HOLTEK

Sub-1GHz OOK/FSK RF Transmitter Touch Flash MCU

BC66F2235/BC66F2245/BC66F2255

Revision: V1.00 Date: November 02, 2021

Features

CPU Features

- Operating voltage
  - f_{SYS}=8MHz: 2.0V~3.6V
  - f_{SYS}=16MHz: 2.0V~3.6V
- Up to 0.25μs instruction cycle with 16MHz system clock at V_{DD}=5V
- Power saving and wake-up functions to reduce power consumption
- Oscillator types
  - RF External High Speed Crystal – HXT
  - Internal High Speed 8MHz RC – HIRC
  - Internal Low Speed 32kHz RC – LIRC
- Multi-mode operation: FAST, SLOW, IDLE, SLEEP and DEEP SLEEP
- Fully integrated internal oscillators require no external components
- All instructions executed in 1~3 instruction cycles
- Table read instructions
- 115 powerful instructions
- 8-level subroutine nesting
- Bit manipulation instruction

Peripheral Features

- Flash Program Memory: 2K×16~8K×16
- RAM Data Memory: 256×8
- Touch Key Data Memory: 96×8
- Up to 16 touch key functions – fully integrated without requiring external components
- In Application Programming function – IAP
- Internal On-Chip Debug Support function – OCDS
- Watchdog Timer function
- Up to 23 bidirectional I/O lines
- Up to 2 pin-shared external interrupts
- Multiple Timer Modules for time measurement, compare match output or PWM output or single pulse output function
  - 2 Compact type 10-bit Timer Modules – CTM0~CTM1
  - 1 Periodic type 10-bit Timer Module – PTM0
- Universal Serial Interface Module – USIM for SPI, I^2C or UART communication
- Dual Time-Base functions for generation of fixed time interrupt signals
- Up to 4 external channel 12-bit resolution A/D converter
- Integrated 1.5V LDO
- Low voltage reset function
- Low voltage detect function
- Package types: 16-pin NSOP-EP, 24-pin SSOP-EP, 32-pin QFN

RF Transmitter Features

- Complete ultra high frequency OOK/FSK transmitter with low transmission phase noise
- Supports 315/433/868/915MHz frequency bands
- Programmable channel setting with < 2kHz resolution
- OOK symbol rates 0.5Ksps~25Ksps, FSK data rates 0.5Kbps~100Kbps
- Output power up to +13dBm (software controlled output power: 0dBm, +10dBm, +13dBm)
- RF external crystal: 16MHz

General Description

This series of devices are Flash Memory A/D type 8-bit high performance RISC architecture microcontrollers with fully integrated Touch Key functions and an integrated RF transmitter function, giving them the great flexibility for use in a wide range of wireless with touch key control applications such as industrial control, consumer products, subsystem controllers, etc. With all touch key functions provided internally, completely eliminating the need for external components, these devices have all the features to offer designers a reliable and easy means of implementing touch keys within their products applications.

For memory features, the Flash Memory offers users the convenience of multi-programming features. Other memory includes an area of RAM Data Memory. By using Holtek's In Application Programming technology, users have a convenient means to directly store their measured data in the Flash Program Memory as well as having the ability to easily update their application programs.

Analog feature includes a multi-channel 12-bit A/D converter. With regard to internal timers, these devices include multiple and extremely flexible Timer Modules providing functions for timing, pulse generation and PWM output operations. Communication with the outside world is catered for by including fully integrated SPI, I^2C and UART interface functions, three popular interfaces which provide designers with a means of easy communication with external peripheral hardware. Protective features such as an internal Watchdog Timer, Low Voltage Reset and Low Voltage Detector coupled with excellent noise immunity and ESD protection ensure that reliable operation is maintained in hostile electrical environments.

A full choice of external high, internal high and low oscillators is provided including two fully integrated system oscillators which require no external components for their implementation. The ability to operate and switch dynamically between a range of operating modes using different clock sources gives users the ability to optimise the conflicting demands of microcontroller performance and power consumption.

The integrated RF transmitter can operate at the 315MHz, 433MHz, 868MHz and 915MHz frequency bands. The addition of a crystal and a limited number of external components is all that is required to create a complete and versatile RF transmitter system. These devices include an internal power amplifier and are capable of delivering +13dBm (Max.) into a 50Ω load. Such a power level enables a small form factor transmitter to operate near the maximum transmission regulation limits. These devices can operate with OOK – On-Off Keying and FSK – Frequency Shift Keying receiver types. The FSK data rate is up to 100Kbps, allowing these devices to support more complicated control protocols.

The inclusion of flexible I/O programming features, Time-Base functions along with many other features ensure that these devices will be highly capable of providing cost effective Sub-1GHz RF transmitter solutions for remote wireless applications.

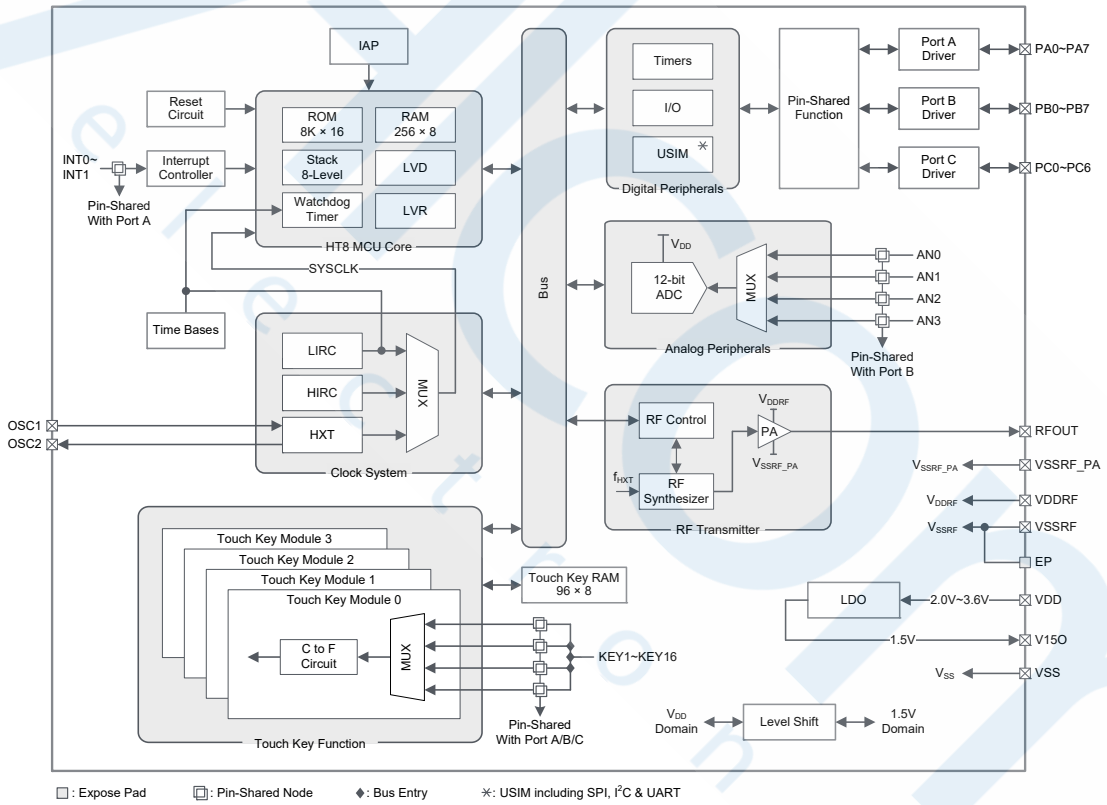
Selection Table

Most features are common to these devices, the main features distinguishing them are Program Memory capacity, I/O count, external interrupt count, A/D channel count, Touch key count and package type. The following table summarises the main features of each device.

Part No.	V_{DD}	ROM	RAM	Touch Key RAM	I/O	External Interrupt	A/D	Time Base	Timer Module
BC66F2235	2.0V~3.6V	2K×16	256×8	96×8	8	1	12-bit×1	2	10-bit CTM×2 10-bit PTM×1
BC66F2245		4K×16			15	2	12-bit×4		
BC66F2255		8K×16			23	2	12-bit×4		

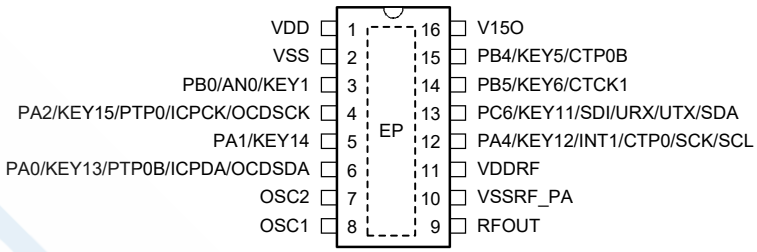
Part No.	Integrated RF		Touch Key	USIM (SPI+I²C+UART)	LDO	Stack	Package
	Band	Type					
BC66F2235	315~915MHz	OOK/FSK TX	8	√	√	8	16NSOP-EP
BC66F2245			14				24SSOP-EP
BC66F2255			16				32QFN

Block Diagram

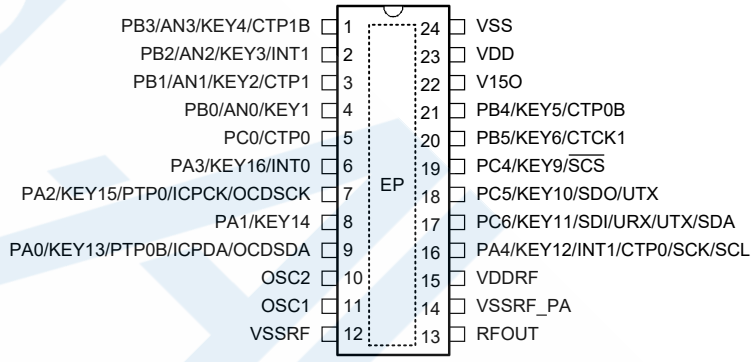


Note: The figure illustrates the Block Diagram of the device with maximum features, the functional differences between the series of devices are provided in the Selection Table.

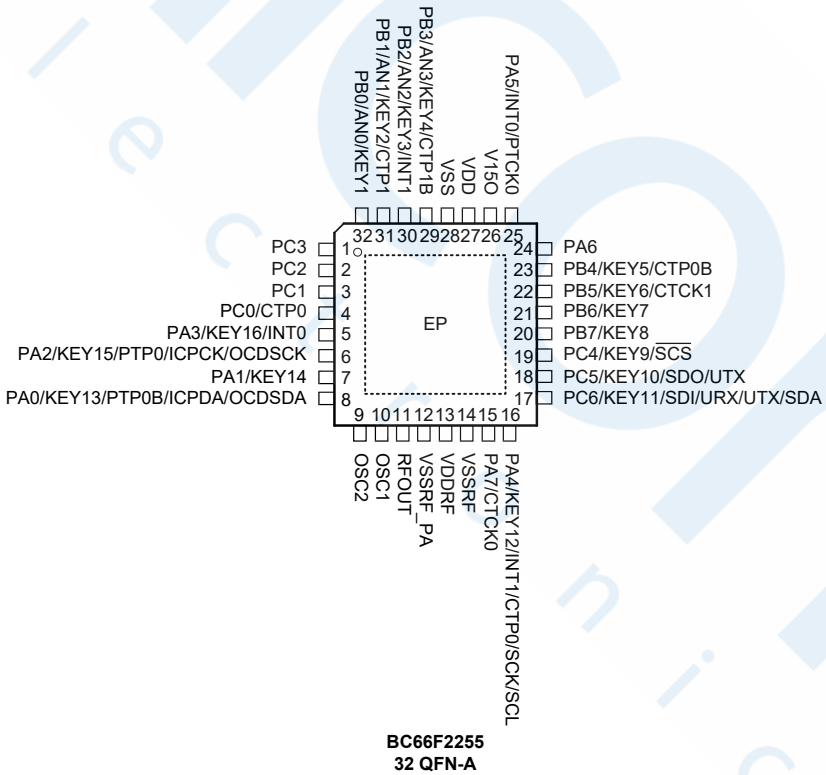
Pin Assignment



BC66F2235
16 NSOP-EP-A



BC66F2245
24 SSOP-EP-A



BC66F2255
32 QFN-A

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

2. The OCDSDA and OCDSCK pins are used as the OCDS dedicated pins.

Pin Description

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

BC66F2235

Pin Name	Function	OPT	I/T	O/T	Description
PA0/KEY13/PTP0B/ ICPDA/OCDSDA	PA0	PAPU PAWU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	KEY13	PAS0	AN	—	Touch key input
	PTP0B	PAS0	—	CMOS	PTM0 inverted output
	ICPDA	—	ST	CMOS	ICP data/address
	OCDSDA	—	ST	CMOS	OCDS data/address
PA1/KEY14	PA1	PAPU PAWU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	KEY14	PAS0	AN	—	Touch key input
PA2/KEY15/PTP0/ ICPCK/OCDSCK	PA2	PAPU PAWU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	KEY15	PAS0	AN	—	Touch key input
	PTP0	PAS0	—	CMOS	PTM0 output
	ICPCK	—	ST	—	ICP clock
	OCDSCK	—	ST	—	OCDS clock
PA4/KEY12/INT1/ CTP0/SCK/SCL	PA4	PAPU PAWU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	KEY12	PAS1	AN	—	Touch key input
	INT1	PAS1 IFS INTEG INTC0	ST	—	External interrupt 1 input
	CTP0	PAS1	—	CMOS	CTM0 output
	SCK	PAS1	ST	CMOS	SPI serial clock
	SCL	PAS1	ST	NMOS	I^2C clock line
PB0/AN0/KEY1	PB0	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	AN0	PBS0	AN	—	A/D Converter external input channel
	KEY1	PBS0	AN	—	Touch key input
PB4/KEY5/CTP0B	PB4	PBPU PBS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	KEY5	PBS1	AN	—	Touch key input
	CTP0B	PBS1	—	CMOS	CTM0 inverted output
PB5/KEY6/CTCK1	PB5	PBPU PBS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	KEY6	PBS1	AN	—	Touch key input
	CTCK1	PBS1	ST	—	CTM1 clock input

Pin Name	Function	OPT	I/T	O/T	Description
PC6/KEY11/SDI/URX/UTX/SDA	PC6	PCPU PCS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	KEY11	PCS1	AN	—	Touch key input
	SDI	PCS1	ST	—	SPI serial data input
	URX/UTX	PCS1	ST	CMOS	UART serial data input in full-duplex communication or UART serial data input / output in Single Wire Mode communication
	SDA	PCS1	ST	NMOS	I^2C data line
OSC1	OSC1	—	HXT	—	RF oscillator input
OSC2	OSC2	—	—	HXT	RF oscillator output
RFOUT	RFOUT	—	—	AN	RF power amplifier output
VDD	VDD	—	PWR	—	Digital positive power supply
V15O	V15O	—	—	PWR	Internal 1.5V LDO output
VSS	VSS	—	PWR	—	Digital negative power supply
VDDRF	VDDRF	—	PWR	—	RF positive power supply
VSSRF_PA	VSSRF_PA	—	PWR	—	RF power amplifier negative power supply
EP	EP	—	PWR	—	Exposed pad, must be connected to ground

Legend: I/T: Input type; O/T: Output type;
OPT: Optional by register option; PWR: Power;
ST: Schmitt Trigger input; CMOS: CMOS output;
NMOS: NMOS output; AN: Analog signal;
HXT: High frequency crystal oscillator

BC66F2245

Pin Name	Function	OPT	I/T	O/T	Description
PA0/KEY13/PTP0B/ICPDA/OCDSDA	PA0	PAPU PAWU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	KEY13	PAS0	AN	—	Touch key input
	PTP0B	PAS0	—	CMOS	PTM0 inverted output
	ICPDA	—	ST	CMOS	ICP data/address
	OCDSDA	—	ST	CMOS	OCDS data/address
PA1/KEY14	PA1	PAPU PAWU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	KEY14	PAS0	AN	—	Touch key input
PA2/KEY15/PTP0/ICPCK/OCDSCK	PA2	PAPU PAWU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	KEY15	PAS0	AN	—	Touch key input
	PTP0	PAS0	—	CMOS	PTM0 output
	ICPCK	—	ST	—	ICP clock
	OCDSCK	—	ST	—	OCDS clock
PA3/KEY16/INT0	PA3	PAPU PAWU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	KEY16	PAS0	AN	—	Touch key input
	INT0	PAS0 IFS INTEG INTC0	ST	—	External interrupt 0 input

Pin Name	Function	OPT	I/T	O/T	Description
PA4/KEY12/INT1/CTP0/SCK/SCL	PA4	PAPU PAWU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	KEY12	PAS1	AN	—	Touch key input
	INT1	PAS1 IFS INTEG INTC0	ST	—	External interrupt 1 input
	CTP0	PAS1	—	CMOS	CTM0 output
	SCK	PAS1	ST	CMOS	SPI serial clock
	SCL	PAS1	ST	NMOS	I^2C clock line
PB0/AN0/KEY1	PB0	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	AN0	PBS0	AN	—	A/D Converter external input channel
	KEY1	PBS0	AN	—	Touch key input
PB1/AN1/KEY2/CTP1	PB1	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	AN1	PBS0	AN	—	A/D Converter external input channel
	KEY2	PBS0	AN	—	Touch key input
	CTP1	PBS0	—	CMOS	CTM1 output
PB2/AN2/KEY3/INT1	PB2	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	AN2	PBS0	AN	—	A/D Converter external input channel
	KEY3	PBS0	AN	—	Touch key input
	INT1	PBS0 IFS INTEG INTC0	ST	—	External interrupt 1 input
PB3/AN3/KEY4/CTP1B	PB3	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	AN3	PBS0	AN	—	A/D Converter external input channel
	KEY4	PBS0	AN	—	Touch key input
	CTP1B	PBS0	—	CMOS	CTM1 inverted output
PB4/KEY5/CTP0B	PB4	PBPU PBS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	KEY5	PBS1	AN	—	Touch key input
	CTP0B	PBS1	—	CMOS	CTM0 inverted output
PB5/KEY6/CTCK1	PB5	PBPU PBS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	KEY6	PBS1	AN	—	Touch key input
	CTCK1	PBS1	ST	—	CTM1 clock input
PC0/CTP0	PC0	PCPU PCS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	CTP0	PCS0	—	CMOS	CTM0 output
PC4/KEY9/$\overline{SCS}$	PC4	PCPU PCS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	KEY9	PCS1	AN	—	Touch key input
	$\overline{SCS}$	PCS1	ST	CMOS	SPI slave chip select

Pin Name	Function	OPT	I/T	O/T	Description
PC5/KEY10/SDO/UTX	PC5	PCPU PCS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	KEY10	PCS1	AN	—	Touch key input
	SDO	PCS1	—	CMOS	SPI serial data output
	UTX	PCS1	—	CMOS	UART serial data output
PC6/KEY11/SDI/ URX/UTX/SDA	PC6	PCPU PCS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	KEY11	PCS1	AN	—	Touch key input
	SDI	PCS1	ST	—	SPI serial data input
	URX/UTX	PCS1	ST	CMOS	UART serial data input in full-duplex communication or UART serial data input / output in Single Wire Mode communication
	SDA	PCS1	ST	NMOS	I^2C data line
OSC1	OSC1	—	HXT	—	RF oscillator input
OSC2	OSC2	—	—	HXT	RF oscillator output
RFOUT	RFOUT	—	—	AN	RF power amplifier output
VDD	VDD	—	PWR	—	Digital positive power supply
V15O	V15O	—	—	PWR	Internal 1.5V LDO output
VSS	VSS	—	PWR	—	Digital negative power supply
VDDRF	VDDRF	—	PWR	—	RF positive power supply
VSSRF	VSSRF	—	PWR	—	RF negative power supply
VSSRF_PA	VSSRF_PA	—	PWR	—	RF power amplifier negative power supply
EP	EP	—	PWR	—	Exposed pad, must be connected to ground

Legend: I/T: Input type;
O/T: Output type;
OPT: Optional by register option;
PWR: Power;
ST: Schmitt Trigger input;
CMOS: CMOS output;
NMOS: NMOS output;
AN: Analog signal;
HXT: High frequency crystal oscillator

BC66F2255

Pin Name	Function	OPT	I/T	O/T	Description
PA0/KEY13/PTP0B/ ICPDA/OCDSDA	PA0	PAPU PAWU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	KEY13	PAS0	AN	—	Touch key input
	PTP0B	PAS0	—	CMOS	PTM0 inverted output
	ICPDA	—	ST	CMOS	ICP data/address
	OCDSDA	—	ST	CMOS	OCDS data/address
PA1/KEY14	PA1	PAPU PAWU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	KEY14	PAS0	AN	—	Touch key input
PA2/KEY15/PTP0/ ICPCK/OCDSCK	PA2	PAPU PAWU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	KEY15	PAS0	AN	—	Touch key input
	PTP0	PAS0	—	CMOS	PTM0 output
	ICPCK	—	ST	—	ICP clock
	OCDSCK	—	ST	—	OCDS clock

Pin Name	Function	OPT	I/T	O/T	Description
PA3/KEY16/INT0	PA3	PAPU PAWU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	KEY16	PAS0	AN	—	Touch key input
	INT0	PAS0 IFS INTEG INTC0	ST	—	External interrupt 0 input
PA4/KEY12/INT1/ CTP0/SCK/SCL	PA4	PAPU PAWU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	KEY12	PAS1	AN	—	Touch key input
	INT1	PAS1 IFS INTEG INTC0	ST	—	External interrupt 1 input
	CTP0	PAS1	—	CMOS	CTM0 output
	SCK	PAS1	ST	CMOS	SPI serial clock
	SCL	PAS1	ST	NMOS	I^2C clock line
PA5/INT0/PTCK0	PA5	PAPU PAWU	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	INT0	IFS INTEG INTC0	ST	—	External interrupt 0 input
	PTCK0	—	ST	—	PTM0 clock input
PA6	PA6	PAPU PAWU	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
PA7/CTCK0	PA7	PAPU PAWU	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	CTCK0	—	ST	—	CTM0 clock input
PB0/AN0/KEY1	PB0	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	AN0	PBS0	AN	—	A/D Converter external input channel
	KEY1	PBS0	AN	—	Touch key input
PB1/AN1/KEY2/CTP1	PB1	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	AN1	PBS0	AN	—	A/D Converter external input channel
	KEY2	PBS0	AN	—	Touch key input
	CTP1	PBS0	—	CMOS	CTM1 output
PB2/AN2/KEY3/INT1	PB2	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	AN2	PBS0	AN	—	A/D Converter external input channel
	KEY3	PBS0	AN	—	Touch key input
	INT1	PBS0 IFS INTEG INTC0	ST	—	External interrupt 1 input
PB3/AN3/KEY4/ CTP1B	PB3	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	AN3	PBS0	AN	—	A/D Converter external input channel
	KEY4	PBS0	AN	—	Touch key input
	CTP1B	PBS0	—	CMOS	CTM1 inverted output

Pin Name	Function	OPT	I/T	O/T	Description
PB4/KEY5/CTP0B	PB4	PBPU PBS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	KEY5	PBS1	AN	—	Touch key input
	CTP0B	PBS1	—	CMOS	CTM0 inverted output
PB5/KEY6/CTCK1	PB5	PBPU PBS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	KEY6	PBS1	AN	—	Touch key input
	CTCK1	PBS1	ST	—	CTM1 clock input
PB6/KEY7	PB6	PBPU PBS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	KEY7	PBS1	AN	—	Touch key input
PB7/KEY8	PB7	PBPU PBS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	KEY8	PBS1	AN	—	Touch key input
PC0/CTP0	PC0	PCPU PCS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	CTP0	PCS0	—	CMOS	CTM0 output
PC1~PC3	PCn	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up
PC4/KEY9/$\overline{SCS}$	PC4	PCPU PCS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	KEY9	PCS1	AN	—	Touch key input
	$\overline{SCS}$	PCS1	ST	CMOS	SPI slave chip select
PC5/KEY10/SDO/UTX	PC5	PCPU PCS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	KEY10	PCS1	AN	—	Touch key input
	SDO	PCS1	—	CMOS	SPI serial data output
	UTX	PCS1	—	CMOS	UART serial data output
PC6/KEY11/SDI/ URX/UTX/SDA	PC6	PCPU PCS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	KEY11	PCS1	AN	—	Touch key input
	SDI	PCS1	ST	—	SPI serial data input
	URX/UTX	PCS1	ST	CMOS	UART serial data input in full-duplex communication or UART serial data input / output in Single Wire Mode communication
	SDA	PCS1	ST	NMOS	I^2C data line
OSC1	OSC1	—	HXT	—	RF oscillator input
OSC2	OSC2	—	—	HXT	RF oscillator output
RFOUT	RFOUT	—	—	AN	RF power amplifier output
VDD	VDD	—	PWR	—	Digital positive power supply
V15O	V15O	—	—	PWR	Internal 1.5V LDO output
VSS	VSS	—	PWR	—	Digital negative power supply
VDDRF	VDDRF	—	PWR	—	RF positive power supply
VSSRF	VSSRF	—	PWR	—	RF negative power supply
VSSRF_PA	VSSRF_PA	—	PWR	—	RF power amplifier negative power supply
EP	EP	—	PWR	—	Exposed pad, must be connected to ground

Legend: I/T: Input type;
OPT: Optional by register option;
ST: Schmitt Trigger input;
NMOS: NMOS output;
HXT: High frequency crystal oscillator
O/T: Output type;
PWR: Power;
CMOS: CMOS output;
AN: Analog signal;

Absolute Maximum Ratings

Supply Voltage ..... V_{SS}-0.3V to 3.6V
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_{OH} Total ..... -80mA
I_{OL} Total 80mA
Total Power Dissipation 500mW
ESD HBM ..... ±2kV
ESD MM ..... ±400V

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 device reliability.

These devices are ESD sensitive. HBM (Human Body Mode) is based on MIL-STD-883H Method 3015.8. MM (Machine Mode) is based on JEDEC EIA/JESD22-A115.

D.C. Characteristics

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

Operating Voltage Characteristics

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	MCU Operating Voltage – HXT (RF External Crystal)	—	$f_{SYS}=f_{HXT}$=16MHz	2.0	—	3.6	V
	MCU Operating Voltage – HIRC	—	$f_{SYS}=f_{HIRC}$=8MHz	2.0	—	3.6	V
	MCU Operating Voltage – LIRC	—	$f_{SYS}=f_{LIRC}$=32kHz	2.0	—	3.6	V
V_{15O}	LDO Operating Voltage	2.0V~3.6V	—	-10%	1.5	+10%	V

Operating Current Characteristics

Ta=-40°C~85°C

Symbol	Operating Mode	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{DD}	SLOW Mode – LIRC	3V	$f_{SYS}=f_{LIRC}$=32kHz, RF TX power off, LCMD=0	—	42	70	μA
		3V	$f_{SYS}=f_{LIRC}$=32kHz, RF TX power off, LCMD=1	—	30	50	μA
	FAST Mode – HIRC	3V	$f_{SYS}=f_{HIRC}$=8MHz, RF TX power off	—	0.76	1.50	mA
	FAST Mode – HXT	3V	f_{SUB} on, $f_{SYS}=f_{HXT}$=16MHz, RF TX power off	—	1.1	1.6	mA

Note: When using the characteristic table data, the following notes should be taken into consideration:
1. Any digital inputs are setup in a non floating condition.

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

Standby Current Characteristics

Ta=25°C, unless otherwise specified

Symbol	Standby Mode	Test Conditions		Min.	Typ.	Max.	Max. @85°C	Unit
		V_{DD}	Conditions					
I_{STB}	DEEP SLEEP Mode (LDO Off)	3V	f_{LIRC} off	—	0.4	1.0	1.0	μA
		3V	f_{LIRC} on	—	1.2	3.0	3.0	μA
	IDLE1 Mode – HXT (LDO On)	3V	f_{SUB} on, f_{SYS}=f_{HXT}=16MHz	—	600	900	900	μA

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

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

A.C. Characteristics

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

Internal High Speed Oscillator – HIRC – Frequency Accuracy

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

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Temp.				
f_{HIRC}	8MHz Writer Trimmed HIRC Frequency	3V	25°C	-2%	8	+2%	MHz
		3V±0.1V	0°C~70°C	-5%	8	+5%	
		2.0V~3.6V	0°C~70°C	-7%	8	+7%	
		2.0V~3.6V	-40°C~85°C	-10%	8	+10%	

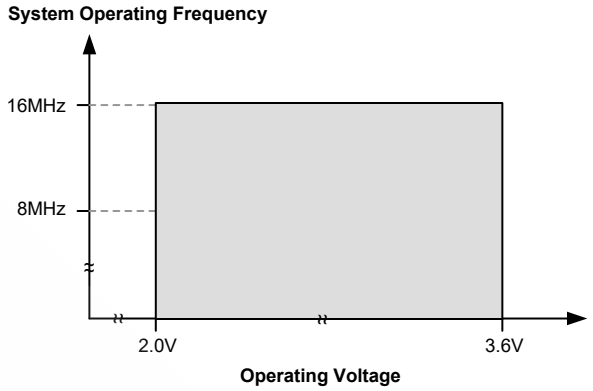
Note: 1. The 3V values for V_{DD} are provided as this is the fixed voltages at which the HIRC frequency is trimmed by the writer.

2. The row below the 3V trim voltage row is provided to show the values for the full V_{DD} range operating voltage. It is recommended that the trim voltage is fixed at 3V for application voltage ranges from 2.0V to 3.6V.

Internal Low Speed Oscillator Characteristics – LIRC

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Temp.				
f_{LIRC}	LIRC Frequency	3V	25°C	-10%	32	+10%	kHz
t_{START}	LIRC Start Up Time	—	-40°C~85°C	—	—	500	μs

Operating Frequency Characteristic Curves



System Start Up Time Characteristics

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
t_{SST}	System Start-up Time (Wake-up from Condition where f_{SYS} is off)	—	$f_{SYS}=f_H \sim f_H/64$, $f_H=f_{HXT}$	—	128	—	t_{HXT}
			$f_{SYS}=f_H \sim f_H/64$, $f_H=f_{HIRC}$	—	16	—	t_{HIRC}
			$f_{SYS}=f_{SUB}=f_{LIRC}$	—	2	—	t_{LIRC}
	System Start-up Time (Wake-up from Condition where f_{SYS} is on)	—	$f_{SYS}=f_H \sim f_H/64$, $f_H=f_{HXT}$ or f_{HIRC}	—	2	—	t_H
			$f_{SYS}=f_{SUB}=f_{LIRC}$	—	2	—	t_{SUB}
	System Speed Switch Time (FAST to Slow Mode or SLOW to FAST Mode)	—	f_{HXT} switches from off → on	—	1024	—	t_{HXT}
			f_{HIRC} switches from off → on	—	16	—	t_{HIRC}
	System Start-up Time (WDT Time-out Hardware Cold Reset)	—	—	—	0	—	t_H
t_{RSTD}	System Reset Delay Time (Reset Source from Power-on Reset or LVR Hardware Reset)	—	RR_{POR}=5V/ms	25	50	100	ms
	System Reset Delay Time (LVRC/WDTC/RSTC Software Reset)	—	—				
	System Reset Delay Time (Reset Source from WDT Overflow)	—	—	8.3	16.7	33.3	
t_{SRESET}	Minimum Software Reset Pulse Width to Reset	—	—	45	90	120	μs

Note: 1. For the System Start-up time values, whether f_{SYS} is on or off depends upon the mode type and the chosen f_{SYS} system oscillator. Details are provided in the System Operating Modes section.

2. The time units, shown by the symbols t_{HIRC}, t_{SYS} etc. are the inverse of the corresponding frequency values as provided in the frequency tables. For example $t_{HIRC}=1/f_{HIRC}$, $t_{SYS}=1/f_{SYS}$ etc.

3. If the LIRC is used as the system clock and if it is off when in the SLEEP/DEEP SLEEP Mode, then an additional LIRC start up time, t_{START}, as provided in the LIRC frequency table, must be added to the t_{SST} time in the table above.

4. The System Speed Switch Time is effectively the time taken for the newly activated oscillator to start up.

Input/Output Characteristics

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{IL}	Input Low Voltage for I/O Ports	3V	—	0	—	0.9	V
		—	—	0	—	$0.3V_{DD}$	
V_{IH}	Input High Voltage for I/O Ports	3V	—	2.1	—	3.0	V
		—	—	$0.7V_{DD}$	—	V_{DD}	
I_{OL}	Sink Current for I/O Ports	3V	V_{OL}=0.1V_{DD}	10	20	—	mA
I_{OH}	Source Current for I/O Ports	3V	V_{OH}=0.9V_{DD}	-5	-10	—	mA
R_{PH}	Pull-high Resistance for I/O Ports (Note)	3V	—	20	30	60	kΩ
V_{OL}	Output Low Voltage for I/O Ports	3V	I_{OL}=10mA	—	—	0.3	V
V_{OH}	Output High Voltage for I/O Ports	3V	I_{OH}=-5mA	2.7	—	—	V
t_{TCK}	TM Clock Input Minimum Pulse Width	—	—	0.3	—	—	μs
t_{INT}	Interrupt Input Pin Minimum Pulse Width	—	—	10	—	—	μs

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

Memory Electrical Characteristics

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

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
Flash Program Memory							
V_{DD}	V_{DD} for Read	—	—	1.9	—	3.6	V
	V_{DD} for Erase / Write	—	—	2.2	—	3.6	
t_{DEW}	Erase / Write Cycle Time	—	—	—	2	3	ms
I_{DDPGM}	Programming / Erase Current on V_{DD}	—	—	—	—	5.0	mA
E_P	Cell Endurance	—	—	10K	—	—	E/W
t_{RETD}	ROM Data Retention Time	—	Ta=25°C	—	40	—	Year
RAM Data Memory							
V_{DD}	V_{DD} for Read / Write	—	—	V_{DDmin}	—	V_{DDmax}	V
V_{DR}	RAM Data Retention Voltage	—	—	1.0	—	—	V

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

LVD/LVR Electrical Characteristics

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{LVR}	Low Voltage Reset Voltage	—	LVR enable, voltage select 1.9V	Typ. -0.1	1.9	Typ. +0.1	V
V_{LVD}	Low Voltage Detection Voltage	—	LVD enable, voltage select 2.0V	-5%	2.0	+5%	V
		—	LVD enable, voltage select 2.2V		2.2		
		—	LVD enable, voltage select 2.4V		2.4		
		—	LVD enable, voltage select 2.7V		2.7		
		—	LVD enable, voltage select 2.9V		2.9		
		—	LVD enable, voltage select 3.0V		3.0		
		—	LVD enable, voltage select 3.3V		3.3		
I_{OP}	Operating Current	3V	LVD enable, LVR enable	—	25	35	μA
t_{BGS}	V_{BG} Turn On Stable Time	—	No load	—	—	150	μs
t_{LVDS}	LVDO Stable Time	—	For LVR enable, LVD off → on	—	—	15	μs
		—	For LVR disable, LVD off → on	—	—	150	μs
t_{LVR}	Minimum Low Voltage Width to Reset	—	—	120	240	480	μs
t_{LVD}	Minimum Low Voltage Width to Interrupt	—	—	60	120	240	μs

A/D Converter Characteristics

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	Operating Voltage	—	—	2.0	—	3.6	V
V_{ADI}	Input Voltage	—	—	0	—	V_{DD}	V
N_R	Resolution	—	—	—	—	12	Bit
DNL	Differential Non-linearity	—	t_{ADCK}=0.5μs	-3	—	3	LSB
INL	Integral Non-linearity	—	t_{ADCK}=0.5μs	-4	—	4	LSB
I_{ADC}	Additional Current Consumption for A/D Converter Enable	2.0V	No load, t_{ADCK}=0.5μs	—	400	500	μA
		3.0V		—	700	850	
		3.6V		—	850	1050	
t_{ADCK}	Clock Period	—	—	0.5	—	10	μs
t_{ON2ST}	A/D Converter On-to-Start Time	—	—	4	—	—	μs
t_{ADC}	Conversion Time (Including A/D Sample and Hold Time)	—	—	—	16	—	t_{ADCK}

RF Transmitter Electrical Characteristics

Ta=-40°C~85°C

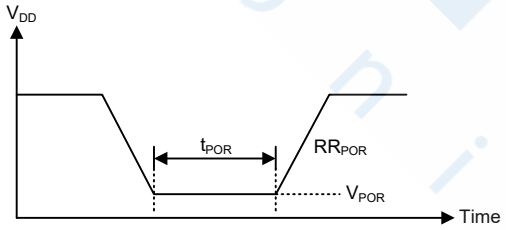
Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DDRF}	RF Operating Voltage	—	—	2.0	3.0	3.6	V
I_{STBRF}	RF Power Down Current	—	—	—	—	1	μA
f_{XTAL}	RF Operating XTAL	—	—	—	16	—	MHz
f_{XTALV}	RF Operating XTAL Variation	—	—	-20	—	+20	ppm
f_{OP}	RF Operating Frequency	—	315/433/868/915MHz band	300	—	930	MHz

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
f_{FSK}	FSK Frequency Deviation	—	—	10	—	150	kHz
f_{STEP}	RF Frequency Step	—	f_{OP}=315/433MHz	—	—	50	Hz
		—	f_{OP}=868/915MHz	—	—	100	Hz
R_{FSK}	Air Data Rate (FSK)	—	—	0.5	—	100	Kbps
R_{OOK}	Air Symbol Rate (OOK)	—	—	0.5	—	25	Ksps
P_{RF}	Output Power	3.0V	—	0	10	13	dBm
P_{RFV}	Output Power Variation	2.0V~3.3V	P_{RF}=10dBm (f_{OP}=433MHz)	—	—	6	dBm
		2.5V~3.3V	P_{RF}=10dBm (f_{OP}=433MHz)	—	—	3	dBm
I_{DDTX}	Transmitter Operating Current	3.0V	P_{RF}=0dBm (f_{OP}=315/433MHz)	—	10	—	mA
		3.0V	P_{RF}=10dBm (f_{OP}=315/433MHz)	—	17	—	mA
		3.0V	P_{RF}=13dBm (f_{OP}=315/433MHz)	—	23	—	mA
		3.0V	P_{RF}=0dBm (f_{OP}=868/915MHz)	—	11	—	mA
		3.0V	P_{RF}=10dBm (f_{OP}=868/915MHz)	—	20	—	mA
		3.0V	P_{RF}=13dBm (f_{OP}=868/915MHz)	—	26	—	mA
PN_{TX}	Transmitter Phase Noise	—	P_{RF}=10dBm, 100kHz from Carrier	—	—	-80	dBc/Hz
		—	P_{RF}=10dBm, 1MHz from Carrier	—	—	-100	dBc/Hz
SE_{TX}	Transmitter Spurious Emission (P_{RF}=10dBm, f_{OP}=433MHz)	—	f<1GHz	—	—	-36	dBm
		—	47MHz<f<74MHz, 87.5MHz<f<118MHz, 174MHz<f<230MHz, 470MHz<f<790MHz	—	—	-54	dBm
		—	2nd Harmonic	—	—	-30	dBm
		—	3rd Harmonic	—	—	-30	dBm
		—	2nd Harmonic (other frequencies)	—	—	-30	dBm
		—	3rd Harmonic (other frequencies)	—	—	-30	dBm

Power-on Reset Characteristics

Ta=-40°C~85°C

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



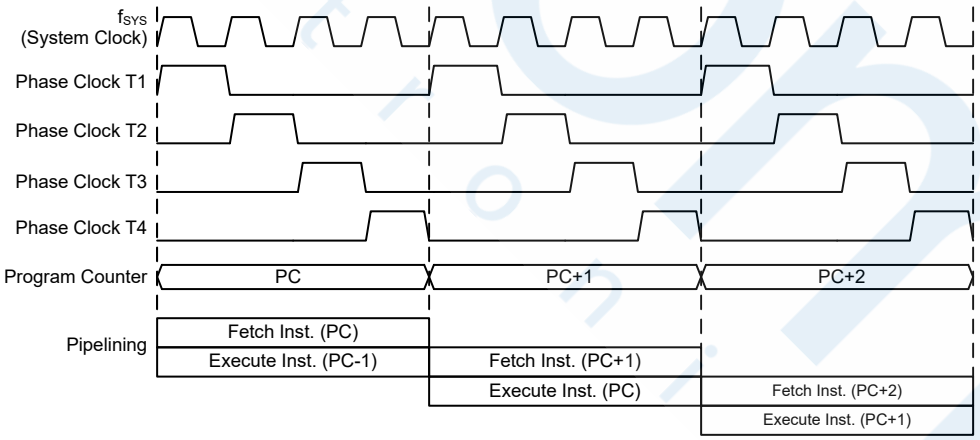
System Architecture

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

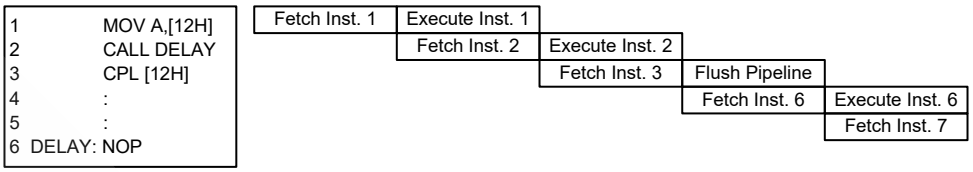
Clocking and Pipelining

The main system clock, derived from either an HXT, 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.

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



System Clocking and Pipelining



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 Byte Register, are directly addressable by the application program.

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

Device	Program Counter	
	High Byte	Low Byte (PCL)
BC66F2235	PC10~PC8	PCL7~PCL0
BC66F2245	PC11~PC8	PCL7~PCL0
BC66F2255	PC12~PC8	PCL7~PCL0

Program Counter

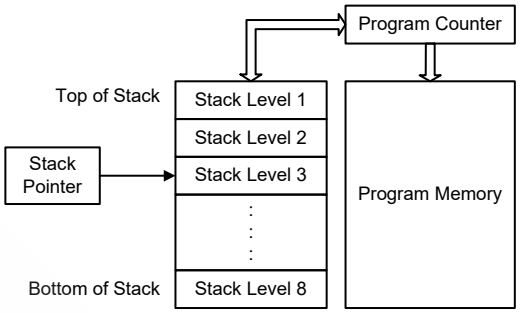
The lower byte of the Program Counter, known as the Program Counter Low Byte register or PCL, is available for program control and is a readable and writeable register. By transferring data directly into this register, a short program jump can be executed directly; however, as only this low byte is available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

Stack

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

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

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



Arithmetic and Logic Unit – ALU

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

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

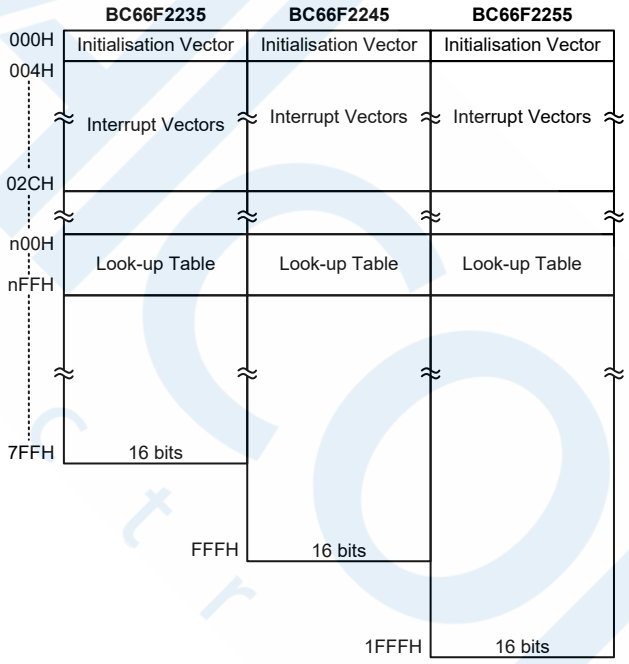
Flash Program Memory

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

Structure

The Program Memory capacity varies from 2K×16 to 8K×16 bits for these devices. 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 separate table pointer registers.

Device	Capacity
BC66F2235	2K×16
BC66F2245	4K×16
BC66F2255	8K×16



Program Memory Structure

Special Vectors

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

Look-up Table

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

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

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

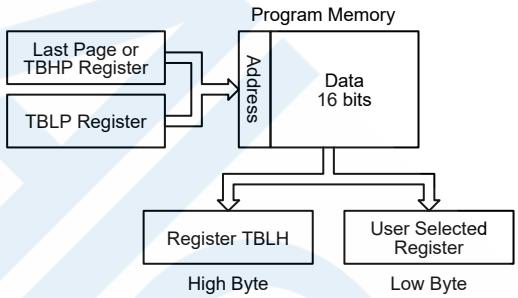


Table Program Example

The following example shows how the table pointer and table data is defined and retrieved from the microcontroller. This example uses raw table data located in the Program Memory which is stored there using the ORG statement. The value at this ORG statement is "700H" which refers to the start address of the last page within the 2K words Program Memory of the BC66F2235. The table pointer low byte register is setup here to have an initial value of "06H". This will ensure that the first data read from the data table will be at the Program Memory address "706H" or 6 locations after the start of the last page. Note that the value for the table pointer is referenced to the specific address pointed by TBHP and TBLP if the "TABRD [m]" or "LTABRD [m]" instruction is being used. The high byte of the table data which in this case is equal to zero will be transferred to the TBLH register automatically when the "TABRD [m]" or "LTABRD [m]" instruction is executed.

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

Table Read Program Example

```
ds .section 'data'
tempreg1 db?        ; temporary register #1
tempreg2 db?        ; temporary register #2
code0 .section 'code'
mov a,06h           ; initialise low table pointer - note that this address is referenced
mov tblp,a          ; to the last page or the page that tbhp pointed
mov a,07h           ; initialise high table pointer
mov tbhp,a          ; it is not necessary to set tbhp if executing tabrdl
:
:
tabrd tempreg1      ; transfers value in table referenced by table pointer data at
                    ; program memory address "706H" transferred to tempreg1 and TBLH
dec tblp            ; reduce value of table pointer by one
tabrd tempreg2      ; transfers value in table referenced by table pointer data at program
                    ; memory address "705H" transferred to tempreg2 and TBLH
                    ; in this example the data "1AH" is transferred to tempreg1 and data "0FH"
                    ; to tempreg2
                    ; the value "00H" will be transferred to the high byte register TBLH
:
:
org 700h            ; sets initial address of last page
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.

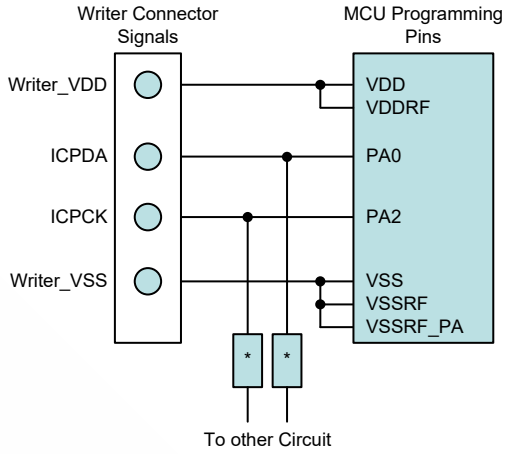
As an additional convenience, Holtek has provided a means of programming the microcontroller in-circuit using a 4-pin interface. This provides manufacturers with the possibility of manufacturing their circuit boards complete with a programmed or un-programmed microcontroller, and then programming or 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 these devices.

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

Note: When using e-Socket to program these devices, users should add a 1μF capacitor to the V15O pin to ensure the programming reliability.

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

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



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

On-Chip Debug Support – OCDS

These devices also provide an "On-Chip Debug" function to debug the MCU during the development process. Users can use the OCDS function to emulate the 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 OCDS function for debugging, other functions which are shared with the OCDSDA and OCDSCK pins in these devices will have no effect. However, the two OCDS pins which are pin-shared with the ICP programming pins are still used as the Flash Memory programming pins for ICP. For more detailed OCDS information, refer to the corresponding document named "Holtek e-Link for 8-bit MCU OCDS User's Guide".

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

In Application Programming – IAP

Flash type Program Memory provides the user with a means of convenient and easy upgrades and modifications to their programs on the same device. The provision of the IAP function offers users the convenience of Flash Memory multi-programming features. The convenience of the IAP function is that it can execute the updated program procedure using its internal firmware, without requiring an external Program Writer or PC. In addition, the IAP interface can also be any type of communication protocol, such as UART, using I/O pins. Regarding the internal firmware, the user can select versions provided by Holtek or create their own. The following section illustrates the procedures regarding how to implement the IAP firmware.

Flash Memory Read/Write Size

The Flash memory Erase operation is carried out in a block format while the Write operation is carried out in 4-word format and the Read operation is carried out in a word format. The block size is assigned with a capacity of 256 words. Note that the Erase operation should be executed before the Write operation is executed.

When the Flash Memory Erase/Write Function is successfully enabled, the CFWEN bit will be set high. When the CFWEN bit is set high, the data can be written into the data registers. The FWT bit is used to initiate the write process and then indicate the write operation status. This bit is set high by application programs to initiate a write process and will be cleared by hardware if the write process is finished.

The Read operation can be carried out by executing a specific read procedure. The FRDEN bit is used to enable the read function and the FRD bit is used to initiate the read process by application programs and then indicate the read operation status. When the read process is finished, this bit will be cleared by hardware.

Device	Program Memory Size	Operations		
		Erase	Write	Read
BC66F2235	2K×16	256 words/block	4 words/time	1 word/time
BC66F2245	4K×16			
BC66F2255	8K×16			

IAP Operation Format

Erase Block	Write Unit	FARH[2:0]	FARL[7:2]	FARL[1:0]
0	0	000	0000 00	xx
	1	000	0000 01	xx
	:	:	:	:
	62	000	1111 10	xx
	63	000	1111 11	xx
1	64	001	0000 00	xx
	65	001	0000 01	xx
	:	:	:	:
	126	001	1111 10	xx
	127	001	1111 11	xx
:	:	:	:	:
7	448	111	0000 00	xx
	449	111	0000 01	xx
	:	:	:	:
	510	111	1111 10	xx
	511	111	1111 11	xx

"x": don't care

Erase Block Number and Selection – BC66F2235

Erase Block	Write Unit	FARH[3:0]	FARL[7:2]	FARL[1:0]
0	0	0000	0000 00	xx
	1	0000	0000 01	xx
	:	:	:	:
	62	0000	1111 10	xx
	63	0000	1111 11	xx
1	64	0001	0000 00	xx
	65	0001	0000 01	xx
	:	:	:	:
	126	0001	1111 10	xx
	127	0001	1111 11	xx
:	:	:	:	:
15	960	1111	0000 00	xx
	961	1111	0000 01	xx
	:	:	:	:
	1022	1111	1111 10	xx
	1023	1111	1111 11	xx

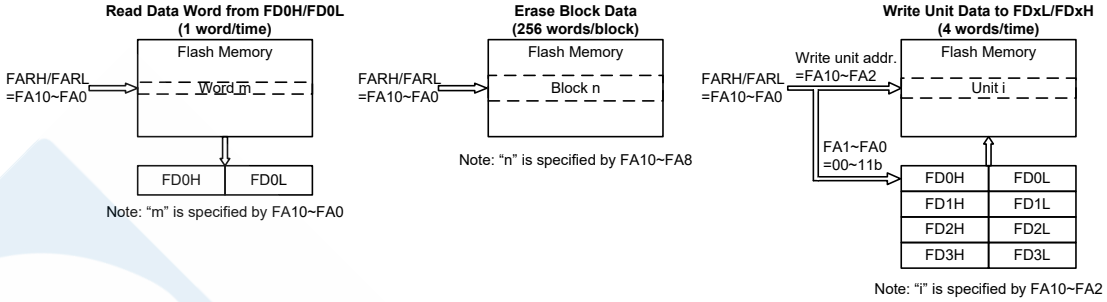
"x": don't care

Erase Block Number and Selection – BC66F2245

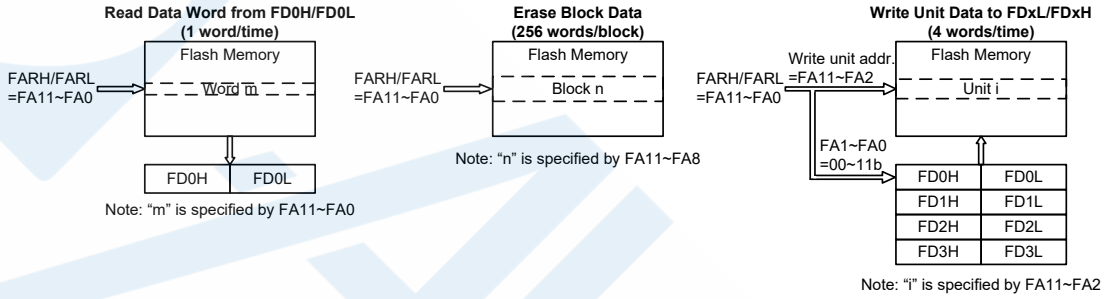
Erase Block	Write Unit	FARH[4:0]	FARL[7:2]	FARL[1:0]
0	0	00000	0000 00	xx
	1	00000	0000 01	xx
	:	:	:	:
	62	00000	1111 10	xx
	63	00000	1111 11	xx
1	64	00001	0000 00	xx
	65	00001	0000 01	xx
	:	:	:	:
	126	00001	1111 10	xx
	127	00001	1111 11	xx
:	:	:	:	:
31	1984	11111	0000 00	xx
	1985	11111	0000 01	xx
	:	:	:	:
	2046	11111	1111 10	xx
	2047	11111	1111 11	xx

"x": don't care

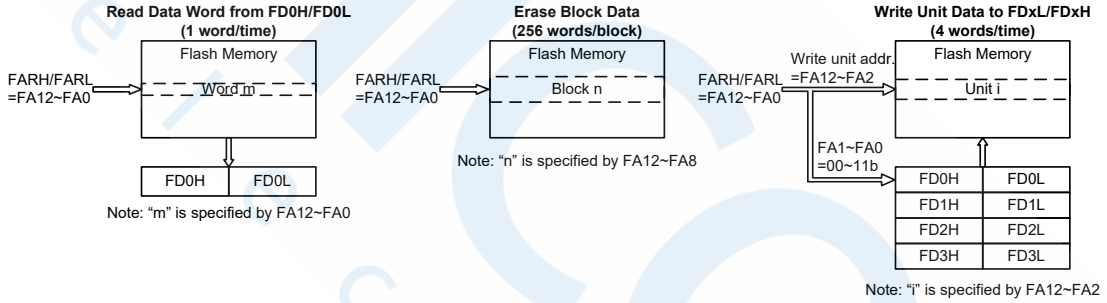
Erase Block Number and Selection – BC66F2255



Flash Memory IAP Read/Erase/Write Structure – BC66F2235



Flash Memory IAP Read/Erase/Write Structure – BC66F2245



Flash Memory IAP Read/Erase/Write Structure – BC66F2255

IAP Flash Program Memory Registers

There are two address registers, four 16-bit data registers and two control registers. Read and Write operations to the Flash memory are carried out using 16-bit data operations using the address and data registers and the control register. Several registers control the overall operation of the internal Flash Program Memory. The address registers are named FARL and FARH, the data registers are named FDnL and FDnH and the control registers are named FC0 and FC1. As the address, data register pairs and the control registers are located in Sector 1, they can be addressed directly only using the corresponding extended instructions or can be read from or written to indirectly using the MP1H/MP1L or MP2H/MP2L Memory Pointer pairs and Indirect Addressing Register, IAR1 or IAR2.

Register Name	Bit							
	7	6	5	4	3	2	1	0
FC0	CFWEN	FMOD2	FMOD1	FMOD0	FWPEN	FWT	FRDEN	FRD
FC1	D7	D6	D5	D4	D3	D2	D1	D0
FARL	FA7	FA6	FA5	FA4	FA3	FA2	FA1	FA0
FARH (BC66F2235)	—	—	—	—	—	FA10	FA9	FA8
FARH (BC66F2245)	—	—	—	—	FA11	FA10	FA9	FA8
FARH (BC66F2255)	—	—	—	FA12	FA11	FA10	FA9	FA8
FD0L	D7	D6	D5	D4	D3	D2	D1	D0
FD0H	D15	D14	D13	D12	D11	D10	D9	D8
FD1L	D7	D6	D5	D4	D3	D2	D1	D0
FD1H	D15	D14	D13	D12	D11	D10	D9	D8
FD2L	D7	D6	D5	D4	D3	D2	D1	D0
FD2H	D15	D14	D13	D12	D11	D10	D9	D8
FD3L	D7	D6	D5	D4	D3	D2	D1	D0
FD3H	D15	D14	D13	D12	D11	D10	D9	D8

IAP Register List

- **FC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	CFWEN	FMOD2	FMOD1	FMOD0	FWPEN	FWT	FRDEN	FRD
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 **CFWEN**: Flash Memory Erase/Write function enable control
0: Flash memory erase/write function is disabled
1: Flash memory erase/write function has been successfully enabled
When this bit is cleared to 0 by application program, the Flash memory erase/write function is disabled. Note that this bit cannot be set high by application programs. Writing a "1" into this bit results in no action. This bit is used to indicate the Flash memory erase/write function status. When this bit is set to 1 by the hardware, it means that the Flash memory erase/write function is enabled successfully. Otherwise, the Flash memory erase/write function is disabled if the bit is zero.

Bit 6~4 **FMOD2~FMOD0**: Flash memory Mode selection
000: Write Mode
001: Block Erase Mode
010: Reserved
011: Read Mode
100: Reserved
101: Reserved
110: Flash memory Erase/Write function Enable Mode
111: Reserved
These bits are used to select the Flash Memory operation modes. Note that the "Flash memory Erase/Write function Enable Mode" should first be successfully enabled before the Erase or Write Flash memory operation is executed.

Bit 3 **FWPEN**: Flash memory Erase/Write function enable procedure Trigger
0: Erase/Write function enable procedure is not triggered or procedure timer times out
1: Erase/Write function enable procedure is triggered and procedure timer starts to count

This bit is used to activate the Flash memory Erase/Write function enable procedure and an internal timer. It is set by the application programs and then cleared by the hardware when the internal timer times out. The correct patterns must be written into the FD1L/FD1H, FD2L/FD2H and FD3L/FD3H register pairs respectively as soon as possible after the FWPEN bit is set high.

Bit 2 **FWT**: Flash memory write initiate control
0: Do not initiate Flash memory write or indicating that a Flash memory write process has completed
1: Initiate Flash memory write process
This bit is set by software and cleared by the hardware when the Flash memory write process has completed.

Bit 1 **FRDEN**: Flash memory read enable control
0: Flash memory read disable
1: Flash memory read enable
This is the Flash memory Read Enable Bit which must be set high before any Flash memory read operations are carried out. Clearing this bit to zero will inhibit Flash memory read operations.

Bit 0 **FRD**: Flash memory read initiate control
0: Do not initiate Flash memory read or indicating that a Flash memory read process has completed
1: Initiate Flash memory read process
This bit is set by software and cleared by the hardware when the Flash memory read process has completed.

Note: 1. The FWT, FRDEN and FRD bits cannot be set to "1" at the same time with a single instruction.
2. Ensure that the f_{SUB} clock is stable before executing the erase or write operation.
3. Note that the CPU will be stopped when a read, write or erase operation is successfully activated.
4. Ensure that the read, erase or write operation is totally complete before executing other operations.

• **FC1 Register**

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

Bit 7~0 **D7~D0**: Chip Reset Pattern
When a specific value of "55H" is written into this register, a reset signal will be generated to reset the whole chip.

• **FARL Register**

Bit	7	6	5	4	3	2	1	0
Name	FA7	FA6	FA5	FA4	FA3	FA2	FA1	FA0
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 **FA7~FA0**: Flash Memory Address bit 7 ~ bit 0

• **FARH Register – BC66F2235**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	FA10	FA9	FA8
R/W	—	—	—	—	—	R/W	R/W	R/W
POR	—	—	—	—	—	0	0	0

Bit 7~3 Unimplemented, read as "0"

Bit 2~0 **FA10~FA8**: Flash Memory Address bit 10 ~ bit 8

• **FARH Register – BC66F2245**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	FA11	FA10	FA9	FA8
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 **FA11~FA8**: Flash Memory Address bit 11 ~ bit 8

• **FARH Register – BC66F2255**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	FA12	FA11	FA10	FA9	FA8
R/W	—	—	—	R/W	R/W	R/W	R/W	R/W
POR	—	—	—	0	0	0	0	0

Bit 7~5 Unimplemented, read as "0"

Bit 4~0 **FA12~FA8**: Flash Memory Address bit 12 ~ bit 8

• **FD0L Register**

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

Bit 7~0 **D7~D0**: The first Flash Memory data word bit 7 ~ bit 0

Note that data written into the low byte data register FD0L will only be stored in the FD0L register and not loaded into the lower 8-bit write buffer.

• **FD0H Register**

Bit	7	6	5	4	3	2	1	0
Name	D15	D14	D13	D12	D11	D10	D9	D8
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 **D15~D8**: The first Flash Memory data word bit 15 ~ bit 8

Note that when 8-bit data is written into the high byte data register FD0H, the whole 16 bits of data stored in the FD0H and FD0L registers will simultaneously be loaded into the 16-bit write buffer after which the contents of the Flash memory address register pair, FARH and FARL, will be incremented by one.

• **FD1L Register**

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

Bit 7~0 **D7~D0**: The second Flash Memory data word bit 7 ~ bit 0

• **FD1H Register**

Bit	7	6	5	4	3	2	1	0
Name	D15	D14	D13	D12	D11	D10	D9	D8
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 **D15~D8**: The second Flash Memory data word bit 15 ~ bit 8

• **FD2L Register**

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

Bit 7~0 **D7~D0**: The third Flash Memory data word bit 7 ~ bit 0

• **FD2H Register**

Bit	7	6	5	4	3	2	1	0
Name	D15	D14	D13	D12	D11	D10	D9	D8
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 **D15~D8**: The third Flash Memory data word bit 15 ~ bit 8

• **FD3L Register**

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

Bit 7~0 **D7~D0**: The fourth Flash Memory data word bit 7 ~ bit 0

• **FD3H Register**

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

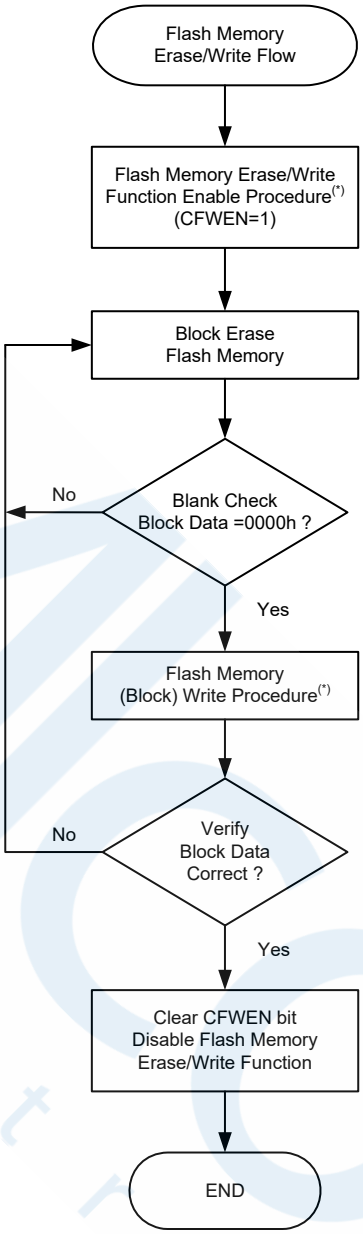
Bit 7~0 **D15~D8**: The fourth Flash Memory data word bit 15 ~ bit 8

Flash Memory Erase/Write Flow

It is important to understand the Flash memory Erase/Write flow before the Flash memory contents are updated. Users can refer to the corresponding operation procedures when developing their IAP program to ensure that the Flash memory contents are correctly updated.

Flash Memory Erase/Write Flow Descriptions

1. Activate the "Flash Memory Erase/Write function enable procedure" first. When the Flash Memory Erase/Write function is successfully enabled, the CFWEN bit in the FC0 register will automatically be set high by hardware. After this, Erase or Write operations can be executed on the Flash memory. Refer to the "Flash Memory Erase/Write Function Enable Procedure" for details.
2. Configure the Flash memory address to select the desired erase block and then erase this block.
3. Execute a Blank Check operation to ensure whether the block erase operation is successful or not. The "TABRD" instruction should be executed to read the Flash memory contents and to check if the contents is 0000h or not. If the Flash memory block erase operation fails, users should go back to Step 2 and execute the block erase operation again.
4. Write data into the specific block. Refer to the "Flash Memory Write Procedure" for details.
5. Execute the "TABRD" instruction to read the Flash memory contents and check if the written data is correct or not. If the data read from the Flash memory is different from the written data, it means that the block write operation has failed, go back to Step 2 and execute the block erase operation again.
6. the CFWEN bit to disable the Flash Memory Erase/Write function enable mode if the current block Erase and Write operations are complete if no more blocks need to be erased or written.



Flash Memory Erase/Write Flow

Note: The Flash Memory Erase/Write Function Enable procedure and Flash Memory Write procedure will be described in the following sections.

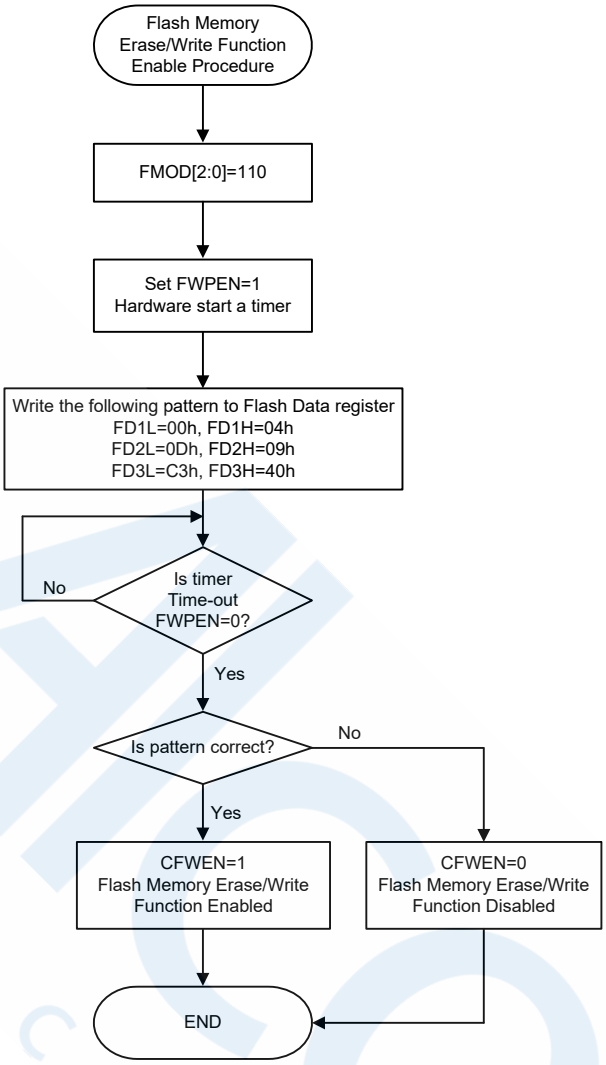
Flash Memory Erase/Write Function Enable Procedure

The Flash Memory Erase/Write Function Enable Mode is specially designed to prevent the Flash memory contents from being wrongly modified. In order to allow users to change the Flash memory data using the IAP control registers, users must first enable the Flash memory Erase/Write function.

Flash Memory Erase/Write Function Enable Procedure Description

1. Write data "110" to the FMOD [2:0] bits in the FC0 register to select the Flash Memory Erase/Write Function Enable Mode.
2. Set the FWPEN bit in the FC0 register to "1" to activate the Flash Memory Erase/Write Function. This will also activate an internal timer.
3. Write the correct data pattern into the Flash data registers, FD1L~FD3L and FD1H~FD3H, as soon as possible after the FWPEN bit is set high. The enable Flash memory erase/write function data pattern is 00H, 0DH, C3H, 04H, 09H and 40H corresponding to the FD1L~FD3L and FD1H~FD3H registers respectively.
4. Once the timer has timed out, the FWPEN bit will automatically be cleared to 0 by hardware regardless of the input data pattern.
5. If the written data pattern is incorrect, the Flash memory erase/write function will not be enabled successfully and the above steps should be repeated. If the written data pattern is correct, the Flash memory erase/write function will be enabled successfully.
6. Once the Flash memory erase/write function is enabled, the Flash memory contents can be updated by executing the block erase and write operations using the IAP control registers.

To disable the Flash memory erase/write function, the CFWEN bit in the FC0 register can be cleared. There is no need to execute the above procedure.



Flash Memory Erase/Write Function Enable Procedure

Flash Memory Write Procedure

After the Flash memory erase/write function has been successfully enabled as the CFWEN bit is set high, the data to be written into the flash memory can be filled into the data registers. The selected Flash memory block data should be erased by properly configuring the IAP control registers before the data write procedure is executed.

The block size is 256 words, whose address is specified by the memory address bits, FA10~FA8, FA11~FA8 or FA12~FA8.

Flash Memory Write Procedure Description

For each write operation the desired write unit address should first be placed in the FARL and FARH registers and the data placed in the FD0L/FD0H~FD3L/FD3H registers. The number of the write operation is 4 words each time, therefore, the available write unit address is only specified by the FA10~FA2, FA11~FA2 or FA12~FA2 bits in the FARH and FARL registers and the content of FA1~FA0 in the FARL register is not used to specify the unit address.

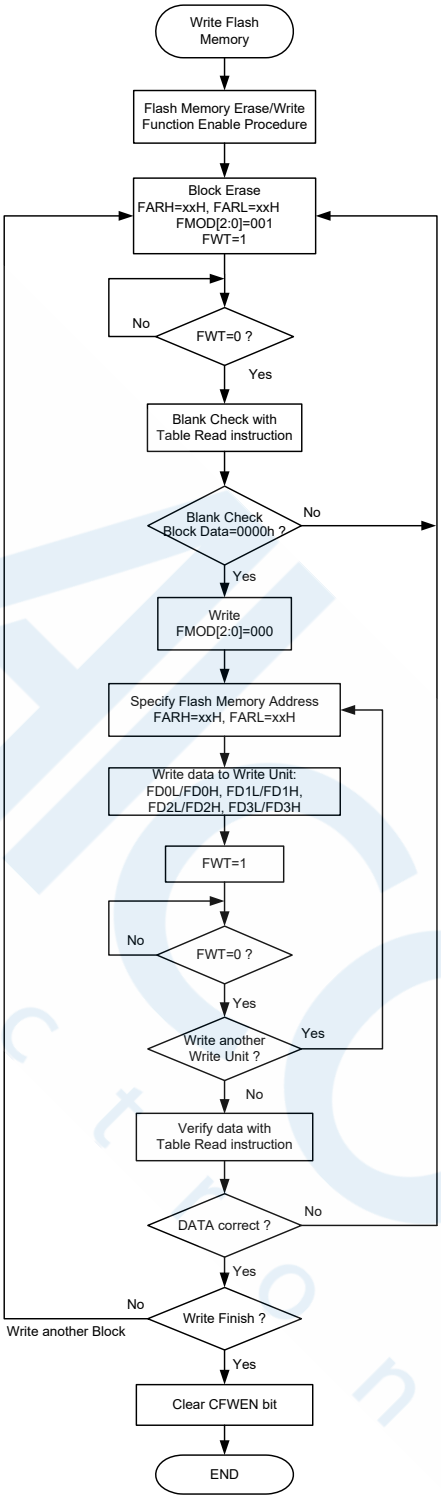
1. Activate the "Flash Memory Erase/Write function enable procedure". Check the CFWEN bit value and then execute the erase/write operations if the CFWEN bit is set high. Refer to the "Flash Memory Erase/Write function enable procedure" for more details.
2. Set the FMOD[2:0] bits to "001" to select the erase operation. Set the FWT bit high to erase the desired block which is specified by the FARH and FARL registers. Wait until the FWT bit goes low.
3. Execute a Blank Check operation using the table read instruction to ensure that the erase operation has successfully completed.

   Go to step 2 if the erase operation is not successful.

   Go to step 4 if the erase operation is successful.
4. Set the FMOD[2:0] bits to "000" to select the write operation.
5. Setup the desired start address in the FARH and FARL registers. Write the desired data words consecutively into the FD0L/FD0H~FD3L/FD3H registers.
6. Set the FWT bit high to write the data words to the Flash memory at four consecutive addresses starting from FARL[1:0]=00b. Wait until the FWT bit goes low.
7. Verify the data using the table read instruction to ensure that the write operation has successfully completed.

   If the write operation has not successfully completed, then go to step 2.

   Go to step 8 if the write operation is successful.
8. Clear the CFWEN bit low to disable the Flash memory erase/write function.



Flash Memory Write Procedure

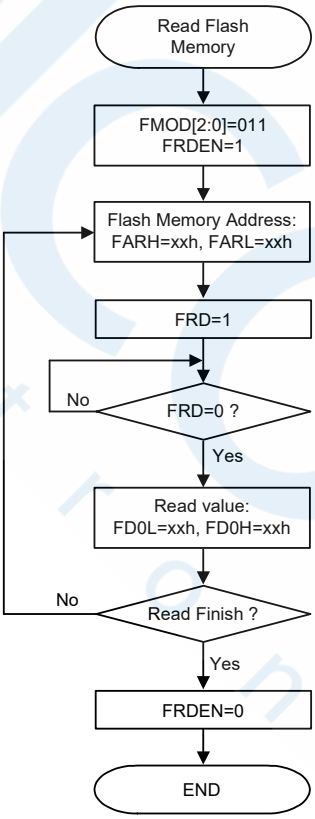
Note: 1. When the erase or write operation is successfully activated, all CPU operations will temporarily cease.
2. It will take a typical time of 2.2ms for the FWT bit state changing from high to low.
3. When the starting address specified by FARH/FARL is not a multiple of 4, the data cannot be correctly written to the four addresses starting from the specified starting address.

Important Points to Note for Flash Memory Write Operations

1. The "Flash Memory Erase/Write Function Enable Procedure" must be successfully activated before the Flash Memory erase/write operation is executed.
2. The Flash Memory erase operation is executed to erase a whole block.
3. After the data is written into the Flash memory the Flash memory contents must be read out using the table read instruction, TABRD, and checked if it is correct or not. If the data written into the Flash memory is incorrect, erase the block and then activate a write operation on the same Flash memory block. The data check, block erase and data re-write steps should be repeatedly executed until the data written into the Flash memory is correct.
4. The system frequency should be setup to the maximum application frequency when data write and data check operations are executed using the IAP function.

Flash Memory Read Procedure

To activate the Flash Memory Read procedure, the FMOD[2:0] bits should be set to "011" to select the Flash memory read mode and the FRDEN bit should be set high to enable the read function. The desired Flash memory address should be written into the FARH and FARL registers and then the FRD bit should be set high. After this the Flash memory read operation will be activated. The data stored in the specified address can be read from the data registers, FD0H and FD0L, when the FRD bit goes low. There is no need to first activate the Flash Memory Erase/Write Function Enable Procedure before the Flash memory read operation is executed.



Flash Memory Read Procedure

Note: 1. When the read operation is successfully activated, all CPU operations will temporarily cease.
2. It will take a typical time of three instruction cycles for the FRD bit state changing from high to low.

Data Memory

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

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

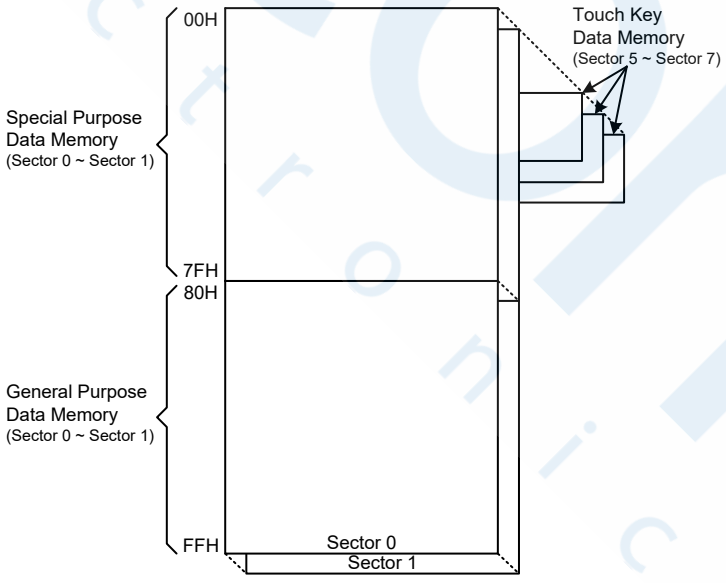
These devices also provide dedicated memory areas for the Touch Key data storage. In this chapter, only the General Purpose Data Memory and the Special Function Register Data Memory are introduced. More information about the Touch Key Data Memory can be obtained in its corresponding chapter.

Structure

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

Device	Special Purpose Data Memory	General Purpose Data Memory		Touch Key Data Memory	
	Located Sectors	Capacity	Sector: Address	Capacity	Sector: Address
BC66F2235	0, 1	256×8	0: 80H~FFH 1: 80H~FFH	96×8	Sector 5: 00H~1FH Sector 6: 00H~1FH Sector 7: 00H~1FH
BC66F2245					
BC66F2255					

Data Memory Summary



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 extended instructions 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 has 9 valid bits for these devices, the high byte indicates a sector and the low byte indicates a specific address.

General Purpose Data Memory

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

Special Purpose Data Memory

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

	Sector 0	Sector 1
00H	IAR0	
01H	MP0	
02H	IAR1	
03H	MP1L	
04H	MP1H	
05H	ACC	
06H	PCL	
07H	TBLP	
08H	TBLH	
09H	TBHP	
0AH	STATUS	
0BH		
0CH	IAR2	
0DH	MP2L	
0EH	MP2H	
0FH	RSTFC	
10H	INTC0	
11H	INTC1	
12H	INTC2	
13H		
14H	PA	
15H	PAC	
16H	PAPU	
17H	PAWU	
18H	PB	PC
19H	PBC	PCC
1AH	PBPU	PCPU
1BH	INTEG	
1CH	SCC	
1DH	HIRCC	
1EH	HXTC	
1FH		
20H	LVDC	SIMC0
21H	LVRC	SIMC1/UUCR1
22H	WDTC	SIMC2/SIMA/UUCR2
23H	RSTC	SIMD/UTXR_RXR
24H	PSCR0	SIMTOC/UBRG
25H	PSCR1	UUCR3
26H	PWRC	UUSR
27H	RF_PWR	
28H	MFI0	
29H	MFI1	
2AH	MFI2	
2BH	IFS	
2CH	SADC0	
2DH	SADC1	
2EH	SADOH	
2FH	SADOL	
30H	TB0C	
31H	TB1C	
32H	PTM0AH	
33H	PTM0AL	
34H	PTM0C0	
35H	PTM0C1	
36H	PTM0DH	
37H	PTM0DL	
38H	PTM0RPH	
39H	PTM0RPL	
3AH	CTM0C0	CTM1C0
3BH	CTM0C1	CTM1C1
3CH	CTM0DL	CTM1DL
3DH	CTM0DH	CTM1DH
3EH	CTM0AL	CTM1AL
3FH	CTM0AH	CTM1AH

	Sector 0	Sector 1
40H	PAS0	FC0
41H	PAS1	FC1
42H	PBS0	
43H	PBS1	FARL
44H	PCS0	FARH
45H	PCS1	FD0L
46H		FD0H
47H		FD1L
48H		FD1H
49H		FD2L
4AH		FD2H
4BH		FD3L
4CH		FD3H
4DH		
4EH		
4FH		
50H	RF_OPER	
51H	RF_CLK1	
52H	RF_CLK2	TKTMR
53H	RF_FIFO_CTRL1	TKC0
54H	RF_FIFO_CTRL2	TKC1
55H	RF_FIFO_CTRL3	TKC2
56H	RF_FIFO_CTRL4	TK16DL
57H	RF_MOD1	TK16DH
58H	RF_MOD2	TKM0C0
59H		TKM0C1
5AH	RF_MOD4	TKM0C2
5BH		TKM016DL
5CH		TKM016DH
5DH		TKM0ROL
5EH		TKM0ROH
5FH		TKM0TH16L
60H	RF_OPMOD	TKM0TH16H
61H	RF_SX1	TKM0THS
62H	RF_SX2	TKM1C0
63H	RF_SX3	TKM1C1
64H	RF_SX4	TKM1C2
65H		TKM116DL
66H		TKM116DH
67H	RF_CP3	TKM1ROL
68H	RF_OD1	TKM1ROH
69H		TKM1TH16L
6AH		TKM1TH16H
6BH	RF_VCO1	TKM1THS
6CH		TKM2C0
6DH		TKM2C1
6EH	RF_TX1	TKM2C2
6FH	RF_TX2	TKM216DL
70H	RF_DFC_CAL	TKM216DH
71H	RF_LDO	TKM2ROL
72H	RF_XO1	TKM2ROH
73H		TKM2TH16L
74H		TKM2TH16H
75H		TKM2THS
76H		TKM3C0
77H		TKM3C1
78H		TKM3C2
79H		TKM316DL
7AH		TKM316DH
7BH		TKM3ROL
7CH		TKM3ROH
7DH		TKM3TH16L
7EH		TKM3TH16H
7FH		TKM3THS

: Unused, read as 00H

: Reserved, cannot be changed

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 only from Sector 0 while the IAR1 register together with the MP1L/MP1H register pair and IAR2 register together with the MP2L/MP2H register pair can access data from any Data Memory Sector. As the Indirect Addressing Registers are not physically implemented, reading the Indirect Addressing Registers will return a result of "00H" and writing to the registers will result in no operation.

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

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

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

Indirect Addressing Program Example 1

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

Indirect Addressing Program Example 2

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

The important point to note here is that in the examples 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 SC flag, CZ flag, zero flag (Z), carry flag (C), auxiliary carry flag (AC), overflow flag (OV), power down flag (PDF), and watchdog time-out flag (TO). These arithmetic/logical operation and system management flags are used to record the status and operation of the microcontroller.

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

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

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

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

• **STATUS Register**

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

"x": unknown

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

Bit 6 **CZ**: The operational result of different flags for different instructions
For SUB/SUBM/LSUB/LSUBM instructions, the CZ flag is equal to the Z flag.
For SBC/SBCM/LSBC/LSBCM instructions, the CZ flag is the "AND" operation result which is performed by the previous operation CZ flag and current operation zero flag.
For other instructions, the CZ flag will not be affected.

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

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

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

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

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

Bit 0 **C**: Carry flag
0: No carry-out
1: An operation results in a carry during an addition operation or if a borrow does not take place during a subtraction operation
The "C" flag is also affected by a rotate through carry instruction.

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 selection and operation are selected through the relevant control registers.

Oscillator Overview

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

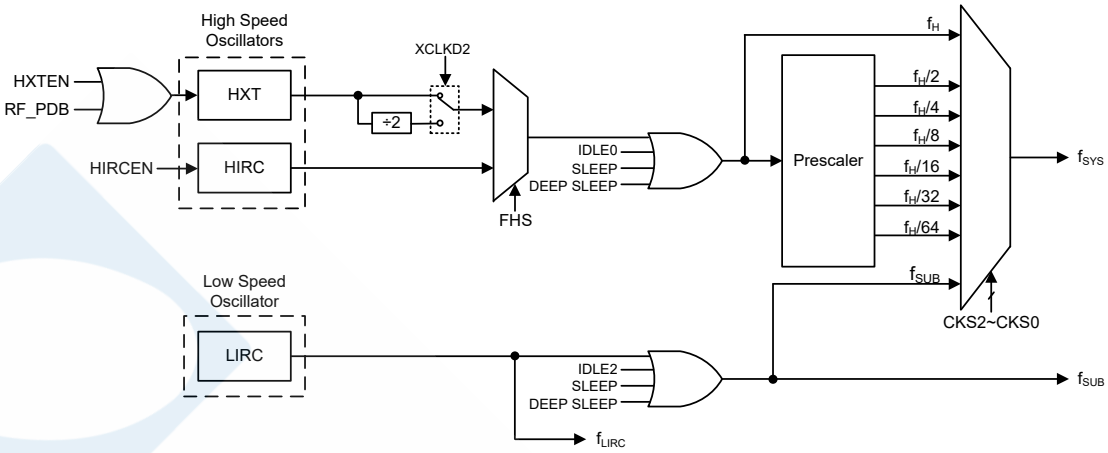
Type	Name	Frequency	Pins
External High Speed Crystal	HXT	16MHz	OSC1/OSC2
Internal High Speed RC	HIRC	8MHz	—
Internal Low Speed RC	LIRC	32kHz	—

Oscillator Types

System Clock Configurations

There are several oscillator sources, two high speed oscillators and one low speed oscillator. The high speed system clocks are sourced from an external crystal/ceramic oscillator, HXT, and an internal 8MHz RC oscillator, HIRC. The low speed oscillator is the internal 32kHz RC oscillator, LIRC. Selecting whether the low or high speed oscillator is used as the system oscillator is implemented using the CKS2~CKS0 bits in the SCC register and as the system clock can be dynamically selected.

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

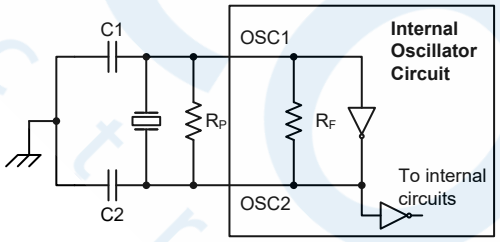


System Clock Configurations

External Crystal/Ceramic Oscillator – HXT

The External Crystal/Ceramic System Oscillator is one of the high frequency oscillators. It is also used as the clock source for the RF circuitry. For most crystal oscillator configurations, the simple connection of a crystal across OSC1 and OSC2 will create the necessary phase shift and feedback for oscillation, without requiring external capacitors. However, for some crystal types and frequencies, to ensure oscillation, it may be necessary to add two small value capacitors, C1 and C2. Using a ceramic resonator will usually require two small value capacitors, C1 and C2, to be connected as shown for oscillation to occur. The values of C1 and C2 should be selected in consultation with the crystal or resonator manufacturer's specification.

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



Note: 1. R_P is normally not required. C1 and C2 are required.
2. The capacitor load is internally integrated and determined by the RF_XO1 register.

Crystal/Resonator Oscillator

Crystal Oscillator C1 and C2 Values		
Crystal Frequency	**C1**	**C2**
16MHz	0pF	0pF
Note: C1 and C2 values are for guidance only.		

Crystal Recommended Capacitor Values

Internal High Speed RC Oscillator – HIRC

The internal RC oscillator is a fully integrated system oscillator requiring no external components. The internal RC oscillator has a fixed frequency of 8MHz. 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 that if this internal system clock option is selected, it requires no external pins for its operation.

Internal 32kHz Oscillator – LIRC

The Internal 32kHz System Oscillator is the low frequency oscillator. It is a fully integrated RC oscillator with a typical frequency of 32kHz at 3V, requiring no external components for its implementation. Device trimming during the manufacturing process and the inclusion of internal frequency compensation circuits are used to ensure that the influence of the power supply voltage, temperature and process variations on the oscillation frequency are minimised.

Operating Modes and System Clocks

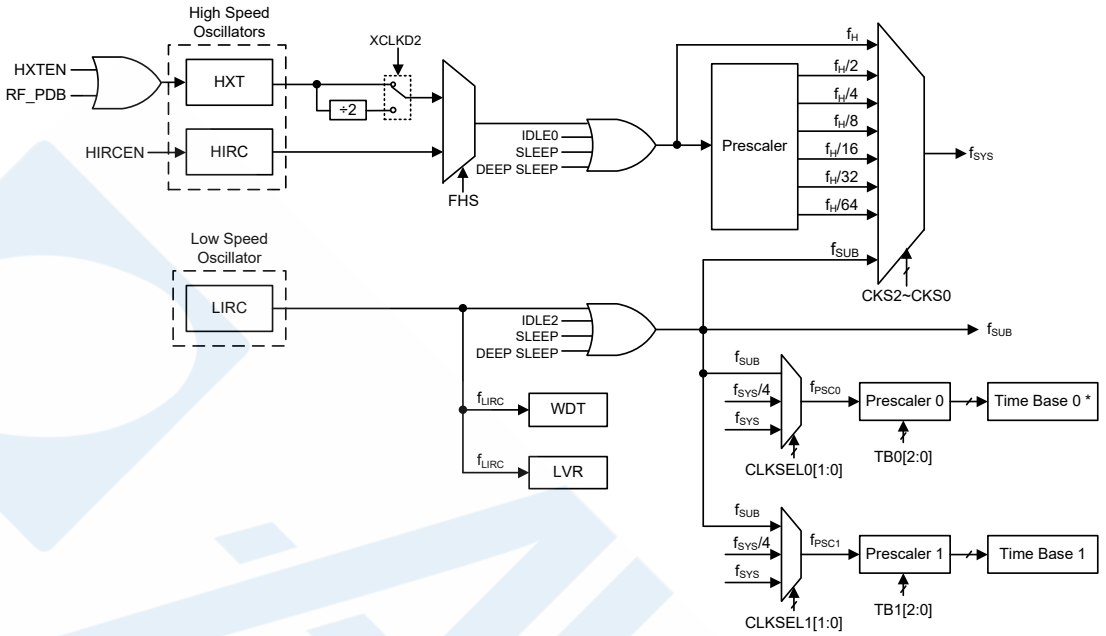
Present day applications require that their microcontrollers have high performance but often still demand that they consume as little power as possible, conflicting requirements that are especially true in battery powered portable applications. The fast clocks required for high performance will by their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

System Clocks

These devices have many different clock sources for both the CPU and peripheral function operation. By providing the user with a wide range of clock options using register programming, a clock system can be configured to obtain maximum application performance.

The main system clock, can come from a high frequency, f_H, or low frequency, f_{SUB}, source, and is selected using the CKS2~CKS0 bits in the SCC register. The high speed system clock can be sourced from either the HXT or HIRC oscillator, selected via configuring the FHS bit in the SCC register. The HXT oscillator is enabled using the HXTEN and RF_PDB bits. Its frequency can be divided by two using the XCLKD2 bit. These latter two bits are described in the RF register section.

The low speed system clock source can be sourced from the internal clock f_{SUB}. If f_{SUB} is selected then it is sourced from the LIRC oscillator. The other choice, which is a divided version of the high speed system oscillator has a range of $f_H/2$~$f_H/64$.



Device Clock Configurations

Note: 1. When the system clock source f_{SYS} is switched to f_{SUB} from f_H, the high speed oscillator will stop to conserve the power or continue to oscillate to provide the clock source, f_H~f_H/64, for peripheral circuit to use, which is determined by configuring the corresponding high speed oscillator enable control bit.

2. When the system enters the DEEP SLEEP Mode, the Time Base 0 clock source is f_{LIRC}, which is determined by the WDT enable/disable status.

System Operation Modes

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

Operation Mode	LDO	CPU	Register Setting				f_{SYS}	f_H	f_{SUB}	f_{LIRC}	f_{HXT}
			PWDN	FHIDEN	FSIDEN	CKS2~CKS0					
FAST	On	On	0	x	x	000~110	f_H~f_H/64	On	On	On	On/Off (3)
SLOW	On	On	0	x	x	111	f_{SUB}	On/Off (1)	On	On	On/Off (3)
IDLE0	On	Off	0	0	1	000~110	Off	Off	On	On	On/Off (3)
						111	On				
IDLE1	On	Off	0	1	1	xxx	On	On	On	On	On/Off (3)
IDLE2	On	Off	0	1	0	000~110	On	On	Off	On/Off (2)	On/Off (3)
						111	Off				
SLEEP	On	Off	0	0	0	xxx	Off	Off	Off	On/Off (2)	On/Off (3)
DEEP SLEEP	Off	Off	1	0	0	xxx	Off	Off	Off	On/Off (2)	Off

"x": Don't care

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

2. The f_{LIRC} clock can be switched on or off which is controlled by the WDT function being enabled or disabled in the IDEL2, SLEEP or DEEP SLEEP mode. However, the WDT is no function in the DEEP SLEEP mode.
3. The f_{HXT} can be switched on or off by the HXTEN bit in the HXTC control register in other modes except the DEEP SLEEP mode. When the HXTEN bit is set to 1, the f_{HXT} will be turned on and provide clock source for the RF transmitter, Time Bases, etc.
4. When the MCU enters the DEEP SLEEP mode, the Time Base 0 clock source will come from f_{LIRC} (WDTC[7:3]≠10101b). After the MCU wakes up from the DEEP SLEEP mode, the Time Base 0 clock source will come from f_{PSC0} (INTC0 will be reset).
5. After the MCU wakes up from the DEEP SLEEP mode, most registers will be reset. Therefore, the system settings should be restored using the application program.

FAST Mode

This is one of the main operating modes where the microcontroller has all of its functions operational and where the system clock is provided by one of the high speed oscillators and the 1.5V LDO is turned on with the PWDN bit in the PWRC register being low. This mode operates allowing the microcontroller to operate normally with a clock source sourced from one of the high speed oscillators, either the HXT or HIRC oscillators. The high speed oscillator will however first be divided by a ratio ranging from 1 to 64, the actual ratio being selected by the CKS2~CKS0 bits in the SCC register. Although a high speed oscillator is used, running the microcontroller at a divided clock ratio reduces the operating current.

SLOW Mode

This is also a mode where the microcontroller operates normally although now with a slower speed clock source and where the 1.5V LDO is turned on with the PWDN bit in the PWRC register being low. The clock source used will be from f_{SUB}. The f_{SUB} clock is derived from the LIRC oscillator.

SLEEP Mode

The SLEEP Mode is entered when an HALT instruction is executed and when the FHIDEN and FSIDEN bits are low and the PWDN bit in the PWRC register is low. In the SLEEP mode the CPU will be stopped. The f_{SUB} clock provided to the peripheral function will also be stopped, too. However the f_{LIRC} clock can continues to operate if the WDT function is enabled.

IDLE0 Mode

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

IDLE1 Mode

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

IDLE2 Mode

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

DEEP SLEEP Mode

The DEEP SLEEP Mode is entered when an HALT instruction is executed and when the PWDN bit in the PWRC register is high. In the DEEP SLEEP Mode the CPU will be stopped. The f_{SUB} clock provided to the peripheral function will also be stopped. If the WDT function is enabled, the f_{LIRC} clock can continue to operate. However, the WDT is no function in the DEEP SLEEP mode irrespective of whether the WDT function is enabled or not.

Control Registers

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

Register Name	Bit							
	7	6	5	4	3	2	1	0
SCC	CKS2	CKS1	CKS0	—	FHS	D2	FHIDEN	FSIDEN
HIRCC	—	—	—	—	—	—	HIRCF	HIRCEN
HXTC	—	—	—	—	—	—	HXTF	HXTEN
PWRC	PA_WAKE	TK_WAKE	—	TB0_WAKE	POF33V	LCMD	IO_ISO_EN	PWDN

System Operating Mode Control Register List

• SCC Register

Bit	7	6	5	4	3	2	1	0
Name	CKS2	CKS1	CKS0	—	FHS	D2	FHIDEN	FSIDEN
R/W	R/W	R/W	R/W	—	R/W	R/W	R/W	R/W
POR	0	0	1	—	0	0	0	0

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

Bit 4 Unimplemented, read as "0"

Bit 3 **FHS**: High Frequency clock selection
0: HIRC – internal high frequency oscillator
1: HXT – external crystal
When this bit is set high to select the external HXT oscillator as the system clock source, the actual clock input to the MCU is determined by the external crystal connected to the OSC1 and OSC2 pins together with the XCLKD2 bit in the RF_CLK1 register. For example, if f_{HXT}=16MHz and XCLKD2=0, then f_H=16MHz.

Bit 2 **D2**: Reserved bit

Bit 1 **FHIDEN**: High Frequency oscillator control when the CPU is switched off
0: Disable
1: Enable
This bit is used to control whether the high speed oscillator is activated or stopped when the CPU is switched off by executing an "HALT" instruction.

Bit 0 **FSIDEN**: Low Frequency oscillator control when CPU is switched off
0: Disable
1: Enable
This bit is used to control whether the low speed oscillator is activated or stopped when the CPU is switched off by executing an "HALT" instruction.

Note: A certain delay is required before the relevant clock is successfully switched to the target clock source after any clock switching setup using the CKS2~CKS0 bits or FHS bit. A proper delay time must be arranged before executing the following operations which require immediate reaction with the target clock source.

Clock switching delay time=$4\times t_{SYS}+[0\sim(1.5\times t_{Curr.}+0.5\times t_{Tar.})]$, where $t_{Curr.}$ indicates the current clock period, $t_{Tar.}$ indicates the target clock period and t_{SYS} indicates the current system clock period.

• **HIRCC Register**

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

Bit 7~2 Unimplemented, read as "0"

Bit 1 **HIRCF**: HIRC oscillator stable flag
0: HIRC unstable
1: HIRC stable
This bit is used to indicate whether the HIRC oscillator is stable or not. When the HIRCEN bit is set to 1 to enable the HIRC oscillator, the HIRCF bit will first be cleared to 0 and then set to 1 after the HIRC oscillator is stable.

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

• **HXTC Register**

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

Bit 7~2 Unimplemented, read as "0"

Bit 1 **HXTF**: HXT oscillator stable flag
0: HXT unstable
1: HXT stable
This bit is used to indicate whether the HXT oscillator is stable or not. When the HXTEN bit is set to 1 to enable the HXT oscillator, the HXTF bit will first be cleared to 0 and then set to 1 after the HXT oscillator is stable.

Bit 0 **HXTEN**: HXT oscillator enable control
0: Disable
1: Enable

• **PWRC Register**

Bit	7	6	5	4	3	2	1	0
Name	PA_WAKE	TK_WAKE	—	TB0_WAKE	POF33V	LCMD	IO_ISO_EN	PWDN
R/W	R/W	R/W	—	R/W	R/W	R/W	R/W	R/W
POR	0	0	—	0	1	0	0	0

Bit 7 **PA_WAKE**: Port A wake-up MCU from DEEP SLEEP Mode flag
0: No Port A wake-up MCU from DEEP SLEEP Mode occurred
1: Port A wake-up MCU from DEEP SLEEP Mode occurred
This bit indicates whether the MCU has been woken up from the DEEP SLEEP Mode by the port A wake-up function. The PAWU register should be properly configured together with the PA_WAKE bit being cleared before the MCU enters the DEEP SLEEP Mode.

Bit 6 **TK_WAKE**: Touch key wake-up MCU from DEEP SLEEP Mode flag
0: No Touch key wake-up MCU from DEEP SLEEP Mode occurred
1: Touch key wake-up MCU from DEEP SLEEP Mode occurred
This bit indicates whether the MCU has been woken up from the DEEP SLEEP Mode by the touch key function. The touch key related registers should be properly configured together with the TK_WAKE bit being cleared before the MCU enters the DEEP SLEEP Mode.

Bit 5 Unimplemented, read as "0"

Bit 4 **TB0_WAKE**: Time Base 0 wake-up MCU from DEEP SLEEP Mode flag
0: No Time Base 0 wake-up MCU from DEEP SLEEP Mode occurred
1: Time Base 0 wake-up MCU from DEEP SLEEP Mode occurred
This bit indicates whether the MCU has been woken up from the DEEP SLEEP Mode by the Time Base 0 interrupt. The TB0C register should be properly configured together with the TB0_WAKE bit being cleared before the MCU enters the DEEP SLEEP Mode.

Bit 3 **POF33V**: 3.3V power domain power on reset flag
0: No 3.3V power domain power on reset occurred
1: 3.3V power domain power on reset occurred
This bit is set high by hardware when a 3.3V power domain power on reset occurs. It is cleared by application program or when the "CLR WDT" instruction is executed.

Bit 2 **LCMD**: 1.5V LDO low current mode selection
0: 1.5V LDO normal mode
1: 1.5V LDO low current mode

Bit 1 **IO_ISO_EN**: I/O isolation mode selection
0: I/O in normal operation mode
1: I/O in isolation mode (I/O status remains unchanged)
Before entering the DEEP SLEEP Mode this bit must be set high to latch I/O ports, so that the I/O port status will remain unchanged when in the DEEP SLEEP Mode. After the MCU is woken up from the DEEP SLEEP Mode, this bit must be cleared using application program to de-latch I/O ports.

Bit 0 **PWDN**: DEEP SLEEP Mode control
0: MCU under normal operation
1: MCU enters the DEEP SLEEP Mode after HALT (1.5V LDO off by hardware)
If this bit is set high, after executing the HALT instruction, the MCU will enter the DEEP SLEEP Mode, in which case the 1.5V LDO will be turned off by hardware. Before entering the DEEP SLEEP Mode, users should previously store the 1.5V domain system settings to Data Memory and latch the I/O ports using the application program.

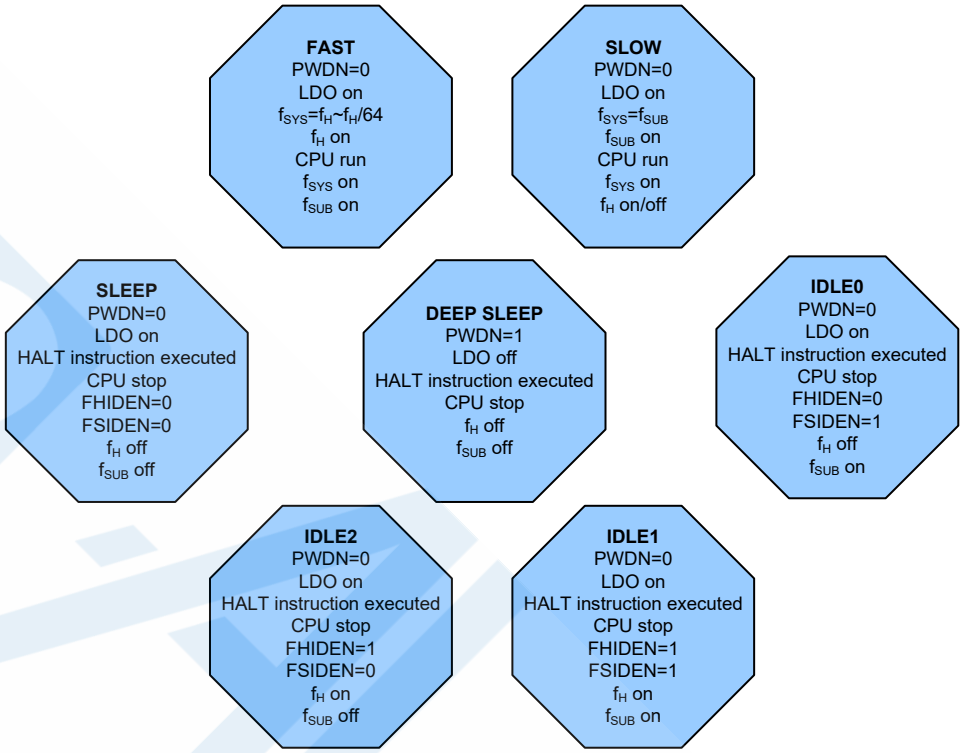
There are several important notes should be emphasized:

1. Before the MCU enters the DEEP SLEEP mode, the system settings should be backed up using the application program in advance.
2. When PWDN=1 and the HALT instruction is executed, the MCU will enter the DEEP SLEEP mode.
3. When the MCU enters the DEEP SLEEP mode, the Data Memory will keep the latest values.
4. When the MCU enters the DEEP SLEEP mode, the RF_PDB bit in the RF_PWR register will be cleared to zero by hardware.
5. The following conditions will wake up the MCU from the DEEP SLEEP mode and the program will resume execution from 0000h.
   - An external falling edge on Port A
   - Touch key
   - Time Base 0 Interrupt
6. To wake up the MCU from the DEEP SLEEP mode by the PA port, the PAWU register should be properly configured together with the PA_WAKE bit being cleared before the MCU enters the DEEP SLEEP mode.
7. To wake up the MCU from the DEEP SLEEP mode by the TB0 interrupt, the TB0C register should be properly configured together with the TB0_WAKE bit being cleared before the MCU enters the DEEP SLEEP mode. The TB0 interrupt wake-up function is irrespective of whether the interrupt is enabled or not.
8. To wake up the MCU from the DEEP SLEEP mode by the touch key function, the touch key related registers should be properly configured together with the TK_WAKE bit being cleared before the MCU enters the DEEP SLEEP Mode.
9. When the MCU enters the DEEP SLEEP mode, the TB0 clock source will come from f_{LIRC} irrespective of the Prescaler 0 clock source selection.
10. After the MCU wakes up from the DEEP SLEEP mode, all registers will be reset. Therefore the system settings should be restored using the application program.

Operating Mode Switching

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

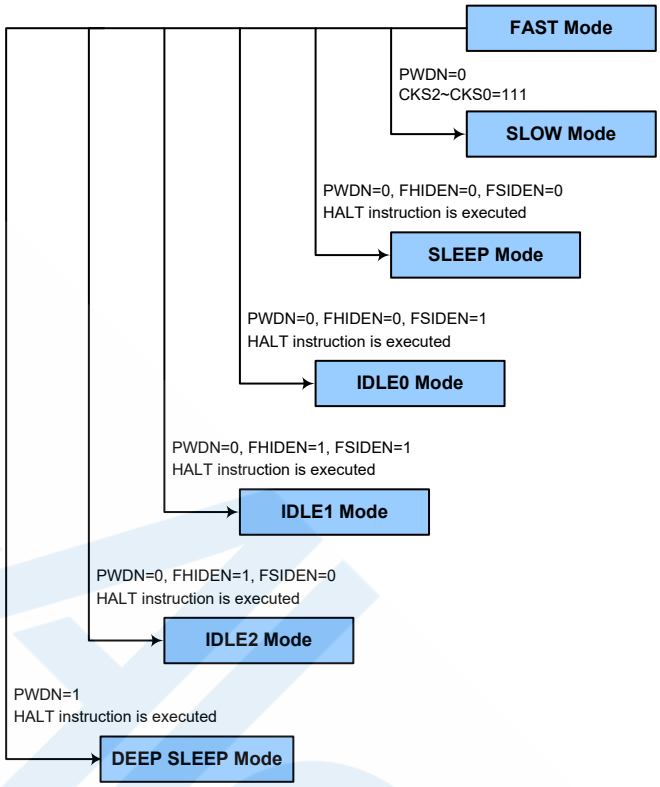
In simple terms, Mode Switching between the FAST Mode and SLOW Mode is executed using the CKS2~CKS0 bits in the SCC register while Mode Switching from the FAST/SLOW Modes to the IDLE/SLEEP/DEEP SLEEP Modes is executed via the HALT instruction. When an HALT instruction is executed, whether these devices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



FAST Mode to SLOW Mode Switching

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

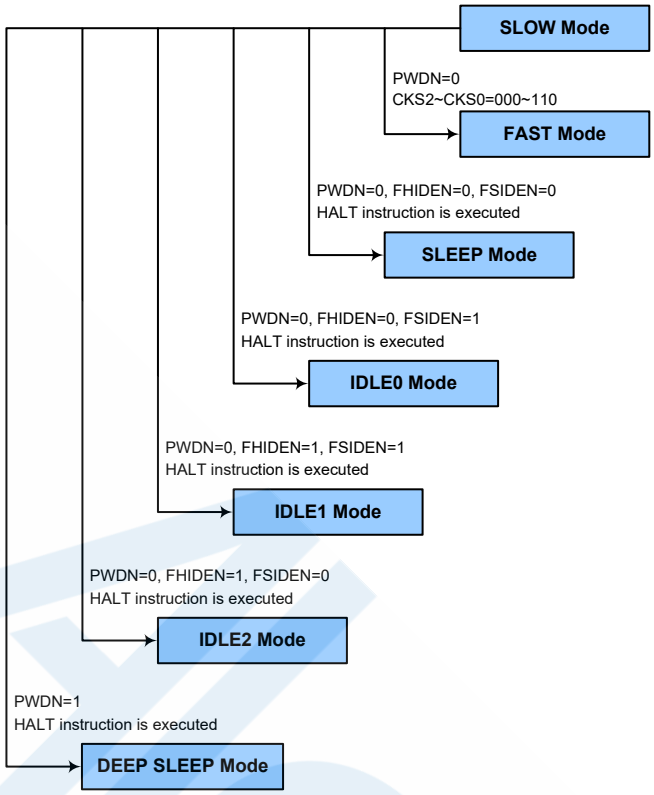
The SLOW Mode is sourced from the LIRC oscillator and therefore requires this oscillator to be stable before full mode switching occurs.



SLOW Mode to FAST Mode Switching

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

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



Entering the SLEEP Mode

There is only one way for these devices to enter the SLEEP Mode and that is to execute the "HALT" instruction in the application program with the PWDN bit in the PWRC register equal to "0" and both the FHIDEN and FSIDEN bits in the SCC 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.
- The Data Memory contents and registers will maintain their present condition.
- The I/O ports will maintain their present conditions.
- In the status register, the Power Down flag, PDF, will be set and the Watchdog time-out flag, TO, will be cleared.
- The WDT will be cleared and resume counting if the WDT function is enabled. If the WDT function is disabled, the WDT will be cleared and then stopped.

Entering the IDLE0 Mode

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

- The f_H clock will be stopped and the application program will stop at the "HALT" instruction, but the f_{SUB} clock will be on.
- The Data Memory contents and registers will maintain their present condition.
- The I/O ports will maintain their present conditions.

- In the status register, the Power Down flag, PDF, will be set and the Watchdog time-out flag, TO, will be cleared.
- The WDT will be cleared and resume counting if the WDT function is enabled. If the WDT function is disabled, the WDT will be cleared and then stopped.

Entering the IDLE1 Mode

There is only one way for these devices to enter the IDLE1 Mode and that is to execute the "HALT" instruction in the application program with the PWDN bit in the PWRC register equal to "0" and both the FHIDEN and FSIDEN bits in the SCC register equal to "1". When this instruction is executed under the conditions described above, the following will occur:

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

Entering the IDLE2 Mode

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

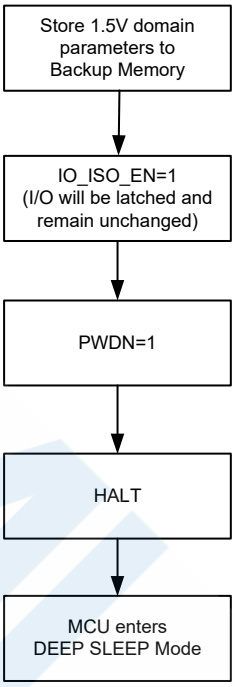
- The f_H clock will be on but the f_{SUB} clock will be off and the application program will stop at the "HALT" instruction.
- The Data Memory contents and registers will maintain their present condition.
- The I/O ports will maintain their present conditions.
- In the status register, the Power Down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

Entering the DEEP SLEEP Mode

There is only one way for these devices to enter the DEEP SLEEP Mode and that is to execute the "HALT" instruction in the application program with the PWDN bit in the PWRC register equal to "1". When this instruction is executed under the conditions described above, the following will occur:

- The system clock will be stopped and the application program will stop at the "HALT" instruction.
- The Data Memory contents and registers will maintain their present condition. It is recommended to previously store the important 1.5V domain system settings in the Data Memory.
- The I/O ports will maintain their present conditions if the IO_ISO_EN bit in the PWRC register is set high.
- In the status register, the Power Down flag, PDF, will be set and the Watchdog time-out flag, TO, will be cleared.
- The WDT will be cleared and then stopped irrespective of whether the WDT function is enabled or not.

The accompany flowchart shows the program procedure for entering the DEEP SLEEP Mode.



Standby Current Considerations

As the main reason for entering the DEEP SLEEP, SLEEP or IDLE Mode is to keep the current consumption of these devices to as low a value as possible, perhaps only in the order of several micro-amps except in the IDLE1 and IDLE2 Mode, there are other considerations which must also be taken into account by the circuit designer if the power consumption is to be minimised. Special attention must be made to the I/O pins on these devices. All high-impedance input pins must be connected to either a fixed high or low level as any floating input pins could create internal oscillations and result in increased current consumption. This also applies to these devices which have different package types, as there may be unbonded pins. These must either be setup as outputs or if setup as inputs must have pull-high resistors connected.

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

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

Wake-up

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

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

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

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

- An external falling edge on Port A
- A Time Base 0 interrupt
- Touch key

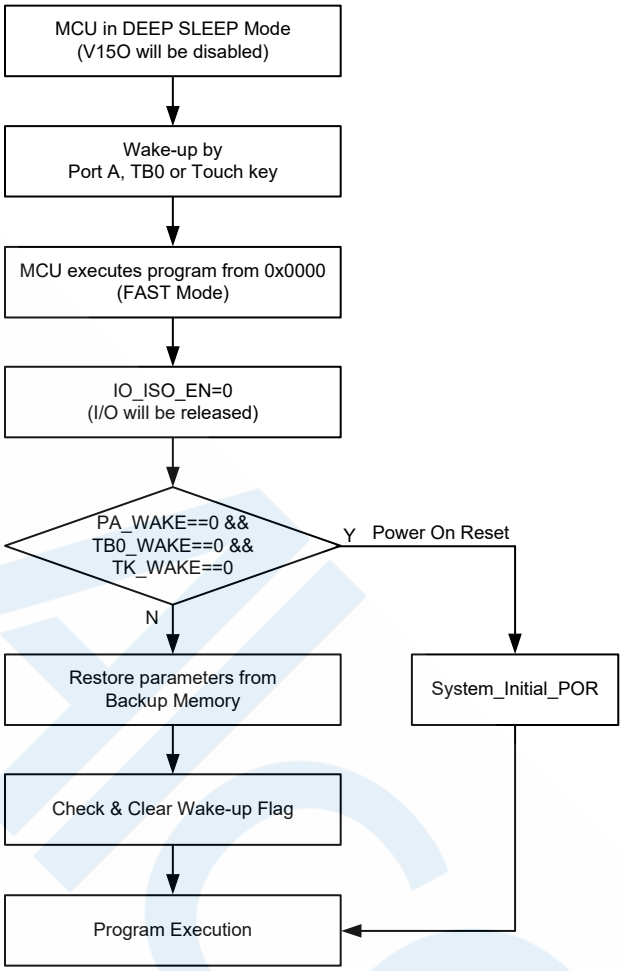
When these devices execute the "HALT" instruction, the PDF flag will be set to 1. The PDF flag will be cleared to 0 if these devices experience a system power-up or executes the clear Watchdog Timer instruction. If the system is woken up by a WDT overflow, a Watchdog Timer reset will be initiated and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

Each pin on Port A can be setup using the PAWU register to permit a negative transition on the pin to wake-up the system. When a Port A pin wakes up the system from the SLEEP or IDLE Mode, the program will resume execution at the instruction following the "HALT" instruction. However, when a Port A pin wakes up the system from the DEEP SLEEP Mode, the program will resume execution from 0000h.

If the system is woken up from the SLEEP or IDLE Mode 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 these devices will not be immediately serviced, but will rather be serviced later when the related interrupt is finally enabled or when a stack level becomes free. The other situation is where the related interrupt is enabled and the stack is not full, in which case the regular interrupt response takes place. If the system is woken up from the DEEP SLEEP Mode by the Time Base 0 interrupt, the program will resume execution from 0000h.

If an interrupt request flag is set high before entering the DEEP SLEEP, SLEEP or IDLE Mode, the wake-up function of the related interrupt will be disabled.

The accompany flowchart shows the program procedure for waking up from the DEEP SLEEP Mode.


MCU in DEEP SLEEP Mode
(V15O will be disabled)
Wake-up by
Port A, TB0 or Touch key
MCU executes program from 0x0000
(FAST Mode)
IO_ISO_EN=0
(I/O will be released)
PA_WAKE==0 &&
TB0_WAKE==0 &&
TK_WAKE==0
Y
Power On Reset
N
Restore parameters from
Backup Memory
System_Initial_POR
Check & Clear Wake-up Flag
Program Execution

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

Watchdog Timer Control Register

A single register, WDTC, controls the required time-out period as well as the WDT enable/disable and software reset MCU operation.

• **WDTC Register**

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

Bit 7~3 **WE4~WE0**: WDT function software control

If the WDT is always enabled determined by the configuration option:
10101: Enable
01010: Enable
Other values: Reset MCU

If the WDT is controlled by the WDTC register determined by the configuration option:
10101: Disable
01010: Enable
Other values: Reset MCU

The WE4~WE0 bits setup is determined by the WDT configuration option selection.

When these bits are changed to any other values except 10101B and 01010B due to environmental noise the microcontroller will be reset, this reset operation will be activated after a delay time, t_{SRESET}, and the WRF bit in the RSTFC register will be set high.

Note that the WDT is no function in the DEEP SLEEP mode irrespective of whether the WDT function is enabled or not.

Bit 2~0 **WS2~WS0**: WDT time-out period selection

000: $2^8/f_{LIRC}$
001: $2^{10}/f_{LIRC}$
010: $2^{12}/f_{LIRC}$
011: $2^{14}/f_{LIRC}$
100: $2^{15}/f_{LIRC}$
101: $2^{16}/f_{LIRC}$
110: $2^{17}/f_{LIRC}$
111: $2^{18}/f_{LIRC}$

These three bits determine the division ratio of the Watchdog Timer source clock, which in turn determines the timeout period.

• **RSTFC Register**

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

"x": unknown

Bit 7~4 Unimplemented, read as "0"

Bit 3 **RSTF**: Reset control register software reset flag
Refer to the Internal Reset Control section.

Bit 2 **LVRF**: LVR function reset flag
Refer to the Low Voltage Reset section.

Bit 1 **LRF**: LVR control register software reset flag
Refer to the Low Voltage Reset section.

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

Watchdog Timer Operation

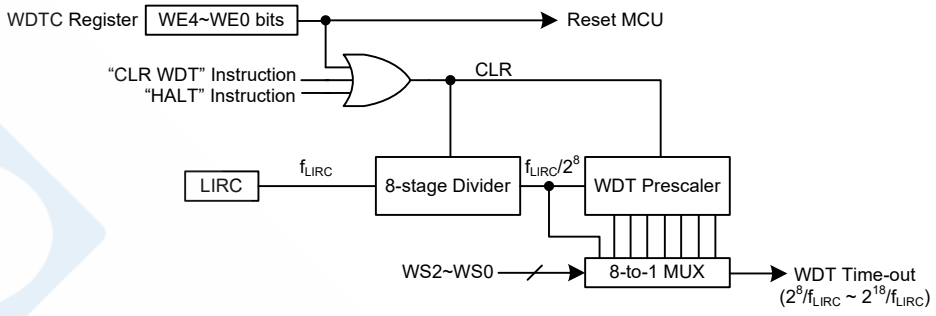
The Watchdog Timer operates by providing a device reset when its timer overflows. This means that in the application program and during normal operation the user has to strategically clear the Watchdog Timer before it overflows to prevent the Watchdog Timer from executing a reset. This is done using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

Configuration Option	WE4~WE0 Bits	WDT Function
WDT always enabled	10101B or 01010B	Enable
	Any other value	Reset MCU
WDT controlled by WDTC	10101B	Disable
	01010B	Enable
	Any other value	Reset MCU

Watchdog Timer Function Control

Under normal program operation, a Watchdog Timer time-out will initialise a device reset and set the status bit TO. However, if the system is in the DEEP SLEEP, 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 except 01010B and 10101B written into the WE4~WE0 bits, the second is using the Watchdog Timer software clear instruction, the third is via a HALT instruction. There is only one method of using software instruction to clear the Watchdog Timer. That is to use the single "CLR WDT" instruction to clear the WDT.

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



Watchdog Timer

Reset and Initialisation

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

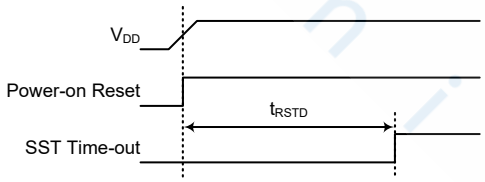
In addition to the power-on reset, another reset exists in the form of a Low Voltage Reset, LVR, where a full reset is implemented in situations where the power supply voltage falls below a certain threshold. Another type of reset is when the Watchdog Timer overflows and resets the microcontroller. All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

Reset Functions

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

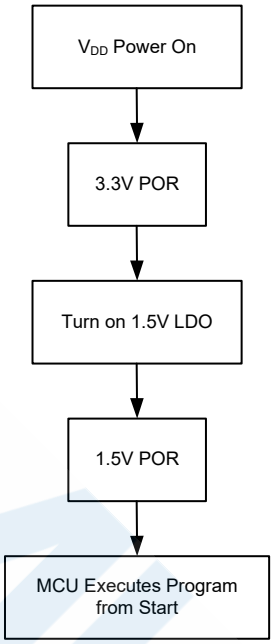
Power-on Reset

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



Power-on Reset Timing Chart

The accompany flowchart shows the program procedure for power on.



Internal Reset Control

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

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

Internal Reset Function Control

- **RSTC Register**

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

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

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

• **RSTFC Register**

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

"x": unknown

Bit 7~4 Unimplemented, read as "0"

Bit 3 **RSTF**: Reset control register software reset flag
0: Not occurred
1: Occurred
This bit is set to 1 by the RSTC control register software reset and cleared by the application program. Note that this bit can only be cleared to 0 by the application program.

Bit 2 **LVRF**: LVR function reset flag
Refer to the Low Voltage Reset section.

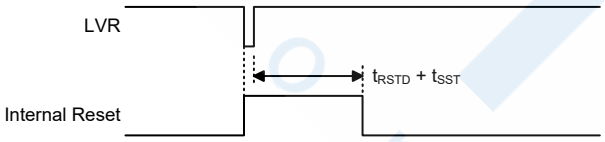
Bit 1 **LRF**: LVR control register software reset flag
Refer to the Low Voltage Reset section.

Bit 0 **WRF**: WDT control register software reset flag
Refer to the Watchdog Timer Control Register section.

Low Voltage Reset – LVR

The microcontrollers contain a low voltage reset circuit in order to monitor the supply voltage of these devices and provide an MCU reset should the value fall below a certain predefined level.

The LVR function is enabled/disabled using the related configuration option with a specific LVR voltage V_{LVR}. If the supply voltage of these devices drops to within a range of 0.9V~V_{LVR} such as might occur when changing the battery in battery powered applications, the LVR will automatically reset these devices internally and the LVRF bit in the RSTFC register will also be set to 1. For a valid LVR signal, a low supply voltage, i.e., a voltage in the range between 0.9V~V_{LVR} must exist for a time greater than that specified by t_{LVR} in the LVD/LVR Characteristics. If the low supply voltage state does not exceed this value, the LVR will ignore the low supply voltage and will not perform a reset function. The actual V_{LVR} value is set by the LVS7~LVS0 bits in the LVRC register with a fixed value of 1.9V. If the LVS7~LVS0 bits are changed to some different values by environmental noise, the LVR will reset these devices after a delay time, t_{SRESET}. When this happens, the LRF bit in the RSTFC register will be set to 1. After power on the register will have the value of 01010101B. Note that the LVR function will be automatically disabled when these devices enter the DEEP SLEEP, SLEEP or IDLE mode.



Low Voltage Reset Timing Chart

• **LVRC Register**

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

Bit 7~0 **LVS7~LVS0**: LVR Voltage Select control
01010101: 1.9V
00110011: 1.9V
10011001: 1.9V
10101010: 1.9V
Any other value: Generates MCU reset – register is reset to POR value

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

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

• **RSTFC Register**

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

"x": unknown

Bit 7~4 Unimplemented, read as "0"

Bit 3 **RSTF**: Reset control register software reset flag
Refer to the Internal Reset Control section.

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

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

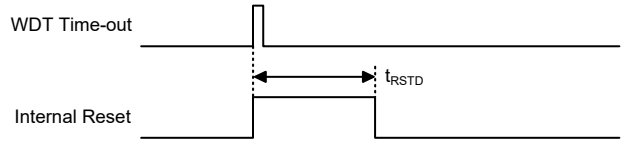
Bit 0 **WRF**: WDT control register software reset flag
Refer to the Watchdog Timer Control Register section.

IAP Reset

When a specific value of "55H" is written into the FC1 register, a reset signal will be generated to reset the whole device. Refer to the IAP section for more associated details.

Watchdog Time-out Reset during Normal Operation

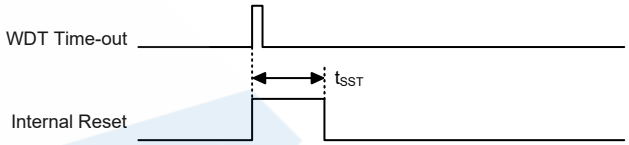
When the Watchdog Time-out Reset during normal operation in the FAST or SLOW mode occurs, the Watchdog time-out flag TO will be set to "1".



WDT Time-out Reset during Normal Operation Timing Chart

Watchdog Time-out Reset during SLEEP/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 and PDF flags will be set to "1". Refer to the System Start Up Time Characteristics for t_{SST} details.



WDT Time-out Reset during SLEEP/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 DEEP SLEEP, 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 or wake-up from DEEP SLEEP Mode
u	u	LVR reset during FAST or SLOW Mode operation
1	u	WDT time-out reset during FAST or SLOW Mode operation
1	1	WDT time-out reset during IDLE or SLEEP Mode operation

"u" stands for unchanged

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

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

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

Register	BC66F2235	BC66F2245	BC66F2255	Power On Reset	Reset (Wake-up from DEEP SLEEP)	WDT Time-out (Normal Operation)	WDT Time-out (SLEEP/IDLE)
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	xxxx xxxx	uuuu uuuu	uuuu uuuu
PCL	•	•	•	0000 0000	0000 0000	0000 0000	0000 0000
TBLP	•	•	•	xxxx xxxx	xxxx xxxx	uuuu uuuu	uuuu uuuu
TBLH	•	•	•	xxxx xxxx	xxxx xxxx	uuuu uuuu	uuuu uuuu
TBHP	•			---- -xxx	---- -xxx	---- -uuu	---- -uuu
		•		---- xxxx	---- xxxx	---- uuuu	---- uuuu
			•	---x xxxx	---x xxxx	---u uuuu	---u uuuu
STATUS	•	•	•	xx00 xxxx	xx00 xxxx	uu1u uuuu	uu11 uuuu
IAR2	•	•	•	0000 0000	0000 0000	0000 0000	uuuu uuuu
MP2L	•	•	•	0000 0000	0000 0000	0000 0000	uuuu uuuu
MP2H	•	•	•	0000 0000	0000 0000	0000 0000	uuuu uuuu
RSTFC	•	•	•	---- 0x00	---- 0x00	---- uuuu	---- uuuu
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
PA	•	•	•	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAC	•	•	•	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAPU	•	•	•	0000 0000	0000 0000	0000 0000	uuuu uuuu
PAWU	•	•	•	0000 0000	0000 0000	0000 0000	uuuu uuuu
PB	•	•	•	1111 1111	1111 1111	1111 1111	uuuu uuuu
PBC	•	•	•	1111 1111	1111 1111	1111 1111	uuuu uuuu
PBPU	•	•	•	0000 0000	0000 0000	0000 0000	uuuu uuuu
PC	•	•	•	-111 1111	-111 1111	-111 1111	-uuu uuuu
PCC	•	•	•	-111 1111	-111 1111	-111 1111	-uuu uuuu
PCPU	•	•	•	-000 0000	-000 0000	-000 0000	-uuu uuuu
INTEG	•	•	•	---- 0000	---- 0000	---- 0000	---- uuuu
SCC	•	•	•	001- 0000	001- 0000	001- 0000	uuu- uuuu
HIRCC	•	•	•	---- --01	---- --01	---- --01	---- --uu
HXTC	•	•	•	---- --00	---- --00	---- --00	---- --uu
LVDC	•	•	•	--00 0000	--00 0000	--00 0000	--uu uuuu
LVRC	•	•	•	0101 0101	0101 0101	0101 0101	uuuu uuuu
WDTC	•	•	•	0101 0011	0101 0011	0101 0011	uuuu uuuu
RSTC	•	•	•	0101 0101	0101 0101	0101 0101	uuuu uuuu
PSC0R	•	•	•	---- --00	---- --00	---- --00	---- --uu
PSC1R	•	•	•	---- --00	---- --00	---- --00	---- --uu
PWRC	•	•	•	00-0 1000	xx-x uuu0	uu-u uuuu	uu-u uuuu
RF_PWR	•	•	•	---- ---0	---- ---0	---- ---0	---- ---u
MFI0	•	•	•	0000 0000	0000 0000	0000 0000	uuuu uuuu
MFI1	•	•	•	--00 ---00	--00 --00	--00 --00	--uu --uu
MFI2	•	•	•	-000 -000	-000 -000	-000 -000	-uuu -uuu
IFS	•	•	•	---- --00	---- --00	---- --00	---- --uu

Register	BC66F2235	BC66F2245	BC66F2255	Power On Reset	Reset (Wake-up from DEEP SLEEP)	WDT Time-out (Normal Operation)	WDT Time-out (SLEEP/IDLE)
SADC0	•	•	•	0000 0000	0000 0000	0000 0000	uuuu uuuu
SADC1	•	•	•	0000 0000	0000 0000	0000 0000	uuuu uuuu
SADOH	•	•	•	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu (ADRFS=0) ---- uuuu (ADRFS=1)
SADOL	•	•	•	xxxx ----	xxxx ----	xxxx ----	uuuu ---- (ADRFS=0) uuuu uuuu (ADRFS=1)
TB0C	•	•	•	0--- -000	u--- -uuu	0--- -000	u--- -uuu
TB1C	•	•	•	0--- -000	0--- -000	0--- -000	u--- -uuu
PTM0AH	•	•	•	---- --00	---- --00	---- --00	---- --uu
PTM0AL	•	•	•	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM0C0	•	•	•	0000 0---	0000 0---	0000 0---	uuuu u---
PTM0C1	•	•	•	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM0DH	•	•	•	---- --00	---- --00	---- --00	---- --uu
PTM0DL	•	•	•	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM0RPH	•	•	•	---- --00	---- --00	---- --00	---- --uu
PTM0RPL	•	•	•	0000 0000	0000 0000	0000 0000	uuuu uuuu
CTM0C0	•	•	•	0000 0000	0000 0000	0000 0000	uuuu uuuu
CTM0C1	•	•	•	0000 0000	0000 0000	0000 0000	uuuu uuuu
CTM0DL	•	•	•	0000 0000	0000 0000	0000 0000	uuuu uuuu
CTM0DH	•	•	•	---- --00	---- --00	---- --00	---- --uu
CTM0AL	•	•	•	0000 0000	0000 0000	0000 0000	uuuu uuuu
CTM0AH	•	•	•	---- --00	---- --00	---- --00	---- --uu
PAS0	•	•	•	0000 0000	0000 0000	0000 0000	uuuu uuuu
PAS1	•	•	•	---- --00	---- --00	---- --00	---- --uu
PBS0	•	•	•	0000 0000	0000 0000	0000 0000	uuuu uuuu
PBS1	•	•	•	0000 0000	0000 0000	0000 0000	uuuu uuuu
PCS0	•	•	•	---- --00	---- --00	---- --00	---- --uu
PCS1	•	•	•	--00 0000	--00 0000	--00 0000	--uu uuuu
RF_OPER	•	•	•	--00 ---0	--00 ---0	--00 ---0	--uu ---u
RF_CLK1	•	•	•	1000 ---0	1000 ---0	1000 ---0	uuuu ---u
RF_CLK2	•	•	•	0000 0000	0000 0000	0000 0000	uuuu uuuu
RF_FIFO_CTRL1	•	•	•	0000 0000	0000 0000	0000 0000	uuuu uuuu
RF_FIFO_CTRL2	•	•	•	0011 1111	0011 1111	0011 1111	uuuu uuuu
RF_FIFO_CTRL3	•	•	•	0000 0000	0000 0000	0000 0000	uuuu uuuu
RF_FIFO_CTRL4	•	•	•	0--- 0001	0--- 0001	0--- 0001	u--- uuuu
RF_MOD1	•	•	•	1001 0000	1001 0000	1001 0000	uuuu uuuu
RF_MOD2	•	•	•	---- -001	---- -001	---- -001	---- -uuu
RF_MOD4	•	•	•	0000 1110	0000 1110	0000 1110	uuuu uuuu
RF_OPMOD	•	•	•	---- -000	---- -000	---- -000	---- -uuu
RF_SX1	•	•	•	-011 0110	-011 0110	-011 0110	-uuu uuuu
RF_SX2	•	•	•	0000 1010	0000 1010	0000 1010	uuuu uuuu
RF_SX3	•	•	•	1101 0111	1101 0111	1101 0111	uuuu uuuu

Register	BC66F2235	BC66F2245	BC66F2255	Power On Reset	Reset (Wake-up from DEEP SLEEP)	WDT Time-out (Normal Operation)	WDT Time-out (SLEEP/IDLE)
RF_SX4	●	●	●	---- 0011	---- 0011	---- 0011	---- uuuu
RF_CP3	●	●	●	1100 1010	1100 1010	1100 1010	uuuu uuuu
RF_OD1	●	●	●	0000 0100	0000 0100	0000 0100	uuuu uuuu
RF_VCO1	●	●	●	0001 0000	0001 0000	0001 0000	uuuu uuuu
RF_TX1	●	●	●	1000 1000	1000 1000	1000 1000	uuuu uuuu
RF_TX2	●	●	●	1000 0010	1000 0010	1000 0010	uuuu uuuu
RF_DFC_CAL	●	●	●	00-0 0000	00-0 0000	00-0 0000	uu-u uuuu
RF_LDO	●	●	●	0001 1000	0001 1000	0001 1000	uuuu uuuu
RF_XO1	●	●	●	10-1 0101	10-1 0101	10-1 0101	uu-u uuuu
SIMC0	●	●	●	1110 0000	1110 0000	1110 0000	uuuu uuuu
SIMC1* (UMD=0)	●	●	●	1000 0001	1000 0001	1000 0001	uuuu uuuu
UUCR1* (UMD=1)	●	●	●	0000 00x0	0000 00x0	0000 00x0	uuuu uuuu
SIMC2/SIMA/UUCR2	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
SIMD/UTXR_RXR	●	●	●	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
SIMTOC* (UMD=0)	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
UBRG* (UMD=1)	●	●	●	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
UUCR3	●	●	●	---- ---0	---- ---0	---- ---0	---- ---u
UUSR	●	●	●	0000 1011	0000 1011	0000 1011	uuuu uuuu
CTM1C0	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
CTM1C1	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
CTM1DL	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
CTM1DH	●	●	●	---- --00	---- --00	---- --00	---- --uu
CTM1AL	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
CTM1AH	●	●	●	---- --00	---- --00	---- --00	---- --uu
FC0	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
FC1	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
FARL	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
FARH	●			---- -000	---- -000	---- -000	---- -uuu
		●		---- 0000	---- 0000	---- 0000	---- uuuu
			●	---0 0000	---0 0000	---0 0000	---u uuuu
FD0L	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
FD0H	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
FD1L	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
FD1H	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
FD2L	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
FD2H	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
FD3L	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
FD3H	●	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKTMR	●	●	●	0000 0000	uuuu uuuu	0000 0000	uuuu uuuu
TKC0	●	●	●	0000 0010	uuuu uuuu	0000 0010	uuuu uuuu
TKC1	●	●	●	0000 0011	uuuu uuuu	0000 0011	uuuu uuuu
TKC2	●	●	●	---- -001	---- -uuu	---- -001	---- -uuu
TK16DL	●	●	●	0000 0000	uuuu uuuu	0000 0000	uuuu uuuu
TK16DH	●	●	●	0000 0000	uuuu uuuu	0000 0000	uuuu uuuu
TKM0C0	●	●	●	--0- 0000	--u- uuuu	--0- 0000	--u- uuuu

Register	BC66F2235	BC66F2245	BC66F2255	Power On Reset	Reset (Wake-up from DEEP SLEEP)	WDT Time-out (Normal Operation)	WDT Time-out (SLEEP/IDLE)
TKM0C1	•	•	•	0-00 0000	u-uu uuuu	0-00 0000	u-uu uuuu
TKM0C2	•	•	•	1110 0100	uuuu uuuu	1110 0100	uuuu uuuu
TKM016DL	•	•	•	0000 0000	uuuu uuuu	0000 0000	uuuu uuuu
TKM016DH	•	•	•	0000 0000	uuuu uuuu	0000 0000	uuuu uuuu
TKM0ROL	•	•	•	0000 0000	uuuu uuuu	0000 0000	uuuu uuuu
TKM0ROH	•	•	•	---- --00	---- --uu	---- --00	---- --uu
TKM0TH16L	•	•	•	0000 0000	uuuu uuuu	0000 0000	uuuu uuuu
TKM0TH16H	•	•	•	0000 0000	uuuu uuuu	0000 0000	uuuu uuuu
TKM0THS	•	•	•	0000 0000	uuuu uuuu	0000 0000	uuuu uuuu
TKM1C0	•	•	•	--0- 0000	--u- uuuu	--0- 0000	--u- uuuu
TKM1C1	•	•	•	0-00 0000	u-uu uuuu	0-00 0000	u-uu uuuu
TKM1C2	•	•	•	1110 0100	uuuu uuuu	1110 0100	uuuu uuuu
TKM116DL	•	•	•	0000 0000	uuuu uuuu	0000 0000	uuuu uuuu
TKM116DH	•	•	•	0000 0000	uuuu uuuu	0000 0000	uuuu uuuu
TKM1ROL	•	•	•	0000 0000	uuuu uuuu	0000 0000	uuuu uuuu
TKM1ROH	•	•	•	---- --00	---- --uu	---- --00	---- --uu
TKM1TH16L	•	•	•	0000 0000	uuuu uuuu	0000 0000	uuuu uuuu
TKM1TH16H	•	•	•	0000 0000	uuuu uuuu	0000 0000	uuuu uuuu
TKM1THS	•	•	•	0000 0000	uuuu uuuu	0000 0000	uuuu uuuu
TKM2C0	•	•	•	--0- 0000	--u- uuuu	--0- 0000	--u- uuuu
TKM2C1	•	•	•	0-00 0000	u-uu uuuu	0-00 0000	u-uu uuuu
TKM2C2	•	•	•	1110 0100	uuuu uuuu	1110 0100	uuuu uuuu
TKM216DL	•	•	•	0000 0000	uuuu uuuu	0000 0000	uuuu uuuu
TKM216DH	•	•	•	0000 0000	uuuu uuuu	0000 0000	uuuu uuuu
TKM2ROL	•	•	•	0000 0000	uuuu uuuu	0000 0000	uuuu uuuu
TKM2ROH	•	•	•	---- --00	---- --uu	---- --00	---- --uu
TKM2TH16L	•	•	•	0000 0000	uuuu uuuu	0000 0000	uuuu uuuu
TKM2TH16H	•	•	•	0000 0000	uuuu uuuu	0000 0000	uuuu uuuu
TKM2THS	•	•	•	0000 0000	uuuu uuuu	0000 0000	uuuu uuuu
TKM3C0	•	•	•	--0- 0000	--u- uuuu	--0- 0000	--u- uuuu
TKM3C1	•	•	•	0-00 0000	u-uu uuuu	0-00 0000	u-uu uuuu
TKM3C2	•	•	•	1110 0100	uuuu uuuu	1110 0100	uuuu uuuu
TKM316DL	•	•	•	0000 0000	uuuu uuuu	0000 0000	uuuu uuuu
TKM316DH	•	•	•	0000 0000	uuuu uuuu	0000 0000	uuuu uuuu
TKM3ROL	•	•	•	0000 0000	uuuu uuuu	0000 0000	uuuu uuuu
TKM3ROH	•	•	•	---- --00	---- --uu	---- --00	---- --uu
TKM3TH16L	•	•	•	0000 0000	uuuu uuuu	0000 0000	uuuu uuuu
TKM3TH16H	•	•	•	0000 0000	uuuu uuuu	0000 0000	uuuu uuuu
TKM3THS	•	•	•	0000 0000	uuuu uuuu	0000 0000	uuuu uuuu

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

Input/Output Ports

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

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

Register Name	Bit							
	7	6	5	4	3	2	1	0
PA	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0
PAC	PAC7	PAC6	PAC5	PAC4	PAC3	PAC2	PAC1	PAC0
PAPU	PAPU7	PAPU6	PAPU5	PAPU4	PAPU3	PAPU2	PAPU1	PAPU0
PAWU	PAWU7	PAWU6	PAWU5	PAWU4	PAWU3	PAWU2	PAWU1	PAWU0
PB	PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0
PBC	PBC7	PBC6	PBC5	PBC4	PBC3	PBC2	PBC1	PBC0
PBPU	PBPU7	PBPU6	PBPU5	PBPU4	PBPU3	PBPU2	PBPU1	PBPU0
PC	—	PC6	PC5	PC4	PC3	PC2	PC1	PC0
PCC	—	PCC6	PCC5	PCC4	PCC3	PCC2	PCC1	PCC0
PCPU	—	PCPU6	PCPU5	PCPU4	PCPU3	PCPU2	PCPU1	PCPU0

"—": Unimplemented, read as "0"

I/O Logic Function Register List

Note: 1. The table shows the situation in the largest package, refer to the "Pin Assignment" section for the actual available bits of each device.

2. For less pin count devices, the unavailable bits in these registers are reserved and should be properly configured to avoid unwanted power consumption resulting from floating input conditions. Refer to the "Standby Current Considerations" section.

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 a digital input have the capability of being connected to an internal pull-high resistor. These pull-high resistors are selected using registers, namely PAPU~PCPU, and are implemented using weak PMOS transistors.

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

• **PxPU Register**

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

PxPUn: I/O Port x Pin pull-high function control
0: Disable
1: Enable

The PxPUn bit is used to control the pin pull-high function. Here the "x" can be A, B or C. However, the actual available bits for each I/O Port of each device may be different.

Port A Wake-up

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

Note that the wake-up function can be controlled by the wake-up control registers only when the pin is selected as a general purpose input and the MCU enters the DEEP SLEEP, SLEEP or IDLE Mode.

• **PAWU Register**

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

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

I/O Port Control Registers

Each I/O port has its own control register known as PAC~PCC, 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.

• **PxC Register**

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

PxCn: I/O Port x Pin type selection
0: Output
1: Input
The PxCn bit is used to control the pin type selection. Here the "x" can be A, B or C. However, the actual available bits for each I/O Port of each device may be different.

Pin-shared Functions

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

Pin-shared Function Selection Registers

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

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

Register Name	Bit							
	7	6	5	4	3	2	1	0
PAS0	PAS07	PAS06	PAS05	PAS04	PAS03	PAS02	PAS01	PAS00
PAS1	—	—	—	—	—	—	PAS11	PAS10
PBS0	PBS07	PBS06	PBS05	PBS04	PBS03	PBS02	PBS01	PBS00
PBS1	PBS17	PBS16	PBS15	PBS14	PBS13	PBS12	PBS11	PBS10
PCS0	—	—	—	—	—	—	PCS01	PCS00
PCS1	—	—	PCS15	PCS14	PCS13	PCS12	PCS11	PCS10
IFS	—	—	—	—	—	—	INT1PS	INT0PS

Pin-shared Function Selection Register List

• **PAS0 Register**

Bit	7	6	5	4	3	2	1	0
Name	PAS07	PAS06	PAS05	PAS04	PAS03	PAS02	PAS01	PAS00
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 **PAS07~PAS06**: PA3 pin-shared function selection
00: PA3/INT0
01: PA3/INT0
10: KEY16
11: KEY16
Note: For the BC66F2235 device, these bits are reserved and should be fixed at "00".

Bit 5~4 **PAS05~PAS04**: PA2 pin-shared function selection
00: PA2
01: PTP0
10: KEY15
11: KEY15

Bit 3~2 **PAS03~PAS02**: PA1 pin-shared function selection
00: PA1
01: PA1
10: KEY14
11: KEY14

Bit 1~0 **PAS01~PAS00**: PA0 pin-shared function selection
00: PA0
01: PTP0B
10: KEY13
11: KEY13

• **PAS1 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	PAS11	PAS10
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2 Unimplemented, read as "0"

Bit 1~0 **PAS11~PAS10**: PA4 pin-shared function selection
00: PA4/INT1
01: CTP0
10: SCL/SCK
11: KEY12

• **PBS0 Register**

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

Bit 7~6 **PBS07~PBS06**: PB3 pin-shared function selection
00: PB3
01: CTP1B
10: AN3
11: KEY4
Note: For the BC66F2235 device, these bits are reserved and should be fixed at "00".

Bit 5~4 **PBS05~PBS04**: PB2 pin-shared function selection
00: PB2/INT1
01: PB2/INT1
10: AN2
11: KEY3
Note: For the BC66F2235 device, these bits are reserved and should be fixed at "00".

Bit 3~2 **PBS03~PBS02**: PB1 pin-shared function selection
00: PB1
01: CTP1
10: AN1
11: KEY2
Note: For the BC66F2235 device, these bits are reserved and should be fixed at "00".

Bit 1~0 **PBS01~PBS00**: PB0 pin-shared function selection
00: PB0
01: PB0
10: AN0
11: KEY1

• **PBS1 Register**

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

Bit 7~6 **PBS17~PBS16**: PB7 pin-shared function selection
00: PB7
01: PB7
10: KEY8
11: KEY8
Note: For the BC66F2235 and BC66F2245 devices, these bits are reserved and should be fixed at "00".

Bit 5~4 **PBS15~PBS14**: PB6 pin-shared function selection
00: PB6
01: PB6
10: KEY7
11: KEY7
Note: For the BC66F2235 and BC66F2245 devices, these bits are reserved and should be fixed at "00".

Bit 3~2 **PBS13~PBS12**: PB5 pin-shared function selection
00: PB5/CTCK1
01: PB5/CTCK1
10: KEY6
11: KEY6

Bit 1~0 **PBS11~PBS10**: PB4 pin-shared function selection
00: PB4
01: CTP0B
10: KEY5
11: KEY5

• **PCS0 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	PCS01	PCS00
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2 Unimplemented, read as "0"

Bit 1~0 **PCS01~PCS00**: PC0 pin-shared function selection
00: PC0
01: CTP0
10: CTP0
11: CTP0
Note: For the BC66F2235 device, these bits are reserved and should be fixed at "00".

• **PCS1 Register**

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

Bit 7~6 Unimplemented, read as "0"

Bit 5~4 **PCS15~PCS14**: PC6 pin-shared function selection
00: PC6
01: PC6
10: SDI/URX/UTX/SDA
11: KEY11

Bit 3~2 **PCS13~PCS12**: PC5 pin-shared function selection
00: PC5
01: PC5
10: SDO/UTX
11: KEY10
Note: For the BC66F2235 device, these bits are reserved and should be fixed at "00".

Bit 1~0 **PCS11~PCS10**: PC4 pin-shared function selection
00: PC4
01: PC4
10: $\overline{\text{SCS}}$
11: KEY9
Note: For the BC66F2235 device, these bits are reserved and should be fixed at "00".

• **IFS Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	INT1PS	INT0PS
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

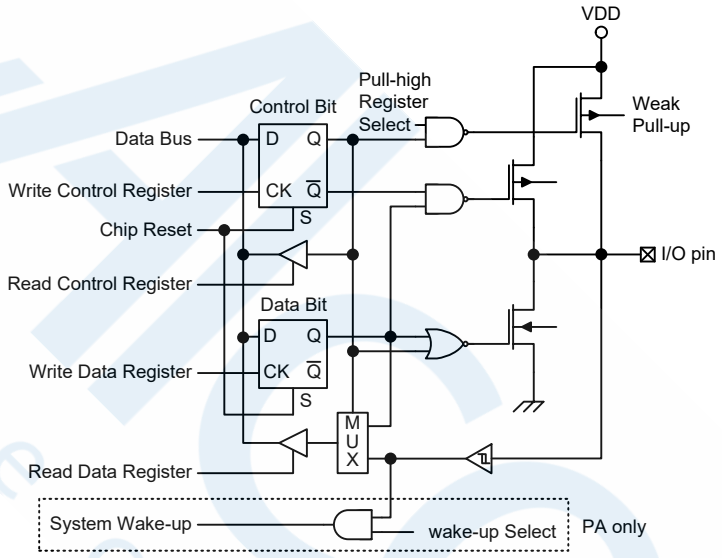
Bit 7~2 Unimplemented, read as "0"

Bit 1 **INT1PS**: INT1 input source pin selection
For BC66F2235:
0: PA4
1: Reserved
For BC66F2245/BC66F2255:
0: PA4
1: PB2

Bit 0 **INT0PS**: INT0 input source pin selection
For BC66F2245:
0: PA3
1: Reserved
For BC66F2255:
0: PA3
1: PA5
Note: For the BC66F2235 device, these bits are reserved and should be fixed at "00".

I/O Pin Structures

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



Logic Function Input/Output Structure

Programming Considerations

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

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

Timer Modules – TM

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

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

Introduction

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

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

TM Function Summary

TM Operation

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

TM Clock Source

The clock source which drives the main counter in each TM can originate from various sources. The selection of the required clock source is implemented using the xTnCK2~xTnCK0 bits in the xTMn control registers, where “x” stands for C or P type TM and “n” stands for the specific TM serial number. The clock source can be a ratio of the system clock, f_{SYS}, or the internal high clock, f_H, the f_{SUB} clock source or the external xTCKn pin. The xTCKn pin clock source is used to allow an external signal to drive the TM as an external clock source for event counting.

TM Interrupts

The Compact or Periodic type TM each has two internal interrupt, one for each of the internal comparator A or comparator P, which generate a TM interrupt when a compare match condition

occurs. When a TM interrupt is generated, it can be used to clear the counter and also to change the state of the TM output pin.

TM External Pins

The TMs each have no or one input pin with the label xTCKn. The xTMn input pin, xTCKn, is essentially a clock source for the xTMn and is selected using the xTnCK2~xTnCK0 bits in the xTMnC0 register. This external TM input pin allows an external clock source to drive the internal TM. The xTCKn input pin can be chosen to have either a rising or falling active edge. The PTCKn pins are also used as the external trigger input pin in single pulse output mode for the PTMn.

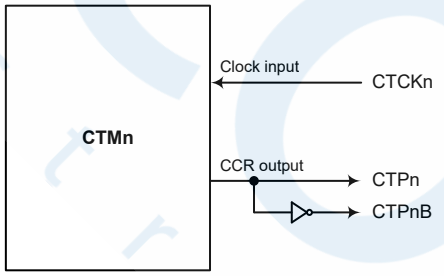
The TMs each have no or two output pins, with the label xTPn and xTPnB. When the TM is in the Compare Match Output Mode, these pins can be controlled by the TM to switch to a high or low level or to toggle when a compare match situation occurs. The external xTPn and xTPnB output pins are also the pins where the xTMn generates the PWM output waveform.

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

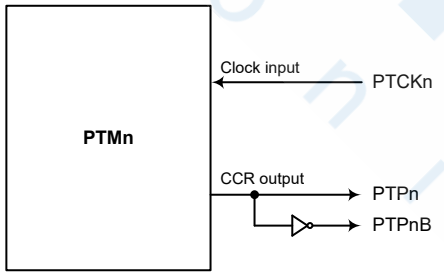
The number of external pins for each TM type is different, the details are provided in the accompanying table.

Device	CTM		PTM	
	Input	Output	Input	Output
BC66F2235	—; CTCK1	CTP0, CTP0B; —	—	PTP0, PTP0B
BC66F2245	—; CTCK1	CTP0, CTP0B; CTP1, CTP1B	—	PTP0, PTP0B
BC66F2255	CTCK0; CTCK1	CTP0, CTP0B; CTP1, CTP1B	PTCK0	PTP0, PTP0B

TM External Pins



CTM Function Pin Block Diagram (n=0~1)

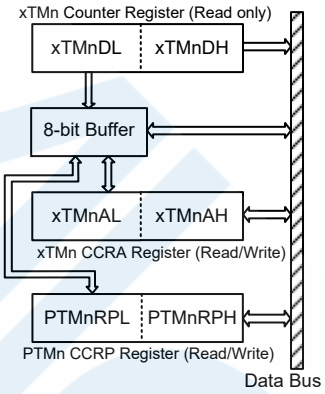


PTM Function Pin Block Diagram (n=0)

Programming Considerations

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

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



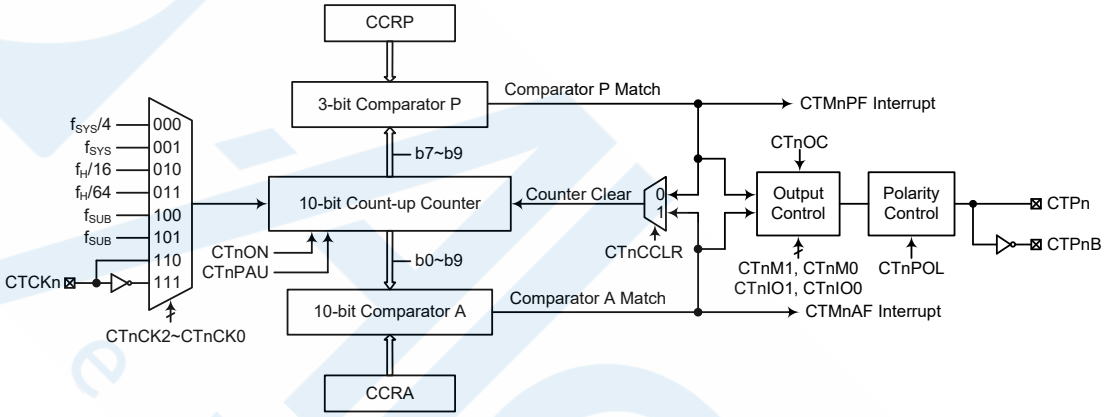
The following steps show the read and write procedures:

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

Compact Type TM – CTM

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

Device	CTM Core	CTM Input Pins	CTM Output Pins
BC66F2235	10-bit CTM (CTM0, CTM1)	—; CTCK1	CTP0, CTP0B; —
BC66F2245		—; CTCK1	CTP0, CTP0B; CTP1, CTP1B
BC66F2255		CTCK0; CTCK1	CTP0, CTP0B; CTP1, CTP1B



Note: The CTMn external pins are pin-shared with other functions, therefore before using the CTMn function, ensure that the pin-shared function registers have been set properly to enable the CTMn pin function. The CTCKn pins, if used, must also be set as an input by setting the corresponding bits in the port control register.

10-bit Compact Type TM Block Diagram (n=0~1)

Compact Type TM Operation

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

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

Compact Type TM Register Description

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

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

10-bit Compact Type TM Register List (n=0~1)

- **CTMnC0 Register**

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

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

Bit 6~4 **CTnCK2~CTnCK0**: CTMn counter clock selection
000: $f_{SYS}/4$
001: f_{SYS}
010: $f_H/16$
011: $f_H/64$
100: f_{SUB}
101: f_{SUB}
110: CTCKn rising edge clock
111: CTCKn falling edge clock
These three bits are used to select the clock source for the CTMn. The external pin clock source can be chosen to be active on the rising or falling edge. The clock source f_{SYS} is the system clock, while f_H and f_{SUB} are other internal clocks, the details of which can be found in the Operating Modes and System Clocks section.

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

pin will be reset to its initial condition, as specified by the CTnOC bit, when the CTnON bit changes from low to high.

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

Comparator P match period=
000: 1024 CTMn clocks
001~111: (1~7)×128 CTMn clocks

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

• **CTMnC1 Register**

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

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

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

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

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

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

Timer/Counter Mode
Unused

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

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

output pin when a compare match occurs. After the CTMn output pin changes state, it can be reset to its initial level by changing the level of the CTnON bit from low to high.

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

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

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

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

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

Bit 2 **CTnPOL**: CTMn CTPn output polarity control
0: Non-invert
1: Invert

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

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

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

Bit 0 **CTCCLR**: CTMn counter clear condition selection
0: Comparator P match
1: Comparator A match

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

- **CTMnDL Register**

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

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

• **CTMnDH Register**

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

Bit 7~2 Unimplemented, read as "0"

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

• **CTMnAL Register**

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

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

• **CTMnAH Register**

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

Bit 7~2 Unimplemented, read as "0"

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

Compact Type TM Operation Modes

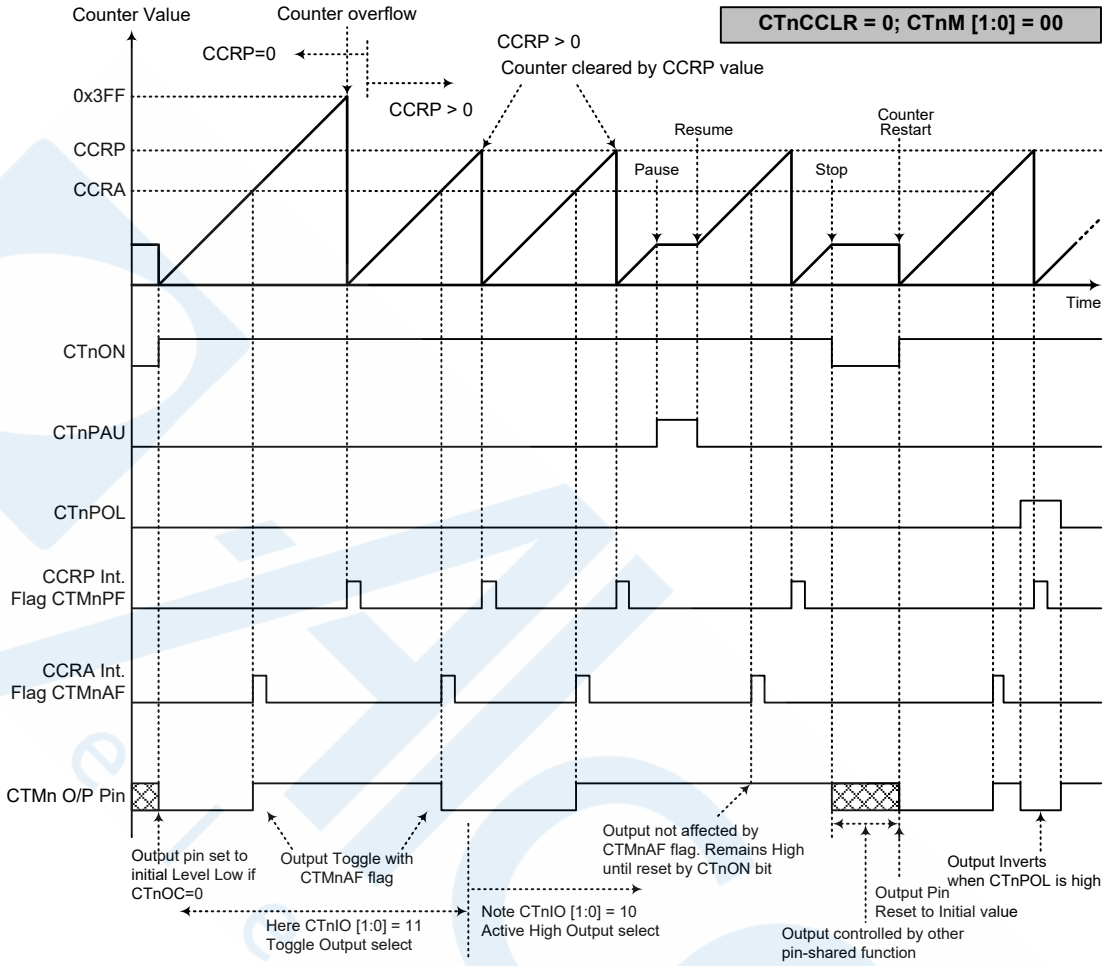
The Compact Type TM can operate in one of three operating modes, Compare Match Output Mode, PWM Output Mode or Timer/Counter Mode. The operating mode is selected using the CTnM1 and CTnM0 bits in the CTMnC1 register.

Compare Match Output Mode

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

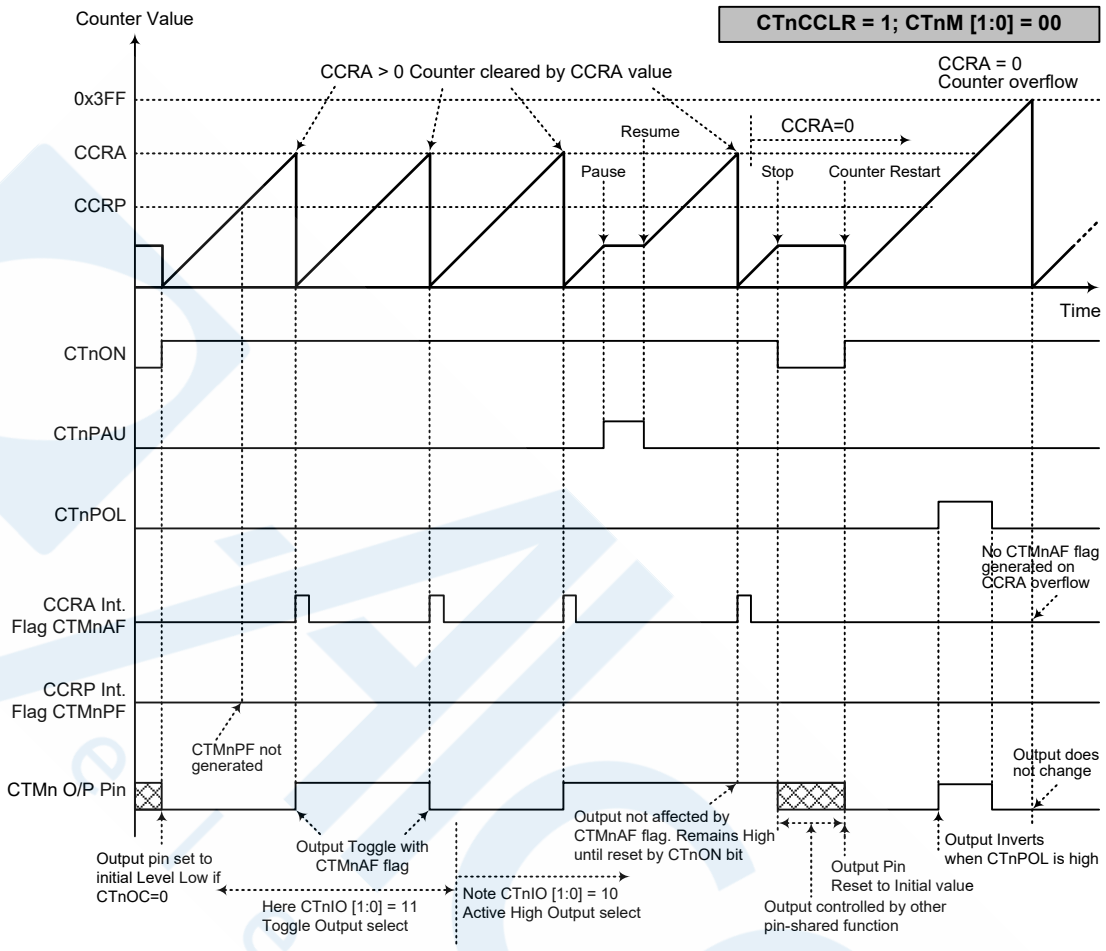
If the CTnCCLR bit in the CTMnC1 register is high then the counter will be cleared when a compare match occurs from Comparator A. However, here only the CTMnAF interrupt request flag will be generated even if the value of the CCRP bits is less than that of the CCRA registers. Therefore when CTnCCLR is high no CTMnPF interrupt request flag will be generated. If the CCRA bits are all zero, the counter will overflow when it reaches its maximum 10-bit, 3FF Hex, value. However, here the CTMnAF interrupt request flag will not be generated.

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



Compare Match Output Mode – CTnCCLR=0 (n=0~1)

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



Compare Match Output Mode – CTnCCLR=1 (n=0~1)

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

Timer/Counter Mode

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

PWM Output Mode

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

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

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

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

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

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

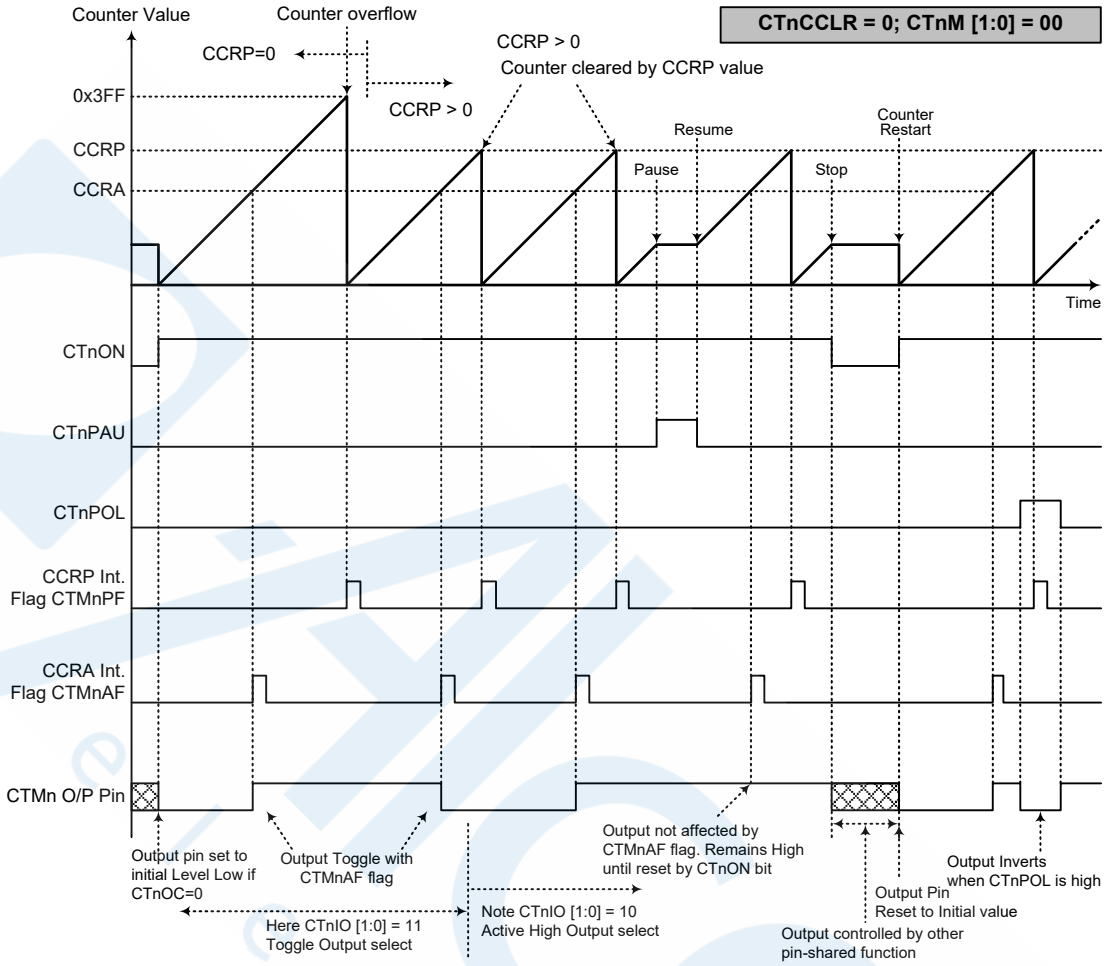
The CTMn PWM output frequency=$(f_{SYS}/4)/(4\times128)=f_{SYS}/2048$=8kHz, duty=128/(4×128)=25%.

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

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

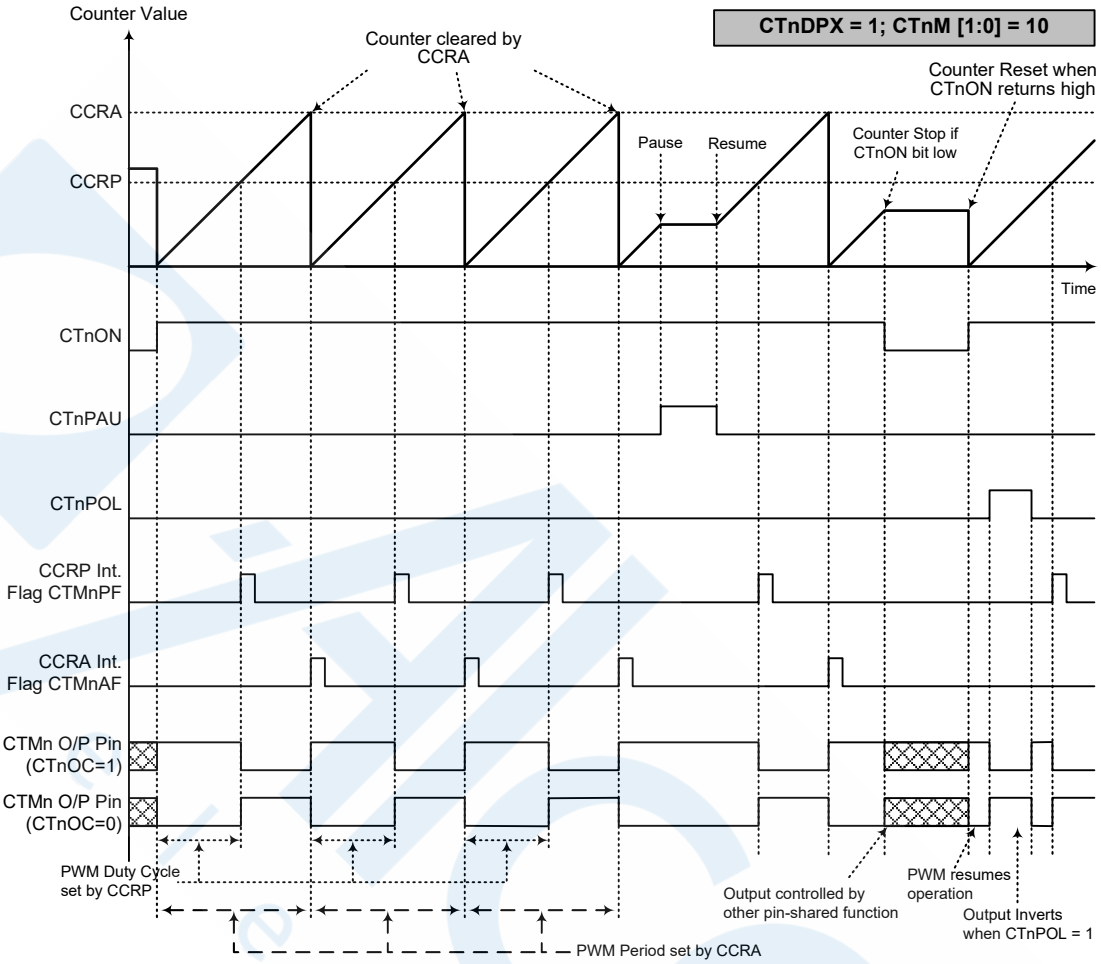
CCRP	1~7	0
Period	CCRA	
Duty	CCRP×128	1024

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



PWM Output Mode – CTnDPX=0 (n=0~1)

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



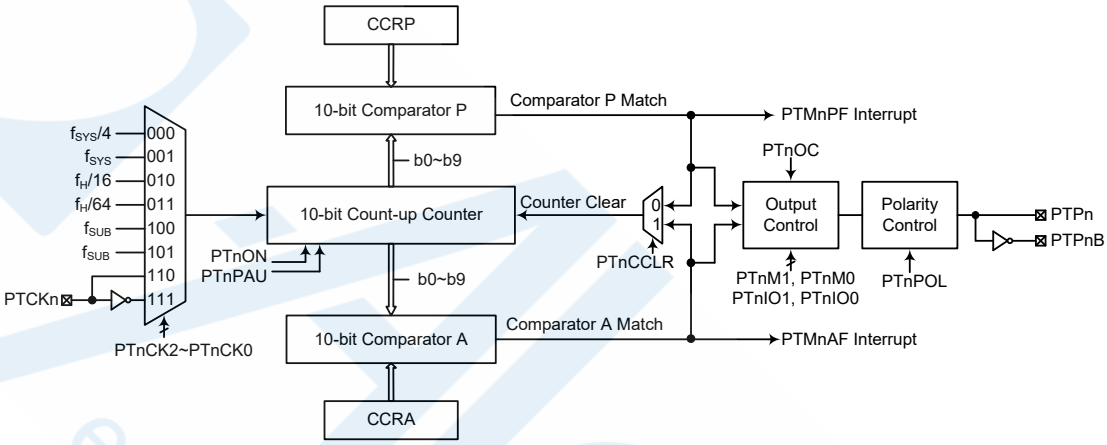
PWM Output Mode – CTnDPX=1 (n=0~1)

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

Periodic Type TM – PTM

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

Device	PTM Core	PTM Input Pins	PTM Output Pins
BC66F2235	10-bit PTM (PTM0)	—	PTP0, PTP0B
BC66F2245		—	PTP0, PTP0B
BC66F2255		PTCK0	PTP0, PTP0B



Note: The PTMn external pins are pin-shared with other functions, so before using the PTMn function the pin-shared function registers must be set properly to enable the PTMn pin function. The PTCKn pin, if used, must also be set as an input by setting the corresponding bits in the port control register.

10-bit Periodic Type TM Block Diagram (n=0)

Periodic TM Operation

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

The only way of changing the value of the 10-bit counter using the application program is to clear the counter by changing the 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 PTMn 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 two output pins. All operating setup conditions are selected using relevant internal registers.

Periodic Type TM Register Description

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

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

10-bit Periodic TM Register List (n=0)

- **PTMnC0 Register**

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

Bit 7 **PTnPAU**: PTMn counter pause control
0: Run
1: Pause

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

Bit 6~4 **PTnCK2~PTnCK0**: PTMn counter clock selection
000: $f_{SYS}/4$
001: f_{SYS}
010: $f_H/16$
011: $f_H/64$
100: f_{SUB}
101: f_{SUB}
110: PTCKn rising edge clock
111: PTCKn falling edge clock

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

Bit 3 **PTnON**: PTMn counter on/off control
0: Off
1: On

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

If the PTMn is in the Compare Match Output Mode, PWM Output Mode or Single Pulse Output Mode then the PTMn output pin will be reset to its initial condition, as specified by the PTnOC bit, when the PTnON bit changes from low to high.

Bit 2~0 Unimplemented, read as "0"

• **PTMnC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	PTnM1	PTnM0	PTnIO1	PTnIO0	PTnOC	PTnPOL	D1	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**: PTMn operating mode selection
00: Compare Match Output Mode
01: Undefined
10: PWM Output Mode or Single Pulse Output Mode
11: Timer/Counter Mode

These bits setup the required operating mode for the PTMn. To ensure reliable operation the PTMn should be switched off before any changes are made to the PTnM1 and PTnM0 bits. In the Timer/Counter Mode, the PTMn output pin state is undefined.

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

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

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

Timer/Counter Mode
Unused

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

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

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

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

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

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

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

Bit 2 **PTnPOL**: PTMn PTPn output polarity control
0: Non-inverted
1: Inverted
This bit controls the polarity of the PTPn output pin. When the bit is set high the PTMn output pin will be inverted and not inverted when the bit is zero. It has no effect if the PTMn is in the Timer/Counter Mode.

Bit 1 **D1**: Reserved, must be fixed at "0"

Bit 0 **PTnCCLR**: PTMn counter clear condition selection
0: Comparator P match
1: Comparator A match
This bit is used to select the method which clears the counter. Remember that the Periodic TM contains two comparators, Comparator A and Comparator P, either of which can be selected to clear the internal counter. With the PTnCCLR bit set high, the counter will be cleared when a compare match occurs from the Comparator A. When the bit is low, the counter will be cleared when a compare match occurs from the Comparator P or with a counter overflow. A counter overflow clearing method can only be implemented if the CCRP bits are all cleared to zero. The PTnCCLR bit is not used in the PWM Output or Single Pulse Output 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 **D7~D0**: 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 **D9~D8**: 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 **D7~D0**: 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 **D9~D8**: PTMn CCRA High Byte Register bit 1 ~ bit 0
PTMn 10-bit CCRA bit 9 ~ bit 8

• **PTMnRPL Register**

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

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

• **PTMnRPH Register**

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

Bit 7~2 Unimplemented, read as "0"

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

Periodic Type TM Operation Modes

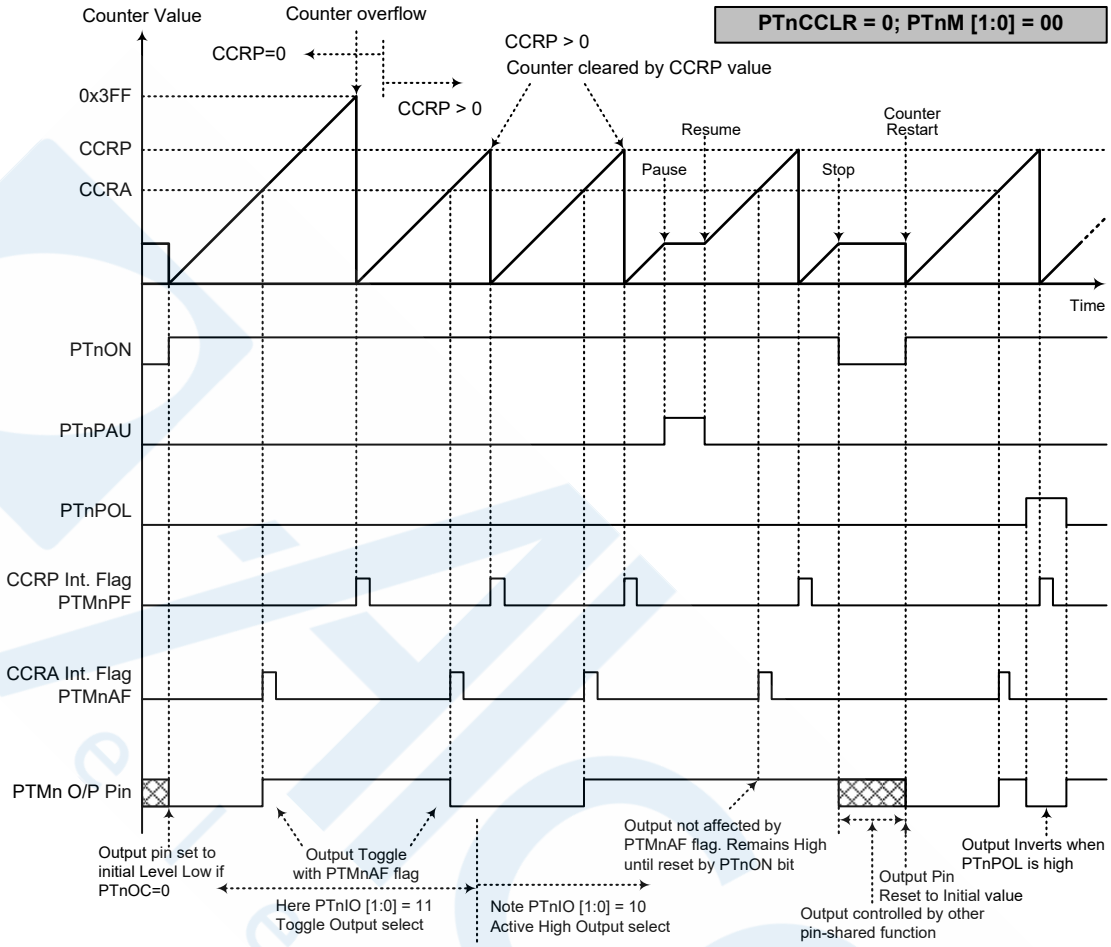
The Periodic Type TM can operate in one of four operating modes, Compare Match Output Mode, PWM Output Mode, Single Pulse Output Mode or Timer/Counter Mode. The operating mode is selected using the PTnM1 and PTnM0 bits in the PTMnC1 register.

Compare Match Output Mode

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

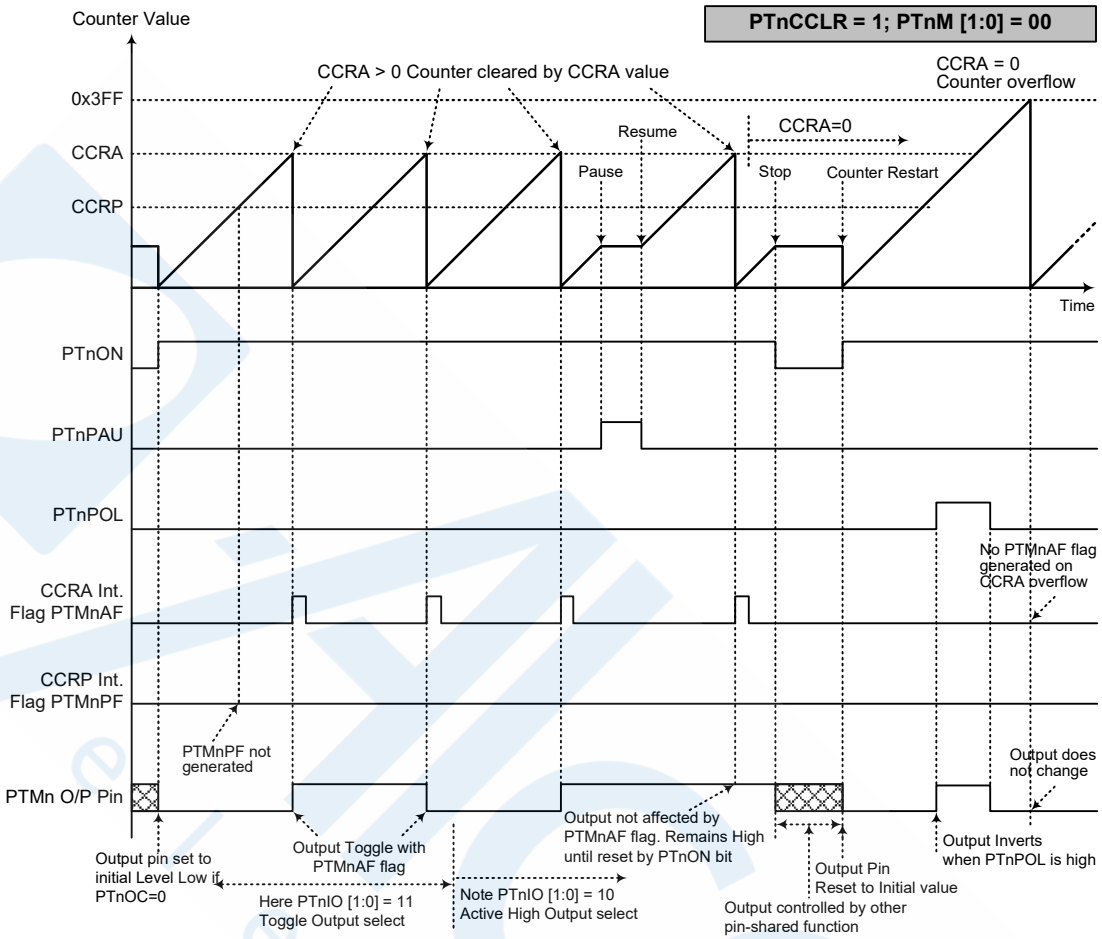
If the PTnCCLR bit in the PTMnC1 register is high then the counter will be cleared when a compare match occurs from Comparator A. However, here only the PTMnAF interrupt request flag will be generated even if the value of the CCRP bits is less than that of the CCRA registers. Therefore when PTnCCLR is high no PTMnPF interrupt request flag will be generated. In the Compare Match Output Mode, the CCRA cannot be cleared to zero. If the CCRA bits are all zero, the counter will overflow when it reaches its maximum 10-bit, 3FF Hex, value, however here the PTMnAF interrupt request flag will not be generated.

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



Compare Match Output Mode – PTnCCLR=0 (n=0)

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



Compare Match Output Mode – PTnCCLR=1 (n=0)

Note: 1. With PTnCCLR=1, a Comparator A match will clear the counter
2. The PTMn output pin is controlled only by the PTMnAF flag
3. The output pin is reset to its initial state by a PTnON bit rising edge
4. A PTMnPF flag is not generated when PTnCCLR=1

Timer/Counter Mode

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

PWM Output Mode

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

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

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

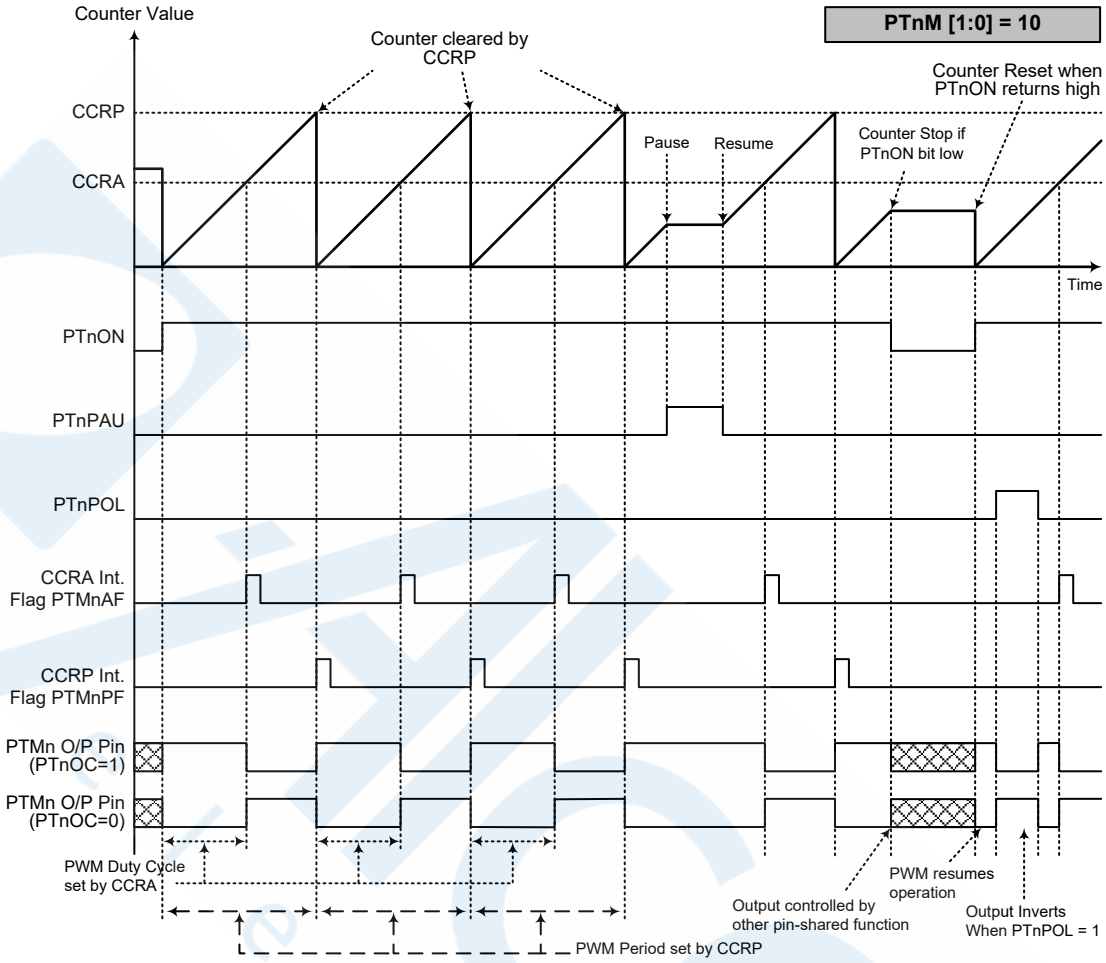
- **10-bit PTMn, PWM Output Mode, Edge-aligned Mode**

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=8kHz, duty=128/512=25%,

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



PWM Output Mode (n=0)

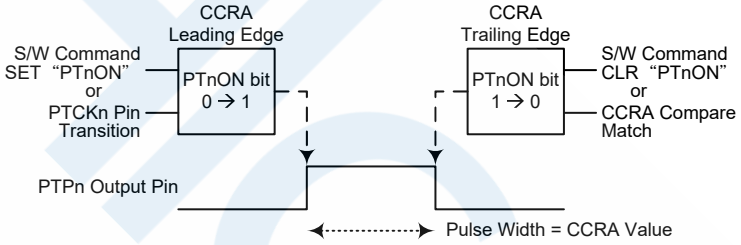
Note: 1. The counter is cleared by CCRP.
2. A counter clear sets the PWM Period
3. The internal PWM function continues running even when PTnIO[1:0]=00 or 01
4. The PTnCCLR bit has no influence on PWM operation

Single Pulse Output Mode

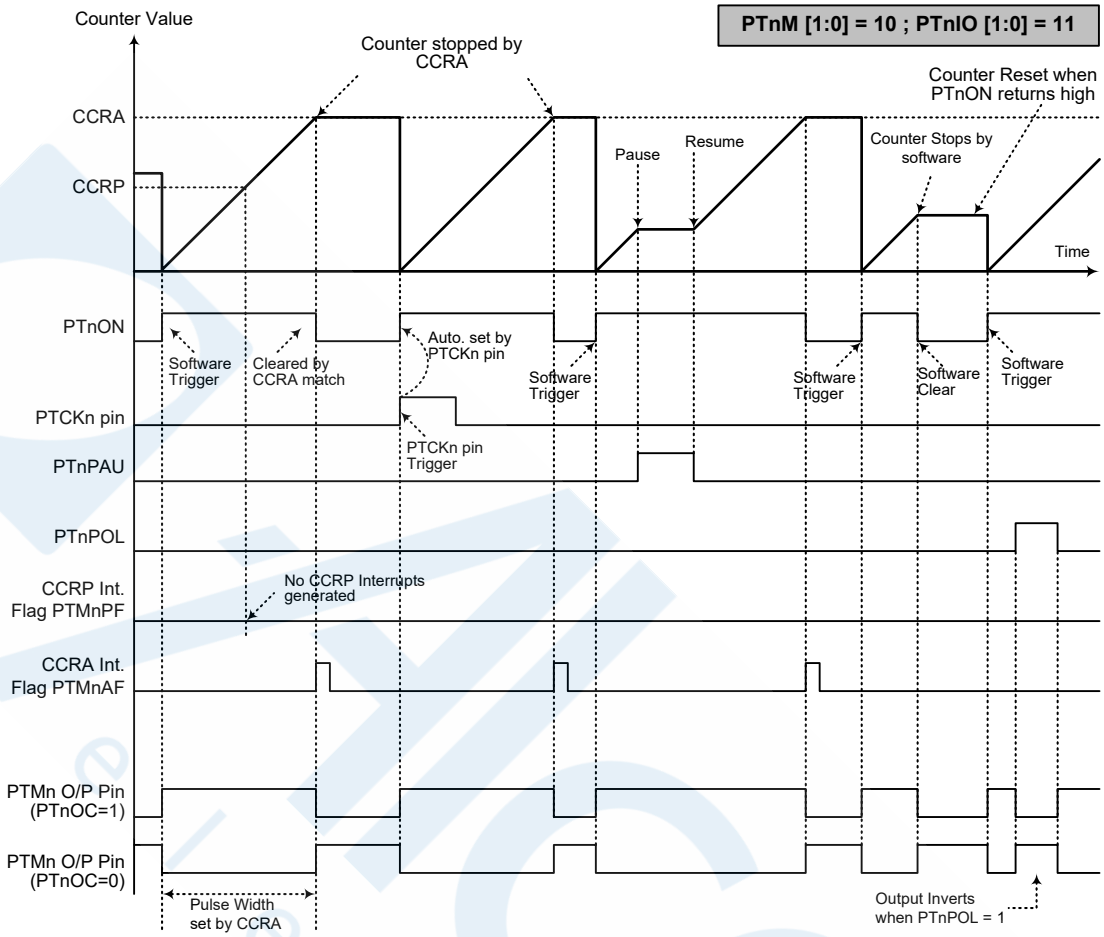
To select this mode, bits PTnM1 and PTnM0 in the PTMnC1 register should be set to 10 respectively and also the PTnIO1 and PTnIO0 bits should be set to 11 respectively. The Single Pulse Output Mode, as the name suggests, will generate a single shot pulse on the PTMn output pin.

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

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



Single Pulse Generation (n=0)



Single Pulse Output Mode (n=0)

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

Analog to Digital Converter

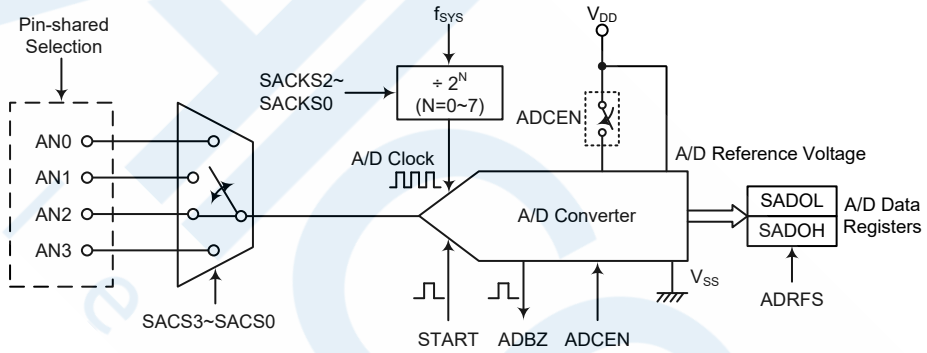
The need to interface to real world analog signals is a common requirement for many electronic systems. However, to properly process these signals by a microcontroller, they must first be converted into digital signals by A/D converters. By integrating the A/D conversion electronic circuitry into the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

These devices contain a multi-channel analog to digital converter which can directly interface to external analog signals, such as that from sensors or other control signals and convert these signals directly into a 12-bit digital value.

Device	External Input Channels	A/D Channel Select Bits
BC66F2235	1: AN0	SACS3~SACS0
BC66F2245 BC66F2255	4: AN0~AN3	SACS3~SACS0

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 four registers. A read only register pair exists to store the A/D converter data 12-bit single value. The remaining two registers are control registers which configures the operating and control function of the A/D converter.

Register Name	Bit							
	7	6	5	4	3	2	1	0
SADOL (ADRFS=0)	D3	D2	D1	D0	—	—	—	—
SADOL (ADRFS=1)	D7	D6	D5	D4	D3	D2	D1	D0
SADOH (ADRFS=0)	D11	D10	D9	D8	D7	D6	D5	D4
SADOH (ADRFS=1)	—	—	—	—	D11	D10	D9	D8
SADC0	START	ADBZ	ADCEN	ADRFS	SACS3	SACS2	SACS1	SACS0
SADC1	D7	D6	D5	D4	D3	SACKS2	SACKS1	SACKS0

A/D Converter Register List

A/D Converter Data Registers – SADOL, SADOH

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

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

A/D Converter Data Registers

A/D Converter Control Registers – SADC0, SADC1

To control the function and operation of the A/D converter, two control registers known as SADC0 and SADC1 are provided. These 8-bit registers define functions such as the selection of which analog channel is connected to the internal A/D converter, the digitised data format, the A/D clock source as well as controlling the start function and monitoring the A/D converter busy status. As these devices contain only one actual analog to digital converter hardware circuit, each of the external analog signal inputs must be routed to the converter. The SACS3~SACS0 bits in the SADC0 register are used to determine which external channel input is selected to be converted.

The relevant pin-shared function selection bits determine which pins on I/O Ports 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. When the pin is selected to be an A/D input, its original function whether it is an I/O or other pin-shared function will be removed. In addition, any internal pull-high resistor connected to the pin will be automatically removed if the pin is selected to be an A/D converter input.

- **SADC0 Register**

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

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

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

Bit 5 **ADCEN**: A/D converter function enable control
0: Disable
1: Enable
This bit controls the A/D internal function. This bit should be set to one to enable the A/D converter. If the bit is cleared to zero, then the A/D converter will be switched off

reducing the device power consumption. When the A/D converter function is disabled, the contents of the A/D data register pair known as SADOH and SADOL will be unchanged.

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

Bit 3~0 **SACS3~SACS0**: A/D converter external analog channel input selection
For BC66F2235:
0000: AN0
0001~1111: Non-existed channel, input floating if selected
For BC66F2245/BC66F2255:
0000: AN0
0001: AN1
0010: AN2
0011: AN3
0100~1111: Non-existed channel, input floating if selected

• **SADC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	D5	D4	D3	SACKS2	SACKS1	SACKS0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~3 **D7~D3**: Reserved, must be fixed at "00000"

Bit 2~0 **SACKS2~SACKS0**: A/D conversion clock source selection
000: f_{SYS}
001: $f_{SYS}/2$
010: $f_{SYS}/4$
011: $f_{SYS}/8$
100: $f_{SYS}/16$
101: $f_{SYS}/32$
110: $f_{SYS}/64$
111: $f_{SYS}/128$
These three bits are used to select the clock source for the A/D converter.

A/D Converter Reference Voltage

The reference voltage supply to the A/D converter is supplied from the internal A/D power supply voltage, V_{DD}. The analog input values must not be allowed to exceed the value of the reference voltage, V_{DD}.

A/D Converter Input Signals

All the external A/D analog channel input pins are pin-shared with the I/O pins as well as other functions. The corresponding control bits for each A/D external input pin in the PBS0 register determine whether the input pins are set as A/D converter analog inputs or whether they have other functions. If the pin is set to be as an A/D analog channel input, 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 set through register programming, will be automatically disconnected if the pins are set as A/D inputs. Note that it is not necessary to first set the A/D pin as an input in the port control register to enable the A/D input as when the pin-shared function control bits enable an A/D input, the status of the port control register will be overridden.

A/D Converter Operation

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

The ADBZ bit in the SADC0 register is used to indicate whether the analog to digital conversion process is in progress or not. This bit will be automatically set to 1 at the next ADCLK after an A/D conversion is successfully initiated. When the A/D conversion is complete, the ADBZ will be cleared to 0. In addition, the corresponding A/D interrupt request flag will be set in the interrupt control register, and if the associated 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 poll the ADBZ bit in the SADC0 register to check whether it has been cleared as an alternative method of detecting the end of an A/D conversion cycle. If ADCLK is set to a low time, the polling result may vary from 0→1→0.

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

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

A/D Clock Period Examples

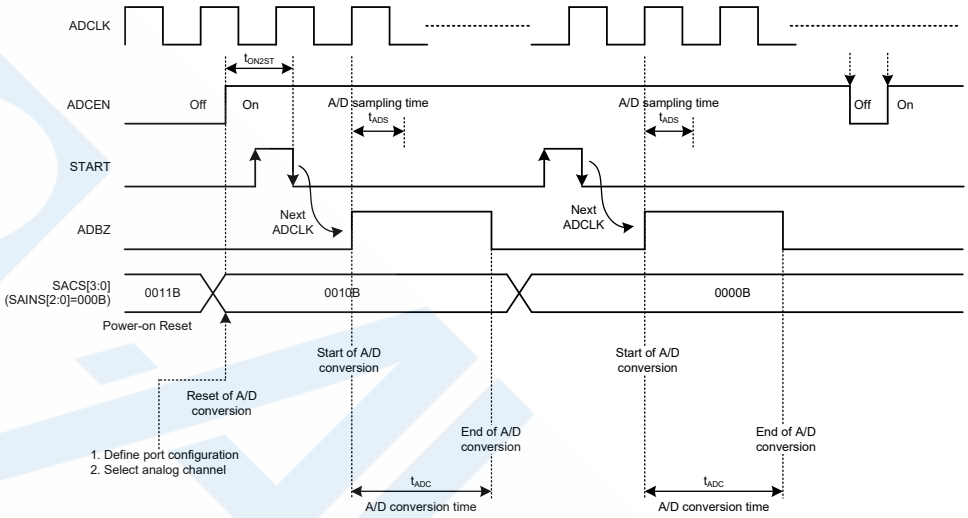
Controlling the power on/off function of the A/D converter circuitry is implemented using the ADCEN bit in the SADC0 register. This bit must be set high to power on the A/D converter. When the ADCEN bit is set high to power on the A/D converter internal circuitry, a certain delay, as indicated in the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

Conversion Rate and Timing Diagram

A complete A/D conversion contains two parts, data sampling and data conversion. A total of 16 A/D clock periods for an external input A/D conversion which is defined as t_{ADC} are necessary.

Maximum single A/D conversion rate = 1/(A/D clock period × 16)

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

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 SACKS2~SACKS0 in the SADC1 register.
- Step 2
  Enable the A/D converter by setting the ADCEN bit in the SADC0 register to "1".
- Step 3
  Select which external channel is to be connected to the internal A/D converter by correctly configuring the SACS3~SACS0 bits in the SADC0 register. The corresponding pin should be configured as A/D input function by configuring the relevant pin-shared function control bits.
- Step 4
  Select A/D converter output data format by setting the ADRFS bit in the SADC0 register.
- Step 5
  If the A/D conversion interrupt is used, the interrupt control registers must be correctly configured to ensure the A/D interrupt function is active. The master interrupt control bit, EMI, and the A/D conversion interrupt control bit, ADE, must both be set high in advance.
- Step 6
  The A/D conversion procedure can now be initialised by setting the START bit from low to high and then low again.

- Step 7
  If A/D conversion is in progress, the ADBZ flag will be set high. After the A/D conversion process is complete, the ADBZ flag will go low and then the output data can be read from SADOH and SADOL registers.

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

Programming Considerations

During microcontroller operations where the A/D converter is not being used, the A/D internal circuitry can be switched off to reduce power consumption, by clearing bit ADCEN to 0 in the SADC0 register. When this happens, the internal A/D converter circuits will not consume power irrespective of what analog voltage is applied to their input lines. If the A/D converter input lines are used as normal I/Os, then care must be taken as if the input voltage is not at a valid logic level, then this may lead to some increase in power consumption.

A/D Conversion Function

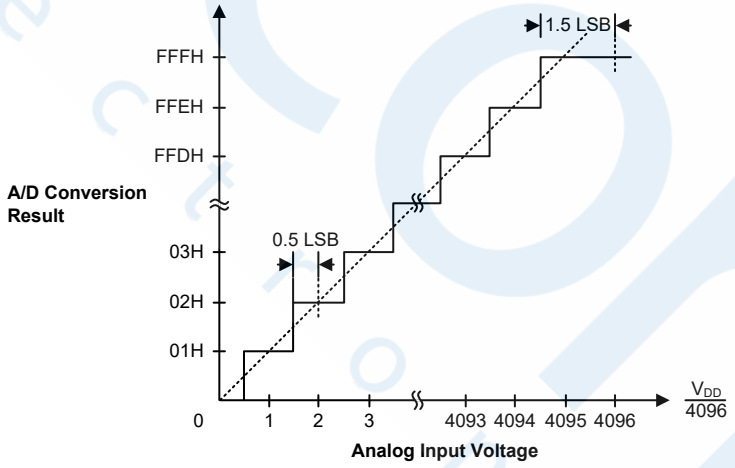
As these devices contain a 12-bit A/D converter, its full-scale converted digitised value is equal to FFFH. Since the full-scale analog input value is equal to the actual A/D converter reference voltage, V_{DD}, this gives a single bit analog input value of V_{DD} divided by 4096.

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

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

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

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



Ideal A/D Transfer Function

A/D Conversion Programming Examples

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

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

```
clr ADE                ; disable ADC interrupt
mov a,03h              ; select fSYS/8 as A/D clock
mov SADC1,a
mov a,02h              ; set PBS0 to configure pin AN0
mov PBS0,a
mov a,20h
mov SADC0,a            ; enable A/D 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  ADBZ               ; poll the SADC0 register ADBZ bit to detect end of A/D conversion
jmp polling_EOC        ; continue polling
mov a,SADOL            ; read low byte conversion result value
mov SADOL_buffer,a     ; save result to user defined register
mov a,SADOH            ; read high byte conversion result value
mov SADOH_buffer,a     ; save result to user defined register
:
jmp start_conversion   ; start next A/D conversion
```

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

```
clr ADE                ; disable ADC interrupt
mov a,03h              ; select fSYS/8 as A/D clock
mov SADC1,a
mov a,02h              ; set PBS0 to configure pin AN0
mov PBS0,a
mov a,20h
mov SADC0,a            ; enable A/D 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 ADC interrupt request flag
set ADE                ; enable ADC interrupt
set EMI                ; enable global interrupt
:
ADC_ISR:               ; ADC interrupt service routine
mov acc_stack,a        ; save ACC to user defined memory
mov a,STATUS
mov status_stack,a     ; save STATUS to user defined memory
:
mov a,SADOL            ; read low byte conversion result value
mov SADOL_buffer,a     ; save result to user defined register
mov a,SADOH            ; read high byte conversion result value
mov SADOH_buffer,a     ; save result to user defined register
:
EXIT_INT_ISR:
mov a,status_stack
mov STATUS,a           ; restore STATUS from user defined memory
mov a,acc_stack        ; restore ACC from user defined memory
reti
```

Universal Serial Interface Module – USIM

These devices contain a Universal Serial Interface Module, which includes the four-line SPI interface, the two-line I²C interface and the two-line/single-wire UART interface types, to allow an easy method of communication with external peripheral hardware. Having relatively simple communication protocols, these serial interface types allow the microcontroller to interface to external SPI, I²C or UART based hardware such as sensors, Flash or EEPROM memory, etc. The USIM interface pins are pin-shared with other I/O pins therefore the USIM interface functional pins must first be selected using the corresponding pin-shared function selection bits. As all the interface types share the same pins and registers, the choice of whether the UART, SPI or I²C type is used is made using the UART mode selection bit, named UMD, and the SPI/I²C operating mode control bits, named SIM2~SIM0, in the SIMC0 register. These pull-high resistors of the USIM pin-shared I/O are selected using pull-high control registers when the USIM function is enabled and the corresponding pins are used as USIM input pins.

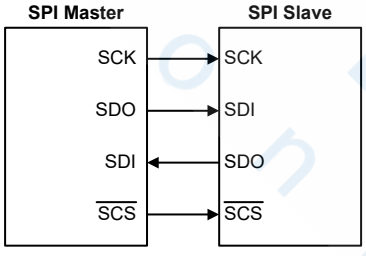
SPI Interface

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

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

SPI Interface Operation

The SPI interface is a full duplex synchronous serial data link. It is a four line interface with pin names SDI, SDO, SCK and $\overline{SCS}$. Pins SDI and SDO are the Serial Data Input and Serial Data Output lines, the SCK pin is the Serial Clock line and $\overline{SCS}$ is the Slave Select line. As the SPI interface pins are pin-shared with normal I/O pins and with the I²C/UART function pins, the SPI interface pins must first be selected by configuring the pin-shared function selection bits and setting the correct bits in the SIMC0 and SIMC2 registers. Communication between devices connected to the SPI interface is carried out in a slave/master mode with all data transfer initiations being implemented by the master. The Master also controls the clock signal. As the device only contains a single $\overline{SCS}$ pin only one slave device can be utilised. The $\overline{SCS}$ pin is controlled by software, set CSEN bit to 1 to enable $\overline{SCS}$ pin function, set CSEN bit to 0 the $\overline{SCS}$ pin will be floating state.



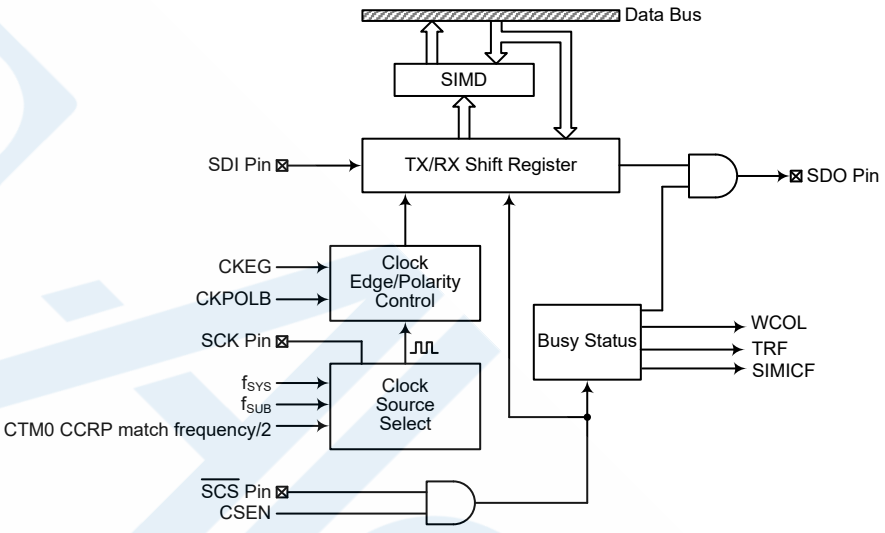
SPI Master/Slave Connection

The SPI function offers the following features:

- Full duplex synchronous data transfer
- Both Master and Slave modes

- LSB first or MSB first data transmission modes
- Transmission complete flag
- Rising or falling active clock edge

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



SPI Block Diagram

SPI Registers

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

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

SPI Register List

SPI Data Register

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

"x": unknown

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

SPI Control Registers

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

• **SIMC0 Register**

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

Bit 7~5 **SIM2~SIM0**: USIM SPI/I²C 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 CTM0 CCRP match frequency/2
101: SPI slave mode
110: I²C slave mode
111: Unused mode

When the UMD bit is cleared to zero, these bits setup the SPI or I²C operating mode of the USIM 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 CTM0 and f_{SUB}. If the SPI Slave Mode is selected then the clock will be supplied by an external Master device.

Bit 4 **UMD**: UART mode selection bit
0: SPI or I²C mode
1: UART mode

This bit is used to select the UART mode. When this bit is cleared to zero, the actual SPI or I²C mode can be selected using the SIM2~SIM0 bits. Note that the UMD bit must be cleared to zero for SPI or I²C mode.

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

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

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

The bit is the overall on/off control for the USIM SPI/I²C interface. When the SIMEN bit is cleared to zero to disable the USIM SPI/I²C interface, the SDI, SDO, SCK and $\overline{SCS}$, or SDA and SCL lines will lose their SPI or I²C function and the USIM operating current will be reduced to a minimum value. When the bit is high the USIM SPI/I²C interface is enabled. If the USIM is configured to operate as an SPI interface via the UMD and SIM2~SIM0 bits, the contents of the SPI control registers will remain at the

previous settings when the SIMEN bit changes from low to high and should therefore be first initialised by the application program. If the USIM is configured to operate as an I²C interface via the UMD and SIM2~SIM0 bits and the SIMEN bit changes from low to high, the contents of the I²C control bits such as HTX and TXAK will remain at the previous settings and should therefore be first initialised by the application program while the relevant I²C flags such as HCF, HAAS, HBB, SRW and RXAK will be set to their default states.

Bit 0 **SIMICF**: USIM SPI Incomplete Flag
0: USIM SPI incomplete condition is not occurred
1: USIM SPI incomplete condition is occurred
This bit is only available when the USIM is configured to operate in an SPI slave mode. If the SPI operates in the slave mode with the SIMEN and CSEN bits both being set high but the $\overline{SCS}$ line is pulled high by the external master device before the SPI data transfer is completely finished, the SIMICF bit will be set high 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
POR	0	0	0	0	0	0	0	0

Bit 7~6 **D7~D6**: Undefined bits
These bits can be read or written by application program.

Bit 5 **CKPOLB**: SPI clock line base condition selection
0: The SCK line will be high when the clock is inactive
1: The SCK line will be low when the clock is inactive
The CKPOLB bit determines the base condition of the clock line, if the bit is high, then the SCK line will be low when the clock is inactive. When the CKPOLB bit is low, then the SCK line will be high when the clock is inactive.

Bit 4 **CKEG**: SPI SCK active clock edge type selection
CKPOLB=0
0: SCK is high base level and data capture at SCK rising edge
1: SCK is high base level and data capture at SCK falling edge
CKPOLB=1
0: SCK is low base level and data capture at SCK falling edge
1: SCK is low base level and data capture at SCK rising edge
The CKEG and CKPOLB bits are used to setup the way that the clock signal outputs and inputs data on the SPI bus. These two bits must be configured before data transfer is executed otherwise an erroneous clock edge may be generated. The CKPOLB bit determines the base condition of the clock line, if the bit is high, then the SCK line will be low when the clock is inactive. When the CKPOLB bit is low, then the SCK line will be high when the clock is inactive. The CKEG bit determines active clock edge type which depends upon the condition of CKPOLB bit.

Bit 3 **MLS**: SPI data shift order
0: LSB first
1: MSB first
This is the data shift select bit and is used to select how the data is transferred, either MSB or LSB first. Setting the bit high will select MSB first and low for LSB first.

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

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

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

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

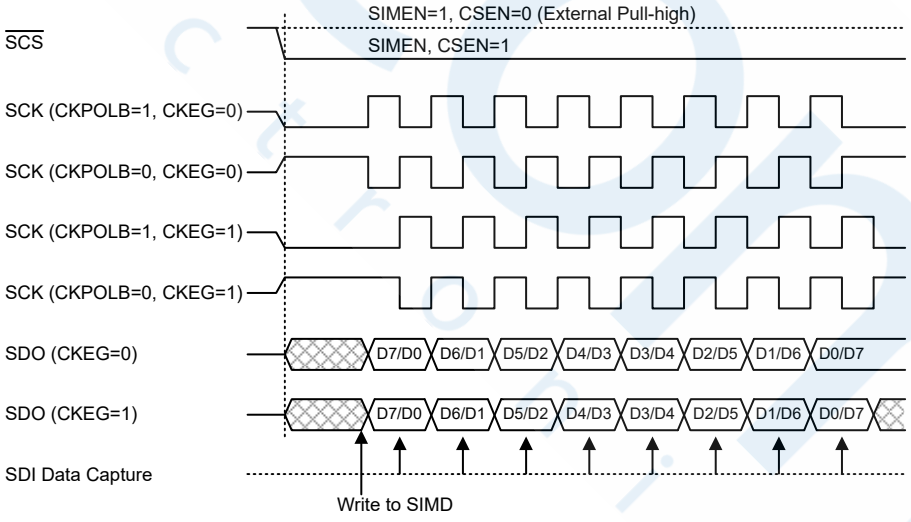
Bit 0 **TRF**: SPI Transmit/Receive complete flag
0: SPI data is being transferred
1: SPI data transmission is completed

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

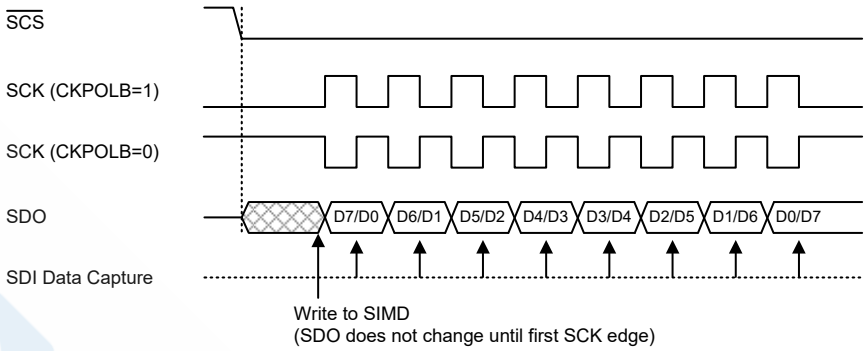
SPI Communication

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

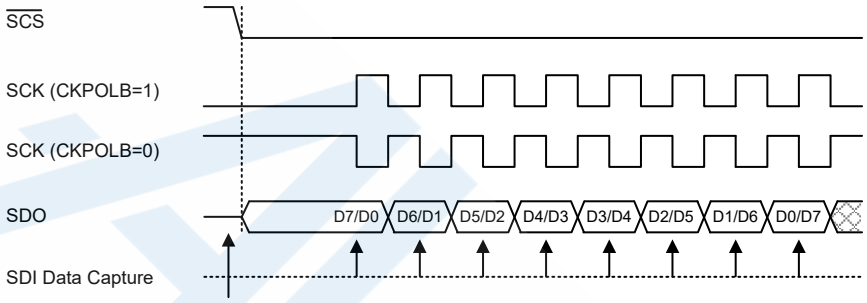
The SPI will continue to function in certain IDLE Modes if the clock source used by the SPI interface is still active.



SPI Master Mode Timing

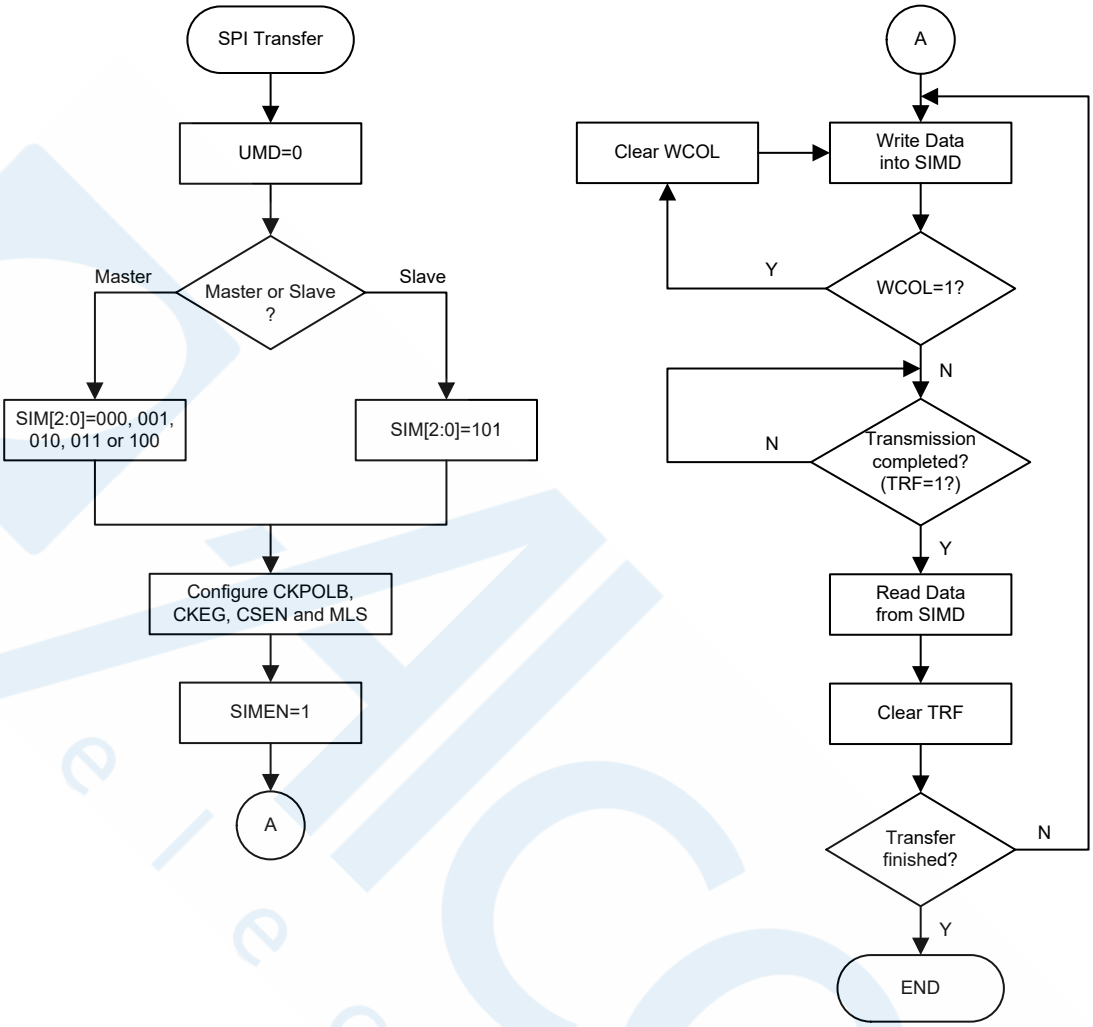


SPI Slave Mode Timing – CKEG=0



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

SPI Bus Enable/Disable

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

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

SPI Operation Steps

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

The CSEN bit in the SIMC2 register controls the $\overline{SCS}$ pin function of the SPI interface. Setting this bit high will enable the SPI interface by allowing the $\overline{SCS}$ line to be active, which can then be used to control the SPI interface. If the CSEN bit is low, the SPI interface will be disabled and the $\overline{SCS}$ line will be in a floating condition and can therefore not be used for control of the SPI interface. If the CSEN bit and the SIMEN bit in the SIMC0 are set high, this will place the SDI

line in a floating condition and the SDO line high. If in Master Mode the SCK line will be either high or low depending upon the clock polarity selection bit CKPOLB in the SIMC2 register. If in Slave Mode the SCK line will be in a floating condition. If the SIMEN bit is low, then the bus will be disabled and $\overline{SCS}$, SDI, SDO and SCK will all become I/O pins or the other functions using the corresponding pin-shared control bits. In the Master Mode the Master will always generate the clock signal. The clock and data transmission will be initiated after data has been written into the SIMD register. In the Slave Mode, the clock signal will be received from an external master device for both data transmission and reception. The following sequences show the order to be followed for data transfer in both Master and Slave Mode.

Master Mode:

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

Slave Mode:

- Step 1
  Select the SPI Slave mode using the UMD and SIM2~SIM0 bits in the SIMC0 control register
- Step 2
  Setup the CSEN bit and setup the MLS bit to choose if the data is MSB or LSB first, this setting must be the same with the Master devices.
- Step 3
  Setup the SIMEN bit in the SIMC0 control register to enable the SPI interface.
- Step 4
  For write operations: write the data to the SIMD register, which will actually place the data into the TXRX buffer. Then wait for the master clock SCK and $\overline{SCS}$ signal. After this, go to step 5.

For read operations: the data transferred in on the SDI line will be stored in the TXRX buffer until all the data has been received at which point it will be latched into the SIMD register.

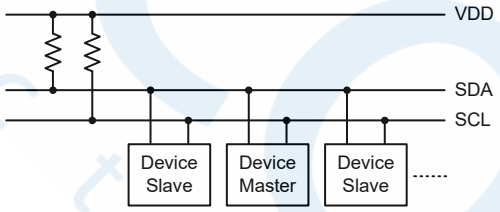
- Step 5
  Check the WCOL bit if set high then a collision error has occurred so return to step 4. If equal to zero then go to the following step.
- Step 6
  Check the TRF bit or wait for a USIM SPI serial bus interrupt.
- Step 7
  Read data from the SIMD register.
- Step 8
  Clear TRF.
- Step 9
  Go to step 4.

Error Detection

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

I²C Interface

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

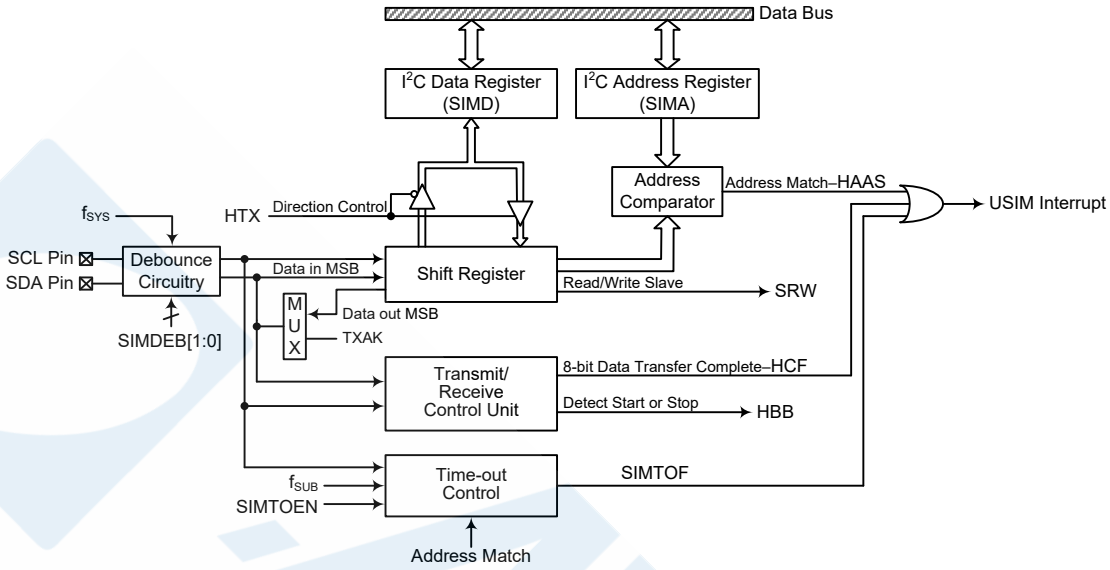


I²C Master Slave Bus Connection

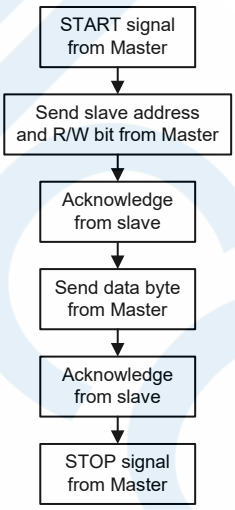
I²C Interface Operation

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

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



I²C Block Diagram



I²C Interface Operation

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

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

I²C Minimum f_{SYS} Frequency Requirements

I²C Registers

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

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

I²C Register List

I²C Data Register

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

- **SIMD Register**

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

"x": unknown

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

I²C Address Register

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

- **SIMA Register**

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

Bit 7~1 **SIMA6~SIMA0**: I²C slave address
SIMA6~SIMA0 is the 7-bit I²C slave address.

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

I²C Control Registers

There are three control registers for the I²C interface, SIMC0, SIMC1 and SIMTOC. The SIMC0 register is used to control the enable/disable function and to set the data transmission clock

frequency. The SIMC1 register contains the relevant flags which are used to indicate the I²C communication status. Another register, SIMTOC, is used to control the I²C time-out function and is described in the corresponding section.

• **SIMC0 Register**

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

Bit 7~5 **SIM2~SIM0**: USIM SPI/I²C 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 CTM0 CCRP match frequency/2
101: SPI slave mode
110: I²C slave mode
111: Unused mode
When the UMD bit is cleared to zero, these bits setup the SPI or I²C operating mode of the USIM 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 CTM0 and f_{SUB}. If the SPI Slave Mode is selected then the clock will be supplied by an external Master device.

Bit 4 **UMD**: UART mode selection bit
0: SPI or I²C mode
1: UART mode
This bit is used to select the UART mode. When this bit is cleared to zero, the actual SPI or I²C mode can be selected using the SIM2~SIM0 bits. Note that the UMD bit must be cleared to zero for SPI or I²C mode.

Bit 3~2 **SIMDEB1~SIMDEB0**: I²C Debounce Time Selection
00: No debounce
01: 2 system clock debounce
10: 4 system clock debounce
11: 4 system clock debounce
These bits are used to select the I²C debounce time when the USIM is configured as the I²C interface function by setting the UMD bit to "0" and SIM2~SIM0 bits to "110".

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

Bit 0 **SIMICF**: USIM SPI Incomplete Flag
This bit is only available when the USIM is configured to operate in an SPI slave mode. Refer to the SPI register section.

• **SIMC1 Register**

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

Bit 7 **HCF**: 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 cleared to "0" when the bus is free which will occur when a STOP signal is detected.

Bit 4 **HTX**: I²C slave device is transmitter or 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 bits of data, this bit will be transmitted to the bus on the 9th clock from the slave device. The slave device must always set TXAK bit to "0" before further data is received.

Bit 2 **SRW**: I²C Slave Read/Write flag
0: Slave device should be in receive mode
1: Slave device should be in transmit mode
The SRW flag is the I²C Slave Read/Write flag. This flag determines whether the master device wishes to transmit or receive data from the I²C bus. When the transmitted address and slave address is match, that is when the HAAS flag is set high, the slave device will check the SRW flag to determine whether it should be in transmit mode or receive mode. If the SRW flag is high, the master is requesting to read data from the bus, so the slave device should be in transmit mode. When the SRW flag is zero, the master will write data to the bus, therefore the slave device should be in receive mode to read this data.

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

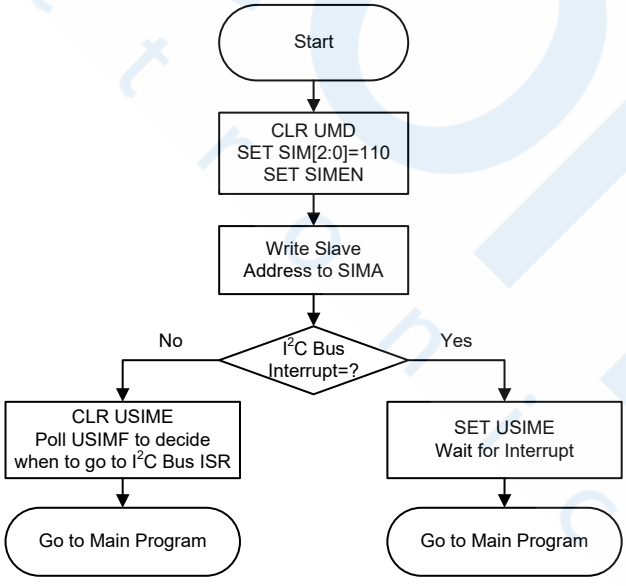
Bit 0 **RXAK**: I²C Bus Receive acknowledge flag
0: Slave receive acknowledge flag
1: Slave does not receive acknowledge flag

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

I²C Bus Communication

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

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



I²C Bus Initialisation Flow Chart

I²C Bus Start Signal

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

I²C Slave Address

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

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

I²C Bus Read/Write Signal

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

I²C Bus Slave Address Acknowledge Signal

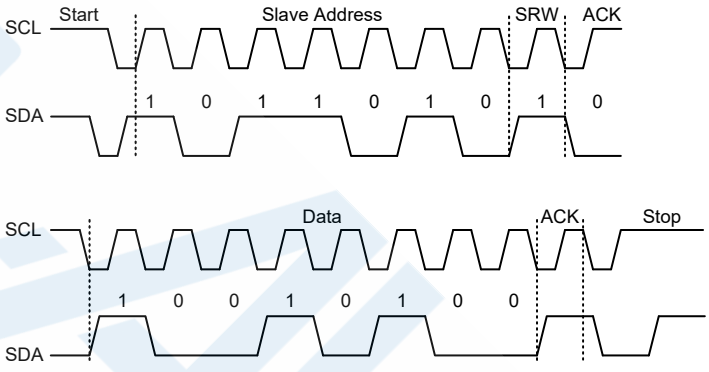
After the master has transmitted a calling address, any slave device on the I²C bus, whose own internal address matches the calling address, must generate an acknowledge signal. The acknowledge signal will inform the master that a slave device has accepted its calling address. If no acknowledge signal is received by the master then a STOP signal must be transmitted by the master to end the communication. When the HAAS flag is high, the addresses have matched and the slave device must check the SRW flag to determine if it is to be a transmitter or a receiver. If the SRW flag is high, the slave device should be setup to be a transmitter so the HTX bit in the SIMC1 register should be set to "1". If the SRW flag is low, then the microcontroller slave device should be setup as a receiver and the HTX bit in the SIMC1 register should be 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 bits of data, the receiver must transmit an acknowledge signal, level "0", before it can receive the next data byte. If the slave transmitter does not receive an acknowledge bit signal from the master receiver, then the slave transmitter will release the SDA line to allow the master to send

a STOP signal to release the I²C Bus. The corresponding data will be stored in the SIMD register. If setup as a transmitter, the slave device must first write the data to be transmitted into the SIMD register. If setup as a receiver, the slave device must read the transmitted data from the SIMD register.

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

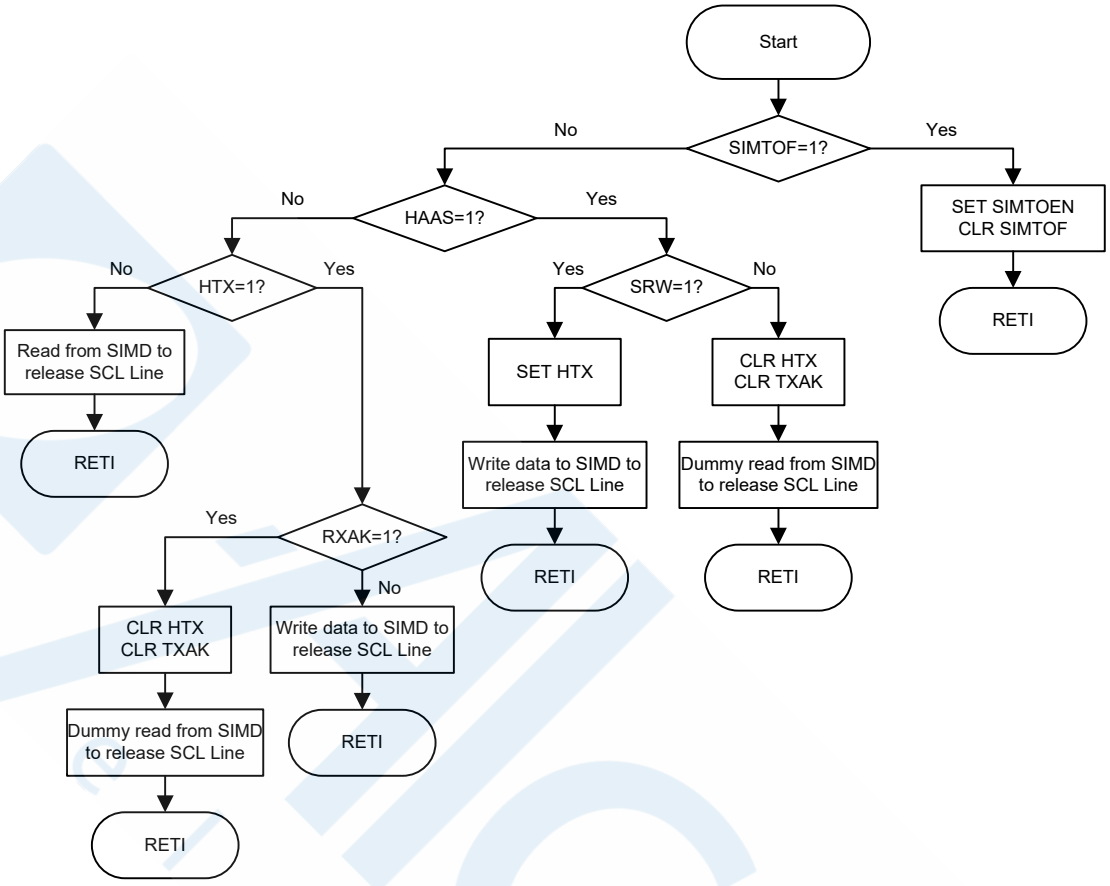


S=Start (1 bit)
SA=Slave Address (7 bits)
SR=SRW bit (1 bit)
M=Slave device send acknowledge bit (1 bit)
D=Data (8 bits)
A=ACK (RXAK bit for transmitter, TXAK bit for receiver, 1 bit)
P=Stop (1 bit)

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

I²C Communication Timing Diagram

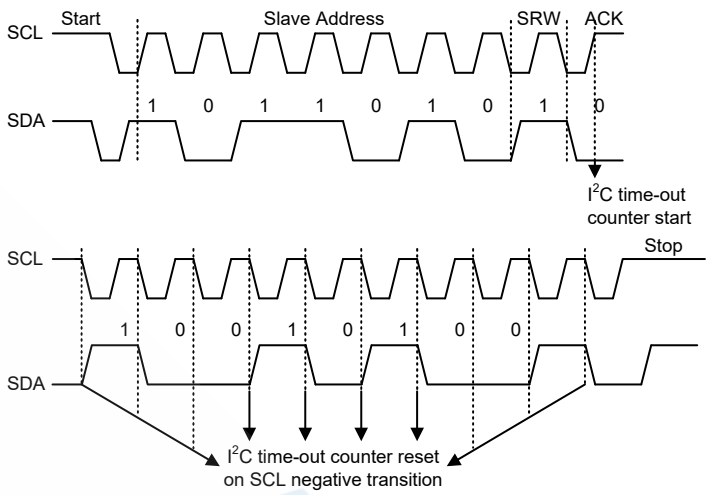
Note: When a slave address is matched, the device must be placed in either the transmit mode and then write data to the SIMD register, or in the receive mode where it must implement a dummy read from the SIMD register to release the SCL line.



I²C Bus ISR Flow Chart

I²C Time-out Control

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



I²C Time-out

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

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

I²C Registers after Time-out

The SIMTOF flag can be cleared by the application program. There are 64 time-out periods which can be selected using the SIMTOS[5:0] bits in the SIMTOC register. The time-out time is given by the formula: (1~64)×(32/f_{SUB}). This gives a time-out period which ranges from about 1ms to 64ms.

- **SIMTOC Register**

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

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

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

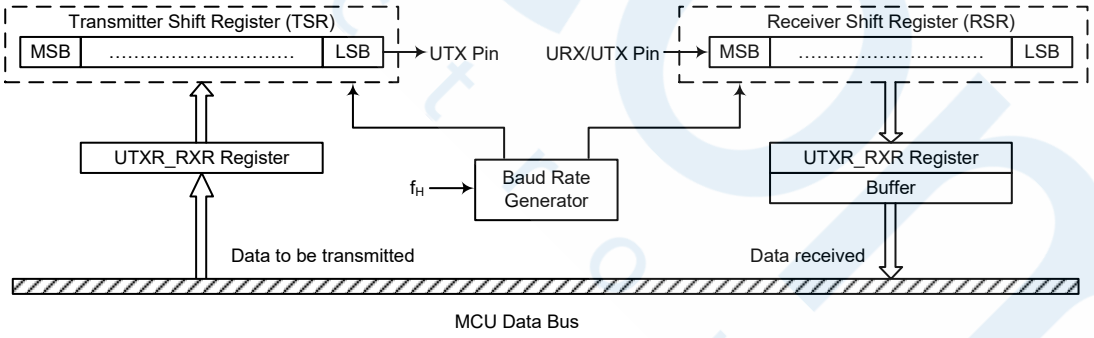
Bit 5~0 **SIMTOS5~SIMTOS0**: USIM I²C Time-out period selection
I²C time-out clock source is f_{SUB}/32.
I²C time-out time is equal to (SIMTOS[5:0]+1)×(32/f_{SUB}).

UART Interface

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

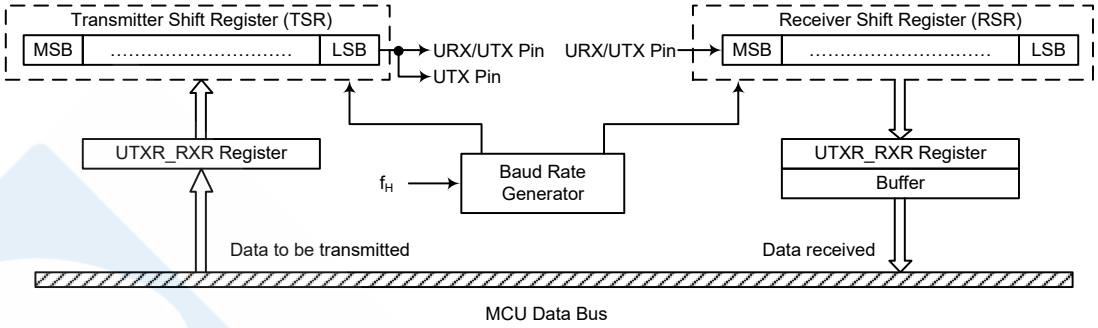
The integrated UART function contains the following features:

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



Note: For the BC66F2235 device, the UART function is unavailable when USWM=0 and it operates properly only when the Single Wire Mode is enabled.

UART Data Transfer Block Diagram – USWM=0



UART Data Transfer Block Diagram – USWM=1

UART External Pins

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

UART Single Wire Mode

The UART function also supports a Single Wire Mode communication which is selected using the USWM bit in the UUCR3 register. When the USWM bit is set high, the UART function will be in the single wire mode. In the single wire mode, a single URX/UTX pin can be used to transmit and receive data depending upon the corresponding control bits. When the URXEN bit is set high, the URX/UTX pin is used as a receiver pin. When the URXEN bit is cleared to zero and the UTXEN bit is set high, the URX/UTX pin will act as a transmitter pin.

It is recommended not to set both the URXEN and UTXEN bits high in the single wire mode. If both the URXEN and UTXEN bits are set high, the URXEN bit will have the priority and the UART will act as a receiver.

It is important to note that the functional description in this UART chapter, which is described from the full-duplex communication standpoint, also applies to the half-duplex (single wire mode) communication except the pin usage. In the single wire mode, the UTX pin mentioned in this chapter should be replaced by the URX/UTX pin to understand the whole UART single wire mode function.

In the single wire mode, the data can also be transmitted on the UTX pin in a transmission operation with proper software configurations. Therefore, the data will be output on the URX/UTX and UTX pins.

UART Data Transfer Scheme

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

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

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

UART Status and Control Registers

There are seven control registers associated with the UART function. The UMD bit in the SIMC0 register can be used to select the UART interface. The USWM bit in the UUCR3 register is used to enable/disable the UART Single Wire Mode. The UUSR, UUCR1 and UUCR2 registers control the overall function of the UART, while the UBRG register controls the Baud rate. The actual data to be transmitted and received on the serial interface is managed through the UTXR_RXR data register. Note that UART related registers and their POR values are only available when the UART mode is selected by setting the UMD bit in the SIMC0 register to "1".

Register Name	Bit							
	7	6	5	4	3	2	1	0
SIMC0	SIM2	SIM1	SIM0	UMD	SIMDEB1	SIMDEB0	SIMEN	SIMICF
UUSR	UPERR	UNF	UFERR	UOERR	URIDLE	URXIF	UTIDLE	UTXIF
UUCR1	UREN	UBNO	UPREN	UPRT	USTOPS	UTXBRK	URX8	UTX8
UUCR2	UTXEN	URXEN	UBRGH	UADDEN	UWAKE	URIE	UTIIE	UTEIE
UUCR3	—	—	—	—	—	—	—	USWM
UTXR_RXR	UTXRX7	UTXRX6	UTXRX5	UTXRX4	UTXRX3	UTXRX2	UTXRX1	UTXRX0
UBRG	UBRG7	UBRG6	UBRG5	UBRG4	UBRG3	UBRG2	UBRG1	UBRG0

UART Register List

- **SIMC0 Register**

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

Bit 7~5 **SIM2~SIM0**: USIM SPI/I^2C Operating Mode Control
When the UMD bit is cleared to zero, these bits setup the SPI or I^2C operating mode of the USIM function. Refer to the SPI or I^2C register section for more details.

Bit 4 **UMD**: UART mode selection bit
0: SPI or I^2C mode
1: UART mode
This bit is used to select the UART mode. When this bit is cleared to zero, the actual SPI or I^2C mode can be selected using the SIM2~SIM0 bits. Note that the UMD bit must be cleared to zero for SPI or I^2C mode.

Bit 3~2 **SIMDEB1~SIMDEB0**: I^2C Debounce Time Selection
Refer to the I^2C register section.

Bit 1 **SIMEN**: USIM SPI/I^2C Enable Control
This bit is only available when the USIM is configured to operate in an SPI or I^2C mode with the UMD bit cleared to zero. Refer to the SPI or I^2C register section for more details.

Bit 0 **SIMICF**: USIM SPI Incomplete Flag
Refer to the SPI register section.

- **UUSR Register**

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

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

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

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

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

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

Bit 3 **URIDLE**: Receiver status
0: Data reception is in progress (Data being received)
1: No data reception is in progress (Receiver is idle)
The URIDLE flag is the receiver status flag. When this read only flag is "0", it indicates that the receiver is between the initial detection of the start bit and the completion of the stop bit. When the flag is "1", it indicates that the receiver is idle. Between the completion of the stop bit and the detection of the next start bit, the

URIDLE bit is "1" indicating that the UART receiver is idle and the URX/UTX pin stays in logic high condition.

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

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

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

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

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

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

• UUCR1 Register

The UUCR1 register together with the UUCR2 and UUCR3 registers are the three 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, single wire mode communication etc. Further explanation on each of the bits is given below:

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

"x": unknown

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

The UREN bit is the UART enable bit. When this bit is equal to "0", the UART will be disabled and the URX/UTX pin as well as the UTX pin will be set in a floating state. When the bit is equal to "1", the UART will be enabled if the UMD bit is set and the UTX and URX/UTX pins will function as defined by the USWM mode selection bit together with the UTXEN and URXEN enable control bits.

When the UART is disabled, it will empty the buffer so any character remaining in the buffer will be discarded. In addition, the value of the baud rate counter will be reset. If the UART is disabled, all error and status flags will be reset. Also the UTXEN, URXEN, UTXBRK, URXIF, UOERR, UFERR, UPERR and UNF bits will be cleared to zero, while the UTIDLE, UTXIF and URIDLE bits will be set high. Other control bits in UUCR1, UUCR2, UUCR3 and UBRG registers will remain unaffected. If the UART is active and the UREN bit is cleared to zero, all pending transmissions and receptions will be terminated and the module will be reset as defined above. When the UART is re-enabled, it will restart in the same configuration.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

• **UUCR2 Register**

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

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

Bit 7 **UTXEN**: UART Transmitter enabled control
0: UART transmitter is disabled
1: UART transmitter is enabled

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

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

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

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

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

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

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

The bit named UADDEN is the address detect function enable control bit. When this bit is equal to "1", the address detect function is enabled. When it occurs, if the 8th bit, which corresponds to UTXRX7 if UBNO=0 or the 9th bit, which corresponds to URX8 if UBNO=1, has a value of "1", then the received word will be identified as an address, rather than data. If the corresponding interrupt is enabled, an interrupt request will be generated each time the received word has the address bit set, which is the 8th or 9th bit depending on the value of UBNO. If the address bit known as the 8th or 9th bit of the received word is "0" with the address detect function being enabled, an interrupt will not be generated and the received data will be discarded.

Bit 3 **UWAKE**: URX/UTX pin wake-up UART function enable control
0: URX/UTX pin wake-up UART function is disabled
1: URX/UTX pin wake-up UART function is enabled
This bit is used to control the wake-up UART function when a falling edge on the URX/UTX pin occurs. Note that this bit is only available when the UART clock (f_H) is switched off. There will be no URX/UTX pin wake-up UART function if the UART clock (f_H) exists. If the UWAKE bit is set high as the UART clock (f_H) is switched off, a UART wake-up request will be initiated when a falling edge on the URX/UTX pin occurs. When this request happens and the corresponding interrupt is enabled, an URX/UTX pin wake-up UART interrupt will be generated to inform the MCU to wake up the UART function by switching on the UART clock (f_H) via the application program. Otherwise, the UART function cannot resume even if there is a falling edge on the URX/UTX pin when the UWAKE bit is cleared to zero.

Bit 2 **URIE**: Receiver interrupt enable control
0: Receiver related interrupt is disabled
1: Receiver related interrupt is enabled
This bit enables or disables the receiver interrupt. If this bit is equal to "1" and when the receiver overrun flag UOERR or receive data available flag URXIF is set, the USIM interrupt request flag USIMF will be set. If this bit is equal to "0", the USIM interrupt request flag USIMF will not be influenced by the condition of the UOERR or URXIF flags.

Bit 1 **UTIIE**: Transmitter Idle interrupt enable control
0: Transmitter idle interrupt is disabled
1: Transmitter idle interrupt is enabled
This bit enables or disables the transmitter idle interrupt. If this bit is equal to "1" and when the transmitter idle flag UTIDLE is set, due to a transmitter idle condition, the USIM interrupt request flag USIMF will be set. If this bit is equal to "0", the USIM interrupt request flag USIMF will not be influenced by the condition of the UTIDLE flag.

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

• UUCR3 Register

The UUCR3 register is used to enable the UART Single Wire Mode communication. As the name suggests in the single wire mode the UART communication can be implemented in one single line, URX/UTX, together with the control of the URXEN and UTXEN bits in the UUCR2 register.

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

Bit 7~1 Unimplemented, read as "0"

Bit 0 **USWM**: Single Wire Mode enable control
0: Disable, the URX/UTX pin is used as UART receiver function only
1: Enable, the URX/UTX pin can be used as UART receiver or transmitter function controlled by the URXEN and UTXEN bits
Note that when the Single Wire Mode is enabled, if both the URXEN and UTXEN bits are high, the URX/UTX pin will just be used as UART receiver input.
Note: For the BC66F2235 device, this bit should be set high when the UART mode is selected.

• **UTXR_RXR Register**

The UTXR_RXR register is the data register which is used to store the data to be transmitted on the UTX pin or being received from the URX/UTX pin.

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

"x": unknown

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

• **UBRG Register**

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

"x": unknown

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

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

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

Baud rate=f_H/[16×(N+1)] if UBRGH=1.

Baud Rate Generator

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

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

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

Calculating the Baud Rate and Error Values

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

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

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

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

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

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

UART Setup and Control

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

Enabling/Disabling the UART Interface

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

Clearing the UREN bit will disable the UTX and URX/UTX pin and allow these pins to be used as normal I/O or other pin-shared functional pins by configuring the corresponding pin-shared control bits. When the UART function is disabled the buffer will be reset to an empty condition, at the same time discarding any remaining residual data. Disabling the UART will also reset the error and status flags with bits UTXEN, URXEN, UTXBRK, URXIF, UOERR, UFERR, UPERR and UNF being cleared while bits UTIDLE, UTXIF and URIDLE will be set. The remaining control bits in the UUCR1, UUCR2, UUCR3 and UBRG registers will remain unaffected. If the UREN bit in the UUCR1 register is cleared while the UART is active, then all pending transmissions and receptions will be immediately suspended and the UART will be reset to a condition as defined above. If the UART is then subsequently re-enabled, it will restart again in the same configuration.

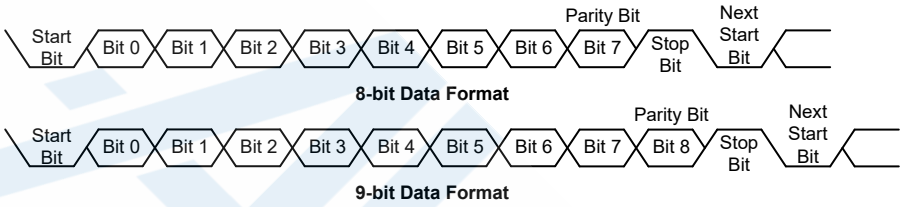
Data, Parity and Stop Bit Selection

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

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

Transmitter Receiver Data Format

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



8-bit Data Format

9-bit Data Format

UART Transmitter

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

Transmitting Data

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

- Make the correct selection of the UBNO, UPRT, UPREN and USTOPS bits to define the required word length, parity type and number of stop bits.

- Setup the UBRG register to select the desired baud rate.
- Set the UTXEN bit ensure that the UTX pin is used as a UART transmitter pin.
- Access the UUSR register and write the data that is to be transmitted into the UTXR_RXR register. Note that this step will clear the UTXIF bit.

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

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

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

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

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

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

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

Transmitting Break

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

UART Receiver

The UART is capable of receiving word lengths of either 8 or 9 bits. If the UBNO bit is set, the word length will be set to 9 bits with the MSB being stored in the URX8 bit of the UUCR1 register. At the receiver core lies the Receive Serial Shift Register, commonly known as the RSR. The data which is received on the URX/UTX pin input 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 URX/UTX 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 URX/UTX pin input is sampled three times by a majority detect circuit to determine the logic level that has been placed onto the URX/UTX 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 URX/UTX pin input, LSB first. In the read mode, the UTXR_RXR register forms a buffer between the internal bus and the receiver shift register. The UTXR_RXR register is a two byte deep FIFO data buffer, where two bytes can be held in the FIFO while a third byte can continue to be received. Note that the application program must ensure that the data is read from UTXR_RXR before the third byte has been completely shifted in, otherwise this third byte will be discarded and an overrun error UOERR will be subsequently indicated. The steps to initiate a data transfer can be summarised as follows:

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

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

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

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

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

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

Receiving Break

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

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

Idle Status

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

the next start bit, the URIDLE flag will have a high value, which indicates the receiver is in an idle condition.

Receiver Interrupt

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

Managing Receiver Errors

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

Overrun Error – UOERR

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

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

- The UOERR flag in the UUSR register will be set.
- The UTXR_RXR contents will not be lost.
- The shift register will be overwritten.
- An interrupt will be generated if the URIE bit is set.

The UOERR flag can be cleared by an access to the UUSR register followed by a read to the UTXR_RXR register.

Noise Error – UNF

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

- The read only noise flag, UNF, in the UUSR register will be set on the rising edge of the URXIF bit.
- Data will be transferred from the Shift register to the UTXR_RXR register.
- No interrupt will be generated. However this bit rises at the same time as the URXIF bit which itself generates an interrupt.

Note that the UNF flag is reset by a UUSR register read operation followed by a UTXR_RXR register read operation.

Framing Error – UFERR

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

Parity Error – UPERR

The read only parity error flag, UPERR, in the UUSR register, is set if the parity of the received word is incorrect. This error flag is only applicable if the parity is enabled, UPREN=1, and if the parity type, odd or even is selected. The read only UPERR flag and the received data will be recorded in the UUSR and UTXR_RXR registers respectively. It is cleared on any reset, it should

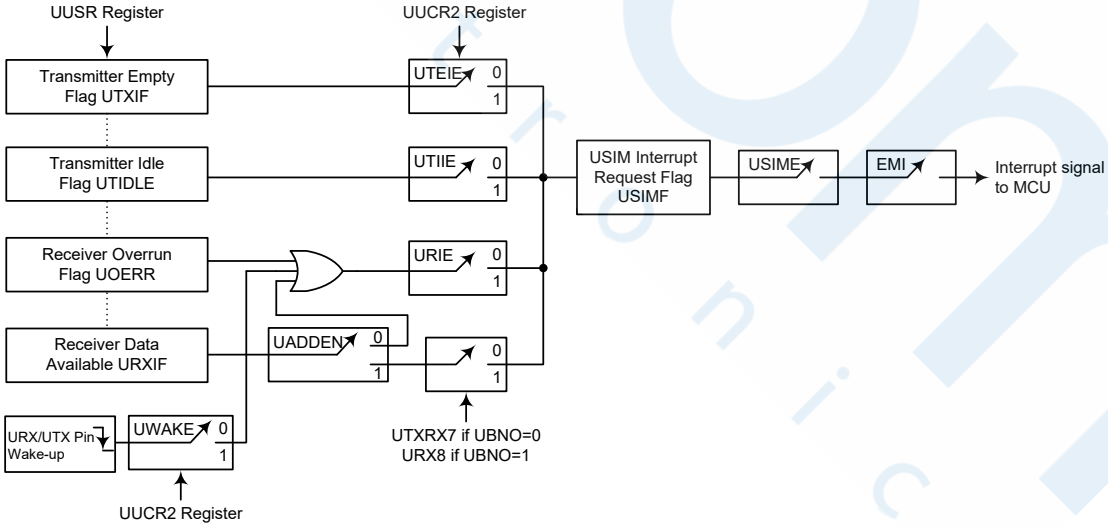
be noted that the flags, UFERR and UPERR, in the UUSR register should first be read by the application program before reading the data word.

UART Interrupt Structure

Several individual UART conditions can trigger an USIM interrupt. When these conditions exist, a low pulse will be generated to get the attention of the microcontroller. These conditions are a transmitter data register empty, transmitter idle, receiver data available, receiver overrun, address detect and an URX/UTX pin wake-up. When any of these conditions are created, if the global interrupt enable bit and the USIM interrupt control bit are enabled and the stack is not full, the program will jump to its corresponding interrupt vector where it can be serviced before returning to the main program. Four of these conditions have the corresponding UUSR register flags which will generate an USIM interrupt if its associated interrupt enable control bit in the UUCR2 register is set. The two transmitter interrupt conditions have their own corresponding enable control bits, while the two receiver interrupt conditions have a shared enable control bit. These enable bits can be used to mask out individual USIM UART mode interrupt sources.

The address detect condition, which is also an USIM UART mode interrupt source, does not have an associated flag, but will generate an USIM interrupt when an address detect condition occurs if its function is enabled by setting the UADDEN bit in the UUCR2 register. An URX/UTX pin wake-up, which is also an USIM UART mode interrupt source, does not have an associated flag, but will generate an USIM interrupt if the UART clock (f_H) source is switched off and the UWAKE and URIE bits in the UUCR2 register are set when a falling edge on the URX/UTX pin occurs. Note that in the event of an URX/UTX 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 stabilise before the system resumes normal operation.

Note that the UUSR register flags are read only and cannot be cleared or set by the application program, neither will they be cleared when the program jumps to the corresponding interrupt servicing routine, as is the case for some of the other interrupts. The flags will be cleared automatically when certain actions are taken by the UART, the details of which are given in the UART register section. The overall UART interrupt can be disabled or enabled by the USIM interrupt enable control bit in the interrupt control register of the microcontroller to decide whether the interrupt requested by the UART module is masked out or allowed.



UART Interrupt Structure

Address Detect Mode

Setting the Address Detect Mode bit, UADDEN, in the UUCR2 register, enables this special mode. If this bit is enabled then an additional qualifier will be placed on the generation of a Receiver Data Available interrupt, which is requested by the URXIF flag. If the UADDEN bit is enabled, then when data is available, an interrupt will only be generated, if the highest received bit has a high value. Note that the USIME and EMI interrupt enable bits must also be enabled for correct interrupt generation. This highest address bit is the 9th bit if UBNO=1 or the 8th bit if UBNO=0. If this bit is high, then the received word will be defined as an address rather than data. A Data Available interrupt will be generated every time the last bit of the received word is set. If the UADDEN bit is not enabled, then a Receiver Data Available interrupt will be generated each time the URXIF flag is set, irrespective of the data last bit status. The address detect mode and parity enable are mutually exclusive functions. Therefore if the address detect mode is enabled, then to ensure correct operation, the parity function should be disabled by resetting the parity enable bit UPREN to zero.

UADDEN	9th bit if UBNO=1, 8th bit if UBNO=0	USIM Interrupt Generated
0	0	√
	1	√
1	0	×
	1	√

UADDEN Bit Function

UART Power Down and Wake-up

When the UART clock (f_H) is off, the UART will cease to function, all clock sources to the module are shutdown. If the UART clock (f_H) is off while a transmission is still in progress, then the transmission will be paused until the UART clock source derived from the microcontroller is activated. In a similar way, if the MCU enters the IDLE or SLEEP mode while receiving data, then the reception of data will likewise be paused. When the MCU enters the IDLE or SLEEP mode, note that the UUSR, UUCR1, UUCR2, UUCR3, UTXR_RXR, as well as the UBRG register will not be affected. It is recommended to make sure first that the UART data transmission or reception has been finished before the microcontroller enters the IDLE or SLEEP mode.

The UART function contains a receiver URX/UTX pin wake-up function, which is enabled or disabled by the UWAKE bit in the UUCR2 register. If this bit, along with the UART mode selection bit, UMD, the UART enable bit, UREN, the receiver enable bit, URXEN and the receiver interrupt bit, URIE, are all set when the UART clock (f_H) is off, then a falling edge on the URX/UTX pin will trigger an URX/UTX pin wake-up UART interrupt. Note that as it takes certain system clock cycles after a wake-up, before normal microcontroller operation resumes, any data received during this time on the URX/UTX pin will be ignored.

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

Touch Key Function

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

Touch Key Structure

The touch keys are pin-shared with the I/O pins, with the desired function chosen via the corresponding pin-shared selection register bits. Keys are organised into several groups, with each group known as a module and having a module number, M0 to M3. Each module is a fully independent set of four Touch Keys and each Touch Key has its own oscillator. Each module contains its own control logic circuits and register set. Examination of the register names will reveal the module number it is referring to.

Device	Total Key Number	Touch Key Module	Touch Key
BC66F2235	8	M0	KEY1
		M1	KEY5~KEY6
		M2	KEY11~KEY12
		M3	KEY13~KEY15
BC66F2245	14	M0	KEY1~KEY4
		M1	KEY5~KEY6
		M2	KEY9~KEY12
		M3	KEY13~KEY16
BC66F2255	16	M0	KEY1~KEY4
		M1	KEY5~KEY8
		M2	KEY9~KEY12
		M3	KEY13~KEY16

Touch Key Structure

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. These devices have four Touch Key Modules.

Register Name	Description
TKTMR	Touch key time slot 8-bit counter preload register
TKC0	Touch key function control register 0
TKC1	Touch key function control register 1
TKC2	Touch key function control register 2
TK16DL	Touch key function 16-bit counter low byte
TK16DH	Touch key function 16-bit counter high byte
TKMn16DL	Touch key module n 16-bit C/F counter low byte
TKMn16DH	Touch key module n 16-bit C/F counter high byte
TKMnROL	Touch key module n reference oscillator capacitor selection low byte
TKMnROH	Touch key module n reference oscillator capacitor selection high byte
TKMnC0	Touch key module n control register 0
TKMnC1	Touch key module n control register 1
TKMnC2	Touch key module n control register 2
TKMnTH16L	Touch key module n 16-bit threshold low byte

Register Name	Description
TKMnTH16H	Touch key module n 16-bit threshold high byte
TKMnTHS	Touch key module n threshold comparison flag

Touch Key Function Register Definition (n=0~3)

Register Name	Bit							
	7	6	5	4	3	2	1	0
TKTMR	D7	D6	D5	D4	D3	D2	D1	D0
TKC0	TKRAMC	TKRCOV	TKST	TKCFOV	TK16OV	TKMOD1	TKMOD0	TKBUSY
TKC1	D7	D6	D5	TSCS	TK16S1	TK16S0	TKFS1	TKFS0
TKC2	—	—	—	—	—	TSC	ASMP1	ASMP0
TK16DL	D7	D6	D5	D4	D3	D2	D1	D0
TK16DH	D15	D14	D13	D12	D11	D10	D9	D8
TKMn16DL	D7	D6	D5	D4	D3	D2	D1	D0
TKMn16DH	D15	D14	D13	D12	D11	D10	D9	D8
TKMnROL	D7	D6	D5	D4	D3	D2	D1	D0
TKMnROH	—	—	—	—	—	—	D9	D8
TKMnC0	—	—	MnDFEN	—	MnSOFC	MnSOF2	MnSOF1	MnSOF0
TKMnC1	MnTSS	—	MnROEN	MnKOEN	MnK4EN	MnK3EN	MnK2EN	MnK1EN
TKMnC2	MnSK31	MnSK30	MnSK21	MnSK20	MnSK11	MnSK10	MnSK01	MnSK00
TKMnTH16L	D7	D6	D5	D4	D3	D2	D1	D0
TKMnTH16H	D15	D14	D13	D12	D11	D10	D9	D8
TKMnTHS	MnK4THF	MnK3THF	MnK2THF	MnK1THF	MnK4THS	MnK3THS	MnK2THS	MnK1THS

Touch Key Function Register List (n=0~3)

- **TKTMR Register**

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

Bit 7~0 **D7~D0**: Touch key time slot 8-bit counter preload register

The touch key time slot counter preload register is used to determine the touch key time slot overflow time. If the time slot unit period is obtained by a 5-bit counter, it is equal to 32 time slot clock cycles. If the time slot unit period is obtained by bypassing a 5-bit counter, it is equal to 1 time slot clock cycle. Therefore, the time slot counter overflow time is equal to the following equation shown.

If MnTSS=0, Time slot counter overflow time=(256-TKTMR[7:0])×32t_{TSC}, while if MnTSS=1, Time slot counter overflow time=(256-TKTMR[7:0])×t_{TSC}, where t_{TSC} is the time slot counter clock period.

• **TKC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	TKRAMC	TKRCOV	TKST	TKCFOV	TK16OV	TKMOD1	TKMOD0	TKBUSY
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R
POR	0	0	0	0	0	0	1	0

Bit 7 **TKRAMC**: Touch key data memory access control
0: Accessed by MCU
1: Accessed by touch key module

This bit determines that the touch key data memory is used by the MCU or the touch key module. However, the touch key module will have the priority to access the touch key data memory when the touch key module operates in the auto scan mode or the periodic auto scan mode, i.e., the TKST bit state is changed from 0 to 1 when the TKMOD1~TKMOD0 bits are set to 00, 10 or 11. After the touch key auto scan or the periodic auto scan operation is completed, i.e., the TKBUSY bit state is changed from 1 to 0, the touch key data memory access will be controlled by the TKRAMC bit. Therefore, it is recommended to set the TKRAMC bit to 1 when the touch key module operates in the auto scan mode or the periodic auto scan mode. Otherwise, the contents of the touch key data memory may be modified as this data memory space is configured by the touch key module followed by the MCU access. When the TKRAMC bit is set to 1, which means the touch key data memory is controlled by the touch key module, reading the touch key data memory by MCU will obtain uncertain values.

Bit 6 **TKRCOV**: Touch key time slot counter overflow flag
0: No overflow occurs
1: Overflow occurs

This bit can be accessed by application program. When this bit is set by touch key time slot counter overflow, the corresponding touch key TKRCOV interrupt request flag will be set. However, if this bit is set by application program, the touch key TKRCOV interrupt request flag will not be affected. Therefore, this bit cannot be set by application program but must be cleared to 0 by application program.

In the auto scan mode, if the time slot counter of module 0 or all modules, selected by the TSCS bit, overflows but the touch key auto scan operation is not completed, the TKRCOV bit will not be set. At this time, all touch key modules' 16-bit C/F counter, touch key function 16-bit counter and 5-bit time slot unit period counter will be automatically cleared but the 8-bit time slot counter will be reloaded from the 8-bit time slot counter preload register. When the touch key auto scan operation is completed, the TKRCOV bit and the Touch Key TKRCOV Interrupt request flag, TKRCOVF, will be set and all modules' key and reference oscillators will automatically stop. All touch key modules' 16-bit C/F counter, touch key function 16-bit counter, 5-bit time slot unit period counter and 8-bit time slot counter will be automatically switched off.

In the periodic auto scan mode, the TKRCOV bit is cleared to zero during the auto scan operation period. Only at the end of the last scan operation in the TB0 time-out cycle, the 16-bit C/F counter content will be written into the corresponding touch key data memory, and then the TKRCOV bit will be set high by the hardware circuit. The other actions in this mode are the same as those in the auto scan mode except the above mentioned.

In the manual scan mode, if the time slot counter of module 0 or all modules, selected by the TSCS bit, overflows, the TKRCOV bit and the Touch Key TKRCOV Interrupt request flag, TKRCOVF, will be set and all modules' key and reference oscillators will automatically stop. All touch key modules' 16-bit C/F counter, touch key function 16-bit counter, 5-bit time slot unit period counter and 8-bit time slot counter will be automatically switched off.

Bit 5 **TKST**: Touch key detection start control
0: Stopped or no operation
0→1: Start detection
In all modules, the touch key module 16-bit C/F counter, touch key function 16-bit counter and 5-bit time slot unit period counter will automatically be cleared when this bit is cleared to zero. However, the 8-bit programmable time slot counter will not be cleared. When this bit is changed from low to high, the touch key module 16-bit C/F counter, touch key function 16-bit counter, 5-bit time slot unit period counter and 8-bit time slot counter will be switched on together with the key and reference oscillators to drive the corresponding counters.

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

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

Bit 2~1 **TKMOD1~TKMOD0**: Touch key scan mode selection
00: Auto scan mode
01: Manual scan mode
10/11: Periodic auto scan mode
In the manual scan mode, the reference oscillator capacitor value should be properly configured before the scan operation begins and the touch key module 16-bit C/F counter value should be read by application program after the scan operation finishes.
In the auto scan mode, the data movement which is described above is implemented by hardware. The individual reference oscillator capacitor value and 16-bit C/F counter content for all the scanned keys will be read from and written into a dedicated Touch Key Data Memory area. In the auto scan mode, the keys to be scanned can be arranged in a specific sequence which is determined by the MnSK3[1:0]~MnSK0[1:0] bits in the TKMnC2 register. The scan operation will not be stopped until all arranged keys are scanned.
In the periodic auto scan mode, the touch key scan operation will be implemented automatically on a periodic basis, which can be determined by the ASMP1~ASMP0 bits in the TKC2 register. Only at the end of the last scan operation in the TB0 time-out cycle, the 16-bit C/F counter content for all scanned keys will be written into the corresponding touch key data memory. In addition, when any key C/F counter value is less than the lower threshold if MnKmTHS=0, or larger than the upper threshold if MnKmTHS=1, the TKTH signal will be set high. The other actions in this mode are the same as those in the auto scan mode except the above mentioned.

Bit 0 **TKBUSY**: Touch key scan operation busy flag
0: Not busy – No scan operation is executed or scan operation is completed
1: Busy – Scan operation is executing
This bit indicates whether the touch key scan operation is executing or not. It is set to 1 when the TKST bit is set high to start the scan operation.
In the manual scan mode this bit is cleared to 0 automatically when the time slot counter of module 0 or all modules, selected by the TSCS bit, overflows. In the auto scan mode this bit is cleared to 0 automatically when the touch key scan operation is completed. In the periodic auto scan mode this bit is cleared to 0 automatically when the last scan operation in the TB0 time-out cycle is completed, or when any key C/F counter value is less than the lower threshold if MnKmTHS=0, or when the value is larger than the upper threshold if MnKmTHS=1.

• **TKC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	D5	TSCS	TK16S1	TK16S0	TKFS1	TKFS0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	1	1

Bit 7~5 **D7~D5**: Data bits for test only

These bits are used for test purpose only and must be kept as "000" for normal operations.

Bit 4 **TSCS**: Touch key time slot counter selection
0: Each touch key module uses its own time slot counter
1: All touch key modules use Module 0 time slot counter

Bit 3~2 **TK16S1~TK16S0**: Touch key function 16-bit counter clock source selection
00: f_{SYS}
01: $f_{SYS}/2$
10: $f_{SYS}/4$
11: $f_{SYS}/8$

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

• **TKC2 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	TSC	ASMP1	ASMP0
R/W	—	—	—	—	—	R/W	R/W	R/W
POR	—	—	—	—	—	0	0	1

Bit 7~3 Unimplemented, read as "0"

Bit 2 **TSC**: Time slot control
0: Time slot 0~3
1: Time slot 0 only
This bit is used to configure time slot, if the TSC bit is set to "1", only time slot 0 is active to execute the touch key related operations and time slot 1~3 are invalid. It can reduce power consumption and implement the 1-key wake-up function from the IDLE, SLEEP or DEEP SLEEP mode. The desired wake-up key in time slot 0 is selected by the MnSK01~MnSK00 bits in the TKMnC2 register.

Bit 1~0 **ASMP1~ASMP0**: Periodic auto scan mode period selection
00: 2^{14}/PSC0
01: 2^{13}/PSC0
10: 2^{12}/PSC0
11: 2^{11}/PSC0
These bits are used to determine the touch key scan period and only available when the touch key function is configured to operate in the periodic auto scan mode. The number of touch key scan times is obtained by the TB0 time-out period, TB0, and the periodic auto scan mode period, t_{KEY}, using the equation, $N=TB0/t_{KEY}$. The TB0 time-out period, t_{TB0}, is selected by the TB02~TB00 bits.
For example, if the f_{PSC0} selects from f_{SUB} and the TB0 time-out period is 2^{15}/PSC0 by setting the TB0C[2:0] bits to 100, then TB0 is equal to 1s. Therefore, the number of touch key scan times is 2/4/8/16 times in a TB0 time-out cycle when the ASMP[1:0] bits are set to 00/01/10/11 respectively. It is extremely important to ensure that the periodic auto scan mode period t_{KEY} does not exceed the TB0 time-out period TB0 in applications.

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

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

This register pair is used to store the touch key function 16-bit counter value. This 16-bit counter can be used to calibrate the reference or key oscillator frequency. When the touch key time slot counter overflows in the manual scan mode, this 16-bit counter will be stopped and the counter content will be unchanged. However, this 16-bit counter content will be cleared to zero at the end of the time slot 0, slot 1 and slot 2 but kept unchanged at the end of the time slot 3 in the auto scan mode or the periodic auto scan mode. This register pair will be cleared to zero when the TKST bit is cleared to zero.

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

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

This register pair is used to store the touch key module n 16-bit C/F counter value. This 16-bit C/F counter will be stopped and the counter content will be kept unchanged when the touch key time slot counter overflows in the manual scan mode. However, this 16-bit C/F counter content will be cleared to zero at the end of the time slot 0, slot 1 and slot 2 after it is written to the touch key data memory but kept unchanged at the end of the time slot 3 when the auto scan mode or the periodic auto scan mode is selected. This register pair will be cleared to zero when the TKST bit is cleared to zero.

• **TKMnROH/TKMnROL – Touch Key Module n Reference Oscillator Capacitor Selection Register Pair**

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

This register pair is used to store the touch key module n reference oscillator capacitor value. This register pair will be loaded with the corresponding next time slot capacitor value from the dedicated touch key data memory at the end of the current time slot when the auto scan mode or the periodic auto scan mode is selected.

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

• **TKMnC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	MnDFEN	—	MnSOFC	MnSOF2	MnSOF1	MnSOF0
R/W	—	—	R/W	—	R/W	R/W	R/W	R/W
POR	—	—	0	—	0	0	0	0

Bit 7~6 Unimplemented, read as "0"

Bit 5 **MnDFEN**: Touch key module n multi-frequency control
0: Disable
1: Enable
This bit is used to control the touch key oscillator frequency doubling function. When this bit is set to 1, the key oscillator frequency will be doubled.

Bit 4 Unimplemented, read as "0"

Bit 3 **MnSOFC**: Touch key module n C/F oscillator frequency hopping function control selection
0: Controlled by the MnSOF2~MnSOF0 bits
1: Controlled by hardware circuit
This bit is used to select the touch key oscillator frequency hopping function control method. When this bit is set to 1, the key oscillator frequency hopping function is controlled by the hardware circuit regardless of the MnSOF2~MnSOF0 bits value.

Bit 2~0 **MnSOF2~MnSOF0**: Touch key module n reference and key oscillators hopping frequency selection (MnSOFC=0)
000: 1.020MHz
001: 1.040MHz
010: 1.059MHz
011: 1.074MHz
100: 1.085MHz
101: 1.099MHz
110: 1.111MHz
111: 1.125MHz
These bits are used to select the touch key oscillator frequency for the hopping function. Note that these bits are only available when the MnSOFC bit is cleared to 0.
The frequencies mentioned here are only for the condition where the key and reference oscillator frequency is selected to be 1MHz, these values will be changed when the external or internal capacitor has different values. Users can adjust the key and reference oscillator frequency in scale when any other frequency is selected.

• **TKMnC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	MnTSS	—	MnROEN	MnKOEN	MnK4EN	MnK3EN	MnK2EN	MnK1EN
R/W	R/W	—	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	—	0	0	0	0	0	0

Bit 7 **MnTSS**: Touch key module n time slot counter clock source selection
0: Touch key module n reference oscillator
1: f_{LIRC}
When the MnTSS bit is set high, the touch key module n time slot counter clock comes from the f_{LIRC} oscillator, which can remain on even when these devices enter the IDLE, SLEEP or DEEP SLEEP mode. The touch key module n time slot counter clock is then can bypass the 5-bit counter to reduce the overflow time. In the periodic auto scan mode, setting the MnTSS and TSC bits high can reduce the touch key standby power.
Note: If the 5-bit counter is bypassed, the touch key module n Reference and Key oscillator hopping frequency is selected by the MnSOF2~MnSOF0 bits regardless of the MnSOFC bit setting.

Bit 6 Unimplemented, read as "0"

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

In the manual scan mode, this bit is used to enable/disable the touch key module n reference oscillator. The module n reference oscillator should first be enabled before setting the TKST bit from low to high if the module n reference oscillator is required to be used and will be disabled automatically when the TKBUSY bit is changed from high to low.

In the auto scan mode or the periodic auto scan mode, this bit is controlled by hardware automatically. For the module 0, when the TKST bit is from low to high, the M0ROEN bit will be set high automatically. For the other module n (n≠0), if the condition that MnK4EN~MnK1EN ≠ 0000B, MnTSS=0 and TSCS=0 is satisfied, the MnROEN bit will be set high automatically when the TKST bit is from low to high. In other conditions, the MnROEN bit will be unaffected by the TKST bit settings. When the TKBUSY bit is changed from high to low, the MnROEN bit will automatically be cleared to zero to disable the reference oscillator.

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

In the manual scan mode, this bit is used to enable/disable the module n key oscillator. The module n key oscillator should first be enabled before setting the TKST bit from low to high if the relevant key is enabled to be scanned and will be disabled automatically when the TKBUSY bit is changed from high to low.

In the auto scan mode or the periodic auto scan mode, this bit is controlled by hardware automatically. The MnKOEN bit will be set high automatically to enable the key oscillator when the TKST bit is from low to high. When the TKBUSY bit is changed from high to low, the MnKOEN bit will automatically be cleared to zero to disable the key oscillator.

Bit 3 **MnK4EN**: Touch key module n Key 4 enable control

MnK4EN	Touch Key Module n – Mn			
	M0	M1	M2	M3
0: Disable	I/O or other functions			
1: Enable	KEY4	KEY8	KEY12	KEY16
BC66F2235	—	—	√	—
BC66F2245	√	—	√	√
BC66F2255	√	√	√	√

Note: "—" indicates that the corresponding bit for the device is reserved and must be fixed at "0".

Bit 2 **MnK3EN**: Touch key module n Key 3 enable control

MnK3EN	Touch Key Module n – Mn			
	M0	M1	M2	M3
0: Disable	I/O or other functions			
1: Enable	KEY3	KEY7	KEY11	KEY15
BC66F2235	—	—	√	√
BC66F2245	√	—	√	√
BC66F2255	√	√	√	√

Note: "—" indicates that the corresponding bit for the device is reserved and must be fixed at "0".

Bit 1 **MnK2EN**: Touch key module n Key 2 enable control

MnK2EN	Touch Key Module n – Mn			
	M0	M1	M2	M3
0: Disable	I/O or other functions			
1: Enable	KEY2	KEY6	KEY10	KEY14
BC66F2235	—	√	—	√
BC66F2245	√	√	√	√
BC66F2255	√	√	√	√

Note: "—" indicates that the corresponding bit for the device is reserved and must be fixed at "0".

Bit 0 **MnK1EN**: Touch key module n Key 1 enable control

MnK1EN	Touch Key Module n – Mn			
	M0	M1	M2	M3
0: Disable	I/O or other functions			
1: Enable	KEY1	KEY5	KEY9	KEY13
BC66F2235	√	√	—	√
BC66F2245	√	√	√	√
BC66F2255	√	√	√	√

Note: "—" indicates that the corresponding bit for the device is reserved and must be fixed at "0".

- **TKMnC2 Register**

This register is used to select the desired scan key in the time slots 0~3 of the touch key module n. It should be noted that if any key is disabled, the touch key module n Reference and Key oscillators of the corresponding time slot will not oscillate.

Bit	7	6	5	4	3	2	1	0
Name	MnSK31	MnSK30	MnSK21	MnSK20	MnSK11	MnSK10	MnSK01	MnSK00
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	1	1	1	0	0	1	0	0

Bit 7~6 **MnSK31~MnSK30**: Touch key module n time slot 3 key scan selection
00: Key 1
01: Key 2
10: Key 3
11: Key 4
These bits are used to select the desired scan key in time slot 3 in the auto scan mode or the periodic auto scan mode. The settings for these bits are invalid when TKMOD1~TKMOD0=01 or TSC=1.

Bit 5~4 **MnSK21~MnSK20**: Touch key module n time slot 2 key scan selection
00: Key 1
01: Key 2
10: Key 3
11: Key 4
These bits are used to select the desired scan key in time slot 2 in the auto scan mode or the periodic auto scan mode. The settings for these bits are invalid when TKMOD1~TKMOD0=01 or TSC=1.

Bit 3~2 **MnSK11~MnSK10**: Touch key module n time slot 1 key scan selection
00: Key 1
01: Key 2
10: Key 3
11: Key 4

These bits are used to select the desired scan key in time slot 1 in the auto scan mode or the periodic auto scan mode. The settings for these bits are invalid when TKMOD1~TKMOD0=01 or TSC=1.

Bit 1~0 **MnSK01~MnSK00**: Touch key module n time slot 0 key scan selection
00: Key 1
01: Key 2
10: Key 3
11: Key 4

These bits are used to select the desired scan key in time slot 0 in the auto scan mode or the periodic auto scan mode or used as the multiplexer for scan key selection in the manual mode.

- **TKMnTH16H/TKMnTH16L – Touch Key Module n 16-bit Threshold Register Pair**

Register	TKMnTH16H								TKMnTH16L							
Bit	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Name	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This register pair is used to store the touch key module n 16-bit upper/lower threshold value. This register pair will be loaded with the corresponding next time slot 16-bit upper/lower threshold value from the dedicated touch key data memory automatically by the hardware at the end of the current time slot when the periodic auto scan mode is selected. After the touch key module n dedicated touch key, Key m (m=1~4), scan operation is completed, the 16-bit C/F counter content, TKMn16DH/TKMn16DL, will be compared with the TKMnTH16H/TKMnTH16L value by the hardware. When this value is less than the lower threshold if MnKmTHS=0, or larger than the upper threshold if MnKmTHS=1, then the MnKmTHF flag will be set high, and an interrupt signal will be generated.

- **TKMnTHS Register**

Bit	7	6	5	4	3	2	1	0
Name	MnK4THF	MnK3THF	MnK2THF	MnK1THF	MnK4THS	MnK3THS	MnK2THS	MnK1THS
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 **MnK4THF**: Touch key module n Key 4 upper/lower threshold comparison flag
0: Not less than lower threshold or not larger than upper threshold
1: Less than lower threshold or larger than upper threshold

Bit 6 **MnK3THF**: Touch key module n Key 3 upper/lower threshold comparison flag
0: Not less than lower threshold or not larger than upper threshold
1: Less than lower threshold or larger than upper threshold

Bit 5 **MnK2THF**: Touch key module n Key 2 upper/lower threshold comparison flag
0: Not less than lower threshold or not larger than upper threshold
1: Less than lower threshold or larger than upper threshold

Bit 4 **MnK1THF**: Touch key module n Key 1 upper/lower threshold comparison flag
0: Not less than lower threshold or not larger than upper threshold
1: Less than lower threshold or larger than upper threshold

Bit 3 **MnK4THS**: Touch key module n Key 4 upper or lower threshold comparison selection
0: Lower threshold comparison
1: Upper threshold comparison

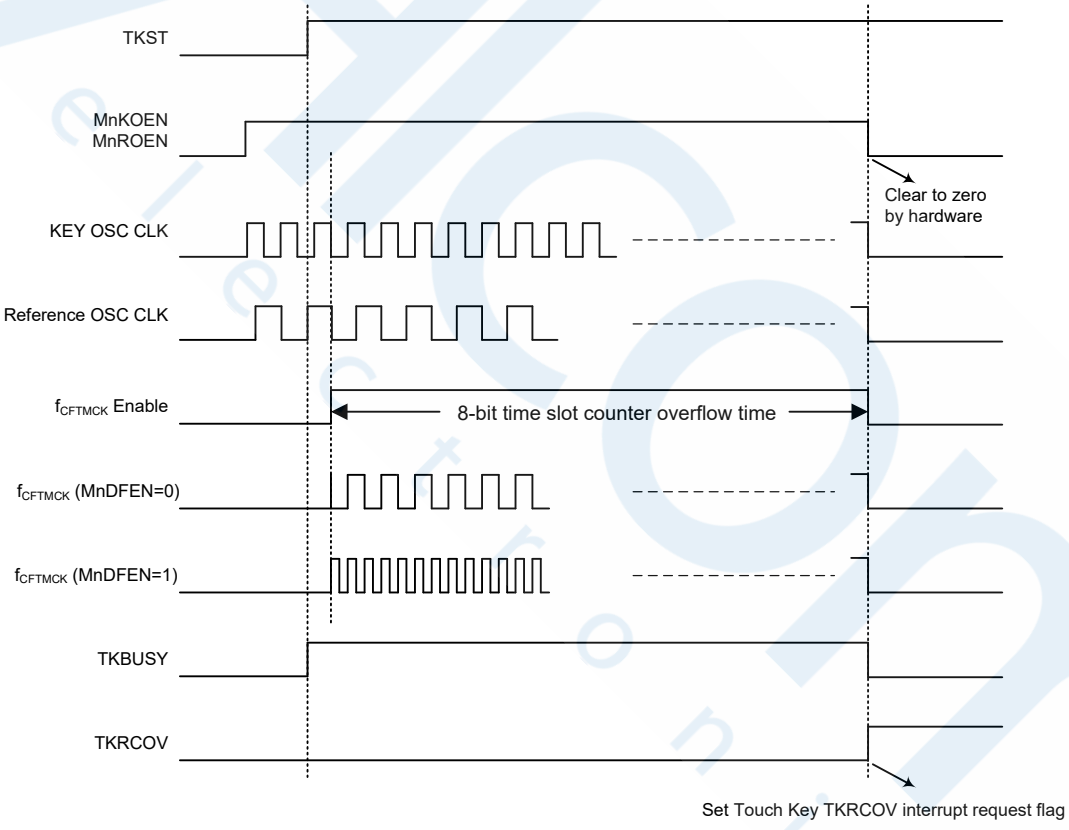
Bit 2 **MnK3THS**: Touch key module n Key 3 upper or lower threshold comparison selection
0: Lower threshold comparison
1: Upper threshold comparison

Bit 1 **MnK2THS**: Touch key module n Key 2 upper or lower threshold comparison selection
0: Lower threshold comparison
1: Upper threshold comparison

Bit 0 **MnK1THS**: Touch key module n Key 1 upper or lower threshold comparison selection
0: Lower threshold comparison
1: Upper threshold comparison

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 sensing 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 sensing oscillator during this fixed time period touch key actions can be determined.



Touch Key Manual Scan Mode Timing Diagram

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

During the reference clock fixed interval, the number of clock cycles generated by the sensing 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 TKRCOV interrupt signal will be generated in the manual scan mode.

Using the TSCS bit in the TKC1 register can select the module 0 time slot counter as the time slot counter for all modules. All modules use the same started signal, TKST, in the TKC0 register. The touch key module 16-bit C/F counter, touch key function 16-bit counter, 5-bit time slot unit period counter in all modules will be automatically cleared when the TKST bit is cleared to zero, but the 8-bit programmable time slot counter will not be cleared. The overflow time is set by users. When the TKST bit changes from low to high, the 16-bit C/F counter, touch key function 16-bit counter, 5-bit time slot unit period counter and 8-bit time slot timer counter will be automatically switched on.

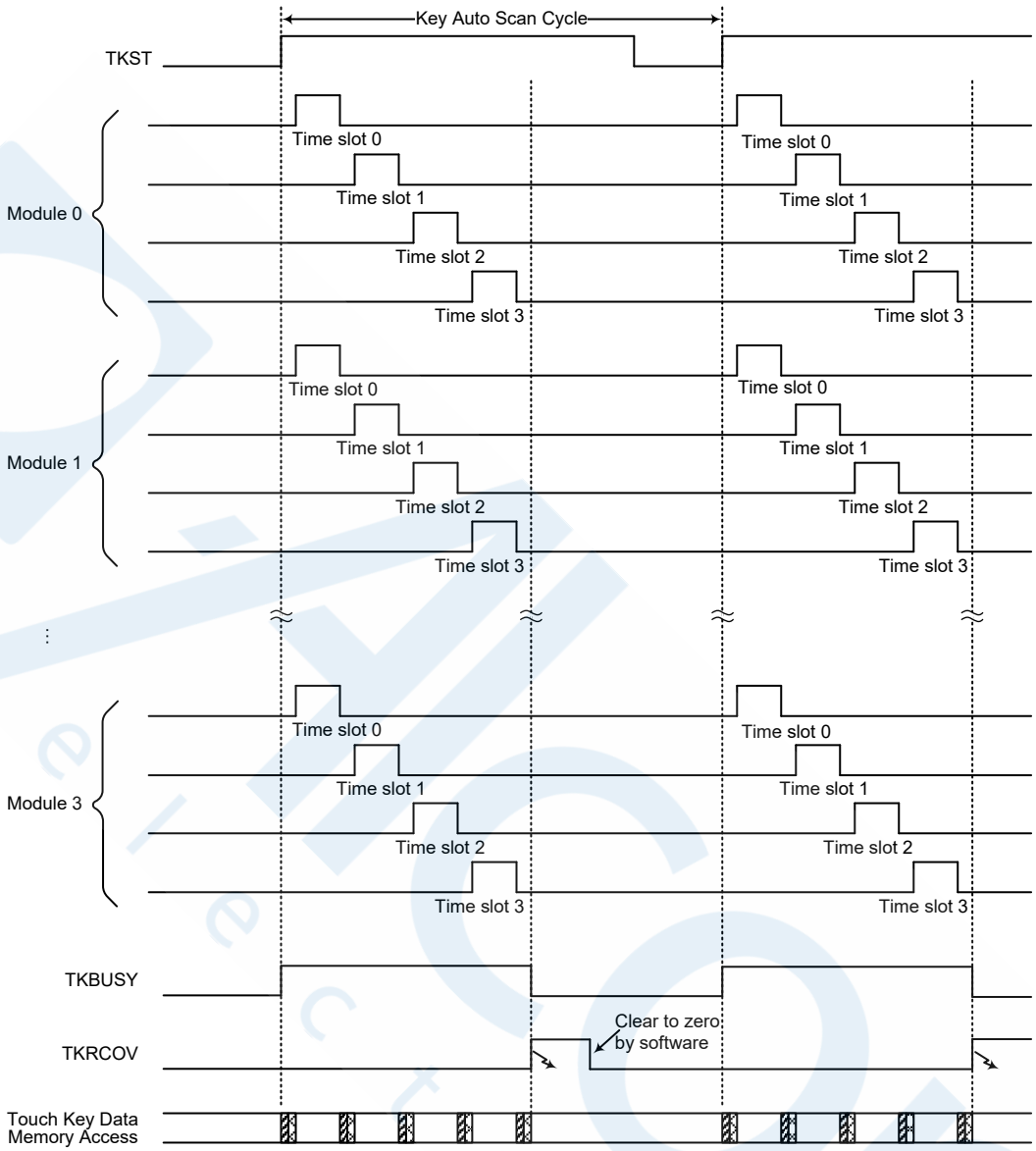
The key oscillator and reference oscillator in all modules will be automatically stopped and the 16-bit C/F counter, touch key function 16-bit counter, 5-bit time slot unit period counter and 8-bit time slot timer counter will be automatically switched off when the time slot counter overflows. The clock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

When the time slot counter in all the touch key modules or in the touch key module 0 overflows, an actual touch key TKRCOV 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 module 3. Each touch key module has an identical structure.

Auto Scan Mode

There are three scan modes contained for the touch key function. The auto scan mode, the periodic auto scan mode and the manual scan mode are selected using the TKMOD1~TKMOD0 bits in the TKC0 register. The auto scan mode can minimise the load of the application program and improve the touch key scan operation performance. When the TKMOD1~TKMOD0 bits are set to 00, the auto scan mode is selected to scan the module keys in a specific sequence determined by the MnSK3[1:0]~MnSK0[1:0] bits in the TKMnC2 register. The TSC bit in the TKC2 register is used to configure time slot.



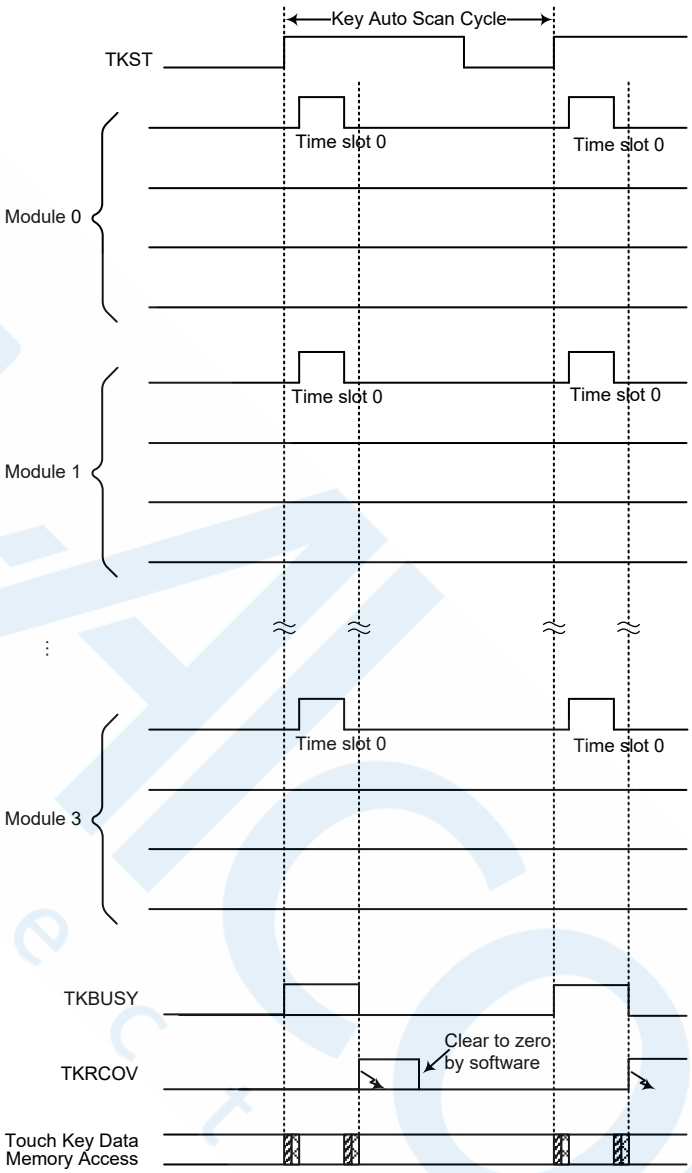
: Set Touch Key interrupt request flag

: Read 2N bytes from Touch Key Data Memory to TKMnROH/TKMnROL registers

: Write 2N bytes from TKMn16DH/TKMn16DL registers to Touch Key Data Memory

N = Touch Key Module Number; n = Module Serial Number

Touch Key Auto Scan Mode Timing Diagram – TSC=0



: Set Touch Key interrupt request flag

: Read 2N bytes from Touch Key Data Memory to TKMnROH/TKMnROL registers

: Write 2N bytes from TKMn16DH/TKMn16DL registers to Touch Key Data Memory

N = Touch Key Module Number; n = Module Serial Number

Touch Key Auto Scan Mode Timing Diagram – TSC=1

In the auto scan mode the module n key oscillator and reference oscillator which are required to be used will be enabled by hardware automatically when the TKST bit is set from low to high and disabled automatically when the TKBUSY bit changes from high to low. If the TSC bit is cleared to zero, time slot 0~3 are active. When the TKST bit is set from low to high in the auto scan mode, the internal capacitor value of the reference oscillator for the selected key to be scanned in the time slot 0 will first be read from a specific location of the dedicated touch key data memory and loaded into the corresponding TKMnROH/TKMnROL registers. Then the 16-bit C/F counter value will be written into the corresponding location of the last time slot 3 scanned key in the touch key data memory. After this, the selected key will start to be scanned in time slot 0. At the end of the time slot 0 key scan operation, the reference oscillator internal capacitor value for the next selected key will be read from the touch key data memory and loaded into the next TKMnROH/TKMnROL registers. Then the 16-bit C/F counter value of the current scanned key will be written into the corresponding touch key data memory. The whole auto scan operation will sequentially be carried out in the above specific way from time slot 0 to time slot 3. At the end of the time slot 3 key scan operation, the reference oscillator internal capacitor value for the time slot 0 selected key will again be read from the touch key data memory and loaded into the corresponding TKMnROH/TKMnROL registers. Then the 16-bit C/F counter value will be written into the relevant location of the time slot 3 scanned key in the touch key data memory. After all the selected keys are scanned, the TKRCOV bit will be set high and the TKBUSY bit will be cleared to zero as well as an auto scan mode operation is completed. If the TSC bit is set high, only time slot 0 is active to execute the touch key related operations and time slot 1~3 are invalid.

Periodic Auto Scan Mode

In addition to those actions mentioned in the auto scan mode, the periodic auto scan mode provides periodic auto scan and C/F counter upper/lower threshold comparison functions. When the TKMOD1~TKMOD0 bits are set to 10 or 11, the periodic auto scan mode is selected to scan the module keys automatically and periodically. Note that this mode is generally used in the IDLE mode, in order to monitor the touch key state and minimise power consumption.

In the periodic auto scan mode the touch key scan operation will be implemented automatically on a periodic basis, which can be determined by the ASMP1~ASMP0 bits in the TKC2 register. The number of touch key scan times depends upon the TB0 time-out period and the periodic auto scan mode period. Each auto scan operation will sequentially be carried out in a specific way from time slot 0 to time slot 3 like the auto scan mode. The reference oscillator internal capacitor value for each time slot selected key will be read from the touch key data memory and loaded into the TKMnROH/TKMnROL registers. However, only at the end of the last scan operation in the TB0 time-out cycle, the 16-bit C/F counter value for all scanned keys will be written into the corresponding touch key data memory.

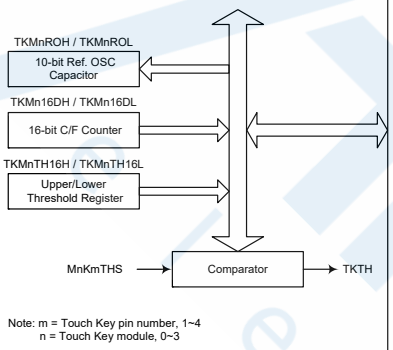
In addition, the 16 bit upper/lower threshold value for the selected key to be scanned in the time slot 0 will be read from a specific location of the dedicated touch key data memory and loaded into the corresponding TKMnTH16H/TKMnTH16L registers before the selected key will start to be scanned in time slot 0. The TKMnTH16H/TKMnTH16L register pair will be loaded with the corresponding next time slot 16-bit upper/lower threshold value from the dedicated touch key data memory automatically by the hardware at the end of the current time slot. Each touch key has its own independent upper/lower threshold comparator. The upper/lower threshold comparison function will automatically be enabled in the periodic auto scan mode. When any key C/F counter value is less than the lower threshold if MnKmTHS=0, or larger than the upper threshold if MnKmTHS=1, this indicates that the touch key state changes, then the MnKmTHF flag will be set high by the hardware, and an interrupt signal will be generated. Note that if the touch key module TKTH interrupt occurs, 1-byte data will be written to the TKMnROL register because the TKMnROH/

TKMnROL register pair will be loaded with the corresponding next time slot capacitor value from the dedicated touch key data memory and the 16-bit C/F counter content, TKMn16DH/TKMn16DL, and the TKMnTH16H/TKMnTH16L value are compared at the same time.

As the periodic auto scan operation is implemented using the TB0 counter clock to reduce power consumption, when the TB0ON is cleared the TB0 counter will stop, the periodic auto scan operation time will be affected.

Touch Key Data Memory

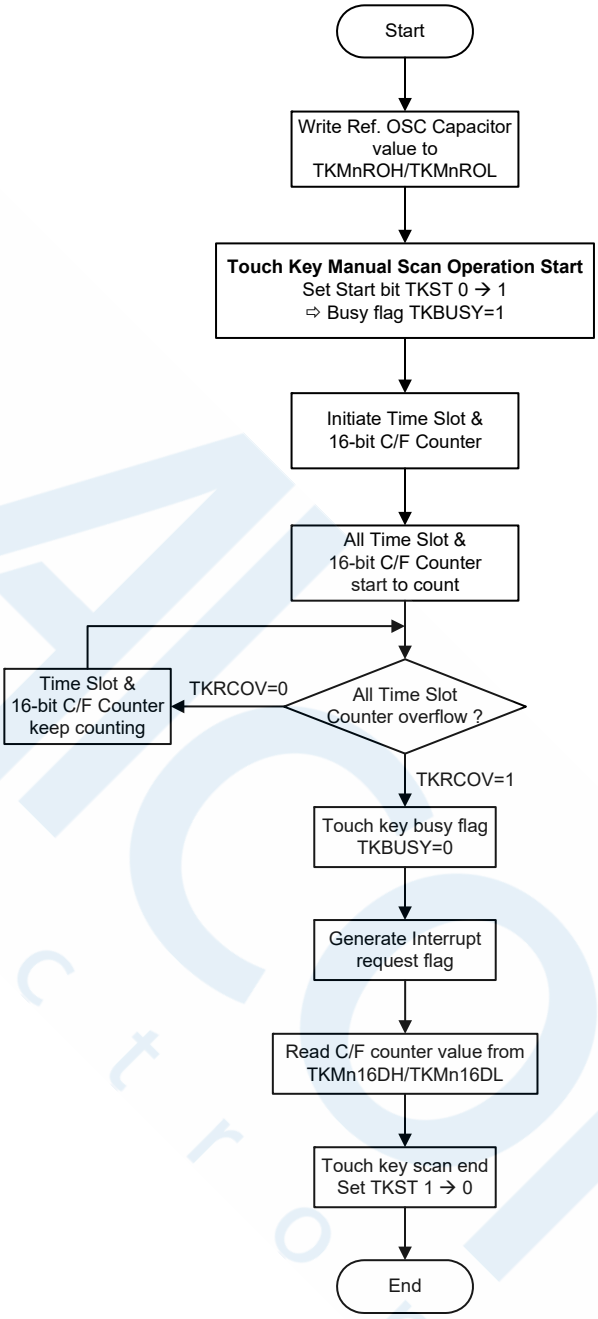
These devices provide three dedicated Data Memory area for the touch key auto scan mode or periodic auto scan mode. The first area is used to store the 16-bit C/F counter values of the touch key modules and located in Data Memory Sector 5. The second area is used to store the reference oscillator internal capacitor values of the touch key modules and located in Data Memory Sector 6. The last area is used to store the 16-bit upper/lower threshold value of the touch key modules and located in Data Memory Sector 7.



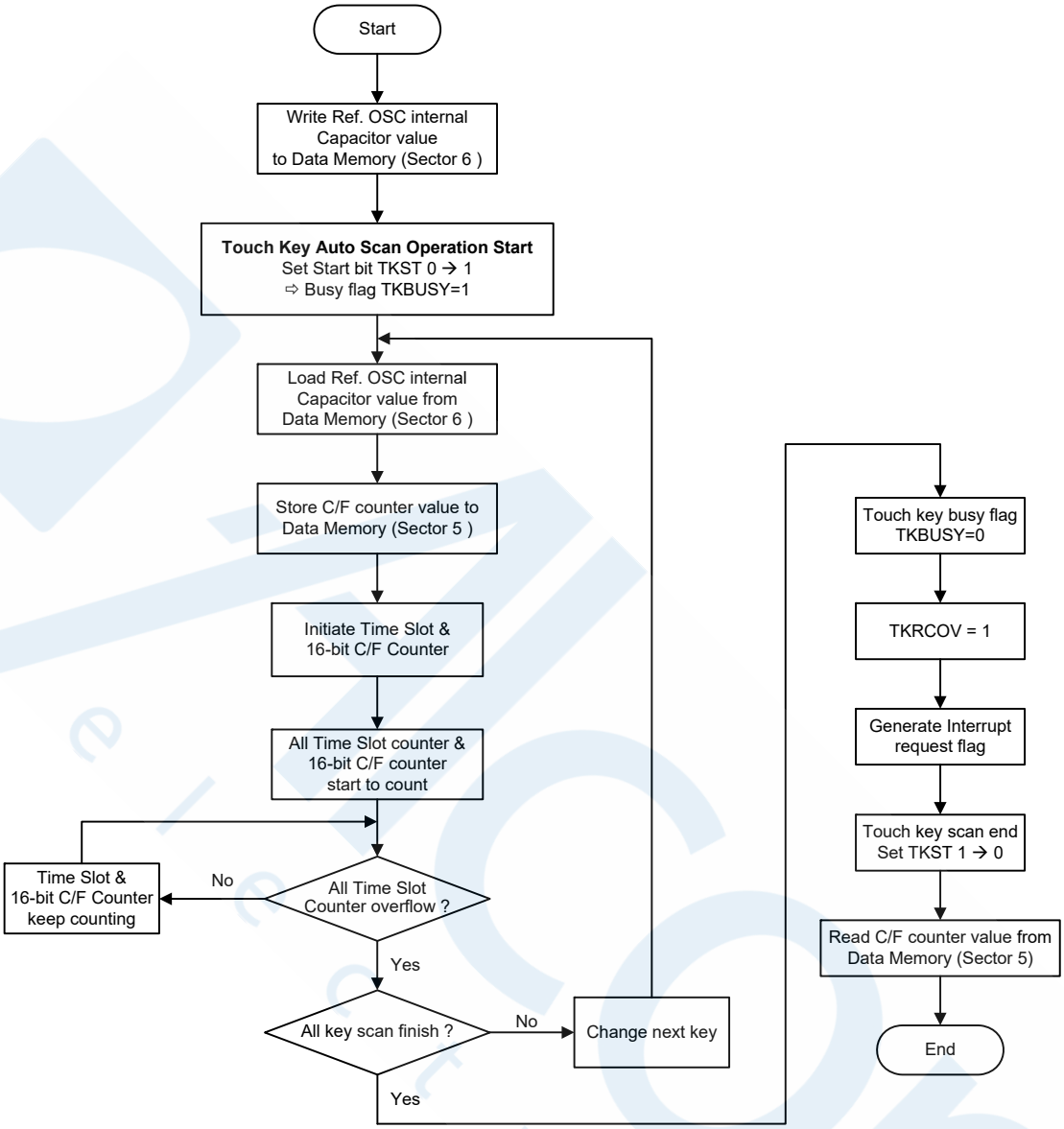
	16-bit C/F Counter Value (Sector 5)	Ref. OSC Capacitor Value (Sector 6)	Upper/Lower Threshold Value (Sector 7)
00H	TKM016DL_K1	TKM0ROL_K1	TKM0TH16L_K1
01H	TKM016DH_K1	TKM0ROH_K1	TKM0TH16H_K1
02H	TKM016DL_K2	TKM0ROL_K2	TKM0TH16L_K2
Module 0	TKM016DH_K2	TKM0ROH_K2	TKM0TH16H_K2
	TKM016DL_K3	TKM0ROL_K3	TKM0TH16L_K3
	TKM016DH_K3	TKM0ROH_K3	TKM0TH16H_K3
	TKM016DL_K4	TKM0ROL_K4	TKM0TH16L_K4
	TKM016DH_K4	TKM0ROH_K4	TKM0TH16H_K4
	TKM116DL_K1	TKM1ROL_K1	TKM1TH16L_K1
	TKM116DH_K1	TKM1ROH_K1	TKM1TH16H_K1
	TKM116DL_K2	TKM1ROL_K2	TKM1TH16L_K2
Module 1	TKM116DH_K2	TKM1ROH_K2	TKM1TH16H_K2
	TKM116DL_K3	TKM1ROL_K3	TKM1TH16L_K3
	TKM116DH_K3	TKM1ROH_K3	TKM1TH16H_K3
	TKM116DL_K4	TKM1ROL_K4	TKM1TH16L_K4
	TKM116DH_K4	TKM1ROH_K4	TKM1TH16H_K4
	⋮	⋮	⋮
	TKM316DL_K1	TKM3ROL_K1	TKM3TH16L_K1
	TKM316DH_K1	TKM3ROH_K1	TKM3TH16H_K1
	TKM316DL_K2	TKM3ROL_K2	TKM3TH16L_K2
Module 3	TKM316DH_K2	TKM3ROH_K2	TKM3TH16H_K2
	TKM316DL_K3	TKM3ROL_K3	TKM3TH16L_K3
	TKM316DH_K3	TKM3ROH_K3	TKM3TH16H_K3
	TKM316DL_K4	TKM3ROL_K4	TKM3TH16L_K4
1FH	TKM316DH_K4	TKM3ROH_K4	TKM3TH16H_K4

Touch Key Data Memory Map

Touch Key Scan Operation Flowchart



Touch Key Manual Scan Mode Flowchart – TKMOD[1:0]=01, TSCS=0



Touch Key Auto Scan Mode Flowchart – TKMOD[1:0]=00, TSCS=0

Touch Key Interrupts

The touch key has two independent interrupts, known as touch key TKRCOV interrupt and touch key module TKTH interrupt. In the manual scan mode, when the time slot counter in all the touch key modules or in the touch key module 0 overflows, an actual touch key TKRCOV interrupt will take place. In the auto scan mode, when the touch key auto scan operation is completed, the Touch Key TKRCOV Interrupt request flag, TKRCOVF, will be set. In the periodic auto scan mode, only after the last scan operation in the TB0 time-out cycle completes the 16-bit C/F counter content is written into the corresponding touch key data memory, then the Touch Key TKRCOV Interrupt request flag, TKRCOVF, will be set. The touch keys mentioned here are the keys which are enabled. The 16-bit C/F counter, 16-bit counter, 5-bit time slot unit period counter and 8-bit time slot counter in the module will be automatically cleared. When any key C/F counter value is less than the lower threshold if MnKmTHS=0, or larger than the upper threshold if MnKmTHS=1, a touch key module TKTH interrupt will take place. More details regarding the touch key interrupt is located in the interrupt section of the datasheet.

Programming Considerations

After the relevant registers are setup, the touch key detection process is initiated when changing the TKST bit from low to high. This will enable and synchronise all relevant oscillators. The TKRCOV flag which is the time slot counter flag will go high when the counter overflows in the manual scan mode. When this happens an interrupt signal will be generated. In the auto scan mode, if the time slot counter overflows but the touch key auto scan operation is not completed, the TKRCOV bit will not be set. When the touch key auto scan operation is completed, the TKRCOV bit and the Touch Key TKRCOV Interrupt request flag, TKRCOVF, will be set. In the periodic auto scan mode, the TKRCOV bit is cleared to zero during the auto scan operation period. Only at the end of the last scan operation in the TB0 time-out cycle, the 16-bit C/F counter content will be written into the corresponding touch key data memory, and then the TKRCOV bit will be set high by the hardware circuit. The TKTH signal which is the threshold comparison indication signal will go high when a certain threshold comparison condition occurs. When this happens an interrupt signal will also be generated. As the TKRCOV flag will not be automatically cleared, it has to be cleared by the application program.

The TKCFOV flag which is the 16-bit C/F counter overflow flag will go high when any of the Touch Key Module 16-bit C/F counter overflows. As this flag will not be automatically cleared, it has to be cleared by the application program. The TK16OV flag which is the 16-bit counter overflow flag will go high when the 16-bit counter overflows. As this flag will not be automatically cleared, it has to be cleared by the application program.

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

RF Transmitter

The RF transmitter is a fully integrated transmitter, which is capable of using both Frequency-Shift Keying (FSK) and On-Off Keying (OOK) modulation modes for data streaming. It has two main operating modes, Burst Mode and Direct Mode. The RF transmitter operates in the 315/433/868/915MHz frequency bands.

RF Transmitter Abbreviation Notes

CP: Charge Pump

DFC: Digital Frequency Centering

FIFO: TX FIFO

MMD: Multi-Mode Divider

PAD: Power Amplifier Driver

SX: Synthesizer

TXD: Data from FIFO or DTXD bit, determined by the DIR_EN bit

VCO: Voltage Control Oscillator

XO: External Crystal Output

XCLK: RF circuit main clock. Controlled by configuring the XCLKD2 bit and enabled by setting the XCLK_EN bit high.

XCLK_MCU: MCU system clock. Controlled by configuring the XCLKD2 bit and enabled by setting the HXTEN bit in the HXTC register high.

RF Transmitter Control Registers

The RF transmitter is controlled by a series of registers. These registers control the overall RF function including power down control, operation mode selection, clock division selection, FIFO data configuration, modulator control, fractional-N synthesizer control, charge pump (CP) control, multi-mode divider (MMD) control, voltage control oscillator (VCO) control, TX power fine tune, VCO DFC calibration, RF LDO control and RF external crystal output, etc.

Register Name	Bit							
	7	6	5	4	3	2	1	0
RF_PWR	—	—	—	—	—	—	—	RF_PDB
RF_OPER	—	—	FSK_EN	DIR_EN	—	—	—	TX_STROBE
RF_CLK1	XCLK_EN	XCLKINV	XCLKR_RDY	XCLKD2	—	—	—	RST_RF
RF_CLK2	DTR7	DTR6	DTR5	DTR4	DTR3	DTR2	DTR1	DTR0
RF_FIFO_CTRL1	FFDATA7	FFDATA6	FFDATA5	FFDATA4	FFDATA3	FFDATA2	FFDATA1	FFDATA0
RF_FIFO_CTRL2	FFLEN7	FFLEN6	FFLEN5	FFLEN4	FFLEN3	FFLEN2	FFLEN1	FFLEN0
RF_FIFO_CTRL3	RST_TX_FF	FFSA6	FFSA5	FFSA4	FFSA3	FFSA2	FFSA1	FFSA0
RF_FIFO_CTRL4	DTXD	—	—	—	TXFFLT	FFMG_EN	FFMG1	FFMG0
RF_MOD1	FDEV7	FDEV6	FDEV5	FDEV4	FDEV3	FDEV2	FDEV1	FDEV0
RF_MOD2	—	—	—	—	—	FDEV10	FDEV9	FDEV8
RF_MOD4	D7	D6	D5	OOKDT_TS2	OOKDT_TS1	OOKDT_TS0	OOKDT_POR	OOKDT_EN
RF_OPMOD	—	—	—	—	—	ACAL_EN	TX_EN	SX_EN
RF_SX1	—	DN6	DN5	DN4	DN3	DN2	DN1	DN0
RF_SX2	DK7	DK6	DK5	DK4	DK3	DK2	DK1	DK0
RF_SX3	DK15	DK14	DK13	DK12	DK11	DK10	DK9	DK8
RF_SX4	—	—	—	—	DK19	DK18	DK17	DK16
RF_CP3	DLY_SYN2	DLY_SYN1	DLY_SYN0	D4	D3	D2	D1	D0
RF_OD1	D7	D6	D5	D4	D3	D2	D1	D0
RF_VCO1	VCO_SWHB	D6	D5	D4	D3	D2	D1	D0
RF_TX1	DLY_PAD2	DLY_PAD1	DLY_PAD0	D4	CT_PARD2	CT_PARD1	CT_PARD0	PAD_EN

Register Name	Bit							
	7	6	5	4	3	2	1	0
RF_TX2	D7	D6	D5	D4	D3	D2	D1	D0
RF_DFC_CAL	CT_MMDLDO1	CT_MMDLDO0	—	D4	D3	D2	D1	D0
RF_LDO	D7	D6	D5	D4	D3	D2	D1	D0
RF_XO1	XSHIFT1	XSHIFT0	—	XO_TRIM4	XO_TRIM3	XO_TRIM2	XO_TRIM1	XO_TRIM0

RF Transmitter Control Register List

Most of the RF transmitter control register details will be described in this section, however several registers will be described in their respective other sections.

- **RF_PWR Register – RF Power Down Control Register**

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

Bit 7~1 Unimplemented, read as "0"

Bit 0 **RF_PDB**: RF power down control
0: RF Power down mode
1: RF Active mode
This bit determines whether the RF circuit is in the active or power down mode. This bit will be cleared to 0 when a power on reset, an LVR reset or a WDT time-out during normal operation occurs or when the MCU enters the DEEP SLEEP Mode by setting the PWDN bit in the PWRC register to 1 and executing a HALT instruction.

- **RF_CLK1 Register – RF Clock Control Register 1**

Bit	7	6	5	4	3	2	1	0
Name	XCLK_EN	XCLKINV	XCLKR_RDY	XCLKD2	—	—	—	RST_RF
R/W	R/W	R/W	R	R/W	—	—	—	R/W
POR	1	0	0	0	—	—	—	0

Bit 7 **XCLK_EN**: Clock auto gating for the RF internal digital circuit
0: Disable
1: Enable
Writing data to the FIFO requires enabling XCLK. The XCLK_EN bit should be cleared before the RF circuit enters the power down mode with the RF_PDB bit cleared, this will prevent XCLK clock glitch from occurring and guarantee that the XCLK domain register will not affected by the clock lost.

Bit 6 **XCLKINV**: XCLK clock invert control
0: XCLK clock does not invert
1: XCLK clock inverts

Bit 5 **XCLKR_RDY**: XCLK ready flag
0: XCLK is not ready
1: XCLK is ready
When HXTEN=0 and RF_PDB=1, indicating that the MCU operates with the f_{HIRC} clock, read this bit. It is read only and used to indicate whether the XCLK debounce is completed and ready to operate.

Bit 4 **XCLKD2**: XCLK clock divided by 2 control
0: XCLK clock is not divided by 2
1: XCLK clock is divided by 2
It is recommended to clear this bit to zero.

Bit 3~1 Unimplemented, read as "0"

Bit 0 **RST_RF**: Reset RF control register
0: Self-clear RF control register is complete
1: Reset RF control register
When this bit is set to 1, all RF associated registers except the RF_PWR register will be reset. It needs one instruction cycle to complete the self-clear enable action. Do not set the RST_RF bit high in two continuous instructions.

• **RF_CLK2 Register – RF Clock Control Register 2**

Bit	7	6	5	4	3	2	1	0
Name	DTR7	DTR6	DTR5	DTR4	DTR3	DTR2	DTR1	DTR0
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 **DTR7~DTR0**: RF data rate setting
RF data rate=100kHz/(DTR[7:0]+1)

• **RF_CP3 Register – RF CP Control Register 3**

Bit	7	6	5	4	3	2	1	0
Name	DLY_SYN2	DLY_SYN1	DLY_SYN0	D4	D3	D2	D1	D0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	1	1	0	0	1	0	1	0

Bit 7~5 **DLY_SYN2~DLY_SYN0**: Synthesiser delay get ready time
000: 16μs
001: 20μs
010: 24μs
011: 28μs
100: 32μs
101: 36μs
110: 40μs
111: 100μs

Bit 4~0 **D4~D0**: Reserved bits, fill with 11100b

• **RF_OD1 Register – RF MMD and OD Control Register 1**

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	1	0	0

Bit 7~0 **D7~D0**: Multi-mode divider and output division
This register is used to control the multi-mode divider and output division in the RF circuitry. Note that different setting values should be written into this register for different RF frequency band applications. The recommended setting values are summarised in the following table.

RF Frequency Band	315MHz	433MHz	868/915MHz
RF_OD1 Setting Values	A8H	A4H	00H

• **RF_VCO1 Register – RF VCO Control Register 1**

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

Bit 7 **VCO_SWHB**: VCO 2.5GHz frequency band switch control
0: Other frequency bands
1: 315MHz Band
This bit is used to select the VCO 315MHz frequency band. If this bit is set to 1, the VCO 315MHz is selected. Other VCO frequency bands except 315MHz will be selected when this bit is cleared to 0.

Bit 6~0 **D6~D0**: Frequency Band control

Note that different setting values should be written into this register for different RF frequency band applications. The recommended setting values are summarised in the following table.

RF Frequency Band	315MHz	433MHz	868/915MHz
RF_VCO1 Setting Values	A8H	38H	38H

• **RF_TX1 Register – RF TX Control Register 1**

Bit	7	6	5	4	3	2	1	0
Name	DLY_PAD2	DLY_PAD1	DLY_PAD0	D4	CT_PARD2	CT_PARD1	CT_PARD0	PAD_EN
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	1	0	0	0	1	0	0	0

Bit 7~5 **DLY_PAD2~DLY_PAD0**: TX PA driver stable time (Used in FSK mode only)
000: 16μs
001: 20μs
010: 24μs
011: 28μs
100: 32μs
101: 36μs
110: 40μs
111: 50μs

Bit 4 **D4**: Reserved bit

Bit 3~1 **CT_PARD2~CT_PARD0**: PA ramp up/down trim
000: 5μs
001: 9μs
010: 13μs
011: 17μs
100: 21μs
101: 25μs
110: 29μs
111: 33μs

Bit 0 **PAD_EN**: TX PA driver enable control
0: Disable
1: Enable

The recommended setting values are summarised in the following table.

RF Modulation Mode	OOK	FSK
RF_TX1 Setting Values	86H	84H

• **RF_TX2 Register – RF TX Control Register 2**

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

Bit 7~0 **D7~D0**: Reserved bits

The recommended setting values are summarised in the following table.

RF Band		315MHz	433MHz	868/915MHz
RF_TX2 Setting Values	0dBm	3AH	3AH	3AH
	10dBm	8AH	8AH	8AH
	13dBm	B2H	AAH	BAH

• **RF_DFC_CAL Register – RF VCO DFC Calibration Control Register**

Bit	7	6	5	4	3	2	1	0
Name	CT_MMDLDO1	CT_MMDLDO0	—	D4	D3	D2	D1	D0
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 7~6 **CT_MMDLDO1~CT_MMDLDO0**: MMD LDO voltage setting
00: 1.35V
01: 1.5V (Recommended value)
10: 1.65V
11: 1.8V

Bit 5 Unimplemented, read as "0"

Bit 4~0 **D4~D0**: Reserved bits, fill with 10000b

• **RF_LDO Register – RF LDO Control 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	1	1	0	0	0

Bit 7~0 **D7~D0**: RF synthesizer and VCO LDO function control

This register is used to control the overall LDO functions for the RF synthesizer and VCO circuitry. Note that different setting values should be written into this register for different RF frequency band applications. The recommended setting values are summarised in the following table.

RF Frequency Band	315MHz	433MHz	868/915MHz
RF_LDO Setting Values	64H	74H	74H

• **RF_XO1 Register – RF XO Control Register 1**

Bit	7	6	5	4	3	2	1	0
Name	XSHIFT1	XSHIFT0	—	XO_TRIM4	XO_TRIM3	XO_TRIM2	XO_TRIM1	XO_TRIM0
R/W	R/W	R/W	—	R/W	R/W	R/W	R/W	R/W
POR	1	0	—	1	0	1	0	1

Bit 7~6 **XSHIFT1~XSHIFT0**: Internal capacitor load coarse shift for the crystal oscillator
Capacitor step=4.5pF, XSHIFT[1:0]=0~3
Total Capacitor load≈7+XSHIFT[1:0]×4.5 + XO_TRIM[4:0]×0.2, unit: pF

Bit 5 Unimplemented, read as "0"

Bit 4~0 **XO_TRIM4~XO_TRIM0**: Internal capacitor load trim value for the crystal oscillator
Capacitor step=0.2pF, XO_TRIM[4:0]=0~31
Total Capacitor load≈7+XSHIFT[1:0]×4.5+XO_TRIM[4:0]×0.2, unit: pF

Modulation Modes and Operating Modes Selection

Modulation Modes

There are two RF modulation modes for these devices, FSK mode and OOK mode, which are selected by the FSK_EN bit.

In the OOK modulation mode, the RFOUT pin outputs the RF carry signal with a frequency f_C that is selected by channel code DN[6:0] and DK[19:0]. The RF carry signal on/off is controlled by transmitting data at the required data rate.

In the FSK modulation mode, the RFOUT pin outputs the RF signal with (f_C+f_{DEV}) for data "1" and (f_C-f_{DEV}) for data "0". The RF carry signal on/off is controlled by transmitting data at the determined data rate.

Operating Modes

There are two RF operating modes for these devices, Burst Mode and Direct Mode, which are selected by the DIR_EN bit.

In the Burst Mode, the data to be transmitted is from the FIFO and the RF block is controlled by a state machine. Users just need to write data to the FIFO and set the TX_STROBE bit high to start the transmission and wait for its completion. This is an easy mode to use.

In the Direct Mode, the data to be transmitted is configured by the DTXD bit in the RF_FIFO_CTRL4 register. The on/off timing for the RF functional block is controlled by specific program sequences as described below. This mode provides the maximum flexibility but requires users to take more attention to the block control and data to be transmitted.

- **RF_OPER Register – RF Operation Control Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	FSK_EN	DIR_EN	—	—	—	TX_STROBE
R/W	—	—	R/W	R/W	—	—	—	R/W
POR	—	—	0	0	—	—	—	0

Bit 7~6 Unimplemented, read as "0"

Bit 5 **FSK_EN**: RF output modulation mode selection
0: OOK Mode, PAD enable is controlled by TXD directly
1: FSK Mode, PAD enable is forced to 1 and TXD modulates the TX frequency

Bit 4 **DIR_EN**: RF output operating mode selection
0: Burst Mode, TX data is read from FIFO, RF is controlled by state machine (if FIFO transmission is complete, RF TX will enter the standby mode)
1: Direct Mode, TX data comes from DTXD bit in the RF_FIFO_CTRL4 register directly and RF state is controlled by SX_EN (first enable) and TX_EN (second enable)

Bit 3~1 Unimplemented, read as "0"

Bit 0 **TX_STROBE**: TX strobe to start the burst mode data transfer from RF
0: TX packet has been sent completely
1: Initiate a TX transmission
Setting this bit high will initiate the TX transmission procedure automatically, in which condition, all register settings are not allowed to change. This bit will be cleared to 0 automatically when a TX packet has been sent completely, in which condition, a BMTCF interrupt request will be generated. Users should start the next transmission after this bit returns to 0. Users can force the TX transmission to teminate by writting a 0 value to this bit.

• **RF_MOD1 Register – RF Modulator Control Register 1**

Bit	7	6	5	4	3	2	1	0
Name	FDEV7	FDEV6	FDEV5	FDEV4	FDEV3	FDEV2	FDEV1	FDEV0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	1	0	0	1	0	0	0	0

Bit 7~0 **FDEV7~FDEV0**: Set the frequency deviation in FSK mode
These bits together with the FDEV10~FDEV9 bits in the RF_MOD2 register set the frequency deviation in FSK mode.

• **RF_MOD2 Register – RF Modulator Control Register 2**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	FDEV10	FDEV9	FDEV8
R/W	—	—	—	—	—	R/W	R/W	R/W
POR	—	—	—	—	—	0	0	1

Bit 7~3 Unimplemented, read as "0"

Bit 2~0 **FDEV10~FDEV8**: Set the frequency deviation in FSK mode
These bits together with the FDEV7~FDEV0 bits in the RF_MOD1 register set the frequency deviation in FSK mode.
FDEV[10:0]=DEC2HEX $((2^{17} - 1)/f_{XTAL})\times f_D$)
For example, if f_{XTAL}=16MHz, f_D=50kHz, then FDEV[10:0]=199H.
Note: When using in the 868/915MHz frequency bands, if the frequency deviation is 150kHz, it should be paid attention to the safety specification.

• **RF_MOD4 Register – RF Modulator Control Register 4**

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	D5	OOKDT_TS2	OOKDT_TS1	OOKDT_TS0	OOKDT_POR	OOKDT_EN
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	1	1	1	0

Bit 7~5 **D7~D5**: Reserved bits

Bit 4~2 **OOKDT_TS2~OOKDT_TS0**: OOK duty cycle tune time selection
000: 1μs
001: 3μs
010: 5μs
011: 10μs
100: 15μs
101: 20μs
110: 25μs
111: 30μs

Bit 1 **OOKDT_POR**: OOK duty cycle tune polarity
0: Extend 0 duty cycle
1: Extend 1 duty cycle

Bit 0 **OOKDT_EN**: OOK duty cycle tune enable control
0: Disable
1: Enable

The recommended setting values are summarised in the following table.

RF Modulation Mode	OOK	FSK
RF_MOD4 Setting Values	0DH	15H

• **RF_OPMOD Register – RF Operation Mode Control Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	ACAL_EN	TX_EN	SX_EN
R/W	—	—	—	—	—	R/W	R/W	R/W
POR	—	—	—	—	—	0	0	0

Bit 7~3 Unimplemented, read as "0"

Bit 2 **ACAL_EN**: Auto calibration enable control
0: Disable
1: Enable
This bit can be set or reset by the application program to activate or terminate the auto calibration function. It will automatically be cleared to 0 by hardware when the auto calibration is completed.

Bit 1 **TX_EN**: Transmitter control (for Direct Mode only)
0: Disable
1: Enable
This bit is only used in the Direct Mode, it is not used in the Burst Mode.

Bit 0 **SX_EN**: Synthesizer (PLL On) control (for Direct Mode only)
0: Disable
1: Enable
This bit is only used in the Direct Mode, it is not used in the Burst Mode.

TX FIFO Mode in Burst Mode

In the Burst mode, the data to be transmitted comes from the FIFO into which data is pre-written by the MCU. There are three FIFO modes to support various applications. These are Simple FIFO mode, Block FIFO mode and Extend FIFO mode. To use the FIFO in the burst mode, set the RST_TX_FF bit in the RF_FIFO_CTRL3 register to 1 to reset the FIFO pointer and buffer. After this, the FIFO is in the initial state same as power on reset.

Simple FIFO Mode

This FIFO mode is used in general applications. The data length should not exceed 128 bytes. To use the simple FIFO mode, the MCU must write the data to be transmitted to the FIFO by accessing the RF_FIFO_CTRL1 register (FFDATA[7:0]). The transmit sequence to the transmitter is first written byte first out and is MSB first out in each byte. Users should first determine the transmit data packet format such as the preamble, ID code and packet encoding such as the FEC (Forward Error Correction), CRC (Cyclic Redundancy Check), whitening and etc. After the FIFO has been filled completely, clear the FFSA[6:0] bits in the RF_FIFO_CTRL3 register to 0 and configure the FFLEN[7:0] bits in the RF_FIFO_CTRL2 register to the desired transmission length in bytes. Then configure the FSK_EN, DIR_EN and TX_STROBE bits to start the transmission. After the current transmission is complete, the data will be kept in the FIFO to wait for next transmission.

Block FIFO Mode

The Block FIFO mode is used to support multi-key code applications. Users should write all the key code to the FIFO beforehand. When a key is pressed, the MCU will detect the key, set the FFSA[6:0] bits in the RF_FIFO_CTRL3 register as the start address of the target key code, and then set the FFLEN[7:0] bits in the RF_FIFO_CTRL2 register to indicate the key code length and finally set the TX_STROBE bit in the RF_OPER register to start the transmission. The maximum FIFO length is also limited to 128 bytes.

Extend FIFO Mode

The Extend FIFO mode is used for large data packet length up to 256 bytes. The physical FIFO length is 128 bytes. To extend the available transmit length in one packet, a handshake mechanism is needed between the MCU and FIFO.

Set the FFMG[1:0] bits in the RF_FIFO_CTRL4 register to determine the FIFO data length margin and set the FFMG_EN bit to enable the margin detect function to remind the MCU. The MCU should write data to FIFO as soon as possible when receiving this remind signal, to avoid the transmission being terminated by FIFO data length low to zero.

- **RF_FIFO_CTRL1 Register – RF FIFO Control Register 1**

Bit	7	6	5	4	3	2	1	0
Name	FFDATA7	FFDATA6	FFDATA5	FFDATA4	FFDATA3	FFDATA2	FFDATA1	FFDATA0
R/W	W	W	W	W	W	W	W	W
POR	0	0	0	0	0	0	0	0

Bit 7~0 **FFDATA7~FFDATA0**: FIFO data port
Writing data to this port will write data to the FIFO. Reading data from this port will read data from the top of the FIFO.

- **RF_FIFO_CTRL2 Register – RF FIFO Control Register 2**

Bit	7	6	5	4	3	2	1	0
Name	FFLEN7	FFLEN6	FFLEN5	FFLEN4	FFLEN3	FFLEN2	FFLEN1	FFLEN0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	1	1	1	1	1	1

Bit 7~0 **FFLEN7~FFLEN0**: TX data length (used for Burst mode only)
The number of data bytes to be transmitted is FFLEN[7:0]+1. In the Simple FIFO mode and Block FIFO mode, the FFLEN[7:0] is limited to 7Fh. Users should not set this register to be greater than this value.

- **RF_FIFO_CTRL3 Register – RF FIFO Control Register 3**

Bit	7	6	5	4	3	2	1	0
Name	RST_TX_FF	FFSA6	FFSA5	FFSA4	FFSA3	FFSA2	FFSA1	FFSA0
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 **RST_TX_FF**: TX FIFO reset control
0: Do not reset TX FIFO to its initial state or the reset action is completed
1: Reset TX FIFO to initial state
Setting this bit high will reset the TX FIFO to its initial state. After the reset action is complete, this bit will be automatically cleared.

Bit 6~0 **FFSA6~FFSA0**: FIFO start address to send out data - used for the Block FIFO mode

- **RF_FIFO_CTRL4 Register – RF FIFO Control Register 4**

Bit	7	6	5	4	3	2	1	0
Name	DTXD	—	—	—	TXFFLT	FFMG_EN	FFMG1	FFMG0
R/W	R/W	—	—	—	R	R/W	R/W	R/W
POR	0	—	—	—	0	0	0	1

Bit 7 **DTXD**: Direct mode TX data setting

Bit 6~4 Unimplemented, read as “0”

Bit 3 **TXFFLT**: TX FIFO length status flag (read only)
0: TX FIFO length is greater than the threshold length specified by FFMG[1:0]
1: TX FIFO length is less than or equal to threshold length specified by FFMG[1:0]
If the FFMG_EN bit equals to 0, the TXFFLT bit always keeps at 0. If the FFMG_EN bit equals to 1, the TXFFLT bit indicates the TX FIFO length status. When this bit is set high by hardware, it indicates that the TX FIFO length is less than or equal to the threshold length specified by FFMG[1:0], in which case, an FFMGF interrupt request will be generated.

Bit 2 **FFMG_EN**: TX FIFO length margin detect enable control
0: Disable
1: Enable

Bit 1~0 **FFMG1~FFMG0**: TX FIFO length margin - indicates the remained number of data bytes in the FIFO
00: 4 bytes
01: 8 bytes
10: 16 bytes
11: 32 bytes

RF Channel Setup

The RF channel setup is implemented using four registers, the RF_SX1~RF_SX4 registers.

• RF_SX1 Register – RF Fractional-N Synthesizer Control Register 1

Bit	7	6	5	4	3	2	1	0
Name	—	DN6	DN5	DN4	DN3	DN2	DN1	DN0
R/W	—	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	0	1	1	0	1	1	0

Bit 7 Unimplemented, read as "0"

Bit 6~0 **DN6~DN0**: Integer of dividend for MMD, supports 32 to 127
Since the MMD data is generated from a delta-sigma modulator, the DN support range will be affected. The real range will be (32+3)~(127-4), i.e. 35~123.
Set initial value to XO=16MHz and TX band=433.92MHz:
For example, if Crystal=XO=16MHz and RF_OD1=04H;
VCO frequency=TX frequency×4=433.92MHz×4=1735.68MHz;
$XO\times(DN+DK/2^{20})=16MHz\times(DN+DK/2^{20})$=VCO frequency/2=1735.68MHz/2=867.84MHz;
$(DN+DK/2^{20})=54.24$, so DN[6:0]=36H=54, DK[19:0]=3D70AH=251658.

• RF_SX2 Register – RF Fractional-N Synthesizer Control Register 2

Bit	7	6	5	4	3	2	1	0
Name	DK7	DK6	DK5	DK4	DK3	DK2	DK1	DK0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	1	0	1	0

Bit 7~0 **DK7~DK0**: Low byte of 20-bit fractional of dividend for MMD
Set an initial value to implement XO=16MHz and TX band=433.92MHz.

• RF_SX3 Register – RF Fractional-N Synthesizer Control Register 3

Bit	7	6	5	4	3	2	1	0
Name	DK15	DK14	DK13	DK12	DK11	DK10	DK9	DK8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	1	1	0	1	0	1	1	1

Bit 7~0 **DK15~DK8**: Middle byte of 20-bit fractional of dividend for MMD
Set an initial value to implement XO=16MHz and TX band=433.92MHz.

• **RF_SX4 Register – RF Fractional-N Synthesizer Control Register 4**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	DK19	DK18	DK17	DK16
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	1	1

Bit 7~4 Unimplemented, read as "0"

Bit 3~0 **DK19~DK16**: High byte of 20-bit fractional of dividend for MMD
Set an initial value to implement XO=16MHz and TX band=433.92MHz.

RF Channel Setup Description (VCO operating frequency: 1.7~1.9GHz, 2.5GHz)

The RF_OD1 register must be configured according to the following two rules:

1. Whether the VCO operating frequency is in the specified range.

2. The calculated DN and DK values are within the available range of RF_SX1~RF_SX4 registers.

The following are five frequency configuration examples (if Crystal=XO=16MHz):

- 315MHz
  RF_OD1=08H
  VCO frequency=TX Frequency×8=315MHz×8=2520MHz
  XO×(DN+DK/2^{20})=16MHz×(DN+DK/2^{20})=VCO frequency/2=2520MHz/2=1260MHz
  (DN+DK/2^{20})=78.75 → DN[6:0]=4EH=78, DK[19:0]=C0000H=786432
  Note: The VCO_SWHB bit should be set to '1' by the user program for switching the VCO to the 2.5GHz band when the TX frequency is 315MHz.
- 433MHz
  RF_OD1=04H
  VCO frequency=TX Frequency×4=433MHz×4=1732MHz
  XO×(DN+DK/2^{20})=16MHz×(DN+DK/2^{20})=VCO frequency/2=1732MHz/2=866MHz
  (DN+DK/2^{20})=54.125 → DN[6:0]=36H=54, DK[19:0]=20000H=131072
- 433.92MHz (default setting)
  RF_OD1=04H
  VCO frequency=TX Frequency×4=433.92MHz×4=1735.68MHz
  XO×(DN+DK/2^{20})=16MHz×(DN+DK/2^{20})=VCO frequency/2=1735.68MHz/2=867.84MHz
  (DN+DK/2^{20})=54.24 → DN[6:0]=36H=54, DK[19:0]=3D70AH=251658
- 868MHz
  RF_OD1=00H
  VCO frequency=TX Frequency×2=868MHz×2=1736MHz
  XO×(DN+DK/2^{20})=16MHz×(DN+DK/2^{20})=VCO frequency/2=1736MHz/2=868MHz
  (DN+DK/2^{20})=54.25 → DN[6:0]=36H=54, DK[19:0]=40000H=262144
- 915MHz
  RF_OD1=00H
  VCO frequency=TX Frequency×2=915MHz×2=1830MHz
  XO×(DN+DK/2^{20})=16MHz×(DN+DK/2^{20})=VCO frequency/2=1830MHz/2=915MHz
  (DN+DK/2^{20})=57.1875 → DN[6:0]=39H=57, DK[19:0]=30000H=196608

Software Programming Guide

In order to help users to implement the desired transmission, this section provides the reference setup steps for the Burst mode and Direct Mode. The differences between these two modes are the generation of the data to be transmitted and the RF block on/off control, more details are described below.

Burst Mode

1. Set the XCLKD2 bit properly according to the selected crystal oscillator.
2. Set the RF_PDB bit to 1 and the RF circuit will start to operate with the XCLK clock.
3. If users select HXT as the system clock source, check the HXTF bit. When this bit is equal to 1 it indicates that the MCU clock is ready. The MCU can use the XCLK_MCU clock as its main clock.
4. Set the RF related configuration registers.
5. If it is the first time to power up or the operation environment has large variations, set the ACAL_EN bit in the RF_OPMOD register to start an auto calibration process and poll this bit to wait for the end of the calibration process.
6. Write data to the FIFO. Set FFLEN[7:0] for the desired transmission length in bytes. Set DTR[7:0] for the data rate. Set FDEV[10:0] for the frequency deviation in FSK mode.
7. Set DIR_EN=0 to select the Burst Mode, set FSK_EN=0(OOK)/1(FSK), set TX_STROBE=1 to start the transmission. Then the data in FIFO will be automatically transferred to the RFOUT pin with FSK or OOK modulation.
8. Poll the TX_STROBE bit until it is cleared to 0 by hardware. When the TX_STROBE bit equals to 1, any write accesses to RF configuration registers are not allowed.
9. If users want to resend the previous data, just check the TX_STROBE bit, when it is in a low state set it to 1 again to initiate another transmission.
10. If TX_STROBE is in a high state and users want to forcedly terminate this transmission, just clear the TX_STROBE to zero then wait at least 1μs to let the state machine turn off the Power Amplifier Driver. Then users can reset TX_STROBE to start a new transmission.
11. Then clear RF_PDB to zero to turn off the RF circuit.

Direct Mode

1. Set the XCLKD2 bit properly according to the selected crystal oscillator.
2. Set the RF_PDB bit to 1 and the RF circuit will start to operate with the XCLK clock.
3. If HXT is selected as the system clock source, check the HXTF bit. When this bit is equal to 1 this indicates that the MCU clock is ready. The MCU can use the XCLK_MCU clock as its main clock.
4. Set the RF related configuration registers.
5. If it is the first time to power up or the operation environment has large variations, set the ACAL_EN bit in the RF_OPMOD register to start the auto calibration process and poll this bit to wait for the end of the calibration process.
6. Write the DTXD bit with the first data bit to be transmitted. Set FDEV[10:0] for the frequency deviation in the FSK mode.
7. Set DIR_EN=1 to select the Direct Mode and implement a delay time (about 20μs) to wait for the related circuit ready, set FSK_EN=0(OOK)/1(FSK).
8. Set the SX_EN bit to enable all the synthesizer block functions and implement a delay time (about 20μs) to wait for the synthesizer to stabilise.
9. Set the TX_EN bit to enable the Power Amplifier block function and then start to transmit data via the RFOUT pin.
10. Continue to update the DTXD bit with a desired data rate until all data transmissions are completed.
11. Set TX_EN=0 and delay 20μs, then set SX_EN=0 and delay 20μs.
12. Then clear RF_PDB to zero to turn off the RF circuit.

Note: When in the Direct Mode and a data transfer has completed, to allow the RF circuit to enter the power down mode to save power consumption, users must configure the related bits in the correct order as described in step11 and step12, otherwise an undesirable standby current will be generated.

Low Voltage Detector – LVD

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

LVD Register

The Low Voltage Detector function is controlled using a single register with the name LVDC. Three bits in this register, V_{LVD2}~V_{LVD0}, are used to select one of seven fixed voltages below which a low voltage condition will be determined. A low voltage condition is indicated when the LVDO bit is set. If the LVDO bit is low, this indicates that the V_{DD} voltage is above the preset low voltage value. The ENLVD 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	ENLVD	D3	VLVD2	VLVD1	VLVD0
R/W	—	—	R	R/W	R/W	R/W	R/W	R/W
POR	—	—	0	0	0	0	0	0

Bit 7~6 Unimplemented, read as "0"

Bit 5 **LVDO**: LVD Output flag
0: No Low Voltage Detected
1: Low Voltage Detected

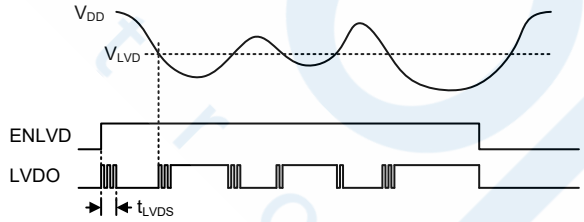
Bit 4 **ENLVD**: Low Voltage Detector Enable control
0: Disable
1: Enable

Bit 3 **D3**: Reserved bit

Bit 2~0 **VLVD2~VLVD0**: LVD Voltage selection
000: 2.4V
001: 2.0V
010: 2.2V
011: 2.4V
100: 2.7V
101: 2.9V
110: 3.0V
111: 3.3V

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 2.0V and 3.3V. 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. When these devices enter the SLEEP or DEEP SLEEP Mode, the low voltage detector will automatically be disabled even if the ENLVD bit is high. After enabling the Low Voltage Detector, a time delay t_{LVDS} should be allowed for the circuitry to stabilise before reading the LVDO bit. Note also that as the V_{DD} voltage may rise and fall rather slowly, at the voltage nears that of V_{LVD}, there may be multiple bit LVDO transitions.



LVD Operation

The Low Voltage Detector also has its own interrupt which is contained within one of the Multi-function interrupts, providing an alternative means of low voltage detection, in addition to polling the LVDO bit. The interrupt will only be generated after a delay of t_{LVD} after the LVDO bit has been set high by a low voltage condition, i.e., V_{DD} falls below the preset LVD voltage. In this case, the LVF interrupt request flag will be set, causing an interrupt to be generated. This will cause these devices to wake up from the IDLE Mode, however if the Low Voltage Detector wake up function is not required then the LVF flag should be first set high before these devices enter the IDLE Mode.

Interrupts

Interrupts are an important part of any microcontroller system. When an external event or an internal function such as a Timer Module requires microcontroller attention, their corresponding interrupt will enforce a temporary suspension of the main program allowing the microcontroller to direct attention to their respective needs. These devices contain several external interrupt and internal interrupt functions. The external interrupts are generated by the action of the external INT0~INT1 pins, while the internal interrupts are generated by various internal functions such as the TMs, LVD and Time bases, etc.

Interrupt Registers

Overall interrupt control, which basically means the setting of request flags when certain microcontroller conditions occur and the setting of interrupt enable bits by the application program, is controlled by a series of registers, located in the Special Purpose Data Memory, as shown in the accompanying table. The registers falls into three categories. The first is the INTC0~INTC2 registers which setup the primary interrupts, the second is the MFI0~MFI2 registers which setup the Multi-function interrupts. Finally there is an INTEG register to setup the external interrupt trigger edge type.

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

Function	Enable Bit	Request Flag	Notes
Global	EMI	—	—
INTn Pin	INTnE	INTnF	n=1 for BC66F2235 n=0~1 for BC66F2245/ BC66F2255
Time Bases	TBnE	TBnF	n=0~1
Multi-function	MFnE	MFnF	n=0~2
LVD	LVE	LVF	—
RF TX FIFO Length Margin Detect	FFMGE	FFMGF	—
RF Burst Mode Transmit Complete	BMTCE	BMTCF	—
Touch key TKRCOV	TKRCOVE	TKRCOVF	—
Touch key module TKTH	TKTHE	TKTHF	—
CTM	CTMnPE	CTMnPF	n=0~1
	CTMnAE	CTMnAF	
PTM	PTMnPE	PTMnPF	n=0
	PTMnAE	PTMnAF	

Interrupt Register Bit Naming Conventions

Register Name	Bit							
	7	6	5	4	3	2	1	0
INTEG	—	—	—	—	INT1S1	INT1S0	INT0S1	INT0S0
INTC0	—	TB0F	INT1F	INT0F	TB0E	INT1E	INT0E	EMI
INTC1	MF2F	MF1F	MF0F	TB1F	MF2E	MF1E	MF0E	TB1E
INTC2	ADF	USIMF	TKTHF	TKRCOVF	ADE	USIME	TKTHE	TKRCOVE
MFI0	CTM1AF	CTM1PF	CTM0AF	CTM0PF	CTM1AE	CTM1PE	CTM0AE	CTM0PE
MFI1	—	—	PTM0AF	PTM0PF	—	—	PTM0AE	PTM0PE
MFI2	—	LVF	FFMGF	BMTCF	—	LVE	FFMGE	BMTCE

Interrupt Register List

• **INTEG Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	INT1S1	INT1S0	INT0S1	INT0S0
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0

Bit 7~4 Unimplemented, read as "0"

Bit 3~2 **INT1S1~INT1S0**: Interrupt edge control for INT1 pin
00: Disable
01: Rising edge
10: Falling edge
11: Rising and falling edges

Bit 1~0 **INT0S1~INT0S0**: Interrupt edge control for INT0 pin
00: Disable
01: Rising edge
10: Falling edge
11: Rising and falling edges

Note: For the BC66F2235 device, these bits are reserved and should be fixed at "00".

• **INTC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	TB0F	INT1F	INT0F	TB0E	INT1E	INT0E	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 **INT1F**: INT1 interrupt request flag
0: No request
1: Interrupt request

Bit 4 **INT0F**: INT0 interrupt request flag
0: No request
1: Interrupt request

Note: For the BC66F2235 device, this bit is reserved and should be fixed at "0".

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

Bit 2 **INT1E**: INT1 interrupt control
0: Disable
1: Enable

Bit 1 **INT0E**: INT0 interrupt control
0: Disable
1: Enable

Note: For the BC66F2235 device, this bit is reserved and should be fixed at "0".

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

• **INTC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	MF2F	MF1F	MF0F	TB1F	MF2E	MF1E	MF0E	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 **MF2F**: Multi-function interrupt 2 request flag
0: No request
1: Interrupt request

Bit 6 **MF1F**: Multi-function interrupt 1 request flag
0: No request
1: Interrupt request

Bit 5 **MF0F**: Multi-function interrupt 0 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 **MF2E**: Multi-function interrupt 2 control
0: Disable
1: Enable

Bit 2 **MF1E**: Multi-function interrupt 1 control
0: Disable
1: Enable

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

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

• **INTC2 Register**

Bit	7	6	5	4	3	2	1	0
Name	ADF	USIMF	TKTHF	TKRCOVF	ADE	USIME	TKTHE	TKRCOVE
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7 **ADF**: A/D Converter interrupt request flag
0: No request
1: Interrupt request

Bit 6 **USIMF**: USIM interrupt request flag
0: No request
1: Interrupt request

Bit 5 **TKTHF**: Touch key module TKTH interrupt request flag
0: No request
1: Interrupt request

Bit 4 **TKRCOVF**: Touch key TKRCOV interrupt request flag
0: No request
1: Interrupt request

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

Bit 2 **USIME**: USIM interrupt control
0: Disable
1: Enable

Bit 1 **TKTHE**: Touch key module TKTH interrupt control
0: Disable
1: Enable

Bit 0 **TKRCOVE**: Touch key TKRCOV interrupt control
0: Disable
1: Enable

• **MFI0 Register**

Bit	7	6	5	4	3	2	1	0
Name	CTM1AF	CTM1PF	CTM0AF	CTM0PF	CTM1AE	CTM1PE	CTM0AE	CTM0PE
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7 **CTM1AF**: CTM1 Comparator A match interrupt request flag
0: No request
1: Interrupt request

Bit 6 **CTM1PF**: CTM1 Comparator P match interrupt request flag
0: No request
1: Interrupt request

Bit 5 **CTM0AF**: CTM0 Comparator A match interrupt request flag
0: No request
1: Interrupt request

Bit 4 **CTM0PF**: CTM0 Comparator P match interrupt request flag
0: No request
1: Interrupt request

Bit 3 **CTM1AE**: CTM1 Comparator A match interrupt control
0: Disable
1: Enable

Bit 2 **CTM1PE**: CTM1 Comparator P match interrupt control
0: Disable
1: Enable

Bit 1 **CTM0AE**: CTM0 Comparator A match interrupt control
0: Disable
1: Enable

Bit 0 **CTM0PE**: CTM0 Comparator P match interrupt control
0: Disable
1: Enable

• **MFI1 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	PTM0AF	PTM0PF	—	—	PTM0AE	PTM0PE
R/W	—	—	R/W	R/W	—	—	R/W	R/W
POR	—	—	0	0	—	—	0	0

Bit 7~6 Unimplemented, read as "0"

Bit 5 **PTM0AF**: PTM0 Comparator A match interrupt request flag
0: No request
1: Interrupt request

Bit 4 **PTM0PF**: PTM0 Comparator P match interrupt request flag
0: No request
1: Interrupt request

Bit 3~2 Unimplemented, read as "0"

Bit 1 **PTM0AE**: PTM0 Comparator A match interrupt control
0: Disable
1: Enable

Bit 0 **PTM0PE**: PTM0 Comparator P match interrupt control
0: Disable
1: Enable

• **MFI2 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	LVF	FFMGF	BMTCF	—	LVE	FFMGE	BMTCE
R/W	—	R/W	R/W	R/W	—	R/W	R/W	R/W
POR	—	0	0	0	—	0	0	0

Bit 7 Unimplemented, read as "0"

Bit 6 **LVF**: LVD interrupt request flag
0: No request
1: Interrupt request

Bit 5 **FFMGF**: RF TX FIFO Length Margin Detect interrupt request flag
0: No request
1: Interrupt request

Bit 4 **BMTCF**: RF Burst Mode Transmit Complete interrupt request flag
0: No request
1: Interrupt request

Bit 3 Unimplemented, read as "0"

Bit 2 **LVE**: LVD interrupt control
0: Disable
1: Enable

Bit 1 **FFMGE**: RF TX FIFO Length Margin Detect interrupt control
0: Disable
1: Enable

Bit 0 **BMTCE**: RF Burst Mode Transmit Complete interrupt control
0: Disable
1: Enable

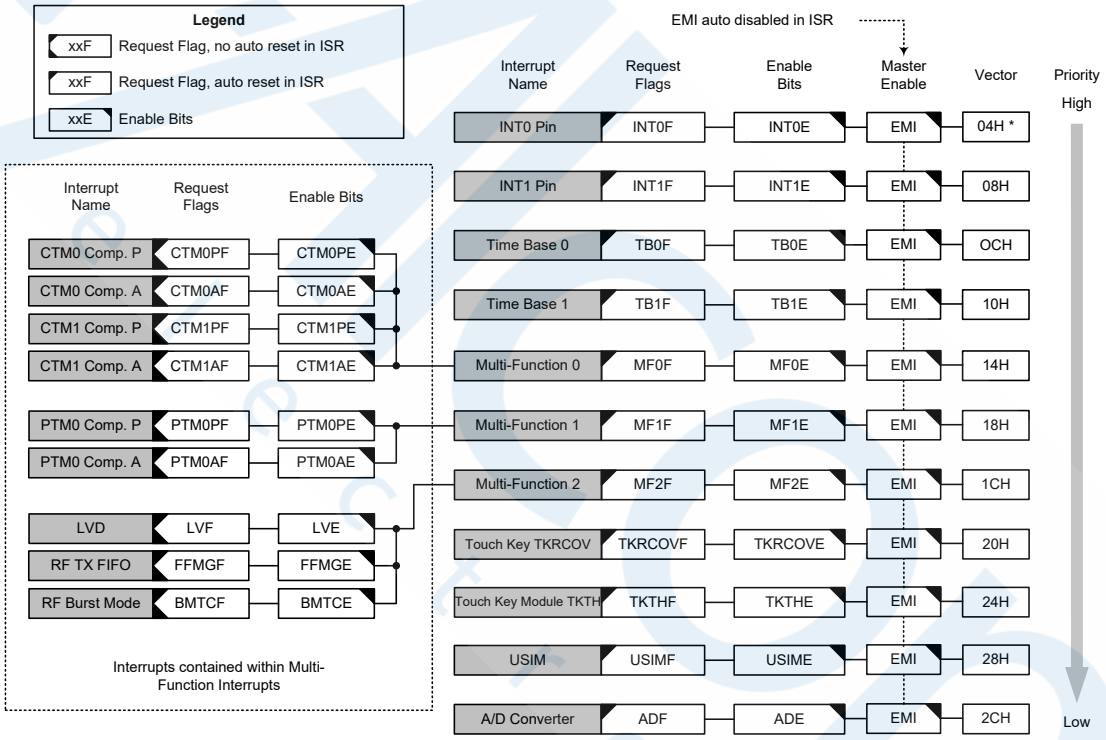
Interrupt Operation

When the conditions for an interrupt event occur, such as a TM Comparator P or Comparator A or A/D conversion completion, etc., the relevant interrupt request flag will be set. Whether the request flag actually generates a program jump to the relevant interrupt vector is determined by the condition of the interrupt enable bit. If the enable bit is set high then the program will jump to its relevant vector; if the enable bit is zero then although the interrupt request flag is set an actual interrupt will not be generated and the program will not jump to the relevant interrupt vector. The global interrupt enable bit, if cleared to zero, will disable all interrupts.

When an interrupt is generated, the Program Counter, which stores the address of the next instruction to be executed, will be transferred onto the stack. The Program Counter will then be loaded with a new address which will be the value of the corresponding interrupt vector. The microcontroller will then fetch its next instruction from this interrupt vector. The instruction at this vector will usually be a "JMP" which will jump to another section of program which is known as the interrupt service routine. Here is located the code to control the appropriate interrupt. The interrupt service routine must be terminated with a "RETI", which retrieves the original Program Counter address from the stack and allows the microcontroller to continue with normal execution at the point where the interrupt occurred.

The various interrupt enable bits, together with their associated request flags, are shown in the accompanying diagrams with their order of priority. Some interrupt sources have their own individual vector while others share the same multi-function interrupt vector. Once an interrupt subroutine is serviced, all 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 these devices if it is in SLEEP or IDLE Mode, however to prevent a wake-up from occurring the corresponding flag should be set before these devices are in SLEEP or IDLE Mode.



Note: For the BC66F2235 device, the 04H interrupt vector is reserved.

Interrupt Structure

External Interrupts

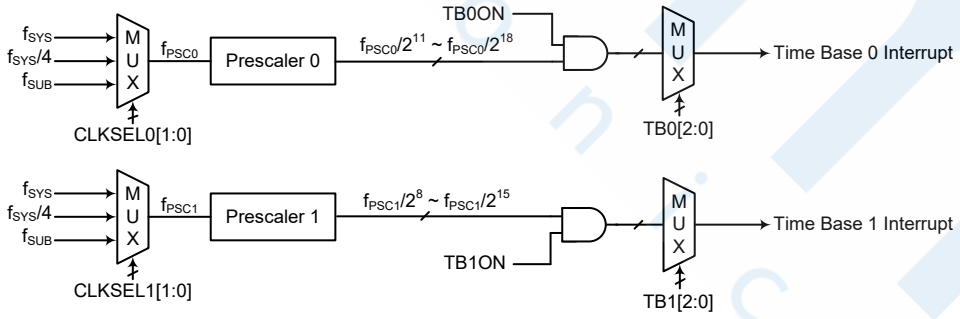
The external interrupts are controlled by signal transitions on the pins INT0~INT1. An external interrupt request will take place when the external interrupt request flags, INT0F~INT1F, are set, which will occur when a transition, whose type is chosen by the edge select bits, appears on the external interrupt pins. To allow the program to branch to their respective interrupt vector addresses, the global interrupt enable bit, EMI, and respective external interrupt enable bit, INT0E~INT1E, must first be set. Additionally the correct interrupt edge type must be selected using the INTEG register to enable the external interrupt function and to choose the trigger edge type. As the external interrupt pins are pin-shared with I/O pins, they can only be configured as external interrupt pins if their external interrupt enable bit in the corresponding interrupt register has been set and the external interrupt pin is selected by the corresponding pin-shared function selection bits. The pin must also be setup as an input by setting the corresponding bit in the port control register. When the interrupt is enabled, the stack is not full and the correct transition type appears on the external interrupt pin, a subroutine call to the external interrupt vector, will take place. When the interrupt is serviced, the external interrupt request flags, INT0F~INT1F, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts. Note that any pull-high resistor selections on the external interrupt pins will remain valid even if the pin is used as an external interrupt input.

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

Time Base Interrupts

The function of the Time Base Interrupts is to provide regular time signals in the form of an internal interrupt. They are controlled by the overflow signal from their respective internal timers. When this happens their respective interrupt request flags, TB0F or TB1F will be set. To allow the program to branch to their respective interrupt vector addresses, the global interrupt enable bit, EMI and Time Base enable bit, TB0E or TB1E, must first be set. When the interrupt is enabled, the stack is not full and the Time Base overflows, a subroutine call to their respective vector locations will take place. When the interrupt is serviced, the interrupt request flag, TB0F or TB1F, will be automatically reset and the EMI bit will be cleared to disable other interrupts.

The purpose of the Time Base Interrupts is to provide an interrupt signal at fixed time periods. Their respective clock source, f_{PSC0} or f_{PSC1}, originates from the internal clock source f_{SYS}, $f_{SYS}/4$ or f_{SUB} and then passes through a divider, the division ratio of which is selected by programming the appropriate bits in the TB0C or TB1C registers to obtain longer interrupt periods whose value ranges. The clock source which in turn controls the Time Base interrupt period, is selected using the CLKSEL0[1:0] and CLKSEL1[1:0] bits in the PSC0R and PSC1R register respectively.



Time Base Interrupts

• **PSC0R Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	CLKSEL01	CLKSEL00
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2 Unimplemented, read as "0"

Bit 1~0 **CLKSEL01~CLKSEL00**: Prescaler 0 clock source f_{PSC0} selection
00: f_{SYS}
01: $f_{SYS}/4$
1x: f_{SUB}

Care must be taken that when the MCU enters the DEEP SLEEP Mode, the Time Base 0 clock source will come from f_{LIRC}, irrespective of the Prescaler 0 clock source selection. After the MCU wakes up from the DEEP SLEEP mode, the Time Base 0 clock source will come from f_{PSC0}.

• **PSC1R Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	CLKSEL11	CLKSEL10
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2 Unimplemented, read as "0"

Bit 1~0 **CLKSEL11~CLKSEL10**: Prescaler 1 clock source f_{PSC1} selection
00: f_{SYS}
01: $f_{SYS}/4$
1x: f_{SUB}

• **TB0C Register**

Bit	7	6	5	4	3	2	1	0
Name	TB0ON	—	—	—	—	TB02	TB01	TB00
R/W	R/W	—	—	—	—	R/W	R/W	R/W
POR	0	—	—	—	—	0	0	0

Bit 7 **TB0ON**: Time Base 0 Control
0: Disable
1: Enable

Bit 6~3 Unimplemented, read as "0"

Bit 2~0 **TB02~TB00**: Time Base 0 time-out period selection
000: $2^{11}/f_{PSC0}$
001: $2^{12}/f_{PSC0}$
010: $2^{13}/f_{PSC0}$
011: $2^{14}/f_{PSC0}$
100: $2^{15}/f_{PSC0}$
101: $2^{16}/f_{PSC0}$
110: $2^{17}/f_{PSC0}$
111: $2^{18}/f_{PSC0}$

• **TB1C Register**

Bit	7	6	5	4	3	2	1	0
Name	TB1ON	—	—	—	—	TB12	TB11	TB10
R/W	R/W	—	—	—	—	R/W	R/W	R/W
POR	0	—	—	—	—	0	0	0

Bit 7 **TB1ON**: Time Base 1 Control
0: Disable
1: Enable

Bit 6~3 Unimplemented, read as "0"

Bit 2~0 **TB12~TB10**: Time Base 1 time-out period selection
000: $2^8/f_{PSC1}$
001: $2^9/f_{PSC1}$
010: $2^{10}/f_{PSC1}$
011: $2^{11}/f_{PSC1}$
100: $2^{12}/f_{PSC1}$
101: $2^{13}/f_{PSC1}$
110: $2^{14}/f_{PSC1}$
111: $2^{15}/f_{PSC1}$

Multi-function Interrupts

Within these devices there are three Multi-function interrupts. Unlike the other independent interrupts, these interrupts have no independent source, but rather are formed from other existing interrupt sources, namely the TM Interrupts, LVD Interrupt, RF FFMG Length Margin Detect Interrupt and RF Burst Mode Transmit Complete Interrupt.

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

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

TM Interrupts

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

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

the TM interrupt request flags will not be automatically cleared, they have to be cleared by the application program.

LVD Interrupt

The Low Voltage Detector Interrupt is contained within the Multi-function Interrupt. An LVD Interrupt request will take place when the LVD Interrupt request flag, LVF, is set, which occurs when the Low Voltage Detector function detects a low power supply voltage. To allow the program to branch to its interrupt vector address, the global interrupt enable bit, EMI, Low Voltage Interrupt enable bit, LVE, and associated Multi-function interrupt enable bit, must first be set. When the interrupt is enabled, the stack is not full and a low voltage condition occurs, a subroutine call to the relevant Multi-function Interrupt vector, will take place. When the Low Voltage Interrupt is serviced, the EMI bit will be automatically cleared to disable other interrupts, however only the Multi-function interrupt request flag will be also automatically cleared. As the LVF flag will not be automatically cleared, it has to be cleared by the application program.

RF TX FIFO Length Margin Detect Interrupt

The RF TX FIFO Length Margin Detect interrupt is contained within the Multi-function Interrupt. A TX FIFO Length Margin Detect Interrupt request will take place when its interrupt request flag, FFMGF, is set, which occurs when the TX FIFO length is less than the threshold length. To allow the program to branch to its interrupt vector address, the global interrupt enable bit, EMI, and TX FIFO Length Margin Detect Interrupt enable bit, FFMGE, and associated Multi-function interrupt enable bit, must first be set. When the interrupt is enabled, the stack is not full and the previously mentioned condition occurs, a subroutine call to the relevant Multi-function interrupt vector will take place. When the interrupt is serviced, the EMI bit will be automatically cleared to disable other interrupts, however only the Multi-function interrupt request flag will be also automatically cleared. As the FFMGF flag will not be automatically cleared, it has to be cleared by the application program.

RF Burst Mode Transmit Complete Interrupt

The RF Burst Mode Transmit Complete interrupt is contained within the Multi-function Interrupt. A RF Burst Mode Transmit Complete Interrupt request will take place when its interrupt request flag, BMTCF, is set, which occurs when the TX packet has been sent completely in the Burst Mode. To allow the program to branch to its interrupt vector address, the global interrupt enable bit, EMI, and RF Burst Mode Transmit Complete Interrupt enable bit, BMTCE, and associated Multi-function interrupt enable bit, must first be set. When the interrupt is enabled, the stack is not full and the previously mentioned condition occurs, a subroutine call to the relevant Multi-function interrupt vector will take place. When the interrupt is serviced, the EMI bit will be automatically cleared to disable other interrupts, however only the Multi-function interrupt request flag will be also automatically cleared. As the BMTCF flag will not be automatically cleared, it has to be cleared by the application program.

Touch Key TKRCOV Interrupt

A Touch Key TKRCOV Interrupt request will take place when the Touch Key TKRCOV Interrupt request flag, TKRCOVF, is set, which occurs when the time slot counter overflows in the manual mode or all the touch key auto scan operations finish in the auto scan mode or the 16-bit C/F counter content is written into the corresponding touch key data memory at the end of the last scan operation in the TB0 time-out cycle in the periodic auto scan mode. To allow the program to branch to its interrupt vector address, the global interrupt enable bit, EMI, and Touch Key TKRCOV Interrupt enable bit, TKRCOVE, must first be set. When the interrupt is enabled, the stack is not full and any

of the above described situations occurs, a subroutine call to its interrupt vector, will take place. When the interrupt is serviced, the Touch Key TKRCOV Interrupt request flag, TKRCOVF, will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts.

Touch Key Module TKTH Interrupt

A Touch Key Module TKTH Interrupt request will take place when the Touch Key Module TKTH Interrupt request flag, TKTHF, is set, which occurs when the Touch Key Module 16-bit C/F counter is less than the lower threshold if MnKmTHS=0, or larger than the upper threshold if MnKmTHS=1. To allow the program to branch to its interrupt vector address, the global interrupt enable bit, EMI, and Touch Key Module TKTH Interrupt enable bit, TKTHE, must first be set. When the interrupt is enabled, the stack is not full and any of the above described threshold comparison conditions occurs, a subroutine call to its interrupt vector, will take place. When the interrupt is serviced, the Touch Key Module TKTH Interrupt request flag, TKTHF, will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts.

Universal Serial Interface Module Interrupt

The Universal Serial Interface Module Interrupt, also known as the USIM interrupt, will take place when the USIM Interrupt request flag, USIMF, is set. As the USIM interface can operate in three modes which are SPI mode, I^2C mode and UART mode, the USIMF flag can be set by different conditions depending on the selected interface mode.

If the SPI or I^2C mode is selected, the USIM interrupt can be triggered when a byte of data has been received or transmitted by the SPI/I^2C interface, or an I^2C slave address match occurs, or an I^2C bus time-out occurs. If the UART mode is selected, several individual UART conditions including a transmitter data register empty, transmitter idle, receiver data available, receiver overrun, address detect and an URX/UTX pin wake-up, can generate a USIM interrupt with the USIMF flag bit set high.

To allow the program to branch to its interrupt vector address, the global interrupt enable bit, EMI, and the Serial Interface Interrupt enable bit, USIME, must first be set. When the interrupt is enabled, the stack is not full and any of the above described situations occurs, a subroutine call to the respective Interrupt vector, will take place. When the interrupt is serviced, the Universal Serial Interface Interrupt flag, USIMF, will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts.

Note that if the USIM interrupt is triggered by the UART interface, after the interrupt has been serviced, the UUSR register flags will only be cleared when certain actions are taken by the UART, the details of which are given in the UART section.

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 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.

Interrupt Wake-up Function

Each of the interrupt functions has the capability of waking up the microcontroller when in the SLEEP or IDLE Mode. A wake-up is generated when an interrupt request flag changes from low to high and is independent of whether the interrupt is enabled or not. Therefore, even though these devices are in the SLEEP or IDLE Mode and its system oscillator stopped, situations such as external edge transitions on the external interrupt pins or a low power supply voltage may cause their respective interrupt flag to be set high and consequently generate an interrupt.

However, only the Time Base 0 interrupt has the capability of waking up the microcontroller when in the DEEP SLEEP Mode, where the Time Base 0 clock source comes from f_{LIRC} and the wake-up status is indicated by the TB0_WAKE flag.

Care must 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 these devices enter the DEEP SLEEP, SLEEP or IDLE Mode. The interrupt enable bits have no effect on the interrupt wake-up function.

Programming Considerations

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

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

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

Every interrupt has the capability of waking up the microcontroller when it is in the SLEEP or IDLE Mode and the Time Base 0 interrupt can also wake up the microcontroller when it is in the DEEP SLEEP 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 request flag should be first set high before enter the DEEP SLEEP, SLEEP or IDLE Mode.

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

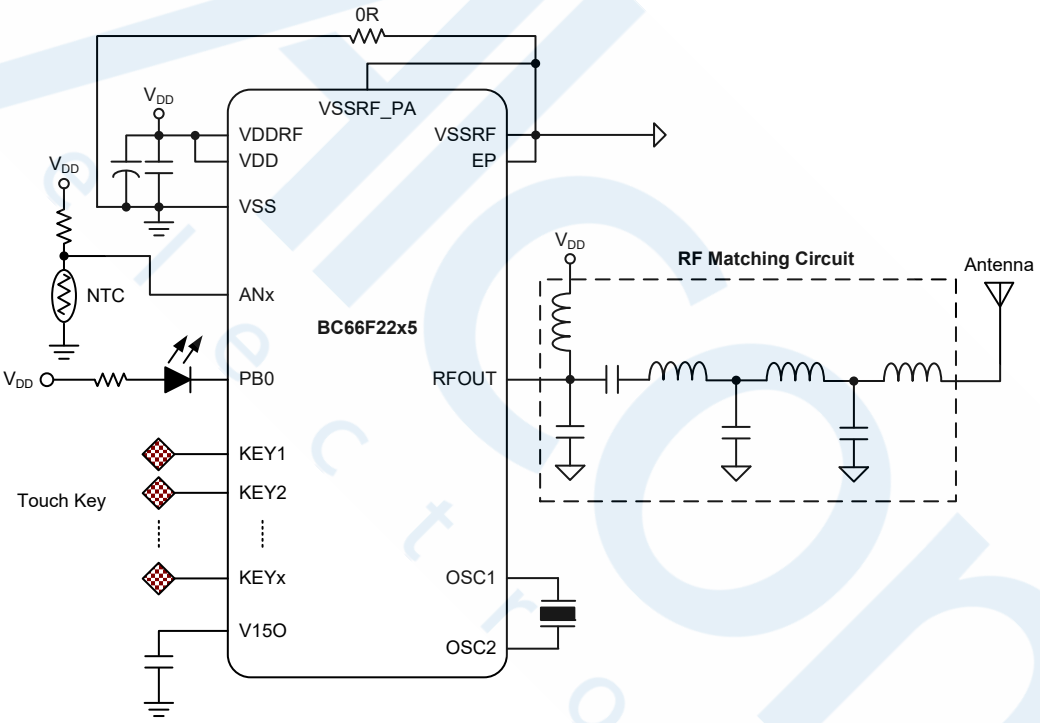
To return from an interrupt subroutine, either a RET or RETI instruction may be executed. The RETI instruction in addition to executing a return to the main program also automatically sets the EMI bit high to allow further interrupts. The RET instruction however only executes a return to the main program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

Configuration Options

Configuration options refer to certain options within the MCU that are programmed into these devices during the programming process. During the development process, these options are selected using the HT-IDE software development tools. All options must be defined for proper system function, the details of which are shown in the table.

No.	Options
Watchdog Timer Option	
1	Watchdog Timer Function: 1. Always enable 2. By WDTC control
LVR Option	
2	LVR Function: 1. Disable 2. Enable

Application Circuits



Instruction Set

Introduction

Central to the successful operation of any microcontroller is its instruction set, which is a set of program instruction codes that directs the microcontroller to perform certain operations. In the case of Holtek microcontroller, a comprehensive and flexible set of over 60 instructions is provided to enable programmers to implement their application with the minimum of programming overheads.

For easier understanding of the various instruction codes, they have been subdivided into several functional groupings.

Instruction Timing

Most instructions are implemented within one instruction cycle. The exceptions to this are branch, call, or table read instructions where two instruction cycles are required. One instruction cycle is equal to 4 system clock cycles, therefore in the case of an 8MHz system oscillator, most instructions would be implemented within 0.5μs and branch or call instructions would be implemented within 1μs. Although instructions which require one more cycle to implement are generally limited to the JMP, CALL, RET, RETI and table read instructions, it is important to realize that any other instructions which involve manipulation of the Program Counter Low register or PCL will also take one more cycle to implement. As instructions which change the contents of the PCL will imply a direct jump to that new address, one more cycle will be required. Examples of such instructions would be "CLR PCL" or "MOV PCL, A". For the case of skip instructions, it must be noted that if the result of the comparison involves a skip operation then this will also take one more cycle, if no skip is involved then only one cycle is required.

Moving and Transferring Data

The transfer of data within the microcontroller program is one of the most frequently used operations. Making use of 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:
$[m] \leftarrow [m] - 1$
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:
$ACC \leftarrow [m] - 1$
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

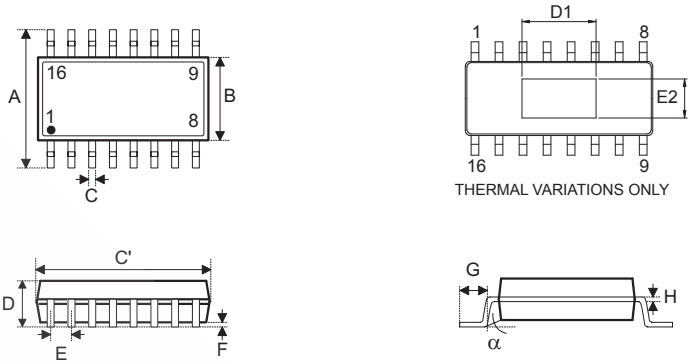
Package Information

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

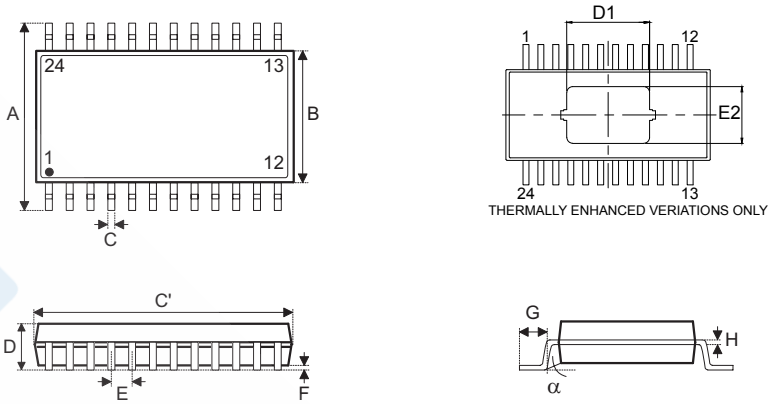
16-pin NSOP-EP (150mil) Outline Dimensions



Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	—	0.236 BSC	—
B	—	0.154 BSC	—
C	0.012	—	0.020
C’	—	0.390 BSC	—
D	—	—	0.069
D1	0.059	—	—
E	—	0.050 BSC	—
E2	0.039	—	—
F	0.000	—	0.006
G	0.016	—	0.050
H	0.004	—	0.010
α	0°	—	8°

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	—	6.00 BSC	—
B	—	3.90 BSC	—
C	0.31	—	0.51
C’	—	9.90 BSC	—
D	—	—	1.75
D1	1.50	—	—
E	—	1.27 BSC	—
E2	1.00	—	—
F	0.00	—	0.15
G	0.40	—	1.27
H	0.10	—	0.25
α	0°	—	8°

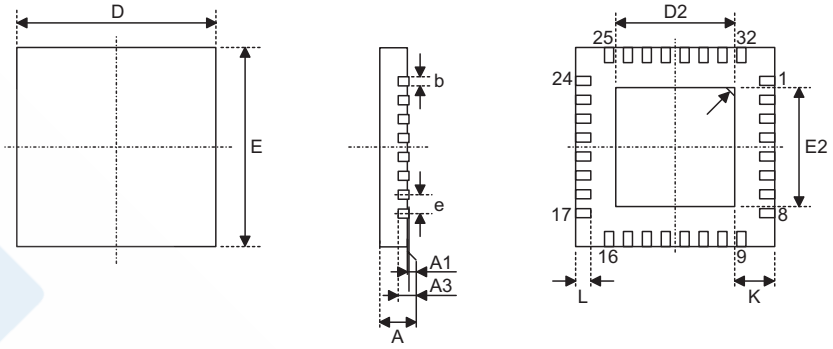
24-pin SSOP-EP (150mil) Outline Dimensions



Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	—	0.236 BSC	—
B	—	0.154 BSC	—
C	0.008	—	0.012
C’	—	0.341 BSC	—
D	—	—	0.069
D1	—	0.140	—
E	—	0.025 BSC	—
E2	—	0.096	—
F	0.000	—	0.004
G	0.016	—	0.050
H	0.004	—	0.010
α	0°	—	8°

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	—	6.00 BSC	—
B	—	3.90 BSC	—
C	0.20	—	0.30
C’	—	8.66 BSC	—
D	—	—	1.75
D1	—	3.56	—
E	—	0.635 BSC	—
E2	—	2.44	—
F	0.00	—	0.10
G	0.41	—	1.27
H	0.10	—	0.25
α	0°	—	8°

SAW Type 32-pin QFN (4mm×4mm×0.75mm) Outline Dimensions



Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	0.028	0.030	0.031
A1	0.000	0.001	0.002
A3	—	0.008 BSC	—
b	0.006	0.008	0.010
D	—	0.157 BSC	—
E	—	0.157 BSC	—
e	—	0.016 BSC	—
D2	0.104	0.106	0.108
E2	0.104	0.106	0.108
L	0.014	0.016	0.018
K	0.008	—	—

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	0.70	0.75	0.80
A1	0.00	0.02	0.05
A3	—	0.203 BSC	—
b	0.15	0.20	0.25
D	—	4.00 BSC	—
E	—	4.00 BSC	—
e	—	0.40 BSC	—
D2	2.65	2.70	2.75
E2	2.65	2.70	2.75
L	0.35	0.40	0.45
K	0.20	—	—