Ultra-Low Power Touch Key Flash MCU

BS83A02L/BS83B04L

Revision: V1.10 Date: February 10, 2020

www.holtek.com

Table of Contents

Features – BS83A02L

CPU Features

- Operating voltage
  - f_{SYS}=8MHz: 1.8V~5.5V
- Up to 0.5μs instruction cycle with 8MHz system clock at V_{DD}=5V
- Power down and wake-up functions to reduce power consumption
- Oscillator types
  - Internal High Speed 8MHz RC – HIRC
  - Internal Low Speed 2kHz RC – LIRC
- Multi-mode operation: FAST, SLOW, IDLE and SLEEP
- Fully integrated internal oscillators require no external components
- All instructions executed in one or two instruction cycles
- Table read instructions
- 63 powerful instructions
- 2-level subroutine nesting
- Bit manipulation instruction

Peripheral Features

- Flash Program Memory: 1K×14
- RAM Data Memory: 64×8
- 2 touch key functions – fully integrated without requiring external components
- Watchdog Timer function
- 4 bidirectional I/O lines
- Single external interrupt line shared with I/O pin
- Single 8-bit programmable Timer/Event Counter with overflow interrupt and prescaler
- Single Time-Base function for generation of fixed time interrupt signals
- Low voltage reset function
- Package types: 6-pin DFN/SOT23, 8-pin SOP

Features – BS83B04L

CPU Features

- Operating Voltage
  - f_{SYS}=2MHz: 1.8V~5.5V
  - f_{SYS}=4MHz: 1.8V~5.5V
  - f_{SYS}=8MHz: 1.8V~5.5V
- Up to 0.5μs instruction cycle with 8MHz system clock at V_{DD}=5V
- Power down and wake-up functions to reduce power consumption
- Oscillator types
  - Internal High Speed 2/4/8MHz RC – HIRC
  - Internal Low Speed 2kHz RC – LIRC

- Multi-mode operation: FAST, SLOW, IDLE and SLEEP
- Fully integrated internal oscillators require no external components
- All instructions executed in one or two instruction cycles
- Table read instructions
- 63 powerful instructions
- 4-level subroutine nesting
- Bit manipulation instruction

Peripheral Features

- Flash Program Memory: 2K×16
- Data Memory: 128×8
- True EEPROM Memory: 32×8
- 4 touch key functions – fully integrated without requiring external components
- Watchdog Timer function
- 8 bidirectional I/O lines
- Single external interrupt line shared with I/O pin
- Single Timer Module for time measurement, compare match output, PWM output function
- Single Time-Base function for generation of fixed time interrupt signals
- I^2C interface
- Low voltage reset function
- Package types: 8-pin SOP, 10-pin DFN/MSOP

General Description

The series of devices are Flash Memory type 8-bit high performance RISC architecture microcontrollers with fully integrated touch key functions. With all touch key functions provided internally and with the convenience of Flash Memory multi-programming features, each device has all the features to offer designers a reliable and easy means of implementing Touch Keyes within their products applications.

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

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

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

Selection Table

Most features are common to all devices and the main features distinguishing them are Oscillator Types, Memory capacity, I/O count, Touch key count, Time Base, Timers, Interface, Stack and Package types. The following table summarises the main features of each device.

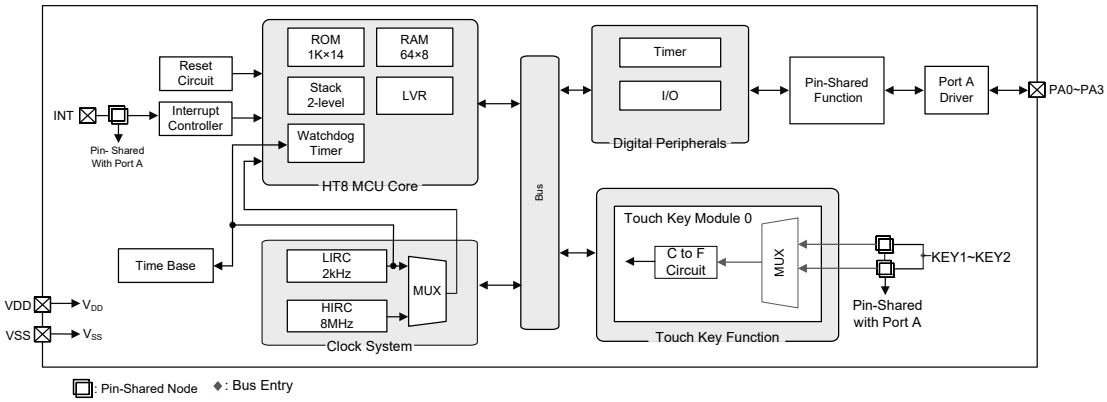
Device	Oscillator Types	Program Memory	Data Memory	True EEPROM	I/O	External Interrupt
BS83A02L	8MHz HIRC 2kHz LIRC	1K×14	64×8	—	4	1
BS83B04L	2/4/8MHz HIRC 2kHz LIRC	2K×16	128×8	32×8	8	1

Device	Time Base	Timer Module	Timer/Event Counter	Touch Key	Interface	Stack	Packages
BS83A02L	1	—	8-bit Timer×1	4	—	2	6DFN/SOT23; 8SOP
BS83B04L	1	10-bit CTM×1	—	8	I²C×1	4	8SOP; 10DFN/MSOP

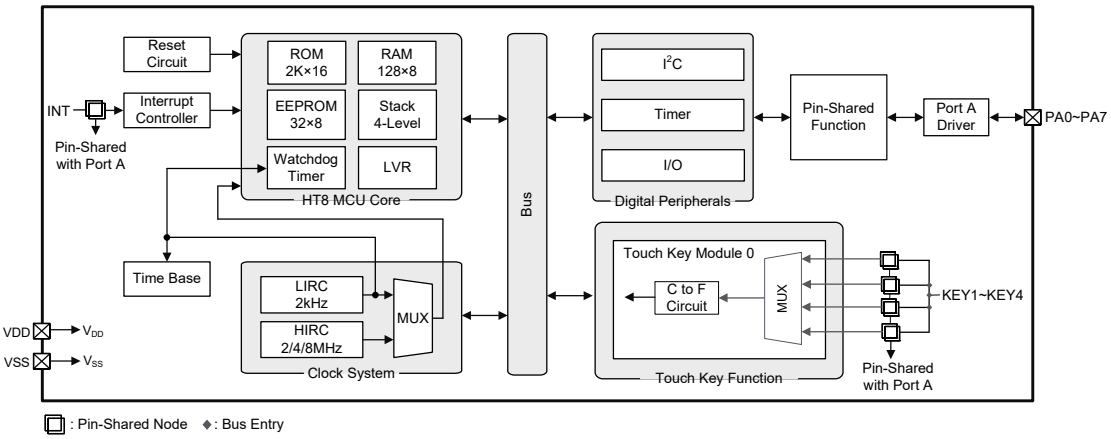
Note: As devices exist in more than one package format, the table reflects the situation for the package with the most pins.

Block Diagram

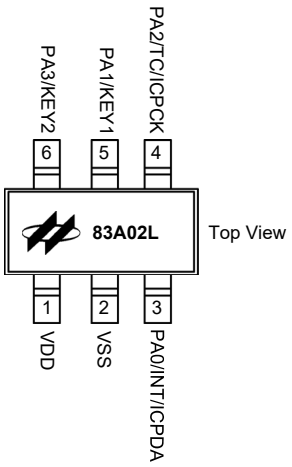
BS83A02L



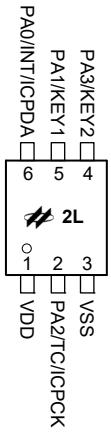
BS83B04L



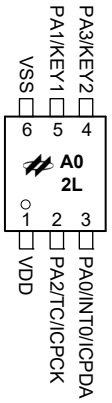
Pin Assignment



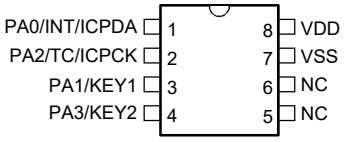
BS83A02L
SOT23-6



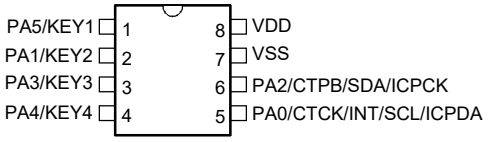
BS83A02L
6 DFN-A (2×2×0.35mm)



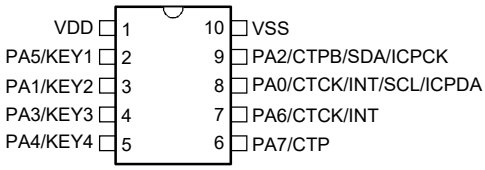
BS83A02L
6 DFN-B (2×2×0.75mm)



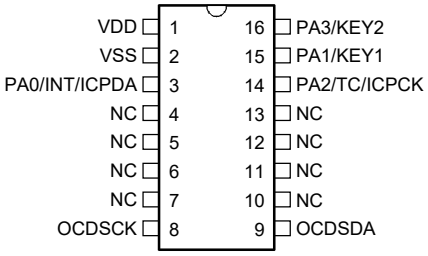
BS83A02L
8 SOP-A



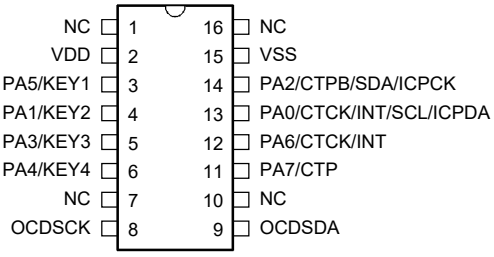
BS83B04L
8 SOP-A



BS83B04L
10 MSOP-A/DFN-A



BS83AV02L
16 NSOP-A



BS83BV04L
16 NSOP-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 16-pin NSOP package type is only for OCDS EV chips. The OCDSDA and OCDSCK pins are the OCDS dedicated pins.
3. For less pin-count package types there will be unbonded pins which should be properly configured to avoid unwanted current consumption resulting from floating input conditions. Refer to the "Standby Current Considerations" and "Input/Output Ports" sections.

Pin Descriptions

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

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

BS83A02L

Pin Name	Function	OPT	I/T	O/T	Description
PA0/INT/ICPDA	PA0	PAPU PAWU	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	INT	—	ST	—	External interrupt input
	ICPDA	—	ST	CMOS	ICP data/address
PA1/KEY1	PA1	PAPU PAWU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	KEY1	PAS0	AN	—	Touch key input
PA2/TC/ICPCK	PA2	PAPU PAWU	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	TC	—	ST	—	Timer/Event Counter external clock input
	ICPCK	—	ST	—	ICP clock
PA3/KEY2	PA3	PAPU PAWU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	KEY2	PAS0	AN	—	Touch key input
VDD	VDD	—	PWR	—	Positive power supply
VSS	VSS	—	PWR	—	Negative power supply, ground
NC	NC	—	—	—	No connection
For 16-pin NSOP package type only					
OCDSDA	OCDSDA	—	ST	CMOS	OCDS data/address pin, for EV chip only
OCDSCK	OCDSCK	—	ST	—	OCDS clock pin, for EV chip only

Legend: I/T: Input type; O/T: Output type;
OPT: Optional by register option;
PWR: Power; ST: Schmitt Trigger input;
CMOS: CMOS output; AN: Analog signal.

BS83B04L

Pin Name	Function	OPT	I/T	O/T	Description
PA0/CTCK/INT/SCL/ICPDA	PA0	PAPU PAWU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	CTCK	PAS0 IFS	ST	—	CTM clock input
	INT	PAS0 IFS INTEG INTC0	ST	—	External Interrupt input
	SCL	PAS0	ST	NMOS	I²C clock line
	ICPDA	—	ST	CMOS	ICP data/address
PA1/KEY2	PA1	PAPU PAWU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	KEY2	PAS0	AN	—	Touch key input
PA2/CTPB/SDA/ICPCK	PA2	PAPU PAWU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	CTPB	PAS0	—	CMOS	CTM inverted output
	SDA	PAS0	ST	NMOS	I²C data line
	ICPCK	—	ST	—	ICP clock pin
PA3/KEY3	PA3	PAPU PAWU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	KEY3	PAS0	AN	—	Touch key input
PA4/KEY4	PA4	PAPU PAWU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	KEY4	PAS1	AN	—	Touch key input
PA5/KEY1	PA5	PAPU PAWU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	KEY1	PAS1	AN	—	Touch key input
PA6/CTCK/INT	PA6	PAPU PAWU	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	CTCK	IFS	ST	—	CTM clock input
	INT	IFS INTEG INTC0	ST	—	External Interrupt input
PA7/CTP	PA7	PAPU PAWU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	CTP	PAS1	—	CMOS	CTM output
VDD	VDD	—	PWR	—	Power supply
VSS	VSS	—	PWR	—	Ground
For 16-pin NSOP package type only					
OCDSDA	OCDSDA	—	ST	CMOS	OCDS data/address pin, for EV chip only
OCDSCK	OCDSCK	—	ST	—	OCDS clock pin, for EV chip only

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

Absolute Maximum Ratings

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

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

D.C. Electrical 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, can all exert an influence on the measured values. Note that the 2MHz or 4MHz operating frequency is used for the BS83B04L only.

Operating Voltage Characteristics

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
V_{DD}	Operating Voltage – HIRC	$f_{SYS}=f_{HIRC}$=2MHz	1.8	—	5.5	V
		$f_{SYS}=f_{HIRC}$=4MHz	1.8	—	5.5	
		$f_{SYS}=f_{HIRC}$=8MHz	1.8	—	5.5	
	Operating Voltage – LIRC	$f_{SYS}=f_{LIRC}$=2kHz	1.8	—	5.5	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	1.8V	f_{SYS}=2kHz	—	4	8	μA
		3V		—	5	10	
		5V		—	15	30	
	FAST Mode – HIRC	1.8V	f_{SYS}=2MHz	—	0.15	0.25	mA
		3V		—	0.2	0.3	
		5V		—	0.4	0.6	
		1.8V	f_{SYS}=4MHz	—	0.3	0.5	mA
		3V		—	0.4	0.6	
		5V		—	0.8	1.2	
		1.8V	f_{SYS}=8MHz	—	0.6	1.0	mA
		3V		—	0.8	1.2	
		5V		—	1.6	2.4	

Note: When using the characteristic table data, the following notes should be taken into consideration:
1. Any digital inputs are set 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

BS83A02L

Ta=25°C

Symbol	Standby Mode	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{STB}	SLEEP Mode	1.8V	WDT on	—	40	100	nA
		3V		—	60	120	
		5V		—	120	300	
	SLEEP Mode – 1 Key Wake-up	3V	WDT on, f_{SUB} on	—	150	250	nA
	IDLE0 Mode – LIRC	1.8V	f_{SUB} on	—	1.2	2.4	μA
		3V		—	1.5	3.0	
		5V		—	2.5	5.0	
	IDLE1 Mode – HIRC	1.8V	f_{SUB} on, f_{SYS}=8MHz	—	200	400	μA
		3V		—	300	500	
		5V		—	700	900	

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

1. Any digital inputs are set 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.

BS83B04L

Ta=25°C

Symbol	Standby Mode	Test Conditions		Min.	Typ.	Max	Unit
		V_{DD}	Conditions				
I_{STB}	SLEEP Mode	1.8V	WDT on	—	90	150	nA
		3V		—	90	150	
		5V		—	160	320	
	SLEEP Mode – 1 Key Wake-up	3V	WDT on, f_{SUB} on	—	150	250	nA
	IDLE0 Mode – LIRC	1.8V	f_{SUB} on	—	1.2	2.4	μA
		3V		—	1.5	3.0	
		5V		—	2.5	5.0	
	IDLE1 Mode – HIRC	1.8V	f_{SUB} on, f_{SYS}=2MHz	—	90	180	μA
		3V		—	100	200	
		5V		—	180	250	
		1.8V	f_{SUB} on, f_{SYS}=4MHz	—	100	200	μA
		3V		—	160	250	
		5V		—	300	600	
		1.8V	f_{SUB} on, f_{SYS}=8MHz	—	200	400	μA
		3V		—	300	500	
		5V		—	700	900	

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

1. Any digital inputs are set 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.

High Speed Internal Oscillator – HIRC – Frequency Accuracy

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

BS83A02L

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Temp.				
f_{HIRC}	8MHz Writer Trimmed HIRC Frequency	3V/5V	25°C	-1%	8	+1%	MHz
			-40°C~85°C	-2%	8	+2%	
		2.2V~5.5V	25°C	-2.5%	8	+2.5%	
			-40°C~85°C	-3%	8	+3%	
		1.8V~5.5V	25°C	-5%	8	+3%	
			-40°C~85°C	-10%	8	+5%	

Note: 1. The 3V/5V values for V_{DD} are provided as these are the two selectable fixed voltages at which the HIRC frequency is trimmed by the writer.
2. The row below the 3V/5V trim voltage row is provided to show the values for the full V_{DD} range operating voltage. It is recommended that the trim voltage is fixed at 3V for application voltage ranges from 1.8V to 3.6V and fixed at 5V for application voltage ranges from 3.3V to 5.5V.

BS83B04L

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Temp.				
f_{HIRC}	2MHz Writer Trimmed HIRC Frequency	3V/5V	25°C	-1%	2	+1%	MHz
			-40°C~85°C	-4%	2	+4%	
		2.2V~5.5V	25°C	-6%	2	+6%	
			-40°C~85°C	-7%	2	+7%	
		1.8V~5.5V	25°C	-10%	2	+10%	
			-40°C~85°C	-15%	2	+15%	
	4MHz Writer Trimmed HIRC Frequency	3V/5V	25°C	-1%	4	+1%	MHz
			-40°C~85°C	-3%	4	+2%	
		2.2V~5.5V	25°C	-2.5%	4	+2.5%	
			-40°C~85°C	-4%	4	+4%	
		1.8V~5.5V	25°C	-8%	4	+8%	
			-40°C~85°C	-13%	4	+13%	
	8MHz Writer Trimmed HIRC Frequency	3V/5V	25°C	-0.5%	8	+0.5%	MHz
			-40°C~85°C	-3%	8	+3%	
		2.2V~5.5V	25°C	-3%	8	+3%	
			-40°C~85°C	-4%	8	+4%	
		1.8V~5.5V	25°C	-8%	8	+8%	
			-40°C~85°C	-13%	8	+13%	

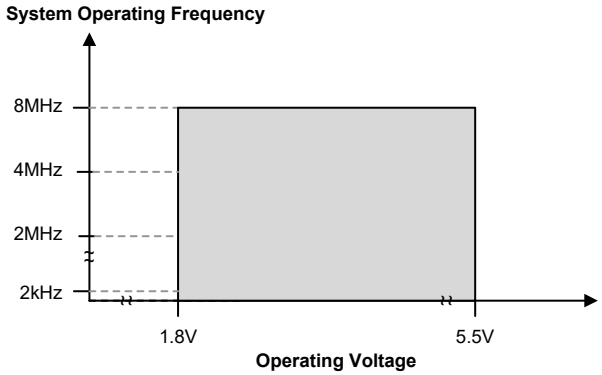
Note: 1. The 3V/5V values for V_{DD} are provided as these are the two selectable fixed voltages at which the HIRC frequency is trimmed by the writer.
2. The row below the 3V/5V trim voltage row is provided to show the values for the full V_{DD} range operating voltage. It is recommended that the trim voltage is fixed at 3V for application voltage ranges from 1.8V to 3.6V and fixed at 5V for application voltage ranges from 3.3V to 5.5V.

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

Low Speed Internal Oscillator Characteristics – LIRC

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Temp.				
f_{LIRC}	LIRC Frequency	3V	-40°C~85°C	-20%	2	+20%	kHz
		1.8V~5.5V	-40°C~85°C	-40%	2	+40%	
t_{START}	LIRC Start up Time	—	-40°C~85°C	—	—	100	μs

Operating Frequency Characteristic Curves



Note: The 2MHz or 4MHz operating frequency for the BS83B04L only.

System Start Up Time Characteristics

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
t_{SST}	System Start-up Time Wake-up from condition where f_{SYS} is off	—	$f_{SYS}=f_H\sim f_H/64$, $f_H=f_{HIRC}$	—	16	—	t_{HIRC}
			$f_{SYS}=f_{SUB}=f_{LIRC}$	—	2	—	t_{LIRC}
	System Start-up Time Wake-up from condition where f_{SYS} is on	—	$f_{SYS}=f_H\sim f_H/64$, $f_H=f_{HIRC}$	—	2	—	t_H
			$f_{SYS}=f_{SUB}=f_{LIRC}$	—	2	—	t_{SUB}
	System Speed Switch Time FAST to SLOW Mode or SLOW to FAST Mode	—	f_{HIRC} switches from off → on	—	16	—	t_{HIRC}
t_{RSTD}	System Reset Delay Time Reset Source from Power-on Reset or LVR Hardware Reset	—	RR_{POR}=5V/ms	42	48	54	ms
	System Reset Delay Time Reset Source from LVRC/WDTC/RSTC Software Reset (5)	—	—				
	System Reset Delay Time Reset Source from WDT Overflow	—	—	14	16	18	ms
t_{SRESET}	Minimum Software Reset 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}, etc., are the inverse of the corresponding frequency values as provided in the frequency tables. For example, $t_{HIRC}=1/f_{HIRC}$, etc.

3. If the LIRC is used as the system clock and if it is off when in the SLEEP Mode, then an additional LIRC start up time, t_{START}, as provided in the LIRC frequency table, must be added to the t_{SST} time in the table above.
4. The System Speed Switch Time is effectively the time taken for the newly activated oscillator to start up.
5. The LVRC software reset is used for the BS83A02L only while the RSTC software reset is used for the BS83B04L only.

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 or Input Pins	5V	—	0	—	1.5	V
		—	—	0	—	$0.2V_{DD}$	
V_{IH}	Input High Voltage for I/O Ports or Input Pins	5V	—	3.5	—	5.0	V
		—	—	$0.8V_{DD}$	—	V_{DD}	
I_{OL}	Sink Current for I/O Pins – BS83A02L	1.8V	$V_{OL}=0.1V_{DD}$	2.5	5.0	—	mA
		3V		5	10	—	
		5V		10	20	—	
	Sink Current for I/O Pins – BS83B04L	3V	$V_{OL}=0.1V_{DD}$	16	32	—	mA
		5V		32	65	—	
I_{OH}	Source Current for I/O Pins – BS83A02L	1.8V	$V_{OH}=0.9V_{DD}$	-1.5	-2.0	—	mA
		3V		-4	-5	—	
		5V		-8	-10	—	
	Source Current for I/O Pins – BS83B04L	3V	$V_{OH}=0.9V_{DD}$	-4	-8	—	mA
		5V		-8	-16	—	
I_{LEAK}	Input Leakage Current	5V	$V_{IN}=V_{DD}$ or $V_{IN}=V_{SS}$	—	—	±1	μA
R_{PH}	Pull-high Resistance for I/O Ports (Note)	3V	LVPU=0, For BS83A02L, PAPU=0FH; For BS83B04L, PAPU=FFH	20	60	100	kΩ
		5V		10	30	50	
		3V	LVPU=1, For BS83A02L, PAPU=0FH; For BS83B04L, PAPU=FFH	6.67	15.00	23.00	
		5V		3.5	7.5	12.0	
t_{TC}	TC Input Pin Minimum Pulse Width – BS83A02L	—	—	25	—	—	ns
t_{TCK}	TM TCK Input Pin Minimum Pulse Width – BS83B04L	—	—	0.3	—	—	μs
t_{INT}	External Interrupt Minimum Pulse Width	—	—	10	—	—	μs

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

Memory Characteristics

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

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{RW}	V_{DD} for Read / Write	—	—	V_{DDmin}	—	V_{DDmax}	V
Flash Program / EEPROM Memory (EEPROM is used for BS83B04L only)							
t_{DEW}	Erase / Write Cycle Time – Flash Program Memory	—	—	—	2	3	ms
	Write Cycle Time – Data EEPROM Memory	—	—	—	4	6	
I_{DDPGM}	Programming / Erase current on V_{DD}	—	—	—	—	5.0	mA
E_P	Cell Endurance – Flash Program Memory	—	—	10K	—	—	E/W
	Cell Endurance – Data EEPROM Memory	—	—	100K	—	—	
t_{RETD}	ROM Data Retention time	—	Ta=25°C	—	40	—	Year
RAM Data Memory							
V_{DR}	RAM Data Retention Voltage	—	Device in SLEEP Mode	1.0	—	—	V

LVR Electrical Characteristics

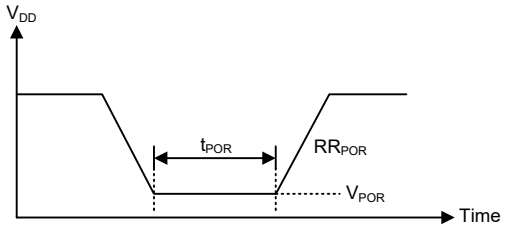
Ta=-40°C~85°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{LVR}	Low Voltage Reset Voltage	—	LVR enable, voltage is 1.7V	-5%	1.7	+5%	V
I_{LVR}	Operating Current	3V	LVR enable, V_{LVR}=1.7V	—	—	15	μA
		5V		—	15	25	
t_{LVR}	Minimum Low Voltage Width to Reset	—	—	125	750	1875	μs

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

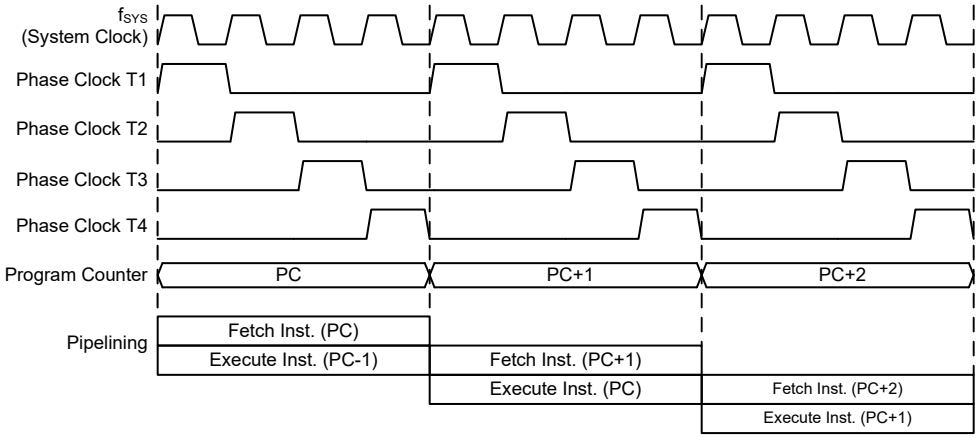


System Architecture

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

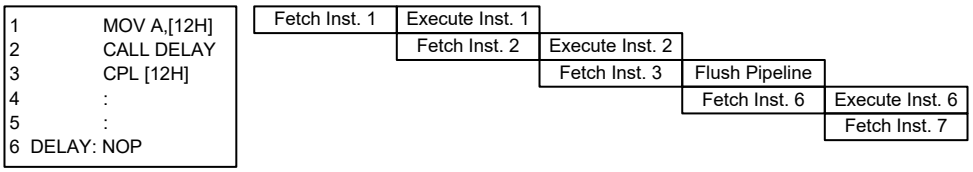
Clocking and Pipelining

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



System Clocking and Pipelining

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



Instruction Fetching

Program Counter

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

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

Device	Program Counter	
	High Byte	Low Byte (PCL)
BS83A02L	PC9~PC8	PCL7~PCL0
BS83B04L	PC10~PC8	PCL7~PCL0

Program Counter

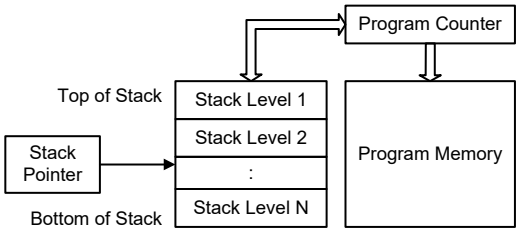
The lower byte of the Program Counter, known as the Program Counter Low register or PCL, is available for program control and is a readable and writeable register. By transferring data directly into this register, a short program jump can be executed directly; however, as only this low byte is available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

Stack

This is a special part of the memory which is used to save the contents of the Program Counter only. The stack is organized into up to 4 levels and neither part of the data nor part of the program space, and is neither readable nor writeable. The activated level is indexed by the Stack Pointer, and is neither readable nor writeable. At a subroutine call or interrupt acknowledge signal, the contents of the Program Counter are pushed onto the stack. At the end of a subroutine or an interrupt routine, signaled by a return instruction, RET or RETI, the Program Counter is restored to its previous value from the stack. After a device reset, the Stack Pointer will point to the top of the stack.

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

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



Note: N=2 for BS83A02L while N=4 for BS83B04L.

Arithmetic and Logic Unit – ALU

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

- Arithmetic operations:
  ADD, ADDM, ADC, ADCM, SUB, SUBM, SBC, SBCM, DAA
- Logic operations:
  AND, OR, XOR, ANDM, ORM, XORM, CPL, CPLA
- Rotation:
  RRA, RR, RRCA, RRC, RLA, RL, RLCA, RLC
- Increment and Decrement:
  INCA, INC, DECA, DEC
- Branch decision:
  JMP, SZ, SZA, SNZ, SIZ, SDZ, SIZA, SDZA, CALL, RET, RETI

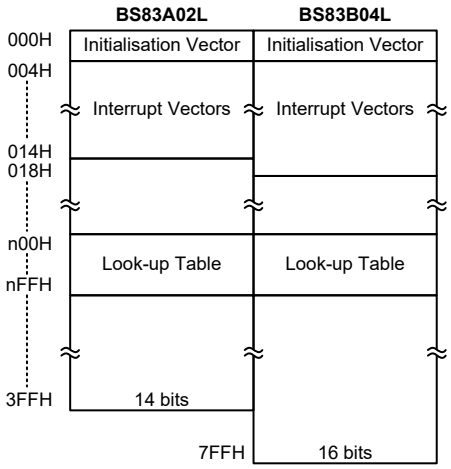
Flash Program Memory

The Program Memory is the location where the user code or program is stored. For 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.

Device	Capacity
BS83A02L	1K×14
BS83B04L	2K×16

Structure

The Program Memory has a capacity of 1K×14 to 2K×16 bits. The Program Memory is addressed by the Program Counter and also contains data, table information and interrupt entries. Table data, which can be setup in any location within the Program Memory, is addressed by a separate table pointer register.



Program Memory Structure

Special Vectors

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

Look-up Table

Any location within the Program Memory can be defined as a look-up table where programmers can store fixed data. To use the look-up table, the table pointer must first be setup by placing the address of the look up data to be retrieved in the table pointer register, TBLP and TBHP. These registers define the total address of the look-up table. The TBHP table pointer register is used for the BS83B04L only.

After setting up the table pointer, the table data can be retrieved from the Program Memory using the "TABRD[m]" or "TABRDL[m]" instructions, respectively. When the instruction is executed, the lower order table byte from the Program Memory will be transferred to the user defined Data Memory register [m] as specified in the instruction. The higher order table data byte from the Program Memory will be transferred to the TBLH special register. Any unused bits in this transferred higher order byte will be read as "0".

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

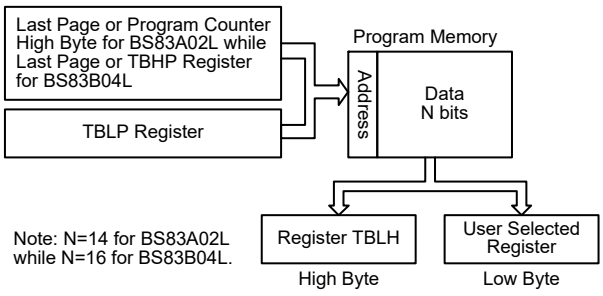


Table Program Example

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 "300H" which refers to the start address of the last page within the 1K Program Memory of the BS83A02L device. The table pointer is setup here to have an initial value of "06H". This will ensure that the first data read from the data table will be at the Program Memory address "306H" or 6 locations after the start of the last page. Note that the value for the table pointer is referenced to the first address of the present page if the "TABRD[m]" instruction is being used. 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.

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

Table Read Program Example

```
tempreg1 db ?          ; temporary register #1
tempreg2 db ?          ; temporary register #2
:
:
mov a,06h              ; initialise low table pointer - note that this address is referenced
mov tblp,a             ; to the last page or present page
:
:
tabrdl tempreg1        ; transfers value in table referenced by table pointer,
                       ; data at program memory address "306H" transferred to tempreg1
dec tblp               ; reduce value of table pointer by one
tabrdl tempreg2        ; transfers value in table referenced by table pointer,
                       ; data at program memory address "305H" transferred to tempreg2
                       ; in this example the data "1AH" is transferred to tempreg1 and
                       ; data "0FH" to register tempreg2
:
:
org 300h               ; sets initial address of program memory
dc 00Ah, 00Bh, 00Ch, 00Dh, 00Eh, 00Fh, 01Ah, 01Bh
```

In Circuit Programming – ICP

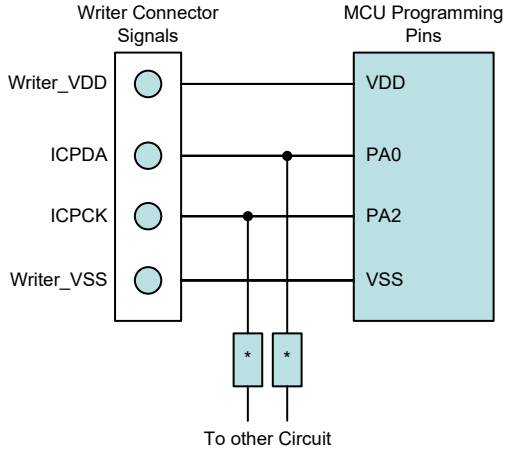
The provision of Flash type Program Memory provides the user with a means of convenient and easy upgrades and modifications to their programs on the same device.

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

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

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

There are EV chips named BS83AV02L and BS83BV04L which are used to emulate the real MCU devices named BS83A02L and BS83B04L respectively. The EV chip devices also provide an "On-Chip Debug" function to debug the real MCU devices during the development process. The EV chips and the real MCU devices are almost functionally compatible except for "On-Chip Debug" function and package type. Users can use the EV chip devices to emulate the real chip devices behavior by connecting the OCDSDA and OCDSCK pins to the Holtek HT-IDE development tools. The OCDSDA pin is the OCDS Data/Address input/output pin while the OCDSCK pin is the OCDS clock input pin. When users use the EV chip devices for debugging, the corresponding pin functions shared with the OCDSDA and OCDSCK pins in the real MCU devices will have no effect in the EV chip. However, the two OCDS pins which are pin-shared with the ICP programming pins are still used as the Flash Memory programming pins for ICP. For more detailed OCDS information, refer to the corresponding document named "Holtek e-Link for 8-bit MCU OCDS User's Guide".

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

Data Memory

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

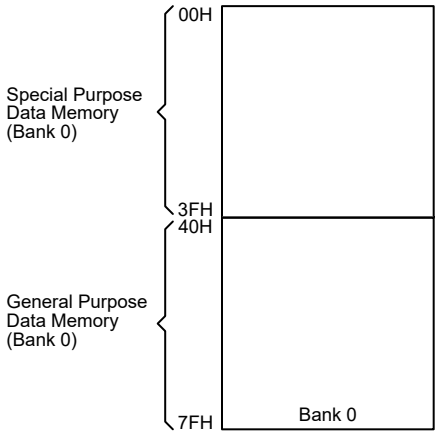
Categorized into two types, the first of these is an area of RAM where special function registers are located. These registers have fixed locations and are necessary for correct operation of the devices. Many of these registers can be read from and written to directly under program control, however, 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. For the BS83B04L, there is another area of Data Memory reserved for the Touch Key Data Memory.

Structure

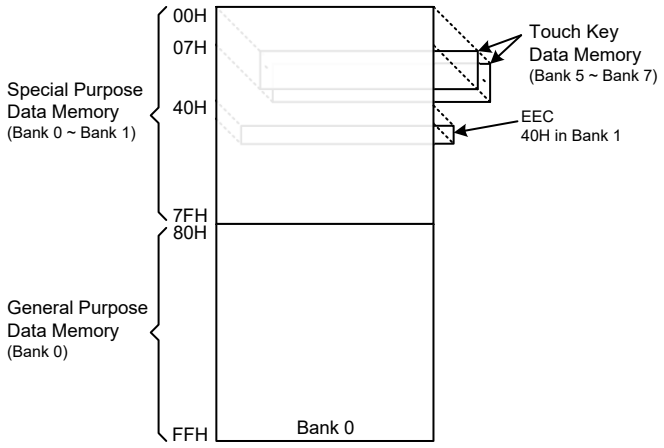
The overall Data Memory is subdivided into several banks. The Special Purpose Data Memory registers are accessible in bank 0, with the exception of the EEC register at address 40H, which is accessible in Bank 1. For the BS83B04L, the Touch Key Data Memory is located in Bank 5~Bank 7. Switching among the different Data Memory banks is achieved by setting the Bank Pointer to the correct value. The start address of the Data Memory for the devices is the address 00H.

Device	Special Purpose Data Memory	General Purpose Data Memory		Touch Key Data Memory	
	Available Banks	Capacity	Bank: Address	Capacity	Bank: Address
BS83A02L	Bank 0	64×8	Bank 0: 40H~7FH	—	—
BS83B04L	Bank 0 Bank 1: 40H (EEC only)	128×8	Bank 0: 80H~FFH	24×8	Bank 5: 00H~07H Bank 6: 00H~07H Bank 7: 00H~07H

Data Memory Summary



Data Memory Structure – BS83A02L



Data Memory Structure – BS83B04L

- **TDn Register – BS83A02L (n=0~5)**

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**: Temporary Data register
When the data memory capacity is not enough for applications, it can be temporarily used for data storage.

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

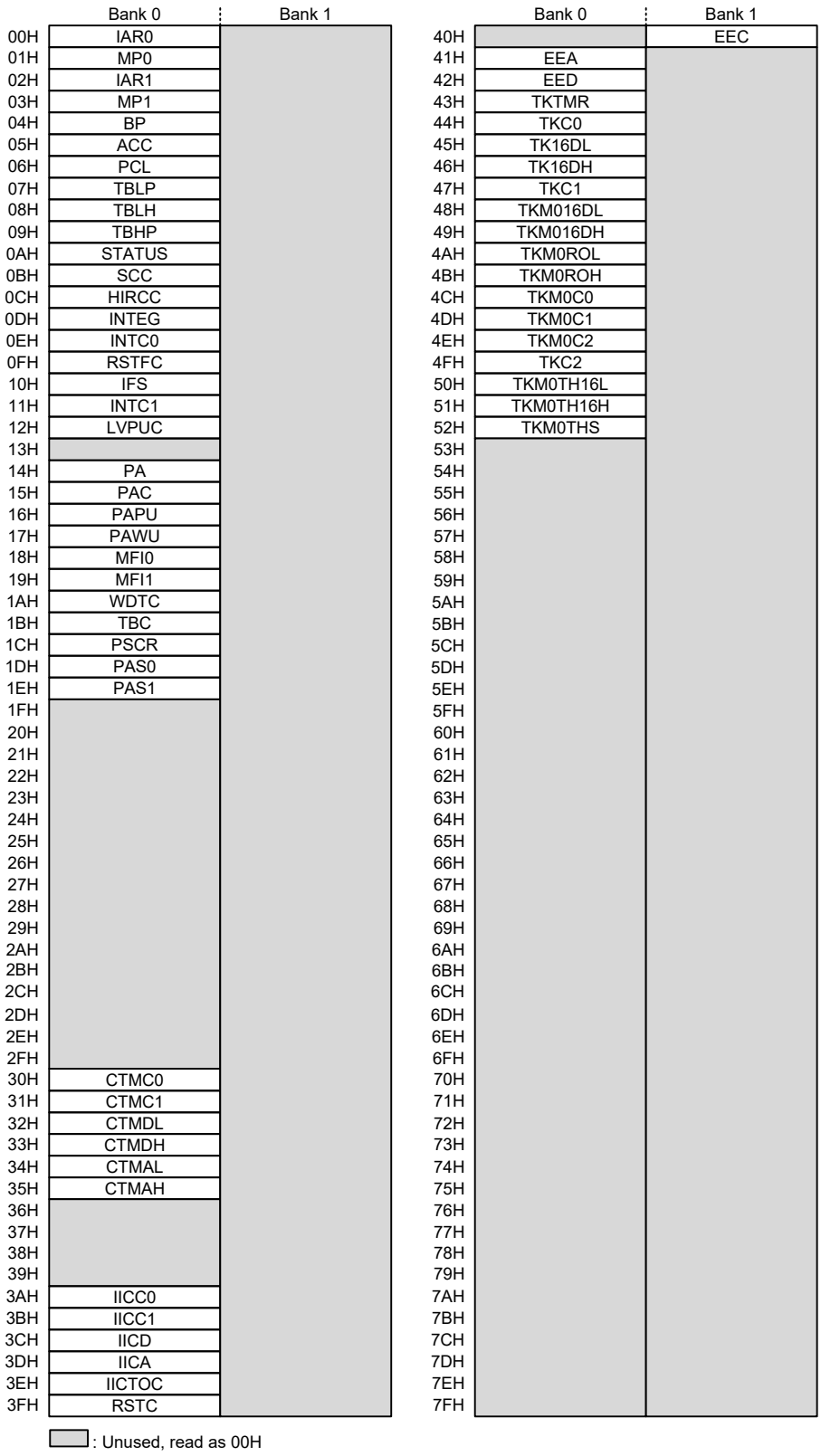
This area of Data Memory is where registers, necessary for the correct operation of the microcontroller, are stored. Most of the registers are both readable and writeable but some are protected and are readable only, the details of which are located under the relevant Special Function Register section.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

	Bank 0
00H	IAR0
01H	MP0
02H	
03H	
04H	
05H	ACC
06H	PCL
07H	TBLP
08H	TBLH
09H	
0AH	STATUS
0BH	SCC
0CH	HIRCC
0DH	
0EH	INTC0
0FH	RSTFC
10H	INTEG
11H	INTC1
12H	LVRC
13H	LVPUC
14H	PA
15H	PAC
16H	PAPU
17H	PAWU
18H	
19H	
1AH	WDTC
1BH	TBC
1CH	PSC0R
1DH	PAS0
1EH	TD0
1FH	TD1

	Bank 0
20H	TK1M0TH16L
21H	TK1M0TH16H
22H	TK2M0TH16L
23H	TK2M0TH16H
24H	TKM0K1ROCL
25H	TKM0K1ROCH
26H	TKM0K2ROCL
27H	TKM0K2ROCH
28H	TKM0K1CNTL
29H	TKM0K1CNTH
2AH	TKM0K2CNTL
2BH	TKM0K2CNTH
2CH	TD2
2DH	TD3
2EH	TD4
2FH	TMRC
30H	TMR
31H	TD5
32H	TKC2
33H	TKM0C2
34H	TKM0THS
35H	TKTMR
36H	TKC0
37H	TK16DL
38H	TK16DH
39H	TKC1
3AH	TKM016DL
3BH	TKM016DH
3CH	TKM0ROL
3DH	TKM0ROH
3EH	TKM0C0
3FH	TKM0C1

: Unused, read as 00H

Special Purpose Data Memory Structure – BS83A02L



Special Purpose Data Memory Structure – BS83B04L

Special Function Register Description

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

Indirect Addressing Registers – IAR0, IAR1

The Indirect Addressing Registers, IAR0 and IAR1, although having their locations in normal RAM register space, do not actually physically exist as normal registers. The method of indirect addressing for RAM data manipulation uses these Indirect Addressing Registers and Memory Pointers, in contrast to direct memory addressing, where the actual memory address is specified. Actions on the IAR0 and IAR1 registers will result in no actual read or write operation to these registers but rather to the memory location specified by their corresponding Memory Pointers, MP0 or MP1. Acting as a pair, IAR0 and MP0 can together access data from Bank 0 while the IAR1 and MP1 register pair can access data from any bank. As the Indirect Addressing Registers are not physically implemented, reading the Indirect Addressing Registers indirectly will return a result of "00H" and writing to the registers indirectly will result in no operation. Note that the IAR1 Indirect Addressing Register is used for the BS83B04L only.

Memory Pointers – MP0, MP1

Two Memory Pointers, known as MP0 and MP1 are provided. These Memory Pointers are physically implemented in the Data Memory and can be manipulated in the same way as normal registers providing a convenient way with which to address and track data. When any operation to the relevant Indirect Addressing Registers is carried out, the actual address that the microcontroller is directed to is the address specified by the related Memory Pointer. MP0, together with Indirect Addressing Register, IAR0, are used to access data from Bank 0, while MP1 and IAR1 are used to access data from all banks according to BP register. Direct Addressing can only be used with Bank 0, all other Banks must be addressed indirectly using MP1 and IAR1. Note that the MP1 Memory Pointer is used for the BS83B04L only.

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

Indirect Addressing Program Example

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

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

Bank Pointer – BP

For the BS83B04L, the Data Memory is divided into several banks. Selecting the required Data Memory area is achieved using the Bank Pointer.

The Data Memory is initialised to Bank 0 after a reset, except for a WDT time-out reset in the SLEEP or IDLE Mode, in which case, the Data Memory bank remains unaffected. Directly addressing the Data Memory will always result in Bank 0 being accessed irrespective of the value of the Bank Pointer. Accessing data from banks other than Bank 0 must be implemented using Indirect Addressing.

• **BP Register – BS83B04L**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	DMBP2	DMBP1	DMBP0
R/W	—	—	—	—	—	R/W	R/W	R/W
POR	—	—	—	—	—	0	0	0

Bit 7~3 Unimplemented, read as "0"

Bit 2~0 **DMBP2~DMBP0**: Data Memory Banks selection
000: Bank 0
001: Bank 1
010~100: Unimplemented
101: Bank 5
110: Bank 6
111: Bank 7

Accumulator – ACC

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

Program Counter Low Register – PCL

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

Look-up Table Registers – TBLP, TBHP, TBLH

These three special function registers are used to control operation of the look-up table which is stored in the Program Memory. TBLP and TBHP are the table pointers and indicate the location where the table data is located, note that the TBHP register is used for the BS83B04L only. Their values must be setup before any table read commands are executed. Their value can be changed, for example using the "INC" or "DEC" instructions, allowing for easy table data pointing and reading. TBLH is the location where the high order byte of the table data is stored after a table read data instruction has been executed. Note that the lower order table data byte is transferred to a user defined location.

Status Register – STATUS

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

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

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

- C is set if an operation results in a carry during an addition operation or if a borrow does not take place during a subtraction operation; otherwise C is cleared. C is also affected by a rotate through carry instruction.
- AC is set if an operation results in a carry out of the low nibbles in addition, or no borrow from the high nibble into the low nibble in subtraction; otherwise AC is cleared.
- Z is set if the result of an arithmetic or logical operation is zero; otherwise Z is cleared.
- OV is set if an operation results in a carry into the highest-order bit but not a carry out of the highest-order bit, or vice versa; otherwise OV is cleared.
- PDF is cleared by a system power-up or executing the "CLR WDT" instruction. PDF is set by executing the "HALT" instruction.
- TO is cleared by a system power-up or executing the "CLR WDT" or "HALT" instruction. TO is set by a WDT time-out.

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

• STATUS Register

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

"x": Unknown

Bit 7~6 Unimplemented, read as "0"

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

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

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

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

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

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

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

EEPROM Data Memory – BS83B04L

The BS83B04L contains an area of internal EEPROM Data Memory. EEPROM is by its nature a non-volatile form of re-programmable memory, with data retention even when its power supply is removed. By incorporating this kind of data memory, a whole new host of application possibilities are made available to the designer. The availability of EEPROM storage allows information such as product identification numbers, calibration values, specific user data, system setup data or other product information to be stored directly within the product microcontroller. The process of reading and writing data to the EEPROM memory has been reduced to a very trivial affair.

Capacity	Address
32×8	00H~1FH

EEPROM Data Memory Structure

The EEPROM Data Memory capacity varies from 32×8 bits, according to the device selected. Unlike the Program Memory and RAM Data Memory, the EEPROM Data Memory is not directly mapped into memory space and is therefore not directly addressable in the same way as the other types of memory. Read and Write operations to the EEPROM are carried out in single byte operations using an address and a data register in Bank 0 and a single control register in Bank 1.

EEPROM Registers

Three registers control the overall operation of the internal EEPROM Data Memory. These are the address register, EEA, the data register, EED and a single control register, EEC. As both the EEA and EED registers are located in Bank 0, they can be directly accessed in the same way as any other Special Function Register. The EEC register however, being located in Bank 1, can be read from or written to indirectly using the MP1 Memory Pointer and Indirect Addressing Register, IAR1. Because the EEC control register is located at address 40H in Bank 1, the MP1 Memory Pointer must first be set to the value 40H and the Bank Pointer register, BP, set to the value, 01H, before any operations on the EEC register are executed.

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

EEPROM Register List

• **EEA Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	EEA4	EEA3	EEA2	EEA1	EEA0
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 **EEA4~EEA0**: Data EEPROM address bit 4 ~ bit 0

• **EED Register**

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

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

• **EEC Register**

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

Bit 7~4 Unimplemented, read as "0"

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

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

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

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

Note: 1. The WREN, WR, RDEN and RD cannot be set high at the same time in one instruction. The WR and RD cannot be set high at the same time.
2. Ensure that the f_{32K} clock is stable before executing the write operation.

3. Ensure that the write operation is totally complete before changing the EEC register content.

Reading Data from the EEPROM

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

Writing Data to the EEPROM

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

Write Protection

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

EEPROM Interrupt

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

Programming Considerations

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

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

Programming Examples

Reading data from the EEPROM – polling method

```
MOV A, EEPROM_ADRES       ; user defined address
MOV EEA, A
MOV A, 040H               ; setup memory pointer MP1
MOV MP1, A                ; MP1 points to EEC register
MOV A, 01H                ; setup Bank Pointer
MOV BP, A
SET IAR1.1                ; set RDEN bit, enable read operations
SET IAR1.0                ; start Read Cycle - set RD bit
BACK:
SZ  IAR1.0                ; check for read cycle end
JMP BACK
CLR IAR1                  ; disable EEPROM read if no more read operations are required
CLR BP
MOV A, EED                ; move read data to register
MOV READ_DATA, A
```

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

Writing Data to the EEPROM – polling method

```
MOV A, EEPROM_ADRES       ; user defined address
MOV EEA, A
MOV A, EEPROM_DATA        ; user defined data
MOV EED, A
MOV A, 040H               ; setup memory pointer MP1
MOV MP1, A                ; MP1 points to EEC register
MOV A, 01H                ; setup Bank Pointer
MOV BP, A
CLR EMI
SET IAR1.3                ; set WREN bit, enable write operations
SET IAR1.2                ; start Write Cycle - set WR bit - executed immediately
                          ; after set WREN bit
SET EMI
BACK:
SZ  IAR1.2                ; check for write cycle end
JMP BACK
CLR BP
```

Oscillators

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

Oscillator Overview

In addition to being the source of the main system clock the oscillators also provide clock sources for the Watchdog Timer and Time Base Interrupts. Two fully integrated internal oscillators, requiring no external components, are provided to form a wide range of both fast and slow system oscillators. The higher frequency oscillator provides 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, the devices have the flexibility to optimize the performance/power ratio, a feature especially important in power sensitive portable applications.

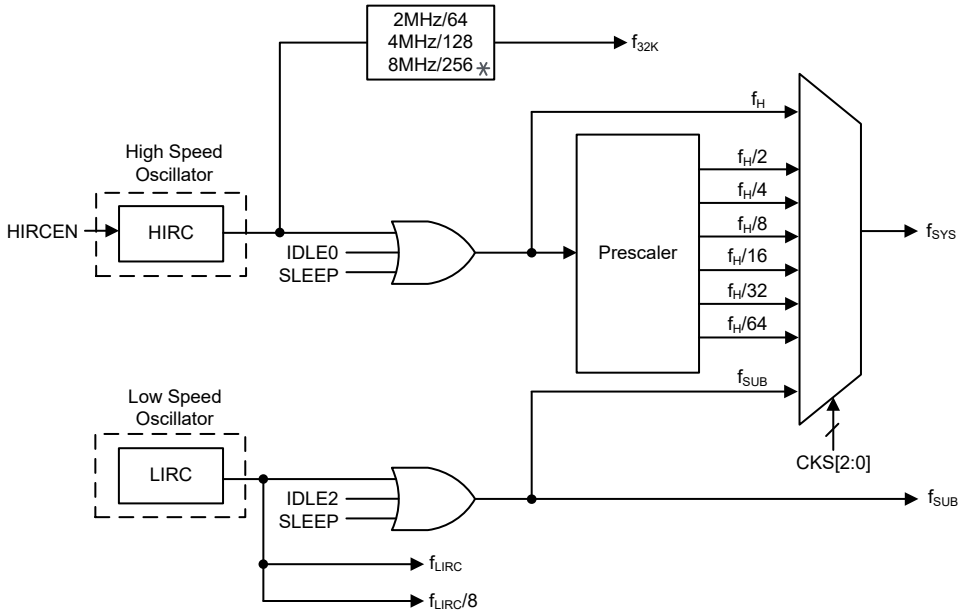
Device	Type	Name	Frequency
BS83A02L	Internal High Speed RC	HIRC	8MHz
	Internal Low Speed RC	LIRC	2kHz
BS83B04L	Internal High Speed RC	HIRC	2/4/8MHz
	Internal Low Speed RC	LIRC	2kHz

Oscillator Types

System Clock Configurations

There are two methods of generating the system clock, one high speed oscillator and one low speed oscillator. For the BS83A02L, the high speed oscillator is the internal 8MHz RC oscillator, HIRC. For the BS83B04L, the high speed oscillator is the internal 2/4/8MHz RC oscillator. The low speed oscillator is the internal 2kHz 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 frequency of the slow speed or high speed system clock is determined using the CKS2~CKS0 bits in the SCC register. Note that two oscillator selections must be made namely one high speed and one low speed system oscillator. It is not possible to choose a no-oscillator selection for either the high or low speed oscillator.



*: 2MHz/64 or 4MHz/128 for BS83B04L only.

System Clock Configurations

Internal High Speed RC Oscillator – HIRC

The internal RC oscillator is a fully integrated system oscillator requiring no external components. For the BS83A02L, the internal RC oscillator has a fixed frequency of 8MHz. For the BS83B04L, the internal RC oscillator has three fixed frequencies of 2MHz, 4MHz and 8MHz, which is selected using a configuration option. The HIRC1~HIRC0 bits in the HIRCC register must also be setup to match the selected configuration option frequency. Setting up these bits is necessary to ensure that the HIRC frequency accuracy specified in the A.C. Characteristics is achieved. Device trimming during the manufacturing process and the inclusion of internal frequency compensation circuits are used to ensure that the influence of the power supply voltage, temperature and process variations on the oscillation frequency are minimised.

Internal 2kHz Oscillator – LIRC

The Internal 2kHz System Oscillator is the low frequency oscillator. It is a fully integrated RC oscillator with a typical frequency of 2kHz at full voltage range, 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.

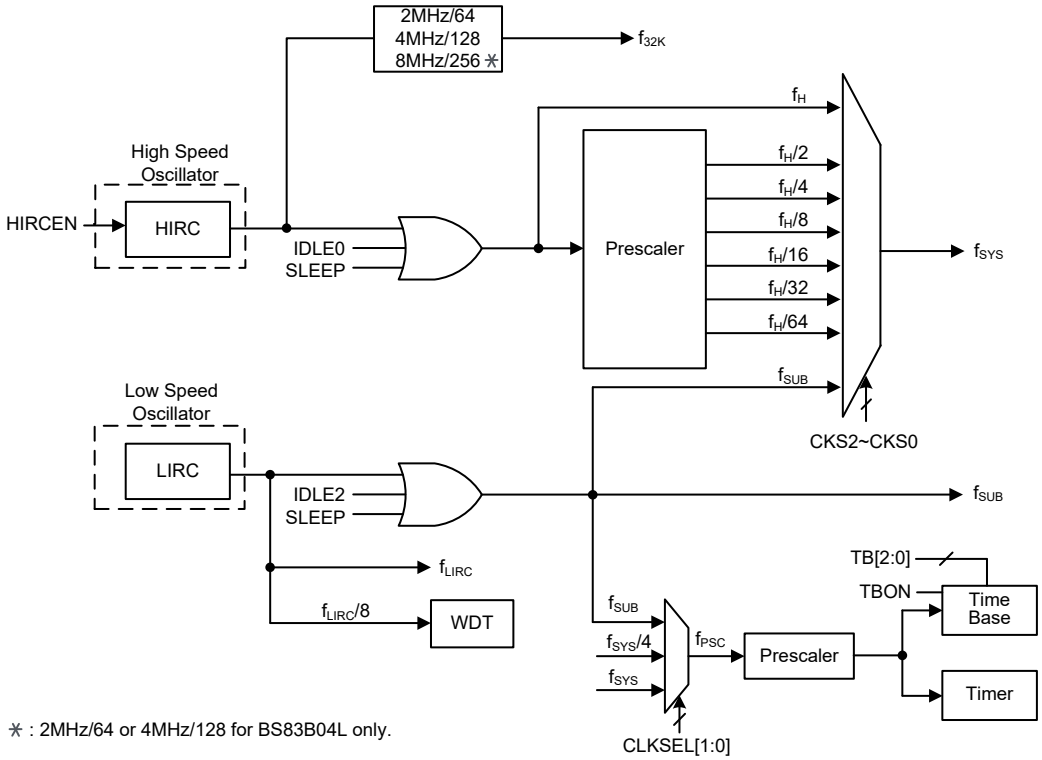
Operating Modes and System Clocks

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

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

The main system clock, can come from either a high frequency, f_H, or low frequency, f_{SUB}, source, and is selected using the CKS2~CKS0 bits in the SCC register. The high speed system clock is sourced from the HIRC oscillator. The low speed system clock source can be sourced from the 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: 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.

System Operation Modes

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

Operation Mode	CPU	Register Setting			f_{SYS}	f_H	f_{SUB}	f_{LIRC}	$f_{LIRC}/8$
		FHIDEN	FSIDEN	CKS2~CKS0					
FAST	On	x	x	000~110	f_H~$f_H/64$	On	On	On	On
SLOW	On	x	x	111	f_{SUB}	On/Off(1)	On	On	On
IDLE0	Off	0	1	000~110	Off	Off	On	On	On
				111	On				
IDLE1	Off	1	1	xxx	On	On	On	On	On
IDLE2	Off	1	0	000~110	On	On	Off	On	On
				111	Off				
SLEEP	Off	0	0	xxx	Off	Off	Off	On/Off(2)	On(3)

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 by configuring the touch key function M0TSS bit being enabled or disabled in the SLEEP mode.
3. The $f_{LIRC}/8$ clock will be switched on since the WDT function is always enabled in the SLEEP mode.

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. This mode operates allowing the microcontroller to operate normally with a clock source will come from HIRC oscillator. The high speed oscillator will however first be divided by a ratio ranging from 1 to 64, the actual ratio being selected by the CKS2~CKS0 bits in the SCC register. Although a high speed oscillator is used, running the microcontroller at a divided clock ratio reduces the operating current.

SLOW Mode

This is also a mode where the microcontroller operates normally although now with a slower speed clock source. The clock source used will be from f_{SUB}. The f_{SUB} clock is derived from the LIRC oscillator. Running the microcontroller in this mode allows it to run with much lower operating currents.

SLEEP Mode

The SLEEP Mode is entered when a HALT instruction is executed and when the FHIDEN and FSIDEN bit are 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 $f_{LIRC}/8$ clock still continue to operate since the WDT function is always enabled.

IDLE0 Mode

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

IDLE1 Mode

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

IDLE2 Mode

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

Control Registers

The registers, SCC and HIRCC, are used to control the internal clocks within the devices.

Register Name	Bit							
	7	6	5	4	3	2	1	0
SCC	CKS2	CKS1	CKS0	—	—	—	FHIDEN	FSIDEN
HIRCC (BS83A02L)	—	—	—	—	—	—	HIRCF	HIRCEN
HIRCC (BS83B04L)	—	—	—	—	HIRC1	HIRC0	HIRCF	HIRCEN

System Operating Mode Control Register List

- **SCC Register**

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

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

Bit 4~2 Unimplemented, read as "0"

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

• **HIRCC Register – BS83A02L**

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

• **HIRCC Register – BS83B04L**

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

Bit 7~4 Unimplemented, read as "0"

Bit 3~2 **HIRC1~HIRC0**: HIRC frequency selection
00: 2MHz
01: 4MHz
10: 8MHz
11: 2MHz
It is recommended that the HIRC frequency selected by these two bits should be the same with the frequency determined by the configuration option to achieve the HIRC frequency accuracy specified in the A.C. characteristics.

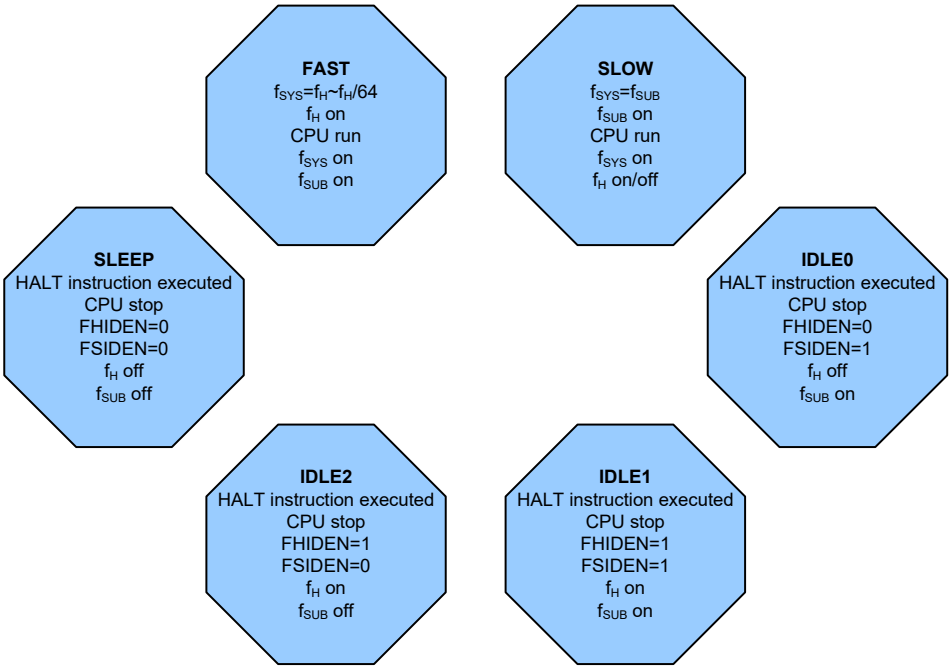
Bit 1 **HIRCF**: HIRC oscillator stable flag
0: HIRC unstable
1: HIRC stable
This bit is used to indicate whether the HIRC oscillator is stable or not. When the HIRCEN bit is set to 1 to enable the HIRC oscillator, the HIRCF bit will first be cleared to 0 and then set to 1 after the HIRC oscillator is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

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

Operating Mode Switching

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

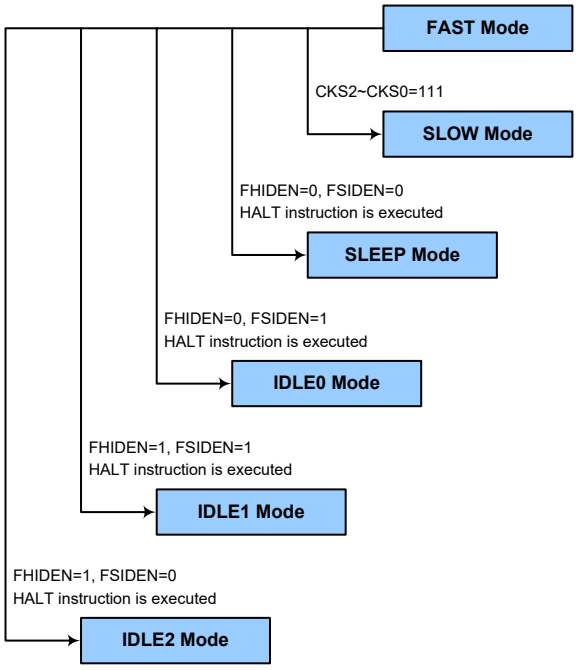
In simple terms, Mode Switching between the FAST Mode and SLOW Mode is executed using the CKS2~CKS0 bits in the SCC register while Mode Switching from the FAST/SLOW Modes to the IDLE/SLEEP Modes is executed via the HALT instruction. When a HALT instruction is executed, whether the device enters the IDLE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



FAST Mode to SLOW Mode Switching

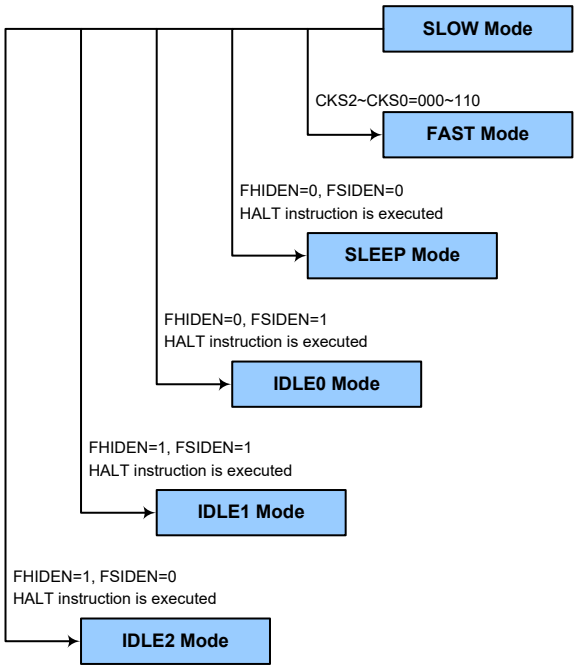
When running in the FAST Mode, which uses the high speed system oscillator, and therefore consumes more power, the system clock can switch to run in the SLOW Mode by set the CKS2~CKS0 bits to "111" in the SCC register. This will then use the low speed system oscillator which will consume less power.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

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

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

Entering the IDLE0 Mode

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

- The f_H clock will be stopped and the application program will stop at the "HALT" instruction, but

the f_{SUB} clock will be on.

- The Data Memory contents and registers will maintain their present condition.
- The I/O ports will maintain their present conditions.
- In the status register, the Power Down flag, PDF, will be set and the Watchdog time-out flag, TO, will be cleared.
- The WDT will be cleared and resume counting as the WDT function is always enabled.

Entering the IDLE1 Mode

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

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

Entering the IDLE2 Mode

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

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

Standby Current Considerations

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

When the devices execute the "HALT" instruction, the PDF flag will be set to 1. The PDF flag will be cleared to 0 if the 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 wake-up occurs, the program will resume execution at the instruction following the "HALT" instruction. If the system is woken up by an interrupt, then two possible situations may occur. The first is where the related interrupt is disabled or the interrupt is enabled but the stack is full, in which case the program will resume execution at the instruction following the "HALT" instruction. In this situation, the interrupt which woke up the 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 an interrupt request flag is set high before entering the SLEEP or IDLE Mode, the wake-up function of the related interrupt will be disabled.

Watchdog Timer

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

Watchdog Timer Clock Source

The Watchdog Timer clock source is provided by the internal clock, f_{WDT}, which is supplied by the LIRC oscillator with the output frequency of $f_{LIRC}/8$. The LIRC internal oscillator has an approximate frequency of 2kHz 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 2^5 to 2^{11} to give longer timeouts, the actual value being chosen using the WS2~WS0 bits in the WDTC register.

Watchdog Timer Control Register

A single register, WDTC, controls the required timeout period as well as the enable and reset 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
01010/10101: Enable
Other values: Reset MCU
When these bits are changed to any other values 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.

Bit 2~0 **WS2~WS0**: WDT time-out period selection
000: $2^5/f_{WDT}$
001: $2^5/f_{WDT}$
010: $2^6/f_{WDT}$
011: $2^7/f_{WDT}$
100: $2^8/f_{WDT}$
101: $2^9/f_{WDT}$
110: $2^{10}/f_{WDT}$
111: $2^{11}/f_{WDT}$
These three bits determine the division ratio of the Watchdog Timer source clock, which in turn determines the timeout period. The internal clock, f_{WDT} is supplied by the LIRC oscillator with the output frequency of $f_{LIRC}/8$.

• RSTFC Register – BS83A02L

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

"x": Unknown

Bit 7~3 Unimplemented, read as "0"

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 high by the WDT Control register software reset and cleared to zero by the application program. Note that this bit can only be cleared to zero by the application program.

• **RSTFC Register – BS83B04L**

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

"x": Unknown

Bit 7~4 Unimplemented, read as "0"

Bit 3 **RSTF**: Reset control register software reset flag
Describe elsewhere.

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

Bit 1 Unimplemented, read as "0"

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 the device. There are five bits, WE4~WE0, in the WDTC register to offer the enable and reset control of the Watchdog Timer. The WDT function will be enabled if the WE4~WE0 bits are equal to 01010B or 10101B. If the WE4~WE0 bits are set to any other values, other than 01010B and 10101B, it will reset the device after a delay time, t_{SRESET}. After power on these bits will have a value of 01010B.

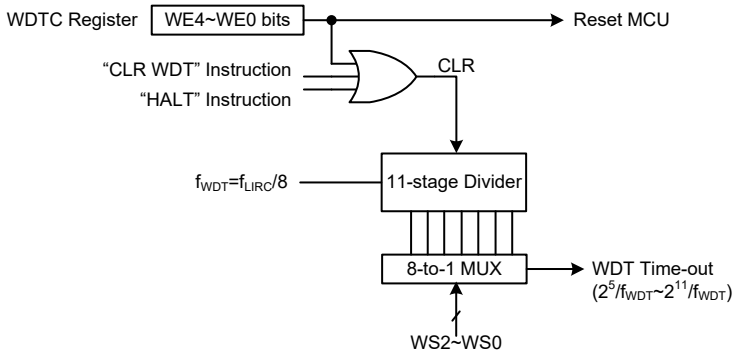
WE4~WE0 Bits	WDT Function
01010B/10101B	Enable
Any other value	Reset MCU

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

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

The maximum time out period is when the 2^{11} division ratio is selected. As an example, with a 250Hz f_{WDT} as its source clock, this will give a maximum watchdog period of around 8s for the 2^{11} division ratio and a minimum timeout of 128ms for the 2^5 division ration.



Watchdog Timer

Reset and Initialisation

A reset function is a fundamental part of any microcontroller ensuring that the 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.

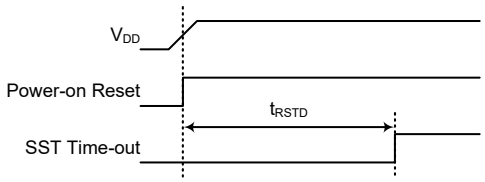
In addition to the power-on reset, another reset exists in the form of a Low Voltage Reset, LVR, where a full reset is implemented in situations where the power supply voltage falls below a certain threshold. Another type of reset is when the Watchdog Timer overflows and resets the microcontroller. All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

Internal Reset Control – BS83B04L

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

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

Internal Reset Function Control

- **RSTC Register – BS83B04L**

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 – BS83B04L**

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

"x": Unknown

Bit 7~4 Unimplemented, read as "0"

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

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

Bit 1 Unimplemented, read as "0"

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

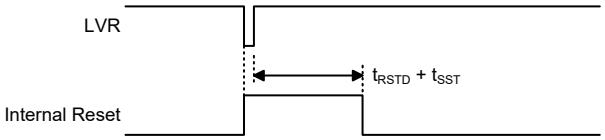
Low Voltage Reset – LVR

The microcontroller contains a low voltage reset circuit in order to monitor the supply voltage of the device and provides an MCU reset should the value fall below a certain predefined level. If the supply voltage of the device drops to within a range of 0.9V~V_{LVR} such as might occur when changing the battery in battery powered applications, the LVR will automatically reset the device internally and the LVRF bit in the RSTFC register will also be set high. For a valid LVR signal, a low supply voltage, i.e., a voltage in the range between 0.9V~V_{LVR} must exist for a time greater than that specified by t_{LVR} in the LVR 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.

For the BS83A02L, the LVR function can be controlled by the LVRC register. If the LVS7~LVS0 bits are set to 01011010B, the LVR function is enabled with a fixed LVR voltage of 1.7V. If the LVS7~LVS0 bits are set to 10100101B, the LVR function is disabled. If the LVS7~LVS0 bits are changed to some different values by environmental noise, the LVR will reset the device after a delay time, t_{SRESET}. When this happens, the LRF bit in the RSTFC register will be set high. After power on the register will have the value of 01011010B.

For the BS83B04L, the LVR function is always enabled in the FAST or SLOW mode with a specific LVR voltage V_{LVR}, which is fixed at 1.7V.

Note that the LVR function will be automatically disabled when the device enters the IDLE or SLEEP mode.



Low Voltage Reset Timing Chart

- **LVRC Register – BS83A02L**

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	1	0	1	0

Bit 7~0 **LVS7~LVS0**: LVR voltage select control
01011010: 1.7V
10100101: Disable
Any other value: Generates MCU reset – register is reset to POR value

When an actual low voltage condition occurs, as specified 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 01011010B and 10100101B, 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 – BS83A02L**

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

"x": Unknown

Bit 7~3 Unimplemented, read as "0"

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

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

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

• **RSTFC Register – BS83B04L**

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

"x": Unknown

Bit 7~4 Unimplemented, read as "0"

Bit 3 **RSTF**: Reset control register software reset flag
Describe elsewhere.

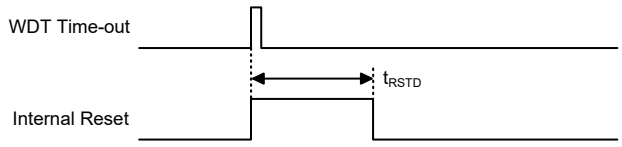
Bit 2 **LVRF**: LVR function reset flag
0: Not occur
1: Occurred
This bit is set to 1 when a specific Low Voltage Reset situation condition occurs. This bit can only be cleared to 0 by the application program.

Bit 1 Unimplemented, read as "0"

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

Watchdog Time-out Reset during Normal Operation

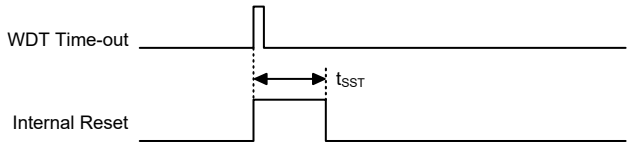
When a Watchdog time-out Reset during normal operations in the FAST or SLOW mode, the Watchdog time-out flag TO will be set to "1".



WDT Time-out Reset during Normal Operation Timing Chart

Watchdog Time-out Reset during SLEEP or IDLE Mode

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



WDT Time-out Reset during SLEEP or IDLE Timing Chart

Reset Initial Conditions

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

TO	PDF	Reset Conditions
0	0	Power-on reset
u	u	LVR reset during 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 Base	Clear after reset, WDT begins counting
Timers	Timers will be turned off
Input/Output Ports	I/O ports will be setup as inputs
Stack Pointer	Stack Pointer will point to the top of the stack

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

Register	BS83A02L	BS83B04L	Power On Reset	LVR Reset (Normal Operation)	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)
IAR0	●	●	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
MP0	●		0xxx xxxx	0uuu uuuu	0uuu uuuu	0uuu uuuu
		●	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
IAR1		●	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
MP1		●	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
BP		●	---- -000	---- -000	---- -000	---- -000

Register	BS83A02L	BS83B04L	Power On Reset	LVR Reset (Normal Operation)	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)
ACC	●	●	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
PCL	●	●	0000 0000	0000 0000	0000 0000	0000 0000
TBLP	●	●	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLH	●		--xx xxxx	--uu uuuu	--uu uuuu	--uu uuuu
		●	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBHP		●	---- -xxx	---- -uuu	---- -uuu	---- -uuu
STATUS	●	●	--00 xxxx	--uu uuuu	--1u uuuu	--11 uuuu
SCC	●	●	000- --00	000- --00	000- --00	uuu- --uu
HIRCC	●		---- --01	---- --01	---- --01	---- --uu
		●	---- 0001	---- 0001	---- 0001	---- uuuu
INTC0	●	●	-000 0000	-000 0000	-000 0000	-uuu uuuu
RSTFC	●		---- -x00	---- -1uu	---- -uuu	---- -uuu
		●	---- 0x-0	---- u1-u	---- uu-u	---- uu-u
INTEG	●	●	---- --00	---- --00	---- --00	---- --uu
IFS		●	---- --00	---- --00	---- --00	---- --uu
INTC1	●		--00 --00	--00 --00	--00 --00	--uu --uu
		●	-000 -000	-000 -000	-000 -000	-uuu -uuu
LVRC	●		0101 1010	uuuu uuuu	0101 1010	uuuu uuuu
LVPUC	●	●	---- ---0	---- ---0	---- ---0	---- ---u
PA	●		---- 1111	---- 1111	---- 1111	---- uuuu
		●	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAC	●		---- 1111	---- 1111	---- 1111	---- uuuu
		●	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAPU	●		---- 0000	---- 0000	---- 0000	---- uuuu
		●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PAWU	●		---- 0000	---- 0000	---- 0000	---- uuuu
		●	0000 0000	0000 0000	0000 0000	uuuu uuuu
MFI0		●	--00 --00	--00 --00	--00 --00	--uu --uu
MFI1		●	--00 --00	--00 --00	--00 --00	--uu --uu
WDTC	●	●	0101 0100	0101 0100	0101 0100	uuuu uuuu
TBC	●	●	0--- -000	0--- -000	0--- -000	u--- -uuu
PSCR	●	●	---- --00	---- --00	---- --00	---- --uu
PAS0	●		00-- 00--	00-- 00--	00-- 00--	uu-- uu--
		●	0000 0000	0000 0000	0000 0000	uuuu uuuu
PAS1		●	00-- 0000	00-- 0000	00-- 0000	uu-- uuuu
CTMC0		●	0000 0000	0000 0000	0000 0000	uuuu uuuu
CTMC1		●	0000 0000	0000 0000	0000 0000	uuuu uuuu
CTMDL		●	0000 0000	0000 0000	0000 0000	uuuu uuuu
CTMDH		●	---- --00	---- --00	---- --00	---- --uu
CTMAL		●	0000 0000	0000 0000	0000 0000	uuuu uuuu

Register	BS83A02L	BS83B04L	Power On Reset	LVR Reset (Normal Operation)	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)
CTMAH		●	---- --00	---- --00	---- --00	---- --uu
IICC0		●	---- 000-	---- 000-	---- 000-	---- uuu-
IICC1		●	1000 0001	1000 0001	1000 0001	uuuu uuuu
IICD		●	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
IICA		●	0000 000-	0000 000-	0000 000-	uuuu uuu-
IICTOC		●	0000 0000	0000 0000	0000 0000	uuuu uuuu
RSTC		●	0101 0101	0101 0101	0101 0101	uuuu uuuu
EEA		●	---0 0000	---0 0000	---0 0000	---u uuuu
EED		●	0000 0000	0000 0000	0000 0000	uuuu uuuu
EEC		●	---- 0000	---- 0000	---- 0000	---- 0000
TK1M0TH16L	●		0000 0000	0000 0000	0000 0000	uuuu uuuu
TK1M0TH16H	●		0000 0000	0000 0000	0000 0000	uuuu uuuu
TK2M0TH16L	●		0000 0000	0000 0000	0000 0000	uuuu uuuu
TK2M0TH16H	●		0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM0K1ROCL	●		0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM0K1ROCH	●		---- --00	---- --00	---- --00	---- --uu
TKM0K2ROCL	●		0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM0K2ROCH	●		---- --00	---- --00	---- --00	---- --uu
TKM0K1CNTL	●		0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM0K1CNTH	●		0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM0K2CNTL	●		0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM0K2CNTH	●		0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM0TH16L		●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM0TH16H		●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TMRC	●		00-0 1000	00-0 1000	00-0 1000	uu-u uuuu
TMR	●		0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM0THS	●		--00 --00	--00 --00	--00 --00	--uu --uu
		●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM0C2	●		---- -1-0	---- -1-0	---- -1-0	---- -u-u
		●	1110 0100	1110 0100	1110 0100	uuuu uuuu
TKC2	●	●	---- -001	---- -001	---- -001	---- -uuu
TKTMR	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKC0	●		-000 0010	-000 0010	-000 0010	-uuu uuuu
		●	0000 0010	0000 0010	0000 0010	uuuu uuuu
TK16DL	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TK16DH	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKC1	●	●	000- 0011	000- 0011	000- 0011	Uuu- uuuu
TKM016DL	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM016DH	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM0ROL	●	●	0000 0000	0000 0000	0000 0000	uuuu uuuu

Register	BS83A02L	BS83B04L	Power On Reset	LVR Reset (Normal Operation)	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)
TKM0ROH	•	•	---- --00	---- --00	---- --00	---- --uu
TKM0C0	•	•	--0- 0000	--0- 0000	--0- 0000	--u- uuuu
TKM0C1	•	•	0-00 --00	0-00 --00	0-00 --00	u-uu --uu
TD0	•		0000 0000	0000 0000	0000 0000	uuuu uuuu
TD1	•		0000 0000	0000 0000	0000 0000	uuuu uuuu
TD2	•		0000 0000	0000 0000	0000 0000	uuuu uuuu
TD3	•		0000 0000	0000 0000	0000 0000	uuuu uuuu
TD4	•		0000 0000	0000 0000	0000 0000	uuuu uuuu
TD5	•		0000 0000	0000 0000	0000 0000	uuuu uuuu

Note: “u” stands for unchanged
“x” stands for unknown
“-” stands for unimplemented

Input/Output Ports

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

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

Device	Register Name	Bit							
		7	6	5	4	3	2	1	0
BS83A02L	PA	—	—	—	—	PA3	PA2	PA1	PA0
	PAC	—	—	—	—	PAC3	PAC2	PAC1	PAC0
	PAPU	—	—	—	—	PAPU3	PAPU2	PAPU1	PAPU0
	PAWU	—	—	—	—	PAWU3	PAWU2	PAWU1	PAWU0
BS83B04L	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
BS83A02L/ BS83B04L	LVPUC	—	—	—	—	—	—	—	LVPU

“—”: Unimplemented, read as “0”

I/O Logic Function Register List

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 the relevant pull-high control registers PAPU and LVPUC and are implemented using weak PMOS transistors. The PAPU registers is used to determine whether the pull-high function is enabled or not while the LVPUC register is used to select the pull-high resistors value for low voltage power supply applications.

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.

Note that the LVPU bit in the LVPUC register is only available when the corresponding pin pull-high function is enabled by setting the relevant pull-high control bit high. This bit will have no effect when the pull-high function is disabled.

• PAPU Register – BS83A02L

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	PAPU3	PAPU2	PAPU1	PAPU0
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 **PAPU3~PAPU0**: PA3~PA0 pull-high function control
0: Disable
1: Enable

• PAPU Register – BS83B04L

Bit	7	6	5	4	3	2	1	0
Name	PAPU7	PAPU6	PAPU5	PAPU4	PAPU3	PAPU2	PAPU1	PAPU0
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 **PAPU7~PAPU0**: PA7~PA0 pull-high function control
0: Disable
1: Enable

• LVPUC Register

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

Bit 7~1 Unimplemented, read as "0"

Bit 0 **LVPU**: Pull-high resistor selection when low voltage power supply
0: All pin pull high resistor is 60kΩ @ 3V
1: All pin pull high resistor is 15kΩ @ 3V

Port A Wake-up

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

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

• **PAWU Register – BS83A02L**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	PAWU3	PAWU2	PAWU1	PAWU0
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 **PAWU3~PAWU0**: PA3~PA0 wake-up function control
0: Disable
1: Enable

• **PAWU Register – BS83B04L**

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**: PA7~PA0 wake-up function control
0: Disable
1: Enable

I/O Port Control Registers

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

• **PAC Register – BS83A02L**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	PAC3	PAC2	PAC1	PAC0
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	1	1	1	1

Bit 7~4 Unimplemented, read as "0"

Bit 3~0 **PAC3~PAC0**: PA3~PA0 Pin type selection
0: Output
1: Input

• **PAC Register – BS83B04L**

Bit	7	6	5	4	3	2	1	0
Name	PAC7	PAC6	PAC5	PAC4	PAC3	PAC2	PAC1	PAC0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	1	1	1	1	1	1	1	1

Bit 7~0 **PAC7~PAC0**: PA7~PA0 Pin type selection
0: Output
1: Input

Pin-shared Functions

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

Pin-shared Function Selection Registers

The limited number of supplied pins in a package can impose restrictions on the amount of functions a certain device can contain. However, by allowing the same pins to share several different functions and providing a means of function selection, a wide range of different functions can be incorporated into even relatively small package sizes. The device includes Port A Output Function Selection register “n”, labeled as PASn, 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 the digital input pin, such as INT, CTCK, TC etc., which shares the same pin-shared control configuration with its corresponding general purpose I/O function when setting the relevant pin-shared control bit. To select this pin function, in addition to the necessary pin-shared control and peripheral functional setup aforementioned, it must also be set 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.

Device	Register Name	Bit							
		7	6	5	4	3	2	1	0
BS83A02L	PAS0	PAS07	PAS06	—	—	PAS03	PAS02	—	—
BS83B04L	PAS0	PAS07	PAS06	PAS05	PAS04	PAS03	PAS02	PAS01	PAS00
	PAS1	PAS17	PAS16	—	—	PAS13	PAS12	PAS11	PAS10
	IFS	—	—	—	—	—	—	IFS1	IFS0

Pin-shared Function Selection Register List

• **PAS0 Register – BS83A02L**

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

Bit 7~6 **PAS07~PAS06**: PA3 Pin-Shared function selection
00/01/10: PA3
11: KEY2

Bit 5~4 Unimplemented, read as "0"

Bit 3~2 **PAS03~PAS02**: PA1 Pin-Shared function selection
00/01/10: PA1
11: KEY1

Bit 1~0 Unimplemented, read as "0"

• **PAS0 Register – BS83B04L**

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
00/01/10: PA3
11: KEY3

Bit 5~4 **PAS05~PAS04**: PA2 Pin-Shared function selection
00/01: PA2
10: CTPB
11: SDA

Bit 3~2 **PAS03~PAS02**: PA1 Pin-Shared function selection
00/01/10: PA1
11: KEY2

Bit 1~0 **PAS01~PAS00**: PA0 Pin-Shared function selection
00/01/10: PA0/CTCK/INT
11: SCL

• **PAS1 Register – BS83B04L**

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

Bit 7~6 **PAS17~PAS16**: PA7 Pin-Shared function selection
00/01/10: PA7
11: CTP

Bit 5~4 Unimplemented, read as "0"

Bit 3~2 **PAS13~PAS12**: PA5 Pin-Shared function selection
00/01/10: PA5
11: KEY1

Bit 1~0 **PAS11~PAS10**: PA4 Pin-Shared function selection
00/01/10: PA4
11: KEY4

• **IFS Register – BS83B04L**

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

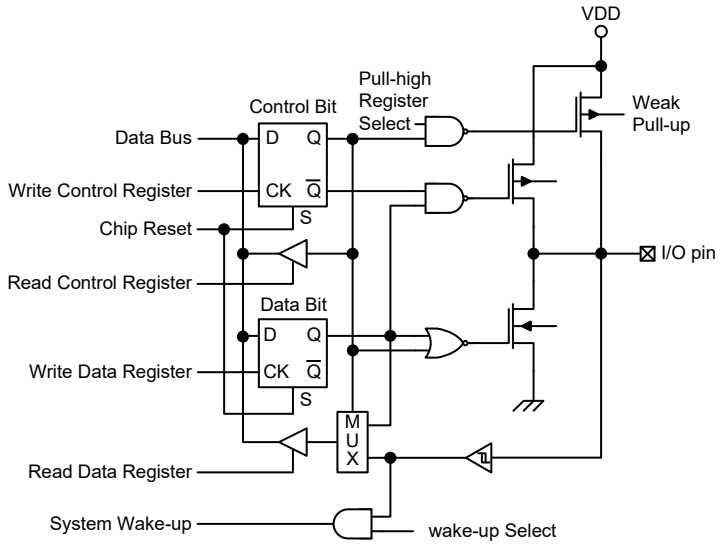
Bit 7~2 Unimplemented, read as "0"

Bit 1 **IFS1**: CTCK input source pin selection
0: PA6
1: PA0

Bit 0 **IFS0**: INT input source pin selection
0: PA0
1: PA6

I/O Pin Structures

The accompanying diagram illustrates the internal structure 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 I/O logic function. The wide range of pin-shared structures does not permit all types to be shown.



Logic Function Input/Output Structure

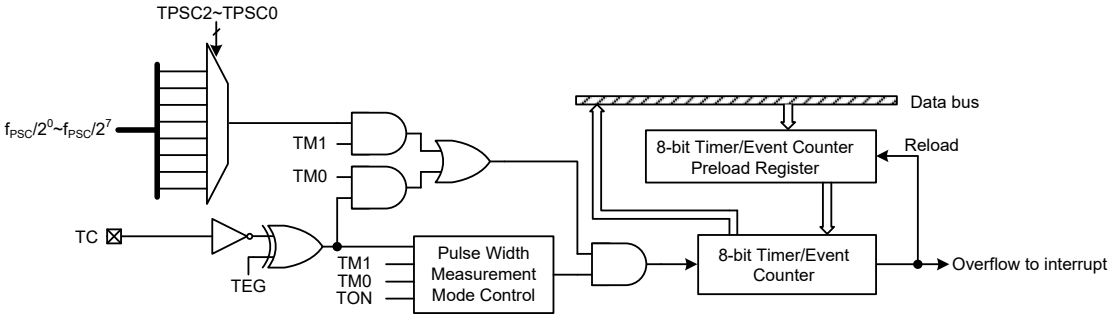
Programming Considerations

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

Port A has the additional capability of providing wake-up functions. When the devices are in the SLEEP or IDLE Mode, various methods are available to wake the 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/Event Counter – BS83A02L

The provision of the Timer/Event Counter forms an important part of any microcontroller, giving the designer a means of carrying out time related functions. The BS83A02L contains an 8-bit Timer/Event Counter, which contains an 8-bit programmable count-up counter and the clock may come from an external or internal clock source. As the timer has three different operating modes, it can be configured to operate as a general timer, an external event counter or a pulse width measurement device.



Note: The Timer/Event Counter internal clocks are sourced from a prescaler, which is the f_{PSC} clock divided by a ratio of 2^0 to 2^7. Refer to the Time Base Interrupt section for details on the f_{PSC} clock.

8-bit Timer/Event Counter

Timer/Event Counter Input Clock Source

The Timer/Event Counter clock source can originate from various sources, an internal clock or an external pin. The internal clock source is used when the timer is in the Timer Mode and Pulse Width Measurement Mode. For the Timer/Event Counter, this internal clock source is the divided version of f_{PSC}, which is selected by the TPSC2~TPSC0 bits in the TMRC Timer/Event Control Register.

An external clock source is used when the Timer/Event Counter is in the Event Counter Mode, the clock source is provided on the external TC pin. Depending upon the condition of the TEG bit, each high to low or low to high transition on the external timer pin will increase the counter by one.

Timer/Event Counter Registers

There are two registers related to the Timer/Event Counter. The first is the TMR register that contains the actual value of the timer and into which an initial value can be preloaded. Writing to the TMR register will transfer the specified data to the Timer/Event Counter. Reading the TMR register will read the contents of the Timer/Event Counter. The second is the TMRC control register, which is used to define the operating mode, control the counting enable or disable and select the active edge.

Register Name	Bit							
	7	6	5	4	3	2	1	0
TMRC	TM1	TM0	—	TON	TEG	TPSC2	TPSC1	TPSC0
TMR	D7	D6	D5	D4	D3	D2	D1	D0

Timer/Event Counter Register List

Timer Register – TMR

The timer register TMR is the place where the actual timer value is stored. The value in the timer register increases by one each time an internal clock pulse is received or an external transition occurs on the external timer pin. The timer will count from the initial value loaded by the preload register to the full count of FFH for the 8-bit Timer/Event Counter, at which point the timer overflows and an internal interrupt signal is generated. The timer value will then be loaded with the preload register value and continue counting.

Note that to achieve a maximum full range count of FFH, the preload register must first be cleared. Note that if the Timer/Event Counter is in an off condition and data is written to its preload register, this data will be immediately written into the actual counter. However, if the counter is enabled and counting, any new data written into the preload data register during this period will remain in the preload register and will only be written into the actual counter until an overflow occurs.

- **TMR 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**: Timer preload register data byte

Timer Control Register – TMRC

The flexible features of the Holtek microcontroller Timer/Event Counter are implemented by operating in three different modes, the options of which are determined by the contents of control register bits.

The Timer Control Register is known as TMRC. It is the Timer Control Register together with its corresponding timer register that controls the full operation of the Timer/Event Counter. Before the timer can be used, it is essential that the Timer Control Register is fully programmed with the right data to ensure its correct operation, a process that is normally carried out during program initialisation.

To select which of the three modes the timer is to operate in, namely the Timer Mode, the Event Counter Mode or the Pulse Width Measurement Mode, the TM1~TM0 bits in the Timer Control Register must be set to the required logic levels. The timer-on bit TON provides the basic on/off control of the respective timer. Setting the bit to high allows the counter to run. Clearing the bit stops the counter. When the internal clock is used, it can be selected by properly setting the TPSC2~TPSC0 bits. The internal clock selection will have no effect if an external clock source is used. If the timer is in the Event Counter or Pulse Width Measurement Mode, the active transition edge type is selected by the logic level of the TEG bit in the Timer Control Register.

- **TMRC Register**

Bit	7	6	5	4	3	2	1	0
Name	TM1	TM0	—	TON	TEG	TPSC2	TPSC1	TPSC0
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 7~6 **TM1~TM0**: Timer/Event Counter operating mode selection
00: Unused
01: Event Counter Mode
10: Timer Mode
11: Pulse Width Measurement Mode

Bit 5 Unimplemented, read as "0"

Bit 4 **TON**: Timer/Event Counter counting enable control
0: Disable
1: Enable

Bit 3 **TEG**: Timer/Event Counter active edge selection
Event Counter Mode
0: Count on rising edge
1: Count on falling edge
Pulse Width Measurement Mode
0: Start counting on falling edge, stop on rising edge
1: Start counting on rising edge, stop on falling edge

Bit 2~0 **TPSC2~TPSC0**: Timer internal clock selection
000: f_{PSC}
001: $f_{PSC}/2$
010: $f_{PSC}/2^2$
011: $f_{PSC}/2^3$
100: $f_{PSC}/2^4$
101: $f_{PSC}/2^5$
110: $f_{PSC}/2^6$
111: $f_{PSC}/2^7$

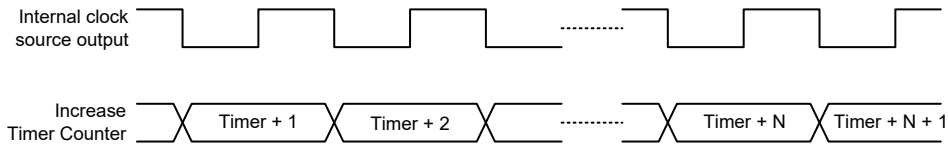
Timer/Event Counter Operating Modes

The Timer/Event Counter can operate in one of three operating modes, Timer Mode, Event Counter Mode or Pulse Width Measurement Mode. The operating mode is selected using the TM1 and TM0 bits in the TMRC register.

Timer Mode

To select this mode, bits TM1 and TM0 in the TMRC register should be set to "10" respectively. In this mode, the Timer/Event Counter can be utilised to measure fixed time intervals, providing an internal interrupt signal each time the Timer/Event Counter overflows.

When operating in this mode the internal clock is used as the timer clock. The internal clock source is from the Prescaler, and is selected by the TPSC2~TPSC0 bits in the TMRC register. The timer-on bit TON must be set high to enable the timer to run. Each time an internal clock high to low transition occurs, the timer increases by one. When the timer reaches its maximum 8-bit, FFH Hex, value and overflows, an interrupt signal is generated and the timer will reload the value already loaded into the preload register and continue counting. It should be noted that in the Timer mode, even if the device is in the IDLE/SLEEP mode, if the selected internal clock is still activated and a timer overflow occurs, it will generate a timer interrupt and corresponding wake-up source.



Timer Mode Timing Chart

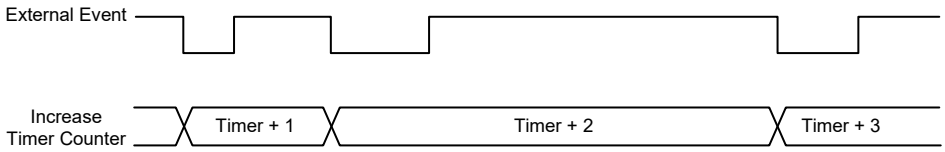
Event Counter Mode

To select this mode, bits TM1 and TM0 in the TMRC register should be set to "01" respectively. In this mode, a number of externally changing logic events, occurring on the external timer TC pin, can be recorded by the Timer/Event Counter.

When operating in this mode, the external timer pin, TC, is used as the Timer/Event Counter clock

source. After the other bits in the Timer Control Register have been properly configured, the enable bit TON, can be set high to enable the Timer/Event Counter. If the Active Edge Selection bit, TEG, is low, the Timer/Event Counter will increase each time the TC pin receives a low to high transition. If the TEG bit is high, the counter will increase each time the TC pin receives a high to low transition. When it is full and overflows, an interrupt signal is generated and the Timer/Event Counter will reload the value already loaded into the preload register and continue counting.

The TC pin must be set as an input by setting the corresponding bit in the port control register. It should be noted that in the Event Counter mode, even if the device is in the IDLE/SLEEP Mode, the Timer/Event Counter will continue to record externally changing logic events on the TC pin. As a result, when the timer overflows it will generate a timer interrupt and corresponding wake-up source.



Event Counter Mode Timing Chart (TEG=1)

Pulse Width Measurement Mode

To select this mode, bits TM1 and TM0 in the TMRC register should be set to "11" respectively. In this mode, the Timer/Event Counter can be utilised to measure the width of external pulses applied to the external timer pin.

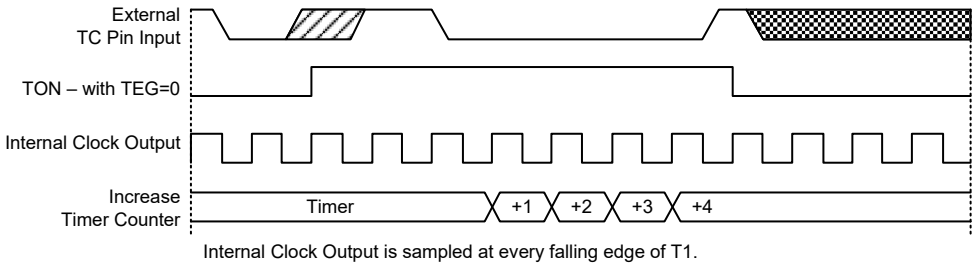
When operating in this mode the internal clock is used as the timer clock. The internal clock source is from the Prescaler, the division ratio is determined by the TPSC2~TPSC0 bits in the TMRC register. After the other bits in the Timer Control Register have been properly configured, the enable bit TON, can be set high to enable the Timer/Event Counter, however it will not actually start counting until an active edge is received on the TC pin.

If the active Edge Selection bit TEG is low, once a high to low transition has been received on the TC pin, the Timer/Event Counter will start counting based on the internal selected clock source until the TC pin returns to its original high level. At this point the enable bit will be automatically reset to zero and the Timer/Event Counter will stop counting. If the Active Edge Selection bit is high, the Timer/Event Counter will begin counting once a low to high transition has been received on the TC pin and stop counting when the external timer pin returns to its original low level. As before, the enable bit will then be automatically reset to zero. It is important to note that in the pulse width measurement Mode, the enable bit is automatically reset to zero when the external control signal on the TC pin returns to its original level, whereas in the other two modes the enable bit can only be reset to zero under application program control.

The residual value in the Timer/Event Counter, which can now be read by the program, therefore represents the length of the pulse received on the TC pin. As the enable bit has now been reset, any further transitions on the external timer pin will be ignored. The timer cannot begin further pulse width measurement until the enable bit is set high again by the application program. In this way, single shot pulse measurements can be easily made. It should be noted that in this mode the Timer/Event Counter is controlled by logical transitions on the external timer pin and not by the logic level. When the Timer/Event Counter is full and overflows, an interrupt signal is generated and the Timer/Event Counter will reload the value already loaded into the preload register and continue counting.

The TC pin must be set as an input by setting the corresponding bit in the port control register. It should be noted that in the Pulse Width Measurement mode, even if the device is in the IDLE/SLEEP Mode, the Timer/Event Counter will continue to increase if the internal clock source is still

activated and the required logical transitions occur on the external TC pin. As a result, when the timer overflows it will generate a timer interrupt and corresponding wake-up source.



Pulse Width Measurement Mode Timing Chart (TEG=0)

Programming Considerations

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

When the Timer/Event Counter is read, or if data is written to the preload register, the clock is inhibited to avoid errors, however as this may result in a counting error, it should be taken into account by the programmer. Care must be taken to ensure that the timers are properly initialised before using them for the first time. The associated timer interrupt enable bit in the interrupt control register must be properly set otherwise the internal interrupt associated with the timer will remain inactive. The active edge selection, timer operating mode selection and internal clock selection bits in timer control register must also be correctly issued to ensure the timer is properly configured for the required applications. It is also important to ensure that a desired initial value is first loaded into the timer register before the timer is switched on. After the timer has been initialised the timer can be turned on and off by controlling the enable bit in the timer control register.

When the Timer/Event Counter overflows, its corresponding interrupt request flag in the interrupt control register will be set to generate an interrupt signal. If the Timer/Event Counter interrupt is enabled, this will in turn allow program branch to its interrupt vector. However irrespective of whether the interrupt is enabled or not, a Timer/Event Counter overflow will also generate a wake-up signal if the device is in the IDLE/SLEEP mode. This situation may occur if the Timer/Event Counter internal clock source is still activated or if the external signal continues to change state. In such cases, the Timer/Event Counter will continue to count and if an overflow occurs the device will be woken up. To prevent such a wake-up from occurring, the timer interrupt request flag should first be set high before issuing the “HALT” instruction to enter the IDLE/SLEEP mode.

Timer Module (TM) – BS83B04L

One of the most fundamental functions in any microcontroller device is the ability to control and measure time. To implement time related functions the BS83B04L includes a Timer Module, abbreviated to the name TM. The TM is multi-purpose timing units and serve to provide operations such as Timer/Counter, Compare Match Output as well as being the functional unit for the generation of PWM signals. The TM has two individual interrupts. The addition of input and output pins for each TM ensures that users are provided with timing units with a wide and flexible range of features.

The general features of the Compact type TM are described here with more detailed information provided in the individual Compact type TM section.

Introduction

The device contains a Compact type TM. The main features of the CTM are summarised in the accompanying table.

Function	CTM
Timer/Counter	√
Compare Match Output	√
PWM Output	√
PWM Alignment	Edge
PWM Adjustment Period & Duty	Duty or Period

CTM Function Summary

TM Operation

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

TM Clock Source

The clock source which drives the main counter in each TM can originate from various sources. The selection of the required clock source is implemented using the CTCK2~CTCK0 bits in the CTM control registers. 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 CTCK pin. The CTCK pin clock source is used to allow an external signal to drive the TM as an external clock source or for event counting.

TM Interrupts

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

TM External Pins

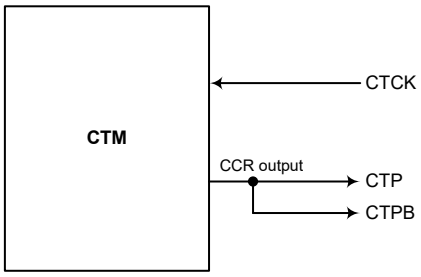
The Compact type TM has one TM input pin, with the label CTCK. The CTM input pin, CTCK, is essentially a clock source for the CTM and is selected using the CTCK2~CTCK0 bits in the CTMC0 register. This external TM input pin allows an external clock source to drive the internal TM. The

CTCK input pin can be chosen to have either a rising or falling active edge.

The TM has two output pins with the label CTP and CTPB. 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 CTP and CTPB output pins are also the pins where the TM generates the PWM output waveform. As the TM input and output pins are pin-shared with other functions, the TM input and output function must first be setup using relevant pin-shared function selection register described in the Pin-shared Function section.

CTM	
Input	Output
CTCK	CTP, CTPB

CTM External Pins

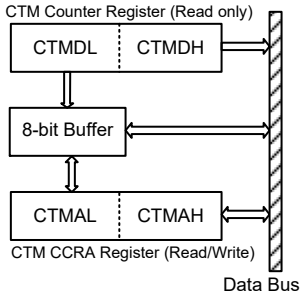


CTM Function Pin Block Diagram

Programming Considerations

The TM Counter Registers and the Capture/Compare CCRA 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 registers are implemented in the way shown in the following diagram and accessing this register pair is carried out in a specific way described above, it is recommended to use the "MOV" instruction to access the CCRA low byte register, named CTMAL, in the following access procedures. Accessing the CCRA low byte register without following these access procedures will result in unpredictable values.



The following steps show the read and write procedures:

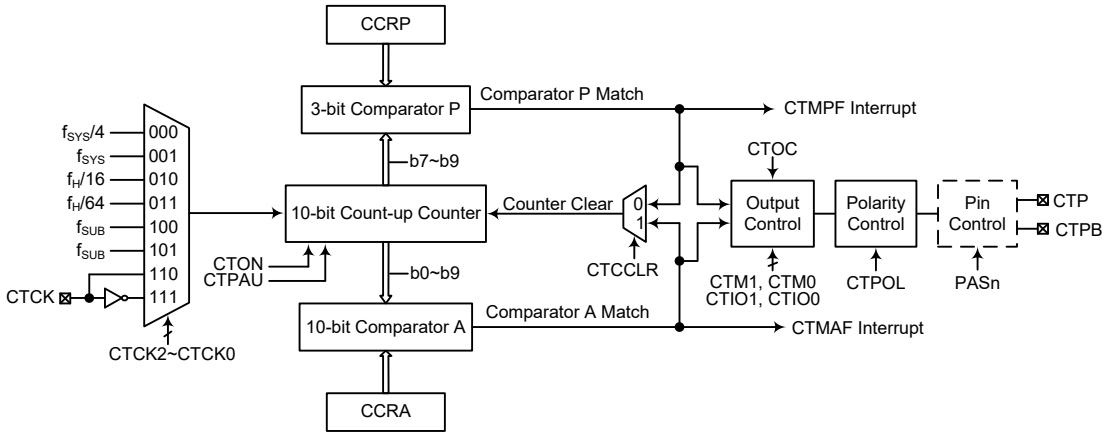
- Writing Data to CCRA
  - Step 1. Write data to Low Byte CTMAL
    - Note that here data is only written to the 8-bit buffer.

- Step 2. Write data to High Byte CTMAH
  - 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
  - Step 1. Read data from the High Byte CTMDH, CTMAH
    - 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 CTMDL, CTMAL
    - This step reads data from the 8-bit buffer.

Compact Type (CTM) – BS83B04L

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



Note: CTPB is the inverted output of CTP.

Compact Type TM Block Diagram

Compact TM Operation

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

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

Compact Type TM Register Description

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

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

10-bit Compact TM Register List

- **CTMC0 Register**

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

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

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

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

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

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

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

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

Bit 2~0 **CTRP2~CTRP0**: CTM CCRP 3-bit register, compared with the CTM Counter bit 9~bit 7
Comparator P Match Period:
000: 1024 CTM clocks
001: 128 CTM clocks
010: 256 CTM clocks
011: 384 CTM clocks
100: 512 CTM clocks
101: 640 CTM clocks
110: 768 CTM clocks
111: 896 CTM clocks

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

• CTMC1 Register

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

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

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

Bit 5~4 **CTIO1~CTIO0**: Select CTM external pin, CTP, function
Compare Match Output Mode
00: No change
01: Output low
10: Output high
11: Toggle output
PWM Output Mode
00: PWM output inactive state
01: PWM output active state
10: PWM output
11: Undefined
Timer/counter Mode
Unused

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

In the Compare Match Output Mode, the CTIO1~CTIO0 bits determine how the CTM output pin changes state when a compare match occurs from the Comparator A. The CTM output pin can be setup to switch high, switch low or to toggle its present state

when a compare match occurs from the Comparator A. When the CTIO1~CTIO0 bits are both zero, then no change will take place on the output. The initial value of the CTM output pin should be setup using the CTOC bit. Note that the output level requested by the CTIO1~CTIO0 bits must be different from the initial value setup using the CTOC bit otherwise no change will occur on the CTM output pin when a compare match occurs. After the CTM output pin changes state it can be reset to its initial level by changing the level of the CTON bit from low to high.

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

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

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

PWM Output Mode
0: Active low
1: Active high

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

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

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

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

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

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

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

• **CTMDL Register**

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

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

• **CTMDH Register**

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

Bit 7~2 Unimplemented, read as "0"

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

• **CTMAL Register**

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

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

• **CTMAH Register**

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

Bit 7~2 Unimplemented, read as "0"

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

Compact Type TM Operating Modes

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

Compare Match Output Mode

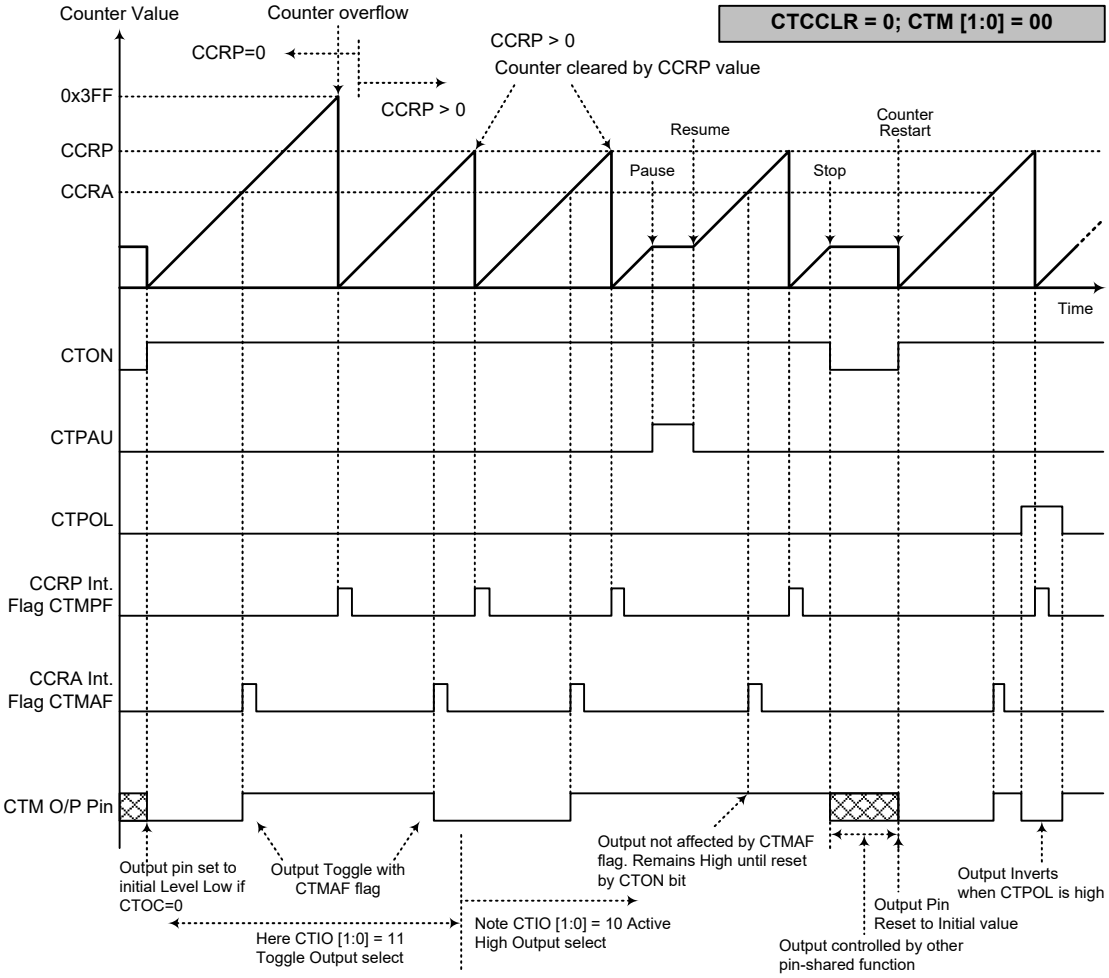
To select this mode, bits CTM1 and CTM0 in the CTMC1 register, should be set to 00 respectively. In this mode once the counter is enabled and running it can be cleared by three methods. These are a counter overflow, a compare match from Comparator A and a compare match from Comparator P. When the CTCCLR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

If the CTCCLR bit in the CTMC1 register is high, then the counter will be cleared when a compare

match occurs from Comparator A. However, here only the CTMAF interrupt request flag will be generated even if the value of the CCRP bits is less than that of the CCRA registers. Therefore, when CTCCLR is high no CTMPF interrupt request flag will be generated.

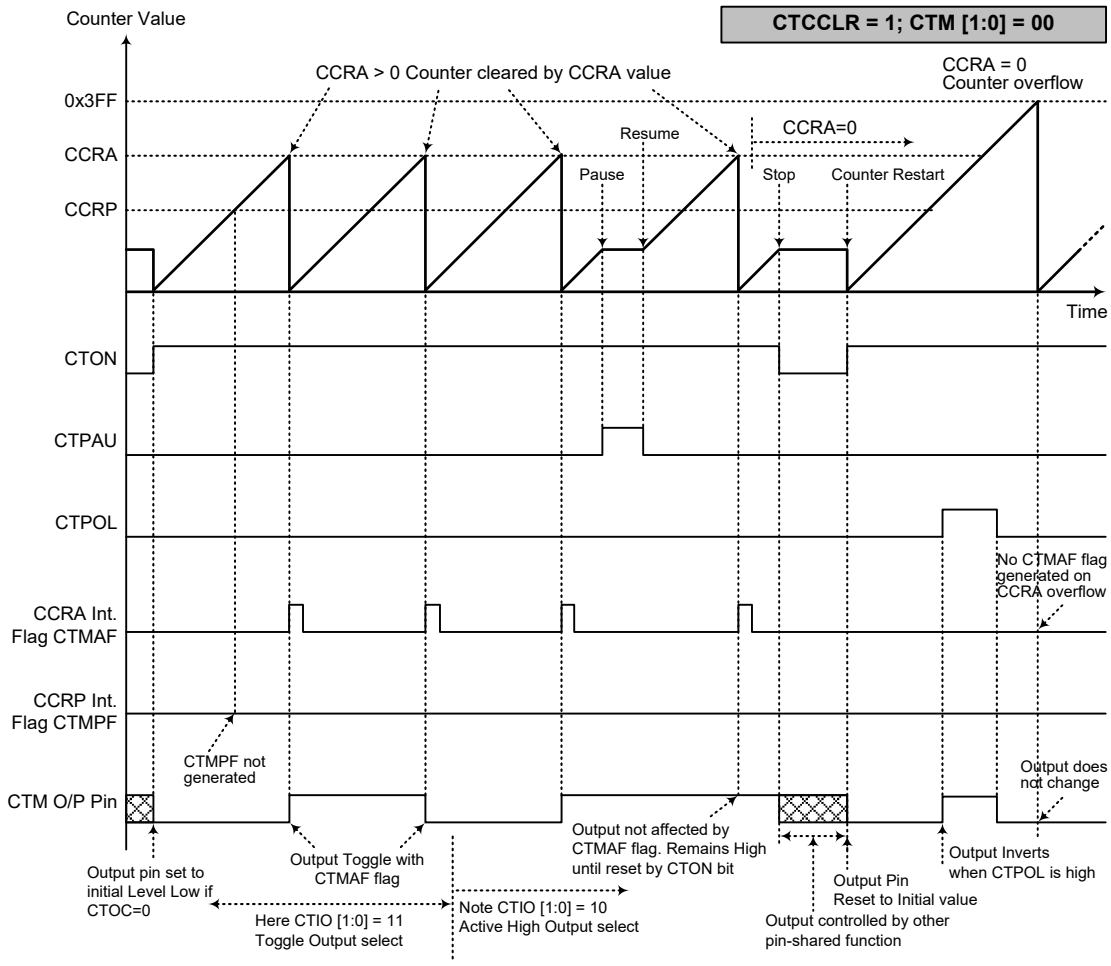
If the CCRA bits are all zero, the counter will overflow when it reaches its maximum 10-bit, 3FF Hex, value, however here the CTMAF interrupt request flag will not be generated.

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



Compare Match Output Mode – CTCCLR=0

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



Compare Match Output Mode – CTCCLR=1

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

Timer/Counter Mode

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

PWM Output Mode

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

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

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

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

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

If f_{SYS}=8MHz, CTM clock source is f_{SYS}/4, CCRP=100b, CCRA=128,

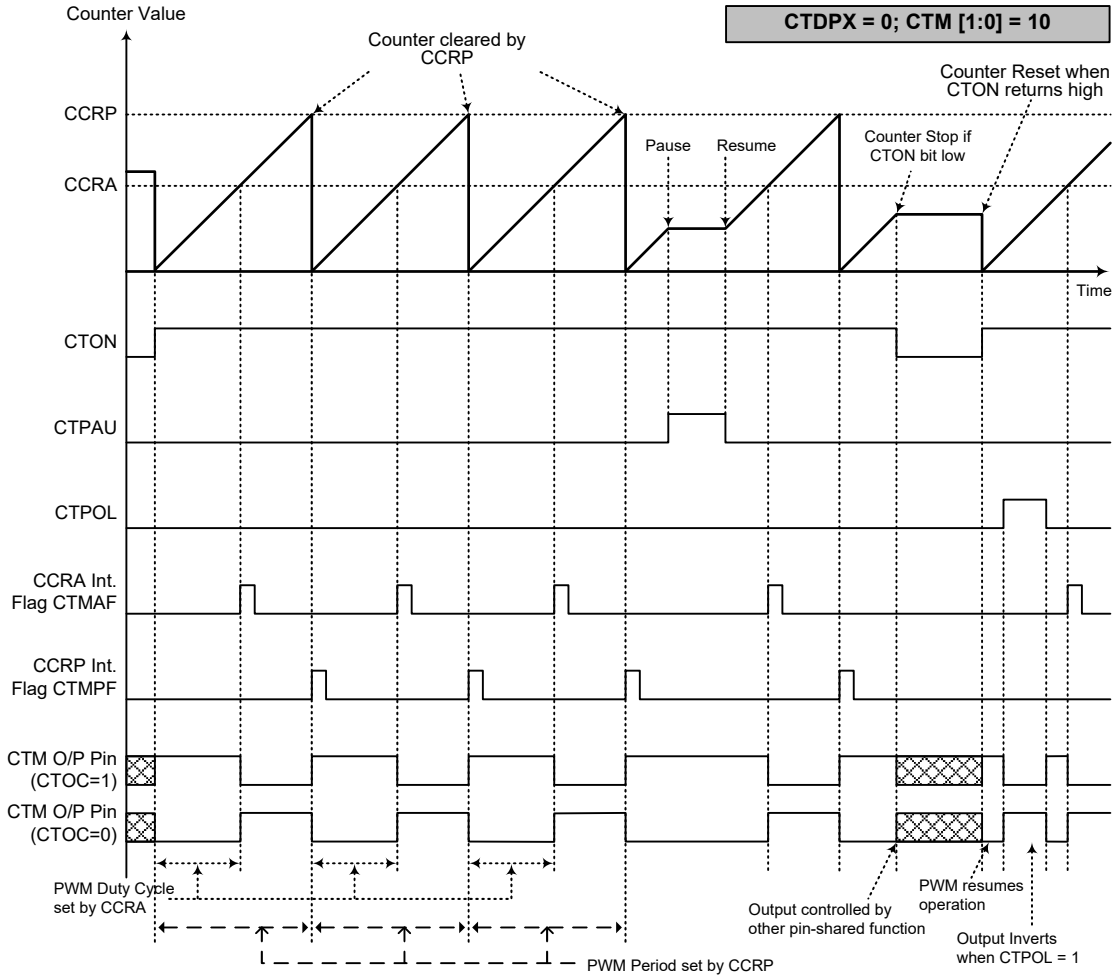
The CTM PWM output frequency=(f_{SYS}/4)/512=f_{SYS}/2048=4kHz, duty=128/512=25%.

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%.

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

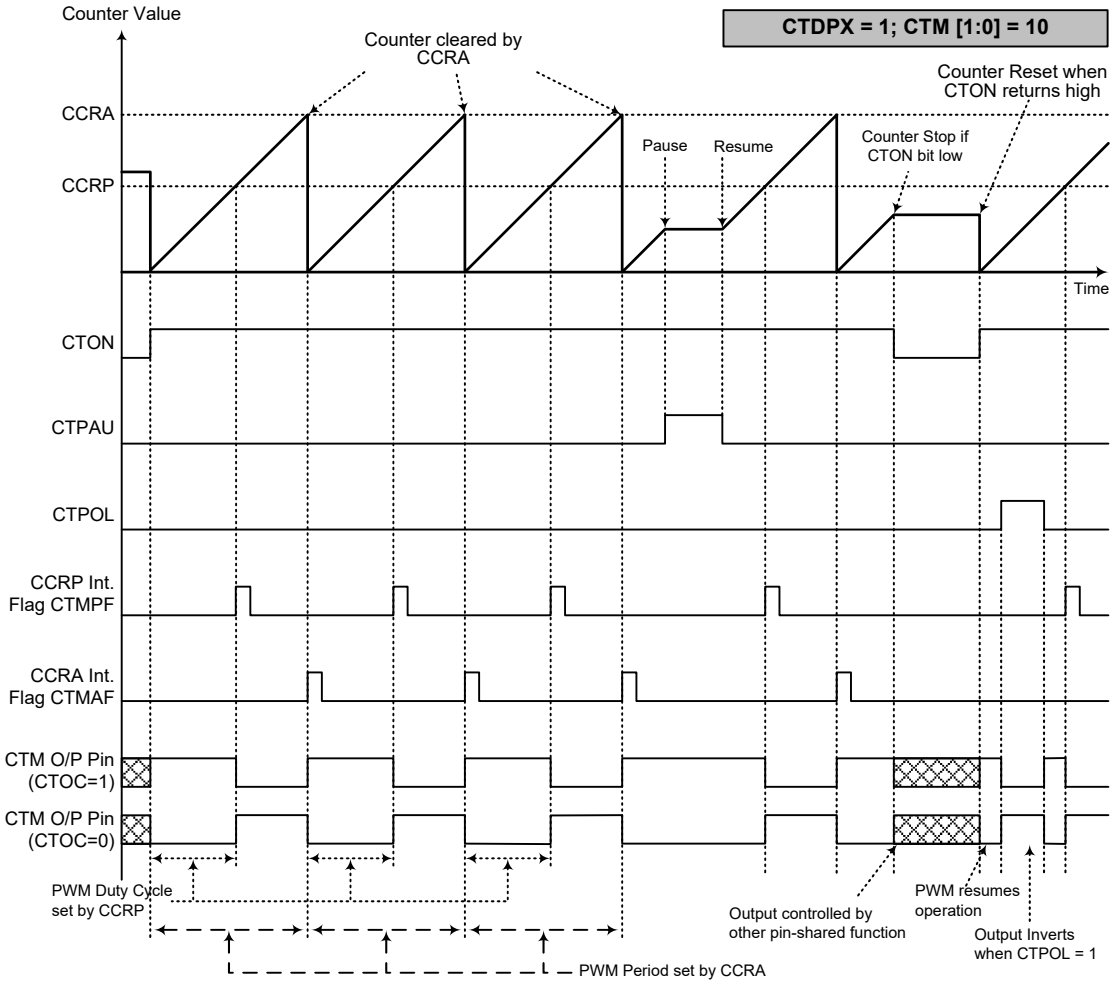
CCRP	001b	010b	011b	100b	101b	110b	111b	000b
Period	CCRA							
Duty	128	256	384	512	640	768	896	1024

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



PWM Output Mode – CTDPX=0

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



PWM Output Mode – CTDPX=1

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

Touch Key Function – BS83A02L

The device provides two 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 pin-shared selection register bit. Keys are organised into one group, known as a module. The module is a fully independent set of two Touch Keys and has its own oscillator. The module contains its own control logic circuits and register set.

Total Key Number	Touch Key	Shared I/O Pin
2	KEY1~KEY2	PA1, PA3

Touch Key Structure

Touch Key Register Definition

The touch key module 0, which contains two touch key functions, has its own suite registers. The following table shows the register set for the touch key module 0.

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
TKM016DL	Touch key module 0 16-bit C/F counter low byte
TKM016DH	Touch key module 0 16-bit C/F counter high byte
TKM0ROL	Touch key module 0 reference oscillator capacitor selection low byte
TKM0ROH	Touch key module 0 reference oscillator capacitor selection high byte
TKM0C0	Touch key module 0 control register 0
TKM0C1	Touch key module 0 control register 1
TKM0C2	Touch key module 0 control register 2
TK1M0TH16L	Touch key module 0 KEY1 16-bit threshold low byte
TK1M0TH16H	Touch key module 0 KEY1 16-bit threshold high byte
TK2M0TH16L	Touch key module 0 KEY2 16-bit threshold low byte
TK2M0TH16H	Touch key module 0 KEY2 16-bit threshold high byte
TKM0K1ROCL	Touch key module 0 KEY1 Reference OSC internal capacitor selection low byte
TKM0K1ROCH	Touch key module 0 KEY1 Reference OSC internal capacitor selection high byte
TKM0K2ROCL	Touch key module 0 KEY2 Reference OSC internal capacitor selection low byte
TKM0K2ROCH	Touch key module 0 KEY2 Reference OSC internal capacitor selection high byte
TKM0K1CNTL	Touch key module 0 KEY1 16-bit counter low byte
TKM0K1CNTH	Touch key module 0 KEY1 16-bit counter high byte
TKM0K2CNTL	Touch key module 0 KEY2 16-bit counter low byte
TKM0K2CNTH	Touch key module 0 KEY2 16-bit counter high byte
TKM0THS	Touch key module 0 threshold comparison flag

Touch Key Function Register Definition

Register Name	Bit							
	7	6	5	4	3	2	1	0
TKTMR	D7	D6	D5	D4	D3	D2	D1	D0
TKC0	—	TKRCOV	TKST	TKCFOV	TK16OV	TKMOD1	TKMOD0	TKBUSY
TKC1	D7	D6	D5	—	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
TKM016DL	D7	D6	D5	D4	D3	D2	D1	D0
TKM016DH	D15	D14	D13	D12	D11	D10	D9	D8
TKM0ROL	D7	D6	D5	D4	D3	D2	D1	D0
TKM0ROH	—	—	—	—	—	—	D9	D8
TKM0C0	—	—	M0DFEN	—	M0SOFC	M0SOF2	M0SOF1	M0SOF0
TKM0C1	M0TSS	—	M0ROEN	M0KOEN	—	—	M0K2EN	M0K1EN
TKM0C2	—	—	—	—	—	M0SK10	—	M0SK00
TK1M0TH16L	D7	D6	D5	D4	D3	D2	D1	D0
TK1M0TH16H	D15	D14	D13	D12	D11	D10	D9	D8
TK2M0TH16L	D7	D6	D5	D4	D3	D2	D1	D0
TK2M0TH16H	D15	D14	D13	D12	D11	D10	D9	D8
TKM0K1ROCL	D7	D6	D5	D4	D3	D2	D1	D0
TKM0K1ROCH	—	—	—	—	—	—	D9	D8
TKM0K2ROCL	D7	D6	D5	D4	D3	D2	D1	D0
TKM0K2ROCH	—	—	—	—	—	—	D9	D8
TKM0K1CNTL	D7	D6	D5	D4	D3	D2	D1	D0
TKM0K1CNTH	D15	D14	D13	D12	D11	D10	D9	D8
TKM0K2CNTL	D7	D6	D5	D4	D3	D2	D1	D0
TKM0K2CNTH	D15	D14	D13	D12	D11	D10	D9	D8
TKMTHS	—	—	M0K2THF	M0K1THF	—	—	M0K2THS	M0K1THS

Touch Key Function Register List

- **TKTMR Register**

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

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

The touch key time slot counter preload register is used to determine the touch key time slot overflow time. The time slot unit period is equal to t_{STC} or $32t_{STC}$. It can be obtained by a 5-bit counter and equal to 32 time slot clock cycles. It also can be obtained by bypassing a 5-bit counter and equal to 1 time slot clock cycle. Therefore, the time slot counter overflow time is equal to the following equation shown.

If M0TSS=0, Time slot counter overflow time=(256-TKTMR[7:0])×$32t_{TSC}$, while if M0TSS=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	—	TKRCOV	TKST	TKCFOV	TK16OV	TKMOD1	TKMOD0	TKBUSY
R/W	—	R/W	R/W	R/W	R/W	R/W	R/W	R
POR	—	0	0	0	0	0	1	0

Bit 7 Unimplemented, read as "0"

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 can not be set by application program but must be cleared to 0 by application program.

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. At this time, the touch key module 0 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 module keys and reference oscillators will automatically stop. The touch key module 0 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 WDT time-out cycle, the 16-bit C/F counter content will be written into the TKM0KnCNTH/TKM0KnCNTL registers, 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 overflows, the TKRCOV bit and the Touch Key TKRCOV Interrupt request flag, TKRCOVF, will be set and all module keys and reference oscillators will automatically stop. The touch key module 0 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

The touch key module 0 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 0 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 0 16-bit C/F counter overflow flag
0: No overflow occurs
1: Overflow occurs

This bit is set by touch key module 0 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 by 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 0 16-bit C/F counter value should be read after the scan operation finishes by application program.

In the auto scan mode, the data movement which is described above is implemented by hardware. The individual reference oscillator capacitor value and 16-bit C/F counter content for all scanned keys will be read from the TKM0KnROCH/TKM0KnROCL and TKM0KnCNTH/TKM0KnCNTL registers. In the auto scan mode, the keys to be scanned can be arranged in a specific sequence which is determined by the M0SK10 and M0SK00 bits in the TKM0C2 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 WDT time-out cycle, the 16-bit C/F counter content for all scanned keys will be written into the TKM0KnCNTH/TKM0KnCNTL registers. In addition, when any key C/F counter value is less than the lower threshold if M0KnTHS=0, or larger than the upper threshold if M0KnTHS=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 for all touch key scan modes.

In the manual scan mode this bit is cleared to 0 automatically when the touch key time slot counter 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 WDT time-out cycle is completed, or when any key C/F counter value is less than the lower threshold if M0KnTHS=0, or when the value is larger than the upper threshold if M0KnTHS=1.

• TKC1 Register

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	D5	—	TK16S1	TK16S0	TKFS1	TKFS0
R/W	R/W	R/W	R/W	—	R/W	R/W	R/W	R/W
POR	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 Unimplemented, read as “0”

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	0

Bit 7~3 Unimplemented, read as "0"

Bit 2 **TSC**: Time slot control
0: Time slot 0~1 (default)
1: Time slot 0 only
The bit is used for configure time slot function. If the TSC bit is set high, it can reduce power consumption in the IDLE or SLEEP mode.

Bit 1~0 **ASMP1~ASMP0**: Periodic auto scan mode period selection
00: $t_{WDT}/2$
01: $t_{WDT}/4$ (default)
10: $t_{WDT}/8$
11: $t_{WDT}/16$
These bits are used to determine the touch key scan period and only available when the touch key fucntion is configured to operate in the periodic auto scan mode. The number of touch key scan times is obtained by the WDT time-out period, t_{WDT}, and the periodic auto scan mode period, t_{KEY}, using the equation, $N=t_{WDT}/t_{KEY}$. The WDT time-out period t_{WDT} is selected by the WS2~WS0 bits.
For example, if the WDT time-out period is $2^{11}/f_{LIRC}$ by setting the WS[2:0] bits to 100, then t_{WDT} is equal to 1.024s. Therefore, the number of touch key scan times is 2/4/8/16 times in a WDT 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 WDT time-out period t_{WDT} 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 but kept unchanged at the end of the time slot 1 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.

• **TKM016DH/TKM016DL – Touch Key Module 0 16-bit C/F Counter Register Pair**

Register	TKM016DH								TKM016DL							
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 0 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 and is written to the TKM0KnCNTH/TKM0KnCNTL registers at the end of the time

slot 0 but kept unchanged at the end of the time slot 1 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.

- **TKM0ROH/TKM0ROL – Touch Key Module 0 Reference Oscillator Capacitor Selection Register Pair**

Register	TKM0ROH								TKM0ROL							
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 0 reference oscillator capacitor value. This register pair will be loaded with the corresponding next time slot capacitor value from the TKM0KnROCH/TKM0KnROCL registers 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=$\frac{\text{TKM0RO[9:0]}\times 30\text{pF}}{1024}$

- **TKM0C0 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	M0DFEN	—	M0SOFC	M0SOF2	M0SOF1	M0SOF0
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 **M0DFEN**: Touch key module 0 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 **M0SOFC**: Touch key module 0 C/F oscillator frequency hopping function control selection
0: Controlled by the M0SOF2~M0SOF0
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 M0SOF2~M0SOF0 bits value.

Bit 2~0 **M0SOF2~M0SOF0**: Touch key module 0 reference and key oscillators hopping frequency selection (M0SOFC=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 M0SOFC bit is cleared to 0.
The frequency mentioned here will be changed when the external or internal capacitor is with different values. If the touch key operates at 1MHz frequency, users can adjust the frequency in scale when any other frequency is selected.

• **TKM0C1 Register**

Bit	7	6	5	4	3	2	1	0
Name	M0TSS	—	M0ROEN	M0KOEN	—	—	M0K2EN	M0K1EN
R/W	R/W	—	R/W	R/W	—	—	R/W	R/W
POR	0	—	0	0	—	—	0	0

Bit 7 **M0TSS**: Touch key module 0 time slot counter clock source selection
0: Touch key module 0 reference oscillator
1: f_{LIRC}

Bit 6 Unimplemented, read as "0"

Bit 5 **M0ROEN**: Touch key module 0 reference oscillator control
0: Disable
1: Enable
This bit is used to enable the touch key module 0 reference oscillator. In the auto scan mode or the periodic auto scan mode, the reference oscillator will automatically be enabled by setting the M0ROEN bit high when the TKST bit is set from low to high if the reference oscillator is selected as the time slot clock source. The combination of the M0TSS and M0K2EN~M0K1EN bits determines whether the reference oscillator is used or not. When the TKBUSY bit is changed from high to low, the M0ROEN bit will automatically be cleared to zero to disable the reference oscillator.
In the manual scan mode the reference oscillator should first be enabled before setting the TKST bit from low to high if the reference oscillator is selected to be used and will be disabled when the TKBUSY bit is changed from high to low.

Bit 4 **M0KOEN**: Touch key module 0 key oscillator control
0: Disable
1: Enable
This bit is used to enable the touch key module 0 key oscillator. In the auto scan mode or the periodic auto scan mode, the key oscillator will automatically be enabled by setting the M0KOEN bit high when the TKST bit is set form low to high. When the TKBUSY bit is changed from high to low, the M0KOEN bit will automatically be cleared to zero to disable the key oscillator.
In the manual scan mode, the key oscillator should first be enabled before setting the TKST bit from low to high if the relevant key is enabled to be scanned and will be disabled when the TKBUSY bit is changed from high to low.

Bit 3~2 Unimplemented, read as "0"

Bit 1 **M0K2EN**: Touch key module 0 KEY 2 control
0: isable
1: Enable

Bit 0 **M0K1EN**: Touch key module 0 KEY 1 control
0: Disable
1: Enable

• **TKM0C2 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	M0SK10	—	M0SK00
R/W	—	—	—	—	—	R/W	—	R/W
POR	—	—	—	—	—	1	—	0

Bit 7~3 Unimplemented, read as "0"

Bit 2 **M0SK10**: Touch key module 0 time slot 1 key scan selection
0: KEY1
1: KEY2
These bits are used to select the desired scan key in time slot 1 and only available in the auto scan mode or the periodic auto scan mode.

Bit 1 Unimplemented, read as "0"

Bit 0 **M0SK00**: Touch key module 0 time slot 0 key scan selection
0: KEY1
1: KEY2
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 select in the manual mode.

- **TKnM0TH16H/TKnM0TH16L – Touch Key Module 0 KEYn 16-bit Threshold Register Pair (n=1~2)**

Register	TKnM0TH16H								TKnM0TH16L							
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

These two register pairs are used to store the touch key module 0 KEY1~KEY2 16-bit upper/lower threshold value respectively. After the touch key module 0 KEYn scan operation is completed, the 16-bit C/F counter content, TKM016DH/TKM016DL, will be compared with the TKnM0TH16H/TKnM0TH16L value by the hardware. When this value is less than the the lower threshold if M0KnTHS=0, or larger than the upper threshold if M0KnTHS=1, then the M0KnTHF flag will be set high, and an interrupt signal will be generated.

- **TKM0KnROCH/TKM0KnROCL – Touch Key Module 0 KEYn Reference Oscillator Capacitor Selection Register Pair (n=1~2)**

Register	TKM0KnROCH								TKM0KnROCL							
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

These two register pairs are used to store the touch key module 0 KEYn reference oscillator capacitor value.

- **TKM0KnCNTH/TKM0KnCNTL – Touch Key Module 0 KEYn 16-bit Counter Register Pair (n=1~2)**

Register	TKM0KnCNTH								TKM0KnCNTL							
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

These two register pairs are used to store the Touch key module 0 KEYn 16-bit counter contents.

• **TKM0THS Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	M0K2THF	M0K1THF	—	—	M0K2THS	M0K1THS
R/W	—	—	R/W	R/W	—	—	R/W	R/W
POR	—	—	0	0	—	—	0	0

Bit 7~6 Unimplemented, read as "0"

Bit 5 **M0K2THF**: Touch key module 0 KEY2 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 **M0K1THF**: Touch key module 0 KEY1 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

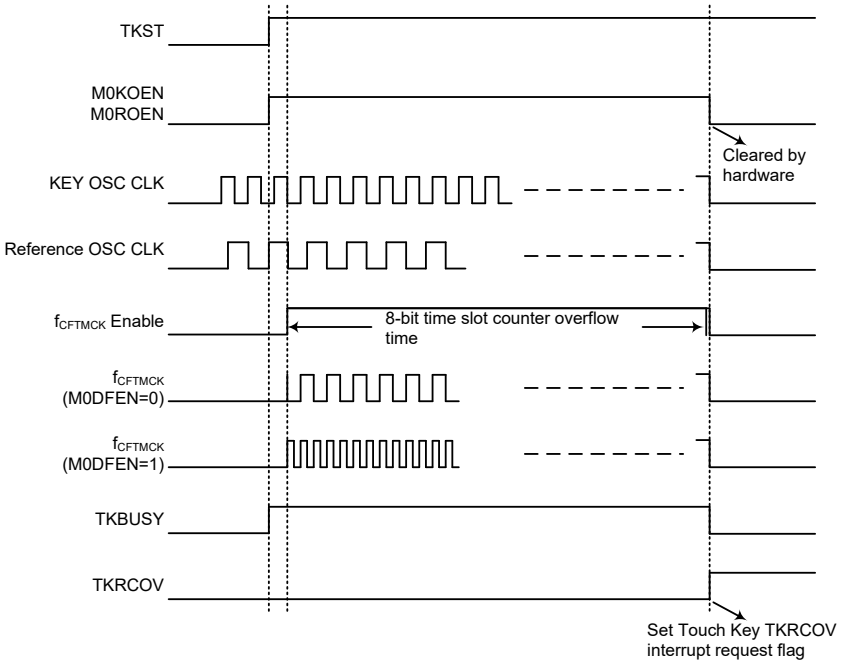
Bit 3~2 Unimplemented, read as "0"

Bit 1 **M0K2THS**: Touch key module 0 KEY2 upper or lower threshold comparison selection
0: Lower threshold comparison
1: Upper threshold comparison

Bit 0 **M0K1THS**: Touch key module 0 KEY1 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 sense oscillator, touch actions can be sensed by measuring these frequency changes. Using an internal programmable divider the reference clock is used to generate a fixed time period. By counting a number of generated clock cycles from the sense oscillator during this fixed time period touch key actions can be determined.



Touch Key Manual Scan Mode Timing Diagram

The touch key module 0 contains two touch key inputs, namely KEY1~KEY2, which are shared with logical I/O pins, and the desired function is selected using register bits. The touch key has its own independent sense oscillator. There are therefore two sense oscillators within the touch key module 0.

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

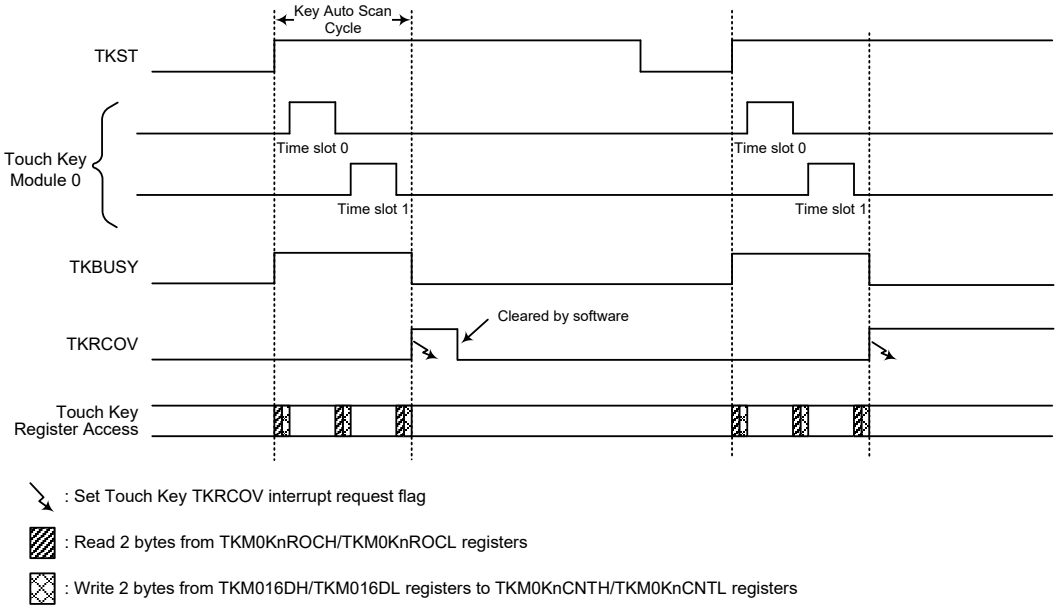
The touch key module 0 16-bit C/F counter, 16-bit counter, 5-bit time slot unit period counter in the module will be automatically cleared when the TKST bit is cleared to zero, but the 8-bit programmable time slot counter will not be cleared. The overflow time is setup by user. When the TKST bit changes from low to high, the 16-bit C/F counter, 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 the module 0 will be automatically stopped and the 16-bit C/F counter, 16-bit counter, 5-bit time slot unit period counter and 8-bit time slot timer counter will be automatically switched off when the time slot counter overflows. The clock source for the time slot counter is sourced from the reference oscillator or f_{LIRC} which is selected using the M0TSS bit in the TKM0C1 register. The reference oscillator and key oscillator will be enabled by setting the M0ROEN bit and M0KOEN bits in the TKM0C1 register.

When the time slot counter 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.

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 minisize 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 M0SK10 and M0SK00 bits in the TKM0C2 register.



Touch Key Auto Scan Mode Timing Diagram

In the auto scan mode, the key oscillator and reference oscillator will automatically be enabled when the TKST bit is set from low to high and disabled automatically when the TKBUSY bit changes from high to low. When the TKST bit is set from low to high in the auto scan mode, the internal capacitor value of the reference oscillator for the selected key to be scanned in the time slot 0 will first be read from the TKM0KnROCH/TKM0KnROCL registers and loaded into the TKM0ROH/TKM0ROL registers. Then the 16-bit C/F counter value will be written into the corresponding location of the time slot 1 scanned key in the TKM0KnCNTH/TKM0KnCNTL registers. 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 TKM0KnROCH/TKM0KnROCL registers and loaded into the next TKM0ROH/TKM0ROL registers. Then the 16-bit C/F counter value of the current scanned key will be written into the TKM0KnCNTH/TKM0KnCNTL registers. The whole auto scan operation will sequentially be carried out in the above specific way from time slot 0 to time slot 1. At the end of the time slot 1 key scan operation, the reference oscillator internal capacitor value for the time slot 0 selected key will again be read from the TKM0K1ROCH/TKM0K1ROCL registers and loaded into the TKM0ROH/TKM0ROL registers. Then the 16-bit C/F counter value will be written into the TKM0KnCNTH/TKM0KnCNTL registers. After two 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.

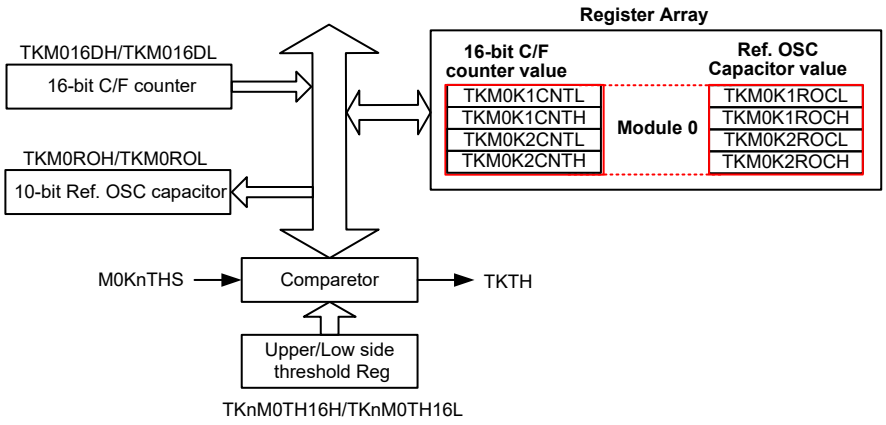
Periodic Auto Scan Mode

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 WDT 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 1 like the auto scan mode. The reference oscillator internal capacitor value for each time slot selected key will be read from the TKM0KnROCH/TKM0KnROCL registers and loaded into the TKM0ROH/TKM0ROL registers. However, only at the end of the last scan operation in the WDT time-out cycle, the 16-bit C/F counter value for all scanned keys will be written into the TKM0KnCNTH/TKM0KnCNTL registers.

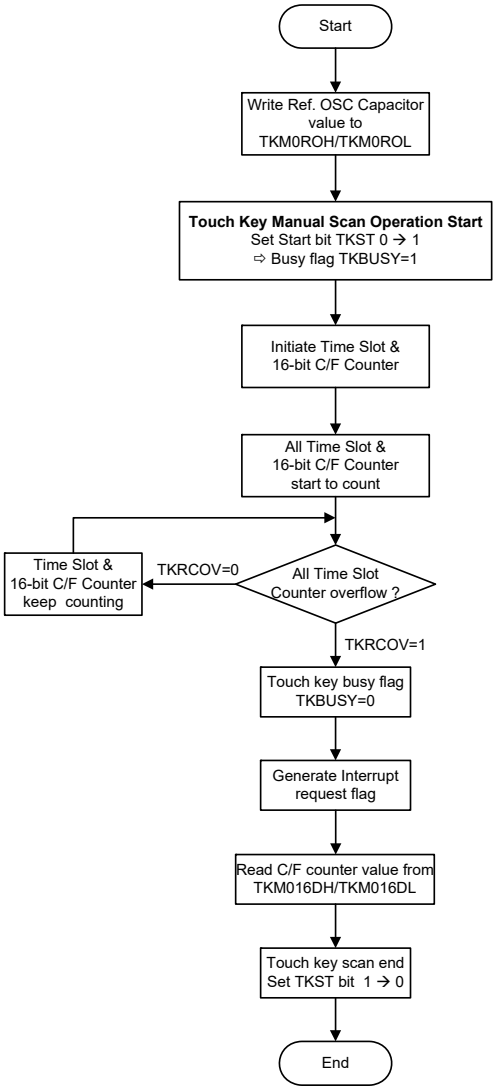
In addition, each touch key has its own independent upper/lower threshold comparator. The upper/lower threshold comparison function will automatically be enabled in the periodic auto scan mode. When any key C/F counter value is less than the lower threshold if M0KnTHS=0, or larger than the upper threshold if M0KnTHS=1, this indicates that the touch key state changes, then the M0KnTHF flag will be set high by the hardware, and an interrupt signal will be generated.

As the periodic auto scan operation is implemented using the WDT counter clock to reduce power consumption, when the WDT is cleared the WDT counter will be reset, the periodic auto scan operation time will be affected but the number of touch key times will not be affected.

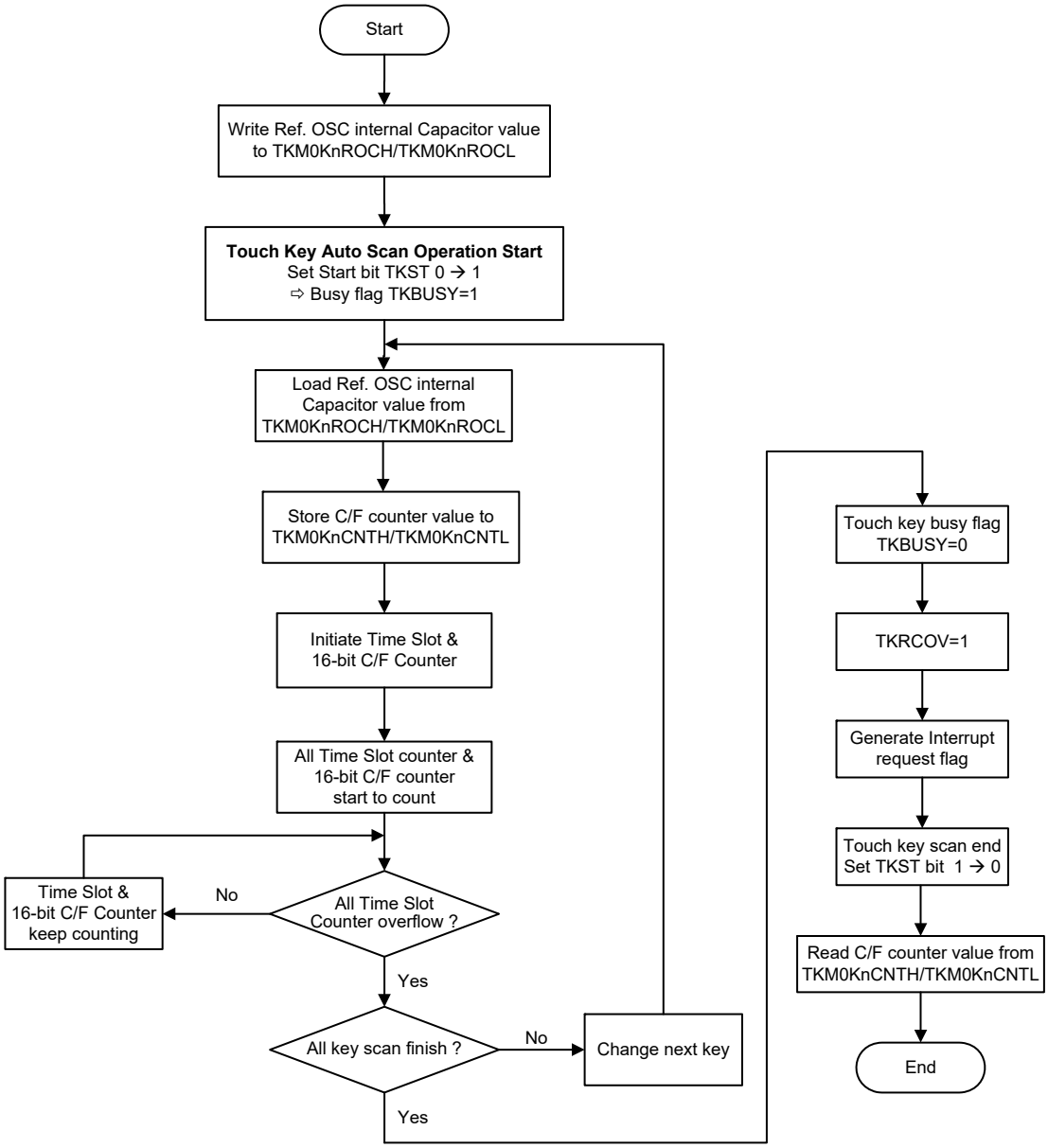


Touch Key Register Allocation

Touch Key Scan Operation Flowchart



Touch Key Manual Scan Mode Flowchart



Touch Key Auto Scan Mode Flowchart

Touch Key Interrupts

The touch key has two independent interrupts, known as touch key TKRCOV interrupt and touch key threshold TKTH interrupt. In the manual scan mode, when the touch key module 0 time slot counter 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, after the last scan operation in the WDT time-out cycle completes the 16-bit C/F counter content is written into the TKM0KnCNTH/TKM0KnCNTL registers, 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 M0KnTHS=0, or larger than the upper threshold if M0KnTHS=1, a touch key threshold 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 WDT time-out cycle, the 16-bit C/F counter content will be written into the TKM0KnCNTH/TKM0KnCNTL registers, 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 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 0 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.

Touch Key Function – BS83B04L

The BS83B04L provides multiple touch key functions. The touch key function is fully integrated and requires no external components, allowing touch key functions to be implemented by the simple manipulation of internal registers.

Touch Key Structure

The touch keys are pin shared with the I/O pins, with the desired function chosen via the pin-shared selection register bit. Keys are organised into one group, known as a module. The module is a fully independent set of four Touch Keys and has its own oscillator. The module contains its own control logic circuits and register set.

Total Key Number	Touch Key	Shared I/O Pin
4	KEY1~KEY4	PA5, PA1, PA3, PA4

Touch Key Structure

Touch Key Register Definition

The touch key module 0, which contains four touch key functions, is controlled using several registers. The following table shows the register set for the touch key module 0.

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
TKM016DL	Touch key module 0 16-bit C/F counter low byte
TKM016DH	Touch key module 0 16-bit C/F counter high byte
TKM0ROL	Touch key module 0 reference oscillator capacitor selection low byte
TKM0ROH	Touch key module 0 reference oscillator capacitor selection high byte
TKM0C0	Touch key module 0 control register 0
TKM0C1	Touch key module 0 control register 1
TKM0C2	Touch key module 0 control register 2
TKM0TH16L	Touch key module 0 16-bit threshold low byte
TKM0TH16H	Touch key module 0 16-bit threshold high byte
TKM0THS	Touch key module 0 threshold comparison flag

Touch Key Function Register Definition

Register Name	Bit							
	7	6	5	4	3	2	1	0
TKTMR	D7	D6	D5	D4	D3	D2	D1	D0
TKC0	TKRAMC	TKRCOV	TKST	TKCFOV	TK16OV	TKMOD1	TKMOD0	TKBUSY
TKC1	D7	D6	D5	—	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
TKM016DL	D7	D6	D5	D4	D3	D2	D1	D0
TKM016DH	D15	D14	D13	D12	D11	D10	D9	D8
TKM0ROL	D7	D6	D5	D4	D3	D2	D1	D0

Register Name	Bit							
	7	6	5	4	3	2	1	0
TKM0ROH	—	—	—	—	—	—	D9	D8
TKM0C0	—	—	M0DFEN	—	M0SOFC	M0SOF2	M0SOF1	M0SOF0
TKM0C1	M0TSS	—	M0ROEN	M0KOEN	M0K4EN	M0K3EN	M0K2EN	M0K1EN
TKM0C2	M0SK31	M0SK30	M0SK21	M0SK20	M0SK11	M0SK10	M0SK01	M0SK00
TKM0TH16L	D7	D6	D5	D4	D3	D2	D1	D0
TKM0TH16H	D15	D14	D13	D12	D11	D10	D9	D8
TKM0THS	M0K4THF	M0K3THF	M0K2THF	M0K1THF	M0K4THS	M0K3THS	M0K2THS	M0K1THS

Touch Key Function Register List

- **TKTMR Register**

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

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

The touch key time slot counter preload register is used to determine the touch key time slot overflow time. The time slot unit period is equal to t_{STC} or $32t_{STC}$. It can be obtained by a 5-bit counter and equal to 32 time slot clock cycles. It also can be obtained by bypassing a 5-bit counter and equal to 1 time slot clock cycle. Therefore, the time slot counter overflow time is equal to the following equation shown.

If M0TSS=0, Time slot counter overflow time=(256-TKTMR[7:0])×$32t_{TSC}$, while if M0TSS=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.

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 can not be set by application program but must be cleared to 0 by application program.

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. At this time, the touch key module 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 module keys and reference oscillators will automatically stop. The touch key module 16-bit C/F counter, touch key function 16-bit counter, 5-bit time slot unit period counter and 8-bit time slot counter will be automatically switched off.

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 WDT 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 overflows, the TKRCOV bit and the Touch Key TKRCOV Interrupt request flag, TKRCOVF, will be set and all module keys and reference oscillators will automatically stop. The touch key module 16-bit C/F counter, touch key function 16-bit counter, 5-bit time slot unit period counter and 8-bit time slot counter will be automatically switched off.

Bit 5 **TKST**: Touch key detection start control
0: Stopped or no operation
0→1: Start detection

The touch key module 0 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 0 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 0 16-bit C/F counter overflow flag
0: No overflow occurs
1: Overflow occurs

This bit is set by touch key module 0 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 by 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 0 16-bit C/F counter value should be read after the scan operation finishes by application program.

In the auto scan mode, the data movement which is described above is implemented by hardware. The individual reference oscillator capacitor value and 16-bit C/F counter content for all scanned keys will be read from 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 M0SK3[1:0]~M0SK0[1:0] bits in the TKM0C2

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 WDT 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 M0KnTHS=0, or larger than the upper threshold if M0KnTHS=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 touch key time slot counter 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 WDT time-out cycle is completed, or when any key C/F counter value is less than the lower threshold if M0KnTHS=0, or when the value is larger than the upper threshold if M0KnTHS=1.

• TKC1 Register

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	D5	—	TK16S1	TK16S0	TKFS1	TKFS0
R/W	R/W	R/W	R/W	—	R/W	R/W	R/W	R/W
POR	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 Unimplemented, read as “0”

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 (default)
1: Time slot 0 only

This bit is used to configure time slot, if TSC 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 IDLE or SLEEP mode. The desired wake-up key in time slot 0 is selected by the M0SK01~M0SK00 bits in the TKM0C2 register.

Bit 1~0 **ASMP1~ASMP0**: Periodic auto scan mode period selection
00: $2^{14}/f_{LIRC}$
01: $2^{13}/f_{LIRC}$
10: $2^{12}/f_{LIRC}$
11: $2^{11}/f_{LIRC}$

These bits are used to determine the touch key scan period and only available when the touch key fucntion is configured to operate in the periodic auto scan mode. The number of touch key scan times is obtained by the WDT time-out period, t_{WDT}, and the periodic auto scan mode period, t_{KEY}, using the equation, $N=t_{WDT}/t_{KEY}$. The WDT time-out period t_{WDT} is selected by the WS2~WS0 bits.

For example, if the WDT time-out period is $2^{11}/f_{WDT}=2^{14}/f_{LIRC}$ by setting the WS[2:0] bits to 111, then t_{WDT} is about equal to 8s. Therefore, the number of touch key scan times is 1/2/4/8 times in a WDT 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 WDT time-out period t_{WDT} 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 set low.

• TKM016DH/TKM016DL – Touch Key Module 0 16-bit C/F Counter Register Pair

Register	TKM016DH								TKM016DL							
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 0 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 set low.

• **TKM0ROH/TKM0ROL – Touch Key Module 0 Reference Oscillator Capacitor Selection Register Pair**

Register	TKM0ROH								TKM0ROL							
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 0 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=$\frac{\text{TKM0RO[9:0]}\times 50\text{pF}}{1024}$

• **TKM0C0 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	M0DFEN	—	M0SOFC	M0SOF2	M0SOF1	M0SOF0
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 **M0DFEN**: Touch key module 0 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 **M0SOFC**: Touch key module 0 C/F oscillator frequency hopping function control selection
0: Controlled by the M0SOF2~M0SOF0
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 M0SOF2~M0SOF0 bits value.

Bit 2~0 **M0SOF2~M0SOF0**: Touch key module 0 reference and key oscillators hopping frequency selection (M0SOFC=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 M0SOFC bit is cleared to 0.
The frequency mentioned here will be changed when the external or internal capacitor is with different values. If the touch key operates at 1MHz frequency, users can adjust the frequency in scale when any other frequency is selected.

• **TKM0C1 Register**

Bit	7	6	5	4	3	2	1	0
Name	M0TSS	—	M0ROEN	M0KOEN	M0K4EN	M0K3EN	M0K2EN	M0K1EN
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 **M0TSS**: Touch key module 0 time slot counter clock source selection
0: Touch key module 0 reference oscillator
1: f_{LIRC}

When the M0TSS bit is high, the touch key module 0 time slot counter clock comes from the f_{LIRC} oscillator. which can remain on even when the device enters the IDLE and SLEEP modes. The touch key module 0 time slot counter clock is then can bypass the 5-bit counter to reduce the overflow time. In the periodic auto scan mode, set the M0TSS and TSC bits high can reduce the touch key standby power.

Note: If the 5-bit counter is bypassed, the touch key module 0 Reference and Key oscillators hopping frequency is selected by the M0SOF2~M0SOF0 bits regardless of the M0SOFC bit setting.

Bit 6 Unimplemented, read as "0"

Bit 5 **M0ROEN**: Touch key module 0 reference oscillator control
0: Disable
1: Enable

This bit is used to enable the touch key module 0 reference oscillator. In the auto scan mode or the periodic auto scan mode, the reference oscillator will automatically be enabled by setting the M0ROEN bit high when the TKST bit is set from low to high if the reference oscillator is selected as the time slot clock source. The combination of the M0TSS and M0K4EN~M0K1EN bits determines whether the reference oscillator is used or not. When the TKBUSY bit is changed from high to low, the M0ROEN bit will automatically be cleared to zero to disable the reference oscillator.

In the manual scan mode, the reference oscillator should first be enabled before setting the TKST bit from low to high if the reference oscillator is selected to be used and will be disabled when the TKBUSY bit is changed from high to low.

Bit 4 **M0KOEN**: Touch key module 0 key oscillator control
0: Disable
1: Enable

This bit is used to enable the touch key module 0 key oscillator. In the auto scan mode or the periodic auto scan mode, the key oscillator will automatically be enabled by setting the M0KOEN bit high when the TKST bit is set form low to high. When the TKBUSY bit is changed from high to low, the M0KOEN bit will automatically be cleared to zero to disable the key oscillator.

In the manual scan mode, the key oscillator should first be enabled before setting the TKST bit from low to high if the relevant key is enabled to be scanned and will be disabled when the TKBUSY bit is changed from high to low.

Bit 3 **M0K4EN**: Touch key module 0 KEY 4 control
0: Disable
1: Enable

Bit 2 **M0K3EN**: Touch key module 0 KEY 3 control
0: Disable
1: Enable

Bit 1 **M0K2EN**: Touch key module 0 KEY 2 control
0: Disable
1: Enable

Bit 0 **M0K1EN**: Touch key module 0 KEY 1 control
0: Disable
1: Enable

• **TKM0C2 Register**

This register is used to select the desired scan key in the time slots 0~3 of the touch key module 0. It should be noted that if any key is disabled, the touch key module 0 Reference and Key oscillators of the corresponding time slot will not oscillate.

Bit	7	6	5	4	3	2	1	0
Name	M0SK31	M0SK30	M0SK21	M0SK20	M0SK11	M0SK10	M0SK01	M0SK00
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 **M0SK31~M0SK30**: Touch key module 0 time slot 3 key scan select
00: KEY1
01: KEY2
10: KEY3
11: KEY4
These bits are used to select the desired scan key in time slot 3. The settings for these bits are invalid when TKMOD1~TKMOD0=01 or TSC=1.

Bit 5~4 **M0SK21~M0SK20**: Touch key module 0 time slot 2 key scan select
00: KEY1
01: KEY2
10: KEY3
11: KEY4
These bits are used to select the desired scan key in time slot 2. The settings for these bits are invalid when TKMOD1~TKMOD0=01 or TSC=1.

Bit 3~2 **M0SK11~M0SK10**: Touch key module 0 time slot 1 key scan select
00: KEY1
01: KEY2
10: KEY3
11: KEY4
These bits are used to select the desired scan key in time slot 1. The settings for these bits are invalid when TKMOD1~TKMOD0=01 or TSC=1.

Bit 1~0 **M0SK01~M0SK00**: Touch key module 0 time slot 0 key scan select
00: KEY1
01: KEY2
10: KEY3
11: KEY4
These bits are used to select the desired scan key in time slot 0 in the auto scan mode or the periodic auto scan mode or used as the multiplexer for scan key select in the manual mode.

• **TKM0TH16H/TKM0TH16L – Touch Key Module 0 16-bit Threshold Register Pair**

Register	TKM0TH16H								TKM0TH16L							
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 0 16-bit upper/lower threshold value. This register pair will be loaded with the corresponding upper/lower threshold value from the dedicated touch key data memory automatically by the hardware before scanning the touch key. After the touch key module 0 dedicated touch key, KEYn (n=1~4), scan operation is completed, the 16-bit C/F counter content, TKM016DH/TKM016DL, will be compared with the TKM0TH16H/TKM0TH16L value by the hardware. When this value is less than the lower threshold if M0KnTHS=0, or larger than the upper threshold if M0KnTHS=1, then the M0KnTHF flag will be set high, and an interrupt signal will be generated.

• **TKM0THS Register**

Bit	7	6	5	4	3	2	1	0
Name	M0K4THF	M0K3THF	M0K2THF	M0K1THF	M0K4THS	M0K3THS	M0K2THS	M0K1THS
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 **M0K4THF**: Touch key module 0 KEY4 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 **M0K3THF**: Touch key module 0 KEY3 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 **M0K2THF**: Touch key module 0 KEY2 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 **M0K1THF**: Touch key module 0 KEY1 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 **M0K4THS**: Touch key module 0 KEY4 upper or lower threshold comparison selection
0: Lower threshold comparison
1: Upper threshold comparison

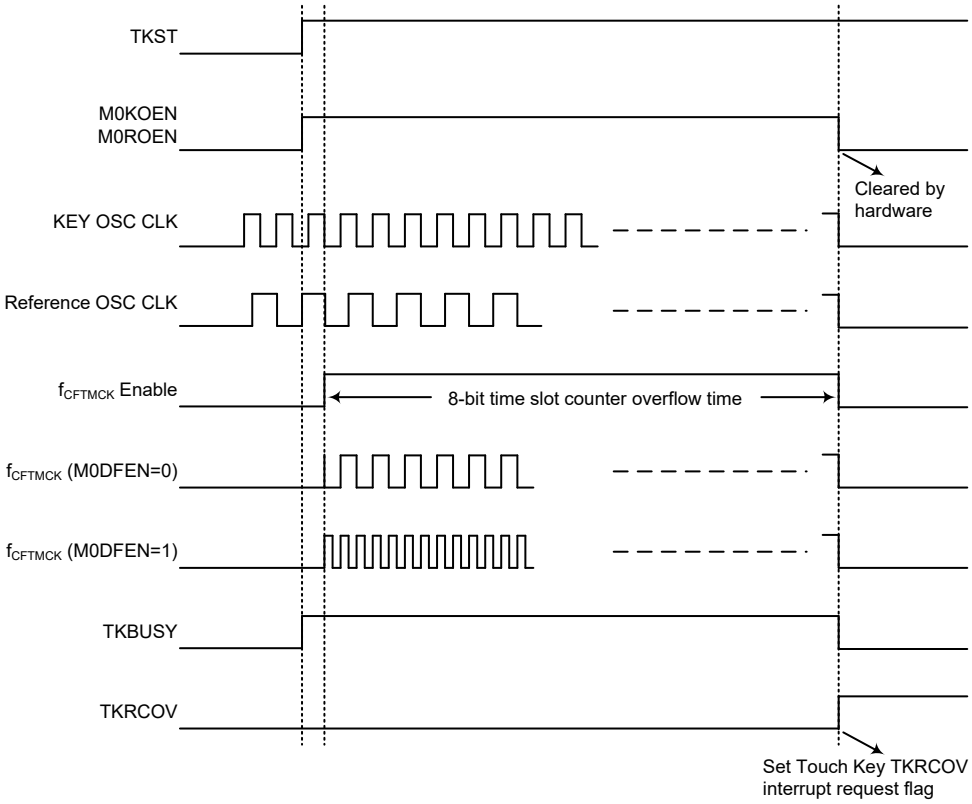
Bit 2 **M0K3THS**: Touch key module 0 KEY3 upper or lower threshold comparison selection
0: Lower threshold comparison
1: Upper threshold comparison

Bit 1 **M0K2THS**: Touch key module 0 KEY2 upper or lower threshold comparison selection
0: Lower threshold comparison
1: Upper threshold comparison

Bit 0 **M0K1THS**: Touch key module 0 KEY1 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 sense oscillator, touch actions can be sensed by measuring these frequency changes. Using an internal programmable divider which the reference clock is used to generate a fixed time period. By counting a number of generated clock cycles from the sense oscillator during this fixed time period touch key actions can be determined.



Touch Key Manual Scan Mode Timing Diagram

The touch key module 0 contains four touch key inputs, namely KEY1~KEY4, which are shared with logical I/O pins, and the desired function is selected using register bits. The touch key has its own independent sense oscillator. There are therefore four sense oscillators within the touch key module 0.

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

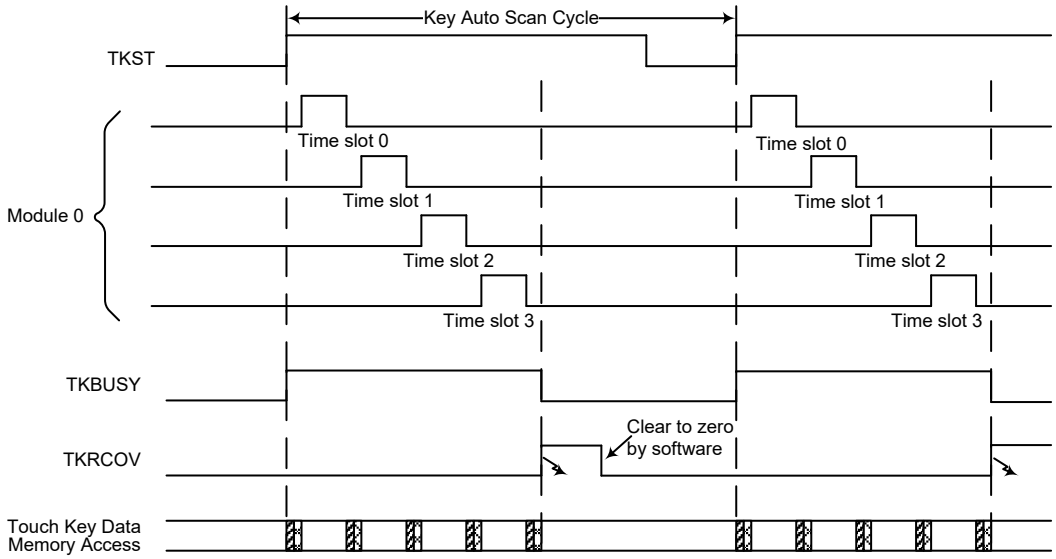
The touch key module 0 16-bit C/F counter, 16-bit counter, 5-bit time slot unit period counter in the module will be automatically cleared when the TKST bit is cleared to zero, but the 8-bit programmable time slot counter will not be cleared. The overflow time is setup by user. When the TKST bit changes from low to high, the 16-bit C/F counter, 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 the module 0 will be automatically stopped and the 16-bit C/F counter, 16-bit counter, 5-bit time slot unit period counter and 8-bit time slot timer counter will be automatically switched off when the time slot counter overflows. The clock source for the time slot counter is sourced from the reference oscillator or f_{LIRC} which is selected using the M0TSS bit in the TKM0C1 register. The reference oscillator and key oscillator will be enabled by setting the M0ROEN bit and M0KOEN bits in the TKM0C1 register.

When the time slot counter 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.

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 minisize 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 selcted to scan the module keys in a specific sequence determined by the M0SK3[1:0]~M0SK0[1:0] bits in the TKM0C2 register. The TSC bit in the TKC2 register is used to configure time slot.

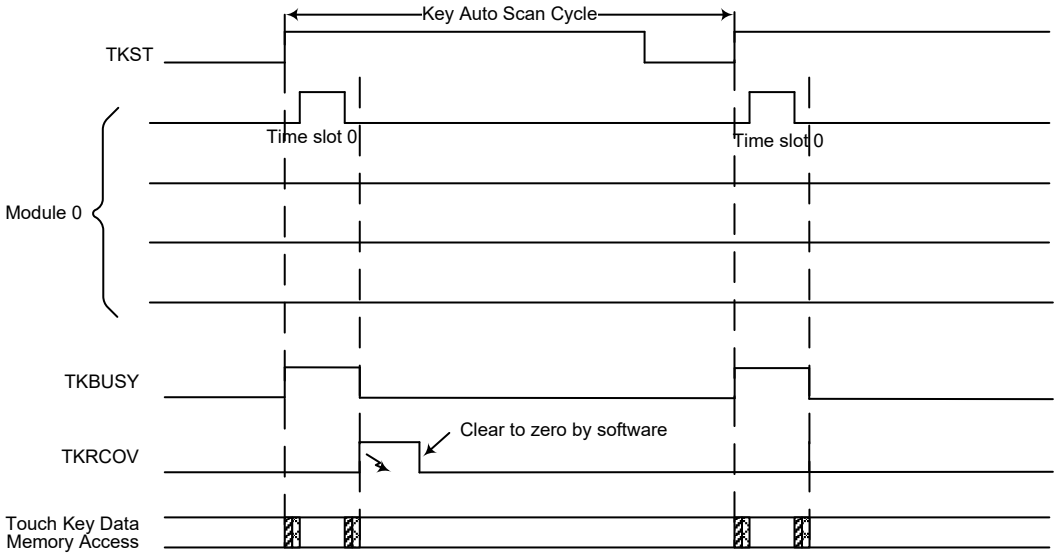


: Set Touch Key interrupt request flag

: Read 2N bytes from Touch Key Data Memory to TKM0ROH/TKM0ROL registers

: Write 2N bytes from TKM016DH/TKM016DL registers to Touch Key Data Memory

Touch Key Auto Scan Mode Timing Diagram – TSC=0



: Set Touch Key interrupt request flag

: Read 2N bytes from Touch Key Data Memory to TKM0ROH/TKM0ROL registers

: Write 2N bytes from TKM016DH/TKM016DL registers to Touch Key Data Memory

Touch Key Auto Scan Mode Timing Diagram – TSC=1

In the auto scan mode the key oscillator and reference oscillator will automatically be enabled when the TKST bit is set from low to high and disabled automatically when the TKBUSY bit changes from high to low. If the TSC bit is cleared to low, 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 TKM0ROH/TKM0ROL registers. Then the 16-bit C/F counter value will be written into the corresponding location of the time slot 3 scanned key in the touch key data memory. After this, the selected key will start to be scanned in time slot 0. At the end of the time slot 0 key scan operation, the reference oscillator internal capacitor value for the next selected key will be read from the touch key data memory and loaded into the TKM0ROH/TKM0ROL 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 TKM0ROH/TKM0ROL 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 selected keys are all 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, 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.

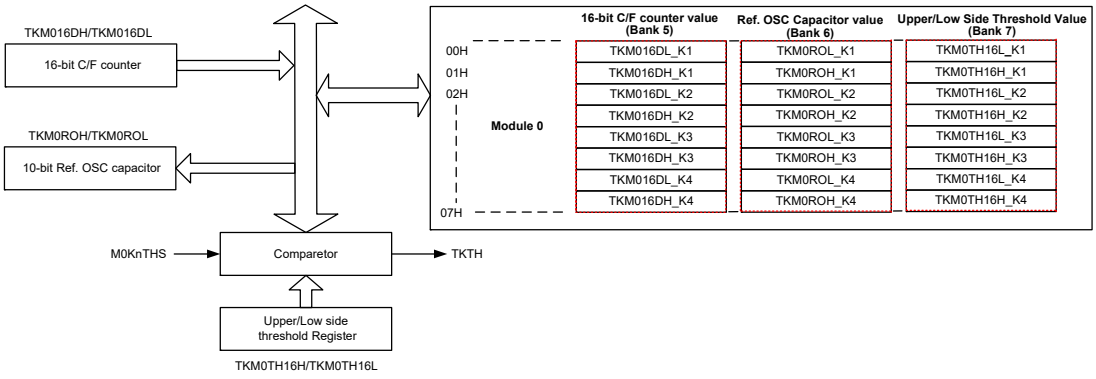
In the periodic auto scan mode the touch key scan operation will be implemented automatically on a periodic basis, which can be determined by the ASMP1~ASMP0 bits in the TKC2 register. The number of touch key scan times depends upon the WDT time-out period and the periodic auto scan mode period. Each 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 TKM0ROH/TKM0ROL registers. However, only at the end of the last scan operation in the WDT 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 TKM0TH16H/TKM0TH16L registers before the selected key will start to be scanned in time slot 0. The TKM0TH16H/TKM0TH16L 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. 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 M0KnTHS=0, or larger than the upper threshold if M0KnTHS=1, this indicates that the touch key state changes, then the M0KnTHF flag will be set high by the hardware, and an interrupt signal will be generated. Note that if the touch key threshold TKTH interrupt occurs, 1-byte data will be written to the TKM0ROL register because the TKM0ROH/TKM0ROL 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, TKM016DH/TKM016DL, and the TKM0TH16H/TKM0TH16L value are compared at the same time.

As the periodic auto scan operation is implemented using the WDT counter clock to reduce power consumption, when the WDT is cleared the WDT counter will be reset, the periodic auto scan operation time will be affected but the number of touch key times will not be affected.

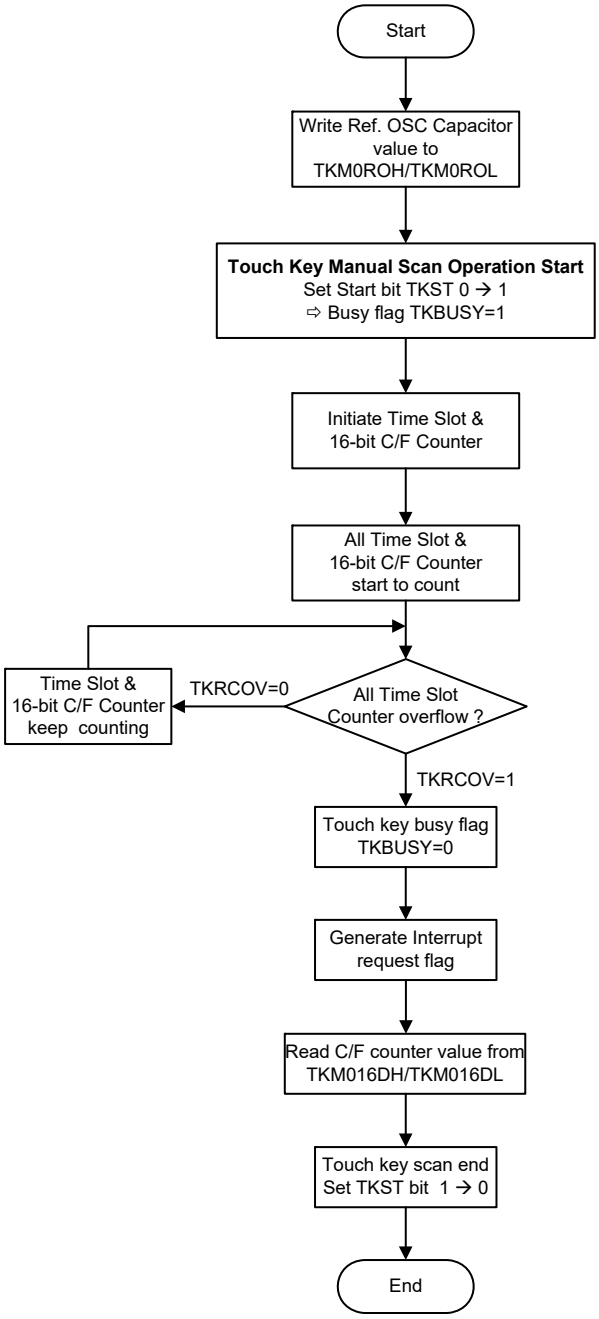
Touch Key Data Memory

The device provides three dedicated Data Memory area. The first area is used to store the 16-bit C/F counter values of the touch key module and located in Data Memory Bank 5, The second area is used to store the reference oscillator internal capacitor values of the touch key module and located in Data Memory Bank 6. The last area is used to store the 16-bit upper/lower threshold value of the touch key module and located in Data Memory Bank 7.

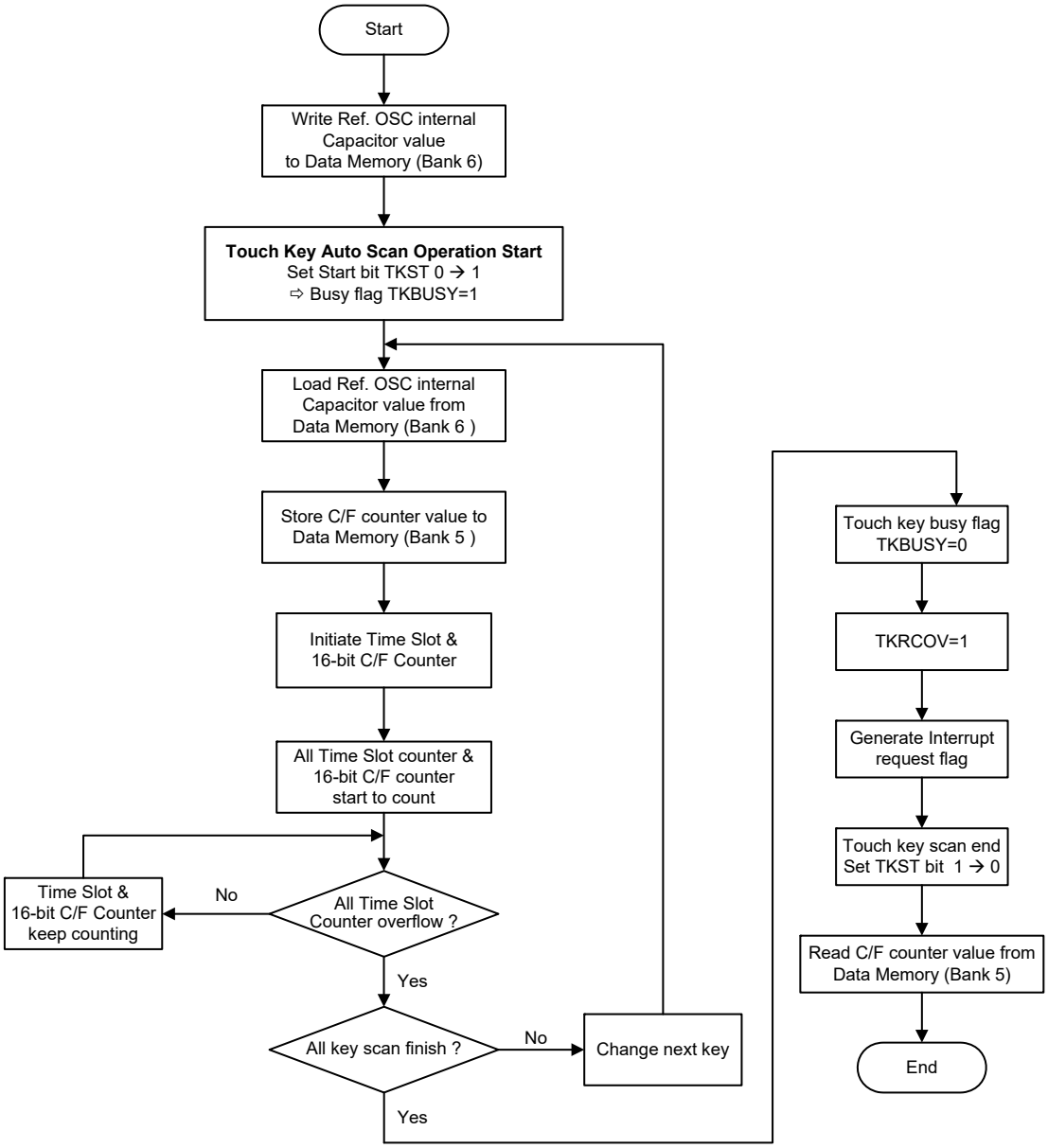


Touch Key Register Memory Map

Touch Key Scan Operation Flowchart



Touch Key Manual Scan Mode Flowchart



Touch Key Auto Scan Mode Flowchart

Touch Key Interrupts

The touch key has two independent interrupts, known as touch key TKRCOV interrupt and touch key threshold TKTH interrupt. In the manual scan mode, when the touch key module 0 time slot counter 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, after the last scan operation in the WDT 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 M0KnTHS=0, or larger than the upper threshold if M0KnTHS=1, a touch key threshold 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 WDT 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.

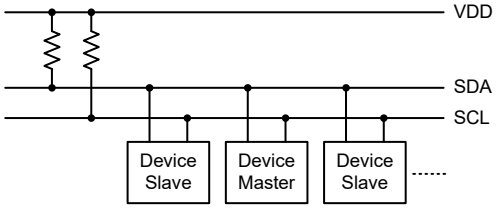
The TKCFOV flag which is the 16-bit C/F counter overflow flag will go high when any of the Touch Key Module 0 16-bit C/F counter overflows. As this flag will not be automatically cleared, it has to be cleared by the application program.

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.

I²C Interface – BS83B04L

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.



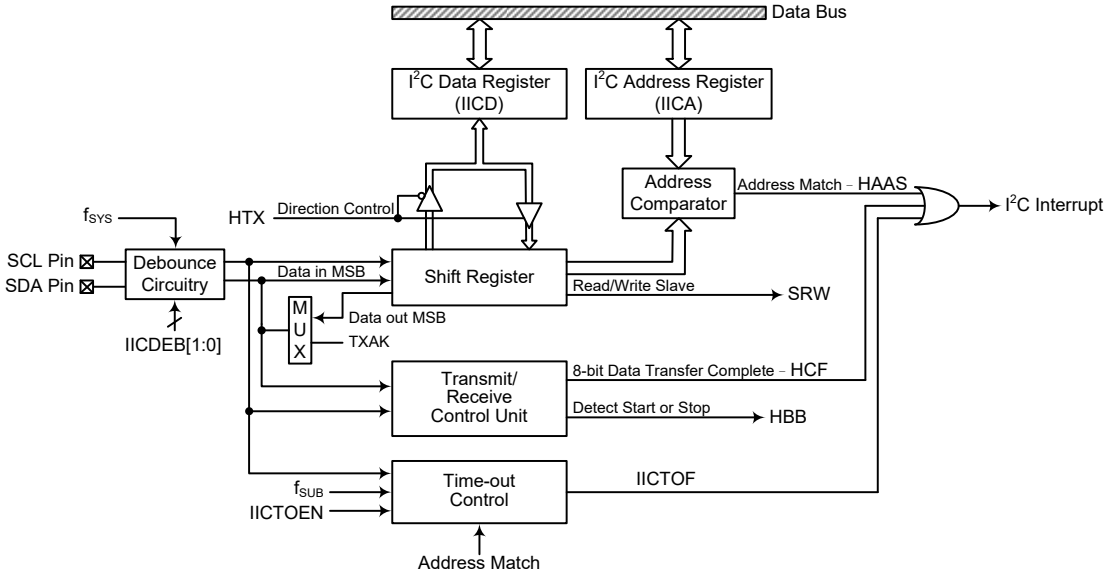
I²C Master Slave Bus Connection

I²C Interface Operation

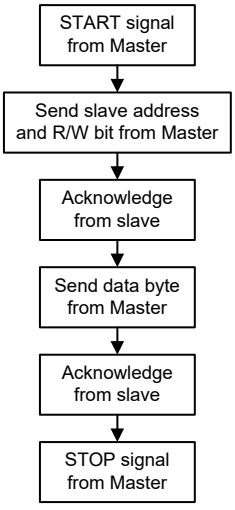
The I²C serial interface is a two-line interface, a serial data line, SDA, and serial clock line, SCL. As many devices may be connected together on the same bus, their outputs are both open drain types. For this reason, it is necessary that external pull-high resistors are connected to these outputs. 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 IICDEB1 and IICDEB0 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 Requirement

I²C Registers

There are three control registers associated with the I²C bus, IICC0, IICC1 and IICTOC, one address register IICA and one data register, IICD.

Register Name	Bit							
	7	6	5	4	3	2	1	0
IICC0	—	—	—	—	IICDEB1	IICDEB0	IICEN	—
IICC1	HCF	HAAS	HBB	HTX	TXAK	SRW	IAMWU	RXAK
IICD	D7	D6	D5	D4	D3	D2	D1	D0
IICA	IICA6	IICA5	IICA4	IICA3	IICA2	IICA1	IICA0	—
IICTOC	IICTOEN	IICTOF	IICTOS5	IICTOS4	IICTOS3	IICTOS2	IICTOS1	IICTOS0

I²C Register List

I²C Data Register

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

- **IICD 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**: I²C data register bit 7 ~ bit 0

I²C Address Register

The IICA register is the location where the 7-bit slave address of the slave device is stored. Bits 7~1 of the IICA 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 IICA register, the slave device will be selected.

• **IICA Register**

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

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

Bit 0 Unimplemented, read as "0"

I²C Control Registers

There are three control registers for the I²C interface, IICC0, IICC1 and IICTOC. The IICC0 register is used to control the enable/disable function and to set the data transmission clock frequency. The IICC1 register contains the relevant flags which are used to indicate the I²C communication status. Another register, IICTOC, is used to control the I²C time-out function and is described in the corresponding section.

• **IICC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	IICDEB1	IICDEB0	IICEN	—
R/W	—	—	—	—	R/W	R/W	R/W	—
POR	—	—	—	—	0	0	0	—

Bit 7~4 Unimplemented, read as "0"

Bit 3~2 **IICDEB1~IICDEB0**: I²C Debounce Time Selection
00: No debounce
01: 2 system clock debounce
10/11: 4 system clock debounce
Note that the I²C debounce circuit will operate normally if the system clock, f_{SYS}, is derived from the f_H clock or the IAMWU bit is equal to 0. Otherwise, the debounce circuit will have no effect and be bypassed.

Bit 1 **IICEN**: I²C Enable Control
0: Disable
1: Enable
The bit is the overall on/off control for the I²C interface. When the IICEN bit is cleared to zero to disable the I²C interface, the SDA and SCL lines will lose their I²C function and the I²C operating current will be reduced to a minimum value. When the bit is high the I²C interface is enabled. If the IICEN 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 Unimplemented, read as "0"

• **IICC1 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 set 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 to 1 to enable the I²C address match wake up from the SLEEP or IDLE Mode. If the IAMWU bit has been set before entering either the SLEEP or IDLE mode to enable the I²C address match wake up, then this bit must be cleared by the application program after wake-up to ensure correction device operation.

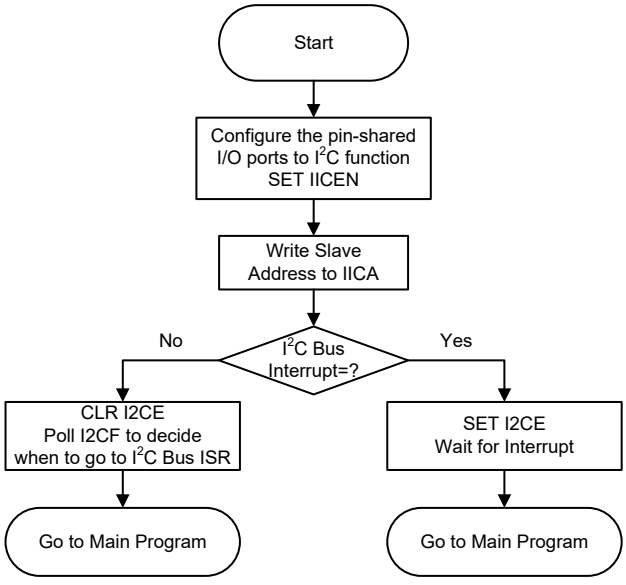
Bit 0 **RXAK**: I²C Bus Receive acknowledge flag
0: Slave receive acknowledge flag
1: Slave does not receive acknowledge flag

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 IICC1 register will be set and an I²C interrupt will be generated. After entering the interrupt service routine, the slave device must first check the condition of the HAAS and IICTOF 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
  Configure the corresponding pin-shared function as the I²C functional pins and set the IICEN bit in the IICC0 register to "1" to enable the I²C bus.
- Step 2
  Write the slave address of the device to the I²C bus address register IICA.
- Step 3
  Set the I2CE interrupt enable bit of the interrupt control register to enable the I²C interrupt.



I²C Bus Initialisation Flowchart

I²C Bus Start Signal

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

I²C Slave Address

The transmission of a START signal by the master will be detected by all devices on the I²C bus. To determine which slave device the master wishes to communicate with, the address of the slave device will be sent out immediately following the START signal. All slave devices, after receiving this 7-bit address data, will compare it with their own 7-bit slave address. If the address sent out by the master matches the internal address of the microcontroller slave device, then an internal I²C bus interrupt signal will be generated. The next bit following the address, which is the 8th bit, defines the read/write status and will be saved to the SRW bit of the IICC1 register. The slave device will then transmit an acknowledge bit, which is a low level, as the 9th bit. The slave device will also set the status flag HAAS when the addresses match.

As an I²C bus interrupt can come from three sources, when the program enters the interrupt subroutine, the HAAS and IICTOF 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 IICD register, or in the receive mode where it must implement a dummy read from the IICD register to release the SCL line.

I²C Bus Read/Write Signal

The SRW bit in the IICC1 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

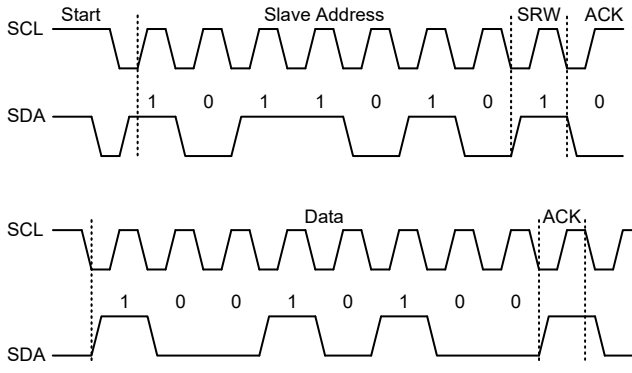
After the master has transmitted a calling address, any slave device on the I²C bus, whose own internal address matches the calling address, must generate an acknowledge signal. The acknowledge signal will inform the master that a slave device has accepted its calling address. If no acknowledge signal 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 IICC1 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 IICC1 register should be set 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 IICD register. If setup as a transmitter, the slave device must first write the data to be transmitted into the IICD register. If setup as a receiver, the slave device must read the transmitted data from the IICD 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 IICC1 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.

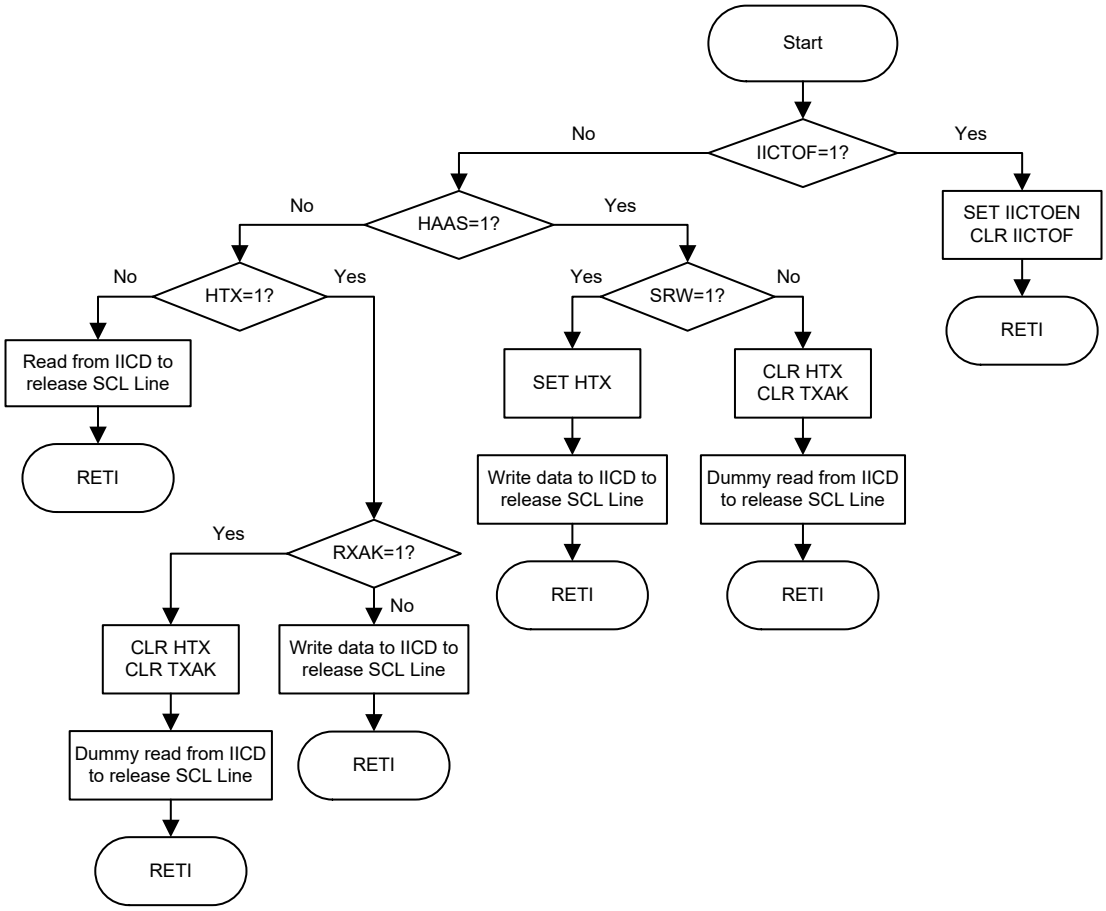


S=Start (1 bit)
SA=Slave Address (7 bits)
SR=SRW bit (1 bit)
M=Slave device send acknowledge bit (1 bit)
D=Data (8 bits)
A=ACK (RXAK bit for transmitter, TXAK bit for receiver, 1 bit)
P=Stop (1 bit)

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

I²C Communication Timing Diagram

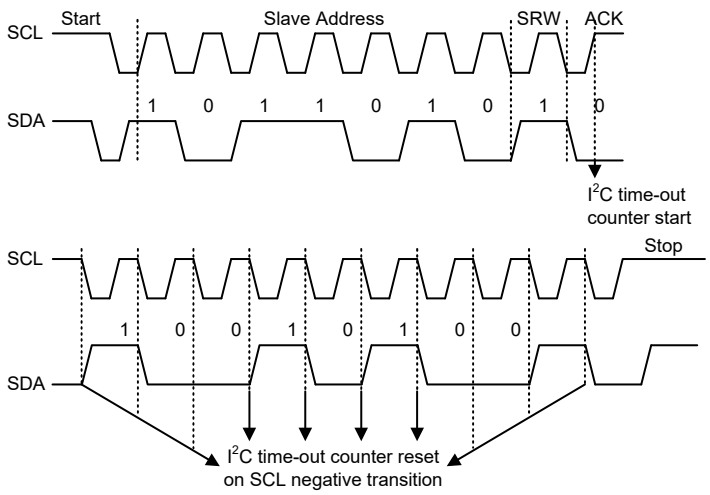
Note: When a slave address is matched, the device must be placed in either the transmit mode and then write data to the IICD register, or in the receive mode where it must implement a dummy read from the IICD register to release the SCL line.



I²C Bus ISR Flowchart

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

Registers	After I²C Time-out
IICD, IICA, IICC0	No change
IICC1	Reset to POR condition

I²C Registers after Time-out

The IICTOF flag can be cleared by the application program. There are 64 time-out periods which can be selected using IICTOS bit field in the IICTOC register. The time-out time is given by the formula: $[(1\sim64)\times32]/f_{SUB}$. This gives a time-out period which ranges from about 1ms to 64ms.

- **IICTOC Register**

Bit	7	6	5	4	3	2	1	0
Name	IICTOEN	IICTOF	IICTOS5	IICTOS4	IICTOS3	IICTOS2	IICTOS1	IICTOS0
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 **IICTOEN**: I²C Time-out control
0: Disable
1: Enable

Bit 6 **IICTOF**: I²C Time-out flag
0: No time-out occurred
1: Time-out occurred

Bit 5~0 **IICTOS5~IICTOS0**: I²C Time-out period selection
I²C time-out clock source is $f_{SUB}/32$.
I²C time-out time is equal to $(IICTOS[5:0]+1)\times(32/f_{SUB})$.

Interrupts

Interrupts are an important part of any microcontroller system. When an external event or an internal function such as a Touch action requires microcontroller attention, its corresponding interrupt will enforce a temporary suspension of the main program allowing the microcontroller to direct attention to their respective needs. The device contains several external interrupt and internal interrupt functions. The external interrupts are generated by the action of the external INT pin, while the internal interrupts are generated by various internal functions such as the Touch Keys, Timer/Event Counter, TM, EEPROM, Time Base and I²C etc.

Interrupt Registers

Overall interrupt control, which basically means the setting of request flags when certain microcontroller conditions occur and the setting of interrupt enable bits by the application program, is controlled by a series of registers, located in the Special Purpose Data Memory, as shown in the accompanying table. The number of registers falls into three categories. The first is the INTC0~INTC1 registers which setup the primary interrupts, the second is the MFI0~MFI1 registers which setup the Multi-function interrupts. Finally there is an INTEG register to setup the external interrupt trigger edge type.

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

Function	Enable Bit	Request Flag	Notes
Global	EMI	—	—
INT Pin	INTE	INTF	—
Time Base	TBE	TBF	—
Touch key TKRCOV	TKRCOVE	TKRCOVF	—
Touch key threshold TKTH	TKTHE	TKTHF	—
Timer/Event Counter	TE	TF	For BS83A02L only
I²C	I2CE	I2CF	For BS83B04L only
EEPROM Write Operation	DEE	DEF	For BS83B04L only
Multi-function	MFnE	MFnF	For BS83B04L only, n=0~1
CTM	CTMPE	CTMPF	For BS83B04L only
	CTMAE	CTMAF	For BS83B04L only

Interrupt Register Bit Naming Conventions

Register Name	Bit							
	7	6	5	4	3	2	1	0
INTEG	—	—	—	—	—	—	INTS1	INTS0
INTC0 (BS83A02L)	—	TKTHF	TKRCOVF	INTF	TKTHE	TKRCOVE	INTE	EMI
INTC0 (BS83B04L)	—	MF1F	MF0F	INTF	MF1E	MF0E	INTE	EMI
INTC1 (BS83A02L)	—	—	TBF	TF	—	—	TBE	TE
INTC1 (BS83B04L)	—	DEF	TBF	I2CF	—	DEE	TBE	I2CE
MFI0 (BS83B04L)	—	—	TKTHF	TKRCOVF	—	—	TKTHE	TKRCOVE
MFI1 (BS83B04L)	—	—	CTMAF	CTMPF	—	—	CTMAE	CTMPE

Interrupt Register List

• **INTEG Register**

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

Bit 7~2 Unimplemented, read as "0"

Bit 1~0 **INTS1~INTS0**: Interrupt edge control for INT pin
00: Disable
01: Rising edge
10: Falling edge
11: Rising and falling edges

• **INTC0 Register – BS83A02L**

Bit	7	6	5	4	3	2	1	0
Name	—	TKTHF	TKRCOVF	INTF	TKTHE	TKRCOVE	INTE	EMI
R/W	—	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	0	0	0	0	0	0	0

Bit 7 Unimplemented, read as "0"

Bit 6 **TKTHF**: Touch key threshold TKTH interrupt request flag
0: No request
1: Interrupt request

Bit 5 **TKRCOVF**: Touch key TKRCOV interrupt request flag
0: No request
1: Interrupt request

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

Bit 3 **TKTHE**: Touch key threshold TKTH interrupt control
0: Disable
1: Enable

Bit 2 **TKRCOVE**: Touch key TKRCOV interrupt control
0: Disable
1: Enable

Bit 1 **INTE**: INT interrupt control
0: Disable
1: Enable

Bit 0 **EMI**: Global interrupt control
0: Disable
1: Enable

• **INTC0 Register – BS83B04L**

Bit	7	6	5	4	3	2	1	0
Name	—	MF1F	MF0F	INTF	MF1E	MF0E	INTE	EMI
R/W	—	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	0	0	0	0	0	0	0

Bit 7 Unimplemented, read as "0"

Bit 6 **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 **INTF**: INT interrupt request flag
0: No request
1: Interrupt request

Bit 3 **MF1E**: Multi-function interrupt 1 control
0: Disable
1: Enable

Bit 2 **MF0E**: Multi-function interrupt 0 control
0: Disable
1: Enable

Bit 1 **INTE**: INT interrupt control
0: Disable
1: Enable

Bit 0 **EMI**: Global interrupt control
0: Disable
1: Enable

• **INTC1 Register – BS83A02L**

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

Bit 7~6 Unimplemented, read as "0"

Bit 5 **TBF**: Time Base interrupt request flag
0: No request
1: Interrupt request

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

Bit 3~2 Unimplemented, read as "0"

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

Bit 1 **TE**: Timer/Event Counter interrupt control
0: Disable
1: Enable

• **INTC1 Register – BS83B04L**

Bit	7	6	5	4	3	2	1	0
Name	—	DEF	TBF	I2CF	—	DEE	TBE	I2CE
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 **DEF**: Data EEPROM interrupt request flag
0: No request
1: Interrupt request

Bit 5 **TBF**: Time Base interrupt request flag
0: No request
1: Interrupt request

Bit 4 **I2CF**: I^2C interrupt request flag
0: No request
1: Interrupt request

Bit 3 Unimplemented, read as "0"

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

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

Bit 0 **I2CE**: I^2C interrupt control
0: Disable
1: Enable

• **MFI0 Register – BS83B04L**

Bit	7	6	5	4	3	2	1	0
Name	—	—	TKTHF	TKRCOVF	—	—	TKTHE	TKRCOVE
R/W	—	—	R/W	R/W	—	—	R/W	R/W
POR	—	—	0	0	—	—	0	0

Bit 7~6 Unimplemented, read as "0"

Bit 5 **TKTHF**: Touch key threshold 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 3~2 Unimplemented, read as "0"

Bit 1 **TKTHE**: Touch key threshold TKTH interrupt control
0: Disable
1: Enable

Bit 0 **TKRCOVE**: Touch key TKRCOV interrupt control
0: Disable
1: Enable

• **MFI1 Register – BS83B04L**

Bit	7	6	5	4	3	2	1	0
Name	—	—	CTMAF	CTMPF	—	—	CTMAE	CTMPE
R/W	—	—	R/W	R/W	—	—	R/W	R/W
POR	—	—	0	0	—	—	0	0

Bit 7~6 Unimplemented, read as "0"

Bit 5 **CTMAF**: CTM Comparator A match interrupt request flag
0: No request
1: Interrupt request

Bit 4 **CTMPF**: CTM Comparator P match interrupt request flag
0: No request
1: Interrupt request

Bit 3~2 Unimplemented, read as "0"

Bit 1 **CTMAE**: CTM Comparator A match interrupt control
0: Disable
1: Enable

Bit 0 **CTMPE**: CTM Comparator P match interrupt control
0: Disable
1: Enable

Interrupt Operation

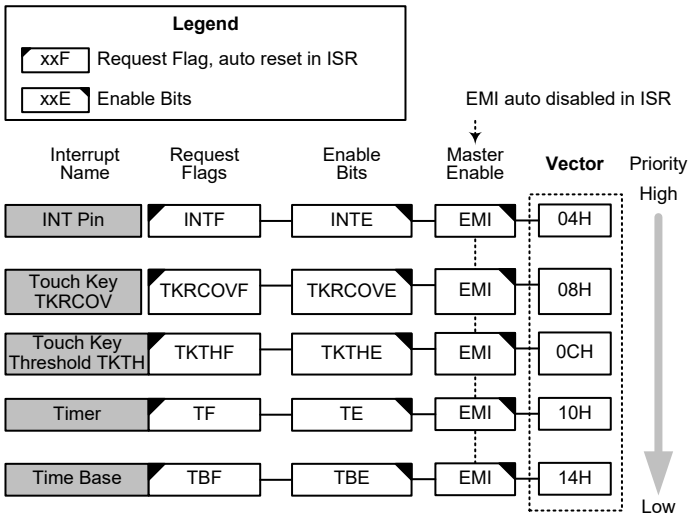
When the conditions for an interrupt event occur, such as a Touch Key Counter overflow etc., the relevant interrupt request flag will be set. Whether the request flag actually generates a program jump to the relevant interrupt vector is determined by the condition of the interrupt enable bit. If the enable bit is set high, then the program will jump to its relevant vector; if the enable bit is zero then although the interrupt request flag is set an actual interrupt will not be generated and the program will not jump to the relevant interrupt vector. The global interrupt enable bit, if cleared to zero, will disable all interrupts.

When an interrupt is generated, the Program Counter, which stores the address of the next instruction to be executed, will be transferred onto the stack. The Program Counter will then be loaded with a new address which will be the value of the corresponding interrupt vector. The microcontroller will then fetch its next instruction from this interrupt vector. The instruction at this vector will usually be a "JMP" 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.

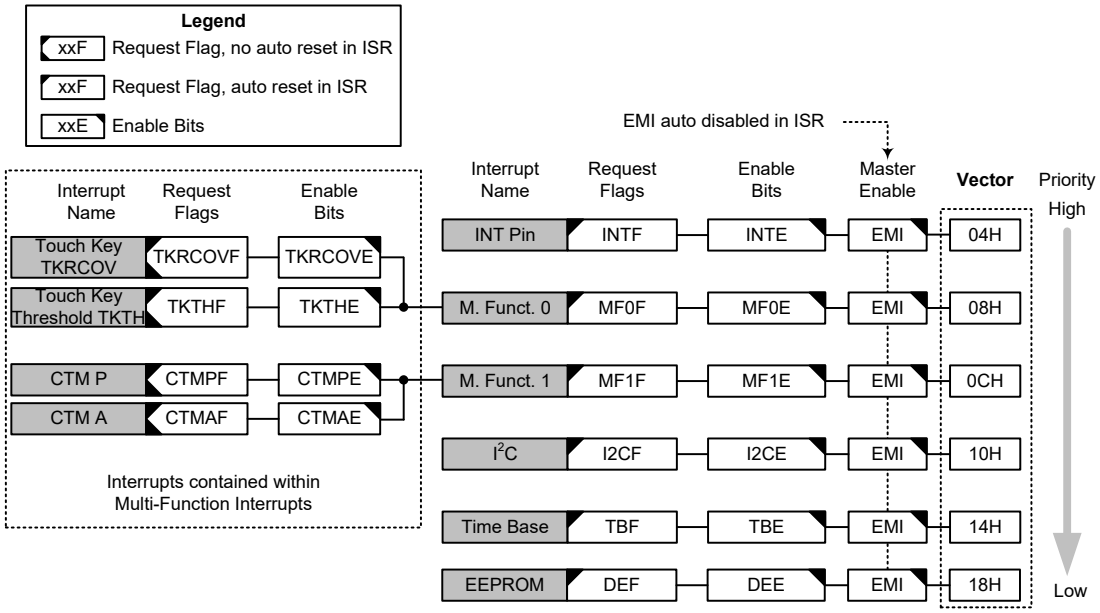
The various interrupt enable bit, together with their associated request flags, are shown in the accompanying diagrams with their order of priority. Some interrupt sources have their own individual vector while others share the same multi-function interrupt vector. Once an interrupt subroutine is serviced,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

If an interrupt requires immediate servicing while the program is already in another interrupt service routine, the EMI bit should be set after entering the routine, to allow interrupt nesting. If the stack is full, the interrupt request will not be acknowledged, even if the related interrupt is enabled, until the Stack Pointer is decremented. If immediate service is desired, the stack must be prevented from

becoming full. In case of simultaneous requests, the accompanying diagram shows the priority that is applied. All of the interrupt request flags when set will wake-up the device if it is in SLEEP or IDLE Mode, however to prevent a wake-up from occurring the corresponding flag should be set before the device is in SLEEP or IDLE Mode.



Interrupt Structure – BS83A02L



Interrupt Structure – BS83B04L

External Interrupt

The external interrupt is controlled by signal transitions on the INT pin. An external interrupt request will take place when the external interrupt request flag, INTF, is set, which will occur when a transition, whose type is chosen by the edge select bits, appears on the external interrupt pin. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and respective external interrupt enable bit, INTE, must first be set. Additionally,

the correct interrupt edge type must be selected using the INTEG register to enable the external interrupt function and to choose the trigger edge type. As the external interrupt pin is pin-shared with I/O pin, it can only be configured as external interrupt pin if its external interrupt enable bit in the corresponding interrupt register has been set and the external interrupt pin is selected by the corresponding pin-shared function selection bits. The pin must also be setup as an input by setting the corresponding bit in the port control register.

When the interrupt is enabled, the stack is not full and the correct transition type appears on the external interrupt pin, a subroutine call to the external interrupt vector, will take place. When the interrupt is serviced, the external interrupt request flag, INTF, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts. Note that any pull-high resistor selections on the external interrupt pins will remain valid even if the pin is used as an external interrupt input.

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

EEPROM Interrupt

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

Timer/Event Counter Interrupt

An actual Timer/Event Counter interrupt will take place when the Timer/Event Counter request flag, TF, is set, which occurs when the Timer/Event Counter overflows. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and the Timer/Event Counter Interrupt enable bit, TE, must first be set. When the interrupt is enabled, the stack is not full and the Timer/Event Counter overflows, a subroutine call to its interrupt vector, will take place. When the interrupt is serviced, the Timer/Event Counter Interrupt flag, TF, will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts.

I²C Interrupt

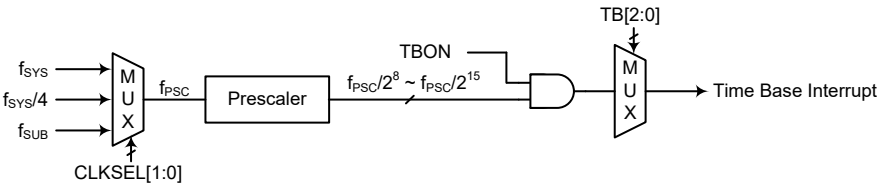
An I²C interrupt request will take place when the I²C Interrupt request flag, I2CF, is set, which occurs when a byte of data has been received or transmitted by the I²C interface, or an I²C slave address match occurs, or an I²C bus time-out occurs. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and the Serial Interface Interrupt enable bit, I2CE, 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 Serial Interface Interrupt flag, I2CF, will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts.

Time Base Interrupt

The function of the Time Base Interrupt is to provide regular time signal in the form of an internal interrupt. It is controlled by the overflow signals from their respective timer functions. When it

happens, its respective interrupt request flag, TBF will be set. To allow the program to branch to its respective interrupt vector addresses, the global interrupt enable bit, EMI and Time Base enable bit, TBE, must first be set. When the interrupt is enabled, the stack is not full and the Time Base overflows, a subroutine call to its respective vector location will take place. When the interrupt is serviced, the respective interrupt request flag, TBF, will be automatically reset and the EMI bit will be cleared to disable other interrupts.

The purpose of the Time Base Interrupt is to provide an interrupt signal at fixed time periods. Its clock source, f_{PSC}, originates from the internal clock source f_{SYS}, $f_{SYS}/4$ or f_{SUB} and then passes through a divider, the division ratio of which is selected by programming the appropriate bits in the TBC register to obtain longer interrupt periods whose value ranges. The clock source which in turn controls the Time Base interrupt period is selected using the CLKSEL1~CLKSEL0 bits in the PSCR register.



Time Base Interrupt

- **PSCR Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	CLKSEL1	CLKSEL0
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2 Unimplemented, read as "0"

Bit 1~0 **CLKSEL1~CLKSEL0**: Prescaler clock source selection
00: f_{SYS}
01: $f_{SYS}/4$
10/11: f_{SUB}

- **TBC Register**

Bit	7	6	5	4	3	2	1	0
Name	TBON	—	—	—	—	TB2	TB1	TB0
R/W	R/W	—	—	—	—	R/W	R/W	R/W
POR	0	—	—	—	—	0	0	0

Bit 7 **TBON**: Time Base Control
0: Disable
1: Enable

Bit 6~3 Unimplemented, read as "0"

Bit 2~0 **TB2~TB0**: Select Time Base Time-out Period selection
000: $2^8/f_{PSC}$
001: $2^9/f_{PSC}$
010: $2^{10}/f_{PSC}$
011: $2^{11}/f_{PSC}$
100: $2^{12}/f_{PSC}$
101: $2^{13}/f_{PSC}$
110: $2^{14}/f_{PSC}$
111: $2^{15}/f_{PSC}$

Multi-function Interrupt

Within the BS83B04L there are two 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 touch key TKRCOV interrupt, touch key threshold TKTH interrupt and TM interrupts.

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

However, it must be noted that, although the Multi-function Interrupt 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.

Touch Key TKRCOV Interrupt

For the BS83A02L, the Touch Key TKRCOV Interrupt is an individual interrupt source with its own interrupt vector. A Touch Key TKRCOV Interrupt request will take place when the Touch Key TKRCOV Interrupt request flag, TKRCOVF, is set, which occurs when the touch key time slot counter overflow flag is set high. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, Touch Key TKRCOV Interrupt enable bit, TKRCOVE, must first be set. When the interrupt is enabled, the stack is not full and the touch key time slot counter overflows or all the scan operations are completed, 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.

For the BS83B04L, the Touch Key TKRCOV Interrupt is contained within the Multi-function 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 touch key time slot counter overflows or all the scan operations are completed. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, Touch Key TKRCOV Interrupt enable bit, TKRCOVE, and associated Multi-function interrupt enable bit, must first be set. When the interrupt is enabled, the stack is not full and the touch key time slot counter overflows, a subroutine calls to the Multi-function Interrupt vector, will take place. When the Touch Key TKRCOV 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 TKRCOVF flag will not be automatically cleared, it has to be cleared by the application program.

Touch Key Threshold TKTH Interrupt

For the BS83A02L, the Touch Key Threshold TKTH Interrupt is an individual interrupt source with its own interrupt vector. A Touch Key Threshold TKTH Interrupt request will take place when the Touch Key Threshold TKTH Interrupt request flag, TKTHF, is set, which occurs when the Touch Key Module 0 16-bit C/F counter is less than the lower threshold if M0KnTHS=0, or larger than the upper threshold if M0KnTHS=1. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, Touch Key Threshold 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 Threshold TKTH Interrupt request flag, TKTHF, will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts.

For the BS83B04L, the Touch Key Threshold TKTH Interrupt is contained within the Multi-function Interrupt. A Touch Key Threshold TKTH Interrupt request will take place when the Touch Key Threshold 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 M0KnTHS=0, or larger than the upper threshold if M0KnTHS=1. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, Touch Key Threshold TKTH Interrupt enable bit, TKTHE, and associated Multi-function interrupt enable bit, 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 calls to the Multi-function Interrupt vector, will take place. When the Touch Key Threshold TKTH 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 TKTHF flag will not be automatically cleared, it has to be cleared by the application program.

TM Interrupt

The Compact Type TM 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. There are two interrupt request flags and two enable control bits. A TM interrupt request will take place when any of the TM request flags are set, a situation which occurs when a TM comparator P or A match situation happens.

To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, respective TM Interrupt enable bit, and relevant Multi-function Interrupt enable bit, MFnE, must first be set. When the interrupt is enabled, the stack is not full and a TM comparator match situation occurs, a subroutine call to the relevant Multi-function Interrupt vector locations, will take place. When the TM interrupt is serviced, the EMI bit will be automatically cleared to disable other interrupts, however only the related MFnF flag will be automatically cleared. As the TM interrupt request flags will not be automatically cleared, they have to be cleared by the application program.

Interrupt Wake-up Function

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

Programming Considerations

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

Where a certain interrupt is contained within a Multi-function interrupt, then when the interrupt

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

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

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

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

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

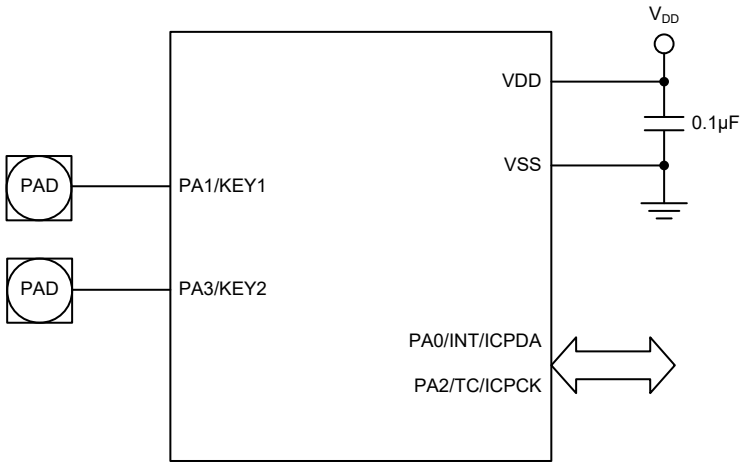
Configuration Options – BS83B04L

Configuration options refer to certain options within the BS83B04L that are programmed into the device during the programming process. During the development process, these options are selected using the HT-IDE software development tools. As these options are programmed into the device using the hardware programming tools, once they are selected they cannot be changed later using the application program. All options must be defined for proper system function, the details of which are shown in the table.

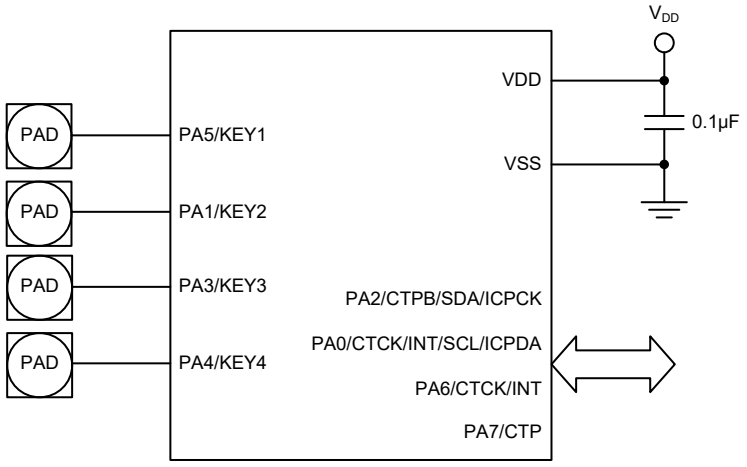
No.	Options
Oscillator Option	
1	HIRC frequency selection – f_{HIRC}: 2MHz, 4MHz or 8MHz

Note: When the HIRC has been configured at a frequency shown in this table, the HIRC1~HIRC0 bits should also be setup to select the same frequency to achieve the HIRC frequency accuracy specified in the A.C. Characteristics.

Application Circuits



BS83A02L



BS83B04L

Instruction Set

Introduction

Central to the successful operation of any microcontroller is its instruction set, which is a set of program instruction codes that directs the microcontroller to perform certain operations. In the case of Holtek microcontroller, a comprehensive and flexible set of over 60 instructions is provided to enable programmers to implement their application with the minimum of programming overheads.

For easier understanding of the various instruction codes, they have been subdivided into several functional groupings.

Instruction Timing

Most instructions are implemented within one instruction cycle. The exceptions to this are branch, call, or table read instructions where two instruction cycles are required. One instruction cycle is equal to 4 system clock cycles, therefore in the case of an 8MHz system oscillator, most instructions would be implemented within 0.5μs and branch or call instructions would be implemented within 1μs. Although instructions which require one more cycle to implement are generally limited to the JMP, CALL, RET, RETI and table read instructions, it is important to realize that any other instructions which involve manipulation of the Program Counter Low register or PCL will also take one more cycle to implement. As instructions which change the contents of the PCL will imply a direct jump to that new address, one more cycle will be required. Examples of such instructions would be "CLR PCL" or "MOV PCL, A". For the case of skip instructions, it must be noted that if the result of the comparison involves a skip operation then this will also take one more cycle, if no skip is involved then only one cycle is required.

Moving and Transferring Data

The transfer of data within the microcontroller program is one of the most frequently used operations. Making use of three kinds of MOV instructions, data can be transferred from registers to the Accumulator and vice-versa as well as being able to move specific immediate data directly into the Accumulator. One of the most important data transfer applications is to receive data from the input ports and transfer data to the output ports.

Arithmetic Operations

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

Logical and Rotate Operation

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

Branches and Control Transfer

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

Bit Operations

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

Table Read Operations

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

Other Operations

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

Instruction Set Summary

The following table depicts a summary of the instruction set categorised according to function and can be consulted as a basic instruction reference using the following listed conventions.

Table Conventions

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

Mnemonic	Description	Cycles	Flag Affected
Arithmetic			
ADD A,[m]	Add Data Memory to ACC	1	Z, C, AC, OV
ADDM A,[m]	Add ACC to Data Memory	1Note	Z, C, AC, OV
ADD A,x	Add immediate data to ACC	1	Z, C, AC, OV
ADC A,[m]	Add Data Memory to ACC with Carry	1	Z, C, AC, OV
ADCM A,[m]	Add ACC to Data memory with Carry	1Note	Z, C, AC, OV
SUB A,x	Subtract immediate data from the ACC	1	Z, C, AC, OV
SUB A,[m]	Subtract Data Memory from ACC	1	Z, C, AC, OV
SUBM A,[m]	Subtract Data Memory from ACC with result in Data Memory	1Note	Z, C, AC, OV
SBC A,[m]	Subtract Data Memory from ACC with Carry	1	Z, C, AC, OV
SBCM A,[m]	Subtract Data Memory from ACC with Carry, result in Data Memory	1Note	Z, C, AC, OV
DAA [m]	Decimal adjust ACC for Addition with result in Data Memory	1Note	C
Logic Operation			
AND A,[m]	Logical AND Data Memory to ACC	1	Z
OR A,[m]	Logical OR Data Memory to ACC	1	Z
XOR A,[m]	Logical XOR Data Memory to ACC	1	Z
ANDM A,[m]	Logical AND ACC to Data Memory	1Note	Z
ORM A,[m]	Logical OR ACC to Data Memory	1Note	Z
XORM A,[m]	Logical XOR ACC to Data Memory	1Note	Z
AND A,x	Logical AND immediate Data to ACC	1	Z
OR A,x	Logical OR immediate Data to ACC	1	Z
XOR A,x	Logical XOR immediate Data to ACC	1	Z
CPL [m]	Complement Data Memory	1Note	Z
CPLA [m]	Complement Data Memory with result in ACC	1	Z
Increment & Decrement			
INCA [m]	Increment Data Memory with result in ACC	1	Z
INC [m]	Increment Data Memory	1Note	Z
DECA [m]	Decrement Data Memory with result in ACC	1	Z
DEC [m]	Decrement Data Memory	1Note	Z
Rotate			
RRA [m]	Rotate Data Memory right with result in ACC	1	None
RR [m]	Rotate Data Memory right	1Note	None
RRCA [m]	Rotate Data Memory right through Carry with result in ACC	1	C
RRC [m]	Rotate Data Memory right through Carry	1Note	C
RLA [m]	Rotate Data Memory left with result in ACC	1	None
RL [m]	Rotate Data Memory left	1Note	None
RLCA [m]	Rotate Data Memory left through Carry with result in ACC	1	C
RLC [m]	Rotate Data Memory left through Carry	1Note	C

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].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 or current page) to TBLH and Data Memory	2Note	None
TABRDL [m]	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
CLR WDT1	Pre-clear Watchdog Timer	1	TO, PDF
CLR WDT2	Pre-clear Watchdog Timer	1	TO, PDF
SWAP [m]	Swap nibbles of Data Memory	1Note	None
SWAPA [m]	Swap nibbles of Data Memory with result in ACC	1	None
HALT	Enter power down mode	1	TO, PDF

Note: 1. For skip instructions, if the result of the comparison involves a skip then up to two cycles are required, if no skip takes place only one cycle is required.
2. Any instruction which changes the contents of the PCL will also require 2 cycles for execution.
3. For the "CLR WDT1" and "CLR WDT2" instructions the TO and PDF flags may be affected by the execution status. The TO and PDF flags are cleared after both "CLR WDT1" and "CLR WDT2" instructions are consecutively executed. Otherwise the TO and PDF flags remain unchanged.

Instruction Definition

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

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

Operation ACC ← ACC + [m] + C

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

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

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

Operation [m] ← ACC + [m] + C

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

ADD A,[m] Add Data Memory to ACC

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

Operation ACC ← ACC + [m]

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

ADD A,x Add immediate data to ACC

Description The contents of the Accumulator and the specified immediate data are added. The result is stored in the Accumulator.

Operation ACC ← ACC + x

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

ADDM A,[m] Add ACC to Data Memory

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

Operation [m] ← ACC + [m]

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

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

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

Operation ACC ← ACC "AND" [m]

Affected flag(s) Z

AND A,x Logical AND immediate data to ACC

Description Data in the Accumulator and the specified immediate data perform a 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

CLR WDT1 Pre-clear Watchdog Timer

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

Operation WDT cleared
TO ← 0
PDF ← 0

Affected flag(s) TO, PDF

CLR WDT2 Pre-clear Watchdog Timer

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

Operation WDT cleared
TO ← 0
PDF ← 0

Affected flag(s) TO, PDF

CPL [m] Complement Data Memory

Description Each bit of the specified Data Memory is logically complemented (1′s complement). Bits which previously contained a 1 are changed to 0 and vice versa.

Operation [m] ← $\overline{[m]}$

Affected flag(s) Z

CPLA [m] Complement Data Memory with result in ACC

Description Each bit of the specified Data Memory is logically complemented (1′s complement). Bits which previously contained a 1 are changed to 0 and vice versa. The complemented result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

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

Affected flag(s) Z

DAA [m] Decimal-Adjust ACC for addition with result in Data Memory

Description Convert the contents of the Accumulator value to a BCD (Binary Coded Decimal) value resulting from the previous addition of two BCD variables. If the low nibble is greater than 9 or if AC flag is set, then a value of 6 will be added to the low nibble. Otherwise the low nibble remains unchanged. If the high nibble is greater than 9 or if the C flag is set, then a value of 6 will be added to the high nibble. Essentially, the decimal conversion is performed by adding 00H, 06H, 60H or 66H depending on the Accumulator and flag conditions. Only the C flag may be affected by this instruction which indicates that if the original BCD sum is greater than 100, it allows multiple precision decimal addition.

Operation [m] ← ACC + 00H or
[m] ← ACC + 06H or
[m] ← ACC + 60H or
[m] ← ACC + 66H

Affected flag(s) C

DEC [m] Decrement Data Memory

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

Operation [m] ← [m] − 1

Affected flag(s) Z

DECA [m] Decrement Data Memory with result in ACC

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

Operation ACC ← [m] − 1

Affected flag(s) Z

HALT Enter power down mode

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

Operation TO ← 0
PDF ← 1

Affected flag(s) TO, PDF

INC [m] Increment Data Memory

Description Data in the specified Data Memory is incremented by 1.

Operation [m] ← [m] + 1

Affected flag(s) Z

INCA [m] Increment Data Memory with result in ACC

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

Operation ACC ← [m] + 1

Affected flag(s) Z

JMP addr Jump unconditionally

Description The contents of the Program Counter are replaced with the specified address. Program execution then continues from this new address. As this requires the insertion of a dummy instruction while the new address is loaded, it is a two cycle instruction.

Operation Program Counter ← addr

Affected flag(s) None

MOV A,[m] Move Data Memory to ACC

Description The contents of the specified Data Memory are copied to the Accumulator.

Operation ACC ← [m]

Affected flag(s) None

MOV A,x Move immediate data to ACC

Description The immediate data specified is loaded into the Accumulator.

Operation ACC ← x

Affected flag(s) None

MOV [m],A Move ACC to Data Memory

Description The contents of the Accumulator are copied to the specified Data Memory.

Operation [m] ← ACC

Affected flag(s) None

NOP No operation

Description No operation is performed. Execution continues with the next instruction.

Operation No operation

Affected flag(s) None

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

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

Operation ACC ← ACC ″OR″ [m]

Affected flag(s) Z

OR A,x Logical OR immediate data to ACC

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

Operation ACC ← ACC ″OR″ x

Affected flag(s) Z

ORM A,[m] Logical OR ACC to Data Memory

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

Operation [m] ← ACC ″OR″ [m]

Affected flag(s) Z

RET Return from subroutine

Description The Program Counter is restored from the stack. Program execution continues at the restored address.

Operation Program Counter ← Stack

Affected flag(s) None

RET A,x Return from subroutine and load immediate data to ACC

Description The Program Counter is restored from the stack and the Accumulator loaded with the specified immediate data. Program execution continues at the restored address.

Operation Program Counter ← Stack
ACC ← x

Affected flag(s) None

RETI Return from interrupt

Description The Program Counter is restored from the stack and the interrupts are re-enabled by setting the EMI bit. EMI is the master interrupt global enable bit. If an interrupt was pending when the RETI instruction is executed, the pending Interrupt routine will be processed before returning to the main program.

Operation Program Counter ← Stack
EMI ← 1

Affected flag(s) None

RL [m] Rotate Data Memory left

Description The contents of the specified Data Memory are rotated left by 1 bit with bit 7 rotated into bit 0.

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

Affected flag(s) None

RLA [m] Rotate Data Memory left with result in ACC

Description The contents of the specified Data Memory are rotated left by 1 bit with bit 7 rotated into bit 0. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

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

Affected flag(s) None

RLC [m] Rotate Data Memory left through Carry

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

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

Affected flag(s) C

RLCA [m] Rotate Data Memory left through Carry with result in ACC

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

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

Affected flag(s) C

RR [m] Rotate Data Memory right

Description The contents of the specified Data Memory are rotated right by 1 bit with bit 0 rotated into bit 7.

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

Affected flag(s) None

RRA [m] Rotate Data Memory right with result in ACC

Description: Data in the specified Data Memory is rotated right by 1 bit with bit 0 rotated into bit 7. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

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

Affected flag(s): None

RRC [m] Rotate Data Memory right through Carry

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

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

Affected flag(s): C

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

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

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

Affected flag(s): C

SBC A,[m] Subtract Data Memory from ACC with Carry

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

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

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

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

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

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

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

SDZ [m] Skip if decrement Data Memory is 0

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

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

Affected flag(s): None

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

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

Operation ACC ← [m] − 1
Skip if ACC=0

Affected flag(s) None

SET [m] Set Data Memory

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

Operation [m] ← FFH

Affected flag(s) None

SET [m].i Set bit of Data Memory

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

Operation [m].i ← 1

Affected flag(s) None

SIZ [m] Skip if increment Data Memory is 0

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

Operation [m] ← [m] + 1
Skip if [m]=0

Affected flag(s) None

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

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

Operation ACC ← [m] + 1
Skip if ACC=0

Affected flag(s) None

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

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

Operation Skip if [m].i ≠ 0

Affected flag(s) None

SUB A,[m] Subtract Data Memory from ACC

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

Operation ACC ← ACC − [m]

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

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

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

Operation [m] ← ACC − [m]

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

SUB A,x Subtract immediate data from ACC

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

Operation ACC ← ACC − x

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

SWAP [m] Swap nibbles of Data Memory

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

Operation [m].3~[m].0 ↔ [m].7~[m].4

Affected flag(s) None

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

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

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

Affected flag(s) None

SZ [m] Skip if Data Memory is 0

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

Operation Skip if [m]=0

Affected flag(s) None

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

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

Operation ACC ← [m]
Skip if [m]=0

Affected flag(s) None

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

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

Operation Skip if [m].i=0

Affected flag(s) None

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

Description The low byte of the program code addressed by the table pointer (TBHP and TBLP or only TBLP if no 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

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

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

Operation ACC ← ACC ″XOR″ [m]

Affected flag(s) Z

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

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

Operation [m] ← ACC ″XOR″ [m]

Affected flag(s) Z

XOR A,x Logical XOR immediate data to ACC

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

Operation ACC ← ACC ″XOR″ x

Affected flag(s) Z

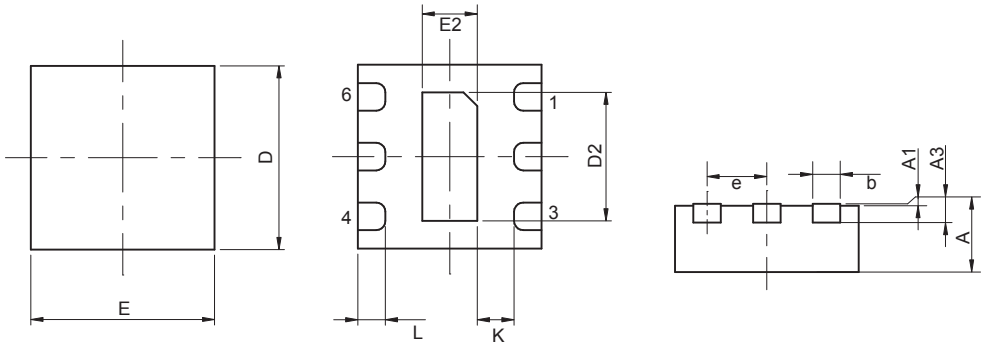
Package Information

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

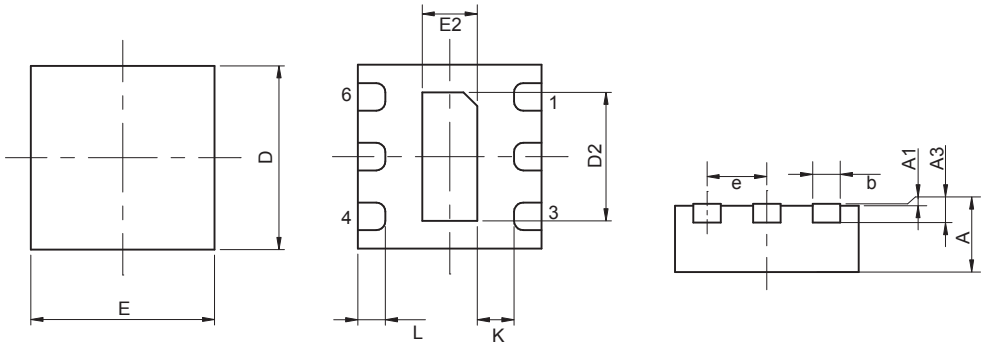
6-pin DFN (2mm×2mm×0.35mm) Outline Dimensions



Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	0.012	0.014	0.016
A1	0.000	0.001	0.002
A3	—	0.005 BSC	—
b	0.010	0.012	0.014
D	—	0.079 BSC	—
E	—	0.079 BSC	—
e	—	0.026 BSC	—
D2	0.053	0.055	0.057
E2	0.022	0.024	0.026
L	0.010	0.012	0.014
K	0.008	—	—

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	0.30	0.35	0.40
A1	0.00	0.02	0.05
A3	—	0.127 BSC	—
b	0.25	0.30	0.35
D	—	2.00 BSC	—
E	—	2.00 BSC	—
e	—	0.65 BSC	—
D2	1.35	1.40	1.45
E2	0.55	0.60	0.65
L	0.25	0.30	0.35
K	0.20	—	—

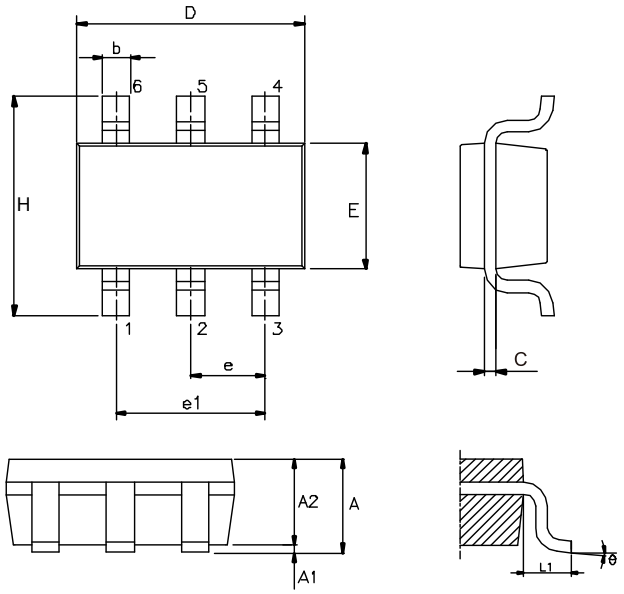
6-pin DFN (2mm×2mm×0.75mm) Outline Dimensions



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	0.010	0.012	0.014
D	—	0.079 BSC	—
E	—	0.079 BSC	—
e	—	0.026 BSC	—
D2	0.053	0.055	0.057
E2	0.022	0.024	0.026
L	0.010	0.012	0.014
K	0.008	—	—

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	0.700	0.750	0.800
A1	0.000	0.020	0.050
A3	—	0.200 BSC	—
b	0.250	0.300	0.350
D	—	2.00 BSC	—
E	—	2.00 BSC	—
e	—	0.65 BSC	—
D2	1.350	1.400	1.450
E2	0.550	0.600	0.650
L	0.250	0.300	0.350
K	0.200	—	—

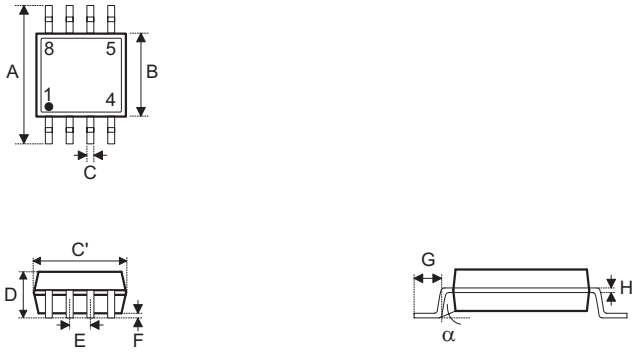
6-pin SOT23-6 Outline Dimensions



Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	—	—	0.057
A1	—	—	0.006
A2	0.035	0.045	0.051
b	0.012	—	0.020
C	0.003	—	0.009
D	—	0.114 BSC	—
E	—	0.063 BSC	—
e	—	0.037 BSC	—
e1	—	0.075 BSC	—
H	—	0.110 BSC	—
L1	—	0.024 BSC	—
θ	0°	—	8°

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	—	—	1.45
A1	—	—	0.15
A2	0.90	1.15	1.30
b	0.30	—	0.50
C	0.08	—	0.22
D	—	2.90 BSC	—
E	—	1.60 BSC	—
e	—	0.95 BSC	—
e1	—	1.90 BSC	—
H	—	2.80 BSC	—
L1	—	0.60 BSC	—
θ	0°	—	8°

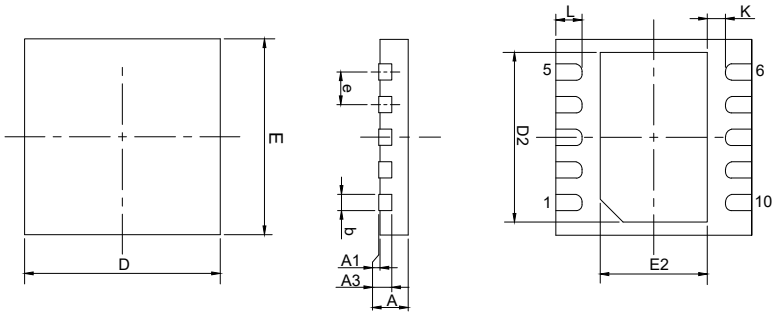
8-pin SOP (150mil) Outline Dimensions



Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	—	0.236 BSC	—
B	—	0.154 BSC	—
C	0.012	—	0.020
C′	—	0.193 BSC	—
D	—	—	0.069
E	—	0.050 BSC	—
F	0.004	—	0.010
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′	—	4.90 BSC	—
D	—	—	1.75
E	—	1.27 BSC	—
F	0.10	—	0.25
G	0.40	—	1.27
H	0.10	—	0.25
α	0°	—	8°

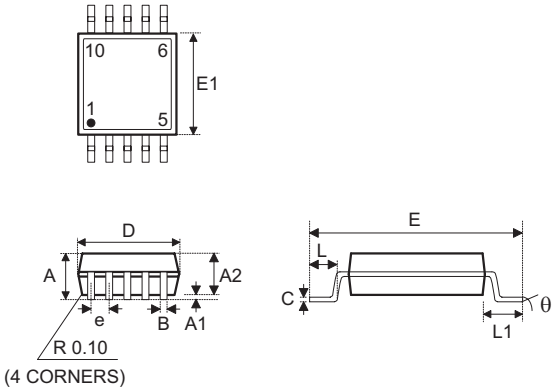
10-pin DFN (3mm×3mm×0.75mm) Outline Dimensions



Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	0.028	0.030	0.031
A1	0.000	0.001	0.002
A3	—	0.080 BSC	—
b	0.007	0.010	0.012
D	—	0.118 BSC	—
E	—	0.118 BSC	—
e	—	0.020 BSC	—
D2	0.087	0.091	0.093
E2	0.061	0.065	0.067
L	0.012	0.016	0.020
K	0.008	—	—

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	0.700	0.750	0.800
A1	0.000	0.020	0.050
A3	—	0.203 BSC	—
b	0.180	0.250	0.300
D	—	3.000 BSC	—
E	—	3.000 BSC	—
e	—	0.500 BSC	—
D2	2.200	2.300	2.350
E2	1.550	1.650	1.700
L	0.300	0.400	0.450
K	0.200	—	—

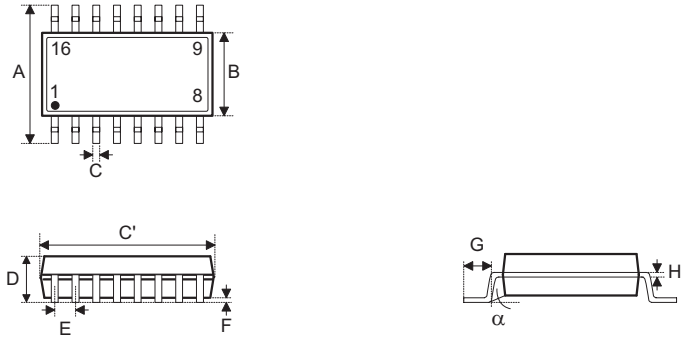
10-pin MSOP (118mil) Outline Dimensions



Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	—	—	0.043
A1	0.000	—	0.006
A2	0.030	0.033	0.037
B	0.007	—	0.013
C	0.003	—	0.009
D	—	0.118 BSC	—
E	—	0.193 BSC	—
E1	—	0.118 BSC	—
e	—	0.020 BSC	—
L	0.016	0.024	0.031
L1	—	0.037 BSC	—
y	—	0.004	—
θ	0°	—	8°

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	—	—	1.10
A1	0.00	—	0.15
A2	0.75	0.85	0.95
B	0.17	—	0.33
C	0.08	—	0.23
D	—	3.00 BSC	—
E	—	4.90 BSC	—
E1	—	3.00 BSC	—
e	—	0.50 BSC	—
L	0.40	0.60	0.80
L1	—	0.95 BSC	—
y	—	0.10	—
θ	0°	—	8°

16-pin NSOP (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
E	—	0.050 BSC	—
F	0.004	—	0.010
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
E	—	1.27 BSC	—
F	0.10	—	0.25
G	0.40	—	1.27
H	0.10	—	0.25
α	0°	—	8°