Touch A/D Flash MCU with LED/LCD Driver

BS86B12A-3/BS86C16A-3/BS86D20A-3

Revision: V1.50 Date: October 27, 2021

www.holtek.com

Table of Contents

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
- Three Oscillators
  - High Speed Internal RC -- HIRC: 8/12/16MHz
  - Low Speed Internal RC -- LIRC: 32kHz
  - Low speed External Crystal -- LXT: 32768Hz (for BS86C16A-3/BS86D20A-3 only)
- Multi-mode operation: NORMAL, 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

- Flash Program Memory: 2K×16~8K×16
- RAM Data Memory: 384×8~768×8
- True EEPROM Memory: 64×8
- Fully integrated 12/16/20 touch key functions -- require no external components
- Watchdog Timer function
- Up to 26 bidirectional I/O lines
- PMOS Source Current Adjustable
- Software controlled 4-SCOM lines LCD driver with 1/3 bias
- One external interrupt line shared with I/O pin
- Multiple Timer Module for time measure, input capture, compare match output, PWM output or single pulse output function
- Dual Time-Base functions for generation of fixed time interrupt signals
- Multi-channel 12-Bit resolution A/D converter
- Serial Interfaces Module – SIM for SPI or I^2C
- UART Interface
- Low voltage reset function
- Low voltage detect function
- Flash program memory can be re-programmed up to 10,000 times
- Flash program memory data retention > 40 years
- True EEPROM data memory can be re-programmed up to 100,000 times
- True EEPROM data memory data retention > 40 years
- Package: 20/24/28-pin SOP

Development Tools

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

https://www.holtek.com/esk-bs-210 (BS86C16A-3/BS86D20A-3 only)

General Description

These devices are a series of Flash Memory type 8-bit high performance RISC architecture microcontrollers with fully integrated touch key functions. With all touch key functions provided internally and with the convenience of Flash Memory multi-programming features, these devices have 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 feature includes a multi-channel 12-bit A/D converter. 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 a fully integrated system oscillator 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 these 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.

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

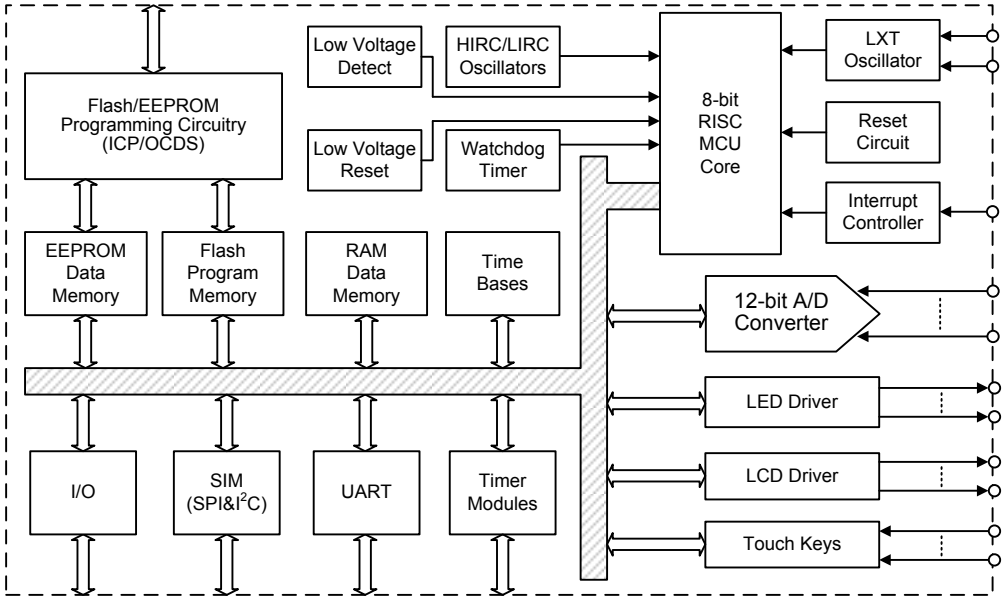
Selection Table

Most features are common to these devices, the main features distinguishing them are Memory capacity, I/O count, LCD driver segment count, Touch Key count, stack capacity amd package types. The following table summarises the main features of each device.

Part No.	V_{DD}	Program Memory	Data Memory	Data EEPROM	I/O	Ext. Int.	A/D
BS86B12A-3	2.7V~5.5V	2K×16	384×8	64×8	22	1	12-bit×8
BS86C16A-3	2.7V~5.5V	4K×16	512×8	64×8	26	1	12-bit×8
BS86D20A-3	2.7V~5.5V	8K×16	768×8	64×8	26	1	12-bit×8

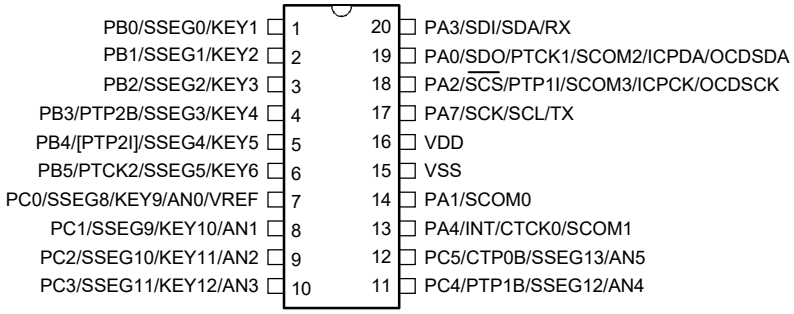
Part No.	LCD Driver	Timer Module	Touch Key	Interface (SPI/I²C)	UART	Time Base	Stack	Package
BS86B12A-3	16×4	10-bit CTM×1 10-bit PTM×2	12	√	√	2	6	20/24SOP
BS86C16A-3	20×4	10-bit CTM×1 10-bit PTM×2	16	√	√	2	6	24/28SOP
BS86D20A-3	20×4	10-bit CTM×1 10-bit PTM×2	20	√	√	2	8	24/28SOP

Block Diagram

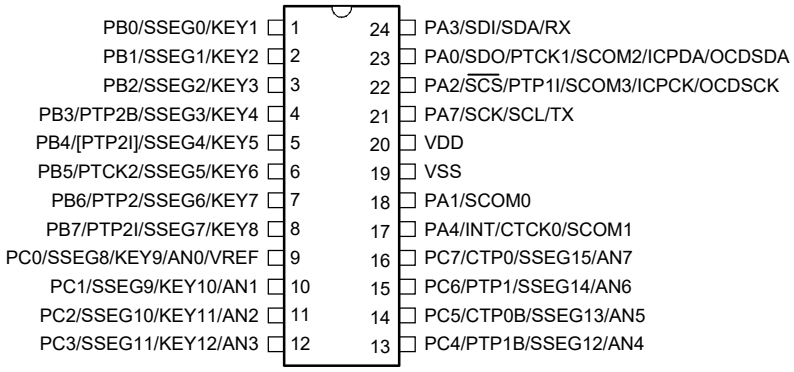


Note: The LXT oscillator is only for the BS86C16A-3 and BS86D20A-3.

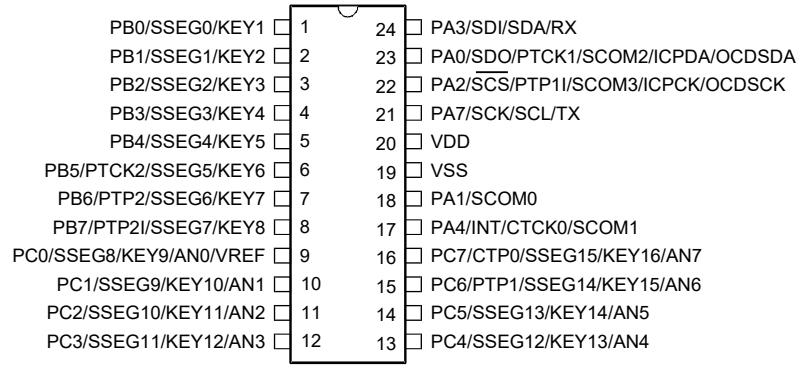
Pin Assignment



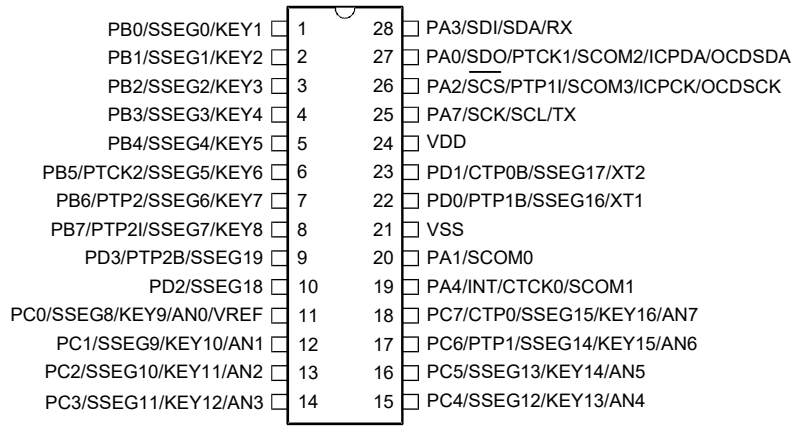
BS86B12A-3/BS86BV12A
20 SOP-A



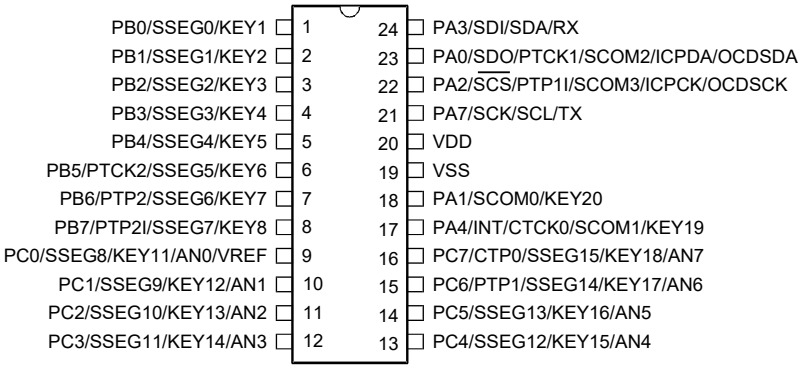
BS86B12A-3/BS86BV12A
24 SOP-A



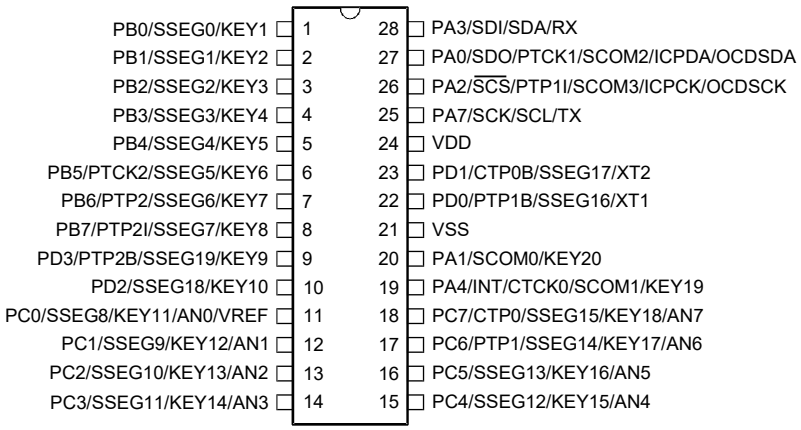
BS86C16A-3/BS86CV16A-3
24 SOP-A



BS86C16A-3/BS86CV16A-3
28 SOP-A



BS86D20A-3/BS86DV20A-3
24 SOP-A



BS86D20A-3/BS86DV20A-3
28 SOP-A

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

2. The OCDSDA and OCDSCK pins are the OCDS dedicated pins and only available for the BS86BV12A/ BS86CV16A-3/BS86DV20A-3 devices, which are the OCDS EV chips for the BS86B12A-3/BS86C16A-3/ BS86D20A-3 devices respectively.

Pin Descriptions

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

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

BS86B12A-3

Pin Name	Function	OP	I/T	O/T	Description
PA0/SDO/PTCK1/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
	PTCK1	PTM1C0	ST	—	PTM1 clock input
	SCOM2	SLCDC0	—	SCOM	LCD driver output for LCD panel common
	ICPDA	—	ST	CMOS	In-circuit programming address/data pin
	OCDSDA	—	ST	CMOS	On-chip debug support data/address pin, for EV chip only.
PA1/SCOM0	PA1	PAWU PAPU	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	SCOM0	SLCDC0	—	SCOM	LCD driver output for LCD panel common
PA2/$\overline{SCS}$/PTP1I/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
	PTP1I	PTM1C0 PTM1C1	ST	—	PTM1 input
	SCOM3	SLCDC0	—	SCOM	LCD driver output for LCD panel common
	ICPCK	—	ST	—	In-circuit programming clock pin
	OCDSCK	—	ST	—	On-chip debug support 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
	RX	UCR1	ST	—	UART receiver data input
PA4/INT/CTCK0/SCOM1	PA4	PAWU PAPU	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	INT	INTC0 INTEG	ST	—	External interrupt
	CTCK0	CTM0C0	ST	—	CTM0 clock input
	SCOM1	SLCDC0	ST	SCOM	LCD driver output for LCD panel common
PA7/SCK/SCL/TX	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
	TX	UCR1	—	CMOS	UART transmitter data output
PB0/SSEG0/KEY1	PB0	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SSEG0	SLCDC1	—	CMOS	LCD driver output for LCD panel segment
	KEY1	TKM0C1	NSI	—	Touch key input

Pin Name	Function	OP	I/T	O/T	Description
PB1/SSEG1/ KEY2	PB1	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SSEG1	SLCDC1	—	CMOS	LCD driver output for LCD panel segment
	KEY2	TKM0C1	NSI	—	Touch key input
PB2/SSEG2/ KEY3	PB2	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SSEG2	SLCDC1	—	CMOS	LCD driver output for LCD panel segment
	KEY3	TKM0C1	NSI	—	Touch key input
PB3/PTP2B/ SSEG3/KEY4	PB3	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	PTP2B	TMPC	—	CMOS	PTM2 output
	SSEG3	SLCDC1	—	—	LCD driver output for LCD panel segment
	KEY4	TKM0C1	NSI	—	Touch key input
PB4/[PTP2I]/ SSEG4/KEY5	PB4	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	PTP2I	PTM2C0 PTM2C1 IFS	ST	—	PTM2 input
	SSEG4	SLCDC1	—	CMOS	LCD driver output for LCD panel segment
	KEY5	TKM1C1	NSI	—	Touch key input
PB5/PTCK2/ SSEG5/KEY6	PB5	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	PTCK2	PTM2C0	ST	—	PTM2 clock input
	SSEG5	SLCDC1	—	CMOS	LCD driver output for LCD panel segment
	KEY6	TKM1C1	NSI	—	Touch key input
PB6/PTP2/ SSEG6/KEY7	PB6	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	PTP2	TMPC	—	CMOS	PTM2 output
	SSEG6	SLCDC1	—	CMOS	LCD driver output for LCD panel segment
	KEY7	TKM1C1	NSI	—	Touch key input
PB7/PTP2I/ SSEG7/KEY8	PB7	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	PTP2I	PTM2C0 PTM2C1 IFS	ST	—	PTM2 input
	SSEG7	SLCDC1	—	CMOS	LCD driver output for LCD panel segment
	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	LCD driver output for LCD panel segment
	KEY9	TKM2C1	NSI	—	Touch key input
	AN0	ACERL	AN	—	A/D Converter intput
	VREF	ADCR1	AN	—	A/D Converter reference input
PC1/SSEG9/ KEY10/AN1	PC1	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SSEG9	SLCDC2	—	CMOS	LCD driver output for LCD panel segment
	KEY10	TKM2C1	NSI	—	Touch key input
	AN1	ACERL	AN	—	A/D Converter intput
PC2/SSEG10/ KEY11/AN2	PC2	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SSEG10	SLCDC2	—	CMOS	LCD driver output for LCD panel segment
	KEY11	TKM2C1	NSI	—	Touch key input
	AN2	ACERL	AN	—	A/D Converter intput
PC3/SSEG11/ KEY12/AN3	PC3	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SSEG11	SLCDC2	—	CMOS	LCD driver output for LCD panel segment
	KEY12	TKM2C1	NSI	—	Touch key input
	AN3	ACERL	AN	—	A/D Converter intput

Pin Name	Function	OP	I/T	O/T	Description
PC4/PTP1B/ SSEG12/AN4	PC4	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	PTP1B	TMPC	—	CMOS	PTM1 output
	SSEG12	SLCDC2			LCD driver output for LCD panel segment
	AN4	ACERL	AN	—	A/D Converter intput
PC5/CTP0B/ SSEG13/AN5	PC5	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	CTP0B	TMPC	—	CMOS	CTM0 output
	SSEG13	SLCDC2	—	CMOS	LCD driver output for LCD panel segment
	AN5	ACERL	AN	—	A/D Converter intput
PC6/PTP1/ SSEG14/AN6	PC6	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	PTP1	TMPC	—	CMOS	PTM1 output
	SSEG14	SLCDC2	—	CMOS	LCD driver output for LCD panel segment
	AN6	ACERL	AN	—	A/D Converter intput
PC7/CTP0/ SSEG15/AN7	PC7	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	CTP0	TMPC	—	CMOS	CTM0 output
	SSEG15	SLCDC2	—	CMOS	LCD driver output for LCD panel segment
	AN7	ACERL	AN	—	A/D Converter intput
VDD	VDD	—	PWR	—	Power supply
VSS	VSS	—	PWR	—	Ground

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

BS86C16A-3

Pin Name	Function	OP	I/T	O/T	Description
PA0/SDO/ PTCK1/ 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
	PTCK1	PTM1C0	ST	—	PTM1 clock input
	SCOM2	SLCDC0	—	SCOM	LCD driver output for LCD panel common
	ICPDA	—	ST	CMOS	In-circuit programming address/data pin
	OCDSDA	—	ST	CMOS	On-chip debug support data/address pin, for EV chip only.
PA1/SCOM0	PA1	PAWU PAPU	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	SCOM0	SLCDC0	—	SCOM	LCD driver output for LCD panel common
PA2/$\overline{\text{SCS}}$/PTP1I/ SCOM3/ICPCK/ OCDSCK	PA2	PAWU PAPU	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	$\overline{\text{SCS}}$	SIMC0	ST	CMOS	SPI slave select
	PTP1I	PTM1C0 PTM1C1	ST	—	PTM1 input
	SCOM3	SLCDC0	—	SCOM	LCD driver output for LCD panel common
	ICPCK	—	ST	—	In-circuit programming clock pin
	OCDSCK	—	ST	—	On-chip debug support clock pin, for EV chip only.

Pin Name	Function	OP	I/T	O/T	Description
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
	RX	UCR1	ST	—	UART receiver data input
PA4/INT/CTCK0/ SCOM1	PA4	PAWU PAPU	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	INT	INTC0 INTEG	ST	—	External interrupt
	CTCK0	CTM0C0	ST	—	CTM0 clock input
	SCOM1	SLCDC0	—	SCOM	LCD driver output for LCD panel common
PA7/SCK/SCL/TX	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
	TX	UCR1	—	CMOS	UART transmitter data output
PB0/SSEG0/ KEY1	PB0	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SSEG0	SLCDC1	—	CMOS	LCD driver output for LCD panel segment
	KEY1	TKM0C1	NSI	—	Touch key input
PB1/SSEG1/ KEY2	PB1	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SSEG1	SLCDC1	—	CMOS	LCD driver output for LCD panel segment
	KEY2	TKM0C1	NSI	—	Touch key input
PB2/SSEG2/ KEY3	PB2	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SSEG2	SLCDC1	—	CMOS	LCD driver output for LCD panel segment
	KEY3	TKM0C1	NSI	—	Touch key input
PB3/SSEG3/ KEY4	PB3	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SSEG3	SLCDC1	—	CMOS	LCD driver output for LCD panel segment
	KEY4	TKM0C1	NSI	—	Touch key input
PB4/SSEG4/ KEY5	PB4	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SSEG4	SLCDC1	—	CMOS	LCD driver output for LCD panel segment
	KEY5	TKM1C1	NSI	—	Touch key input
PB5/PTCK2/ SSEG5/KEY6	PB5	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	PTCK2	PTM2C0	ST	—	PTM2 clock input
	SSEG5	SLCDC1	—	CMOS	LCD driver output for LCD panel segment
	KEY6	TKM1C1	NSI	—	Touch key input
PB6/PTP2/ SSEG6/KEY7	PB6	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	PTP2	TMPC	—	CMOS	PTM2 output
	SSEG6	SLCDC1	—	CMOS	LCD driver output for LCD panel segment
	KEY7	TKM1C1	NSI	—	Touch key input
PB7/PTP2I/ SSEG7/KEY8	PB7	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	PTP2I	PTM2C0 PTM2C1	ST	—	PTM2 input
	SSEG7	SLCDC1	—	CMOS	LCD driver output for LCD panel segment
	KEY8	TKM1C1	NSI	—	Touch key input

Pin Name	Function	OP	I/T	O/T	Description
PC0/SSEG8/ KEY9/AN0/VREF	PC0	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SSEG8	SLCDC2	—	CMOS	LCD driver output for LCD panel segment
	KEY9	TKM2C1	NSI	—	Touch key input
	AN0	ACERL	AN	—	A/D Converter intput
	VREF	ADCR1	AN	—	A/D Converter reference input
PC1/SSEG9/ KEY10/AN1	PC1	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SSEG9	SLCDC2	—	CMOS	LCD driver output for LCD panel segment
	KEY10	TKM2C1	NSI	—	Touch key input
	AN1	ACERL	AN	—	A/D Converter intput
PC2/SSEG10/ KEY11/AN2	PC2	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SSEG10	SLCDC2	—	CMOS	LCD driver output for LCD panel segment
	KEY11	TKM2C1	NSI	—	Touch key input
	AN2	ACERL	AN	—	A/D Converter intput
PC3/SSEG11/ KEY12/AN3	PC3	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SSEG11	SLCDC2	—	CMOS	LCD driver output for LCD panel segment
	KEY12	TKM2C1	NSI	—	Touch key input
	AN3	ACERL	AN	—	A/D Converter intput
PC4/SSEG12/ KEY13/AN4	PC4	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SSEG12	SLCDC2	—	CMOS	LCD driver output for LCD panel segment
	KEY13	TKM3C1	NSI	—	Touch key input
	AN4	ACERL	AN	—	A/D Converter intput
PC5/SSEG13/ KEY14/AN5	PC5	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SSEG13	SLCDC2	—	CMOS	LCD driver output for LCD panel segment
	KEY14	TKM3C1	NSI	—	Touch key input
	AN5	ACERL	AN	—	A/D Converter intput
PC6/PTP1/ SSEG14/ KEY15/AN6	PC6	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	PTP1	TMPC	—	CMOS	PTM1 output
	SSEG14	SLCDC2	—	CMOS	LCD driver output for LCD panel segment
	KEY15	TKM3C1	NSI	—	Touch key input
	AN6	ACERL	AN	—	A/D Converter intput
PC7/CTP0/ SSEG15/ KEY16/AN7	PC7	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	CTP0	TMPC	—	CMOS	CTM0 output
	SSEG15	SLCDC2	—	CMOS	LCD driver output for LCD panel segment
	KEY16	TKM3C1	NSI	—	Touch key input
	AN7	ACERL	AN	—	A/D Converter intput
PD0/PTP1B/ SSEG16/XT1	PD0	PDPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	PTP1B	TMPC	—	CMOS	PTM1 output
	SSEG16	SLCDC3	—	CMOS	LCD driver output for LCD panel segment
	XT1	CO	LXT	—	LXT pin

Pin Name	Function	OP	I/T	O/T	Description
PD1/CTP0B/ SSEG17/XT2	PD1	PDPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	CTP0B	TMPC	—	CMOS	CTM0 output
	SSEG17	SLCDC3	—	CMOS	LCD driver output for LCD panel segment
	XT2	CO	—	LXT	LXT pin
PD2/SSEG18	PD2	PDPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SSEG18	SLCDC3	—	CMOS	LCD driver output for LCD panel segment
PD3/PTP2B/ SSEG19	PD3	PDPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	PTP2B	TMPC	—	CMOS	PTM2 output
	SSEG19	SLCDC3	—	CMOS	LCD driver output for LCD panel segment
VDD	VDD	—	PWR	—	Power supply
VSS	VSS	—	PWR	—	Ground

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

BS86D20A-3

Pin Name	Function	OP	I/T	O/T	Description
PA0/SDO/PTCK1/ 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
	PTCK1	PTM1C0	ST	—	PTM1 clock input
	SCOM2	SLCDC0	—	SCOM	LCD driver output for LCD panel common
	ICPDA	—	ST	CMOS	In-circuit programming address/data pin
	OCDSDA	—	ST	CMOS	On-chip debug support data/address pin, for EV chip only.
PA1/SCOM0/ KEY20	PA1	PAWU PAPU	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	SCOM0	SLCDC0	—	SCOM	LCD driver output for LCD panel common
	KEY20	TKM4C1	NSI	—	Touch key input
PA2/$\overline{\text{SCS}}$/PTP1I/ SCOM3/ICPCK/ OCDSCK	PA2	PAWU PAPU	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	$\overline{\text{SCS}}$	SIMC0	ST	CMOS	SPI slave select
	PTP1I	PTM1C0 PTM1C1	ST	—	PTM1 input
	SCOM3	SLCDC0	—	SCOM	LCD driver output for LCD panel common
	ICPCK	—	ST	—	In-circuit programming clock pin
	OCDSCK	—	ST	—	On-chip debug support 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
	RX	UCR1	ST	—	UART receiver data input

Pin Name	Function	OP	I/T	O/T	Description
PA4/INT/CTCK0/ SCOM1/KEY19	PA4	PAWU PAPU	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	INT	INTC0 INTEG	ST	—	External interrupt
	CTCK0	CTM0C0	ST	—	CTM0 clock input
	SCOM1	SLCDC0	—	SCOM	LCD driver output for LCD panel common
	KEY19	TKM4C1	NSI	—	Touch key input
PA7/SCK/SCL/TX	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
	TX	UCR1	—	CMOS	UART transmitter data output
PB0/SSEG0/ KEY1	PB0	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SSEG0	SLCDC1	—	CMOS	LCD driver output for LCD panel segment
	KEY1	TKM0C1	NSI	—	Touch key input
PB1/SSEG1/ KEY2	PB1	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SSEG1	SLCDC1	—	CMOS	LCD driver output for LCD panel segment
	KEY2	TKM0C1	NSI	—	Touch key input
PB2/SSEG2/ KEY3	PB2	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SSEG2	SLCDC1	—	CMOS	LCD driver output for LCD panel segment
	KEY3	TKM0C1	NSI	—	Touch key input
PB3/SSEG3/ KEY4	PB3	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SSEG3	SLCDC1	—	CMOS	LCD driver output for LCD panel segment
	KEY4	TKM0C1	NSI	—	Touch key input
PB4/SSEG4/ KEY5	PB4	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SSEG4	SLCDC1	—	CMOS	LCD driver output for LCD panel segment
	KEY5	TKM1C1	NSI	—	Touch key input
PB5/PTCK2/ SSEG5/KEY6	PB5	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	PTCK2	PTM2C0	ST	—	PTM2 clock input
	SSEG5	SLCDC1	—	CMOS	LCD driver output for LCD panel segment
	KEY6	TKM1C1	NSI	—	Touch key input
PB6/PTP2/ SSEG6/KEY7	PB6	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	PTP2	TMPC	—	CMOS	PTM2 output
	SSEG6	SLCDC1	—	CMOS	LCD driver output for LCD panel segment
	KEY7	TKM1C1	NSI	—	Touch key input
PB7/PTP2I/ SSEG7/KEY8	PB7	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	PTP2I	PTM2C0 PTM2C1	ST	—	PTM2 input
	SSEG7	SLCDC1	—	CMOS	LCD driver output for LCD panel segment
	KEY8	TKM1C1	NSI	—	Touch key input

Pin Name	Function	OP	I/T	O/T	Description
PC0/SSEG8/ KEY11/AN0/ VREF	PC0	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SSEG8	SLCDC2	—	CMOS	LCD driver output for LCD panel segment
	KEY11	TKM2C1	NSI	—	Touch key input
	AN0	ACERL	AN	—	A/D Converter intput
	VREF	ADCR1	AN	—	A/D Converter reference input
PC1/SSEG9/ KEY12/AN1	PC1	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SSEG9	SLCDC2	—	CMOS	LCD driver output for LCD panel segment
	KEY12	TKM2C1	NSI	—	Touch key input
	AN1	ACERL	AN	—	A/D Converter intput
PC2/SSEG10/ KEY13/AN2	PC2	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SSEG10	SLCDC2	—	CMOS	LCD driver output for LCD panel segment
	KEY13	TKM2C1	NSI	—	Touch key input
	AN2	ACERL	AN	—	A/D Converter intput
PC3/SSEG11/ KEY14/AN3	PC3	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SSEG11	SLCDC2	—	CMOS	LCD driver output for LCD panel segment
	KEY14	TKM3C1	NSI	—	Touch key input
	AN3	ACERL	AN	—	A/D Converter intput
PC4/SSEG12/ KEY15/AN4	PC4	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SSEG12	SLCDC2	—	CMOS	LCD driver output for LCD panel segment
	KEY15	TKM3C1	NSI	—	Touch key input
	AN4	ACERL	AN	—	A/D Converter intput
PC5/SSEG13/ KEY16/AN5	PC5	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SSEG13	SLCDC2	—	CMOS	LCD driver output for LCD panel segment
	KEY16	TKM3C1	NSI	—	Touch key input
	AN5	ACERL	AN	—	A/D Converter intput
PC6/PTP1/ SSEG14/ KEY17/AN6	PC6	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	PTP1	TMPC	—	CMOS	PTM1 output
	SSEG14	SLCDC2	—	CMOS	LCD driver output for LCD panel segment
	KEY17	TKM4C1	NSI	—	Touch key input
	AN6	ACERL	AN	—	A/D Converter intput
PC7/CTP0/ SSEG15/ KEY18/AN7	PC7	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	CTP0	TMPC	—	CMOS	CTM0 output
	SSEG15	SLCDC2	—	CMOS	LCD driver output for LCD panel segment
	KEY18	TKM4C1	NSI	—	Touch key input
	AN7	ACERL	AN	—	A/D Converter intput
PD0/PTP1B/ SSEG16/XT1	PD0	PDPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	PTP1B	TMPC	—	CMOS	PTM1 output
	SSEG16	SLCDC3	—	CMOS	LCD driver output for LCD panel segment
	XT1	CO	LXT	—	LXT pin

Pin Name	Function	OP	I/T	O/T	Description
PD1/CTP0B/ SSEG17/XT2	PD1	PDPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	CTP0B	TMPC	—	CMOS	CTM0 output
	SSEG17	SLCDC3	—	CMOS	LCD driver output for LCD panel segment
	XT2	CO	—	LXT	LXT pin
PD2/SSEG18/ KEY10	PD2	PDPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	SSEG18	SLCDC3	—	CMOS	LCD driver output for LCD panel segment
	KEY10	TKM2C1	NSI	—	Touch key input
PD3/PTP2B/ SSEG19/KEY9	PD3	PDPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	PTP2B	TMPC	—	CMOS	PTM2 output
	SSEG19	SLCDC3	—	CMOS	LCD driver output for LCD panel segment
	KEY9	TKM2C1	NSI	—	Touch key input
VDD	VDD	—	PWR	—	Power supply
VSS	VSS	—	PWR	—	Ground

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

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	Operating Voltage (HIRC)	—	f_{SYS}=8MHz	2.7	—	5.5	V
			f_{SYS}=12MHz	2.7	—	5.5	V
			f_{SYS}=16MHz	4.5	—	5.5	V
I_{DD}	Operating Current (Normal) (HIRC, $f_{SYS}=f_H$, $f_S=f_{SUB}$)	3V	No load, f_H=8MHz, ADC off, WDT enable, LVR enable	—	1.2	1.8	mA
		5V		—	2.2	3.3	mA
		3V	No load, f_H=12MHz, ADC off, WDT enable, LVR enable	—	1.6	2.4	mA
		5V		—	3.3	5.0	mA
		5V	No load, f_H=16MHz, ADC off, WDT enable, LVR enable	—	4.0	6.0	mA
	Operating Current (Normal) (HIRC, $f_{SYS}=f_L$, $f_S=f_{SUB}$)	3V	No load, f_H =12MHz, $f_L=f_H/2$, ADC off, WDT enable, LVR enable	—	1.2	2.0	mA
		5V		—	2.2	3.3	mA
		5V	No load, f_H =12MHz, $f_L=f_H/64$, ADC off, WDT enable, LVR enable	—	0.8	1.2	mA
		3V		—	1.5	2.3	mA
	Operating Current (Slow) (LXT/LIRC, $f_{SYS}=f_L$, $f_S=f_{SUB}$) (BS86C16A-3/BS86D20A-3 only)	3V	No load, f_{SYS}=LXT, ADC off, WDT enable, LVR enable, LXTLP=0	—	19	38	μA
		5V		—	48	96	μA
		3V	No load, f_{SYS}=LXT, ADC off, WDT enable, LVR enable, LXTLP=1	—	16	32	μA
		5V		—	36	72	μA
		3V	No load, f_{SYS}=LIRC, ADC off, WDT enable, LVR enable	—	16	32	μA
		5V		—	36	72	μA
	Operating Current (Slow) (LIRC, $f_{SYS}=f_L$, $f_S=f_{SUB}$) (BS86B12A-3 only)	3V	No load, f_{SYS}=LIRC, ADC off, WDT enable, LVR enable	—	16	32	μA
		5V		—	36	72	μA
I_{STB}	IDLE1 Mode Standby Current (HIRC, $f_{SYS}=f_H$, $f_S=f_{SUB}$)	3V	No load, system HALT, ADC off, WDT enable, f_{SYS}=12MHz	—	0.9	1.4	mA
		5V		—	1.4	2.1	mA
	IDLE0 Mode Standby Current (HIRC, f_{SYS}=off, $f_S=f_{SUB}$)	3V	No load, system HALT, ADC off, WDT enable, f_{SYS}=12MHz	—	1.4	3.0	μA
		5V		—	2.7	5.0	μA
	IDLE1 Mode Standby Current (HIRC, $f_{SYS}=f_L$, $f_S=f_{SUB}$)	3V	No load, system HALT, ADC off, WDT enable, f_{SYS}=12MHz/64	—	0.7	1.1	mA
		5V		—	1.4	2.1	mA
	IDLE0 Mode Standby Current (HIRC, f_{SYS}=off, $f_S=f_{SUB}$)	3V	No load, system HALT, ADC off, WDT enable, f_{SYS}=12MHz/64	—	1.3	3.0	μA
		5V		—	2.3	5.0	μA
	IDLE1 Mode Standby Current (LIRC, $f_{SYS}=f_L=f_{LIRC}$, $f_S=f_{SUB}=f_{LIRC}$)	3V	No load, system HALT, ADC off, WDT enable, f_{SYS}=LIRC	—	1.9	4.0	μA
		5V		—	3.3	7.0	μA

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{STB}	IDLE0 Mode Standby Current (LXT/LIRC, f_{SYS}=off, f_S=f_{SUB}) (BS86C16A-3/BS86D20A-3 only)	3V	No load, system HALT, ADC off, WDT enable, LXTLP=0 (LXT on)	—	5	10	μA
		5V		—	18	30	μA
		3V	No load, system HALT, ADC off, WDT enable, LXTLP=1 (LXT on)	—	2.5	5	μA
		5V		—	6	10	μA
		3V	No load, system HALT, ADC off, WDT enable (LIRC on)	—	1.3	3.0	μA
		5V		—	2.4	5.0	μA
	IDLE0 Mode Standby Current (LIRC, f_{SYS}=off, f_S=f_{SUB}) (BS86B12A-3 only)	3V	No load, system HALT, ADC off, WDT enable (LIRC on)	—	1.3	3.0	μA
		5V		—	2.4	5.0	μA
	SLEEP Mode Standby Current (HIRC, f_{SYS}=off, f_S=f_{SUB}=off)	3V	No load, system HALT, ADC off, WDT disable (LXT and LIRC off)	—	0.1	1	μA
		5V		—	0.3	2	μA
	SLEEP Mode Standby Current (LXT/LIRC, f_{SYS}=off, f_S=f_{SUB}=off) (BS86C16A-3/BS86D20A-3 only)	3V	No load, system HALT, ADC off, WDT disable (LXT and LIRC off)	—	0.1	1	μA
		5V		—	0.3	2	μA
V_{IL}	Input Low Voltage for I/O Ports or Input Pins	5V	—	0	—	1.5	V
		—		0	—	$0.2V_{DD}$	V
V_{IH}	Input High Voltage for I/O Ports or Input Pins	5V	—	3.5	—	5.0	V
		—		$0.8V_{DD}$	—	V_{DD}	V
V_{LVR}	Low Voltage Reset Voltage	—	LVR enable, 2.55V	-5%	2.55	+5%	V
V_{LVD}	Low Voltage Detector Voltage	—	LVDEN=1, V_{LVD}=2.7V	-5%	2.7	+5%	V
			LVDEN=1, V_{LVD}=3.0V	-5%	3.0	+5%	V
			LVDEN=1, V_{LVD}=3.3V	-5%	3.3	+5%	V
			LVDEN=1, V_{LVD}=3.6V	-5%	3.6	+5%	V
			LVDEN=1, V_{LVD}=4.0V	-5%	4.0	+5%	V
I_{OL}	I/O Port Sink Current	3V	V_{OL}=0.1V_{DD}	16	32	—	mA
		5V	V_{OL}=0.1V_{DD}	32	64	—	mA
I_{OH}	I/O Port Source Current	3V	V_{OH}=0.9V_{DD}, PxPS=00	-1.0	-2.0	—	mA
		5V	V_{OH}=0.9V_{DD}, PxPS=00	-2.0	-4.0	—	mA
		3V	V_{OH}=0.9V_{DD}, PxPS=01	-1.75	-3.5	—	mA
		5V	V_{OH}=0.9V_{DD}, PxPS=01	-3.5	-7.0	—	mA
		3V	V_{OH}=0.9V_{DD}, PxPS=10	-2.5	-5.0	—	mA
		5V	V_{OH}=0.9V_{DD}, PxPS=10	-5.0	-10	—	mA
		3V	V_{OH}=0.9V_{DD}, PxPS=11	-5.5	-11	—	mA
		5V	V_{OH}=0.9V_{DD}, PxPS=11	-11	-22	—	mA
R_{PH}	Pull-high Resistance for I/O Ports	3V	—	20	60	100	kΩ
		5V	—	10	30	50	kΩ

A.C. Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
f_{SYS}	System Clock (HIRC)	3V/5V	Ta=25°C	-2%	8	+2%	MHz
				-2%	12	+2%	MHz
		5V		-2%	16	+2%	MHz
t_{TIMER}	Timer Input Pulse Width	—	—	0.3	—	—	μs
f_{LIRC}	System Clock (32kHz)	5V	Ta=25°C	-10%	32	+10%	kHz
f_{LXT}	System Clock (LXT)	—	—	—	32768	—	Hz
t_{INT}	Interrupt Pulse Width	—	—	10	—	—	μs
t_{LVR}	Low Voltage Width to Reset	—	—	120	240	480	μs
t_{LVD}	Low Voltage Width to Interrupt	—	—	60	120	240	μs
t_{LVDS}	LVDO stable time	—	—	—	—	15	μs
t_{EERD}	EEPROM Read Time	—	—	1	2	4	t_{SYS}
t_{EEWR}	EEPROM Write Time	—	—	1	2	4	ms
t_{RSTD}	System Reset Delay Time (Power on reset, LVR reset, WDT S/W reset (WDTC))	—	—	25	50	100	ms
	System Reset Delay Time (WDT normal reset)	—	—	8.3	16.7	33.3	ms
t_{SST}	System Start-up Timer Period (Wake-up from HALT)	—	f_{SYS}=LXT	1024	—	—	t_{SYS}
			f_{SYS}=HIRC	16	—	—	
			f_{SYS}=LIRC	2	—	—	
	System Start-up Timer Period (Wake-up from HALT, f_{SYS} on at HALT state)	—	—	2	—	—	

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

Sensor Oscillator Electrical Characteristics

Ta=25°C

Touch Key RC OSC=500kHz

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Condition				
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 Capacitance	5V	*f_{SENOSC}=500kHz	5	10	20	pF
C_{REFOSC}	Reference Oscillator Internal Capacitance	5V	*f_{SENOSC}=500kHz	5	10	20	pF
f_{KEYOSC}	Sensor (KEY) Oscillator Operating Frequency	5V	*External Capacitance =7,8,9,10,11,12,13,14,15, … 50pF	100	500	1000	kHz
$f_{REFYOSC}$	Reference Oscillator Operating Frequency	5V	*Internal Capacitance =7,8,9,10,11,12,13,14,15, … 50pF	100	500	1000	kHz

Note: *f_{SENOSC}=500kHz: adjust the KEYn capacitor to make the Sensor Oscillator frequency =500kHz.

*f_{REFOSC}=500kHz: adjust the Reference Oscillator internal capacitor to make the Reference Oscillator frequency =500kHz.

Touch Key RC OSC=1000kHz

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Condition				
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 Capacitance	5V	*f_{SENOSC}=1000kHz	5	10	20	pF
C_{REFOSC}	Reference Oscillator Internal Capacitance	5V	*f_{SENOSC}=1000kHz	5	10	20	pF
f_{KEYOSC}	Sensor (KEY) Oscillator Operating Frequency	5V	*External Capacitance =1,2,3,4,5,6,7,8,9,10,11,12,13, 14,15, … 50pF	150	1000	2500	kHz
$f_{REFYOSC}$	Reference Oscillator Operating Frequency	5V	*Internal Capacitance =1,2,3,4,5,6,7,8,9,10,11,12,13, 14,15, … 50pF	150	1000	2500	kHz

Note: *f_{SENOSC}=1000kHz: adjust the KEYn capacitor to make the Sensor Oscillator frequency =1000kHz.
*f_{REFOSC}=1000kHz: adjust the Reference Oscillator internal capacitor to make the Reference Oscillator frequency =1000kHz.

Touch Key RC OSC=1500kHz

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Condition				
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 Capacitance	3V	*f_{SENOSC}=1500kHz	4	8	16	pF
		5V		5	10	20	
C_{REFOSC}	Reference Oscillator Internal Capacitance	3V	*f_{SENOSC}=1500kHz	4	8	16	pF
		5V		5	10	20	
f_{KEYOSC}	Sensor (KEY) Oscillator Operating Frequency	3V	*External Capacitance =1,2,3,4,5,6,7,8,9,10,11,12,13, 14,15, … 50pF	150	1500	3000	kHz
		5V		150	1500	3000	
$f_{REFYOSC}$	Reference Oscillator Operating Frequency	3V	*Internal Capacitance =1,2,3,4,5,6,7,8,9,10,11,12,13, 14,15, … 50pF	150	1500	3000	kHz
		5V		150	1500	3000	

Note: *f_{SENOSC}=1500kHz: adjust the KEYn capacitor to make the Sensor Oscillator frequency =1500kHz.
*f_{REFOSC}=1500kHz: adjust the Reference Oscillator internal capacitor to make the Reference Oscillator frequency =1500kHz.

Touch Key RC OSC=2000kHz

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Condition				
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 Capacitance	3V	*f_{SENOSC}=2000kHz	4	8	16	pF
		5V		5	10	20	
C_{REFOSC}	Reference Oscillator Internal Capacitance	3V	*f_{SENOSC}=2000kHz	4	8	16	pF
		5V		5	10	20	
f_{KEYOSC}	Sensor (KEY) Oscillator Operating Frequency	3V	*External Capacitance =1,2,3,4,5,6,7,8,9,10,11,12,13, 14,15, … 50pF	150	2000	4000	kHz
		5V		150	2000	4000	
$f_{REFYOSC}$	Reference Oscillator Operating Frequency	3V	*Internal Capacitance =1,2,3,4,5,6,7,8,9,10,11,12,13, 14,15, … 50pF	150	2000	4000	kHz
		5V		150	2000	4000	

Note: *f_{SENOSC}=2000kHz: adjust the KEYn capacitor to make the Sensor Oscillator frequency =2000kHz.
*f_{REFOSC}=2000kHz: adjust the Reference Oscillator internal capacitor to make the Reference Oscillator frequency =2000kHz

A/D Converter Electrical Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
AV_{DD}	A/D Converter Operating Voltage	—	—	2.7	—	5.5	V
V_{ADI}	A/D Converter Input Voltage	—	—	0	—	V_{REF}	V
V_{REF}	A/D Converter Reference Voltage	—	—	2	—	AV_{DD}	V
V_{BG}	Bandgap Reference with Buffer Voltage	—	—	-3%	1.09	+3%	V
DNL	Differential Non-linearity	3V	V_{REF}=AV_{DD}=V_{DD}, t_{ADCK}=0.5μs, Ta=25°C	-3	—	+3	LSB
		5V					
		3V	V_{REF}=AV_{DD}=V_{DD}, t_{ADCK}=0.5μs, Ta=-40°C~85°C	-6	—	+6	LSB
		5V					
INL	Integral Non-linearity	3V	V_{REF}=AV_{DD}=V_{DD}, t_{ADCK}=0.5μs, Ta=25°C	-4	—	+4	LSB
		5V					
		3V	V_{REF}=AV_{DD}=V_{DD}, t_{ADCK}=0.5μs, Ta=-40°C~85°C	-8	—	+8	LSB
		5V					
I_{ADC}	Additional Power Consumption if A/D Converter is used	3V	No load (t_{ADCK}=0.5μs)	—	0.9	1.35	mA
		5V	No load (t_{ADCK}=0.5μs)	—	1.2	1.8	mA
I_{BG}	Additional Power Consumption if V_{BG} Reference with Buffer is used	—	—	—	200	300	μA
t_{ADCK}	A/D Converter Clock Period	—	—	0.5	—	10	μs
t_{ADC}	A/D Conversion Time (Include Sample and Hold Time)	—	12-Bit ADC	—	16	—	t_{ADCK}
t_{ADS}	A/D Converter Sampling Time	—	—	—	4	—	t_{ADCK}

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
t_{ON2ST}	A/D Converter On-to-Start Time	—	—	2	—	—	μs
t_{BG}	V_{BG} Turn-on Stable Time	—	—	—	—	200	μs

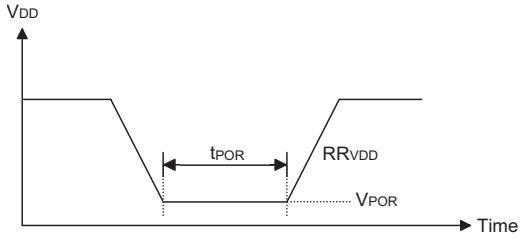
LCD Electrical Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{BIAS}	V_{DD}/3 bias Current for LCD	5V	ISEL[1:0]=00B	5.8	8.3	10.8	μA
			ISEL[1:0]=01B	11.7	16.7	21.7	
			ISEL[1:0]=10B	35	50	65	
			ISEL[1:0]=11B	70	100	130	
V_{SCOM}	1/3 bias LCD COM Output (1/3 V_{DD})	2.2V~5.5V	No load	0.317×V_{DD}	(1/3)×V_{DD}	0.35×V_{DD}	V
	1/3 bias LCD COM Output (2/3 V_{DD})	2.2V~5.5V	No load	0.634×V_{DD}	(2/3)×V_{DD}	0.7×V_{DD}	V
V_{SSEG}	1/3 bias LCD SEG Output (1/3 V_{DD})	2.2V~5.5V	No load	0.317×V_{DD}	(1/3)×V_{DD}	0.35×V_{DD}	V
	1/3 bias LCD SEG Output (2/3 V_{DD})	2.2V~5.5V	No load	0.634×V_{DD}	(2/3)×V_{DD}	0.7×V_{DD}	V

Power-on Reset Characteristics

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_{VDD}	V_{DD} Raising 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

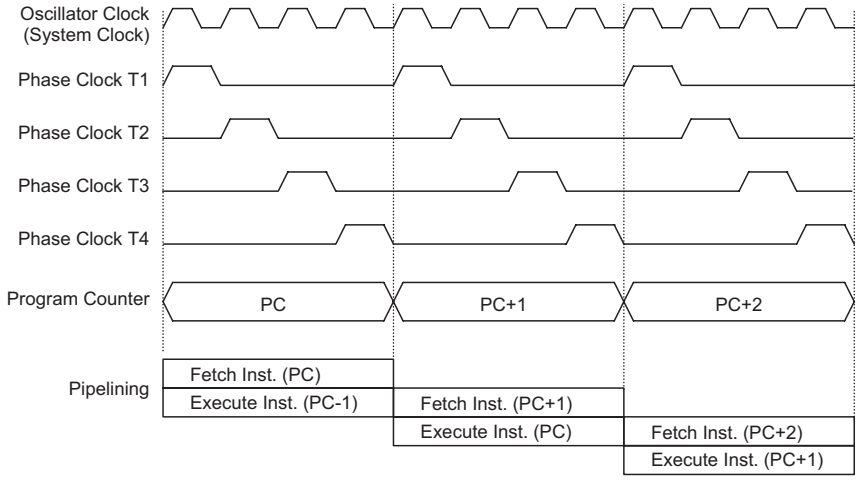


System Architecture

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

Clocking and Pipelining

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

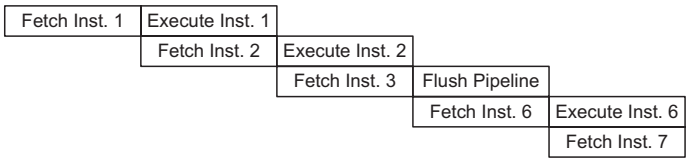


System Clock and Pipelining

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

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

   DELAY:
```



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.

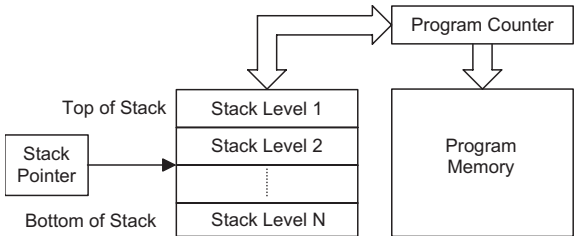
Device	Program Counter	
	Program Counter High Byte	PCL Register
BS86B12A-3	PC10~PC8	PCL7~PCL0
BS86C16A-3	PC11~PC8	
BS86D20A-3	PC12~PC8	

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

Stack

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

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



Device	Stack Levels
BS86B12A-3	6
BS86C16A-3	6
BS86D20A-3	8

Arithmetic and Logic Unit – ALU

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

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

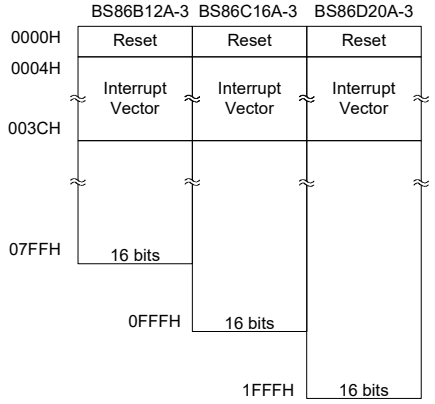
Flash Program Memory

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

Structure

The Program Memory has a capacity of 2K×16 Bits 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 register.

Device	Capacity
BS86B12A-3	2Kx16
BS86C16A-3	4K×16
BS86D20A-3	8K×16



Program Memory Structure

Special Vectors

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

Look-up Table

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

After setting up the table pointer pair, 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 Data Memory Sector 0. If the memory [m] is located in Data Memory other sectors, the data can be retrieved from the program memory using the "LTABRD[m]" or "LTABRDL[m]" instructions respectively. When the instruction is executed, the lower order table byte from the Program Memory will be transferred to the user defined Data Memory register [m] as specified in the instruction. The higher order table data byte from the Program Memory will be transferred to the TBLH special register.

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

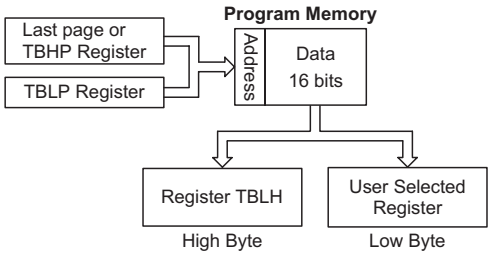


Table Program Example

The following example shows how the table pointer and table data is defined and retrieved from the microcontroller. This example uses raw table data located in the Program Memory which is stored there using the ORG statement. The value at this ORG statement is "0F00H" which refers to the start address of the last page within the 4K words Program Memory of the BS86C16A-3. The table pointer is setup here to have an initial value of "06H". This will ensure that the first data read from the data table will be at the Program Memory address "0F06H" 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 specific page if the "TABRD [m]" instruction is being used. The high byte of the table data which in this case is equal to zero will be transferred to the TBLH register automatically when the "TABRD [m]" instruction is executed.

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

In Circuit Programming – ICP

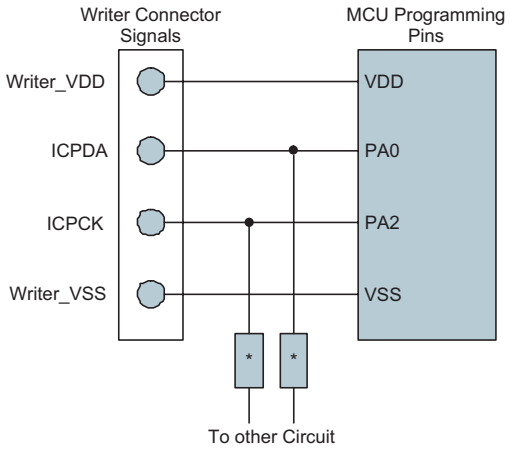
The provision of Flash type Program Memory provides the user with a means of convenient and easy upgrades and modifications to their programs on the same device. As an additional convenience, Holtek has provided a means of programming the microcontroller in-circuit using a 4-pin interface. This provides 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.

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

Holtek Write Pins	MCU Programming Pins	Function
ICPDA	PA0	Serial data/address input/output
ICPCK	PA2	Serial Clock input
VDD	VDD	Power Supply
VSS	VSS	Ground

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

During the programming process the PA0 and PA2 I/O pins for data and clock programming purposes. The user must there take care to ensure that no other outputs are connected to these two pins.



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

On-Chip Debug Support – OCDS

There are three EV chips named BS86BV12A, BS86CV16A-3 and BS86DV20A-3, which are used to emulate the BS86B12A-3, BS86C16A-3 and BS86D20A-3 devices respectively. Each EV chip device also provides an “On-Chip Debug” function to debug the corresponding MCU device during the development process. The EV chip and the actual MCU device are almost functionally compatible except for the “On-Chip Debug” function. Users can use the EV chip device to emulate the real chip device behavior by connecting the OCDSDA and OCDSCK pins to the 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 for debugging, other functions which are shared with the OCDSDA and OCDSCK pins in the actual MCU device will

have no effect in the EV chip. However, the two OCDS pins which are pin-shared with the ICP programming pins are still used as the Flash Memory programming pins for ICP. For a more detailed OCDS description, refer to the corresponding document named "Holtek e-Link for 8-Bit MCU OCDS User's Guide".

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

RAM Data Memory

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

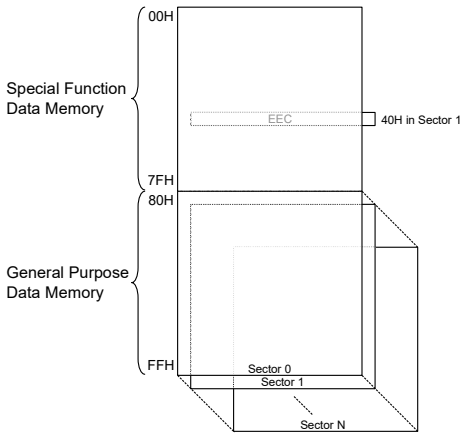
Structure

Divided into two types, the first of these is an area of RAM, known as the Special Function Data Memory. Here are located registers which are necessary for correct operation of the device. Many of these registers can be read from and written to directly under program control, however, some remain protected from user manipulation. The second area of Data Memory is known as the General Purpose Data Memory, which is reserved for general purpose use. All locations within this area are read and write accessible under program control.

The overall Data Memory is subdivided into several sectors for the devices. The Special Purpose Data Memory registers addressed from 00H~7FH in Data Memory are common and accessible in all sectors, with the exception of the EEC register at address 40H which is only accessible in Sector 1. Switching between the different Data Memory sectors is achieved by properly setting the Memory Pointers to the correct value. The start address of the Data Memory for all devices is the address 00H.

Device	Special Puroise Data Memory		General Puroise Data Memory	
	Capacity	Sectors	Capacity	Sectors
BS86B12A-3	384 × 8	Sector 0~2: 00H~7FH (EEC register at 40H only accessible in Sector 1)	384 × 8	Sector 0: 80H~FFH Sector 1: 80H~FFH Sector 2: 80H~FFH
BS86C16A-3	512 × 8	Sector 0~3: 00H~7FH (EEC register at 40H only accessible in Sector 1)	512 × 8	Sector 0: 80H~FFH Sector 1: 80H~FFH Sector 2: 80H~FFH Sector 3: 80H~FFH
BS86D20A-3	768 × 8	Sector 0~5: 00H~7FH (EEC register at 40H only accessible in Sector 1)	768 × 8	Sector 0: 80H~FFH Sector 1: 80H~FFH Sector 2: 80H~FFH Sector 3: 80H~FFH Sector 4: 80H~FFH Sector 5: 80H~FFH

Data Memory Sturcture



N=2 for BS86B12A-3; N=3 for BS86C16A-3; N=5 for BS86D20A-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 composed of two bytes, the high byte indicates a sector and the low byte indicates a specific address.

General Purpose Data Memory

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

Special Purpose Data Memory

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

	Sector 0~2
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	
1EH	EEA
1FH	EED
20H	PB
21H	PBC
22H	PBPU
23H	SIMTOC
24H	SIMC0
25H	SIMC1
26H	SIMD
27H	SIMC2/SIMA
28H	USR
29H	UCR1
2AH	UCR2
2BH	BRG
2CH	TXR_RXR
2DH	ADRL
2EH	ADRH
2FH	ADCR0
30H	ADCR1
31H	ACERL
32H	TMPC
33H	SLCDC0
34H	SLCDC1
35H	SLCDC2
36H	
37H	LVDC
38H	IFS
39H	PC
3AH	PCC
3BH	PCPU
3CH	
3DH	CTRL
3EH	
3FH	

	Sector 0, 2	Sector 1
40H		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	PTM1C0	
68H	PTM1C1	
69H	PTM1DL	
6AH	PTM1DH	
6BH	PTM1AL	
6CH	PTM1AH	
6DH	PTM1RPL	
6EH	PTM1RPH	
6FH		
70H		
71H		
72H		
73H		
74H		
75H	PTM2C0	
76H	PTM2C1	
77H	PTM2DL	
78H	PTM2DH	
79H	PTM2AL	
7AH	PTM2AH	
7BH	PTM2RPL	
7CH	PTM2RPH	
7DH		
7EH		
7FH		

: Unused, read as 00H

BS86B12A-3 Special Purpose Data Memory

	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	
1EH	EEA
1FH	EED
20H	PB
21H	PBC
22H	PBPU
23H	SIMTOC
24H	SIMC0
25H	SIMC1
26H	SIMD
27H	SIMC2/SIMA
28H	USR
29H	UCR1
2AH	UCR2
2BH	BRG
2CH	TXR_RXR
2DH	ADRL
2EH	ADRH
2FH	ADCR0
30H	ADCR1
31H	ACERL
32H	TMPC
33H	SLCDC0
34H	SLCDC1
35H	SLCDC2
36H	SLCDC3
37H	LVDC
38H	
39H	PC
3AH	PCC
3BH	PCPU
3CH	
3DH	CTRL
3EH	
3FH	

	Sector 0, 2, 3	Sector 1
40H		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	PTM1C0	
68H	PTM1C1	
69H	PTM1DL	
6AH	PTM1DH	
6BH	PTM1AL	
6CH	PTM1AH	
6DH	PTM1RPL	
6EH	PTM1RPH	
6FH		
70H		
71H		
72H		
73H		
74H		
75H	PTM2C0	
76H	PTM2C1	
77H	PTM2DL	
78H	PTM2DH	
79H	PTM2AL	
7AH	PTM2AH	
7BH	PTM2RPL	
7CH	PTM2RPH	
7DH		
7EH		
7FH		

: Unused, read as 00H

BS86C16A-3 Special Purpose Data Memory

	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	
1EH	EEA
1FH	EED
20H	PB
21H	PBC
22H	PBPU
23H	SIMTOC
24H	SIMC0
25H	SIMC1
26H	SIMD
27H	SIMC2/SIMA
28H	USR
29H	UCR1
2AH	UCR2
2BH	BRG
2CH	TXR_RXR
2DH	ADRL
2EH	ADRH
2FH	ADCR0
30H	ADCR1
31H	ACERL
32H	TMPC
33H	SLCDC0
34H	SLCDC1
35H	SLCDC2
36H	SLCDC3
37H	LVDC
38H	
39H	PC
3AH	PCC
3BH	PCPU
3CH	
3DH	CTRL
3EH	
3FH	

	Sector 0, 2~5	Sector 1
40H		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	PTM1C0	
68H	PTM1C1	
69H	PTM1DL	
6AH	PTM1DH	
6BH	PTM1AL	
6CH	PTM1AH	
6DH	PTM1RPL	
6EH	PTM1RPH	
6FH	TKM416DL	
70H	TKM416DH	
71H	TKM4ROL	
72H	TKM4ROH	
73H	TKM4C0	
74H	TKM4C1	
75H	PTM2C0	
76H	PTM2C1	
77H	PTM2DL	
78H	PTM2DH	
79H	PTM2AL	
7AH	PTM2AH	
7BH	PTM2RPL	
7CH	PTM2RPH	
7DH		
7EH		
7FH		

: Unused, read as 00H

BS86D20A-3 Special Purpose Data Memory

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 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 sector. As the Indirect Addressing Registers are not physically implemented, reading the Indirect Addressing Registers directly will return a result of "00H" and writing to the registers directly will result in no operation.

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

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

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

Indirect Addressing Program Example

Example 1

```
data .section ´data´
adres1   db ?
adres2   db ?
adres3   db ?
adres4   db ?
block    db ?
code .section at 0 ´code´
org00h
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 Data Memory addresses.

Direct Addressing Program Example using extended instructions

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

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

Accumulator – ACC

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

Program Counter Low Byte Register – PCL

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

Look-up Table Registers – TBLP, TBHP, TBLH

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

Status Register – STATUS

This 8-Bit register contains the zero flag (Z), carry flag (C), auxiliary carry flag (AC), overflow flag (OV), 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/W	R/W	R	R	R/W	R/W	R/W	R/W
POR	x	x	0	0	x	x	x	x

“x” unknown

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

Bit 6 **CZ**: The the operational result of different flags for different instuctions.
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 willl 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
C is also affected by a rotate through carry instruction.

EEPROM Data Memory

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

EEPROM Data Memory Structure

The EEPROM Data Memory capacity is 64×8 Bits. Unlike the Program Memory and RAM Data Memory, the EEPROM Data Memory is not directly mapped and is therefore not directly accessible 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.

Device	Capacity	Address
BS86B12A-3	64×8	00H~3FH
BS86C16A-3		
BS86D20A-3		

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 way as any other Special Function Register. The EEC register however, being located in Sector 1, cannot be directly addressed directly and can only be read from or written to indirectly using the MP1H/MP1L or MP2H/MP2L Memory Pointer 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.

EEPROM Control Registers List

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

- **EEA Register**

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

Bit 7 ~ 6 Unimplemented, read as "0"

Bit 5 ~ 0 Data EEPROM address
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 Data EEPROM data
Data EEPROM data Bit 7 ~ Bit 0

• **EEC Register**

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

Bit 7 ~ 4 Unimplemented, read as "0"

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

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

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

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

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

Writing Data to the EEPROM

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

Write Protection

Protection against inadvertent write operation is provided in several ways. After the device is powered-on the Write Enable Bit in the control register will be cleared preventing any write operations. Also at power-on the Memory Pointer high byte register, MP1H or MP2H, will be reset to zero, which means that Data Memory Sector 0 will be selected. As the EEPROM control register is located in Sector 1, this adds a further measure of protection against spurious write operations. During normal program operation, ensuring that the Write Enable Bit in the control register is cleared will safeguard against incorrect write operations.

EEPROM Interrupt

The EEPROM write interrupt is generated when an EEPROM write cycle has ended. The EEPROM interrupt must first be enabled by setting the DEE Bit in the relevant interrupt register. When an EEPROM write cycle ends, the DEF request flag will be set. If the global and EEPROM write interrupts 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 devices should not enter the IDLE or SLEEP mode until the EEPROM read or write operation is totally completed. 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 read if no more read operations are required
CLR MP1H
MOV A, EED                ; move read data to register
MOV READ_DATA, A
```

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

Writing Data to the EEPROM – polling method

```
MOV A, EEPROM_ADRES       ; user defined address
MOV EEA, A
MOV A, EEPROM_DATA        ; user defined data
MOV EED, A
MOV A, 040H               ; setup memory pointer 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 - executed immediately after
                          ; setting WREN Bit
SET EMI
BACK:
SZ IAR1.2                 ; check for write cycle end
JMP BACK
CLR MP1H
```

Oscillators

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

Oscillator Overview

All the devices include two internal oscillators and some devices also include an external oscillator. In addition to being the source of the main system clock the oscillators also provide clock sources for the Watchdog Timer, Time Bases and TMs. External oscillator requiring some external components as well as fully integrated internal oscillators requiring no external components, are provided to form a wide range of both fast and slow system oscillators. For the BS86C16A-3 and BS86D20A-3 devices, the low speed oscillators are selected through the configuration option. 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 devices have the flexibility to optimize the performance/power ratio, a feature especially important in power sensitive portable applications.

Device	Type	Name	Freq.	Pins
All three devices	Internal High Speed RC	HIRC	8/12/16MHz	—
	Internal Low Speed RC	LIRC	32kHz	—
BS86C16A-3/BS86D20A-3	External Low Speed Crystal	LXT	32768Hz	XT1/XT2

Oscillator Types

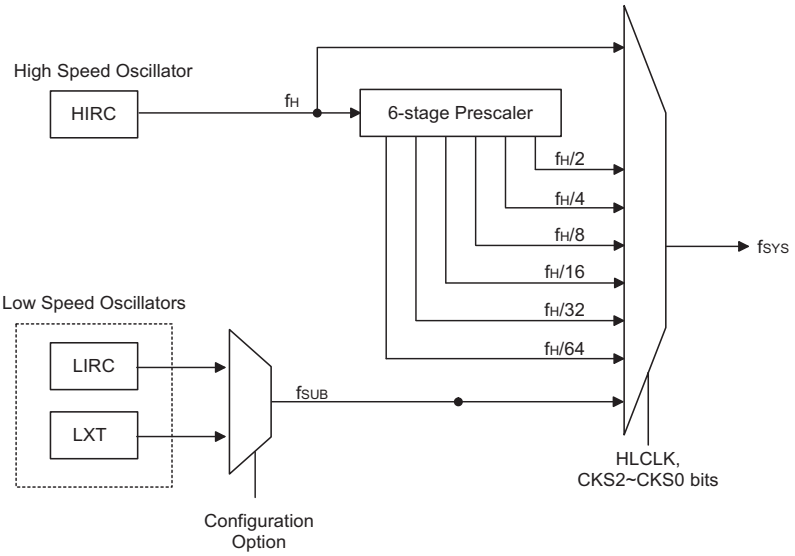
System Clock Configurations

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

The actual source clock used for the low speed oscillator comes from the LIRC oscillator or is chosen via configuration option depending on the selected device. The frequency of the slow speed or high speed system clock is also determined using the HLCLK Bit 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.

Internal 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 8 MHz 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.



Note: For the BS86B12A-3 device, f_{SUB} is directly sourced from the LIRC oscillator without configuration option.

System Clock Configurations

Internal 32kHz Oscillator – LIRC

The Internal 32kHz System Oscillator is one of the low frequency oscillators. It is a fully integrated RC oscillator with a typical frequency of 32kHz at 5V, requiring no external components for its implementation. Device trimming during the manufacturing process and the inclusion of internal frequency compensation circuits are used to ensure that the influence of the power supply voltage, temperature and process variations on the oscillation frequency are minimised.

External 32.768kHz Crystal Oscillator – LXT

For the BS86C16A-3 and BS86D20A-3 devices, the External 32.768kHz Crystal System Oscillator is one of the low frequency oscillator choices, which is selected via configuration option. This clock source has a fixed frequency of 32.768kHz and requires a 32.768kHz crystal to be connected between pins XT1 and XT2. The external resistor and capacitor components connected to the 32.768kHz crystal are necessary to provide oscillation. For applications where precise frequencies are essential, these components may be required to provide frequency compensation due to different crystal manufacturing tolerances. During power-up there is a time delay associated with the LXT oscillator waiting for it to start-up.

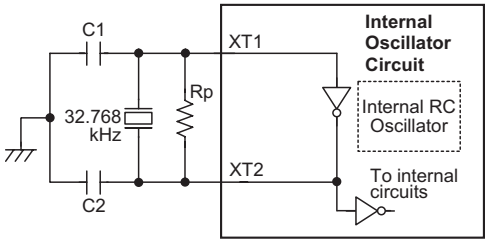
When the microcontroller enters the SLEEP or IDLE Mode, the system clock is switched off to stop microcontroller activity and to conserve power. However, in many microcontroller applications it may be necessary to keep the internal timers operational even when the microcontroller is in the SLEEP or IDLE 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 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 pins 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 pins or other pin-shared functions.
- If the LXT oscillator is used for any clock source, the 32.768kHz crystal should be connected to the XT1/XT2 pins.

For oscillator stability and to minimize 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. Rp, C1 and C2 are required.
2. Although not shown pins have a parasitic capacitance of around 7pF.

External LXT Oscillator

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

32.768kHz Crystal Recommended Capacitor Values

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 Bit	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.

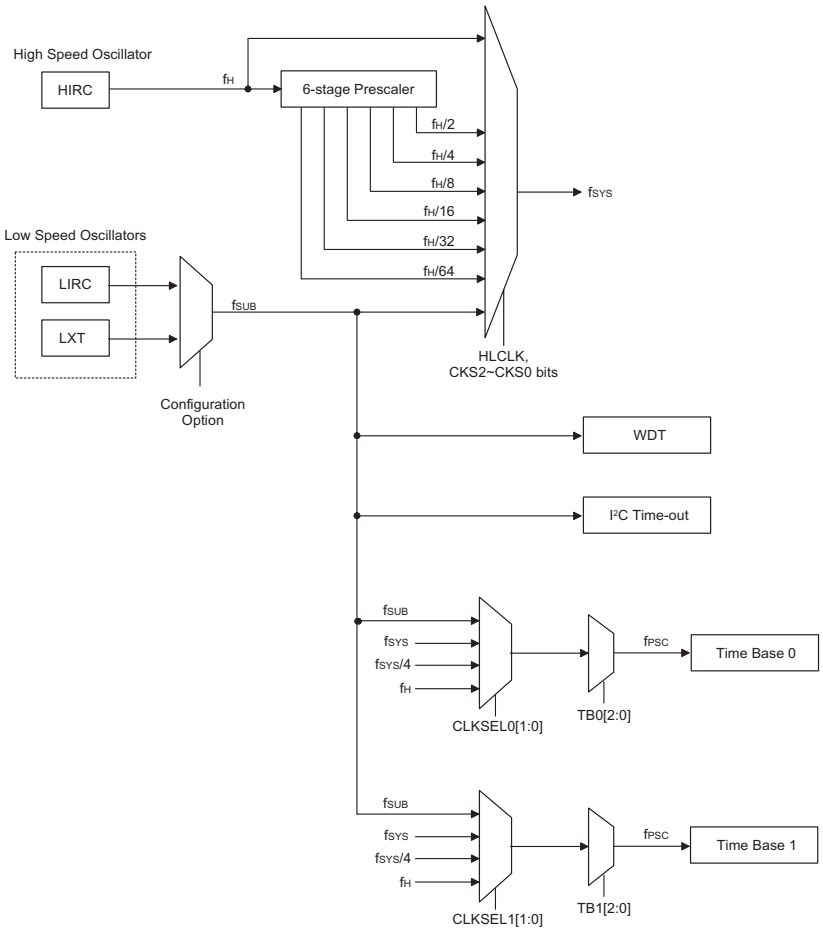
It should be noted that, no matter what condition the LXTLP Bit is set to, the LXT oscillator will always function normally, the only difference is that it will take more time to start up if in the Low-power mode.

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

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 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 $f_H/2$~$f_H/64$.



System Clock Configurations

Note: For the BS86B12A-3 device, f_{SUB} is directly sourced from the LIRC oscillator without configuration option. 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 NORMAL Mode and SLOW Mode. The remaining three modes, the SLEEP, IDLE0 and IDLE1 Mode are used when the microcontroller CPU is switched off to conserve power.

Operating Mode	Description		
	CPU	f_{SYS}	f_{SUB}
NORMAL Mode	On	f_H~f_H/64	On
SLOW Mode	On	f_{SUB}	On
ILDE0 Mode	Off	Off	On
IDLE1 Mode	Off	On	On
SLEEP Mode	Off	Off	Off

NORMAL Mode

As the name suggests this is one of the main operating modes where the microcontroller has all of its functions operational and where the system clock is provided by the high speed oscillator. 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}. Running the microcontroller in this mode allows 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 the f_{SUB} clock will be stopped too, the Watchdog Timer function is disabled.

IDLE0 Mode

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

IDLE1 Mode

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

Control Register

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

• **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**: The 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 1024 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 is the IDLE Mode Control Bit and determines what happens when the HALT instruction is executed. If this Bit is high, when a HALT instruction is executed the device will enter the IDLE Mode. In the IDLE1 Mode the CPU will stop running but the system clock will continue to keep the peripheral functions operational, if FSYSON Bit is high. If FSYSON Bit is low, the CPU and the system clock will all stop in IDLE0 mode. If the Bit is low the device will enter the SLEEP Mode when a HALT instruction is executed.

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

• **CTRL Register**

Bit	7	6	5	4	3	2	1	0
Name	FSYSON	—	HIRCS1	HIRCS0	LXTLP	LVRF	D1	WRF
R/W	R/W	—	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	—	0	0	0	x	0	0

"x" unknown

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

Bit 6 Unimplemented, read as "0".

Bit 5 ~ 4 **HIRCS1~HIRCS0**: HIRC frequency clock select
00: 8MHz
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
This bit is only available for the BS86C16A-3 and BS86D20A-3 devices.

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

Bit 1 Undefined Bit
This Bit can be read or written by user application program.

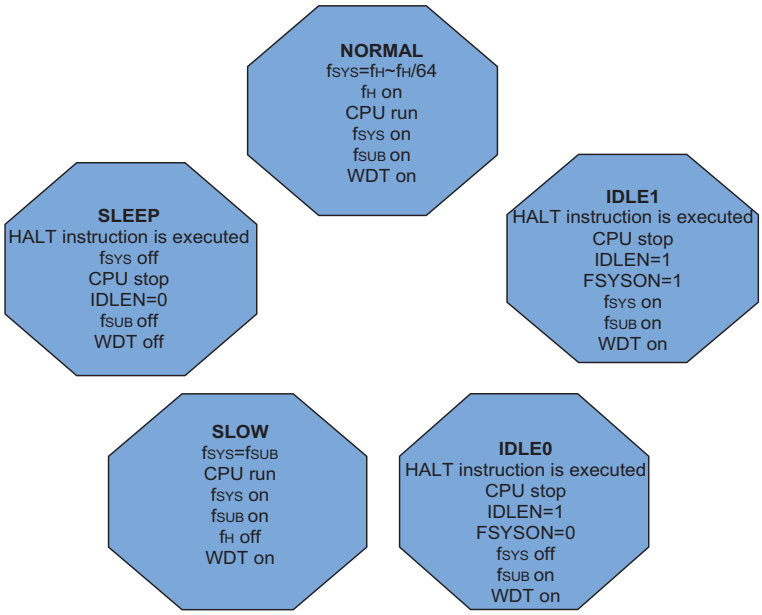
Bit 0 **WRF**: WDT Control register software reset flag
Describe elsewhere

Operating Mode Switching

The 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 NORMAL Mode and SLOW Mode is executed using the HLCLK Bit and CKS2~CKS0 Bits in the SMOD register while Mode Switching from the NORMAL/SLOW Modes to the SLEEP/IDLE Modes is executed via the HALT instruction. When a HALT instruction is executed, whether the devices enter the IDLE Mode or the SLEEP Mode is determined by the condition of the IDLEN Bit in the SMOD register and FSYSON in the CTRL register.

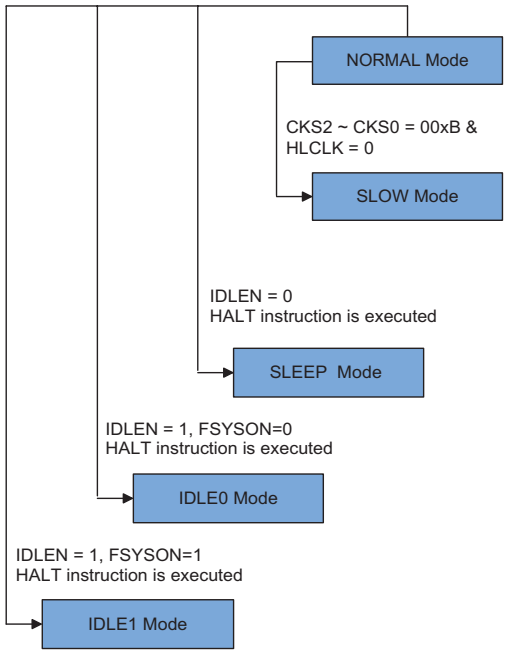
When the HLCLK Bit switches to a low level, which implies that clock source is switched from the high speed clock source, f_H, to the clock source, $f_H/2$~$f_H/64$ or f_{SUB}. If the clock is from the f_{SUB}, the high speed clock source will stop running to conserve power. When this happens it must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



NORMAL Mode to SLOW Mode Switching

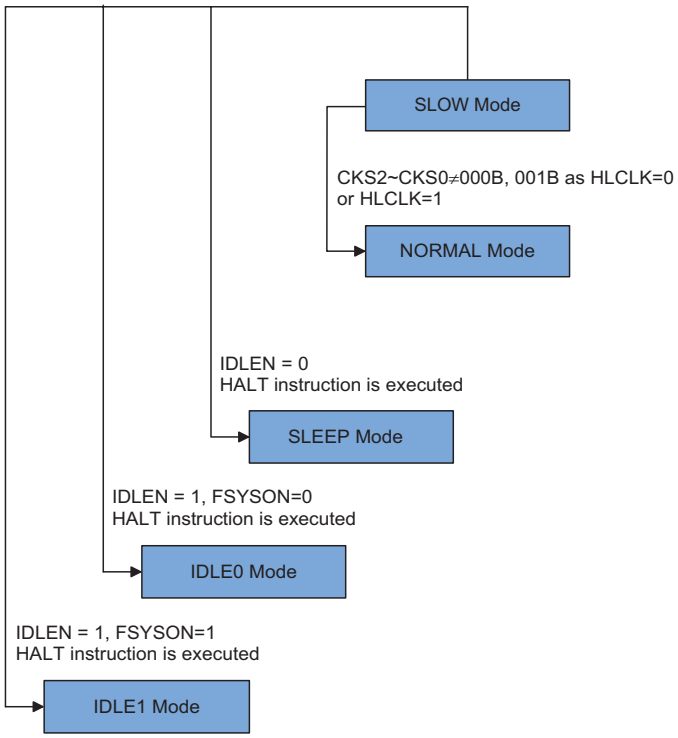
When running in the NORMAL Mode, which uses the high speed system oscillator, and therefore consumes more power, the system clock can switch to run in the SLOW Mode by clearing the HLCLK Bit to zero and setting the CKS2~CKS0 Bits to "000" or "001" in the SMOD register.This will then use the low speed 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 oscillators and therefore requires the selected oscillator to be stable before full mode switching occurs. This is monitored using the LTO Bit in the SMOD register.



SLOW Mode to NORMAL Mode Switching

In SLOW Mode the system uses either the LXT or LIRC low speed system oscillator. To switch back to the NORMAL Mode, where the high speed system oscillator is used, the HLCLK Bit should be set high or HLCLK Bit is low, but CKS2~CKS0 is set to "010", "011", "100", "101", "110" or "111". As a certain amount of time will be required for the high frequency clock to stabilise, the status of the HTO Bit is checked. The amount of time required for high speed system oscillator stabilization is 15~16 clock cycles.



Entering the SLEEP Mode

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

- The system clock and the f_{SUB} 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 WDT will be cleared and stop counting.
- The I/O ports will maintain their present conditions.
- In the status register, the Power Down flag, PDF, will be set and the Watchdog time-out flag, TO, will be cleared.

Entering the IDLE0 Mode

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

Entering the IDLE1 Mode

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

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

Standby Current Considerations

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

Care must also be taken with the loads, which are connected to I/O pins, which are setup as outputs. These should be placed in a condition in which minimum current is drawn or connected only to external circuits that do not draw current, such as other CMOS inputs. In the IDLE1 Mode the system oscillator is on, if the system oscillator is from the high speed system oscillator, the additional standby current will also be perhaps in the order of several hundred micro-amps.

Wake-up

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

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

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

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

System Oscillator	Wake-up Time (SLEEP Mode)	Wake-up Time (IDLE0 Mode)	Wake-up Time (IDLE1 Mode)
HIRC	15~16 HIRC cycles		1~2 HIRC cycles
LIRC	1~2 LIRC cycles		1~2 LIRC cycles
LXT	1024 LXT cycles		1~2 LXT cycles

Wake-Up Time

Programming Considerations

The high speed and low speed oscillators both use the same SST counter. For example, if the system is woken up from the SLEEP Mode the HIRC oscillator needs to start-up from an off state.

If the device is woken up from the SLEEP Mode to the NORMAL Mode, the high speed system oscillator needs an SST period. The device will execute the first instruction after HTO is high. 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 which 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 32kHz 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 operation. The WDTC register is initiated to 01010011B at any reset except 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 software control
01010B or 10101B: Enabled
Other values: Reset MCU (Reset will be active after 2~3 LIRC clock for debounce time.)
If the MCU reset is caused by the WE [4:0] in WDTC software reset, the WRF flag of CTRL register will be set.

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 timeout period.

• CTRL Register

Bit	7	6	5	4	3	2	1	0
Name	FSYSON	—	HIRCS1	HIRCS0	LXTLP	LVRF	D1	WRF
R/W	R/W	—	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	—	0	0	0	x	0	0

"x" unknown

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

Bit 6 Unimplemented, read as "0"

Bit 5~ 4 **HIRCS1~HIRCS0**: HIRC frequency clock select
Describe elsewhere

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

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

Bit 1 Undefined Bit
This Bit can be read or written by user application program.

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.

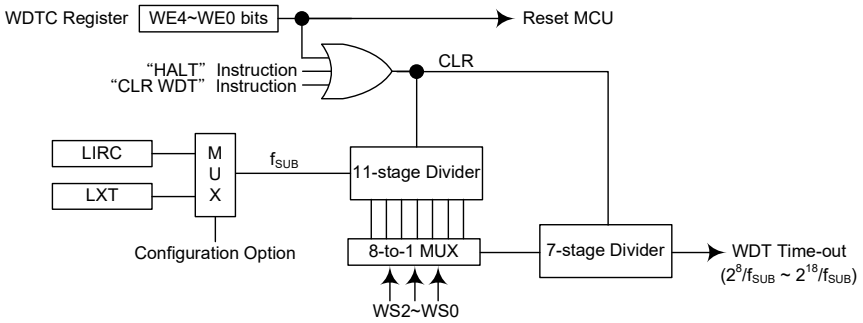
Watchdog Timer Operation

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 WDT instruction will not be executed in the correct manner, in which case the Watchdog Timer will overflow and reset the device. There are five Bits, WE4~WE0, in the WDTC register to enable the WDT function. When the WE4~WE0 Bits value is equal to 01010B or 10101B, the WDT function is enabled. However, if the WE4~WE0 Bits are changed to any other values except 01010B and 10101B, which is caused by the environmental noise, it will reset the microcontroller after 2~3 LIRC clock cycles. After power on these Bits will have a value of 01010B.

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

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

The maximum time-out period is when the 2^{18} division ratio is selected. As an example, with a 32kHz LIRC oscillator as its source clock, this will give a maximum watchdog period of around 8 seconds for the 2^{18} division ratio, and a minimum timeout of 7.8ms for the 2^8 division ration.



Note: For the BS86B12A-3 device, the f_{SUB} is supplied only by the LIRC oscillator.

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.

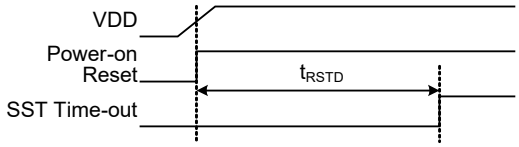
Another type of reset is when the Watchdog Timer overflows and resets the microcontroller. All typesof reset operations result in different register conditions being setup. Another reset exists in the form of a Low Voltage Reset, LVR, where a full reset, is implemented in situations where the power supply voltage falls below a certain threshold.

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.

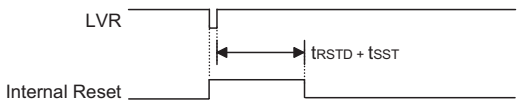


Note: t_{RSTD} is power-on delay, typical time = 50ms

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 high. For a valid LVR signal, a low voltage, i.e., a voltage in the range between 0.9V~V_{LVR} must exist for greater than the value t_{LVR} specified in the A.C. characteristics. If the low voltage state does not exceed this value, the LVR will ignore the low supply voltage and will not perform a reset function. The actual V_{LVR} is fixed at a voltage value of 2.55V. Note that the LVR function will be automatically disabled when the device enters the SLEEP or IDLE mode.



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

Low Voltage Reset Timing Chart

• **CTRL Register**

Bit	7	6	5	4	3	2	1	0
Name	FSYSON	—	HIRCS1	HIRCS0	LXTLP	LVRF	D1	WRF
R/W	R/W	—	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	—	0	0	0	x	0	0

"x" unknown

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

Bit 6 Unimplemented, read as "0".

Bit 5 ~ 4 **HIRCS1~HIRCS0**: HIRC frequency clock select
Describe elsewhere

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

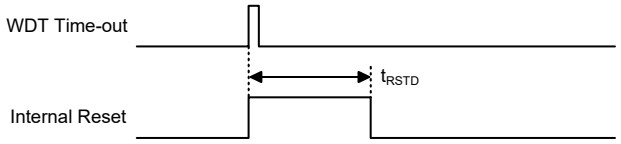
Bit 2 **LVRF**: LVR function reset flag
0: Not occur
1: Occurred
This Bit is set high when a specific Low Voltage Reset situation condition occurs. This Bit can only be cleared to zero by the application program.

Bit 1 Undefined Bit
This Bit can be read or written by user application program.

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

Watchdog Time-out Reset during Normal Operation

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

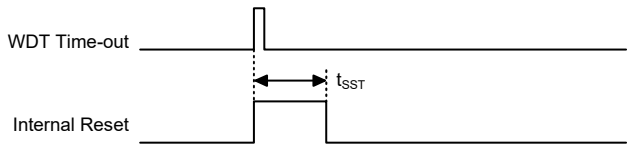


Note: t_{RSTD} is power-on delay, typical time=16.7ms

WDT Time-out Reset during Normal Operation Timing Chart

Watchdog Time-out Reset during SLEEP or IDLE Mode

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



Note: The t_{SST} is 15~16 clock cycles if the system clock source is provided by the HIRC.
The t_{SST} is 1~2 clock for the LIRC. The t_{SST} is 1024 clock for the LXT.

WDT Time-out Reset during SLEEP or IDLE Timing Chart

Reset Initial Conditions

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

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

Note: "u" stands for unchanged

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

Item	Condition After RESET
Program Counter	Reset to zero
Interrupts	All interrupts will be disabled
WDT	Clear after reset, WDT begins counting
Timer Modules	Timer Modules will be turned off
Input/Output Ports	I/O ports will be setup as inputs, AN0~AN7 as A/D input pins
Stack Pointer	Stack Pointer will point to the top of the stack

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

Register	BS86B12A-3	BS86C16A-3	BS86D20A-3	Power On Reset	LVR Reset (Normal Operation)	WDT Time-out (Normal Operation)	WDT Time-out (HALT)*
Program Counter	•	•	•	0000H	0000H	0000H	0000H
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	•			---- -xxx	---- -xxx	---- -uuu	---- -uuu
		•		---- xxxx	---- uuuu	---- uuuu	---- uuuu
			•	---x xxxx	---u uuuu	---u uuuu	---u uuuu
STATUS	•	•	•	xx00 xxxx	uuuu uuuu	uu1u uuuu	uu11 uuuu
SMOD	•	•	•	000- 0011	000- 0011	000- 0011	uuu- uuuu
IAR2	•	•	•	0000 0000	0000 0000	0000 0000	uuuu uuuu
MP2L	•	•	•	0000 0000	0000 0000	0000 0000	uuuu uuuu
MP2H	•	•	•	0000 0000	0000 0000	0000 0000	uuuu uuuu

Register	BS86B12A-3	BS86C16A-3	BS86D20A-3	Power On Reset	LVR Reset (Normal Operation)	WDT Time-out (Normal Operation)	WDT Time-out (HALT)*
INTEG	●	●	●	---- --00	---- --00	---- --00	---- --uu
INTC0	●	●	●	-000 0000	-000 0000	-000 0000	-uuu uuuu
INTC1	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
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
SLEDC0	●	●	●	0101 0101	0101 0101	0101 0101	uuuu uuuu
SLEDC1	●			---- 0101	---- 0101	---- 0101	---- uuuu
		●	●	--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
EEA	●	●	●	--00 0000	--00 0000	--00 0000	--uu uuuu
EED	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PB	●	●	●	1111 1111	1111 1111	1111 1111	uuuu uuuu
PBC	●	●	●	1111 1111	1111 1111	1111 1111	uuuu uuuu
PBPU	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
SIMTOC	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
SIMC0 (SPI Mode)	●	●	●	111- --00	111- --00	111- --00	uuu- --uu
SIMC0 (I^2C Mode)	●	●	●	111- 000-	111- 000-	111- 000-	uuu- uuu-
SIMC1	●	●	●	1000 0001	1000 0001	1000 0001	uuuu uuuu
SIMD	●	●	●	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
SIMC2	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
SIMA	●	●	●	0000 000-	0000 000-	0000 000-	uuuu uuu-
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
ADRL (ADRFS=0)	●	●	●	xxxx ----	xxxx ----	xxxx ----	uuuu ----
ADRL (ADRFS=1)	●	●	●	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
ADRH (ADRFS=0)	●	●	●	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
ADRH (ADRFS=1)	●	●	●	---- xxxx	---- xxxx	---- xxxx	---- uuuu
ADCR0	●	●	●	0110 -000	0110 -000	0110 -000	uuuu -uuu
ADCR1	●	●	●	00-0 -000	00-0 -000	00-0 -000	uu-u -uuu
ACERL	●	●	●	1111 1111	1111 1111	1111 1111	uuuu uuuu
TMPC	●	●	●	--00 0000	--00 0000	--00 0000	--uu uuuu

Register	BS86B12A-3	BS86C16A-3	BS86D20A-3	Power On Reset	LVR Reset (Normal Operation)	WDT Time-out (Normal Operation)	WDT Time-out (HALT)*
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
LVDC	●	●	●	--00 -000	--00 -000	--00 -000	--uu -uuu
IFS	●			---- ---0	---- ---0	---- ---0	---- ---u
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
CTRL	●			0-00 -x00	0-00 -100	0-00 -x00	u-uu -uuu
		●	●	0-00 0x00	0-00 0100	0-00 0x00	u-uu uuuu
PD		●	●	---- 1111	---- 1111	---- 1111	---- uuuu
PDC		●	●	---- 1111	---- 1111	---- 1111	---- uuuu
PDPU		●	●	---- 0000	---- 0000	---- 0000	---- uuuu
TKTMR	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKC0	●	●	●	-000 0000	-000 0000	-000 0000	-uuu uuuu
TK16DL	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TK16DH	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKC1	●	●	●	---- --11	---- --11	---- --11	---- --uu
TKM016DL	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM016DH	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM0ROL	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM0ROH	●	●	●	---- --00	---- --00	---- --00	---- --uu
TKM0C0	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM0C1	●	●	●	0-00 0000	0-00 0000	0-00 0000	u-uu uuuu
TKM116DL	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM116DH	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM1ROL	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM1ROH	●	●	●	---- --00	---- --00	---- --00	---- --uu
TKM1C0	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM1C1	●	●	●	0-00 0000	0-00 0000	0-00 0000	u-uu uuuu
TKM216DL	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM216DH	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
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
CTM0C0	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
CTM0C1	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu

Register	BS86B12A-3	BS86C16A-3	BS86D20A-3	Power On Reset	LVR Reset (Normal Operation)	WDT Time-out (Normal Operation)	WDT Time-out (HALT)*
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
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
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
PTM2C0	•	•	•	0000 0---	0000 0---	0000 0---	uuuu u---
PTM2C1	•	•	•	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM2DL	•	•	•	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM2DH	•	•	•	---- --00	---- --00	---- --00	---- --uu
PTM2AL	•	•	•	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM2AH	•	•	•	---- --00	---- --00	---- --00	---- --uu
PTM2RPL	•	•	•	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM2RPH	•	•	•	---- --00	---- --00	---- --00	---- --uu
EEC	•	•	•	---- 0000	---- 0000	---- 0000	---- uuuu

Note: "*" stands for "warm reset"
"-" not implement
"u" stands for "unchanged"
"x" stands for "unknown"

Input/Output Ports

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

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

Device	Register Name	Bit							
		7	6	5	4	3	2	1	0
BS86B12A-3 BS86C16A-3 BS86D20A-3	PAWU	PAWU7	—	—	PAWU4	PAWU3	PAWU2	PAWU1	PAWU0
	PAPU	PAPU7	—	—	PAPU4	PAPU3	PAPU2	PAPU1	PAPU0
	PA	PA7	—	—	PA4	PA3	PA2	PA1	PA0
	PAC	PAC7	—	—	PAC4	PAC3	PAC2	PAC1	PAC0
	PBPU	PBPU7	PBPU6	PBPU5	PBPU4	PBPU3	PBPU2	PBPU1	PBPU0
	PB	PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0
	PBC	PBC7	PBC6	PBC5	PBC4	PBC3	PBC2	PBC1	PBC0
	PCPU	PCPU7	PCPU6	PCPU5	PCPU4	PCPU3	PCPU2	PCPU1	PCPU0
	PC	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0
	PCC	PCC7	PCC6	PCC5	PCC4	PCC3	PCC2	PCC1	PCC0
BS86C16A-3 BS86D20A-3	PDPU	—	—	—	—	PDPU3	PDPU2	PDPU1	PDPU0
	PD	—	—	—	—	PD3	PD2	PD1	PD0
	PDC	—	—	—	—	PDC3	PDC2	PDC1	PDC0

I/O Register List

PAWUn: PA wake-up function control
0: Disable
1: Enable

PAn/PBn/PCn/PDn: I/O Data Bit
0: Data 0
1: Data 1

PACn/PBCn/PCCn/PDCn: I/O Type selection
0: Output
1: Input

PAPUn/PBPUn/PCPUn/PDPUn: I/O Pull-high function control
0: Disable
1: Enable

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 registers PAPU~PDPU, and are implemented using weak PMOS transistors.

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.

I/O Port Control Registers

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

Pin-remapping Function

There is an IFS register which is used to select the PTP2I pin function for the BS86B12A-3 device.

- **IFS Register**

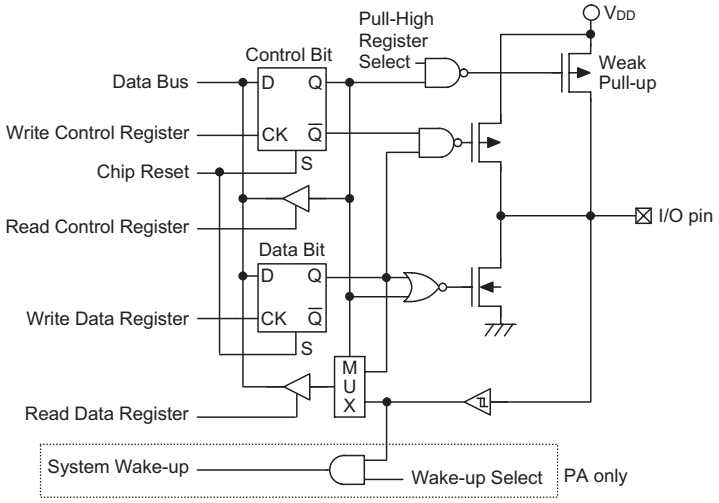
Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	—	PTP2IS
R/W	—	—	—	—	—	—	—	R/W
POR	—	—	—	—	—	—	—	0

Bit 7 ~ 1 Unimplemented, read as "0"

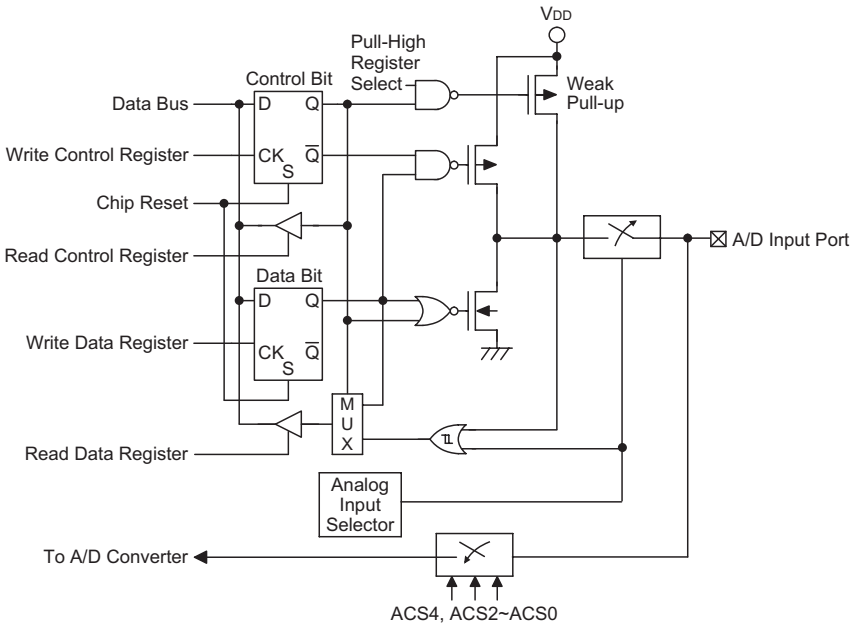
Bit 0 **PTP2IS**: PTP2I pin remapping control
0: PTP2I on PB7 (default)
1: PTP2I on PB4

I/O Pin Structures

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



Generic Input/Output Structure



A/D Input/Output Sturcture

Source Current Selection

The source current of each pin in these devices can be configured with different source current which is selected by the corresponding pin source current select Bits. These source current Bits are available when the corresponding pin is configured as a CMOS output. Otherwise, these select Bits have no effect. Users should refer to the D.C. Characteristics section to obtain the exact value for different applications.

• **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	1	0	1	0	1	0	1

Bit 7 ~ 6 **PBPS3~PBPS2**: PB7~PB4 source current select
00 : source = Level 0 (min.)
01 : source = Level 1
10 : source = Level 2
11 : source = Level 3 (max.)
These Bits are available when the corresponding pin is configured as a CMOS output.

Bit 5 ~ 4 **PBPS1~PBPS0**: PB3~PB0 source current select
00 : source = Level 0 (min.)
01 : source = Level 1
10 : source = Level 2
11 : source = Level 3 (max.)
These Bits are available when the corresponding pin is configured as a CMOS output.

Bit 3 ~ 2 **PAPS3~PAPS2**: PA7 and PA4 source current select
00 : source = Level 0 (min.)
01 : source = Level 1
10 : source = Level 2
11 : source = Level 3 (max.)
These Bits are available when the corresponding pin is configured as a CMOS output.

Bit 1 ~ 0 **PAPS1~PAPS0**: PA3~PA0 source current select
00 : source = Level 0 (min.)
01 : source = Level 1
10 : source = Level 2
11 : source = Level 3 (max.)
These Bits are available when the corresponding pin is configured as a CMOS output.

• **SLEDC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	PDPS1	PDPS0	PCPS3	PCPS2	PCPS1	PCPS0
R/W	—	—	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	—	0	1	0	1	0	1

Bit 7 ~ 6 Unimplemented, read as "0"

Bit 5 ~ 4 **PDPS1~PDPS0**: PD3~PD0 source current select
00 : source = Level 0 (min.)
01 : source = Level 1
10 : source = Level 2
11 : source = Level 3 (max.)
These Bits are available when the corresponding pin is configured as a CMOS output.
Note: These bits are only available for the BS86C16A-3 and BS86D20A-3 devices.

Bit 3 ~ 2 **PCPS3~PCPS2**: PC7~PC4 source current select
00 : source = Level 0 (min.)
01 : source = Level 1
10 : source = Level 2
11 : source = Level 3 (max.)
These Bits are available when the corresponding pin is configured as a CMOS output.

Bit 1 ~ 0 **PCPS1~PCPS0**: PC3~PC0 source current select
00 : source = Level 0 (min.)
01 : source = Level 1
10 : source = Level 2
11 : source = Level 3 (max.)
These Bits are available when the corresponding pin is configured as a CMOS output.

Programming Considerations

Within the user program, one of the first things to consider is port initialisation. After a reset, all of the I/O data and port control registers will be set high. This means that all I/O pins will default to an input state, the level of which depends on the other connected circuitry and whether pull-high selections have been chosen. If the port control registers, PAC~PDC, 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, PA~PD, 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 device is the ability to control and measure time. To implement time related functions the device includes several Timer Modules, abbreviated to the name TM. The TMs are multi-purpose timing units and serve to provide operations such as Timer/Counter, Input Capture, Compare Match Output and Single Pulse Output as well as being the functional unit for the generation of PWM signals. Each of the TMs has two individual interrupts. The addition of input and output pins for each TM ensures that users are provided with timing units with a wide and flexible range of features.

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

Introduction

Each device contains a 10-Bit Compact TM, CTM0, and two 10-Bit Periodic TMs, PTM1 and PTM2. Although similar in nature, the different TM types vary in their feature complexity. The common features to the Compact and Periodic TMs will be described in this section and the detailed operation will be described in corresponding sections. The main features and differences between the two types of TMs are summarised in the accompanying table.

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

TM Function Summary

CTM0	PTM1	PTM2
10-Bit CTM	10-Bit PTM	10-Bit PTM

TM Name/Type Reference

TM Operation

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

TM Clock Source

The clock source which drives the main counter in each TM can originate from various sources. The selection of the required clock source is implemented using the xTnCK2~xTnCK0 Bits in the xTMn control registers, where “x” can stand for C or P and “n” is the serial number. The clock source can be a ratio of either 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 or for event counting.

TM Interrupts

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

TM External Pins

Each of the TMs, irrespective of what type, has one or two TM input pins, with the label xTCKn and xTPnI respectively. The TM input pin, xTCKn, is essentially a clock source for the TM 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 TM 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 PTM.

The other TM 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.

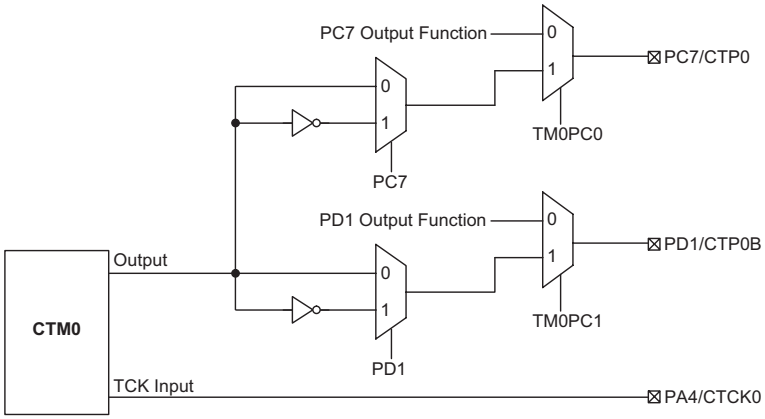
The TMs each has two output pins. When the TM is in the Compare Match Output Mode, these pins can be controlled by the TM to switch to a high or low level or to toggle when a compare match situation occurs. The external xTPn and xTPnB output pins are also the pins where the TM generates the PWM output waveform. As the TM output pins are pin-shared with other function, the TM output function must first be setup using the associated register. A single 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	CTM0	PTM1	PTM2
BS86B12A-3	CTCK0 CTP0, CTP0B	PTCK1, PTP1I PTP1, PTP1B	PTCK2, PTP2I PTP2, PTP2B
BS86C16A-3			
BS86D20A-3			

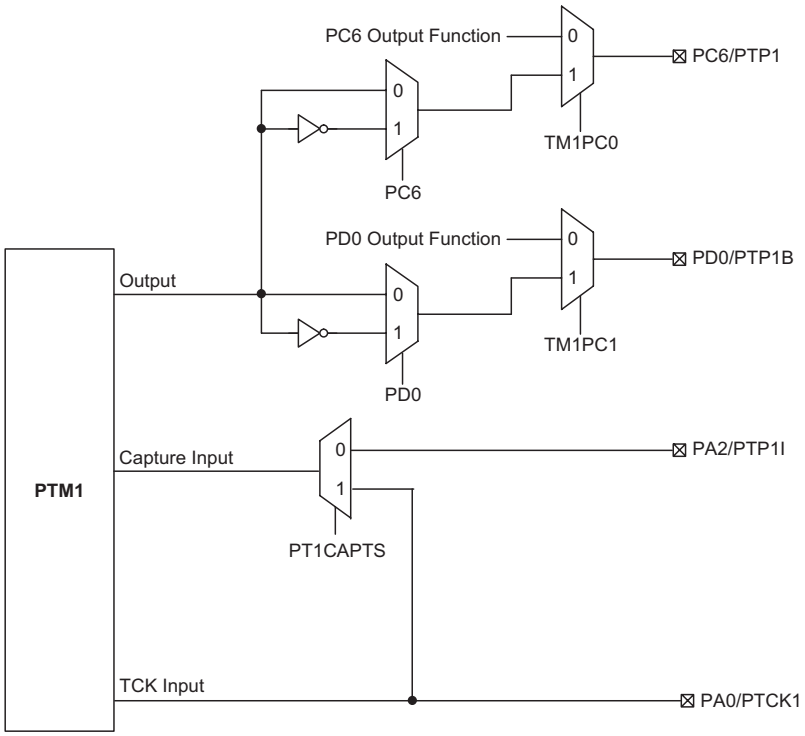
TM Input/Output Pins

TM Input/Output Pin Control Register

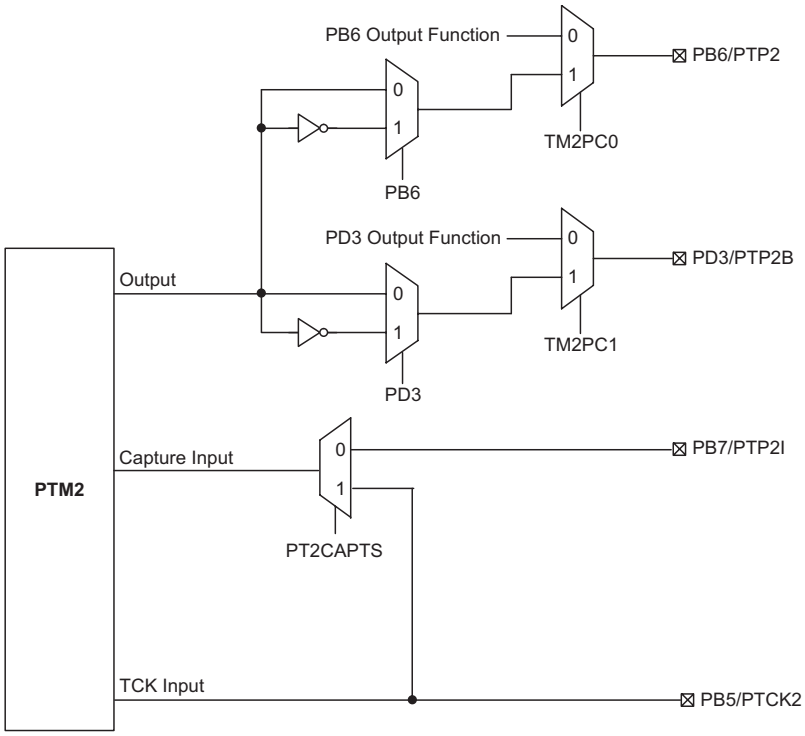
Selecting to have a TM input/output or whether to retain its other shared function is implemented using one register, with a single Bit in each register corresponding to a TM input/output pin. Setting the Bit high will setup the corresponding pin as a TM input/output, if reset to zero the pin will retain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



CTM0 Function Pin Control Block Diagram



PTM1 Function Pin Control Block Diagram



PTM2 Function Pin Control Block Diagram

• **TMPC Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	TM2PC1	TM2PC0	TM1PC1	TM1PC0	TM0PC1	TM0PC0
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 **TM2PC1**: PTP2B pin Control
0: Disabled
1: Enabled

Bit 4 **TM2PC0**: PTP2 pin control
0: Disabled
1: Enabled

Bit 3 **TM1PC1**: PTP1B pin Control
0: Disabled
1: Enabled

Bit 2 **TM1PC0**: PTP1 pin control
0: Disabled
1: Enabled

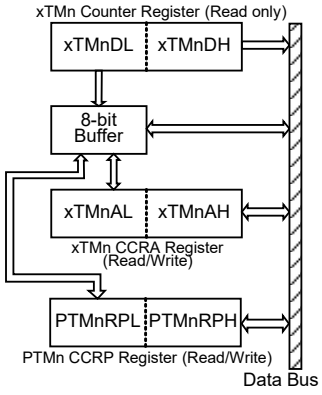
Bit 1 **TM0PC1**: CTP0B pin Control
0: Disabled
1: Enabled

Bit 0 **TM0PC0**: CTP0 pin Control
0: Disabled
1: Enabled

Programming Considerations

The TM Counter Registers and the Capture/Compare CCRA and CCRP registers, being 10-Bit, 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 registers is carried out in a specific way described above, it is recommended to use the "MOV" instruction to access the CCRA and CCRP low byte registers, named xTMnAL and PTMnRPL, in the following access procedures. Accessing the CCRA or CCRP low byte register without following these access procedures will result in unpredictable values.

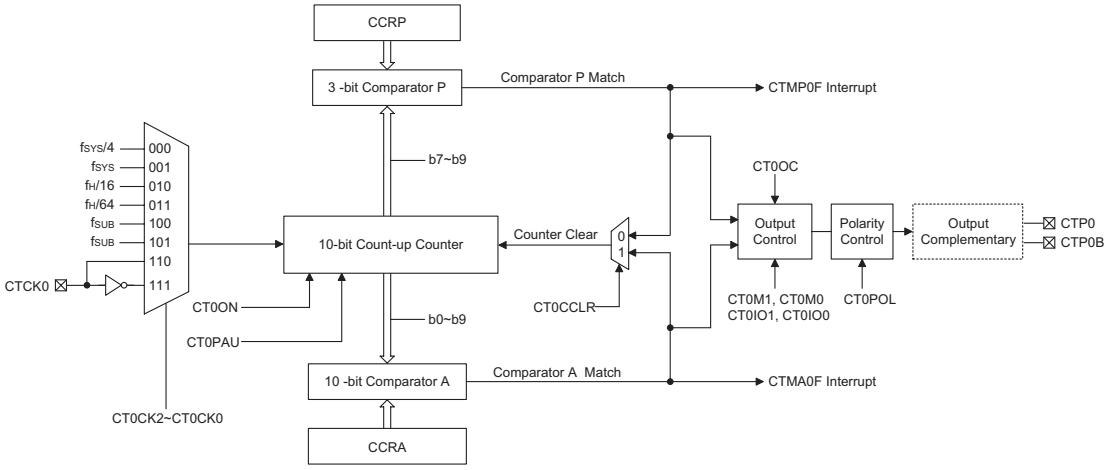


The following steps show the read and write procedures:

- 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

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



Compact Type TM Block Diagram

Compact TM Operation

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

The only way of changing the value of the 10-Bit counter using the application program, is to clear the counter by changing the CT0ON 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 CTM0 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 one output pin. All operating setup conditions are selected using relevant internal registers.

Compact Type TM Register Description

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

Register Name	Bit							
	7	6	5	4	3	2	1	0
CTM0C0	CT0PAU	CT0CK2	CT0CK1	CT0CK0	CT0ON	CT0RP2	CT0RP1	CT0RP0
CTM0C1	CT0M1	CT0M0	CT0IO1	CT0IO0	CT0OC	CT0POL	CT0DPX	CT0CCLR
CTM0DL	D7	D6	D5	D4	D3	D2	D1	D0
CTM0DH	—	—	—	—	—	—	D9	D8
CTM0AL	D7	D6	D5	D4	D3	D2	D1	D0
CTM0AH	—	—	—	—	—	—	D9	D8

Compact TM Register List

- **CTM0C0 Register**

Bit	7	6	5	4	3	2	1	0
Name	CT0PAU	CT0CK2	CT0CK1	CT0CK0	CT0ON	CT0RP2	CT0RP1	CT0RP0
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 **CT0PAU**: CTM0 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 CTM0 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 **CT0CK2~CT0CK0**: Select CTM0 Counter clock
000: $f_{SYS}/4$
001: f_{SYS}
010: $f_H/16$
011: $f_H/64$
100: f_{SUB}
101: f_{SUB}
110: CTCK0 rising edge clock
111: CTCK0 falling edge clock
These three Bits are used to select the clock source for the CTM0. 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 **CT0ON**: CTM0 Counter On/Off Control
0: Off
1: On
This Bit controls the overall on/off function of the CTM0. Setting the Bit high enables the counter to run, clearing the Bit disables the CTM0. Clearing this Bit to zero will stop the counter from counting and turn off the CTM0 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.
If the CTM0 is in the Compare Match Output Mode then the CTM0 output pin will be reset to its initial condition, as specified by the CT0OC Bit, when the CT0ON Bit changes from low to high.

Bit 2~0 **CT0RP2~CT0RP0**: CTM0 CCRP 3-Bit register, compared with the CTM0 Counter Bit 9 ~ Bit 7

Comparator P Match Period
000: 1024 CTM0 clocks
001: 128 CTM0 clocks
010: 256 CTM0 clocks
011: 384 CTM0 clocks
100: 512 CTM0 clocks
101: 640 CTM0 clocks
110: 768 CTM0 clocks
111: 896 CTM0 clocks

These three Bits are used to setup the value on the internal CCRP 3-Bit register, which are then compared with the internal counter's highest three Bits. The result of this comparison can be selected to clear the internal counter if the CT0CCLR Bit is set to zero. Setting the CT0CCLR Bit to zero ensures that a compare match with the CCRP values will reset the internal counter. As the CCRP Bits are only compared with the highest three counter Bits, the compare values exist in 128 clock cycle multiples. Clearing all three Bits to zero is in effect allowing the counter to overflow at its maximum value.

• CTM0C1 Register

Bit	7	6	5	4	3	2	1	0
Name	CT0M1	CT0M0	CT0IO1	CT0IO0	CT0OC	CT0POL	CT0DPX	CT0CCLR
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 **CT0M1~CT0M0**: Select CTM0 Operating Mode
00: Compare Match Output Mode
01: Undefined
10: PWM Mode
11: Timer/Counter Mode

These Bits setup the required operating mode for the CTM0. To ensure reliable operation the CTM0 should be switched off before any changes are made to the CT0M1 and CT0M0 Bits. In the Timer/Counter Mode, the CTM0 output pin control must be disabled.

Bit 5~4 **CT0IO1~CT0IO0**: Select CTP0 output function

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

PWM 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 CTM0 output pin changes state when a certain condition is reached. The function that these Bits select depends upon in which mode the CTM0 is running.

In the Compare Match Output Mode, the CT0IO1 and CT0IO0 Bits determine how the CTM0 output pin changes state when a compare match occurs from the Comparator A. The CTM0 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 CTM0 output pin should be setup using the CT0OC Bit in the CTM0C1 register. Note that

the output level requested by the CT0IO1 and CT0IO0 Bits must be different from the initial value setup using the CT0OC Bit otherwise no change will occur on the CTM0 output pin when a compare match occurs. After the CTM0 output pin changes state it can be reset to its initial level by changing the level of the CT0ON Bit from low to high. In the PWM Mode, the CT0IO1 and CT0IO0 Bits determine how the CTM0 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 CT0IO1 and CT0IO0 Bits only after the CTM0 has been switched off. Unpredictable PWM outputs will occur if the CT0IO1 and CT0IO0 Bits are changed when The CTM0 is running.

Bit 3 **CT0OC**: CTP0 Output control Bit

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

PWM Mode
0: Active low
1: Active high

This is the output control Bit for the CTM0 output pin. Its operation depends upon whether CTM0 is being used in the Compare Match Output Mode or in the PWM Mode. It has no effect if the CTM0 is in the Timer/Counter Mode. In the Compare Match Output Mode it determines the logic level of the CTM0 output pin before a compare match occurs. In the PWM Mode it determines if the PWM signal is active high or active low.

Bit 2 **CT0POL**: CTP0 Output polarity Control
0: Non-invert
1: Invert

This Bit controls the polarity of the CTP0 output pin. When the Bit is set high the CTM0 output pin will be inverted and not inverted when the Bit is zero. It has no effect if the CTM0 is in the Timer/Counter Mode.

Bit 1 **CT0DPX**: CTM0 PWM period/duty Control
0: CCRP - period; CCRA - duty
1: CCRP - duty; CCRA - period

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

Bit 0 **CT0CCLR**: Select CTM0 Counter clear condition
0: CTM0 Comparatror P match
1: CTM0 Comparatror A match

This Bit is used to select the method which clears the counter. Remember that the Compact CTM0 contains two comparators, Comparator A and Comparator P, either of which can be selected to clear the internal counter. With the CT0CCLR 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 CT0CCLR Bit is not used in the PWM Mode.

• **CTM0DL 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 CTM0 Counter Low Byte Register Bit 7 ~ Bit 0
CTM0 10-Bit Counter Bit 7 ~ Bit 0

• **CTM0DH 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 CTM0 Counter High Byte Register Bit 1 ~ Bit 0
CTM0 10-Bit Counter Bit 9 ~ Bit 8

• **CTM0AL 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 CTM0 CCRA Low Byte Register Bit 7 ~ Bit 0
CTM0 10-Bit CCRA Bit 7 ~ Bit 0

• **CTM0AH 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 CTM0 CCRA High Byte Register Bit 1 ~ Bit 0
CTM0 10-Bit CCRA Bit 9 ~ Bit 8

Compact Type TM Operating Modes

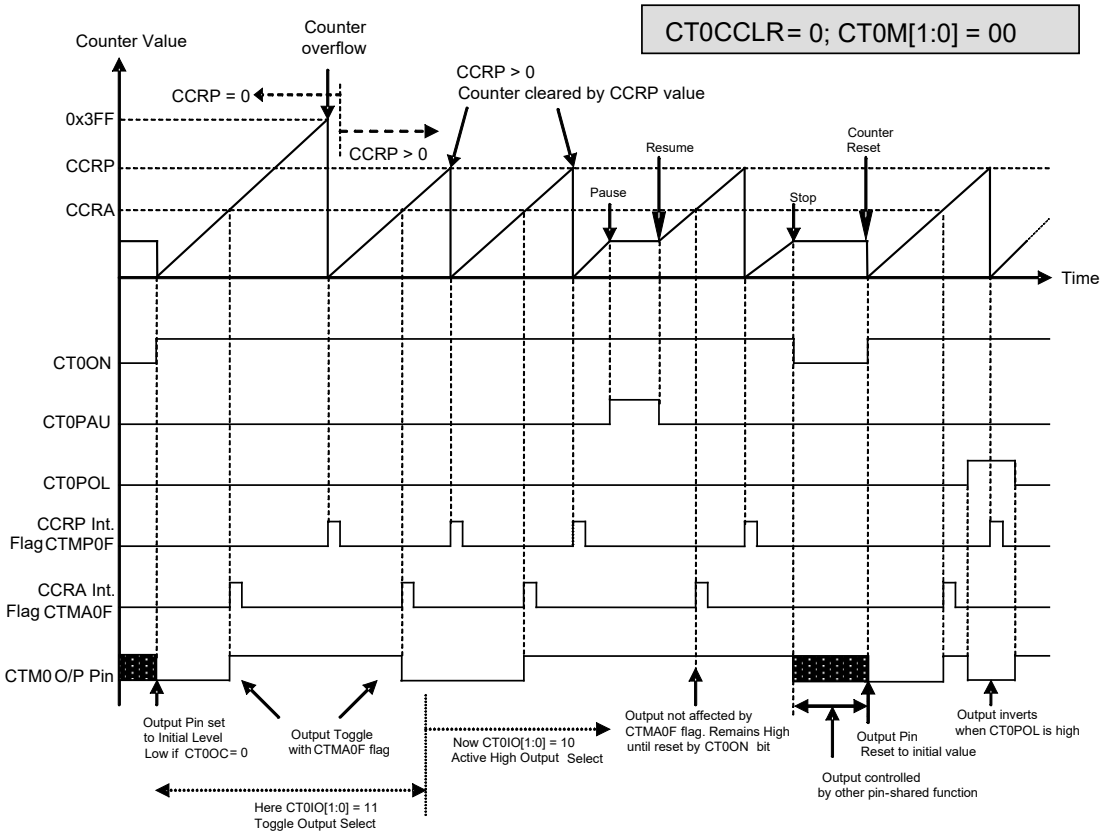
The Compact Type TM can operate in one of three operating modes, Compare Match Output Mode, PWM Output Mode or Timer/Counter Mode. The operating mode is selected using the CT0M1 and CT0M0 Bits in the CTM0C1 register.

Compare Match Output Mode

To select this mode, Bits CT0M1 and CT0M0 in the CTM0C1 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 CT0CCLR 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 CTMA0F and CTMP0F interrupt request flags for Comparator A and Comparator P respectively, will both be generated.

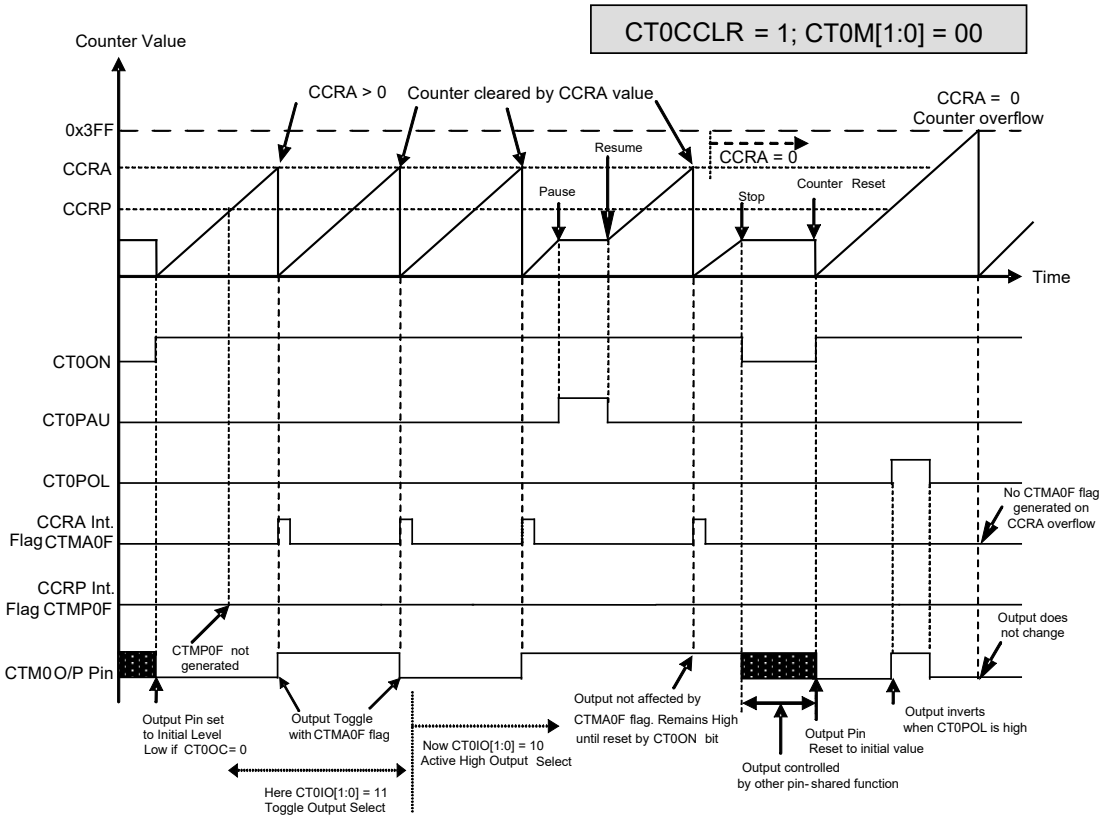
If the CT0CCLR Bit in the CTM0C1 register is high then the counter will be cleared when a compare match occurs from Comparator A. However, here only the CTMA0F interrupt request flag will be generated even if the value of the CCRP Bits is less than that of the CCRA registers. Therefore when CT0CCLR is high no CTMP0F interrupt request flag will be generated. If the CCRA Bits are all zero, the counter will overflow when its reaches its maximum 10-Bit, 3FF Hex, value, however here the CTMA0F interrupt request flag will not be generated.

As the name of the mode suggests, after a comparison is made, the CTM0 output pin will change state. The CTM0 output pin condition however only changes state when a CTMA0F interrupt request flag is generated after a compare match occurs from Comparator A. The CTMP0F interrupt request flag, generated from a compare match occurs from Comparator P, will have no effect on the CTM0 output pin. The way in which the CTM0 output pin changes state are determined by the condition of the CT0IO1 and CT0IO0 Bits in the CTM0C1 register. The CTM0 output pin can be selected using the CT0IO1 and CT0IO0 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 CTM0 output pin, which is setup after the CT0ON Bit changes from low to high, is setup using the CT0OC Bit. Note that if the CT0IO1 and CT0IO0 Bits are zero then no pin change will take place.



Compare Match Output Mode – CT0CCLR=0

Note: 1. With CT0CCLR=0, a Comparator P match will clear the counter
2. The CTM0 output pin controlled only by the CTMA0F flag
3. The output pin reset to initial state by a CT0ON Bit rising edge



Compare Match Output Mode – CT0CCLR=1

Note: 1. With CT0CCLR=1, a Comparator A match will clear the counter
2. The CTM output pin controlled only by the CTMA0F flag
3. The output pin reset to initial state by a CT0ON rising edge
4. The CTMP0F flags is not generated when CT0CCLR=1

Timer/Counter Mode

To select this mode, Bits CT0M1 and CT0M0 in the CTM0C1 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 CTM0 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 CTM0 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 CT0M1 and CT0M0 in the CTM0C1 register should be set to 10 respectively. The PWM function within the CTM0 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 CTM0 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 CT0CCLR Bit has no effect on the PWM operation. Both of the CCRA and CCRP registers are used to generate the PWM waveform, one register is used to clear the internal counter and thus control the PWM waveform frequency, while the other one is used to control the duty cycle. Which register is used to control either frequency or duty cycle is determined using the CT0DPX Bit in the CTM0C1 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 CT0OC Bit In the CTM0C1 register is used to select the required polarity of the PWM waveform while the two CT0IO1 and CT0IO0 Bits are used to enable the PWM output or to force the CTM0 output pin to a fixed high or low level. The CT0POL Bit is used to reverse the polarity of the PWM output waveform.

- **CTM, PWM Mode, Edge-aligned Mode, CT0DPX=0**

CCRP	001b	010b	011b	100b	101b	110b	111b	000b
Period	128	256	384	512	640	768	896	1024
Duty	CCRA							

If f_{SYS}=16MHz, CTM0 clock source is f_{SYS}/4, CCRP=100b, CCRA=128,

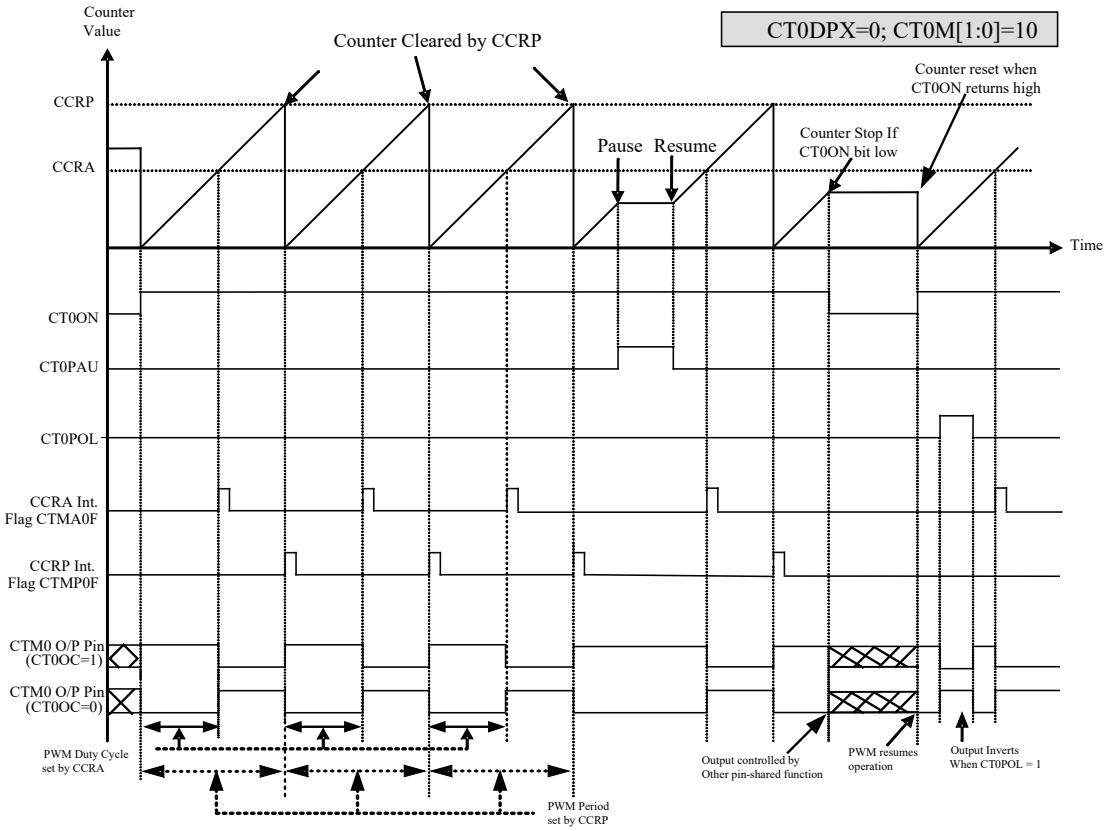
The CTM0 PWM output frequency=(f_{SYS}/4) / 512=f_{SYS}/2048=7.8125 kHz, duty=128/512=25%.

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

- **CTM, PWM Mode, Edge-aligned Mode, CT0DPX=1**

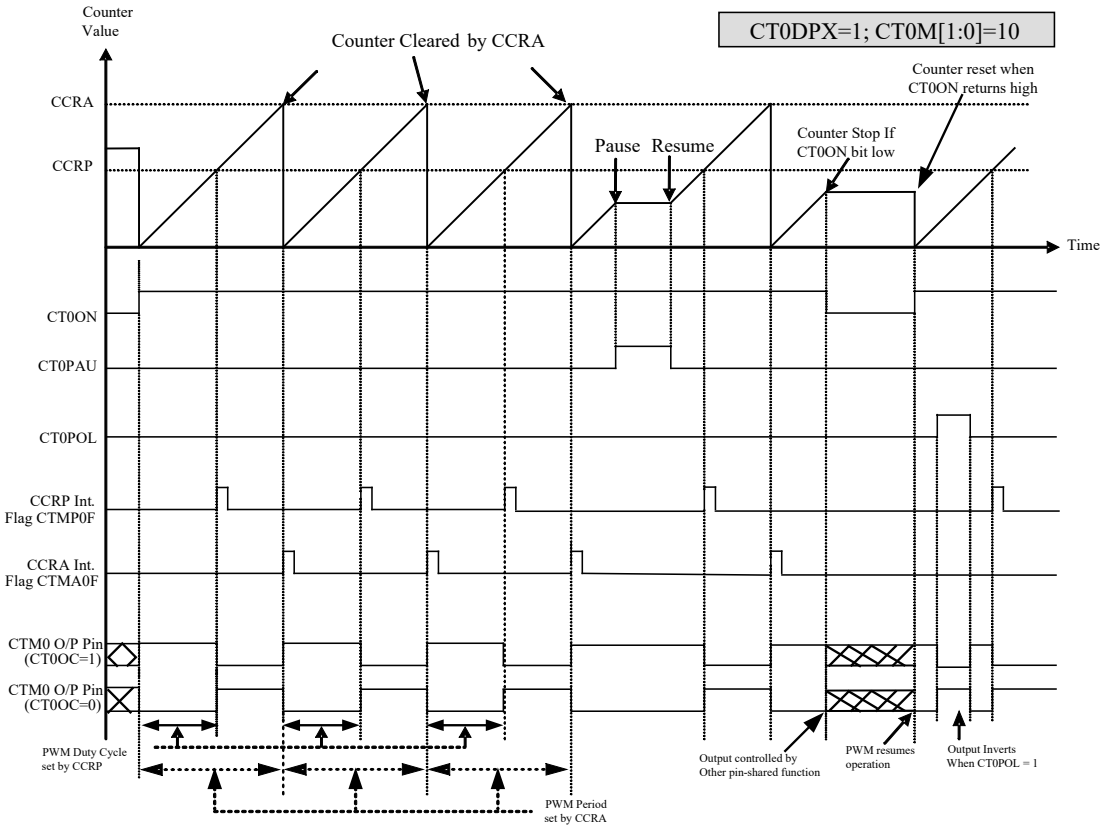
CCRP	001b	010b	011b	100b	101b	110b	111b	000b
Period	CCRA							
Duty	128	256	384	512	640	768	896	1024

The PWM output period is determined by the CCRA register value together with the CTM0 clock while the PWM duty cycle is defined by the CCRP register value.



PWM Mode – CT0DPX=0

Note: 1. Here CT0DPX=0 – Counter cleared by CCRP
2. A counter clear sets PWM Period
3. The internal PWM function continues running even when CT0IO[1:0]=00 or 01
4. The CT0CCLR Bit has no influence on PWM operation

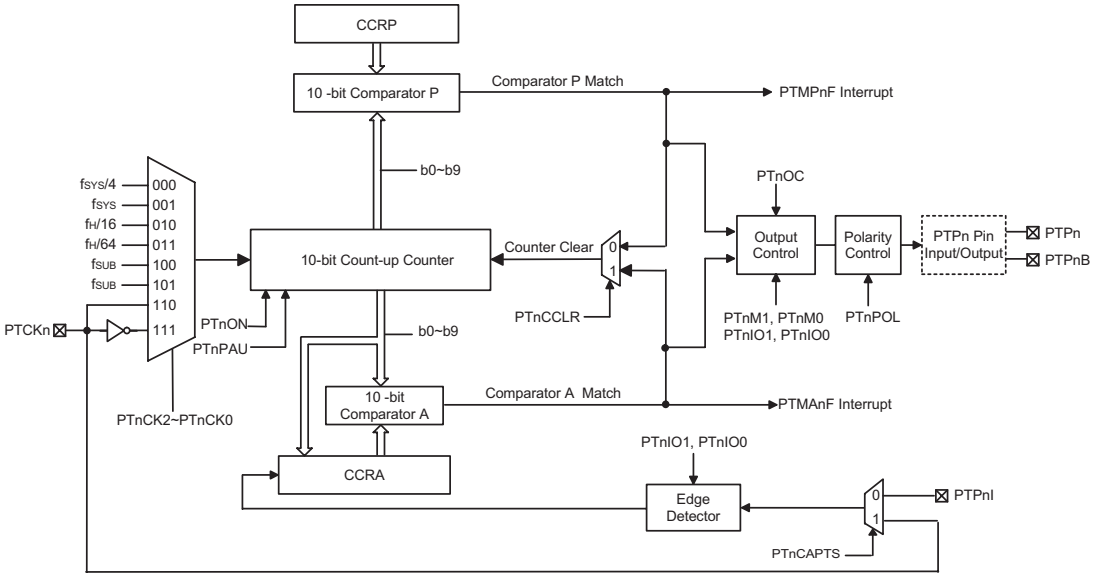


PWM Mode – CT0DPX=1

Note: 1. Here CT0DPX=1 – Counter cleared by CCRA
2. A counter clear sets PWM Period
3. The internal PWM function continues even when CT0IO[1:0]=00 or 01
4. The CT0CCLR Bit has no influence on PWM operation

Periodic Type TM – PTM1 & PTM2

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 be controlled with two external input pins and can drive two external output pins.



Note: 1. PTPnB is the inverse signal of PTPn.

2. For the BS86B12A-3 device, the PTP2I pin source can be selected using the IFS register.

Periodic Type TM Block Diagram (n=1 or 2)

Periodic TM Operation

At the core is a 10 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 10-Bit wide.

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 more than one output pin. All operating setup conditions are selected using relevant internal registers.

Periodic Type TM Register Description

Overall operation of the Periodic Type TM is controlled using a series of registers. A read only register pair exists to store the internal counter 10-Bit value, while two read/write register pairs exist to store the internal 10-Bit CCRA value 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	D7	D6	D5	D4	D3	D2	D1	D0
PTMnRPH	—	—	—	—	—	—	D9	D8

10-Bit Periodic TM Register List (n=1 or 2)

- **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 PTM will remain powered up and continue to consume power. The counter will retain its residual value when this Bit changes from low to high and resume counting from this value when the Bit changes to a low value again.

Bit 6 ~ 4 **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 PTM. The external pin clock source can be chosen to be active on the rasing 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 PTM. Setting the Bit high enables the counter to run, clearing the Bit disables the PTM. Clearing this Bit to zero will stop the counter from counting and turn off the PTM which will reduce its power consumption. When the Bit changes state from low to high the internal counter value will be reset to zero, however when the Bit changes from high to low, the internal counter will retain its residual value until the Bit returns high again.

If the PTM is in the Compare Match Output Mode, PWM output Mode or Single Pulse Output Mode then the PTM output pin will be reset to its initial condition, as specified by the 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 Mode or Single Pulse Output Mode
11: Timer/Counter Mode

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

Bit 5 ~ 4 **PTnIO1~PTnIO0**: Select PTMn external pin function

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

PWM 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 falling/rising edge of PTPnI or PTCKn
11: Input capture disabled

Timer/counter Mode:
Unused

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

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

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

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

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

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

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

Bit 2 **PTnPOL**: PTPn Output polarity Control
0: Non-invert
1: Invert

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

Bit 1 **PTnCAPTS**: PTMn capture trigger source select
0: From PTPnI
1: From PTCKn

Bit 0 **PTnCCLR**: Select PTMn Counter clear condition
0: PTM Comparatror P match
1: PTM Comparatror 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, Single Pulse or Input Capture Mode.

- **PTMnDL Register**

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

Bit 7 ~ 0 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	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 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	—	—	—	—	—	—	D9	D8
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

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

Periodic Type TM Operating 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.

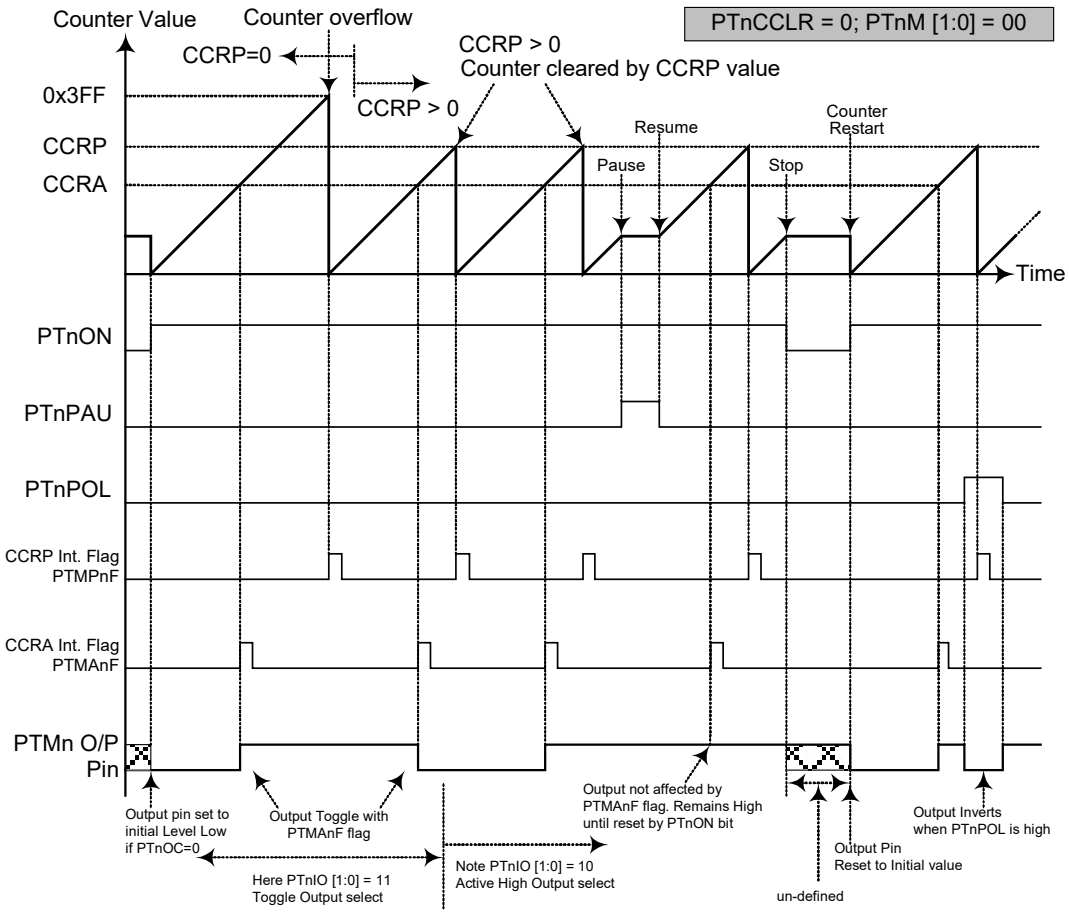
Compare Match Output Mode

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 PTMAnF and PTMPnF 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 PTMAnF 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 PTMPnF interrupt request flag will be generated. In the Compare Match Output Mode, the CCRA can not be cleared to zero.

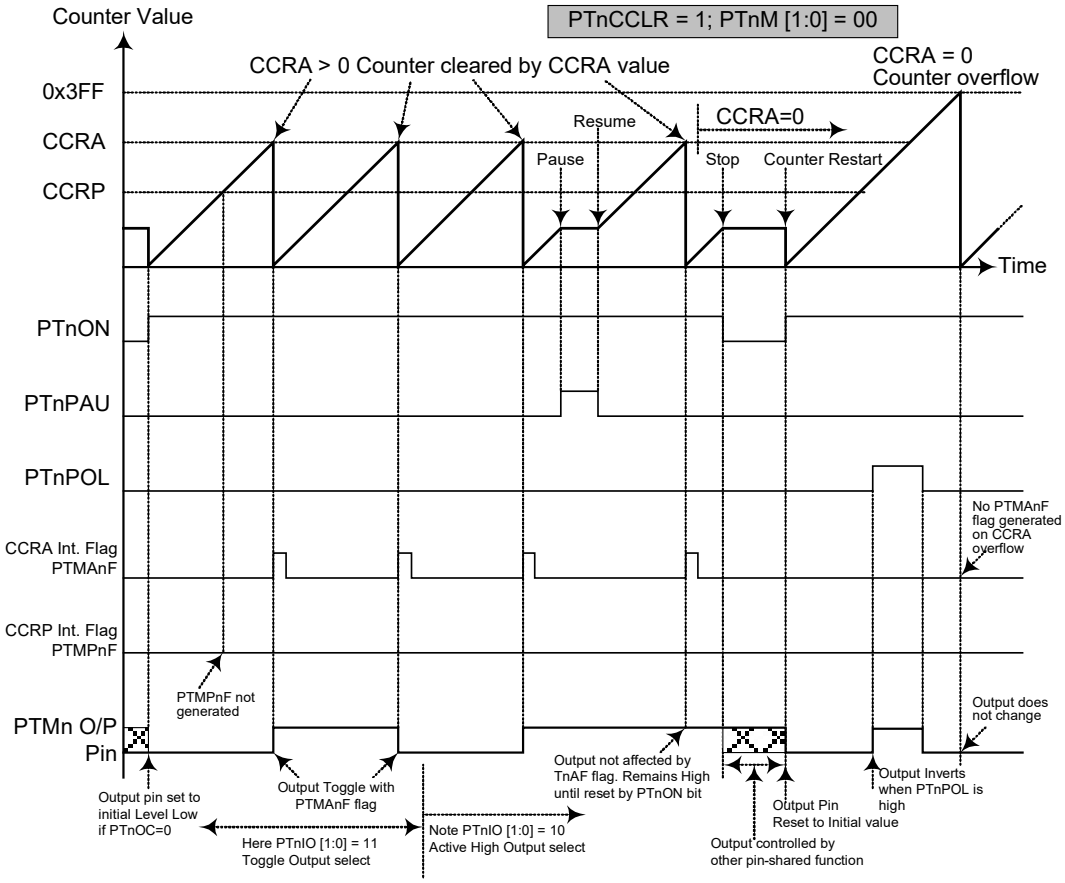
If the CCRA Bits are all zero, the counter will overflow when its reaches its maximum 10-Bit, 3FF Hex, value, however here the PTMAnF interrupt request flag will not be generated.

As the name of the mode suggests, after a comparison is made, the PTM output pin, will change state. The PTM output pin condition however only changes state when a PTMAnF interrupt request flag is generated after a compare match occurs from Comparator A. The PTMPnF interrupt request flag, generated from a compare match occurs from Comparator P, will have no effect on the PTM output pin. The way in which the PTM output pin changes state are determined by the condition of the PTnIO1 and PTnIO0 Bits in the PTMnC1 register. The PTM 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 PTM 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 PTM output pin is controlled only by the PTMAnF flag
3. The output pin is reset to itsinitial state by a PTnON Bit rising edge
4. n=1 or 2



Compare Match Output Mode – PTnCCLR=1

Note: 1. With PTnCCLR=1 a Comparator A match will clear the counter
2. The PTM output pin is controlled only by the PTMAnF flag
3. The output pin is reset to its initial state by a PTnON Bit rising edge
4. A PTMPnF flag is not generated when PTnCCLR=1
5. n=1 or 2

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 PTM output pin is not used. Therefore the above description and Timing Diagrams for the Compare Match Output Mode can be used to understand its function.

PWM Output Mode

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

As both the period and duty cycle of the PWM waveform can be controlled, the choice of generated waveform is extremely flexible. In the PWM Output Mode, the PTnCCLR Bit has no effect on the PWM operation. Both of the CCRA and CCRP registers are used to generate the PWM waveform, one register is used to clear the internal counter and thus control the PWM waveform frequency, while the other one is used to control the duty cycle. 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 PTM output pin to a fixed high or low level. The PTnPOL Bit is used to reverse the polarity of the PWM output waveform.

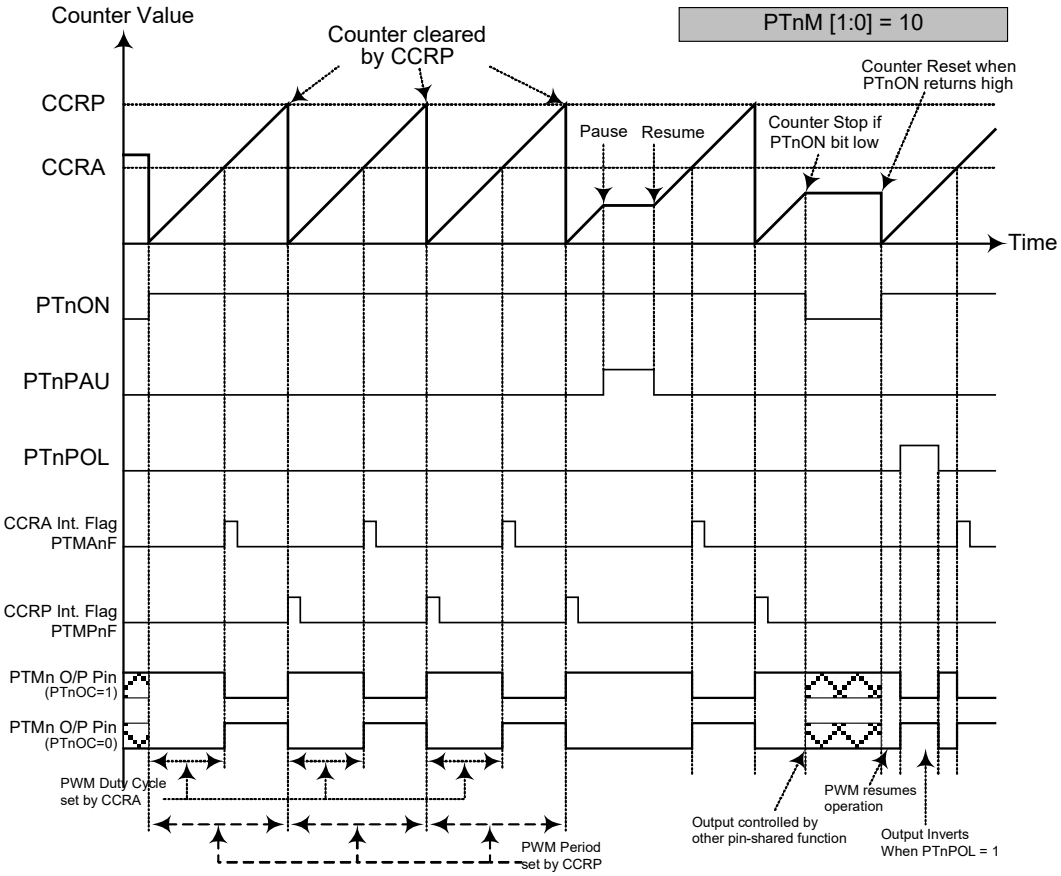
- **10-Bit PTM, PWM Mode, Edge-aligned Mode**

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

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

The PTMn PWM output frequency = (f_{SYS}/4)/512=f_{SYS}/2048=7.8125kHz, duty=128/512=25%,

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



PWM Output Mode

Note: 1. A counter clear sets the PWM Period

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

3. The PTnCCLR Bit has no influence on PWM operation

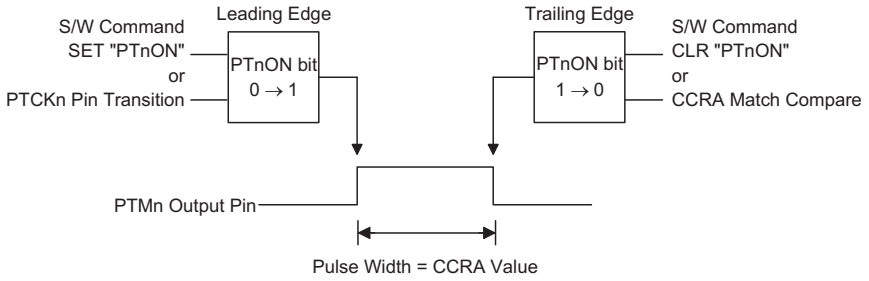
4. n=1 or 2

Single Pulse Mode

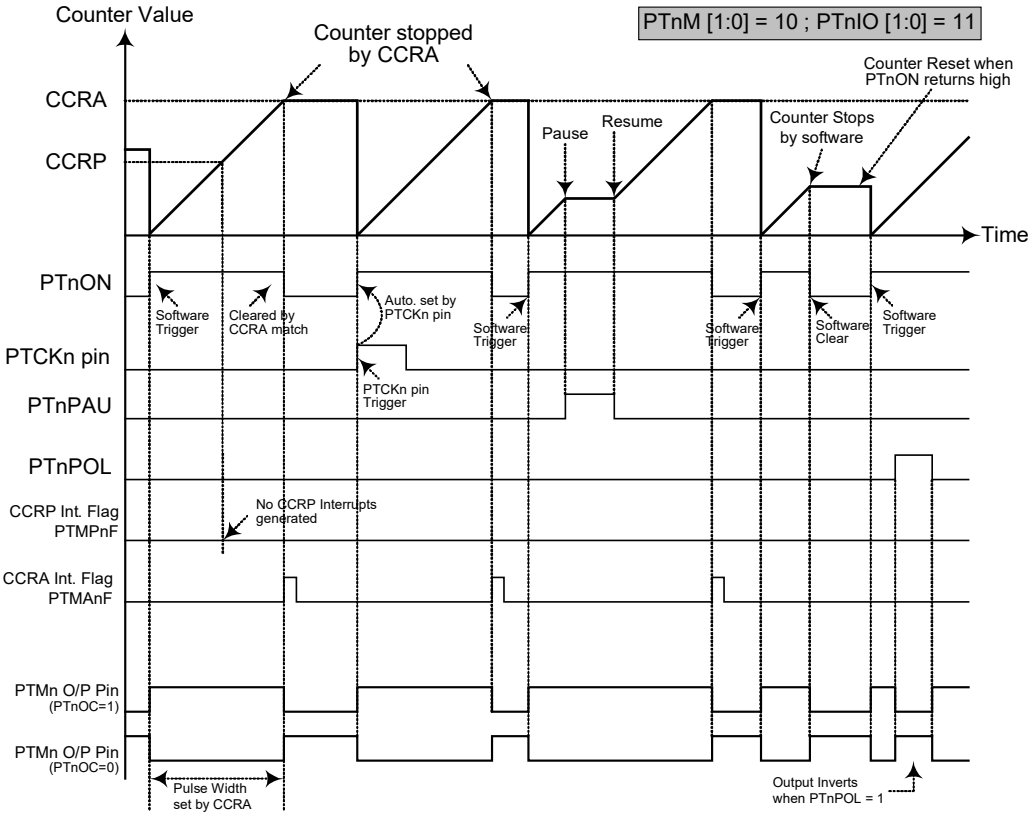
To select this mode, Bits PTnM1 and PTnM0 in the PTMnC1 register should be set to 10 respectively and also the PTnIO1 and PTnIO0 Bits should be set to 11 respectively. The Single Pulse Output Mode, as the name suggests, will generate a single shot pulse on the PTM output pin.

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 PTM 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 Bit is not used in this Mode.



Single Pulse Generation (n= 1 or 2)



Single Pulse Mode

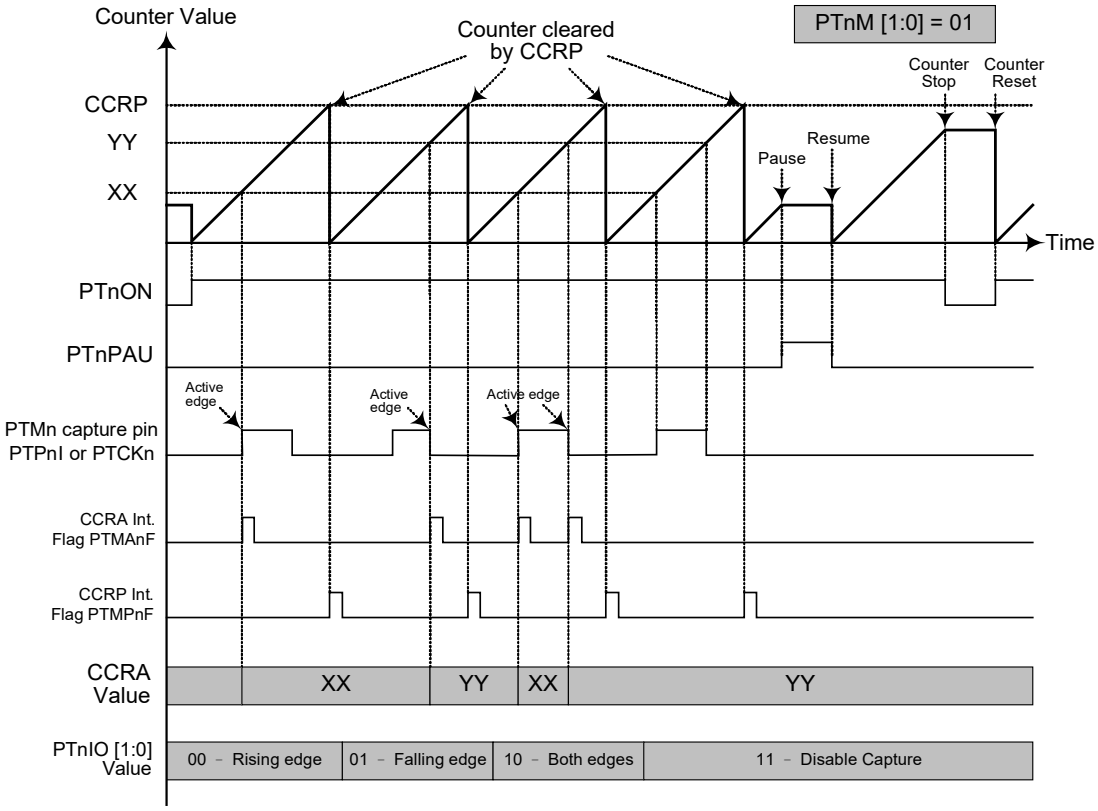
Note: 1. Counter stopped by CCRA
2. CCRP is not used
3. The pulse is triggered by the PTCKn pin or by setting the PTnON Bit high
4. A PTCKn pin active edge will automatically set the PTnON Bit hight
5. In the Single Pulse Mode, PTnIO[1:0] must be set to “11” and can not be changed.
6. n=1 or 2

Capture Input Mode

To select this mode Bits PTnM1 and PTnM0 in the PTMnC1 register should be set to 01 respectively. This mode enables external signals to capture and store the present value of the internal counter and can therefore be used for applications such as pulse width measurements. The external signal is supplied on the PTPnI or PTCKn pin which is selected using the 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 PTM 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 PTM interrupt will also be generated. Counting the number of overflow interrupt signals from the CCRP can be a useful method in measuring long pulse widths. The 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 Capture Input 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 PTM 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=1 or 2

Analog to Digital Converter

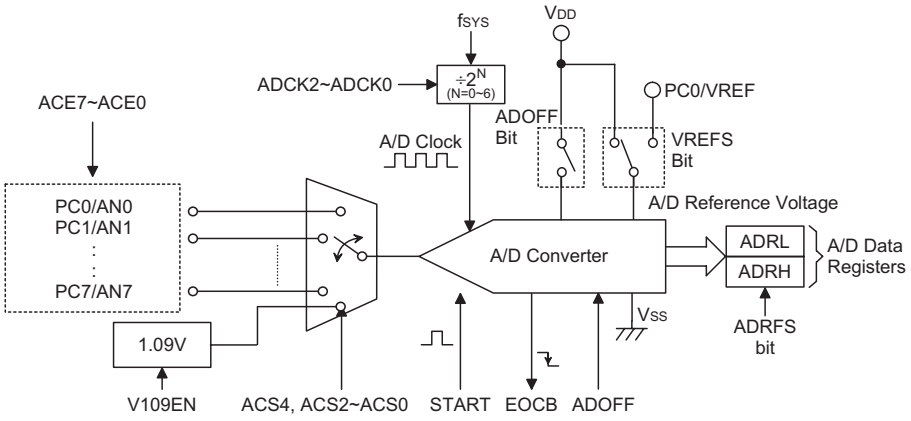
The need to interface to real world analog signals is a common requirement for many electronic systems. However, to properly process these signals by a microcontroller, they must first be converted into digital signals by A/D converters. By integrating the A/D conversion electronic circuitry into the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

The 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.

Device	Input Channels	A/D Channel Select Bits	Input Pins
BS86B12A-3	8	ACS4, ACS2~ACS0	AN0~AN7
BS86C16A-3			
BS86D20A-3			

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



A/D Converter Structure

A/D Converter Register Description

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

Name	Bit							
	7	6	5	4	3	2	1	0
ADRL(ADRFS=0)	D3	D2	D1	D0	—	—	—	—
ADRL(ADRFS=1)	D7	D6	D5	D4	D3	D2	D1	D0
ADRH(ADRFS=0)	D11	D10	D9	D8	D7	D6	D5	D4
ADRH(ADRFS=1)	—	—	—	—	D11	D10	D9	D8
ADCR0	START	EOCB	ADOFF	ADRFS	—	ACS2	ACS1	ACS0
ADCR1	ACS4	V109EN	—	VREFS	—	ADCK2	ADCK1	ADCK0
ACERL	ACE7	ACE6	ACE5	ACE4	ACE3	ACE2	ACE1	ACE0

A/D Converter Register List

A/D Converter Data Registers – ADRL, ADRH

As the devices contain an internal 12-Bit A/D converter, it requires two data registers to store the converted value. These are a high byte register, known as ADRH, and a low byte register, known as ADRL. After the conversion process takes place, these registers can be directly read by the microcontroller to obtain the digitized conversion value. As only 12 Bits of the 16-Bit register space is utilized, the format in which the data is stored is controlled by the ADRFS Bit in the ADCR0 register as shown in the accompanying table. D0~D11 are the A/D conversion result data Bits. Any unused Bits will be read as zero.

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

A/D Data Registers

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

To control the function and operation of the A/D converter, three control registers known as ADCR0, ADCR1, ACERL are provided. These 8-Bit registers define functions such as the selection of which analog channel is connected to the internal A/D converter, the digitized data format, the A/D clock source as well as controlling the start function and monitoring the A/D converter end of conversion status. The ACS2~ACS0 Bits in the ADCR0 register and ACS4 Bit is the ADCR1 register define the A/D Converter input channel number. As the devices contain only one actual analog to digital converter hardware circuit, each of the individual 8 analog inputs must be routed to the converter. It is the function of the ACS4 and ACS2 ~ ACS0 Bits to determine which analog channel input signals or internal 1.09V is actually connected to the internal A/D converter.

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

• ADCR0 Register

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

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

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

Bit 5 **ADOFF** : A/D Converter module power on/off control Bit
0: A/D Converter module power on
1: A/D Converter module power off
This Bit controls the power to the A/D internal function. This Bit should be cleared to zero to enable the A/D converter. If the Bit is set high then the A/D converter will be switched off reducing the device power consumption. As the A/D converter will consume a limited amount of power, even when not executing a conversion, this may be an important consideration in power sensitive battery powered applications.
Note: 1. it is recommended to set ADOFF=1 before entering IDLE/SLEEP Mode for saving power.
2. ADOFF=1 will power down the A/D Converter module.

Bit 4 **ADRFS**: A/D Converter Data Format Control
0: A/D Converter Data MSB is ADRH Bit 7, LSB is ADRL Bit 4
1: A/D Converter Data MSB is ADRH Bit 3, LSB is ADRL Bit 0
This Bit controls the format of the 12-Bit converted A/D value in the two A/D data registers. Details are provided in the A/D data register section.

Bit 3 Unimplemented, read as "0"

Bit 2 ~ 0 **ACS2 ~ ACS0**: Select A/D channel (when ACS4 is "0")
000: AN0
001: AN1
010: AN2
011: AN3
100: AN4
101: AN5
110: AN6
111: AN7
These are the A/D channel select control Bits. As there is only one internal hardware A/D converter each of the eight A/D inputs must be routed to the internal converter using these Bits. If Bit ACS4 in the ADCR1 register is set high then the internal 1.09V will be routed to the A/D Converter.

• ADCR1 Register

Bit	7	6	5	4	3	2	1	0
Name	ACS4	V109EN	—	VREFS	—	ADCK2	ADCK1	ADCK0
R/W	R/W	R/W	—	R/W	—	R/W	R/W	R/W
POR	0	0	—	0	—	0	0	0

Bit 7 **ACS4**: Select Internal 1.09V as A/D Converter input Control
0: Disable
1: Enable
This Bit enables 1.09V to be connected to the A/D converter. The V109EN Bit must first have been set to enable the bandgap circuit 1.09V voltage to be used by the A/D converter. When the ACS4 Bit is set high, the bandgap 1.09V voltage will be routed to the A/D converter and the other A/D input channels disconnected.

Bit 6 **V109EN**: Internal 1.09V Control
0: Disable
1: Enable
This Bit controls the internal bandgap circuit on/off function to the A/D converter. When the Bit is set high the bandgap 1.09V voltage can be used by the A/D converter. If 1.09V is not used by the A/D converter and the LVR function is disabled then the bandgap reference circuit will be automatically switched off to conserve power. When 1.09V is switched on for use by the A/D converter, a time t_{BG} should be allowed for the bandgap circuit to stabilise before implementing an A/D conversion.

Bit 5 Unimplemented, read as "0"

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

Bit 3 Unimplemented, read as "0"

Bit 2 ~ 0 **ADCK2 ~ ADCK0**: Select A/D Converter clock source
000: f_{SYS}
001: $f_{SYS}/2$
010: $f_{SYS}/4$
011: $f_{SYS}/8$
100: $f_{SYS}/16$
101: $f_{SYS}/32$
110: $f_{SYS}/64$
111: Undefined
These three Bits are used to select the clock source for the A/D converter.

- **ACERL Register**

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

Bit 7 **ACE7**: Define PC7 is A/D input or not
0: Not A/D input
1: A/D input, AN7

Bit 6 **ACE6**: Define PC6 is A/D input or not
0: Not A/D input
1: A/D input, AN6

Bit 5 **ACE5**: Define PC5 is A/D input or not
0: Not A/D input
1: A/D input, AN5

Bit 4 **ACE4**: Define PC4 is A/D input or not
0: Not A/D input
1: A/D input, AN4

Bit 3 **ACE3**: Define PC3 is A/D input or not
0: Not A/D input
1: A/D input, AN3

Bit 2 **ACE2**: Define PC2 is A/D input or not
0: Not A/D input
1: A/D input, AN2

Bit 1 **ACE1**: Define PC1 is A/D input or not
0: Not A/D input
1: A/D input, AN1

Bit 0 **ACE0**: Define PC0 is A/D input or not
0: Not A/D input
1: A/D input, AN0

A/D Operation

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

The EOCB Bit in the ADCR0 register is used to indicate when the analog to digital conversion process is complete. This Bit will be automatically set to "0" by the microcontroller after a conversion cycle has ended. In addition, the corresponding A/D interrupt request flag will be set in the interrupt control register, and if the interrupts are enabled, an appropriate internal interrupt signal will be generated. This A/D internal interrupt signal will direct the program flow to the associated A/D internal interrupt address for processing. If the A/D internal interrupt is disabled, the microcontroller can be used to poll the EOCB Bit in the ADCR0 register to check whether it has been cleared as an alternative method of detecting the end of an A/D conversion cycle.

The clock source for the A/D converter, which originates from the system clock f_{SYS}, can be chosen to be either f_{SYS} or a subdivided version of f_{SYS}. The division ratio value is determined by the ADCK2~ADCK0 Bits in the ADCR1 register.

Although the A/D clock source is determined by the system clock f_{SYS}, and by Bits ADCK2~ADCK0, there are some limitations on the maximum A/D clock source speed that can be selected. As the minimum value of permissible A/D clock period, t_{ADCK}, is from 0.5μs to 10μs, care must be taken for system clock frequencies. For example, if the system clock operates at a frequency of 4MHz, the ADCK2~ADCK0 Bits should not be set to 000B or 110B. Doing so will give A/D clock periods that are less than the minimum A/D clock period or greater than the maximum A/D clock period which may result in inaccurate A/D conversion values.

Refer to the following table for examples, where values marked with an asterisk * show where, 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})							
	ADCK2, ADCK1, ADCK0 =000 (f_{SYS})	ADCK2, ADCK1, ADCK0 =001 (f_{SYS}/2)	ADCK2, ADCK1, ADCK0 =010 (f_{SYS}/4)	ADCK2, ADCK1, ADCK0 =011 (f_{SYS}/8)	ADCK2, ADCK1, ADCK0 =100 (f_{SYS}/16)	ADCK2, ADCK1, ADCK0 =101 (f_{SYS}/32)	ADCK2, ADCK1, ADCK0 =110 (f_{SYS}/64)	ADCK2, ADCK1, ADCK0 =111
1MHz	1μs	2μs	4μs	8μs	16μs*	32μs*	64μs*	Undefined
2MHz	500ns	1μs	2μs	4μs	8μs	16μs*	32μs*	Undefined
4MHz	250ns*	500ns	1μs	2μs	4μs	8μs	16μs*	Undefined
8MHz	125ns*	250ns*	500ns	1μs	2μs	4μs	8μs	Undefined
12MHz	83ns*	167ns*	333ns*	667ns	1.33μs	2.67μs	5.33μs	Undefined
16MHz	62.5ns*	125ns*	250ns*	500ns	1μs	2μs	4μs	Undefined

A/D Clock Period Examples

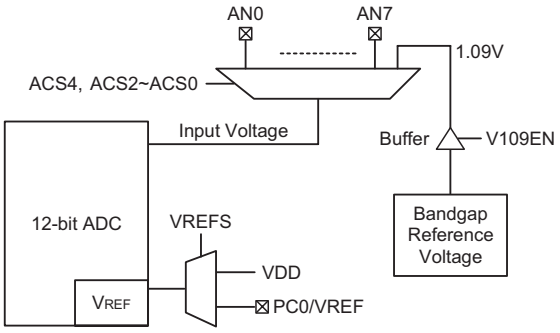
Controlling the power on/off function of the A/D converter circuitry is implemented using the ADOFF Bit in the ADCR0 register. This Bit must be zero to power on the A/D converter. When the ADOFF Bit is cleared to zero to power on the A/D converter internal circuitry a certain delay, as indicated in the timing diagram, must be allowed before an A/D conversion is initiated. Even if no pins are selected for use as A/D inputs by clearing the ACE7~ACE0 Bits in the ACERL register, if the ADOFF Bit is zero then some power will still be consumed. In power conscious applications it is therefore recommended that the ADOFF is set high to reduce power consumption when the A/D converter function is not being used.

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

A/D Input Pins

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

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



A/D Input Structure

Summary of A/D Conversion Steps

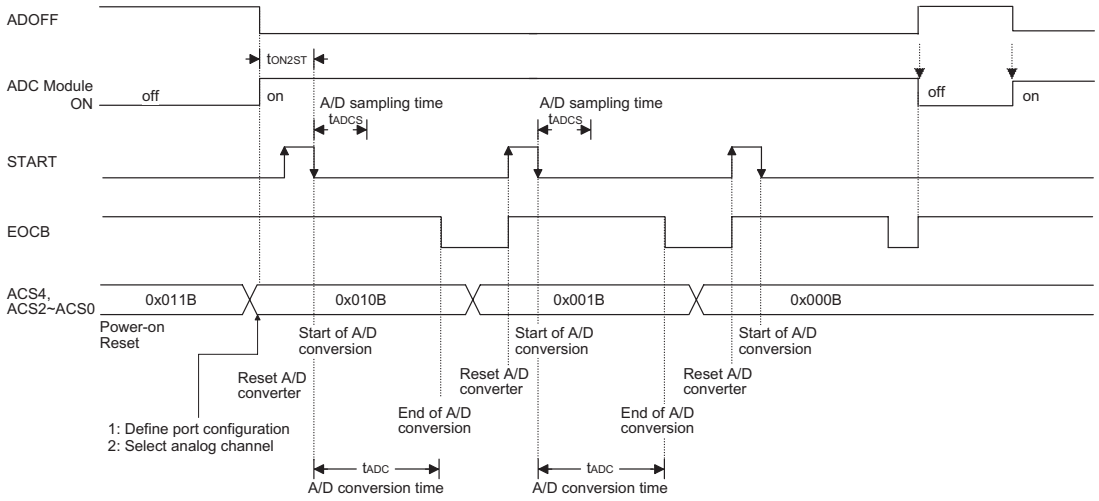
The following summarises the individual steps that should be executed in order to implement an A/D conversion process.

- Step 1
  Select the required A/D conversion clock by correctly programming Bits ADCK2~ADCK0 in the ADCR1 register.
- Step 2
  Enable the A/D by clearing the ADOFF Bit in the ADCR0 register to zero.
- Step 3
  Select which channel is to be connected to the internal A/D converter by correctly programming the ACS4, ACS2~ACS0 Bits which are also contained in the ADCR1 and ADCR0 registers.
- Step 4
  Select which pins are to be used as A/D inputs and configure them by correctly programming the ACE7~ACE0 Bits in the ACERL register.

- Step 5
  If the interrupts are to be used, the interrupt control registers must be correctly configured to ensure the A/D converter interrupt function is active. The master interrupt control Bit, EMI, and the A/D converter interrupt Bit, ADE, must both be set high to do this.
- Step 6
  The analog to digital conversion process can now be initialised by setting the START Bit in the ADCR0 register from low to high and then low again. Note that this Bit should have been originally cleared to zero.
- Step 7
  To check when the analog to digital conversion process is complete, the EOCB Bit in the ADCR0 register can be polled. The conversion process is complete when this Bit goes low. When this occurs the A/D data registers ADRL and ADRH can be read to obtain the conversion value. As an alternative method, if the interrupts are enabled and the stack is not full, the program can wait for an A/D interrupt to occur.

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

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



A/D Conversion Timing

Programming Considerations

During microcontroller operations where the A/D converter is not being used, the A/D internal circuitry can be switched off to reduce power consumption, by setting Bit ADOFF high in the ADCR0 register. When this happens, the internal A/D converter circuits will not consume power irrespective of what analog voltage is applied to their input lines. If the A/D converter input lines are used as normal I/Os, then care must be taken as if the input voltage is not at a valid logic level, then this may lead to some increase in power consumption.

A/D Transfer Function

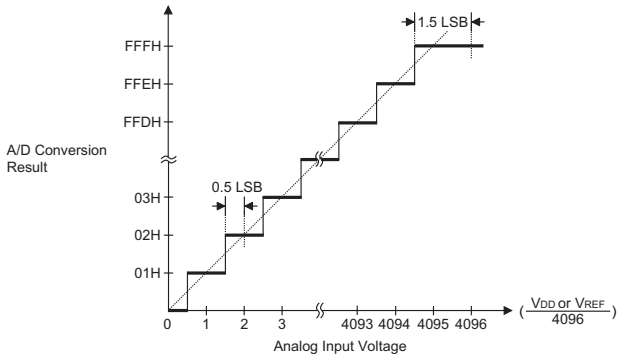
As the device contains a 12-Bit A/D converter, its full-scale converted digitized value is equal to FFFH. Since the full-scale analog input value is equal to the V_{DD} or V_{REF} voltage, this gives a single Bit analog input value of V_{DD} or V_{REF} divided by 4096.

$$1\ LSB = (V_{DD} \text{ or } V_{REF}) / 4096$$

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

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

The diagram shows the ideal transfer function between the analog input value and the digitised output value for the A/D converter. Except for the digiti ├ ed 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 digitized value will change at a point 1.5 LSB below the V_{DD} or V_{REF} level.



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 EOCB Bit in the ADCR0 register is used to detect when the conversion cycle is complete, whereas in the second example, the A/D interrupt is used to determine when the conversion is complete.

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

```
clr     ADE             ; disable A/D Converter interrupt
mov     a,03H
mov     ADCR1,a         ; select fSYS/8 as A/D clock and switch off 1.09V
clr     ADOFF
mov     a,0Fh           ; setup ACERL to configure pins AN0~AN3
mov     ACERL,a
mov     a,01h
mov     ADCR0,a         ; enable and connect AN0 channel to A/D converter
:
start_conversion:
clr     START           ; high pulse on start Bit to initiate conversion
set     START           ; reset A/D
clr     START           ; start A/D
polling_EOC:
sz      EOCB            ; poll the ADCR0 register EOCB Bit to detect end of A/D conversion
jmp     polling_EOC     ; continue polling
mov     a,ADRL          ; read low byte conversion result value
mov     ADRL_buffer,a   ; save result to user defined register
mov     a,ADRH          ; read high byte conversion result value
```

```
mov     ADRH_buffer,a     ; save result to user defined register
:
:
jmp     start_conversion  ; start next A/D conversion
```

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

```
clr     ADE               ; disable A/D Converter interrupt
mov     a,03H
mov     ADCR1,a           ; select fSYS/8 as A/D clock and switch off 1.09V
clr     ADOFF
mov     a,0Fh             ; setup ACERL to configure pins AN0~AN3
mov     ACERL,a
mov     a,01h
mov     ADCR0,a           ; enable and connect AN0 channel to A/D converter
Start_conversion:
clr     START             ; high pulse on START Bit to initiate conversion
set     START             ; reset A/D
clr     START             ; start A/D
clr     ADF               ; clear A/D Converter interrupt request flag
set     ADE               ; enable A/D Converter interrupt
set     EMI               ; enable global interrupt
:
:
; A/D Converter interrupt service routine
ADC_ISR:
mov     acc_stack,a       ; save ACC to user defined memory
mov     a,STATUS
mov     status_stack,a    ; save STATUS to user defined memory
:
:
mov     a,ADRL            ; read low byte conversion result value
mov     adrl_buffer,a     ; save result to user defined register
mov     a,ADRH            ; read high byte conversion result value
mov     adrh_buffer,a     ; save result to user defined register
:
:
EXIT_INT_ISR:
mov     a,status_stack
mov     STATUS,a          ; restore STATUS from user defined memory
mov     a,acc_stack       ; restore ACC from user defined memory
reti
```

Touch Key Function

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

Touch Key Structure

The touch keys are pin shared with the PA ~ PD logic I/O pins, with the desired function chosen via 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	Keys - n	Touch Key Module	Touch Key	Shared I/O Pin
BS86B12A-3	12	M0	Key1~Key4	PB0~PB3
		M1	Key5~Key8	PB4~PB7
		M2	Key9~Key12	PC0~PC3
BS86C16A-3	16	M0	Key1~Key4	PB0~PB3
		M1	Key5~Key8	PB4~PB7
		M2	Key9~Key12	PC0~PC3
		M3	Key13~Key16	PC4~PC7
BS86D20A-3	20	M0	Key1~Key4	PB0~PB3
		M1	Key5~Key8	PB4~PB7
		M2	Key9~Key12	PD3, PD2, PC0, PC1
		M3	Key13~Key16	PC2~PC5
		M4	Key17~Key20	PC6, PC7, PA4, PA1

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 BS86B12A-3 has a range of M0 to M2, the BS86C16A-3 has a range of M0 to M3, the BS86D20A-3 has a range of M0 to M4.

Name	Usage
TKTMR	Touch Key 8-Bit timer/counter register
TKC0	Counter on-off and clear control/reference clock control/Start Bit
TK16DL	Touch key module 16-Bit counter low byte contents
TK16DH	Touch key module 16-Bit counter high byte contents
TKC1	Touch key OSC frequency select
TKMn16DL	Module n 16-Bit counter low byte contents
TKMn16DH	Module n 16-Bit counter high byte contents
TKMnROL	Reference OSC internal capacitor select
TKMnROH	Reference OSC internal capacitor select
TKMnC0	Control Register 0 Multiplexer Key Select
TKMnC1	Control Register 1 Key oscillator control/Reference oscillator control/ Touch key or I/O select

Register Listing (n=0~4)

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 Module (n=0~4)

- **TKTMR Register**

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

Bit 7~0 Touch Key 8-Bit timer/counter register
Time slot counter overflow set-up time is (256-TKTMR[7:0])×32

- **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/W
POR	—	0	0	0	0	0	0	0

Bit 7 Unimplemented, read as "0"

Bit 6 **TKRCOV**: Time slot counter overflow flag
0: No overflow
1: Overflow
If module 0 or all module time slot counter, selected by the TSCS Bit, is overflow, the Touch Key Interrupt request flag, TKMF, will be set and all module key OSCs and ref OSCs auto stop. All module 16-Bit C/F counter, 16-Bit counter, 5-Bit time slot counter and 8-Bit time slot timer counter will be automatically switched off.

Bit 5 **TKST**: Start Touch Key detection control Bit
0: Stopped
0->1: Started
In all modules the16-Bit C/F counter, 16-Bit counter, 5-Bit time slot counter will be automatically cleared when this Bit is cleared to zero (8-Bit programmable time slot counter will not be cleared, which overflow time is setup by user). When this Bit changes from low to high, the 16-Bit C/F counter, 16-Bit counter, 5-Bit time slot counter and 8-Bit time slot timer counter will be automatically on and enable key OSC and ref OSC output clock input to these counters.

Bit 4 **TKCFOV**: Touch key module 16-Bit C/F counter overflow flag
0: Not overflow
1: Overflow
When the touch key module 16-bit C/F counter overflows, this bit will be set to 1. As this flag will not be automatically cleared, it has to be cleared by the application program.

Bit 3 **TK16OV**: Touch key module 16-Bit counter overflow flag
0: Not overflow
1: Overflow
When the touch key module 16-bit counter overflows, this bit will be set to 1. As this flag will not be automatically cleared, it has to be cleared by the application program.

Bit 2 **TSCS**: Touch Key time slot counter select
0: Each Module uses its own time slot counter.
1: All Touch Key Module use Module 0 time slot counter.

Bit 1~0 **TK16S1~ TK16S0**: The touch key module 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 OSC frequency select
00: 500kHz
01: 1000 kHz
10: 1500 kHz
11: 2000 kHz

• TK16DL 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 Touch key module 16-Bit counter low byte contents

• TK16DH Register

Bit	7	6	5	4	3	2	1	0
Name	D15	D14	D13	D12	D11	D10	D9	D8
R/W	R	R	R	R	R	R	R	R
POR	0	0	0	0	0	0	0	0

Bit 7~0 Touch key module 16-Bit counter high byte contents

• TKMn16DL Register

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

Bit 7~0 Module n 16-Bit counter low byte contents

• **TKMn16DH Register**

Bit	7	6	5	4	3	2	1	0
Name	D15	D14	D13	D12	D11	D10	D9	D8
R/W	R	R	R	R	R	R	R	R
POR	0	0	0	0	0	0	0	0

Bit 7~0 Module n 16-Bit counter high byte contents

• **TKMnROL 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 Reference OSC inernal capacitor select
OSC inernal capacitor select : (TKMnRO[9:0] × 50pF) / 1024

• **TKMnROH 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 Reference OSC inernal capacitor select
OSC inernal capacitor select: (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		Module Number				
MnMXS1	MnMXS0	M0	M1	M2	M3	M4
0	0	Key 1	Key 5	Key 9	Key 13	Key 17
0	1	Key 2	Key 6	Key 10	Key 14	Key 18
1	0	Key 3	Key 7	Key 11	Key 15	Key 19
1	1	Key 4	Key 8	Key 12	Key 16	Key 20

Bit 5 **MnDFEN**: Multi-frequency control
0: Disable
1: Enable

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**: C to F OSC frequency hopping function control
0: The frequency hopping function is controlled by MnSOF2 ~ MnSOF0 Bits
1: The frequency hopping function is controlled by hardware regardless of what is the state of MnSOF2~ MnSOF0 Bits
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

• **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**: Time slot counter clock select
0: Reference oscillator
1: $f_{SYS}/4$

Bit 6 Unimplemented, read as "0"

Bit 5 **MnROEN**: Reference OSC control
0: Disable
1: Enable

Bit 4 **MnKOEN**: Key OSC control
0: Disable
1: Enable

Bit 3~0 **MnK4IO~ MnK1IO**: I/O pin or touch key function select

MnK4IO	M0	M1	M2	M3	M4
	PB3/Key 4	PB7/Key 8	PC3/Key 12 or PC1/Key 12	PC7/Key 16 or PC5/Key 16	PA1/Key 20
0	I/O				
1	Touch key				

MnK3IO	M0	M1	M2	M3	M4
	PB2/Key 3	PB6/Key 7	PC2/Key 11 or PC0/Key 11	PC6/Key15 or PC4/Key 15	PA4/Key 19
0	I/O				
1	Touch key				

MnK2IO	M0	M1	M2	M3	M4
	PB1/Key 2	PB5/Key 6	PC1/Key 10 or PD2/Key 10	PC5/Key 14 or PC3/Key 14	PC7/Key 18
0	I/O				
1	Touch key				

MnK1IO	M0	M1	M2	M3	M4
	PB0/Key 1	PB4/Key 5	PC0/Key 9 or PD3/Key 9	PC4/Key 13 or PC2/Key 13	PC6/Key 17
0	I/O				
1	Touch key input				

Touch Key Operation

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

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

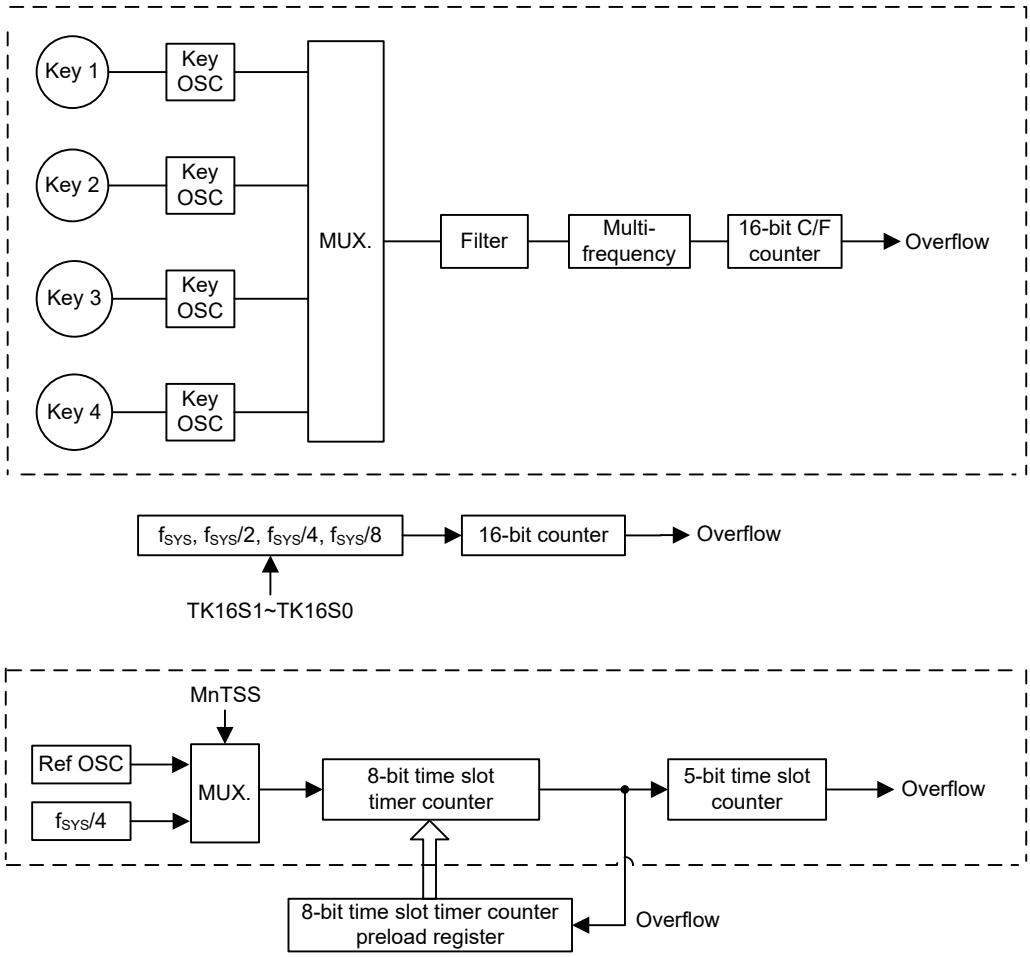
During this reference clock fixed interval, the number of clock cycles generated by the sense oscillator is measured, and it is this value that is used to determine if a touch action has been made or not. At the end of the fixed reference clock time interval a Touch Key interrupt signal will be generated.

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. The16-Bit C/F counter, 16-Bit counter, 5-Bit time slot counter in all modules will be automatically cleared when this 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 this Bit changes from low to high, the 16-Bit C/F counter, 16-Bit counter, 5-Bit time slot counter and 8-Bit time slot timer counter will be automatically switched on.

The key oscillator and reference oscillator in all modules will be automatically stopped and the 16-Bit C/F counter, 16-Bit counter, 5-Bit time slot counter and 8-Bit time slot timer counter will be automatically switched off when the 5-Bit time slot counter overflows. The clock source for the time slot counter and 8+5 Bit counter, is sourced from the reference oscillator or $f_{SYS}/4$. The reference oscillator and key oscillator will be enabled by setting the MnROEN Bit and MnKOEN Bits in the TKMnC1 register.

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

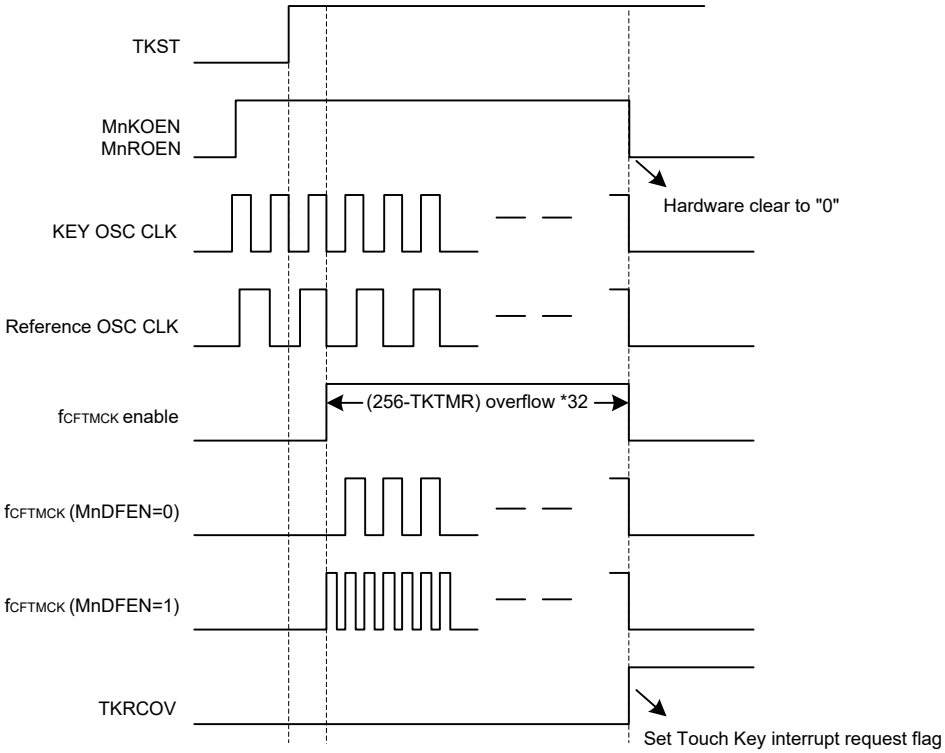
Each touch key module consists of four touch keys, Key1 ~ Key4 are contained in module 0, Key5 ~ Key8 are contained in module 1, Key9 ~ Key12 are contained in module 2, Key13 ~ Key16 are contained in the module 3 and Key17 ~ Key20 are contained in the module 4. Each touch key module has an identical structure.

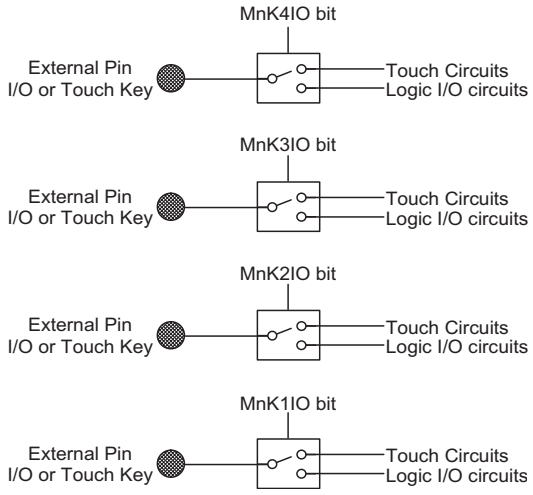


Note: 1. Each touch key module contains the content in the red dash line.
2. The content in the black dash line is the module number (0~n). Each module contains 4 touch keys.

Touch Key Module Block Diagram

The touch key sense oscilltor and reference oscillator timing diagram is shown in the following figure:





Touch Key or I/O Function Select

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 counter and 8-Bit time slot counter in all modules will be automatically cleared.

Programming Considerations

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

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

Serial Interface Module – SIM

The devices contain a Serial Interface Module, which includes both the four-line SPI interface and the two-line I^2C interface types, to allow an easy method of communication with external peripheral hardware. Having relatively simple communication protocols, these serial interface types allow the microcontroller to interface to external SPI or I^2C based hardware such as sensors, Flash or EEPROM memory, etc. The SIM interface pins are pin-shared with other I/O pins therefore the SIM interface function must first be selected by setting the SIM enable/disable Bit. 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 SIM operating mode 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 shall not enter the device to HALT status by application program during processing SIM communication.

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

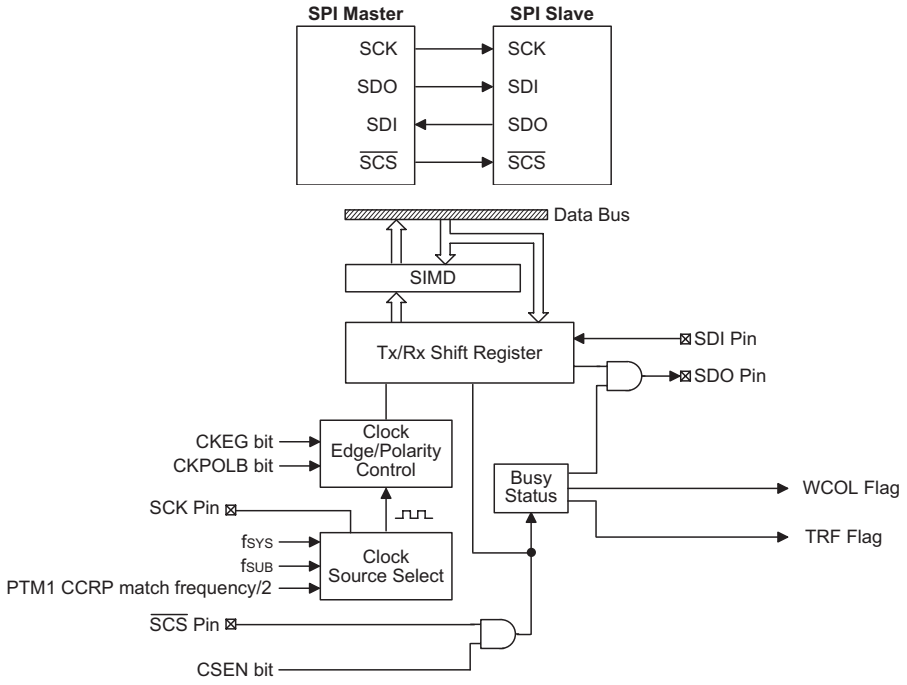
SPI Interface Operation

The SPI interface is a full duplex synchronous serial data link. It is a four line interface with pin names SDI, SDO, SCK and $\overline{SCS}$. Pins SDI and SDO are the Serial Data Input and Serial Data Output lines, SCK is the Serial Clock line and $\overline{SCS}$ is the Slave Select line. As the SPI interface pins are pin-shared with normal I/O pins and with the I^2C function pins, the SPI interface must first be enabled by setting the correct Bits in the SIMC0 and SIMC2 registers. 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 high to enable the $\overline{SCS}$ pin function, clear the CSEN Bit to zero, the $\overline{SCS}$ pin will be in floating state.

The SPI function in these devices offers the following features:

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

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



SPI Block Diagram

SPI Registers

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

Register Name	Bit							
	7	6	5	4	3	2	1	0
SIMC0	SIM2	SIM1	SIM0	—	SIMDBNC1	SIMDBNC0	SIMEN	SIMICF
SIMC2	D7	D6	CKPOLB	CKEG	MLS	CSEN	WCOL	TRF
SIMD	D7	D6	D5	D4	D3	D2	D1	D0

SPI Registers List

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

• **SIMD Register**

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

"x" unknow

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	—	SIMDBNC1	SIMDBNC0	SIMEN	SIMICF
R/W	R/W	R/W	R/W	—	R/W	R/W	R/W	R/W
FOR	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 PTM1 CCRP compare match frequency/2
101: SPI slave mode
110: I²C slave mode
111: Unused mode

These Bits setup the overall operating mode of the SIM function. As well as selecting if the I²C or SPI function, they are used to control the SPI Master/Slave selection and the SPI Master clock frequency. The SPI clock is a function of the system clock but can also be chosen to be sourced from f_{SUB} or the PTM1. 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 **SIMDBNC1~SIMDBNC0**: I²C Debounce Time Selection
Described in I²C registers section.

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

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

Bit 0 **SIMICF**: SPI Incompleted Flag
0: No SPI incomplete transfer occurs
1: SPI incomplete transfer 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 high but the $\overline{SCS}$ pin is is pulled high by the external master device before the SPI data transfer is completely finished, the SIMICF Bit will be set to "1" by hareware 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 high if the SIMICF Bit is set high 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
FOR	0	0	0	0	0	0	0	0

Bit 7~6 Undefined Bit

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

Bit 5 **CKPOLB**: 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

1: MSB

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

Bit 2 **CSEN**: SPI $\overline{SCS}$ pin Control

0: Disable

1: Enable

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

Bit 1 **WCOL**: SPI Write Collision flag

0: No collision

1: Collision

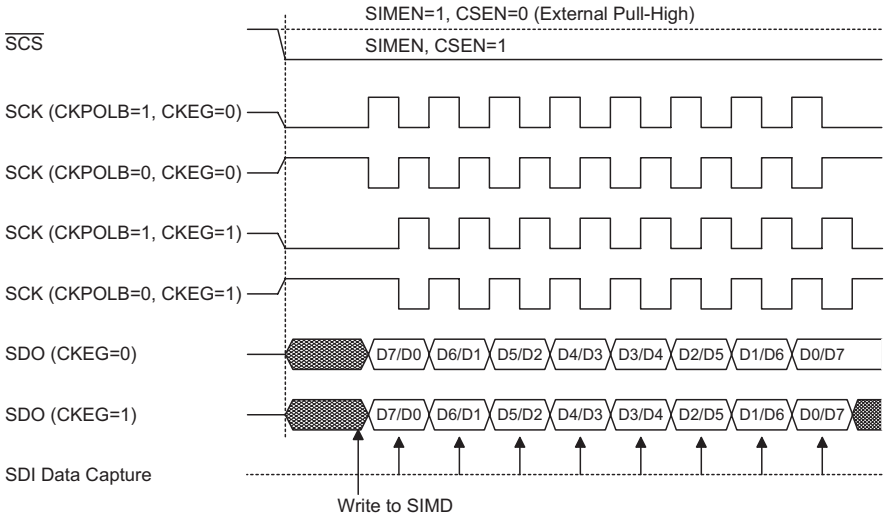
The WCOL flag is used to detect if a data collision has occurred. If this Bit is high it means that data has been attempted to be written to the SIMD register during a data transfer operation. This writing operation will be ignored if data is being transferred. The Bit can be cleared by the application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

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

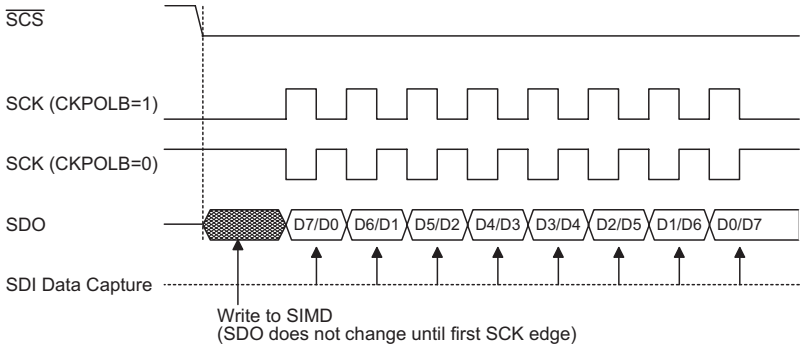
SPI Communication

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

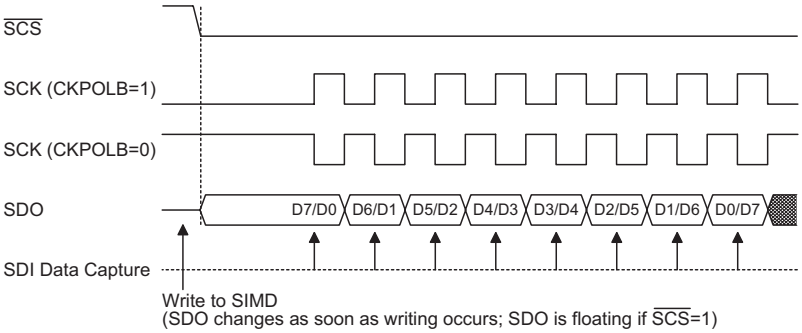
The SPI will continue to function in specific IDLE Modes if the clock source used by the SPI interface is still active.



SPI Master Mode Timing

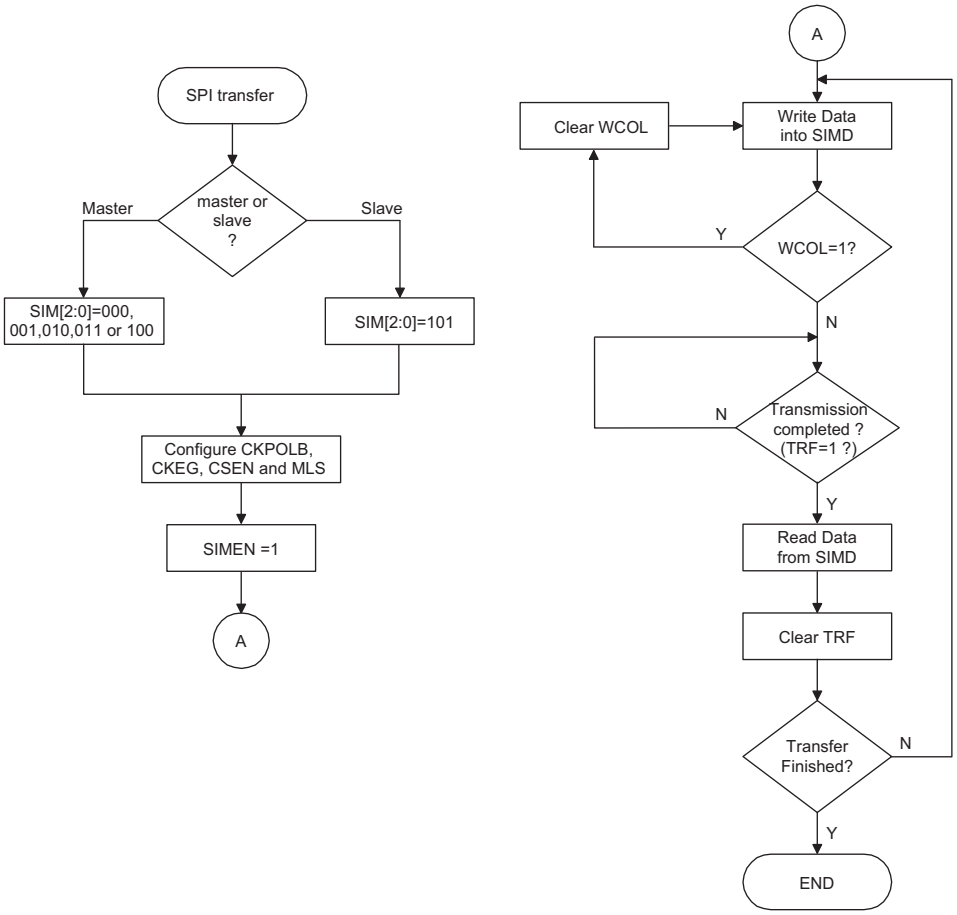


SPI Slave Mode Timing – CKEG=0



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

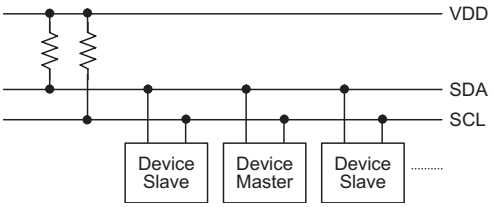
SPI Slave Mode Timing – CKEG=1



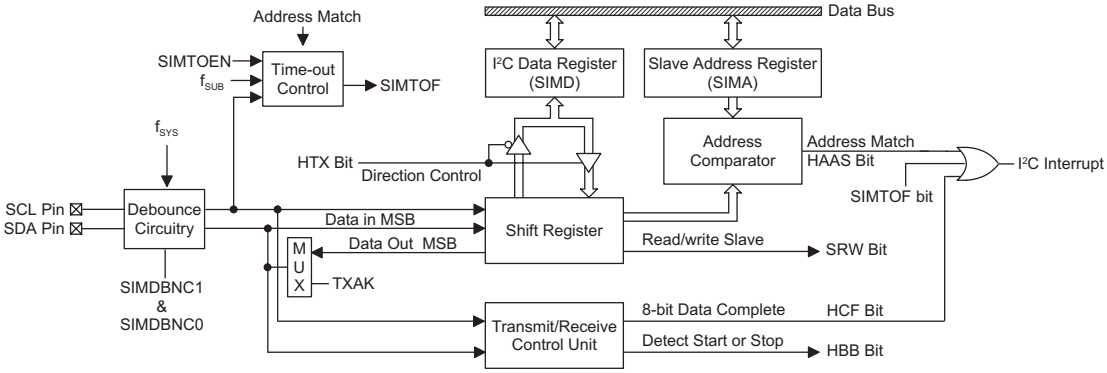
SPI Transfer Control Flowchart

I²C Interface

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



I²C Master/Slave Bus Connection

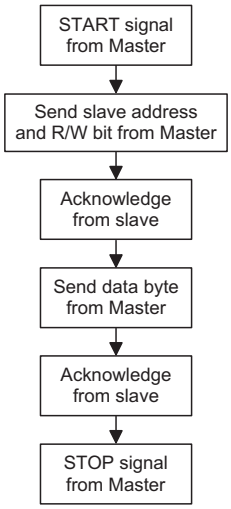


I²C Block Diagram

I²C Interface Operation

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

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



The SIMDBNC1 and SIMDBNC0 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 Debounce	f_{SYS} > 2MHz	f_{SYS} > 5MHz
2 system clock debounce	f_{SYS} > 4MHz	f_{SYS} > 10MHz
4 system clock debounce	f_{SYS} > 8MHz	f_{SYS} > 20MHz

I²C Minimum f_{SYS} Frequency Requirements

I²C Registers

There are four control registers associated with the I²C bus, SIMC0, SIMC1, SIMTOC and SIMA and one data register, SIMD. The SIMD register, which is shown in the above SPI section, is used to store the data being transmitted and received on the I²C bus. Before the microcontroller 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 microcontroller can read it from the SIMD register. Any transmission or reception of data from the I²C bus must be made via the SIMD register. Note that the SIMA register also has the name SIMC2 which is used by the SPI function. The SIMEN Bit, SIM2~SIM0 Bits and SIMDBNC1~SIMDBNC0 Bits in register SIMC0 are used by the I²C interface. The SIMTOC register is used for the I²C time-out control.

Register Name	Bit							
	7	6	5	4	3	2	1	0
SIMC0	SIM2	SIM1	SIM0	—	SIMDBNC1	SIMDBNC0	SIMEN	SIMICF
SIMC1	HCF	HAAS	HBB	HTX	TXAK	SRW	IAMWU	RXAK
SIMD	D7	D6	D5	D4	D3	D2	D1	D0
SIMA	A6	A5	A4	A3	A2	A1	A0	—
SIMTOC	SIMTOEN	SIMTOF	SIMTOS5	SIMTOS4	SIMTOS3	SIMTOS2	SIMTOS1	SIMTOS0

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

• **SIMD Register**

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

"x" unknown

The SIMA register is also used by the SPI interface but has the name SIMC2. The SIMA register is the location where the 7-Bit slave address of the slave device is stored. Bits 7~1 of the SIMA register define the device slave address. Bit 0 is not defined.

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

• **SIMA Register**

Bit	7	6	5	4	3	2	1	0
Name	A6	A5	A4	A3	A2	A1	A0	—
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	—
FOR	0	0	0	0	0	0	0	—

Bit 7~1 **A6~A0**: I²C slave address
A6~A0 is I²C slave address Bit 7~ Bit 1.

Bit 0 Unimplemented, read as "0"

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

• **SIMC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	SIM2	SIM1	SIM0	—	SIMDBNC1	SIMDBNC0	SIMEN	SIMICF
R/W	R/W	R/W	R/W	—	R/W	R/W	R/W	R/W
FOR	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 PTM1 CCRP compare match frequency/2
101: SPI slave mode
110: I²C slave mode
111: Unused mode

These Bits setup the overall operating mode of the SIM function. As well as selecting if the I²C or SPI function, they are used to control the SPI Master/Slave selection and the SPI Master clock frequency. The SPI clock is a function of the system clock but can also be chosen to be sourced from f_{SUB} or the PTM1. 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 **SIMDBNC1~SIMDBNC0**: I²C Debounce Time Selection
00: No debounce
01: 2 system clock debounce
10: 4 system clock debounce
11: 4 system clock debounce

Bit 1 **SIMEN**: SIM Control
0: Disable
1: Enable
The Bit is the overall on/off control for the SIM interface. When the SIMEN Bit is cleared to zero to disable the SIM interface, the SDI, SDO, SCK and $\overline{SCS}$, or SDA and SCL lines will lose their SPI or I²C function and the SIM operating current will be reduced to a minimum value. When the Bit is high the SIM interface is enabled. If the SIM is configured to operate as an SPI interface via the SIM2~SIM0 Bits, the contents of the SPI control registers will remain at the previous settings when the SIMEN Bit changes from low to high and should therefore be first initialised by the application program. If the SIM is configured to operate as an I²C interface via the SIM2~SIM0 Bits and the SIMEN Bit changes from low to high, the contents of the I²C control Bits such as HTX and TXAK will remain at the previous settings and should therefore be first initialised by the application program while the relevant I²C flags such as HCF, HAAS, HBB, SRW and RXAK will be set to their default states.

Bit 0 **SIMICF**: SPI Incompleted Flag
Described in the SPI register section.

• SIMC1 Register

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

Bit 7 **HCF**: I²C Bus data transfer completion flag
0: Data is being transferred
1: Completion of an 8-Bit data transfer
The HCF flag is the data transfer flag. This flag will be zero when data is being transferred. Upon completion of an 8-Bit data transfer the flag will go high and an interrupt will be generated.

Bit 6 **HAAS**: I²C Bus address match flag
0: Not address match
1: Address match
The HAAS flag is the address match flag. This flag is used to determine if the slave device address is the same as the master transmit address. If the addresses match then this Bit will be high, if there is no match then the flag will be low.

Bit 5 **HBB**: I²C Bus busy flag
0: I²C Bus is not busy
1: I²C Bus is busy
The HBB flag is the I²C busy flag. This flag will be "1" when the I²C bus is busy which will occur when a START signal is detected. The flag will be set to "0" when the bus is free which will occur when a STOP signal is detected.

Bit 4 **HTX**: I²C slave device transmitter/receiver selection
0: Slave device is the receiver
1: Slave device is the transmitter

Bit 3 **TXAK**: I²C Bus transmit acknowledge flag
0: Slave send acknowledge flag
1: Slave do not send acknowledge flag

The TXAK Bit is the transmit acknowledge flag. After the slave device receipt of 8-Bit of data, this Bit will be transmitted to the bus on the 9th clock from the slave device. The slave device must always clear the TXAK Bit to zero 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 application program after wake-up to ensure correction device operation.

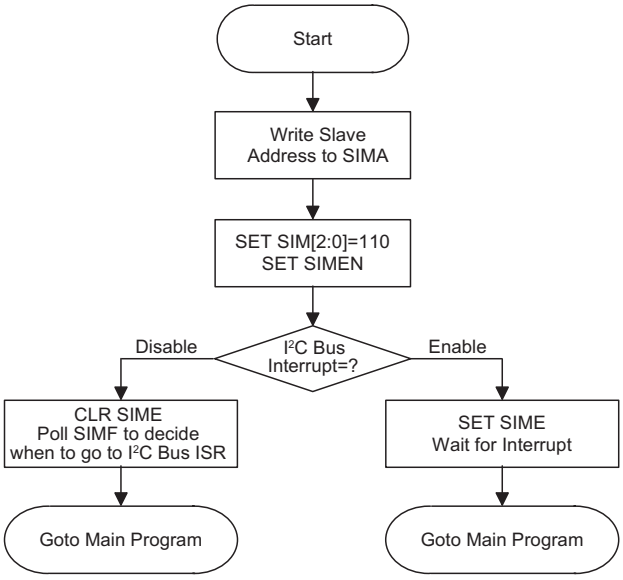
Bit 0 **RXAK**: I²C Bus Receive acknowledge flag
0: Slave receives acknowledge flag
1: Slave do not receive acknowledge flag

The RXAK flag is the receiver acknowledge flag. When the RXAK flag is "0", it means that a acknowledge signal has been received at the 9th clock, after 8 Bits of data have been transmitted. When the slave device in the transmit mode, the slave device checks the RXAK flag to determine if the master receiver wishes to receive the next byte. The slave transmitter will therefore continue sending out data until the RXAK flag is "1". When this occurs, the slave transmitter will release the SDA line to allow the master to send a STOP signal to release the 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 Bit and SIMTOF Bit to determine whether the interrupt source originates from an address match or I²C communication time-out or from the completion of an 8-Bit data transfer. During a data transfer, note that after the 7-Bit slave address has been transmitted, the following Bit, which is the 8th Bit, is the read/write Bit whose value will be placed in the SRW Bit. This Bit will be checked by the slave device to determine whether to go into transmit or receive mode. Before any transfer of data to or from the I²C bus, the microcontroller must initialize the bus, the following are steps to achieve this:

- Step 1
  Set the SIM2~SIM0 Bits and SIMEN Bit in the SIMC0 register to "110" and "1" repectively 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 related interrupt enable Bit of the interrupt control register to enable the SIM interrupt.



I²C Bus Initialisation Flow Chart

I²C Bus Start Signal

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

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 Bit and SIMTOF Bit should be examined to see whether the interrupt source has come from a matching slave address or I²C communication time-out or from the completion of a data byte transfer. When a slave address is matched, the device must be placed in either the transmit mode and then write data to the SIMD register, or in the receive mode where it must implement a dummy read from the SIMD register to release the SCL line.

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.

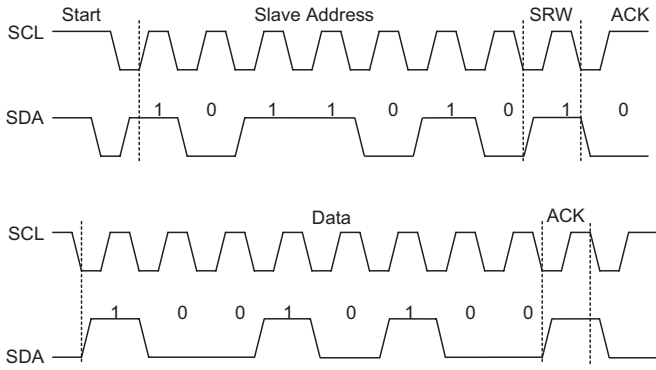
I²C Bus Slave Address Acknowledge Signal

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 cleared to "0".

I²C Bus Data and Acknowledge Signal

The transmitted data is 8-Bit wide and is transmitted after the slave device has acknowledged receipt of its slave address. The order of serial Bit transmission is the MSB first and the LSB last. After receipt of 8-Bit 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.

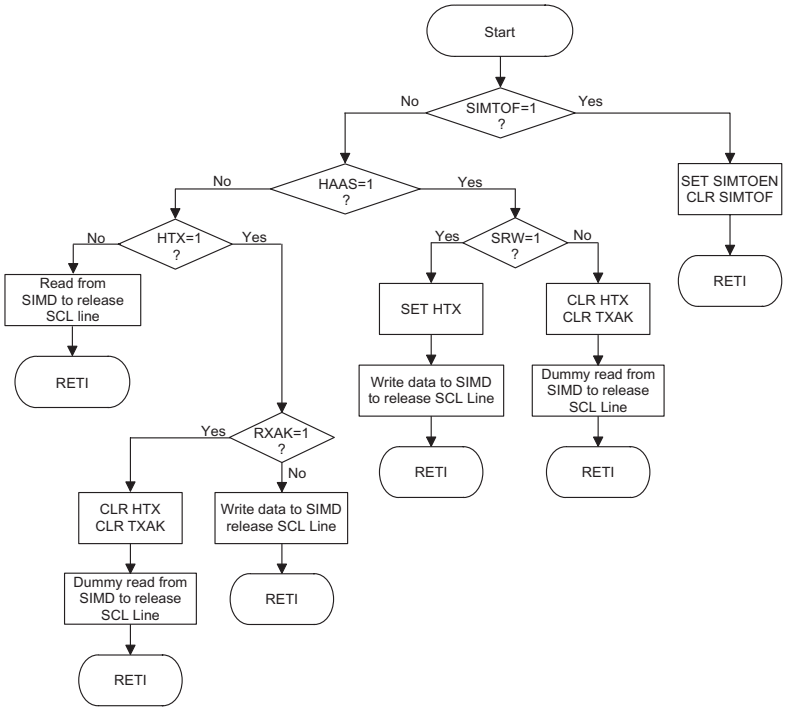


S=Start (1 bit)
SA=Slave Address (7 bits)
SR=SRW bit (1 bit)
M=Slave device send acknowledge bit (1 bit)
D=Data (8 bits)
A=ACK (RXAK bit for transmitter, TXAK bit for receiver 1 bit)
P=Stop (1 bit)

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

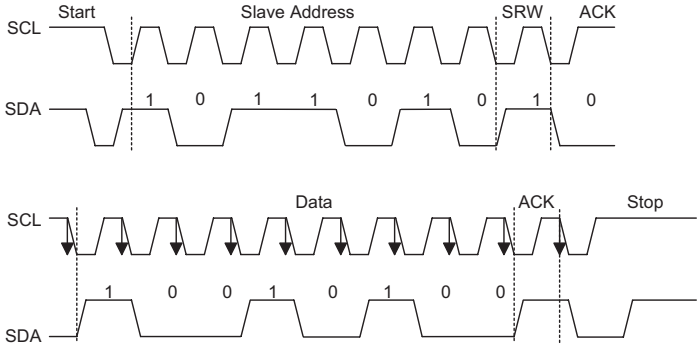
I²C Communication Timing Diagram



I²C Bus ISR Flow Chart

I²C Interface Time-out Function

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



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)

I²C Time-out

The time-out counter starts counting on an I²C bus "START" and "address match" condition, and is cleared by an SCL falling edge. Before the next SCL falling edge arrives, if the time elapsed is greater than the time-out setup by the SIMTOC register, then a time-out condition will occur. The time-out function will stop when an I²C "STOP" condition occurs.

When an I²C time-out counter overflow occurs, the counter will stop and the SIMTOEN Bit will be cleared to zero and the SIMTOF Bit will be set high to indicate that a time-out condition as 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

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

$$((1\sim64) \times 32)/f_{SUB}$$

This gives a range of 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
FOR	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 by time-out and cleared by application program.

Bit 5~0 **SIMTOS5~SIMTOS0**: SIM I²C Time-Out period Selection
The I²C Time-Out clock source is $f_{SUB}/32$.
The I²C Time-Out period is equal to (SIMTOS[5: 0]+1) × $(32/f_{SUB})$

UART Interface

The devices contain an integrated full-duplex asynchronous serial communications UART interface that enables communication with external devices that contain a serial interface. The UART function has many features and can transmit and receive data serially by transferring a frame of data with eight or nine data Bits per transmission as well as being able to detect errors when the data is overwritten or incorrectly framed. The UART function 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, Universal Asynchronous Receiver and Transmitter (UART) 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)
- Transmitter and receiver enabled independently
- 2-byte Deep FIFO Receive Data Buffer
- Transmit and Receive Multiple Interrupt Generation Sources:
  - Transmitter Empty
  - Transmitter Idle
  - Receiver Full
  - Receiver Overrun
  - Address Mode Detect
  - RX pin wake-up interrupt (RX enable, RX falling edge)

UART External Pin Interfacing

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 Bit, the TXEN and RXEN Bits, if set, will automatically setup these I/O or other pin-shared functional pins to their respective TX output and RX input conditions and disable any pull-high resistor option which may exist on the TX pin. However, the pull-high register related to the RX pin is controlled by the corresponding I/O pull-high function control bit. When the TX or RX pin function is disabled by clearing the UARTEN and TXEN or RXEN Bit, the TX or RX pin can be used as a general purpose I/O or other pin-shared functional pin.

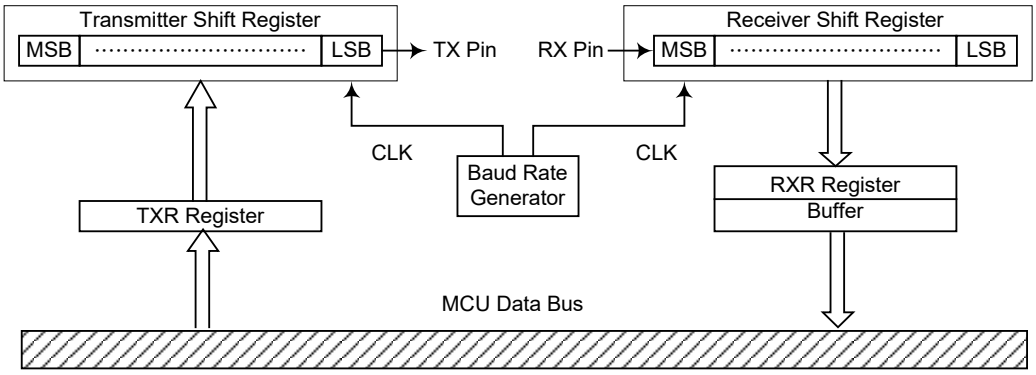
UART Data Transfer Scheme

The block 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 Data Transfer Scheme

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 register.

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 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 Register List

• USR Register

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

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

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

The PERR flag is the parity error flag. When this read only flag is "0", it indicates a parity error has not been detected. When the flag is "1", it indicates that the parity of the received word is incorrect. This error flag is applicable only if 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 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 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 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 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 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 RXR data register status
0: RXR data register is empty
1: 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 RXR read data register is empty. When the flag is "1", it indicates that the RXR read data register contains new data. When the contents of the shift register are transferred to the 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 is cleared when the USR register is read with RXIF set, followed by a read from the RXR register, and if the RXR register has no data available.

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

The TIDLE flag is known as the transmission complete flag. When this read only flag is "0", it indicates that a transmission is in progress. This flag will be set 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 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 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 data register. The TXIF flag is cleared by reading the UART status register (USR) with TXIF set and then writing to the TXR 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 two UART control registers that are used to set the various options for the UART function, such as overall on/off control, parity control, data transfer Bit length etc. Further explanation on each of the Bits is given below.

Bit	7	6	5	4	3	2	1	0
Name	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	x	0

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

Bit 4 **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 for transmitter
0: One stop Bit format is used
1: Two stop Bits format is used
This Bit determines if one or two stop Bits are to be used for the transmitter. When this Bit is equal to "1", two stop Bits are used. If this Bit is equal to "0", then only one stop Bit is used.

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

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

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

• UCR2 Register

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

Bit	7	6	5	4	3	2	1	0
Name	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 enabled control
0: UART transmitter is disabled
1: UART transmitter is enabled
The Bit named TXEN is the Transmitter Enable Bit. When this Bit is equal to "0", the transmitter will be disabled with any pending data transmissions being aborted. In addition the buffers will be reset. In this situation the TX pin will be 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. If this situation occurs, the TX pin will be used as an I/O or other pin-shared functional pin.

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

The Bit named RXEN is the Receiver Enable Bit. When this Bit is equal to "0", the receiver will be disabled with any pending data receptions being aborted. In addition the receive buffers will be reset. In this situation the RX 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. If this situation occurs, the RX pin will be used as an I/O or other pin-shared functional pin.

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 this Bit is equal to "1", the high speed mode is selected. If the Bit is equal to "0", the low speed mode is selected.

Bit 4 **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 detect function enable control Bit. When this Bit is equal to "1", the address detect function is enabled. When it occurs, if the 8th Bit, which corresponds to RX7 if BNO=0 or the 9th Bit, which corresponds to RX8 if BNO=1, has a value of "1", then the received word will be identified as an address, rather than data. If the corresponding interrupt is enabled, an interrupt request will be generated each time the received word has the address Bit set, which is the 8th or 9th Bit depending on the value of BNO. If the address Bit known as the 8th or 9th Bit of the received word is "0" with the address detect function being enabled, an interrupt will not be generated and the received data will be discarded.

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

This bit is used to control the wake-up UART function when a falling edge on the RX pin occurs. Note that this bit is only available when the UART clock source is switched off. There will be no RX pin wake-up UART function if the UART clock exists. If the WAKE bit is set to 1 as the UARTn clock is switched off, a UARTn 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 via the application program. 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

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

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

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

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

• TXR_RXR Register

The TXR_RXRn register is the data register which is used to store the data to be transmitted on the TXn pin or being received from the RXn pin.

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

"x" unknown

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

Baud Rate Generator

To setup the speed of the serial data communication, the UART function contains its own dedicated baud rate generator. The baud rate is controlled by its own internal free running 8-Bit timer, the period of which is determined by two factors. The first of these is the value placed in the baud rate register BRG and the second is the value of the BRGH Bit with the control register UCR2. 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	x	x	x	x	x	x	x	x

"x" unknown

Bit 7~0 **BRG7~BRG0**: Baud Rate values
By programming the BRGH Bit in 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%

The following table shows actual values of baud rate and error values for the two values of BRGH.

Baud Rate K/BPS	f_{SYS}=8MHz					
	Baud Rates for BRGH=0			Baud Rates for BRGH=1		
	BRG	Kbaud	Error (%)	BRG	Kbaud	Error (%)
0.3	—	—	—	—	—	—
1.2	103	1.202	0.16	—	—	—
2.4	51	2.404	0.16	207	2.404	0.16
4.8	25	4.808	0.16	103	4.808	0.16
9.6	12	9.615	0.16	51	9.615	0.16
19.2	6	17.8857	-6.99	25	19.231	0.16
38.4	2	41.667	8.51	12	38.462	0.16
57.6	1	62.500	8.51	8	55.556	-3.55
115.2	0	125	8.51	3	125	8.51
250	—	—	—	1	250	0

Baud Rates and Error Values

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 UART transmitter and receiver are functionally independent, they both use the same data format and baud rate. In all cases stop Bits will be used for data transmission.

Enabling/disabling the UART Interface

The basic on/off function of the internal UART function is controlled using the UARTEN Bit in the UCR1 register. If the UARTEN, TXEN and RXEN Bits are set, then these two UART pins will act as normal TX output pin and RX 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 allow these two pins to be used as normal I/O or other pin-shared functional pins. When the UART function is disabled the buffer will be reset to an empty condition, at the same time discarding any remaining residual data. Disabling the UART will also reset the error and status flags with Bits TXEN, RXEN, TXBRK, RXIF,

OERR, FERR, PERR and NF 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

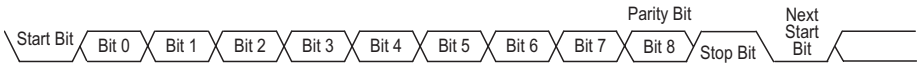
The format of the data to be transferred is composed of various factors such as data Bit length, parity on/off, parity type, address Bits and the number of stop Bits. These factors are determined by the setup of various Bits within the UCR1 register. The BNO Bit controls the number of data Bits which can be set to either 8 or 9, the PRT Bit controls the choice of odd or even parity, the PREN Bit controls the parity on/off function and the STOPS Bit decides whether one or two stop Bits are to be used. The following table shows various formats for data transmission. The address Bit identifies the frame as an address character. The number of stop Bits, which can be either one or two, is independent of the data length and are only to be used for Transmitter. There is only one stop Bit for Receiver.

Start Bit	Data Bits	Address Bits	Parity Bits	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 register. The data to be transmitted is loaded into this TXR 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 register, if it is available. It should be noted that the TSR register, unlike many other registers, is not directly mapped into the Data Memory area and as such is not available to the application program for direct read/write operations. An actual transmission of data will normally be enabled when the TXEN Bit is set, but the data will not be transmitted until the TXR 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 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 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 will then return to the I/O or other pin-shared function.

Transmitting Data

When the UART is transmitting data, the data is shifted on the TX pin from the shift register, with the least significant Bit first. In the transmit mode, the TXR 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 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 is "0", data will be inhibited from being written to the TXR register. Clearing the TXIF flag is always achieved using the following software sequence:

- A USR register access
- A TXR register write execution

The read-only TXIF flag is set by the UART hardware and if set indicates that the TXR register is empty and that other data can now be written into the TXR 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 register will place the data into the TXR 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 register will place the data directly into the shift register, resulting in the commencement of data transmission, and the TXIF Bit being immediately set. When a frame transmission is complete, which happens after stop Bits are sent or after the break frame, the TIDLE Bit will be set. To clear the TIDLE Bit the following software sequence is used:

- A USR register access
- A TXR 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 break characters will be sent on the next transmission. Break character transmission consists of a start Bit, followed by 13×N '0' Bits and stop Bits, where N=1, 2, etc. If a break character is to be transmitted then the 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 highs at the end of the last break character will ensure that the start Bit of the next frame is recognized.

UART Receiver

The UART is capable of receiving word lengths of either 8 or 9 Bits can be selected by programming the BNO Bit in the UCR register. If the BNO Bit is set, the word length will be set to 9 Bits with the MSB being stored in the RX8 Bit of the UCR1 register. At the receiver core lies the Receive Serial Shift Register, commonly known as the RSR. The data which is received on the RX 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 RXR register is a two byte deep FIFO data buffer, where two bytes can be held in the FIFO while a third byte can continue to be received. Note that the application program must ensure that the data is read from RXR before the third byte has been completely shifted in, otherwise this third 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 BNO, PRT and PREN Bits to define the word length and parity type.
- 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 RXR register has data available, at least one character can be read.
- When the contents of the shift register have been transferred to the 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:

- A USR register access
- An RXR register read execution

Receive Break

Any break character received by the UART will be managed as a framing error. The receiver will count and expect a certain number of Bit times as specified by the values programmed into the BNO and one STOPS Bit. If the break is much longer than 13 Bit times, the reception will be considered as complete after the number of Bit times specified by BNO and one STOP Bit. 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. If a long break signal has been detected and the receiver has received a start Bit, the data Bits and the invalid stop Bit, which sets the FERR flag, 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. A break is regarded as a character that contains only zeros with the FERR flag set. 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, 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 Bit is "1", when a word is transferred from the Receive Shift Register, RSR, to the Receive Data Register, RXR. An overrun error can also generate an interrupt if RIE is "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 Flag

The RXR register is composed of a two byte deep FIFO data buffer, where two bytes can be held in the FIFO register, while a third byte can continue to be received. Before this third byte has been entirely shifted in, the data should be read from the 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 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 RXR register.

Noise Error – NF Flag

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 RXR register.
- No interrupt will be generated. However this Bit rises at the same time as the RXIF Bit which itself generates an interrupt.

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

Framing Error – FERR Flag

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, only the first stop Bit is detected, it must be high. If the first stop Bit is low, the FERR flag will be set. The FERR flag is buffered along with the received data and is cleared on any reset.

Parity Error – PERR Flag

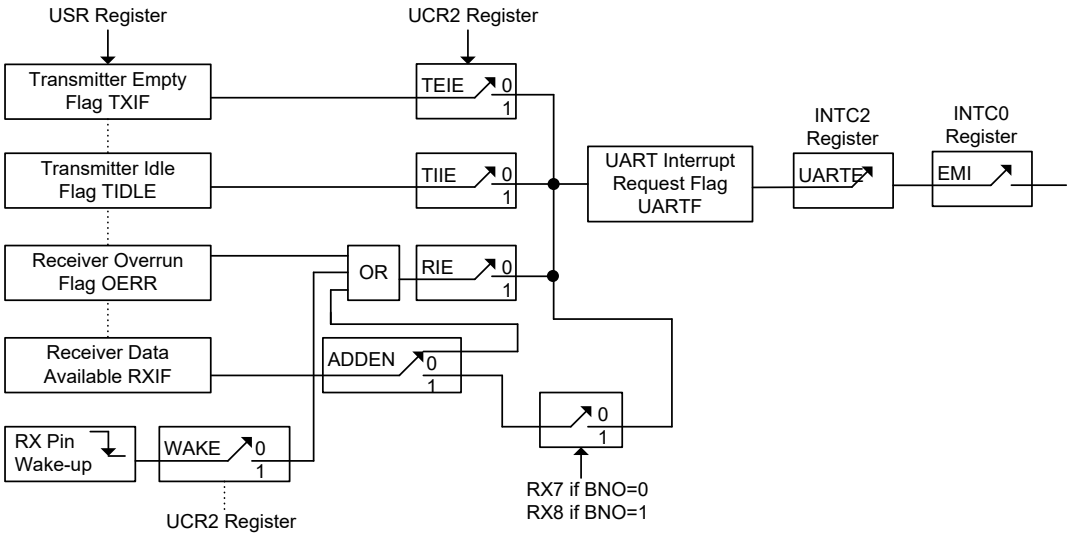
The read only parity error flag, PERR, in the USR register, is set if the parity of the received word is incorrect. This error flag is only applicable if the parity is enabled, PREN Bit is "1", and if the parity type, odd or even is selected. The read only PERR flag is buffered along with the received data bytes. It is cleared on any reset. It should be noted that the FERR and PERR flags are buffered along with the corresponding word and should be read before reading the data word.

UART Module 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 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 Scheme

Address Detect Mode

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

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

ADDEN Bit Function

UART Power Down and Wake-up

When the the device system clock is switched off, the UART will cease to function. If the device executes the "HALT" instruction and switches off the system clock 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 device executes the "HALT" instruction and switches off the system clock while receiving data, then the reception of data will likewise be paused. When the device enters the IDLE or SLEEP 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 IDLE or SLEEP 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 device enters the IDLE0 or SLEEP Mode, then a falling edge on the RX pin will wake up the UART function from the IDLE0 or SLEEP Mode. 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, UARTE, must also be set. If these two 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.

Interrupts

Interrupts are an important part of any microcontroller system. When an external event or an internal function such as a Touch Action or Timer/Event Counter overflow requires microcontroller attention, their corresponding interrupt will enforce a temporary suspension of the main program allowing the microcontroller to direct attention to their respective needs. The devices contain several external interrupt and internal interrupt functions. The external interrupt is generated by the action of the external INT pin and Touch Keys, while the internal interrupts are generated by various internal functions such as Timer Modules, Time Bases, SIM, LVD, EEPROM, A/D converter and UART.

Interrupt Registers

Overall interrupt control, which basically means the setting of request flags when certain microcontroller conditions occur and the setting of interrupt enable Bits by the application program, is controlled by a series of registers, located in the Special Purpose Data Memory, as shown in the accompanying table. The registers fall into two categories. The first is the INTC0~INTC3 registers which setup the primary interrupts, the second is the INTEG register to setup the external interrupt trigger edge type.

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

Function	Enable Bit	Request Flag	Notes
Global	EMI	—	—
INT Pin	INTE	INTF	—
Touch Key Module	TKME	TKMF	—
SIM	SIME	SIMF	—
UART	UARTE	UARTF	—
EEPROM	DEE	DEF	—
LVD	LVDE	LVDF	—
Time Base	TBnE	TBnF	n=0 or 1
A/D Converter	ADE	ADF	—
TM	CTMPnE	CTMPnF	n=0
	PTMPnE	PTMPnF	n=1 or 2
	CTMAnE	CTMAnF	n=0
	PTMAnE	PTMAnF	n=1 or 2

Interrupt Register Bit Naming Conventions

Interrupt Register Contents

Name	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
INTEG	—	—	—	—	—	—	INTS1	INTS0
INTC0	—	TB0F	TKMF	INTF	TB0E	TKME	INTE	EMI
INTC1	PTMA1F	PTMP1F	CTMA0F	CTMP0F	PTMA1E	PTMP1E	CTMA0E	CTMP0E
INTC2	UARTF	DEF	SIMF	TB1F	UARTE	DEE	SIME	TB1E
INTC3	PTMA2F	PTMP2F	ADF	LVDF	PTMA2E	PTMP2E	ADE	LVDE

- **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**: Defines INT interrupt active edge
00: Disabled interrupt
01: Rising Edge interrupt
10: Falling Edge interrupt
11: Dual Edge interrupt

- **INTC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	TB0F	TKMF	INTF	TB0E	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 **TB0F**: Time Base 0 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 pin interrupt request flag
0: No request
1: Interrupt request

Bit 3 **TB0E**: Time Base 0 interrupt control
0: Disable
1: Enable

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

Bit 1 **INTE**: INT pin 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	PTMA1F	PTMP1F	CTMA0F	CTMP0F	PTMA1E	PTMP1E	CTMA0E	CTMP0E
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 **PTMA1F**: PTM1 CCRA comparator interrupt request flag
0: No request
1: Interrupt request

Bit 6 **PTMP1F**: PTM1 CCRP comparator interrupt request flag
0: No request
1: Interrupt request

Bit 5 **CTMA0F**: CTM0 CCRA comparator interrupt request flag
0: No request
1: Interrupt request

Bit 4 **CTMP0F**: CTM0 CCRP comparator interrupt request flag
0: No request
1: Interrupt request

Bit 3 **PTMA1E**: PTM1 CCRA comparator interrupt control
0: Disable
1: Enable

Bit 2 **PTMP1E**: PTM1 CCRP comparator interrupt control
0: Disable
1: Enable

Bit 1 **CTMA0E**: CTM0 CCRA comparator interrupt control
0: Disable
1: Enable

Bit 0 **CTMP0E**: CTM0 CCRP comparator interrupt control
0: Disable
1: Enable

• **INTC2 Register**

Bit	7	6	5	4	3	2	1	0
Name	UARTF	DEF	SIMF	TB1F	UARTE	DEE	SIME	TB1E
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 **UARTF**: UART 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 **TB1F**: Time Base 1 interrupt request flag
0: No request
1: Interrupt request

Bit 3 **UARTE**: UART interrupt request control
0: Disable
1: Enable

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

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

Bit 0 **TB1E**: Time Base 1 interrupt control
0: Disable
1: Enable

• **INTC3 Register**

Bit	7	6	5	4	3	2	1	0
Name	PTMA2F	PTMP2F	ADF	LVDF	PTMA2E	PTMP2E	ADE	LVDE
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 **PTMA2F**: PTM2 CCRA comparator interrupt request flag
0: No request
1: Interrupt request

Bit 6 **PTMP2F**: PTM2 CCRP comparator 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 **LVDF**: LVD interrupt request flag
0: No request
1: Interrupt request

Bit 3 **PTMA2E**: PTM2 CCRA comparator interrupt control
0: Disable
1: Enable

Bit 2 **PTMP2E**: PTM2 CCRP comparator interrupt control
0: Disable
1: Enable

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

Bit 0 **LVDE**: LVD interrupt control
0: Disable
1: Enable

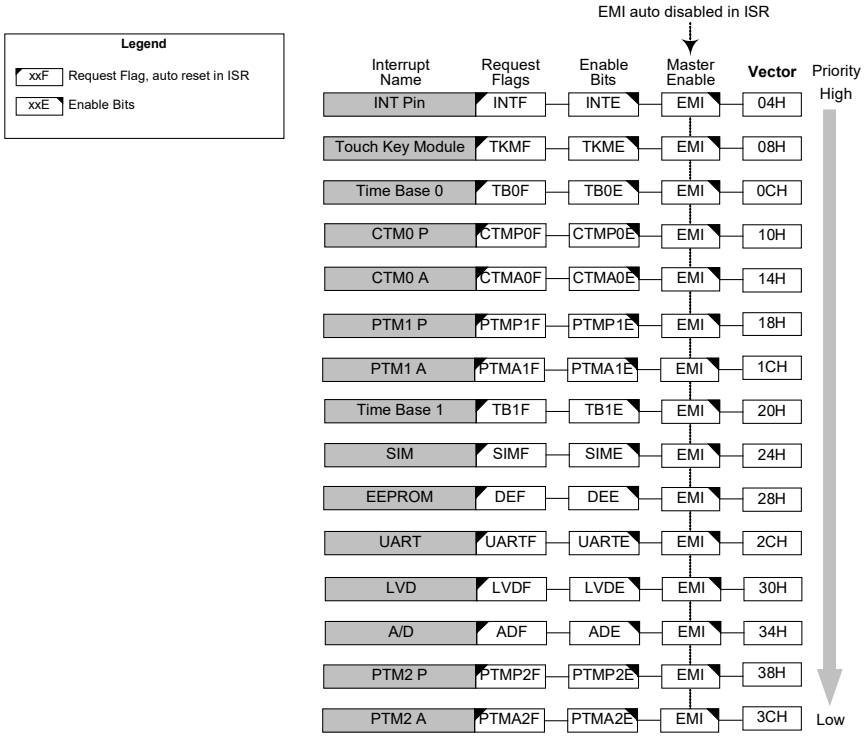
Interrupt Operation

When the conditions for an interrupt event occur, such as a Touch Key Counter overflow, a TM Comparator P or Comparator A match, 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. Every interrupt source has its own individual vector. Once an interrupt subroutine is serviced, all the other interrupts will be blocked, as the global interrupt enable Bit, EMI Bit will be cleared automatically. This will prevent any further interrupt nesting from occurring. However, if other interrupt requests occur during this interval, although the interrupt will not be immediately serviced, the request flag will still be recorded.

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



Interrupt Structure

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 pin, it can only be configured as external interrupt pin if its external interrupt enable Bit in the corresponding interrupt register has been set. The pin must also be setup as an input by setting the corresponding Bit in the port control register. When the interrupt is enabled, the stack is not full and the correct transition type appears on the external interrupt pin, a subroutine call to the external interrupt vector, will take place. When the interrupt is serviced, the external interrupt request flag, INTF, will be automatically reset and the EMI Bit will be automatically cleared to disable other interrupts. Note that any pull-high resistor selections on the external interrupt pin will remain valid even if the pin is used as an external interrupt input.

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

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 timer function. When this happens its interrupt request flags TBnF will be set. To allow the program to branch to its interrupt vector address, 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 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. Each Time Base clock source originates from an independent internal prescaler.

Each 15-Bit prescaler can source from f_{SYS}, $f_{SYS}/4$, f_{SUB} or f_H, selected by CLKSELn1~CLKSELn0 Bits in the PSCR register.

• **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**: Time Base 1 prescaler clock source 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**: Time Base 0 prescaler clock source selection
00: f_{SYS}
01: $f_{SYS}/4$
10: f_{SUB}
11: f_H

• **TBC Register**

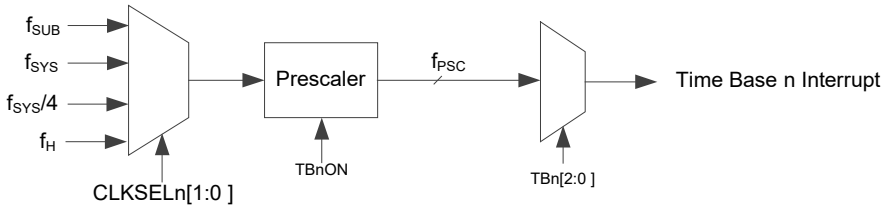
Bit	7	6	5	4	3	2	1	0
Name	TB1ON	TB12	TB11	TB10	TB0ON	TB02	TB01	TB00
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7 **TB1ON**: Time Base 1 enable/disable control
0: Disable
1: Enable

Bit 6~4 **TB12 ~ TB10**: Select Time Base 1 Time-out Period
000: $2^8/f_{PSC}$
001: $2^9/f_{PSC}$
010: $2^{10}/f_{PSC}$
011: $2^{11}/f_{PSC}$
100: $2^{12}/f_{PSC}$
101: $2^{13}/f_{PSC}$
110: $2^{14}/f_{PSC}$
111: $2^{15}/f_{PSC}$

Bit 3 **TB0ON**: Time Base 0 enable/disable control
0: Disable
1: Enable

Bit 2~0 **TB02 ~ TB00**: Select Time Base 1 Time-out Period
000: $2^8/f_{PSC}$
001: $2^9/f_{PSC}$
010: $2^{10}/f_{PSC}$
011: $2^{11}/f_{PSC}$
100: $2^{12}/f_{PSC}$
101: $2^{13}/f_{PSC}$
110: $2^{14}/f_{PSC}$
111: $2^{15}/f_{PSC}$



Time Base Structure (n=0 or 1)

TM Interrupts

The Compact and Periodic type TMs each has two internal interrupts, the internal comparator A or comparator P, which generates a TM interrupt when a compare match condition occurs. For each of the Compact and Periodic Type TMs, there are two interrupt request flags, CTMP0F/CTMA0F and PTMPnF/PTMAnF, and two enable Bits, CTMP0E/CTMA0E and PTMPnE/PTMAnE. 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 macth situation happens. To allow the program to branch to its respective interrupt vector address, the global interrupt enable Bit, EMI, the respective TM interrupt enable Bit must first be set. When the interrupt is enabled, the stack is not full and a TM comparator match situation occurs, a subroutine call to the relevant TM interrupt vector location, will take place. When the TM interrupt is serviced, the TM interrupt request flag will be automatically reset and the EMI Bit will be automatically cleared to disable other interrupts.

EEPROM Interrupt

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

LVD Interrupt

An LVD Interrupt request will take place when the LVD Interrupt request flag, LVDF, is set, which occurs when the Low Voltage Detector function detects a low power supply voltage. To allow the program to branch to its respective interrupt vector address, the global interrupt enable Bit, EMI, and Low Voltage Interrupt enable Bit, LVDE, must first be set. When the interrupt is enabled, the stack is not full and a low voltage condition occurs, a subroutine call to the LVD Interrupt vector, will take place. When the Low Voltage Interrupt is serviced, the LVDF flag will be automatically cleared and the EMI Bit will be automatically cleared to disable other interrupts.

Touch Key Interrupt

For a Touch Key interrupt to occur, the global interrupt enable Bit, EMI, and the Touch Key interrupt enable TKME must be first set. An actual Touch Key interrupt will take place when the Touch Key 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 timer interrupt vector, will take place. When the interrupt is serviced, the Touch Key interrupt request flag, TKMF, will be automatically reset and the EMI Bit will be automatically cleared to disable other interrupts.

Serial Interface Module Interrupt

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

UART Interrupt

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. To allow the program to branch to the respective interrupt vector addresses, the global interrupt enable Bit, EMI, and UART interrupt enable Bit, UARTE, must first be set. When the interrupt is enabled, the stack is not full and any of these conditions are created, a subroutine call to the UART Interrupt vector will take place. When the interrupt is serviced, the UART Interrupt flag, UARTF, will be automatically cleared. The EMI Bit will also be automatically cleared to disable other interrupts. However, the USR register flags will be cleared automatically when certain actions are taken by the UART, the details of which are given in the UART section.

Interrupt Wake-up Function

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

Programming Considerations

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

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

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

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

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

SCOM and SSEG Function for LCD

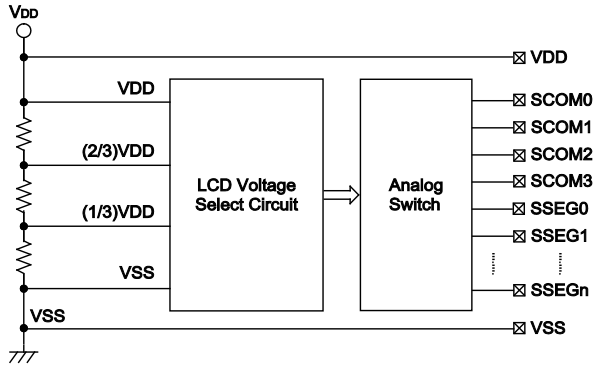
The devices have the capability of driving external LCD panels. The common pins for LCD driving, SCOM0~SCOM3, SSEG0~SSEG19, are pin shared with certain pins on the I/O ports. The LCD signals (COM and SEG) are generated using the application program.

Device	COMs	SEGs
BS86B12A-3	SCOM0~SCOM3	SSEG0~SSEG15
BS86C16A-3		SSEG0~SSEG19
BS86D20A-3		

LCD Operation

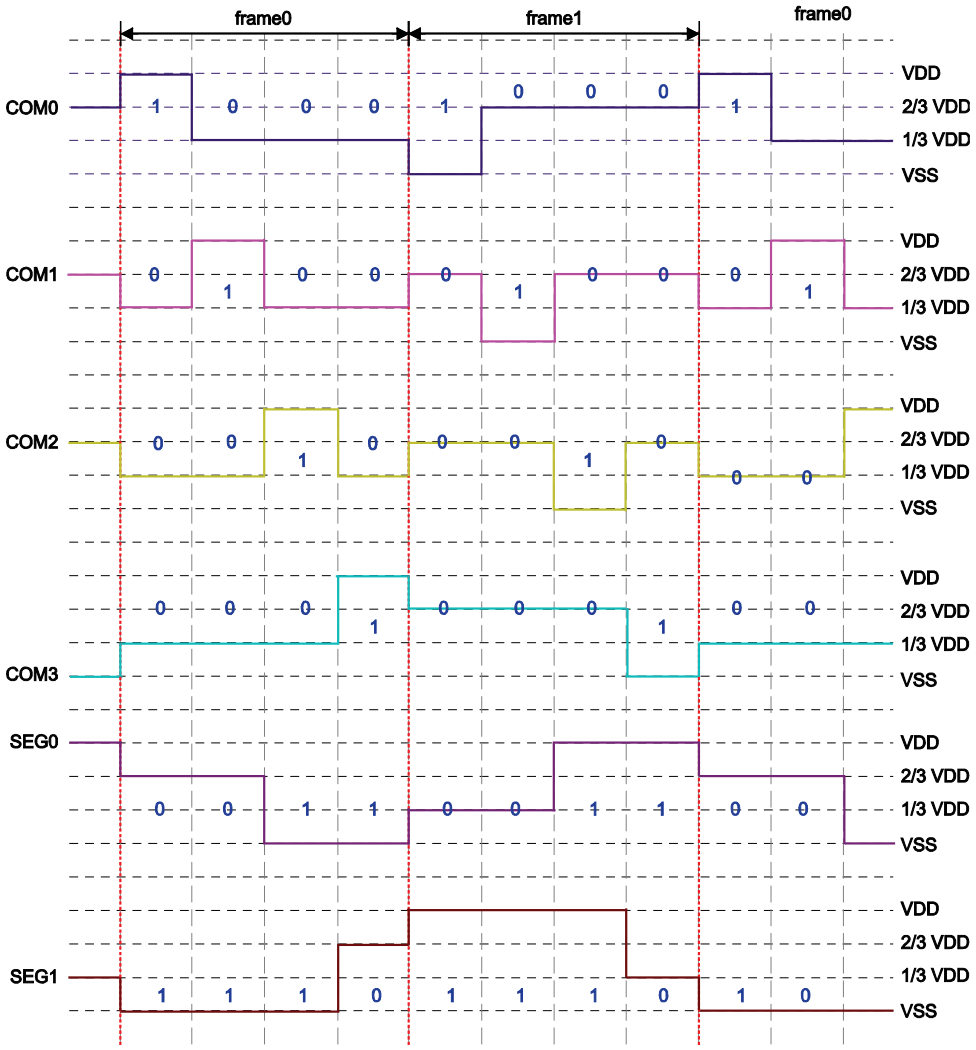
An external LCD panel can be driven using the devices by configuring the I/O pins as common pins and configuring the I/O pins as segment pins. The LCD driver function is controlled using the LCD control registers which in addition to controlling the overall on/off function also controls the SCOM and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

The LCDEN Bit in the SLCDC0 register is the overall master control for the LCD driver, however this Bit is used in conjunction with the COMnEN and SEGnEN Bits to select which I/O Port pins are used for LCD driving. Note that the Port Control register does not need to first setup the pins as outputs to enable the LCD driver operation.



LCD Driver Structure

The accompanying waveform diagram shows a typical 1/3 Bias LCD waveform generated using the application program. Note that the depiction of a "1" in the diagram illustrates an illuminated LCD pixel. The COM signal polarity generated on pins SCOM0~SCOM3, whether 0 or 1, are generated using the corresponding I/O data register Bits, which are Bits PA0~PA2, PA4 in the PA register.



Note: The logical values shown in the diagram are the PA I/O register Bit values, PA0~PA2, PA4.

1/3 Bias LCD Waveform

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

In Frame 0

To select Frame 0 clear the FRAME Bit to 0.

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

In Frame 1

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

The COM0~COMn waveform is controlled by the application program using the FRAME Bit, and the corresponding I/O data register for the respective COM pin to determine whether the COM0~COMn output has a value of either V_{DD}, V_{SS} or V_{bias}. The SEG0~SEGm waveform is controlled in a similar way using the FRAME Bit and the corresponding I/O data register for the respective SEG pin to determine whether the SEG0~SEGn output has a value of either V_{DD}, V_{SS} or V_{bias}.

LCD Bias Control

The LCD COM and SEG driver enable a range of selections to be provided to suit the requirement of the LCD panel which are being used. The bias resistor choice is implemented using the ISEL1 and ISEL0 Bits in the SLCDC0 register.

• **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**: Fram 0 or Frame 1 output selection
0: Frame 0
1: Frame 1

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

Bit 4 **LCDEN**: SCOM and SSEG module on/off control
0: Disable
1: Enable
The SCOMn and SSEGm lines can be enabled using COMnEN and SEGmEN if LCDEN=1. When LCDEN=0, then the SCOMn and SSEGm outputs will be fixed at a V_{SS} level.

Bit 3 **COM3EN**: SCOM3 or other function selection
0: Other function
1: SCOM3

Bit 2 **COM2EN**: SCOM2 or other function selection
0: Other function
1: SCOM2

Bit 1 **COM1EN**: SCOM1 or other function selection
0: Other function
1: SCOM1

Bit 0 **COM0EN**: SCOM0 or other function selection
0: Other function
1: 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 or other function selection
0: Other function
1: 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 or other function selection
0: Other function
1: SSEG15~SSEG8

• **SLCDC3 Register – BS86C16A-3/BS86D20A-3 only**

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 or other function selection
0: Other function
1: SSEG19~SSEG16

Low Voltage Detector – LVD

Each device has a Low Voltage Detector function, also known as LVD. This enables the device to monitor the power supply voltage, V_{DD}, and provides 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 detemined. 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	—	VLVD2	VLVD1	VLVD0
R/W	—	—	R	R/W	—	R/W	R/W	R/W
POR	—	—	0	0	—	0	0	0

Bit 7 ~ 6 Unimplemented, read as "0"

Bit 5 **LVDO**: LVD Output Flag
0: No Low Voltage Detect
1: Low Voltage Detect

Bit 4 **LVDEN**: Low Voltage Detector Control
0: Disable
1: Enable

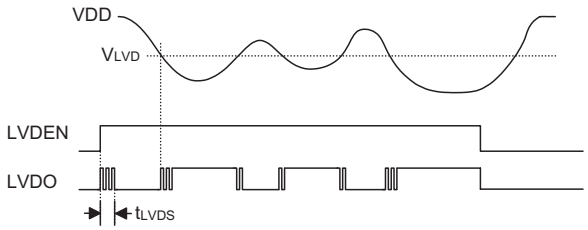
Bit 3 Unimplemented, read as "0"

Bit 2~0 **VLVD2 ~ VLVD0**: Select LVD Voltage
000: 2.0V
001: 2.2V
010: 2.4V
011: 2.7V
100: 3.0V
101: 3.3V
110: 3.6V
111: 4.0V

Note: The V_{LVR} of these two devices is fixed at 2.55V, so 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 in SLEEP mode the low voltage detector will be automatically disabled even if the LVDEN Bit is high. After enabling the Low Voltage Detector, a time delay t_{LVDS} should be allowed for the circuitry to stabilise before reading the LVDO Bit. Note also that as the V_{DD} voltage may rise and fall rather slowly, at the voltage nears that of V_{LVD}, there may be multiple Bit LVDO transitions.



LVD Operation

The Low Voltage Detector also has its own interrupt, providing an alternative means of low voltage detection, in addition to polling the LVDO Bit. The interrupt will only be generated after a delay of t_{LVD} after the LVDO Bit has been set high by a low voltage condition. In this case, the LVDF 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 IDLE Mode, however if the Low Voltage Detector wake up function is not required then the LVDF flag should be first set high before the device enters the IDLE Mode.

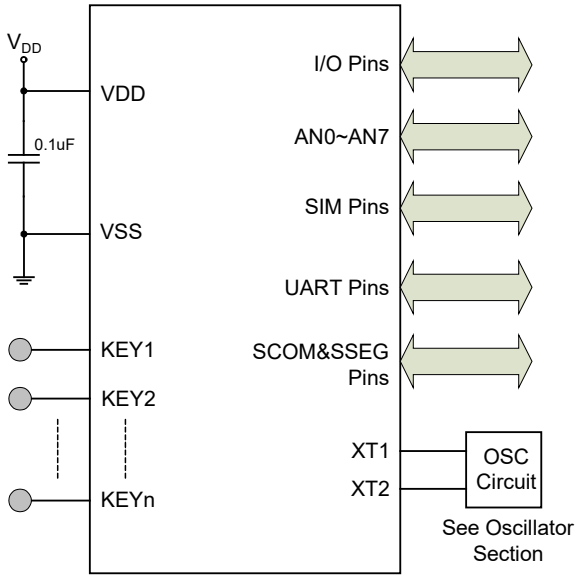
Configuration Options

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, LXT
2	HIRC frequency selection: 8MHz, 12MHz, 16MHz

Note: 1. The low speed system oscillator selection is only for the BS86C16A-3 and BS86D20A-3.
2. When the HIRC has been configurated at a frequency shown in this table, the HIRCS1 and HIRCS0 Bits is recommended to be setup to select the same frequency to keep the HIRC frequency accuracy spedified in the A.C. characteristics.

Application Circuit



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 several kinds of MOV instructions, data can be transferred from registers to the Accumulator and vice-versa as well as being able to move specific immediate data directly into the Accumulator. One of the most important data transfer applications is to receive data from the input ports and transfer data to the output ports.

Arithmetic Operations

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

Logical and Rotate Operation

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

Branches and Control Transfer

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

Bit Operations

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

Table Read Operations

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

Other Operations

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

Instruction Set Summary

The instructions related to the data memory access in the following table can be used when the desired data memory is located in Data Memory sector 0.

Table Conventions

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

Mnemonic	Description	Cycles	Flag Affected
Arithmetic			
ADD A,[m]	Add Data Memory to ACC	1	Z, C, AC, OV, SC
ADDM A,[m]	Add ACC to Data Memory	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 (specific page) 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 two cycles are required, if no skip takes place only one cycle is required.

2. Any instruction which changes the contents of the PCL will also require 2 cycles for execution.

Extended Instruction Set

The extended instructions are used to support the full range address access for the data memory. When the accessed data memory is located in any data memory sector except sector 0, the extended instruction can be used to directly access the data memory instead of using the indirect addressing access. This can not only reduce the use of Flash memory space but also improve the CPU execution efficiency.

Mnemonic	Description	Cycles	Flag Affected
Arithmetic			
LADD A,[m]	Add Data Memory to ACC	2	Z, C, AC, OV, SC
LADDM A,[m]	Add ACC to Data Memory	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 (specific page) 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 (specific page) 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 three cycles are required, if no skip takes place two cycles is required.

2. Any extended instruction which changes the contents of the PCL register will also require three cycles for execution.

Instruction Definition

ADC A,[m] Add Data Memory to ACC with Carry

Description The contents of the specified Data Memory, Accumulator and the carry flag are added. The result is stored in the Accumulator.

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

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

SBC A, x Subtract immediate data from ACC with Carry

Description: The immediate data and the complement of the carry flag are subtracted from the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.

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

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

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

SDZ [m] Skip if decrement Data Memory is 0

Description: The contents of the specified Data Memory are first decremented by 1. If the result is 0 the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation:
[m] ← [m] − 1
Skip if [m]=0

Affected flag(s): None

SDZA [m] Skip if decrement Data Memory is zero with result in ACC

Description: The contents of the specified Data Memory are first decremented by 1. If the result is 0, the following instruction is skipped. The result is stored in the Accumulator but the specified Data Memory contents remain unchanged. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0, the program proceeds with the following instruction.

Operation:
ACC ← [m] − 1
Skip if ACC=0

Affected flag(s): None

SET [m] Set Data Memory

Description Each bit of the specified Data Memory is set to 1.

Operation [m] ← FFH

Affected flag(s) None

SET [m].i Set bit of Data Memory

Description Bit i of the specified Data Memory is set to 1.

Operation [m].i ← 1

Affected flag(s) None

SIZ [m] Skip if increment Data Memory is 0

Description The contents of the specified Data Memory are first incremented by 1. If the result is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation [m] ← [m] + 1
Skip if [m]=0

Affected flag(s) None

SIZA [m] Skip if increment Data Memory is zero with result in ACC

Description The contents of the specified Data Memory are first incremented by 1. If the result is 0, the following instruction is skipped. The result is stored in the Accumulator but the specified Data Memory contents remain unchanged. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation ACC ← [m] + 1
Skip if ACC=0

Affected flag(s) None

SNZ [m].i Skip if bit i of Data Memory is not 0

Description If bit i of the specified Data Memory is not 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is 0 the program proceeds with the following instruction.

Operation Skip if [m].i ≠ 0

Affected flag(s) None

SNZ [m] Skip if Data Memory is not 0

Description The contents of the specified Data Memory are read out and then written back to the specified Data Memory again. If the specified Data Memory is not 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is 0 the program proceeds with the following instruction.

Operation Skip if [m]≠ 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 The contents of the specified Data Memory are read out and then written back to the specified Data Memory again. If the contents of the specified Data Memory is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation Skip if [m]=0

Affected flag(s) None

SZA [m] Skip if Data Memory is 0 with data movement to ACC

Description The contents of the specified Data Memory are copied to the Accumulator. If the value is zero, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation ACC ← [m]
Skip if [m]=0

Affected flag(s) None

SZ [m].i Skip if bit i of Data Memory is 0

Description If bit i of the specified Data Memory is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0, the program proceeds with the following instruction.

Operation Skip if [m].i=0

Affected flag(s) None

TABRD [m] Read table (specific page) to TBLH and Data Memory

Description The low byte of the program code (specific page) addressed by the table pointer (TBLP and TBHP) is moved to the specified Data Memory and the high byte moved to TBLH.

Operation [m] ← program code (low byte)
TBLH ← program code (high byte)

Affected flag(s) None

TABRDL [m] Read table (last page) to TBLH and Data Memory

Description The low byte of the program code (last page) addressed by the table pointer (TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.

Operation [m] ← program code (low byte)
TBLH ← program code (high byte)

Affected flag(s) None

ITABRD [m] Increment table pointer low byte first and read table (specific page) to TBLH and Data Memory

Description Increment table pointer low byte, TBLP, first and then the program code (specific page) addressed by the table pointer (TBHP and TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.

Operation [m] ← program code (low byte)
TBLH ← program code (high byte)

Affected flag(s) None

ITABRDL [m] Increment table pointer low byte first and read table (last page) to TBLH and Data Memory

Description Increment table pointer low byte, TBLP, first and then the low byte of the program code (last page) addressed by the table pointer (TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.

Operation [m] ← program code (low byte)
TBLH ← program code (high byte)

Affected flag(s) None

XOR A,[m] Logical XOR Data Memory to ACC

Description Data in the Accumulator and the specified Data Memory perform a bitwise logical XOR operation. The result is stored in the Accumulator.

Operation ACC ← ACC ″XOR″ [m]

Affected flag(s) Z

XORM A,[m] Logical XOR ACC to Data Memory

Description Data in the specified Data Memory and the Accumulator perform a bitwise logical XOR operation. The result is stored in the Data Memory.

Operation [m] ← ACC ″XOR″ [m]

Affected flag(s) Z

XOR A,x Logical XOR immediate data to ACC

Description Data in the Accumulator and the specified immediate data perform a bitwise logical XOR operation. The result is stored in the Accumulator.

Operation ACC ← ACC ″XOR″ x

Affected flag(s) Z

Extended Instruction Definition

The extended instructions are used to directly access the data stored in any data memory sections.

LADC A,[m] Add Data Memory to ACC with Carry

Description The contents of the specified Data Memory, Accumulator and the carry flag are added. The result is stored in the Accumulator.

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

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

LSBCM A,[m] Subtract Data Memory from ACC with Carry and result in Data Memory

Description: The contents of the specified Data Memory and the complement of the carry flag are subtracted from the Accumulator. The result is stored in the Data Memory. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.

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

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

LSDZ [m] Skip if decrement Data Memory is 0

Description The contents of the specified Data Memory are first decremented by 1. If the result is 0 the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a three cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation [m] ← [m] − 1
Skip if [m]=0

Affected flag(s) None

LSDZA [m] Skip if decrement Data Memory is zero with result in ACC

Description The contents of the specified Data Memory are first decremented by 1. If the result is 0, the following instruction is skipped. The result is stored in the Accumulator but the specified Data Memory contents remain unchanged. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a three cycle instruction. If the result is not 0, the program proceeds with the following instruction.

Operation ACC ← [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 three 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 three 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 bit i of Data Memory is not 0

Description If bit i of the specified Data Memory is not 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a three cycle instruction. If the result is 0 the program proceeds with the following instruction.

Operation Skip if [m].i ≠ 0

Affected flag(s) None

LSNZ [m] Skip if Data Memory is not 0

Description The contents of the specified Data Memory are read out and then written to the specified Data Memory again. If the content of the specified Data Memory is not 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a three cycle instruction. If the result is 0 the program proceeds with the following instruction.

Operation Skip if [m] ≠ 0

Affected flag(s) None

LSUB A,[m] Subtract Data Memory from ACC

Description The specified Data Memory is subtracted from the contents of the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.

Operation ACC ← ACC − [m]

Affected flag(s) OV, Z, AC, C, SC, CZ

LSUBM A,[m] Subtract Data Memory from ACC with result in Data Memory

Description The specified Data Memory is subtracted from the contents of the Accumulator. The result is stored in the Data Memory. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.

Operation [m] ← ACC − [m]

Affected flag(s) OV, Z, AC, C, SC, CZ

LSWAP [m] Swap nibbles of Data Memory

Description The low-order and high-order nibbles of the specified Data Memory are interchanged.

Operation [m].3~[m].0 ↔ [m].7~[m].4

Affected flag(s) None

LSWAPA [m] Swap nibbles of Data Memory with result in ACC

Description The low-order and high-order nibbles of the specified Data Memory are interchanged. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.

Operation ACC.3~ACC.0 ← [m].7~[m].4
ACC.7~ACC.4 ← [m].3~[m].0

Affected flag(s) None

LSZ [m] Skip if Data Memory is 0

Description The contents of the specified Data Memory are read out and then written to the specified Data Memory again. If the contents of the specified Data Memory is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a three cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation Skip if [m]=0

Affected flag(s) None

LSZA [m] Skip if Data Memory is 0 with data movement to ACC

Description The contents of the specified Data Memory are copied to the Accumulator. If the value is zero, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a three cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation ACC ← [m]
Skip if [m]=0

Affected flag(s) None

LSZ [m].i Skip if bit i of Data Memory is 0

Description If bit i of the specified Data Memory is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a three cycle instruction. If the result is not 0, the program proceeds with the following instruction.

Operation Skip if [m].i=0

Affected flag(s) None

LTABRD [m] Read table (specific page) to TBLH and Data Memory

Description The low byte of the program code (specific page) addressed by the table pointer (TBHP and TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.

Operation [m] ← program code (low byte)
TBLH ← program code (high byte)

Affected flag(s) None

LTABRDL [m] Read table (last page) to TBLH and Data Memory

Description The low byte of the program code (last page) addressed by the table pointer (TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.

Operation [m] ← program code (low byte)
TBLH ← program code (high byte)

Affected flag(s) None

LITABRD [m] Increment table pointer low byte first and read table (specific page) to TBLH and Data Memory

Description Increment table pointer low byte, TBLP, first and then the program code (specific page) addressed by the table pointer (TBHP and TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.

Operation [m] ← program code (low byte)
TBLH ← program code (high byte)

Affected flag(s) None

LITABRDL [m] Increment table pointer low byte first and read table (last page) to TBLH and Data Memory

Description Increment table pointer low byte, TBLP, first and then the low byte of the program code (last page) addressed by the table pointer (TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.

Operation [m] ← program code (low byte)
TBLH ← program code (high byte)

Affected flag(s) None

LXOR A,[m] Logical XOR Data Memory to ACC

Description Data in the Accumulator and the specified Data Memory perform a bitwise logical XOR operation. The result is stored in the Accumulator.

Operation ACC ← ACC ″XOR″ [m]

Affected flag(s) Z

LXORM A,[m] Logical XOR ACC to Data Memory

Description Data in the specified Data Memory and the Accumulator perform a bitwise logical XOR operation. The result is stored in the Data Memory.

Operation [m] ← ACC ″XOR″ [m]

Affected flag(s) Z

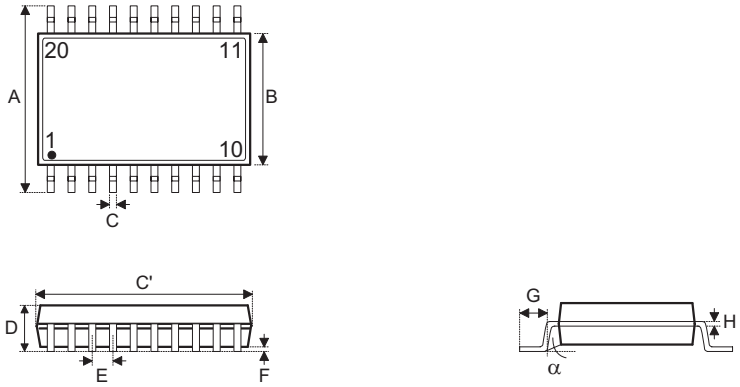
Package Information

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

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

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

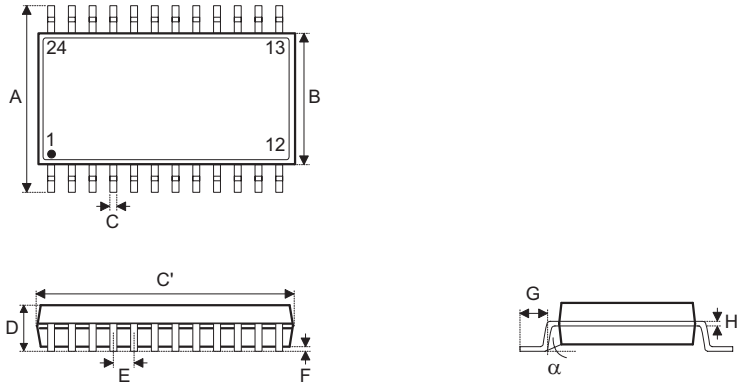
20-pin SOP (300mil) Outline Dimensions



Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	—	0.406 BSC	—
B	—	0.295 BSC	—
C	0.012	—	0.020
C’	—	0.504 BSC	—
D	—	—	0.104
E	—	0.050 BSC	—
F	0.004	—	0.012
G	0.016	—	0.050
H	0.008	—	0.013
α	0°	—	8°

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	—	10.30 BSC	—
B	—	7.50 BSC	—
C	0.31	—	0.51
C’	—	12.80 BSC	—
D	—	—	2.65
E	—	1.27 BSC	—
F	0.10	—	0.30
G	0.40	—	1.27
H	0.20	—	0.33
α	0°	—	8°

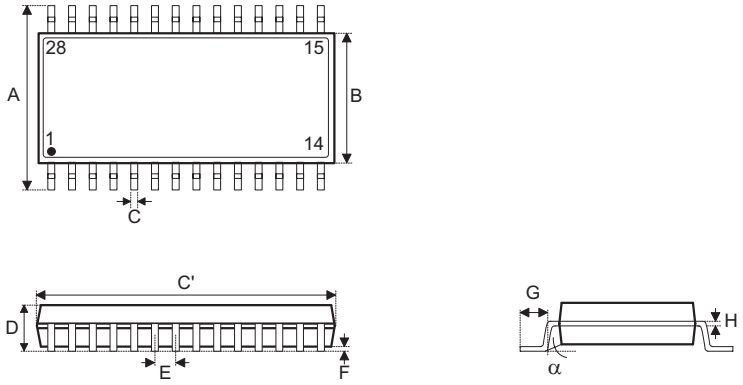
24-pin SOP(300mil) Outline Dimensions



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°