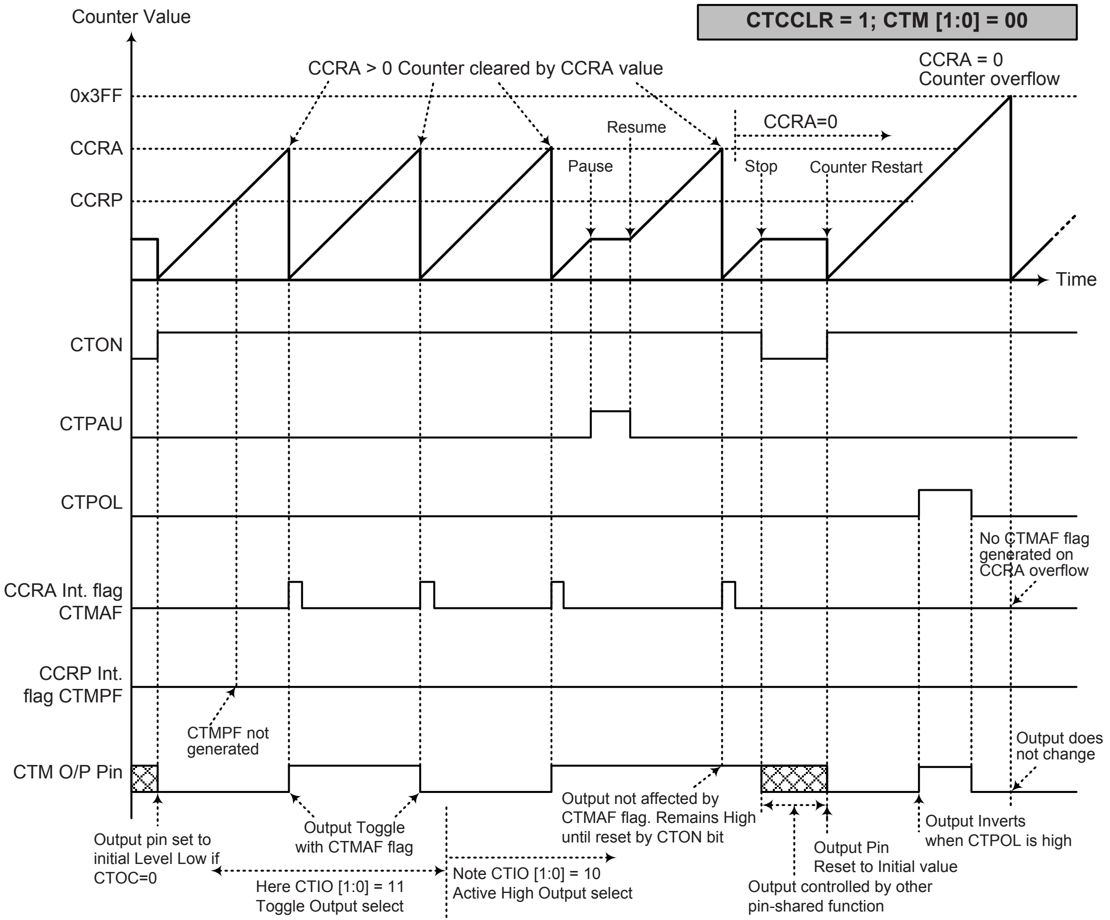
If the CCRA bits are all zero, the counter will overflow when its reaches its maximum 10-bit, 3FF Hex, value, however here the CTMAF interrupt request flag will not be generated.

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


Counter Value
Counter overflow
CTCCLR = 0; CTM [1:0] = 00
CCRP=0
CCRP > 0
CCRP > 0
Counter cleared by CCRP value
0x3FF
CCRP
CCRA
Resume
Counter Restart
Pause
Stop
Time
CTON
CTPAU
CTPOL
CCRP Int. flag CTMPF
CCRA Int. flag CTMAF
CTM O/P Pin
Output pin set to initial Level Low if CTOC=0
Output Toggle with CTMAF flag
Output not affected by CTMAF flag. Remains High until reset by CTON bit
Output Inverts when CTPOL is high
Output Pin Reset to Initial value
Here CTIO [1:0] = 11 Toggle Output select
Note CTIO [1:0] = 10 Active High Output select
Output controlled by other pin-shared function

Compare Match Output Mode – CTCCLR=0

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


Counter Value
CTCCLR = 1; CTM [1:0] = 00
CCRA > 0 Counter cleared by CCRA value
CCRA = 0
Counter overflow
0x3FF
CCRA
CCRP
Resume
CCRA=0
Pause
Stop
Counter Restart
Time
CTON
CTPAU
CTPOL
No CTMAF flag generated on CCRA overflow
CCRA Int. flag CTMAF
CCRP Int. flag CTMPF
CTMPF not generated
Output does not change
CTM O/P Pin
Output not affected by CTMAF flag. Remains High until reset by CTON bit
Output pin set to initial Level Low if CTOC=0
Output Toggle with CTMAF flag
Output Pin Reset to Initial value
Output Inverts when CTPOL is high
Here CTIO [1:0] = 11 Toggle Output select
Note CTIO [1:0] = 10 Active High Output select
Output controlled by other pin-shared function

Compare Match Output Mode – CTCCLR=1

Notes: 1. With CTCCLR=1, a Comparator A match will clear the counter
2. The CTM output pin controlled only by the CTMAF flag
3. The output pin reset to initial state by a CTON rising edge
4. The CTMPF flags is not generated when CTCCLR=1

Timer/Counter Mode

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

PWM Output Mode

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

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

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

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

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

If f_{SYS}=8MHz, CTM clock source is f_{SYS}/4, CCRP=2, CCRA=128,

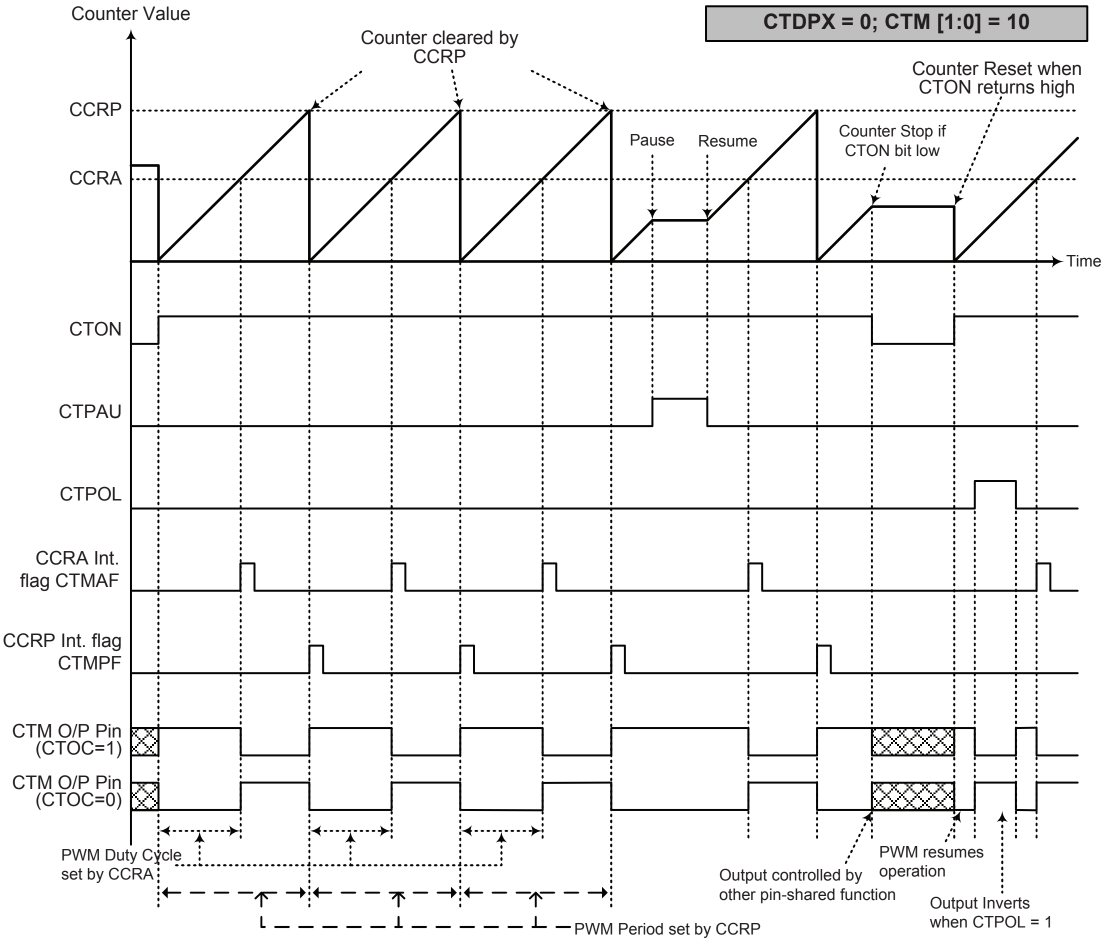
The CTM PWM output frequency=(f_{SYS}/4)/(2×128)=f_{SYS}/1024=7.812kHz, duty=128/(2×128)=50%.

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

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

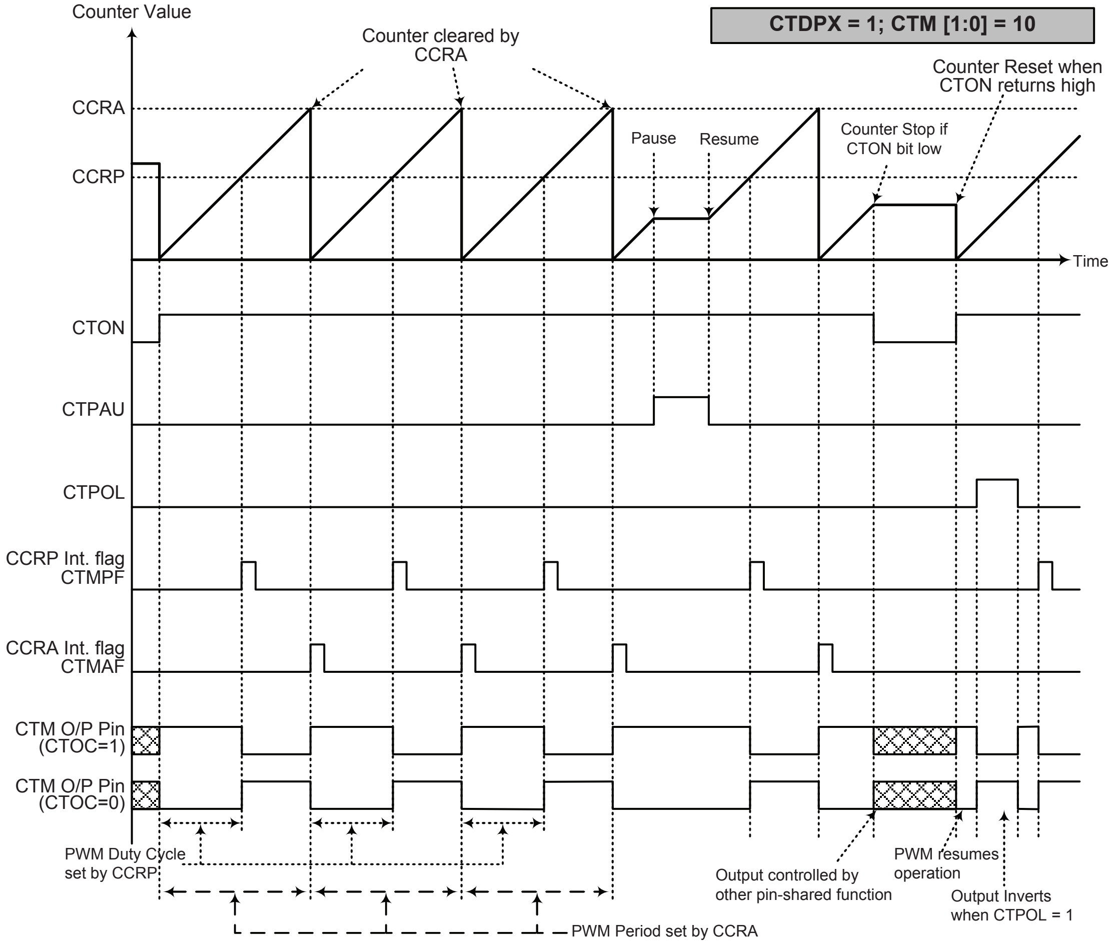
CCRP	1~7	0
Period	CCRA	
Duty	CCRP × 128	1024

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


CTDPX = 0; CTM [1:0] = 10
Counter Value
Counter cleared by CCRP
Counter Reset when CTON returns high
CCRP
CCRA
Pause
Resume
Counter Stop if CTON bit low
Time
CTON
CTPAU
CTPOL
CCRA Int. flag CTMAF
CCRP Int. flag CTMPF
CTM O/P Pin (CTOC=1)
CTM O/P Pin (CTOC=0)
PWM Duty Cycle set by CCRA
PWM resumes operation
Output controlled by other pin-shared function
Output Inverts when CTPOL = 1
PWM Period set by CCRP

PWM Output Mode – CTDPX=0

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

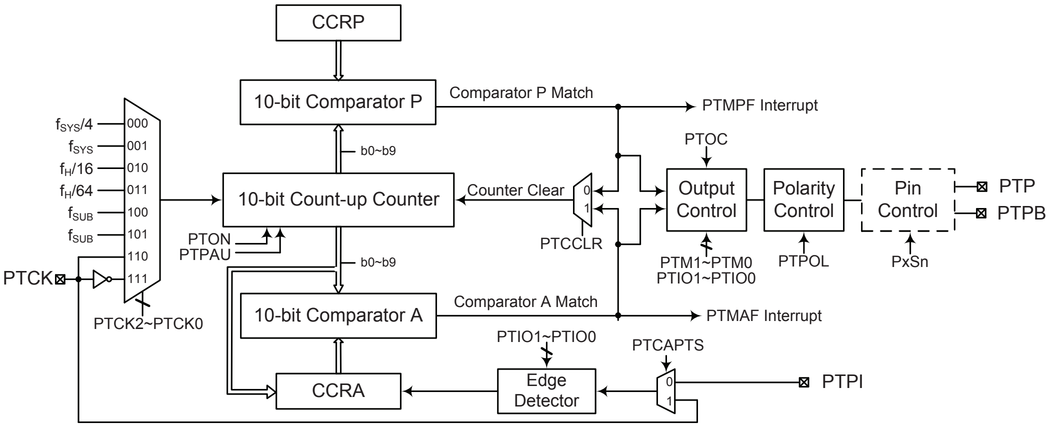

Counter Value
CTDPX = 1; CTM [1:0] = 10
Counter cleared by CCRA
Counter Reset when CTON returns high
CCRA
CCRP
Pause
Resume
Counter Stop if CTON bit low
Time
CTON
CTPAU
CTPOL
CCRP Int. flag CTMPF
CCRA Int. flag CTMAF
CTM O/P Pin (CTOC=1)
CTM O/P Pin (CTOC=0)
PWM Duty Cycle set by CCRP
Output controlled by other pin-shared function
PWM resumes operation
Output Inverts when CTPOL = 1
PWM Period set by CCRA

PWM Output Mode – CTDPX=1

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

Periodic Type TM – PTM

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


CCRP
10-bit Comparator P
Comparator P Match
PTMPF Interrupt
PTOC
b0~b9
$f_{SYS}/4$
f_{SYS}
$f_H/16$
$f_H/64$
f_{SUB}
f_{SUB}
000
001
010
011
100
101
110
111
10-bit Count-up Counter
Counter Clear
0
1
Output Control
Polarity Control
Pin Control
PTP
PTPB
PTON
PTPAU
PTCCLR
PTM1~PTM0
PTIO1~PTIO0
PTPOL
PxSn
PTCK
b0~b9
PTCK2~PTCK0
10-bit Comparator A
Comparator A Match
PTMAF Interrupt
PTIO1~PTIO0
PTCAPTS
CCRA
Edge Detector
0
1
PTPI

Note: The PTPB is the inverted output of the PTP pin.

10-bit Periodic Type TM Block Diagram

Periodic TM Operation

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

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

Periodic Type TM Register Description

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

Register Name	Bit							
	7	6	5	4	3	2	1	0
PTMC0	PTPAU	PTCK2	PTCK1	PTCK0	PTON	—	—	—
PTMC1	PTM1	PTM0	PTIO1	PTIO0	PTOC	PTPOL	PTCAPTS	PTCCLR
PTMDL	D7	D6	D5	D4	D3	D2	D1	D0
PTMDH	—	—	—	—	—	—	D9	D8
PTMAL	D7	D6	D5	D4	D3	D2	D1	D0
PTMAH	—	—	—	—	—	—	D9	D8
PTMRPL	D7	D6	D5	D4	D3	D2	D1	D0
PTMRPH	—	—	—	—	—	—	D9	D8

10-bit Periodic Type TM Register List

- **PTMDL Register**

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

Bit 7~0 PTM Counter Low Byte Register bit 7 ~ bit 0
PTM 10-bit Counter bit 7 ~ bit 0

- **PTMDH Register**

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

Bit 7~2 Unimplemented, read as “0”

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

- **PTMAL Register**

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

Bit 7~0 PTM CCRA Low Byte Register bit 7 ~ bit 0
PTM 10-bit CCRA bit 7 ~ bit 0

- **PTMAH Register**

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

Bit 7~2 Unimplemented, read as “0”

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

- **PTMRPL Register**

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

Bit 7~0 PTM CCRP Low Byte Register bit 7 ~ bit 0
PTM 10-bit CCRP bit 7 ~ bit 0

- **PTMRPH Register**

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

Bit 7~2 Unimplemented, read as “0”

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

- **PTMC0 Register**

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

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

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

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

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

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

This bit controls the overall on/off function of the PTM. Setting the bit high enables the counter to run while clearing the bit to 0 disables the PTM. Clearing this bit to zero will stop the counter from counting and turn off the PTM which will reduce its power consumption. When the bit changes state from low to high the internal counter value will be reset to zero, however when the bit changes from high to low, the internal counter will retain its residual value until the bit returns high again. If the PTM is in the Compare Match Output Mode then the PTM output pin will be reset to its initial condition, as specified by the PTOC bit, when the PTON bit changes from low to high.

Bit 2~0 Unimplemented, read as “0”

- **PTMC1 Register**

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

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

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

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

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

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

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

Timer/Counter Mode
Unused

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

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

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

Bit 3	**PTOC**: PTM PTP Output control Compare Match Output Mode 0: Initial low 1: Initial high PWM Output Mode/Single Pulse Output Mode 0: Active low 1: Active high This is the output control bit for the PTM output pin. Its operation depends upon whether PTM is being used in the Compare Match Output Mode or in the PWM Output Mode/Single Pulse Output Mode. It has no effect if the PTM is in the Timer/Counter Mode. In the Compare Match Output Mode it determines the logic level of the PTM output pin before a compare match occurs. In the PWM Output Mode/Single Pulse Output Mode it determines if the PWM signal is active high or active low.
Bit 2	**PTPOL**: PTM PTP Output polarity control 0: Non-invert 1: Invert This bit controls the polarity of the PTP output pin. When the bit is set high the PTM output pin will be inverted and not inverted when the bit is zero. It has no effect if the PTM is in the Timer/Counter Mode.
Bit 1	**PTCAPTS**: PTM Capture Trigger Source selection 0: From PTPI pin 1: From PTCK pin
Bit 0	**PTCCLR**: PTM Counter Clear condition selection 0: Comparator P match 1: Comparator A match This bit is used to select the method which clears the counter. Remember that the Periodic TM contains two comparators, Comparator A and Comparator P, either of which can be selected to clear the internal counter. With the PTCCLR bit set high, the counter will be cleared when a compare match occurs from the Comparator A. When the bit is low, the counter will be cleared when a compare match occurs from the Comparator P or with a counter overflow. A counter overflow clearing method can only be implemented if the CCRP bits are all cleared to zero. The PTCCLR bit is not used in the PWM Output, Single Pulse Output or Capture Input Mode.

Periodic Type TM Operation Modes

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

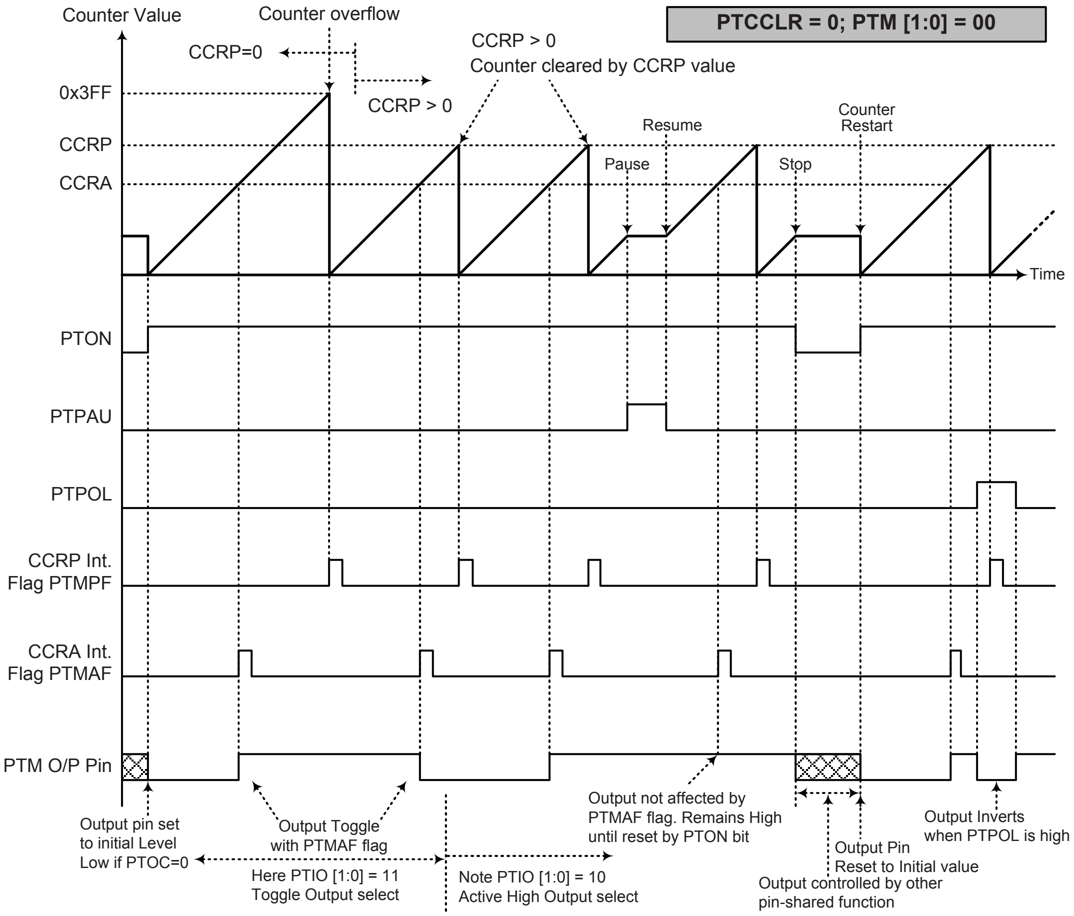
Compare Match Output Mode

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

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

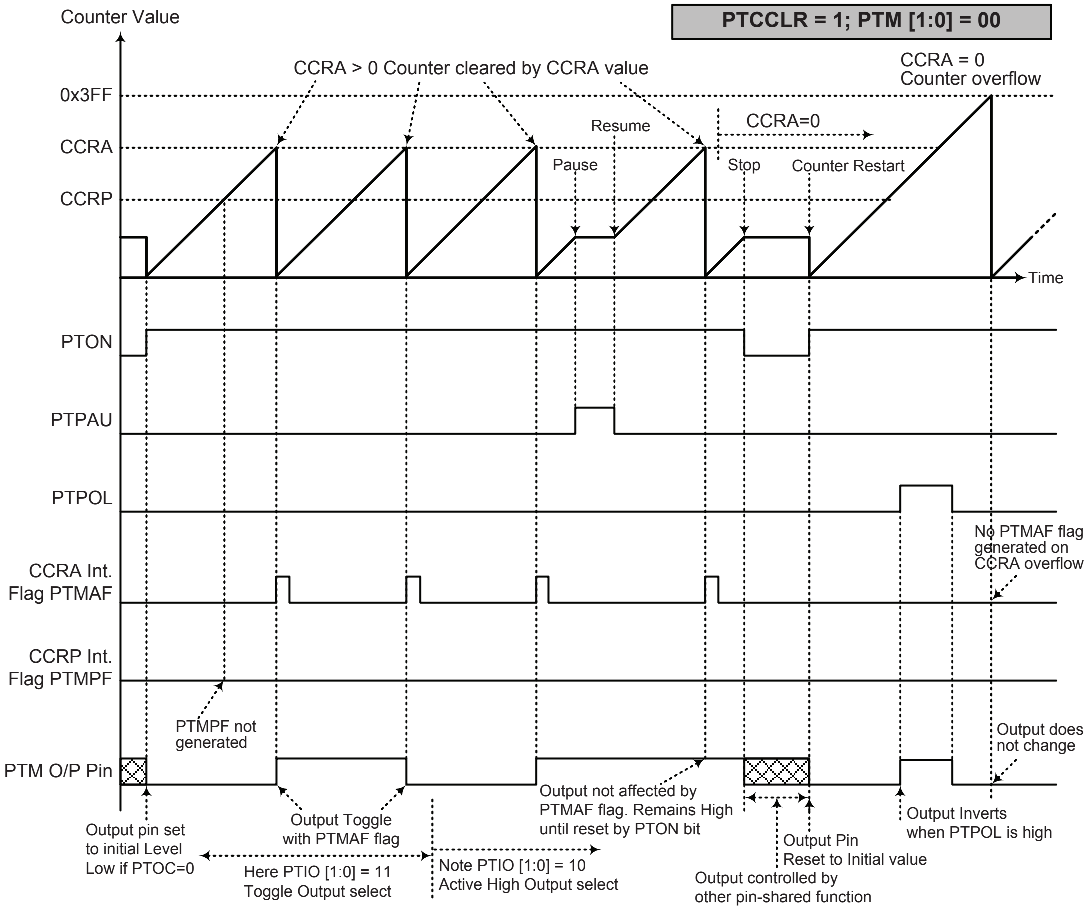
If the CCRA bits are all zero, the counter will overflow when its reaches its maximum 10-bit, 3FF Hex, value, however here the PTMnAF interrupt request flag will not be generated.

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


PTCCLR = 0; PTM [1:0] = 00
Counter Value
Counter overflow
CCRP=0
CCRP > 0
CCRP > 0
Counter cleared by CCRP value
0x3FF
CCRP
CCRA
Resume
Pause
Counter Restart
Stop
Time
PTON
PTPAU
PTPOL
CCRP Int. Flag PTMPF
CCRA Int. Flag PTMAF
PTM O/P Pin
Output pin set to initial Level Low if PTOC=0
Output Toggle with PTMAF flag
Here PTIO [1:0] = 11 Toggle Output select
Note PTIO [1:0] = 10 Active High Output select
Output not affected by PTMAF flag. Remains High until reset by PTON bit
Output Pin Reset to Initial value
Output controlled by other pin-shared function
Output Inverts when PTPOL is high

Compare Match Output Mode – PTCCLR=0

Notes: 1. With PTCCLR=0, a Comparator P match will clear the counter
2. The PTM output pin is controlled only by the PTMAF flag
3. The output pin is reset to its initial state by a PTON bit rising edge
4. The 10-bit PTM maximum counter value is 0x3FF


PTCCLR = 1; PTM [1:0] = 00
Counter Value
0x3FF
CCRA
CCRP
CCRA > 0 Counter cleared by CCRA value
CCRA = 0
Counter overflow
Resume
CCRA=0
Pause
Stop
Counter Restart
Time
PTON
PTPAU
PTPOL
CCRA Int.
Flag PTMAF
No PTMAF flag generated on CCRA overflow
CCRP Int.
Flag PTMPF
PTMPF not generated
Output does not change
PTM O/P Pin
Output not affected by PTMAF flag. Remains High until reset by PTON bit
Output pin set to initial Level Low if PTOC=0
Output Toggle with PTMAF flag
Here PTIO [1:0] = 11 Toggle Output select
Note PTIO [1:0] = 10 Active High Output select
Output Pin Reset to Initial value
Output controlled by other pin-shared function
Output Inverts when PTPOL is high

Compare Match Output Mode – PTCCLR=1

Notes: 1. With PTCCLR=1, a Comparator A match will clear the counter
2. The PTM output pin is controlled only by the PTMAF flag
3. The output pin is reset to its initial state by a PTON bit rising edge
4. A PTMPF flag is not generated when PTCCLR =1
5. The 10-bit PTM maximum counter value is 0x3FF

Timer/Counter Mode

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

PWM Output Mode

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

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

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

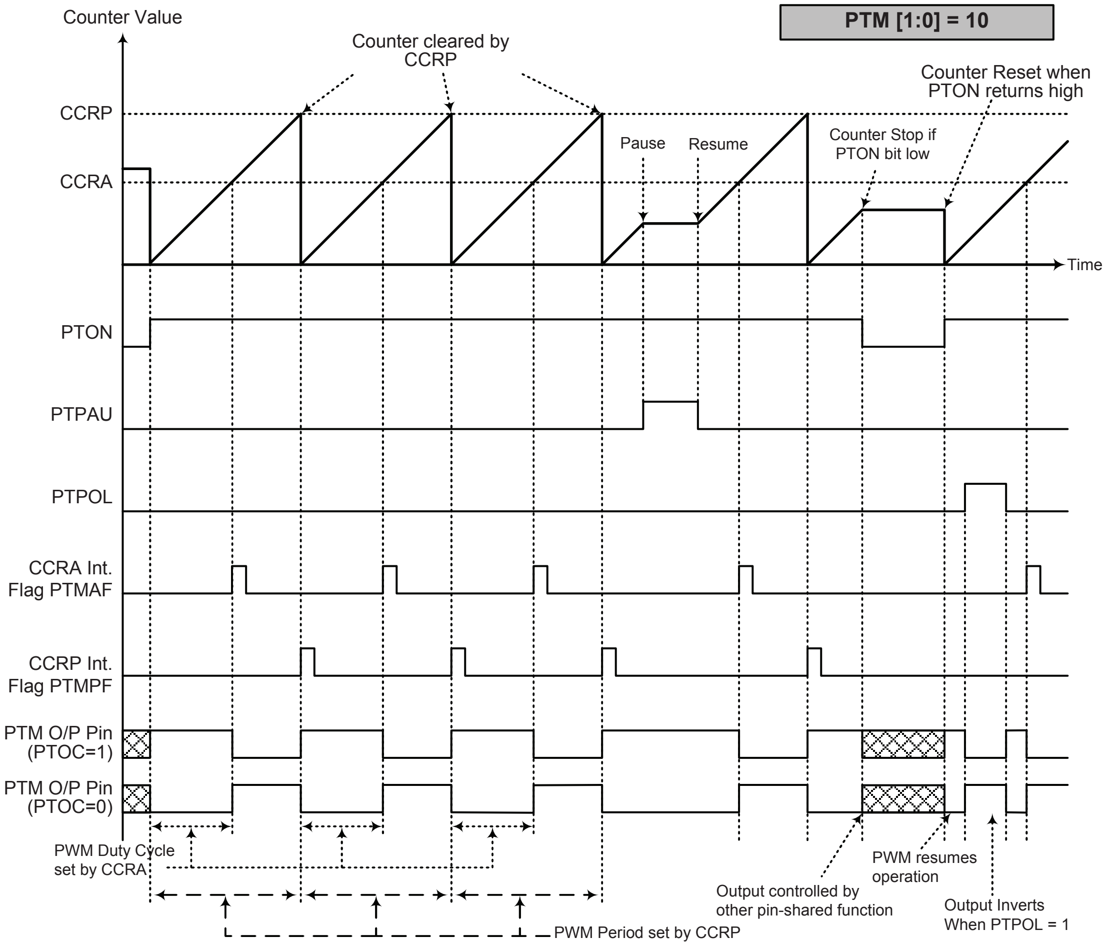
- **10-bit PTM, PWM Output Mode, Edge-aligned Mode**

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

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

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

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


Counter Value
PTM [1:0] = 10
Counter cleared by CCRP
Counter Reset when PTON returns high
CCRP
CCRA
Pause
Resume
Counter Stop if PTON bit low
Time
PTON
PTPAU
PTPOL
CCRA Int. Flag PTMAF
CCRP Int. Flag PTMPF
PTM O/P Pin (PTOC=1)
PTM O/P Pin (PTOC=0)
PWM Duty Cycle set by CCRA
PWM resumes operation
Output controlled by other pin-shared function
Output Inverts When PTPOL = 1
PWM Period set by CCRP

PWM Output Mode

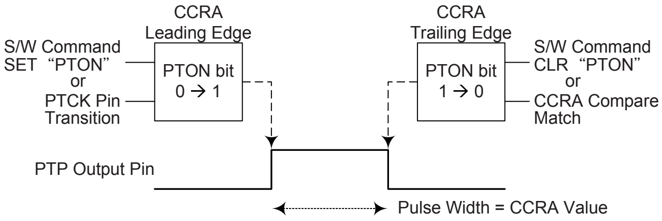
Notes: 1. The counter is cleared by CCRP
2. A counter clear sets the PWM Period
3. The internal PWM function continues running even when PTIO [1:0]=00 or 01
4. The PTCCLR bit has no influence on PWM operation

Single Pulse Output Mode

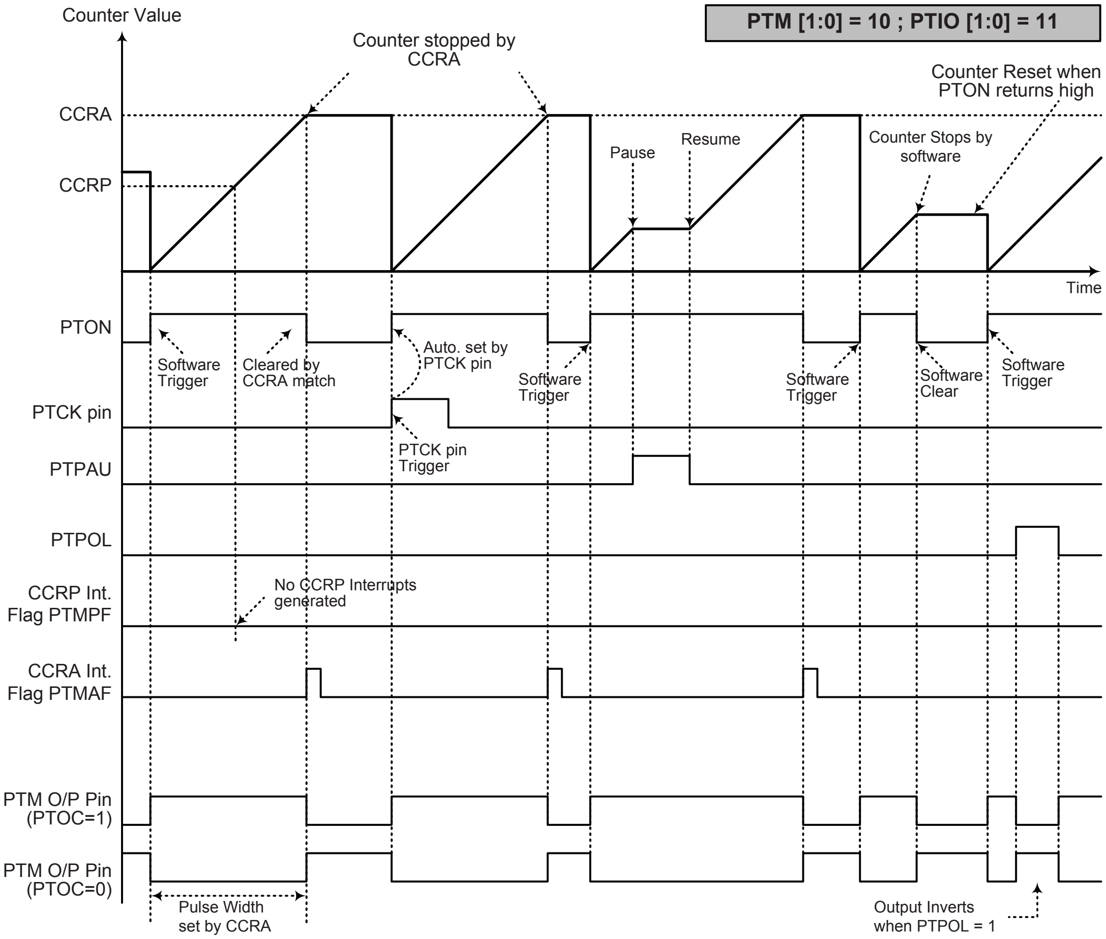
To select this mode, bits PTM1 and PTM0 in the PTMC1 register should be set to 10 respectively and also the PTIO1 and PTIO0 bits should be set to 11 respectively. The Single Pulse Output Mode, as the name suggests, will generate a single shot pulse on the PTM output pin.

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

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


CCRA
Leading Edge
S/W Command
SET "PTON"
or
PTCK Pin
Transition
PTON bit
0 → 1
CCRA
Trailing Edge
PTON bit
1 → 0
S/W Command
CLR "PTON"
or
CCRA Compare
Match
PTP Output Pin
Pulse Width = CCRA Value

Single Pulse Generation


Counter Value
PTM [1:0] = 10 ; PTIO [1:0] = 11
Counter stopped by CCRA
Counter Reset when PTON returns high
CCRA
CCRP
Pause
Resume
Counter Stops by software
Time
PTON
Software Trigger
Cleared by CCRA match
Auto. set by PTCK pin
Software Trigger
Software Trigger
Software Clear
Software Trigger
PTCK pin
PTCK pin Trigger
PTPAU
PTPOL
CCRP Int. Flag PTMPF
No CCRP Interrupts generated
CCRA Int. Flag PTMAF
PTM O/P Pin (PTOC=1)
PTM O/P Pin (PTOC=0)
Pulse Width set by CCRA
Output Inverts when PTPOL = 1

Single Pulse Output Mode

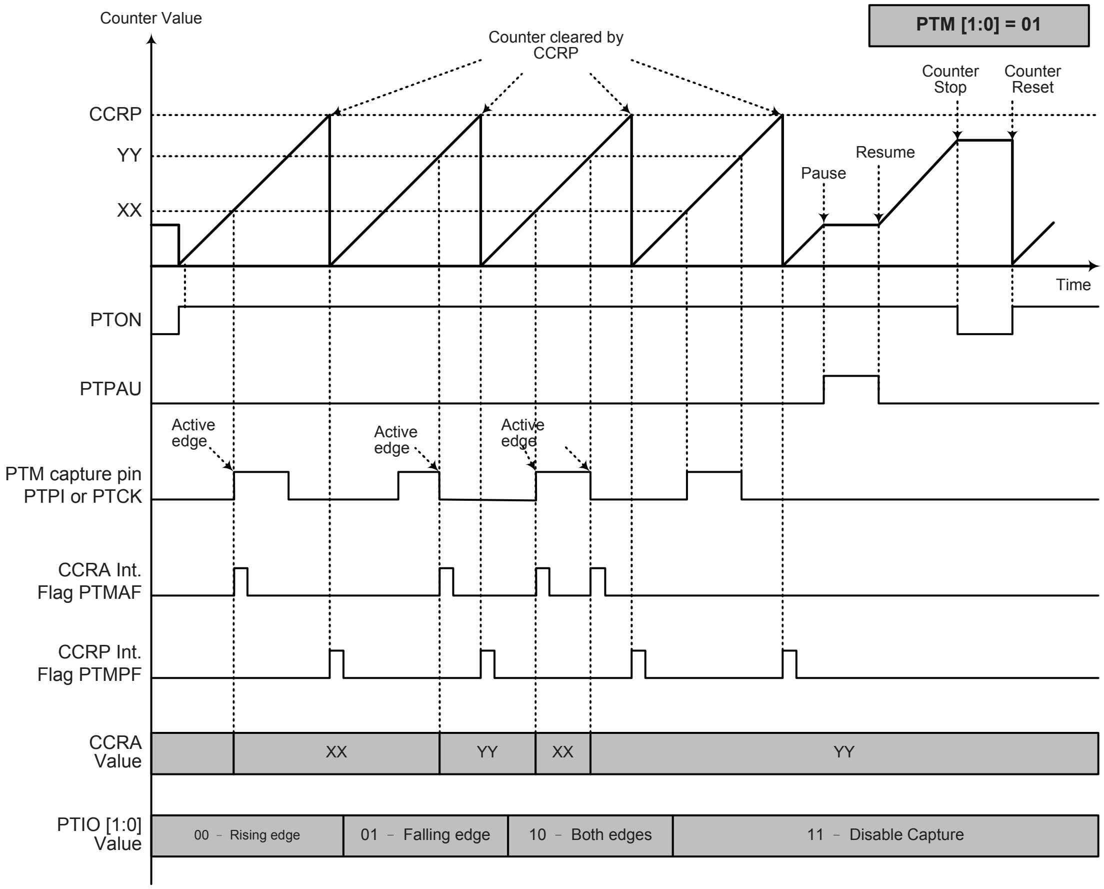
Notes: 1. Counter stopped by CCRA
2. CCRP is not used
3. The pulse triggered by the PTCK pin or by setting the PTON bit high
4. A PTCK pin active edge will automatically set the PTON bit high
5. In the Single Pulse Output Mode, PTIO [1:0] must be set to "11" and can not be changed

Capture Input Mode

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

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

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


Counter Value
PTM [1:0] = 01
Counter cleared by CCRP
Counter Stop
Counter Reset
CCRP
YY
XX
Resume
Pause
Time
PTON
PTPAU
Active edge
Active edge
Active edge
PTM capture pin PTPI or PTCK
CCRA Int. Flag PTMAF
CCRP Int. Flag PTMPF
CCRA Value
XX
YY
XX
YY
PTIO [1:0] Value
00 – Rising edge
01 – Falling edge
10 – Both edges
11 – Disable Capture

Capture Input Mode

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

Universal Serial Interface Module – USIM

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

SPI Interface

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

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

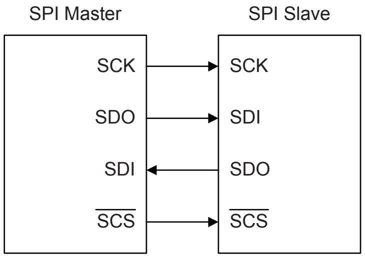
SPI Interface Operation

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

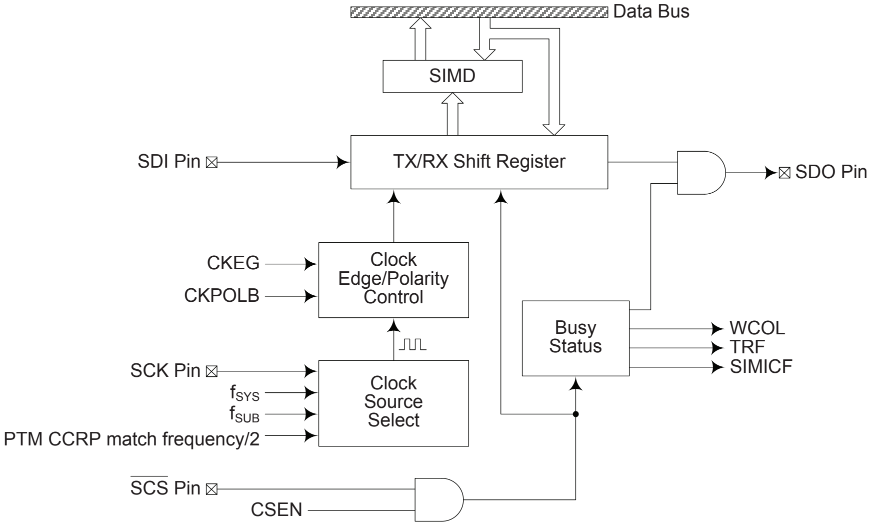
The SPI function in these devices offer the following features:

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

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


SPI Master
SPI Slave
SCK
SDO
SDI
SCS
SCK
SDI
SDO
SCS

SPI Master/Slave Connection


Data Bus
SIMD
SDI Pin
TX/RX Shift Register
SDO Pin
CKEG
CKPOLB
Clock Edge/Polarity Control
Busy Status
WCOL
TRF
SIMICF
SCK Pin
f_{SYS}
f_{SUB}
PTM CCRP match frequency/2
Clock Source Select
SCS Pin
CSEN

SPI Block Diagram

SPI Registers

There are three internal registers which control the overall operation of the SPI interface. These are the SIMD data register and two control registers, SIMC0 and SIMC2. Note that the SIMC2 and SIMD registers and their POR values are only available when the SPI mode is selected by properly configuring the UMD and SIM2~SIM0 bits in the SIMC0 register.

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

SPI Register List

SPI Data Register

The SIMD register is used to store the data being transmitted and received. The same register is used by both the SPI and 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

Bit 7~0 **D7~D0**: USIM SPI/I^{2}C data register bit 7 ~ bit 0

SPI Control Registers

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

- **SIMC0 Register**

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

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

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

Bit 4 **UMD**: UART mode selection bit
0: SPI or I^{2}C mode
1: UART mode

This bit is used to select the UART mode. When this bit is cleared to zero, the actual SPI or I^{2}C mode can be selected using the SIM2~SIM0 bits. Note that the UMD bit must be set low for SPI or I^{2}C mode.

Bit 3~2 **SIMDEB1~SIMDEB0**: I^{2}C Debounce Time Selection

These bits are only available when the USIM is configured to operate in the I^{2}C mode. Refer to the I^{2}C register section.

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

0: Disable
1: Enable

The bit is the overall on/off control for the USIM SPI/I²C interface. When the SIMEN bit is cleared to zero to disable the USIM SPI/I²C interface, the SDI, SDO, SCK and $\overline{SCS}$, or SDA and SCL lines will lose their SPI or I²C function and the USIM operating current will be reduced to a minimum value. When the bit is high the USIM SPI/I²C interface is enabled. If the USIM is configured to operate as an SPI interface via the UMD and SIM2~SIM0 bits, the contents of the SPI control registers will remain at the previous settings when the SIMEN bit changes from low to high and should therefore be first initialised by the application program. If the USIM is configured to operate as an I²C interface via the UMD and SIM2~SIM0 bits and the SIMEN bit changes from low to high, the contents of the I²C control bits such as HTX and TXAK will remain at the previous settings and should therefore be first initialised by the application program while the relevant I²C flags such as HCF, HAAS, HBB, SRW and RXAK will be set to their default states.

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

0: USIM SPI incomplete condition is not occurred
1: USIM SPI incomplete condition is occurred

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

- **SIMC2 Register**

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

Bit 7~6 **D7~D6**: Undefined bits

These bits can be read or written by the application program.

Bit 5 **CKPOLB**: SPI clock line base condition selection

0: The SCK line will be high when the clock is inactive
1: The SCK line will be low when the clock is inactive

The CKPOLB bit determines the base condition of the clock line, if the bit is high, then the SCK line will be low when the clock is inactive. When the CKPOLB bit is low, then the SCK line will be high when the clock is inactive.

Bit 4 **CKEG**: SPI SCK clock active edge type selection

CKPOLB=0
0: SCK is high base level and data capture at SCK rising edge
1: SCK is high base level and data capture at SCK falling edge

CKPOLB=1
0: SCK is low base level and data capture at SCK falling edge
1: SCK is low base level and data capture at SCK rising edge

The CKEG and CKPOLB bits are used to setup the way that the clock signal outputs and inputs data on the SPI bus. These two bits must be configured before data transfer is executed otherwise an erroneous clock edge may be generated. The CKPOLB bit determines the base condition of the clock line, if the bit is high, then the SCK line will be low when the clock is inactive. When the CKPOLB bit is low, then the SCK line will be high when the clock is inactive. The CKEG bit determines active clock edge type which depends upon the condition of CKPOLB bit.

Bit 3 **MLS**: SPI data shift order
0: LSB first
1: MSB first

This is the data shift select bit and is used to select how the data is transferred, either MSB or LSB first. Setting the bit high will select MSB first and low for LSB first.

Bit 2 **CSEN**: SPI $\overline{SCS}$ pin control
0: Disable
1: Enable

The CSEN bit is used as an enable/disable for the $\overline{SCS}$ pin. If this bit is low, then the $\overline{SCS}$ pin will be disabled and placed into a floating condition. If the bit is high the $\overline{SCS}$ pin will be enabled and used as a select pin.

Bit 1 **WCOL**: SPI write collision flag
0: No collision
1: Collision

The WCOL flag is used to detect if a data collision has occurred. If this bit is high it means that data has been attempted to be written to the SIMD register during a data transfer operation. This writing operation will be ignored if data is being transferred. The bit can be cleared to 0 by the application program.

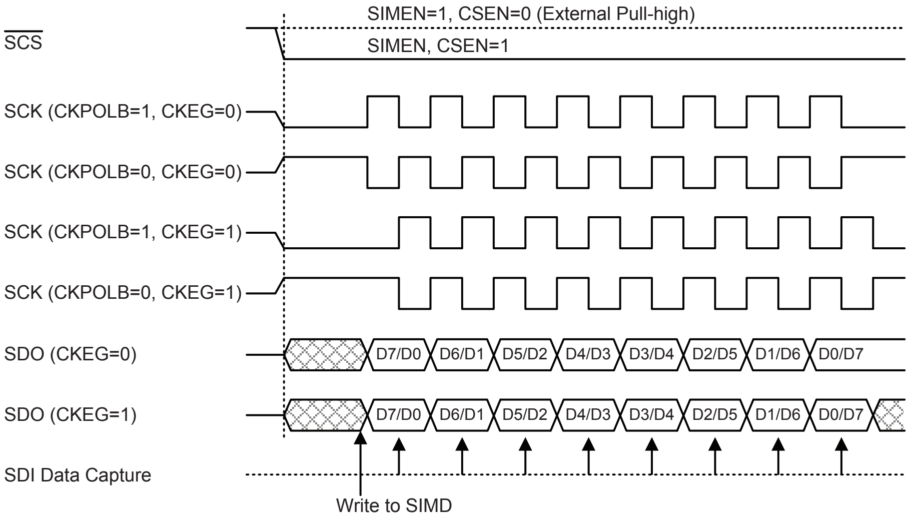
Bit 0 **TRF**: SPI Transmit/Receive complete flag
0: SPI data is being transferred
1: SPI data transmission is completed

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

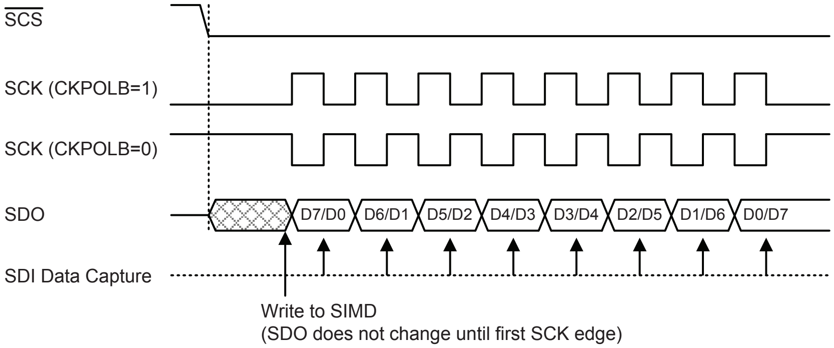
SPI Communication

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

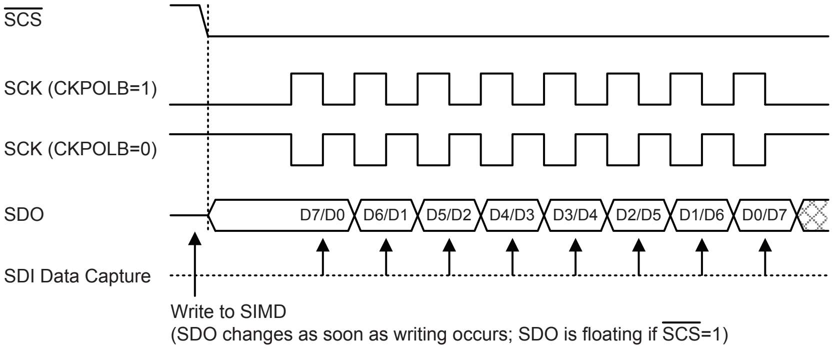
The SPI will continue to function in certain IDLE Modes if the clock source used by the SPI interface is still active.


SIMEN=1, CSEN=0 (External Pull-high)
SIMEN, CSEN=1
$\overline{\text{SCS}}$
SCK (CKPOLB=1, CKEG=0)
SCK (CKPOLB=0, CKEG=0)
SCK (CKPOLB=1, CKEG=1)
SCK (CKPOLB=0, CKEG=1)
SDO (CKEG=0)
SDO (CKEG=1)
D7/D0
D6/D1
D5/D2
D4/D3
D3/D4
D2/D5
D1/D6
D0/D7
SDI Data Capture
Write to SIMD

SPI Master Mode Timing

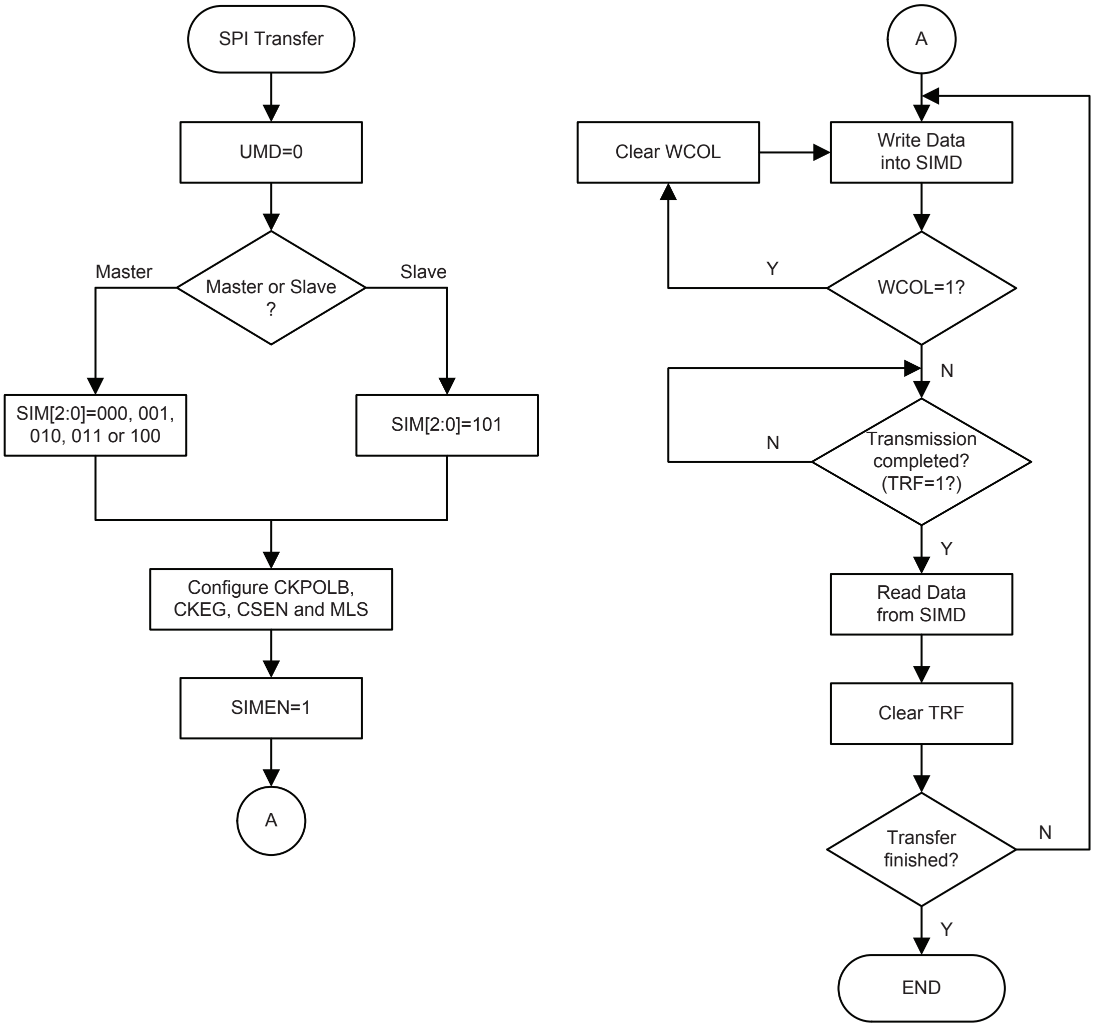

$\overline{\text{SCS}}$
SCK (CKPOLB=1)
SCK (CKPOLB=0)
SDO
D7/D0
D6/D1
D5/D2
D4/D3
D3/D4
D2/D5
D1/D6
D0/D7
SDI Data Capture
Write to SIMD
(SDO does not change until first SCK edge)

SPI Slave Mode Timing – CKEG=0


$\overline{\text{SCS}}$
SCK (CKPOLB=1)
SCK (CKPOLB=0)
SDO
D7/D0
D6/D1
D5/D2
D4/D3
D3/D4
D2/D5
D1/D6
D0/D7
SDI Data Capture
Write to SIMD
(SDO changes as soon as writing occurs; SDO is floating if $\overline{\text{SCS}}$=1)

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

SPI Slave Mode Timing – CKEG=1


SPI Transfer
UMD=0
Master or Slave ?
Master
Slave
SIM[2:0]=000, 001, 010, 011 or 100
SIM[2:0]=101
Configure CKPOLB, CKEG, CSEN and MLS
SIMEN=1
A
A
Clear WCOL
Write Data into SIMD
Y
WCOL=1?
N
N
Transmission completed? (TRF=1?)
Y
Read Data from SIMD
Clear TRF
Transfer finished?
N
Y
END

SPI Transfer Control Flowchart

SPI Bus Enable/Disable

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

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

SPI Operation Steps

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

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

Master Mode

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

Slave Mode

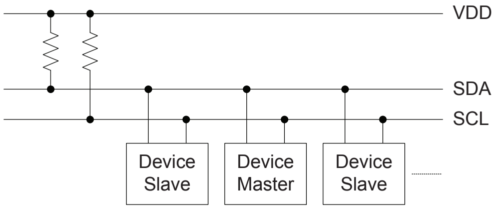
- Step 1
  Select the SPI Slave mode using the UMD and SIM2~SIM0 bits in the SIMC0 control register
- Step 2
  Setup the CSEN bit and setup the MLS bit to choose if the data is MSB or LSB first, this setting must be the same with the Master devices.
- Step 3
  Setup the SIMEN bit in the SIMC0 control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

Error Detection

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

I^{2}C Interface

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

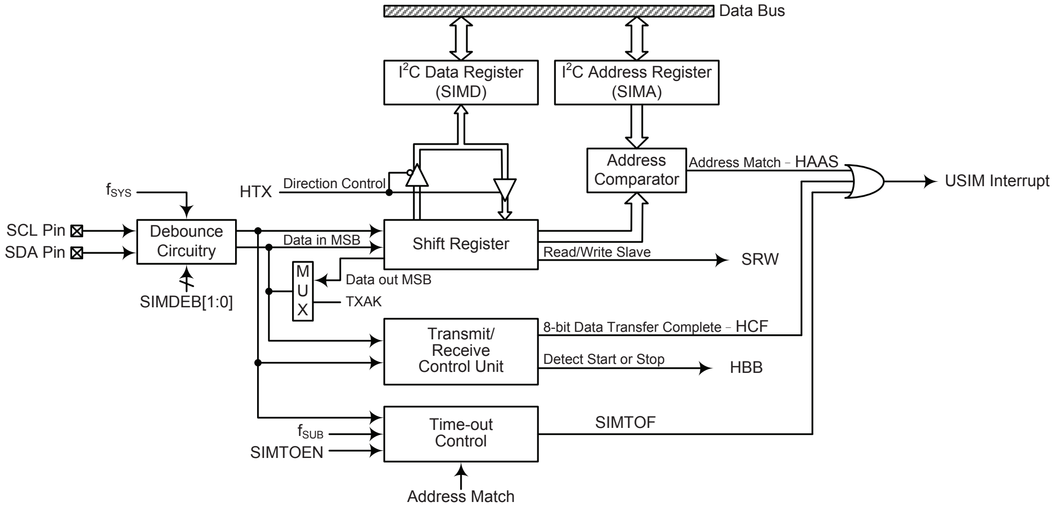


I^{2}C Master Slave Bus Connection

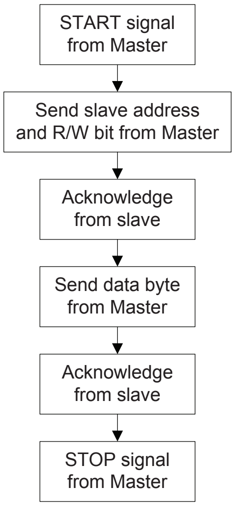
I²C Interface Operation

The I²C serial interface is a two line interface, a serial data line, SDA, and serial clock line, SCL. As many devices may be connected together on the same bus, their outputs are both open drain types. For this reason it is necessary that external pull-high resistors are connected to these outputs. Note that no chip select line exists, as each device on the I²C bus is identified by a unique address which will be transmitted and received on the I²C bus.

When two devices communicate with each other on the bidirectional I²C bus, one is known as the master device and one as the slave device. Both master and slave can transmit and receive data, however, it is the master device that has overall control of the bus. For the device, which only operates in slave mode, there are two methods of transferring data on the I²C bus, the slave transmit mode and the slave receive mode. The pull-high control function pin-shared with SCL/SDA pin is still applicable even if I²C device is activated and the related internal pull-high register could be controlled by its corresponding pull-high control register.


Data Bus
I²C Data Register (SIMD)
I²C Address Register (SIMA)
Address Comparator
Address Match – HAAS
USIM Interrupt
f_SYS
HTX
Direction Control
SCL Pin
SDA Pin
Debounce Circuitry
SIMDEB[1:0]
Data in MSB
Shift Register
Read/Write Slave
SRW
M U X
Data out MSB
TXAK
Transmit/ Receive Control Unit
8-bit Data Transfer Complete – HCF
Detect Start or Stop
HBB
Time-out Control
f_SUB
SIMTOEN
SIMTOF
Address Match

I²C Block Diagram


START signal from Master
Send slave address and R/W bit from Master
Acknowledge from slave
Send data byte from Master
Acknowledge from slave
STOP signal from Master

I²C Interface Operation

The SIMDEB1 and SIMDEB0 bits determine the debounce time of the I²C interface. This uses the internal clock to in effect add a debounce time to the external clock to reduce the possibility of glitches on the clock line causing erroneous operation. The debounce time, if selected, can be chosen to be either 2 or 4 system clocks. To achieve the required I²C data transfer speed, there exists a relationship between the system clock, f_{SYS}, and the I²C debounce time. For either the I²C Standard or Fast mode operation, users must take care of the selected system clock frequency and the configured debounce time to match the criterion shown in the following table.

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

I²C Minimum f_{SYS} Frequency Requirements

I²C Registers

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

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

I²C Register List

I²C Data Register

The SIMD register is used to store the data being transmitted and received. The same register is used by both the SPI and I²C functions. Before the device writes data to the I²C bus, the actual data to be transmitted must be placed in the SIMD register. After the data is received from the I²C bus, the device can read it from the SIMD register. Any transmission or reception of data from the I²C bus must be made via the SIMD register.

- **SIMD Register**

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

"x": Unknown

Bit 7~0 **D7~D0**: USIM SPI/I²C data register bit 7 ~ bit 0

I²C Address Register

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

- **SIMA Register**

Bit	7	6	5	4	3	2	1	0
Name	SIMA6	SIMA5	SIMA4	SIMA3	SIMA2	SIMA1	SIMA0	D0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~1 **SIMA6~SIMA0**: I^{2}C slave address
SIMA6~SIMA0 is the I^{2}C slave address bit 6~bit 0.

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

I^{2}C Control Registers

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

- **SIMC0 Register**

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

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

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

Bit 4 **UMD**: UART mode selection bit
0: SPI or I^{2}C mode
1: UART mode

This bit is used to select the UART mode. When this bit is cleared to zero, the actual SPI or I^{2}C mode can be selected using the SIM2~SIM0 bits. Note that the UMD bit must be set low for SPI or I^{2}C mode.

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

These bits are used to select the I^{2}C debounce time when the USIM is configured as the I^{2}C interface function by setting the UMD bit to “0” and the SIM2~SIM0 bits to “110”.

Bit 1 **SIMEN**: USIM SPI/I^{2}C Enable Control
0: Disable
1: Enable

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

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

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

- **SIMC1 Register**

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

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

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

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

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

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

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

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

Bit 3 **TXAK**: I^{2}C Bus transmit acknowledge flag
0: Slave send acknowledge flag
1: Slave do not send acknowledge flag

The TXAK bit is the transmit acknowledge flag. After the slave device receipt of 8 bits of data, this bit will be transmitted to the bus on the 9th clock from the slave device. The slave device must always set TXAK bit to “0” before further data is received.

Bit 2 **SRW**: I²C Slave Read/Write flag
0: Slave device should be in receive mode
1: Slave device should be in transmit mode

The SRW flag is the I²C Slave Read/Write flag. This flag determines whether the master device wishes to transmit or receive data from the I²C bus. When the transmitted address and slave address 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**: I²C Address Match Wake-up control
0: Disable
1: Enable

This bit should be set to 1 to enable the I²C address match wake up from the SLEEP or IDLE Mode. If the IAMWU bit has been set before entering either the SLEEP or IDLE mode to enable the I²C address match wake up, then this bit must be cleared to 0 by the application program after wake-up to ensure correction device operation.

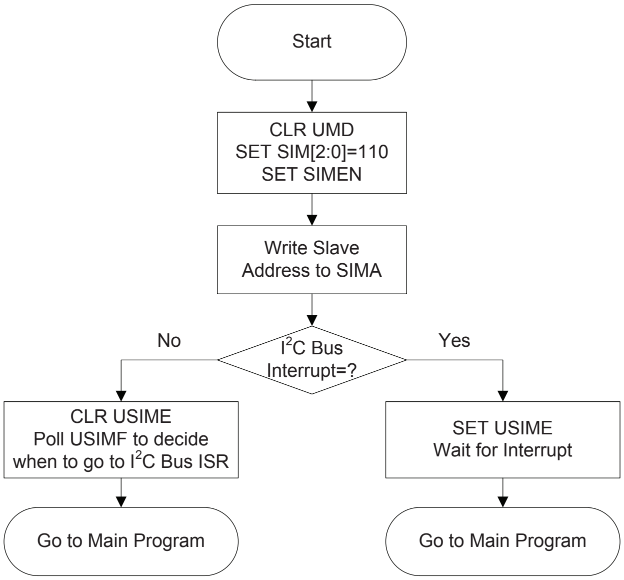
Bit 0 **RXAK**: I²C Bus Receive acknowledge flag
0: Slave receive acknowledge flag
1: Slave does not receive acknowledge flag

The RXAK flag is the receiver acknowledge flag. When the RXAK flag is "0", it means that a acknowledge signal has been received at the 9th clock, after 8 bits of data have been transmitted. When the slave device in the transmit mode, the slave device checks the RXAK flag to determine if the master receiver wishes to receive the next byte. The slave transmitter will therefore continue sending out data until the RXAK flag is "1". When this occurs, the slave transmitter will release the SDA line to allow the master to send a STOP signal to release the I²C Bus.

I²C Bus Communication

Communication on the I²C bus requires four separate steps, a START signal, a slave device address transmission, a data transmission and finally a STOP signal. When a START signal is placed on the I²C bus, all devices on the bus will receive this signal and be notified of the imminent arrival of data on the bus. The first seven bits of the data will be the slave address with the first bit being the MSB. If the address of the slave device matches that of the transmitted address, the HAAS bit in the SIMC1 register will be set and an USIM interrupt will be generated. After entering the interrupt service routine, the slave device must first check the condition of the HAAS and SIMTOF bits to determine whether the interrupt source originates from an address match or from the completion of an 8-bit data transfer completion or from the I²C bus time-out occurrence. During a data transfer, note that after the 7-bit slave address has been transmitted, the following bit, which is the 8th bit, is the read/write bit whose value will be placed in the SRW bit. This bit will be checked by the slave device to determine whether to go into transmit or receive mode. Before any transfer of data to or from the I²C bus, the microcontroller must initialise the bus, the following are steps to achieve this:

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



I²C Bus Initialisation Flow Chart

I²C Bus Start Signal

The START signal can only be generated by the master device connected to the I²C bus and not by the slave device. This START signal will be detected by all devices connected to the I²C bus. When detected, this indicates that the I²C bus is busy and therefore the HBB bit will be set. A START condition occurs when a high to low transition on the SDA line takes place when the SCL line remains high.

I²C Slave Address

The transmission of a START signal by the master will be detected by all devices on the I²C bus. To determine which slave device the master wishes to communicate with, the address of the slave device will be sent out immediately following the START signal. All slave devices, after receiving this 7-bit address data, will compare it with their own 7-bit slave address. If the address sent out by the master matches the internal address of the microcontroller slave device, then an internal USIM I²C bus interrupt signal will be generated. The next bit following the address, which is the 8th bit, defines the read/write status and will be saved to the SRW bit of the SIMC1 register. The slave device will then transmit an acknowledge bit, which is a low level, as the 9th bit. The slave device will also set the status flag HAAS when the addresses match.

As an USIM I²C bus interrupt can come from three sources, when the program enters the interrupt subroutine, the HAAS and SIMTOF bits should be examined to see whether the interrupt source has come from a matching slave address or from the completion of a data byte transfer or from the I²C bus time-out occurrence. When a slave address is matched, the device must be placed in either the transmit mode and then write data to the SIMD register, or in the receive mode where it must implement a dummy read from the SIMD register to release the SCL line.

I²C Bus Read/Write Signal

The SRW bit in the SIMC1 register defines whether the master device wishes to read data from the I²C bus or write data to the I²C bus. The slave device should examine this bit to determine if it is to be a transmitter or a receiver. If the SRW flag is "1" then this indicates that the master device wishes to read data from the I²C bus, therefore the slave device must be setup to send data to the I²C bus as a transmitter. If the SRW flag is "0" then this indicates that the master wishes to send data to the I²C bus, therefore the slave device must be setup to read data from the I²C bus as a receiver.

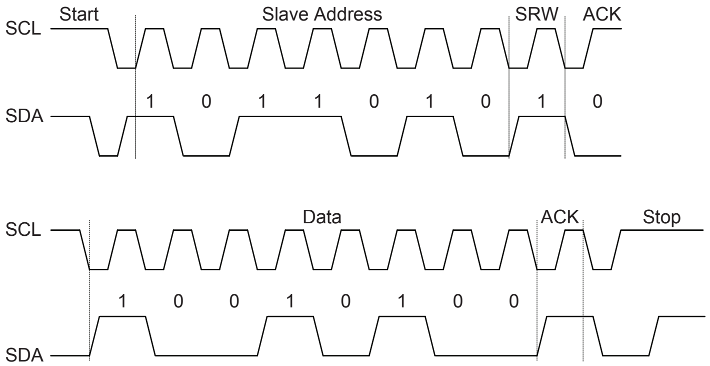
I^{2}C Bus Slave Address Acknowledge Signal

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

I^{2}C Bus Data and Acknowledge Signal

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

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

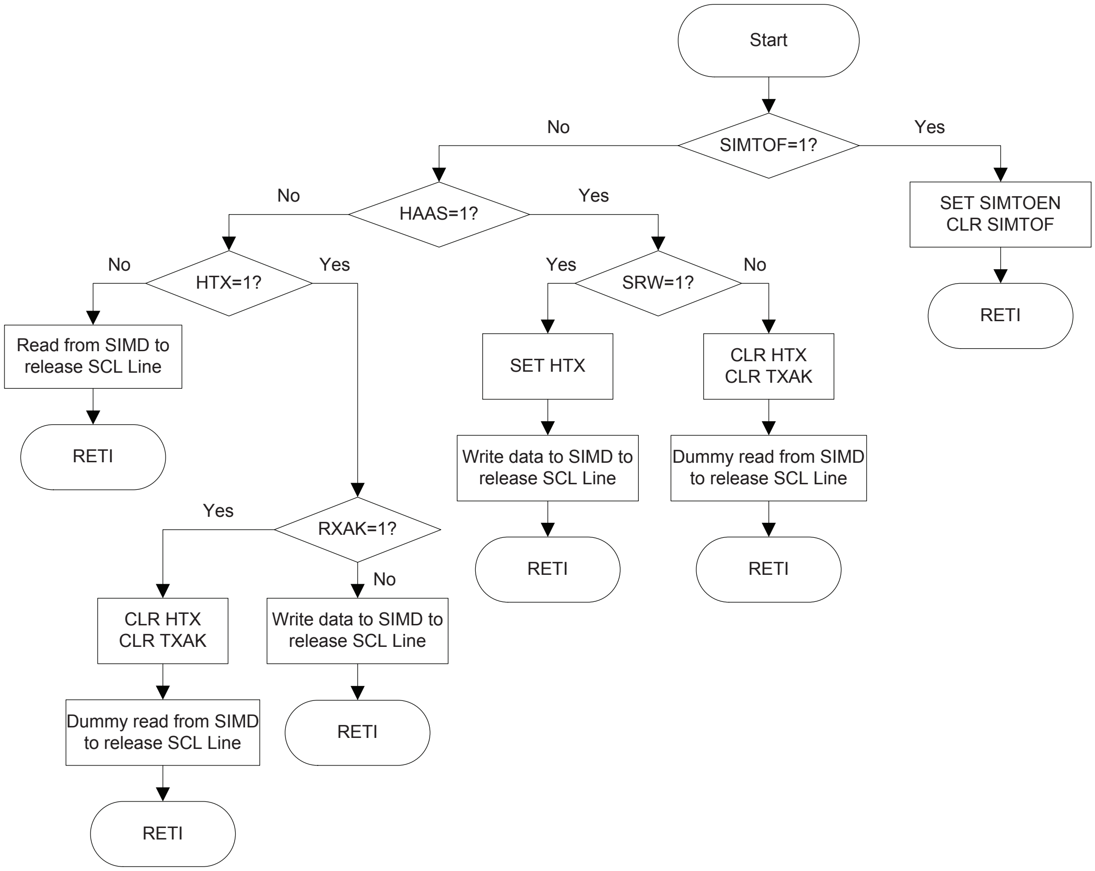


S=Start (1 bit)
SA=Slave Address (7 bits)
SR=SRW bit (1 bit)
M=Slave device send acknowledge bit (1 bit)
D=Data (8 bits)
A=ACK (RXAK bit for transmitter, TXAK bit for receiver, 1 bit)
P=Stop (1 bit)

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

I^{2}C Communication Timing Diagram

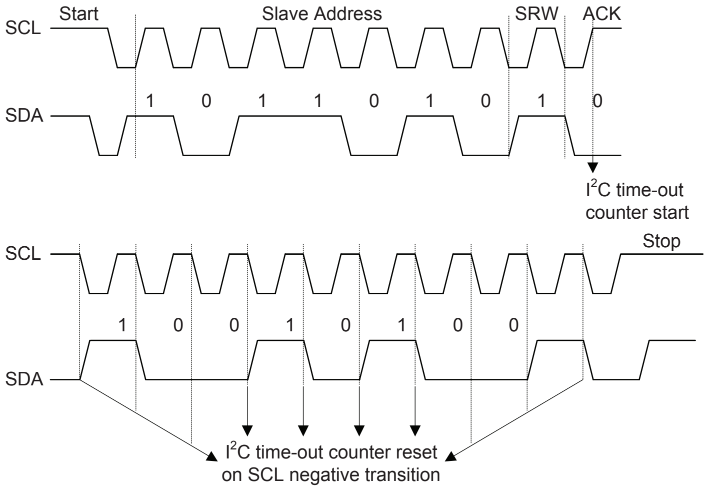
Note: When a slave address is matched, the device must be placed in either the transmit mode and then write data to the SIMD register, or in the receive mode where it must implement a dummy read from the SIMD register to release the SCL line.


Start
SIMTOF=1?
No
Yes
SET SIMTOEN
CLR SIMTOF
RETI
HAAS=1?
No
Yes
HTX=1?
No
Yes
Read from SIMD to release SCL Line
RETI
SRW=1?
Yes
No
SET HTX
Write data to SIMD to release SCL Line
RETI
CLR HTX
CLR TXAK
Dummy read from SIMD to release SCL Line
RETI
RXAK=1?
Yes
No
CLR HTX
CLR TXAK
Dummy read from SIMD to release SCL Line
RETI
Write data to SIMD to release SCL Line
RETI

I^{2}C Bus ISR Flow Chart

I^{2}C Time-out Control

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


SCL
Start
Slave Address
SRW
ACK
SDA
1
0
1
1
0
1
0
1
0
I²C time-out counter start
SCL
Stop
1
0
0
1
0
1
0
0
SDA
I²C time-out counter reset on SCL negative transition

I²C Time-out

When an I²C time-out counter overflow occurs, the counter will stop and the SIMTOEN bit will be cleared to zero and the SIMTOF bit will be set high to indicate that a time-out condition has occurred. The time-out condition will also generate an interrupt which uses the USIM interrupt vector. When an I²C time-out occurs, the I²C internal circuitry will be reset and the registers will be reset into the following condition:

Registers	After I²C Time-out
SIMD, SIMA, SIMC0	No change
SIMC1	Reset to POR condition

I²C Registers after Time-out

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

- **SIMTOC Register**

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

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

Bit 6 **SIMTOF**: USIM I²C Time-out flag
0: No time-out occurred
1: Time-out occurred
This bit is set high when time-out occurs and can only be cleared to 0 by application program.

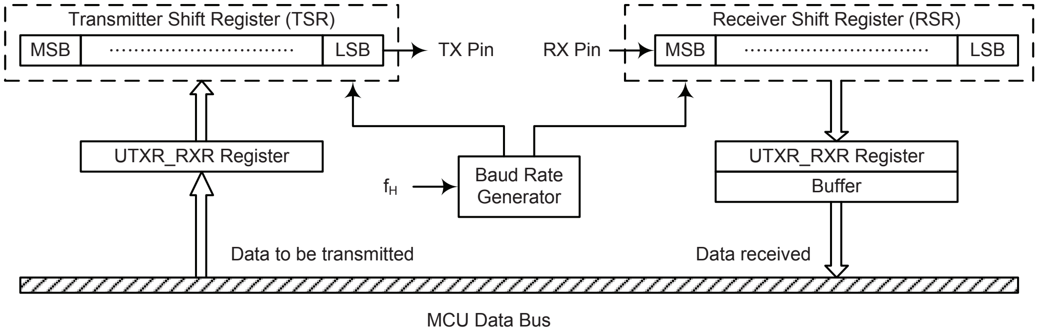
Bit 5~0 **SIMTOS5~SIMTOS0**: USIM I²C Time-out period selection
I²C time-out clock source is $f_{SUB}/32$.
I²C time-out time is equal to $(\text{SIMTOS}[5:0]+1) \times (32/f_{SUB})$.

UART Interface

The devices contain an integrated full-duplex asynchronous serial communications UART interface that enables communication with external devices that contain a serial interface. The UART function has many features and can transmit and receive data serially by transferring a frame of data with eight or nine data bits per transmission as well as being able to detect errors when the data is overwritten or incorrectly framed. The UART function shares the same internal interrupt vector with the SPI and I^2C interfaces which can be used to indicate when a reception occurs or when a transmission terminates.

The integrated UART function contains the following features:

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


Transmitter Shift Register (TSR)
MSB
LSB
TX Pin
RX Pin
Receiver Shift Register (RSR)
MSB
LSB
UTXR_RXR Register
fH
Baud Rate Generator
UTXR_RXR Register
Buffer
Data to be transmitted
Data received
MCU Data Bus

UART Data Transfer Block Diagram

UART External Pins

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

UART Data Transfer Scheme

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

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

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

UART Status and Control Registers

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

Register Name	Bit							
	7	6	5	4	3	2	1	0
SIMC0	SIM2	SIM1	SIM0	UMD	SIMDEB1	SIMDEB0	SIMEN	SIMICF
UUSR	UPERR	UNF	UFERR	UOERR	URIDLE	URXIF	UTIDLE	UTXIF
UUCR1	UREN	UBNO	UPREN	UPRT	USTOPS	UTXBRK	URX8	UTX8
UUCR2	UTXEN	URXEN	UBRGH	UADDEN	UWAKE	URIE	UTIIE	UTEIE
UTXR_RXR	UTXRX7	UTXRX6	UTXRX5	UTXRX4	UTXRX3	UTXRX2	UTXRX1	UTXRX0
UBRG	UBRG7	UBRG6	UBRG5	UBRG4	UBRG3	UBRG2	UBRG1	UBRG0

UART Register List

- **SIMC0 Register**

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

Bit 7~5 **SIM2~SIM0**: USIM SPI/I^{2}C Operating Mode Control

When the UMD bit is cleared to zero, these bits setup the SPI or I^{2}C operating mode of the USIM function. Refer to the SPI or I^{2}C register section for more details.

Bit 4 **UMD**: UART mode selection bit

0: SPI or I^{2}C mode
1: UART mode

This bit is used to select the UART mode. When this bit is cleared to zero, the actual SPI or I^{2}C mode can be selected using the SIM2~SIM0 bits. Note that the UMD bit must be set low for SPI or I^{2}C mode.

Bit 3~2 **SIMDEB1~SIMDEB0**: I^{2}C Debounce Time Selection

Refer to the I^{2}C register section.

Bit 1 **SIMEN**: USIM SPI/I^{2}C Enable Control

This bit is only available when the USIM is configured to operate in an SPI or I^{2}C mode with the UMD bit set low. Refer to the SPI or I^{2}C register section for more details.

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

Refer to the SPI register section.

- **UUSR Register**

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

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

Bit 7 **UPERR**: Parity error flag

0: No parity error is detected
1: Parity error is detected

The UPERR flag is the parity error flag. When this read only flag is "0", it indicates a parity error has not been detected. When the flag is "1", it indicates that the parity of the received word is incorrect. This error flag is applicable only if Parity mode (odd or even) is selected. The flag can also be cleared to 0 by a software sequence which involves a read to the status register UUSR followed by an access to the UTXR_RXR data register.

Bit 6 **UNF**: Noise flag

0: No noise is detected
1: Noise is detected

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

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

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

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

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

Bit 3 **URIDLE**: Receiver status
0: Data reception is in progress (Data being received)
1: No data reception is in progress (Receiver is idle)

The URIDLE flag is the receiver status flag. When this read only flag is “0”, it indicates that the receiver is between the initial detection of the start bit and the completion of the stop bit. When the flag is “1”, it indicates that the receiver is idle. Between the completion of the stop bit and the detection of the next start bit, the URIDLE bit is “1” indicating that the UART receiver is idle and the RX pin stays in logic high condition.

Bit 2 **URXIF**: Receive UTXR_RXR data register status
0: UTXR_RXR data register is empty
1: UTXR_RXR data register has available data

The URXIF flag is the receive data register status flag. When this read only flag is “0”, it indicates that the UTXR_RXR read data register is empty. When the flag is “1”, it indicates that the UTXR_RXR read data register contains new data. When the contents of the shift register are transferred to the UTXR_RXR register, an interrupt is generated if URIE=1 in the UUCR2 register. If one or more errors are detected in the received word, the appropriate receive-related flags UNF, UFERR, and/or UPERR are set within the same clock cycle. The URXIF flag will eventually be cleared to 0 when the UUSR register is read with URXIF set, followed by a read from the UTXR_RXR register, and if the UTXR_RXR register has no more new data available.

Bit 1 **UTIDLE**: Transmission idle
0: Data transmission is in progress (Data being transmitted)
1: No data transmission is in progress (Transmitter is idle)

The UTIDLE flag is known as the transmission complete flag. When this read only flag is “0”, it indicates that a transmission is in progress. This flag will be set high when the UTXIF flag is “1” and when there is no transmit data or break character being transmitted. When UTIDLE is equal to “1”, the TX pin becomes idle with the pin state in logic high condition. The UTIDLE flag is cleared to 0 by reading the UUSR register with UTIDLE set and then writing to the UTXR_RXR register. The flag is not generated when a data character or a break is queued and ready to be sent.

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

The UTXIF flag is the transmit data register empty flag. When this read only flag is “0”, it indicates that the character is not transferred to the transmitter shift register. When the flag is “1”, it indicates that the transmitter shift register has received a character from the UTXR_RXR data register. The UTXIF flag is cleared to 0 by reading the UART status register (UUSR) with UTXIF set and then writing to the UTXR_RXR data register. Note that when the UTXEN bit is set, the UTXIF flag bit will also be set since the transmit data register is not yet full.

- **UUCR1 Register**

The UUCR1 register together with the UUCR2 register are the two UART control registers that are used to set the various options for the UART function, such as overall on/off control, parity control, data transfer bit length etc. Further explanation on each of the bits is given below:

Bit	7	6	5	4	3	2	1	0
Name	UREN	UBNO	UPREN	UPRT	USTOPS	UTXBRK	URX8	UTX8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R	W
POR	0	0	0	0	0	0	x	0

"x": Unknown

Bit 7 **UREN**: UART function enable control
0: Disable UART. TX and RX pins are in a floating state
1: Enable UART. TX and RX pins function as UART pins

The UREN bit is the UART enable bit. When this bit is equal to "0", the UART will be disabled and the RX pin as well as the TX pin will be set in a floating state. When the bit is equal to "1", the UART will be enabled if the UMD bit is set and the TX and RX pins will function as defined by the UTXEN and URXEN enable control bits.

When the UART is disabled, it will empty the buffer so any character remaining in the buffer will be discarded. In addition, the value of the baud rate counter will be reset. If the UART is disabled, all error and status flags will be reset. Also the UTXEN, URXEN, UTXBRK, URXIF, UOERR, UFERR, UPERR and UNF bits will be cleared to 0, while the UTIDLE, UTXIF and URIDLE bits will be set. Other control bits in UUCR1, UUCR2 and UBRG registers will remain unaffected. If the UART is active and the UREN bit is cleared to 0, all pending transmissions and receptions will be terminated and the module will be reset as defined above. When the UART is re-enabled, it will restart in the same configuration.

Bit 6 **UBNO**: Number of data transfer bits selection
0: 8-bit data transfer
1: 9-bit data transfer

This bit is used to select the data length format, which can have a choice of either 8-bit or 9-bit format. When this bit is equal to "1", a 9-bit data length format will be selected. If the bit is equal to "0", then an 8-bit data length format will be selected. If 9-bit data length format is selected, then bits URX8 and UTX8 will be used to store the 9th bit of the received and transmitted data respectively.

Bit 5 **UPREN**: Parity function enable control
0: Parity function is disabled
1: Parity function is enabled

This is the parity enable bit. When this bit is equal to "1", the parity function will be enabled. If the bit is equal to "0", then the parity function will be disabled.

Bit 4 **UPRT**: Parity type selection bit
0: Even parity for parity generator
1: Odd parity for parity generator

This bit is the parity type selection bit. When this bit is equal to "1", odd parity type will be selected. If the bit is equal to "0", then even parity type will be selected.

Bit 3 **USTOPS**: Number of Stop bits selection
0: One stop bit format is used
1: Two stop bits format is used

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

Bit 2 **UTXBRK**: Transmit break character
0: No break character is transmitted
1: Break characters transmit

The UTXBRK bit is the Transmit Break Character bit. When this bit is "0", there are no break characters and the TX pin operates normally. When the bit is "1", there are transmit break characters and the transmitter will send logic zeros. When this bit is equal to "1", after the buffered data has been transmitted, the transmitter output is held low for a minimum of a 13-bit length and until the UTXBRK bit is reset.

Bit 1 **URX8**: Receive data bit 8 for 9-bit data transfer format (read only)

This bit is only used if 9-bit data transfers are used, in which case this bit location will store the 9th bit of the received data known as URX8. The UBNO bit is used to determine whether data transfers are in 8-bit or 9-bit format.

Bit 0 **UTX8**: Transmit data bit 8 for 9-bit data transfer format (write only)

This bit is only used if 9-bit data transfers are used, in which case this bit location will store the 9th bit of the transmitted data known as UTX8. The UBNO bit is used to determine whether data transfers are in 8-bit or 9-bit format.

- **UUCR2 Register**

The UUCR2 register is the second of the two UART control registers and serves several purposes. One of its main functions is to control the basic enable/disable operation of the UART Transmitter and Receiver as well as enabling the various USIM UART mode interrupt sources. The register also serves to control the baud rate speed, receiver wake-up enable and the address detect enable. Further explanation on each of the bits is given below:

Bit	7	6	5	4	3	2	1	0
Name	UTXEN	URXEN	UBRGH	UADDEN	UWAKE	URIE	UTIIE	UTEIE
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7 **UTXEN**: UART Transmitter enabled control

0: UART transmitter is disabled
1: UART transmitter is enabled

The bit named UTXEN is the Transmitter Enable Bit. When this bit is equal to “0”, the transmitter will be disabled with any pending data transmissions being aborted. In addition the buffers will be reset. In this situation the TX pin will be set in a floating state.

If the UTXEN bit is equal to “1” and the UMD and UREN bit are also equal to “1”, the transmitter will be enabled and the TX pin will be controlled by the UART. Clearing the UTXEN bit to 0 during a transmission will cause the data transmission to be aborted and will reset the transmitter. If this situation occurs, the TX pin will be set in a floating state.

Bit 6 **URXEN**: UART Receiver enabled control

0: UART receiver is disabled
1: UART receiver is enabled

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

Bit 5 **UBRGH**: Baud Rate speed selection

0: Low speed baud rate
1: High speed baud rate

The bit named UBRGH selects the high or low speed mode of the Baud Rate Generator. This bit, together with the value placed in the baud rate register UBRG, controls the Baud Rate of the UART. If this bit is equal to “1”, the high speed mode is selected. If the bit is equal to “0”, the low speed mode is selected.

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

The bit named UADDEN is the address detect function enable control bit. When this bit is equal to “1”, the address detect function is enabled. When it occurs, if the 8th bit, which corresponds to URX7 if UBNO=0 or the 9th bit, which corresponds to URX8 if UBNO=1, has a value of “1”, then the received word will be identified as an address, rather than data. If the corresponding interrupt is enabled, an interrupt request will be generated each time the received word has the address bit set, which is the 8th or 9th bit depending on the value of UBNO. If the address bit known as the 8th or 9th bit of the received word is “0” with the address detect function being enabled, an interrupt will not be generated and the received data will be discarded.

Bit 3 **UWAKE**: RX pin wake-up UART function enable control
0: RX pin wake-up UART function is disabled
1: RX pin wake-up UART function is enabled

This bit is used to control the wake-up UART function when a falling edge on the RX pin occurs. Note that this bit is only available when the UART clock (f_H) is switched off. There will be no RX pin wake-up UART function if the UART clock (f_H) exists. If the UWAKE bit is set to 1 as the UART clock (f_H) is switched off, a UART wake-up request will be initiated when a falling edge on the RX pin occurs. When this request happens and the corresponding interrupt is enabled, an RX pin wake-up UART interrupt will be generated to inform the MCU to wake up the UART function by switching on the UART clock (f_H) via the application program. Otherwise, the UART function cannot resume even if there is a falling edge on the RX pin when the UWAKE bit is cleared to 0.

Bit 2 **URIE**: Receiver interrupt enable control
0: Receiver related interrupt is disabled
1: Receiver related interrupt is enabled

This bit enables or disables the receiver interrupt. If this bit is equal to “1” and when the receiver overrun flag UOERR or receive data available flag URXIF is set, the USIM interrupt request flag USIMF will be set. If this bit is equal to “0”, the USIM interrupt request flag USIMF will not be influenced by the condition of the UOERR or URXIF flags.

Bit 1 **UTIIE**: Transmitter Idle interrupt enable control
0: Transmitter idle interrupt is disabled
1: Transmitter idle interrupt is enabled

This bit enables or disables the transmitter idle interrupt. If this bit is equal to “1” and when the transmitter idle flag UTIDLE is set, due to a transmitter idle condition, the USIM interrupt request flag USIMF will be set. If this bit is equal to “0”, the USIM interrupt request flag USIMF will not be influenced by the condition of the UTIDLE flag.

Bit 0 **UTEIE**: Transmitter Empty interrupt enable control
0: Transmitter empty interrupt is disabled
1: Transmitter empty interrupt is enabled

This bit enables or disables the transmitter empty interrupt. If this bit is equal to “1” and when the transmitter empty flag UTXIF is set, due to a transmitter empty condition, the USIM interrupt request flag USIMF will be set. If this bit is equal to “0”, the USIM interrupt request flag USIMF will not be influenced by the condition of the UTXIF flag.

- **UTXR_RXR Register**

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

Bit	7	6	5	4	3	2	1	0
Name	UTXRX7	UTXRX6	UTXRX5	UTXRX4	UTXRX3	UTXRX2	UTXRX1	UTXRX0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	x	x	x	x	x	x	x	x

"x": Unknown

Bit 7~0 **UTXRX7~UTXRX0**: UART Transmit/Receive Data bit 7 ~ bit 0

- **UBRG Register**

Bit	7	6	5	4	3	2	1	0
Name	UBRG7	UBRG6	UBRG5	UBRG4	UBRG3	UBRG2	UBRG1	UBRG0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	x	x	x	x	x	x	x	x

"x": Unknown

Bit 7~0 **UBRG7~UBRG0**: Baud Rate values

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

Note: Baud rate=f_H / [64 × (N+1)] if UBRGH=0.
Baud rate=f_H / [16 × (N+1)] if UBRGH=1.

Baud Rate Generator

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

UUCR2 UBRGH Bit	0	1
Baud Rate (BR)	f_H / [64 (N+1)]	f_H / [16 (N+1)]

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

Calculating the Baud Rate and Error Values

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

From the above table the desired baud rate BR=f_H / [64 (N+1)]

Re-arranging this equation gives N=[f_H / (BR×64)] - 1

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

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

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

UART Setup and Control

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

Enabling/Disabling the UART Interface

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

Clearing the UREN bit will disable the TX and RX pins and allow these two pins to be used as normal I/O or other pin-shared functional pins by configuring the corresponding pin-shared control bits. When the UART function is disabled the buffer will be reset to an empty condition, at the same time discarding any remaining residual data. Disabling the UART will also reset the error and status flags with bits UTXEN, URXEN, UTXBRK, URXIF, UOERR, UFERR, UPERR and UNF being cleared while bits UTIDLE, UTXIF and URIDLE will be set. The remaining control bits in the UUCR1, UUCR2 and UBRG registers will remain unaffected. If the UREN bit in the UUCR1 register is cleared while the UART is active, then all pending transmissions and receptions will be immediately suspended and the UART will be reset to a condition as defined above. If the UART is then subsequently re-enabled, it will restart again in the same configuration.

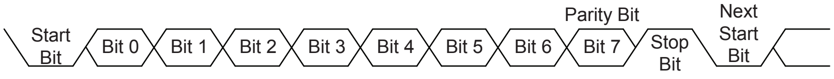
Data, Parity and Stop Bit Selection

The format of the data to be transferred is composed of various factors such as data bit length, parity on/off, parity type, address bits and the number of stop bits. These factors are determined by the setup of various bits within the UUCR1 register. The UBNO bit controls the number of data bits which can be set to either 8 or 9, the UPRT bit controls the choice of odd or even parity, the UPREN bit controls the parity on/off function and the USTOPS bit decides whether one or two stop bits are to be used. The following table shows various formats for data transmission. The address bit, which is the MSB of the data byte, identifies the frame as an address character or data if the address detect function is enabled. The number of stop bits, which can be either one or two, is independent of the data length and is only used for the transmitter. There is only one stop bit for the receiver.

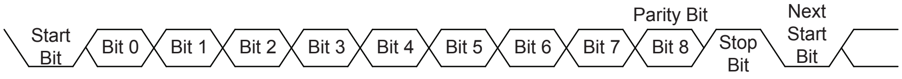
Start Bit	Data Bits	Address Bit	Parity Bit	Stop Bit
Example of 8-bit Data Formats				
1	8	0	0	1
1	7	0	1	1
1	7	1	0	1
Example of 9-bit Data Formats				
1	9	0	0	1
1	8	0	1	1
1	8	1	0	1

Transmitter Receiver Data Format

The following diagram shows the transmit and receive waveforms for both 8-bit and 9-bit data formats.



8-bit Data Format



9-bit Data Format

UART Transmitter

Data word lengths of either 8 or 9 bits can be selected by programming the UBNO bit in the UUCR1 register. When UBNO bit is set, the word length will be set to 9 bits. In this case the 9th bit, which is the MSB, needs to be stored in the UTX8 bit in the UUCR1 register. At the transmitter core lies the Transmitter Shift Register, more commonly known as the TSR, whose data is obtained from the transmit data register, which is known as the UTXR_RXR register. The data to be transmitted is loaded into this UTXR_RXR register by the application program. The TSR register is not written to with new data until the stop bit from the previous transmission has been sent out. As soon as this stop bit has been transmitted, the TSR can then be loaded with new data from the UTXR_RXR register, if it is available. It should be noted that the TSR register, unlike many other registers, is not directly mapped into the Data Memory area and as such is not available to the application program for direct read/write operations. An actual transmission of data will normally be enabled when the UTXEN bit is set, but the data will not be transmitted until the UTXR_RXR register has been loaded with data and the baud rate generator has defined a shift clock source. However, the transmission can also be initiated by first loading data into the UTXR_RXR register, after which the UTXEN bit can be set. When a transmission of data begins, the TSR is normally empty, in which case a transfer to the UTXR_RXR register will result in an immediate transfer to the TSR. If during a transmission the UTXEN bit is cleared, the transmission will immediately cease and the transmitter will be reset. The TX output pin can then be configured as the I/O or other pin-shared function by configuring the corresponding pin-shared control bits.

Transmitting Data

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

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

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

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

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

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

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

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

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

Transmit Break

If the UTXBRK bit is set then break characters will be sent on the next transmission. Break character transmission consists of a start bit, followed by 13×N ‘0’ bits and stop bits, where N=1, 2, etc. If a break character is to be transmitted then the UTXBRK bit must be first set by the application program, and then cleared to generate the stop bits. Transmitting a break character will not generate a transmit interrupt. Note that a break condition length is at least 13 bits long. If the UTXBRK bit is continually kept at a logic high level then the transmitter circuitry will transmit continuous break characters. After the application program has cleared the UTXBRK bit, the transmitter will finish transmitting the last break character and subsequently send out one or two stop bits. The automatic logic highs at the end of the last break character will ensure that the start bit of the next frame is recognized.

UART Receiver

The UART is capable of receiving word lengths of either 8 or 9 bits. If the UBNO bit is set, the word length will be set to 9 bits with the MSB being stored in the URX8 bit of the UUCR1 register. At the receiver core lies the Receive Serial Shift Register, commonly known as the RSR. The data which is received on the RX external input pin is sent to the data recovery block. The data recovery block operating speed is 16 times that of the baud rate, while the main receive serial shifter operates at the baud rate. After the RX pin is sampled for the stop bit, the received data in RSR is transferred to the receive data register, if the register is empty. The data which is received on the external RX input pin is sampled three times by a majority detect circuit to determine the logic level that has been placed onto the RX pin. It should be noted that the RSR register, unlike many other registers, is not directly mapped into the Data Memory area and as such is not available to the application program for direct read/write operations.

Receiving Data

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

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

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

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

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

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

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

Receive Break

Any break character received by the UART will be managed as a framing error. The receiver will count and expect a certain number of bit times as specified by the values programmed into the UBNO bit plus one stop bit. If the break is much longer than 13 bit times, the reception will be considered as complete after the number of bit times specified by UBNO plus one stop bit. The URXIF bit is set, UFERR is set, zeros are loaded into the receive data register, interrupts are generated if appropriate and the URIDLE bit is set. A break is regarded as a character that contains only zeros with the UFERR flag set. If a long break signal has been detected, the receiver will regard it as a data frame including a start bit, data bits and the invalid stop bit and the UFERR flag will be set. The receiver must wait for a valid stop bit before looking for the next start bit. The receiver will not make the assumption that the break condition on the line is the next start bit. The break character will be loaded into the buffer and no further data will be received until stop bits are received. It should be noted that the URIDLE read only flag will go high when the stop bits have not yet been received. The reception of a break character on the UART registers will result in the following:

- The framing error flag, UFERR, will be set.
- The receive data register, UTXR_RXR, will be cleared.
- The UOERR, UNF, UPERR, URIDLE or URXIF flags will possibly be set.

Idle Status

When the receiver is reading data, which means it will be in between the detection of a start bit and the reading of a stop bit, the receiver status flag in the UUSR register, otherwise known as the URIDLE flag, will have a zero value. In between the reception of a stop bit and the detection of the next start bit, the URIDLE flag will have a high value, which indicates the receiver is in an idle condition.

Receiver Interrupt

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

Managing Receiver Errors

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

Overrun Error – UOERR

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

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

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

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

Noise Error – UNF

Over-sampling is used for data recovery to identify valid incoming data and noise. If noise is detected within a frame the following will occur:

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

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

Framing Error – UFERR

The read only framing error flag, UFERR, in the UUSR register, is set if a zero is detected instead of stop bits. If two stop bits are selected, both stop bits must be high; otherwise the UFERR flag will be set. The UFERR flag and the received data will be recorded in the UUSR and UTXR_RXR registers respectively, and the flag is cleared in any reset.

Parity Error – UPERR

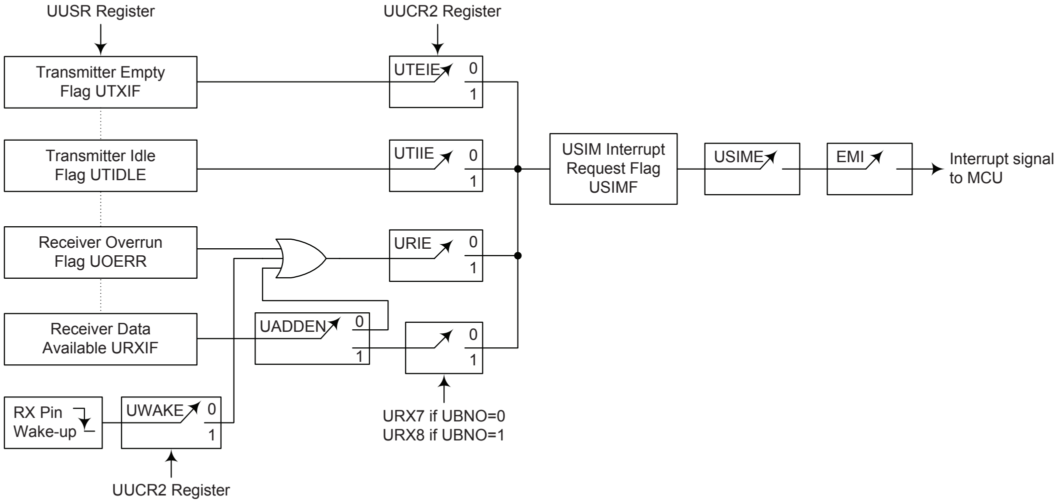
The read only parity error flag, UPERR, in the UUSR register, is set if the parity of the received word is incorrect. This error flag is only applicable if the parity is enabled, UPREN=1, and if the parity type, odd or even is selected. The read only UPERR flag and the received data will be recorded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

UART Interrupt Structure

Several individual UART conditions can trigger an USIM interrupt. When these conditions exist, a low pulse will be generated to get the attention of the microcontroller. These conditions are a transmitter data register empty, transmitter idle, receiver data available, receiver overrun, address detect and an RX pin wake-up. When any of these conditions are created, if the global interrupt enable bit and the USIM interrupt control bit are enabled and the stack is not full, the program will jump to its corresponding interrupt vector where it can be serviced before returning to the main program. Four of these conditions have the corresponding UUSR register flags which will generate an USIM interrupt if its associated interrupt enable control bit in the UUCR2 register is set. The two transmitter interrupt conditions have their own corresponding enable control bits, while the two receiver interrupt conditions have a shared enable control bit. These enable bits can be used to mask out individual USIM UART mode interrupt sources.

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

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


UUSR Register
UUCR2 Register
Transmitter Empty Flag UTXIF
UTEIE
0
1
Transmitter Idle Flag UTIDLE
UTIIE
0
1
USIM Interrupt Request Flag USIMF
USIME
EMI
Interrupt signal to MCU
Receiver Overrun Flag UOERR
URIE
0
1
Receiver Data Available URXIF
UADDEN
0
1
0
1
URX7 if UBNO=0
URX8 if UBNO=1
RX Pin Wake-up
UWAKE
0
1
UUCR2 Register

UART Interrupt Structure

Address Detect Mode

Setting the Address Detect Mode bit, UADDEN, in the UUCR2 register, enables this special mode. If this bit is enabled then an additional qualifier will be placed on the generation of a Receiver Data Available interrupt, which is requested by the URXIF flag. If the UADDEN bit is enabled, then when data is available, an interrupt will only be generated, if the highest received bit has a high value. Note that the USIME and EMI interrupt enable bits must also be enabled for correct interrupt generation. This highest address bit is the 9th bit if UBNO=1 or the 8th bit if UBNO=0. If this bit is high, then the received word will be defined as an address rather than data. A Data Available interrupt will be generated every time the last bit of the received word is set. If the UADDEN bit is not enabled, then a Receiver Data Available interrupt will be generated each time the URXIF flag is set, irrespective of the data last bit status. The address detect mode and parity enable are mutually exclusive functions. Therefore if the address detect mode is enabled, then to ensure correct operation, the parity function should be disabled by resetting the parity enable bit UPREN to zero.

UADDEN	Bit 9 if UBNO=1, Bit 8 if UBNO=0	USIM Interrupt Generated
0	0	√
	1	√
1	0	×
	1	√

UADDEN Bit Function

UART Power Down and Wake-up

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

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

For a UART wake-up interrupt to occur, in addition to the bits for the wake-up being set, the global interrupt enable bit, EMI, and the USIM interrupt enable bit, USIME, must be set. If the EMI and USIME bits are not set then only a wake up event will occur and no interrupt will be generated. Note also that as it takes certain system clock cycles after a wake-up before normal microcontroller resumes, the USIM interrupt will not be generated until after this time has elapsed.

Touch Key Function

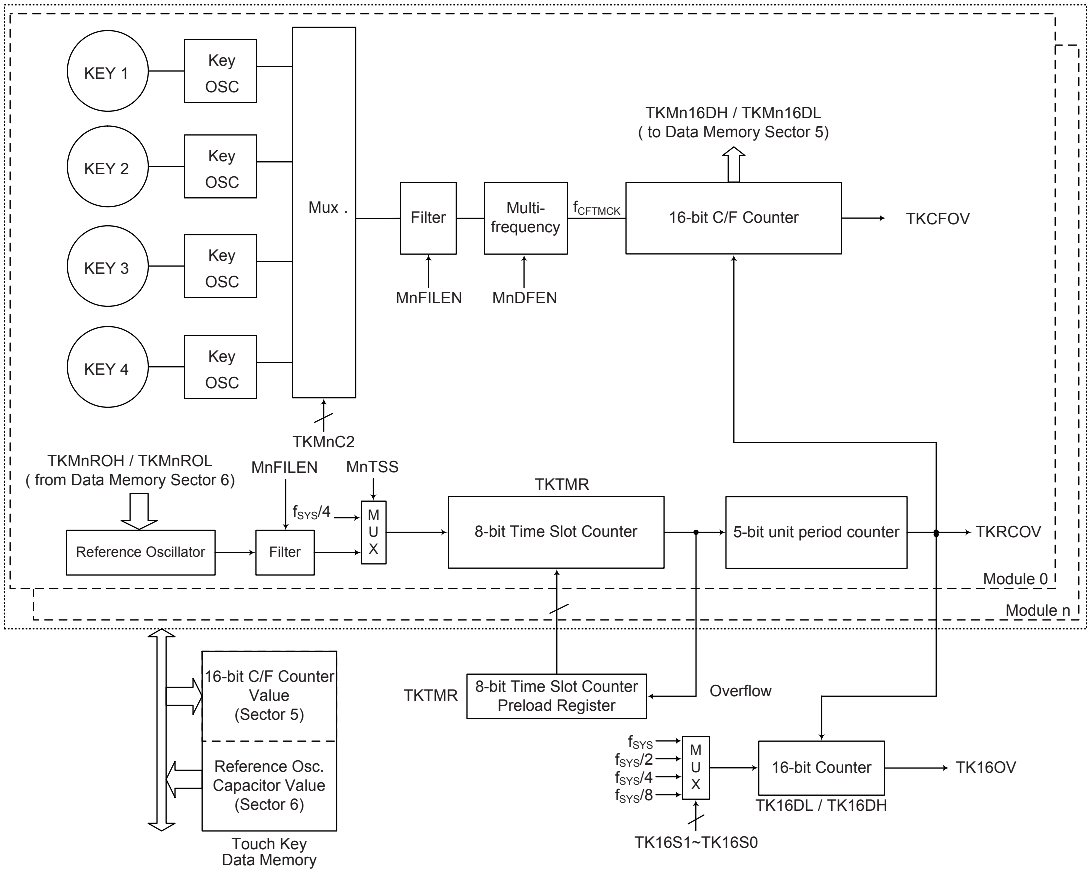
Each device provides multiple touch key functions. The touch key function is fully integrated and requires no external components, allowing touch key functions to be implemented by the simple manipulation of internal registers.

Touch Key Structure

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

Device	Total Key Number	Touch Key Module		Touch Key
BS83B24C	24	Mn (n=0~5)	M0	KEY1~KEY4
			M1	KEY5~KEY8
			M2	KEY9~KEY12
			M3	KEY13~KEY16
			M4	KEY17~KEY20
			M5	KEY21~KEY24
BS83C40C	40	Mn (n=0~9)	M0	KEY1~KEY4
			M1	KEY5~KEY8
			M2	KEY9~KEY12
			M3	KEY13~KEY16
			M4	KEY17~KEY20
			M5	KEY21~KEY24
			M6	KEY25~KEY28
			M7	KEY29~KEY32
			M8	KEY33~KEY36
			M9	KEY37~KEY40

Touch Key Structure


KEY 1
KEY 2
KEY 3
KEY 4
Key OSC
Key OSC
Key OSC
Key OSC
Mux .
TKMnC2
Filter
MnFILEN
Multi-frequency
MnDFEN
f_{CFTMCK}
16-bit C/F Counter
TKMn16DH / TKMn16DL
(to Data Memory Sector 5)
TKCFOV
TKMnROH / TKMnROL
(from Data Memory Sector 6)
Reference Oscillator
MnFILEN
Filter
$f_{SYS}/4$
MnTSS
MUX
TKTMR
8-bit Time Slot Counter
5-bit unit period counter
TKRCOV
Module 0
Module n
16-bit C/F Counter Value (Sector 5)
Reference Osc. Capacitor Value (Sector 6)
Touch Key Data Memory
TKTMR
8-bit Time Slot Counter Preload Register
Overflow
f_{SYS}
$f_{SYS}/2$
$f_{SYS}/4$
$f_{SYS}/8$
MUX
TK16S1~TK16S0
16-bit Counter
TK16DL / TK16DH
TK16OV

Notes: 1. The structure contained in the dash line is identical for each touch key module which contains four touch keys.

2. When MnTSS=0 and MnROEN=1 or when MnTSS=1, the touch key function 16-bit counter can operate normally.

Touch Key Function Block Diagram

Touch Key Register Definition

Each touch key module, which contains four touch key functions, has its own suite registers. The following table shows the register set for each touch key module. The Mn within the register name refers to the Touch Key module number. The series of devices has up to ten Touch Key Modules depending upon the selected device.

Name	Description
TKTMR	Touch key time slot 8-bit counter preload register
TKC0	Touch key function Control register 0
TKC1	Touch key function Control register 1
TK16DL	Touch key function 16-bit counter low byte
TK16DH	Touch key function 16-bit counter high byte
TKMn16DL	Touch key module n 16-bit C/F counter low byte
TKMn16DH	Touch key module n 16-bit C/F counter high byte
TKMnROL	Touch key module n reference oscillator capacitor selection low byte
TKMnROH	Touch key module n reference oscillator capacitor selection high byte
TKMnC0	Touch key module n Control register 0
TKMnC1	Touch key module n Control register 1
TKMnC2	Touch key module n Control register 2

Touch Key Module Register Definition

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

Touch Key Function Register List

- **TKTMR Register**

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

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

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

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

- **TKC0 Register**

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

Bit 7 **TKRAMC**: Touch key data memory access control
0: Accessed by MCU
1: Accessed by touch key module

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

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

This bit can be accessed by application program. Note that this bit can not be set by application program but must be cleared to 0 by application program.

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

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

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

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

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

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

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

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

Bit 2 Unimplemented, read as "0"

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

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

In the auto scan mode the data movement which is described above is implemented by hardware. The individual reference oscillator capacitor value and 16-bit C/F counter content for all scanned keys will be read from and written into a dedicated Touch Key Data Memory area. The scan operation will not be stopped until all arranged keys are scanned.

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

This bit indicates whether the touch key scan operation is executing or not. It is set to 1 when the TKST bit is set high to start the scan operation for all touch key scan modes. In the auto scan mode this bit is cleared to 0 automatically when the touch key scan operation is completed. In the manual scan mode this bit is cleared to 0 automatically when the touch key time slot counter overflows.

- **TKC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	D5	TSCS	TK16S1	TK16S0	TKFS1	TKFS0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	1	1

Bit 7~5 **D7~D5**: Data bits for test only

These bits are used for test purpose only and must be kept as "000" for normal operations.

Bit 4 **TSCS**: Touch key time slot counter selection
0: Each module use own time slot counter.
1: All touch key module use module 0 time slot counter.

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

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

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

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

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

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

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

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

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

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

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

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

- **TKMnC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	MnDFEN	MnFILEN	MnSOFC	MnSOF2	MnSOF1	MnSOF0
R/W	—	—	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	—	0	0	0	0	0	0

Bit 7~6 Uninplemented. Read as “0”

Bit 5 **MnDFEN**: Touch key module n multi-frequency control
0: Disable
1: Enable
This bit is used to control the touch key oscillator frequency doubling function. When this bit is set to 1, the key oscillator frequency will be doubled.

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

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

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

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

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

- **TKMnC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	MnTSS	—	MnROEN	MnKOEN	MnK4EN	MnK3EN	MnK2EN	MnK1EN
R/W	R/W	—	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	—	0	0	0	0	0	0

Bit 7 **MnTSS**: Touch key module n time slot counter clock source select
0: Touch key module n reference oscillator
1: $f_{SYS}/4$

Bit 6 Unimplemented, read as "0"

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

This bit is used to enable the touch key module reference oscillator. In the auto scan mode, the reference oscillator will automatically be enabled by setting the MnROEN bit high when the TKST bit is set from low to high if the reference oscillator is selected as the time slot clock source. The combination of the MnTSS and MnK4EN~MnK1EN bits determines whether the reference oscillator is used or not. When the TKBUSY bit is changed from high to low, the MnROEN bit will automatically be set low to disable the reference oscillator.

In the manual scan mode the reference oscillator should first be enabled before setting the TKST bit from low to high if the reference oscillator is selected to be used and will be disabled when the TKBUSY bit is changed from high to low.

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

This bit is used to enable the touch key module key oscillator. In the auto scan mode, the key oscillator will automatically be enabled by setting the MnKOEN bit high when the TKST bit is set form low to high. When the TKBUSY bit is changed from high to low, the MnKOEN bit will automatically be set low to disable the key oscillator.

In the manual scan mode the key oscillator should first be enabled before setting the TKST bit from low to high if the relevant key is enabled to be scanned and will be disabled when the TKBUSY bit is changed from high to low.

Bit 3 **MnK4EN**: Touch key module n KEY4 enable control

MnK4EN	Touch Key Module n – Mn									
	M0	M1	M2	M3	M4	M5	M6	M7	M8	M9
0: Disable	I/O or other functions									
1: Enable	KEY4	KEY8	KEY12	KEY16	KEY20	KEY24	KEY28	KEY32	KEY36	KEY40
BS83B24C	√	√	√	√	√	√	—	—	—	—
BS83C40C	√	√	√	√	√	√	√	√	√	√

Bit 2 **MnK3EN**: Touch key module n KEY3 enable control

MnK3EN	Touch Key Module n – Mn									
	M0	M1	M2	M3	M4	M5	M6	M7	M8	M9
0: Disable	I/O or other functions									
1: Enable	KEY3	KEY7	KEY11	KEY15	KEY19	KEY23	KEY27	KEY31	KEY35	KEY39
BS83B24C	√	√	√	√	√	√	—	—	—	—
BS83C40C	√	√	√	√	√	√	√	√	√	√

Bit 1 **MnK2EN**: Touch key module n KEY2 enable control

MnK2EN	Touch Key Module n – Mn									
	M0	M1	M2	M3	M4	M5	M6	M7	M8	M9
0: Disable	I/O or other functions									
1: Enable	KEY2	KEY6	KEY10	KEY14	KEY18	KEY22	KEY26	KEY30	KEY34	KEY38
BS83B24C	√	√	√	√	√	√	—	—	—	—
BS83C40C	√	√	√	√	√	√	√	√	√	√

Bit 0 **MnK1EN**: Touch key module n KEY1 enable control

MnK1EN	Touch Key Module n – Mn									
	M0	M1	M2	M3	M4	M5	M6	M7	M8	M9
0: Disable	I/O or other functions									
1: Enable	KEY1	KEY5	KEY9	KEY13	KEY17	KEY21	KEY25	KEY29	KEY33	KEY37
BS83B24C	√	√	√	√	√	√	—	—	—	—
BS83C40C	√	√	√	√	√	√	√	√	√	√

- **TKMnC2 Register**

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

Bit 7~6 **MnSK31~MnSK30**: Touch key module n time slot 3 key scan selection
00: KEY1
01: KEY2
10: KEY3
11: KEY4

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

Bit 5~4 **MnSK21~MnSK20**: Touch key module n time slot 2 key scan selection
00: KEY1
01: KEY2
10: KEY3
11: KEY4

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

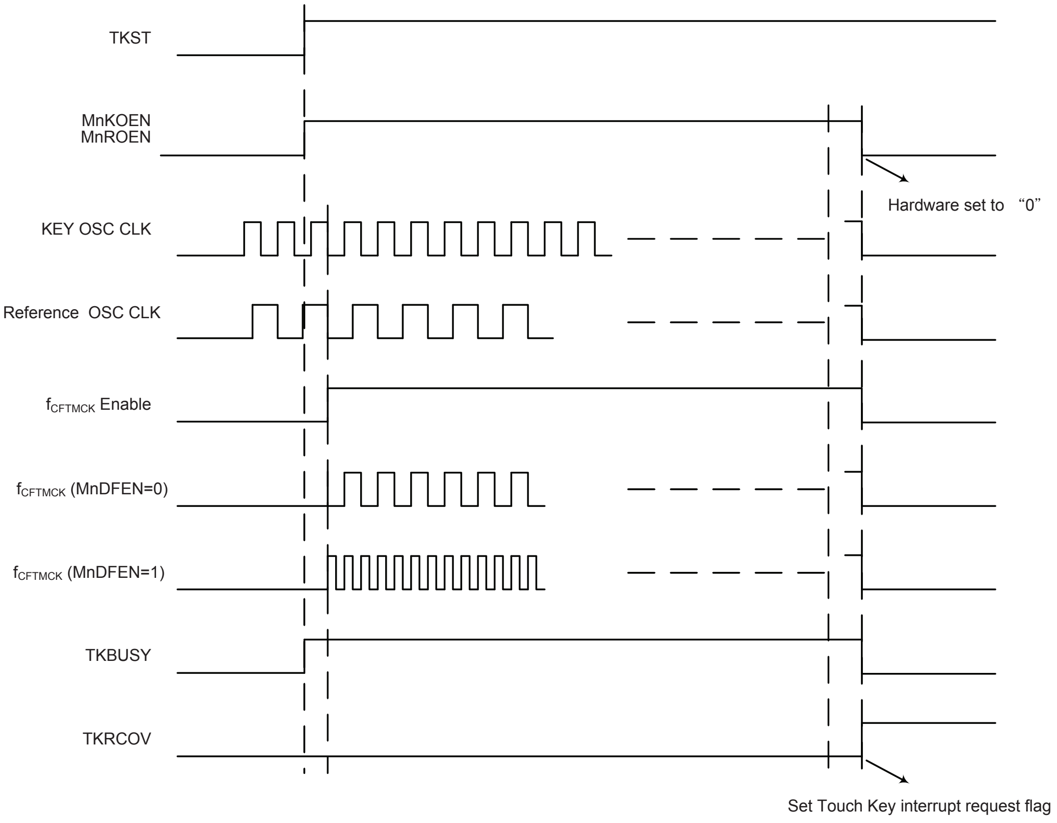
Bit 3~2 **MnSK11~MnSK10**: Touch key module n time slot 1 key scan selection
00: KEY1
01: KEY2
10: KEY3
11: KEY4
These bits are used to select the desired scan key in time slot 1 and only available in the auto scan mode.

Bit 1~0 **MnSK01~MnSK00**: Touch key module n time slot 0 key scan selection
00: KEY1
01: KEY2
10: KEY3
11: KEY4
These bits are used to select the desired scan key in time slot 0 in the auto scan mode or used as the multiplexer for scan key selection in the manual mode.

Touch Key Operation

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

Each touch key module contains four touch key inputs which are shared with logical I/O pins, and the desired function is selected using register bits. Each touch key has its own independent sense oscillator. Therefore, there are four sense oscillators within each touch key module.


TKST
MnKOEN
MnROEN
Hardware set to "0"
KEY OSC CLK
Reference OSC CLK
f_{CFTMCK} Enable
f_{CFTMCK} (MnDFEN=0)
f_{CFTMCK} (MnDFEN=1)
TKBUSY
TKRCOV
Set Touch Key interrupt request flag

Touch Key Manual Scan Mode Timing Diagram

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

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

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

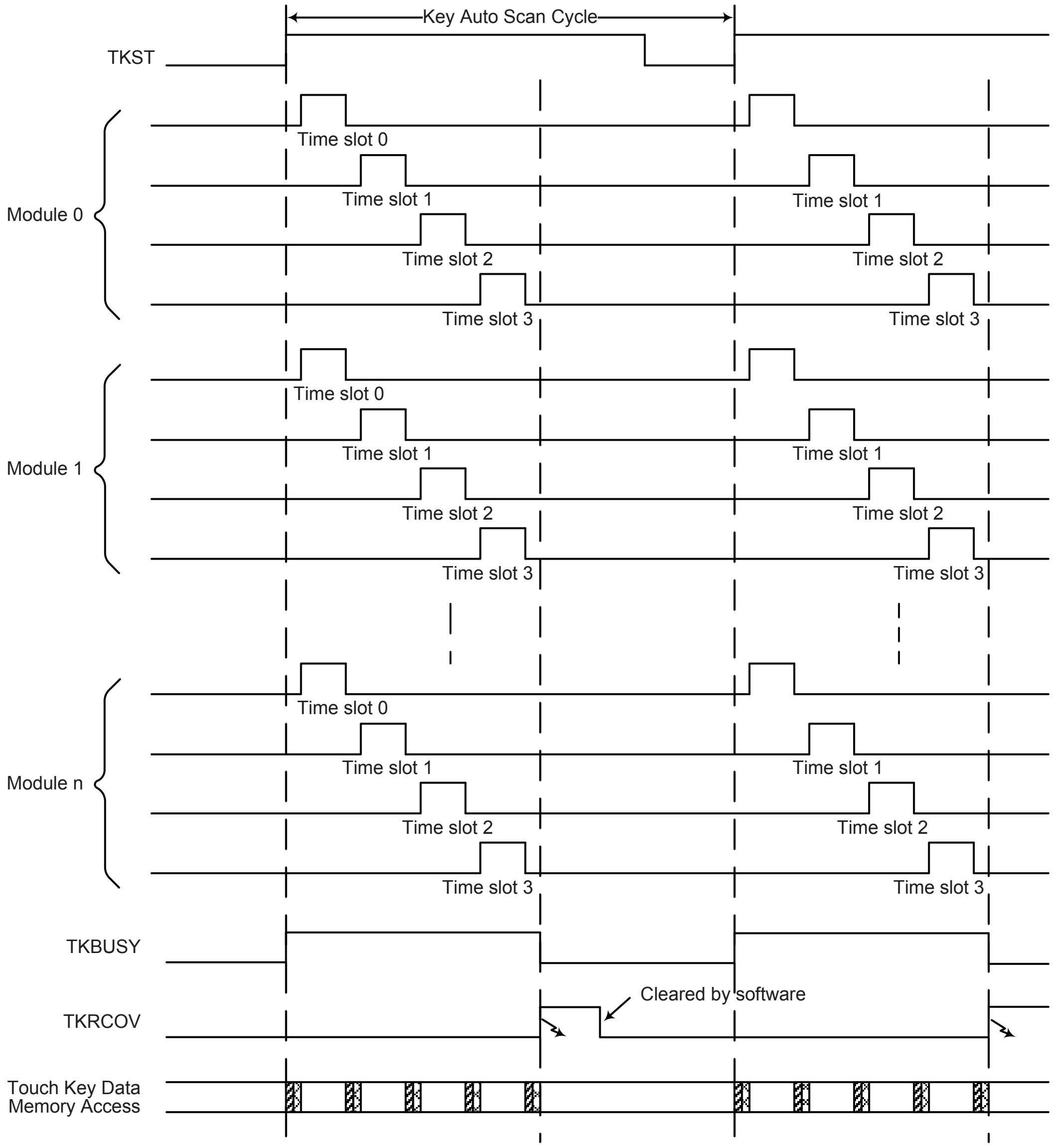
When the time slot counter in all the touch key modules or in the touch key module 0 overflows, an actual touch key interrupt will take place. The touch keys mentioned here are the keys which are enabled.

Each touch key module consists of four touch keys, KEY1~KEY4 are contained in module 0, KEY5~KEY8 are contained in module 1, KEY9~KEY12 are contained in module 2, etc. Each touch key module has an identical structure.

Auto Scan Mode

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

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


Key Auto Scan Cycle
TKST
Module 0
Time slot 0
Time slot 1
Time slot 2
Time slot 3
Time slot 1
Time slot 2
Time slot 3
Module 1
Time slot 0
Time slot 1
Time slot 2
Time slot 3
Time slot 1
Time slot 2
Time slot 3
Module n
Time slot 0
Time slot 1
Time slot 2
Time slot 3
Time slot 1
Time slot 2
Time slot 3
TKBUSY
TKRCOV
Cleared by software
Touch Key Data Memory Access

↘ : Set Touch Key interrupt request flag

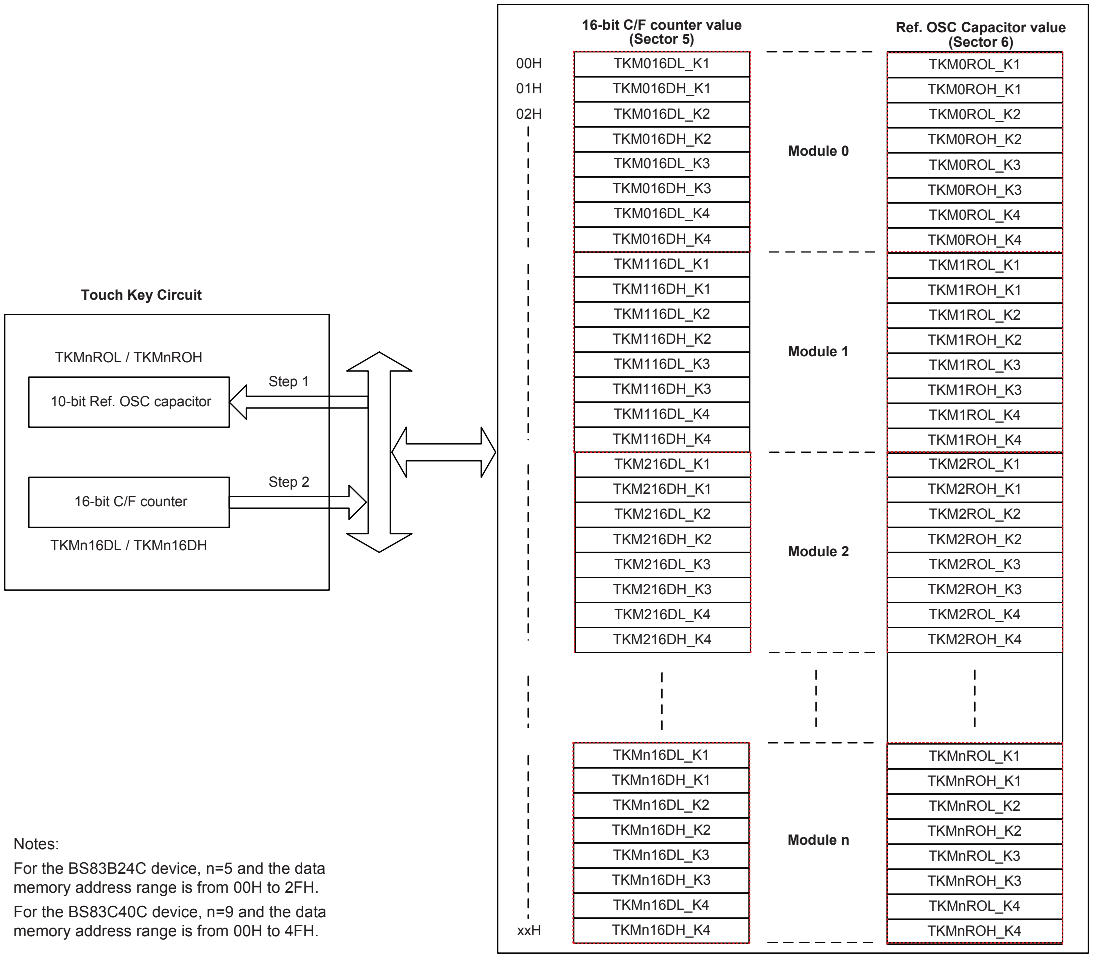
▨ : Read 2N bytes from Touch Key Data Memory to TKMnROH/TKMnTROL registers

⊠ : Write 2N bytes from TKMn16DH/TKMn16DL registers to Touch Key Data Memory
N = Touch Key Module Number; n = Module Serial Number

Touch Key Auto Scan Mode Timing Diagram

Touch Key Data Memory

The devices provide two dedicated Data Memory areas for the touch key auto scan mode. One area is used to store the 16-bit C/F counter values of the touch key module and located in Data Memory Sector 5. The other area is used to store the reference oscillator internal capacitor values of the touch key module and located in Data Memory Sector 6.


Touch Key Circuit
TKMnROL / TKMnROH
10-bit Ref. OSC capacitor
Step 1
16-bit C/F counter
Step 2
TKMn16DL / TKMn16DH
16-bit C/F counter value (Sector 5)
Ref. OSC Capacitor value (Sector 6)
00H
01H
02H
xxH
TKM016DL_K1
TKM016DH_K1
TKM016DL_K2
TKM016DH_K2
TKM016DL_K3
TKM016DH_K3
TKM016DL_K4
TKM016DH_K4
TKM116DL_K1
TKM116DH_K1
TKM116DL_K2
TKM116DH_K2
TKM116DL_K3
TKM116DH_K3
TKM116DL_K4
TKM116DH_K4
TKM216DL_K1
TKM216DH_K1
TKM216DL_K2
TKM216DH_K2
TKM216DL_K3
TKM216DH_K3
TKM216DL_K4
TKM216DH_K4
TKMn16DL_K1
TKMn16DH_K1
TKMn16DL_K2
TKMn16DH_K2
TKMn16DL_K3
TKMn16DH_K3
TKMn16DL_K4
TKMn16DH_K4
Module 0
Module 1
Module 2
Module n
TKM0ROL_K1
TKM0ROH_K1
TKM0ROL_K2
TKM0ROH_K2
TKM0ROL_K3
TKM0ROH_K3
TKM0ROL_K4
TKM0ROH_K4
TKM1ROL_K1
TKM1ROH_K1
TKM1ROL_K2
TKM1ROH_K2
TKM1ROL_K3
TKM1ROH_K3
TKM1ROL_K4
TKM1ROH_K4
TKM2ROL_K1
TKM2ROH_K1
TKM2ROL_K2
TKM2ROH_K2
TKM2ROL_K3
TKM2ROH_K3
TKM2ROL_K4
TKM2ROH_K4
TKMnROL_K1
TKMnROH_K1
TKMnROL_K2
TKMnROH_K2
TKMnROL_K3
TKMnROH_K3
TKMnROL_K4
TKMnROH_K4

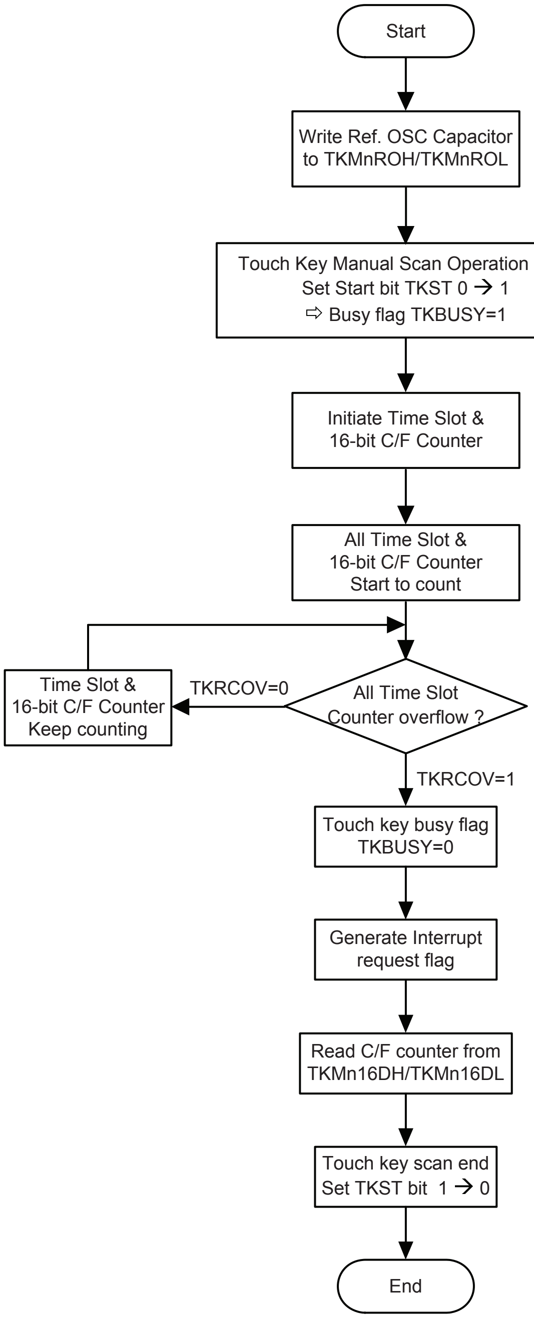
Notes:

For the BS83B24C device, n=5 and the data memory address range is from 00H to 2FH.

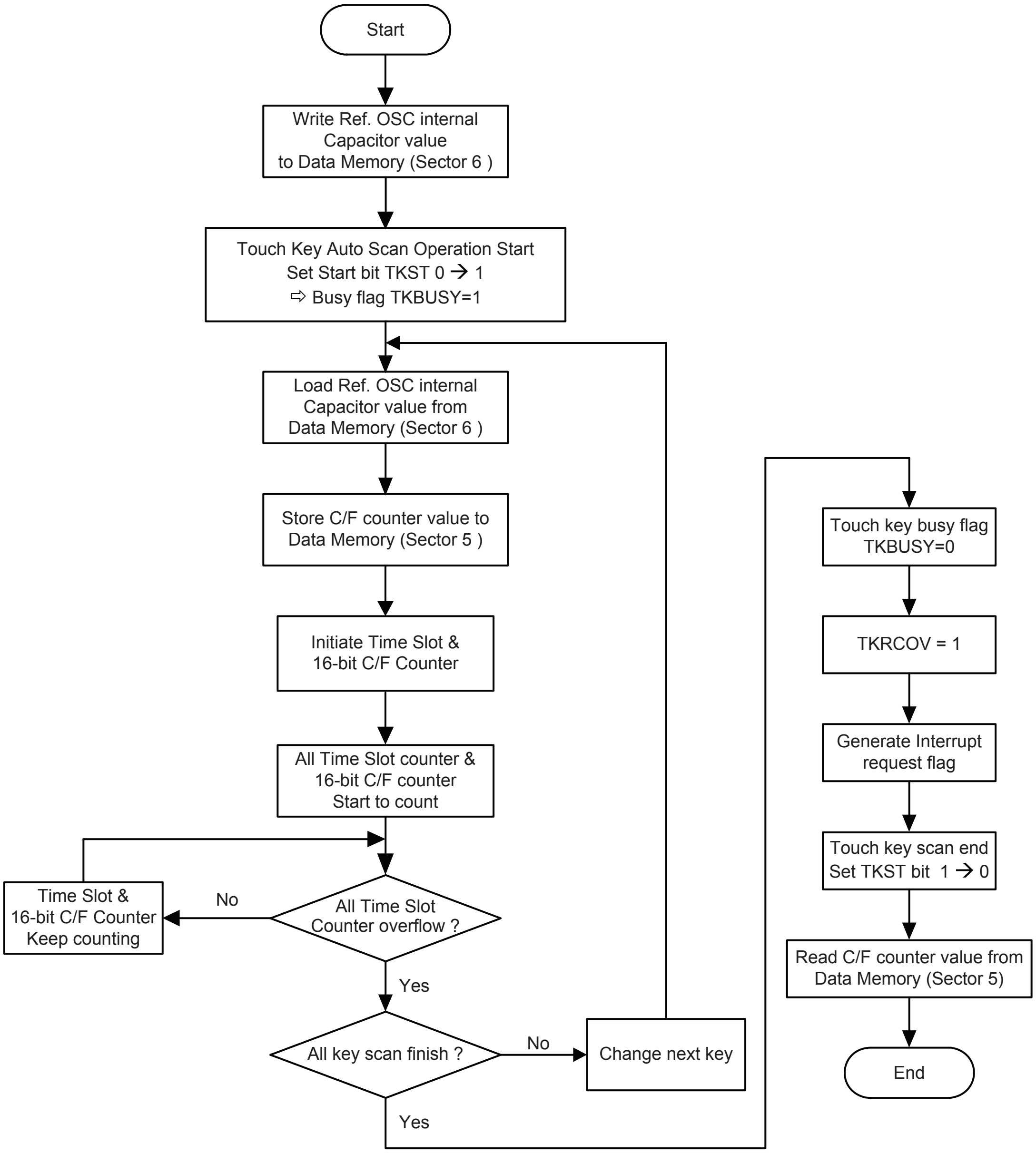
For the BS83C40C device, n=9 and the data memory address range is from 00H to 4FH.

Touch Key Data Memory Map

Touch Key Scan Operation Flowchart


Start
Write Ref. OSC Capacitor to TKMnROH/TKMnROL
Touch Key Manual Scan Operation Start
Set Start bit TKST 0 → 1
⇨ Busy flag TKBUSY=1
Initiate Time Slot & 16-bit C/F Counter
All Time Slot & 16-bit C/F Counter Start to count
All Time Slot Counter overflow ?
TKRCOV=0
Time Slot & 16-bit C/F Counter Keep counting
TKRCOV=1
Touch key busy flag TKBUSY=0
Generate Interrupt request flag
Read C/F counter from TKMn16DH/TKMn16DL
Touch key scan end
Set TKST bit 1 → 0
End

Touch Key Manual Scan Mode Flowchart


Start
Write Ref. OSC internal Capacitor value to Data Memory (Sector 6)
Touch Key Auto Scan Operation Start
Set Start bit TKST 0 → 1
⇨ Busy flag TKBUSY=1
Load Ref. OSC internal Capacitor value from Data Memory (Sector 6)
Store C/F counter value to Data Memory (Sector 5)
Initiate Time Slot & 16-bit C/F Counter
All Time Slot counter & 16-bit C/F counter Start to count
All Time Slot Counter overflow ?
No
Time Slot & 16-bit C/F Counter Keep counting
Yes
All key scan finish ?
No
Change next key
Yes
Touch key busy flag TKBUSY=0
TKRCOV = 1
Generate Interrupt request flag
Touch key scan end
Set TKST bit 1 → 0
Read C/F counter value from Data Memory (Sector 5)
End

Touch Key Auto Scan Mode Flowchart

Touch Key Interrupt

The touch key only has single interrupt, when the time slot counter in all the touch key modules or in the touch key module 0 overflows, an actual touch key interrupt will take place. The touch keys mentioned here are the keys which are enabled. The 16-bit C/F counter, 16-bit counter, 5-bit time slot unit period counter and 8-bit time slot counter in all modules will be automatically cleared.

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

The TK16OV flag which is the 16-bit counter overflow flag will go high when the 16-bit counter overflows. As this flag will not be automatically cleared, it has to be cleared by the application program. More details regarding the touch key interrupt is located in the interrupt section of the datasheet.

Programming Considerations

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

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

Interrupts

Interrupts are an important part of any microcontroller system. When an external event or an internal function such as a Timer Module requires microcontroller attention, their corresponding interrupt will enforce a temporary suspension of the main program allowing the microcontroller to direct attention to their respective needs. These devices contain an external interrupt and several internal interrupt functions. The external interrupt is generated by the action of the external INT pin, while the internal interrupts are generated by various internal functions such as the Timer Modules (TM), Time Bases, EEPROM, Touch Key module and USIM module.

Interrupt Registers

Overall interrupt control, which basically means the setting of request flags when certain microcontroller conditions occur and the setting of interrupt enable bits by the application program, is controlled by a series of registers, located in the Special Purpose Data Memory. The registers fall into three categories. The first is the INTC0~INTC2 registers which setup the primary interrupts, the second is the MFI0~MFI1 registers which setup the Multi-function interrupts. Finally there is an INTEG register to setup the external interrupts 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	Note
Global	EMI	—	—
INT Pin	INTE	INTF	—
Touch Key Module	TKME	TKMF	—
Multi-function	MFnE	MFnF	n=0 for BS83B24C n=0~1 for BS83C40C
USIM	USIME	USIMF	—
Time Bases	TBnE	TBnF	n=0~1
EEPROM	DEE	DEF	—
CTM	CTMPE	CTMPF	Only for BS83C40C
	CTMAE	CTMAF	
PTM	PTMPE	PTMPF	—
	PTMAE	PTMAF	

Interrupt Register Bit Naming Conventions

Register Name	Bit							
	7	6	5	4	3	2	1	0
INTEG	—	—	—	—	—	—	INTS1	INTS0
INTC0	—	MF0F	TKMF	INTF	MF0E	TKME	INTE	EMI
INTC1	DEF	TB1F	TB0F	USIMF	DEE	TB1E	TB0E	USIME
MFI0	—	—	PTMAF	PTMPF	—	—	PTMAE	PTMPE

Interrupt Register List – BS83B24C

Register Name	Bit							
	7	6	5	4	3	2	1	0
INTEG	—	—	—	—	—	—	INTS1	INTS0
INTC0	—	MF0F	TKMF	INTF	MF0E	TKME	INTE	EMI
INTC1	DEF	TB1F	TB0F	USIMF	DEE	TB1E	TB0E	USIME
INTC2	—	—	—	MF1F	—	—	—	MF1E
MFI0	—	—	PTMAF	PTMPF	—	—	PTMAE	PTMPE
MFI1	—	—	CTMAF	CTMPF	—	—	CTMAE	CTMPE

Interrupt Register List – BS83C40C

- **INTEG Register**

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

Bit 7~2 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**

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

Bit 7 Unimplemented, read as “0”

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

Bit 5 **TKMF**: Touch Key Module interrupt request flag
0: No request
1: Interrupt request

Bit 4 **INTF**: External Interrupt request flag
0: No request
1: Interrupt request

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

Bit 2 **TKME**: Touch Key Module interrupt control
0: Disable
1: Enable

Bit 1 **INTE**: External Interrupt control
0: Disable
1: Enable

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

- **INTC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	DEF	TB1F	TB0F	USIMF	DEE	TB1E	TB0E	USIME
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7 **DEF**: Data EEPROM interrupt request flag
0: No request
1: Interrupt request

Bit 6 **TB1F**: Time Base 1 interrupt request flag
0: No request
1: Interrupt request

Bit 5 **TB0F**: Time Base 0 interrupt request flag
0: No request
1: Interrupt request

Bit 4 **USIMF**: USIM Module interrupt request flag
0: No request
1: Interrupt request

Bit 3 **DEE**: Data EEPROM interrupt control
0: Disable
1: Enable

Bit 2 **TB1E**: Time Base 1 interrupt control
0: Disable
1: Enable

Bit 1 **TB0E**: Time Base 0 interrupt control
0: Disable
1: Enable

Bit 0 **USIME**: USIM Module interrupt control
0: Disable
1: Enable

- **INTC2 Register – BS83C40C**

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

Bit 7~5 Unimplemented, read as “0”

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

Bit 3~1 Unimplemented, read as “0”

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

- **MFI0 Register**

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

Bit 7~6 Unimplemented, read as “0”

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

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

Bit 3~2 Unimplemented, read as "0"

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

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

- **MFI1 Register – BS83C40C**

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

Bit 7~6 Unimplemented, read as "0"

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

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

Bit 3~2 Unimplemented, read as "0"

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

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

Interrupt Operation

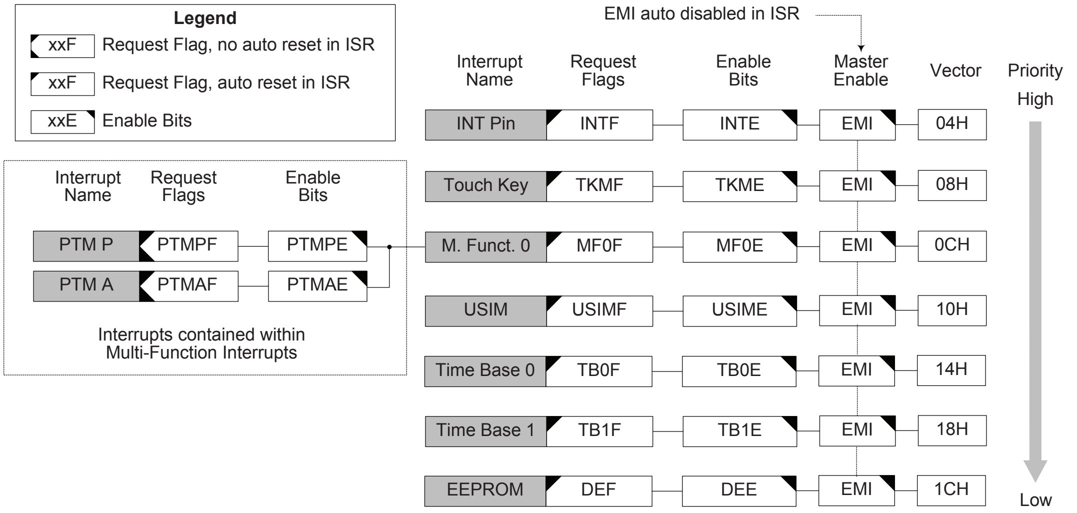
When the conditions for an interrupt event occur, such as a TM Comparator P, Comparator A match or an EEPROM Write cycle ends etc., the relevant interrupt request flag will be set. Whether the request flag actually generates a program jump to the relevant interrupt vector is determined by the condition of the interrupt enable bit. If the enable bit is set high then the program will jump to its relevant vector; if the enable bit is zero then although the interrupt request flag is set an actual interrupt will not be generated and the program will not jump to the relevant interrupt vector. The global interrupt enable bit, if cleared to zero, will disable all interrupts.

When an interrupt is generated, the Program Counter, which stores the address of the next instruction to be executed, will be transferred onto the stack. The Program Counter will then be loaded with a new address which will be the value of the corresponding interrupt vector. The microcontroller will then fetch its next instruction from this interrupt vector. The instruction at this vector will usually be a "JMP" which will jump to another section of program which is known as the interrupt service routine. Here is located the code to control the appropriate interrupt. The interrupt service routine must be terminated with a "RETI", which retrieves the original Program Counter address from the stack and allows the microcontroller to continue with normal execution at the point where the interrupt occurred.

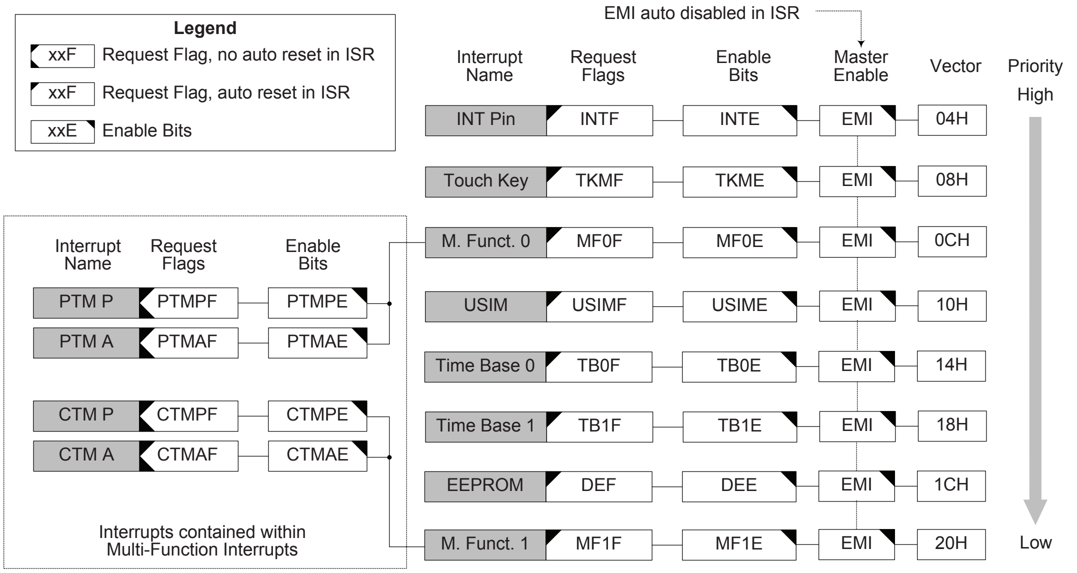
The various interrupt enable bits, together with their associated request flags, are shown in the accompanying diagrams with their order of priority. Some interrupt sources have their own individual vector while others share the same multi-function interrupt vector. Once an interrupt subroutine is serviced, all the other interrupts will be blocked, as the global interrupt enable bit, EMI bit will be cleared automatically. This will prevent any further interrupt nesting from occurring.

However, if other interrupt requests occur during this interval, although the interrupt will not be immediately serviced, the request flag will still be recorded.

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


Legend
xxF Request Flag, no auto reset in ISR
xxF Request Flag, auto reset in ISR
xxE Enable Bits
EMI auto disabled in ISR
Interrupt Name
Request Flags
Enable Bits
Master Enable
Vector
Priority
High
Low
INT Pin INTF INTE EMI 04H
Touch Key TKMF TKME EMI 08H
M. Funct. 0 MF0F MF0E EMI 0CH
USIM USIMF USIME EMI 10H
Time Base 0 TB0F TB0E EMI 14H
Time Base 1 TB1F TB1E EMI 18H
EEPROM DEF DEE EMI 1CH
PTM P PTMPF PTMPE
PTM A PTMAF PTMAE
Interrupts contained within Multi-Function Interrupts

Interrupt Structure – BS83B24C


Legend
xxF Request Flag, no auto reset in ISR
xxF Request Flag, auto reset in ISR
xxE Enable Bits
EMI auto disabled in ISR
Interrupt Name
Request Flags
Enable Bits
Master Enable
Vector
Priority
High
Low
INT Pin INTF INTE EMI 04H
Touch Key TKMF TKME EMI 08H
M. Funct. 0 MF0F MF0E EMI 0CH
USIM USIMF USIME EMI 10H
Time Base 0 TB0F TB0E EMI 14H
Time Base 1 TB1F TB1E EMI 18H
EEPROM DEF DEE EMI 1CH
M. Funct. 1 MF1F MF1E EMI 20H
PTM P PTMPF PTMPE
PTM A PTMAF PTMAE
CTM P CTMPF CTMPE
CTM A CTMAF CTMAE
Interrupts contained within Multi-Function Interrupts

Interrupt Structure – BS83C40C

External Interrupt

The external interrupts are 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 pins. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and respective external interrupt enable bit, 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 pins are pin-shared with I/O pins, they can only be configured as external interrupt pins if their external interrupt enable bit in the corresponding interrupt register has been set and the external interrupt pin is selected by the corresponding pin-shared function selection bits. The pin must also be setup as an input by setting the corresponding bit in the port control register.

When the interrupt is enabled, the stack is not full and the correct transition type appears on the external interrupt pin, a subroutine call to the external interrupt vector, will take place. When the interrupt is serviced, the external interrupt request 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 pins will remain valid even if the pin is used as an external interrupt input. The INTEG register is used to select the type of active edge that will trigger the external interrupt. A choice of either rising or falling or both edge types can be chosen to trigger an external interrupt. Note that the INTEG register can also be used to disable the external interrupt function.

Multi-function Interrupt

Within these devices there are several Multi-function interrupts. Unlike the other independent interrupts, these interrupts have no independent source, but rather are formed from other existing interrupt sources, namely the TM interrupts.

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

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

Timer Module Interrupts

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

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

EEPROM Interrupt

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

USIM Interrupt

The Universal Serial Interface Module Interrupt, also known as the USIM interrupt, will take place when the USIM Interrupt request flag, USIMF, is set. As the USIM interface can operate in three modes which are SPI mode, I^2C mode and UART mode, the USIMF flag can be set by different conditions depending on the selected interface mode.

If the SPI or I^2C mode is selected, the USIM interrupt can be triggered when a byte of data has been received or transmitted by the USIM SPI or I^2C interface, or an I^2C slave address match occurs, or an I^2C bus time-out occurs. If the UART mode is selected, several individual UART conditions including a transmitter data register empty, transmitter idle, receiver data available, receiver overrun, address detect and an RX pin wake-up, can generate a USIM interrupt with the USIMF flag bit set high.

To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and the Universal Serial Interface Module Interrupt enable bit, USIME, must first be set. When the interrupt is enabled, the stack is not full and any of the above described situations occurs, a subroutine call to the respective Interrupt vector, will take place. When the interrupt is serviced, the Universal Serial Interface Module Interrupt flag, USIMF, will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts.

Note that if the USIM interrupt is triggered by the UART interface, after the interrupt has been servied, the UUSR register flags will be cleared automatically when certain actions are taken by the UART, the details of which are given in the UART section.

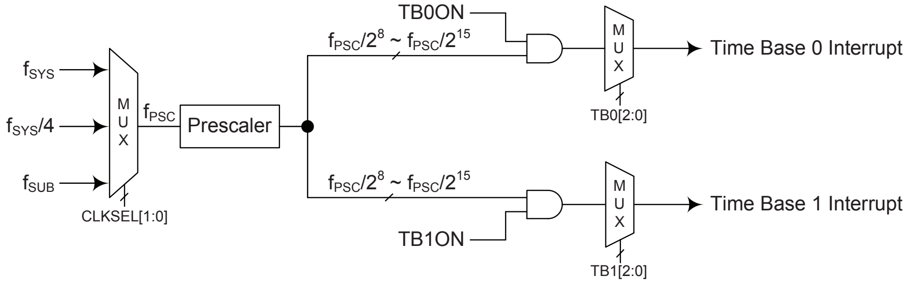
Touch Key Interrupt

An Touch Key Interrupt request will take place when the Touch Key Interrupt request flag, TKMF, is set, which occurs when the touch key time slot counter overflows. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and the Touch Key Interrupt enable bit, TKME, must first be set. When the interrupt is enabled, the stack is not full and the touch key time slot counter overflows, a subroutine call to the Touch Key Interrupt vector, will take place. When the Touch Key Interrupt is serviced, the TKRMF flag will be automatically cleared, the EMI bit will also be automatically cleared to disable other interrupts.

Time Base Interrupts

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

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



Time Base Interrupts

- **PSCR Register**

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

Bit 7~2	Unimplemented, read as "0"
Bit 1~0	**CLKSEL1~CLKSEL0**: Prescaler clock source selection 00: f_{SYS} 01: $f_{SYS}/4$ 1x: f_{SUB}

- **TB0C Register**

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

Bit 7 **TB0ON**: Time Base 0 Control
0: Disable
1: Enable

Bit 6~3 Unimplemented, read as "0"

Bit 2~0 **TB02~TB00**: Select Time Base 0 Time-out Period
000: $2^8/f_{PSC}$
001: $2^9/f_{PSC}$
010: $2^{10}/f_{PSC}$
011: $2^{11}/f_{PSC}$
100: $2^{12}/f_{PSC}$
101: $2^{13}/f_{PSC}$
110: $2^{14}/f_{PSC}$
111: $2^{15}/f_{PSC}$

- **TB1C Register**

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

Bit 7 **TB1ON**: Time Base 1 Control
0: Disable
1: Enable

Bit 6~3 Unimplemented, read as "0"

Bit 2~0 **TB12~TB10**: Select Time Base 1 Time-out Period
000: $2^8/f_{PSC}$
001: $2^9/f_{PSC}$
010: $2^{10}/f_{PSC}$
011: $2^{11}/f_{PSC}$
100: $2^{12}/f_{PSC}$
101: $2^{13}/f_{PSC}$
110: $2^{14}/f_{PSC}$
111: $2^{15}/f_{PSC}$

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 or a low power supply voltage may cause their respective interrupt flag to be set high and consequently generate an interrupt. Care must therefore be taken if spurious wake-up situations are to be avoided. If an interrupt wake-up function is to be disabled then the corresponding interrupt request flag should be set high before the device enters the SLEEP or IDLE Mode. The interrupt enable bits have no effect on the interrupt wake-up function.

Programming Considerations

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

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

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

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

As only the Program Counter is pushed onto the stack, then when the interrupt is serviced, if the contents of the accumulator, status register or other registers are altered by the interrupt service program, their contents should be saved to the memory at the beginning of the interrupt service routine. To return from an interrupt subroutine, either a RET or RETI instruction may be executed. The RETI instruction in addition to executing a return to the main program also automatically sets the EMI bit high to allow further interrupts. The RET instruction however only executes a return to the main program leaving the EMI bit in its present zero state and therefore disabling the execution of further interrupts.

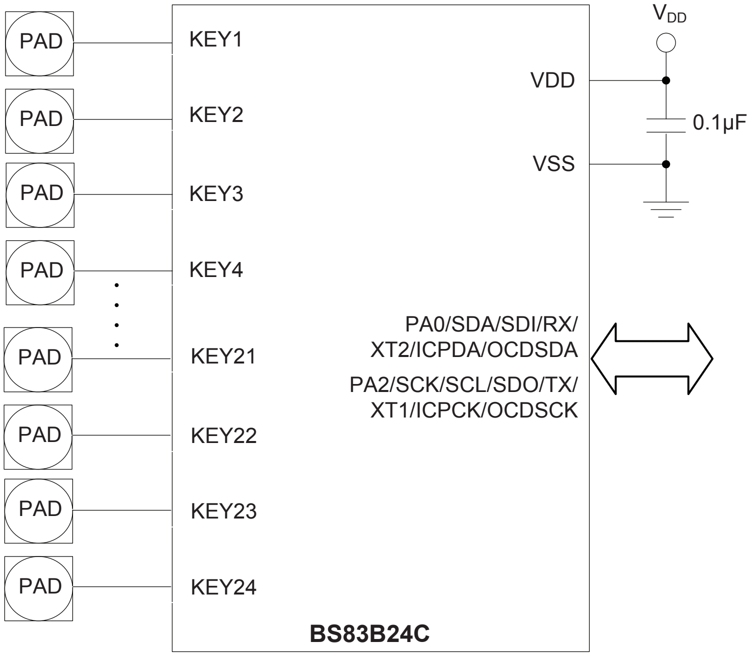
Configuration Options

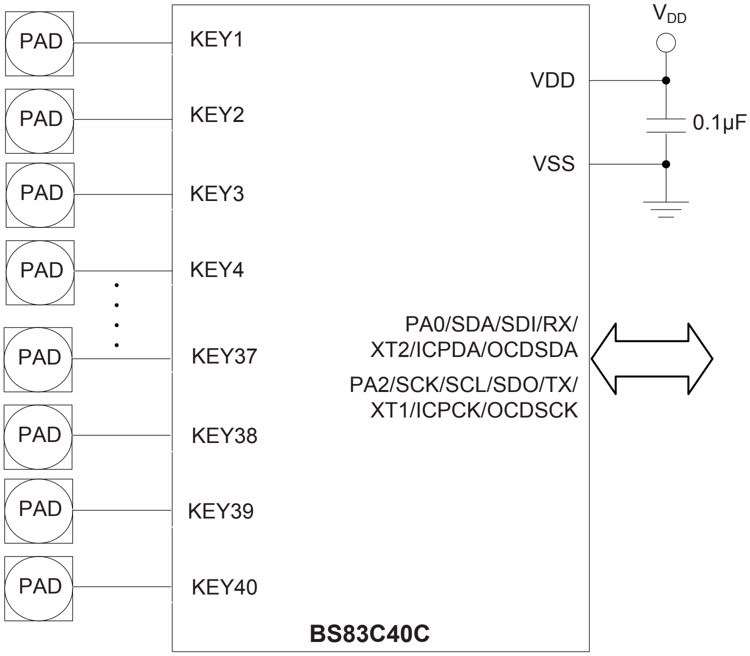
Configuration options refer to certain options within the MCU that are programmed into the device during the programming process. During the development process, these options are selected using the HT-IDE software development tools. As these options are programmed into the device using the hardware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

No.	Options
Oscillator Option	
1	HIRC frequency selection: 1. 8MHz 2. 12MHz 3. 16MHz

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

Application Circuits


PAD
KEY1
KEY2
KEY3
KEY4
KEY21
KEY22
KEY23
KEY24
VDD
VSS
V_DD
0.1μF
PA0/SDA/SDI/RX/
XT2/ICPDA/OCDSDA
PA2/SCK/SCL/SDO/TX/
XT1/ICPCK/OCDSCK
BS83B24C


PAD
KEY1
KEY2
KEY3
KEY4
KEY37
KEY38
KEY39
KEY40
VDD
VSS
V_DD
0.1μF
PA0/SDA/SDI/RX/
XT2/ICPDA/OCDSDA
PA2/SCK/SCL/SDO/TX/
XT1/ICPCK/OCDSCK
BS83C40C

Instruction Set

Introduction

Central to the successful operation of any microcontroller is its instruction set, which is a set of program instruction codes that directs the microcontroller to perform certain operations. In the case of Holtek microcontroller, a comprehensive and flexible set of over 60 instructions is provided to enable programmers to implement their application with the minimum of programming overheads.

For easier understanding of the various instruction codes, they have been subdivided into several functional groupings.

Instruction Timing

Most instructions are implemented within one instruction cycle. The exceptions to this are branch, call, or table read instructions where two instruction cycles are required. One instruction cycle is equal to 4 system clock cycles, therefore in the case of an 8MHz system oscillator, most instructions would be implemented within 0.5μs and branch or call instructions would be implemented within 1μs. Although instructions which require one more cycle to implement are generally limited to the JMP, CALL, RET, RETI and table read instructions, it is important to realize that any other instructions which involve manipulation of the Program Counter Low register or PCL will also take one more cycle to implement. As instructions which change the contents of the PCL will imply a direct jump to that new address, one more cycle will be required. Examples of such instructions would be "CLR PCL" or "MOV PCL, A". For the case of skip instructions, it must be noted that if the result of the comparison involves a skip operation then this will also take one more cycle, if no skip is involved then only one cycle is required.

Moving and Transferring Data

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

Arithmetic Operations

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

Logical and Rotate Operation

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

Branches and Control Transfer

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

Bit Operations

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

Table Read Operations

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

Other Operations

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

Instruction Set Summary

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

Table Conventions

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

Mnemonic	Description	Cycles	Flag Affected
Arithmetic			
ADD A,[m]	Add Data Memory to ACC	1	Z, C, AC, OV, SC
ADDM A,[m]	Add ACC to Data Memory	1^{Note}	Z, C, AC, OV, SC
ADD A,x	Add immediate data to ACC	1	Z, C, AC, OV, SC
ADC A,[m]	Add Data Memory to ACC with Carry	1	Z, C, AC, OV, SC
ADCM A,[m]	Add ACC to Data memory with Carry	1^{Note}	Z, C, AC, OV, SC
SUB A,x	Subtract immediate data from the ACC	1	Z, C, AC, OV, SC, CZ
SUB A,[m]	Subtract Data Memory from ACC	1	Z, C, AC, OV, SC, CZ
SUBM A,[m]	Subtract Data Memory from ACC with result in Data Memory	1^{Note}	Z, C, AC, OV, SC, CZ
SBC A,x	Subtract immediate data from ACC with Carry	1	Z, C, AC, OV, SC, CZ
SBC A,[m]	Subtract Data Memory from ACC with Carry	1	Z, C, AC, OV, SC, CZ
SBCM A,[m]	Subtract Data Memory from ACC with Carry, result in Data Memory	1^{Note}	Z, C, AC, OV, SC, CZ
DAA [m]	Decimal adjust ACC for Addition with result in Data Memory	1^{Note}	C
Logic Operation			
AND A,[m]	Logical AND Data Memory to ACC	1	Z
OR A,[m]	Logical OR Data Memory to ACC	1	Z
XOR A,[m]	Logical XOR Data Memory to ACC	1	Z
ANDM A,[m]	Logical AND ACC to Data Memory	1^{Note}	Z
ORM A,[m]	Logical OR ACC to Data Memory	1^{Note}	Z
XORM A,[m]	Logical XOR ACC to Data Memory	1^{Note}	Z
AND A,x	Logical AND immediate Data to ACC	1	Z
OR A,x	Logical OR immediate Data to ACC	1	Z
XOR A,x	Logical XOR immediate Data to ACC	1	Z
CPL [m]	Complement Data Memory	1^{Note}	Z
CPLA [m]	Complement Data Memory with result in ACC	1	Z
Increment & Decrement			
INCA [m]	Increment Data Memory with result in ACC	1	Z
INC [m]	Increment Data Memory	1^{Note}	Z
DECA [m]	Decrement Data Memory with result in ACC	1	Z
DEC [m]	Decrement Data Memory	1^{Note}	Z
Rotate			
RRA [m]	Rotate Data Memory right with result in ACC	1	None
RR [m]	Rotate Data Memory right	1^{Note}	None
RRCA [m]	Rotate Data Memory right through Carry with result in ACC	1	C
RRC [m]	Rotate Data Memory right through Carry	1^{Note}	C
RLA [m]	Rotate Data Memory left with result in ACC	1	None
RL [m]	Rotate Data Memory left	1^{Note}	None
RLCA [m]	Rotate Data Memory left through Carry with result in ACC	1	C
RLC [m]	Rotate Data Memory left through Carry	1^{Note}	C

Mnemonic	Description	Cycles	Flag Affected
Data Move			
MOV A,[m]	Move Data Memory to ACC	1	None
MOV [m],A	Move ACC to Data Memory	1^Note	None
MOV A,x	Move immediate data to ACC	1	None
Bit Operation			
CLR [m].i	Clear bit of Data Memory	1^Note	None
SET [m].i	Set bit of Data Memory	1^Note	None
Branch Operation			
JMP addr	Jump unconditionally	2	None
SZ [m]	Skip if Data Memory is zero	1^Note	None
SZA [m]	Skip if Data Memory is zero with data movement to ACC	1^Note	None
SZ [m].i	Skip if bit i of Data Memory is zero	1^Note	None
SNZ [m]	Skip if Data Memory is not zero	1^Note	None
SNZ [m].i	Skip if bit i of Data Memory is not zero	1^Note	None
SIZ [m]	Skip if increment Data Memory is zero	1^Note	None
SDZ [m]	Skip if decrement Data Memory is zero	1^Note	None
SIZA [m]	Skip if increment Data Memory is zero with result in ACC	1^Note	None
SDZA [m]	Skip if decrement Data Memory is zero with result in ACC	1^Note	None
CALL addr	Subroutine call	2	None
RET	Return from subroutine	2	None
RET A,x	Return from subroutine and load immediate data to ACC	2	None
RETI	Return from interrupt	2	None
Table Read Operation			
TABRD [m]	Read table (specific page) to TBLH and Data Memory	2^Note	None
TABRDL [m]	Read table (last page) to TBLH and Data Memory	2^Note	None
ITABRD [m]	Increment table pointer TBLP first and Read table to TBLH and Data Memory	2^Note	None
ITABRDL [m]	Increment table pointer TBLP first and Read table (last page) to TBLH and Data Memory	2^Note	None
Miscellaneous			
NOP	No operation	1	None
CLR [m]	Clear Data Memory	1^Note	None
SET [m]	Set Data Memory	1^Note	None
CLR WDT	Clear Watchdog Timer	1	TO, PDF
SWAP [m]	Swap nibbles of Data Memory	1^Note	None
SWAPA [m]	Swap nibbles of Data Memory with result in ACC	1	None
HALT	Enter power down mode	1	TO, PDF

Note: 1. For skip instructions, if the result of the comparison involves a skip then up to three cycles are required, if no skip takes place only one cycle is required.

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

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

Extended Instruction Set

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

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

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

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

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

Instruction Definition

ADC A,[m]	Add Data Memory to ACC with Carry
Description	The contents of the specified Data Memory, Accumulator and the carry flag are added. The result is stored in the Accumulator.
Operation	ACC ← ACC + [m] + C
Affected flag(s)	OV, Z, AC, C, SC

ADCM A,[m]	Add ACC to Data Memory with Carry
Description	The contents of the specified Data Memory, Accumulator and the carry flag are added. The result is stored in the specified Data Memory.
Operation	[m] ← ACC + [m] + C
Affected flag(s)	OV, Z, AC, C, SC

ADD A,[m]	Add Data Memory to ACC
Description	The contents of the specified Data Memory and the Accumulator are added. The result is stored in the Accumulator.
Operation	ACC ← ACC + [m]
Affected flag(s)	OV, Z, AC, C, SC

ADD A,x	Add immediate data to ACC
Description	The contents of the Accumulator and the specified immediate data are added. The result is stored in the Accumulator.
Operation	ACC ← ACC + x
Affected flag(s)	OV, Z, AC, C, SC

ADDM A,[m]	Add ACC to Data Memory
Description	The contents of the specified Data Memory and the Accumulator are added. The result is stored in the specified Data Memory.
Operation	[m] ← ACC + [m]
Affected flag(s)	OV, Z, AC, C, SC

AND A,[m]	Logical AND Data Memory to ACC
Description	Data in the Accumulator and the specified Data Memory perform a bitwise logical AND operation. The result is stored in the Accumulator.
Operation	ACC ← ACC ″AND″ [m]
Affected flag(s)	Z

AND A,x	Logical AND immediate data to ACC
Description	Data in the Accumulator and the specified immediate data perform a bit wise logical AND operation. The result is stored in the Accumulator.
Operation	ACC ← ACC ″AND″ x
Affected flag(s)	Z

ANDM A,[m]	Logical AND ACC to Data Memory
Description	Data in the specified Data Memory and the Accumulator perform a bitwise logical AND operation. The result is stored in the Data Memory.
Operation	[m] ← ACC ″AND″ [m]
Affected flag(s)	Z

CALL addr	Subroutine call
Description	Unconditionally calls a subroutine at the specified address. The Program Counter then increments by 1 to obtain the address of the next instruction which is then pushed onto the stack. The specified address is then loaded and the program continues execution from this new address. As this instruction requires an additional operation, it is a two cycle instruction.
Operation	Stack ← Program Counter + 1 Program Counter ← addr
Affected flag(s)	None

CLR [m]	Clear Data Memory
Description	Each bit of the specified Data Memory is cleared to 0.
Operation	[m] ← 00H
Affected flag(s)	None

CLR [m].i	Clear bit of Data Memory
Description	Bit i of the specified Data Memory is cleared to 0.
Operation	[m].i ← 0
Affected flag(s)	None

CLR WDT	Clear Watchdog Timer
Description	The TO, PDF flags and the WDT are all cleared.
Operation	WDT cleared TO ← 0 PDF ← 0
Affected flag(s)	TO, PDF

CPL [m]	Complement Data Memory
Description	Each bit of the specified Data Memory is logically complemented (1′s complement). Bits which previously contained a 1 are changed to 0 and vice versa.
Operation	$[m] \leftarrow \overline{[m]}$
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] ← ACC + 00H or [m] ← ACC + 06H or [m] ← ACC + 60H or [m] ← ACC + 66H
Affected flag(s)	C

DEC [m]	Decrement Data Memory
Description	Data in the specified Data Memory is decremented by 1.
Operation	[m] ← [m] − 1
Affected flag(s)	Z

DECA [m]	Decrement Data Memory with result in ACC
Description	Data in the specified Data Memory is decremented by 1. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.
Operation	ACC ← [m] − 1
Affected flag(s)	Z

HALT	Enter power down mode
Description	This instruction stops the program execution and turns off the system clock. The contents of the Data Memory and registers are retained. The WDT and prescaler are cleared. The power down flag PDF is set and the WDT time-out flag TO is cleared.
Operation	TO ← 0 PDF ← 1
Affected flag(s)	TO, PDF

INC [m]	Increment Data Memory
Description	Data in the specified Data Memory is incremented by 1.
Operation	[m] ← [m] + 1
Affected flag(s)	Z

INCA [m]	Increment Data Memory with result in ACC
Description	Data in the specified Data Memory is incremented by 1. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.
Operation	ACC ← [m] + 1
Affected flag(s)	Z

JMP addr	Jump unconditionally
Description	The contents of the Program Counter are replaced with the specified address. Program execution then continues from this new address. As this requires the insertion of a dummy instruction while the new address is loaded, it is a two cycle instruction.
Operation	Program Counter ← addr
Affected flag(s)	None

MOV A,[m]	Move Data Memory to ACC
Description	The contents of the specified Data Memory are copied to the Accumulator.
Operation	ACC ← [m]
Affected flag(s)	None

MOV A,x	Move immediate data to ACC
Description	The immediate data specified is loaded into the Accumulator.
Operation	ACC ← x
Affected flag(s)	None

MOV [m],A	Move ACC to Data Memory
Description	The contents of the Accumulator are copied to the specified Data Memory.
Operation	[m] ← ACC
Affected flag(s)	None

NOP	No operation
Description	No operation is performed. Execution continues with the next instruction.
Operation	No operation
Affected flag(s)	None

OR A,[m]	Logical OR Data Memory to ACC
Description	Data in the Accumulator and the specified Data Memory perform a bitwise logical OR operation. The result is stored in the Accumulator.
Operation	ACC ← ACC ″OR″ [m]
Affected flag(s)	Z

OR A,x	Logical OR immediate data to ACC
Description	Data in the Accumulator and the specified immediate data perform a bitwise logical OR operation. The result is stored in the Accumulator.
Operation	ACC ← ACC ″OR″ x
Affected flag(s)	Z

ORM A,[m]	Logical OR ACC to Data Memory
Description	Data in the specified Data Memory and the Accumulator perform a bitwise logical OR operation. The result is stored in the Data Memory.
Operation	[m] ← ACC ″OR″ [m]
Affected flag(s)	Z

RET	Return from subroutine
Description	The Program Counter is restored from the stack. Program execution continues at the restored address.
Operation	Program Counter ← Stack
Affected flag(s)	None

RET A,x	Return from subroutine and load immediate data to ACC
Description	The Program Counter is restored from the stack and the Accumulator loaded with the specified immediate data. Program execution continues at the restored address.
Operation	Program Counter ← Stack ACC ← x
Affected flag(s)	None

RETI	Return from interrupt
Description	The Program Counter is restored from the stack and the interrupts are re-enabled by setting the EMI bit. EMI is the master interrupt global enable bit. If an interrupt was pending when the RETI instruction is executed, the pending Interrupt routine will be processed before returning to the main program.
Operation	Program Counter ← Stack EMI ← 1
Affected flag(s)	None

RL [m]	Rotate Data Memory left
Description	The contents of the specified Data Memory are rotated left by 1 bit with bit 7 rotated into bit 0.
Operation	[m].(i+1) ← [m].i; (i=0~6) [m].0 ← [m].7
Affected flag(s)	None

RLA [m]	Rotate Data Memory left with result in ACC
Description	The contents of the specified Data Memory are rotated left by 1 bit with bit 7 rotated into bit 0. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.
Operation	ACC.(i+1) ← [m].i; (i=0~6) ACC.0 ← [m].7
Affected flag(s)	None

RLC [m]	Rotate Data Memory left through Carry
Description	The contents of the specified Data Memory and the carry flag are rotated left by 1 bit. Bit 7 replaces the Carry bit and the original carry flag is rotated into bit 0.
Operation	[m].(i+1) ← [m].i; (i=0~6) [m].0 ← C C ← [m].7
Affected flag(s)	C

RLCA [m]	Rotate Data Memory left through Carry with result in ACC
Description	Data in the specified Data Memory and the carry flag are rotated left by 1 bit. Bit 7 replaces the Carry bit and the original carry flag is rotated into the bit 0. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.
Operation	ACC.(i+1) ← [m].i; (i=0~6) ACC.0 ← C C ← [m].7
Affected flag(s)	C

RR [m]	Rotate Data Memory right
Description	The contents of the specified Data Memory are rotated right by 1 bit with bit 0 rotated into bit 7.
Operation	[m].i ← [m].(i+1); (i=0~6) [m].7 ← [m].0
Affected flag(s)	None

RRA [m]	Rotate Data Memory right with result in ACC
Description	Data in the specified Data Memory is rotated right by 1 bit with bit 0 rotated into bit 7. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.
Operation	ACC.i ← [m].(i+1); (i=0~6) ACC.7 ← [m].0
Affected flag(s)	None

RRC [m]	Rotate Data Memory right through Carry
Description	The contents of the specified Data Memory and the carry flag are rotated right by 1 bit. Bit 0 replaces the Carry bit and the original carry flag is rotated into bit 7.
Operation	[m].i ← [m].(i+1); (i=0~6) [m].7 ← C C ← [m].0
Affected flag(s)	C

RRCA [m]	Rotate Data Memory right through Carry with result in ACC
Description	Data in the specified Data Memory and the carry flag are rotated right by 1 bit. Bit 0 replaces the Carry bit and the original carry flag is rotated into bit 7. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.
Operation	ACC.i ← [m].(i+1); (i=0~6) ACC.7 ← C C ← [m].0
Affected flag(s)	C

SBC A,[m]	Subtract Data Memory from ACC with Carry
Description	The contents of the specified Data Memory and the complement of the carry flag are subtracted from the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.
Operation	ACC ← ACC − [m] − $\overline{C}$
Affected flag(s)	OV, Z, AC, C, SC, CZ

SBC A, x	Subtract immediate data from ACC with Carry
Description	The immediate data and the complement of the carry flag are subtracted from the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.
Operation	ACC ← ACC - [m] - $\overline{C}$
Affected flag(s)	OV, Z, AC, C, SC, CZ

SBCM A,[m]	Subtract Data Memory from ACC with Carry and result in Data Memory
Description	The contents of the specified Data Memory and the complement of the carry flag are subtracted from the Accumulator. The result is stored in the Data Memory. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.
Operation	[m] ← ACC − [m] − $\overline{C}$
Affected flag(s)	OV, Z, AC, C, SC, CZ

SDZ [m]	Skip if decrement Data Memory is 0
Description	The contents of the specified Data Memory are first decremented by 1. If the result is 0 the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.
Operation	[m] ← [m] − 1 Skip if [m]=0
Affected flag(s)	None

SDZA [m]	Skip if decrement Data Memory is zero with result in ACC
Description	The contents of the specified Data Memory are first decremented by 1. If the result is 0, the following instruction is skipped. The result is stored in the Accumulator but the specified Data Memory contents remain unchanged. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0, the program proceeds with the following instruction.
Operation	ACC ← [m] − 1 Skip if ACC=0
Affected flag(s)	None

SET [m]	Set Data Memory
Description	Each bit of the specified Data Memory is set to 1.
Operation	[m] ← 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 Data Memory is not 0
Description	If the specified Data Memory is not 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is 0 the program proceeds with the following instruction.
Operation	Skip if [m].i ≠ 0
Affected flag(s)	None

SNZ [m]	Skip if Data Memory is not 0
Description	If the specified Data Memory is not 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is 0 the program proceeds with the following instruction.
Operation	Skip if [m]≠ 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, SC, CZ

SUBM A,[m]	Subtract Data Memory from ACC with result in Data Memory
Description	The specified Data Memory is subtracted from the contents of the Accumulator. The result is stored in the Data Memory. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.
Operation	[m] ← ACC − [m]
Affected flag(s)	OV, Z, AC, C, SC, CZ

SUB A,x	Subtract immediate data from ACC
Description	The immediate data specified by the code is subtracted from the contents of the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.
Operation	ACC ← ACC − x
Affected flag(s)	OV, Z, AC, C, SC, CZ

SWAP [m]	Swap nibbles of Data Memory
Description	The low-order and high-order nibbles of the specified Data Memory are interchanged.
Operation	[m].3~[m].0 ↔ [m].7~[m].4
Affected flag(s)	None

SWAPA [m]	Swap nibbles of Data Memory with result in ACC
Description	The low-order and high-order nibbles of the specified Data Memory are interchanged. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.
Operation	ACC.3~ACC.0 ← [m].7~[m].4 ACC.7~ACC.4 ← [m].3~[m].0
Affected flag(s)	None

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

SZA [m]	Skip if Data Memory is 0 with data movement to ACC
Description	The contents of the specified Data Memory are copied to the Accumulator. If the value is zero, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.
Operation	ACC ← [m] Skip if [m]=0
Affected flag(s)	None

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

TABRD [m]	Read table (specific page) to TBLH and Data Memory
Description	The low byte of the program code (specific page) addressed by the table pointer pair (TBLP and TBHP) is moved to the specified Data Memory and the high byte moved to TBLH.
Operation	[m] ← program code (low byte) TBLH ← program code (high byte)
Affected flag(s)	None

TABRDL [m]	Read table (last page) to TBLH and Data Memory
Description	The low byte of the program code (last page) addressed by the table pointer (TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.
Operation	[m] ← program code (low byte) TBLH ← program code (high byte)
Affected flag(s)	None

ITABRD [m]	Increment table pointer low byte first and read table to TBLH and Data Memory
Description	Increment table pointer low byte, TBLP, first and then the program code addressed by the table pointer (TBHP and TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.
Operation	[m] ← program code (low byte) TBLH ← program code (high byte)
Affected flag(s)	None

ITABRDL [m]	Increment table pointer low byte first and read table (last page) to TBLH and Data Memory
Description	Increment table pointer low byte, TBLP, first and then the low byte of the program code (last page) addressed by the table pointer (TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.
Operation	[m] ← program code (low byte) TBLH ← program code (high byte)
Affected flag(s)	None

XOR A,[m]	Logical XOR Data Memory to ACC
Description	Data in the Accumulator and the specified Data Memory perform a bitwise logical XOR operation. The result is stored in the Accumulator.
Operation	ACC ← ACC ″XOR″ [m]
Affected flag(s)	Z

XORM A,[m]	Logical XOR ACC to Data Memory
Description	Data in the specified Data Memory and the Accumulator perform a bitwise logical XOR operation. The result is stored in the Data Memory.
Operation	[m] ← ACC ″XOR″ [m]
Affected flag(s)	Z

XOR A,x	Logical XOR immediate data to ACC
Description	Data in the Accumulator and the specified immediate data perform a bitwise logical XOR operation. The result is stored in the Accumulator.
Operation	ACC ← ACC ″XOR″ x
Affected flag(s)	Z

Extended Instruction Definition

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

LADC A,[m]	Add Data Memory to ACC with Carry
Description	The contents of the specified Data Memory, Accumulator and the carry flag are added. The result is stored in the Accumulator.
Operation	ACC ← ACC + [m] + C
Affected flag(s)	OV, Z, AC, C, SC

LADCM A,[m]	Add ACC to Data Memory with Carry
Description	The contents of the specified Data Memory, Accumulator and the carry flag are added. The result is stored in the specified Data Memory.
Operation	[m] ← ACC + [m] + C
Affected flag(s)	OV, Z, AC, C, SC

LADD A,[m]	Add Data Memory to ACC
Description	The contents of the specified Data Memory and the Accumulator are added. The result is stored in the Accumulator.
Operation	ACC ← ACC + [m]
Affected flag(s)	OV, Z, AC, C, SC

LADDM A,[m]	Add ACC to Data Memory
Description	The contents of the specified Data Memory and the Accumulator are added. The result is stored in the specified Data Memory.
Operation	[m] ← ACC + [m]
Affected flag(s)	OV, Z, AC, C, SC

LAND A,[m]	Logical AND Data Memory to ACC
Description	Data in the Accumulator and the specified Data Memory perform a bitwise logical AND operation. The result is stored in the Accumulator.
Operation	ACC ← ACC ″AND″ [m]
Affected flag(s)	Z

LANDM A,[m]	Logical AND ACC to Data Memory
Description	Data in the specified Data Memory and the Accumulator perform a bitwise logical AND operation. The result is stored in the Data Memory.
Operation	[m] ← ACC ″AND″ [m]
Affected flag(s)	Z

LCLR [m]	Clear Data Memory
Description	Each bit of the specified Data Memory is cleared to 0.
Operation	[m] ← 00H
Affected flag(s)	None

LCLR [m].i	Clear bit of Data Memory
Description	Bit i of the specified Data Memory is cleared to 0.
Operation	[m].i ← 0
Affected flag(s)	None

LCPL [m]	Complement Data Memory
Description	Each bit of the specified Data Memory is logically complemented (1′s complement). Bits which previously contained a 1 are changed to 0 and vice versa.
Operation	[m] ← $\overline{[m]}$
Affected flag(s)	Z

LCPLA [m]	Complement Data Memory with result in ACC
Description	Each bit of the specified Data Memory is logically complemented (1′s complement). Bits which previously contained a 1 are changed to 0 and vice versa. The complemented result is stored in the Accumulator and the contents of the Data Memory remain unchanged.
Operation	ACC ← $\overline{[m]}$
Affected flag(s)	Z

LDAA [m]	Decimal-Adjust ACC for addition with result in Data Memory
Description	Convert the contents of the Accumulator value to a BCD (Binary Coded Decimal) value resulting from the previous addition of two BCD variables. If the low nibble is greater than 9 or if AC flag is set, then a value of 6 will be added to the low nibble. Otherwise the low nibble remains unchanged. If the high nibble is greater than 9 or if the C flag is set, then a value of 6 will be added to the high nibble. Essentially, the decimal conversion is performed by adding 00H, 06H, 60H or 66H depending on the Accumulator and flag conditions. Only the C flag may be affected by this instruction which indicates that if the original BCD sum is greater than 100, it allows multiple precision decimal addition.
Operation	[m] ← ACC + 00H or [m] ← ACC + 06H or [m] ← ACC + 60H or [m] ← ACC + 66H
Affected flag(s)	C

LDEC [m]	Decrement Data Memory
Description	Data in the specified Data Memory is decremented by 1.
Operation	[m] ← [m] − 1
Affected flag(s)	Z

LDECA [m]	Decrement Data Memory with result in ACC
Description	Data in the specified Data Memory is decremented by 1. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.
Operation	ACC ← [m] − 1
Affected flag(s)	Z

LINC [m]	Increment Data Memory
Description	Data in the specified Data Memory is incremented by 1.
Operation	[m] ← [m] + 1
Affected flag(s)	Z

LINCA [m]	Increment Data Memory with result in ACC
Description	Data in the specified Data Memory is incremented by 1. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.
Operation	ACC ← [m] + 1
Affected flag(s)	Z

LMOV A,[m]	Move Data Memory to ACC
Description	The contents of the specified Data Memory are copied to the Accumulator.
Operation	ACC ← [m]
Affected flag(s)	None

LMOV [m],A	Move ACC to Data Memory
Description	The contents of the Accumulator are copied to the specified Data Memory.
Operation	[m] ← ACC
Affected flag(s)	None

LOR A,[m]	Logical OR Data Memory to ACC
Description	Data in the Accumulator and the specified Data Memory perform a bitwise logical OR operation. The result is stored in the Accumulator.
Operation	ACC ← ACC ″OR″ [m]
Affected flag(s)	Z

LORM A,[m]	Logical OR ACC to Data Memory
Description	Data in the specified Data Memory and the Accumulator perform a bitwise logical OR operation. The result is stored in the Data Memory.
Operation	[m] ← ACC ″OR″ [m]
Affected flag(s)	Z

LRL [m]	Rotate Data Memory left
Description	The contents of the specified Data Memory are rotated left by 1 bit with bit 7 rotated into bit 0.
Operation	[m].(i+1) ← [m].i; (i=0~6) [m].0 ← [m].7
Affected flag(s)	None

LRLA [m]	Rotate Data Memory left with result in ACC
Description	The contents of the specified Data Memory are rotated left by 1 bit with bit 7 rotated into bit 0. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.
Operation	ACC.(i+1) ← [m].i; (i=0~6) ACC.0 ← [m].7
Affected flag(s)	None

LRLC [m]	Rotate Data Memory left through Carry
Description	The contents of the specified Data Memory and the carry flag are rotated left by 1 bit. Bit 7 replaces the Carry bit and the original carry flag is rotated into bit 0.
Operation	[m].(i+1) ← [m].i; (i=0~6) [m].0 ← C C ← [m].7
Affected flag(s)	C

LRLCA [m]	Rotate Data Memory left through Carry with result in ACC
Description	Data in the specified Data Memory and the carry flag are rotated left by 1 bit. Bit 7 replaces the Carry bit and the original carry flag is rotated into the bit 0. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.
Operation	ACC.(i+1) ← [m].i; (i=0~6) ACC.0 ← C C ← [m].7
Affected flag(s)	C

LRR [m]	Rotate Data Memory right
Description	The contents of the specified Data Memory are rotated right by 1 bit with bit 0 rotated into bit 7.
Operation	[m].i ← [m].(i+1); (i=0~6) [m].7 ← [m].0
Affected flag(s)	None

LRRA [m]	Rotate Data Memory right with result in ACC
Description	Data in the specified Data Memory is rotated right by 1 bit with bit 0 rotated into bit 7. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.
Operation	ACC.i ← [m].(i+1); (i=0~6) ACC.7 ← [m].0
Affected flag(s)	None

LRRC [m]	Rotate Data Memory right through Carry
Description	The contents of the specified Data Memory and the carry flag are rotated right by 1 bit. Bit 0 replaces the Carry bit and the original carry flag is rotated into bit 7.
Operation	[m].i ← [m].(i+1); (i=0~6) [m].7 ← C C ← [m].0
Affected flag(s)	C

LRRCA [m]	Rotate Data Memory right through Carry with result in ACC
Description	Data in the specified Data Memory and the carry flag are rotated right by 1 bit. Bit 0 replaces the Carry bit and the original carry flag is rotated into bit 7. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.
Operation	ACC.i ← [m].(i+1); (i=0~6) ACC.7 ← C C ← [m].0
Affected flag(s)	C

LSBC A,[m]	Subtract Data Memory from ACC with Carry
Description	The contents of the specified Data Memory and the complement of the carry flag are subtracted from the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.
Operation	$\text{ACC} \leftarrow \text{ACC} - [\text{m}] - \overline{\text{C}}$
Affected flag(s)	OV, Z, AC, C, SC, CZ

LSBCM A,[m]	Subtract Data Memory from ACC with Carry and result in Data Memory
Description	The contents of the specified Data Memory and the complement of the carry flag are subtracted from the Accumulator. The result is stored in the Data Memory. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.
Operation	$[\text{m}] \leftarrow \text{ACC} - [\text{m}] - \overline{\text{C}}$
Affected flag(s)	OV, Z, AC, C, SC, CZ

LSDZ [m]	Skip if decrement Data Memory is 0
Description	The contents of the specified Data Memory are first decremented by 1. If the result is 0 the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a 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

LSDZA [m]	Skip if decrement Data Memory is zero with result in ACC
Description	The contents of the specified Data Memory are first decremented by 1. If the result is 0, the following instruction is skipped. The result is stored in the Accumulator but the specified Data Memory contents remain unchanged. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a 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

LSET [m]	Set Data Memory
Description	Each bit of the specified Data Memory is set to 1.
Operation	[m] ← FFH
Affected flag(s)	None

LSET [m].i	Set bit of Data Memory
Description	Bit i of the specified Data Memory is set to 1.
Operation	[m].i ← 1
Affected flag(s)	None

LSIZ [m]	Skip if increment Data Memory is 0
Description	The contents of the specified Data Memory are first incremented by 1. If the result is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a 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

LSIZA [m]	Skip if increment Data Memory is zero with result in ACC
Description	The contents of the specified Data Memory are first incremented by 1. If the result is 0, the following instruction is skipped. The result is stored in the Accumulator but the specified Data Memory contents remain unchanged. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a 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

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

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

LSUB A,[m]	Subtract Data Memory from ACC
Description	The specified Data Memory is subtracted from the contents of the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.
Operation	ACC ← ACC − [m]
Affected flag(s)	OV, Z, AC, C, SC, CZ

LSUBM A,[m]	Subtract Data Memory from ACC with result in Data Memory
Description	The specified Data Memory is subtracted from the contents of the Accumulator. The result is stored in the Data Memory. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.
Operation	[m] ← ACC − [m]
Affected flag(s)	OV, Z, AC, C, SC, CZ

LSWAP [m]	Swap nibbles of Data Memory
Description	The low-order and high-order nibbles of the specified Data Memory are interchanged.
Operation	[m].3~[m].0 ↔ [m].7~[m].4
Affected flag(s)	None

LSWAPA [m]	Swap nibbles of Data Memory with result in ACC
Description	The low-order and high-order nibbles of the specified Data Memory are interchanged. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.
Operation	ACC.3~ACC.0 ← [m].7~[m].4 ACC.7~ACC.4 ← [m].3~[m].0
Affected flag(s)	None

LSZ [m]	Skip if Data Memory is 0
Description	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

LSZA [m]	Skip if Data Memory is 0 with data movement to ACC
Description	The contents of the specified Data Memory are copied to the Accumulator. If the value is zero, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.
Operation	ACC ← [m] Skip if [m]=0
Affected flag(s)	None

LSZ [m].i	Skip if bit i of Data Memory is 0
Description	If bit i of the specified Data Memory is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a 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

LTABRD [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) TBLH ← program code (high byte)
Affected flag(s)	None

LTABRDL [m]	Read table (last page) to TBLH and Data Memory
Description	The low byte of the program code (last page) addressed by the table pointer (TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.
Operation	[m] ← program code (low byte) TBLH ← program code (high byte)
Affected flag(s)	None

LITABRD [m]	Increment table pointer low byte first and read table to TBLH and Data Memory
Description	Increment table pointer low byte, TBLP, first and then the program code addressed by the table pointer (TBHP and TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.
Operation	[m] ← program code (low byte) TBLH ← program code (high byte)
Affected flag(s)	None

LITABRDL [m]	Increment table pointer low byte first and read table (last page) to TBLH and Data Memory
Description	Increment table pointer low byte, TBLP, first and then the low byte of the program code (last page) addressed by the table pointer (TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.
Operation	[m] ← program code (low byte) TBLH ← program code (high byte)
Affected flag(s)	None

LXOR A,[m]	Logical XOR Data Memory to ACC
Description	Data in the Accumulator and the specified Data Memory perform a bitwise logical XOR operation. The result is stored in the Accumulator.
Operation	ACC ← ACC ″XOR″ [m]
Affected flag(s)	Z

LXORM A,[m]	Logical XOR ACC to Data Memory
Description	Data in the specified Data Memory and the Accumulator perform a bitwise logical XOR operation. The result is stored in the Data Memory.
Operation	[m] ← ACC ″XOR″ [m]
Affected flag(s)	Z

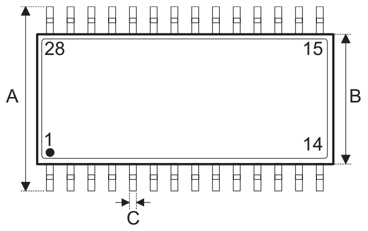
Package Information

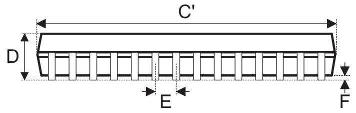
Note that the package information provided here is for consultation purposes only. As this information may be updated at regular intervals users are reminded to consult the Holtek website for the latest version of the Package/Carton Information.

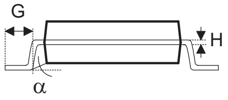
Additional supplementary information with regard to packaging is listed below. Click on the relevant section to be transferred to the relevant website page.

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

28-pin SOP (300mil) Outline Dimensions



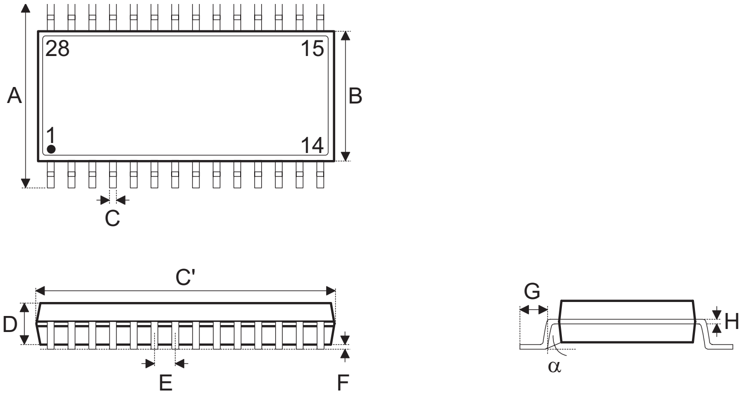




Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	—	0.406 BSC	—
B	—	0.295 BSC	—
C	0.012	—	0.020
C’	—	0.705 BSC	—
D	—	—	0.104
E	—	0.050 BSC	—
F	0.004	—	0.012
G	0.016	—	0.050
H	0.008	—	0.013
α	0°	—	8°

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	—	10.30 BSC	—
B	—	7.50 BSC	—
C	0.31	—	0.51
C’	—	17.90 BSC	—
D	—	—	2.65
E	—	1.27 BSC	—
F	0.10	—	0.30
G	0.40	—	1.27
H	0.20	—	0.33
α	0°	—	8°

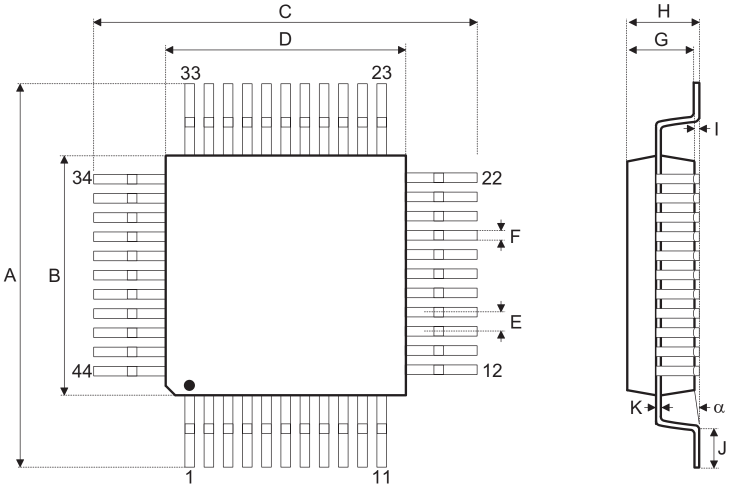
28-pin SSOP (150mil) Outline Dimensions



Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	—	0.236 BSC	—
B	—	0.154 BSC	—
C	0.008	—	0.012
C’	—	0.390 BSC	—
D	—	—	0.069
E	—	0.025 BSC	—
F	0.004	—	0.0098
G	0.016	—	0.050
H	0.004	—	0.010
α	0°	—	8°

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	—	6.000 BSC	—
B	—	3.900 BSC	—
C	0.20	—	0.30
C’	—	9.900 BSC	—
D	—	—	1.75
E	—	0.635 BSC	—
F	0.10	—	0.25
G	0.41	—	1.27
H	0.10	—	0.25
α	0°	—	8°

44-pin LQFP (10mm×10mm) (FP2.0mm) Outline Dimensions



Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	—	0.472 BSC	—
B	—	0.394 BSC	—
C	—	0.472 BSC	—
D	—	0.394 BSC	—
E	—	0.032 BSC	—
F	0.012	0.015	0.018
G	0.053	0.055	0.057
H	—	—	0.063
I	0.002	—	0.006
J	0.018	0.024	0.030
K	0.004	—	0.008
α	0°	—	7°

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	—	12.00 BSC	—
B	—	10.00 BSC	—
C	—	12.00 BSC	—
D	—	10.00 BSC	—
E	—	0.80 BSC	—
F	0.30	0.37	0.45
G	1.35	1.40	1.45
H	—	—	1.60
I	0.05	—	0.15
J	0.45	0.60	0.75
K	0.09	—	0.20
α	0°	—	7°

Copyright© 2018 by HOLTEK SEMICONDUCTOR INC.

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