



---

Touch I/O OTP MCU

**BS23A02CA/BS23B04CA/BS23B08CA**

Revision: V1.10    Date: August 22, 2024

[www.holtek.com](http://www.holtek.com)

# Table of Contents

<b>Features .....</b>	<b>5</b>
CPU Features .....	5
Peripheral Features.....	5
<b>General Description.....</b>	<b>6</b>
<b>Selection Table.....</b>	<b>6</b>
<b>Block Diagram.....</b>	<b>7</b>
<b>Pin Assignment.....</b>	<b>8</b>
<b>Pin Description .....</b>	<b>10</b>
<b>Absolute Maximum Ratings.....</b>	<b>14</b>
<b>D.C. Characteristics.....</b>	<b>15</b>
Operating Voltage Characteristics.....	15
Standby Current Characteristics .....	15
Operating Current Characteristics.....	15
<b>A.C. Characteristics.....</b>	<b>16</b>
High Speed Internal Oscillator – HIRC – Frequency Accuracy .....	16
Low Speed Internal Oscillator Characteristics – LIRC .....	16
Operating Frequency Characteristic Curves .....	16
System Start Up Time Characteristics .....	17
<b>Input/Output Characteristics .....</b>	<b>17</b>
<b>Memory Characteristics .....</b>	<b>18</b>
<b>LVR Electrical Characteristics .....</b>	<b>18</b>
<b>I<sup>2</sup>C Electrical Characteristics .....</b>	<b>18</b>
<b>Power-on Reset Characteristics.....</b>	<b>19</b>
<b>System Architecture .....</b>	<b>20</b>
Clocking and Pipelining.....	20
Program Counter.....	21
Stack .....	21
Arithmetic and Logic Unit – ALU .....	22
<b>OTP Program Memory.....</b>	<b>22</b>
Structure.....	22
Special Vectors .....	23
Look-up Table.....	23
Table Program Example.....	24
In Circuit Programming – ICP .....	25
On-Chip Debug Support – OCDS .....	26
OTP ROM Parameter Program – ORPP – BS23B04CA and BS23B08CA.....	26
<b>Data Memory .....</b>	<b>29</b>
Structure.....	29
General Purpose Data Memory .....	30
Special Purpose Data Memory .....	30

<b>Special Function Register Description.....</b>	<b>34</b>
Indirect Addressing Register – IAR0, IAR1 .....	34
Memory Pointer – MP0, MP1 .....	34
Bank Pointer – BP – BS23B08CA.....	35
Accumulator – ACC.....	35
Program Counter Low Byte Register – PCL.....	35
Look-up Table Registers – TBLP, TBLH .....	35
Status Register – STATUS .....	36
<b>Oscillators .....</b>	<b>37</b>
Oscillator Overview .....	37
System Clock Configurations .....	37
Internal High Speed RC Oscillator – HIRC .....	38
Internal 32kHz Oscillator – LIRC.....	38
<b>Operating Modes and System Clocks .....</b>	<b>38</b>
System Clocks .....	38
System Operation Modes.....	39
Control Register .....	41
Operating Mode Switching .....	42
Standby Current Considerations .....	45
Wake-up .....	45
<b>Watchdog Timer.....</b>	<b>46</b>
Watchdog Timer Clock Source.....	46
Watchdog Timer Control Register .....	46
Watchdog Timer Operation .....	47
<b>Reset and Initialisation.....</b>	<b>48</b>
Reset Functions .....	48
Reset Initial Conditions .....	50
<b>Input/Output Ports .....</b>	<b>53</b>
Pull-high Resistors .....	54
Port A Wake-up .....	54
I/O Port Control Registers .....	55
Pin-shared Functions .....	55
I/O Pin Structure.....	60
Programming Considerations.....	60
<b>8-bit Timer/Event Counter – BS23B04CA and BS23B08CA .....</b>	<b>61</b>
Timer/Event Counter Input Clock Source.....	61
Timer/Event Counter Registers.....	61
Timer/Event Counter Operating Modes.....	64
<b>Touch Key Function .....</b>	<b>68</b>
Touch Key Structure.....	68
Touch Key Register Definition .....	69
Touch Key Operation.....	74
Touch Key Interrupt.....	75
Programming Considerations.....	75

<b>I<sup>2</sup>C Interface – BS23B04CA and BS23B08CA .....</b>	<b>76</b>
I <sup>2</sup> C Interface Operation.....	76
I <sup>2</sup> C Registers .....	77
I <sup>2</sup> C Bus Communication .....	80
I <sup>2</sup> C Time-out Control.....	84
<b>Interrupts .....</b>	<b>85</b>
Interrupt Registers.....	86
Interrupt Operation .....	89
External Interrupt.....	90
Timer/Event Counter Interrupt.....	90
Touch Key Module Interrupt .....	91
I <sup>2</sup> C Interrupt .....	91
Multi-function Interrupts.....	91
Time Base Interrupt.....	92
Interrupt Wake-up Function.....	93
Programming Considerations.....	94
<b>Configuration Options.....</b>	<b>94</b>
<b>Application Circuits.....</b>	<b>95</b>
<b>Instruction Set.....</b>	<b>96</b>
Introduction .....	96
Instruction Timing .....	96
Moving and Transferring Data .....	96
Arithmetic Operations.....	96
Logical and Rotate Operation .....	97
Branches and Control Transfer .....	97
Bit Operations .....	97
Table Read Operations .....	97
Other Operations.....	97
<b>Instruction Set Summary .....</b>	<b>98</b>
Table Conventions.....	98
<b>Instruction Definition.....</b>	<b>100</b>
<b>Package Information .....</b>	<b>110</b>
6-pin SOT23 Outline Dimensions.....	111
8-pin SOP (150mil) Outline Dimensions .....	112
10-pin MSOP Outline Dimensions .....	113
16-pin NSOP (150mil) Outline Dimensions.....	114

## Features

### CPU Features

- Operating voltage
  - ♦  $f_{SYS}=8\text{MHz}$ : 2.0V~5.5V
- Up to 0.5 $\mu\text{s}$  instruction cycle with 8MHz system clock at  $V_{DD}=5\text{V}$
- Power down and wake-up functions to reduce power consumption
- Oscillator types
  - ♦ Internal High Speed 8MHz RC – HIRC
  - ♦ Internal Low Speed 32kHz 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
- 61 powerful instructions
- Up to 6-level subroutine nesting
- Bit manipulation instruction

### Peripheral Features

- OTP Program Memory: (1K-16) $\times$ 14~(2K-16) $\times$ 15
- Data Memory: 64 $\times$ 8~256 $\times$ 8
- OTP ROM Parameter Program – ORPP (for BS23B04CA and BS23B08CA)
- Watchdog Timer function
- Up to 14 bidirectional I/O lines
- Single external interrupt line shared with I/O pin
- Multiple 8-bit programmable Timer/Event Counters with PWM output (for BS23B04CA and BS23B08CA)
- I<sup>2</sup>C interface (for BS23B04CA and BS23B08CA)
- Up to 8 touch key function
- Up to two Time-Base functions for generation of fixed time interrupt signals
- Low voltage reset function
- Package types: SOT23-6, 8-pin SOP, 10-pin MSOP, 16-pin NSOP

## General Description

The devices are OTP type 8-bit high performance RISC architecture microcontrollers, designed for Touch Key product applications.

For memory features, the devices are supplied with One-Time Programmable, OTP memory. Other memory includes an area of RAM Data Memory.

Analog features include multiple extremely flexible Timer/Event Counters. 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 are provided and the two fully integrated system oscillators require no external components for their implementation. The ability to operate and switch dynamically between a range of operating modes using different clock sources gives users the ability to optimize microcontroller operation and minimize power consumption. Easy communication with the outside world is provided using the internal I<sup>2</sup>C interface.

The devices include 2~8 touch keys which can detect human body contact using external touch pads. The high level of device integration enable applications to be implemented with a minimum number of external components.

Special internal circuitry is also employed to ensure excellent power noise rejection to reduce the possibility of false detections, increasing the touch switch application reliability under adverse environmental conditions. With auto-calibration, low standby current, excellent resistance to voltage fluctuation and other features, this range of touch key devices provide a simple and effective means of implementing touch key operation in a wide variety of applications.

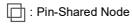
## Selection Table

Most features are common to these devices, the main features distinguishing of them are Memory capacity, I/O count, Touch key count, Time Base, Timer count, stack capacity and package types, etc. The following table summarises the main features of each device.

Part No.	V <sub>DD</sub>	ROM	RAM	I/O	Touch key
BS23A02CA	2.0V~5.5V	(1K-16)×14	64×8	6	2
BS23B04CA	2.0V~5.5V	(2K-16)×15	128×8	8	4
BS23B08CA	2.0V~5.5V	(2K-16)×15	256×8	14	8

Part No.	Time Base	Timer	I <sup>2</sup> C	ORPP	Stack	Package
BS23A02CA	1	—	—	—	2	SOT23-6 8SOP
BS23B04CA	1	8-bit×2	√	√	4	8SOP 10MSOP
BS23B08CA	2	8-bit×4	√	√	6	16NSOP

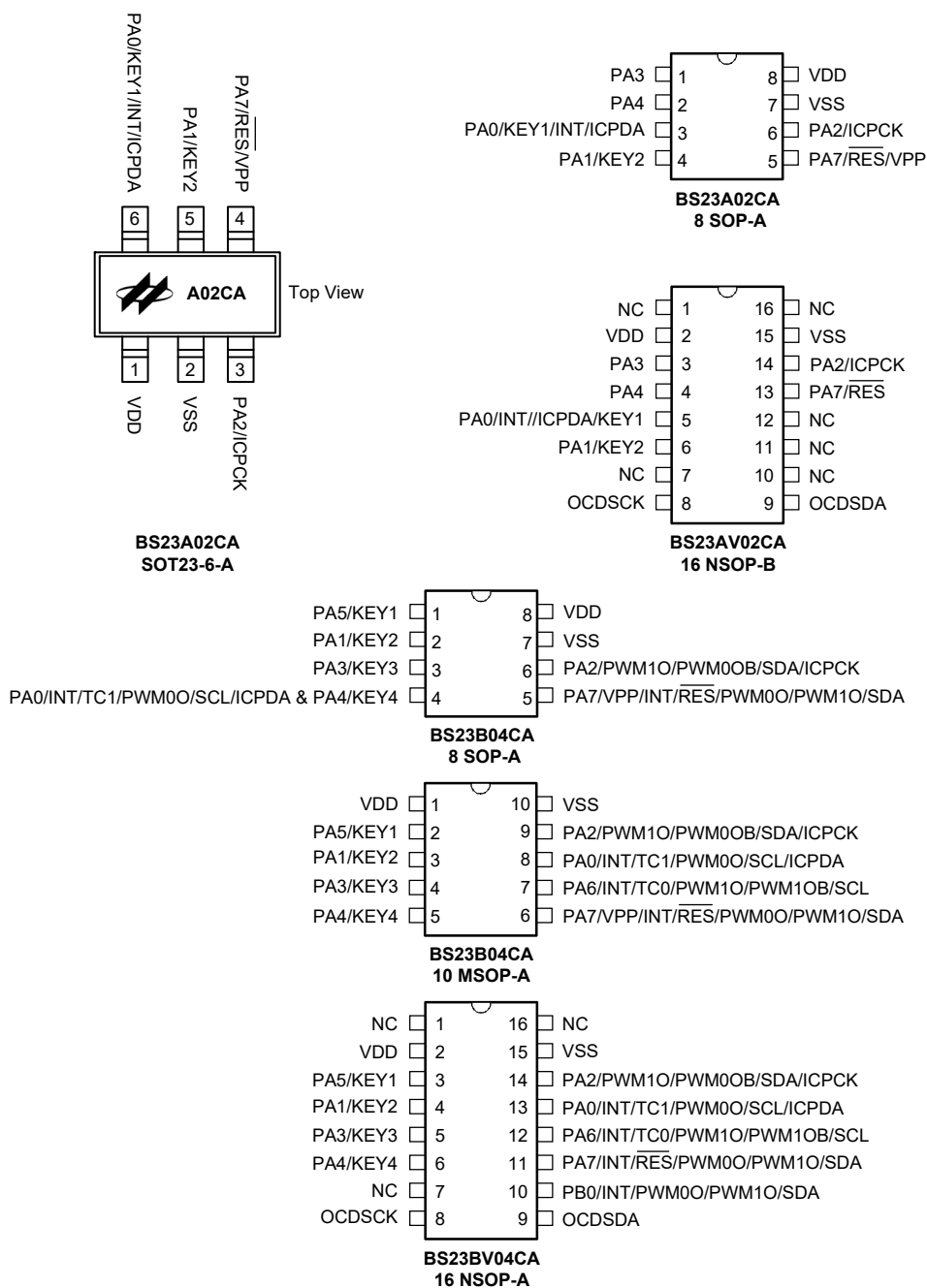
## BS23A02CA



\*: PB0 for the BS23BV04CA 16 NSOP-A only

 : Pin-Shared Node

## Pin Assignment





PB0/KEY1	1	16	PA1/TC1/PWM0O/PWM2O/PWM3O
PB1/KEY2	2	15	PA4/INT/TC2/PWM1O/PWM0OB/SCL
PB2/KEY3	3	14	PA3/TC3/PWM2O/PWM1OB/SDA
PB3/KEY4	4	13	PA0/INT/PWM0O/SDA/ICPDA/OCDSDA
PB4/KEY5	5	12	PA2/PWM1O/PWM3OB/SCL/ICPCK/OCDSCK
PB5/KEY6	6	11	PA7/VPP/INT/ $\overline{\text{RES}}$ /TC0/PWM3O/PWM2OB
PB6/KEY7	7	10	VDD
PB7/KEY8	8	9	VSS

**BS23B08CA/BS23BV08CA**  
**16 NSOP-A**

- Note:
1. If the pin-shared pin functions have multiple outputs, the desired pin-shared function is determined by the corresponding software control bits.
  2. The OCDSDA and OCDSCK pins are supplied as the OCDS dedicated pins and as such only available for the BS23AV02CA/BS23BV04CA/BS23BV08CA device (Flash type) which is the OCDS EV chip for the BS23A02CA/BS23B04CA/BS23B08CA device (OTP type).
  3. The VPP pin is the High Voltage input OTP programming and only available for the BS23A02CA/BS23B04CA/BS23B08CA device.
  4. For the less pin count package types there will be unbonded pins which should be properly configured to avoid unwanted power consumption resulting from floating input conditions. Refer to the “Standby Current Considerations” and “Input/Output Ports” sections.
  5. 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.
  6. The BS23AV02CA 16-pin NSOP-B provides e-FADP08N-BS soft board.
  7. For the BS23B04CA 8-pin SOP-A, on Pin 4 there are two sets of pin functions, the PA0/INT/TC1/PWM0O/SCL/ICPDA or PA4/KEY4, however they cannot be used simultaneously. If the PA0/INT/TC1/PWM0O/SCL/ICPDA function set is used, the PA4 should be configured as an input with pull-high function disabled. If the PA4/KEY4 function set is used, the PA0 should be configured as an input with pull-high function disabled.
  8. The PB0 for the BS23BV04CA 16-pin NSOP-A only, which should be set to the general purpose I/O with pull-up function.
  9. The BS23B04CA PB0 port and the BS23B08CA PA5 port are 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 Description

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

### BS23A02CA

Pin Name	Function	OPT	I/T	O/T	Description
PA0/INT/ICPDA/ KEY1	PA0	PAPU PAWU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	INT	INTEG INTC0 PAS0	ST	—	External interrupt input
	ICPDA	—	ST	CMOS	ICP Data/Address pin
	KEY1	PAS0	AN	—	Touch key input
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/ICPCK	PA2	PAPU PAWU	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	ICPCK	—	ST	—	ICP clock pin
PA3	PA3	PAPU PAWU	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
PA4	PA4	PAPU PAWU	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
PA7/ $\overline{\text{RES}}$ /VPP	PA7	PAPU PAWU	ST	NMOS	General purpose I/O. Register enabled pull-up and wake-up
	$\overline{\text{RES}}$	CO	ST	—	External reset input
	VPP	—	PWR	—	High Voltage input OTP programming pin, not available for EV chip
VDD	VDD	—	PWR	—	Positive power supply
VSS	VSS	—	PWR	—	Negative power supply, ground
<b>For 16-pin NSOP package type only</b>					
OCSDA	OCSDA	—	ST	CMOS	OCDS address/data pin, for EV chip only
OCDSCK	OCDSCK	—	ST	—	OCDS clock pin, for EV chip only
NC	NC	—	—	—	No connection

Legend: I/T: Input type;

O/T: Output type;

OPT: Optional by configuration option (CO) or register option;

PWR: Power;

ST: Schmitt Trigger input;

CMOS: CMOS output;

CO: Configuration option;

AN: Analog signal.

**BS23B04CA**

Pin Name	Function	OPT	I/T	O/T	Description
PA0/INT/TC1/ PWM0O/SCL/ ICPDA	PA0	PAPU PAWU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	INT	INTEG INTC0 PAS0 IFS	ST	—	External interrupt input
	TC1	PAS0	ST	—	Timer/Event Counter 1 clock input
	PWM0O	PAS0	—	CMOS	PWM0 signal output
	SCL	PAS0 IFS	ST	CMOS	I <sup>2</sup> C clock line
	ICPDA	—	ST	CMOS	ICP Data/Address pin
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/PWM1O/ PWM0OB/SDA/ ICPCK	PA2	PAPU PAWU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	PWM1O	PAS0	—	CMOS	PWM1 signal output
	PWM0OB	PAS0	—	CMOS	PWM0 signal inverter output
	SDA	PAS0 IFS	ST	NMOS	I <sup>2</sup> C 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/INT/TC0/ PWM1O/ PWM1OB/SCL	PA6	PAPU PAWU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	INT	INTEG INTC0 PAS1 IFS	ST	—	External interrupt input
	TC0	PAS1	ST	—	Timer/Event Counter 0 clock input
	PWM1O	PAS1	—	CMOS	PWM1 signal output
	PWM1OB	PAS1	—	CMOS	PWM1 signal inverter output
	SCL	PAS1 IFS	ST	CMOS	I <sup>2</sup> C clock line

Pin Name	Function	OPT	I/T	O/T	Description
PA7/VPP/INT/ RES/TC0/ PWM0O/ PWM1O/SDA	PA7	PAPU PAWU PAS1	ST	NMOS	General purpose I/O. Register enabled pull-up and wake-up
	VPP	—	PWR	—	High Voltage input OTP programming pin, not available for EV chip
	INT	INTEG INTC0 IFS PAS1	ST	—	External interrupt input
	RES	CO	ST	—	External reset input
	PWM0O	PAS1	—	CMOS	PWM0 signal output
	PWM1O	PAS1	—	CMOS	PWM1 signal output
	SDA	PAS1 IFS	ST	NMOS	I <sup>2</sup> C data line
VDD	VDD	—	PWR	—	Positive power supply
VSS	VSS	—	PWR	—	Negative power supply, ground
<b>For 16-pin NSOP package type only</b>					
PB0/INT/ PWM0O/ PWM1O/SDA	PB0	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	INT	INTEG INTC0 IFS	ST	—	External interrupt input
	PWM0O	PBS0	—	CMOS	PWM0 signal output
	PWM1O	PBS0	—	CMOS	PWM1 signal output
	SDA	PBS0 IFS	ST	NMOS	I <sup>2</sup> C data line
OCDSDA	OCDSDA	—	ST	CMOS	OCDS address/data pin, for EV chip only
OCDSCK	OCDSCK	—	ST	—	OCDS clock pin, for EV chip only
NC	NC	—	—	—	No connection

Legend: I/T: Input type;

O/T: Output type;

OPT: Optional by configuration option (CO) or register option;

PWR: Power;

ST: Schmitt Trigger input;

CMOS: CMOS output;

CO: Configuration option;

AN: Analog signal.

### BS23B08CA

Pin Name	Function	OPT	I/T	O/T	Description
PA0/INT/ PWM0O/SDA/ ICPDA/OCDSDA	PA0	PAPU PAWU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	INT	INTEG INTC0 PAS0 IFS	ST	—	External interrupt input
	PWM0O	PAS0	—	CMOS	PWM0 signal output
	SDA	PAS0 IFS	ST	NMOS	I <sup>2</sup> C data line
	ICPDA	—	ST	CMOS	ICP Data/Address pin
	OCDSDA	—	ST	CMOS	OCDS address/data pin, for EV chip only

Pin Name	Function	OPT	I/T	O/T	Description
PA1/TC1/ PWM0O/ PWM2O/ PWM3O	PA1	PAPU PAWU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	TC1	PAS0	ST	—	Timer/Event Counter 1 clock input
	PWM0O	PAS0	—	CMOS	PWM0 signal output
	PWM2O	PAS0	—	CMOS	PWM2 signal output
	PWM3O	PAS0	—	CMOS	PWM3 signal output
PA2/PWM1O/ PWM3OB/SCL/ ICPCK/ OCDSCK	PA2	PAPU PAWU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	PWM1O	PAS0	—	CMOS	PWM1 signal output
	PWM3OB	PAS0	—	CMOS	PWM3 signal inverter output
	SCL	PAS0 IFS	ST	CMOS	I <sup>2</sup> C clock line
	ICPCK	—	ST	—	ICP clock pin
	OCDSCK	—	ST	—	OCD clock pin, for EV chip only
PA3/TC3/ PWM2O/ PWM1OB/SDA	PA3	PAPU PAWU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	TC3	PAS0	ST	—	Timer/Event Counter 3 clock input
	PWM2O	PAS0	—	CMOS	PWM2 signal output
	PWM1OB	PAS0	—	CMOS	PWM1 signal inverter output
	SDA	PAS0 IFS	ST	NMOS	I <sup>2</sup> C data line
PA4/INT/TC2/ PWM1O/ PWM0OB/SCL	PA4	PAPU PAWU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	INT	INTEG INTC0 PAS1 IFS	ST	—	External interrupt input
	TC2	PAS1	ST	—	Timer/Event Counter 2 clock input
	PWM1O	PAS1	—	CMOS	PWM1 signal output
	PWM0OB	PAS1	—	CMOS	PWM0 signal inverter output
	SCL	PAS1 IFS	ST	CMOS	I <sup>2</sup> C clock line
PA7/VPP/INT/ RES/TC0/ PWM3O/ PWM2OB	PA7	PAPU PAWU PAS1	ST	NMOS	General purpose I/O. Register enabled pull-up and wake-up
	VPP	—	PWR	—	High Voltage input OTP programming pin, not available for EV chip
	INT	INTEG INTC0 IFS PAS1	ST	—	External interrupt input
	RES	CO	ST	—	External reset input
	TC0	PAS1	ST	—	Timer/Event Counter 0 clock input
	PWM3O	PAS1	—	CMOS	PWM3 signal output
	PWM2OB	PAS1	—	CMOS	PWM2 signal inverter output
PB0/KEY1	PB0	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	KEY1	PBS0	AN	—	Touch key input

Pin Name	Function	OPT	I/T	O/T	Description
PB1/KEY2	PB1	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-up.
	KEY2	PBS0	AN	—	Touch key input
PB2/KEY3	PB2	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-up.
	KEY3	PBS0	AN	—	Touch key input
PB3/KEY4	PB3	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-up.
	KEY4	PBS0	AN	—	Touch key input
PB4/KEY5	PB4	PBPU PBS1	ST	CMOS	General purpose I/O. Register enabled pull-up.
	KEY5	PBS1	AN	—	Touch key input
PB5/KEY6	PB5	PBPU PBS1	ST	CMOS	General purpose I/O. Register enabled pull-up.
	KEY6	PBS1	AN	—	Touch key input
PB6/KEY7	PB6	PBPU PBS1	ST	CMOS	General purpose I/O. Register enabled pull-up.
	KEY7	PBS1	AN	—	Touch key input
PB7/KEY8	PB7	PBPU PBS1	ST	CMOS	General purpose I/O. Register enabled pull-up.
	KEY8	PBS1	AN	—	Touch key input
VDD	VDD	—	PWR	—	Positive power supply
VSS	VSS	—	PWR	—	Negative power supply, ground

Legend: I/T: Input type;

O/T: Output type;

OPT: Optional by configuration option (CO) or register option;

PWR: Power;

ST: Schmitt Trigger input;

CMOS: CMOS output;

CO: Configuration option;

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.....	$-60^{\circ}C$ to $150^{\circ}C$
Operating Temperature.....	$-40^{\circ}C$ to $85^{\circ}C$
$I_{OH}$ 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 the devices. Functional operation of the 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. Characteristics

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

### Operating Voltage Characteristics

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
V <sub>DD</sub>	Operating Voltage – HIRC	f <sub>sys</sub> =f <sub>HIRC</sub> =8MHz	2.0	—	5.5	V
	Operating Voltage – LIRC	f <sub>sys</sub> =f <sub>LIRC</sub> =32kHz	2.0	—	5.5	V

### Standby Current Characteristics

Ta=25°C, unless otherwise specified

Symbol	Standby Mode	Test Conditions		Min.	Typ.	Max.	Max. @85°C	Unit
		V <sub>DD</sub>	Conditions					
I <sub>STB</sub>	SLEEP Mode	2V	WDT off	—	0.45	0.80	7.00	μA
		3V		—	0.45	0.90	8.00	
		5V		—	0.5	2.0	10.0	
		2V	WDT on	—	1.5	3.0	5.5	μA
		3V		—	1.8	3.6	6.5	
		5V		—	3	5	10	
	IDLE0 Mode – LIRC	2V	f <sub>SUB</sub> on	—	2.4	4.0	8.0	μA
		3V		—	3	5	9	
		5V		—	5	10	15	
	IDLE1 Mode – HIRC	2V	f <sub>SUB</sub> on, f <sub>sys</sub> =8MHz	—	288	400	480	μA
		3V		—	360	500	600	
		5V		—	600	800	960	

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.

### Operating Current Characteristics

Ta=-40°C~85°C

Symbol	Operating Mode	Test Conditions		Min.	Typ.	Max.	Unit
		V <sub>DD</sub>	Conditions				
I <sub>DD</sub>	SLOW Mode – LIRC	2V	f <sub>sys</sub> =32kHz	—	30	50	μA
		3V		—	40	60	
		5V		—	60	80	
	FAST Mode – HIRC	2V	f <sub>sys</sub> =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.

## 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.

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V <sub>DD</sub>	Temp.				
f <sub>HIRC</sub>	8MHz Writer Trimmed HIRC Frequency	3V/5V	25°C	-1%	8	+1%	MHz
			-40°C~85°C	-4%	8	+4%	
		2.0V~5.5V	25°C	-3%	8	+3%	
			-40°C~85°C	-5%	8	+5%	

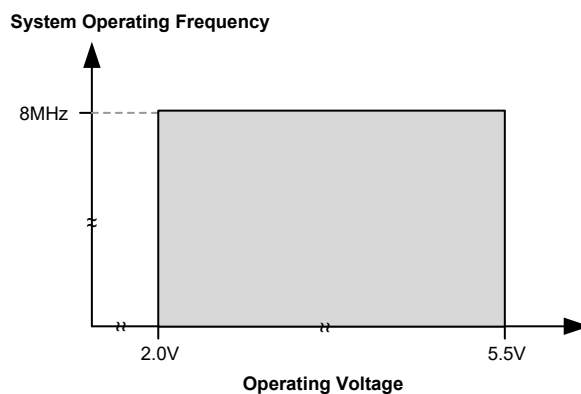
Note: 1. The 3V/5V values for V<sub>DD</sub> are provided as these are the two selectable fixed voltages at which the HIRC frequency is trimmed by the writer.

2. The row below the 3V/5V trim voltage row is provided to show the values for the full V<sub>DD</sub> range operating voltage. It is recommended that the trim voltage is fixed at 3V for application voltage ranges from 2.0V to 3.6V and fixed at 5V for application voltage ranges from 3.3V to 5.5V.

### Low Speed Internal Oscillator Characteristics – LIRC

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V <sub>DD</sub>	Temp.				
f <sub>LIRC</sub>	LIRC Frequency	2.0V~5.5V	25°C	-20%	32	+20%	kHz
			-40°C~85°C	-50%	32	+60%	
t <sub>START</sub>	LIRC Start Up Time	—	-40°C~85°C	—	—	500	μs

### Operating Frequency Characteristic Curves





## System Start Up Time Characteristics

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V <sub>DD</sub>	Conditions				
t <sub>SST</sub>	System Start-up Time (Wake-up from Condition Where f <sub>SYS</sub> is Off)	—	f <sub>SYS</sub> =f <sub>H</sub> ~f <sub>H</sub> /64, f <sub>H</sub> =f <sub>HIRC</sub>	—	128	—	t <sub>SYS</sub>
		—	f <sub>SYS</sub> =f <sub>SUB</sub> =f <sub>LIRC</sub>	—	2	—	t <sub>SUB</sub>
	System Start-up Time (Wake-up from Condition Where f <sub>SYS</sub> is On)	—	f <sub>SYS</sub> =f <sub>H</sub> ~f <sub>H</sub> /64, f <sub>H</sub> =f <sub>HIRC</sub>	—	2	—	t <sub>SYS</sub>
		—	f <sub>SYS</sub> =f <sub>SUB</sub> =f <sub>LIRC</sub>	—	2	—	t <sub>SUB</sub>
	System Speed Switch Time (FAST to SLOW Mode or SLOW to FAST Mode)	—	f <sub>HIRC</sub> switches from off to on	—	128	—	t <sub>HIRC</sub>
t <sub>RSTD</sub>	System Reset Delay Time (Reset Source from Power-on Reset or LVR Hardware Reset)	—	RR <sub>POR</sub> =5V/ms	5	16	80	ms
	System Reset Delay Time (Reset Source from WDT Overflow or RES Pin Reset)	—	—				

- Note: 1. For the System Start-up time values, whether f<sub>SYS</sub> is on or off depends upon the mode type and the chosen f<sub>SYS</sub> system oscillator. Details are provided in the System Operating Modes section.
2. The time units, shown by the symbols, t<sub>HIRC</sub>, etc., are the inverse of the corresponding frequency values as provided in the frequency tables. For example t<sub>HIRC</sub>=1/f<sub>HIRC</sub>, t<sub>LIRC</sub>=1/f<sub>LIRC</sub>, 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<sub>START</sub>, as provided in the LIRC frequency table, must be added to the t<sub>SST</sub> time in the table above.
4. The System Speed Switch Time is effectively the time taken for the newly activated oscillator to start up.

## Input/Output Characteristics

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

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V <sub>DD</sub>	Conditions				
V <sub>IL</sub>	Input Low Voltage for I/O Ports (Except RES)	5V	—	0	—	1.5	V
		—	—	0	—	0.2V <sub>DD</sub>	
	Input Low Voltage for $\overline{\text{RES}}$ Pin	—	V <sub>DD</sub> ≥2.7	0	—	0.4V <sub>DD</sub>	V
		—	2.0≤V <sub>DD</sub> <2.7	0	—	0.3V <sub>DD</sub>	V
V <sub>IH</sub>	Input High Voltage for I/O Ports (Except RES)	5V	—	3.5	—	5.0	V
		—	—	0.8V <sub>DD</sub>	—	V <sub>DD</sub>	
	Input High Voltage for $\overline{\text{RES}}$ Pin	—	—	0.9V <sub>DD</sub>	—	V <sub>DD</sub>	V
I <sub>OL</sub>	Sink Current for I/O Ports	3V	V <sub>OL</sub> =0.1V <sub>DD</sub>	16	32	—	mA
		5V		32	65	—	
I <sub>OH</sub>	Source Current for I/O Ports	3V	V <sub>OH</sub> =0.9V <sub>DD</sub>	-4	-8	—	mA
		5V		-8	-16	—	
R <sub>PH</sub>	Pull-high Resistance for I/O Ports (Note)	3V	—	20	60	100	kΩ
		5V	—	10	30	50	
I <sub>LEAK</sub>	Input Leakage Current for I/O Ports	5V	V <sub>IN</sub> =V <sub>DD</sub> or V <sub>IN</sub> =V <sub>SS</sub>	—	—	±1	μA
t <sub>TC</sub>	TCn Clock Input Minimum Pulse Width (for BS23B04CA and BS23B08CA)	—	—	20	—	—	ns
t <sub>INT</sub>	Interrupt Input Pin Minimum Pulse Width	—	—	0.3	—	—	μs
t <sub>RES</sub>	External Reset Minimum Low Pulse Width	—	—	0.3	—	—	μs

Note: The R<sub>PH</sub> 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<sub>PH</sub> value.

## Memory Characteristics

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

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V <sub>DD</sub>	Conditions				
OTP Program Memory (for BS23B04CA and BS23B08CA)							
V <sub>DD</sub>	V <sub>DD</sub> for Read – ORPP Memory	—	—	2.0	—	5.5	V
	V <sub>DD</sub> for Write – ORPP Memory	—	—	3.0	—	5.5	V
V <sub>PP</sub>	V <sub>PP</sub> for Write – ORPP Memory	—	—	8.25	8.50	8.75	V
t <sub>WR</sub>	Write Cycle Time – ORPP Memory	—	—	—	300	450	μs
E <sub>p</sub>	Cell Endurance – ORPP Memory	—	—	1	—	—	W
t <sub>RETD</sub>	ROM Data Retention Time	—	Ta=25°C	—	40	—	Year
RAM Data Memory							
V <sub>DR</sub>	RAM Data Retention Voltage	—	—	1.0	—	—	V

Note: “W” means Write times.

## LVR Electrical Characteristics

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V <sub>DD</sub>	Conditions				
V <sub>LVR</sub>	Low Voltage Reset Voltage	—	LVR enable, voltage select 1.9V	-5%	1.9	+5%	V
			LVR enable, voltage select 2.1V		2.1		
			LVR enable, voltage select 3.15V		3.15		
			LVR enable, voltage select 4.2V		4.2		
I <sub>LVR</sub>	Operating Current	3V	LVR enable, V <sub>LVR</sub> =1.9V	—	—	15	μA
		5V	LVR enable, V <sub>LVR</sub> =1.9V	—	15	30	
t <sub>LVR</sub>	Minimum Low Voltage Width to Reset	—	—	100	240	1250	μs

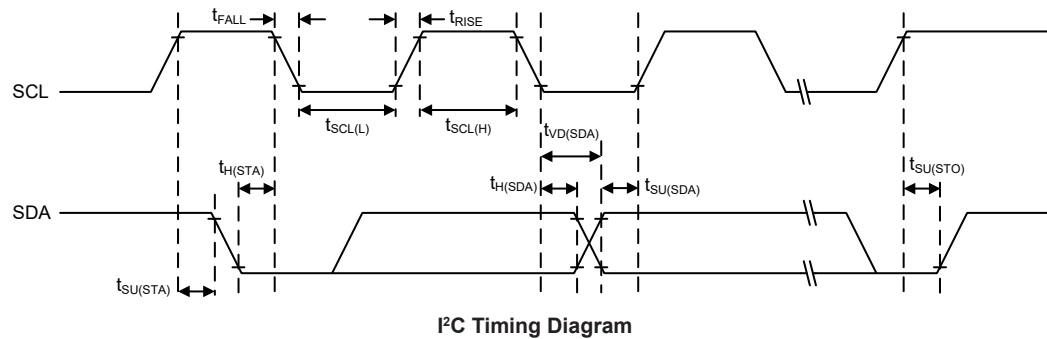
## I<sup>2</sup>C Electrical Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V <sub>DD</sub>	Conditions				
f <sub>I2C</sub>	I <sup>2</sup> C Standard Mode (100kHz) f <sub>sys</sub> Frequency <sup>(Note)</sup>	—	No clock debounce	2	—	—	MHz
			2 system clock debounce	4	—	—	
			4 system clock debounce	4	—	—	
	I <sup>2</sup> C Fast Mode (400kHz) f <sub>sys</sub> Frequency <sup>(Note)</sup>	—	No clock debounce	4	—	—	MHz
			2 system clock debounce	8	—	—	
			4 system clock debounce	8	—	—	
f <sub>SCL</sub>	SCL Clock Frequency	3V/5V	Standard mode	—	—	100	kHz
			Fast mode	—	—	400	
t <sub>SCL(H)</sub>	SCL Clock High Time	3V/5V	Standard mode	3.5	—	—	μs
			Fast mode	0.9	—	—	
t <sub>SCL(L)</sub>	SCL Clock Low Time	3V/5V	Standard mode	3.5	—	—	μs
			Fast mode	0.9	—	—	
t <sub>FALL</sub>	SCL and SDA Fall Time	3V/5V	Standard mode	—	—	1.3	μs
			Fast mode	—	—	0.34	

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V <sub>DD</sub>	Conditions				
t <sub>RISE</sub>	SCL and SDA Rise Time	3V/5V	Standard mode	—	—	1.3	μs
			Fast mode	—	—	0.34	
t <sub>SU(SDA)</sub>	SDA Data Setup Time	3V/5V	Standard mode	0.25	—	—	μs
			Fast mode	0.1	—	—	
t <sub>H(SDA)</sub>	SDA Data Hold Time	3V/5V	—	0.1	—	—	μs
t <sub>VD(SDA)</sub>	SDA Data Valid Time	3V/5V	—	—	—	0.6	μs
t <sub>SU(STA)</sub>	Start Condition Setup Time	3V/5V	Standard mode	3.5	—	—	μs
			Fast mode	0.6	—	—	
t <sub>H(STA)</sub>	Start Condition Hold Time	3V/5V	Standard mode	4.0	—	—	μs
			Fast mode	0.6	—	—	
t <sub>SU(STO)</sub>	Stop Condition Setup Time	3V/5V	Standard mode	3.5	—	—	μs
			Fast mode	0.6	—	—	

Note: Using the debounce function can make the transmission more stable and reduce the probability of communication failure due to interference.

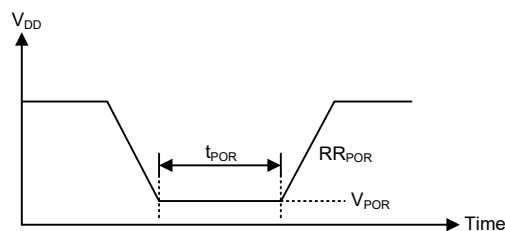


I²C Timing Diagram

## Power-on Reset Characteristics

T<sub>a</sub>=-40°C~85°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V <sub>DD</sub>	Conditions				
V <sub>POR</sub>	V <sub>DD</sub> Start Voltage to Ensure Power-on Reset	—	—	—	—	100	mV
RR <sub>POR</sub>	V <sub>DD</sub> Rising Rate to Ensure Power-on Reset	—	—	0.035	—	—	V/ms
t <sub>POR</sub>	Minimum Time for V <sub>DD</sub> Stays at V <sub>POR</sub> to Ensure Power-on Reset	—	—	1	—	—	ms



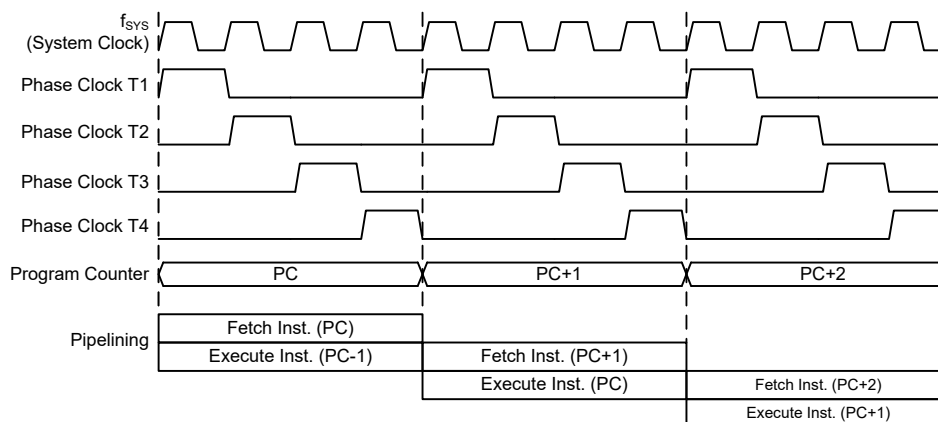
## System Architecture

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

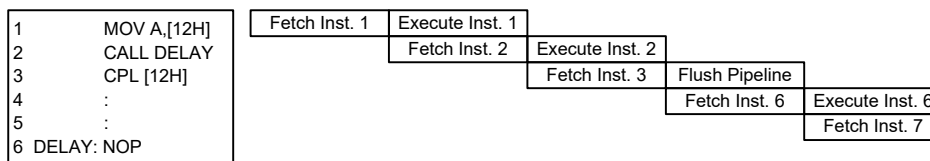
### Clocking and Pipelining

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.

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



**System Clocking and Pipelining**



**Instruction Fetching**

## Program Counter

During program execution, the Program Counter is used to keep track of the address of the next instruction to be executed. It is automatically incremented by one each time an instruction is executed except for instructions, such as “JMP” or “CALL” that demand a jump to a non-consecutive Program Memory address. Only the lower 8 bits, known as the Program Counter Low 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)
BS23A02CA	PC9~PC8	PCL7~PCL0
BS23B04CA/BS23B08CA	PC10~PC8	PCL7~PCL0

**Program Counter**

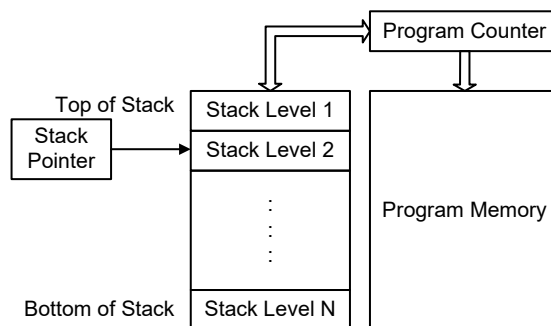
The lower byte of the Program Counter, known as the Program Counter Low register or PCL, is available for program control and is a readable and writeable register. By transferring data directly into this register, a short program jump can be executed directly; however, as only this low byte is available 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 has multiple levels and is neither part of the data nor part of the program space, and is neither readable nor writeable. The activated level is indexed by the Stack Pointer, and is neither readable nor writeable. At a subroutine call or interrupt acknowledge signal, the contents of the Program Counter are pushed onto the stack. At the end of a subroutine or an interrupt routine, signaled by a return instruction, RET or RETI, the Program Counter is restored to its previous value from the stack. After a device reset, the Stack Pointer will point to the top of the stack.

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

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



Device	Stack Levels (N)
BS23A02CA	2
BS23B04CA	4
BS23B08CA	6

## Arithmetic and Logic Unit – ALU

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

- Arithmetic operations: ADD, ADDM, ADC, ADCM, SUB, SUBM, SBC, SBCM, DAA
- 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

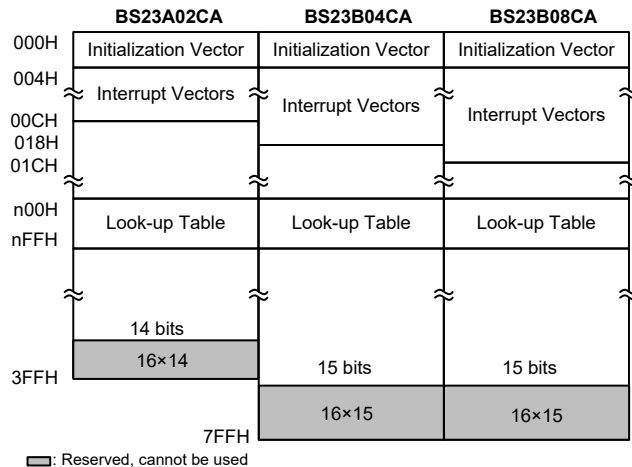
## OTP Program Memory

The Program Memory is the location where the user code or program is stored. The devices are supplied with One-Time Programmable, OTP memory where users can program their application code into the device.

### Structure

The Program Memory has a capacity of  $(1K-16) \times 14 \sim (2K-16) \times 15$  bits. Note that the subtractive  $16 \times 14 \sim 16 \times 15$  bits space is reserved and cannot be used. The Program Memory is addressed by the Program Counter and also contains data, table information and interrupt entries. Table data, which can be set in any location within the Program Memory, is addressed by a separate table pointer register.

Device	Capacity
BS23A02CA	$(1K-16) \times 14$
BS23B04CA/BS23B08CA	$(2K-16) \times 15$



**Program Memory Structure**

## Special Vectors

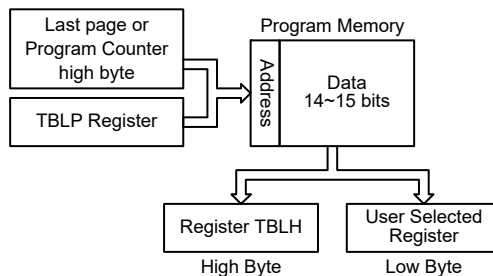
Within the Program Memory, certain locations are reserved for the reset and interrupts. The location 000H is reserved for use by the device reset for program initialisation. After a device reset is initiated, the program will jump to this location and begin execution.

## Look-up Table

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

After setting up the table pointer, the table data can be retrieved from the Program Memory using the “TABRD [m]” or “TABRDL[m]” instructions respectively. When the 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 microcontrollers. 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 words Program Memory of the devices. The table pointer low byte register is set 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 specific address pointed by the TBLP register if the "TABRD [m]" instruction is being used. The high byte of the table data which in this case is equal to zero will be transferred to the TBLH register automatically when the "TABRD [m]" instruction is executed.

Because the TBLH register is a read-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
:
:
tabrd tempreg1 ; transfers value in table referenced by table pointer,
               ; data at program memory address "306H" transferred to tempreg1 and
TBLH
dec tblp      ; reduce value of table pointer by one
tabrd tempreg2 ; transfers value in table referenced by table pointer,
               ; data at program memory address "305H" transferred to tempreg2 and
TBLH
               ; in this example the data "1AH" is transferred to tempreg1 and data
"0FH"
               ; to register tempreg2
               ; the value "00H" will be transferred to the high byte register TBLH
:
:
org 300h      ; sets initial address of program memory
dc 00Ah, 00Bh, 00Ch, 00Dh, 00Eh, 00Fh, 01Ah, 01Bh
```



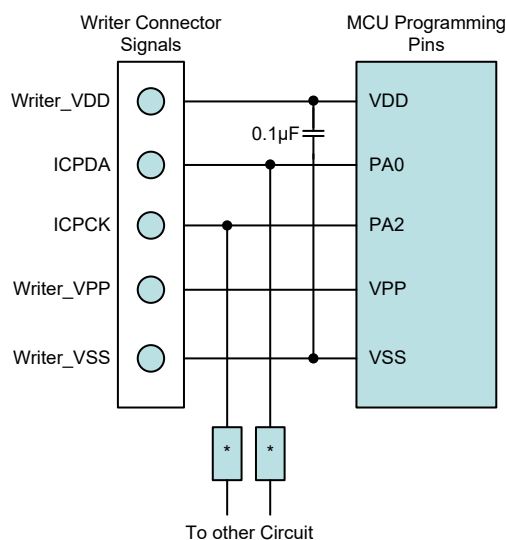
## In Circuit Programming – ICP

The provision of OTP type Program Memory, users can program their application One-Time into the device. As an additional convenience, Holtek has provided a means of programming the microcontroller in-circuit using a 5-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
VPP	VPP	Programming OTP ROM power supply (8.5V)
VDD	VDD	Power Supply. A 0.1 $\mu$ F capacitor is required to be connected between VDD and VSS for programming.
VSS	VSS	Ground

The Program Memory can be programmed serially in-circuit using this 5-wire interface. Data is downloaded and uploaded serially on a single pin with an additional line for the clock. Three 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: 1. A 0.1 $\mu$ F capacitor is required to be connected between VDD and VSS for ICP programming, and as close to these pins as possible.

2. \* may be resistor or capacitor. The resistance of \* must be greater than 1k $\Omega$  or the capacitance of \* must be less than 1nF.

## On-Chip Debug Support – OCDS

There is an EV chip named BS23AV02CA/BS23BV04CA/BS23BV08CA which are used to emulate the devices named BS23A02CA/BS23B04CA/BS23B08CA respectively. The EV chip device also provides an “On-Chip Debug” function to debug the real MCU device during the development process. The EV chip and the real MCU device are almost functionally compatible except for “On-Chip Debug” function and the package type. Users can use the EV chip device to emulate the real chip device behavior by connecting the OCDSDA and OCDSCCK pins to the Holtek HT-IDE development tools. The OCDSDA pin is the OCDS Data/Address input/output pin while the OCDSCCK pin is the OCDS clock input pin. When users use the EV chip for debugging, other functions which are shared with the OCDSDA and OCDSCCK pins in the device will have no effect in the EV chip. 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 Pins	Pin Description
OCDSDA	OCDSDA	On-Chip Debug Support Data/Address input/output
OCDSCCK	OCDSCCK	On-Chip Debug Support Clock input
VDD	VDD	Power Supply
VSS	VSS	Ground

## OTP ROM Parameter Program – ORPP – BS23B04CA and BS23B08CA

This device contains an ORPP function. The provision of the ORPP function offers users the convenience of OTP Memory programming features. Note that the Write operation only writes data to the last page of OTP Program Memory, and the data can only be written once and cannot be erased. Before the write operation is implemented, the VPP pin must be connected to 8.5V.

### ORPP Registers

Three registers control the overall operation of the internal ORPP function. These are data registers ODL and ODH, and a control register OCR.

Registers Name	Bit							
	7	6	5	4	3	2	1	0
OCR	—	—	—	—	WREN	WR	—	—
ODL	D7	D6	D5	D4	D3	D2	D1	D0
ODH	—	D14	D13	D12	D11	D10	D9	D8

ORPP Register List

#### • ODL 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**: ORPP Program Memory data bit 7 ~ bit 0

#### • ODH Register

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

Bit 7 Unimplemented, read as “0”

Bit 6~0 **D14~D8**: ORPP Program Memory data bit 14 ~ bit 8

• **OCR Register**

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

Bit 7~4 Unimplemented, read as “0”

Bit 3 **WREN**: ORPP Write Enable

0: Disable

1: Enable

This is the ORPP Write Enable Bit which must be set high before write operations are carried out. This bit will be automatically reset to zero by the hardware after the write cycle has finished. Clearing this bit to zero will inhibit ORPP write operations.

Bit 2 **WR**: ORPP Write Control

0: Write cycle has finished

1: Activate a write cycle

This is the ORPP 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 1~0 Unimplemented, read as “0”

Note: 1. The WREN and WR cannot be set high at the same time in one instruction.

2. Note that the CPU will be stopped when a write operation is successfully activated.

3. Ensure that the  $f_{SUB}$  clock is stable before executing the write operation.

4. Ensure that the write operation is totally complete before executing other operations.

### ORPP Writing Data to the OTP Program Memory

For ORPP write operation the data to be written should be placed in the ODH and ODL registers and the desired write address should first be placed in the TBLP register. To write data to the OTP Program Memory, the write enable bit, WREN, in the OCR register must first be set high to enable the write function. After this, the WR bit in the OCR register must be immediately set high to initiate a write cycle. These two instructions must be executed in two consecutive instruction cycles to activate a write operation successfully. The global interrupt bit EMI should also first be cleared before implementing any write operations, and then set again after a valid write activation procedure has completed. Note that the CPU will be stopped when a write operation is successfully activated. When the write cycle terminates, the CPU will resume executing the application program. And the WR bit will be automatically cleared to zero by the microcontroller, informing the user that the data has been written to the OTP Program Memory.

### ORPP Reading Data from the OTP Program Memory

For ORPP read operation the desired address should first be placed in the TBLP register. Then the data can be retrieved from the program memory using the “TABRDL [m]” instruction. 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.

### Programming Considerations

Care must be taken that data is not inadvertently written to the OTP Program Memory. Protection can be enhanced by ensuring that the Write Enable bit is normally cleared to zero when not writing. 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 set high again after a write activation procedure has completed. Note that the device should not enter the IDLE or SLEEP mode until the ORPP write operation is totally complete. Otherwise, the ORPP write operation will fail.

### Programming Examples

#### ORPP Reading Data from the OTP Program Memory

```
Tempreg1 db?          ; temporary register
MOV A, 03H
MOV TBLP, A            ; set read address 03H
TABRDL Tempreg1        ; transfers value in table (last page) referenced by table
                        ; pointer, data at program memory address "0703H" transferred
                        ; to tempreg1 and TBLH
```

#### ORPP Writing Data to the OTP Program Memory

```
MOV A, ORPP_ADRES     ; user defined address
MOV TBLP, A
MOV A, ORPP_DATA_L    ; user defined data
MOV ODL, A
MOV A, ORPP_DATA_H
MOV ODH, A
MOV A, 00H
MOV OCR, A
CLR EMI
SET WREN              ; set WREN bit, enable write operation
SET WR               ; start Write Cycle - set WR bit - executed immediately
                    ; after setting WREN bit

SET EMI
BACK:
SZ WR                ; check for write cycle end
JMP BACK
NOP
```

## Data Memory

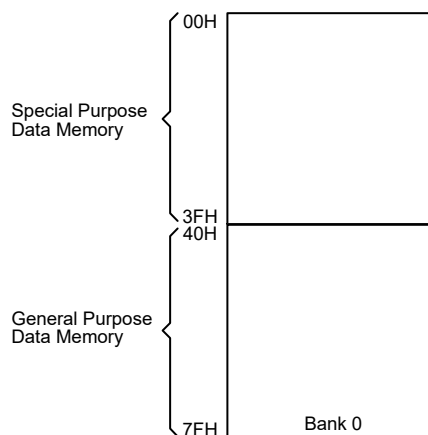
The Data Memory is a volatile area of 8-bit wide RAM internal memory and is the location where temporary information is stored.

### Structure

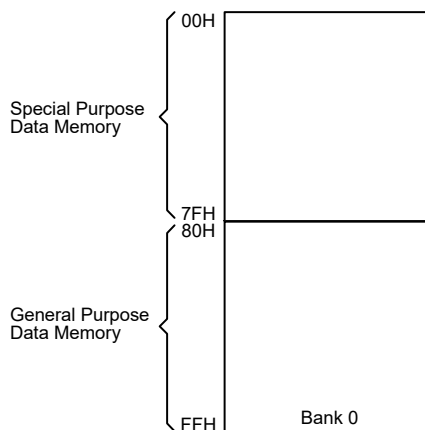
Categorised into two types, the first of these is an area of RAM, known as the Special Function Data Memory. These registers have fixed locations and are necessary for correct operation of the devices. Many of these registers can be read from and written to directly under program control, however, some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

The start address of the Data Memory for the devices is 00H. For the BS23A02CA device, the address range of the Special Purpose Data Memory is from 00H to 3FH while the address range of the General Purpose Data Memory is from 40H to 7FH. For the BS23B04CA and BS23B08CA devices, the address range of the Special Purpose Data Memory is from 00H to 7FH while the address range of the General Purpose Data Memory is from 80H to FFH. Switching between the different Data Memory banks is achieved by setting the Data Memory Bank Pointer to the correct value.

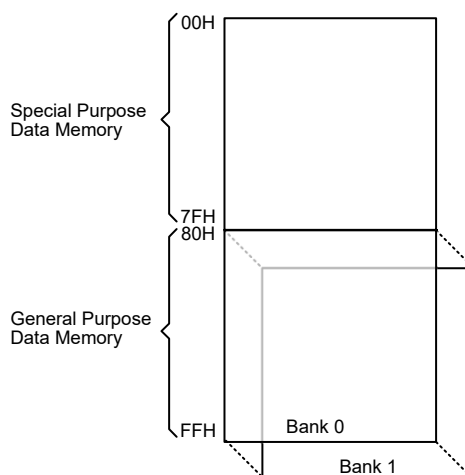
Device	Special Purpose Data Memory	General Purpose Data Memory	
	Located Bank	Capacity	Bank: Address
BS23A02CA	0	64×8	0: 40H~7FH
BS23B04CA	0	128×8	0: 80H~FFH
BS23B08CA	0	256×8	0: 80H~FFH 1: 80H~FFH



**Data Memory Structure – BS23A02CA**



**Data Memory Structure – BS23B04CA**



**Data Memory Structure – BS23B08CA**


## 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	
0CH	INTEG
0DH	WDTC
0EH	TB0C
0FH	
10H	SCC
11H	INTC0
12H	
13H	
14H	PA
15H	PAC
16H	PAPU
17H	PAWU
18H	
19H	TKTMR
1AH	TKC0
1BH	TKC1
1CH	TK16DL
1DH	TK16DH
1EH	TKM0C0
1FH	TKM0C1
20H	TKM016DL
21H	TKM016DH
22H	TKM0ROL
23H	TKM0ROH
24H	PAS0
...	
3FH	

 : Unused, read as 00H

Special Purpose Data Memory – BS23A02CA

	Bank 0
00H	IAR0
01H	MP0
02H	IAR1
03H	MP1
04H	
05H	ACC
06H	PCL
07H	TBLP
08H	TBLH
09H	
0AH	STATUS
0BH	
0CH	INTEG
0DH	WDTC
0EH	TB0C
0FH	
10H	SCC
11H	INTC0
12H	
13H	
14H	PA
15H	PAC
16H	PAPU
17H	PAWU
18H	
19H	TKTMR
1AH	TKC0
1BH	TKC1
1CH	TK16DL
1DH	TK16DH
1EH	TKM0C0
1FH	TKM0C1
20H	TKM016DL
21H	TKM016DH
22H	TKM0ROL
23H	TKM0ROH
24H	IFS
25H	INTC1
26H	PB
27H	PBC
28H	PBPU
29H	TMR0C/PWM0C
2AH	TMR0/PWM0DATA
2BH	TMR1C/PWM1C
2CH	TMR1/PWM1DATA
2DH	OCR
2EH	ODL
2FH	ODH
30H	IICC0
31H	IICC1
32H	IICD
33H	IICA
34H	IICTOC
35H	PAS0
36H	PAS1
37H	PBS0
...	
7FH	

□ : Unused, read as 00H

**Special Purpose Data Memory – BS23B04CA**



 : Unused, read as 00H

August 22, 2024

## 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 Register – IAR0, IAR1

The Indirect Addressing Register, IAR0 and IAR1, although having its location in normal RAM register space, does not actually physically exist as normal registers. The method of indirect addressing for RAM data manipulation uses this Indirect Addressing Register and Memory Pointer, 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 this register but rather to the memory location specified by the corresponding Memory Pointer, MP0 or MP1. Acting as a pair, IAR0 and MP0 or IAR1 and MP1 can together access data from Bank 0. As the Indirect Addressing Register is not physically implemented, reading the Indirect Addressing Register will return a result of “00H” and writing to the register will result in no operation. For BS23A02CA device does not have IAR1 and MP1.

### Memory Pointer – MP0, MP1

One or two Memory Pointer, known as MP0 and MP1 are provided. This Memory Pointer is 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 Indirect Addressing Register is carried out, the actual address that the microcontroller is directed to is the address specified by the Memory Pointer. MP0, together with the 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 the desired data memory bank selected by the DMBP0 bit in BP register. Direct Addressing can only be used within Bank 0, all other Banks must be addressed indirectly using MP1 and IAR1. For BS23A02CA device does not have IAR1 and MP1.

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                ; set size of block
    mov block, a
    mov a, offset adres1      ; Accumulator loaded with first RAM address
    mov MP0, a                ; set memory pointer with first RAM address
loop:
    clr IAR0                  ; clear the data at address defined by MP0
    inc MP0                   ; increase 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 – BS23B08CA

For this device, the Data Memory is divided into two banks, Bank0~Bank1. Selecting the required Data Memory area is achieved using the Bank Pointer. The DMBP0 bit in the BP register is used to select Data Memory Banks.

The Data Memory is initialised to Bank 0 after a reset, except for a WDT time-out reset in IDLE or SLEEP Mode, in which case, the Data Memory bank remains unaffected. It should be noted that the Special Purpose Data Memory is not affected by the bank selection, which means that the Special Function Registers can be accessed from within any bank. Directly addressing the Data Memory will always result in Bank 0 being accessed irrespective of the value of the Bank Pointer. Accessing data from Bank1 must be implemented using Indirect Addressing

### • BP Register

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

Bit 7~1 Unimplemented, read as “0”

Bit 0 **DMBP0**: Data Memory Bank Selection

0: Bank 0

1: Bank 1

## Accumulator – ACC

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

## Program Counter Low Byte Register – PCL

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

## Look-up Table Registers – TBLP, TBLH

The special function register TBLP is used to control operation of the look-up table which is stored in the Program Memory. TBLP is the table pointer and indicates the location where the table data is located. Its value must be set before any table read commands are executed. Its 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 registers are important and if the subroutine can corrupt the status register, precautions must be taken to correctly save it.

### • STATUS Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	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.

## Oscillators

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

### Oscillator Overview

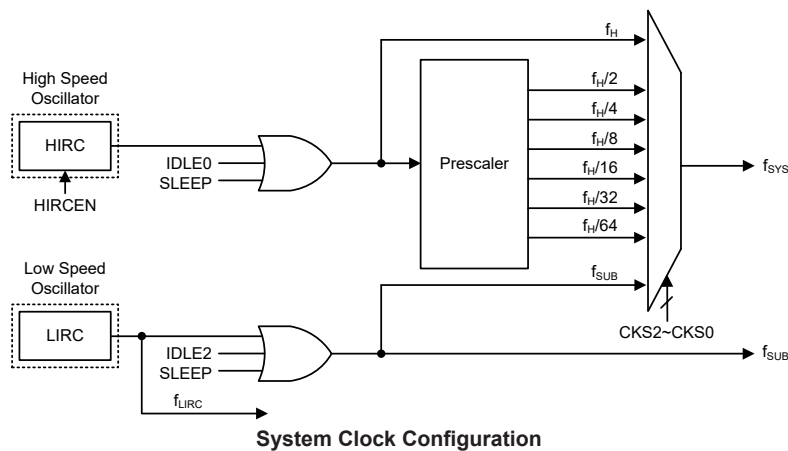
In addition to being the source of the main system clock the oscillators also provide clock sources for the Watchdog Timer and Time Base Interrupt. The fully integrated internal oscillators, requiring no external components, are provided to form a wide range of both fast and slow system oscillators. 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.

Type	Name	Frequency
Internal High Speed RC	HIRC	8MHz
Internal Low Speed RC	LIRC	32kHz

**Oscillator Types**

### System Clock Configurations

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



### **Internal High Speed RC Oscillator – HIRC**

The internal RC oscillator is a fully integrated system oscillator requiring no external components. The internal RC oscillator has one fixed frequency of 8MHz. Device trimming during the manufacturing process and the inclusion of internal frequency compensation circuits are used to ensure that the influence of the power supply voltage, temperature and process variations on the oscillation frequency are minimised.

### **Internal 32kHz Oscillator – LIRC**

The Internal 32kHz System Oscillator is a fully integrated low frequency RC oscillator with a typical frequency of 32kHz, 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.

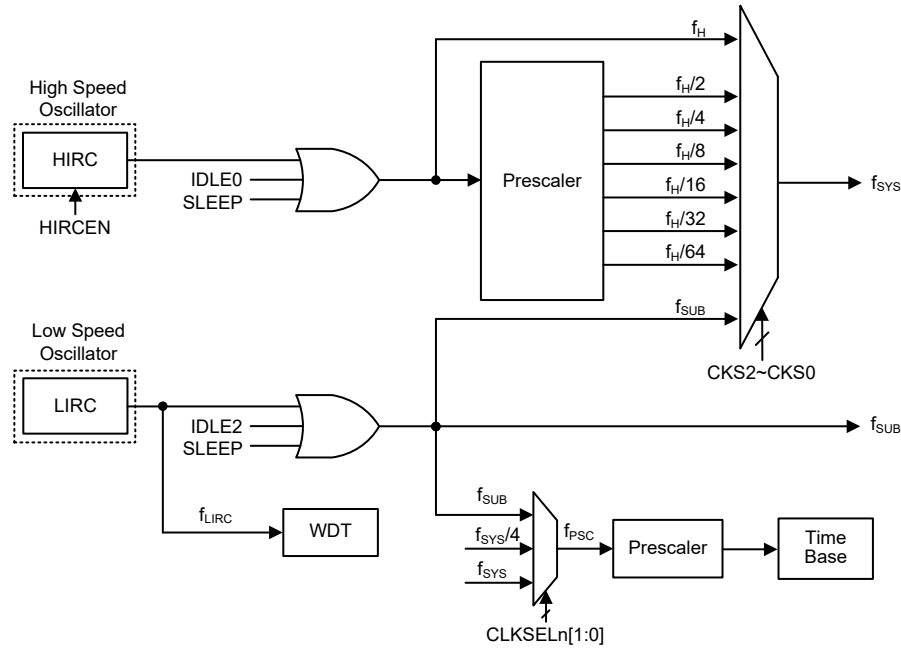
## **Operating Modes and System Clocks**

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 the devices with both high and low speed clock sources and the means to switch between them dynamically, the user can optimise the operation of their microcontroller to achieve the best performance/power ratio.

### **System Clocks**

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

The main system clock, can come from 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 is sourced from the LIRC oscillator. The other choice, which is a divided version of the high speed system oscillator has a range of  $f_H/2 \sim f_H/64$ .



Note: n=0 for BS23A02CA, BS23B04CA; n=0~1 for BS23B08CA.

#### 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,  $f_H \sim f_H/64$ , 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 microcontrollers, each one with its own special characteristics and which can be chosen according to the specific performance and power requirements of the application. There are two modes allowing normal operation of the microcontroller, the FAST Mode and SLOW Mode. The remaining four modes, the SLEEP, IDLE0, IDLE1 and IDLE2 Mode are used when the microcontroller CPU is switched off to conserve power.

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

"x": Don't care

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

2. The  $f_{LIRC}$  clock can be switched on or off which is controlled by the WDT function being enabled or disabled in the SLEEP mode.

**FAST Mode**

This is one of the main operating modes where the microcontrollers have all of their functions operational and where the system clock is provided by the high speed oscillator. This mode operates allowing the microcontrollers to operate normally with a clock source which will come from the high speed oscillator, HIRC. The high speed oscillator will however first be divided by a ratio ranging from 1 to 64, the actual ratio being selected by the CKS2~CKS0 bits in the SCC register. Although a high speed oscillator is used, running the microcontrollers 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  $f_{SUB}$ , which is derived from the LIRC oscillator.

**SLEEP Mode**

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

**IDLE0 Mode**

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

**IDLE1 Mode**

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

**IDLE2 Mode**

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



## Control Register

The SCC register is used to control the system clock and the corresponding oscillator configurations.

### • SCC Register

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

Bit 7~5 **CKS2~CKS0**: System clock selection

000:  $f_H$   
 001:  $f_H/2$   
 010:  $f_H/4$   
 011:  $f_H/8$   
 100:  $f_H/16$   
 101:  $f_H/32$   
 110:  $f_H/64$   
 111:  $f_{SUB}$

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

Bit 4 Unimplemented, read as “0”

Bit 3 **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 2 **HIRCEN**: HIRC oscillator enable control

0: Disable  
 1: Enable

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

0: Disable  
 1: Enable

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

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

0: Disable  
 1: Enable

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

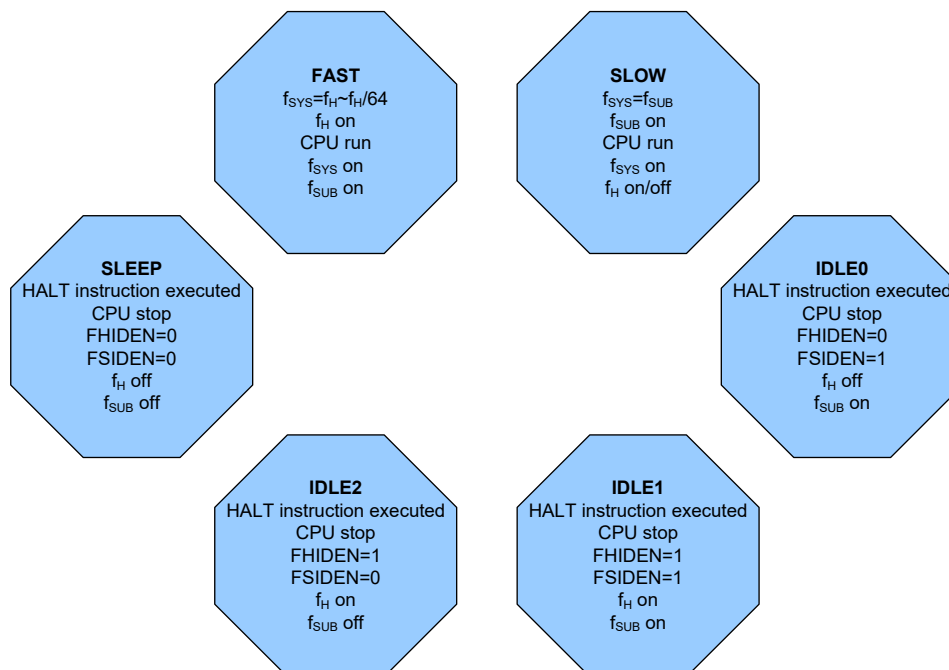
Note: A certain delay is required before the relevant clock is successfully switched to the target clock source after any clock switching setup using the CKS2~CKS0 bits. A proper delay time must be arranged before executing the following operations which require immediate reaction with the target clock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

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

## 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.

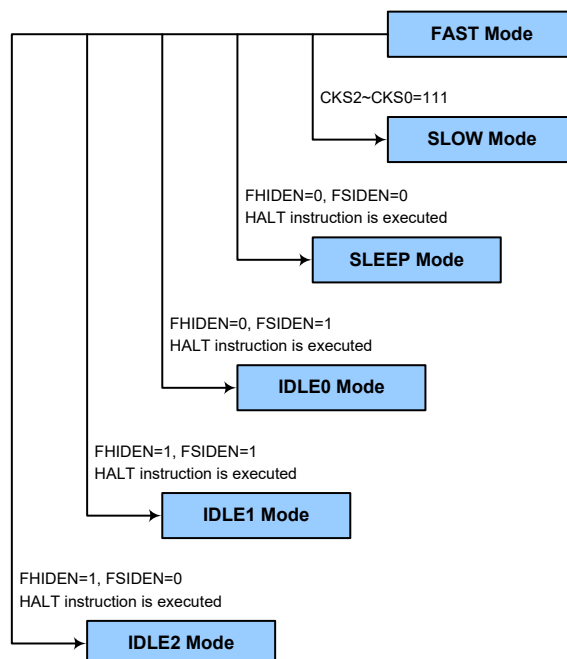
In simple terms, mode switching between the FAST Mode and SLOW Mode is executed using the CKS2~CKS0 bits in the SCC register while mode switching from the FAST/SLOW Modes to the SLEEP/IDLE Modes is executed via the HALT instruction. When an HALT instruction is executed, whether the device enters the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



## FAST Mode to SLOW Mode Switching

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

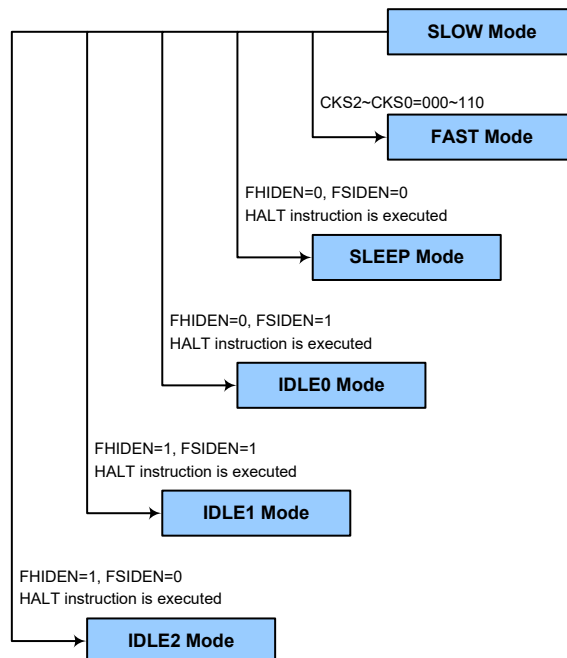
The SLOW Mode system clock is sourced from the LIRC oscillator and therefore requires this oscillator to be stable before full mode switching occurs.



### SLOW Mode to FAST Mode Switching

In the 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  $f_H \sim f_H/64$ .

However, if  $f_H$  is not used in the 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 SCC register. The time duration required for the high speed system oscillator stabilisation is specified in the System Start Up Time Characteristics.



### Entering the SLEEP Mode

There is only one way for the devices to enter the SLEEP Mode and that is to execute the “HALT” instruction in the application program with 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 if the WDT function is enabled. If the WDT function is disabled, the WDT will be cleared and then stopped.

### Entering the IDLE0 Mode

There is only one way for the devices 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 if the WDT function is enabled. If the WDT function is disabled, the WDT will be cleared and then stopped.

### Entering the IDLE1 Mode

There is only one way for the devices 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 if the WDT function is enabled. If the WDT function is disabled, the WDT will be cleared and then stopped.

### Entering the IDLE2 Mode

There is only one way for the devices 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 if the WDT function is enabled. If the WDT function is disabled, the WDT will be cleared and then stopped.

### **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 devices. All high-impedance input pins must be connected to either a fixed high or low level as any floating input pins could create internal oscillations and result in increased current consumption. This also applies to the device which has different package types, as there may be unbonded pins. These must either be set as outputs or if set 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 set 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 device is woken up again, it will take a considerable time for the original system oscillator to restart, stabilise and allow normal operation to resume.

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

- An external falling edge on Port A
- An external  $\overline{RES}$  pin reset
- A system interrupt
- A WDT overflow

When the device executes the “HALT” instruction, it will enter the IDLE or SLEEP mode and the PDF flag will be set to 1. The PDF flag will be cleared to 0 if the device experiences a system power-up or executes the clear Watchdog Timer instruction.

If the system is woken up by an external  $\overline{RES}$  pin reset, the device will experience a full system reset, however, if the device is woken up by a WDT overflow, a Watchdog Timer reset will be initiated. Although both of these wake-up methods will initiate a reset operation, the actual source of the wake-up can be determined by examining the TO and PDF flag. The PDF flag is cleared by a system power-up or executing the clear Watchdog Timer instructions and is set when executing the “HALT” instruction. The TO flag is set if a WDT time-out occurs and causes a wake-up that only resets the Program Counter and Stack Pointer, other flags remain in their original status.

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

## Watchdog Timer

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

### Watchdog Timer Clock Source

The Watchdog Timer clock source is provided by the internal clock,  $f_{LIRC}$  which is sourced from the LIRC oscillator. The LIRC internal oscillator has an approximate frequency of 32kHz and this specified internal clock period can vary with  $V_{DD}$ , temperature and process variations. The Watchdog Timer source clock is then subdivided by a ratio of  $[(2^8-2^0)\sim 2^8]\sim[(2^{15}-2^7)\sim 2^{15}]$  to give longer timeouts, the actual value being chosen using the WS2~WS0 bits in the WDTC register.

### Watchdog Timer Control Register

A single register, WDTC, controls the required time-out period as well as the enable/disable operation.

#### • WDTC Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	WDTEN	WS2	WS1	WS0
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	1	1	1	1

Bit 7~4 Unimplemented, read as “0”

Bit 3 **WDTEN**: WDT function enable control  
 0: Disable  
 1: Enable

Bit 2~0 **WS2~WS0**: WDT time-out period selection  
 000:  $[(2^8-2^0)\sim 2^8]/f_{LIRC}$   
 001:  $[(2^9-2^1)\sim 2^9]/f_{LIRC}$   
 010:  $[(2^{10}-2^2)\sim 2^{10}]/f_{LIRC}$   
 011:  $[(2^{11}-2^3)\sim 2^{11}]/f_{LIRC}$   
 100:  $[(2^{12}-2^4)\sim 2^{12}]/f_{LIRC}$   
 101:  $[(2^{13}-2^5)\sim 2^{13}]/f_{LIRC}$   
 110:  $[(2^{14}-2^6)\sim 2^{14}]/f_{LIRC}$   
 111:  $[(2^{15}-2^7)\sim 2^{15}]/f_{LIRC}$

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

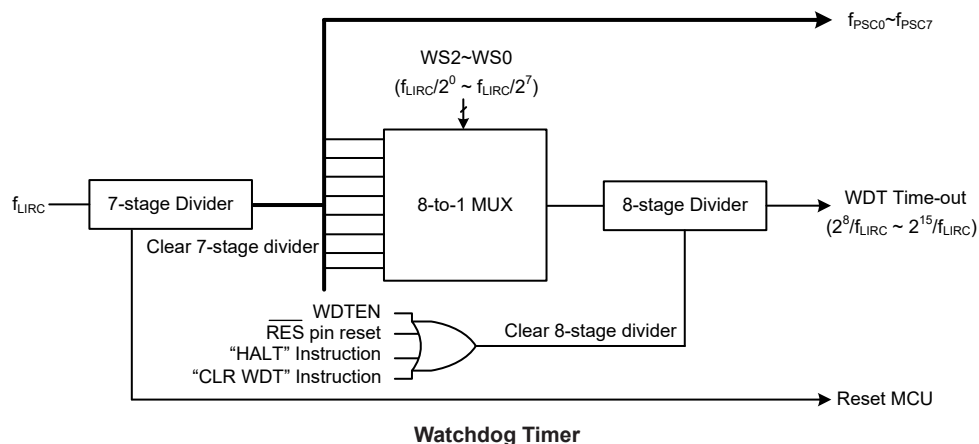
## Watchdog Timer Operation

The Watchdog Timer operates by providing a device reset when its timer overflows. This means that in the application program and during normal operation the user has to strategically clear the Watchdog Timer before it overflows to prevent the Watchdog Timer from executing a reset. This is done using the clear watchdog instruction. If the program malfunctions for whatever reason, jumps to an unknown location, or enters an endless loop, the clear instruction will not be executed in the correct manner, in which case the Watchdog Timer will overflow and reset the device. With regard to the Watchdog Timer enable/disable function, there is one bit, WDTEN, in the WDTC register to offer the enable/disable 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 using the Watchdog Timer software clear instruction, the second is via a HALT instruction. The third is an external hardware reset, which means a low level on the external reset pin.

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

The maximum time-out period is when the  $2^{15}$  division ratio is selected. As an example, with a 32kHz LIRC oscillator as its source clock, this will give a maximum watchdog period of around 1 second for the  $2^{15}$  division ratio, and a minimum timeout of 8ms for the  $2^8$  division ration.



## Reset and Initialisation

A reset function is a fundamental part of any microcontroller ensuring that the device can be set to some predetermined condition irrespective of outside parameters. The most important reset condition is after power is first applied to the microcontrollers. In this case, internal circuitry will ensure that the microcontrollers, 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, situations may arise where it is necessary to forcefully apply a reset condition when the device is running. One example of this is where after power has been applied and the device is already running, the  $\overline{\text{RES}}$  line is forcefully pulled low. In such a case, known as a normal operation reset, some of the registers remain unchanged allowing the device to proceed with normal operation after the reset line is allowed to return high.

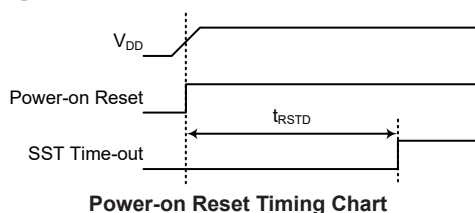
Another reset exists in the form of a Low Voltage Reset, LVR, where a full reset, similar to the  $\overline{\text{RES}}$  reset is implemented in situations where the power supply voltage falls below a certain threshold. Another type of reset is when the Watchdog Timer overflows and resets the microcontroller. 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 both internally and externally.

#### Power-on Reset

The most fundamental and unavoidable reset is the one that occurs after power is first applied to the microcontrollers. 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**

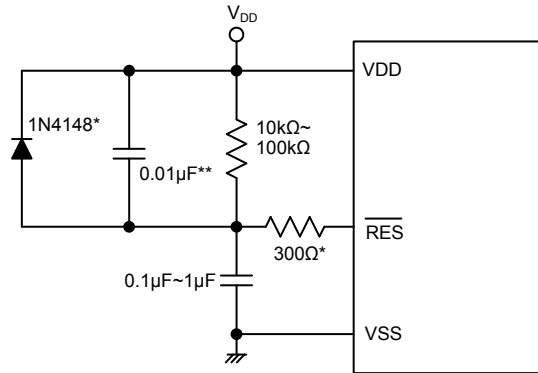
#### $\overline{\text{RES}}$ Pin Reset

As the reset pin is shared with an I/O pin, the reset function must be selected using a configuration option. Although the microcontroller has an internal RC reset function, if the  $V_{DD}$  power supply rise time is not fast enough or does not stabilise quickly at power-on, the internal reset function may be incapable of providing proper reset operation. For this reason it is recommended that an external RC network is connected to the  $\overline{\text{RES}}$  pin, whose additional time delay will ensure that the  $\overline{\text{RES}}$  pin remains low for an extended period to allow the power supply to stabilise. During this time delay, normal operation of the microcontroller will be inhibited. After the  $\overline{\text{RES}}$  line reaches a certain voltage value, the reset delay time  $t_{RSTD}$  is invoked to provide an extra delay time after which the microcontroller will begin normal operation. The abbreviation SST in the figures stands for System Start-up Timer.



For most applications a resistor connected between VDD and the  $\overline{\text{RES}}$  pin and a capacitor connected between VSS and the  $\overline{\text{RES}}$  pin will provide a suitable external reset circuit. Any wiring connected to the  $\overline{\text{RES}}$  pin should be kept as short as possible to minimise any stray noise interference.

For applications that operate within an environment where more noise is present the Enhanced Reset Circuit shown is recommended.

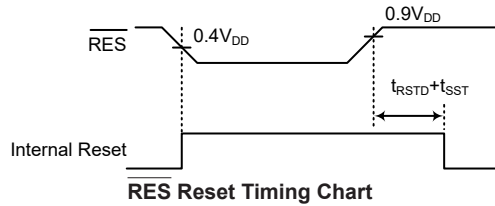


Note: \* It is recommended that this component is added for added ESD protection.

\*\* It is recommended that this component is added in environments where power line noise is significant.

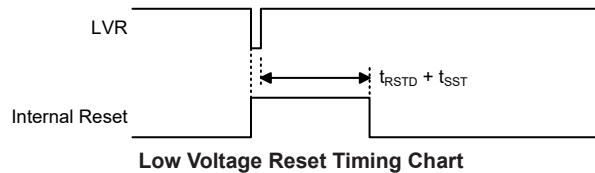
#### External $\overline{\text{RES}}$ Circuit

Pulling the  $\overline{\text{RES}}$  pin low using external hardware will also execute a device reset. In this case, as in the case of other resets, the Program Counter will reset to zero and program execution initiated from this point.



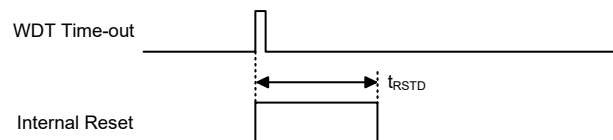
#### Low Voltage Reset – LVR

The microcontroller contains a low voltage reset circuit in order to monitor the supply voltage of the device. The LVR function can be enabled or disabled which is selected by the configuration options. If the supply voltage of the device drops to within a range of 0.9V~V<sub>LVR</sub> such as might occur when changing the battery in battery powered applications, the LVR will automatically reset the device internally. For a valid LVR signal, a low supply voltage, i.e., a voltage in the range between 0.9V~V<sub>LVR</sub> must exist for a time greater than that specified by  $t_{\text{LVR}}$  in the LVR Electrical characteristics. If the low supply voltage state does not exceed this value, the LVR will ignore the low supply voltage and will not perform a reset function. The specified voltage values for V<sub>LVR</sub> can be selected using configuration options.



### Watchdog Time-out Reset during Normal Operation

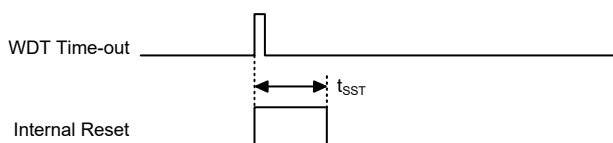
When the Watchdog time-out Reset during normal operations in the FAST or SLOW mode occurs, the Watchdog time-out flag TO will be set to “1”.



WDT Time-out Reset during Normal Operation Timing Chart

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



WDT Time-out Reset during SLEEP 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	RES or 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	Cleared after reset, WDT begins counting
Timer/Event Counter	Timer/Event Counter will be turned off
Input/Output Ports	I/O ports will be set as inputs
Stack Pointer	Stack Pointer will point to the top of the stack

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

Register	BS23A02CA	BS23B04CA	BS23B08CA	Reset (Power On)	WDT Time-out (Normal Operation)	$\overline{\text{RES}}$ Reset (Normal Operation)	WDT Time-out (IDLE/SLEEP)
IAR0	•	•	•	x x x x x x x x	u u u u u u u u	u u u u u u u u	u u u u u u u u
MP0	•	•	•	x x x x x x x x	u u u u u u u u	u u u u u u u u	u u u u u u u u
IAR1		•	•	x x x x x x x x	u u u u u u u u	u u u u u u u u	u u u u u u u u
MP1		•	•	x x x x x x x x	u u u u u u u u	u u u u u u u u	u u u u u u u u
BP			•	- - - - - 0	- - - - - 0	- - - - - 0	- - - - - u
ACC	•	•	•	x x x x x x x x	u u u u u u u u	u u u u u u u u	u u u u u u u u
PCL	•	•	•	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0
TBLP	•	•	•	x x x x x x x x	u u u u u u u u	u u u u u u u u	u u u u u u u u
TBLH	•			- - x x x x x x	- - u u u u u u	- - u u u u u u	- - u u u u u u
		•	•	- x x x x x x x	- u u u u u u u	- u u u u u u u	- u u u u u u u
STATUS	•	•	•	- - 0 0 x x x x	- - 1 u u u u u	- - u u u u u u	- - 1 1 u u u u
INTEG	•	•	•	- - - - - 0 0	- - - - - 0 0	- - - - - 0 0	- - - - - u u
WDTC	•	•	•	- - - - 1 1 1 1	- - - - 1 1 1 1	- - - - 1 1 1 1	- - - - u u u u
TB0C	•	•	•	0 - 0 0 - 0 0 0	0 - 0 0 - 0 0 0	0 - 0 0 - 0 0 0	u - u u - u u u
SCC	•	•	•	1 1 1 - 0 0 0 0	1 1 1 - 0 0 0 0	1 1 1 - 0 0 0 0	u u u - u u u u
INTC0	•	•	•	- 0 0 0 0 0 0 0	- 0 0 0 0 0 0 0	- 0 0 0 0 0 0 0	- u u u u u u u u
PA	•			1 - - 1 1 1 1 1	1 - - 1 1 1 1 1	1 - - 1 1 1 1 1	u - - u u u u u
		•		1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1	u u u u u u u u
			•	1 - - 1 1 1 1 1	1 - - 1 1 1 1 1	1 - - 1 1 1 1 1	u - - u u u u u
PAC	•			1 - - 1 1 1 1 1	1 - - 1 1 1 1 1	1 - - 1 1 1 1 1	u - - u u u u u
		•		1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1	u u u u u u u u
			•	1 - - 1 1 1 1 1	1 - - 1 1 1 1 1	1 - - 1 1 1 1 1	u - - u u u u u
PAPU	•			0 - - 0 0 0 0 0	0 - - 0 0 0 0 0	0 - - 0 0 0 0 0	u - - u u u u u
		•		- 0 0 0 0 0 0 0	- 0 0 0 0 0 0 0	- 0 0 0 0 0 0 0	- u u u u u u u u
			•	0 - - 0 0 0 0 0	0 - - 0 0 0 0 0	0 - - 0 0 0 0 0	- - u u u u u u u
PAWU	•			0 - - 0 0 0 0 0	0 - - 0 0 0 0 0	0 - - 0 0 0 0 0	u - - u u u u u
		•		0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	u u u u u u u u
			•	0 - - 0 0 0 0 0	0 - - 0 0 0 0 0	0 - - 0 0 0 0 0	u - - u u u u u
TB1C			•	0 - 0 0 - 0 0 0	0 - 0 0 - 0 0 0	0 - 0 0 - 0 0 0	u - u u - u u u
TKTMR	•	•	•	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	u u u u u u u u
TKC0	•	•		- 0 0 0 0 - 0 0	- 0 0 0 0 0 - 0	- 0 0 0 0 0 - 0	- u u u u - u u
			•	- 0 0 0 0 0 0 0	- 0 0 0 0 0 0 0	- 0 0 0 0 0 0 0	- u u u u u u u u
TKC1	•	•	•	- - - - - 1 1	- - - - - 1 1	- - - - - 1 1	- - - - - u u
TK16DL	•	•	•	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	u u u u u u u u
TK16DH	•	•	•	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	u u u u u u u u
TKM0C0	•	•	•	0 0 0 - 0 0 0 0	0 0 0 - 0 0 0 0	0 0 0 - 0 0 0 0	u u u - u u u u
TKM0C1	•			0 - 0 0 - - 0 0	0 - 0 0 - - 0 0	0 - 0 0 - - 0 0	u - u u - - u u
		•	•	0 - 0 0 0 0 0 0	0 - 0 0 0 0 0 0	0 - 0 0 0 0 0 0	u - u u u u u u
TKM016DL	•	•	•	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	u u u u u u u u
TKM016DH	•	•	•	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	u u u u u u u u
TKM0ROL	•	•	•	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	u u u u u u u u

Register	BS23A02CA	BS23B04CA	BS23B08CA	Reset (Power On)	WDT Time-out (Normal Operation)	$\overline{\text{RES}}$ Reset (Normal Operation)	WDT Time-out (IDLE/SLEEP)
TKM0ROH	•	•	•	---- --00	---- --00	---- --00	---- --uu
INTC1		•		-000 -000	-000 -000	-000 -000	-uuu -uuu
			•	0000 0000	0000 0000	0000 0000	uuuu uuuu
MFI			•	-000 -000	-000 -000	-000 -000	-uuu -uuu
PB		•		---- ---1	---- ---1	---- ---1	---- ---u
			•	1111 1111	1111 1111	1111 1111	uuuu uuuu
PBC		•		---- ---1	---- ---1	---- ---1	---- ---u
			•	1111 1111	1111 1111	1111 1111	uuuu uuuu
PBPU		•		---- ---0	---- ---0	---- ---0	---- ---u
			•	0000 0000	0000 0000	0000 0000	uuuu uuuu
TMR0C/PWM0C		•	•	00-0 0000	00-0 0000	00-0 0000	uu-u uuuu
TMR0/PWM0DATA		•	•	0000 0000	0000 0000	0000 0000	uuuu uuuu
TMR1C/PWM1C		•	•	00-0 0000	00-0 0000	00-0 0000	uu-u uuuu
TMR1/PWM1DATA		•	•	0000 0000	0000 0000	0000 0000	uuuu uuuu
OCR		•	•	---- 00--	---- 00--	---- 00--	---- uu--
ODL		•	•	0000 0000	0000 0000	0000 0000	uuuu uuuu
ODH		•	•	-000 0000	-000 0000	-000 0000	-uuu uuuu
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
PAS0	•			---- --00	---- --00	---- --00	---- --uu
		•	•	0000 0000	0000 0000	0000 0000	uuuu uuuu
PAS1		•		0000 0000	0000 0000	0000 0000	uuuu uuuu
			•	00-- --00	00-- --00	00-- --00	uu-- --uu
PBS0		•		---- --00	---- --00	---- --00	---- --uu
			•	0000 0000	0000 0000	0000 0000	uuuu uuuu
IFS		•	•	---0 0000	---0 0000	---0 0000	---u uuuu
TKM1C0			•	000- 0000	000- 0000	000- 0000	uuu- uuuu
TKM1C1			•	0-00 0000	0-00 0000	0-00 0000	u-uu uuuu
TKM116DL			•	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM116DH			•	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM1ROL			•	0000 0000	0000 0000	0000 0000	uuuu uuuu
TKM1ROH			•	---- --00	---- --00	---- --00	---- --uu
TMR2C/PWM2C			•	00-0 0000	00-0 0000	00-0 0000	uu-u uuuu
TMR2/PWM2DATA			•	0000 0000	0000 0000	0000 0000	uuuu uuuu
TMR3C/PWM3C			•	00-0 0000	00-0 0000	00-0 0000	uu-u uuuu
TMR3/PWM3DATA			•	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 devices provide bidirectional input/output lines labeled with port names PA~PB. These I/O ports are mapped to the RAM Data Memory with specific addresses as shown in the Special Purpose Data Memory table. All of these I/O ports can be used for input and output operations. For input operation, these ports are non-latching, which means the inputs must be ready at the T2 rising edge of instruction “MOV A, [m]”, where m denotes the port address. For output operation, all the data is latched and remains unchanged until the output latch is rewritten.

Register Name	Bit							
	7	6	5	4	3	2	1	0
PA	PA7	—	—	PA4	PA3	PA2	PA1	PA0
PAC	PAC7	—	—	PAC4	PAC3	PAC2	PAC1	PAC0
PAPU	PAPU7	—	—	PAPU4	PAPU3	PAPU2	PAPU1	PAPU0
PAWU	PAWU7	—	—	PAWU4	PAWU3	PAWU2	PAWU1	PAWU0

“—”: Unimplemented, read as “0”

**I/O Logic Function Register List – BS23A02CA**

Register Name	Bit							
	7	6	5	4	3	2	1	0
PA	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0
PAC	PAC7	PAC6	PAC5	PAC4	PAC3	PAC2	PAC1	PAC0
PAPU	PAPU7	PAPU6	PAPU5	PAPU4	PAPU3	PAPU2	PAPU1	PAPU0
PAW	PAWU7	PAWU6	PAWU5	PAWU4	PAWU3	PAWU2	PAWU1	PAWU0
PB	—	—	—	—	—	—	—	PB0
PBC	—	—	—	—	—	—	—	PBC0
PBPU	—	—	—	—	—	—	—	PBPU0

“—”: Unimplemented, read as “0”

**I/O Logic Function Register List – BS23B04CA**

Register Name	Bit							
	7	6	5	4	3	2	1	0
PA	PA7	—	PA5	PA4	PA3	PA2	PA1	PA0
PAC	PAC7	—	PAC5	PAC4	PAC3	PAC2	PAC1	PAC0
PAPU	PAPU7	—	PAPU5	PAPU4	PAPU3	PAPU2	PAPU1	PAPU0
PAWU	PAWU7	—	PAWU5	PAWU4	PAWU3	PAWU2	PAWU1	PAWU0
PB	PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0
PBC	PBC7	PBC6	PBC5	PBC4	PBC3	PBC2	PBC1	PBC0
PBPU	PBPU7	PBPU6	PBPU5	PBPU4	PBPU3	PBPU2	PBPU1	PBPU0

“—”: Unimplemented, read as “0”

**I/O Logic Function Register List – BS23B08CA**

## 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 and are implemented using weak PMOS transistors.

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

### • PxPU Register

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

Bit 7~0 **PxPU7~PxPU0**: Port x bit 7~bit 0 pull-high function control

0: Disable

1: Enable

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

## Port A Wake-up

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

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

### • PAWU Register

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

**PAWUn**: Port A Pin wake-up function control

0: Disable

1: Enable

The actual available bits may be different.

## I/O Port Control Registers

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

### • PxC Register

Bit	7	6	5	4	3	2	1	0
Name	PxC7	PxC6	PxC5	PxC4	PxAC3	PxAC2	PxC1	PxC0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	1	1	1	1	1	1	1	1

**PxCn:** I/O Port x Pin type selection

0: Output

1: Input

The PxCn bit is used to control the pin type selection. Here the “x” can be A or B. However, the actual available bits for each I/O Port may be different.

## Pin-shared Functions

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

### Pin-shared Function Selection Registers

The limited number of supplied pins in a package can impose restrictions on the amount of functions a certain device can contain. However by allowing the same pins to share several different functions and providing a means of function selection, a wide range of different functions can be incorporated into even relatively small package sizes. The devices include a Port “x” Output Function Selection register “n”, labeled as PxCn, 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 TCn, 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.

Register Name	Bit							
	7	6	5	4	3	2	1	0
PAS0 (BS23A02CA)	—	—	—	—	—	—	PAS01	PAS00
PAS0 (BS23B04CA/ BS23B08CA)	PAS07	PAS06	PAS05	PAS04	PAS03	PAS02	PAS01	PAS00
PAS1 (BS23B04CA)	PAS17	PAS16	PAS15	PAS14	PAS13	PAS12	PAS11	PAS10
PAS1 (BS23B08CA)	PAS17	PAS16	—	—	—	—	PAS11	PAS10
PBS0 (BS23B04CA)	—	—	—	—	—	—	PBS01	PBS00
PBS0 (BS23B08CA)	PBS07	PBS06	PBS05	PBS04	PBS03	PBS02	PBS01	PBS00
IFS (BS23B04CA)	—	—	—	INTPS1	INTPS0	SCLPS	SDAPS1	SDAPS0
IFS (BS23B08CA)	—	—	—	TC0PS	INTPS1	INTPS0	SCLPS	SDAPS

**Pin-shared Function Selection Register List**

• **PAS0 Register – BS23A02CA**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	PAS01	PAS00
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2 Unimplemented, read as “0”

Bit 1 **PAS01**: PA1 Pin-Shared function selection  
 0: PA1  
 1: KEY2

Bit 0 **PAS00**: PA0 Pin-Shared function selection  
 0: PA0/INT  
 1: KEY1

• **PAS0 Register – BS23B04CA**

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

Bit 7~6 **PAS07~PAS06**: PA3 Pin-Shared function selection  
 00: PA3  
 01: KEY3  
 10: PA3  
 11: PA3

Bit 5~4 **PAS05~PAS04**: PA2 Pin-Shared function selection  
 00: PA2  
 01: PWM1O  
 10: PWM0OB  
 11: SDA

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

Bit 1~0 **PAS01~PAS00**: PA0 Pin-Shared function selection  
 00: PA0/INT/TC1  
 01: PWM0O  
 10: SCL  
 11: PA0/INT/TC1



• **PAS0 Register – BS23B08CA**

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

Bit 7~6 **PAS07~PAS06:** PA3 Pin-Shared function selection

00: PA3/TC3  
 01: PWM2O  
 10: PWM1OB  
 11: SDA

Bit 5~4 **PAS05~PAS04:** PA2 Pin-Shared function selection

00: PA2  
 01: PWM1O  
 10: PWM3OB  
 11: SCL

Bit 3~2 **PAS03~PAS02:** PA1 Pin-Shared function selection

00: PA1/TC1  
 01: PWM0O  
 10: PWM2O  
 11: PWM3O

Bit 1~0 **PAS01~PAS00:** PA0 Pin-Shared function selection

00: PA0/INT  
 01: PWM0O  
 10: SDA  
 11: PA0/INT

• **PAS1 Register – BS23B04CA**

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

Bit 7~6 **PAS17~PAS16:** PA7 Pin-Shared function selection

00: PA7/INT/RES  
 01: PWM0O  
 10: PWM1O  
 11: SDA

Bit 5~4 **PAS15~PAS14:** PA6 Pin-Shared function selection

00: PA6/INT/TC0  
 01: PWM1O  
 10: PWM1OB  
 11: SCL

Bit 3~2 **PAS13~PAS12:** PA5 Pin-Shared function selection

00: PA5  
 01: KEY1  
 10: PA5  
 11: PA5

Bit 1~0 **PAS11~PAS10:** PA4 Pin-Shared function selection

00: PA4  
 01: KEY4  
 10: PA4  
 11: PA4

**• PAS1 Register – BS23B08CA**

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

Bit 7~6 **PAS17~PAS16:** PA7 Pin-Shared function selection

00: PA7/INT/TC0/RES/VPP

01: PWM3O

10: PWM2OB

11: PA7/INT/TC0/ $\overline{\text{RES}}$ /VPP

Bit 5~2 Unimplemented, read as “0”

Bit 1~0 **PAS11~PAS10:** PA4 Pin-Shared function selection

00: PA4/INT/TC2

01: PWM1O

10: PWM0OB

11: SCL

**• PBS0 Register – BS23B04CA**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	PBS01	PBS00
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2 Unimplemented, read as “0”

Bit 1~0 **PBS01~PBS00:** PB0 Pin-Shared function selection

00: PB0/INT

01: PWM0O

10: PWM1O

11: SDA

**• PBS0 Register – BS23B08CA**

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

Bit 7 **PBS07:** PB7 Pin-Shared function selection

0: PB7

1: KEY8

Bit 6 **PBS06:** PB6 Pin-Shared function selection

0: PB6

1: KEY7

Bit 5 **PBS05:** PB5 Pin-Shared function selection

0: PB5

1: KEY6

Bit 4 **PBS04:** PB4 Pin-Shared function selection

0: PB4

1: KEY5

Bit 3 **PBS03:** PB3 Pin-Shared function selection

0: PB3

1: KEY4

Bit 2 **PBS02:** PB2 Pin-Shared function selection

0: PB2

1: KEY3

- Bit 1      **PBS01**: PB1 Pin-Shared function selection  
             0: PB1  
             1: KEY2
- Bit 0      **PBS00**: PB0 Pin-Shared function selection  
             0: PB0  
             1: KEY1

• **IFS Register – BS23B04CA**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	INTPS1	INTPS0	SCLPS	SDAPS1	SDAPS0
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 4~3      **INTPS1~INTPS0**: INT input source pin selection  
             00: PA7  
             01: PA0  
             10: PA6  
             11: PB0
- Bit 2      **SCLPS**: SCL input source pin selection  
             0: PA0  
             1: PA6
- Bit 1~0      **SDAPS1~SDAPS0**: SDA input source pin selection  
             00: PA2  
             01: PB0  
             10: PA7  
             11: PA2

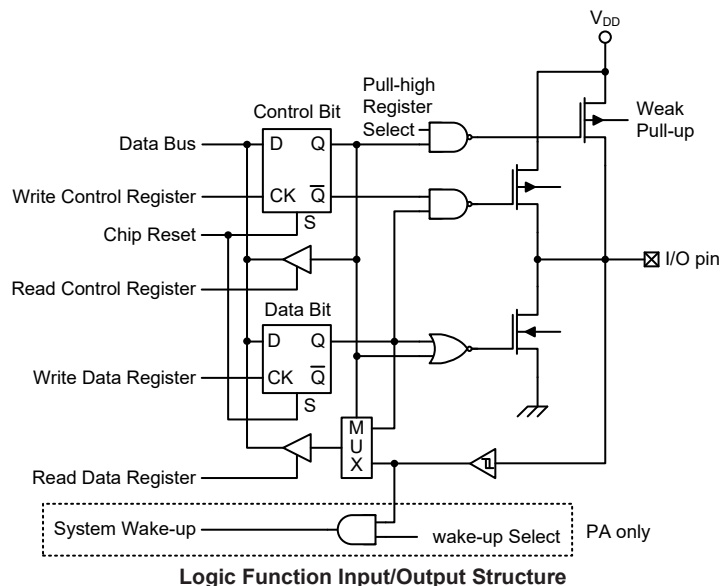
• **IFS Register – BS23B08CA**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	TC0PS	INTPS1	INTPS0	SCLPS	SDAPS
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      **TC0PS**: TC0 input source pin selection  
             0: Reserved  
             1: PA7  
             This bit should be kept as “1”.
- Bit 3~2      **INTPS1~INTPS0**: INT input source pin selection  
             00: PA7  
             01 : PA0  
             10 : PA4  
             11 : Reserved
- Bit 1      **SCLPS**: SCL input source pin selection  
             0: PA2  
             1: PA4
- Bit 0      **SDAPS**: SDA input source pin selection  
             0: PA0  
             1: PA3

## I/O Pin Structure

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.



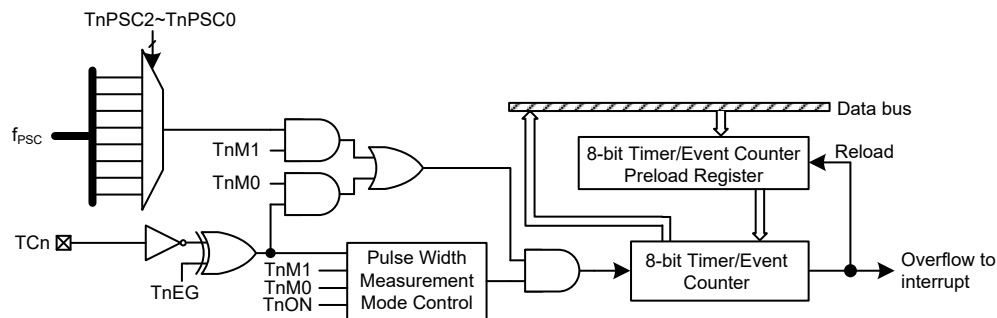
## Programming Considerations

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

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

## 8-bit Timer/Event Counter – BS23B04CA and BS23B08CA

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 devices contain several 8-bit Timer/Event Counters, each of which is an 8-bit programmable count-up counter and the clock may come from an external or internal clock source. As this timer has four different operating modes, it can be configured to operate as a general timer, an external event counter, a pulse width measurement or a pulse width modulation function.



Note: n=0~1 for BS23B04CA; n=0~3 for BS23B08CA.

**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, Pulse Width Measurement Mode and PWM Mode. For the Timer/Event Counter, this internal clock is sourced from the Time Base prescaler, which is selected by the TnPSC2~TnPSC0 bits in the TMRnC 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 TCn pin. Depending upon the condition of the TnEG 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 several registers related to the Timer/Event Counter. The first is the TMRn register that contains the actual value of the timer and into which an initial value can be preloaded. Writing to the TMRn register will transfer the specified data to the Timer/Event Counter. Reading the TMRn register will read the contents of the Timer/Event Counter. The second is the TMRnC control register, which is used to define the operating mode, control the counting enable or disable and select the active edge. Another two registers control the overall operation of the Pulse Width Modulator, the data register, PWMnDATA and the control register, PWMnC. Note that the PWMnC and TMRnC registers have the same register address, the PWMnDATA and TMRn registers also have the same register address.

Register Name	Bit							
	7	6	5	4	3	2	1	0
TMRnC	TnM1	TnM0	—	TnON	TnEG	TnPSC2	TnPSC1	TnPSC0
PWMnC	TnM1	TnM0	—	PWMnDIV2	PWMnDIV1	PWMnDIV0	PWMnSEL	PWMnEN
TMRn	TMRnD7	TMRnD6	TMRnD5	TMRnD4	TMRnD3	TMRnD2	TMRnD1	TMRnD0
PWMnDATA	PWMnD7	PWMnD6	PWMnD5	PWMnD4	PWMnD3	PWMnD2	PWMnD1	PWMnD0

Note: n=0~1 for BS23B04CA; n=0~3 for BS23B08CA.

**Timer/Event Counter Register List**

### Timer Register – TMRn

The timer register TMRn 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. 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.

#### • TMRn register

Bit	7	6	5	4	3	2	1	0
Name	TMRnD7	TMRnD6	TMRnD5	TMRnD4	TMRnD3	TMRnD2	TMRnD1	TMRnD0
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      **TMRnD7~TMRnD0**: Timer preload register byte

### Timer Control Register – TMRnC

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

The Timer Control Register is known as TMRnC. 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 four modes the timer is to operate in, namely the PWM Mode, Timer Mode, the Event Counter Mode or the Pulse Width Measurement Mode, the TnM1~TnM0 bits in the Timer Control Register must be set to the required logic levels. The timer-on bit TnON 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  $f_{PSC}$  is used, it can be selected by properly setting the TnPSC2~TnPSC0 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 TnEG bit in the TMRnC Register.

#### • TMRnC Register

Bit	7	6	5	4	3	2	1	0
Name	TnM1	TnM0	—	TnON	TnEG	TnPSC2	TnPSC1	TnPSC0
R/W	R/W	R/W	—	R/W	R/W	R/W	R/W	R/W
POR	0	0	—	0	0	0	0	0

Bit 7~6      **TnM1~TnM0**: Timer/Event Counter operating mode selection

- 00: PWM Mode
- 01: Event Counter Mode
- 10: Timer Mode
- 11: Pulse Width Measurement Mode

Bit 5	Unimplemented, read as “0”
Bit 4	<b>TnON</b> : Timer/Event Counter counting enable 0: Disable 1: Enable  Note that this bit is invalid in the PWM Mode.
Bit 3	<b>TnEG</b> : 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 PWM Mode Unused
Bit 2~0	<b>TnPSC2~TnPSC0</b> : Timer internal clock selection 000: $f_{PSC}/2^0$ 001: $f_{PSC}/2^1$ 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 PWM Mode Registers – PWMnC, PWMnDATA

There are two registers which control the overall operation of the Pulse Width Modulator channel. These are the data register, PWMnDATA and a single control register, PWMnC. Note that the PWMnC register is the same register address as TMRnC register. Additionally, the PWMnDATA and TMRn registers also have the same register address.

##### • PWMnC Register

Bit	7	6	5	4	3	2	1	0
Name	TnM1	TnM0	—	PWMnDIV2	PWMnDIV1	PWMnDIV0	PWMnSEL	PWMnEN
R/W	R/W	R/W	—	R/W	R/W	R/W	R/W	R/W
POR	0	0	—	0	0	0	0	0

Bit 7~6	<b>TnM1~TnM0</b> : Timer/Event Counter operating mode selection 00: PWM Mode 01: Event Counter Mode 10: Timer Mode 11: Pulse Width Measurement Mode
---------	---

Bit 5 Unimplemented, read as “0”

Bit 4~2	<b>PWMnDIV2~PWMnDIV0</b> : $f_{DIV}$ Frequency selection 000: $f_{DIV}=f_{SYS}$ 001: $f_{DIV}=f_{SYS}/2$ 010: $f_{DIV}=f_{SYS}/3$ 011: $f_{DIV}=f_{SYS}/4$ 100: $f_{DIV}=f_{SYS}/8$ 101: $f_{DIV}=f_{SYS}/16$ 110: $f_{DIV}=f_{SYS}/32$ 111: $f_{DIV}=f_{SYS}/64$
---------	---

Bit 1 **PWMnSEL**: PWMn mode selection  
 0: (6+2) bits mode  
 1: (7+1) bits mode

Bit 0 **PWMnEN**: PWMn enable control  
 0: Disable  
 1: Enable

Note: 1. When the PWMnEN bit is cleared to zero to disable the PWM function, the internal PWMnO signal will be pulled low. However, the external PWMnO pin status will be floating when the multi-functional pin is selected as a PWMnO output and the PWMnEN bit is cleared to zero.

2. After the PWMnEN bit is set high, the first PWM modulation cycle period and cycle may not match the expected waveform. The PWM output will be normal after the first PWM cycle.

3. The PWMnOB is the inverse output of the PWMnO, which can generate a complementary output.

#### • PWMnDATA Register

Bit	7	6	5	4	3	2	1	0
Name	PWMnD7	PWMnD6	PWMnD5	PWMnD4	PWMnD3	PWMnD2	PWMnD1	PWMnD0
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 **PWMnD7~PWMnD0**: PWMn output duty cycle PWMnD bit 7 ~ bit 0

Note that the duty cycle value PWMnD, once being changed, will reflect on the PWMnO output signal immediately. Therefore, a sudden PWM duty change will occur in the current PWM cycle, resulting in undesired waveform, which only lasts a PWM cycle. Starting from the next new PWM cycle, the PWM duty will be in accordance with the new PWMnD value.

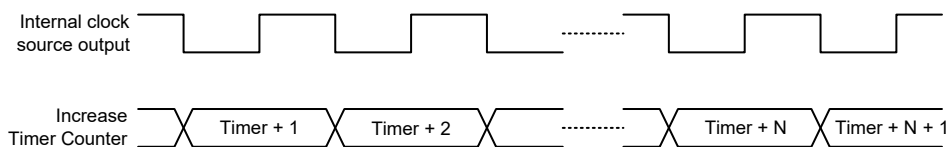
### Timer/Event Counter Operating Modes

The Timer/Event Counter can operate in one of four operating modes, PWM Mode, Timer Mode, Event Counter Mode or Pulse Width Measurement Mode. The operating mode is selected using the TnM1 and TnM0 bits in the TMRnC register.

#### Timer Mode

To select this mode, bits TnM1 and TnM0 in the TMRnC 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  $f_{PSC}$  is used as the timer clock. The clock source is from the Time Base prescaler, and is selected by the TnPSC2~TnPSC0 bits in the TMRnC register. The timer-on bit TnON 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, FF Hex, value and overflows, an interrupt request 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 the timer will continue to count.



**Timer Mode Timing Chart**

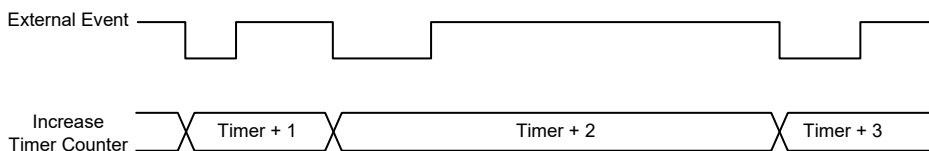


### Event Counter Mode

To select this mode, bits TnM1 and TnM0 in the TMRnC register should be set to “01” respectively. In this mode, a number of externally changing logic events, occurring on the external timer TCn pin, can be recorded by the Timer/Event Counter.

When operating in this mode, the external timer pin, TCn, 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 TnON, can be set high to enable the Timer/Event Counter. If the Active Edge Selection bit, TnEG, is low, the Timer/Event Counter will increase each time the TCn pin receives a low to high transition. If the TnEG bit is high, the counter will increase each time the TCn 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.

As the external timer pin TCn is pin-shared with other pin functions, the TCn pin function must first be selected using relevant pin-shared function selection bits. The pin must also 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 TCn pin. As a result when the timer overflows it will generate a timer interrupt and corresponding wake-up source.



**Event Counter Mode Timing Chart (TnEG=1)**

### Pulse Width Measurement Mode

To select this mode, bits TnM1 and TnM0 in the TMRnC 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.

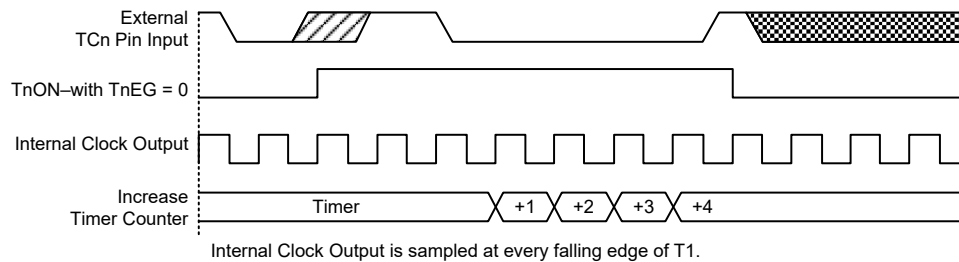
When operating in this mode the internal clock  $f_{PSC}$  is used as the timer clock. The clock source is from the Time Base prescaler, the division ratio is determined by the TnPSC2~TnPSC0 bits in the TMRnC register. After the other bits in the Timer Control Register have been properly configured, the enable bit TnON, 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 TCn pin.

If the active edge selection bit TnEG is low, once a high to low transition has been received on the TCn pin, the Timer/Event Counter will start counting based on the internal selected clock source until the TCn 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 TCn 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 TCn pin returns to its original level, whereas in the event count or timer mode 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 TCn 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.

As the external timer pin TCn is pin-shared with other pin functions, the TCn pin function must first be selected using relevant pin-shared function selection bits. The pin must also be set as an input by setting the corresponding bit in the port control register. It should be noted that in the Pulse Width Capture 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 TCn pin if the internal clock source is still activated and the external signal continues to change state. As a result when the timer overflows it will generate a timer interrupt and corresponding wake-up source.



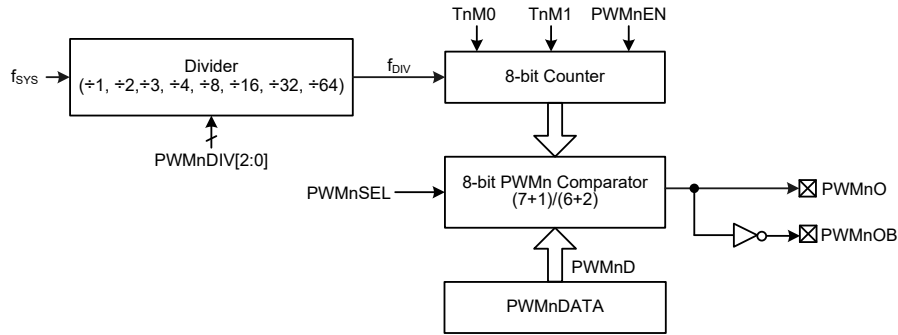
**Pulse Width Measurement Mode Timing Chart (TnEG=0)**

## PWM Mode

To select this mode, bits TnM1 and TnM0 in the PWMnC register should be set to “00” respectively. In this mode, the Timer/Event Counter can be used to make PWM waveform with the clock source coming from the internal selected clock source.

The PWMnC register and PWMnDATA register are assigned to the Pulse Width Modulator channel. The PWM channel has a data register, PWMnDATA, the content of which is an 8-bit data, abbreviated as PWMnD, representing the overall duty cycle of one modulation cycle of the output waveform. To increase the PWM modulation frequency, each modulation cycle is subdivided into two or four individual modulation subsections, known as the (7+1) bits mode or (6+2) bits mode respectively. The PWM counter clock frequency,  $f_{DIV}$ , comes from  $f_{SYS}$  or its division. The required mode, clock source selection and the enable/disable control for the PWM channel are selected using the PWMnC register. Note that when using the PWM, it is only necessary to write the required value into the PWMnDATA register and select the required mode, clock source and enable/disable control using the PWMnC register, the subdivision of the waveform into its sub-modulation cycles is implemented automatically within the microcontroller hardware. Note that the PWMnOB pin is the inverse output of the PWMnO, which can generate a complementary output and supplying more power to connected interfaces such as buzzers.

This method of dividing the original modulation cycle into a further 2 or 4 sub-cycles enables the generation of higher PWM frequencies which allow a wider range of applications to be served. The difference between what is known as the PWM cycle frequency and the PWM modulation frequency should be understood. As the PWM clock is  $f_{DIV}$ , and as the PWM value is 8-bits wide, the overall PWM cycle frequency is  $f_{DIV}/256$ . However, when in the (7+1) mode the PWM modulation frequency will be  $f_{DIV}/128$ , while the PWM modulation frequency for the (6+2) mode will be  $f_{DIV}/64$ .



**PWM Mode Block Diagram**

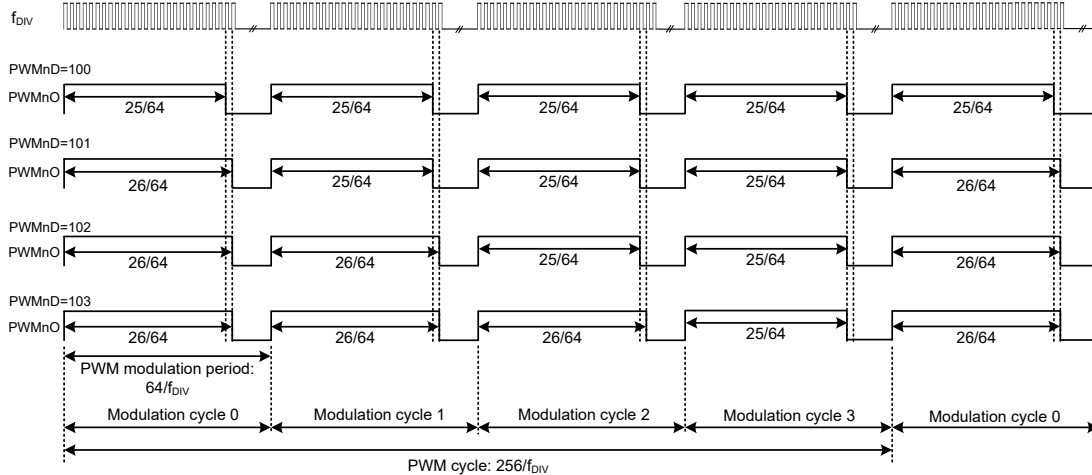
### (6+2) Bits PWM Mode Modulation

A (6+2) bits mode PWM cycle is divided into four modulation cycles, which are named as Modulation cycle 0 ~ Modulation cycle 3. Each modulation cycle has 64 PWM input clock periods. In the (6+2) bits PWM mode, the  $PWMnD$  is divided into two groups. Group 1 is denoted by DC which is the value of  $PWMnD$  bit 7 ~ bit 2. Group 2 is denoted by AC which is the value of  $PWMnD$  bit 1 ~ bit 0. The modulation frequency, modulation cycle duty, PWM cycle frequency and PWM cycle duty of the (6+2) bits mode PWM output signals are summarized in the following table.

Modulation Frequency	Modulation Cycle i	Modulation Cycle Duty		PWM Cycle Frequency	PWM Cycle Duty
$f_{DIV}/64$	$i=0\sim3$	$i < AC$	$(DC+1)/64$	$f_{DIV}/256$	$PWMnD/256$
		$i \geq AC$	$DC/64$		

**(6+2) Bits PWM Mode Summary**

The following diagram illustrates the waveforms associated with the (6+2) bits mode of PWM operation. It is important to note how the single PWM cycle is subdivided into 4 individual modulation cycles, numbered from 0~3 and how the AC value is related to the PWM value. The waveforms of PWM outputs are as shown below.

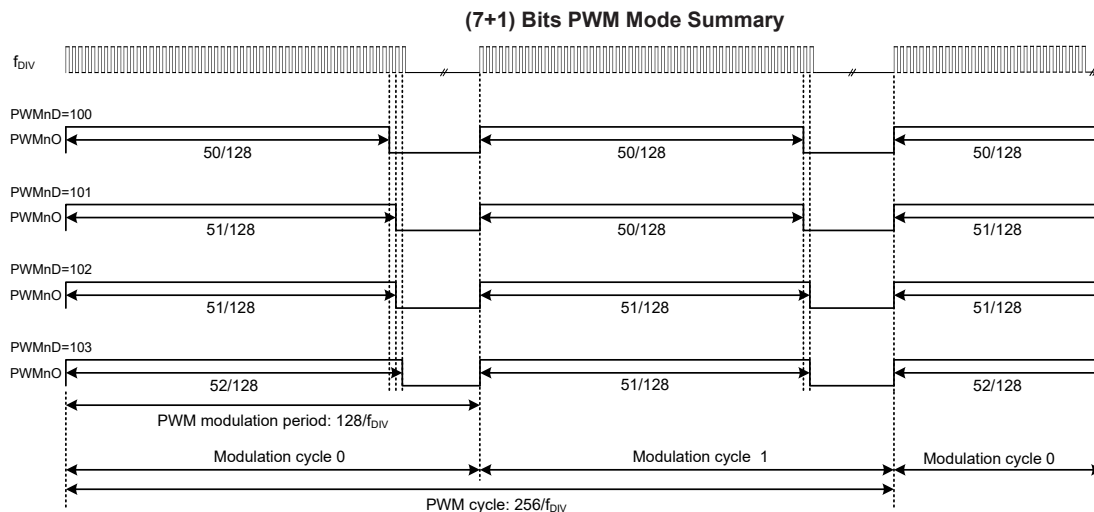


**(6+2) Bits PWM Mode Modulation Waveform**

### (7+1) Bits PWM Mode Modulation

A (7+1) bits mode PWM cycle is divided into two modulation cycles, which is named as Modulation cycle 0 ~ Modulation cycle 1. Each modulation cycle has 128 PWM input clock periods. In the (7+1) bits PWM mode, the PWMnD is divided into two groups. Group 1 is denoted by DC which is the value of PWMnD bit 7 ~ bit 1. Group 2 is denoted by AC which is the value of PWMnD bit 0. The modulation frequency, modulation cycle duty, PWM cycle frequency and PWM cycle duty of the (7+1) bits mode PWM output signals are summarized in the following table.

Modulation Frequency	Modulation Cycle i	Modulation Cycle Duty		PWM Cycle Frequency	PWM Cycle Duty
$f_{DIV}/128$	$i=0\sim1$	$i<AC$	$(DC+1)/128$	$f_{DIV}/256$	PWMnD/256
		$i\geq AC$	$DC/128$		



**(7+1) Bits PWM Mode Modulation Waveform**

## Touch Key Function

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

### Touch Key Structure

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

Device	Total Key Number	Touch Key Module		Touch Key	Shared I/O Pin
BS23A02CA	2	Mn (n=0)	M0	KEY1~KEY2	PA0~PA1
BS23B04CA	4	Mn (n=0)	M0	KEY1~KEY4	PA1, PA3~PA5
BS23B08CA	8	Mn (n=0~1)	M0	KEY1~KEY4	PB0~PB3
			M1	KEY5~KEY8	PB4~PB7

**Touch Key Structure**

## Touch Key Register Definition

Each touch key module, which contains four touch key functions, has its own suite registers. The following table shows the register set for each touch key module. The Mn within the register name refers to the Touch Key module number. The series of devices has up to two Touch Key Modules dependent upon the selected device.

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
TK16DL	Touch key function 16-bit counter low byte
TK16DH	Touch key function 16-bit counter high byte
TKMn16DL	Touch key module n 16-bit C/F counter low byte
TKMn16DH	Touch key module n 16-bit C/F counter high byte
TKMnROL	Touch key module n reference oscillator capacitor select low byte
TKMnROH	Touch key module n reference oscillator capacitor select high byte
TKMnC0	Touch key module n Control register 0
TKMnC1	Touch key module n Control register 1

**Touch Key Function Register Definition (n=0~1)**

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	—	TK16S1	TK16S0
TKC1	—	—	—	—	—	—	TKFS1	TKFS0
TK16DL	D7	D6	D5	D4	D3	D2	D1	D0
TK16DH	D15	D14	D13	D12	D11	D10	D9	D8
TKM0C0	M0MXS1	M0MXS0	M0DFEN	—	M0SOFC	M0SOF2	M0SOF1	M0SOF0
TKM0C1	M0TSS	—	M0ROEN	M0KOEN	—	—	M0K2EN	M0K1EN
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

**Touch Key Function Register List – BS23A02CA**

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	—	TK16S1	TK16S0
TKC1	—	—	—	—	—	—	TKFS1	TKFS0
TK16DL	D7	D6	D5	D4	D3	D2	D1	D0
TK16DH	D15	D14	D13	D12	D11	D10	D9	D8
TKM0C0	M0MXS1	M0MXS0	M0DFEN	—	M0SOFC	M0SOF2	M0SOF1	M0SOF0
TKM0C1	M0TSS	—	M0ROEN	M0KOEN	M0K4EN	M0K3EN	M0K2EN	M0K1EN
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

**Touch Key Function Register List – BS23B04CA**

Register Name	Bit							
	7	6	5	4	3	2	1	0
TKTMR	D7	D6	D5	D4	D3	D2	D1	D0
TKC0	—	TKRCOV	TKST	TKCFOV	TK16OV	TSCS	TK16S1	TK16S0
TKC1	—	—	—	—	—	—	TKFS1	TKFS0
TK16DL	D7	D6	D5	D4	D3	D2	D1	D0
TK16DH	D15	D14	D13	D12	D11	D10	D9	D8
TKM0C0	M0MXS1	M0MXS0	M0DFEN	—	M0SOFC	M0SOF2	M0SOF1	M0SOF0
TKM0C1	M0TSS	—	M0ROEN	M0KOEN	M0K4EN	M0K3EN	M0K2EN	M0K1EN
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
TKM1C0	M1MXS1	M1MXS0	M1DFEN	—	M1SOFC	M1SOF2	M1SOF1	M1SOF0
TKM1C1	M1TSS	—	M1ROEN	M1KOEN	M1K4EN	M1K3EN	M1K2EN	M1K1EN
TKM116DL	D7	D6	D5	D4	D3	D2	D1	D0
TKM116DH	D15	D14	D13	D12	D11	D10	D9	D8
TKM1ROL	D7	D6	D5	D4	D3	D2	D1	D0
TKM1ROH	—	—	—	—	—	—	D9	D8

**Touch Key Function Register List – BS23B08CA**

• **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  
Time slot counter overflow time=(256-TKTMR[7:0])×32  $t_{TSC}$ , where  $t_{TSC}$  is the time slot counter clock period.

• **TKC0 Register**

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

Bit 7 Unimplemented, read as “0”

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

0: No overflow occurs

1: Overflow occurs

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

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

- Bit 5      **TKST**: Touch key detection Start control  
0: Stopped or no operation  
0→1: Start detection  
In all modules the touch key module 16-bit C/F counter, touch key function 16-bit counter and 5-bit time slot unit period counter will automatically be cleared when this bit is cleared to zero. However, the 8-bit programmable time slot counter will not be cleared. When this bit is changed from low to high, the touch key module 16-bit C/F counter, touch key function 16-bit counter, 5-bit time slot unit period counter and 8-bit time slot counter will be switched on together with the key and reference oscillators to drive the corresponding counters.
- Bit 4      **TKCFOV**: Touch key module 16-bit C/F counter overflow flag  
0: No overflow occurs  
1: Overflow occurs  
This bit is set high by the touch key module 16-bit C/F counter overflow and must be cleared to 0 by application programs.
- Bit 3      **TK16OV**: Touch key function 16-bit counter overflow flag  
0: No overflow occurs  
1: Overflow occurs  
This bit is set high by the touch key function 16-bit counter overflow and must be cleared to 0 by application programs.
- Bit 2      **TSCS**: Touch key time slot counter select  
0: Each touch key module uses its own time slot counter  
1: All touch key modules use Module 0 time slot counter  
Note: For the BS23A02CA and BS23B04CA devices, this bit is unimplemented, read as “0”.
- Bit 1~0    **TK16S1~TK16S0**: Touch key function 16-bit counter clock source select  
00:  $f_{SYS}$   
01:  $f_{SYS}/2$   
10:  $f_{SYS}/4$   
11:  $f_{SYS}/8$

• **TKC1 Register**

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

Bit 7~2      Unimplemented, read as “0”

- Bit 1~0      **TKFS1~TKFS0**: Touch Key oscillator and Reference oscillator frequency select  
00: 1MHz  
01: 3MHz  
10: 7MHz  
11: 11MHz

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

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

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

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

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

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

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

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

This register pair is used to store the touch key module n reference oscillator capacitor value.

The reference oscillator internal capacitor value=(TKMnRO[9:0]×50pF)/1024

• **TKMnC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	MnMXS1	MnMXS0	MnDFEN	—	MnSOFc	MnSOF2	MnSOF1	MnSOF0
R/W	R/W	R/W	R/W	—	R/W	R/W	R/W	R/W
POR	0	0	0	—	0	0	0	0

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

Bit	Touch Key Module Number		
MnMXS[1:0]	M0	M0	M1
00	KEY1	KEY1	KEY5
01	KEY2	KEY2	KEY6
10	Reserved	KEY3	KEY7
11	Reserved	KEY4	KEY8
BS23A02CA	√	—	—
BS23B04CA	—	√	—
BS23B08CA	—	√	√

Bit 5 **MnDFEN**: Touch key module n multi-frequency control

0: Disable  
1: Enable

This bit is used to control the touch key oscillator frequency doubling function. When this bit is set to 1, the key oscillator frequency will be doubled.

Bit 4 Unimplemented, read as “0”

Bit 3 **MnSOFc**: Touch key module n C-to-F oscillator frequency hopping function control select

0: Controlled by the MnSOF2~MnSOF0  
1: Controlled by hardware circuit

This bit is used to select the touch key oscillator frequency hopping function control method. When this bit is set to 1, the key oscillator frequency hopping function is controlled by the hardware circuit regardless of the MnSOF2~MnSOF0 bits value.



Bit 2~0 **MnSOF2~MnSOF0**: Touch key module n Reference and Key oscillators hopping frequency select (MnSOFC=0)

000: 1.020MHz  
 001: 1.040MHz  
 010: 1.059MHz  
 011: 1.074MHz  
 100: 1.085MHz  
 101: 1.099MHz  
 110: 1.111MHz  
 111: 1.125MHz

These bits are used to select the touch key oscillator frequency for the hopping function. Note that these bits are only available when the MnSOFC bit is cleared to 0.

The 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.

• **TKMnC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	MnTSS	—	MnROEN	MnKOEN	MnK4EN	MnK3EN	MnK2EN	MnK1EN
R/W	R/W	—	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	—	0	0	0	0	0	0

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

Bit 6 Unimplemented, read as “0”

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

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

Bit 3 **MnK4EN**: Touch key module n Key 4 enable control

MnK4EN	Touch Key Module n – Mn	
	M0	M1
0: Disable	I/O or other functions	
1: Enable	KEY4	KEY8
BS23A02CA	—	—
BS23B04CA	√	—
BS23B08CA	√	√

Bit 2 **MnK3EN**: Touch key module n Key 3 enable control

MnK3EN	Touch Key Module n – Mn	
	M0	M1
0: Disable	I/O or other functions	
1: Enable	KEY3	KEY7
BS23A02CA	—	—
BS23B04CA	√	—
BS23B08CA	√	√

Bit 1 **MnK2EN**: Touch key module n Key 2 enable control

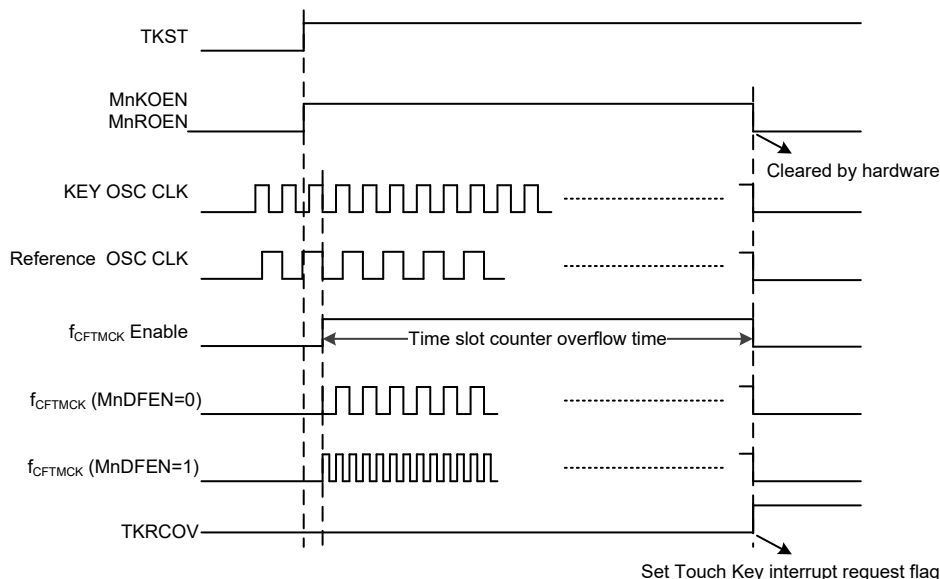
MnK2EN	Touch Key Module n – Mn	
	M0	M1
0: Disable	I/O or other functions	
1: Enable	KEY2	KEY6
BS23A02CA	√	—
BS23B04CA	√	—
BS23B08CA	√	√

Bit 0 **MnK1EN**: Touch key module n Key 1 enable control

MnK1EN	Touch Key Module n – Mn	
	M0	M1
0: Disable	I/O or other functions	
1: Enable	KEY1	KEY5
BS23A02CA	√	—
BS23B04CA	√	—
BS23B08CA	√	√

## Touch Key Operation

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



**Touch Key Scan Mode Timing Diagram**

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

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

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

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

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

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

### **Touch Key Interrupt**

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

### **Programming Considerations**

After the relevant registers are setup, the touch key detection process is initiated by changing the TKST bit from low to high. This will enable and synchronise all relevant oscillators. The TKRCOV flag which is the time slot counter flag will go high when the counter overflows. When this happens an interrupt signal will be generated. As the TKRCOV flag will not be automatically cleared, it has to be cleared by the application program.

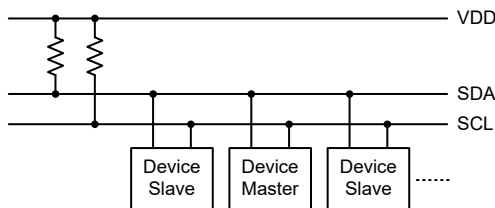
The TKCFOV flag which is the 16-bit C/F counter overflow flag will go high when any of the Touch Key Module 16-bit C/F counter overflows. As this flag will not be automatically cleared, it has to be cleared by the application program.

The TK16OV flag which is the 16-bit counter overflow flag will go high when the 16-bit counter overflows. As this flag will not be automatically cleared, it has to be cleared by the application program.

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

## I<sup>2</sup>C Interface – BS23B04CA and BS23B08CA

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

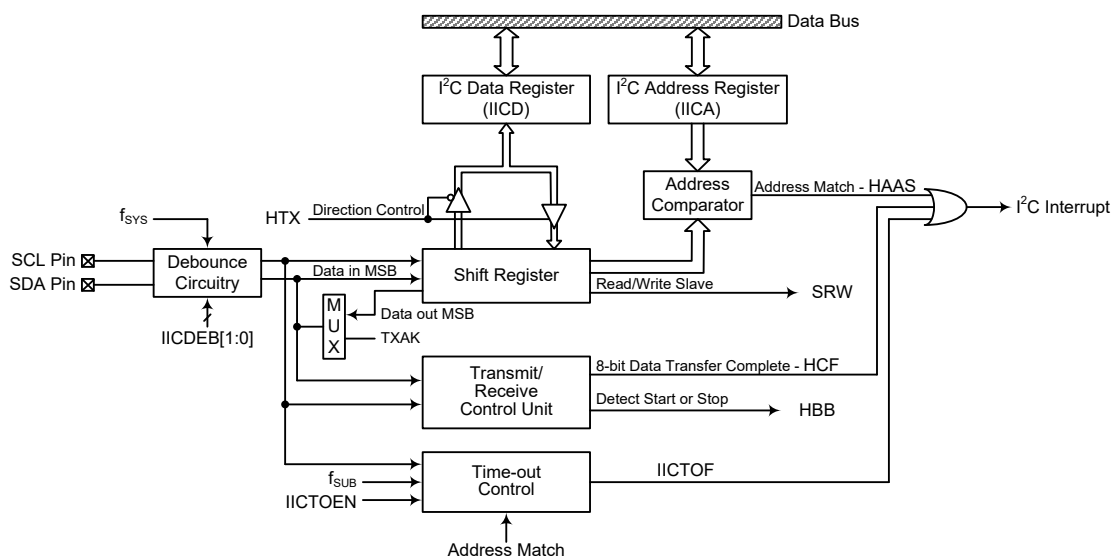


**I<sup>2</sup>C Master Slave Bus Connection**

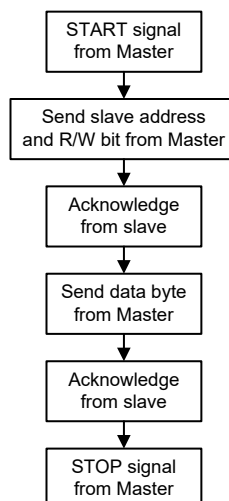
### I<sup>2</sup>C Interface Operation

The I<sup>2</sup>C 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 I<sup>2</sup>C bus is identified by a unique address which will be transmitted and received on the I<sup>2</sup>C bus.

When two devices communicate with each other on the bidirectional I<sup>2</sup>C 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 this device, which only operate in slave mode, there are two methods of transferring data on the I<sup>2</sup>C 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 the I<sup>2</sup>C device is activated and the related internal pull-high function could be controlled by its corresponding pull-high control register. It is suggested that the device should not enter the IDLE/SLEEP mode during the I<sup>2</sup>C communication.



**I<sup>2</sup>C Block Diagram**



### I<sup>2</sup>C Interface Operation

The IICDEB1 and IICDEB0 bits determine the debounce time of the I<sup>2</sup>C 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 I<sup>2</sup>C data transfer speed, there exists a relationship between the system clock,  $f_{SYS}$ , and the I<sup>2</sup>C debounce time. For either the I<sup>2</sup>C 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.

I <sup>2</sup> C Debounce Time Selection	I <sup>2</sup> C Standard Mode (100kHz)	I <sup>2</sup> C Fast Mode (400kHz)
No Debounce	$f_{SYS} > 2\text{MHz}$	$f_{SYS} > 4\text{MHz}$
2 system clock debounce	$f_{SYS} > 4\text{MHz}$	$f_{SYS} > 8\text{MHz}$
4 system clock debounce	$f_{SYS} > 4\text{MHz}$	$f_{SYS} > 8\text{MHz}$

### I<sup>2</sup>C Minimum $f_{SYS}$ Frequency Requirements

### I<sup>2</sup>C Registers

There are three control registers associated with the I<sup>2</sup>C 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

### I<sup>2</sup>C Register List

## I<sup>2</sup>C Data Register

The IICD register is used to store the data being transmitted and received. Before the device writes data to the I<sup>2</sup>C bus, the actual data to be transmitted must be placed in the IICD register. After the data is received from the I<sup>2</sup>C bus, the device can read it from the IICD register. Any transmission or reception of data from the I<sup>2</sup>C 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**: I<sup>2</sup>C data register bit 7 ~ bit 0

## I<sup>2</sup>C 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 I<sup>2</sup>C 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**: I<sup>2</sup>C slave address  
IICA6~IICA0 is the I<sup>2</sup>C slave address bit 6 ~ bit 0.

Bit 0      Unimplemented, read as “0”

## I<sup>2</sup>C Control Registers

There are three control registers for the I<sup>2</sup>C interface, IICC0, IICC1 and IICTOC. The IICC0 register is used to control the enable/disable function and select the debounce time. The IICC1 register contains the relevant flags which are used to indicate the I<sup>2</sup>C communication status. Another register, IICTOC, is used to control the I<sup>2</sup>C 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**: I<sup>2</sup>C debounce time selection

00: No debounce

01: 2 system clock debounce

1x: 4 system clock debounce

Note that the I<sup>2</sup>C 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**: I<sup>2</sup>C enable control  
 0: Disable  
 1: Enable

The bit is the overall on/off control for the I<sup>2</sup>C interface. When the IICEN bit is cleared to zero to disable the I<sup>2</sup>C interface, the SDA and SCL lines will lose their I<sup>2</sup>C function and the I<sup>2</sup>C operating current will be reduced to a minimum value. When the bit is high the I<sup>2</sup>C interface is enabled. If the IICEN bit changes from low to high, the contents of the I<sup>2</sup>C control bits such as HTX and TXAK will remain at the previous settings and should therefore be first initialised by the application program while the relevant I<sup>2</sup>C 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**: I<sup>2</sup>C bus data transfer completion flag  
 0: Data is being transferred  
 1: Completion of an 8-bit data transfer

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

Bit 6 **HAAS**: I<sup>2</sup>C bus address match flag  
 0: Address not match  
 1: Address match

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

Bit 5 **HBB**: I<sup>2</sup>C bus busy flag  
 0: I<sup>2</sup>C Bus is not busy  
 1: I<sup>2</sup>C Bus is busy

The HBB flag is the I<sup>2</sup>C busy flag. This flag will be “1” when the I<sup>2</sup>C bus is busy which will occur when a START signal is detected. The flag will be cleared to “0” when the bus is free which will occur when a STOP signal is detected.

Bit 4 **HTX**: I<sup>2</sup>C slave device is transmitter or receiver selection  
 0: Slave device is the receiver  
 1: Slave device is the transmitter

Bit 3 **TXAK**: I<sup>2</sup>C bus transmit acknowledge flag  
 0: Slave send acknowledge flag  
 1: Slave do not send acknowledge flag

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

Bit 2 **SRW**: I<sup>2</sup>C 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 I<sup>2</sup>C Slave Read/Write flag. This flag determines whether the master device wishes to transmit or receive data from the I<sup>2</sup>C 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.

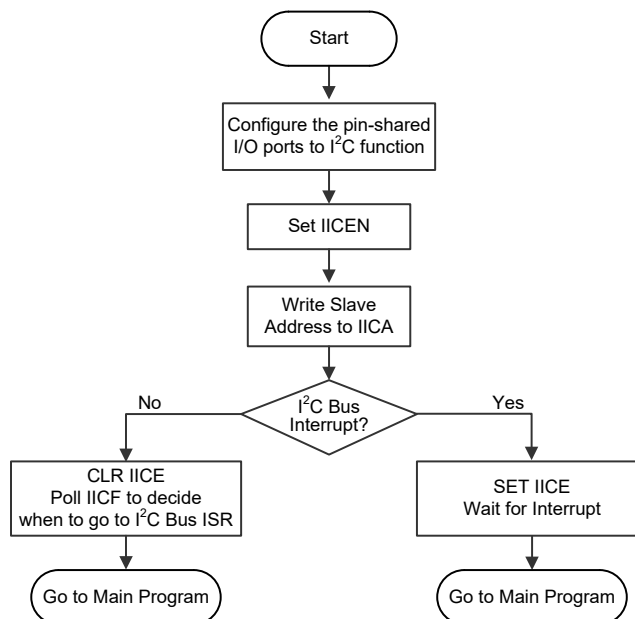
Bit 1	<p><b>IAMWU:</b> I<sup>2</sup>C address match wake-up control</p> <p>0: Disable 1: Enable</p> <p>This bit should be set to 1 to enable the I<sup>2</sup>C address match wake-up from the SLEEP or IDLE Mode. If the IAMWU bit has been set before entering either the SLEEP or IDLE mode to enable the I<sup>2</sup>C address match wake-up, then this bit must be cleared by the application program after wake-up to ensure correction device operation.</p>
Bit 0	<p><b>RXAK:</b> I<sup>2</sup>C bus receive acknowledge flag</p> <p>0: Slave receive acknowledge flag 1: Slave does not receive acknowledge flag</p> <p>The RXAK flag is the receiver acknowledge flag. When the RXAK flag is “0”, it means that a acknowledge signal has been received at the 9th clock, after 8 bits of data have been transmitted. When the slave device in the transmit mode, the slave device checks the RXAK flag to determine if the master receiver wishes to receive the next byte. The slave transmitter will therefore continue sending out data until the RXAK flag is “1”. When this occurs, the slave transmitter will release the SDA line to allow the master to send a STOP signal to release the I<sup>2</sup>C Bus.</p>

### I<sup>2</sup>C Bus Communication

Communication on the I<sup>2</sup>C 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 I<sup>2</sup>C 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 I<sup>2</sup>C 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 I<sup>2</sup>C 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 I<sup>2</sup>C bus, the microcontroller must initialise the bus, the following are steps to achieve this:

- Step 1  
Configure the corresponding pin-shared function as the I<sup>2</sup>C functional pins and set the IICEN bit in the IICC0 register to “1” to enable the I<sup>2</sup>C bus.
- Step 2  
Write the slave address of the device to the I<sup>2</sup>C bus address register IICA.
- Step 3  
Set the IICE interrupt enable bit of the interrupt control register to enable the I<sup>2</sup>C 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.

### **I<sup>2</sup>C Bus Read/Write Signal**

The SRW bit in the IICC1 register defines whether the master device wishes to read data from the I<sup>2</sup>C bus or write data to the I<sup>2</sup>C 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 I<sup>2</sup>C bus, therefore the slave device must be set to send data to the I<sup>2</sup>C bus as a transmitter. If the SRW flag is “0” then this indicates that the master wishes to send data to the I<sup>2</sup>C bus, therefore the slave device must be set to read data from the I<sup>2</sup>C bus as a receiver.

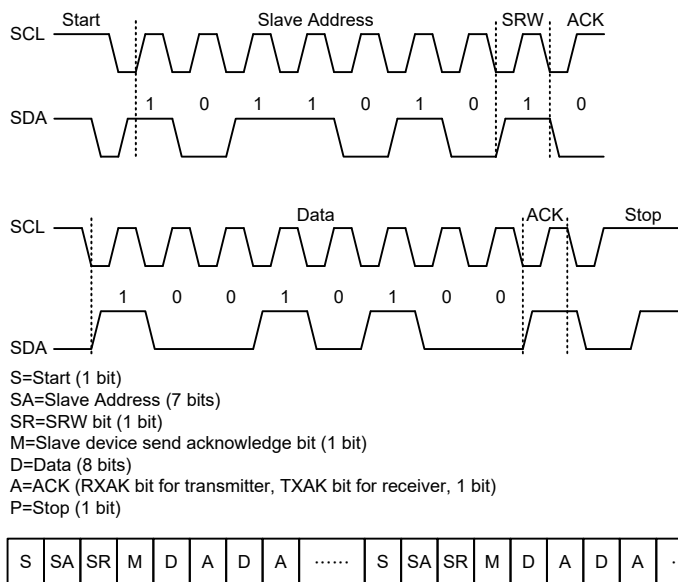
### **I<sup>2</sup>C Bus Slave Address Acknowledge Signal**

After the master has transmitted a calling address, any slave device on the I<sup>2</sup>C bus, whose own internal address matches the calling address, must generate an acknowledge signal. The acknowledge signal will inform the master that a slave device has accepted its calling address. If no acknowledge signal is received by the master then a STOP signal must be transmitted by the master to end the communication. When the HAAS flag is high, the addresses have matched and the slave device must check the SRW flag to determine if it is to be a transmitter or a receiver. If the SRW flag is high, the slave device should be set 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 set as a receiver and the HTX bit in the IICC1 register should be set to “0”.

### **I<sup>2</sup>C Bus Data and Acknowledge Signal**

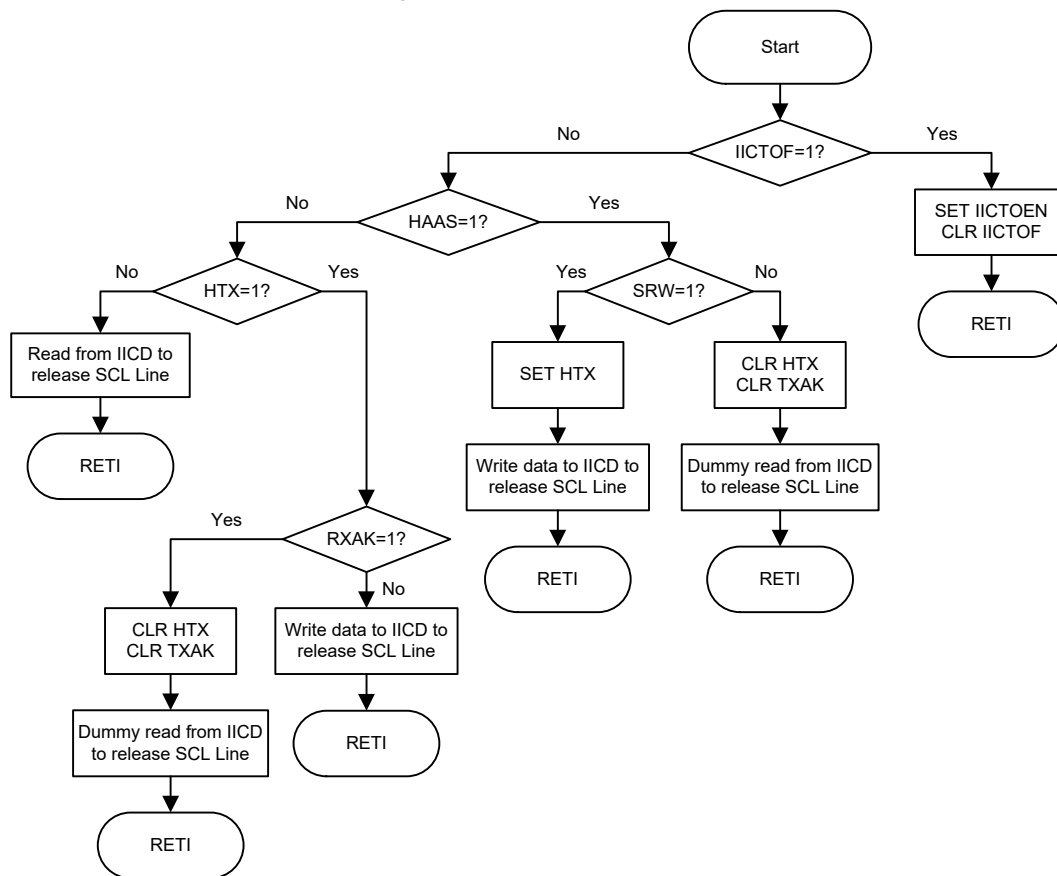
The transmitted data is 8-bit wide and is transmitted after the slave device has acknowledged receipt of its slave address. The order of serial bit transmission is the MSB first and the LSB last. After receipt of 8 bits of data, the receiver must transmit an acknowledge signal, level “0”, before it can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



**I<sup>2</sup>C Communication Timing Diagram**

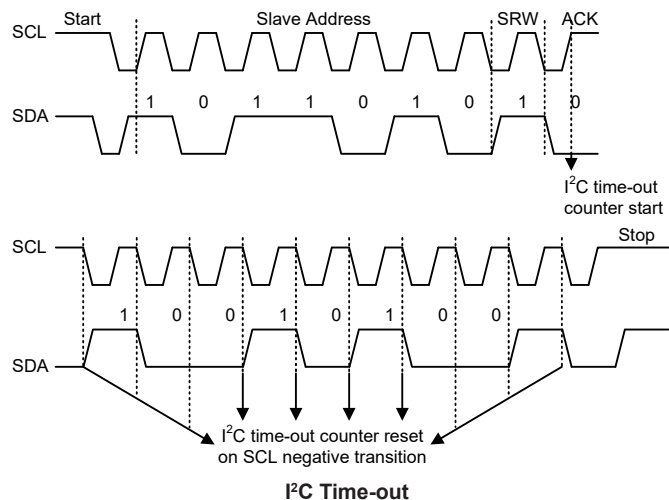
Note: When a slave address is matched, the device must be placed in either the transmit mode and then write data to the IICD register, or in the receive mode where it must implement a dummy read from the IICD register to release the SCL line.



**I<sup>2</sup>C Bus ISR Flowchart**

## I<sup>2</sup>C Time-out Control

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



When an I<sup>2</sup>C 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 I<sup>2</sup>C interrupt vector. When an I<sup>2</sup>C time-out occurs, the I<sup>2</sup>C internal circuitry will be reset and the registers will be reset into the following condition:

Registers	After I <sup>2</sup> C Time-out
IICD, IICA, IICC0	No change
IICC1	Reset to POR condition

I<sup>2</sup>C 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 IICTOS5~IICTOS0 bits in the IICTOC register. The time-out time is given by the formula:  $[(1-64) \times 32] / f_{SUB}$ . 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**: I<sup>2</sup>C Time-out control  
0: Disable  
1: Enable

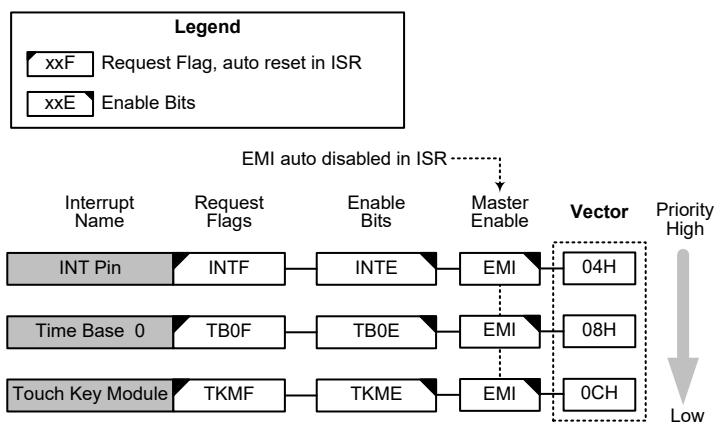
Bit 6 **IICTOF**: I<sup>2</sup>C Time-out flag  
0: No time-out occurred  
1: Time-out occurred

This bit is set high when time-out occurs and can only be cleared to zero by application program.

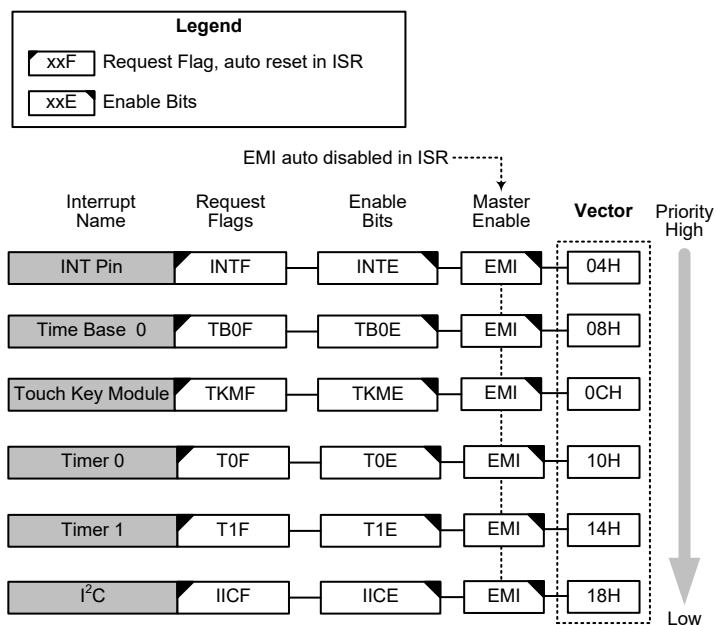
Bit 5~0     **IICTOS5~IICTOS0**: I<sup>2</sup>C Time-out period selection  
I<sup>2</sup>C time-out clock source is  $f_{SUB}/32$ .  
I<sup>2</sup>C time-out time is equal to  $(IICTOS[5:0]+1) \times (32/f_{SUB})$ .

## Interrupts

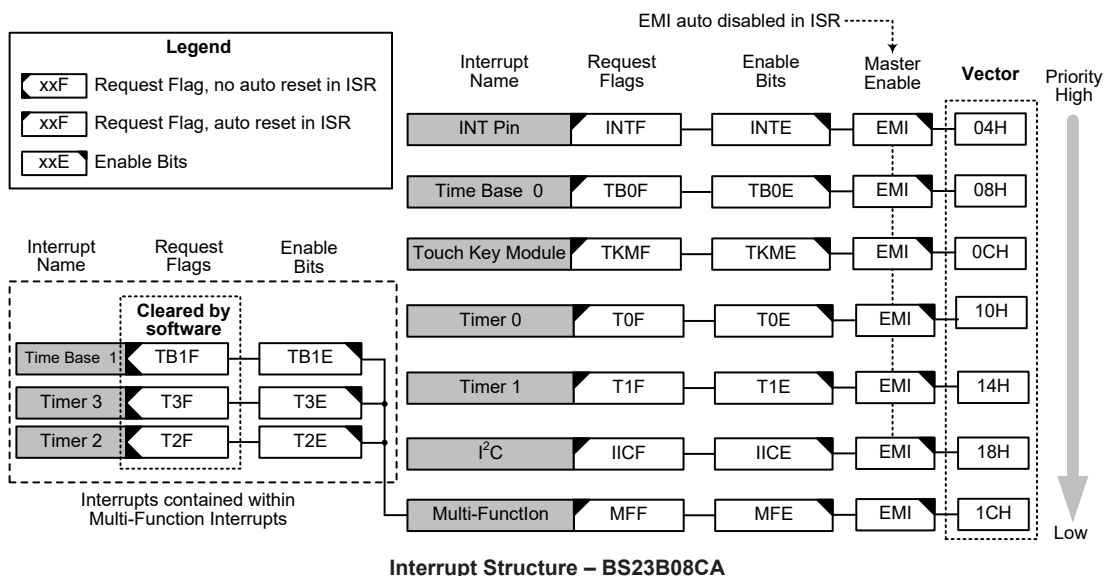
Interrupts are an important part of any microcontroller system. When an external event or an internal function such as a Timer requires microcontroller attention, its corresponding interrupt will enforce a temporary suspension of the main program allowing the microcontroller to direct attention to its needs. The devices contain an external interrupt and several internal interrupt functions. The external interrupt is generated by the action of the external INT pin, while the internal interrupts are generated by internal functions including the Timer/Event Counter and Time Base, etc.



**Interrupt Structure – BS23A02CA**



**Interrupt Structure – BS23B04CA**



## 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 register which sets the primary interrupts, the second is the INTEG register to set the external interrupt trigger edge type. Finally there is the MFI register which setups the Multi-function interrupt.

The interrupt 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	TBnE	TBnF	n=0 for BS23A02CA/BS23B04CA n=0~1 for BS23B08CA
Touch Key Module	TKME	TKMF	—
Timer/Event Counter	TnE	TnF	n=0~1 for BS23B04CA n=0~3 for BS23B08CA
I <sup>2</sup> C	IICE	IICF	—
Multi-function	MFE	MFF	For BS23B08CA only

### Interrupt Register Bit Naming Conventions

Register Name	Bit							
	7	6	5	4	3	2	1	0
INTEG	—	—	—	—	—	—	INTS1	INTS0
INTC0	—	TKMF	TB0F	INTF	TKME	TB0E	INTE	EMI
INTC1 (BS23B04CA)	—	IICF	T1F	T0F	—	IICE	T1E	T0E
INTC1 (BS23B08CA)	MFF	IICF	T1F	T0F	MFE	IICE	T1E	T0E
MFI (BS23B08CA)	—	TB1F	T3F	T2F	—	TB1E	T3E	T2E

### 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**

Bit	7	6	5	4	3	2	1	0
Name	—	T0F	TB0F	INTF	T0E	TB0E	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 **T0F**: Timer/Event Counter 0 interrupt request flag

0: No request

1: Interrupt request

Bit 5 **TB0F**: Time Base 0 interrupt request flag

0: No request

1: Interrupt request

Bit 4 **INTF**: INT interrupt request flag

0: No request

1: Interrupt request

Bit 3 **T0E**: Timer/Event Counter 0 interrupt control

0: Disable

1: Enable

Bit 2 **TB0E**: Time Base 0 interrupt control

0: Disable

1: Enable

Bit 1 **INTE**: INT interrupt control

0: Disable

1: Enable

Bit 0 **EMI**: Global interrupt control

0: Disable

1: Enable

• **INTC1 Register – BS23B04CA**

Bit	7	6	5	4	3	2	1	0
Name	—	IICF	T1F	T0F	—	IICE	T1E	T0E
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 **IICF**: I<sup>2</sup>C interrupt request flag  
0: No request  
1: Interrupt request
- Bit 5 **T1F**: Timer/Event Counter 1 interrupt request flag  
0: No request  
1: Interrupt request
- Bit 4 **T0F**: Timer/Event Counter 0 interrupt request flag  
0: No request  
1: Interrupt request
- Bit 3 Unimplemented, read as “0”
- Bit 2 **IICE**: I<sup>2</sup>C interrupt control  
0: Disable  
1: Enable
- Bit 1 **T1E**: Timer/Event Counter 1 interrupt control  
0: Disable  
1: Enable
- Bit 0 **T0E**: Timer/Event Counter 0 interrupt control  
0: Disable  
1: Enable

• **INTC1 Register – BS23B08CA**

Bit	7	6	5	4	3	2	1	0
Name	MFF	IICF	T1F	T0F	MFE	IICE	T1E	T0E
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 **MFF**: Multi-function interrupt request flag  
0: No request  
1: Interrupt request
- Bit 6 **IICF**: I<sup>2</sup>C interrupt request flag  
0: No request  
1: Interrupt request
- Bit 5 **T1F**: Timer/Event Counter 1 interrupt request flag  
0: No request  
1: Interrupt request
- Bit 4 **T0F**: Timer/Event Counter 0 interrupt request flag  
0: No request  
1: Interrupt request
- Bit 3 **MFE**: Multi-function interrupt control  
0: Disable  
1: Enable
- Bit 2 **IICE**: I<sup>2</sup>C interrupt control  
0: Disable  
1: Enable



- Bit 1      **T1E**: Timer/Event Counter 1 interrupt control  
             0: Disable  
             1: Enable
- Bit 0      **T0E**: Timer/Event Counter 0 interrupt control  
             0: Disable  
             1: Enable

• **MFI Register – BS23B08CA**

Bit	7	6	5	4	3	2	1	0
Name	—	TB1F	T3F	T2F	—	TB1E	T3E	T2E
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      **TB1F**: Time Base 1 interrupt request flag  
             0: No request  
             1: Interrupt request
- Bit 5      **T3F**: Timer/Event Counter 3 interrupt request flag  
             0: No request  
             1: Interrupt request
- Bit 4      **T2F**: Timer/Event Counter 2 interrupt request flag  
             0: No request  
             1: Interrupt request
- Bit 3      Unimplemented, read as “0”
- Bit 2      **TB1E**: Time Base 1 interrupt control  
             0: Disable  
             1: Enable
- Bit 1      **T3E**: Timer/Event Counter 3 interrupt control  
             0: Disable  
             1: Enable
- Bit 0      **T2E**: Timer/Event Counter 2 interrupt control  
             0: Disable  
             1: Enable

## Interrupt Operation

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

When an interrupt is generated, the Program Counter, which stores the address of the next instruction to be executed, will be transferred onto the stack. The Program Counter will then be loaded with a new address which will be the value of the corresponding interrupt vector. The microcontroller will then fetch its next instruction from this interrupt vector. The instruction at this vector will usually be a “JMP” which will jump to another section of program which is known as the interrupt service routine. Here is located the code to control the appropriate interrupt. The interrupt service routine must be terminated with a “RETI”, which retrieves the original Program Counter address from the stack and allows the microcontroller to continue with normal execution at the point where the interrupt occurred.

The various interrupt enable bits, together with their associated request flags, are shown in the accompanying diagram 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.

### **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 the 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 set 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 selection on the external interrupt pin will remain valid even if the pin is used as an external interrupt input.

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

### **Timer/Event Counter Interrupt**

There are up to four Timer/Event Counter interrupts, the Timer/Event Counter interrupt 0/1 is an independent interrupt and the Timer/Event Counter interrupt 2/3 is contained within the Multi-function Interrupt.

An actual Timer/Event Counter interrupt 0/1 will take place when the Timer/Event Counter request flag, TnF, 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, TnE, 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, TnF, will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts.

An actual Timer/Event Counter interrupt 2/3 will take place when the Timer/Event Counter request flag, TnF, 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, the related Multi-function interrupt enable bit and the Timer/Event Counter Interrupt enable bit, TnE, must first be set. When the interrupt is enabled, the stack is not full and the Timer/Event Counter overflows, a subroutine call to the respective Multi-function interrupt vector, will take place. When the interrupt is serviced, the related Multi-function interrupt request flag will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts. However the TnF flag will not be automatically cleared to zero, it has to be cleared by the application program.

### **Touch Key Module Interrupt**

For a Touch Key interrupt to occur, the global interrupt enable bit, EMI, and the Touch Key interrupt enable bit, TKME, must be first set. An actual Touch Key interrupt will take place when the Touch Key interrupt request flag, TMKF, is set, a situation that will occur when the time slot counter overflows. When the interrupt is enabled, the stack is not full and the Touch Key time slot counter overflow occurs, a subroutine call to the relevant interrupt vector, will take place. When the interrupt is serviced, the Touch Key interrupt request flag will be automatically reset and the EMI bit will also be automatically cleared to disable other interrupts.

### **I<sup>2</sup>C Interrupt**

An I<sup>2</sup>C interrupt request will take place when the I<sup>2</sup>C Interrupt request flag, IICF, is set, which occurs when a byte of data has been received or transmitted by the I<sup>2</sup>C interface, or an I<sup>2</sup>C slave address match occurs, or an I<sup>2</sup>C bus time-out occurs. To allow the program to branch to its interrupt vector address, the global interrupt enable bit, EMI, and the I<sup>2</sup>C Interrupt enable bit, IICE, 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 I<sup>2</sup>C Interrupt vector, will take place. When the interrupt is serviced, the I<sup>2</sup>C Interrupt flag, IICF, will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts.

### **Multi-function Interrupts**

Within the BS23B08CA device there is a Multi-function interrupt. Unlike the other independent interrupts, the interrupt has no independent source, but rather is formed from other existing interrupt sources, namely the Timer/Event Counter 2 interrupt, Timer/Event Counter 3 interrupt and Time base 1 interrupt.

A Multi-function interrupt request will take place when the Multi-function interrupt request flag MFF is set. The Multi-function interrupt flag will be set when any of its included functions generate an interrupt request flag. When the Multi-function interrupt is enabled and the stack is not full, and one of the interrupts contained within the Multi-function interrupt occurs, a subroutine call to the Multi-function interrupt vector will take place. When the interrupt is serviced, the related Multi-Function request flag will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts.

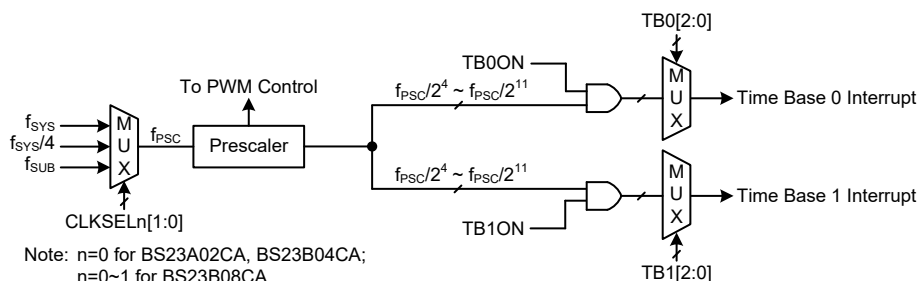
However, it must be noted that, although the Multi-function Interrupt request flag will be automatically reset when the interrupt is serviced, the request flag from the original source of the Multi-function interrupt will not be automatically reset and must be manually reset by the application program.

## Time Base Interrupt

There are up to two Time Base interrupts, the Time Base interrupt 0 is an independent interrupt and the Time Base interrupt 1 is contained within the Multi-function 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 the timer function. When this happens its interrupt request flag TBnF will be set. To allow the program to branch to its interrupt vector address, the global interrupt enable bit, EMI and Time Base enable bit, TBnE, must first be set. When the interrupt is enabled, the stack is not full and the Time Base overflows, a subroutine call to its interrupt vector location will take place. For time base 0, when the interrupt is serviced, the interrupt request flag, TB0F, will be automatically reset and the EMI bit will be cleared to disable other interrupts. For time base 1, when the Time Base interrupt is serviced, the EMI bit will be automatically cleared to disable other interrupts, however only the related MFF flag will be automatically cleared. As the Time Base interrupt request flag, TB1F, will not be automatically cleared, they have to be cleared by the application program.

The purpose of the Time Base Interrupt is to provide an interrupt signal at fixed time periods. Its clock source,  $f_{PSC}$ , originates from the internal clock source  $f_{SYS}$ ,  $f_{SYS}/4$  or  $f_{SUB}$  and then passes through a divider, the division ratio of which is selected by programming the appropriate bits in the TBnC register to obtain longer interrupt periods whose value ranges. The clock source which in turn controls the Time Base interrupt period is selected using the CLKSEL0[1:0] and CLKSEL1[1:0] bits in the TBnC register. It should be noted that as the Time Base clock source is the same as the Timer/Event Counter clock source, care should be taken when programming.



**Time Base Interrupt**

### • TB0C Register

Bit	7	6	5	4	3	2	1	0
Name	TB0ON	—	CLKSEL01	CLKSEL00	—	TB02	TB01	TB00
R/W	R/W	—	R/W	R/W	—	R/W	R/W	R/W
POR	0	—	0	0	—	0	0	0

- Bit 7      **TB0ON:** Time Base 0 control  
0: Disable  
1: Enable
- Bit 6      Unimplemented, read as “0”
- Bit 5~4    **CLKSEL01~CLKSEL00:** Prescaler clock source  $f_{PSC}$  selection  
00:  $f_{SYS}$   
01:  $f_{SYS}/4$   
1x:  $f_{SUB}$
- Bit 3      Unimplemented, read as “0”

Bit 2~0      **TB02~TB00**: Time Base 0 Time-out period selection  
               000:  $2^4/f_{PSC}$   
               001:  $2^5/f_{PSC}$   
               010:  $2^6/f_{PSC}$   
               011:  $2^7/f_{PSC}$   
               100:  $2^8/f_{PSC}$   
               101:  $2^9/f_{PSC}$   
               110:  $2^{10}/f_{PSC}$   
               111:  $2^{11}/f_{PSC}$

• **TB1C Register – BS23B08CA**

Bit	7	6	5	4	3	2	1	0
Name	TB1ON	—	CLKSEL11	CLKSEL10	—	TB12	TB11	TB10
R/W	R/W	—	R/W	R/W	—	R/W	R/W	R/W
POR	0	—	0	0	—	0	0	0

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

Bit 6      Unimplemented, read as “0”

Bit 5~4    **CLKSEL11~CLKSEL10**: Prescaler clock source  $f_{PSC}$  selection  
               00:  $f_{SYS}$   
               01:  $f_{SYS}/4$   
               1x:  $f_{SUB}$

Bit 3      Unimplemented, read as “0”

Bit 2~0    **TB12~TB10**: Time Base 1 Time-out period selection  
               000:  $2^4/f_{PSC}$   
               001:  $2^5/f_{PSC}$   
               010:  $2^6/f_{PSC}$   
               011:  $2^7/f_{PSC}$   
               100:  $2^8/f_{PSC}$   
               101:  $2^9/f_{PSC}$   
               110:  $2^{10}/f_{PSC}$   
               111:  $2^{11}/f_{PSC}$

## Interrupt Wake-up Function

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

## Programming Considerations

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

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

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

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

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

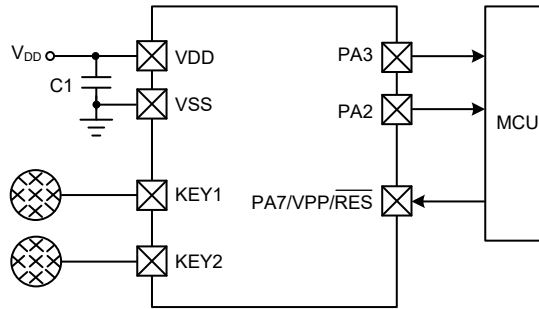
## Configuration Options

Configuration options refer to certain options within the MCU that are programmed into the device during the programming process. During the development process, these options are selected using the HT-IDE software development tools. All options must be defined for proper system function, the details of which are shown in the table.

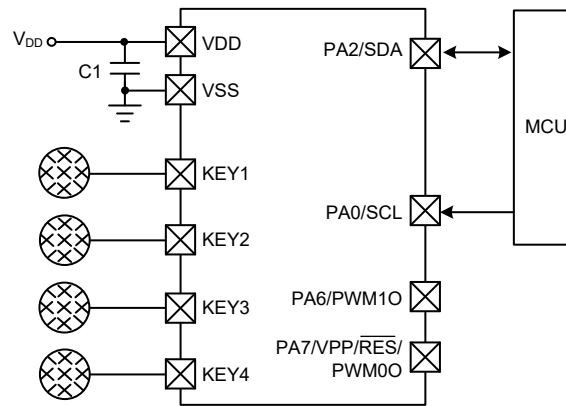
No.	Options
<b>I/O Pin-Shared Options</b>	
1	RES pin reset function selection: 1: I/O port 2: RES pin
<b>LVR Options</b>	
2	LVR function selection: 1: Enable 2: Disable
3	LVR voltage selection: 1: 1.9V 2: 2.1V 3: 3.15V 4: 4.2V

## Application Circuits

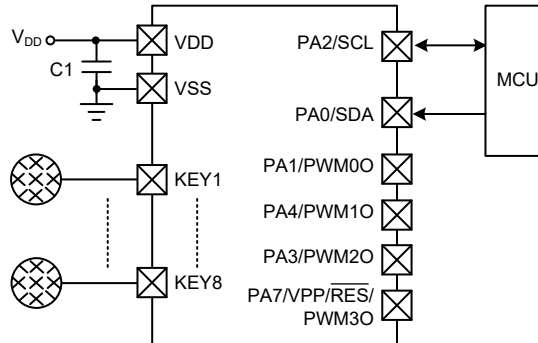
**BS23A02CA**



**BS23B04CA**



**BS23B08CA**



## 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 0.5 $\mu$ s and branch or call instructions would be implemented within 1 $\mu$ s. 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
<b>Arithmetic</b>			
ADD A,[m]	Add Data Memory to ACC	1	Z, C, AC, OV
ADDM A,[m]	Add ACC to Data Memory	1 <sup>Note</sup>	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	1 <sup>Note</sup>	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	1 <sup>Note</sup>	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	1 <sup>Note</sup>	Z, C, AC, OV
DAA [m]	Decimal adjust ACC for Addition with result in Data Memory	1 <sup>Note</sup>	C
<b>Logic Operation</b>			
AND A,[m]	Logical AND Data Memory to ACC	1	Z
OR A,[m]	Logical OR Data Memory to ACC	1	Z
XOR A,[m]	Logical XOR Data Memory to ACC	1	Z
ANDM A,[m]	Logical AND ACC to Data Memory	1 <sup>Note</sup>	Z
ORM A,[m]	Logical OR ACC to Data Memory	1 <sup>Note</sup>	Z
XORM A,[m]	Logical XOR ACC to Data Memory	1 <sup>Note</sup>	Z
AND A,x	Logical AND immediate Data to ACC	1	Z
OR A,x	Logical OR immediate Data to ACC	1	Z
XOR A,x	Logical XOR immediate Data to ACC	1	Z
CPL [m]	Complement Data Memory	1 <sup>Note</sup>	Z
CPLA [m]	Complement Data Memory with result in ACC	1	Z
<b>Increment &amp; Decrement</b>			
INCA [m]	Increment Data Memory with result in ACC	1	Z
INC [m]	Increment Data Memory	1 <sup>Note</sup>	Z
DECA [m]	Decrement Data Memory with result in ACC	1	Z
DEC [m]	Decrement Data Memory	1 <sup>Note</sup>	Z
<b>Rotate</b>			
RRA [m]	Rotate Data Memory right with result in ACC	1	None
RR [m]	Rotate Data Memory right	1 <sup>Note</sup>	None
RRCA [m]	Rotate Data Memory right through Carry with result in ACC	1	C
RRC [m]	Rotate Data Memory right through Carry	1 <sup>Note</sup>	C
RLA [m]	Rotate Data Memory left with result in ACC	1	None
RL [m]	Rotate Data Memory left	1 <sup>Note</sup>	None
RLCA [m]	Rotate Data Memory left through Carry with result in ACC	1	C
RLC [m]	Rotate Data Memory left through Carry	1 <sup>Note</sup>	C

Mnemonic	Description	Cycles	Flag Affected
<b>Data Move</b>			
MOV A,[m]	Move Data Memory to ACC	1	None
MOV [m],A	Move ACC to Data Memory	1 <sup>Note</sup>	None
MOV A,x	Move immediate data to ACC	1	None
<b>Bit Operation</b>			
CLR [m].i	Clear bit of Data Memory	1 <sup>Note</sup>	None
SET [m].i	Set bit of Data Memory	1 <sup>Note</sup>	None
<b>Branch Operation</b>			
JMP addr	Jump unconditionally	2	None
SZ [m]	Skip if Data Memory is zero	1 <sup>Note</sup>	None
SZA [m]	Skip if Data Memory is zero with data movement to ACC	1 <sup>Note</sup>	None
SZ [m].i	Skip if bit i of Data Memory is zero	1 <sup>Note</sup>	None
SNZ [m].i	Skip if bit i of Data Memory is not zero	1 <sup>Note</sup>	None
SIZ [m]	Skip if increment Data Memory is zero	1 <sup>Note</sup>	None
SDZ [m]	Skip if decrement Data Memory is zero	1 <sup>Note</sup>	None
SIZA [m]	Skip if increment Data Memory is zero with result in ACC	1 <sup>Note</sup>	None
SDZA [m]	Skip if decrement Data Memory is zero with result in ACC	1 <sup>Note</sup>	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
<b>Table Read Operation</b>			
TABRD [m]	Read table (specific page or current page) to TBLH and Data Memory	2 <sup>Note</sup>	None
TABRDL [m]	Read table (last page) to TBLH and Data Memory	2 <sup>Note</sup>	None
<b>Miscellaneous</b>			
NOP	No operation	1	None
CLR [m]	Clear Data Memory	1 <sup>Note</sup>	None
SET [m]	Set Data Memory	1 <sup>Note</sup>	None
CLR WDT	Clear Watchdog Timer	1	TO, PDF
SWAP [m]	Swap nibbles of Data Memory	1 <sup>Note</sup>	None
SWAPA [m]	Swap nibbles of Data Memory with result in ACC	1	None
HALT	Enter power down mode	1	TO, PDF

Note: 1. For skip instructions, if the result of the comparison involves a skip then two cycles are required, if no skip takes place only one cycle is required.

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

## Instruction Definition

<b>ADC A,[m]</b>	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	$ACC \leftarrow ACC + [m] + C$
Affected flag(s)	OV, Z, AC, C
<b>ADCM A,[m]</b>	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	$[m] \leftarrow ACC + [m] + C$
Affected flag(s)	OV, Z, AC, C
<b>ADD A,[m]</b>	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	$ACC \leftarrow ACC + [m]$
Affected flag(s)	OV, Z, AC, C
<b>ADD A,x</b>	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	$ACC \leftarrow ACC + x$
Affected flag(s)	OV, Z, AC, C
<b>ADDM A,[m]</b>	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	$[m] \leftarrow ACC + [m]$
Affected flag(s)	OV, Z, AC, C
<b>AND A,[m]</b>	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	$ACC \leftarrow ACC \text{ "AND" } [m]$
Affected flag(s)	Z
<b>AND A,x</b>	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	$ACC \leftarrow ACC \text{ "AND" } x$
Affected flag(s)	Z

<b>ANDM A,[m]</b>	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	$[m] \leftarrow \text{ACC} \text{ "AND" } [m]$
Affected flag(s)	Z
<b>CALL addr</b>	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 $\leftarrow$ Program Counter + 1 Program Counter $\leftarrow$ addr
Affected flag(s)	None
<b>CLR [m]</b>	Clear Data Memory
Description	Each bit of the specified Data Memory is cleared to 0.
Operation	$[m] \leftarrow 00\text{H}$
Affected flag(s)	None
<b>CLR [m].i</b>	Clear bit of Data Memory
Description	Bit i of the specified Data Memory is cleared to 0.
Operation	$[m].i \leftarrow 0$
Affected flag(s)	None
<b>CLR WDT</b>	Clear Watchdog Timer
Description	The TO, PDF flags and the WDT are all cleared.
Operation	WDT cleared $\text{TO} \leftarrow 0$ $\text{PDF} \leftarrow 0$
Affected flag(s)	TO, PDF
<b>CPL [m]</b>	Complement Data Memory
Description	Each bit of the specified Data Memory is logically complemented (1's complement). Bits which previously contained a 1 are changed to 0 and vice versa.
Operation	$[m] \leftarrow \overline{[m]}$
Affected flag(s)	Z
<b>CPLA [m]</b>	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	$\text{ACC} \leftarrow \overline{[m]}$
Affected flag(s)	Z

<b>DAA [m]</b>	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
<b>DEC [m]</b>	Decrement Data Memory
Description	Data in the specified Data Memory is decremented by 1.
Operation	[m] ← [m] – 1
Affected flag(s)	Z
<b>DECA [m]</b>	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
<b>HALT</b>	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
<b>INC [m]</b>	Increment Data Memory
Description	Data in the specified Data Memory is incremented by 1.
Operation	[m] ← [m] + 1
Affected flag(s)	Z
<b>INCA [m]</b>	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

<b>JMP addr</b>	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 $\leftarrow$ addr
Affected flag(s)	None
<b>MOV A,[m]</b>	Move Data Memory to ACC
Description	The contents of the specified Data Memory are copied to the Accumulator.
Operation	ACC $\leftarrow$ [m]
Affected flag(s)	None
<b>MOV A,x</b>	Move immediate data to ACC
Description	The immediate data specified is loaded into the Accumulator.
Operation	ACC $\leftarrow$ x
Affected flag(s)	None
<b>MOV [m],A</b>	Move ACC to Data Memory
Description	The contents of the Accumulator are copied to the specified Data Memory.
Operation	[m] $\leftarrow$ ACC
Affected flag(s)	None
<b>NOP</b>	No operation
Description	No operation is performed. Execution continues with the next instruction.
Operation	No operation
Affected flag(s)	None
<b>OR A,[m]</b>	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 $\leftarrow$ ACC "OR" [m]
Affected flag(s)	Z
<b>OR A,x</b>	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 $\leftarrow$ ACC "OR" x
Affected flag(s)	Z
<b>ORM A,[m]</b>	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] $\leftarrow$ ACC "OR" [m]
Affected flag(s)	Z

<b>RET</b>	Return from subroutine
Description	The Program Counter is restored from the stack. Program execution continues at the restored address.
Operation	Program Counter $\leftarrow$ Stack
Affected flag(s)	None
<b>RET A,x</b>	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 $\leftarrow$ Stack ACC $\leftarrow$ x
Affected flag(s)	None
<b>RETI</b>	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 $\leftarrow$ Stack EMI $\leftarrow$ 1
Affected flag(s)	None
<b>RL [m]</b>	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) $\leftarrow$ [m].i; (i=0~6) [m].0 $\leftarrow$ [m].7
Affected flag(s)	None
<b>RLA [m]</b>	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) $\leftarrow$ [m].i; (i=0~6) ACC.0 $\leftarrow$ [m].7
Affected flag(s)	None
<b>RLC [m]</b>	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) $\leftarrow$ [m].i; (i=0~6) [m].0 $\leftarrow$ C C $\leftarrow$ [m].7
Affected flag(s)	C



<b>RLCA [m]</b>	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	$ACC.(i+1) \leftarrow [m].i; (i=0\sim6)$ $ACC.0 \leftarrow C$ $C \leftarrow [m].7$
Affected flag(s)	C
<b>RR [m]</b>	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	$[m].i \leftarrow [m].(i+1); (i=0\sim6)$ $[m].7 \leftarrow [m].0$
Affected flag(s)	None
<b>RRA [m]</b>	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	$ACC.i \leftarrow [m].(i+1); (i=0\sim6)$ $ACC.7 \leftarrow [m].0$
Affected flag(s)	None
<b>RRC [m]</b>	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	$[m].i \leftarrow [m].(i+1); (i=0\sim6)$ $[m].7 \leftarrow C$ $C \leftarrow [m].0$
Affected flag(s)	C
<b>RRCA [m]</b>	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	$ACC.i \leftarrow [m].(i+1); (i=0\sim6)$ $ACC.7 \leftarrow C$ $C \leftarrow [m].0$
Affected flag(s)	C
<b>SBC A,[m]</b>	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

<b>SBCM A,[m]</b>	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	$[m] \leftarrow ACC - [m] - \bar{C}$
Affected flag(s)	OV, Z, AC, C
<b>SDZ [m]</b>	Skip if decrement Data Memory is 0
Description	The contents of the specified Data Memory are first decremented by 1. If the result is 0 the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.
Operation	$[m] \leftarrow [m] - 1$ Skip if $[m]=0$
Affected flag(s)	None
<b>SDZA [m]</b>	Skip if decrement Data Memory is zero with result in ACC
Description	The contents of the specified Data Memory are first decremented by 1. If the result is 0, the following instruction is skipped. The result is stored in the Accumulator but the specified Data Memory contents remain unchanged. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0, the program proceeds with the following instruction.
Operation	$ACC \leftarrow [m] - 1$ Skip if $ACC=0$
Affected flag(s)	None
<b>SET [m]</b>	Set Data Memory
Description	Each bit of the specified Data Memory is set to 1.
Operation	$[m] \leftarrow FFH$
Affected flag(s)	None
<b>SET [m].i</b>	Set bit of Data Memory
Description	Bit i of the specified Data Memory is set to 1.
Operation	$[m].i \leftarrow 1$
Affected flag(s)	None
<b>SIZ [m]</b>	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	$[m] \leftarrow [m] + 1$ Skip if $[m]=0$
Affected flag(s)	None

<b>SIZA [m]</b>	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	$ACC \leftarrow [m] + 1$ Skip if $ACC=0$
Affected flag(s)	None
<b>SNZ [m].i</b>	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 $[m].i \neq 0$
Affected flag(s)	None
<b>SUB A,[m]</b>	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	$ACC \leftarrow ACC - [m]$
Affected flag(s)	OV, Z, AC, C
<b>SUBM A,[m]</b>	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	$[m] \leftarrow ACC - [m]$
Affected flag(s)	OV, Z, AC, C
<b>SUB A,x</b>	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	$ACC \leftarrow ACC - x$
Affected flag(s)	OV, Z, AC, C
<b>SWAP [m]</b>	Swap nibbles of Data Memory
Description	The low-order and high-order nibbles of the specified Data Memory are interchanged.
Operation	$[m].3 \sim [m].0 \leftrightarrow [m].7 \sim [m].4$
Affected flag(s)	None

<b>SWAPA [m]</b>	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 $\leftarrow$ [m].7~[m].4 ACC.7~ACC.4 $\leftarrow$ [m].3~[m].0
Affected flag(s)	None
<b>SZ [m]</b>	Skip if Data Memory is 0
Description	The contents of the specified Data Memory are read out and then written to the specified Data Memory again. If the contents of the specified Data Memory is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a 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
<b>SZA [m]</b>	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 $\leftarrow$ [m] Skip if [m]=0
Affected flag(s)	None
<b>SZ [m].i</b>	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
<b>TABRD [m]</b>	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] $\leftarrow$ program code (low byte) TBLH $\leftarrow$ program code (high byte)
Affected flag(s)	None

<b>TABRDL [m]</b>	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
<b>XOR A,[m]</b>	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
<b>XORM A,[m]</b>	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
<b>XOR A,x</b>	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

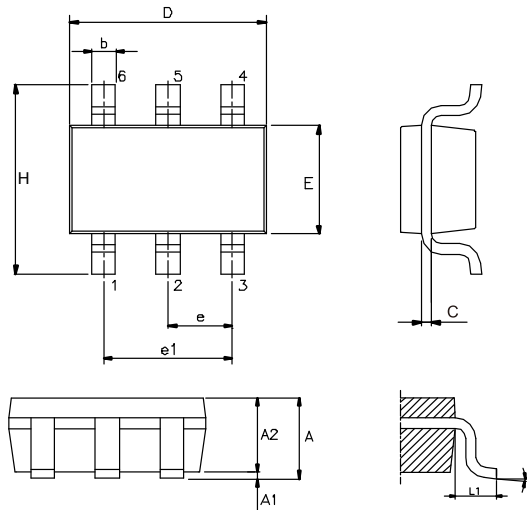
## Package Information

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.

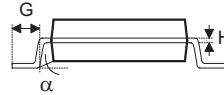
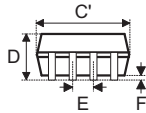
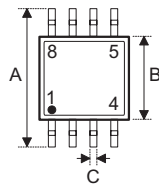
- [Package Information \(include Outline Dimensions, Product Tape and Reel Specifications\)](#)
- [The Operation Instruction of Packing Materials](#)
- [Carton information](#)

**6-pin SOT23 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°

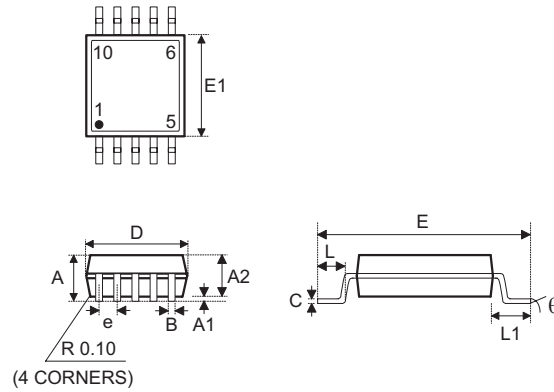
**8-pin SOP (150mil) Outline Dimensions**


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
$\alpha$	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
$\alpha$	0°	—	8°

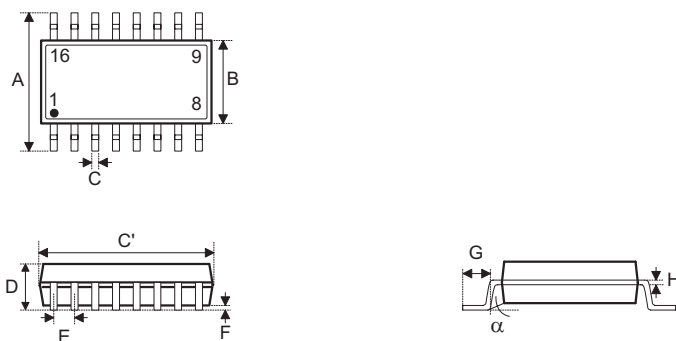


**10-pin MSOP Outline Dimensions**



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
$\alpha$	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
$\alpha$	0°	—	8°

Copyright© 2024 by HOLTEK SEMICONDUCTOR INC. All Rights Reserved.

The information provided in this document has been produced with reasonable care and attention before publication, however, HOLTEK does not guarantee that the information is completely accurate. The information contained in this publication is provided for reference only and may be superseded by updates. HOLTEK disclaims any expressed, implied or statutory warranties, including but not limited to suitability for commercialization, satisfactory quality, specifications, characteristics, functions, fitness for a particular purpose, and non-infringement of any third-party's rights. HOLTEK disclaims all liability arising from the information and its application. In addition, HOLTEK does not recommend the use of HOLTEK's products where there is a risk of personal hazard due to malfunction or other reasons. HOLTEK hereby declares that it does not authorize the use of these products in life-saving, life-sustaining or safety critical components. Any use of HOLTEK's products in life-saving/sustaining or safety applications is entirely at the buyer's risk, and the buyer agrees to defend, indemnify and hold HOLTEK harmless from any damages, claims, suits, or expenses resulting from such use. The information provided in this document, including but not limited to the content, data, examples, materials, graphs, and trademarks, is the intellectual property of HOLTEK (and its licensors, where applicable) and is protected by copyright law and other intellectual property laws. No license, express or implied, to any intellectual property right, is granted by HOLTEK herein. HOLTEK reserves the right to revise the information described in the document at any time without prior notice. For the latest information, please contact us.