Touch A/D Flash MCU with OCVP

BS87B12A-3/BS87C16A-3/BS87D20A-3

Revision: V1.20 Date: December 05, 2016

www.holtek.com

Features

CPU Features

- Operating Voltage:
  - f_{SYS}=8MHz: 2.7V~5.5V
  - f_{SYS}=12MHz: 2.7V~5.5V
  - f_{SYS}=16MHz: 4.5V~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 Type:
  - Internal High Speed RC – HIRC
  - Internal 32kHz RC – LIRC
  - External 32.768kHz Crystal – LXT for BS87C16A-3 & BS87D20A-3 only
- Fully integrated internal 8/12/16 MHz oscillator requires no external components
- Multi-mode operation: FAST, SLOW, IDLE and SLEEP
- All instructions executed in one to three instruction cycles
- Table read instructions
- 115 powerful instructions
- Up to 8-level subroutine nesting
- Bit manipulation instruction

Peripheral Features

- Program Memory: Up to 8K × 16
- Data Memory: Up to 768 × 8
- True EEPROM Memory: 64 × 8
- Up to 20 touch key functions – fully integrated without requiring external components
- Watchdog Timer function
- Up to 42 bidirectional I/O lines
- Programming I/O source current
- Up to 4SCOM & 36SSEG lines Software controlled LCD driver with 1/3 bias
- One external interrupt lines shared with I/O pins
- Multiple Timer Modules for time measure, input capture, compare match output, PWM output function or single pulse output function
- Dual Time-Base functions for generation of fixed time interrupt signals
- Multi-channel 12-bit resolution A/D converter
- Over Current/Voltage Protection – OCVP
- Serial Interfaces Module – SIM for SPI or I^2C
- Fully-duplex Universal Asynchronous Receiver and Transmitter Interface – UART
- Low voltage reset function
- Low voltage detect function
- Wide range of package types

General Description

The series of devices is the Flash Memory type 8-bit high performance RISC architecture microcontroller with fully integrated touch key functions. With all touch key functions provided internally and with the convenience of Flash Memory multi-programming features, this series of devices has all the features to offer designers a reliable and easy means of implementing touch switches within their product applications.

The touch key functions are fully integrated thus completely eliminating the need for external components. In addition to the Flash Program Memory, other memory includes an area of RAM Data Memory as well as an area of true EEPROM Memory for storage of non-volatile data such as serial numbers, calibration data etc. Analog features include a multi-channel 12-bit A/D converter and a Over Current/Voltage Protection module. Protective features such as an internal Watchdog Timer, Low Voltage Reset and Low Voltage Detector functions coupled with excellent noise immunity and ESD protection ensure that reliable operation is maintained in hostile electrical environments.

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

A UART module is contained within this series of devices. This interface can support applications such as data communication networks between microcontrollers, low-cost data links between PCs and peripheral devices, portable and battery operated device communication, etc.

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

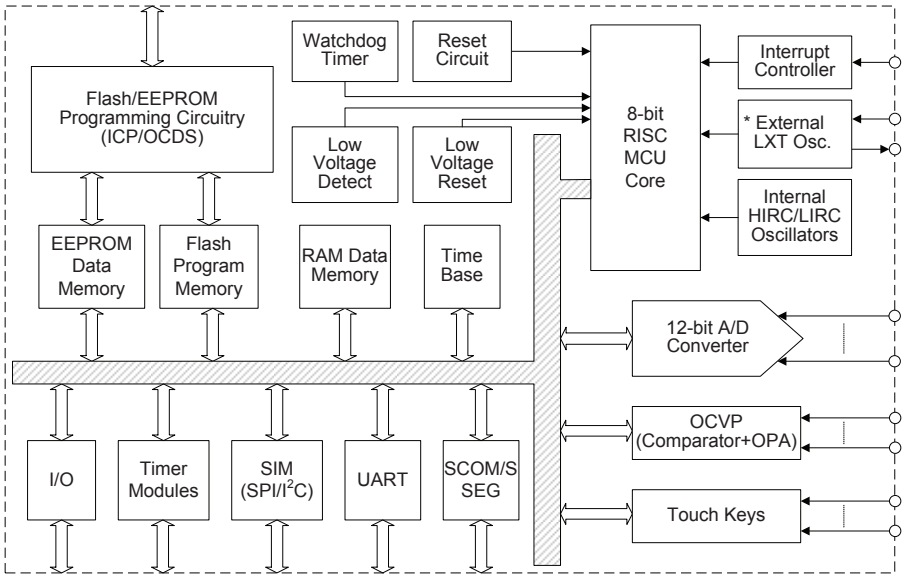
Most features are common to all devices. The main features distinguishing them are Memory capacity, I/O count, Time Module number, SSEG lines and touch key number. The following table summarises the main features of each device.

Part No.	Program Memory	Data Memory	Data EEPROM	I/O	External Interrupt	A/D	Timer Module
BS87B12A-3	3k × 16	384 × 8	64 × 8	22	1	12-bit × 8	10-bit CTM × 1 10-bit PTM × 1
BS87C16A-3	4k × 16	512 × 8	64 × 8	30	1	12-bit × 8	10-bit CTM × 1 10-bit PTM × 2
BS87D20A-3	8k × 16	768 × 8	64 × 8	42	1	12-bit × 8	10-bit CTM × 2 10-bit PTM × 2

Part No.	Time Base	SCOM/ SSEG	Touch Key	UART	SIM	Stacks	Package
BS87B12A-3	2	4/16	12	1	√	6	20NSOP 24SOP
BS87C16A-3	2	4/20	16	1	√	6	24/28SOP 44LQFP
BS87D20A-3	2	4/36	20	1	√	8	28SOP 44LQFP

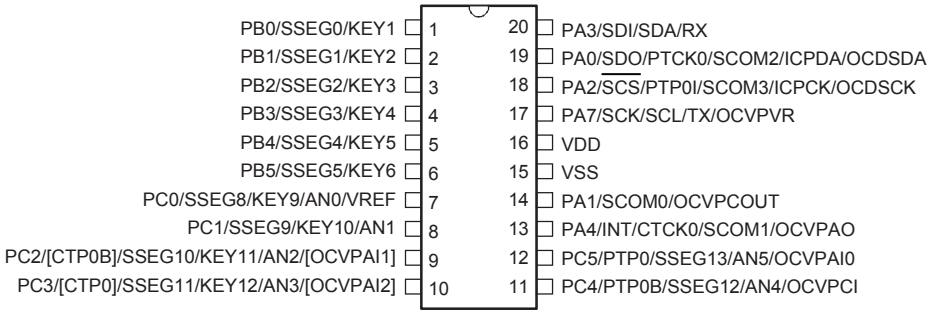
Note: As devices exist in more than one package format, the table reflects the situation for the package with the most pins.

Block Diagram

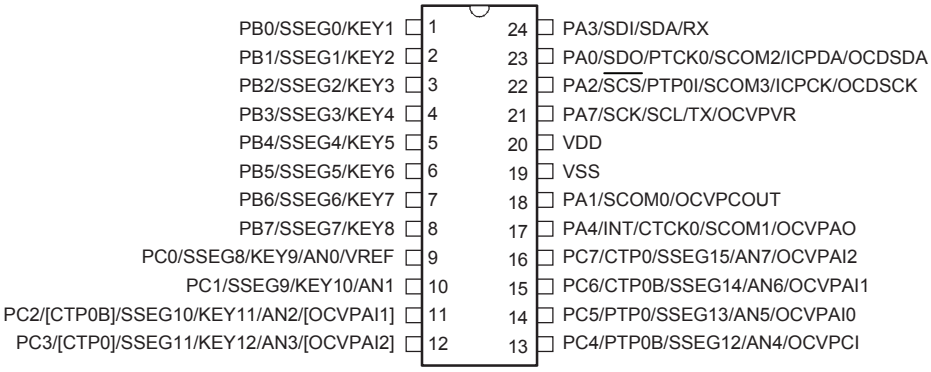


*: The LXT oscillator is only available for the BS87C16A-3 and BS87D20A-3 devices.

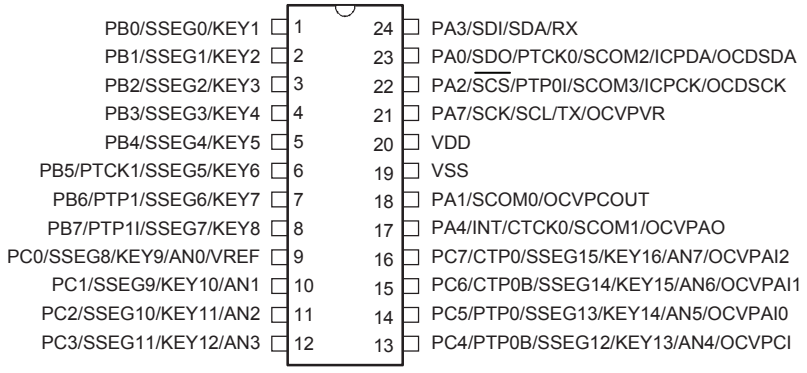
Pin Assignment



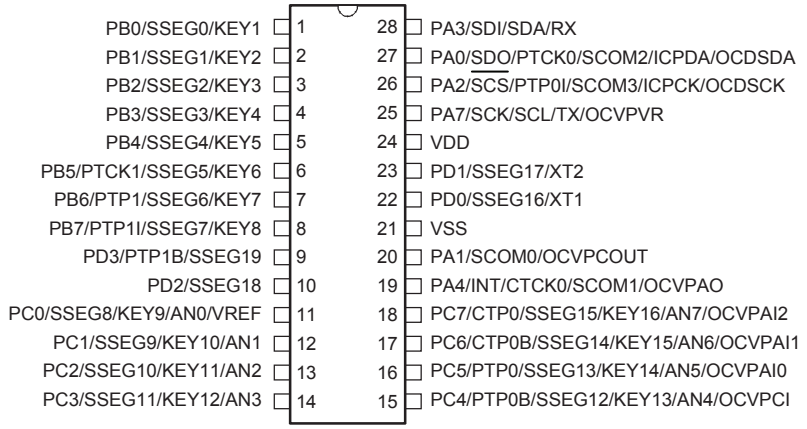
BS87B12A-3/BS87BV12A
20 NSOP-A



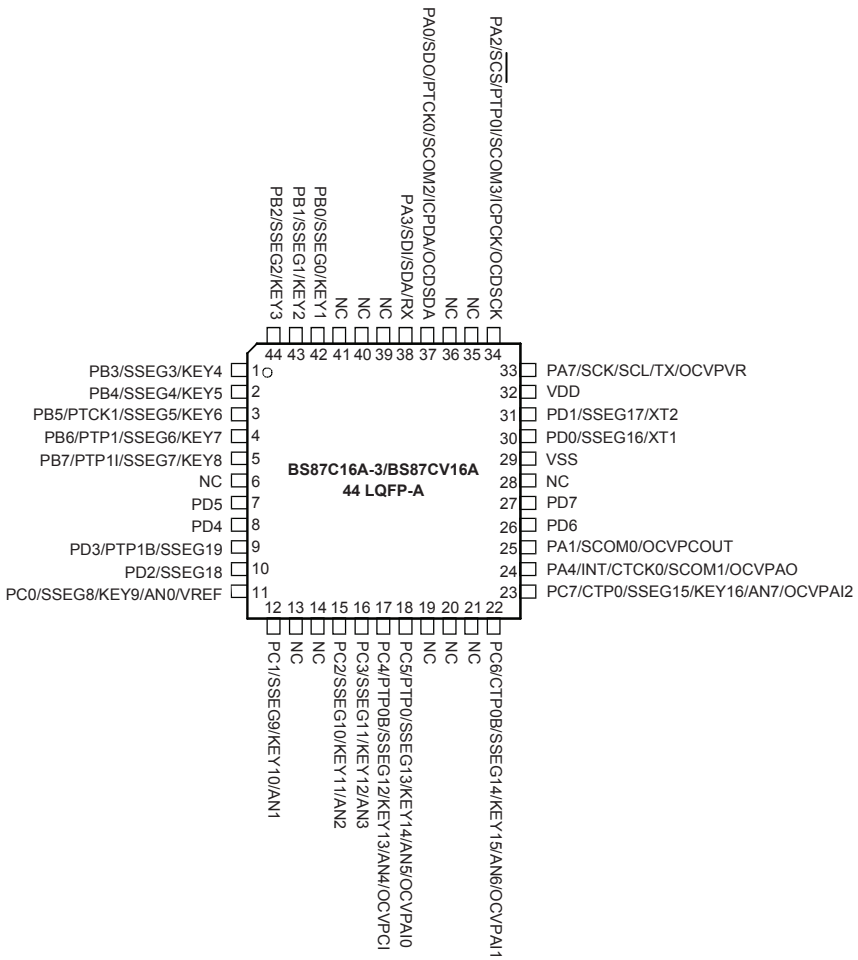
BS87B12A-3/BS87BV12A
24 SOP-A

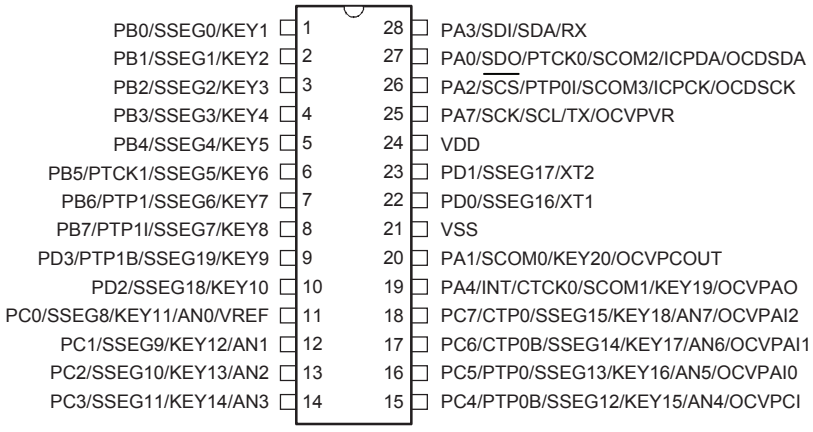


BS87C16A-3/BS87CV16A
24 SOP-A

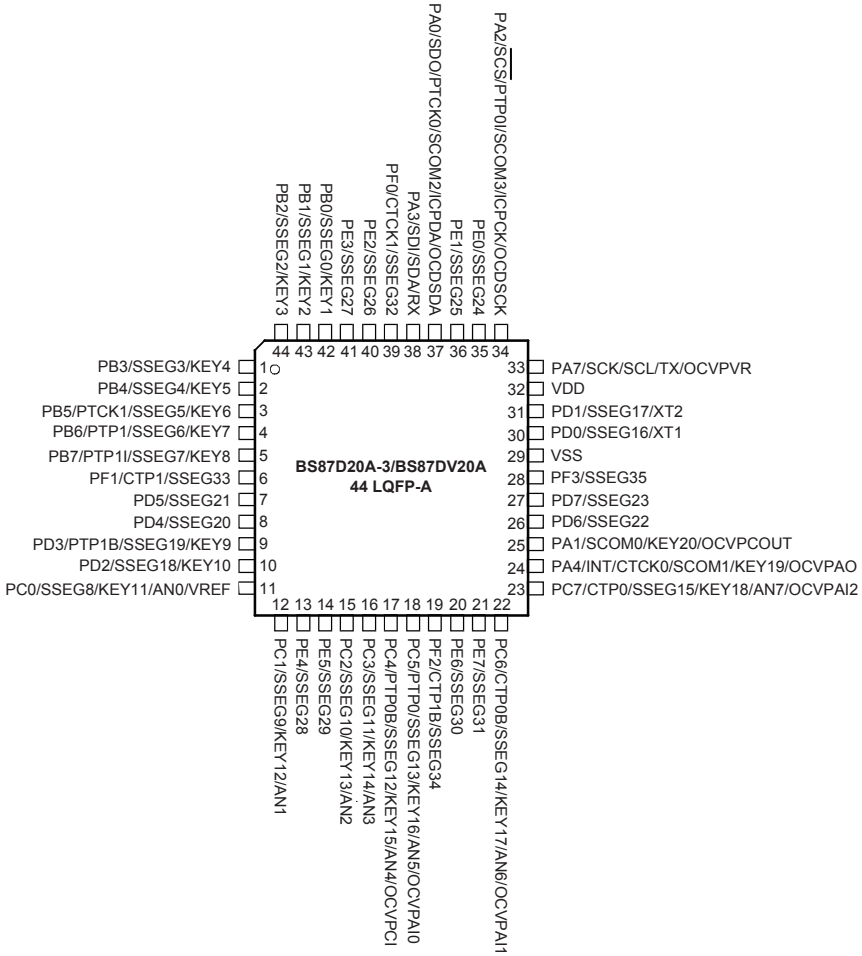


BS87C16A-3/BS87CV16A
28 SOP-A





BS87D20A-3/BS87DV20A
28 SOP-A



Note: 1. If the pin-shared pin functions have multiple outputs simultaneously, its pin name at the right side of the "/" sign can be used for higher priority.

2. The OCDSDA and OCDSCK pins are the OCDS dedicated pins and only available for the BS87BV12A/BS87CV16A/BS87DV20A device which is the OCDS EV chip for the BS87B12A-3/BS87C16A-3/BS87D20A-3 device.

Pin Descriptions

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

As the Pin Description table shows the situation for the package with the most pins, not all pins in the table will be available on small package sizes.

BS87B12A-3

Pad Name	Function	OPT	I/T	O/T	Description
PA0/SDO/PTCK0/ SCOM2/ICPDA/ OCDSDA	PA0	PAWU PAPU	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	SDO	SIMC0	—	CMOS	SPI data output
	PTCK0	PTM0C0	ST	—	PTM0 clock input
	SCOM2	SLCDC0	—	SCOM	Software LCD COM output
	ICPDA	—	ST	CMOS	ICP Data/Address pin
	OCDSDA	—	ST	CMOS	OCDS Data/Address pin, for EV chip only.
PA1/SCOM0/ OCVPCOUT	PA1	PAWU PAPU	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	SCOM0	SLCDC0	—	SCOM	Software LCD COM output
	OCVPCOUT	OCVPC3	—	CMOS	OCVP comparator output
PA2/$\overline{SCS}$/PTP0I/ SCOM3/ICPCK/ OCDSCK	PA2	PAWU PAPU	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	$\overline{SCS}$	SIMC0	ST	CMOS	SPI slave select
	PTP0I	PTM0C0 PTM0C1	ST	—	PTM0 capture input
	SCOM3	SLCDC0	—	SCOM	Software LCD COM output
	ICPCK	—	ST	CMOS	ICP Clock pin
	OCDSCK	—	ST	—	OCDS Clock pin, for EV chip only.
PA3/SDI/SDA/RX	PA3	PAWU PAPU	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	SDI	SIMC0	ST	—	SPI data input
	SDA	SIMC0	ST	NMOS	I^2C data line
	RX	UCR1	ST	—	UART RX serial data input
PA4/INT/CTCK0/ SCOM1/ OCVPAO	PA4	PAWU PAPU	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	INT	INTEG INTC0	ST	—	External Interrupt
	CTCK0	CTM0C0	ST	—	CTM0 clock input
	SCOM1	SLCDC0	—	SCOM	Software LCD COM output
	OCVPAO	OCVPC3	—	AN	OCVP OPA output
PA7/SCK/SCL/TX/ OCVPVR	PA7	PAWU PAPU	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	SCK	SIMC0	ST	CMOS	SPI serial clock
	SCL	SIMC0	ST	NMOS	I^2C clock line
	TX	UCR1	—	CMOS	UART TX serial data output
	OCVPVR	OCVPC1	AN	—	OCVP DAC reference voltage input

Pad Name	Function	OPT	I/T	O/T	Description
PB0/SSEG0/KEY1	PB0	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SSEG0	SLCDC1	—	CMOS	Software LCD SEG output
	KEY1	TKM0C1	NSI	—	Touch key input
PB1/SSEG1/KEY2	PB1	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SSEG1	SLCDC1	—	CMOS	Software LCD SEG output
	KEY2	TKM0C1	NSI	—	Touch key input
PB2/SSEG2/KEY3	PB2	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SSEG2	SLCDC1	—	CMOS	Software LCD SEG output
	KEY3	TKM0C1	NSI	—	Touch key input
PB3/SSEG3/KEY4	PB3	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SSEG3	SLCDC1	—	CMOS	Software LCD SEG output
	KEY4	TKM0C1	NSI	—	Touch key input
PB4/SSEG4/KEY5	PB4	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SSEG4	SLCDC1	—	CMOS	Software LCD SEG output
	KEY5	TKM1C1	NSI	—	Touch key input
PB5/SSEG5/KEY6	PB5	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SSEG5	SLCDC1	—	CMOS	Software LCD SEG output
	KEY6	TKM1C1	NSI	—	Touch key input
PB6/SSEG6/KEY7	PB6	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SSEG6	SLCDC1	—	CMOS	Software LCD SEG output
	KEY7	TKM1C1	NSI	—	Touch key input
PB7/SSEG7/KEY8	PB7	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SSEG7	SLCDC1	—	CMOS	Software LCD SEG output
	KEY8	TKM1C1	NSI	—	Touch key input
PC0/SSEG8/KEY9/ AN0/VREF	PC0	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SSEG8	SLCDC2	—	CMOS	Software LCD SEG output
	KEY9	TKM2C1	NSI	—	Touch key input
	AN0	ACERL	AN	—	A/D Converter analog input
	VREF	TMPC0	AN	—	A/D Converter reference voltage input
PC1/SSEG9/KEY10/ AN1	PC1	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SSEG9	SLCDC2	—	CMOS	Software LCD SEG output
	KEY10	TKM2C1	NSI	—	Touch key input
	AN1	ACERL	AN	—	A/D Converter analog input
PC2/[CTP0B]/ SSEG10/KEY11/ AN2/[OCVPAI1]	PC2	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	[CTP0B]	TMPC0 IFS	—	CMOS	CTM0 inverted output
	SSEG10	SLCDC2	—	CMOS	Software LCD SEG output
	KEY11	TKM2C1	NSI	—	Touch key input
	AN2	ACERL	AN	—	A/D Converter analog input
	[OCVPAI1]	OCVPC3 IFS	AN	—	OCVP OCP input signal 1
PC3/[CTP0]/SSEG11/ KEY12/AN3/ [OCVPAI2]	PC3	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	[CTP0]	TMPC0 IFS	—	CMOS	CTM0 non-inverted output
	SSEG11	SLCDC2	—	CMOS	Software LCD SEG output
	KEY12	TKM2C1	NSI	—	Touch key input
	AN3	ACERL	AN	—	A/D Converter analog input
	[OCVPAI2]	OCVPC3 IFS	AN	—	OCVP OCP input signal 2

Pad Name	Function	OPT	I/T	O/T	Description
PC4/PTP0B/SSEG12/AN4/OCVPCI	PC4	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	PTP0B	TMPC0	—	CMOS	PTM0 inverted output
	SSEG12	SLCDC2	—	CMOS	Software LCD SEG output
	AN4	ACERL	AN	—	A/D Converter analog input
	OCVPCI	OCVPC3	AN	—	OCVP OCP input signal / Comparator non-inverting input signal
PC5/PTP0/SSEG13/AN5/OCVPAI0	PC5	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	PTP0	TMPC0	—	CMOS	PTM0 non-inverted output
	SSEG13	SLCDC2	—	CMOS	Software LCD SEG output
	AN5	ACERL	AN	—	A/D Converter analog input
	OCVPAI0	OCVPC3	AN	—	OCVP OCP input signal 0
PC6/CTP0B/SSEG14/AN6/OCVPAI1	PC6	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	CTP0B	TMPC0 IFS	—	CMOS	CTM0 inverted output
	SSEG14	SLCDC2	—	CMOS	Software LCD SEG output
	AN6	ACERL	AN	—	A/D Converter analog input
	OCVPAI1	OCVPC3 IFS	AN	—	OCVP OCP input signal 1
PC7/CTP0/SSEG15/AN7/OCVPAI2	PC7	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	CTP0	TMPC0 IFS	—	CMOS	CTM0 non-inverted output
	SSEG15	SLCDC2	—	CMOS	Software LCD SEG output
	AN7	ACERL	AN	—	A/D Converter analog input
	OCVPAI2	OCVPC3 IFS	AN	—	OCVP OCP input signal 2
VDD	VDD	—	PWR	—	Positive power supply
VSS	VSS	—	PWR	—	Negative power supply, ground.

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

BS87C16A-3

Pad Name	Function	OPT	I/T	O/T	Description
PA0/SDO/PTCK0/ SCOM2/ICPDA/ OCDSDA	PA0	PAWU PAPU	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	SDO	SIMC0	—	CMOS	SPI data output
	PTCK0	PTM0C0	ST	—	PTM0 clock input
	SCOM2	SLCDC0	—	SCOM	Software LCD COM output
	ICPDA	—	ST	CMOS	ICP Data/Address pin
	OCDSDA	—	ST	CMOS	OCDS Data/Address pin, for EV chip only.
PA1/SCOM0/ OCVPCOUT	PA1	PAWU PAPU	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	SCOM0	SLCDC0	—	SCOM	Software LCD COM output
	OCVPCOUT	OCVPC3	—	CMOS	OCVP comparator output
PA2/$\overline{SCS}$/PTP0I/ SCOM3/ICPCK/ OCDSCK	PA2	PAWU PAPU	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	$\overline{SCS}$	SIMC0	ST	CMOS	SPI slave select
	PTP0I	PTM0C0 PTM0C1	ST	—	PTM0 capture input
	SCOM3	SLCDC0	—	SCOM	Software LCD COM output
	ICPCK	—	ST	CMOS	ICP Clock pin
	OCDSCK	—	ST	—	OCDS Clock pin, for EV chip only.
PA3/SDI/SDA/RX	PA3	PAWU PAPU	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	SDI	SIMC0	ST	—	SPI data input
	SDA	SIMC0	ST	NMOS	I^2C data line
	RX	UCR1	ST	—	UART RX serial data input
PA4/INT/CTCK0/ SCOM1/OCVPAO	PA4	PAWU PAPU	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	INT	INTEG INTC0	ST	—	External Interrupt
	CTCK0	CTM0C0	ST	—	CTM0 clock input
	SCOM1	SLCDC0	—	SCOM	Software LCD COM output
	OCVPAO	OCVPC3	—	AN	OCVP OPA output
PA7/SCK/SCL/TX/ OCVPVR	PA7	PAWU PAPU	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	SCK	SIMC0	ST	CMOS	SPI serial clock
	SCL	SIMC0	ST	NMOS	I^2C clock line
	TX	UCR1	—	CMOS	UART TX serial data output
	OCVPVR	OCVPC1	AN	—	OCVP DAC reference voltage input
PB0/SSEG0/KEY1	PB0	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SSEG0	SLCDC1	—	CMOS	Software LCD SEG output
	KEY1	TKM0C1	NSI	—	Touch key input
PB1/SSEG1/KEY2	PB1	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SSEG1	SLCDC1	—	CMOS	Software LCD SEG output
	KEY2	TKM0C1	NSI	—	Touch key input
PB2/SSEG2/KEY3	PB2	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SSEG2	SLCDC1	—	CMOS	Software LCD SEG output
	KEY3	TKM0C1	NSI	—	Touch key input
PB3/SSEG3/KEY4	PB3	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SSEG3	SLCDC1	—	CMOS	Software LCD SEG output
	KEY4	TKM0C1	NSI	—	Touch key input

Pad Name	Function	OPT	I/T	O/T	Description
PB4/SSEG4/KEY5	PB4	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SSEG4	SLCDC1	—	CMOS	Software LCD SEG output
	KEY5	TKM1C1	NSI	—	Touch key input
PB5/PTCK1/SSEG5/ KEY6	PB5	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	PTCK1	PTM1C0	ST	—	PTM1 clock input
	SSEG5	SLCDC1	—	CMOS	Software LCD SEG output
	KEY6	TKM1C1	NSI	—	Touch key input
PB6/PTP1/SSEG6/ KEY7	PB6	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	PTP1	TMPC0	—	CMOS	PTM1 non-inverted output
	SSEG6	SLCDC1	—	CMOS	Software LCD SEG output
	KEY7	TKM1C1	NSI	—	Touch key input
PB7/PTP1I/SSEG7/ KEY8	PB7	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	PTP1I	PTM1C0 PTM1C1	ST	—	PTM1 capture input
	SSEG7	SLCDC1	—	CMOS	Software LCD SEG output
	KEY8	TKM1C1	NSI	—	Touch key input
PC0/SSEG8/KEY9/ AN0/VREF	PC0	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SSEG8	SLCDC2	—	CMOS	Software LCD SEG output
	KEY9	TKM2C1	NSI	—	Touch key input
	AN0	ACERL	AN	—	A/D Converter analog input
	VREF	TMPC0	AN	—	A/D Converter reference voltage input
PC1/SSEG9/KEY10/ AN1	PC1	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SSEG9	SLCDC2	—	CMOS	Software LCD SEG output
	KEY10	TKM2C1	NSI	—	Touch key input
	AN1	ACERL	AN	—	A/D Converter analog input
PC2/SSEG10/KEY11/ AN2	PC2	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SSEG10	SLCDC2	—	CMOS	Software LCD SEG output
	KEY11	TKM2C1	NSI	—	Touch key input
	AN2	ACERL	AN	—	A/D Converter analog input
PC3/SSEG11/KEY12/ AN3	PC3	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SSEG11	SLCDC2	—	CMOS	Software LCD SEG output
	KEY12	TKM2C1	NSI	—	Touch key input
	AN3	ACERL	AN	—	A/D Converter analog input
PC4/PTP0B/SSEG12/ KEY13/AN4/OCVPCI	PC4	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	PTP0B	TMPC0	—	CMOS	PTM0 inverted output
	SSEG12	SLCDC2	—	CMOS	Software LCD SEG output
	KEY13	TKM3C1	NSI	—	Touch key input
	AN4	ACERL	AN	—	A/D Converter analog input
	OCVPCI	OCVPC3	AN	—	OCVP OCP input signal / Comparator non-inverting input signal
PC5/PTP0/SSEG13/ KEY14/AN5/OCVPAI0	PC5	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	PTP0	TMPC0	—	CMOS	PTM0 non-inverted output
	SSEG13	SLCDC2	—	CMOS	Software LCD SEG output
	KEY14	TKM3C1	NSI	—	Touch key input
	AN5	ACERL	AN	—	A/D Converter analog input
	OCVPAI0	OCVPC3	AN	—	OCVP OCP input signal 0

Pad Name	Function	OPT	I/T	O/T	Description
PC6/CTP0B/SSEG14/ KEY15/AN6/OCVPAI1	PC6	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	CTP0B	TMPC0	—	CMOS	CTM0 inverted output
	SSEG14	SLCDC2	—	CMOS	Software LCD SEG output
	KEY15	TKM3C1	NSI	—	Touch key input
	AN6	ACERL	AN	—	A/D Converter analog input
	OCVPAI1	OCVPC3	AN	—	OCVP OCP input signal 1
PC7/CTP0/SSEG15/ KEY16/AN7/OCVPAI2	PC7	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	CTP0	TMPC0	—	CMOS	CTM0 non-inverted output
	SSEG15	SLCDC2	—	CMOS	Software LCD SEG output
	KEY16	TKM3C1	NSI	—	Touch key input
	AN7	ACERL	AN	—	A/D Converter analog input
	OCVPAI2	OCVPC3	AN	—	OCVP OCP input signal 2
PD0/SEG16/XT1	PD0	PDPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SSEG16	SLCDC3	—	CMOS	Software LCD SEG output
	XT1	CO	LXT	—	LXT pin
PD1/SEG17/XT2	PD1	PDPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SSEG17	SLCDC3	—	CMOS	Software LCD SEG output
	XT2	CO	—	LXT	LXT pin
PD2/SEG18	PD2	PDPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SSEG18	SLCDC3	—	CMOS	Software LCD SEG output
PD3/PTP1B/SEG19	PD3	PDPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	PTP1B	TMPC0	—	CMOS	PTM1 inverted output
	SSEG19	SLCDC3	—	CMOS	Software LCD SEG output
PD4~PD7	PD4~PD7	PDPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
VDD	VDD	—	PWR	—	Positive power supply
VSS	VSS	—	PWR	—	Negative power supply, ground.

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

BS87D20A-3

Pad Name	Function	OPT	I/T	O/T	Description
PA0/SDO/PTCK0/ SCOM2/ICPDA/ OCDSDA	PA0	PAWU PAPU	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	SDO	SIMC0	—	CMOS	SPI data output
	PTCK0	PTM0C0	ST	—	PTM0 clock input
	SCOM2	SLCDC0	—	SCOM	Software LCD COM output
	ICPDA	—	ST	CMOS	ICP Data/Address pin
	OCDSDA	—	ST	CMOS	OCDS Data/Address pin, for EV chip only.
PA1/SCOM0/KEY20/ OCVPCOUT	PA1	PAWU PAPU	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	SCOM0	SLCDC0	—	SCOM	Software LCD COM output
	KEY20	TKM4C1	NSI	—	Touch key input
	OCVPCOUT	OCVPC3	—	CMOS	OCVP comparator output
PA2/$\overline{SCS}$/PTP0I/ SCOM3/ICPCK/ OCDSCK	PA2	PAWU PAPU	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	$\overline{SCS}$	SIMC0	ST	CMOS	SPI slave select
	PTP0I	PTM0C0 PTM0C1	ST	—	PTM0 capture input
	SCOM3	SLCDC0	—	SCOM	Software LCD COM output
	ICPCK	—	ST	CMOS	ICP Clock pin
	OCDSCK	—	ST	—	OCDS Clock pin, for EV chip only.
PA3/SDI/SDA/RX	PA3	PAWU PAPU	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	SDI	SIMC0	ST	—	SPI data input
	SDA	SIMC0	ST	NMOS	I^2C data line
	RX	UCR1	ST	—	UART RX serial data input
PA4/INT/CTCK0/ SCOM1/KEY19 OCVPAO	PA4	PAWU PAPU	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	INT	INTEG INTC0	ST	—	External Interrupt
	CTCK0	CTM0C0	ST	—	CTM0 clock input
	SCOM1	SLCDC0	—	SCOM	Software LCD COM output
	KEY19	TKM4C1	NSI	—	Touch key input
	OCVPAO	OCVPC3	—	AN	OCVP OPA output
PA7/SCK/SCL/TX/ OCVPVR	PA7	PAWU PAPU	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	SCK	SIMC0	ST	CMOS	SPI serial clock
	SCL	SIMC0	ST	NMOS	I^2C clock line
	TX	UCR1	—	CMOS	UART TX serial data output
	OCVPVR	OCVPC1	AN	—	OCVP DAC reference voltage input
PB0/SSEG0/KEY1	PB0	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SSEG0	SLCDC1	—	CMOS	Software LCD SEG output
	KEY1	TKM0C1	NSI	—	Touch key input
PB1/SSEG1/KEY2	PB1	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SSEG1	SLCDC1	—	CMOS	Software LCD SEG output
	KEY2	TKM0C1	NSI	—	Touch key input
PB2/SSEG2/KEY3	PB2	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SSEG2	SLCDC1	—	CMOS	Software LCD SEG output
	KEY3	TKM0C1	NSI	—	Touch key input

Pad Name	Function	OPT	I/T	O/T	Description
PB3/SSEG3/KEY4	PB3	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SSEG3	SLCDC1	—	CMOS	Software LCD SEG output
	KEY4	TKM0C1	NSI	—	Touch key input
PB4/SSEG4/KEY5	PB4	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SSEG4	SLCDC1	—	CMOS	Software LCD SEG output
	KEY5	TKM1C1	NSI	—	Touch key input
PB5/PTCK1/SSEG5/ KEY6	PB5	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	PTCK1	PTM1C0	ST	—	PTM1 clock input
	SSEG5	SLCDC1	—	CMOS	Software LCD SEG output
	KEY6	TKM1C1	NSI	—	Touch key input
PB6/PTP1/SSEG6/ KEY7	PB6	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	PTP1	TMPC	—	CMOS	PTM1 non-inverted output
	SSEG6	SLCDC1	—	CMOS	Software LCD SEG output
	KEY7	TKM1C1	NSI	—	Touch key input
PB7/PTP1I/SSEG7/ KEY8	PB7	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	PTP1I	PTM1C0 PTM1C1	ST	—	PTM1 capture input
	SSEG7	SLCDC1	—	CMOS	Software LCD SEG output
	KEY8	TKM1C1	NSI	—	Touch key input
PC0/SSEG8/KEY11/ AN0/VREF	PC0	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SSEG8	SLCDC2	—	CMOS	Software LCD SEG output
	KEY11	TKM2C1	NSI	—	Touch key input
	AN0	ACERL	AN	—	A/D Converter analog input
	VREF	TMPC	AN	—	A/D Converter reference voltage input
PC1/SSEG9/KEY12/ AN1	PC1	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SSEG9	SLCDC2	—	CMOS	Software LCD SEG output
	KEY12	TKM2C1	NSI	—	Touch key input
	AN1	ACERL	AN	—	A/D Converter analog input
PC2/SSEG10/ KEY13/AN2	PC2	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SSEG10	SLCDC2	—	CMOS	Software LCD SEG output
	KEY13	TKM3C1	NSI	—	Touch key input
	AN2	ACERL	AN	—	A/D Converter analog input
PC3/SSEG11/KEY14/ AN3	PC3	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SSEG11	SLCDC2	—	CMOS	Software LCD SEG output
	KEY14	TKM3C1	NSI	—	Touch key input
	AN3	ACERL	AN	—	A/D Converter analog input
PC4/PTP0B/ SSEG12/KEY15/ AN4/OCVPCI	PC4	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	PTP0B	TMPC	—	CMOS	PTM0 inverted output
	SSEG12	SLCDC2	—	CMOS	Software LCD SEG output
	KEY15	TKM3C1	NSI	—	Touch key input
	AN4	ACERL	AN	—	A/D Converter analog input
	OCVPCI	OCVPC3	AN	—	OCVP OCP input signal / Comparator non-inverting input signal
PC5/PTP0/SSEG13/ KEY16/AN5/ OCVPAI0	PC5	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	PTP0	TMPC	—	CMOS	PTM0 non-inverted output
	SSEG13	SLCDC2	—	CMOS	Software LCD SEG output
	KEY16	TKM3C1	NSI	—	Touch key input
	AN5	ACERL	AN	—	A/D Converter analog input
	OCVPAI0	OCVPC3	AN	—	OCVP OCP input signal 0

Pad Name	Function	OPT	I/T	O/T	Description
PC6/CTP0B/ SSEG14/KEY17/ AN6/OCVPAI1	PC6	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	CTP0B	TMPC	—	CMOS	CTM0 inverted output
	SSEG14	SLCDC2	—	CMOS	Software LCD SEG output
	KEY17	TKM4C1	NSI	—	Touch key input
	AN6	ACERL	AN	—	A/D Converter analog input
	OCVPAI1	OCVPC3	AN	—	OCVP OCP input signal 1
PC7/CTP0/SSEG15/ KEY18/AN7/ OCVPAI2	PC7	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	CTP0	TMPC	—	CMOS	CTM0 non-inverted output
	SSEG15	SLCDC2	—	CMOS	Software LCD SEG output
	KEY18	TKM4C1	NSI	—	Touch key input
	AN7	ACERL	AN	—	A/D Converter analog input
	OCVPAI2	OCVPC3	AN	—	OCVP OCP input signal 2
PD0/SEG16/XT1	PD0	PDPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SSEG16	SLCDC3	—	CMOS	Software LCD SEG output
	XT1	CO	LXT	—	LXT pin
PD1/SEG17/XT2	PD1	PDPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SSEG17	SLCDC3	—	CMOS	Software LCD SEG output
	XT2	CO	—	LXT	LXT pin
PD2/SEG18/KEY10	PD2	PDPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SSEG18	SLCDC3	—	CMOS	Software LCD SEG output
	KEY10	TKM2C1	NSI	—	Touch key input
PD3/PTP1B/SEG19/ KEY9	PD1	PDPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	PTP1B	TMPC	—	CMOS	PTM1 inverted output
	SSEG19	SLCDC3	—	CMOS	Software LCD SEG output
	KEY9	TKM2C1	NSI	—	Touch key input
PD4/SSEG20	PD4	PDPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SSEG20	SLCDC3	—	CMOS	Software LCD SEG output
PD5/SSEG21	PD5	PDPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SSEG21	SLCDC3	—	CMOS	Software LCD SEG output
PD6/SSEG22	PD6	PDPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SSEG22	SLCDC3	—	CMOS	Software LCD SEG output
PD7/SSEG23	PD7	PDPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SSEG23	SLCDC3	—	CMOS	Software LCD SEG output
PE0/SSEG24	PE0	PEPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SSEG24	SLCDC4	—	CMOS	Software LCD SEG output
PE1/SSEG25	PE1	PEPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SSEG25	SLCDC4	—	CMOS	Software LCD SEG output
PE2/SSEG26	PE2	PEPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SSEG26	SLCDC4	—	CMOS	Software LCD SEG output
PE3/SSEG27	PE3	PEPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SSEG27	SLCDC4	—	CMOS	Software LCD SEG output
PE4/SSEG28	PE4	PEPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SSEG28	SLCDC4	—	CMOS	Software LCD SEG output
PE5/SSEG29	PE5	PEPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SSEG29	SLCDC4	—	CMOS	Software LCD SEG output
PE6/SSEG30	PE6	PEPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SSEG30	SLCDC4	—	CMOS	Software LCD SEG output

Pad Name	Function	OPT	I/T	O/T	Description
PE7/SSEG31	PE7	PEPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SSEG31	SLCDC4	—	CMOS	Software LCD SEG output
PF0/CTCK1/SSEG32	PF0	PFPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	CTCK1	CTM1C0	ST	—	CTM1 clock input
	SSEG32	SLCDC5	—	CMOS	Software LCD SEG output
PF1/CTP1/SSEG33	PF1	PFPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	CTP1	TMPC1	ST	—	CTM1 non-inverted output
	SSEG33	SLCDC5	—	CMOS	Software LCD SEG output
PF2/CTP1B/SSEG34	PF2	PFPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	CTP1B	TMPC1	ST	—	CTM1 inverted output
	SSEG34	SLCDC5	—	CMOS	Software LCD SEG output
PF3/SSEG35	PF3	PFPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SSEG35	SLCDC5	—	CMOS	Software LCD SEG output
VDD	VDD	—	PWR	—	Positive power supply
VSS	VSS	—	PWR	—	Negative power supply, ground.

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

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 these devices. Functional operation of these devices at other conditions beyond those listed in the specification is not implied and prolonged exposure to extreme conditions may affect devices 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 to 85°C

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
V_{DD}	Operating Voltage – HIRC	f_{SYS}=8MHz	2.7	—	5.5	V
		f_{SYS}=12MHz	2.7	—	5.5	
		f_{SYS}=16MHz	4.5	—	5.5	
	Operating Voltage – LXT	f_{SYS}=32768Hz	2.7	—	5.5	V
	Operating Voltage – LIRC	f_{SYS}=32kHz	2.7	—	5.5	V

Standby Current Characteristics

Ta=25°C

Symbol	Standby Mode	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{STB}	SLEEP Mode	3V	WDT on	—	1.3	3.0	μA
		5V		—	2.4	5.0	
	IDLE0 Mode – LIRC	3V	f_{SUB}=f_{LIRC}=on	—	1.3	3.0	μA
		5V		—	2.4	5.0	
	IDLE0 Mode – LXT *	3V	f_{SUB}=f_{LXT}=on, LXTLP=1	—	2.5	5.0	μA
		5V		—	6	10	
		3V	f_{SUB}=f_{LXT}=on, LXTLP=0	—	5	10	μA
		5V		—	18	30	
	IDLE1 Mode – HIRC	3V	f_{SUB} on, f_{SYS}=8MHz	—	0.8	1.2	mA
		5V		—	1.0	2.0	
		3V	f_{SUB} on, f_{SYS}=12MHz	—	0.9	1.4	mA
		5V		—	1.4	2.1	
		5V	f_{SUB} on, f_{SYS}=16MHz	—	2.0	4.0	mA

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

- Any digital inputs are setup in a non-floating condition.
- All measurements are taken under conditions of no load and with all peripherals in an off state.
- There are no DC current paths.
- All Standby Current values are taken after a HALT instruction execution thus stopping all instruction execution.
- The LXT oscillator is only available for BS87C16A-3 and BS87D20A-3.

Operating Current Characteristics

Ta=25°C

Symbol	Operating Mode	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{DD}	SLOW Mode – LIRC	3V	f_{SYS}=32kHz	—	25	50	μA
		5V		—	36	72	
	SLOW Mode – LXT *	3V	f_{SYS}=32768Hz, LXTLP=1	—	25	50	μA
		5V		—	36	72	
		3V	f_{SYS}=32768Hz, LXTLP=0	—	30	60	μA
		5V		—	48	96	
	FAST Mode – HIRC	3V	f_{SYS}=8MHz	—	1.2	1.8	mA
		5V		—	2.2	3.3	
		3V	f_{SYS}=12MHz	—	1.6	2.4	mA
		5V		—	3.3	5.0	
		5V	f_{SYS}=16MHz	—	4.0	6.0	mA

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

- Any digital inputs are setup in a non floating condition.
- All measurements are taken under conditions of no load and with all peripherals in an off state.
- There are no DC current paths.
- All Operating Current values are measured using a continuous NOP instruction program loop.
- The LXT oscillator is only available for BS87C16A-3 and BS87D20A-3.

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.

8/12/16 MHz

Symbol	Parameter	Test Conditions		Min	Typ	Max	Unit
		V_{DD}	Temp.				
f_{HIRC}	8 MHz Writer Trimmed HIRC Frequency	3V/5V	25°C	-1%	8	+1%	MHz
			-40°C ~ 85°C	-2%	8	+2%	
		2.7V~5.5V	25°C	-2.5%	8	+2.5%	
			-40°C ~ 85°C	-3%	8	+3%	
	12 MHz 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%	
	16 MHz Writer Trimmed HIRC Frequency	5V	25°C	-1%	16	+1%	MHz
			-40°C ~ 85°C	-2%	16	+2%	
		4.5V~5.5V	25°C	-2.5%	16	+2.5%	
			-40°C ~ 85°C	-3%	16	+3%	

Notes: 1. The 3V/5V values for VDD are provided as these are the two selectable fixed voltages at which the HIRC frequency is trimmed by the writer.
2. The row below the 3V/5V trim voltage row is provided to show the values for the full VDD range operating voltage. It is recommended that the trim voltage is fixed at 3V for application voltage ranges from 2.7V to 3.6V and fixed at 5V for application voltage ranges from 3.3V to 5.5V.
3. The minimum and maximum tolerance values provided in the table are only for the frequency at which the writer trims the HIRC oscillator. After trimming at this chosen specific frequency any change in HIRC oscillator frequency using the oscillator register control bits by the application program will give a frequency tolerance to within ±20%.

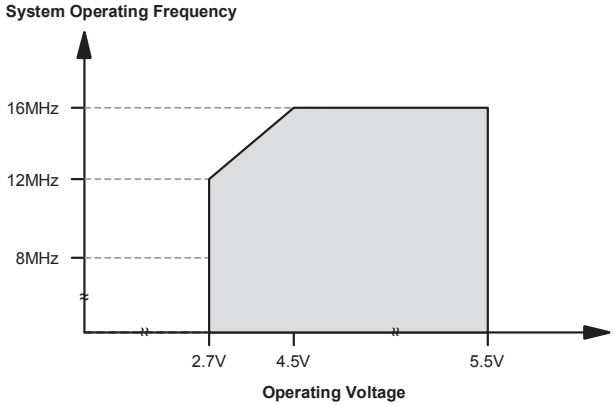
Low Speed Oscillators Characteristics – LIRC & LXT *

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Temp.				
f_{LIRC}	LIRC Oscillator Frequency	5V	25°C	-10%	32	+10%	kHz
f_{LXT}	LXT Oscillator Frequency	—	—	—	32.768	—	kHz

*: The LXT oscillator is only available for BS87C16A-3 and BS87D20A-3.

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	—	$f_{SYS}=f_H \sim f_H/64$, $f_H=f_{HIRC}$	—	16	—	t_{HIRC}
		—	$f_{SYS}=f_{SUB}=f_{LIRC}$	—	2	—	t_{LIRC}
		—	$f_{SYS}=f_{SUB}=*f_{LXT}$	—	128	—	t_{LXT}
	System Start-up Time Wake-up from Condition where f_{SYS} is on	—	$f_{SYS}=f_H \sim f_H/64$, $f_H=f_{HIRC}$	—	2	—	t_H
		—	$f_{SYS}=f_{SUB}=f_{LIRC}$ or $*f_{LXT}$	—	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	—	128	—	t_{LXT}
t_{RSTD}	System Reset Delay Time Reset Source from Power-on Reset or LVR Hardware Reset	—	RR_{POR}=5V/ms	25	50	100	ms
	System Reset Delay Time LVRC/WDTC Software Reset	—	—				
	System Reset Delay Time Reset Source from WDT Overflow	—	—	8.3	16.7	33.3	
t_{SRESET}	Minimum Software Reset 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} etc. are the inverse of the corresponding frequency values as provided in the frequency tables. For example $t_{HIRC}=1/f_{HIRC}$, $t_{SYS}=1/f_{SYS}$ etc.
3. If the LIRC is used as the system clock and if it is off when in the SLEEP Mode, then an additional LIRC start up time, t_{START}, as provided in the LIRC frequency table, must be added to the t_{SST} time in the table above.
4. The System Speed Switch Time is effectively the time taken for the newly activated oscillator to start up.
5. The LXT oscillator is only available for BS87C16A-3 and BS87D20A-3.

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 or Input Pins	5V	—	0	—	1.5	V
		—	—	0	—	$0.2V_{DD}$	
V_{IH}	Input High Voltage for I/O Ports or Input Pins	5V	—	3.5	—	5.0	V
		—	—	$0.8V_{DD}$	—	V_{DD}	
I_{OL}	Sink Current for I/O Pins	3V	V_{OL}=0.1V_{DD}	16	32	—	mA
		5V		32	64	—	
I_{OH}	Source Current for I/O Pins	3V	V_{OH}=0.9V_{DD}, PxPS=00, x=A, B, C, D, E or F	-1.0	-2.0	—	mA
		5V		-2.0	-4.0	—	
		3V	V_{OH}=0.9V_{DD}, PxPS=01, x=A, B, C, D, E or F	-1.75	-3.5	—	
		5V		-3.5	-7.0	—	
		3V	V_{OH}=0.9V_{DD}, PxPS=10, x=A, B, C, D, E or F	-2.5	-5.0	—	
		5V		-5.0	-10.0	—	
		3V	V_{OH}=0.9V_{DD}, PxPS=11, x=A, B, C, D, E or F	-5.5	-11.0	—	
		5V		-11.0	-22.0	—	
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 Minimum Pulse Width	—	—	0.3	—	—	μs
t_{TCK}	TM Clock Input Minimum Pulse Width	—	—	0.3	—	—	μs
t_{INT}	Interrupt Input Pin 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
Program Flash / Data EEPROM Memory							
t_{DEW}	Erase / Write cycle time	—	—	—	4	6	ms
I_{DDPGM}	Programming / Erase current on V_{DD}	—	—	—	—	5.0	mA
E_P	Cell Endurance	—	—	100K	—	—	E/W
t_{RETD}	ROM Data Retention time	—	Ta=25°C	—	40	—	Year
RAM Data Memory							
V_{DR}	RAM Data Retention voltage	—	Device in SLEEP Mode	1.0	—	—	V

LVD/LVR Electrical Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{LVR}	Low Voltage Reset Voltage	—	LVR enabled, voltage select 2.55V	-5%	2.55	+5%	V
V_{LVD}	Low Voltage Detect Voltage	—	LVD enabled, voltage select 2.7V	-5%	2.7	+5%	V
			LVD enabled, voltage select 3.0V		3.0		
			LVD enabled, voltage select 3.3V		3.3		
			LVD enabled, voltage select 3.6V		3.6		
			LVD enabled, voltage select 4.0V		4.0		
t_{LVDS}	LVDO Stable Time	—	For LVR enable, VBGEN=0, LVD off → on	—	—	15	μs
t_{LVR}	Minimum Low Voltage Width to Reset	—	—	120	240	480	μs
t_{LVD}	Minimum Low Voltage Width to Interrupt	—	—	60	120	240	μs

A/D Converter Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	Operating Voltage	—	—	2.7	—	5.5	V
V_{ADI}	Input Voltage	—	—	0	—	V_{REF}	V
V_{REF}	Reference Voltage	—	—	2	—	V_{DD}	V
DNL	Differential Non-linearity	3V	V_{REF}=V_{DD}, t_{ADCK}=0.5μs or 10μs	—	—	±3	LSB
		5V					
INL	Integral Non-linearity	3V	V_{REF}=V_{DD}, t_{ADCK}=0.5μs or 10μs	—	—	±4	LSB
		5V					
I_{ADC}	Additional Current Consumption for A/D Converter Enable	3V	No load, t_{ADCK} =0.5μs	—	1	2	mA
		5V		—	1.5	3	
t_{ADCK}	Clock Period	—	—	0.5	—	10	μs
t_{ON2ST}	A/D Converter On-to-Start Time	—	—	4	—	—	μs
t_{ADS}	Sampling Time	—	—	—	4	—	t_{ADCK}
t_{ADC}	Conversion Time (Including A/D Sample and Hold Time)	—	—	—	16	—	t_{ADCK}

Reference Voltage Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{BG}	Bandgap Reference Voltage	—	—	- 3%	1.09	+ 3%	V
I_{BG}	Additional Current Consumption for Bandgap Reference Voltage Enable	—	—	—	200	300	μA
t_{BG}	V_{BG} Turn-on Stable Time	—	—	—	—	200	μs

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

Touch Key Electrical Characteristics

Ta=25°C

Touch Key RC Oscillator 500kHz mode selected

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{KEYOSC}	Only sensor (KEY) Oscillator Operating Current	3V	*f_{SENOSC}=500kHz	—	30	60	μA
		5V		—	60	120	
I_{REFOSC}	Only reference Oscillator Operating Current	3V	*f_{REFOSC}=500kHz, MnTSS=0	—	30	60	μA
		5V		—	60	120	
		3V	*f_{REFOSC}=500kHz, MnTSS=1	—	30	60	μA
		5V		—	60	120	
C_{KEYOSC}	Sensor (KEY) Oscillator External Capacitor	5V	*f_{SENOSC}=500kHz	5	10	20	pF
C_{REFOSC}	Reference Oscillator Internal Capacitor	5V	*f_{REFOSC}=500kHz	5	10	20	pF
f_{KEYOSC}	Sensor (KEY) Oscillator Operating Frequency	5V	*C_{EXT}=7, 8, 9, 10, 11, 12, 13, 14, 15, … , 50pF	100	500	1000	kHz
f_{REFOSC}	Reference Oscillator Operating Frequency	5V	* C_{INT}=7, 8, 9, 10, 11, 12, 13, 14, 15, … , 50pF	100	500	1000	kHz

Note: 1. f_{SENOSC}=500kHz: Adjust KEYn external capacitor to make sure that the Sensor oscillator frequency is equal to 500kHz.
2. f_{REFOSC}=500kHz: Adjust Reference oscillator internal capacitor to make sure that the reference oscillator frequency is equal to 500kHz.

Touch Key RC Oscillator 1000kHz mode selected

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{KEYOSC}	Only sensor (KEY) Oscillator Operating Current	3V	*f_{SENOSC}=1000kHz	—	40	80	μA
		5V		—	80	160	
I_{REFOSC}	Only reference Oscillator Operating Current	3V	*f_{REFOSC}=1000kHz, MnTSS=0	—	40	80	μA
		5V		—	80	160	
		3V	*f_{REFOSC}=1000kHz, MnTSS=1	—	40	80	μA
		5V		—	80	160	
C_{KEYOSC}	Sensor (KEY) Oscillator External Capacitor	5V	*f_{SENOSC}=1000kHz	5	10	20	pF
C_{REFOSC}	Reference Oscillator Internal Capacitor	5V	*f_{REFOSC}=1000kHz	5	10	20	pF
f_{KEYOSC}	Sensor (KEY) Oscillator Operating Frequency	5V	*C_{EXT}=1, 2, 3, 4, 5, 6, 7, 8, 9, … , 50pF	150	1000	2000	kHz
f_{REFOSC}	Reference Oscillator Operating Frequency	5V	* C_{INT}=1, 2, 3, 4, 5, 6, 7, 8, 9, … , 50pF	150	1000	2000	kHz

Note: 1. f_{SENOSC}=1000kHz: Adjust KEYn external capacitor to make sure that the Sensor oscillator frequency is equal to 1000kHz.
2. f_{REFOSC}=1000kHz: Adjust Reference oscillator internal capacitor to make sure that the reference oscillator frequency is equal to 1000kHz.

Touch Key RC Oscillator 1500kHz mode selected

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{KEYOSC}	Only sensor (KEY) Oscillator Operating Current	3V	*f_{SENOSC}=1500kHz	—	60	120	μA
		5V		—	120	240	
I_{REFOSC}	Only reference Oscillator Operating Current	3V	*f_{REFOSC}=1500kHz, MnTSS=0	—	60	120	μA
		5V		—	120	240	
		3V	*f_{REFOSC}=1500kHz, MnTSS=1	—	60	120	μA
		5V		—	120	240	
C_{KEYOSC}	Sensor (KEY) Oscillator External Capacitor	3V	*f_{SENOSC}=1500kHz	4	8	16	pF
		5V		5	10	20	pF
C_{REFOSC}	Reference Oscillator Internal Capacitor	3V	*f_{REFOSC}=1500kHz	4	8	16	pF
		5V		5	10	20	pF
f_{KEYOSC}	Sensor (KEY) Oscillator Operating Frequency	3V	*C_{EXT}=1, 2, 3, 4, 5, 6, 7, 8, 9, … , 50pF	150	1500	3000	kHz
		5V		150	1500	3000	kHz
f_{REFOSC}	Reference Oscillator Operating Frequency	3V	* C_{INT}=1, 2, 3, 4, 5, 6, 7, 8, 9, … , 50pF	150	1500	3000	kHz
		5V		150	1500	3000	kHz

Note: 1. f_{SENOSC}=1500kHz: Adjust KEYn external capacitor to make sure that the Sensor oscillator frequency is equal to 1500kHz.
2. f_{REFOSC}=1500kHz: Adjust Reference oscillator internal capacitor to make sure that the reference oscillator frequency is equal to 1500kHz.

Touch Key RC Oscillator 2000kHz mode selected

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{KEYOSC}	Only sensor (KEY) Oscillator Operating Current	3V	*f_{SENOSC}=2000kHz	—	80	160	μA
		5V		—	160	320	
I_{REFOSC}	Only reference Oscillator Operating Current	3V	*f_{REFOSC}=2000kHz, MnTSS=0	—	80	160	μA
		5V		—	160	320	
		3V	*f_{REFOSC}=2000kHz, MnTSS=1	—	80	160	μA
		5V		—	160	320	
C_{KEYOSC}	Sensor (KEY) Oscillator External Capacitor	3V	*f_{SENOSC}=2000kHz	4	8	16	pF
		5V		5	10	20	pF
C_{REFOSC}	Reference Oscillator Internal Capacitor	3V	*f_{REFOSC}=2000kHz	4	8	16	pF
		5V		5	10	20	pF
f_{KEYOSC}	Sensor (KEY) Oscillator Operating Frequency	3V	*C_{EXT}=1, 2, 3, 4, 5, 6, 7, 8, 9, … , 50pF	150	2000	4000	kHz
		5V		150	2000	4000	kHz
f_{REFOSC}	Reference Oscillator Operating Frequency	3V	* C_{INT}=1, 2, 3, 4, 5, 6, 7, 8, 9, … , 50pF	150	2000	4000	kHz
		5V		150	2000	4000	kHz

Note: 1. f_{SENOSC}=2000kHz: Adjust KEYn external capacitor to make sure that the Sensor oscillator frequency is equal to 2000kHz.
2. f_{REFOSC}=2000kHz: Adjust Reference oscillator internal capacitor to make sure that the reference oscillator frequency is equal to 2000kHz.

OCVP Electrical Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{OCVP}	Additional Current Consumption for OCVP Enable	3V	DAC V_{REF}=2.5V	—	—	1.25	mA
		5V		—	0.73	1.25	
V_{OS_CMP}	Comparator Input Offset Voltage	3V/5V	Without calibration, OCVPCOF[4:0]=10000	-15	—	15	mV
		3V/5V	With calibration	-4	—	4	
V_{CM_CMP}	Comparator Common Mode Voltage Range	3V/5V	—	V_{SS}	—	V_{DD}-1.4	V
V_{OS_OPA}	OPA Input Offset Voltage	3V/5V	Without calibration, OCVPOOF[5:0]=100000	-15	—	15	mV
		3V/5V	With calibration	-4	—	4	
V_{CM_OPA}	OPA Common Mode Voltage Range	3V/5V	—	V_{SS}	—	V_{DD}-1.4	V
V_{HYS}	Hysteresis	3V/5V	—	20	40	60	mV
V_{OR}	OPA Maximum Output Voltage Range	3V/5V	—	V_{SS}+0.1	—	V_{DD}-0.1	V
Ga	PGA Gain Accuracy (Note)	3V/5V	—	-5	—	5	%
V_{REF}	DAC Reference Voltage	3V/5V	OCVPVRS=1	2	—	V_{DD}	V
DNL	Differential Non-linearity	3V	DAC V_{REF}=V_{DD}	—	—	±2	LSB
		5V		—	—	±1	
INL	Integral Non-linearity	3V	DAC V_{REF}=V_{DD}	—	—	±2	LSB
		5V		—	—	±1.5	

Note: The PGA gain accuracy is guaranteed only when the PGA output voltage meets the V_{OR} specification.

Software Controlled LCD Electrical Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{BIAS}	LCD bias current	5V	ISEL[1:0]=00	5.8	8.3	10.8	μA
			ISEL[1:0]=01	11.7	16.7	21.7	
			ISEL[1:0]=10	35	50	65	
			ISEL[1:0]=11	70	100	130	
V_{SCOM}	LCD COM 1/3 Bias Level Output	—	No load	0.317V_{DD}	0.333V_{DD}	0.35V_{DD}	V
	LCD COM 2/3 Bias Level Output			0.634V_{DD}	0.666V_{DD}	0.7V_{DD}	
V_{SSEG}	LCD SEG 1/3 Bias Level Output	—	No load	0.317V_{DD}	0.333V_{DD}	0.35V_{DD}	V
	LCD SEG 2/3 Bias Level Output			0.634V_{DD}	0.666V_{DD}	0.7V_{DD}	

I²C Characteristics

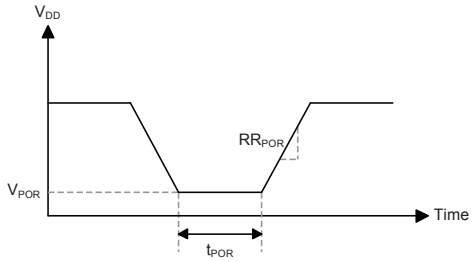
Ta=25°C

Symbol	Parameter	Test Condition		Min.	Typ.	Max.	Unit
		V_{DD}	Condition				
f_{I2C}	System Frequency for I²C Standard Mode (100kHz)	—	No clock debounce	2	—	—	MHz
			2 system clocks debounce	4	—	—	
			4 system clocks debounce	8	—	—	
	System Frequency for I²C Fast Mode (400kHz)	—	No clock debounce	5	—	—	MHz
			2 system clocks debounce	10	—	—	
			4 system clocks debounce	20	—	—	

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



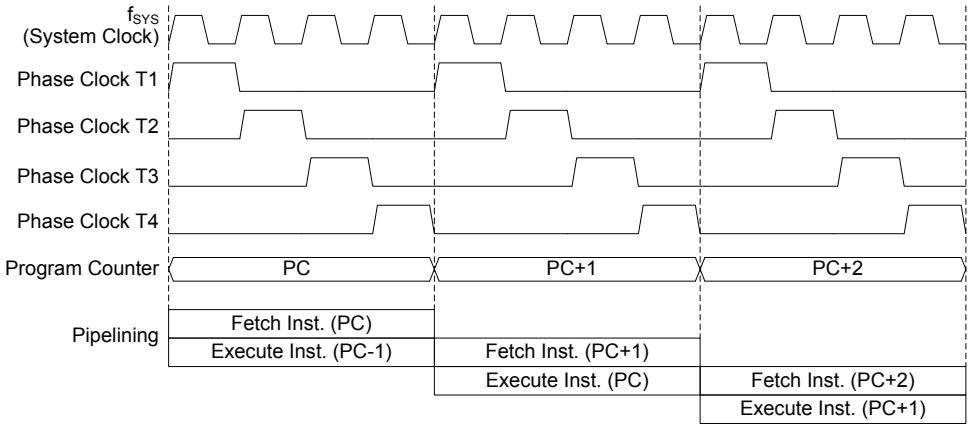
System Architecture

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

Clocking and Pipelining

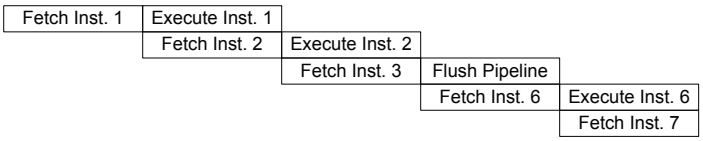
The main system clock, derived from either a HIRC, LXT or LIRC oscillator is subdivided into four internally generated non-overlapping clocks, T1~T4. Note that the LXT oscillator is only available for the BS87C16A-3 and BS87D20A-3 devices. The Program Counter is incremented at the beginning of the T1 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 demand a jump to a non-consecutive Program Memory address. Only the lower 8 bits, known as the Program Counter Low Register, are directly addressable by the application program.

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

Device	Program Counter	
	High Byte	Low Byte (PCL)
BS87B12A-3	PC11~PC8	PC7~PC0
BS87C16A-3	PC11~PC8	PC7~PC0
BS87D20A-3	PC12~PC8	PC7~PC0

Program Counter

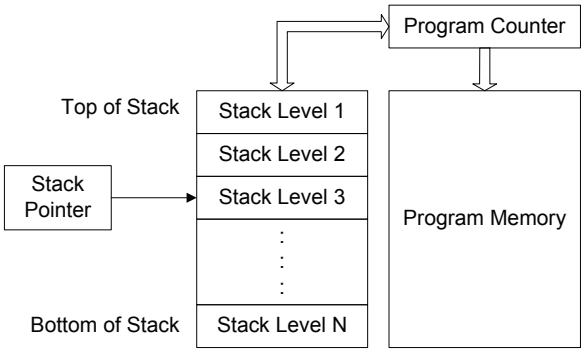
The lower byte of the Program Counter, known as the Program Counter Low register or PCL, is available for program control and is a readable and writeable register. By transferring data directly into this register, a short program jump can be executed directly; however, as only this low byte is available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



Note: N=6 for BS87B12A-3 & BS87C16A-3 while N=8 for BS87D20A-3.

Arithmetic and Logic Unit – ALU

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

- Arithmetic operations:
  ADD, ADDM, ADC, ADCM, SUB, SUBM, SBC, SBCM, DAA
  LADD, LADDM, LADC, LADCM, LSUB, LSUBM, LSBC, LSBCM, LDAA
- Logic operations:
  AND, OR, XOR, ANDM, ORM, XORM, CPL, CPLA
  LAND, LOR, LXOR, LANDM, LORM, LXORM, LCPL, LCPLA
- Rotation:
  RRA, RR, RRCA, RRC, RLA, RL, RLCA, RLC
  LRRA, LRR, 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
  LSZ, LSZA, LSNZ, LSIZ, LSDZ, LSIZA, LSDZA

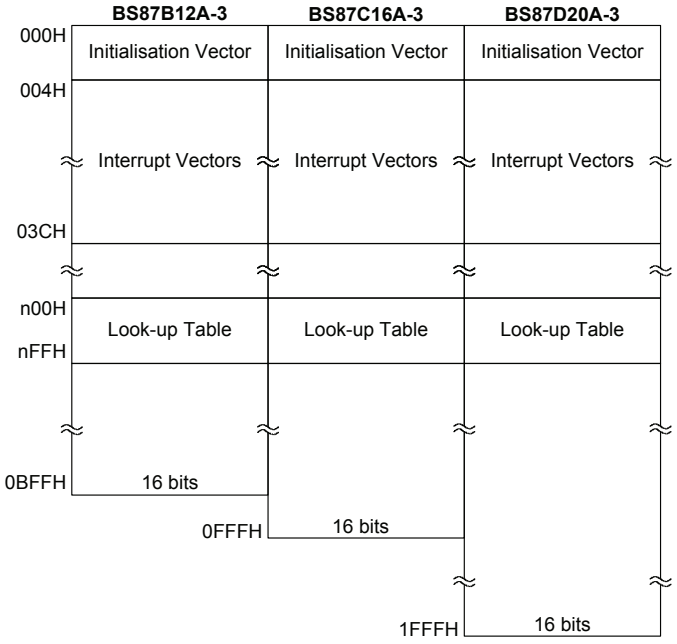
Flash Program Memory

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

Device	Capacity
BS87B12A-3	3K × 16
BS87C16A-3	4K × 16
BS87D20A-3	8K × 16

Structure

The Program Memory has a capacity of 3K×16 to 8K×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 registers.



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 these devices reset for program initialisation. After a device reset is initiated, the program will jump to this location and begin execution.

Look-up Table

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

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

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

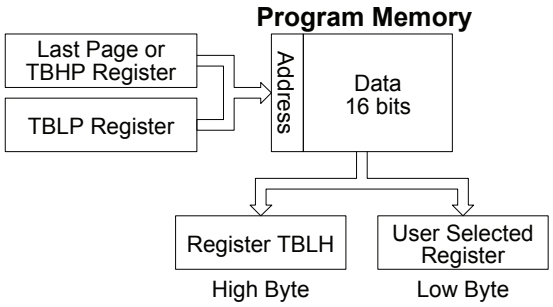


Table Program Example

The accompanying example shows how the table pointer and table data is defined and retrieved from the device. This example uses raw table data located in the last page which is stored there using the ORG statement. The value at this ORG statement is "1F00H" which refers to the start address of the last page within the 8K Program Memory of the device. The table pointer low byte register is setup here to have an initial value of "06H". This will ensure that the first data read from the data table will be at the Program Memory address "1F06H" or 6 locations after the start of the last page. Note that the value for the table pointer is referenced to the first address of the present page pointed by the TBHP register if the "TABRD [m]" instruction is being used. The high byte of the table data which in this case is equal to zero will be transferred to the TBLH register automatically when the "TABRD [m]" instruction is executed.

Because the TBLH register is a read/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,1fh         ; initialise high table pointer
mov  tbhp,a
:
tabrd   tempreg1   ; transfers value in table referenced by table pointer data at program
                   ; memory address "1F06H" 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 "1F05H" transferred to tempreg2 and TBLH in this
                   ; example the data "1AH" is transferred to tempreg1 and data "0FH" to
                   ; register tempreg2
:
org  1F00h         ; sets initial address of program memory
dc      00Ah, 00Bh, 00Ch, 00Dh, 00Eh, 00Fh, 01Ah, 01Bh
:
```

In Circuit Programming – ICP

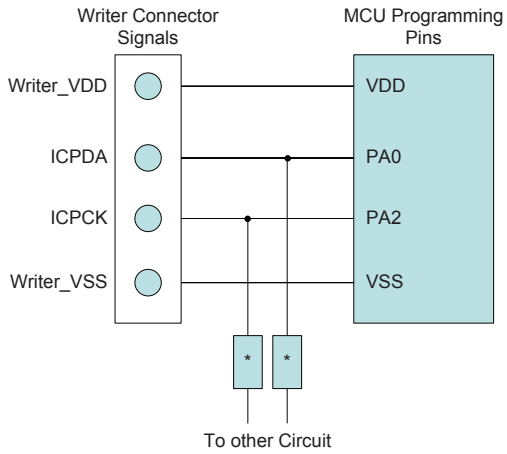
The provision of Flash type Program Memory provides the user with a means of convenient and easy upgrades and modifications to their programs on the same device.

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

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

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

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



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

On-Chip Debug Support – OCDS

There are EV chips named BS87BV12A, BS87CV16A and BS87DV20A which are used to emulate the real MCU device named BS87B12A-3, BS87C16A-3 and BS87D20A-3 respectively. The EV chip device also provides the "On-Chip Debug" function to debug the real MCU device during development process. The EV chips and real MCU devices are almost functional compatible except the "On-Chip Debug" function. Users can use the EV chip device to emulate the real MCU device behaviors by connecting the OCDSDA and OCDSCK pins to the Holtek HT-IDE development tools. The OCDSDA pin is the OCDS Data/Address input/output pin while the OCDSCK pin is the OCDS clock input pin. When users use the EV chip device for debugging, the corresponding pin functions shared with the OCDSDA and OCDSCK pins in the real 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 more detailed OCDS information, refer to the corresponding document named "Holtek e-Link for 8-bit MCU OCDS User's Guide".

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

Data Memory

The Data Memory is an 8-bit wide RAM internal memory and is the location where temporary information is stored.

Structure

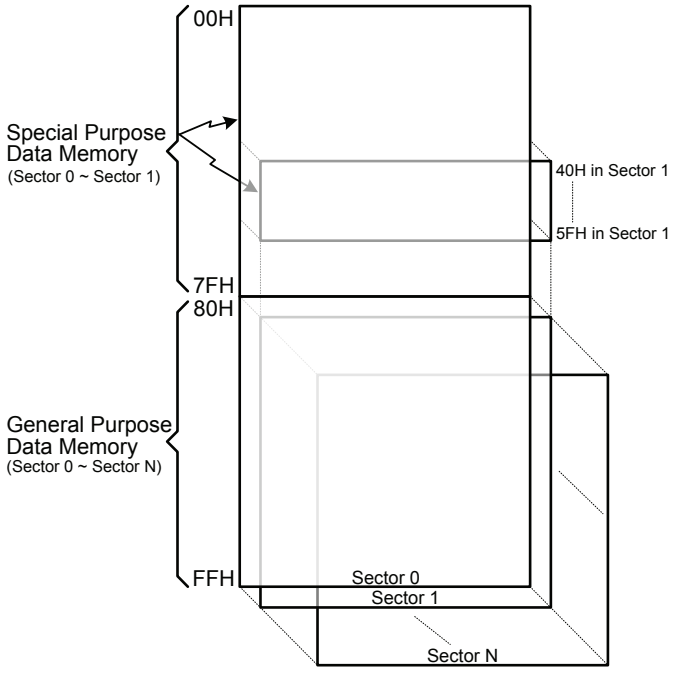
Divided into two types, the first of Data Memory is an area of RAM where special function registers are located. These registers have fixed locations and are necessary for correct operation of the device. Many of these registers can be read from and written to directly under program control, however,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

The 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. Switching between the different Data Memory sectors is achieved by properly setting the Memory Pointers to correct value.

Device	Special Purpose Data Memory	General Purpose Data Memory	
	Sector : Address	Capacity	Sector : Address
BS87B12A-3	0~2: 00H~7FH (EEC @40H only accessible in Sector 1)	384 x 8	0~2: 80H~FFH
BS87C16A-3	0~3: 00H~7FH (EEC @40H only accessible in Sector 1)	512 x 8	0~3: 80H~FFH
BS87D20A-3	0~5: 00H~7FH (EEC @40H only accessible in Sector 1)	768 x 8	0~5: 80H~FFH

Data Memory Summary



Note: N=2 for BS87B12A-3; N=3 for BS87C16A-3; N=5 for BS87D20A-3

Data Memory Structure

Data Memory Addressing

For these devices that support 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 from 10 to 11 bits depending upon which device is selected, the high byte indicates a sector and the low byte indicates a specific address.

General Purpose Data Memory

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

Special Purpose Data Memory

This area of Data Memory is where registers, necessary for the correct operation of the microcontroller, are stored. Most of the registers are both readable and 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 0~2
00H	IAR0
01H	MP0
02H	IAR1
03H	MP1L
04H	MP1H
05H	ACC
06H	PCL
07H	TBLP
08H	TBLH
09H	TBHP
0AH	STATUS
0BH	SMOD
0CH	IAR2
0DH	MP2L
0EH	MP2H
0FH	INTEG
10H	INTC0
11H	INTC1
12H	INTC2
13H	INTC3
14H	PA
15H	PAC
16H	PAPU
17H	PAWU
18H	SLEDC0
19H	SLEDC1
1AH	WDTC
1BH	TBC
1CH	PSCR
1DH	LVRC
1EH	EEA
1FH	EED
20H	PB
21H	PBC
22H	PBPU
23H	SIMTOC
24H	SIMC0
25H	SIMC1
26H	SIMD
27H	SIMA/SIMC2
28H	USR
29H	UCR1
2AH	UCR2
2BH	BRG
2CH	TXR_RXR
2DH	SADOL
2EH	SADOH
2FH	SADC0
30H	SADC1
31H	ACERL
32H	TMPC0
33H	SLCDC0
34H	SLCDC1
35H	SLCDC2
36H	
37H	LVDC
38H	OCVPC0
39H	PC
3AH	PCC
3BH	PCPU
3CH	IFS
3DH	CTRL
3EH	OCVPC1
3FH	OCVPC2

	Sector 0, 2	Sector 1
40H	OCVPC3	EEC
41H		
42H		
43H		
44H	TKTMR	
45H	TKC0	
46H	TK16DL	
47H	TK16DH	
48H	TKC1	
49H	TKM016DL	
4AH	TKM016DH	
4BH	TKM0ROL	
4CH	TKM0ROH	
4DH	TKM0C0	
4EH	TKM0C1	
4FH	TKM116DL	
50H	TKM116DH	
51H	TKM1ROL	
52H	TKM1ROH	
53H	TKM1C0	
54H	TKM1C1	
55H	TKM216DL	
56H	TKM216DH	
57H	TKM2ROL	
58H	TKM2ROH	
59H	TKM2C0	
5AH	TKM2C1	
5BH		
5CH		
5DH		
5EH		
5FH		
60H		
61H	CTM0C0	
62H	CTM0C1	
63H	CTM0DL	
64H	CTM0DH	
65H	CTM0AL	
66H	CTM0AH	
67H	PTM0C0	
68H	PTM0C1	
69H	PTM0DL	
6AH	PTM0DH	
6BH	PTM0AL	
6CH	PTM0AH	
6DH	PTM0RPL	
6EH	PTM0RPH	
6FH		
70H		
71H		
72H		
73H		
74H		
75H		
76H		
77H		
78H		
79H		
7AH		
7BH		
7CH		
7DH	OCVPDA	
7EH	OCVPOCAL	
7FH	OCVPCCAL	

: Unused, read as 00H

Special Purpose Data Memory Structure – BS87B12A-3

	Sector 0~3
00H	IAR0
01H	MP0
02H	IAR1
03H	MP1L
04H	MP1H
05H	ACC
06H	PCL
07H	TBLP
08H	TBLH
09H	TBHP
0AH	STATUS
0BH	SMOD
0CH	IAR2
0DH	MP2L
0EH	MP2H
0FH	INTEG
10H	INTC0
11H	INTC1
12H	INTC2
13H	INTC3
14H	PA
15H	PAC
16H	PAPU
17H	PAWU
18H	SLEDC0
19H	SLEDC1
1AH	WDTC
1BH	TBC
1CH	PSCR
1DH	LVRC
1EH	EEA
1FH	EED
20H	PB
21H	PBC
22H	PBPU
23H	SIMTOC
24H	SIMC0
25H	SIMC1
26H	SIMD
27H	SIMA/SIMC2
28H	USR
29H	UCR1
2AH	UCR2
2BH	BRG
2CH	TXR_RXR
2DH	SADOL
2EH	SADOH
2FH	SADC0
30H	SADC1
31H	ACERL
32H	TMPC0
33H	SLCDC0
34H	SLCDC1
35H	SLCDC2
36H	SLCDC3
37H	LVDC
38H	OCVPC0
39H	PC
3AH	PCC
3BH	PCPU
3CH	MFI
3DH	CTRL
3EH	OCVPC1
3FH	OCVPC2

	Sector 0, 2~3	Sector 1
40H	OCVPC3	EEC
41H	PD	
42H	PDC	
43H	PDPU	
44H	TKTMR	
45H	TKC0	
46H	TK16DL	
47H	TK16DH	
48H	TKC1	
49H	TKM016DL	
4AH	TKM016DH	
4BH	TKM0ROL	
4CH	TKM0ROH	
4DH	TKM0C0	
4EH	TKM0C1	
4FH	TKM116DL	
50H	TKM116DH	
51H	TKM1ROL	
52H	TKM1ROH	
53H	TKM1C0	
54H	TKM1C1	
55H	TKM216DL	
56H	TKM216DH	
57H	TKM2ROL	
58H	TKM2ROH	
59H	TKM2C0	
5AH	TKM2C1	
5BH	TKM316DL	
5CH	TKM316DH	
5DH	TKM3ROL	
5EH	TKM3ROH	
5FH	TKM3C0	
60H	TKM3C1	
61H	CTM0C0	
62H	CTM0C1	
63H	CTM0DL	
64H	CTM0DH	
65H	CTM0AL	
66H	CTM0AH	
67H	PTM0C0	
68H	PTM0C1	
69H	PTM0DL	
6AH	PTM0DH	
6BH	PTM0AL	
6CH	PTM0AH	
6DH	PTM0RPL	
6EH	PTM0RPH	
6FH		
70H		
71H		
72H		
73H		
74H		
75H	PTM1C0	
76H	PTM1C1	
77H	PTM1DL	
78H	PTM1DH	
79H	PTM1AL	
7AH	PTM1AH	
7BH	PTM1RPL	
7CH	PTM1RPH	
7DH	OCVPDA	
7EH	OCVPOCAL	
7FH	OCVPCCAL	

: Unused, read as 00H

Special Purpose Data Memory Structure – BS87C16A-3

	Sector 0~5
00H	IAR0
01H	MP0
02H	IAR1
03H	MP1L
04H	MP1H
05H	ACC
06H	PCL
07H	TBLP
08H	TBLH
09H	TBHP
0AH	STATUS
0BH	SMOD
0CH	IAR2
0DH	MP2L
0EH	MP2H
0FH	INTEG
10H	INTC0
11H	INTC1
12H	INTC2
13H	INTC3
14H	PA
15H	PAC
16H	PAPU
17H	PAWU
18H	SLEDC0
19H	SLEDC1
1AH	WDTC
1BH	TBC
1CH	PSCR
1DH	LVRC
1EH	EEA
1FH	EED
20H	PB
21H	PBC
22H	PBPU
23H	SIMTOC
24H	SIMC0
25H	SIMC1
26H	SIMD
27H	SIMA/SIMC2
28H	USR
29H	UCR1
2AH	UCR2
2BH	BRG
2CH	TXR_RXR
2DH	SADOL
2EH	SADOH
2FH	SADC0
30H	SADC1
31H	ACERL
32H	TMPC0
33H	SLCDC0
34H	SLCDC1
35H	SLCDC2
36H	SLCDC3
37H	LVDC
38H	OCVPC0
39H	PC
3AH	PCC
3BH	PCPU
3CH	MFI
3DH	CTRL
3EH	OCVPC1
3FH	OCVPC2

	Sector 0, 2~5	Sector 1
40H	OCVPC3	EEC
41H	PD	PE
42H	PDC	PEC
43H	PDPU	PEPU
44H	TKTMR	PF
45H	TKC0	PFC
46H	TK16DL	PFPU
47H	TK16DH	SLEDC2
48H	TKC1	SLCDC4
49H	TKM016DL	SLCDC5
4AH	TKM016DH	TMPC1
4BH	TKM0ROL	
4CH	TKM0ROH	
4DH	TKM0C0	
4EH	TKM0C1	
4FH	TKM116DL	
50H	TKM116DH	
51H	TKM1ROL	
52H	TKM1ROH	
53H	TKM1C0	
54H	TKM1C1	
55H	TKM216DL	
56H	TKM216DH	
57H	TKM2ROL	
58H	TKM2ROH	
59H	TKM2C0	
5AH	TKM2C1	CTM1C0
5BH	TKM316DL	CTM1C1
5CH	TKM316DH	CTM1DL
5DH	TKM3ROL	CTM1DH
5EH	TKM3ROH	CTM1AL
5FH	TKM3C0	CTM1AH
60H	TKM3C1	
61H	CTM0C0	
62H	CTM0C1	
63H	CTM0DL	
64H	CTM0DH	
65H	CTM0AL	
66H	CTM0AH	
67H	PTM0C0	
68H	PTM0C1	
69H	PTM0DL	
6AH	PTM0DH	
6BH	PTM0AL	
6CH	PTM0AH	
6DH	PTM0RPL	
6EH	PTM0RPH	
6FH	TKM416DL	
70H	TKM416DH	
71H	TKM4ROL	
72H	TKM4ROH	
73H	TKM4C0	
74H	TKM4C1	
75H	PTM1C0	
76H	PTM1C1	
77H	PTM1DL	
78H	PTM1DH	
79H	PTM1AL	
7AH	PTM1AH	
7BH	PTM1RPL	
7CH	PTM1RPH	
7DH	OCVPDA	
7EH	OCVPOCAL	
7FH	OCVPCCAL	

: Unused, read as 00H

Special Purpose Data Memory Structure – BS87D20A-3

Special Function Register Description

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

Indirect Addressing Registers – IAR0, IAR1, IAR2

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

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

Five Memory Pointers, known as MP0, MP1L, MP1H, MP2L and MP2H, are provided. These Memory Pointers are physically implemented in the Data Memory and can be manipulated in the same way as normal registers providing a convenient way with which to address and track data. When any operation to the relevant Indirect Addressing Registers is carried out, the actual address that the 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 data sectors according to the corresponding MP1H or MP2H register. Direct Addressing can be used in all data sectors using the corresponding instruction which can address all available data memory space.

Indirect Addressing Program Example

- **Example 1**

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

- **Example 2**

```
data .section 'data'
adres1 db ?
adres2 db ?
adres3 db ?
adres4 db ?
block  db ?
code .section at 0 'code'
org 00h
start:
mov a,04h             ; setup size of block
mov block,a
mov a,01h             ; setup the memory sector
mov mp1h,a
mov a,offset adres1   ; Accumulator loaded with first RAM address
mov mp1l,a            ; setup memory pointer with first RAM address
loop:
clr IAR1              ; clear the data at address defined by MP1
inc mp1l              ; increment memory pointer MP1L
sdz block             ; check if last memory location has been cleared
jmp loop
continue:
```

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

Direct Addressing Program Example using extended instructions

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

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

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. The TBLP and TBHP registers are the table pointer pair and indicates the location where the table data is located. Their value must be setup before any table read instructions are executed. Their value can be changed, for example using the "INC" or "DEC" instructions, allowing for easy table data pointing and reading. TBLH is the location where the high order byte of the table data is stored after a table read data instruction has been executed. Note that the lower order table data byte is transferred to a user defined location.

Status Register – STATUS

This 8-bit register contains the zero flag (Z), carry flag (C), auxiliary carry flag (AC), overflow flag (OV), SC flag, CZ flag, power down flag (PDF), and watchdog time-out flag (TO). These arithmetic/logical operation and system management flags are used to record the status and operation of the 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.
- SC is the result of the "XOR" operation which is performed by the OV flag and the MSB of the current instruction operation result.
- CZ is the operational result of different flags for different instructions. Refer to register definitions for more details.

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

STATUS Register

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

"×": 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

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

Device	Capacity	Address
BS87B12A-3	64 × 8	00H ~ 3FH
BS87C16A-3		
BS87D20A-3		

EEPROM Data Memory Structure

The EEPROM Data Memory capacity is 64×8 bits for the series of devices. Unlike the Program Memory and RAM Data Memory, the EEPROM Data Memory is not directly mapped into memory space and is therefore not directly addressable in the same way as the other types of memory. Read and Write operations to the EEPROM are carried out in single byte operations using an address and data register in 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 MP1H/MP1L or MP2H/MP2L Memory Pointer pair and Indirect Addressing Register, IAR1 or IAR2. Because the EEC control register is located at address 40H in sector 1, the Memory Pointer low byte register, MP1L or MP2L, must first be set to the value 40H and the Memory Pointer high byte register, MP1H or MP2H, set to the value, 01H, before any operations on the EEC register are executed.

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

EEPROM Registers List

EEA Register

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

Bit 7~6 Unimplemented, read as 0.

Bit 5~0 **EEA5~EEA0**: Data EEPROM address bit 5 ~ bit 0

EED Register

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

Bit 7~0 **D7~D0**: Data EEPROM data bit 7~bit0

EEC Register

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

Bit 7~4 Unimplemented, read as 0.

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

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

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

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

Note: The WREN, WR, RDEN and RD can not be set to "1" at the same time in one instruction. The WR and RD can not be set to "1" at the same time.

Reading Data from the EEPROM

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

Writing Data to the EEPROM

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

Write Protection

Protection against inadvertent write operation is provided in several ways. After the 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, EEPROM interrupt is 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.

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 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, 040H                    ; setup memory pointer low byte MP1L
MOV  MP1L, A                    ; MP1L points to EEC register
MOV  A, 01H                     ; setup Memory Pointer high byte 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 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 low byte MP1L
MOV  MP1L, A                    ; MP1L points to EEC register
MOV  A, 01H                     ; setup Memory Pointer high byte MP1H
MOV  MP1H, A
CLR  EMI
SET  IAR1.3                     ; set WREN bit, enable write operations
SET  IAR1.2                     ; start Write Cycle - set WR bit
SET  EMI
BACK:
SZ   IAR1.2                     ; check for write cycle end
JMP  BACK
CLR  IAR1                       ; disable EEPROM write
CLR  MP1H
```

Oscillators

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

Oscillator Overview

In addition to being the source of the main system clock the oscillators also provide clock sources for the Watchdog Timer and Time Base Interrupts. External oscillators 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 corresponding configuration options. The higher frequency oscillators provide higher performance but carry with it the disadvantage of higher power requirements, while the opposite is of course true for the lower frequency oscillators. With the capability of dynamically switching between fast and slow system clock, the device has the flexibility to optimize the performance/power ratio, a feature especially important in power sensitive portable applications. Note that the external low speed crystal oscillator, LXT, is only available for the BS87C16A-3 and BS87D20A-3 devices

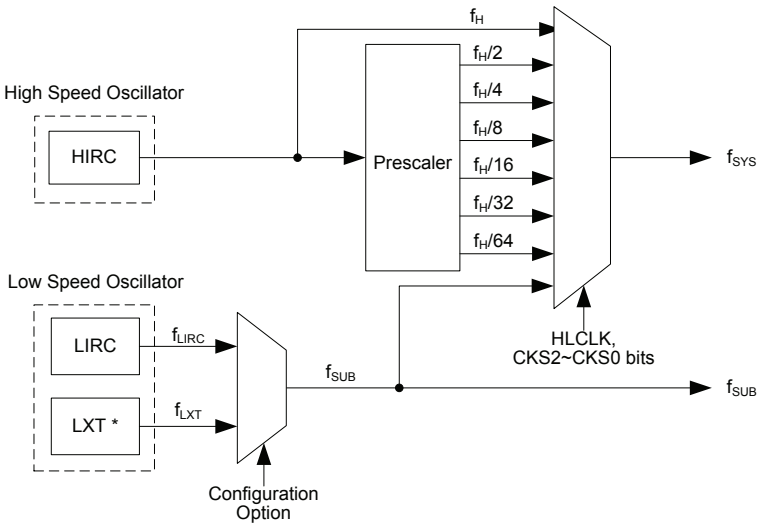
Type	Name	Frequency	Pins
Internal High Speed RC	HIRC	8/12/16 MHz	—
Internal Low Speed RC	LIRC	32 kHz	—
External Low Speed Crystal*	LXT*	32.768 kHz	XT1/XT2

Oscillator Types

System Clock Configurations

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

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



Note: The LXT oscillator is only available for BS87C16A-3 and BS87D20A-3

System Clock Configurations

Internal High Speed RC Oscillator – HIRC

The internal RC oscillator is a fully integrated system oscillator requiring no external components. The internal RC oscillator has a power-on default frequency of 8MHz but can be selected to be either 8MHz, 12MHz or 16MHz via a configuration option and the HIRCS1 and HIRCS0 bits in the CTRL register. Device trimming during the manufacturing process and the inclusion of internal frequency compensation circuits are used to ensure that the influence of the power supply voltage, temperature and process variations on the oscillation frequency are minimised.

Internal 32kHz Oscillator – LIRC

The Internal 32 kHz System Oscillator is one of the low frequency oscillator choices, which is selected via a configuration option for the BS87C16A-3 and BS87D20A-3 devices. For the BS87B12A-3 device there is only one low frequency oscillator known as the LIRC oscillator. It is a fully integrated RC oscillator with a typical frequency of 32 kHz 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.

External 32.768 kHz Crystal Oscillator – LXT – for BS87C16A-3/BS87D20A-3

The External 32.768 kHz Crystal System Oscillator is one of the low frequency oscillator choices, which is selected via a configuration option. This clock source has a fixed frequency of 32.768 kHz and requires a 32.768 kHz crystal to be connected between pins XT1 and XT2. The external resistor and capacitor components connected to the 32.768 kHz 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. During power-up there is a time delay associated with the LXT oscillator waiting for it to start-up.

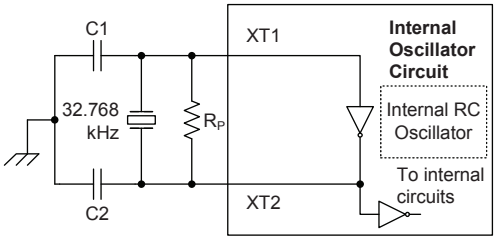
When the microcontroller enters the SLEEP or IDLE Mode, the system clock is switched off to stop microcontroller activity and to conserve power. However, in many microcontroller applications it may be necessary to keep the internal timers operational even when the microcontroller is in the SLEEP or IDLE 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, Rp, is required.

The configuration option determines if the XT1/XT2 pins are used for the LXT oscillator or as I/O or other pin-shared functions.

- If the LXT oscillator is not used for any clock source, the XT1/XT2 pins can be used as normal I/O or other pin-shared functions.
- 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 resistors and capacitors along with interconnecting lines are all located as close to the MCU as possible.



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

External LXT Oscillator

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

32.768kHz Crystal Capacitor Recommended Value

LXT Oscillator Low Power Function

The LXT oscillator can function in one of two modes, the Quick Start Mode and the Low Power Mode. The mode selection is executed using the LXTLP bit in the CTRL register.

LXTLP	LXT Mode
0	Quick Start
1	Low Power

After power on, the LXTLP bit will be automatically cleared to zero ensuring that the LXT oscillator is in the Quick Start operating mode. In the Quick Start Mode the LXT oscillator will power up and stabilise quickly. However, after the LXT oscillator has fully powered up it can be placed into the Low-power mode by setting the LXTLP bit high. The oscillator will continue to run but with reduced current consumption, as the higher current consumption is only required during the LXT oscillator start-up. In power sensitive applications, such as battery applications, where power consumption must be kept to a minimum, it is therefore recommended that the application program sets the LXTLP bit high about 2 seconds after power-on.

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

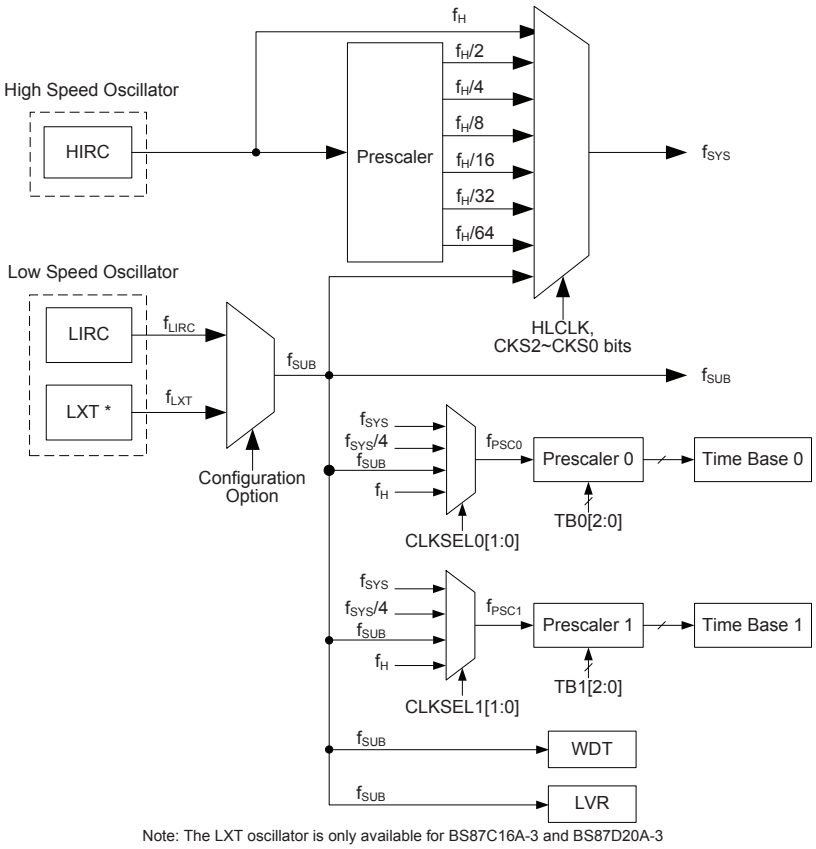
Operating Modes and System Clocks

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

System Clocks

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

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



Note: The LXT oscillator is only available for BS87C16A-3 and BS87D20A-3

Device Clock Configurations

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

System Operation Modes

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

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

FAST Mode

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 high speed oscillator, HIRC. The high speed oscillator will however first be divided by a ratio ranging from 1 to 64, the actual ratio being selected by the CKS2~CKS0 and HLCLK bits in the SMOD register. Although a high speed oscillator is used, running the microcontroller at a divided clock ratio reduces the operating current.

SLOW Mode

This is also a mode where the microcontroller operates normally although now with a slower speed clock source. The clock source used will be from f_{SUB}. The f_{SUB} clock is derived from either the LIRC or LXT oscillator for BS87C16A-3 and BS87D20A-3 devices while the f_{SUB} clock is from the LIRC oscillator for the BS87B12A-3 device. Running the microcontroller in this mode allow it to run with much lower operating currents. In the SLOW Mode, the f_H is off.

SLEEP Mode

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

IDLE0 Mode

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

IDLE1 Mode

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

Control Registers

The register, SMOD, is used to control the internal clocks within the device.

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

System Operating Mode Control Registers List

SMOD Register

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

Bit 7~5 **CKS2~CKS0**: System clock selection when HLCLK is "0"
000: f_{SUB} (LIRC or LXT)
001: f_{SUB} (LIRC or LXT)
010: $f_H/64$
011: $f_H/32$
100: $f_H/16$
101: $f_H/8$
110: $f_H/4$
111: $f_H/2$
These three bits are used to select which clock is used as the system clock source. In addition to the system clock source, which can be either the LXT or LIRC, a divided version of the high speed system oscillator can also be chosen as the system clock source.

Bit 4 Unimplemented, read as 0.

Bit 3 **LTO**: Low speed system oscillator ready flag
0: Not ready
1: Ready
This is the low speed system oscillator ready flag which indicates when the low speed system oscillator is stable after power on reset or a wake-up has occurred. The flag will be low when in the SLEEP Mode but after a wake-up has occurred, the flag will change to a high level after 128 clock cycles if LXT oscillator is used and 1~2 clock cycles if the LIRC oscillator is used.

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

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

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

CTRL Register – BS87B12A-3

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

"×": unknown

CTRL Register – BS87C16A-3 & BS87D20A-3

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

"×": unknown

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

Bit 6 Unimplemented, read as 0.

Bit 5~4 **HIRCS1~HIRCS0**: HIRC frequency clock selection
00: 8 MHz
01: 12 MHz
10: 16 MHz
11: 8 MHz
It is recommended that the HIRC frequency selected by these two bits is the same with the frequency determined by the configuration option to keep the HIRC frequency accuracy specified in the A.C. characteristics.

Bit 3 **LXTLP**: LXT low power control
0: Quick Start mode
1: Low Power mode
Note that this bit is used to selec the operating mode of the LXT oscillator which is only available for BS87C16A-3 and BS87D20A-3 devices. For the BS87B12A-3 device this bit is unimplemented and is read as "0".

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

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

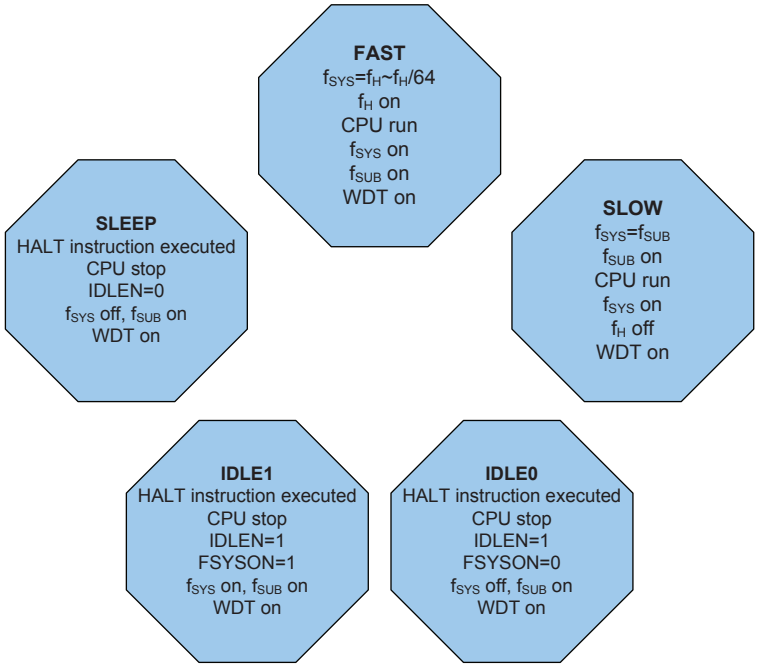
Bit 0 **WRF**: WDT control register software reset flag
Described elsewhere

Operating Mode Switching

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

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

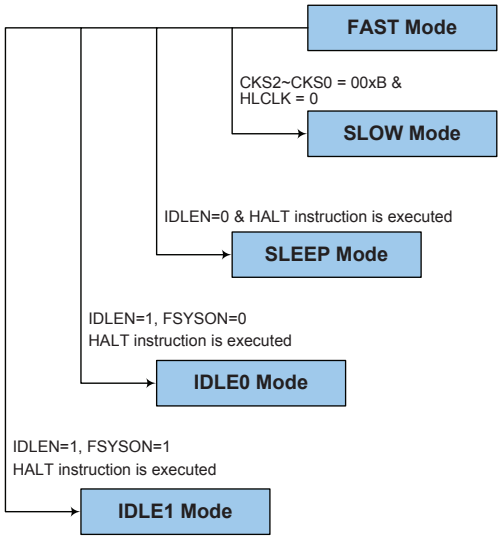
When the HLCLK bit switches to a low level, which implies that clock source is switched from the high speed clock source, f_H, to the clock source, $f_H/2$~ $f_H/64$ or f_{SUB}. If the clock is from the f_{SUB}, the high speed clock source will stop running to conserve power. When this happens it must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



FAST Mode to SLOW Mode Switching

When running in the FAST Mode, which uses the high speed system oscillator, and therefore consumes more power, the system clock can switch to run in the SLOW Mode by clearing the HLCLK bit to zero and setting the CKS2~CKS0 bits to "000" or "001" in the SMOD register. This will then use the low speed system oscillator which will consume less power. Users may decide to do this for certain operations which do not require high performance and can subsequently reduce power consumption.

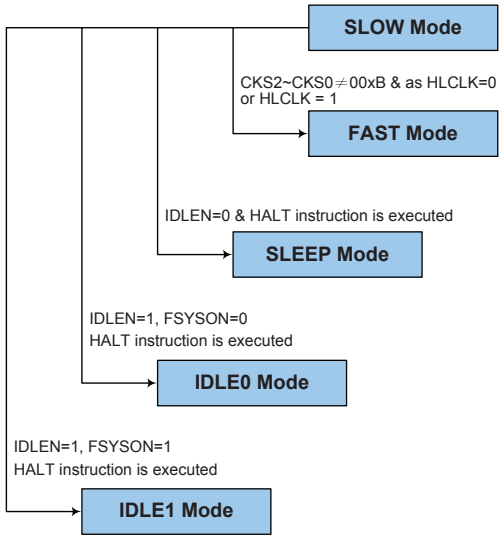
The SLOW Mode is sourced from the LXT or LIRC oscillator and therefore requires the specific oscillator to stable before full mode switching occurs.



SLOW Mode to FAST Mode Switching

In SLOW mode the system uses the f_{SUB} clock derived from either the LXT or LIRC low speed oscillator as system clock. When system clock is switched back to the FAST mode from f_{SUB}, where the high speed system oscillator is used, the HLCLK bit should be set high or HLCLK bit is low but the CKS2~CKS0 bits are set to "010 ~111" and then the system clock will respectively be switched to f_H~ f_H/64.

However, if f_H is not used in SLOW mode and thus switched off, it will take some time to re-oscillate and stabilise when switching back to the FAST mode from the SLOW Mode and the status of the HTO flag should be checked. The time duration required for the high speed system oscillator stabilization is specified in the relevant characteristics.



Entering the SLEEP Mode

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

- The system clock will be stopped and the application program will stop at the "HALT" instruction.
- The Data Memory contents and registers will maintain their present condition.
- The I/O ports will maintain their present conditions.
- In the status register, the Power Down flag PDF will be set, and WDT timeout flag TO will be cleared.
- The WDT will be cleared and resume counting 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 IDLEN bit in the SMOD register equal to "1" and the FSYSON bit in the CTRL register equal to "0". When this instruction is executed under the conditions described above, the following will occur:

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

Entering the IDLE1 Mode

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

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

Standby Current Considerations

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

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

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

Wake-up

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

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

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

When the device executes the "HALT" instruction, 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.

System Oscillator	Wake-up Time (SLEEP Mode)	Wake-up Time (IDLE0 Mode)	Wake-up Time (IDLE1 Mode)
HIRC	15~16 HIRC cycles		1~2 HIRC cycles
LIRC	1~2 LIRC cycles		1~2 LIRC cycles
LXT*	128 LXT cycles		1~2 LXT cycles

Note: The LXT oscillator is only available for the BS87C16A-3 and BS87D20A-3 devices.

Wake-up Time

Programming Considerations

The high speed and low speed oscillators both use the same SST counter. For example, if the system is woken up from the SLEEP Mode the HIRC oscillator needs to start-up from an off state. If the device is woken up from the SLEEP Mode to the FAST Mode, the high speed system oscillator needs an SST period. The device will execute the first instruction after HTO is high. At this time, the LXT oscillator may not be stability if f_{SUB} is from LXT oscillator. The same situation occurs in the power-on state. The LXT oscillator is not ready yet when the first instruction is executed.

Watchdog Timer

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

Watchdog Timer Clock Source

The Watchdog Timer clock source is provided by the internal f_{SUB} clock shich is in turn supplied by either the LXT or LIRC oscillator selected by a configuration option. The LIRC internal oscillator has an approximate frequency of 32 kHz and this specified internal clock period can vary with V_{DD}, temperature and process variations. The LXT oscillator is supplied by an external 32.768 kHz crystal. 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 control. This register controls the overall operation of the Watchdog Timer. The WDTC register is initiated to 01010011B at any reset except the WDT time-out hardware warm reset.

WDTC Register

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

Bit 7~3 **WE4~WE0**: WDT function control
01010 or 10101: Enabled
Other values: Reset MCU
If these bits are changed due to adverse environmental conditions, the microcontroller will be reset. The reset operation will be activated after a delay time, t_{SRESET}, and the WRF bit in the RSTFC register will be set to 1.

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

CTRL Register

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

"×": unknown

Bit 7 **FSYSON**: f_{SYS} Control in IDLE Mode
Described elsewhere
Bit 6 Unimplemented, read as 0.
Bit 5~4 **HIRCS1~HIRCS0**: HIRC frequency clock selection
Described elsewhere

Bit 3 **LXTLP**: LXT low power control
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 by the application program. Note that this bit can only be cleared to zero by the application program.

Watchdog Timer Operation

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

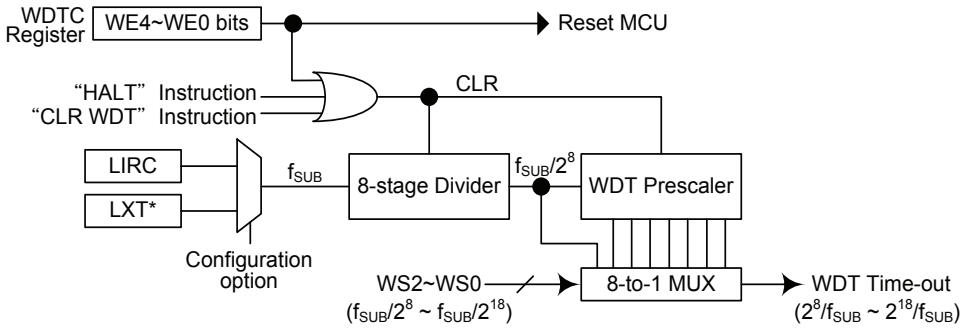
WE4 ~ WE0 Bits	WDT Function
10101B or 01010B	Enable
Any other value	Reset MCU

Watchdog Timer Enable/Reset Control

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

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

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



Note: The LXT oscillator is only available for BS87C16A-3 and BS87D20A-3

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.

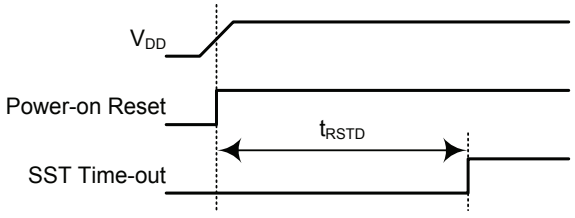
The Watchdog Timer overflow is one of many reset types and will reset the microcontroller. Another reset exists in the form of a Low Voltage Reset, LVR, where a full reset implemented in situations where the power supply voltage falls below a certain threshold. All types of reset operations result in 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 pins will be first set to inputs.

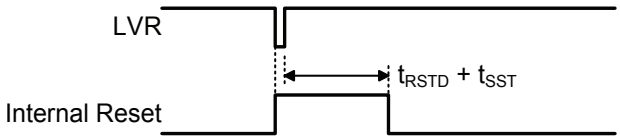


Note: t_{RSTD} is power-on delay with typical time=50 ms

Power-On Reset Timing Chart

Low Voltage Reset – LVR

The microcontroller contains 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 CTRL register will also be set to 1. For a valid LVR signal, a low supply voltage, i.e., a voltage in the range between 0.9V~V_{LVR} must exist for a time greater than that specified by t_{LVR} in the LVD/LVR 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 is fixed at a voltage value of 2.55V by the LVS bits in the LVRC register. If the LVS7~LVS0 bits have any other value, which may perhaps occur due to adverse environmental conditions such as noise, the LVR will reset the device after a delay time, t_{SRESET}. When this happens, the LRF bit in the CTRL register will be set to 1. After power on the register will have the default value of 01010101B. Note that the LVR function will be automatically disabled when the device enters the SLEEP/IDLE mode.



Note: t_{RSTD} is power-on delay with typical time=50 ms

Low Voltage Reset Timing Chart

- **LVRC Register**

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

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

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

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

- **CTRL Register**

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

"×": unknown

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

Bit 6 Unimplemented, read as 0.

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

Bit 3 **LXTLP**: LXT low power control
Described elsewhere

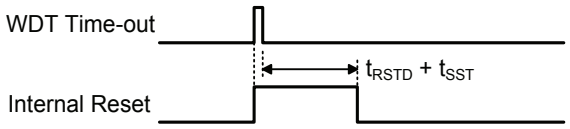
Bit 2 **LVRF**: LVR function reset flag
0: Not occurred
1: Occurred
This bit is set to 1 when a specific low voltage reset condition occurs. Note that this bit can only be cleared to 0 by the application program.

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

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

Watchdog Time-out Reset during Normal Operation

The Watchdog time-out Reset during normal operation is the same as the hardware Low Voltage Reset except that the Watchdog time-out flag TO will be set to "1".

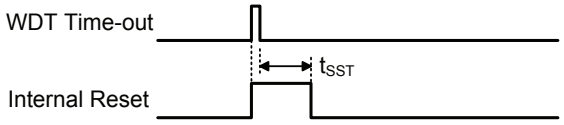


Note: t_{RSTD} is power-on delay with typical time=16.7 ms

WDT Time-out Reset during NORMAL Operation Timing Chart

Watchdog Time-out Reset during SLEEP or IDLE Mode

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



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 Function
0	0	Power-on reset
u	u	LVR reset during NORMAL or SLOW Mode operation
1	u	WDT time-out reset during NORMAL or SLOW Mode operation
1	1	WDT time-out reset during IDLE or SLEEP Mode operation

"u" stands for unchanged

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

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

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

Register	BS87B12A-3	BS87C16A-3	BS87D20A-3	Power On Reset	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
TBHP			●	---x xxxx	---u uuuu	---u uuuu	---u uuuu
STATUS	●	●	●	xx00 xxxx	uuuu uuuu	xx1u uuuu	uu11 uuuu
SMOD	●	●	●	000- 0011	000- 0011	000- 0011	uuu- uuuu
IAR2	●	●	●	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
MP2L	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
MP2H	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
INTEG	●	●	●	---- --00	---- --00	---- --00	---- --uu

Register	BS87B12A-3	BS87C16A-3	BS87D20A-3	Power On Reset	LVR Reset (Normal Operation)	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)*
INTC0	●	●	●	-000 0000	-000 0000	-000 0000	-uuu uuuu
INTC1	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
INTC2	●			000- 000-	000- 000-	000- 000-	uuu- uuu-
INTC2		●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
INTC3	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PA	●	●	●	1--1 1111	1--1 1111	1--1 1111	u--u uuuu
PAC	●	●	●	1--1 1111	1--1 1111	1--1 1111	u--u uuuu
PAPU	●	●	●	0--0 0000	0--0 0000	0--0 0000	u--u uuuu
PAWU	●	●	●	0--0 0000	0--0 0000	0--0 0000	u--u uuuu
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
PD		●	●	1111 1111	1111 1111	1111 1111	uuuu uuuu
PDC		●	●	1111 1111	1111 1111	1111 1111	uuuu uuuu
PDPU		●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
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			●	---- 1111	---- 1111	---- 1111	---- uuuu
PFC			●	---- 1111	---- 1111	---- 1111	---- uuuu
PFPU			●	---- 0000	---- 0000	---- 0000	---- uuuu
SLEDC0	●	●	●	0101 0101	0101 0101	0101 0101	uuuu uuuu
SLEDC1	●			---- 0101	---- 0101	---- 0101	---- uuuu
SLEDC1		●	●	0101 0101	0101 0101	0101 0101	uuuu uuuu
SLEDC2			●	--01 0101	--01 0101	--01 0101	--uu uuuu
WDTC	●	●	●	0101 0011	0101 0011	0101 0011	uuuu uuuu
TBC	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PSCR	●	●	●	--00 --00	--00 --00	--00 --00	--uu --uu
LVRC	●	●	●	0101 0101	0101 0101	0101 0101	uuuu uuuu
EEA	●	●	●	--00 0000	--00 0000	--00 0000	--uu uuuu
EED	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
SIMTOC	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
SIMC0	●	●	●	111- 0000	111- 0000	111- 0000	uuu- uuuu
SIMC1	●	●	●	1000 0001	1000 0001	1000 0001	uuuu uuuu
SIMD	●	●	●	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
SIMA/SIMC2	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
USR	●	●	●	0000 1011	0000 1011	0000 1011	uuuu uuuu
UCR1	●	●	●	0000 00x0	0000 00x0	0000 00x0	uuuu uuuu
UCR2	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
BRG	●	●	●	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TXR_RXR	●	●	●	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu

Register	BS87B12A-3	BS87C16A-3	BS87D20A-3	Power On Reset	LVR Reset (Normal Operation)	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)*
SADOL	●	●	●	xxxx ----	xxxx ----	xxxx ----	uuuu ---- (ADRFS=0)
							uuuu uuuu (ADRFS=1)
SADOH	●	●	●	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu (ADRFS=0)
							---- uuuu (ADRFS=1)
SADC0	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
SADC1	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
ACERL	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TMPC0	●			0--- 0000	0--- 0000	0--- 0000	u--- uuuu
TMPC0		●	●	0-00 0000	0-00 0000	0-00 0000	u-uu uuuu
TMPC1			●	---- --00	---- --00	---- --00	---- --uu
SLCDC0	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
SLCDC1	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
SLCDC2	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
SLCDC3		●		---- 0000	---- 0000	---- 0000	---- uuuu
SLCDC3			●	0000 0000	0000 0000	0000 0000	uuuu uuuu
SLCDC4			●	0000 0000	0000 0000	0000 0000	uuuu uuuu
SLCDC5			●	---- 0000	---- 0000	---- 0000	---- uuuu
LVDC	●	●	●	--00 0000	--00 0000	--00 0000	--uu uuuu
OCVPC0	●	●	●	1100 1000	1100 1000	1100 1000	uuuu uuuu
IFS	●			---- 0000	---- 0000	---- 0000	---- uuuu
MFI		●		--00 --00	--00 --00	--00 --00	--uu --uu
MFI			●	0000 0000	0000 0000	0000 0000	uuuu uuuu
CTRL	●			0-00 -x00	0-00 -x00	0-00 -x00	u-uu -uuu
CTRL		●	●	0-00 0x00	---- u1uu	---- uuuu	---- uuuu
OCVPC1	●	●	●	0000 --00	0000 --00	0000 --00	uuuu --uu
OCVPC2	●	●	●	-000 0000	-000 0000	-000 0000	-uuu uuuu
OCVPC3	●	●	●	0000 --00	0000 --00	0000 --00	uuuu --uu
TKTMR	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKC0	●	●	●	-000 0000	-000 0000	-000 0000	-uuu uuuu
TK16DL	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TK16DH	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKC1	●	●	●	---- --11	---- --11	---- --11	---- --uu
TKM016DL	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM016DH	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM0ROL	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM0ROH	●	●	●	---- --00	---- --00	---- --00	---- --uu
TKM0C0	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM0C1	●	●	●	0-00 0000	0-00 0000	0-00 0000	u-uu uuuu
TKM116DL	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM116DH	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM1ROL	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu

Register	BS87B12A-3	BS87C16A-3	BS87D20A-3	Power On Reset	LVR Reset (Normal Operation)	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)*
TKM1ROH	●	●	●	---- --00	---- --00	---- --00	---- --uu
TKM1C0	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM1C1	●	●	●	0-00 0000	0-00 0000	0-00 0000	u-uu uuuu
TKM216DL	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM216DH	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM2ROL	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM2ROH	●	●	●	---- --00	---- --00	---- --00	---- --uu
TKM2C0	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM2C1	●	●	●	0-00 0000	0-00 0000	0-00 0000	u-uu uuuu
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		●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM3C1		●	●	0-00 0000	0-00 0000	0-00 0000	u-uu 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			●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM4C1			●	0-00 0000	0-00 0000	0-00 0000	u-uu uuuu
CTM0C0	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
CTM0C1	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
CTM0DL	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
CTM0DH	●	●	●	---- --00	---- --00	---- --00	---- --uu
CTM0AL	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
CTM0AH	●	●	●	---- --00	---- --00	---- --00	---- --uu
PTM0C0	●	●	●	0000 0---	0000 0---	0000 0---	uuuu u---
PTM0C1	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM0DL	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM0DH	●	●	●	---- --00	---- --00	---- --00	---- --uu
PTM0AL	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM0AH	●	●	●	---- --00	---- --00	---- --00	---- --uu
PTM0RPL	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM0RPH	●	●	●	---- --00	---- --00	---- --00	---- --uu
PTM1C0		●	●	0000 0---	0000 0---	0000 0---	uuuu u---
PTM1C1		●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM1DL		●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM1DH		●	●	---- --00	---- --00	---- --00	---- --uu
PTM1AL		●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM1AH		●	●	---- --00	---- --00	---- --00	---- --uu
PTM1RPL		●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM1RPH		●	●	---- --00	---- --00	---- --00	---- --uu
CTM1C0			●	0000 0000	0000 0000	0000 0000	uuuu uuuu

Register	BS87B12A-3	BS87C16A-3	BS87D20A-3	Power On Reset	LVR Reset (Normal Operation)	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)*
CTM1C1			•	0000 0000	0000 0000	0000 0000	uuuu uuuu
CTM1DL			•	0000 0000	0000 0000	0000 0000	uuuu uuuu
CTM1DH			•	---- --00	---- --00	---- --00	---- --uu
CTM1AL			•	0000 0000	0000 0000	0000 0000	uuuu uuuu
CTM1AH			•	---- --00	---- --00	---- --00	---- --uu
OCVPDA	•	•	•	0000 0000	0000 0000	0000 0000	uuuu uuuu
OCVPOCAL	•	•	•	0010 0000	0010 0000	0010 0000	uuuu uuuu
OCVPCCAL	•	•	•	0001 0000	0001 0000	0001 0000	uuuu uuuu
EEC	•	•	•	---- 0000	---- 0000	---- 0000	---- uuuu

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

Input/Output Ports

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

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

Register Name	Bit							
	7	6	5	4	3	2	1	0
PA	PA7	—	—	PA4	PA3	PA2	PA1	PA0
PAC	PAC7	—	—	PAC4	PAC3	PAC2	PAC1	PAC0
PAPU	PAPU7	—	—	PAPU4	PAPU3	PAPU2	PAPU1	PAPU0
PAWU	PAWU7	—	—	PAWU4	PAWU3	PAWU2	PAWU1	PAWU0
PB	PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0
PBC	PBC7	PBC6	PBC5	PBC4	PBC3	PBC2	PBC1	PBC0
PBPU	PBPU7	PBPU6	PBPU5	PBPU4	PBPU3	PBPU2	PBPU1	PBPU0
PC	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
IFS	—	—	—	—	CTP0BPS	CTP0PS	OCVPAI2PS	OCVPAI1PS

"—": Unimplemented, read as "0".

I/O Logic Function Registers List – BS87B12A-3

Register Name	Bit							
	7	6	5	4	3	2	1	0
PA	PA7	—	—	PA4	PA3	PA2	PA1	PA0
PAC	PAC7	—	—	PAC4	PAC3	PAC2	PAC1	PAC0
PAPU	PAPU7	—	—	PAPU4	PAPU3	PAPU2	PAPU1	PAPU0
PAWU	PAWU7	—	—	PAWU4	PAWU3	PAWU2	PAWU1	PAWU0
PB	PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0
PBC	PBC7	PBC6	PBC5	PBC4	PBC3	PBC2	PBC1	PBC0
PBPU	PBPU7	PBPU6	PBPU5	PBPU4	PBPU3	PBPU2	PBPU1	PBPU0
PC	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

"—": Unimplemented, read as "0".

I/O Logic Function Registers List – BS87C16A-3

Register Name	Bit							
	7	6	5	4	3	2	1	0
PA	PA7	—	—	PA4	PA3	PA2	PA1	PA0
PAC	PAC7	—	—	PAC4	PAC3	PAC2	PAC1	PAC0
PAPU	PAPU7	—	—	PAPU4	PAPU3	PAPU2	PAPU1	PAPU0
PAWU	PAWU7	—	—	PAWU4	PAWU3	PAWU2	PAWU1	PAWU0
PB	PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0
PBC	PBC7	PBC6	PBC5	PBC4	PBC3	PBC2	PBC1	PBC0
PBPU	PBPU7	PBPU6	PBPU5	PBPU4	PBPU3	PBPU2	PBPU1	PBPU0
PC	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	—	—	—	—	PF3	PF2	PF1	PF0
PFC	—	—	—	—	PFC3	PFC2	PFC1	PFC0
PFPU	—	—	—	—	PFPU3	PFPU2	PFPU1	PFPU0

"—": Unimplemented, read as "0".

I/O Logic Function Registers List – BS87D20A-3

Pull-high Resistors

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

PxPU Register

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

PxPUn: I/O Port x Pin pull-high function control

0: Disable

1: Enable

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

Port A Wake-up

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

PAWU Register

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

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

Bit 6~5 Unimplemented, read as 0.

I/O Port Control Registers

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

PxC Register

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

PxCn: I/O Port x Pin type selection

0: Output

1: Input

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

I/O Port Source Current Selection

These devices support different output source current driving capability for each I/O port. With the selection register, SLEDCn, specific I/O port can support four levels of the source current driving capability. Users should refer to the I/O Port characteristics section to select the desired output source current for different applications.

Register Name	Bit							
	7	6	5	4	3	2	1	0
SLEDC0	PBPS3	PBPS2	PBPS1	PBPS0	PAPS3	PAPS2	PAPS1	PAPS0
SLEDC1	—	—	—	—	PCPS3	PCPS2	PCPS1	PCPS0

I/O Port Source Current Selection Registers List – BS87B12A-3

Register Name	Bit							
	7	6	5	4	3	2	1	0
SLEDC0	PBPS3	PBPS2	PBPS1	PBPS0	PAPS3	PAPS2	PAPS1	PAPS0
SLEDC1	PDPS3	PDPS2	PDPS1	PDPS0	PCPS3	PCPS2	PCPS1	PCPS0

I/O Port Source Current Selection Registers List – BS87C16A-3

Register Name	Bit							
	7	6	5	4	3	2	1	0
SLEDC0	PBPS3	PBPS2	PBPS1	PBPS0	PAPS3	PAPS2	PAPS1	PAPS0
SLEDC1	PDPS3	PDPS2	PDPS1	PDPS0	PCPS3	PCPS2	PCPS1	PCPS0
SLEDC2	—	—	PFPS1	PFPS0	PEPS3	PEPS2	PEPS1	PEPS0

I/O Port Source Current Selection Registers List – BS87D20A-3

SLEDC0 Register

Bit	7	6	5	4	3	2	1	0
Name	PBPS3	PBPS2	PBPS1	PBPS0	PAPS3	PAPS2	PAPS1	PAPS0
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 **PBPS3~PBPS2**: 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 **PBPS1~PBPS0**: 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 **PAPS3~PAPS2**: PA7 and PA4 source current selection
00: source current=Level 0 (min.)
01: source current=Level 1
10: source current=Level 2
11: source current=Level 3 (max.)

Bit 1~0 **PAPS1~PAPS0**: 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 – BS87B12A-3

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	PCPS3	PCPS2	PCPS1	PCPS0
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 **PCPS3~PCPS2**: 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 **PCPS1~PCPS0**: 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 – BS87C16A-3 & BS87D20A-3

Bit	7	6	5	4	3	2	1	0
Name	PDPS3	PDPS2	PDPS1	PDPS0	PCPS3	PCPS2	PCPS1	PCPS0
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 **PDPS3~PDPS2**: 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 **PDPS1~PDPS0**: 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 **PCPS3~PCPS2**: 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 **PCPS1~PCPS0**: 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 – BS87D20A-3

Bit	7	6	5	4	3	2	1	0
Name	—	—	PFPS1	PFPS0	PEPS3	PEPS2	PEPS1	PEPS0
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 **PFPS1~PFPS0**: PF3~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 **PEPS3~PEPS2**: 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 **PEPS1~PEPS0**: 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-remapping Function – BS87B12A-3 only

The flexibility of the microcontroller range is greatly enhanced by the use of pins that have more than one function. Limited numbers of pins can force serious design constraints on designers but by supplying pins with multi-functions, many of these difficulties can be overcome. The way in which the pin function of specific pins is selected is different for each function and a priority order is established where more than one pin function is selected simultaneously. Note that the pin-remapping function is only available for the BS87B12A-3 device.

Pin-remapping Selection Register

The limited number of supplied pins in a package can impose restrictions on the amount of functions a certain device can contain. However by allowing the same pins to share several different functions and providing a means of function selection, a wide range of different functions can be incorporated into even relatively small package sizes.

- **IFS Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	CTP0BPS	CTP0PS	OCVPAI2PS	OCVPAI1PS
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0

Bit 7~4 Unimplemented, read as 0.

Bit 3 **CTP0BPS**: CTP0B pin remapping function selection
0: CTP0B on PC6
1: CTP0B on PC2

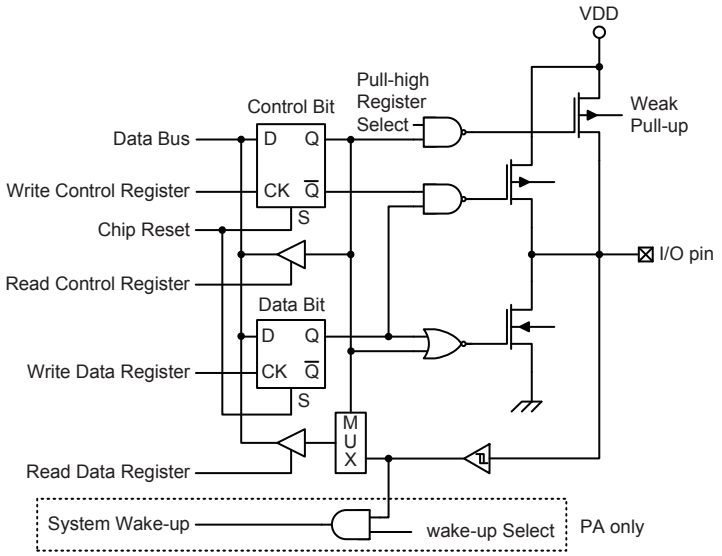
Bit 2 **CTP0PS**: CTP0 pin remapping function selection
0: CTP0 on PC7
1: CTP0 on PC3

Bit 1 **OCVPAI2P**: OCVPAI2 pin remapping function selection
0: OCVPAI2 on PC7
1: OCVPAI2 on PC3

Bit 0 **OCVPAI1P**: OCVPAI1 pin remapping function selection
0: OCVPAI1 on PC6
1: OCVPAI1 on PC2

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 drawing, 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.



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 device includes 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 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 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
BS87B12A-3	CTM0	PTM0
BS87C16A-3	CTM0	PTM0, PTM1
BS87D20A-3	CTM0, CTM1	PTM0, PTM1

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 xTnCK2~xTnCK0 bits in the xTMn control registers, where "x" stands for C or P type TM and "n" stands for the specific TM serial number. The clock source can be a ratio of the system clock, f_{SYS}, or the internal high clock, f_H, the f_{SUB} clock source or the external xTCKn pin. The xTCKn pin clock source is used to allow an external signal to drive the TM as an external clock source for event counting.

TM Interrupts

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

TM External Pins

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

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

The TMs each have two output pin, xTPn and xTPnB. 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 xTPn and xTPnB output pins are also the pins where the xTMn generates the PWM output waveform. As the xTMn output pins are pin-shared with other functions, the TM output function must first be setup using the relevant registers. A signal bit in the register determines if its associated pin is to be used as an external TM output pin or if it is to have another function. The number of external pins for each TM type is different, the details are provided in the accompanying table.

Device	CTM		PTM	
	Input	Output	Input	Output
BS87B12A-3	CTCK0	CTP0, CTP0B	PTCK0, PTP0I	PTP0, PTP0B
BS87C16A-3	CTCK0	CTP0, CTP0B	PTCK0, PTP0I PTCK1, PTP1I	PTP0, PTP0B PTP1, PTP1B
BS87D20A-3	CTCK0 CTCK1	CTP0, CTP0B CTP1, CTP1B	PTCK0, PTP0I PTCK1, PTP1I	PTP0, PTP0B PTP1, PTP1B

TM External Pins

TM Input/Output Pin Control Register

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

TMPC0 Register – BS87B12A-3

Bit	7	6	5	4	3	2	1	0
Name	VREFS	—	—	—	PTM0PC1	PTM0PC0	CTM0PC1	CTM0PC0
R/W	R/W	—	—	—	R/W	R/W	R/W	R/W
POR	0	—	—	—	0	0	0	0

Bit 7 **VREFS**: VREF pin control
0: Disabled
1: Enabled

Bit 6~4 Unimplemented, read as 0.

Bit 3 **PTM0PC1**: PTP0B pin control
0: Disabled
1: Enabled

Bit 2 **PTM0PC0**: PTP0 pin control
0: Disabled
1: Enabled

Bit 1 **CTM0PC1**: CTP0B pin control
0: Disabled
1: Enabled

Bit 0 **CTM0PC0**: CTP0 pin control
0: Disabled
1: Enabled

TMPC0 Register – BS87C16A-3 & BS87D20A-3

Bit	7	6	5	4	3	2	1	0
Name	VREFS	—	PTM1PC1	PTM1PC0	PTM0PC1	PTM0PC0	CTM0PC1	CTM0PC0
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 **VREFS**: VREF pin control
0: Disabled
1: Enabled

Bit 6 Unimplemented, read as 0.

Bit 5 **PTM1PC1**: PTP1B pin control
0: Disabled
1: Enabled

Bit 4 **PTM1PC0**: PTP1 pin control
0: Disabled
1: Enabled

Bit 3 **PTM0PC1**: PTP0B pin control
0: Disabled
1: Enabled

Bit 2 **PTM0PC0**: PTP0 pin control
0: Disabled
1: Enabled

Bit 1 **CTM0PC1**: CTP0B pin control
0: Disabled
1: Enabled

Bit 0 **CTM0PC0**: CTP0 pin control
0: Disabled
1: Enabled

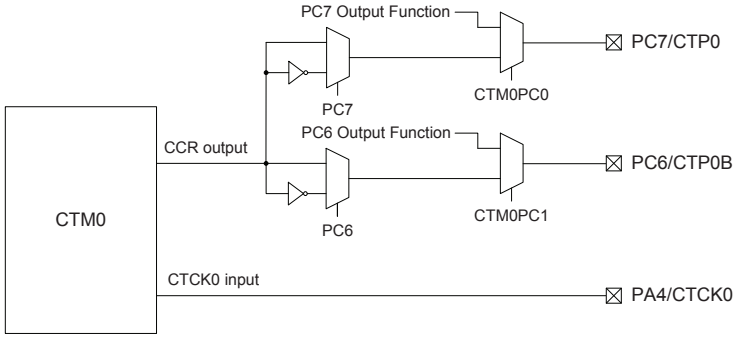
TMPC1 Register – BS87D20A-3

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	CTM1PC1	CTM1PC0
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

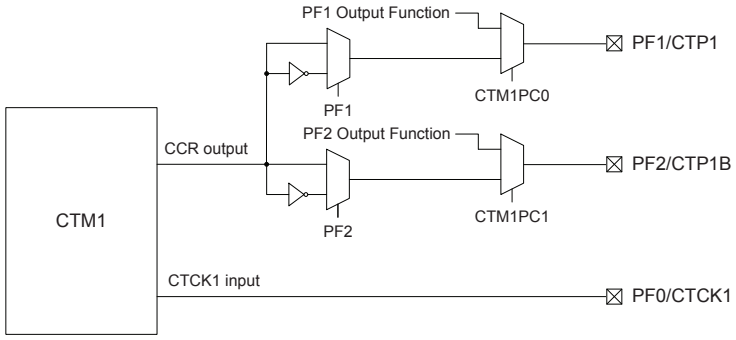
Bit 7~2 Unimplemented, read as 0.

Bit 1 **CTM1PC1**: CTP1B pin control
0: Disabled
1: Enabled

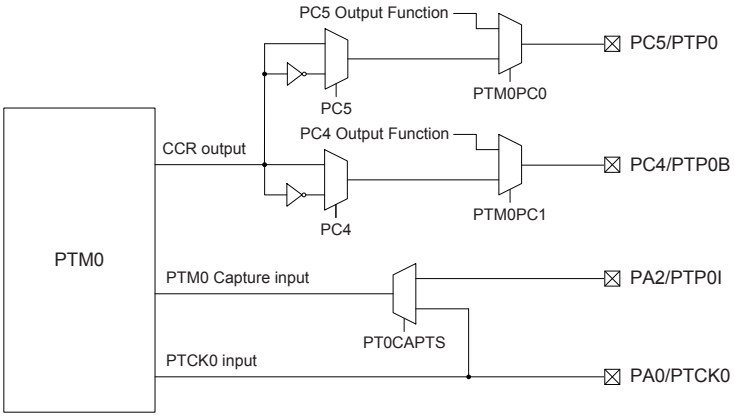
Bit 0 **CTM1PC0**: CTP1 pin control
0: Disabled
1: Enabled



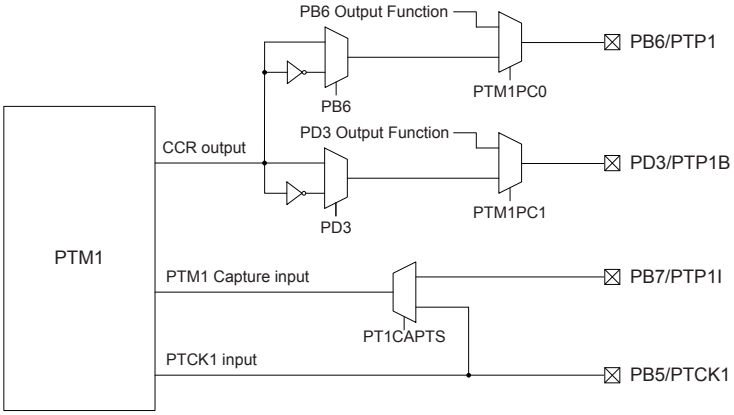
CTM0 Function Pin Control Block Diagram



CTM1 Function Pin Control Block Diagram – BS87D20A-3



PTM0 Function Pin Control Block Diagram

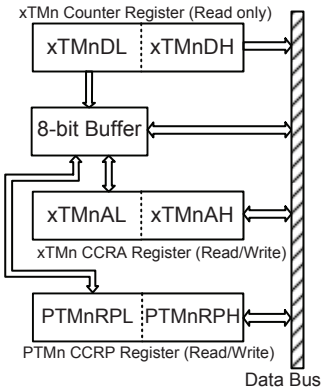


PTM1 Function Pin Control Block Diagram – BS87C16A-3 & BS87D20A-3

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



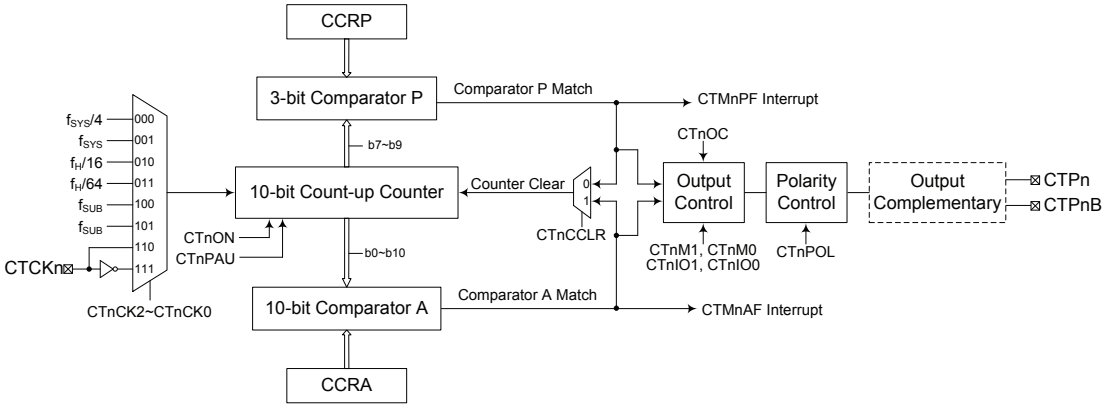
The following steps show the read and write procedures:

- Writing Data to CCRA or CCRP
  - Step 1. Write data to Low Byte xTMnAL or PTMnRPL
    – note that here data is only written to the 8-bit buffer.
  - Step 2. Write data to High Byte xTMnAH or PTMnRPH
    – here data is written directly to the high byte registers and simultaneously data is latched from the 8-bit buffer to the Low Byte registers.
- Reading Data from the Counter Registers and CCRA or CCRP
  - Step 1. Read data from the High Byte xTMnDH, xTMnAH or PTMnRPH
    – here data is read directly from the High Byte registers and simultaneously data is latched from the Low Byte register into the 8-bit buffer.
  - Step 2. Read data from the Low Byte xTMnDL, xTMnAL or PTMnRPL
    – this step reads data from the 8-bit buffer.

Compact Type TM – CTM0, CTM1

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

Device	CTM Core	CTM Input Pin	CTM Output Pin	Note
BS87B12A-3	10-bit CTM (CTM0)	CTCK0	CTP0, CTP0B	n=0
BS87C16A-3	10-bit CTM (CTM0)	CTCK0	CTP0, CTP0B	n=0
BS87D20A-3	10-bit CTM (CTM0, CTM1)	CTCK0 CTCK1	CTP0, CTP0B CTP1, CTP1B	n=0, 1



Compact Type TM Block Diagram – n=0 or 1

Compact Type TM Operation

The size of Compact TM is 10-bit wide and its core is a 10-bit count-up counter which is driven by a user selectable internal or external clock source. There are also two internal comparators with the names, Comparator A and Comparator P. These comparators will compare the value in the counter with CCRP and CCRA registers. The CCRP comparator is 3-bit wide whose value is compared with the highest 3 bits in the counter while the CCRA is the 10 bits and therefore compares all counter bits.

The only way of changing the value of the 10-bit counter using the application program, is to clear the counter by changing the CTnON bit from low to high. The counter will also be cleared automatically by a counter overflow or a compare match with one of its associated comparators. When these conditions occur, a 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 an output pin. All operating setup conditions are selected using relevant internal registers.

Compact Type TM Register Description

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

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

10-bit Compact TypeTM Registers List – n=0 or 1

CTMnDL Register

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

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

CTMnDH Register

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

Bit 7~2: Unimplemented, read as 0.

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

CTMnAL Register

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

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

CTMnAH Register

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

Bit 7~2: Unimplemented, read as 0.

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

CTMnC0 Register

Bit	7	6	5	4	3	2	1	0
Name	CTnPAU	CTnCK2	CTnCK1	CTnCK0	CTnON	CTnRP2	CTnRP1	CTnRP0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7 **CTnPAU**: CTMn Counter Pause control
0: Run
1: Pause

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

Bit 6~4 **CTnCK2~CTnCK0**: Select CTMn Counter clock
000: $f_{SYS}/4$
001: f_{SYS}
010: $f_H/16$
011: $f_H/64$
100: f_{SUB}
101: f_{SUB}
110: CTCKn rising edge clock
111: CTCKn falling edge clock

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

Bit 3 **CTnON**: CTMn Counter On/Off control
0: Off
1: On

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

Bit 2~0 **CTnRP2~CTnRP0**: CTMn CCRP 3-bit register, compared with the CTMn counter bit 9~bit 7

Comparator P match period =
0: 1024 CTMn clocks
1~7: (1~7) x 128 CTMn clocks

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

CTMnC1 Register

Bit	7	6	5	4	3	2	1	0
Name	CTnM1	CTnM0	CTnIO1	CTnIO0	CTnOC	CTnPOL	CTnDPX	CTnCCLR
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 **CTnM1~CTnM0**: Select CTMn Operating Mode
00: Compare Match Output Mode
01: Undefined
10: PWM Output Mode
11: Timer/Counter Mode

These bits setup the required operating mode for the CTMn. To ensure reliable operation the CTMn should be switched off before any changes are made to the CTnM1 and CTnM0 bits. In the Timer/Counter Mode, the CTMn output pin control will be disabled.

Bit 5~4 **CTnIO1~CTnIO0**: Select CTMn external pin CTPn function

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

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

Timer/Counter Mode
Unused

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

In the Compare Match Output Mode, the CTnIO1 and CTnIO0 bits determine how the CTMn output pin changes state when a compare match occurs from the Comparator A. The CTMn output pin can be setup to switch high, switch low or to toggle its present state when a compare match occurs from the Comparator A. When the bits are both zero, then no change will take place on the output. The initial value of the CTMn output pin should be setup using the CTnOC bit in the CTMnC1 register. Note that the output level requested by the CTnIO1 and CTnIO0 bits must be different from the initial value setup using the CTnOC bit otherwise no change will occur on the CTMn output pin when a compare match occurs. After the CTMn output pin changes state, it can be reset to its initial level by changing the level of the CTnON bit from low to high.

In the PWM output mode, the CTnIO1 and CTnIO0 bits determine how the CTMn output pin changes state when a certain compare match condition occurs. The PWM output function is modified by changing these two bits. It is necessary to only change the values of the CTnIO1 and CTnIO0 bits only after the CTMn has been switched off. Unpredictable PWM outputs will occur if the CTnIO1 and CTnIO0 bits are changed when the CTMn is running.

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

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

PWM Output Mode
0: Active low
1: Active high

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

Bit 2 **CTnPOL**: CTMn CTPn Output polarity control
0: Non-inverted
1: Inverted

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

Bit 1 **CTnDPX**: CTMn PWM duty/period control
0: CCRP – period; CCRA – duty
1: CCRP – duty; CCRA – period

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

Bit 0 **CTnCCLR**: CTMn 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 CTMn contains two comparators, Comparator A and Comparator P, either of which can be selected to clear the internal counter. With the CTnCCLR bit set high, the counter will be cleared when a compare match occurs from the Comparator A. When the bit is low, the counter will be cleared when a compare match occurs from the Comparator P or with a counter overflow. A counter overflow clearing method can only be implemented if the CCRP bits are all cleared to zero. The CTnCCLR bit is not used in the PWM Output Mode.

Compact Type TM Operation Modes

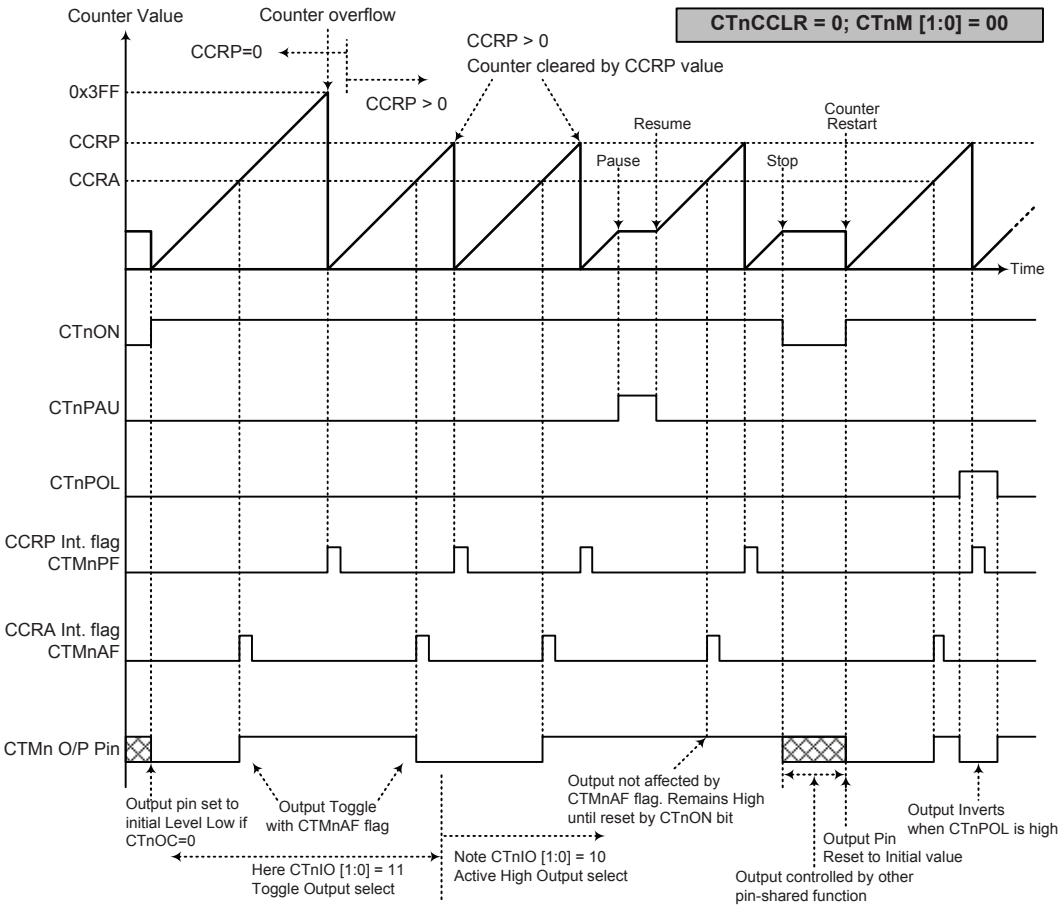
The Compact Type TM can operate in one of three operating modes, Compare Match Output Mode, PWM Output Mode or Timer/Counter Mode. The operating mode is selected using the CTnM1 and CTnM0 bits in the CTMnC1 register.

Compare Match Output Mode

To select this mode, bits CTnM1 and CTnM0 in the CTMnC1 register, should be set to 00 respectively. In this mode once the counter is enabled and running it can be cleared by three methods. These are a counter overflow, a compare match from Comparator A and a compare match from Comparator P. When the CTnCCLR bit is low, there are two ways in which the counter can be cleared. One is when a compare match from Comparator P, the other is when the CCRP bits are all zero which allows the counter to overflow. Here both CTMnAF and CTMnPF interrupt request flags for Comparator A and Comparator P respectively, will both be generated.

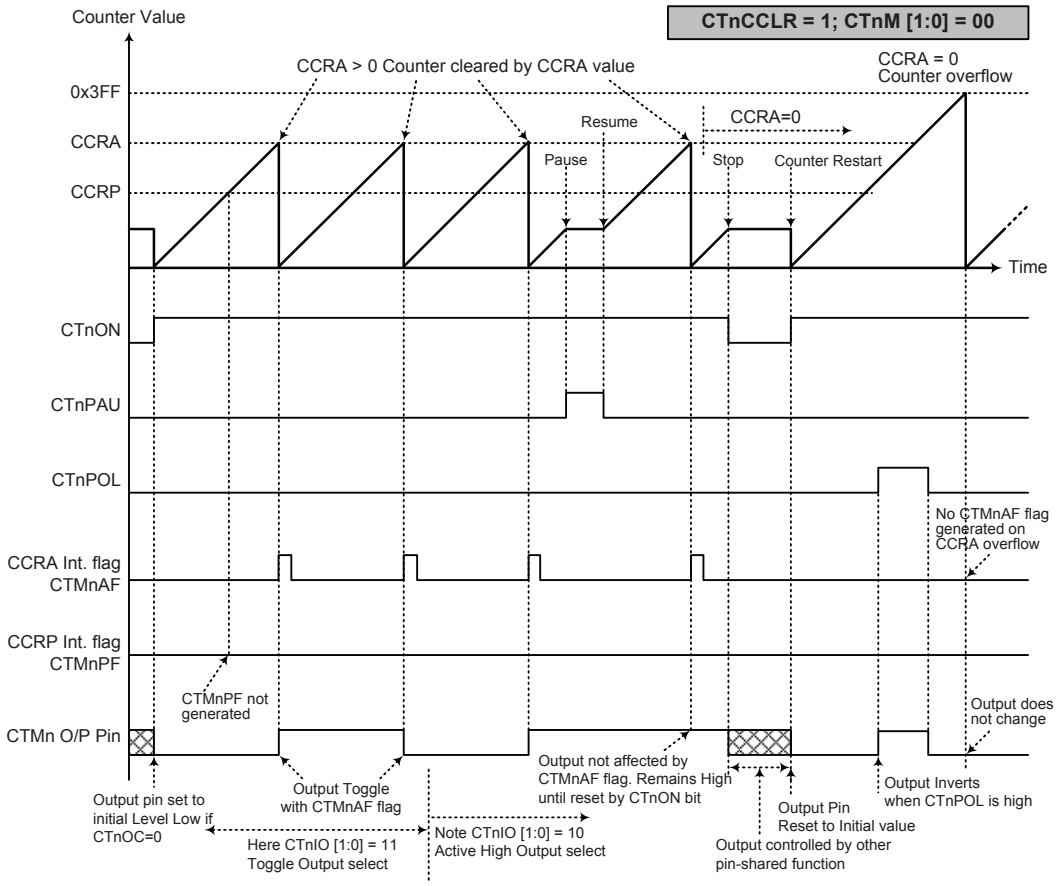
If the CTnCCLR bit in the CTMnC1 register is high then the counter will be cleared when a compare match occurs from Comparator A. However, here only the CTMnAF interrupt request flag will be generated even if the value of the CCRP bits is less than that of the CCRA registers. Therefore when CTnCCLR is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

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



Compare Match Output Mode – CTnCCLR=0

Note: 1. With CTnCCLR=0 a Comparator P match will clear the counter
2. The CTMn output pin is controlled only by the CTMnAF flag
3. The output pin is reset to its initial state by a CTnON bit rising edge
4. n=0 or 0~1 is dependent upon which device is selected



Compare Match Output Mode – CTnCCLR=1

Note: 1. With CTnCCLR=1 a Comparator A match will clear the counter
2. The CTMn output pin is controlled only by the CTMnAF flag
3. The output pin is reset to its initial state by a CTnON bit rising edge
4. A CTMnPF flag is not generated when CTnCCLR=1
5. n=0 or 0~1 is dependent upon which device is selected

Timer/Counter Mode

To select this mode, bits CTnM1 and CTnM0 in the CTMnC1 register should be set to 11 respectively. The Timer/Counter Mode operates in an identical way to the Compare Match Output Mode generating the same interrupt flags. The exception is that in the Timer/Counter Mode the CTMn output pin is not used. Therefore the above description and Timing Diagrams for the Compare Match Output Mode can be used to understand its function. As the CTMn output pin is not used in this mode, the pin can be used as a normal I/O pin or other pin-shared function.

PWM Output Mode

To select this mode, bits CTnM1 and CTnM0 in the CTMnC1 register should be set to 10 respectively. The PWM function within the CTMn is useful for applications which require functions such as motor control, heating control, illumination control, etc. By providing a signal of fixed frequency but of varying duty cycle on the CTMn output pin, a square wave AC waveform can be generated with varying equivalent DC RMS values.

As both the period and duty cycle of the PWM waveform can be controlled, the choice of generated waveform is extremely flexible. In the PWM mode, the CTnCCLR bit has no effect as the PWM period. Both of the CCRA and CCRP registers are used to generate the PWM waveform, one register is used to clear the internal counter and thus control the PWM waveform frequency, while the other one is used to control the duty cycle. Which register is used to control either frequency or duty cycle is determined using the CTnDPX bit in the CTMnC1 register. The PWM waveform frequency and duty cycle can therefore be controlled by the values in the CCRA and CCRP registers.

An interrupt flag, one for each of the CCRA and CCRP, will be generated when a compare match occurs from either Comparator A or Comparator P. The CTnOC bit in the CTMnC1 register is used to select the required polarity of the PWM waveform while the two CTnIO1 and CTnIO0 bits are used to enable the PWM output or to force the CTMn output pin to a fixed high or low level. The CTnPOL bit is used to reverse the polarity of the PWM output waveform.

- **10-bit CTMn, PWM Mode, Edge-aligned Mode, CTnDPX=0**

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

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

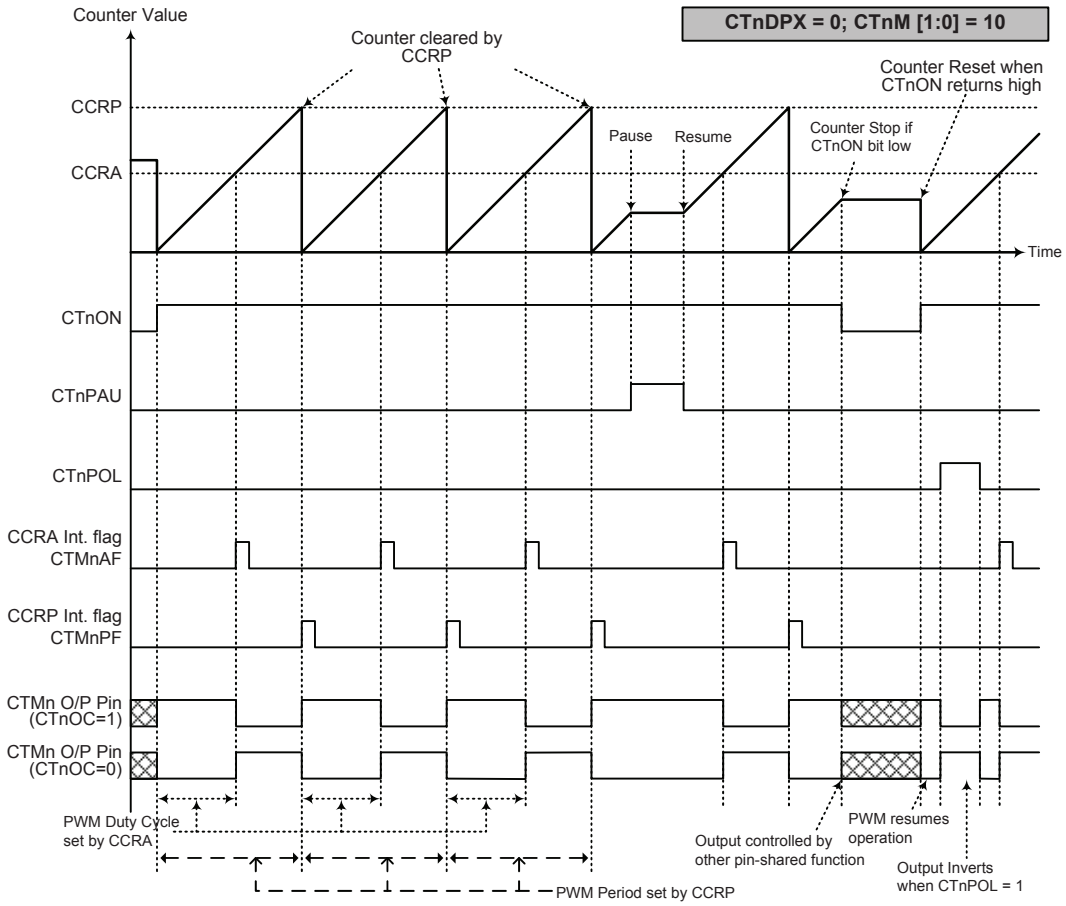
The CTMn PWM output frequency=($f_{SYS}/4$)/(4×128)=$f_{SYS}/2048$=8 kHz, duty=128/(4×128)= 25%.

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

- **10-bit CTMn, PWM Mode, Edge-aligned Mode, CTnDPX=1**

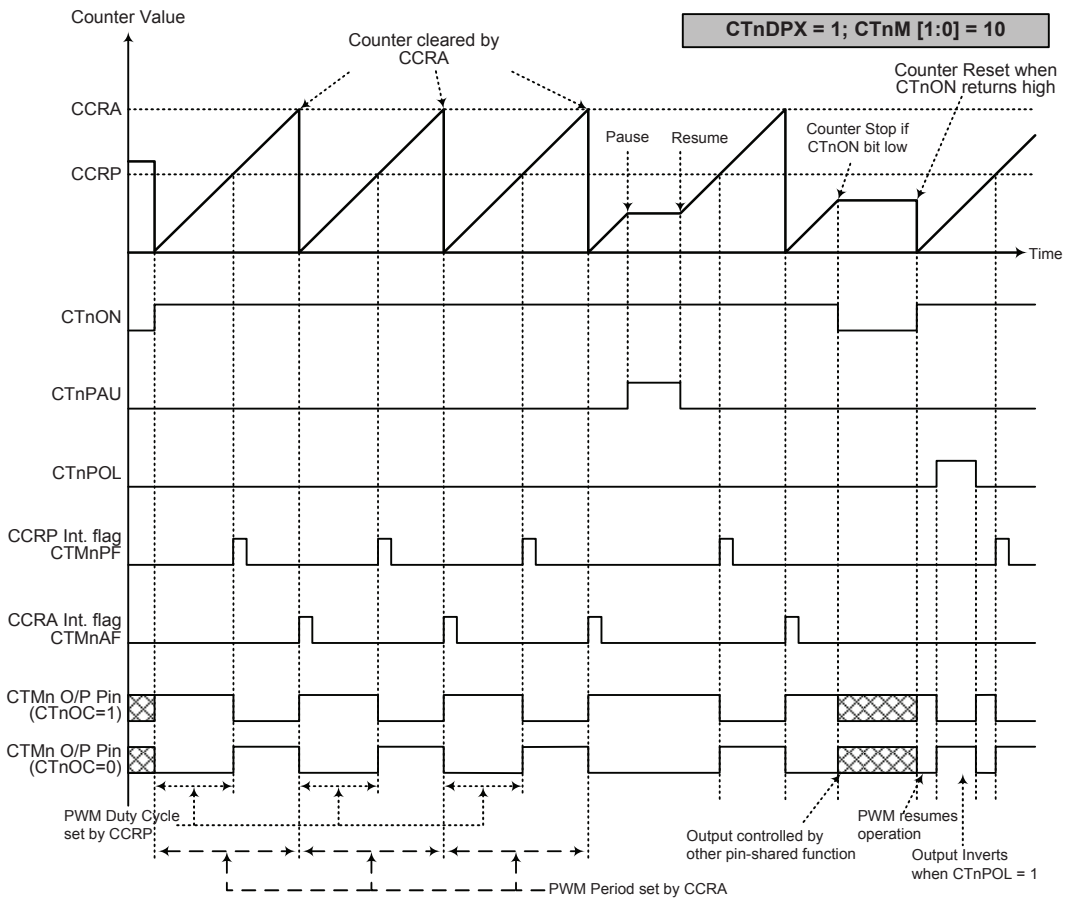
CCRP	1~7	0
Period	CCRA	
Duty	CCRP×128	1024

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



PWM Output Mode – CTnDPX=0

Note: 1. Here CTnDPX=0 – Counter cleared by CCRP
2. A counter clear sets the PWM Period
3. The internal PWM function continues running even when CTnIO [1:0]=00 or 01
4. The CTnCCLR bit has no influence on PWM operation
5. n=0 or 0~1 is dependent upon which device is selected



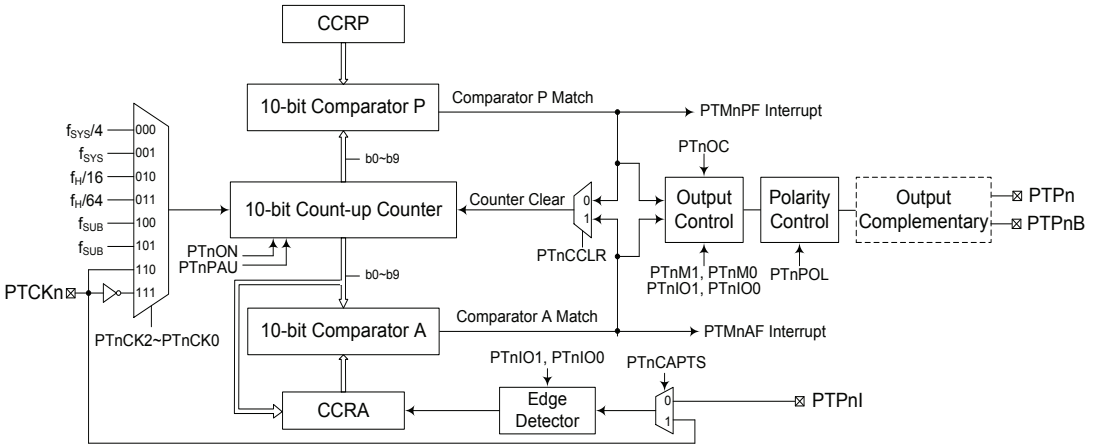
PWM Output Mode – CTnDPX=1

Note: 1. Here CTnDPX=1 – Counter cleared by CCRA
2. A counter clear sets the PWM Period
3. The internal PWM function continues even when CTnIO [1:0]=00 or 01
4. The CTnCCLR bit has no influence on PWM operation
5. n=0 or 0~1 depending upon which device is selected.

Periodic Type TM – PTM

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

Device	PTM Core	PTM Input Pin	PTM Output Pin	Note
BS87B12A-3	10-bit PTM (PTM0)	PTCK0	PTP0, PTP0B	n=0
BS87C16A-3	10-bit PTM (PTM0, PTM1)	PTCK0 PTCK1	PTP0, PTP0B PTP1, PTP1B	n=0, 1
BS87D20A-3	10-bit PTM (PTM0, PTM1)	PTCK0 PTCK1	PTP0, PTP0B PTP1, PTP1B	n=0, 1



10-bit Periodic Type TM Block Diagram – n=0 or 1

Periodic TM Operation

The size of Periodic TM is 10-bit wide and its core is a 10-bit count-up counter which is driven by a user selectable internal or external clock source. There are also two internal comparators with the names, Comparator A and Comparator P. These comparators will compare the value in the counter with CCRP and CCRA registers. The CCRP and CCRA comparators are 10-bit wide whose value is 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 PTnON bit from low to high. The counter will also be cleared automatically by a counter overflow or a compare match with one of its associated comparators. When these conditions occur, a 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 TM is controlled using a series of registers. A read only register pair exists to store the internal counter 10-bit value, while two read/write register pairs exist to store the internal 10-bit CCRA and CCRP value. The remaining two registers are control registers which setup the different operating and control modes.

Register Name	Bit							
	7	6	5	4	3	2	1	0
PTMnC0	PTnPAU	PTnCK2	PTnCK1	PTnCK0	PTnON	—	—	—
PTMnC1	PTnM1	PTnM0	PTnIO1	PTnIO0	PTnOC	PTnPOL	PTnCAPTS	PTnCCLR
PTMnDL	D7	D6	D5	D4	D3	D2	D1	D0
PTMnDH	—	—	—	—	—	—	D9	D8
PTMnAL	D7	D6	D5	D4	D3	D2	D1	D0
PTMnAH	—	—	—	—	—	—	D9	D8
PTMnRPL	PTnRP7	PTnRP6	PTnRP5	PTnRP4	PTnRP3	PTnRP2	PTnRP1	PTnRP0
PTMnRPH	—	—	—	—	—	—	PTnRP9	PTnRP8

10-bit Periodic TM Registers List – n=0 or 1

PTMnDL Register

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

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

PTMnDH Register

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

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

PTMnAL Register

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

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

PTMnAH Register

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

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

PTMnRPL Register

Bit	7	6	5	4	3	2	1	0
Name	PTnRP7	PTnRP6	PTnRP5	PTnRP4	PTnRP3	PTnRP2	PTnRP1	PTnRP0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~0 **PTnRP7~PTnRP0**: PTMn CCRP Low Byte Register bit 7 ~ bit 0
PTMn 10-bit CCRP bit 7 ~ bit 0

PTMnRPH Register

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

Bit 7~2 Unimplemented, read as "0"

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

PTMnC0 Register

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

Bit 7 **PTnPAU**: PTMn Counter Pause control
0: Run
1: Pause
The counter can be paused by setting this bit high. Clearing the bit to zero restores normal counter operation. When in a Pause condition the PTMn will remain powered up and continue to consume power. The counter will retain its residual value when this bit changes from low to high and resume counting from this value when the bit changes to a low value again.

Bit 6~4 **PTnCK2~PTnCK0**: Select PTMn Counter clock
000: $f_{SYS}/4$
001: f_{SYS}
010: $f_H/16$
011: $f_H/64$
100: f_{SUB}
101: f_{SUB}
110: PTCKn rising edge clock
111: PTCKn falling edge clock
These three bits are used to select the clock source for the PTMn. The external pin clock source can be chosen to be active on the rising or falling edge. The clock source f_{SYS} is the system clock, while f_H and f_{SUB} are other internal clocks, the details of which can be found in the oscillator section.

Bit 3 **PTnON**: PTMn Counter On/Off control
0: Off
1: On
This bit controls the overall on/off function of the PTMn. Setting the bit high enables the counter to run while clearing the bit disables the PTMn. Clearing this bit to zero will stop the counter from counting and turn off the PTMn which will reduce its power consumption. When the bit changes state from low to high the internal counter value will be reset to zero, however when the bit changes from high to low, the internal counter will retain its residual value until the bit returns high again. If the PTMn is in the Compare Match Output Mode then the PTMn output pin will be reset to its initial condition, as specified by the PTnOC bit, when the PTnON bit changes from low to high.

Bit 2~0 Unimplemented, read as "0"

PTMnC1 Register

Bit	7	6	5	4	3	2	1	0
Name	PTnM1	PTnM0	PTnIO1	PTnIO0	PTnOC	PTnPOL	PTnCAPTS	PTnCCLR
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 **PTnM1~PTnM0**: Select PTMn 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 PTMn. To ensure reliable operation the PTMn should be switched off before any changes are made to the PTnM1 and PTnM0 bits. In the Timer/Counter Mode, the PTMn output pin control will be disabled.

Bit 5~4 **PTnIO1~PTnIO0**: Select PTMn external pin PTPn or PTPnI 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 PTPnI or PTCKn
01: Input capture at falling edge of PTPnI or PTCKn
10: Input capture at rising/falling edge of PTPnI or PTCKn
11: Input capture disabled
Timer/Counter Mode
Unused
These two bits are used to determine how the PTMn output pin changes state when a certain condition is reached. The function that these bits select depends upon in which mode the PTMn is running.

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

In the PWM Mode, the PTnIO1 and PTnIO0 bits determine how the TM output pin changes state when a certain compare match condition occurs. The PTMn output function is modified by changing these two bits. It is necessary to only change the values of the PTnIO1 and PTnIO0 bits only after the PTMn has been switched off. Unpredictable PWM outputs will occur if the PTnIO1 and PTnIO0 bits are changed when the PTMn is running.

Bit 3 **PTnOC**: PTMn PTPn Output control

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

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

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

Bit 2 **PTnPOL**: PTMn PTPn Output polarity control
0: Non-inverted
1: Inverted

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

Bit 1 **PTnCAPTS**: PTMn Capture Trigger Source selection
0: From PTPnI pin
1: From PTCKn pin

Bit 0 **PTnCCLR**: PTMn Counter Clear condition selection
0: Comparator P match
1: Comparator A match

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

Periodic Type TM Operation Modes

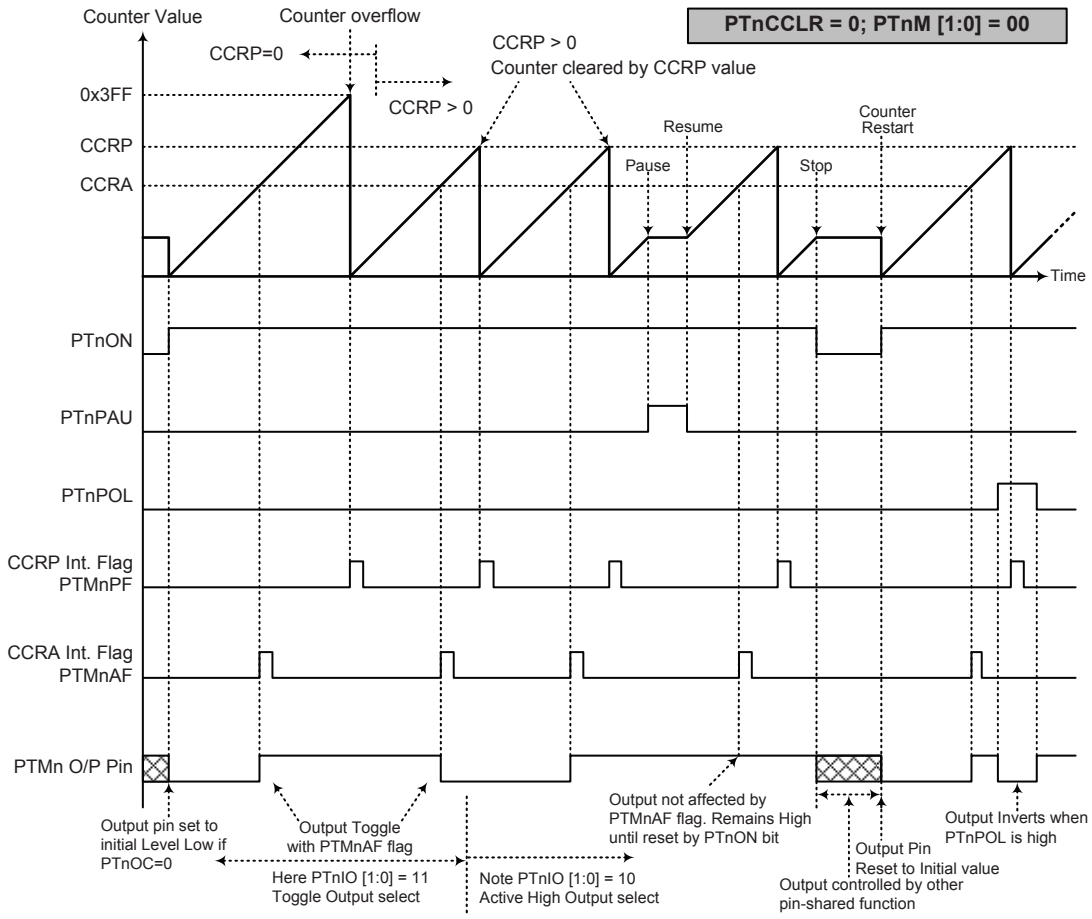
The Periodic Type TM can operate in one of five operating modes, Compare Match Output Mode, PWM Output Mode, Single Pulse Output Mode, Capture Input Mode or Timer/Counter Mode. The operating mode is selected using the PTnM1 and PTnM0 bits in the PTMnC1 register.

Compare Match Output Mode

To select this mode, bits PTnM1 and PTnM0 in the PTMnC1 register, should be set to 00 respectively. In this mode once the counter is enabled and running it can be cleared by three methods. These are a counter overflow, a compare match from Comparator A and a compare match from Comparator P. When the PTnCCLR bit is low, there are two ways in which the counter can be cleared. One is when a compare match from Comparator P, the other is when the CCRP bits are all zero which allows the counter to overflow. Here both PTMnAF and PTMnPF interrupt request flags for Comparator A and Comparator P respectively, will both be generated.

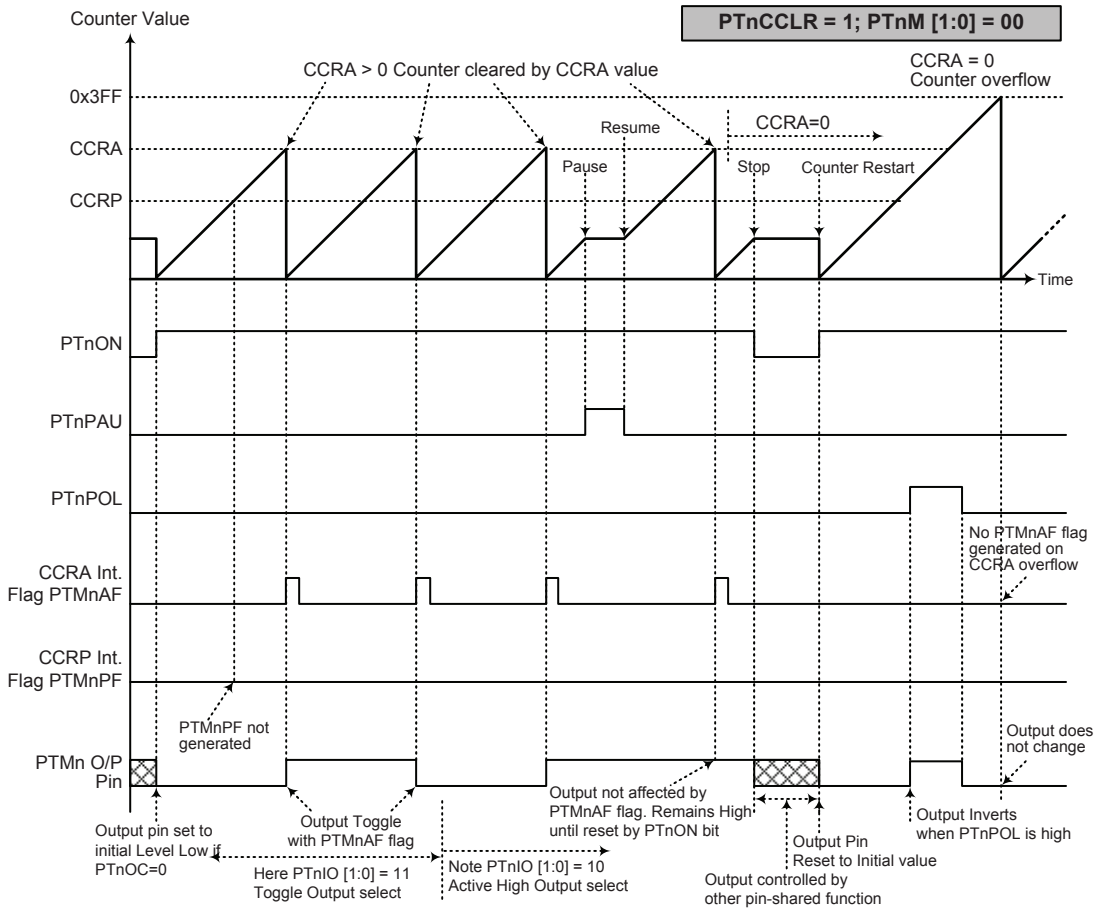
If the PTnCCLR bit in the PTMnC1 register is high then the counter will be cleared when a compare match occurs from Comparator A. However, here only the PTMnAF interrupt request flag will be generated even if the value of the CCRP bits is less than that of the CCRA registers. Therefore when PTnCCLR is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

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



Compare Match Output Mode – PTnCCLR=0

Note: 1. With PTnCCLR=0, a Comparator P match will clear the counter
2. The PTMn output pin is controlled only by the PTMnAF flag
3. The output pin is reset to its initial state by a PTnON bit rising edge
4. The 10-bit PTM maximum counter value is 0x3FF
5. n=0 or 0~1 is dependent upon which device is selected



Compare Match Output Mode – PTnCCLR=1

Note: 1. With PTnCCLR=1, a Comparator A match will clear the counter
2. The PTMn output pin is controlled only by the PTMnAF flag
3. The output pin is reset to its initial state by a PTnON bit rising edge
4. A PTMnPF flag is not generated when PTnCCLR =1
5. The 10-bit PTM maximum counter value is 0x3FF
6. n=0 or 0~1 is dependent upon which device is selected

Timer/Counter Mode

To select this mode, bits PTnM1 and PTnM0 in the PTMnC1 register should be set to 11 respectively. The Timer/Counter Mode operates in an identical way to the Compare Match Output Mode generating the same interrupt flags. The exception is that in the Timer/Counter Mode the PTMn output pin is not used. Therefore the above description and Timing Diagrams for the Compare Match Output Mode can be used to understand its function. As the PTMn output pin is not used in this mode, the pin can be used as a normal I/O pin or other pin-shared function.

PWM Output Mode

To select this mode, bits PTnM1 and PTnM0 in the PTMnC1 register should be set to 10 respectively and also the PTnIO1 and PTnIO0 bits should be set to 10 respectively. The PWM function within the PTMn is useful for applications which require functions such as motor control, heating control, illumination control, etc. By providing a signal of fixed frequency but of varying duty cycle on the PTMn output pin, a square wave AC waveform can be generated with varying equivalent DC RMS values.

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

An interrupt flag, one for each of the CCRA and CCRP, will be generated when a compare match occurs from either Comparator A or Comparator P. The PTnOC bit in the PTMnC1 register is used to select the required polarity of the PWM waveform while the two PTnIO1 and PTnIO0 bits are used to enable the PWM output or to force the PTMn output pin to a fixed high or low level. The PTnPOL bit is used to reverse the polarity of the PWM output waveform.

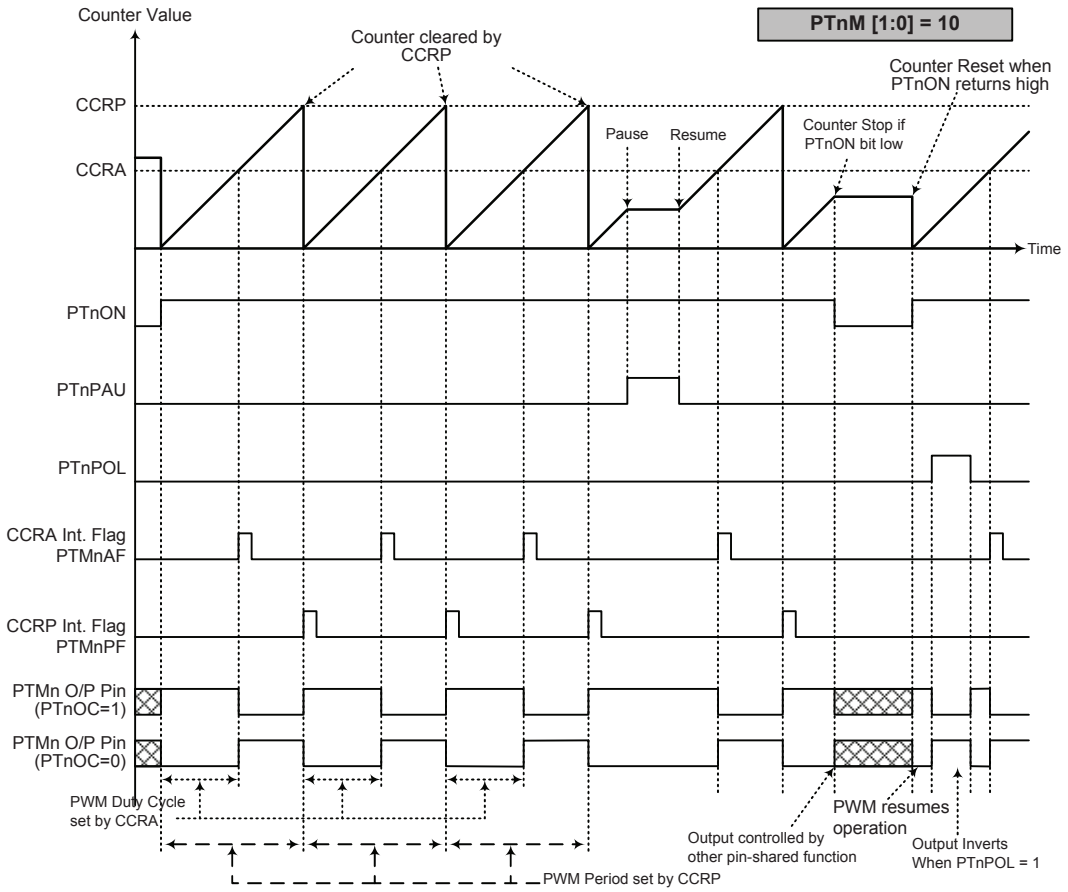
- **10-bit PTMn, PWM Output Mode**

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

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

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

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



PWM Mode

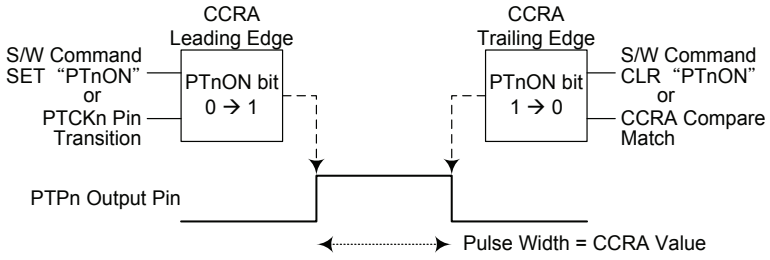
Note: 1. The counter is cleared by CCRP
2. A counter clear sets the PWM Period
3. The internal PWM function continues running even when PTnIO [1:0]=00 or 01
4. The PTnCCLR bit has no influence on PWM operation
5. n=0 or 0~1 is dependent upon which device is selected

Single Pulse Output Mode

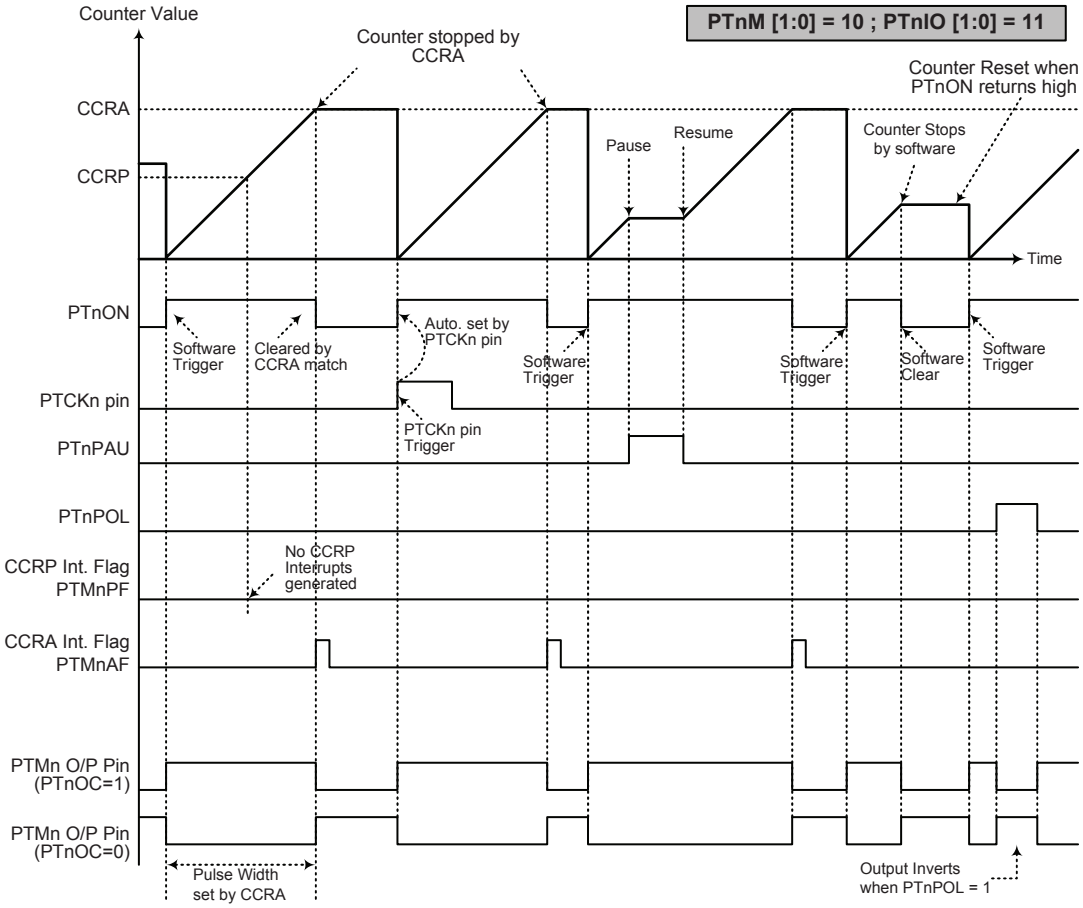
To select this mode, bits PTnM1 and PTnM0 in the PTMnC1 register should be set to 10 respectively and also the PTnIO1 and PTnIO0 bits should be set to 11 respectively. The Single Pulse Output Mode, as the name suggests, will generate a single shot pulse on the PTMn output pin.

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

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



Single Pulse Generation – n=0 or 1



Single Pulse Mode

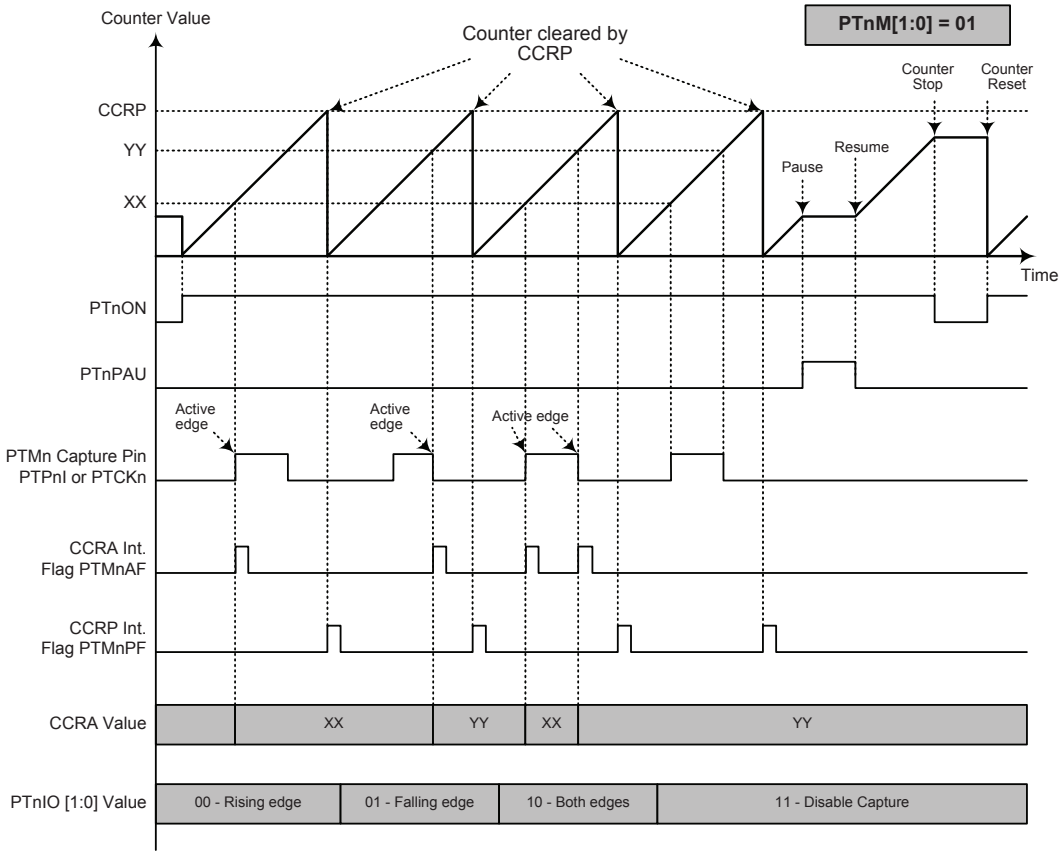
Note: 1. Counter stopped by CCRA
2. CCRP is not used
3. The pulse triggered by the PTCKn pin or by setting the PTnON bit high
4. A PTCKn pin active edge will automatically set the PTnON bit high
5. In the Single Pulse Mode, PTnIO [1:0] must be set to "11" and can not be changed
6. n=0 or 0~1 is dependent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

Capture Input Mode

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

When the required edge transition appears on the PTPnI or PTCKn pin the present value in the counter will be latched into the CCRA registers and a PTMn interrupt generated. Irrespective of what events occur on the PTPnI or PTCKn pin the counter will continue to free run until the PTnON bit changes from high to low. When a CCRP compare match occurs the counter will reset back to zero; in this way the CCRP value can be used to control the maximum counter value. When a CCRP compare match occurs from Comparator P, a PTMn interrupt will also be generated. Counting the number of overflow interrupt signals from the CCRP can be a useful method in measuring long pulse widths. The PTnIO1 and PTnIO0 bits can select the active trigger edge on the PTPnI or PTCKn pin to be a rising edge, falling edge or both edge types. If the PTnIO1 and PTnIO0 bits are both set high, then no capture operation will take place irrespective of what happens on the PTPnI or PTCKn pin, however it must be noted that the counter will continue to run.

As the PTPnI or PTCKn pin is pin shared with other functions, care must be taken if the PTMn 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 PTnCCLR, PTnOC and PTnPOL bits are not used in this Mode.



Capture Input Mode

Note: 1. PTnM [1:0]=01 and active edge set by the PTnIO [1:0] bits
2. A PTMn Capture input pin active edge transfers the counter value to CCRA
3. PTnCCLR bit not used
4. No output function – PTnOC and PTnPOL bits are not used
5. CCRP determines the counter value and the counter has a maximum count value when CCRP is equal to zero
6. n=0 or 0~1 is dependent upon which device is selected

Analog to Digital Converter

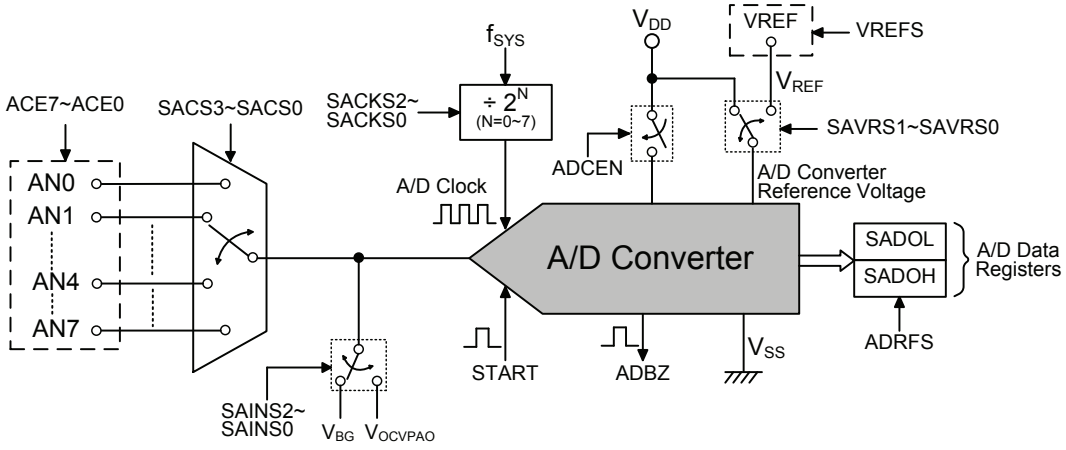
The need to interface to real world analog signals is a common requirement for many electronic systems. However, to properly process these signals by a microcontroller, they must first be converted into digital signals by A/D converters. By integrating the A/D conversion electronic circuitry into the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

A/D Overview

These devices contain a multi-channel analog to digital converter which can directly interface to external analog signals, such as that from sensors or other control signals and convert these signals directly into a 12-bit digital value. It also can convert the internal signals, such as the internal reference voltage, into a 12-bit digital value. The external or internal analog signal to be converted is determined by the SAINS and SACS bit fields. When the external analog signal is to be converted, the corresponding external channel input pin function should first be properly configured and then the desired external channel input should be selected using the SAINS and SACS fields. Note that when the internal analog signal is selected to be converted, all external channel analog input pin function should first be deselected and then properly configured the SAINS and SACS fields. More detailed information about the A/D input signal selection will be described in the "A/D converter Control Registers" and "A/D Converter Input Signals" section respectively.

Device	External Input Channels	Internal Signal	A/D Signal Select
BS87B12A-3 BS87C16A-3 BS87D20A-3	AN0~AN7	V_{BG}, V_{OCVPAO}	SAINS2~SAINS0 SACS3~SACS0

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



A/D Converter Structure

Registers Descriptions

Overall operation of the A/D converter is controlled using five registers. A read only register pair exists to store the A/D Converter data 12-bit value. The remaining three registers, SADC0, SADC1 and ACERL, are control registers which setup the operating conditions and control function of the A/D converter.

Register Name	Bit							
	7	6	5	4	3	2	1	0
SADOL (ADRFS=0)	D3	D2	D1	D0	—	—	—	—
SADOL (ADRFS=1)	D7	D6	D5	D4	D3	D2	D1	D0
S ADOH (ADRFS=0)	D11	D10	D9	D8	D7	D6	D5	D4
SADOH (ADRFS=1)	—	—	—	—	D11	D10	D9	D8
SADC0	START	ADBZ	ADCEN	ADRFS	SACS3	SACS2	SACS1	SACS0
SADC1	SAINS2	SAINS1	SAINS0	SAVRS1	SAVRS0	SACKS2	SACKS1	SACKS0
ACERL	ACE7	ACE6	ACE5	ACE4	ACE3	ACE2	ACE1	ACE0

A/D Converter Registers List

A/D Converter Data Registers – SADOL, SADOH

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

ADRFS	SADOH								SADOL							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
0	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	0	0	0	0
1	0	0	0	0	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

A/D Converter Data Registers

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

To control the function and operation of the A/D converter, three control registers known as SADC0, SADC1 and ACERL are provided. These 8-bit registers define functions such as the selection of which analog signal is connected to the internal A/D converter, the digitised data format, the A/D clock source as well as controlling the start function and monitoring the A/D converter busy status. As these devices contain only one actual analog to digital converter hardware circuit, each of the external and internal analog signals must be routed to the converter. The SAINS field in the SADC1 register and SACS field in the SADC0 register are used to determine which analog signal derived from the external or internal signals will be connected to the A/D converter.

The ACERL register contains the ACE7~ACE0 bits which determine which pins on I/O Ports are used as analog inputs for the A/D converter input and which pins are not. Setting the corresponding bit high will select the A/D input function while clearing the bit to zero will select either the I/O or other pin-shared function. When the pin is selected to be an A/D input, its original function whether it is an I/O or other pin-shared function will be removed. In addition, any internal pull-high resistor connected to the pin will be automatically removed if the pin is selected to be an A/D converter input.

- **SADC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	START	ADBZ	ADCEN	ADRFS	SACS3	SACS2	SACS1	SACS0
R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

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

Bit 6 **ADBZ**: A/D Converter busy flag
0: No A/D conversion is in progress
1: A/D conversion is in progress
This read only flag is used to indicate whether the A/D conversion is in progress or not. When the START bit is set from low to high and then to low again, the ADBZ flag will be set to 1 to indicate that the A/D conversion is initiated. The ADBZ flag will be cleared to 0 after the A/D conversion is complete.

Bit 5 **ADCEN**: A/D Converter function enable control
0: Disable
1: Enable
This bit controls the A/D internal function. This bit should be set to one to enable the A/D converter. If the bit is set low, then the A/D converter will be switched off reducing the device power consumption. When the A/D converter function is disabled, the contents of the A/D data register pair known as SADOH and SADOL will be unchanged.

Bit 4 **ADRFS**: A/D conversion data format select
0: A/D converter data format → SADOH=D [11:4]; SADOL=D [3:0]
1: A/D converter data format → SADOH=D [11:8]; SADOL=D [7:0]
This bit controls the format of the 12-bit converted A/D value in the two A/D data registers. Details are provided in the A/D converter data register section.

Bit 3~0 **SACS3~SACS0**: A/D converter external analog input channel select
0000: External AN0 input
0001: External AN1 input
0010: External AN2 input
0011: External AN3 input
0100: External AN4 input
0101: External AN5 input
0110: External AN6 input
0111: External AN7 input
1xxx: Non-existed channel, input floating if selected.
These bits are used to select which external analog input channel is to be converted. When the external analog input channel is selected, the SAINS bit field must be set to "000", "101" or "11x". Details are summarized in the "A/D Converter Input Signal Selection" table.

- **SADC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	SAINS2	SAINS1	SAINS0	SAVRS1	SAVRS0	SACKS2	SACKS1	SACKS0
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 7~5 **SAINS2~SAINS0**: A/D converter input signal select
000: External source – External analog channel intput, ANn
001: Internal source – Internal signal derived from the internal bandgap reference voltage, V_{BG}
010: Internal source – Reserved, connected to ground
011: Internal source – Internal signal derived from the OCVP OPA output signal, V_{OCVPAO}
100: Internal source – Reserved, connected to ground
101~111: External source – External analog channel intput, ANn

Care must be taken if the SAINS field is set to "001" or "011" to select the internal analog signal to be converted. When the internal analog signal is selected to be converted, the external channel input pin must never be selected as the A/D input signal by properly setting the SACS bit field with a value of "1xxx". Otherwise, the external channel input will be connected together with the internal analog signal. This will result in unpredictable situations such as an irreversible damage.

Bit 4~3 **SAVRS1~SAVRS0**: A/D converter reference voltage select
00: External VREF pin
01: Internal A/D converter power, V_{DD}.
1x: External VREF pin

These bits are used to select the A/D converter reference voltage source. Care must be taken if the SAVRS field is set to "01" to select the internal A/D converter power voltage as the reference voltage source. When the internal reference voltage source is selected, the external VREF cannot be configured as the reference voltage input by properly configuring the VREFS bit in the TMPC0 register. Otherwise, the external input voltage on the VREF pin will be connected to the internal A/D converter power.

Bit 2~0 **SACKS2~SACKS0**: A/D conversion clock source select
000: f_{SYS}
001: $f_{SYS}/2$
010: $f_{SYS}/4$
011: $f_{SYS}/8$
100: $f_{SYS}/16$
101: $f_{SYS}/32$
110: $f_{SYS}/64$
111: $f_{SYS}/128$

These bits are used to select the clock source for the A/D converter. It is recommended that the A/D conversion clock frequency should be in the range from 500 kHz to 1MHz by properly configuring the SACKS2~SACKS0 bits.

- **ACERL Register**

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

Bit 7 **ACE7**: AN7 input pin enable control
0: Disable – not A/D input
1: Enable – A/D input, AN7

Bit 6 **ACE6**: AN6 input pin enable control
0: Disable – not A/D input
1: Enable – A/D input, AN6

Bit 5 **ACE5**: AN5 input pin enable control
0: Disable – not A/D input
1: Enable – A/D input, AN5

Bit 4 **ACE4**: AN4 input pin enable control
0: Disable – not A/D input
1: Enable – A/D input, AN4

Bit 3 **ACE3**: AN3 input pin enable control
0: Disable – not A/D input
1: Enable – A/D input, AN3

Bit 2 **ACE2**: AN2 input pin enable control
0: Disable – not A/D input
1: Enable – A/D input, AN2

Bit 1 **ACE1**: AN1 input pin enable control
0: Disable – not A/D input
1: Enable – A/D input, AN1

Bit 0 **ACE0**: AN0 input pin enable control
0: Disable – not A/D input
1: Enable – A/D input, AN0

A/D Converter Reference Voltage

The actual reference voltage supply to the A/D Converter can be supplied from the positive power supply pin, V_{DD}, or an external reference source supplied on pin VREF determined by the SAVRS bit field in the SADC1 register. When the SAVRS field is set to "01", the A/D converter reference voltage will come from the VDD pin. Otherwise, the A/D converter reference voltage will come from the VREF pin if the SAVRS field is set to any other value except "01". When the VREF pin is selected as the reference voltage supply pin, the VREFS bit in the TMPC0 register should first be properly configured to enable the VREF pin function as it is pin-shared with other functions. However, if the internal A/D converter power is selected as the reference voltage, the external VREF pin must not be configured as the reference voltage input function to avoid the internal connection between the VREF pin to the internal A/D converter power. Note that the analog input values must not be allowed to exceed the value of the selected reference voltage, V_{DD} or V_{REF}.

A/D Converter Input Signals

All of the external A/D analog input pins are pin-shared with the I/O pins as well as other functions. The corresponding pin-shared function selection bits ACEn in the ACERL register determine whether the external input pins are setup as A/D converter analog channel inputs or whether they have other functions. If the corresponding pin is setup to be an A/D converter analog channel input by setting the ACEn bit high, the original pin function will be disabled. In this way, pins can be changed under program control to change their function between A/D inputs and other functions. All pull-high resistors, which are setup through register programming, will be automatically disconnected if the pins are setup as A/D inputs. Note that it is not necessary to first setup the A/D pin as an input in the port control register to enable the A/D input as when the relevant A/D input function selection bits enable an A/D input, the status of the port control register will be overridden.

As these devices contain only one actual analog to digital converter hardware circuit, one of the external or internal analog signals must be routed to the converter. The SAINS2~SAINS0 bits in the SADC1 register are used to determine that the analog signal to be converted comes from the external channel input or internal analog signal. The SACS3~SACS0 bits in the SADC0 register are used to determine which external channel input is selected to be converted. If the SAINS2~SAINS0 bits are set to "000", "101" or "11x", the external channel input will be selected to be converted and the SACS3~SACS0 bits can determine which external channel is selected.

When the SAINS field is set to the value of "001" or "011", the internal analog signal derived from the Bandgap reference voltage or OCVP OPA output signal will be selected. If the internal analog signal is selected to be converted, the external channel signal input must be switched off by setting the SACS field to a value of "1xxx". Otherwise, the internal analog signal will be connected together with the external channel input. This will result in unpredictable situations.

SAINS [2:0]	SACS [3:0]	Input Signals	Description
000, 101, 11x	0000~0111	AN0~AN7	External channel analog input ANn.
	1xxx	—	Floating, no external channel is selected.
001	1xxx	V_{BG}	Internal Bandgap reference voltage
010, 100	1xxx	GND	Connected to the ground.
011	1xxx	V_{OCVPAO}	Internal OCVP OPA output signal

A/D Converter Input Signal Selection

A/D Converter Operation

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

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

The clock source for the A/D converter, which originates from the system clock f_{SYS}, can be chosen to be either f_{SYS} or a subdivided version of f_{SYS}. The division ratio value is determined by the SACKS bit field in the SADC1 register. Although the A/D clock source is determined by the system clock f_{SYS} and by bits SACKS2~SACKS0, there are some limitations on the maximum A/D clock source speed that can be selected. As the recommended range of permissible A/D clock period, t_{ADCK}, is from 0.5μs to 10μ, care must be taken for system clock frequencies. For example, if the system clock operates at a frequency of 8MHz, the SACKS2~SACKS0 bits should not be set to 000, 001 or 111. Doing so will give A/D clock periods that are less than the minimum A/D clock period which may result in inaccurate A/D conversion values. Refer to the following table for examples, where values marked with an asterisk * show where, depending upon the device, special care must be taken, as the values may be less than the specified minimum A/D Clock Period.

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

A/D Clock Period Examples

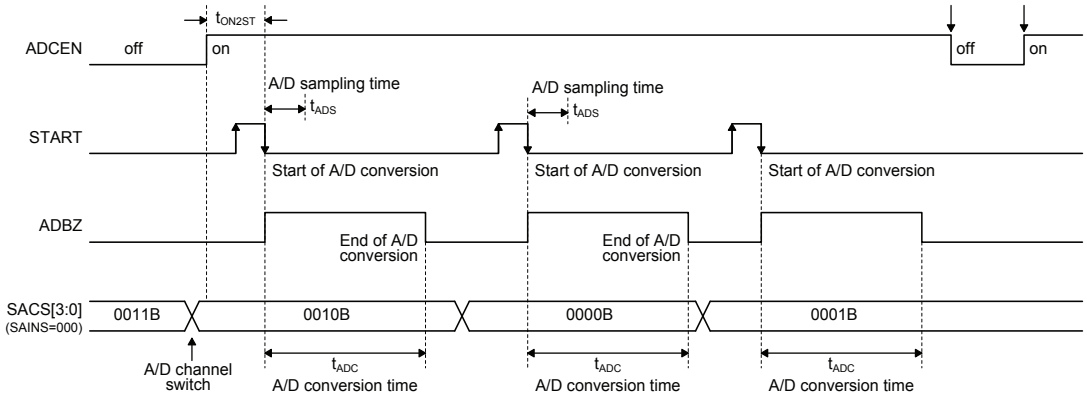
Controlling the power on/off function of the A/D converter circuitry is implemented using the ADCEN bit in the SADC0 register. This bit must be set high to power on the A/D converter. When the ADCEN bit is set high to power on the A/D converter internal circuitry, a certain delay, as indicated in the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

Conversion Rate and Timing Diagram

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

Maximum single A/D conversion rate=A/D clock period/16

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



A/D Conversion Timing – External Channel Input

Summary of A/D Conversion Steps

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

- Step 1
  Select the required A/D conversion clock by properly programming the SACKS2~SACKS0 bits in the SADC1 register.
- Step 2
  Enable the A/D converter by setting the ADCEN bit in the SADC0 register to one.
- Step 3
  Select which signal is to be connected to the internal A/D converter by correctly configuring the SACS and SAINS bit fields
  Selecting the external channel input to be converted, go to Step 4.
  Selecting the internal analog signal to be converted, go to Step 5.
- Step 4
  If the SAINS field is 000, 101 or 11x, the external channel input can be selected. The desired external channel input is selected by configuring the SACS field. When the A/D input signal comes from the external channel input, the corresponding pin should be configured as an A/D input function by selecting the relevant pin-shared function control bits, ACEn. Then go to Step 6.
- Step 5
  If the SAINS field is set to 001 or 011, the relevant internal analog signal can be selected. Before the A/D input signal is selected to come from the internal analog signal by properly configuring the SAINS field, the SACS field must be set to "1xxx" to select a non-existed channel input. Then go to Step 6.
- Step 6
  Select the A/D converter output data format by configuring the ADRFS bit.
- Step 7
  Select the A/D converter reference voltage source by configuring the SAVRS bit field.
  Select the PGA input signal and the desired PGA gain if the PGA output voltage, V_R, is selected as the A/D converter reference voltage.
- Step 8
  If A/D conversion interrupt is used, the interrupt control registers must be correctly configured to ensure the A/D interrupt function is active. The master interrupt control bit, EMI, and the A/D conversion interrupt control bit, ADE, must both be set high in advance.
- Step 9
  The A/D conversion procedure can now be initialized by setting the START bit from low to high and then low again.
- Step 10
  If A/D conversion is in progress, the ADBZ flag will be set high. After the A/D conversion process is complete, the ADBZ flag will go low and then the output data can be read from SADOH and SADOL registers.

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

Programming Considerations

During microcontroller operations where the A/D converter is not being used, the A/D internal circuitry can be switched off to reduce power consumption, by setting bit ADCEN low in the SADC0 register. When this happens, the internal A/D converter circuits will not consume power irrespective of what analog voltage is applied to their input lines. If the A/D converter input lines are used as normal I/Os, then care must be taken as if the input voltage is not at a valid logic level, then this may lead to some increase in power consumption.

A/D Transfer Function

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

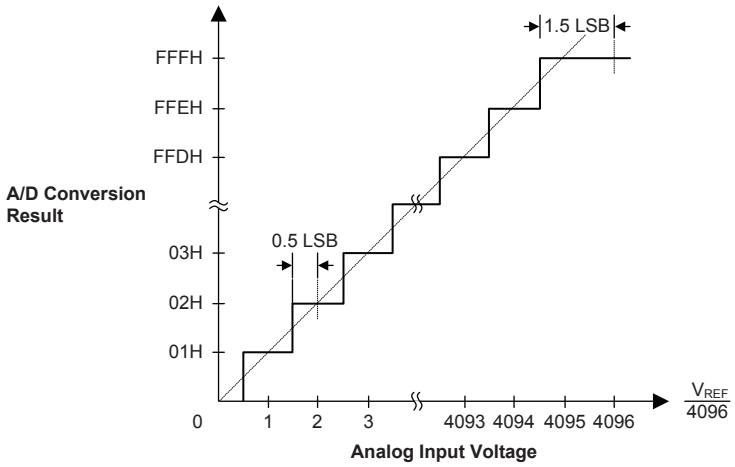
$$1\ LSB=V_{REF} \div 4096$$

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

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

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

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



Ideal A/D Transfer Function

A/D Programming Examples

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

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

```
clr ADE                ; disable ADC interrupt
mov a,03H              ; select fSYS/8 as A/D clock and A/D input
mov SADC1,a            ; signal comes from external channel
mov a,00H              ; select VREF pin voltage as A/D reference voltage source
mov SADC0,a            ; and AN0 is selected as the external channel input
set ADCEN              ; enable the A/D converter
mov a,01H              ; setup ACERL to configure pin AN0 as analog input
mov ACERL,a
:
start_conversion:
clr START              ; high pulse on start bit to initiate conversion
set START              ; reset A/D
clr START              ; start A/D
:
polling_EOC:
sz  ADBZ               ; poll the SADC0 register ADBZ bit to detect end of A/D conversion
jmp polling_EOC        ; continue polling
mov a,SADOL            ; read low byte conversion result value
mov SADOL_buffer,a     ; save result to user defined register
mov a,SADOH            ; read high byte conversion result value
mov SADOH_buffer,a     ; save result to user defined register
:
jmp start_conversion   ; start next A/D conversion
```

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

```
clr ADE                ; disable ADC interrupt
mov a,03H              ; select fSYS/8 as A/D clock and A/D input
mov SADC1,a            ; signal comes from external channel
mov a,00H              ; select VREF pin voltage as A/D reference voltage source
mov SADC0,a            ; and AN0 is selected as the external channel input
set ADCEN              ; enable the A/D converter
mov a,01H              ; setup ACERL to configure pin AN0 as analog input
mov ACERL,a
:
Start_conversion:
clr START              ; high pulse on START bit to initiate conversion
set START              ; reset A/D
clr START              ; start A/D
clr ADF                ; clear ADC interrupt request flag
set ADE                ; enable ADC interrupt
set EMI                ; enable global interrupt
:
:
ADC_ISR:               ; ADC interrupt service routine
mov acc_stack,a        ; save ACC to user defined memory
mov a,STATUS
mov status_stack,a     ; save STATUS to user defined memory
:
mov a,SADOL            ; read low byte conversion result value
mov SADOL_buffer,a     ; save result to user defined register
mov a,SADOH            ; read high byte conversion result value
mov SADOH_buffer,a     ; save result to user defined register
:
EXIT_INT_ISR:
mov a,status_stack
mov STATUS,a           ; restore STATUS from user defined memory
mov a,acc_stack        ; restore ACC from user defined memory
reti
```

Touch Key Function

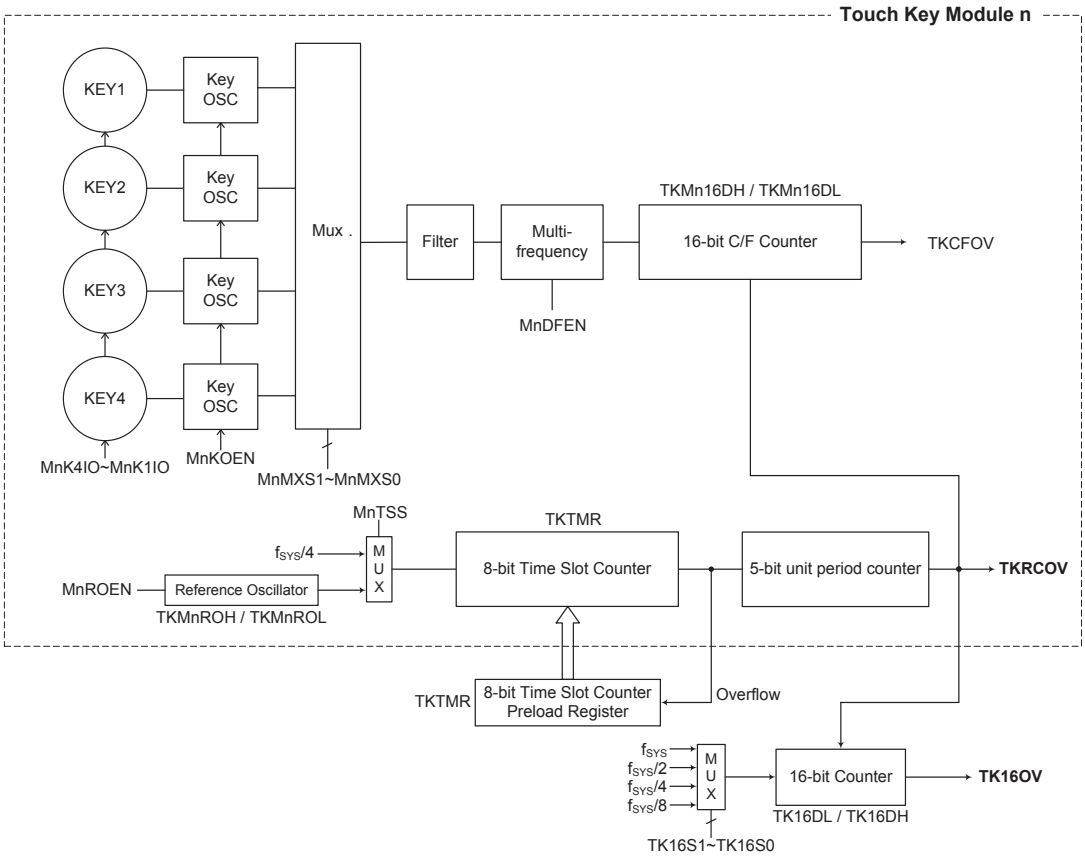
Each device provides multiple touch key functions. The touch key function is fully integrated and requires no external components, allowing touch key functions to be implemented by the simple manipulation of internal registers.

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
BS87B12A-3	12	Mn (n=0~2)	M0	KEY1~KEY4
			M1	KEY5~KEY8
			M2	KEY9~KEY12
BS87C16A-3	16	Mn (n=0~3)	M0	KEY1~KEY4
			M1	KEY5~KEY8
			M2	KEY9~KEY12
			M3	KEY13~KEY16
BS87D20A-3	20	Mn (n=0~4)	M0	KEY1~KEY4
			M1	KEY5~KEY8
			M2	KEY9~KEY12
			M3	KEY13~KEY16
			M4	KEY17~KEY20

Touch Key Structure



Note: The structure contained in the dash line is identical for each touch key module which contains four touch keys.

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 five Touch Key Modules dependent upon the selected device.

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

Touch Key Module Registers List

Register Name	Bit							
	7	6	5	4	3	2	1	0
TKTMR	D7	D6	D5	D4	D3	D2	D1	D0
TKC0	—	TKRCOV	TKST	TKCFOV	TK16OV	TSCS	TK16S1	TK16S0
TK16DL	D7	D6	D5	D4	D3	D2	D1	D0
TK16DH	D15	D14	D13	D12	D11	D10	D9	D8
TKC1	—	—	—	—	—	—	TKFS1	TKFS0
TKMn16DL	D7	D6	D5	D4	D3	D2	D1	D0
TKMn16DH	D15	D14	D13	D12	D11	D10	D9	D8
TKMnROL	D7	D6	D5	D4	D3	D2	D1	D0
TKMnROH	—	—	—	—	—	—	D9	D8
TKMnC0	MnMXS1	MnMXS0	MnDFEN	D4	MnSOFC	MnSOF2	MnSOF1	MnSOF0
TKMnC1	MnTSS	—	MnROEN	MnKOEN	MnK4IO	MnK3IO	MnK2IO	MnK1IO

Touch Key Function Registers 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])×32t_{TSC}, where t_{TSC} is the time slot counter clock period.

TKC0 Register

Bit	7	6	5	4	3	2	1	0
Name	—	TKRCOV	TKST	TKCFOV	TK16OV	TSCS	TK16S1	TK16S0
R/W	—	R/W	R/W	R/W	R/W	R/W	R/W	R
POR	—	0	0	0	0	0	0	0

Bit 7 Unimplemented, read as "0"

Bit 6 **TKRCOV**: Touch key time slot counter overflow flag

0: No overflow occurs

1: Overflow occurs

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

Bit 5 **TKST**: Touch key detection Start control

0: Stopped or no operation

0→1: Start detection

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

Bit 4 **TKCFOV**: Touch key module 16-bit C/F counter overflow flag
0: No overflow occurs
1: Overflow occurs
This bit is set high by the touch key module 16-bit C/F counter overflow and must be cleared to 0 by application programs.

Bit 3 **TK16OV**: Touch key function 16-bit counter overflow flag
0: No overflow occurs
1: Overflow occurs
This bit is set high by the touch key function 16-bit counter overflow and must be cleared to 0 by application programs.

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

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

TKC1 Register

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

Bit 7~2 Unimplemented, read as "0"

Bit 1~0 **TKFS1~TKFS0**: Touch Key oscillator and Reference oscillator frequency select
00: 500 kHz
01: 1000 kHz
10: 1500 kHz
11: 2000 kHz

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

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

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

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

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

This register pair is used to store the touch key module n 16-bit C/F counter value. This 16-bit C/F counter will be stopped and the counter content will be kept unchanged when the touch key time slot counter overflows. This register pair will be cleared to zero when the TKST bit is set low.

TKMnROH/TKMnROL – Touch Key Module n Reference Oscillator Capacitor Select Register Pair

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

This register pair is used to store the touch key module n reference oscillator capacitor value. 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	MnMXS1	MnMXS0	MnDFEN	D4	MnSOFC	MnSOF2	MnSOF1	MnSOF0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 **MnMXS1~MnMXS0**: Multiplexer Key Select

Bit	Touch Key Module Number				
MnMXS[1:0]	M0	M1	M2	M3	M4
00	KEY1	KEY5	KEY9	KEY13	KEY17
01	KEY2	KEY6	KEY10	KEY14	KEY18
10	KEY3	KEY7	KEY11	KEY15	KEY19
11	KEY4	KEY8	KEY12	KEY16	KEY20
BS87B12A-3	√	√	√	—	—
BS87C16A-3	√	√	√	√	—
BS87D20A-3	√	√	√	√	√

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 **D4**: Data bit for test only
The bit is used for test purpose only and must be kept as "0" for normal operations.

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: f_{HOP0} – Min. hopping frequency
001: f_{HOP1}
010: f_{HOP2}
011: f_{HOP3}
100: f_{HOP4} – Selected touch key oscillator frequency
101: f_{HOP5}
110: f_{HOP6}
111: f_{HOP7} – Max. hopping frequency
These bits are used to select the touch key oscillator frequency for the hopping function. Note that these bits are only available when the MnSOFC bit is cleared to 0.

TKMnC1 Register

Bit	7	6	5	4	3	2	1	0
Name	MnTSS	—	MnROEN	MnKOEN	MnK4IO	MnK3IO	MnK2IO	MnK1IO
R/W	R/W	—	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	—	0	0	0	0	0	0

Bit 7 **MnTSS**: Touch key module n time slot counter clock source select
0: Touch key module n reference oscillator
1: $f_{SYS}/4$

Bit 6 Unimplemented, read as "0"

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

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

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

MnK4IO	Touch Key Module n – Mn				
	M0	M1	M2	M3	M4
0: Disable	I/O or other functions				
1: Enable	KEY4	KEY8	KEY12	KEY16	KEY20
BS87B12A-3	√	√	√	—	—
BS87C16A-3	√	√	√	√	—
BS87D20A-3	√	√	√	√	√

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

MnK3IO	Touch Key Module n – Mn				
	M0	M1	M2	M3	M4
0: Disable	I/O or other functions				
1: Enable	KEY3	KEY7	KEY11	KEY15	KEY19
BS87B12A-3	√	√	√	—	—
BS87C16A-3	√	√	√	√	—
BS87D20A-3	√	√	√	√	√

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

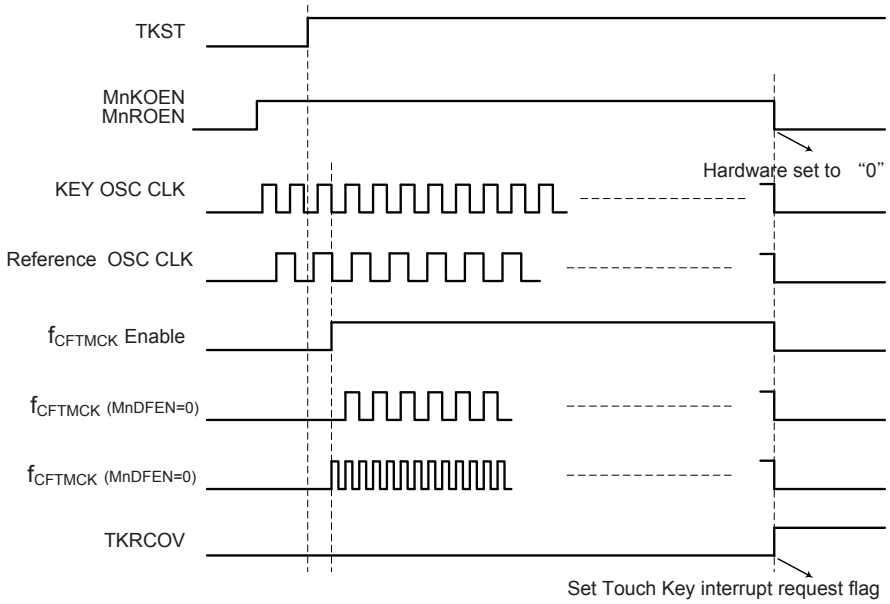
MnK2IO	Touch Key Module n – Mn				
	M0	M1	M2	M3	M4
0: Disable	I/O or other functions				
1: Enable	KEY2	KEY6	KEY10	KEY14	KEY18
BS87B12A-3	√	√	√	—	—
BS87C16A-3	√	√	√	√	—
BS87D20A-3	√	√	√	√	√

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

MnK1IO	Touch Key Module n – Mn				
	M0	M1	M2	M3	M4
0: Disable	I/O or other functions				
1: Enable	KEY1	KEY5	KEY9	KEY13	KEY17
BS87B12A-3	√	√	√	—	—
BS87C16A-3	√	√	√	√	—
BS87D20A-3	√	√	√	√	√

Touch Key Operation

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



Touch Key Scan Mode Timing Diagram

Each touch key module contains four touch key inputs which are shared with logical I/O pins, and the desired function is selected using register bits. Each touch key has its own independent sense oscillator. Therefore, there are four sense oscillators within each touch key module.

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

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

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

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

Each touch key module consists of four touch keys, KEY1 ~ KEY4 are contained in module 0, KEY5 ~ KEY8 are contained in module 1, KEY9 ~ KEY12 are contained in module 2, etc. Each touch key module has an identical structure.

Touch Key Interrupt

The touch key only has single interrupt, when the time slot counter in all the touch key modules or in the touch key module 0 overflows, an actual touch key interrupt will take place. The touch keys mentioned here are the keys which are enabled. The 16-bit C/F counter, 16-bit counter, 5-bit time slot unit period counter and 8-bit time slot counter in all modules will be automatically cleared.

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

The TK16OV flag which is the 16-bit counter overflow flag will go high when the 16-bit counter overflows. As this flag will not be automatically cleared, it has to be cleared by the application program. More details regarding the touch key interrupt is located in the interrupt section of the datasheet.

Programming Considerations

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

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

Serial Interface Module – SIM

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

It is suggested that the user should not enter the power down mode by executing the "HALT" instruction during the SIM communication in progress.

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 devices can be either master or slave. Although the SPI interface specification can control multiple slave devices from a single master, these devices provided only one $\overline{SCS}$ pin. If the master needs to control multiple slave devices from a single master, the master can use I/O pin to select the slave devices.

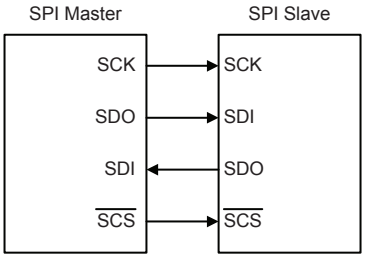
SPI Interface Operation

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

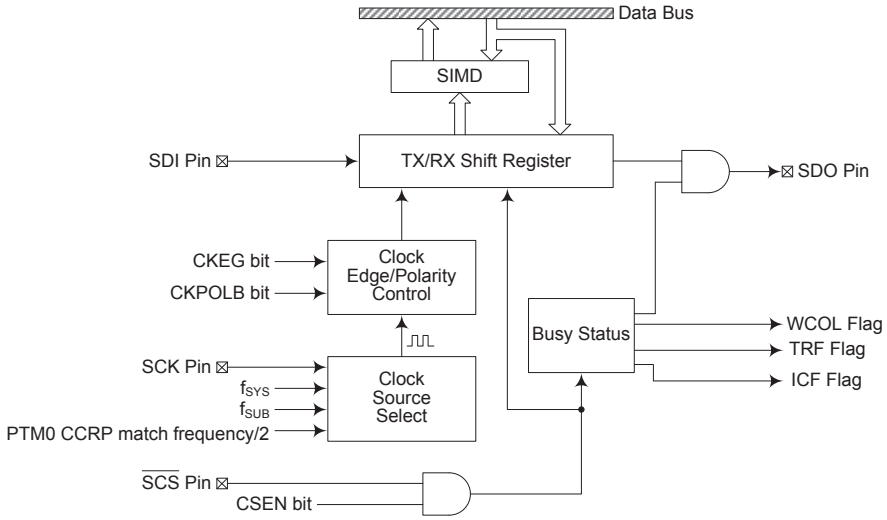
The SPI function in this device offers the following features:

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

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



SPI Master/Slave Connection



SPI Block Diagram

SPI Registers

There are three internal registers which control the overall operation of the SPI interface. These are the SIMD data register and two registers SIMC0 and SIMC2. Note that the SIMC1 register is only used by the I²C interface.

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

SPI Registers List

- **SIMD Register**

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

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	×	×	×	×	×	×	×	×

"×": unknown

There are also two control registers for the SPI interface, SIMC0 and SIMC2. Note that the SIMC2 register also has the name SIMA which is used by the I²C function. The SIMC1 register is not used by the SPI function, only by the I²C function. Register SIMC0 is used to control the enable/disable function and to set the data transmission clock frequency. Register SIMC2 is used for other control functions such as LSB/MSB selection, write collision flag, etc.

- **SIMC0 Register**

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

Bit 7~5 **SIM2~SIM0**: SIM Operating Mode Control
000: SPI master mode; SPI clock is $f_{SYS}/4$
001: SPI master mode; SPI clock is $f_{SYS}/16$
010: SPI master mode; SPI clock is $f_{SYS}/64$
011: SPI master mode; SPI clock is f_{SUB}
100: SPI master mode; SPI clock is PTM0 CCRP match frequency/2
101: SPI slave mode
110: I²C slave mode
111: Non SIM function
These bits setup the overall operating mode of the SIM function. As well as selecting if the I²C or SPI function, they are used to control the SPI Master/Slave selection and the SPI Master clock frequency. The SPI clock is a function of the system clock but can also be chosen to be sourced from PTM0. If the SPI Slave Mode is selected then the clock will be supplied by an external Master device.

Bit 4 Unimplemented, read as "0"

Bit 3~2 **SIMDEB1~SIMDEB0**: I²C Debounce Time Selection
The SIMDEB1~SIMDEB0 bits are only used in the I²C mode and the detailed definition is described in the I²C interface section.

Bit 1 **SIMEN**: SIM Enable Control
0: Disable
1: Enable
The bit is the overall on/off control for the SIM interface. When the SIMEN bit is cleared to zero to disable the SIM interface, the SDI, SDO, SCK and $\overline{SCS}$, or SDA and SCL lines will lose their SPI or I^2C function and the SIM operating current will be reduced to a minimum value. When the bit is high the SIM interface is enabled. The SIM configuration option must have first enabled the SIM interface for this bit to be effective. If the SIM is configured to operate as an SPI interface via the SIM2~SIM0 bits, the contents of the SPI control registers will remain at the previous settings when the SIMEN bit changes from low to high and should therefore be first initialised by the application program. If the SIM is configured to operate as an I^2C interface via the SIM2~SIM0 bits and the SIMEN bit changes from low to high, the contents of the I^2C control bits such as HTX and TXAK will remain at the previous settings and should therefore be first initialised by the application program while the relevant I^2C flags such as HCF, HAAS, HBB, SRW and RXAK will be set to their default states.

Bit 0 **SIMICF**: SIM SPI slave mode Incomplete Transfer Flag
0: SIM SPI slave mode incomplete condition not occurred
1: SIM SPI slave mode incomplete condition occurred
This bit is only available when the SIM is configured to operate in an SPI slave mode. If the SPI operates in the slave mode with the SIMEN and CSEN bits both being set to 1 but the $\overline{SCS}$ line is pulled high by the external master device before the SPI data transfer is completely finished, the SIMICF bit will be set to 1 together with the TRF bit. When this condition occurs, the corresponding interrupt will occur if the interrupt function is enabled. However, the TRF bit will not be set to 1 if the SIMICF bit is set to 1 by software application program.

- **SIMC2 Register**

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

Bit 7~6 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 I/O pin or other pin-shared functions. If the bit is high, the $\overline{SCS}$ pin will be enabled and used as a select pin.

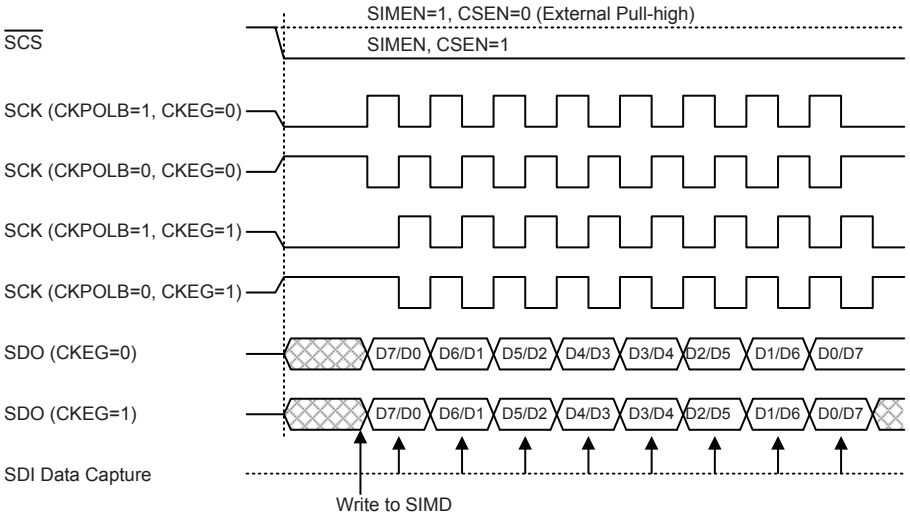
Bit 1 **WCOL**: SPI write collision flag
0: No collision
1: Collision
The WCOL flag is used to detect whether a data collision has occurred or not. If this bit is high, it means that data has been attempted to be written to the SIMD register during a data transfer operation. This writing operation will be ignored if data is being transferred. This bit can be cleared by the application program.

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

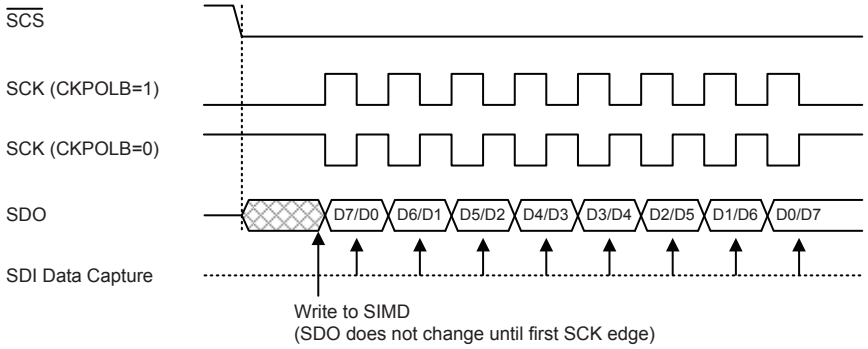
SPI Communication

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

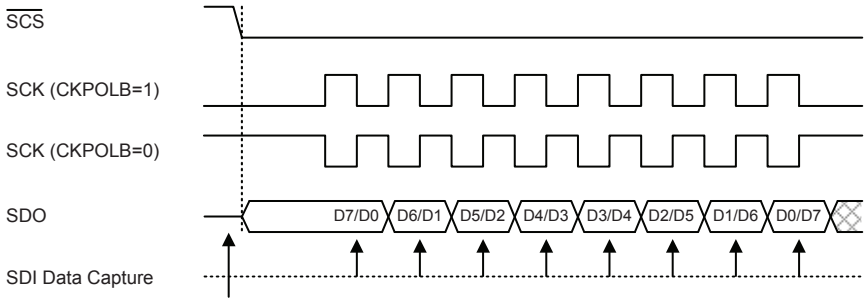
The SPI will continue to function in specific IDLE Mode if the clock source used by the SPI interface is still active.



SPI Master Mode Timing

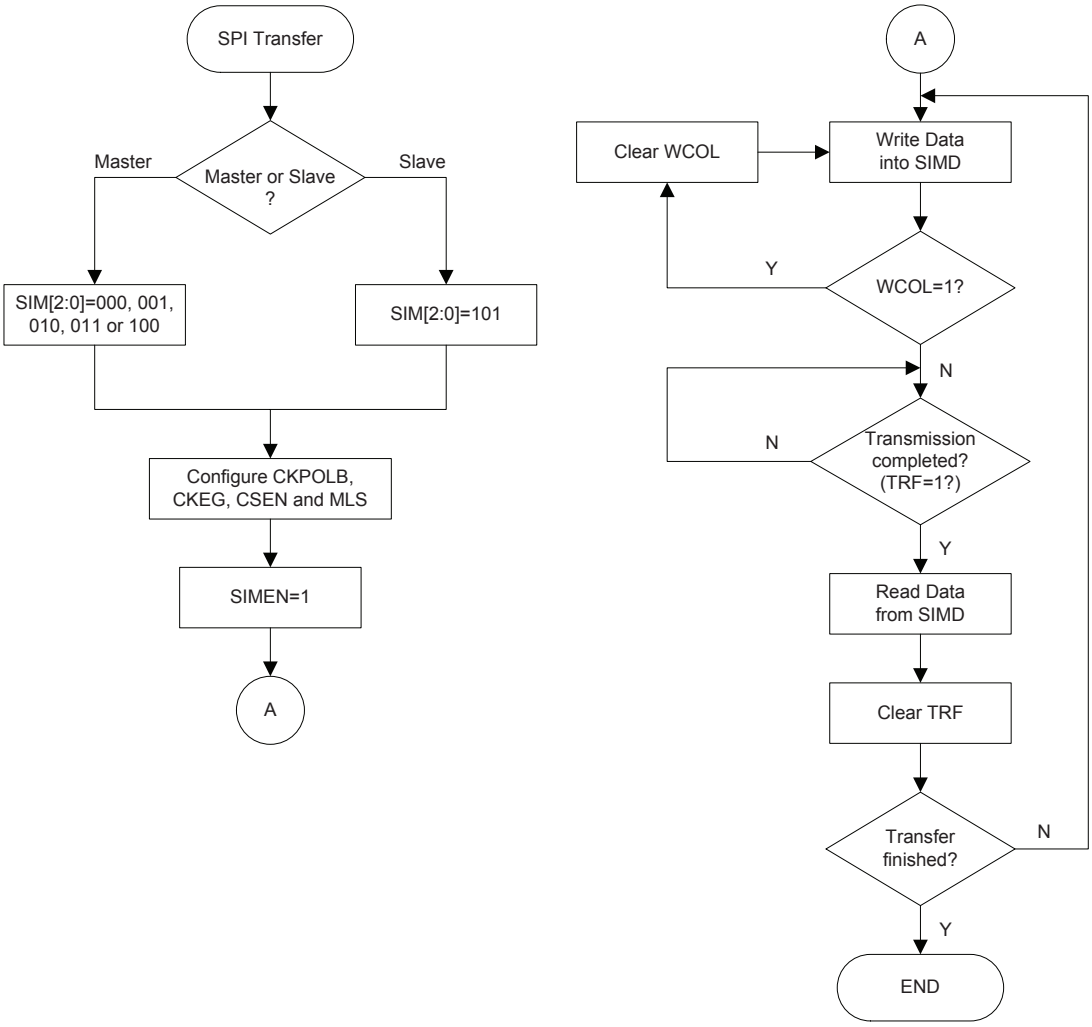


SPI Slave Mode Timing – CKEG=0



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

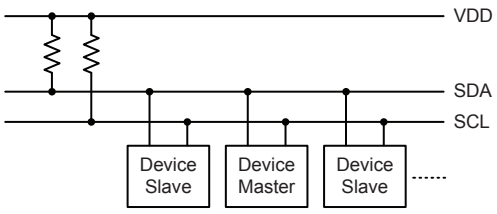
SPI Slave Mode Timing – CKEG=1



SPI Transfer Control Flow Chart

I²C Interface

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

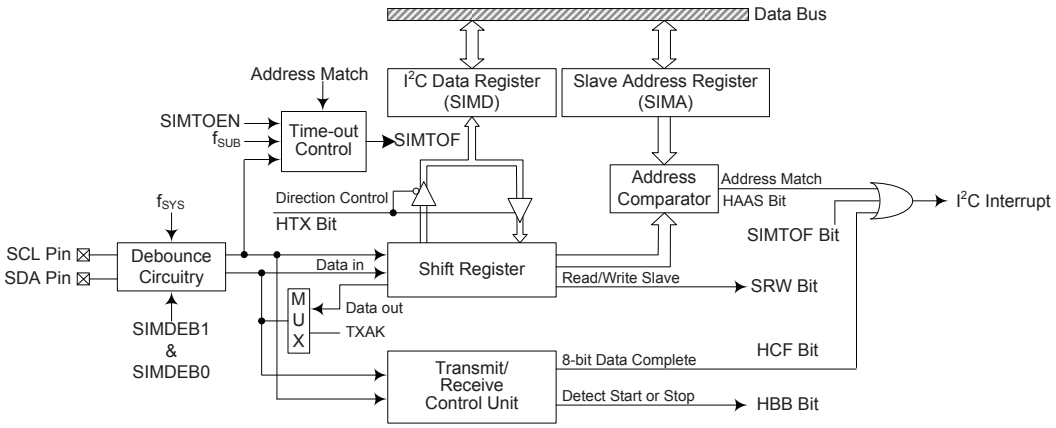


I²C Master/Slave Bus Connection

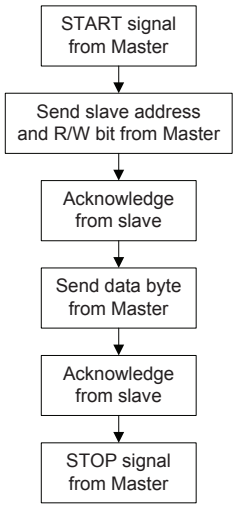
I²C interface Operation

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

When two devices communicate with each other on the bidirectional I²C bus, one is known as the master device and one as the slave device. Both master and slave can transmit and receive data; however, it is the master device that has overall control of the bus. For these devices, which only operate 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 the SCL or SDA pin is still applicable even if the I²C interface is activated and the related internal pull-up resistor could be controlled by its corresponding pull-up control register.



I²C Block Diagram



The SIMDEB1 and SIMDEB0 bits determine the debounce time of the I²C interface. This uses the system clock to in effect add a debounce time to the external clock to reduce the possibility of glitches on the clock line causing erroneous operation. The debounce time, if selected, can be chosen to be either 2 or 4 system clocks. To achieve the required 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 Devounce	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

I²C Registers

There are three control registers associated with the I²C bus, SIMC0, SIMC1 and SIMTOC, one slave address register, SIMA, and one data register, SIMD. Any transmission or reception of data from the I²C bus must be made via the SIMD register. Bit SIMEN and bits SIM2~SIM0 in register SIMC0 are used by the I²C interface as well as the SPI interface. The SIMTOC register is used for the I²C time-out control.

Note that the SIMA register also has the name SIMC2 which is used by the SPI function.

Register Name	Bit							
	7	6	5	4	3	2	1	0
SIMC0	SIM2	SIM1	SIM0	—	SIMDEB1	SIMDEB0	SIMEN	SIMICF
SIMC1	HCF	HAAS	HBB	HTX	TXAK	SRW	IAMWU	RXAK
SIMA	IICA6	IICA5	IICA4	IICA3	IICA2	IICA1	IICA0	—
SIMD	D7	D6	D5	D4	D3	D2	D1	D0
SIMTOC	SIMTOEN	SIMTOF	SIMTOS5	SIMTOS4	SIMTOS3	SIMTOS2	SIMTOS1	SIMTOS0

I²C Registers List

- **SIMD 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.

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	×	×	×	×	×	×	×	×

"×": unknown

- **SIMA 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 implemented.

When a master device, which is connected to the I²C bus, sends out an address, which matches the slave address in the SIMA register, the slave device will be selected. Note that the SIMA register is the same register address as SIMC2 which is used by the SPI interface.

Bit	7	6	5	4	3	2	1	0
Name	IICA6	IICA5	IICA4	IICA3	IICA2	IICA1	IICA0	—
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	—
POR	×	×	×	×	×	×	×	—

"×": unknown

Bit 7~1 **IICA6~IICA0**: I²C slave address
IICA6~IICA0 is the I²C slave address bit 6 ~ bit 0

Bit 0 Unimplemented, read as "0".

There are also two control registers for the I²C interface, SIMC0 and SIMC1. The register SIMC0 is used to control the enable/disable function and to set the data transmission clock frequency. The SIMC1 register contains the relevant flags which are used to indicate the I²C communication status.

- **SIMC0 Register**

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

Bit 7~5 **SIM2~SIM0**: SIM Operating Mode Control
000: SPI master mode; SPI clock is f_{SYS}/4
001: SPI master mode; SPI clock is f_{SYS}/16
010: SPI master mode; SPI clock is f_{SYS}/64
011: SPI master mode; SPI clock is f_{SUB}
100: SPI master mode; SPI clock is PTM0 CCRP match frequency/2
101: SPI slave mode
110: I²C slave mode
111: Non SIM function

These bits setup the overall operating mode of the SIM function. As well as selecting if the I²C or SPI function, they are used to control the SPI Master/Slave selection and the SPI Master clock frequency. The SPI clock is a function of the system clock but can also be chosen to be sourced from PTM0. If the SPI Slave Mode is selected then the clock will be supplied by an external Master device.

Bit 4 Unimplemented, read as "0"

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

These bits are used to select the I²C debounce time when the SIM is configured as the I²C interface function by setting the SIM2~SIM0 bits to "110".

Bit 1 **SIMEN**: SIM Enable Control
0: Disable
1: Enable
The bit is the overall on/off control for the SIM interface. When the SIMEN bit is cleared to zero to disable the SIM interface, the SDI, SDO, SCK and $\overline{SCS}$, or SDA and SCL lines will lose their SPI or I^2C function and the SIM operating current will be reduced to a minimum value. When the bit is high the SIM interface is enabled. The SIM configuration option must have first enabled the SIM interface for this bit to be effective.If the SIM is configured to operate as an SPI interface via the SIM2~SIM0 bits, the contents of the SPI control registers will remain at the previous settings when the SIMEN bit changes from low to high and should therefore be first initialised by the application program. If the SIM is configured to operate as an I^2C interface via the SIM2~SIM0 bits and the SIMEN bit changes from low to high, the contents of the I^2C control bits such as HTX and TXAK will remain at the previous settings and should therefore be first initialised by the application program while the relevant I^2C flags such as HCF, HAAS, HBB, SRW and RXAK will be set to their default states.

Bit 0 **SIMICF**: SIM SPI Incomplete Flag
The SIMICF bit is only used in the SPI mode and the detailed definition is described in the SPI interface section.

- **SIMC1 Register**

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

Bit 7 **HCF**: I^2C Bus data transfer completion flag
0: Data is being transferred
1: Completion of an 8-bit data transfer
The HCF flag is the data transfer flag. This flag will be zero when data is being transferred. Upon completion of an 8-bit data transfer the flag will go high and an interrupt will be generated.

Bit 6 **HAAS**: I^2C Bus data transfer completion flag
0: Not address match
1: Address match
The HAAS flag is the address match flag. This flag is used to determine if the slave device address is the same as the master transmit address. If the addresses match then this bit will be high, if there is no match then the flag will be low.

Bit 5 **HBB**: I^2C Bus busy flag
0: I^2C Bus is not busy
1: I^2C Bus is busy
The HBB flag is the I^2C busy flag. This flag will be "1" when the I^2C bus is busy which will occur when a START signal is detected. The flag will be set to "0" when the bus is free which will occur when a STOP signal is detected.

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

Bit 3 **TXAK**: I^2C bus transmit acknowledge flag
0: Slave send acknowledge flag
1: Slave does not send acknowledge flag
The TXAK bit is the transmit acknowledge flag. After the slave device receipt of 8-bits of data, this bit will be transmitted to the bus on the 9^{th} 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.

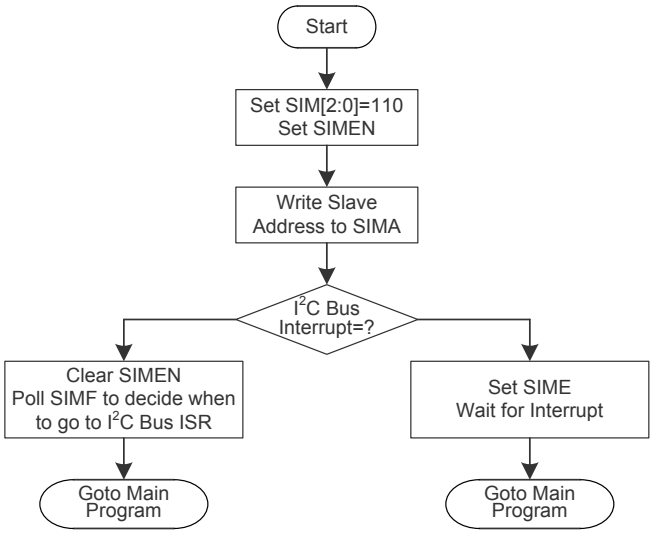
Bit 1 **IAMWU**: I²C Address Match Wake-Up control
0: Disable
1: Enable – must be cleared by the application program after wake-up
This bit should be set to 1 to enable the I²C address match wake up from the SLEEP or IDLE Mode. If the IAMWU bit has been set before entering either the SLEEP or IDLE mode to enable the I²C address match wake up, then this bit must be cleared by the application program after wake-up to ensure correction device operation.

Bit 0 **RXAK**: I²C bus receive acknowledge flag
0: Slave receives acknowledge flag
1: Slave does not receive acknowledge flag
The RXAK flag is the receiver acknowledge flag. When the RXAK flag is "0", it means that a acknowledge signal has been received at the 9th clock, after 8 bits of data have been transmitted. When the slave device in the transmit mode, the slave device checks the RXAK flag to determine if the master receiver wishes to receive the next byte. The slave transmitter will therefore continue sending out data until the RXAK flag is "1". When this occurs, the slave transmitter will release the SDA line to allow the master to send a STOP signal to release the I²C Bus.

I²C Bus Communication

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

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



I²C Bus Inisialisation Flow Chart

I²C Bus Start Signal

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

I²C Slave Address

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

As an I²C bus interrupt can come from three sources, when the program enters the interrupt subroutine, the HAAS and SIMTOF bits should be examined to see whether the interrupt source has come from a matching slave address, the completion of a data byte transfer or the I²C bus time-out occurrence. When a slave address is matched, the devices 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 slave device wishes to read data from the I²C bus or write data to the I²C bus. The slave device should examine this bit to determine if it is to be a transmitter or a receiver. If the SRW flag is "1" then this indicates that the master device wishes to read data from the I²C bus, therefore the slave device must be setup to send data to the I²C bus as a transmitter. If the SRW flag is "0" then this indicates that the master wishes to send data to the I²C bus, therefore the slave device must be setup to read data from the I²C bus as a receiver.

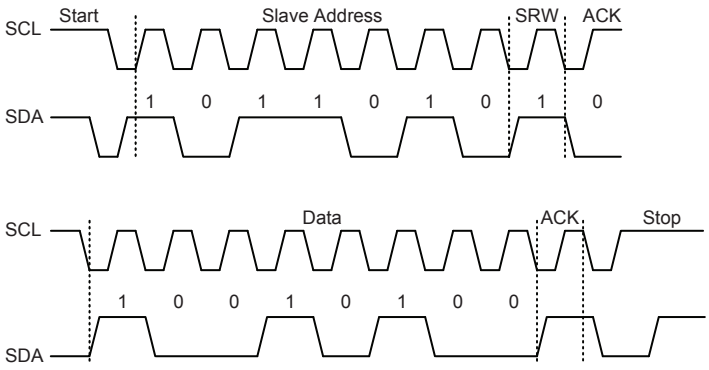
I²C Bus Slave Address Acknowledge Signal

After the master has transmitted a calling address, any slave device on the I²C bus, whose own internal address matches the calling address, must generate an acknowledge signal. The acknowledge signal will inform the master that a slave device has accepted its calling address. If no acknowledge signal is received by the master then a STOP signal must be transmitted by the master to end the communication. When the HAAS flag is high, the addresses have matched and the slave device must check the SRW flag to determine if it is to be a transmitter or a receiver. If the SRW flag is high, the slave device should be setup to be a transmitter so the HTX bit in the SIMC1 register should be set to "1". If the SRW flag is low, then the microcontroller slave device should be setup as a receiver and the HTX bit in the SIMC1 register should be set to "0".

I²C Bus Data and Acknowledge Signal

The transmitted data is 8-bits wide and is transmitted after the slave device has acknowledged receipt of its slave address. The order of serial bit transmission is the MSB first and the LSB last. After receipt of 8-bits of data, the receiver must transmit an acknowledge signal, level "0", before it can receive the next data byte. If the slave transmitter does not receive an acknowledge bit signal from the master receiver, then the slave transmitter will release the SDA line to allow the master to send a STOP signal to release the I²C Bus. The corresponding data will be stored in the SIMD register. If setup as a transmitter, the slave device must first write the data to be transmitted into the SIMD register. If setup as a receiver, the slave device must read the transmitted data from the SIMD register.

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

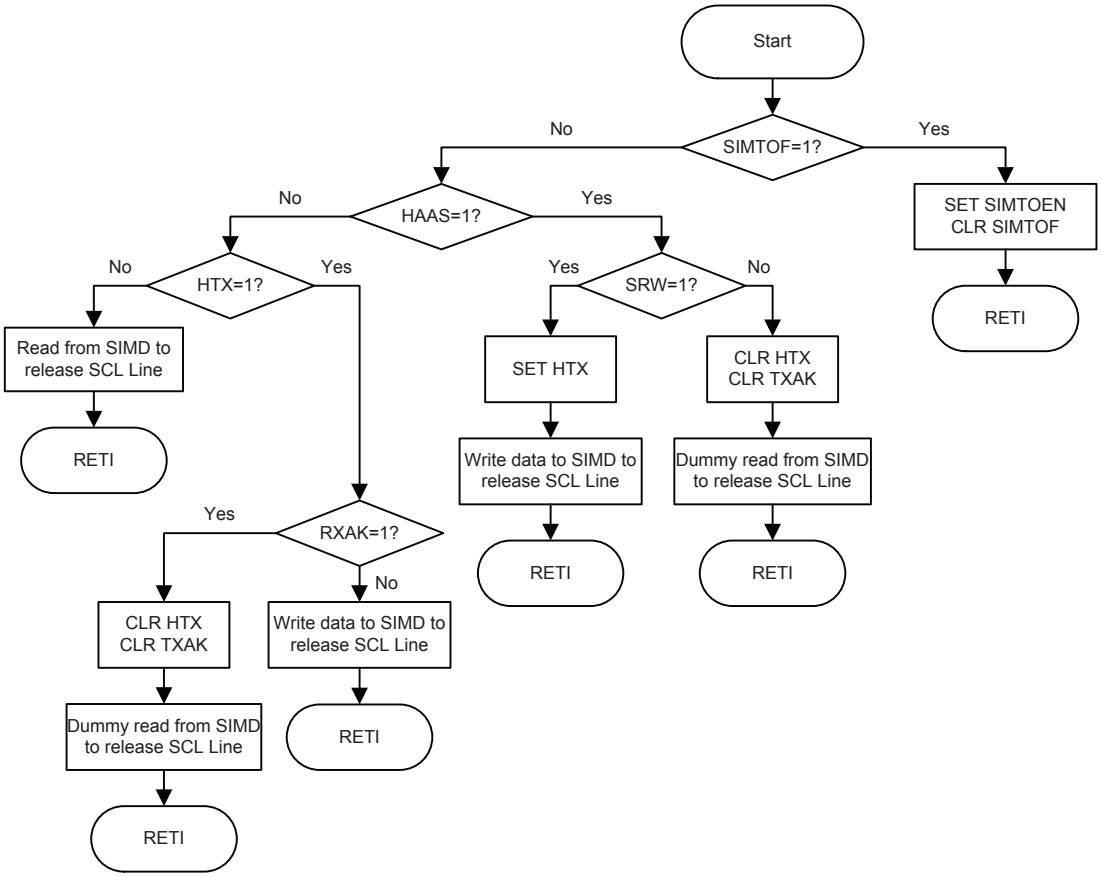


S=Start (1 bit)
SA=Slave Address (7 bits)
SR=SRW bit (1 bit)
M=Slave device send acknowledge bit (1 bit)
D=Data (8 bits)
A=ACK (RXAK bit for transmitter, TXAK bit for receiver, 1 bit)
P=Stop (1 bit)

S	SA	SR	M	D	A	D	A	·····	S	SA	SR	M	D	A	D	A	·····	P

Note: *When a slave address is matched, the devices 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.

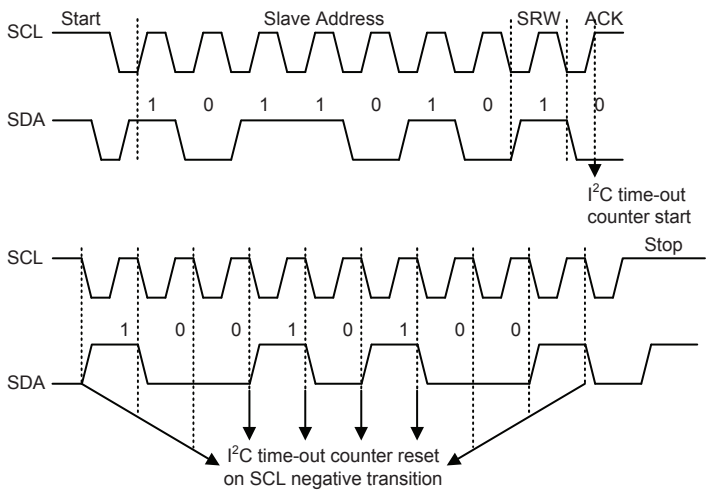
I²C Communication Timing Diagram



I²C Bus ISR Flow Chart

I²C Time-out Control

In order to reduce the I²C lockup problem due to reception of erroneous clock sources, a time-out function is provided. If the clock source connected to the I²C bus is not received for a while, then the I²C circuitry and registers will be reset after a certain time-out period. The time-out counter starts to count on an I²C bus "START" & "address match" condition, and is cleared by an SCL falling edge. Before the next SCL falling edge arrives, if the time elapsed is greater than the time-out period specified by the SIMTOC register, then a time-out condition will occur. The time-out function will stop when an I²C "STOP" condition occurs.



I²C Time-out

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

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

I²C Register after Time-out

The SIMTOF flag can be cleared by the application program. There are 64 time-out period selections which can be selected using the SIMTOS bits in the SIMTOC register. The time-out duration is calculated 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**: SIM I²C Time-out control
0: Disable
1: Enable

Bit 6 **SIMTOF**: SIM I²C Time-out flag
0: No time-out occurred
1: Time-out occurred
This bit is set high by the I²C time-out circuitry and cleared to zero by the application program.

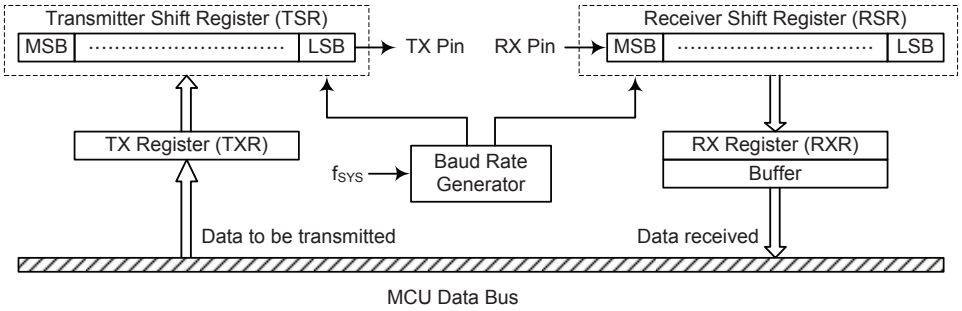
Bit 5~0 **SIMTOS5~SIMTOS0**: SIM I²C Time-out period selection
I²C Time-out clock source is $f_{SUB}/32$
I²C Time-out period is equal to $(SIMTOS[5:0]+1)\times(32/f_{SUB})$

UART Interface

These 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 possesses its own internal interrupt which can be used to indicate when a reception occurs or when a transmission terminates.

The integrated UART function contains the following features:

- Full-duplex, 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



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. Along with the UARTEN, TXEN and RXEN bits, if set, will automatically setup the TX and RX 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 UARTEN, TXEN or RXEN bit, the TX or RX pin will be used as a general purpose I/O or other pin-shared functional pin.

UART Data Transfer Scheme

The above diagram shows the overall data transfer structure arrangement for the UART interface. The actual data to be transmitted from the MCU is first transferred to the TXR register by the application program. The data will then be transferred to the Transmit Shift Register from where it will be shifted out, LSB first, onto the TX pin at a rate controlled by the Baud Rate Generator. Only the TXR 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 RXR register, where it is buffered and can be manipulated by the application program. Only the 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, although referred to in the text, and in application programs, as separate TXR and RXR registers, only exists as a single shared register in the Data Memory. This shared register known as the TXR_RXR register is used for both data transmission and data reception.

UART Status and Control Registers

There are five control registers associated with the UART function. The USR, UCR1 and UCR2 registers control the overall function of the UART, while the BRG register controls the Baud rate. The actual data to be transmitted and received on the serial interface is managed through the TXR_RXR data registers.

Register Name	Bit							
	7	6	5	4	3	2	1	0
USR	PERR	NF	FERR	OERR	RIDLE	RXIF	TIDLE	TXIF
UCR1	UARTEN	BNO	PREN	PRT	STOPS	TXBRK	RX8	TX8
UCR2	TXEN	RXEN	BRGH	ADDEN	WAKE	RIE	TIIE	TEIE
TXR_RXR	TXRX7	TXRX6	TXRX5	TXRX4	TXRX3	TXRX2	TXRX1	TXRX0
BRG	BRG7	BRG6	BRG5	BRG4	BRG3	BRG2	BRG1	BRG0

UART Registers List

USR Register

The USR register is the status register for the UART, which can be read by the program to determine the present status of the UART. All flags within the USR register are read only and further explanations are given below.

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

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

Bit 6 **NF**: Noise flag
0: No noise is detected
1: Noise is detected
The NF flag is the noise flag. When this read only flag is "0", it indicates no noise condition. When the flag is "1", it indicates that the UART has detected noise on the receiver input. The NF flag is set during the same cycle as the RXIF flag but will not be set in the case of as overrun. The NF flag can be cleared by a software sequence which will involve a read to the status register USR followed by an access to the TXR_RXR data register.

Bit 5 **FERR**: Framing error flag
0: No framing error is detected
1: Framing error is detected
The FERR flag is the framing error flag. When this read only flag is "0", it indicates that there is no framing error. When the flag is "1", it indicates that a framing error has been detected for the current character. The flag can also be cleared by a software sequence which will involve a read to the status register USR followed by an access to the TXR_RXR data register.

Bit 4 **OERR**: Overrun error flag
0: No overrun error is detected
1: Overrun error is detected
The OERR flag is the overrun error flag which indicates when the receiver buffer has overflowed. When this read only flag is "0", it indicates that there is no overrun error. When the flag is "1", it indicates that an overrun error occurs which will inhibit further transfers to the TXR_RXR receive data register. The flag is cleared by a software sequence, which is a read to the status register USR followed by an access to the TXR_RXR data register.

Bit 3 **RIDLE**: Receiver status
0: data reception is in progress (data being received)
1: no data reception is in progress (receiver is idle)
The RIDLE flag is the receiver status flag. When this read only flag is "0", it indicates that the receiver is between the initial detection of the start bit and the completion of the stop bit. When the flag is "1", it indicates that the receiver is idle. Between the completion of the stop bit and the detection of the next start bit, the RIDLE bit is "1" indicating that the UART receiver is idle and the RX pin stays in logic high condition.

Bit 2 **RXIF**: Receive TXR_RXR data register status
0: TXR_RXR data register is empty
1: TXR_RXR data register has available data
The RXIF flag is the receive data register status flag. When this read only flag is "0", it indicates that the TXR_RXR read data register is empty. When the flag is "1", it indicates that the TXR_RXR read data register contains new data. When the contents of the shift register are transferred to the TXR_RXR register, an interrupt is generated if RIE=1 in the UCR2 register. If one or more errors are detected in the received word, the appropriate receive-related flags NF, FERR, and/or PERR are set within the same clock cycle. The RXIF flag will eventually be cleared when the USR register is read with RXIF set, followed by a read from the TXR_RXR register, and if the TXR_RXR register has no more new data available.

Bit 1 **TIDLE**: Transmission status
0: data transmission is in progress (data being transmitted)
1: no data transmission is in progress (transmitter is idle)
The TIDLEn flag is known as the transmission complete flag. When this read only flag is "0", it indicates that a transmission is in progress. This flag will be set to "1" when the TXIF flag is "1" and when there is no transmit data or break character being transmitted. When TIDLE is equal to 1, the TX pin becomes idle with the pin state in logic high condition. The TIDLE flag is cleared by reading the USR register with TIDLE set and then writing to the TXR_RXR register. The flag is not generated when a data character or a break is queued and ready to be sent.

Bit 0 **TXIF**: Transmit TXR data register status
0: character is not transferred to the transmit shift register
1: character has transferred to the transmit shift register (TXR_RXR data register is empty)

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

UCR1 Register

The UCR1 register together with the UCR2 register are the 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	UARTEN	BNO	PREN	PRT	STOPS	TXBRK	RX8	TX8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R	W
POR	0	0	0	0	0	0	×	0

"×": unknown

Bit 7 **UARTEN**: UART function enable control
0: Disable UART; TX and RX pins are as I/O or other pin-shared functional pins
1: Enable UART; TX and RX pins function as UART pins

The UARTEN bit is the UART enable bit. When this bit is equal to "0", the UART will be disabled and the RX pin as well as the TX pin will be as the general purpose I/O or other pin-shared functional pins. When the bit is equal to "1", the UART will be enabled and the TX and RX pins will function as defined by the TXEN and RXEN enable control bits. When the UART is disabled, it will empty the buffer so any character remaining in the buffer will be discarded. In addition, the value of the baud rate counter will be reset. If the UART is disabled, all error and status flags will be reset. Also the TXEN, RXEN, TXBRK, RXIF, OERR, FERR, PERR and NF bits will be cleared, while the TIDLE, TXIF and RIDLE bits will be set. Other control bits in UCR1, UCR2 and BRG registers will remain unaffected. If the UART is active and the UARTEN bit is cleared, all pending transmissions and receptions will be terminated and the module will be reset as defined above. When the UART is re-enabled, it will restart in the same configuration.

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

This bit is used to select the data length format, which can have a choice of either 8-bit or 9-bit format. When this bit is equal to "1", a 9-bit data length format will be selected. If the bit is equal to "0", then an 8-bit data length format will be selected. If 9-bit data length format is selected, then bits RX8 and TX8 will be used to store the 9th bit of the received and transmitted data respectively.

Note: 1. If BNO=1 (9-bit data transfer) and parity function is enabled, the 9th bit of data is the parity bit which will not be transferred to RX8.
2. If BNO=0 (8-bit data transfer) and parity function is enabled, the 8th bit of data is the parity bit which will not be transferred to RX7.

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

This bit is the parity function 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 **PRT**: 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 **STOPS**: 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 format are used. If the bit is equal to "0", then only one stop bit format is used.

Bit 2 **TXBRK**: Transmit break character
0: No break character is transmitted
1: Break characters transmit
The TXBRK bit is the Transmit Break Character bit. When this bit is equal to "0", there are no break characters and the TX pin operates normally. When the bit is equal to "1", there are transmit break characters and the transmitter will send logic zeros. When this bit is equal to "1", after the buffered data has been transmitted, the transmitter output is held low for a minimum of a 13-bit length and until the TXBRK bit is reset.

Bit 1 **RX8**: Receive data bit 8 for 9-bit data transfer format (read only)
This bit is only used if 9-bit data transfers are used, in which case this bit location will store the 9th bit of the received data known as RX8. The BNO bit is used to determine whether data transfers are in 8-bit or 9-bit format.

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

UCR2 Register

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

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

Bit 7 **TXEN**: UART Transmitter enable control
0: UART Transmitter is disabled
1: UART Transmitter is enabled
The TXEN bit 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 used as an I/O or other pin-shared functional pin. If the TXEN bit is equal to "1" and the UARTEN bit is also equal to 1, the transmitter will be enabled and the TX pin will be controlled by the UART. Clearing the TXEN bit during a transmission will cause the data transmission to be aborted and will reset the transmitter.

Bit 6 **RXEN**: UART Receiver enable control
0: UART Receiver is disabled
1: UART Receiver is enabled

The RXEN bit 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 receiver buffers will be reset. In this situation the RX pin will be used as an I/O or other pin-shared functional pin. If the RXEN bit is equal to "1" and the UARTEN bit is also equal to 1, the receiver will be enabled and the RX pin will be controlled by the UART. Clearing the RXEN bit during a reception will cause the data reception to be aborted and will reset the receiver.

Bit 5 **BRGH**: Baud Rate speed selection
0: Low speed baud rate
1: High speed baud rate

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

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

The bit named ADDEN is the address detection function enable control bit. When this bit is equal to 1, the address detection function is enabled. When it occurs, if the 8^{th} bit, which corresponds to RX7 if BNO=0, or the 9^{th} bit, which corresponds to RX8 if BNO=1, has a value of "1", then the received word will be identified as an address, rather than data. If the corresponding interrupt is enabled, an interrupt request will be generated each time the received word has the address bit set, which is the 8^{th} or 9^{th} bit depending on the value of the BNO bit. If the address bit known as the 8^{th} or 9^{th} bit of the received word is "0" with the address detection function being enabled, an interrupt will not be generated and the received data will be discarded.

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

The 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_{SYS}, is switched off. There will be no RX pin wake-up UART function if the UART clock, f_{SYS}, exists. If the WAKE bit is equal to 1 and the UART clock, f_{SYS}, 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_{SYS}, via the application programs. Otherwise, the UART function can not resume even if there is a falling edge on the RX pin when the WAKE bit is cleared to 0.

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

The bit enables or disables the receiver interrupt. If this bit is equal to 1 and when the receiver overrun flag OERR or received data available flag RXIF is set, the UART interrupt request flag will be set. If this bit is equal to 0, the UART interrupt request flag will not be influenced by the condition of the OERR or RXIF flags.

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

The bit enables or disables the transmitter idle interrupt. If this bit is equal to 1 and when the transmitter idle flag TIDLE is set, due to a transmitter idle condition, the UART interrupt request flag will be set. If this bit is equal to 0, the UART interrupt request flag will not be influenced by the condition of the TIDLE flag.

Bit 0 **TEIE**: Transmitter Empty interrupt enable control
0: Transmitter empty interrupt is disabled
1: Transmitter empty interrupt is enabled
The bit enables or disables the transmitter empty interrupt. If this bit is equal to 1 and when the transmitter empty flag TXIF is set, due to a transmitter empty condition, the UART interrupt request flag will be set. If this bit is equal to 0, the UART interrupt request flag will not be influenced by the condition of the TXIF flag.

TXR_RXR Register

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

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

"×": unknown

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

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 BRG register and the second is the value of the BRGH bit within the UCR2 control register. The BRGH 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 in the BRG register, N, which is used in the following baud rate calculation formula determines the division factor. Note that N is the decimal value placed in the BRG register and has a range of between 0 and 255.

UCR2 BRGH Bit	0	1
Baud Rate (BR)	f_{SYS} / [64 (N+1)]	f_{SYS} / [16 (N+1)]

By programming the BRGH bit which allows selection of the related formula and programming the required value in the BRG 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 BRG register, there will be an error associated between the actual and requested value. The following example shows how the BRG register value N and the error value can be calculated.

BRG Register

Bit	7	6	5	4	3	2	1	0
Name	BRG7	BRG6	BRG5	BRG4	BRG3	BRG2	BRG1	BRG0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	×	×	×	×	×	×	×	×

"×": unknown

Bit 7~0 **BRG7~BRG0**: Baud Rate values
By programming the BRGH bit in the UCR2 register which allows selection of the related formula described above and programming the required value in the BRG register, the required baud rate can be setup.

Calculating the Baud Rate and Error Values

For a clock frequency of 4MHz, and with BRGH set to 0 determine the BRG 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_{SYS} / [64 (N+1)]

Re-arranging this equation gives N=[f_{SYS} / (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 BRG 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 BNO, PRT, PREN and STOPS bits in the UCR1 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 transmitter and receiver of the UART are functionally independent, they both use the same data format and baud rate. In all cases stop bits will be used for data transmission.

Enabling/Disabling the UART Interface

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

Clearing the UARTEN bit will disable the TX and RX pins and then these two pins can be used as an I/O or other pin-shared functional pins by properly configurations. 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 enable control, the error and status flags with bits TXEN, RXEN, TXBRK, RXIF, OERR, FERR, PERR and NF being cleared while bits TIDLE, TXIF and RIDLE will be set. The remaining control bits in the UCR1, UCR2 and BRG registers will remain unaffected. If the UARTEN bit in the UCR1 register is cleared while the UART is active, then all pending transmissions and receptions will be immediately suspended and the UART will be reset to a condition as defined above. If the UART is then subsequently re-enabled, it will restart again in the same configuration.

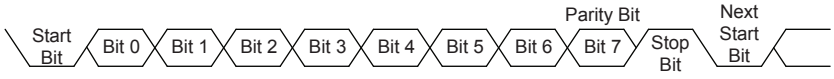
Data, Parity and Stop Bit Selection

The format of the data to be transferred is composed of various factors such as data bit length, parity on/off, parity type, address bits and the number of stop bits. These factors are determined by the setup of various bits within the UCR1 register. The BNO bit controls the number of data bits which can be set to either 8 or 9. The PRT bit controls the choice if odd or even parity. The PREN bit controls the parity on/off function. The STOPS 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.

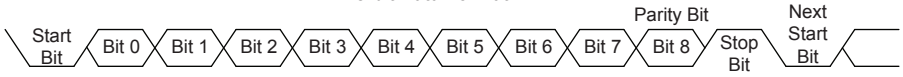
Start Bit	Data Bits	Address Bits	Parity Bit	Stop Bit
Example of 8-bit Data Formats				
1	8	0	0	1
1	7	0	1	1
1	7	1	0	1
Example of 9-bit Data Formats				
1	9	0	0	1
1	8	0	1	1
1	8	1	0	1

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 BNO bit in the UCR1 register. When BNO bit is set, the word length will be set to 9 bits. In this case the 9th bit, which is the MSB, needs to be stored in the TX8 bit in the UCR1 register. At the transmitter core lies the Transmitter Shift Register, more commonly known as the TSR, whose data is obtained from the transmit data register, which is known as the TXR_RXR register. The data to be transmitted is loaded into this TXR_RXR register by the application program. The TSR register is not written to with new data until the stop bit from the previous transmission has been sent out. As soon as this stop bit has been transmitted, the TSR can then be loaded with new data from the TXR_RXR register, if it is available. It should be noted that the TSR register, unlike many other registers, is not directly mapped into the Data Memory area and as such is not available to the application program for direct read/write operations. An actual transmission of data will normally be enabled when the TXENn bit is set, but the data will not be transmitted until the TXR_RXR register has been loaded with data and the baud rate generator has defined a shift clock source. However, the transmission can also be initiated by first loading data into the TXR_RXR register, after which the TXEN bit can be set. When a transmission of data begins, the TSR is normally empty, in which case a transfer to the TXR_RXR register will result in an immediate transfer to the TSR. If during a transmission the TXEN bit is cleared, the transmission will immediately cease and the transmitter will be reset. The TX output pin can then return to the I/O or other pin-shared function by properly configurations.

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 LSB first. In the transmit mode, the TXR_RXR register forms a buffer between the internal bus and the transmitter shift register. It should be noted that if 9-bit data format has been selected, then the MSB will be taken from the TX8 bit in the UCR1 register. The steps to initiate a data transfer can be summarized as follows:

- Make the correct selection of the BNO, PRT, PREN and STOPS bits to define the required word length, parity type and number of stop bits.
- Setup the BRG register to select the desired baud rate.
- Set the TXEN bit to ensure that the UART transmitter is enabled and the TX pin is used as a UART transmitter pin.
- Access the USR register and write the data that is to be transmitted into the TXR_RXR register. Note that this step will clear the TXIF bit.

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

1. A USR register access
2. A TXR_RXR register write execution

The read-only TXIF flag is set by the UART hardware and if set indicates that the TXR_RXR register is empty and that other data can now be written into the TXR_RXR register without overwriting the previous data. If the TEIE bit is set, then the TXIF flag will generate an interrupt. During a data transmission, a write instruction to the TXR_RXR register will place the data into the TXR_RXR register, which will be copied to the shift register at the end of the present transmission. When there is no data transmission in progress, a write instruction to the TXR_RXR register will place the data directly into the shift register, resulting in the commencement of data transmission, and the TXIF bit being immediately set. When a frame transmission is complete, which happens after stop bits are sent or after the break frame, the TIDLE bit will be set. To clear the TIDLEn bit the following software sequence is used:

1. A USR register access
2. A TXR_RXR register write execution

Note that both the TXIF and TIDLE bits are cleared by the same software sequence.

Transmitting Break

If the TXBRK bit is set, then the break characters will be sent on the next transmission. Break character transmission consists of a start bit, followed by 13xN "0" bits, where N=1, 2, etc. If a break character is to be transmitted, then the TXBRK bit must be first set by the application program and then cleared to generate the stop bits. Transmitting a break character will not generate a transmit interrupt. Note that a break condition length is at least 13 bits long. If the TXBRK bit is continually kept at a logic high level, then the transmitter circuitry will transmit continuous break characters. After the application program has cleared the TXBRK bit, the transmitter will finish transmitting the last break character and subsequently send out one or two stop bits. The automatic logic high 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 can be selected by programming the BNO bit in the UCR1 register. When BNO bit is set, the word length will be set to 9 bits. In this case the 9th bit, which is the MSB, will be stored in the RX8 bit in the UCR1 register. At the receiver core lies the Receiver Shift Register more 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 to the shift register, with the least significant bit LSB first. The TXR_RXR register is a two byte deep FIFO data buffer, where two bytes can be held in the FIFO while the 3rd byte can continue to be received. Note that the application program must ensure that the data is read from TXR_RXR before the 3rd byte has been completely shifted in, otherwise the 3rd byte will be discarded and an overrun error OERR will be subsequently indicated. The steps to initiate a data transfer can be summarized as follows:

- Make the correct selection of the BNO, PRT, PREN and STOPS bits to define the required word length, parity type and number of stop bits.
- Setup the BRG register to select the desired baud rate.
- Set the RXEN bit to ensure that the UART receiver is enabled and 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 RXIF bit in the USR register will be set when the TXR_RXR register has data available. There will be at most one more character that can be read.
- When the contents of the shift register have been transferred to the TXR_RXR register and if the RIE bit is set, then an interrupt will be generated.
- If during reception, a frame error, noise error, parity error or an overrun error has been detected, then the error flags can be set.

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

1. A USR register access
2. A TXR_RXR register read execution

Receiving Break

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

- The framing error flag, FERR, will be set.
- The receive data register, TXR_RXR, will be cleared.
- The OERR, NF, PERR, RIDLE or RXIF flags will possibly be set.

Idle Status

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

Receiver Interrupt

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

Managing Receiver Errors

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

Overrun Error – OERR

The TXR_RXR register is composed of a two byte deep FIFO data buffer, where two bytes can be held in the FIFO register, while a 3th byte can continue to be received. Before the 3th byte has been entirely shifted in, the data should be read from the TXR_RXR register. If this is not done, the overrun error flag OERR will be consequently indicated.

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

- The OERR flag in the USR register will be set.
- The TXR_RXR contents will not be lost.
- The shift register will be overwritten.
- An interrupt will be generated if the RIE bit is set.

The OERR flag can be cleared by an access to the USR register followed by a read to the TXR_RXR register.

Noise Error – NF

Over-sampling is used for data recovery to identify valid incoming data and noise. If noise is detected within a frame, the following will occur:

- The read only noise flag, NF, in the USR register will be set on the rising edge of the RXIF bit.
- Data will be transferred from the shift register to the TXR_RXR register.
- No interrupt will be generated. However this bit rises at the same time as the RXIF bit which itself generates an interrupt.

Note that the NF flag is reset by an USR register read operation followed by a TXR_RXR register read operation.

Framing Error – FERR

The read only framing error flag, FERR, in the USR register, is set if a zero is detected instead of stop bits. If two stop bits are selected, both stop bits must be high. Otherwise the FERR flag will be set. The FERR flag and the received data will be recorded in the USR and TXR_RXR registers respectively and the FERR flag will be cleared in any reset.

Parity Error – PERR

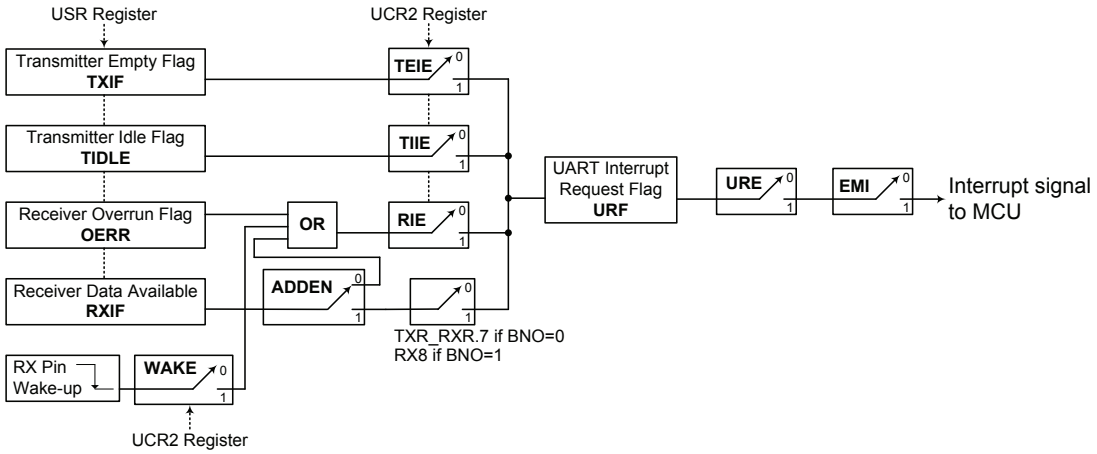
The read only parity error flag, PERR, in the USR register, is set if the parity of the received word is incorrect. This error flag is only applicable if the parity function is enabled, PREN=1, and if the parity type, odd or even, is selected. The read only PERR flag and the received data will be recorded in the USR and TXR_RXR registers respectively and the flag will be cleared on any reset. It should be noted that the FERR and PERR flags in the USR register should first be read by the application programs before reading the data word.

UART Interrupt Structure

Several individual UART conditions can generate a UART interrupt. When these conditions exist, a low pulse will be generated to get the attention of the microcontroller. These conditions are a transmitter data register empty, transmitter idle, receiver data available, receiver overrun, address detect and an RX pin wake-up. When any of these conditions are created, if its corresponding interrupt control is enabled and the stack is not full, the program will jump to its corresponding interrupt vector where it can be serviced before returning to the main program. Four of these conditions have the corresponding USR register flags which will generate a UART interrupt if its associated interrupt enable control bit in the UCR2 register is set. The two transmitter interrupt conditions have their own corresponding enable control bits, while the two receiver interrupt conditions have a shared enable control bit. These enable bits can be used to mask out individual UART interrupt sources.

The address detect condition, which is also a UART interrupt source, does not have an associated flag, but will generate a UART interrupt when an address detect condition occurs if its function is enabled by setting the ADDEN bit in the UCR2 register. An RX pin wake-up, which is also a UART interrupt source, does not have an associated flag, but will generate a UART interrupt if the microcontroller is woken up from the IDLE0 or SLEEP mode by a falling edge on the RX pin if the WAKE and RIE bits in the UCR2 register are set. 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 USR register flags are read only and cannot be cleared or set by the application program, neither will they be cleared when the program jumps to the corresponding interrupt servicing routine, as is the case for some of the other interrupts. The flags will be cleared automatically when certain actions are taken by the UART, the details of which are given in the UART register section. The overall UART interrupt can be disabled or enabled by the related interrupt enable control bits in the interrupt control registers of the microcontroller to decide whether the interrupt requested by the UART module is masked out or allowed.



UART Interrupt Structure

Address Detect Mode

Setting the Address Detect function enable control bit, ADDEN, in the UCR2 register, enables this special function. If this bit is set to 1, then an additional qualifier will be placed on the generation of a Receiver Data Available interrupt, which is requested by the RXIF flag. If the ADDEN bit is equal to 1, then when the data is available, an interrupt will only be generated, if the highest received bit has a high value. Note that the related interrupt enable control bit and the EMI bit of the microcontroller must also be enabled for correct interrupt generation. The highest address bit is the 9th bit if the bit BNO=1 or the 8th bit if the bit BNO=0. If the highest bit is high, then the received word will be defined as an address rather than data. A Data Available interrupt will be generated every time the last bit of the received word is set. If the ADDEN bit is equal to 0, then a Receive Data Available interrupt will be generated each time the RXIF flag is set, irrespective of the data last bit status. The address detection and parity functions are mutually exclusive functions. Therefore, if the address detect function is enabled, then to ensure correct operation, the parity function should be disabled by resetting the parity function enable bit PREN to zero.

ADDEN	Bit 9 if BNO=1 Bit 8 if BNO=0	UART Interrupt Generated
0	0	√
	1	√
1	0	×
	1	√

ADDEN Bit Function

UART Power Down and Wake-up

When the system clock, f_{SYS}, is switched off, the UART will cease to function. If the MCU executes the "HALT" instruction which results in the system clock being switched 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 executes the "HALT" instructure which results in the system clock being switched off while receiving data, then the reception of data will likewise be paused. When the MCU enters the power down mode, note that the USR, UCR1, UCR2, transmit and receive registers, as well as the BRG 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 power down mode.

The UART function contains a receiver RX pin wake-up function, which is enabled or disabled by the WAKE bit in the UCR2 register. If this bit, along with the UART enable bit, UARTEN, the receiver enable bit, RXEN and the receiver interrupt bit, RIE, are all set before the MCU enters the power down mode with the UART clock f_{SYS} being switched off, then a falling edge on the RX pin will initiate a 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.

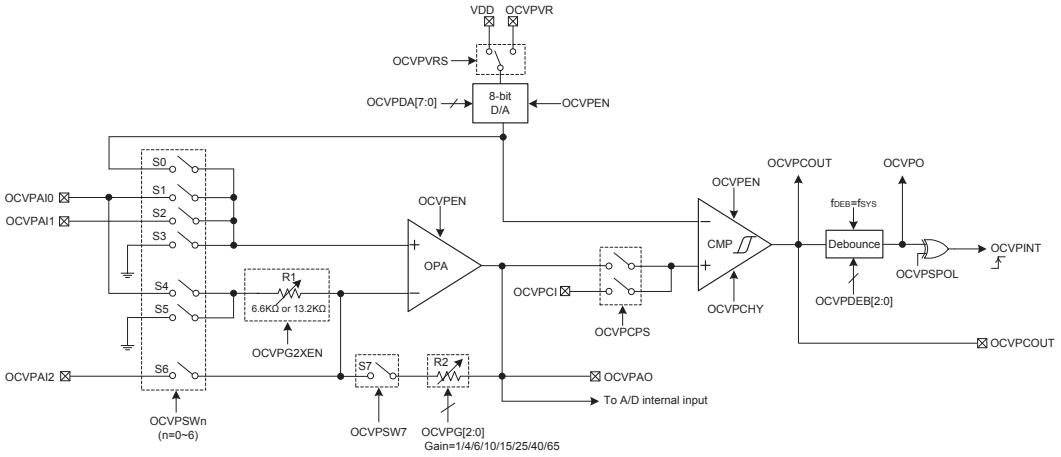
For a UART wake-up interrupt to occur, in addition to the bits for the wake-up being set, the global interrupt enable bit, EMI, and the UART interrupt enable bit, URE, must be set. If the EMI and URE bits are not set then only a wake up event will occur and no interrupt will be generated. Note also that as it takes certain system clock cycles after a wake-up before normal microcontroller resumes, the UART interrupt will not be generated until after this time has elapsed.

Over Current/Voltage Protection Function – OCVP

The series of devices includes a current/voltage protection function which provides a protection mechanism for applications, for example, over current or over voltage protection, etc. The current on the OCVPAI0, OCVPAI1 and OCVPAI2 pins are converted to a relevant voltage level according to the current value using the OCVP operational amplifier. It is then compared with a reference voltage generated by an 8-bit D/A converter. The voltage on the OCVPCI pin is compared with a reference voltage generated by the 8-bit D/A converter. When the OCVPF flag changes from 0 to 1 and if the corresponding interrupt control is enabled, an OCVP interrupt will be generated to indicate a specific current or voltage condition has occurred.

OCVP Operation

The OCVP circuit is used to prevent the input current or voltage from being in an unexpected level range. The current on the OCVPAI0, OCVPAI1 or OCVPAI2 pin is converted to a voltage and then amplified by the OCVP operational amplifier with a programmable gain from 1 to 65 selected by the OCVPG2~OCVPG0 bits in the OCVPC2 register. This is known as the Programmable Gain Amplifier or PGA. This PGA can also be configured to operate in the non-inverting, inverting or input offset cancellation mode determined by the OCVPSW7~OCVPSW0 bits in the OCVPC0 register. After the current is converted and amplified to a specific voltage level, it will be compared with a reference voltage provided by an 8-bit D/A converter. The voltage on the OCVPCI pin is also compared with a reference provided by the 8-bit D/A converter.



OCVP Block Diagram

To compare the OCVPAO output signal or the OCVPCI input signal with the D/A converter output voltage is selected using the OCVPCPS bit in the OCVPC1 register. The 8-bit D/A converter power can be supplied by the external power pin, VDD or OCVPVR, selected by the OCVPVRS bit in the OCVPC1 register. The comparator output, OCVPCOUT, will first be filtered with a certain debounce time period selected by the OCVPDEB2~OCVPDEB0 bits in the OCVPC2 register. Then a filtered OCVP digital comparator output, OCVPO, is obtained to indicate whether a user-defined current or voltage condition occurs or not.

If the the OCVPSPOL bit is cleared to 0 and the comparator inputs force the OCVPO bit to change from 0 to 1, or if the OCVPSPOL bit is set to 1 and the comparator inputs force the OCVPO bit changes from 1 to 0, the corresponding interrupt will be generated if the relevant interrupt control bit is enabled. It is important to note that, only an OCVPINT rising edge can trigger an OCVP interrupt request, as shown in the following diagram, so the OCVPSPOL bit must be properly configured according to user's application requirements.

Note that the debounce clock, f_{DEB}, comes from the system clock, f_{SYS}. The operational amplifier output voltage also can be read out by the A/D converter through an A/D internal input channel. The DAC output voltage is controlled by the OCVPDA register and the DAC output is defined as below:

DAC V_{OUT}=(DAC reference voltage/256)×D[7:0]

OCVP Control Registers

Overall operation of the OCVP function is controlled using several registers. One register is used to provide the reference voltages for the OCVP circuit. There are two registers used to cancel out the operational amplifier and comparator input offset. The remaining four registers are control registers which control the OCVP function, D/A converter reference voltage select, switches on/off control, PGA gain select, comparator non-inverting input select, comparator de-bounce time, comparator hysteresis function and comparator output polarity control, etc. Each OCVP related input or output signal has a corresponding control bit in the OCVPC3 register. For a more detailed description regarding the input offset voltage cancellation procedures, refer to the corresponding input offset cancellation sections.

Register Name	Bit							
	7	6	5	4	3	2	1	0
OCVPC0	OCVPSW7	OCVPSW6	OCVPSW5	OCVPSW4	OCVPSW3	OCVPSW2	OCVPSW1	OCVPSW0
OCVPC1	OCVPEN	OCVPCHY	OCVPO	OCVPSPOL	—	—	OCVPCPS	OCVPVRS
OCVPC2	—	OCVPG2XEN	OCVPG2	OCVPG1	OCVPG0	OCVPDEB2	OCVPDEB1	OCVPDEB0
OCVPC3	OCVPCIEN	OCVPAI2EN	OCVPAI1EN	OCVPAI0EN	—	—	OCVPOPEN	OCVPCPEN
OCVPDA	D7	D6	D5	D4	D3	D2	D1	D0
OCVPOCAL	OCVPOOFM	OCVPORSP	OCVPOOF5	OCVPOOF4	OCVPOOF3	OCVPOOF2	OCVPOOF1	OCVPOOF0
OCVPCCAL	OCVPCOUT	OCVPCOFM	OCVPCRSP	OCVPCOF4	OCVPCOF3	OCVPCOF2	OCVPCOF1	OCVPCOF0

OCVP Registers List

OCVPC0 Register

Bit	7	6	5	4	3	2	1	0
Name	OCVPSW7	OCVPSW6	OCVPSW5	OCVPSW4	OCVPSW3	OCVPSW2	OCVPSW1	OCVPSW0
R/W	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W
POR	1	1	0	0	1	0	0	0

Bit 7 **OCVPSW7**: OCVP switch S7 on/off control
0: Off
1: On

Bit 6 **OCVPSW6**: OCVP switch S6 on/off control
0: Off
1: On

Bit 5 **OCVPSW5**: OCVP switch S5 on/off control
0: Off
1: On

Bit 4 **OCVPSW4**: OCVP switch S4 on/off control
0: Off
1: On

Bit 3 **OCVPSW3**: OCVP switch S3 on/off control
0: Off
1: On

Bit 2 **OCVPSW2**: OCVP switch S2 on/off control
0: Off
1: On

Bit 1 **OCVPSW1**: OCVP switch S1 on/off control
0: Off
1: On

Bit 0 **OCVPSW0**: OCVP switch S0 on/off control
0: Off
1: On

OCVPC1 Register

Bit	7	6	5	4	3	2	1	0
Name	OCVPEN	OCVPCHY	OCVPO	OCVPSPOL	—	—	OCVPCPS	OCVPVRS
R/W	R/W	R/W	R	R/W	—	—	R/W	R/W
POR	0	0	0	0	—	—	0	0

Bit 7 **OCVPEN**: OCVP function enable control
0: Disable
1: Enable
When this bit is cleared to 0, the overall OCVP operation will be disabled and the comparator output, OCVPCOUT, will be equal to 0.

Bit 6 **OCVPCHY**: OCVP Comparator Hysteresis function enable control
0: Disable
1: Enable

Bit 5 **OCVPO**: OCVP Comparator debounced output
This bit is the debounced version of the OCVPCOUT bit

Bit 4 **OCVPSPOL**: OCVPO polarity control
0: Not inverted
1: Inverted

Bit 3~2 Unimplemented, read as "0"

Bit 1 **OCVPCPS**: OCVP Comparator non-inverting input selection
0: From OCVPAO – OPA output
1: From OCVPCI pin

Bit 0 **OCVPVRS**: OCVP D/A converter reference voltage selection
0: From VDD pin
1: From OCVPVR pin

OCVPC2 Register

Bit	7	6	5	4	3	2	1	0
Name	—	OCVPG2XEN	OCVPG2	OCVPG1	OCVPG0	OCVPDEB2	OCVPDEB1	OCVPDEB0
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 **OCVPG2XEN**: OCVP PGA R2/R1 ratio doubling enable control
0: Disable (R1=13.2kΩ)
1: Enable (R1=6.6kΩ)
When this bit is set to 1, the R2/R1 ratio selected by the OCVPG2~OCVPG0 bits will be doubled.

Bit 5~3 **OCVPG2~OCVPG0**: OCVP PGA R2/R1 ratio selection
000: R2/R1=1
001: R2/R1=4
010: R2/R1=6
011: R2/R1=10
100: R2/R1=15
101: R2/R1=25
110: R2/R1=40
111: R2/R1=65

The internal resistors, R1 and R2, should be used when the gain is determined by these bits. This means the S4 or S5 switch together with the S7 switch should be on. Otherwise, the gain accuracy will not be guaranteed. When the OCVPG2XEN bit is set to 1 to enable the R2/R1 ratio doubling function, the above R2/R1 values will be doubled. The calculating formula of the PGA gain for the inverting and non-inverting mode is described in the "Input Voltage Range" section.

Bit 2~0 **OCVPDEB2~OCVPDEB0**: OCVP Comparator output debounce time selection
000: Bypass, no debounce
001: (1~2) × t_{DEB}
010: (3~4) × t_{DEB}
011: (7~8) × t_{DEB}
100: (15~16) × t_{DEB}
101: (31~32) × t_{DEB}
110: (63~64) × t_{DEB}
111: (127~128) × t_{DEB}

Note: $t_{DEB}=1/f_{DEB}$, $f_{DEB}=f_{SYS}$

OCVPC3 Register

Bit	7	6	5	4	3	2	1	0
Name	OCVPCIEN	OCVPAI2EN	OCVPAI1EN	OCVPAI0EN	—	—	OCVPOPEN	OCVPCPEN
R/W	R/W	R/W	R/W	R/W	—	—	R/W	R/W
POR	0	0	0	0	—	—	0	0

Bit 7 **OCVPCIEN**: OCVPCI input signal control
0: Disable
1: Enable

When the OCVPEN bit is set to 0 to disable the OCVP, the OCVPCI input will be forced to disable.

Bit 6 **OCVPAI2EN**: OCVPAI2 input signal control
0: Disable
1: Enable

When the OCVPEN bit is set to 0 to disable the OCVP, the OCVPAI2 input will be forced to disable.

Bit 5 **OCVPAI1EN**: OCVPAI1 input signal control
0: Disable
1: Enable

When the OCVPEN bit is set to 0 to disable the OCVP, the OCVPAI1 input will be forced to disable.

Bit 4 **OCVPAI0EN**: OCVPAI0 input signal control
0: Disable
1: Enable

When the OCVPEN bit is set to 0 to disable the OCVP, the OCVPAI0 input will be forced to disable.

Bit 3~2 Unimplemented, read as "0"

Bit 1 **OCVPOPEN**: OCVPAO output signal control
0: Disable
1: Enable
When the OCVPEN bit is set to 0 to disable the OCVP, the OCVP OPA output will be forced to disable.

Bit 0 **OCVPCPEN**: OCVPCOUT output signal control
0: Disable
1: Enable
When the OCVPEN bit is set to 0 to disable the OCVP, the OCVP comparator output will be forced to disable.

OCVPDA 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 OCVP D/A Converter Data register bit 7 ~ bit 0
OCVP D/A Converter Output=(DAC reference voltage/256)×D[7:0]

OCVPOCAL Register

Bit	7	6	5	4	3	2	1	0
Name	OCVPOOFM	OCVPORSP	OCVPOOF5	OCVPOOF4	OCVPOOF3	OCVPOOF2	OCVPOOF1	OCVPOOF0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	1	0	0	0	0	0

Bit 7 **OCVPOOFM**: OCVP Operational Amplifier operating mode selection
0: Normal operating mode
1: Offset cancellation mode
This bit is used to select the OCVP Operational Amplifier operating mode. To select the Operational Amplifier input offset cancellation mode, the OCVPSW bit field must first be set to 28H and then the OCVPOOFM bit must be set to 1 followed by the OCVPCOFM bit being cleared to 0. Refer to the "Operational Amplifier Input Offset Cancellation" section for the detailed offset cancellation procedures.

Bit 6 **OCVPORSP**: OCVP Operational Amplifier input offset cancellation reference input selection
0: Operational Amplifier inverting input
1: Operational Amplifier non-inverting input

Bit 5~0 **OCVPOOF5~OCVPOOF0**: OCVP Operational Amplifier input offset cancellation value
This 6-bit field is used to perform the Operational Amplifier input offset cancellation operation and the value after the input offset cancellation can be restored into this field. More detailed information is described in the "Operational Amplifier Input Offset Cancellation" section.

OCVPCCAL Register

Bit	7	6	5	4	3	2	1	0
Name	OCVPCOUT	OCVPCOFM	OCVPCRSP	OCVPCOF4	OCVPCOF3	OCVPCOF2	OCVPCOF1	OCVPCOF0
R/W	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	1	0	0	0	0

Bit 7 **OCVPCOUT**: OCVP Comparator output
0: Non-inverting input voltage < D/A converter output voltage
1: Non-inverting input voltage > D/A converter output voltage
This bit is used to indicate whether the non-onverting input voltage is greater than the D/A converter output voltage. If the OCVPCOUT bit is set to 1, it means that the non-onverting input voltage is greater than the D/A converter output voltage. Otherwise, the non-inverting input voltage is less than the D/A converter output voltage. This bit value can be output on the OCVPCOUT pin.

Bit 6 **OCVPCOFM**: OCVP Comparator operating mode selection
0: Normal operating mode
1: Offset cancellation mode
This bit is used to select the OCVP comparator operating mode. To select the comparator input offset cancellation mode, the OCVPSW bit field must first be set to 28H and then the OCVPCOFM bit must be set to 1 followed by the OCVPOOFM bit being cleared to 0. Refer to the "Comparator Input Offset Cancellation" section for the detailed offset cancellation procedures.

Bit 5 **OCVPCRSP**: OCVP Comparator input offset cancellation reference input selection
0: Comparator inverting input
1: Comparator non-inverting input

Bit 4~0 **OCVPCOF4~OCVPCOF0**: OCVP Comparator input offset cancellation value
This 5-bit field is used to perform the comparator input offset cancellation operation and the value after the input offset cancellation can be restored into this field. Refer to the "Comparator Input Offset Cancellation" section for more detailed information.

Input Voltage Range

Together with different PGA operating modes, the input voltage can be positive or negative to provide diverse applications for the device. The PGA output for the positive or negative input voltage is respectively calculated based on different formulas and described by the following examples.

- For $V_{IN} > 0$, the PGA operates in the non-inverting mode and the PGA output is obtained using the formula below:

$$V_{OUT} = (1 + \frac{R_2}{R_1}) \times V_{IN}$$

- When the PGA operates in the non-inverting mode, a unity gain buffer is provided. If the OCVPSW6~OCVPSW4 bits are set to "000", the PGA gain will be equal to 1 and the PGA will act as a unity gain buffer. The switches, S6, S5 and S4, will be switched off internally and the PGA output voltage is equal to V_{IN}.

$$V_{OUT} = V_{IN}$$

- If S3 and S4 are switched on, the input node is OCVPAI0. For $0 > V_{IN} > -0.4$, the PGA operates in the inverting mode and the PGA output is obtained using the formula below. Note that if the input voltage, V_{IN}, is negative, it can not be lower than (-0.4V) which will result in current leakage.

$$V_{OUT} = -\frac{R_2}{R_1} \times V_{IN}$$

Input Offset Cancellation

To operate in the input offset cancellation mode for the OCVP circuit, the OCVPSW7~OCVPSW0 bits should first be set to "28H". For operational amplifier and comparator input offset cancellation, the procedures are similar except for setting the respective control bits.

Operational Amplifier Input Offset Cancellation

Setp 1. Set OCVPSW [7:0]=28H (S3 and S5 are ON, other switches are OFF).
Set OCVPOOFM=1, OCVPCOFM=0, OCVPORSP=1, OCVPCPS=0
Now the OCVP is operating in the Operational Amplifier input offset cancellation mode.

Setp 2. Set OCVPDA [7:0]=40H

Setp 3. Set OCVPOOF [5:0]=000000 and read the OCVPCOUT bit

Setp 4. Increase the OCVPOOF [5:0] value by 1 and then read the OCVPCOUT bit.
If the OCVPCOUT bit state is not changed, then repeat Step 4 until the OCVPCOUT bit state is changed.
If the OCVPCOUT bit state is changed, then record the current OCVPOOF field value as VOOS1 and then go to Step 5.

Setp 5. Set OCVPOOF [5:0]=111111 and read the OCVPCOUT bit

Setp 6. Decrease the OCVPOOF [5:0] value by 1 and then read the OCVPCOUT bit.
If the OCVPCOUT bit state is not changed, then repeat Step 6 until the OCVPCOUT bit state is changed.
If the OCVPCOUT bit state is changed, then record the current OCVPOOF field value as VOOS2 and then go to Step 7.

Setp 7. Restore the operational amplifier input offset cancellation value VOOS into the OCVPOOF [5:0] bit field. The offset cancellation procedure is now finished with the VOOS value as the following formula shown.

$$\mathbf{VOOS} = \frac{\mathbf{VOOS1} + \mathbf{VOOS2}}{\mathbf{2}}$$

Comparator Input Offset Cancellation

Before the offset cancellationexecuted, the hysteresis voltage shoule be zero by setting the OCVPCHY bit to 0.

Setp 1. Set OCVPSW [7:0]=28H (S3 and S5 are ON, other switches are OFF).
Set OCVPCOFM=1, OCVPOOFM=0, OCVPCRSP=0
Now the OCVP is operating in the Comparator input offset cancellation mode.

Setp 2. Set OCVPDA [7:0]=40H

Setp 3. Set OCVPCOF [4:0]=00000 and read the OCVPCOUT bit

Setp 4. Increase the OCVPCOF [4:0] value by 1 and then read the OCVPCOUT bit.
If the OCVPCOUT bit state is not changed, then repeat Step 4 until the OCVPCOUT bit state is changed.
If the OCVPCOUT bit state is changed, then record the current OCVPCOF field value as VCOS1 and then go to Step 5.

Setp 5. Set OCVPCOF [4:0]=11111 and read the OCVPCOUT bit

Setp 6. Decrease the OCVPCOF [4:0] value by 1 and then read the OCVPCOUT bit.
If the OCVPCOUT bit state is not changed, then repeat Step 6 until the OCVPCOUT bit state is changed.
If the OCVPCOUT bit state is changed, then record the current OCVPCOF field value as VCOS2 and then go to Step 7.

Setp 7. Restore the comparator input offset cancellation value VCOS into the OCVPCOF [4:0] bit field. The offset cancellation procedure is now finished with the VCOS value as the following formula shown.

$$\mathbf{VCOS} = \frac{\mathbf{VCOS1} + \mathbf{VCOS2}}{\mathbf{2}}$$

OCVP Interrupt

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

Programming Considerations

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

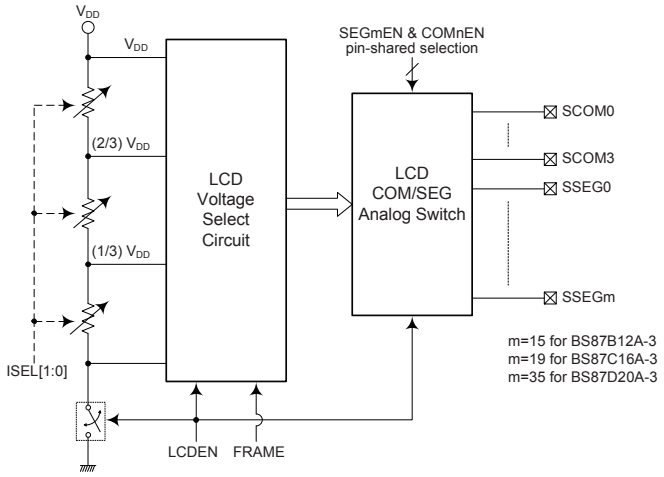
Software Controlled LCD Driver

The devices have the capability of driving external LCD panels. The common pins, SCOM0~SCOM3, and segment pins, SSEG0~SSEGm, where m is dependent upon which device is selected, for LCD driving are pin-shared with certain pins on the I/O ports. The LCD signals (COM and SEG) are generated using the application program.

LCD Operation

An external LCD panel can be driven using the devices by configuring the I/O pins as common pins and segment pins. The LCD driver function is controlled using the LCD control registers which in addition to controlling the overall on/off function also controls the R-type bias current on the SCOMn and SSEGm pins. This enables the LCD COM driver to generate the necessary voltage levels, V_{SS}, (1/3)V_{DD}, (2/3)V_{DD} and V_{DD}, for LCD 1/3 bias operation.

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



Software Controlled LCD Driver Structure

Device	SCOM Pins	SSEG Pins
BS87B12A-3	SCOM0~SCOM3	SSEG0~SSEG15
BS87C16A-3	SCOM0~SCOM3	SSEG0~SSEG19
BS87D20A-3	SCOM0~SCOM3	SSEG0~SSEG35

Software Controlled LCD Driver Pins Summary

LCD Frames

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

- **Frame 0**

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

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

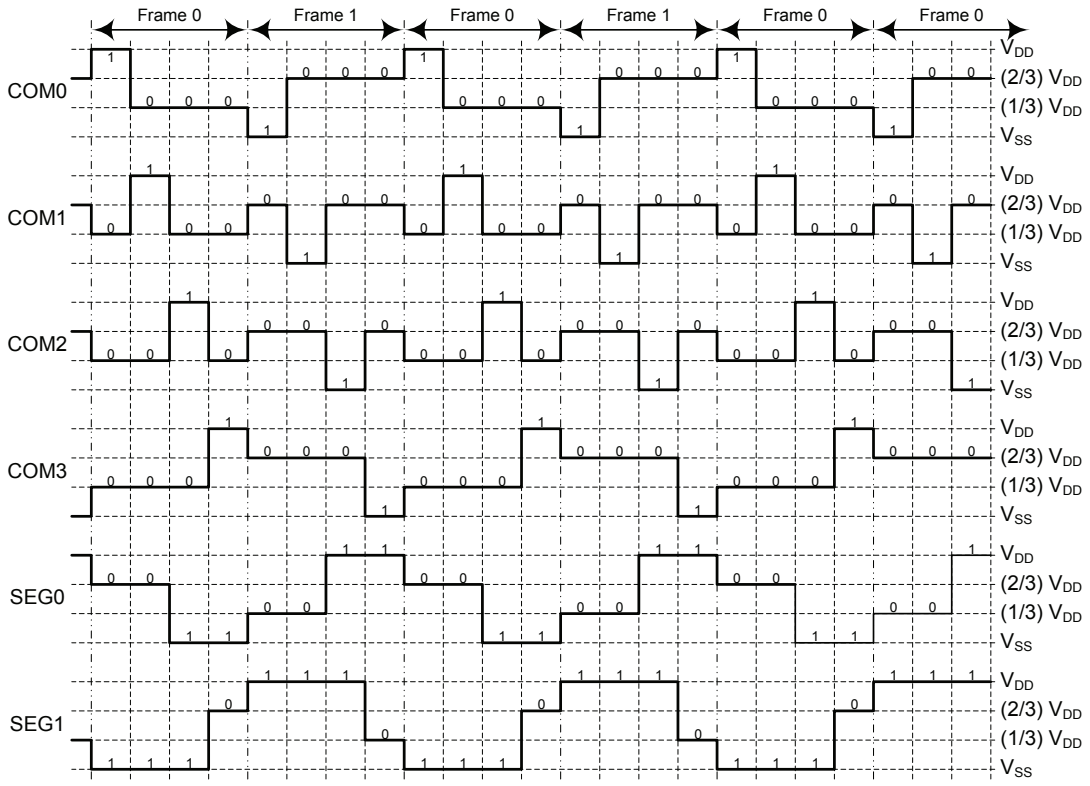
- **Frame 1**

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

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

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

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



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

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

LCD Control Registers

The LCD COM and SEG driver enables a range of selections to be provided to suit the requirement of the LCD panel which are being used. The bias current choice is implemented using the ISEL1 and ISEL0 bits in the SLCDC0 register. All COM and SEG pins are pin-shared with I/O pins and selected as SCOMn and SSEGm pins using the corresponding pin-shared function selection bits.

Register Name	Bit							
	7	6	5	4	3	2	1	0
SLCDC0	FRAME	ISEL1	ISEL0	LCDEN	COM3EN	COM2EN	COM1EN	COM0EN
SLCDC1	SEG7EN	SEG6EN	SEG5EN	SEG4EN	SEG3EN	SEG2EN	SEG1EN	SEG0EN
SLCDC2	SEG15EN	SEG14EN	SEG13EN	SEG12EN	SEG11EN	SEG10EN	SEG9EN	SEG8EN

Software Controlled LCD Registers List – BS87B12A-3

Register Name	Bit							
	7	6	5	4	3	2	1	0
SLCDC0	FRAME	ISEL1	ISEL0	LCDEN	COM3EN	COM2EN	COM1EN	COM0EN
SLCDC1	SEG7EN	SEG6EN	SEG5EN	SEG4EN	SEG3EN	SEG2EN	SEG1EN	SEG0EN
SLCDC2	SEG15EN	SEG14EN	SEG13EN	SEG12EN	SEG11EN	SEG10EN	SEG9EN	SEG8EN
SLCDC3	—	—	—	—	SEG19EN	SEG18EN	SEG17EN	SEG16EN

Software Controlled LCD Registers List – BS87C16A-3

Register Name	Bit							
	7	6	5	4	3	2	1	0
SLCDC0	FRAME	ISEL1	ISEL0	LCDEN	COM3EN	COM2EN	COM1EN	COM0EN
SLCDC1	SEG7EN	SEG6EN	SEG5EN	SEG4EN	SEG3EN	SEG2EN	SEG1EN	SEG0EN
SLCDC2	SEG15EN	SEG14EN	SEG13EN	SEG12EN	SEG11EN	SEG10EN	SEG9EN	SEG8EN
SLCDC3	SEG23EN	SEG22EN	SEG21EN	SEG20EN	SEG19EN	SEG18EN	SEG17EN	SEG16EN
SLCDC4	SEG31EN	SEG30EN	SEG29EN	SEG28EN	SEG27EN	SEG26EN	SEG25EN	SEG24EN
SLCDC5	—	—	—	—	SEG35EN	SEG34EN	SEG33EN	SEG32EN

Software Controlled LCD Registers List – BS87D20A-3

SLCDC0 Register

Bit	7	6	5	4	3	2	1	0
Name	FRAME	ISEL1	ISEL0	LCDEN	COM3EN	COM2EN	COM1EN	COM0EN
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 **FRAME**: Frame 0 or Frame 1 outptu selection
0: Frame 0
1: Frame 1

Bit 6~5 **ISEL1~ISEL0**: SCOM/SSEG typical bias current selection (@V_{DD}=5V)
00: 8.3μA
01: 16.7μA
10: 50μA
11: 100μA

Bit 4 **LCDEN**: Software controlled LCD Driver enable control
0: Disable
1: Enable

The SCOMn and SSEGm lines can be enabled using the corresponding pin-shared selection bits, COMnEN and SEGmEN, if the LCDEN bit is set to 1. When the LCDEN bit is cleared to 0, then the SCOMn and SSEGm outputs will be fixed at a V_{SS} level. Note that the corresponding pin-shared selection bits should first be properly configured before the SCOMn or SSEGm function is enabled.

Bit 3 **COM3EN**: SCOM3 pin function control
0: Disable – I/O or other pin-shared functions
1: Enable – SCOM3

Bit 2 **COM2EN**: SCOM2 pin function control
0: Disable – I/O or other pin-shared functions
1: Enable – SCOM2

Bit 1 **COM1EN**: SCOM1 pin function control
0: Disable – I/O or other pin-shared functions
1: Enable – SCOM1

Bit 0 **COM0EN**: SCOM0 pin function control
0: Disable – I/O or other pin-shared functions
1: Enable – SCOM0

SLCDC1 Register

Bit	7	6	5	4	3	2	1	0
Name	SEG7EN	SEG6EN	SEG5EN	SEG4EN	SEG3EN	SEG2EN	SEG1EN	SEG0EN
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 **SEG7EN~SEG0EN**: SSEG7~SSEG0 individual pin function control
0: Disable – I/O or other pin-shared functions
1: Enable – SSEG7~SSEG0

SLCDC2 Register

Bit	7	6	5	4	3	2	1	0
Name	SEG15EN	SEG14EN	SEG13EN	SEG12EN	SEG11EN	SEG10EN	SEG9EN	SEG8EN
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 **SEG15EN~SEG8EN**: SSEG15~SSEG8 individual pin function control
0: Disable – I/O or other pin-shared functions
1: Enable – SSEG15~SSEG8

SLCDC3 Register – BS87C16A-3

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	SEG19EN	SEG18EN	SEG17EN	SEG16EN
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0

Bit 7~4 Unimplemented, read as "0"

Bit 3~0 **SEG19EN~SEG16EN**: SSEG19~SSEG16 individual pin function control
0: Disable – I/O or other pin-shared functions
1: Enable – SSEG19~SSEG16

SLCDC3 Register – BS87D20A-3

Bit	7	6	5	4	3	2	1	0
Name	SEG23EN	SEG22EN	SEG21EN	SEG20EN	SEG19EN	SEG18EN	SEG17EN	SEG16EN
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 **SEG23EN~SEG16EN**: SSEG23~SSEG16 individual pin function control
0: Disable – I/O or other pin-shared functions
1: Enable – SSEG23~SSEG16

SLCDC4 Register – BS87D20A-3

Bit	7	6	5	4	3	2	1	0
Name	SEG31EN	SEG30EN	SEG29EN	SEG28EN	SEG27EN	SEG26EN	SEG25EN	SEG24EN
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 **SEG31EN~SEG24EN**: SSEG31~SSEG24 individual pin function control
0: Disable – I/O or other pin-shared functions
1: Enable – SSEG31~SSEG24

SLCDC5 Register – BS87D20A-3

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	SEG35EN	SEG34EN	SEG33EN	SEG32EN
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0

Bit 7~4 Unimplemented, read as "0"

Bit 3~0 **SEG35EN~SEG32EN**: SSEG35~SSEG32 individual pin function control
0: Disable – I/O or other pin-shared functions
1: Enable – SSEG35~SSEG32

Low Voltage Detector – LVD

Each device has a Low Voltage Detector function, also known as LVD. This enabled the device to monitor the power supply voltage, V_{DD}, and provide a warning signal should it fall below a certain level. This function may be especially useful in battery applications where the supply voltage will gradually reduce as the battery ages, as it allows an early warning battery low signal to be generated. The Low Voltage Detector also has the capability of generating an interrupt signal.

LVD Register

The Low Voltage Detector function is controlled using a single register with the name LVDC. Three bits in this register, VLVD2~VLVD0, are used to select one of five fixed voltages below which a low voltage condition will be determined. A low voltage condition is indicated when the LVDO bit is set. If the LVDO bit is low, this indicates that the V_{DD} voltage is above the preset low voltage value. The LVDEN bit is used to control the overall on/off function of the low voltage detector. Setting the bit high will enable the low voltage detector. Clearing the bit to zero will switch off the internal low voltage detector circuits. As the low voltage detector will consume a certain amount of power, it may be desirable to switch off the circuit when not in use, an important consideration in power sensitive battery powered applications.

LVDC Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	LVDO	LVDEN	VBGEN	VLVD2	VLVD1	VLVD0
R/W	—	—	R	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 **LVDO**: LVD output flag
0: No Low Voltage Detected
1: Low Voltage Detected

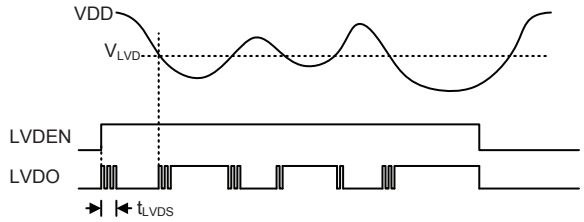
Bit 4 **LVDEN**: Low Voltage Detector Enable control
0: Disable
1: Enable

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

Bit 2~0 **VLVD2~VLVD0**: LVD Voltage selection
000: 2.0V
001: 2.2V
010: 2.4V
011: 2.7V
100: 3.0V
101: 3.3V
110: 3.6V
111: 4.0V
Since the V_{LVR} is fixed at 2.55V, the V_{LVD} should be set to 2.7V~4.0V.

LVD Operation

The Low Voltage Detector function operates by comparing the power supply voltage, V_{DD}, with a pre-specified voltage level stored in the LVDC register. This has a range of between 2.7V and 4.0V. When the power supply voltage, V_{DD}, falls below this pre-determined value, the LVDO bit will be set high indicating a low power supply voltage condition. The Low Voltage Detector function is supplied by a reference voltage which will be automatically enabled. When the device is powered down the low voltage detector will remain active if the LVDEN bit is high. After enabling the Low Voltage Detector, a time delay t_{LVDS} should be allowed for the circuitry to stabilise before reading the LVDO bit. Note also that as the V_{DD} voltage may rise and fall rather slowly, at the voltage nears that of V_{LVD}, there may be multiple bit LVDO transitions.



LVD Operation

The Low Voltage Detector also has its own interrupt which is contained within one of the Multi-function interrupts, providing an alternative means of low voltage detection, in addition to polling the LVDO bit. The interrupt will only be generated after a delay of t_{LVD} after the LVDO bit has been set high by a low voltage condition. When the device is powered down the Low Voltage Detector will remain active if the LVDEN bit is high. In this case, the LVF interrupt request flag will be set, causing an interrupt to be generated if V_{DD} falls below the preset LVD voltage. This will cause the device to wake-up from the SLEEP or IDLE Mode, however if the Low Voltage Detector wake up function is not required then the LVF flag should be first set high before the device enters the SLEEP or IDLE Mode.

Interrupts

Interrupts are an important part of any microcontroller system. When an external event or an internal function such as a Timer Module or an A/D converter requires microcontroller attention, their corresponding interrupt will enforce a temporary suspension of the main program allowing the microcontroller to direct attention to their respective needs. These devices contain several external interrupt and internal interrupts functions. The external interrupts are generated by the action of the external INT0 ~ INT3 pins, while the internal interrupts are generated by various internal functions such as the TMs, Time Base, LVD, EEPROM, SIM, UART and the A/D converter, etc.

Interrupt Registers

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

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

Function	Enable Bit	Request Flag	Notes
Global	EMI	—	—
INT Pin	INTE	INTF	—
Touch Key Module	TKME	TKMF	—
OCVP	OCVPE	OCVPF	—
Multi-function	MFE	MFF	For BS87C16A-3 & BS87D20A-3 only
SIM	SIME	SIMF	—
EEPROM write operation	DEE	DEF	—
UART	URnE	URnF	—
LVD	LVE	LVF	—
A/D Converter	ADE	ADF	—
Time Base	TBnE	TBnF	n=0 ~ 1
PTM	PTMnPE	PTMnPF	n=0 for BS87B12A-3 n=0 ~ 1 for BS87C16A-3 & BS87D20A-3
	PTMnAE	PTMnAF	
CTM	CTMnPE	CTMnPF	n=0 for BS87B12A-3 & BS87C16A-3 n=0 ~ 1 for BS87D20A-3
	CTMnAE	CTMnAF	

Interrupt Register Bit Naming Conventions

Register Name	Bit							
	7	6	5	4	3	2	1	0
INTEG	—	—	—	—	—	—	INTS1	INTS0
INTC0	—	OCVPF	TKMF	INTF	OCVPE	TKME	INTE	EMI
INTC1	PTM0AF	PTM0PF	CTM0AF	CTM0PF	PTM0AE	PTM0PE	CTM0AE	CTM0PE
INTC2	URF	DEF	SIMF	—	URE	DEE	SIME	—
INTC3	TB1F	TB0F	ADF	LVF	TB1E	TB0E	ADE	LVE

Interrupt Registers List – BS87B12A-3

Register Name	Bit							
	7	6	5	4	3	2	1	0
INTEG	—	—	—	—	—	—	INTS1	INTS0
INTC0	—	OCVPF	TKMF	INTF	OCVPE	TKME	INTE	EMI
INTC1	PTM0AF	PTM0PF	CTM0AF	CTM0PF	PTM0AE	PTM0PE	CTM0AE	CTM0PE
INTC2	URF	DEF	SIMF	MFF	URE	DEE	SIME	MFE
INTC3	TB1F	TB0F	ADF	LVF	TB1E	TB0E	ADE	LVE
MFI	—	—	PTM1AF	PTM1PF	—	—	PTM1AE	PTM1PE

Interrupt Registers List – BS87C16A-3

Register Name	Bit							
	7	6	5	4	3	2	1	0
INTEG	—	—	—	—	—	—	INTS1	INTS0
INTC0	—	OCVPF	TKMF	INTF	OCVPE	TKME	INTE	EMI
INTC1	PTM0AF	PTM0PF	CTM0AF	CTM0PF	PTM0AE	PTM0PE	CTM0AE	CTM0PE
INTC2	URF	DEF	SIMF	MFF	URE	DEE	SIME	MFE
INTC3	TB1F	TB0F	ADF	LVF	TB1E	TB0E	ADE	LVE
MFI	CTM1AF	CTM1PF	PTM1AF	PTM1PF	CTM1AE	CTM1PE	PTM1AE	PTM1PE

Interrupt Registers List – BS87D20A-3

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	—	OCVPF	TKMF	INTF	OCVPE	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 **OCVPF**: OCVP interrupt request flag
- 0: no request
- 1: interrupt request

Bit 5 **TKMF**: Touch key module interrupt request flag
- 0: No request
- 1: Interrupt request

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

Bit 3 **OCVPE**: OCVP interrupt control
- 0: Disable
- 1: Enable

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

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

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

INTC1 Register

Bit	7	6	5	4	3	2	1	0
Name	PTM0AF	PTM0PF	CTM0AF	CTM0PF	PTM0AE	PTM0PE	CTM0AE	CTM0PE
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7 **PTM0AF**: PTM0 Comparator A match Interrupt request flag
0: No request
1: Interrupt request

Bit 6 **PTM0PF**: PTM0 Comparator P match Interrupt request flag
0: No request
1: Interrupt request

Bit 5 **CTM0AF**: CTM0 Comparator A match Interrupt request flag
0: No request
1: Interrupt request

Bit 4 **CTM0PF**: CTM0 Comparator P match Interrupt request flag
0: No request
1: Interrupt request

Bit 3 **PTM0AE**: PTM0 Comparator A match Interrupt control
0: Disable
1: Enable

Bit 2 **PTM0PE**: PTM0 Comparator P match Interrupt control
0: Disable
1: Enable

Bit 1 **CTM0AE**: CTM0 Comparator A match Interrupt control
0: Disable
1: Enable

Bit 0 **CTM0PE**: CTM0 Comparator P match Interrupt control
0: Disable
1: Enable

INTC2 Register – BS87B12A-3

Bit	7	6	5	4	3	2	1	0
Name	URF	DEF	SIMF	—	URE	DEE	SIME	—
R/W	R/W	R/W	R/W	—	R/W	R/W	R/W	—
POR	0	0	0	—	0	0	0	—

Bit 7 **URF**: UART transfer interrupt request flag
0: No request
1: Interrupt request

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

Bit 5 **SIMF**: SIM Interrupt request flag
0: No request
1: Interrupt request

Bit 4 Unimplemented, read as "0"

Bit 3 **URE**: UART transfer interrupt control
0: Disable
1: Enable

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

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

Bit 0 Unimplemented, read as "0"

INTC2 Register – BS87C16A-3 & BS87D20A-3

Bit	7	6	5	4	3	2	1	0
Name	URF	DEF	SIMF	MFF	URE	DEE	SIME	MFE
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 **URF**: UART transfer interrupt request flag
0: No request
1: Interrupt request

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

Bit 5 **SIMF**: SIM Interrupt request flag
0: No request
1: Interrupt request

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

Bit 3 **URE**: UART transfer interrupt control
0: Disable
1: Enable

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

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

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

INTC3 Register

Bit	7	6	5	4	3	2	1	0
Name	TB1F	TB0F	ADF	LVF	TB1E	TB0E	ADE	LVE
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 **TB1F**: Time Base 1 interrupt request flag
0: No request
1: Interrupt request

Bit 6 **TB0F**: Time Base 0 interrupt request flag
0: No request
1: Interrupt request

Bit 5 **ADF**: A/D Converter interrupt request flag
0: No request
1: Interrupt request

Bit 4 **LVF**: LVD Interrupt request flag
0: No request
1: Interrupt request

Bit 3 **TB1E**: Time Base 1 interrupt control
0: Disable
1: Enable

Bit 2 **TB0E**: Time Base 0 interrupt control
0: Disable
1: Enable

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

Bit 0 **LVE**: LVD Interrupt control
0: Disable
1: Enable

MFI Register – BS87C16A-3

Bit	7	6	5	4	3	2	1	0
Name	—	—	PTM1AF	PTM1PF	—	—	PTM1AE	PTM1PE
R/W	—	—	R/W	R/W	—	—	R/W	R/W
POR	—	—	0	0	—	—	0	0

Bit 7~6 Unimplemented, read as "0"

Bit 5 **PTM1AF**: PTM1 Comparator A match Interrupt request flag
0: No request
1: Interrupt request

Bit 4 **PTM1PF**: PTM1 Comparator P match Interrupt request flag
0: No request
1: Interrupt request

Bit 3~2 Unimplemented, read as "0"

Bit 1 **PTM1AE**: PTM1 Comparator A match Interrupt control
0: Disable
1: Enable

Bit 0 **PTM1PE**: PTM1 Comparator P match Interrupt control
0: Disable
1: Enable

MFI Register – BS87D20A-3

Bit	7	6	5	4	3	2	1	0
Name	CTM1AF	CTM1PF	PTM1AF	PTM1PF	CTM1AE	CTM1PE	PTM1AE	PTM1PE
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 **CTM1AF**: CTM1 Comparator A match Interrupt request flag
0: No request
1: Interrupt request

Bit 6 **CTM1PF**: CTM1 Comparator P match Interrupt request flag
0: No request
1: Interrupt request

Bit 5 **PTM1AF**: PTM1 Comparator A match Interrupt request flag
0: No request
1: Interrupt request

Bit 4 **PTM1PF**: PTM1 Comparator P match Interrupt request flag
0: No request
1: Interrupt request

Bit 3 **CTM1AE**: CTM1 Comparator A match Interrupt control
0: Disable
1: Enable

Bit 2 **CTM1PE**: CTM1 Comparator P match Interrupt control
0: Disable
1: Enable

Bit 1 **PTM1AE**: PTM1 Comparator A match Interrupt control
0: Disable
1: Enable

Bit 0 **PTM1PE**: PTM1 Comparator P match Interrupt control
0: Disable
1: Enable

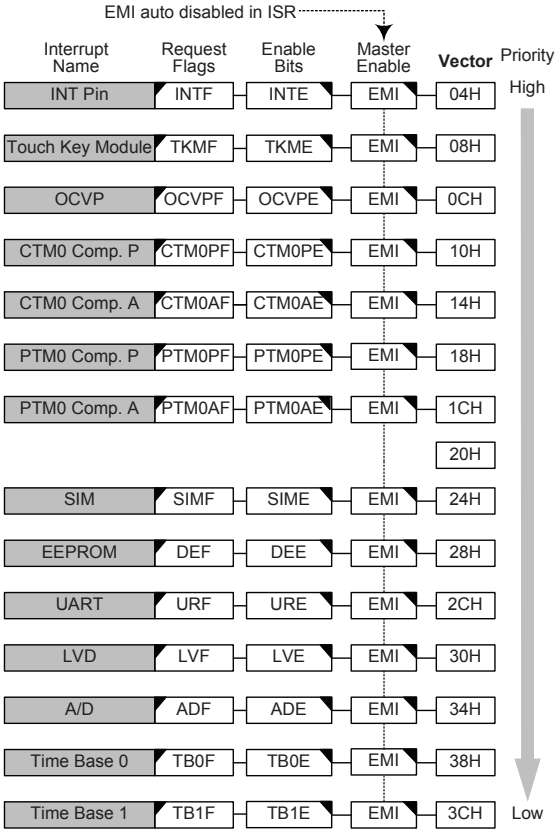
Interrupt Operation

When the conditions for an interrupt event occur, such as a Touch Key Counter overflow, a TM Comparator P or Comparator A match or A/D conversion completion, etc, the relevant interrupt request flag will be set. Whether the request flag actually generates a program jump to the relevant interrupt vector is determined by the condition of the interrupt enable bit. If the enable bit is set high then the program will jump to its relevant vector; if the enable bit is zero then although the interrupt request flag is set an actual interrupt will not be generated and the program will not jump to the relevant interrupt vector. The global interrupt enable bit, if cleared to zero, will disable all interrupts.

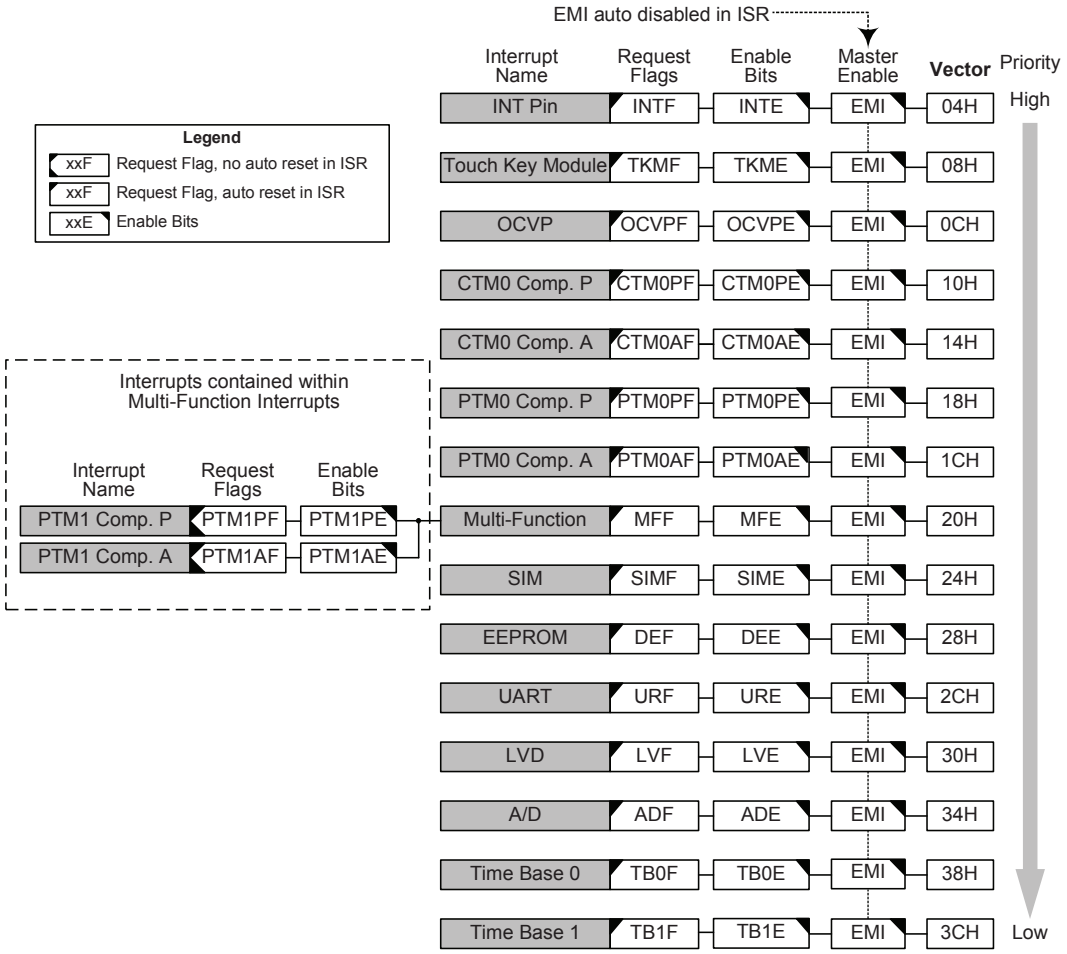
When an interrupt is generated, the Program Counter, which stores the address of the next instruction to be executed, will be transferred onto the stack. The Program Counter will then be loaded with a new address which will be the value of the corresponding interrupt vector. The microcontroller will then fetch its next instruction from this interrupt vector. The instruction at this vector will usually be a JMP which will jump to another section of program which is known as the interrupt service routine. Here is located the code to control the appropriate interrupt. The interrupt service routine must be terminated with a RETI, which retrieves the original Program Counter address from the stack and allows the microcontroller to continue with normal execution at the point where the interrupt occurred.

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

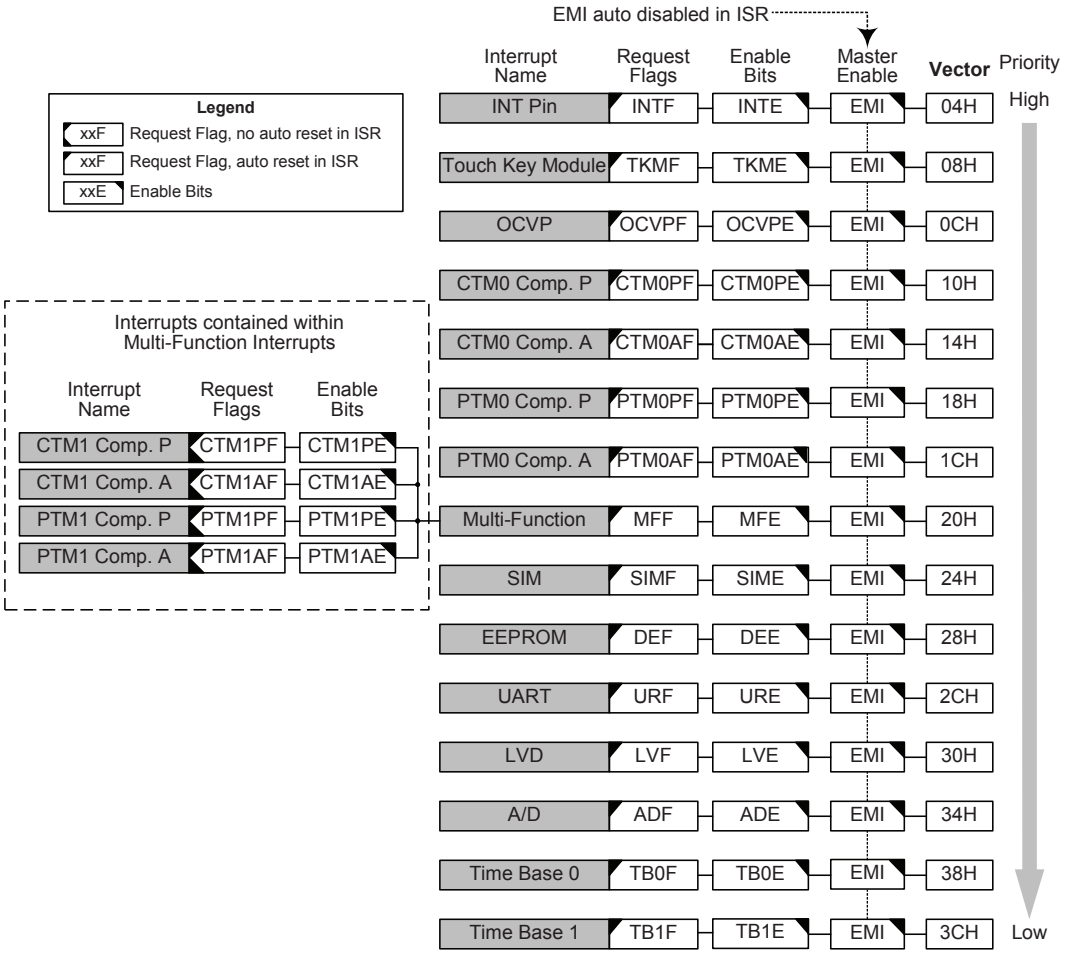
If an interrupt requires immediate servicing while the program is already in another interrupt service routine, the EMI bit should be set after entering the routine to allow interrupt nesting. If the stack is full, the interrupt request will not be acknowledged, even if the related interrupt is enabled,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



Interrupt Structure – BS87B12A-3



Interrupt Structure – BS87C16A-3



Interrupt Structure – BS87D20A-3

External Interrupt

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

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

Touch Key Module Interrupt

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

OCVP Interrupt

An OCVP interrupt request will take place when the OCVP interrupt request flag, OCVPF, is set, which occurs when the OCVP circuit detects a specific current or voltage condition. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and the OCVP interrupt enable bit, OCVPE, must first be set. When the interrupt is enabled, the stack is not full and a user-defined current or voltage condition occurs, a subroutine call to the OCVP interrupt vector will take place. When the OCVP interrupt is serviced, the EMI bit will automatically be cleared to disable other interrupts and the OCVP interrupt request flag will also be automatically cleared.

TM Interrupts

The Compact and Periodic TMs have two interrupts, one comes from the comparator A match situation and the other comes from the comparator P match situation. The CTM0 and PTM0 interrupts have their own individual interrupt vectors respectively while the CTM1 and PTM1 are contained within the Multi-function Interrupts. For the CTM0 and PTM0 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, and respective TM Interrupt enable bit must first be set for CTM0 and PTM0. However, the relevant Multi-function Interrupt enable bit, MFE, must also be set for CTM1 or PTM1. When the interrupt is enabled, the stack is not full and a TM comparator match situation occurs, a subroutine call to the relevant TM Interrupt vector locations will take place. When the TM interrupt is serviced, the EMI bit will be automatically cleared to disable other interrupts. The CTM0 or PTM0 interrupt request flag will automatically be cleared. However, for CTM1 or PTM1 only the related MFF flag will be automatically cleared. The CTM1 or PTM1 interrupt request flags will be kept unchanged. As the CTM or PTM1 interrupt request flags will not be automatically cleared, they have to be cleared by the application program.

Multi-function Interrupt – BS87C16A-3 & BS87D20A-3

Within the specific device there is one Multi-function interrupts. Unlike the other independent interrupts, this interrupt has no independent source, but rather are formed from other existing interrupt sources, namely the PTM1 or CTM1/PTM1 interrupts, dependent upon which device is selected.

A Multi-function interrupt request will take place when the Multi-function interrupt request flag MFF are set. The Multi-function interrupt flag will be set when any of its included functions generate an interrupt request flag. To allow the program to branch to its respective interrupt vector address, when the Multi-function interrupt is enabled and the stack is not full, and one of the interrupts contained within the Multi-function interrupt occurs, a subroutine call to 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 flag will be automatically reset when the interrupt is serviced, the request flags from the original source of the Multi-function interrupt will not be automatically reset and must be manually reset by the application program.

Serial Interface Module Interrupt

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

EEPROM Interrupt

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

UART Transfer Interrupt

The UART Transfer Interrupt is controlled by several UART transfer conditions. When one of these conditions occurs, an interrupt pulse will be generated to get the attention of the microcontroller. These conditions are a transmitter data register empty, transmitter idle, receiver data available, receiver overrun, address detect and an RX pin wake-up. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and UART Interrupt enable bit, URE, must first be set. When the interrupt is enabled, the stack is not full and any of the conditions described above occurs, a subroutine call to the UART Interrupt vector, will take place. When the interrupt is serviced, the UART Interrupt flag, URF, will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts.

LVD Interrupt

The Low Voltage Detector Interrupt is an individual interrupt source with its own interrupt cevtor. An LVD Interrupt request will take place when the LVD Interrupt request flag, LVF, is set, which occurs when the Low Voltage Detector function detects a low power supply voltage. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and Low Voltage Interrupt enable bit, LVE, must first be set. When the interrupt is enabled, the stack is not full and a low voltage condition occurs, a subroutine call to the relevant interrupt vector will take place. When the Low Voltage Interrupt is serviced, the LVF flag will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts.

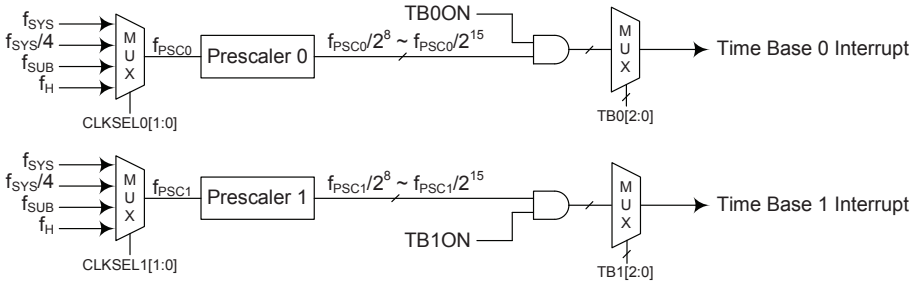
A/D Converter Interrupt

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

Time Base Interrupts

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

The purpose of the Time Base Interrupt is to provide an interrupt signal at fixed time periods. Its clock source, f_{PSC0} or f_{PSC1}, originates from the internal clock source f_{SYS}, $f_{SYS}/4$, f_{SUB} or f_H and then passes through a divider, the division ratio of which is selected by programming the appropriate bits in the TBC register to obtain longer interrupt periods whose value ranges. The clock source which in turn controls the Time Base interrupt period is selected using the CLKSEL0[1:0] and CLKSEL1[1:0] bits in the PSCR register respectively.



Time Base Interrupts

PSCR Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	CLKSEL11	CLKSEL10	—	—	CLKSEL01	CLKSEL00
R/W	—	—	R/W	R/W	—	—	R/W	R/W
POR	—	—	0	0	—	—	0	0

Bit 7~6 unimplemented, read as "0"

Bit 5~4 **CLKSEL11~CLKSEL10**: Prescaler 1 clock source f_{PSC1} selection
00: f_{SYS}
01: $f_{SYS}/4$
10: f_{SUB}
11: f_H

Bit 3~2 unimplemented, read as "0"

Bit 1~0 **CLKSEL01~CLKSEL00**: Prescaler 0 clock source f_{PSC0} selection
00: f_{SYS}
01: $f_{SYS}/4$
10: f_{SUB}
11: f_H

TBC Register

Bit	7	6	5	4	3	2	1	0
Name	TB1ON	TB12	TB11	TB10	TB0ON	TB02	TB01	TB00
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 **TB1ON**: Time Base 1 Enable Control
0: Disable
1: Enable

Bit 6~4 **TB12~TB10**: Time Base 1 time-out period selection
000: $2^8/f_{PSC1}$
001: $2^9/f_{PSC1}$
010: $2^{10}/f_{PSC1}$
011: $2^{11}/f_{PSC1}$
100: $2^{12}/f_{PSC1}$
101: $2^{13}/f_{PSC1}$
110: $2^{14}/f_{PSC1}$
111: $2^{15}/f_{PSC1}$

Bit 3 **TB0ON**: Time Base 0 Enable Control
0: Disable
1: Enable

Bit 2~0 **TB02~TB00**: Time Base 0 time-out period selection
000: $2^8/f_{PSC0}$
001: $2^9/f_{PSC0}$
010: $2^{10}/f_{PSC0}$
011: $2^{11}/f_{PSC0}$
100: $2^{12}/f_{PSC0}$
101: $2^{13}/f_{PSC0}$
110: $2^{14}/f_{PSC0}$
111: $2^{15}/f_{PSC0}$

Interrupt Wake-up Function

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

Programming Considerations

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

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

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

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

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

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

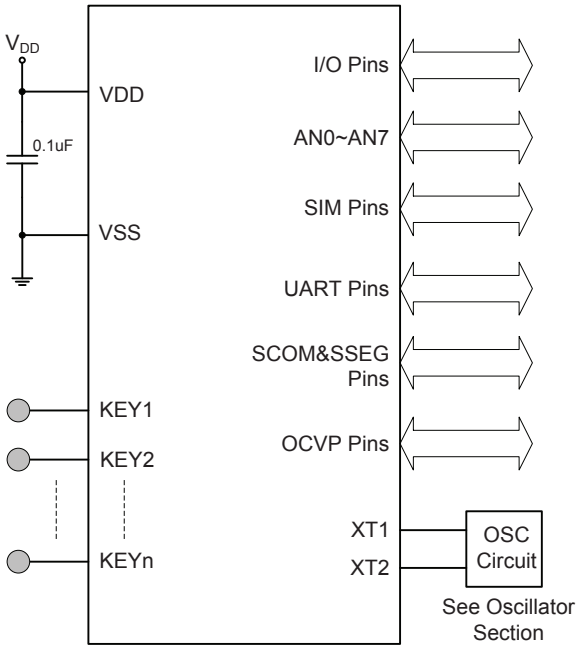
Configuration Options

Configuration options refer to certain options within the MCU that are programmed into the device during the programming process. During the development process, these options are selected using the HT-IDE software development tools. 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	Low Speed System Oscillator Selection – f_{SUB}: LIRC or LXT
2	HIRC Frequency Selection – f_{HIRC}: 8MHz, 12MHz or 16MHz

Note that when the HIRC has been configured at a frequency shown in this table the HIRCS1 and HIRCS0 bits are recommended to be setup to select the same frequency to keep the HIRC frequency accuracy specified in the A.C characteristics.

Application Circuits



Instruction Set

Introduction

Central to the successful operation of any microcontroller is its instruction set, which is a set of program instruction codes that directs the microcontroller to perform certain operations. In the case of Holtek microcontroller, a comprehensive and flexible set of over 60 instructions is provided to enable programmers to implement their application with the minimum of programming overheads.

For easier understanding of the various instruction codes, they have been subdivided into several functional groupings.

Instruction Timing

Most instructions are implemented within one instruction cycle. The exceptions to this are branch, call, or table read instructions where two instruction cycles are required. One instruction cycle is equal to 4 system clock cycles, therefore in the case of an 8MHz system oscillator, most instructions would be implemented within 0.5μs and branch or call instructions would be implemented within 1μs. Although instructions which require one more cycle to implement are generally limited to the JMP, CALL, RET, RETI and table read instructions, it is important to realize that any other instructions which involve manipulation of the Program Counter Low register or PCL will also take one more cycle to implement. As instructions which change the contents of the PCL will imply a direct jump to that new address, one more cycle will be required. Examples of such instructions would be "CLR PCL" or "MOV PCL, A". For the case of skip instructions, it must be noted that if the result of the comparison involves a skip operation then this will also take one more cycle, if no skip is involved then only one cycle is required.

Moving and Transferring Data

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

Arithmetic Operations

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

Logical and Rotate Operation

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

Branches and Control Transfer

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

Bit Operations

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

Table Read Operations

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

Other Operations

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

Instruction Set Summary

The 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	1Note	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	1Note	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	1Note	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	1Note	Z, C, AC, OV, SC, CZ
DAA [m]	Decimal adjust ACC for Addition with result in Data Memory	1Note	C
Logic Operation			
AND A,[m]	Logical AND Data Memory to ACC	1	Z
OR A,[m]	Logical OR Data Memory to ACC	1	Z
XOR A,[m]	Logical XOR Data Memory to ACC	1	Z
ANDM A,[m]	Logical AND ACC to Data Memory	1Note	Z
ORM A,[m]	Logical OR ACC to Data Memory	1Note	Z
XORM A,[m]	Logical XOR ACC to Data Memory	1Note	Z
AND A,x	Logical AND immediate Data to ACC	1	Z
OR A,x	Logical OR immediate Data to ACC	1	Z
XOR A,x	Logical XOR immediate Data to ACC	1	Z
CPL [m]	Complement Data Memory	1Note	Z
CPLA [m]	Complement Data Memory with result in ACC	1	Z
Increment & Decrement			
INCA [m]	Increment Data Memory with result in ACC	1	Z
INC [m]	Increment Data Memory	1Note	Z
DECA [m]	Decrement Data Memory with result in ACC	1	Z
DEC [m]	Decrement Data Memory	1Note	Z
Rotate			
RRA [m]	Rotate Data Memory right with result in ACC	1	None
RR [m]	Rotate Data Memory right	1Note	None
RRCA [m]	Rotate Data Memory right through Carry with result in ACC	1	C
RRC [m]	Rotate Data Memory right through Carry	1Note	C
RLA [m]	Rotate Data Memory left with result in ACC	1	None
RL [m]	Rotate Data Memory left	1Note	None
RLCA [m]	Rotate Data Memory left through Carry with result in ACC	1	C
RLC [m]	Rotate Data Memory left through Carry	1Note	C

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	1Note	None
MOV A,x	Move immediate data to ACC	1	None
Bit Operation			
CLR [m].i	Clear bit of Data Memory	1Note	None
SET [m].i	Set bit of Data Memory	1Note	None
Branch Operation			
JMP addr	Jump unconditionally	2	None
SZ [m]	Skip if Data Memory is zero	1Note	None
SZA [m]	Skip if Data Memory is zero with data movement to ACC	1Note	None
SZ [m].i	Skip if bit i of Data Memory is zero	1Note	None
SNZ [m]	Skip if Data Memory is not zero	1Note	None
SNZ [m].i	Skip if bit i of Data Memory is not zero	1Note	None
SIZ [m]	Skip if increment Data Memory is zero	1Note	None
SDZ [m]	Skip if decrement Data Memory is zero	1Note	None
SIZA [m]	Skip if increment Data Memory is zero with result in ACC	1Note	None
SDZA [m]	Skip if decrement Data Memory is zero with result in ACC	1Note	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	2Note	None
TABRDL [m]	Read table (last page) to TBLH and Data Memory	2Note	None
ITABRD [m]	Increment table pointer TBLP first and Read table to TBLH and Data Memory	2Note	None
ITABRDL [m]	Increment table pointer TBLP first and Read table (last page) to TBLH and Data Memory	2Note	None
Miscellaneous			
NOP	No operation	1	None
CLR [m]	Clear Data Memory	1Note	None
SET [m]	Set Data Memory	1Note	None
CLR WDT	Clear Watchdog Timer	1	TO, PDF
SWAP [m]	Swap nibbles of Data Memory	1Note	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	2Note	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	2Note	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	2Note	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	2Note	Z, C, AC, OV, SC, CZ
LDAA [m]	Decimal adjust ACC for Addition with result in Data Memory	2Note	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	2Note	Z
LORM A,[m]	Logical OR ACC to Data Memory	2Note	Z
LXORM A,[m]	Logical XOR ACC to Data Memory	2Note	Z
LCPL [m]	Complement Data Memory	2Note	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	2Note	Z
LDECA [m]	Decrement Data Memory with result in ACC	2	Z
LDEC [m]	Decrement Data Memory	2Note	Z
Rotate			
LRRA [m]	Rotate Data Memory right with result in ACC	2	None
LRR [m]	Rotate Data Memory right	2Note	None
LRRCA [m]	Rotate Data Memory right through Carry with result in ACC	2	C
LRRC [m]	Rotate Data Memory right through Carry	2Note	C
LRLA [m]	Rotate Data Memory left with result in ACC	2	None
LRL [m]	Rotate Data Memory left	2Note	None
LRLCA [m]	Rotate Data Memory left through Carry with result in ACC	2	C
LRLC [m]	Rotate Data Memory left through Carry	2Note	C
Data Move			
LMOV A,[m]	Move Data Memory to ACC	2	None
LMOV [m],A	Move ACC to Data Memory	2Note	None
Bit Operation			
LCLR [m].i	Clear bit of Data Memory	2Note	None
LSET [m].i	Set bit of Data Memory	2Note	None

Mnemonic	Description	Cycles	Flag Affected
Branch			
LSZ [m]	Skip if Data Memory is zero	2Note	None
LSZA [m]	Skip if Data Memory is zero with data movement to ACC	2Note	None
LSNZ [m]	Skip if Data Memory is not zero	2Note	None
LSZ [m].i	Skip if bit i of Data Memory is zero	2Note	None
LSNZ [m].i	Skip if bit i of Data Memory is not zero	2Note	None
LSIZ [m]	Skip if increment Data Memory is zero	2Note	None
LSDZ [m]	Skip if decrement Data Memory is zero	2Note	None
LSIZA [m]	Skip if increment Data Memory is zero with result in ACC	2Note	None
LSDZA [m]	Skip if decrement Data Memory is zero with result in ACC	2Note	None
Table Read			
LTABRD [m]	Read table to TBLH and Data Memory	3Note	None
LTABRDL [m]	Read table (last page) to TBLH and Data Memory	3Note	None
LITABRD [m]	Increment table pointer TBLP first and Read table to TBLH and Data Memory	3Note	None
LITABRDL [m]	Increment table pointer TBLP first and Read table (last page) to TBLH and Data Memory	3Note	None
Miscellaneous			
LCLR [m]	Clear Data Memory	2Note	None
LSET [m]	Set Data Memory	2Note	None
LSWAP [m]	Swap nibbles of Data Memory	2Note	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] ← $\overline{[m]}$

Affected flag(s) Z

CPLA [m] Complement Data Memory with result in ACC

Description Each bit of the specified Data Memory is logically complemented (1′s complement). Bits which previously contained a 1 are changed to 0 and vice versa. The complemented result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

Operation ACC ← $\overline{[m]}$

Affected flag(s) Z

DAA [m] Decimal-Adjust ACC for addition with result in Data Memory

Description Convert the contents of the Accumulator value to a BCD (Binary Coded Decimal) value resulting from the previous addition of two BCD variables. If the low nibble is greater than 9 or if AC flag is set, then a value of 6 will be added to the low nibble. Otherwise the low nibble remains unchanged. If the high nibble is greater than 9 or if the C flag is set, then a value of 6 will be added to the high nibble. Essentially, the decimal conversion is performed by adding 00H, 06H, 60H or 66H depending on the Accumulator and flag conditions. Only the C flag may be affected by this instruction which indicates that if the original BCD sum is greater than 100, it allows multiple precision decimal addition.

Operation [m] ← ACC + 00H or
[m] ← ACC + 06H or
[m] ← ACC + 60H or
[m] ← ACC + 66H

Affected flag(s) C

DEC [m] Decrement Data Memory

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

Operation [m] ← [m] − 1

Affected flag(s) Z

DECA [m] Decrement Data Memory with result in ACC

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

Operation ACC ← [m] − 1

Affected flag(s) Z

HALT Enter power down mode

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

Operation TO ← 0
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] − 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] − 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 $[m] \leftarrow \overline{[m]}$

Affected flag(s) Z

LCPLA [m] Complement Data Memory with result in ACC

Description Each bit of the specified Data Memory is logically complemented (1′s complement). Bits which previously contained a 1 are changed to 0 and vice versa. The complemented result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

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

Affected flag(s) Z

LDAA [m] Decimal-Adjust ACC for addition with result in Data Memory

Description Convert the contents of the Accumulator value to a BCD (Binary Coded Decimal) value resulting from the previous addition of two BCD variables. If the low nibble is greater than 9 or if AC flag is set, then a value of 6 will be added to the low nibble. Otherwise the low nibble remains unchanged. If the high nibble is greater than 9 or if the C flag is set, then a value of 6 will be added to the high nibble. Essentially, the decimal conversion is performed by adding 00H, 06H, 60H or 66H depending on the Accumulator and flag conditions. Only the C flag may be affected by this instruction which indicates that if the original BCD sum is greater than 100, it allows multiple precision decimal addition.

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

Affected flag(s) C

LDEC [m] Decrement Data Memory

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

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

Affected flag(s) Z

LDECA [m] Decrement Data Memory with result in ACC

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

Operation $ACC \leftarrow [m] - 1$

Affected flag(s) Z

LINC [m] Increment Data Memory

Description Data in the specified Data Memory is incremented by 1.

Operation $[m] \leftarrow [m] + 1$

Affected flag(s) Z

LINCA [m] Increment Data Memory with result in ACC

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

Operation $ACC \leftarrow [m] + 1$

Affected flag(s) Z

LMOV A,[m] Move Data Memory to ACC

Description The contents of the specified Data Memory are copied to the Accumulator.

Operation ACC ← [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 ACC ← ACC − [m] − C

Affected flag(s) OV, Z, AC, C, SC, CZ

LSBCM A,[m] Subtract Data Memory from ACC with Carry and result in Data Memory

Description The contents of the specified Data Memory and the complement of the carry flag are subtracted from the Accumulator. The result is stored in the Data Memory. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.

Operation [m] ← ACC − [m] − 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

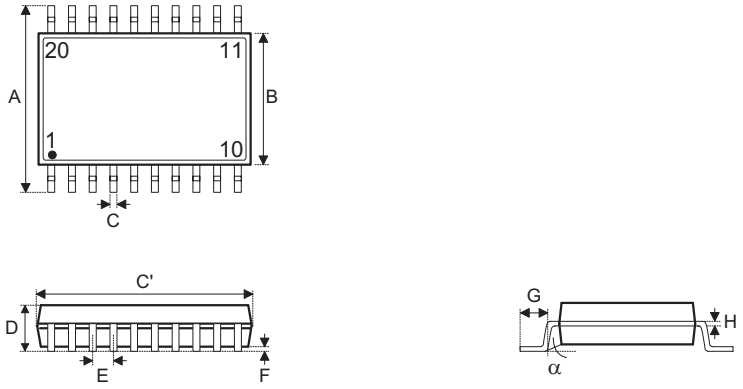
Package Information

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

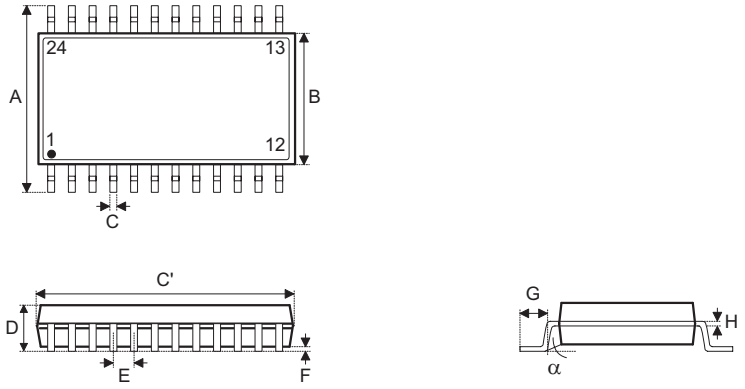
20-pin NSOP (150mil) Outline Dimensions



Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	0.228	0.236	0.244
B	0.146	0.154	0.161
C	0.009	—	0.012
C’	0.382	0.390	0.398
D	—	—	0.069
E	—	0.032 BSC	—
F	0.002	—	0.009
G	0.020	—	0.031
H	0.008	—	0.010
α	0°	—	8°

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	5.80	6.00	6.20
B	3.70	3.90	4.10
C	0.23	—	0.30
C’	9.70	9.90	10.10
D	—	—	1.75
E	—	0.80 BSC	—
F	0.05	—	0.23
G	0.50	—	0.80
H	0.21	—	0.25
α	0°	—	8°

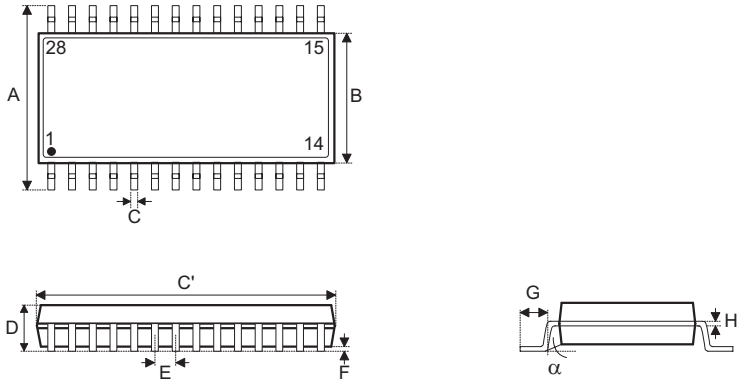
24-pin SOP (300mil) Outline Dimensions



Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	—	0.406 BSC	—
B	—	0.295 BSC	—
C	0.012	—	0.020
C’	—	0.606 BSC	—
D	—	—	0.104
E	—	0.050 BSC	—
F	0.004	—	0.012
G	0.016	—	0.050
H	0.008	—	0.013
α	0°	—	8°

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	—	10.30 BSC	—
B	—	7.5 BSC	—
C	0.31	—	0.51
C’	—	15.4 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°

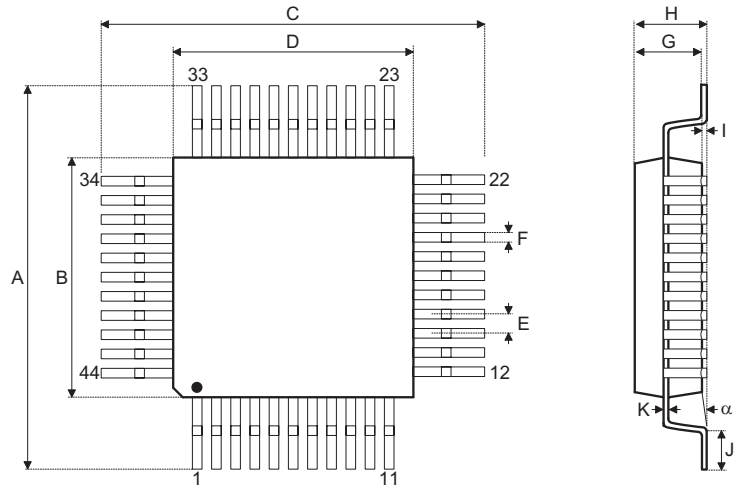
28-pin SOP (300mil) Outline Dimensions



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	—	7.5 BSC	—
C	0.31	—	0.51
C'	—	17.9 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°

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°