



---

Proximity Sensing 2-Key Touch Flash MCU

**BS45F3332**



Singel 3 | B-2550 Kontich | Belgium | Tel. +32 (0)3 458 30 33  
info@alcom.be | www.alcom.be  
Rivium 1e straat 52 | 2909 LE Capelle aan den IJssel | The Netherlands  
Tel. +31 (0)10 288 25 00 | info@alcom.nl | www.alcom.nl

Revision: V1.00 Date: September 14, 2021

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

## Table of Contents

<b>Features</b> .....	<b>6</b>
CPU Features .....	6
Peripheral Features.....	6
<b>General Description</b> .....	<b>7</b>
<b>Block Diagram</b> .....	<b>7</b>
<b>Pin Assignment</b> .....	<b>8</b>
<b>Pin Description</b> .....	<b>9</b>
<b>Absolute Maximum Ratings</b> .....	<b>10</b>
<b>D.C. Characteristics</b> .....	<b>11</b>
Operating Voltage Characteristics.....	11
Operating Current Characteristics.....	11
Standby Current Characteristics .....	11
<b>A.C. Characteristics</b> .....	<b>12</b>
High Speed Internal Oscillator – HIRC – Frequency Accuracy .....	12
Low Speed Internal Oscillator – LIRC – Frequency Accuracy .....	12
Operating Frequency Characteristic Curves .....	12
System Start Up Time Characteristics .....	13
<b>Input/Output Characteristics</b> .....	<b>13</b>
<b>Memory Characteristics</b> .....	<b>14</b>
<b>LVR Electrical Characteristics</b> .....	<b>14</b>
<b>Internal Reference Voltage Characteristics</b> .....	<b>14</b>
<b>A/D Converter Electrical Characteristics</b> .....	<b>15</b>
<b>Operational Amplifier Electrical Characteristics</b> .....	<b>15</b>
<b>Sink Current Generator Electrical Characteristics</b> .....	<b>16</b>
<b>Power-on Reset Characteristics</b> .....	<b>17</b>
<b>System Architecture</b> .....	<b>17</b>
Clocking and Pipelining.....	17
Program Counter.....	18
Stack .....	19
Arithmetic and Logic Unit – ALU .....	19
<b>Flash Program Memory</b> .....	<b>20</b>
Structure.....	20
Special Vectors .....	20
Look-up Table.....	20
Table Program Example.....	21
In Circuit Programming – ICP .....	22
On Chip Debug Support – OCDS .....	23

<b>Data Memory .....</b>	<b>23</b>
Structure.....	23
General Purpose Data Memory .....	24
Special Purpose Data Memory .....	24
<b>Special Function Register Description.....</b>	<b>26</b>
Indirect Addressing Registers – IAR0, IAR1 .....	26
Memory Pointers – MP0, MP1 .....	26
Accumulator – ACC .....	27
Program Counter Low Register – PCL.....	27
Look-up Table Registers – TBLP, TBHP, TBLH .....	27
Status Register – STATUS .....	27
Bank Pointer – BP.....	29
Option Memory Mapping Register – ORMC .....	29
<b>EEPROM Data Memory.....</b>	<b>30</b>
EEPROM Data Memory Structure .....	30
EEPROM Registers .....	30
Reading Data from the EEPROM .....	32
Writing Data to the EEPROM.....	32
Write Protection.....	32
EEPROM Interrupt.....	32
Programming Considerations.....	33
<b>Oscillators .....</b>	<b>34</b>
Oscillator Overview .....	34
System Clock Configurations .....	34
Internal High Speed RC Oscillator – HIRC .....	35
Internal 32kHz Oscillator – LIRC.....	35
<b>Operating Modes and System Clocks .....</b>	<b>35</b>
System Clocks .....	35
System Operation Modes.....	36
Control Registers .....	37
Operating Mode Switching.....	39
Standby Current Considerations .....	42
Wake-up.....	42
<b>Watchdog Timer.....</b>	<b>43</b>
Watchdog Timer Clock Source.....	43
Watchdog Timer Control Register .....	43
Watchdog Timer Operation .....	44
<b>Reset and Initialisation.....</b>	<b>45</b>
Reset Functions .....	45
Reset Initial Conditions .....	48
<b>Input/Output Ports .....</b>	<b>50</b>
Pull-high Resistors .....	51
Port A Wake-up .....	51
I/O Port Control Registers .....	52

I/O Port Source Current Control .....	52
Pin-shared Functions .....	53
I/O Pin Structures .....	56
Programming Considerations .....	56
<b>Timer Modules – TM .....</b>	<b>57</b>
Introduction .....	57
TM Operation .....	57
TM Clock Source .....	57
TM Interrupts .....	57
TM External Pins .....	57
Programming Considerations .....	58
<b>Compact Type TM – CTM .....</b>	<b>59</b>
Compact Type TM Operation .....	59
Compact Type TM Register Description .....	60
Compact Type TM Operating Modes .....	64
<b>Proximity Sensing Circuit .....</b>	<b>70</b>
Proximity Sensing Circuit Operation .....	70
Proximity Sensing Circuit Registers .....	71
Offset Calibration Procedure .....	73
<b>Sink Current Generator .....</b>	<b>74</b>
Sink Current Generator Registers .....	74
<b>Analog to Digital Converter .....</b>	<b>76</b>
A/D Converter Overview .....	76
A/D Converter Register Description .....	77
A/D Converter Operation .....	80
A/D Converter Reference Voltage .....	80
A/D Converter Input Signals .....	81
Conversion Rate and Timing Diagram .....	81
Summary of A/D Conversion Steps .....	82
Programming Considerations .....	83
A/D Conversion Function .....	83
A/D Conversion Programming Examples .....	84
<b>Touch Key Function .....</b>	<b>86</b>
Touch Key Structure .....	86
Touch Key Register Definition .....	86
Touch Key Operation .....	90
Touch Key Interrupt .....	91
Programming Considerations .....	91
<b>Interrupts .....</b>	<b>92</b>
Interrupt Registers .....	92
Interrupt Operation .....	94
External Interrupt .....	95
Touch Key Interrupt .....	96
A/D Converter Interrupt .....	96

TM Interrupts.....	96
EEPROM Interrupt.....	96
Time Base Interrupts.....	96
Interrupt Wake-up Function.....	98
Programming Considerations.....	98
<b>Application Circuits.....</b>	<b>99</b>
Application Circuit for 8-pin SOP.....	99
Application Circuit for 16-pin NSOP.....	100
<b>Instruction Set.....</b>	<b>101</b>
Introduction.....	101
Instruction Timing.....	101
Moving and Transferring Data.....	101
Arithmetic Operations.....	101
Logical and Rotate Operation.....	102
Branches and Control Transfer.....	102
Bit Operations.....	102
Table Read Operations.....	102
Other Operations.....	102
<b>Instruction Set Summary.....</b>	<b>103</b>
Table Conventions.....	103
<b>Instruction Definition.....</b>	<b>105</b>
<b>Package Information.....</b>	<b>114</b>
8-pin SOP (150mil) Outline Dimensions.....	115
16-pin NSOP (150mil) Outline Dimensions.....	116

## Features

### CPU Features

- Operating Voltage
  - ♦  $f_{SYS}=8\text{MHz}$ : 1.8V~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
- 4-level subroutine nesting
- Bit manipulation instruction

### Peripheral Features

- Flash Program Memory: 2K $\times$ 15
- RAM Data Memory: 128 $\times$ 8
- True EEPROM Memory: 32 $\times$ 8
- 2 touch key functions – fully integrated without requiring external components
- Watchdog Timer function
- Up to 13 bidirectional I/O lines
- Single external interrupt line shared with I/O pin
- Timer Module for time measure, compare match output and PWM output function
- Dual Time-Base functions for generation of fixed time interrupt signals
- 4 external channel 10-bit resolution A/D converter with programmable internal reference voltage  $V_{VR}$
- 2-channel sink current generator provides an infrared LED constant current driving of up to 320mA
- Proximity Sensing Circuit
  - ♦ One Operational Amplifier
- Low Voltage Reset function
- Package types: 8-pin SOP, 16-pin NSOP

## General Description

The device is a Flash Memory A/D type 8-bit high performance RISC architecture microcontroller with integrated Proximity Sensing Circuit and Touch Key functions.

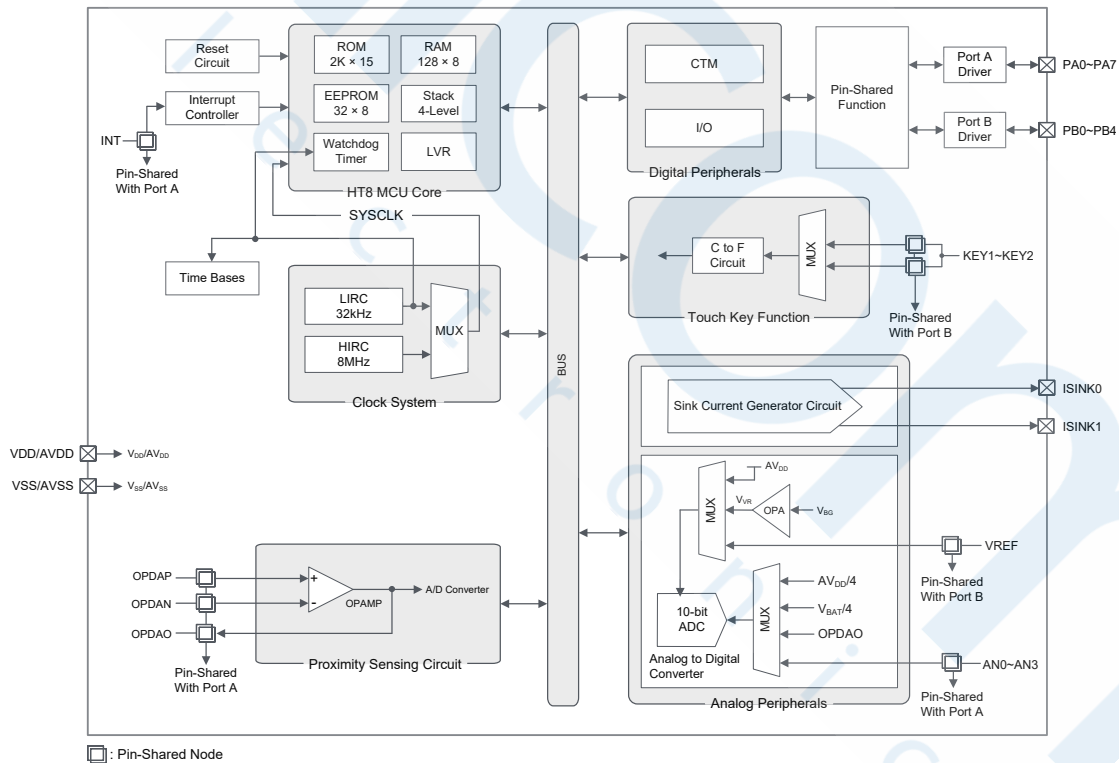
For memory features, the Flash Memory offers users the convenience of multi-programming features. Other memory includes an area of RAM Data Memory as well as an area of true EEPROM memory for storage of non-volatile data such as serial numbers, calibration data etc.

Analog features include a multi-channel 10-bit A/D converter, a Proximity Sensing Circuit mainly composed of an operational amplifier and a 2-channel Sink Current Generator. With regard to internal timers, the device includes extremely flexible Timer Module providing functions for timing, pulse generation and PWM output operations. 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.

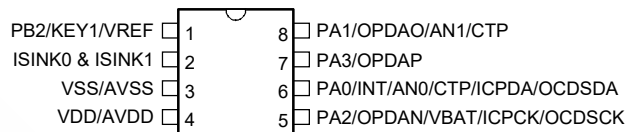
The device also includes fully integrated high and low speed oscillators which require no external components for their implementation. The ability to operate and switch dynamically between a range of operating modes using different clock sources gives users the ability to optimise microcontroller operation and minimise power consumption.

The inclusion of flexible I/O programming features, Time-Base functions, touch keys along with many other features ensure that the device will find excellent use in various Proximity Sensing products.

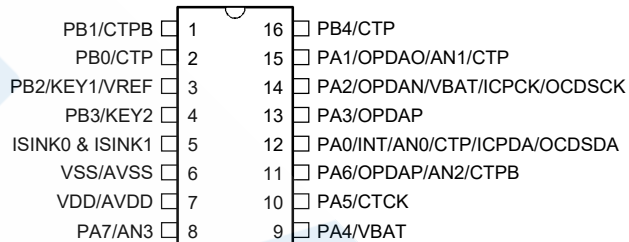
## Block Diagram



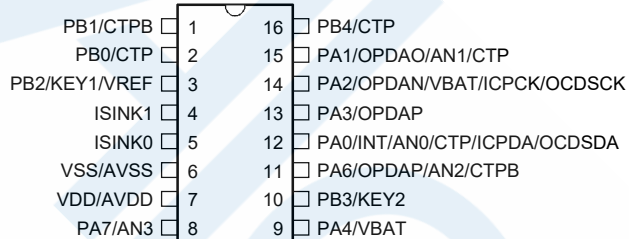
## Pin Assignment



**BS45F3332/BS45V3332**  
**8 SOP-A**



**BS45F3332/BS45V3332**  
**16 NSOP-A**



**BS45F3332/BS45V3332**  
**16 NSOP-B**

- 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 OCSDA and OCDSCK pins are supplied as OCDS dedicated pins and as such only available for the BS45V3332 device which is the OCDS EV chip for the BS45F3332 device.
3. When the ISINK0 and ISINK1 pins are bonded to the same pin location (8SOP-A or 16NSOP-A) or externally connected together by users (16NSOP-B), their corresponding internal switches and current control registers should be properly configured to avoid current leakage problem. Refer to the “Sink Current Generator” section.
4. For less pin-count package types there will be unbonded pins which should be properly configured to avoid unwanted current consumption resulting from floating input conditions. Refer to the “Standby Current Considerations” and “Input/Output Ports” sections.



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

Pin Name	Function	OPT	I/T	O/T	Description
PA0/INT/AN0/CTP/ ICPDA/OCSDSA	PA0	PAPU PAWU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-high and wake-up
	INT	PAS0 INTC0 INTEG	ST	—	External interrupt input
	AN0	PAS0	AN	—	A/D Converter external input channel 0
	CTP	PAS0	—	CMOS	CTM output
	ICPDA	—	ST	CMOS	ICP data/address
	OCSDSA	—	ST	CMOS	OCDS data/address, for EV chip only
PA1/OPDAO/AN1/CTP	PA1	PAPU PAWU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-high and wake-up
	OPDAO	PAS0	—	AN	OPAMP output
	AN1	PAS0	AN	—	A/D Converter external input channel 1
	CTP	PAS0	—	CMOS	CTM output
PA2/OPDAN/VBAT/ ICPCK/OCDSCK	PA2	PAPU PAWU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-high and wake-up
	OPDAN	PAS0	AN	—	OPAMP inverting input
	VBAT	PAS0	AN	—	A/D Converter external input
	ICPCK	—	ST	—	ICP clock pin
PA3/OPDAP	PA3	PAPU PAWU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-high and wake-up
	OPDAP	PAS0	AN	—	OPAMP Non-inverting input
PA4/VBAT	PA4	PAPU PAWU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-high and wake-up
	VBAT	PAS1	AN	—	A/D Converter external input
PA5/CTCK	PA5	PAPU PAWU	ST	CMOS	General purpose I/O. Register enabled pull-high and wake-up
	CTCK	—	ST	—	CTM clock input
PA6/OPDAP/AN2/CTPB	PA6	PAPU PAWU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-high and wake-up
	OPDAP	PAS1	AN	—	OPAMP Non-inverting input
	AN2	PAS1	AN	—	A/D Converter external input channel 2
	CTPB	PAS1	—	CMOS	CTM inverting output
PA7/AN3	PA7	PAPU PAWU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-high and wake-up
	AN3	PAS1	AN	—	A/D Converter external input channel 3

Pin Name	Function	OPT	I/T	O/T	Description
PB0/CTP	PB0	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-high
	CTP	PBS0	—	CMOS	CTM output
PB1/CTPB	PB1	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-high
	CTPB	PBS0	—	CMOS	CTM inverting output
PB2/KEY1/VREF	PB2	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-high
	KEY1	PBS0	AN	—	Touch key input
	VREF	PBS0	AN	—	A/D Converter external reference input
PB3/KEY2	PB3	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-high
	KEY2	PBS0	AN	—	Touch key input
PB4/CTP	PB4	PBPU PBS1	ST	CMOS	General purpose I/O. Register enabled pull-high
	CTP	PBS1	—	CMOS	CTM output
ISINK0	ISINK0	—	—	AN	Sink0 current source
ISINK1	ISINK1	—	—	AN	Sink1 current source
VDD/AVDD	VDD	—	PWR	—	Digital positive power supply
	AVDD	—	PWR	—	Analog positive power supply
VSS/AVSS	VSS	—	PWR	—	Digital negative power supply
	AVSS	—	PWR	—	Analog negative power supply

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

### Absolute Maximum Ratings

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

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

## D.C. Characteristics

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

### Operating Voltage Characteristics

Ta=-40°C~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	1.8	—	5.5	V
	Operating Voltage – LIRC	f <sub>sys</sub> =f <sub>LIRC</sub> =32kHz	1.8	—	5.5	V

### Operating Current Characteristics

Ta=25°C

Symbol	Operating Mode	Test Conditions		Min.	Typ.	Max.	Unit
		V <sub>DD</sub>	Conditions				
I <sub>DD</sub>	SLOW Mode – LIRC	1.8V	f <sub>sys</sub> =32kHz	—	8	16	μA
		3V		—	10	20	
		5V		—	30	50	
	FAST Mode – HIRC	1.8V	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 setup in a non-floating condition.
2. All measurements are taken under conditions of no load and with all peripherals in an off state.
3. There are no DC current paths.
4. All Operating Current values are measured using a continuous NOP instruction program loop.

### Standby Current Characteristics

Ta=25°C, unless otherwise specified

Symbol	Standby Mode	Test Conditions		Min.	Typ.	Max.	Max. @85°C	Unit
		V <sub>DD</sub>	Conditions					
I <sub>STB</sub>	SLEEP Mode	1.8V	WDT off	—	0.11	0.15	2.00	μA
		3V		—	0.11	0.15	2.00	
		5V		—	0.18	0.38	2.90	
	IDLE0 Mode – LIRC	1.8V	f <sub>sub</sub> on	—	2.4	4.0	4.8	μA
		3V		—	3	5	6	
		5V		—	5	10	12	
	IDLE1 Mode – HIRC	1.8V	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 setup in a non-floating condition.
2. All measurements are taken under conditions of no load and with all peripherals in an off state.
3. There are no DC current paths.
4. All Standby Current values are taken after a HALT instruction execution thus stopping all instruction execution.

## A.C. Characteristics

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

### 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	-3%	8	+3%	
		2.2V~5.5V	25°C	-2.5%	8	+2.5%	
			-40°C~85°C	-4%	8	+4%	
		1.8V~5.5V	25°C	-5%	8	+5%	
			-40°C~85°C	-10%	8	+10%	

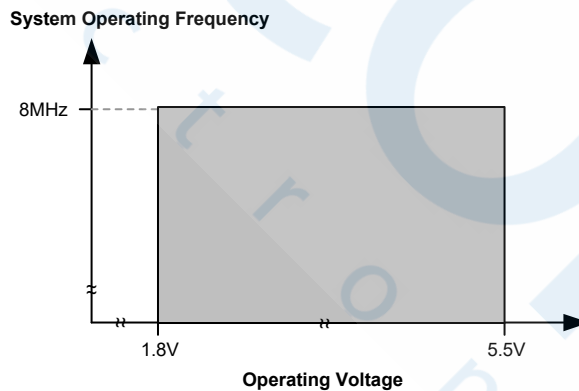
Note: 1. The 3V/5V values for V<sub>DD</sub> are provided as these are the two selectable fixed voltages at which the HIRC frequency is trimmed by the writer.

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 1.8V to 3.6V and fixed at 5V for application voltage ranges from 3.3V to 5.5V.

### Low Speed Internal Oscillator – LIRC – Frequency Accuracy

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V <sub>DD</sub>	Temp.				
f <sub>LIRC</sub>	LIRC Frequency	1.8V~5.5V	-40°C~85°C	-7%	32	+7%	kHz
t <sub>START</sub>	LIRC Start-up Time	—	-40°C~85°C	—	—	100	μs

### Operating Frequency Characteristic Curves



### System Start Up Time Characteristics

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
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>	—	16	—	t <sub>HIRC</sub>
		f <sub>sys</sub> =f <sub>SUB</sub> =f <sub>LIRC</sub>	—	2	—	t <sub>LIRC</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>H</sub>
		f <sub>sys</sub> =f <sub>SUB</sub> =f <sub>LIRC</sub>	—	2	—	t <sub>SUB</sub>
t <sub>RSTD</sub>	System Speed Switch Time (FAST to SLOW Mode or SLOW to FAST Mode)	f <sub>HIRC</sub> switches from off → on	—	16	—	t <sub>HIRC</sub>
	System Reset Delay Time (Reset source from Power-on reset or LVR Hardware Reset)	RR <sub>POR</sub> =5V/ms				
	System Reset Delay Time (LVRC/WBTC Register Software Reset)	—	14	16	18	ms
	System Reset Delay Time (WDT Overflow Reset)	—				
t <sub>SRESET</sub>	Minimum Software Reset Width to Reset	—	45	90	120	μs

- 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>SYS</sub>=1/f<sub>SYS</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

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V <sub>DD</sub>	Conditions				
V <sub>IL</sub>	Input Low Voltage for I/O Ports	5V	—	0	—	1.5	V
		—		0	—	0.2V <sub>DD</sub>	
V <sub>IH</sub>	Input High Voltage for I/O Ports	5V	—	3.5	—	5.0	V
		—		0.8V <sub>DD</sub>	—	V <sub>DD</sub>	
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 (PB0~PB4)	3V	V <sub>OH</sub> = 0.9V <sub>DD</sub>	-4	-8	—	mA
		5V		-8	-16	—	
	Source Current for I/O Pins (PA0~PA7)	3V	V <sub>OH</sub> = 0.9V <sub>DD</sub> , SLEDC[m+1: m] = 00B (m = 0 or 2 or 4 or 6)	-0.7	-1.5	—	mA
		5V		-1.5	-2.9	—	
		3V	V <sub>OH</sub> = 0.9V <sub>DD</sub> , SLEDC[m+1: m] = 01B (m = 0 or 2 or 4 or 6)	-1.3	-2.5	—	mA
		5V		-2.5	-5.1	—	
		3V	V <sub>OH</sub> = 0.9V <sub>DD</sub> , SLEDC[m+1: m] = 10B (m = 0 or 2 or 4 or 6)	-1.8	-3.6	—	mA
		5V		-3.6	-7.3	—	
3V	V <sub>OH</sub> = 0.9V <sub>DD</sub> , SLEDC[m+1: m] = 11B (m = 0 or 2 or 4 or 6)	-4	-8	—	mA		
5V		-8	-16	—			
R <sub>PH</sub>	Pull-high Resistance for I/O Ports <small>(Note)</small>	3V	LVPU=0, PxPU=FFH (Px=PA, PB)	20	60	100	kΩ
		5V		10	30	50	
		3V	LVPU=1, PxPU=FFH (Px=PA, PB)	6.67	15.00	23.00	
		5V		3.5	7.5	12.0	

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V <sub>DD</sub>	Conditions				
I <sub>LEAK</sub>	Input Leakage Current	5V	V <sub>IN</sub> =V <sub>DD</sub> or V <sub>IN</sub> =V <sub>SS</sub>	—	—	±1	μA
t <sub>TCK</sub>	CTCK Input Pin Minimum Pulse Width	—	—	0.3	—	—	μs
t <sub>INT</sub>	Interrupt Pin Minimum Pulse Width	—	—	10	—	—	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				
V <sub>DD</sub>	Operating Voltage for Write – Data EEPROM Memory	—	—	2.2	—	5.5	V
t <sub>DEW</sub>	Erase / Write Cycle Time – Flash Program Memory	—	—	—	2	3	ms
t <sub>EERD</sub>	Read time	—	—	—	—	4	t <sub>sys</sub>
I <sub>DDPGM</sub>	Programming / Erase Current on V <sub>DD</sub>	—	—	—	—	5.0	mA
E <sub>P</sub>	Cell Endurance – Flash Program Memory	—	—	10K	—	—	E/W
	Cell Endurance –Data EEPROM Memory	—	—	100K	—	—	
t <sub>RETD</sub>	ROM Data Retention Time	—	Ta=25°C	—	40	—	Year

Note: “E/W” means Erase/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>DD</sub>	Operating Voltage	—	—	1.8	—	5.5	V
V <sub>LVR</sub>	Low Voltage Reset Voltage	—	LVR enable	-5%	1.7	+5%	V
I <sub>LVRBG</sub>	Operating Current	3V	LVR enable, V <sub>LVR</sub> =1.7V	—	—	15	μA
		5V		—	15	25	
t <sub>LVR</sub>	Minimum Low Voltage Width to Reset	—	—	120	240	480	μs
I <sub>LVR</sub>	Additional Current for LVR Enable	5V	VBGEN=0	—	—	25	μA

## Internal Reference Voltage Characteristics

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V <sub>DD</sub>	Conditions				
t <sub>BGS</sub>	V <sub>BG</sub> Turn On Stable Time	—	No load	—	—	50	μs
I <sub>BG</sub>	Additional Current for Bandgap Reference Enable	—	VBGEN=1, LVR disable	—	—	2	μA

Note: The V<sub>BG</sub> voltage is used as the A/D converter internal signal input.

## A/D Converter Electrical Characteristics

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

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V <sub>DD</sub>	Conditions				
V <sub>DD</sub>	Operating Voltage	—	—	1.8	—	5.5	V
V <sub>ADI</sub>	Input Voltage	—	—	0	—	V <sub>REF</sub>	V
V <sub>REF</sub>	Reference Voltage	—	—	1.6	—	V <sub>DD</sub>	V
N <sub>R</sub>	Resolution	—	—	—	—	10	Bit
DNL	Differential Non-linearity	—	V <sub>REF</sub> =V <sub>DD</sub> , t <sub>ADCK</sub> =0.5μs	-1.5	—	1.5	LSB
INL	Integral Non-linearity	—	V <sub>REF</sub> =V <sub>DD</sub> , t <sub>ADCK</sub> =0.5μs	-2	—	2	LSB
I <sub>ADC</sub>	Additional Current for A/D Converter Enable	1.8V	No load, t <sub>ADCK</sub> =0.5μs	—	300	420	μA
		3V		—	340	500	
		5V		—	500	700	
t <sub>ADCK</sub>	Clock Period	—	—	0.5	—	10.0	μs
t <sub>ON2ST</sub>	A/D Converter On-to-Start Time	—	—	4	—	—	μs
t <sub>ADS</sub>	Sampling Time	—	—	—	4	—	t <sub>ADCK</sub>
t <sub>ADC</sub>	Conversion Time (Including A/D Sample and Hold Time)	—	—	—	14	—	t <sub>ADCK</sub>
GERR	A/D Conversion Gain Error	—	V <sub>REF</sub> =V <sub>DD</sub>	-2	—	2	LSB
OSRR	A/D Conversion Offset Error	—	V <sub>REF</sub> =V <sub>DD</sub>	-2	—	2	LSB
I <sub>PGA</sub>	Additional Current for OPA Enable	3V	No load	—	390	550	μA
		5V		—	500	650	
V <sub>OR</sub>	OPA Maximum Output Voltage Range	3V	—	V <sub>SS</sub> +0.1	—	V <sub>DD</sub> -0.1	V
		5V		V <sub>SS</sub> +0.1	—	V <sub>DD</sub> -0.1	
V <sub>VR</sub>	OPA Fix Voltage Output	1.8V~5.5V	Ta=25°C	-5%	1.6	+5%	V
		1.8V~5.5V	Ta=-40°C~85°C	-10%	1.6	+10%	
		3.2V~5.5V	Ta=25°C	-5%	3.0	+5%	
		3.2V~5.5V	Ta=-40°C~85°C	-10%	3.0	+10%	
		4.2V~5.5V	Ta=25°C	-5%	4.0	+5%	
4.2V~5.5V	Ta=-40°C~85°C	-10%	4.0	+10%			

## Operational Amplifier Electrical Characteristics

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V <sub>DD</sub>	Conditions				
V <sub>DD</sub>	Operating Voltage	—	—	1.8	5.0	5.5	V
I <sub>OPA</sub>	Additional Current for OPA Enable	—	No load, OPDABW=0	—	80	200	μA
			No load, OPDABW=1	—	230	510	
V <sub>OS</sub>	Input Offset Voltage	5V	Without calibration (OPDAOF[5:0]=100000B)	-15	—	+15	mV
			With calibration	-2	—	+2	
I <sub>OS</sub>	Input Offset Current	5V	V <sub>IN</sub> =1/2 V <sub>CM</sub>	—	1	10	nA
V <sub>CM</sub>	Common Mode Voltage Range	—	—	V <sub>SS</sub>	—	V <sub>DD</sub> -1.4	V
PSRR	Power Supply Rejection Ratio	5V	—	50	70	—	dB
CMRR	Common Mode Rejection Ratio	5V	—	50	80	—	dB
A <sub>OL</sub>	Open Loop Gain	—	—	60	80	—	dB



Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V <sub>DD</sub>	Conditions				
SR	Slew Rate	5V	R <sub>LOAD</sub> =1MΩ, C <sub>LOAD</sub> =60pF, OPDABW=0	180	500	—	V/ms
			R <sub>LOAD</sub> =1MΩ, C <sub>LOAD</sub> =60pF, OPDABW=1	600	1800	—	
GBW	Gain Bandwidth	5V	R <sub>LOAD</sub> =1MΩ, C <sub>LOAD</sub> =60pF, OPDABW=0	250	600	—	kHz
			R <sub>LOAD</sub> =1MΩ, C <sub>LOAD</sub> =60pF, OPDABW=1	800	2000	—	
V <sub>OR</sub>	Maximum Output Voltage Range	—	R <sub>LOAD</sub> =5kΩ to V <sub>DD</sub> /2	V <sub>SS</sub> +120	—	V <sub>DD</sub> -180	mV

## Sink Current Generator Electrical Characteristics

T<sub>a</sub>=-40°C~85°C, unless otherwise specified

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V <sub>DD</sub>	Conditions				
V <sub>DD</sub>	Operating Voltage	—	—	1.8	—	5.5	V
I <sub>SINK0</sub>	Sink Current for ISINK0 Pin	5V	(After trimming) T <sub>a</sub> =25°C, V <sub>ISINK0</sub> =3.0V, ISGDATA0[3:0]=0000B, ISST0=0	1.90	2.00	2.10	mA
		—	(After trimming) T <sub>a</sub> =-40°C~85°C, V <sub>ISINK0</sub> =1.0V~4.5V, ISGDATA0[3:0]=0000B, ISST0=0	1.64	2.00	2.36	
		—	(After trimming) T <sub>a</sub> =-40°C~85°C, V <sub>ISINK0</sub> =0.7V~1.0V, ISGDATA0[3:0]=0000B, ISST0=0	1.40	2.00	2.10	
		5V	(After trimming) T <sub>a</sub> =25°C, V <sub>ISINK0</sub> =3.0V, ISGDATA0[3:0]=1111B, ISST0=1	147	160	173	
		—	(After trimming) T <sub>a</sub> =-40°C~85°C, V <sub>ISINK0</sub> =1.0V~4.5V, ISGDATA0[3:0]=1111B, ISST0=1	128	160	189	
		—	(After trimming) T <sub>a</sub> =-40°C~85°C, V <sub>ISINK0</sub> =0.7V~1.0V, ISGDATA0[3:0]=1111B, ISST0=1	112	160	173	
I <sub>SINK1</sub>	Sink Current for ISINK1 Pin	5V	(After trimming) T <sub>a</sub> =25°C, V <sub>ISINK1</sub> =3.0V, ISGDATA1[3:0]=0000B, ISST1=0	1.80	2.00	2.20	mA
		—	(After trimming) T <sub>a</sub> =-40°C~85°C, V <sub>ISINK1</sub> =1.0V~4.5V, ISGDATA1[3:0]=000B, ISST1=0	1.64	2.00	2.36	
		—	(After trimming) T <sub>a</sub> =-40°C~85°C, V <sub>ISINK1</sub> =0.7V~1.0V, ISGDATA1[3:0]=0000B, ISST1=0	1.40	2.00	2.20	
		5V	(After trimming) T <sub>a</sub> =25°C, V <sub>ISINK1</sub> =3.0V, ISGDATA1[3:0]=1111B, ISST1=1	144	160	176	
		—	(After trimming) T <sub>a</sub> =-40°C~85°C, V <sub>ISINK1</sub> =1.0V~4.5V, ISGDATA1[3:0]=1111B, ISST1=1	128	160	189	
		—	(After trimming) T <sub>a</sub> =-40°C~85°C, V <sub>ISINK1</sub> =0.7V~1.0V, ISGDATA1[3:0]=1111B, ISST1=1	112	160	176	

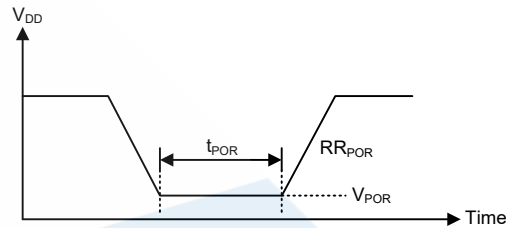
Note: When the ISINK0 and ISINK1 pins are bounded to the same pin location or externally connected together, their corresponding internal switches and current control registers should be properly configured to avoid current leakage problem. Refer to the “Sink Current Generator” section.



## Power-on Reset Characteristics

Ta=-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



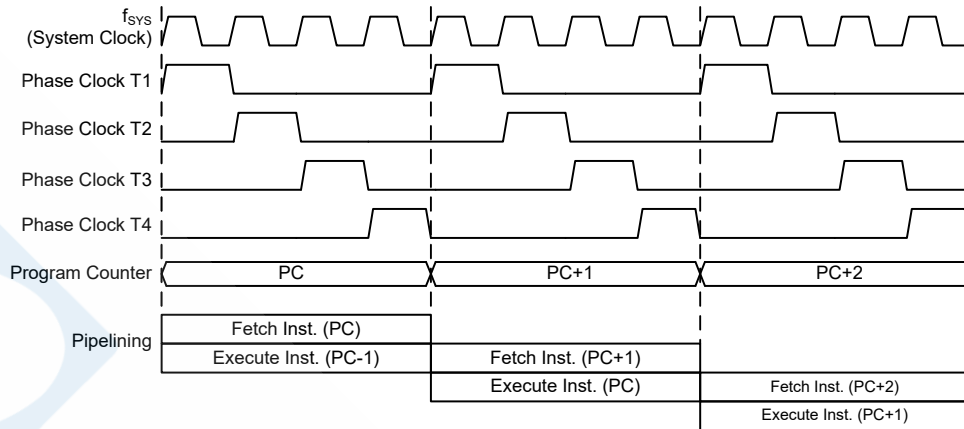
## System Architecture

A key factor in the high-performance features of the Holtek range of microcontrollers is attributed to their internal system architecture. The device takes 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 and A/D control system with maximum reliability and flexibility. This makes the device suitable for low-cost, high-volume production for controller applications.

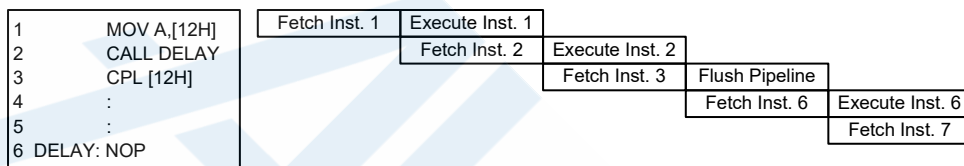
### Clocking and Pipelining

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

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 demands 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.

Program Counter	
High Byte	Low Byte (PCL)
PC10~PC8	PCL7~PCL0

Program Counter

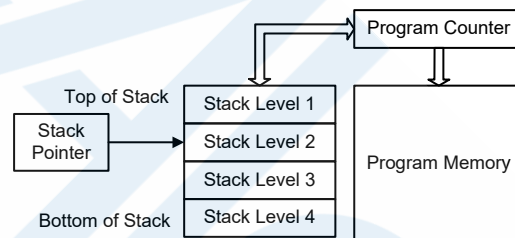
The lower byte of the Program Counter, known as the Program Counter Low register or PCL, is available for program control and is a readable and writeable register. By transferring data directly into this register, a short program jump can be executed directly; however, as only this low byte is available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

## Stack

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

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

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



## Arithmetic and Logic Unit – ALU

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

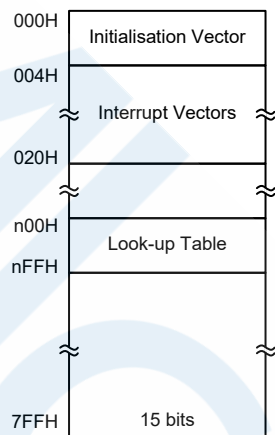
- Arithmetic operations:  
ADD, ADDM, ADC, ADCM, SUB, SUBM, SBC, SBCM, DAA
- Logic operations:  
AND, OR, XOR, ANDM, ORM, XORM, CPL, CPLA
- Rotation:  
RRA, RR, RRCA, RRC, RLA, RL, RLCA, RLC
- Increment and Decrement:  
INCA, INC, DECA, DEC
- Branch decision:  
JMP, SZ, SZA, SNZ, SIZ, SDZ, SIZA, SDZA,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

## Flash Program Memory

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

### Structure

The Program Memory has a capacity of 2K×15 bits. The Program Memory is addressed by the Program Counter and also contains data, table information and interrupt entries. Table data, which can be setup in any location within the Program Memory, is addressed by a separate table pointer register.



**Program Memory Structure**

### Special Vectors

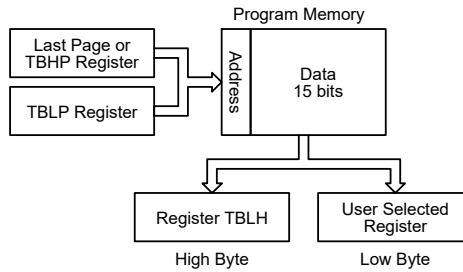
Within the Program Memory, certain locations are reserved for the reset and interrupts. The location 000H is reserved for use by the device reset for program initialisation. After a device reset is initiated, the program will jump to this location and begin execution.

### Look-up Table

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

After setting up the table pointer, the table data can be retrieved from the Program Memory using the “TABRD [m]” or “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.

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



### Table Program Example

The following example shows how the table pointer and table data is defined and retrieved from the microcontroller. This example uses raw table data located in the Program Memory which is stored there using the ORG statement. The value at this ORG statement is “700H” which refers to the start address of the last page within the 2K Program Memory of the device. The table pointer low byte register is setup here to have an initial value of “06H”. This will ensure that the first data read from the data table will be at the Program Memory address “706H” or 6 locations after the start of the last page. Note that the value for the table pointer is referenced to the first address specified by TBLP and TBHP 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 can not be restored, care should be taken to ensure its protection if both the main routine and Interrupt Service Routine use table read instructions. If using the table read instructions, the Interrupt Service Routines may change the value of the TBLH and subsequently cause errors if used again by the main routine. As a rule, it is recommended that simultaneous use of the table read instructions should be avoided. However, in situations where simultaneous use cannot be avoided, the interrupts should be disabled prior to the execution of any main routine table-read instructions. Note that all table related instructions require two instruction cycles to complete their operation.

### Table Read Program Example

```

tempreg1 db ? ; temporary register #1
tempreg2 db ? ; temporary register #2
:
:
mov a,06h ; initialise low table pointer - note that this address is referenced
mov tblp,a ; to the last page or the page that tblp pointed
:
:
tabrd tempreg1 ; transfers value in table referenced by table pointer data at program
; memory address "706H" transferred to tempreg1 and TBLH dec tblp
; reduce value of table pointer by one
tabrd tempreg2 ; transfers value in table referenced by table pointer
; data at program memory address "705H" transferred to tempreg2 and TBLH
; in this example the data "1AH" is transferred to tempreg1
; and data "0FH" to register tempreg2 and the data "00H" is transferred to TBLH
:
:
org 700h ; sets initial address of program memory
dc 00Ah, 00Bh, 00Ch, 00Dh, 00Eh, 00Fh, 01Ah, 01Bh
:
:
  
```

### In Circuit Programming – ICP

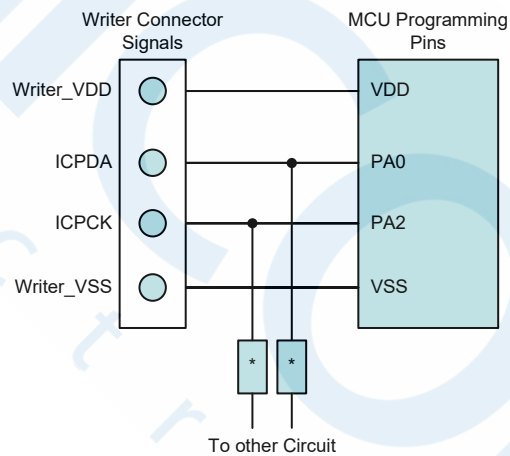
The provision of Flash Type Program Memory provides the user with a means of convenient and easy upgrades and modifications to their programs on the same device. As an additional convenience, a means of programming the microcontroller in-circuit has provided using a 4-pin interface. This provides manufacturers with the possibility of manufacturing their circuit boards complete with a programmed or un-programmed microcontroller, and then programming or upgrading the program at a later stage. This enables product manufacturers to easily keep their manufactured products supplied with the latest program releases without removal and re-insertion of the device.

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

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

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

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



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

### On Chip Debug Support – OCDS

An EV chip exists for the purposes of device emulation. This 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. Users can use the EV chip device to emulate the real chip device behavior by connecting the OCSDA and OCDSCK pins to the Holtek HT-IDE development tools. The OCSDA pin is the OCDS Data/Address input/output pin while the OCDSCK pin is the OCDS clock input pin. When users use the EV chip for debugging, other functions which are shared with the OCSDA and OCDSCK pins in the device will have no effect in the EV chip. However, the two OCDS pins which are pin-shared with the ICP programming pins are still used as the Flash Memory programming pins for ICP. For more detailed OCDS information, refer to the corresponding document named “Holtek e-Link for 8-bit MCU OCDS User’s Guide”.

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

### Data Memory

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

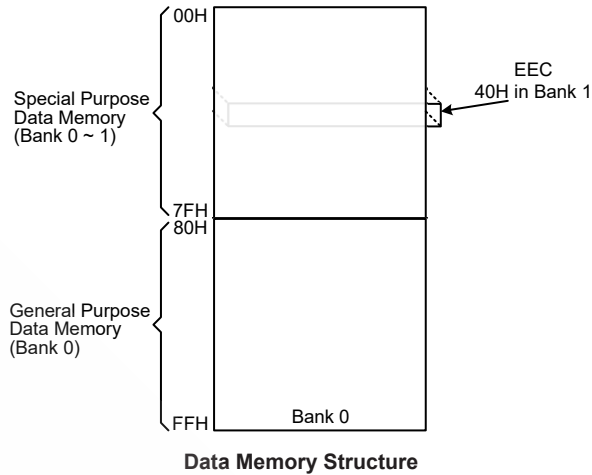
#### Structure

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

The Data Memory is subdivided into two banks, all of which are implemented in 8-bit wide RAM. Switching between the different Data Memory banks is achieved by properly setting the Bank Pointer to the correct value. The address range of the Special Purpose Data Memory for the device is from 00H to 7FH while the General Purpose Data Memory address range is from 80H to FFH.

Special Purpose Data Memory	General Purpose Data Memory	
Located Banks	Capacity	Bank: Address
0: 00H~7FH 1: 40H (EEC only)	128×8	0: 80H~FFH





### General Purpose Data Memory


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

### Special Purpose Data Memory

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



	Bank 0	Bank 1		Bank 0	Bank 1
00H	IAR0		40H		EEC
01H	MP0		41H	OPSWA	
02H	IAR1		42H	OPSWB	
03H	MP1		43H	OPSWC	
04H	BP		44H	OPDC	
05H	ACC		45H	OPDACAL	
06H	PCL		46H	SADC0	
07H	TBLP		47H	SADC1	
08H	TBLH		48H	SADOH	
09H	TBHP		49H	SADOL	
0AH	STATUS		4AH	ORMC	
0BH	SCC		4BH		
0CH	HIRCC				
0DH	LVRC				
0EH	INTEG				
0FH	RSTFC				
10H	INTC0				
11H	INTC1				
12H	INTC2				
13H					
14H	PA				
15H	PAC				
16H	PAPU				
17H	PAWU				
18H	PB				
19H	PBC				
1AH	PBPU				
1BH					
1CH					
1DH	SLEDC				
1EH	VBGC				
1FH	WDTC				
20H	PSC0R				
21H	TB0C				
22H	PSC1R				
23H	TB1C				
24H	EEA				
25H	EED				
26H	PAS0				
27H	PAS1				
28H	PBS0				
29H	PBS1				
2AH					
2BH	TKTMR				
2CH	TKC0				
2DH	TKC1				
2EH	TK16DL				
2FH	TK16DH				
30H	TKMOC0				
31H	TKMOC1				
32H	TKM016DL				
33H	TKM016DH				
34H	TKM0ROL				
35H	TKM0ROH				
36H	CTMC0				
37H	CTMC1				
38H	CTMDL				
39H	CTMDH				
3AH	CTMAL				
3BH	CTMAH				
3CH	ISGENC				
3DH	ISGDATA0				
3EH	ISGDATA1				
3FH	LVPUC		7FH		

 : Unused, read as 00H

**Special Purpose Data Memory Structure**

## Special Function Register Description

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

### Indirect Addressing Registers – IAR0, IAR1

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

### Memory Pointers – MP0, MP1

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

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

#### Indirect Addressing Program

```
data .section 'data'
adres1 db ?
adres2 db ?
adres3 db ?
adres4 db ?
block db ?
code .section at 0 'code'
org 00h
start:
    mov a, 04h           ; setup size of block
    mov block, a
    mov a, offset adres1 ; Accumulator loaded with first RAM address
    mov mp0, a          ; setup memory pointer with first RAM address
loop:
    clr IAR0            ; clear the data at address defined by MP0
    inc mp0             ; 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 example shown above, no reference is made to specific Data Memory addresses.

### **Accumulator – ACC**

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

### **Program Counter Low Register – PCL**

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

### **Look-up Table Registers – TBLP, TBHP, TBLH**

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

### **Status Register – STATUS**

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

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

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

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

- PDF is cleared by a system power-up or executing the “CLR WDT” instruction. PDF is set by executing the “HALT” instruction.
- TO is cleared by a system power-up or executing the “CLR WDT” or “HALT” instruction. TO is set by a WDT time-out.

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

• **STATUS Register**

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

“x”: Unknown

- Bit 7~6 Unimplemented, read as “0”
- Bit 5 **TO**: Watchdog Time-out flag  
0: After power up or executing the “CLR WDT” or “HALT” instruction  
1: A watchdog time-out occurred.
- Bit 4 **PDF**: Power down flag  
0: After power up or executing the “CLR WDT” instruction  
1: By executing the “HALT” instruction
- Bit 3 **OV**: Overflow flag  
0: No overflow  
1: An operation results in a carry into the highest-order bit but not a carry out of the highest-order bit or vice versa.
- Bit 2 **Z**: Zero flag  
0: The result of an arithmetic or logical operation is not zero  
1: The result of an arithmetic or logical operation is zero
- Bit 1 **AC**: Auxiliary flag  
0: No auxiliary carry  
1: An operation results in a carry out of the low nibbles in addition, or no borrow from the high nibble into the low nibble in subtraction
- Bit 0 **C**: Carry flag  
0: No carry-out  
1: An operation results in a carry during an addition operation or if a borrow does not take place during a subtraction operation  
The “C” flag is also affected by a rotate through carry instruction.

### Bank Pointer – BP

For this device, the Data Memory is divided into two banks, Bank 0 and Bank 1. Selecting the required Data Memory area is achieved using the Bank Pointer. Bit 0 of the Bank Pointer is used to select Data Memory Banks 0~1.

The Data Memory is initialised to Bank 0 after a reset, except for a WDT time-out reset in the IDLE or SLEEP Mode, in which case, the Data Memory bank remains unaffected. Directly addressing the Data Memory will always result in Bank 0 being accessed irrespective of the value of the Bank Pointer. Accessing data from Bank 1 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

### Option Memory Mapping Register – ORMC

The ORMC register is used to enable the Option Memory Mapping function. The Option Memory capacity is 32 words. When a specific pattern of 55H and AAH is consecutively written into this register, the Option Memory Mapping function will be enabled and then the Option Memory code can be read by using the table read instruction. The Option Memory addresses 00H~1FH will be mapped to Program Memory last page addresses E0H~FFH.

To successfully enable the Option Memory Mapping function, the specific pattern of 55H and AAH must be written into the ORMC register in two consecutive instruction cycles. It is therefore recommended that the global interrupt bit EMI should first be cleared before writing the specific pattern, and then set high again at a proper time according to users’ requirements after the pattern is successfully written. An internal timer will be activated when the pattern is successfully written. The mapping operation will be automatically finished after a period of  $4 \times T_{LIRC}$ . Therefore, users should read the data in time, otherwise the Option Memory Mapping function needs to be restarted. After the completion of each consecutive write operation to the ORMC register, the timer will recount.

When the table read instructions are used to read the Option Memory code, both “TABRD [m]” and “TABRDL [m]” instructions can be used. However, care must be taken if the “TABRD [m]” instruction is used, the table pointer defined by the TBHP register must be referenced to the last page. Refer to corresponding sections about the table read instruction for more details.

#### • ORMC Register

Bit	7	6	5	4	3	2	1	0
Name	ORMC7	ORMC6	ORMC5	ORMC4	ORMC3	ORMC2	ORMC1	ORMC0
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 **ORMC7~ORMC0**: Option Memory Mapping specific pattern

When a specific pattern of 55H and AAH is written into this register, the Option Memory Mapping function will be enabled. Note that the register content will be cleared after the MCU is woken up from the IDLE/SLEEP mode.

## EEPROM Data Memory

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

### EEPROM Data Memory Structure

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

### EEPROM Registers

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

Register Name	Bit							
	7	6	5	4	3	2	1	0
EEA	—	—	—	EEA4	EEA3	EEA2	EEA1	EEA0
EED	EED7	EED6	EED5	EED4	EED3	EED2	EED1	EED0
EEC	D7	—	—	—	WREN	WR	RDEN	RD

**EEPROM Register List**

#### • EEA Register

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

Bit 7~5 Unimplemented, read as “0”

Bit 4~0 **EEA4~EEA0**: Data EEPROM address bit 4 ~ bit 0

#### • EED Register

Bit	7	6	5	4	3	2	1	0
Name	EED7	EED6	EED5	EED4	EED3	EED2	EED1	EED0
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 **EED7~EED0**: Data EEPROM data bit 7 ~ bit 0

• **EEC Register**

Bit	7	6	5	4	3	2	1	0
Name	D7	—	—	—	WREN	WR	RDEN	RD
R/W	R/W	—	—	—	R/W	R/W	R/W	R/W
POR	0	—	—	—	0	0	0	0

Bit 7      **D7**: Reserved bit, must be fixed at “0”

Bit 6~4    Unimplemented, read as “0”

Bit 3      **WREN**: Data EEPROM Write Enable  
             0: Disable  
             1: Enable

This is the Data EEPROM Write Enable Bit which must be set high before Data EEPROM write operations are carried out. Clearing this bit to zero will inhibit Data EEPROM write operations.

Bit 2      **WR**: EEPROM Write Control  
             0: Write cycle has finished  
             1: Activate a write cycle

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

Bit 1      **RDEN**: Data EEPROM Read Enable  
             0: Disable  
             1: Enable

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

Bit 0      **RD**: EEPROM Read Control  
             0: Read cycle has finished  
             1: Activate a read cycle

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

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

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

3. Ensure that the write operation is totally complete before changing the contents of the EEPROM related registers.



### Reading Data from the EEPROM

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

### Writing Data to the EEPROM

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

### Write Protection

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

### EEPROM Interrupt

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



## Programming Considerations

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

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

## Programming Examples

### Reading data from the EEPROM – polling method

```
MOV A, EEPROM_ADRES      ; user defined address
MOV EEA, A
MOV A, 40H                ; setup memory pointer MP1
MOV MP1, A                ; MP1 points to EEC register
MOV A, 01H                ; setup Bank Pointer
MOV BP, A
SET IAR1.1                ; set RDEN bit, enable read operations
SET IAR1.0                ; start Read Cycle - set RD bit
BACK:
SZ IAR1.0                 ; check for read cycle end
JMP BACK
CLR IAR1                  ; disable EEPROM read if no more read operations are required
CLR BP
MOV A, EED                 ; move read data to register
MOV READ_DATA, A
```

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

### Writing Data to the EEPROM – polling method

```
MOV A, EEPROM_ADRES      ; user defined address
MOV EEA, A
MOV A, EEPROM_DATA       ; user defined data
MOV EED, A
MOV A, 40H                ; setup memory pointer MP1
MOV MP1, A                ; MP1 points to EEC register
MOV A, 01H                ; setup Bank Pointer
MOV BP, A
CLR EMI
SET IAR1.3                ; set WREN bit, enable write operations
SET IAR1.2                ; start Write Cycle - set WR bit - executed immediately
                        ; after setting WREN bit

SET EMI
BACK:
SZ IAR1.2                 ; check for write cycle end
JMP BACK
CLR BP
```

## 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

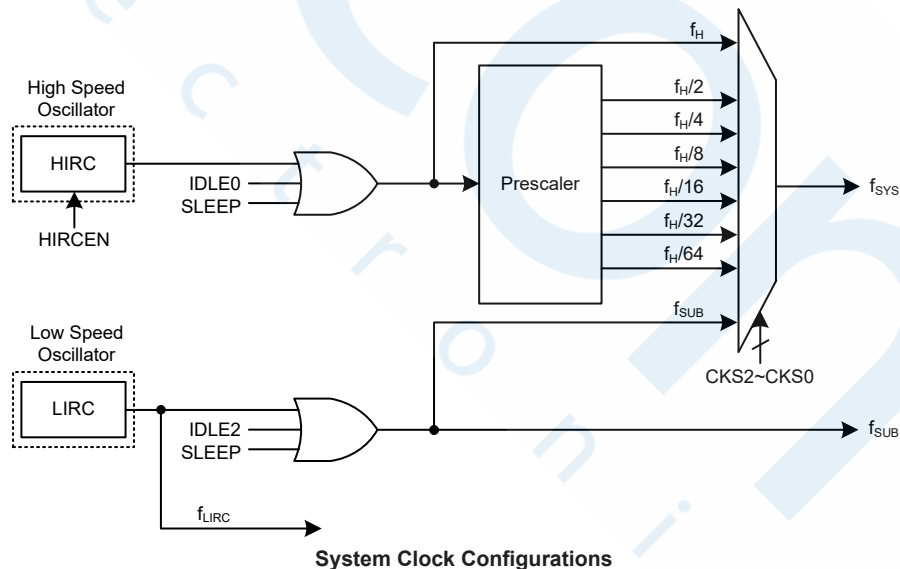
In addition to being the source of the main system clock the oscillators also provide clock sources for the Watchdog Timer and Time Base Interrupts. Two fully integrated internal oscillators, requiring no external components, are provided to form a wide range of both fast and slow system oscillators. The higher frequency oscillator provides higher performance but carry with it the disadvantage of higher power requirements, while the opposite is of course true for the lower frequency oscillator. With the capability of dynamically switching between fast and slow system clock, the device has 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, a high speed oscillator and a 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 as the system clock can be dynamically selected.



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

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

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

The internal 32kHz System Oscillator is also a fully integrated 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.

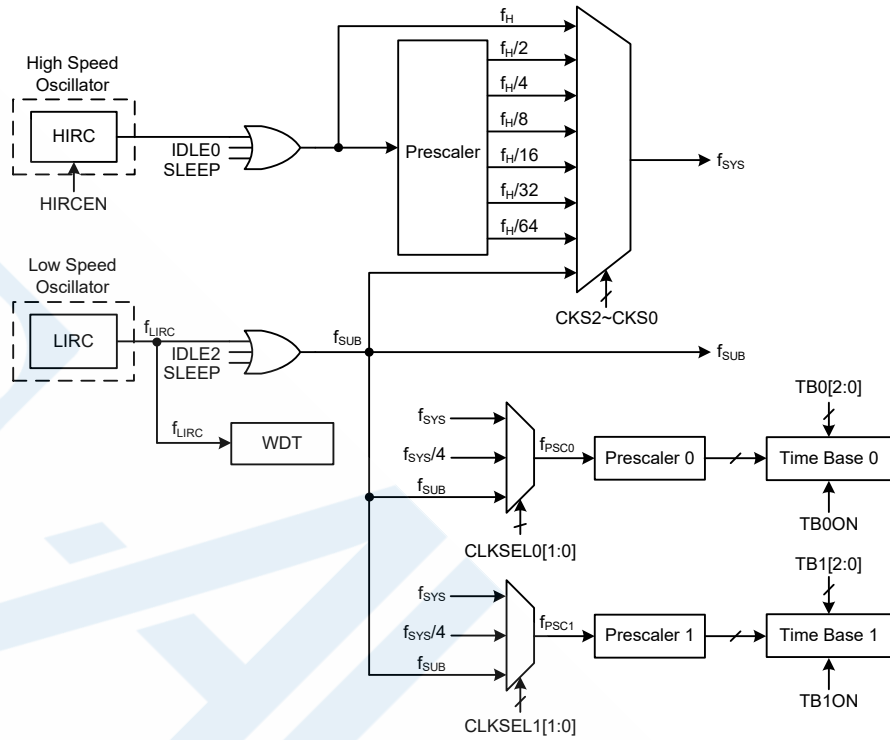
## **Operating Modes and System Clocks**

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

The main system clock, can come from a high frequency,  $f_H$ , or low frequency,  $f_{SUB}$ , source, and is selected using the CKS2~CKS0 bits in the SCC register. The high speed system clock 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  $f_H/2 \sim f_H/64$ .



**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 microcontroller, each one with its own special characteristics and which can be chosen according to the specific performance and power requirements of the application. There are two modes allowing normal operation of the microcontroller, the FAST Mode and SLOW Mode. The remaining four modes, the SLEEP, IDLE0, IDLE1 and IDLE2 Modes 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 will 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 microcontroller has all of its functions operational and where the system clock is provided by the high speed oscillator. This mode operates allowing the microcontroller to operate normally with a clock source 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 microcontroller at a divided clock ratio reduces the operating current.

### SLOW Mode

This is also a mode where the microcontroller operates normally although now with a slower speed clock source. The clock source used will be from  $f_{SUB}$ . The  $f_{SUB}$  clock is derived from the LIRC oscillator.

### SLEEP Mode

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

### IDLE0 Mode

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

### IDLE1 Mode

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

### IDLE2 Mode

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

## Control Registers

The SCC and HIRCC registers are used to control the system clock and the HIRC oscillator configurations.

Register Name	Bit							
	7	6	5	4	3	2	1	0
SCC	CKS2	CKS1	CKS0	—	—	—	FHIDEN	FSIDEN
HIRCC	—	—	—	—	—	—	HIRCF	HIRCEN

**System Operating Mode Control Register List**

• **SCC Register**

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

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

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

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

Bit 4~2     Unimplemented, read as “0”

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

0: Disable  
 1: Enable

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

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

0: Disable  
 1: Enable

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

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  $t_{SYS}$  indicates the current system clock period.

• **HIRCC Register**

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

Bit 7~2     Unimplemented, read as “0”

Bit 1        **HIRCF**: HIRC oscillator stable flag

0: HIRC unstable  
 1: HIRC stable

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

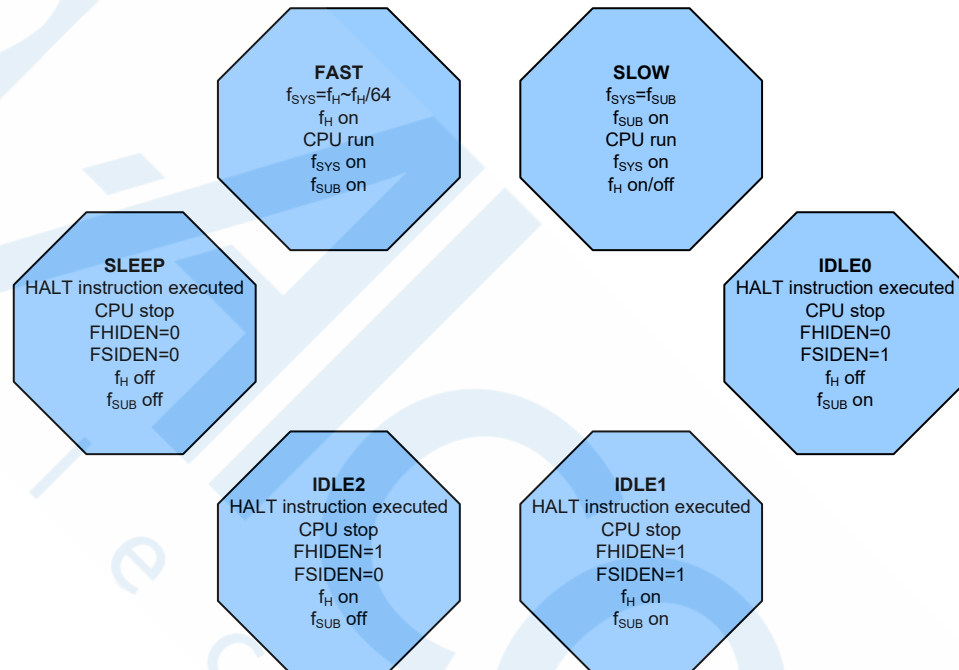
Bit 0        **HIRCEN**: HIRC oscillator enable control

0: Disable  
 1: Enable

## Operating Mode Switching

The device 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.

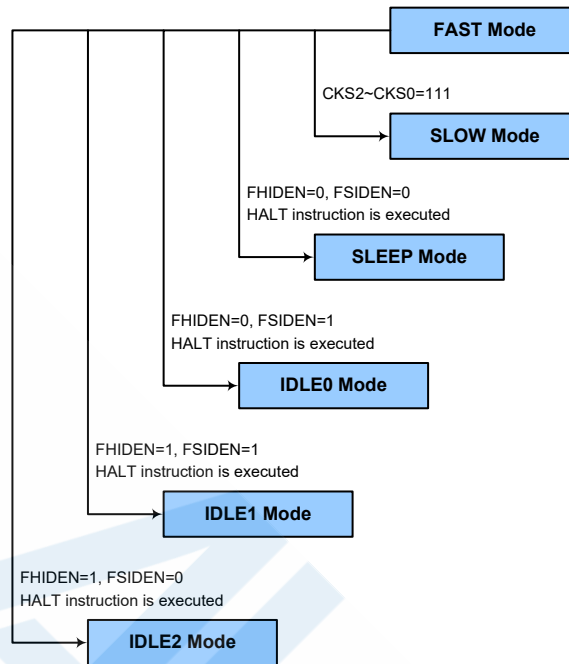
In simple terms, Mode Switching between the FAST Mode and SLOW Mode is executed using the CKS2~CKS0 bits in the SCC register while Mode Switching from the FAST/SLOW Modes to the SLEEP/IDLE Modes is executed via the HALT instruction. When a HALT instruction is executed, whether the device enters the IDLE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



### FAST Mode to SLOW Mode Switching

When running in the FAST Mode, which uses the high speed system oscillator, and therefore consumes more power, the system clock can switch to run in the SLOW Mode by 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.

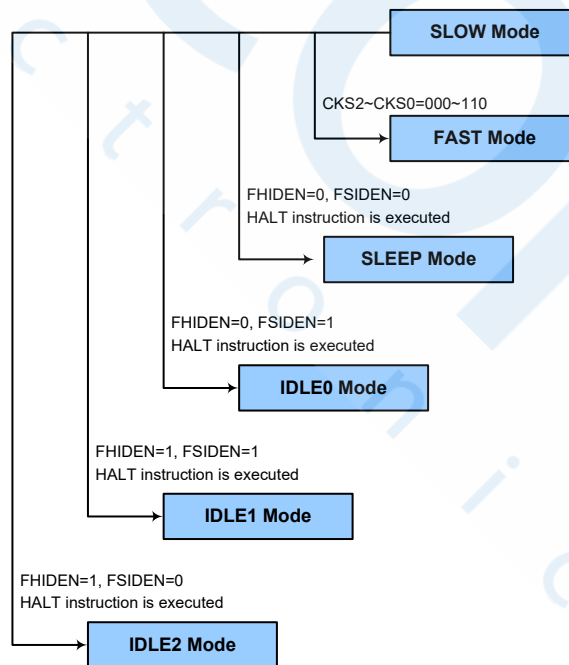
The SLOW Mode system clock is sourced from the LIRC oscillator and therefore requires this oscillator to be stable before full mode switching occurs.



### SLOW Mode to FAST Mode Switching

In SLOW mode the system clock is derived from  $f_{SUB}$ . When system clock is switched back to the FAST mode from  $f_{SUB}$ , the  $CKS2\sim CKS0$  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 SLOW mode and thus switched off, it will take some time to re-oscillate and stabilise when switching to the FAST mode from the SLOW Mode. This is monitored using the HIRCF bit in the HIRCC register. The time duration required for the high speed system oscillator stabilization is specified in the System Start Up Time Characteristics.





### Entering the SLEEP Mode

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

- The system clock will be stopped and the application program will stop at the “HALT” instruction.
- The Data Memory contents and registers will maintain their present condition.
- The I/O ports will maintain their present conditions.
- In the status register, the Power Down flag, PDF, will be set and the Watchdog time-out flag, TO, will be cleared.
- The WDT will be cleared and resume counting if the WDT function is always 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 device to enter the IDLE0 Mode and that is to execute the “HALT” instruction in the application program with the FHIDEN bit in the SCC register equal to “0” and the FSIDEN bit in the SCC register equal to “1”. When this instruction is executed under the conditions described above, the following will occur:

- The  $f_H$  clock will be stopped and the application program will stop at the “HALT” instruction, but the  $f_{SUB}$  clock will be on.
- The Data Memory contents and registers will maintain their present condition.
- The I/O ports will maintain their present conditions.
- In the status register, the Power Down flag, PDF, will be set and the Watchdog time-out flag, TO, will be cleared.
- The WDT will be cleared and resume counting if the WDT function is always 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 device to enter the IDLE1 Mode and that is to execute the “HALT” instruction in the application program with both the FHIDEN and FSIDEN bits in the SCC register equal to “1”. When this instruction is executed under the conditions described above, the following will occur:

- The  $f_H$  and  $f_{SUB}$  clocks will be on but the application program will stop at the “HALT” instruction.
- The Data Memory contents and registers will maintain their present condition.
- The I/O ports will maintain their present conditions.
- In the status register, the Power Down flag, PDF, will be set and the Watchdog time-out flag, TO, will be cleared.
- The WDT will be cleared and resume counting if the WDT function is always 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 device to enter the IDLE2 Mode and that is to execute the “HALT” instruction in the application program with the FHIDEN bit in the SCC register equal to “1” and the FSIDEN bit in the SCC register equal to “0”. When this instruction is executed under the conditions described above, the following will occur:

- The  $f_H$  clock will be on but the  $f_{SUB}$  clock will be off and the application program will stop at the “HALT” instruction.
- The Data Memory contents and registers will maintain their present condition.
- The I/O ports will maintain their present conditions.
- In the status register, the Power Down flag, PDF, will be set and the Watchdog time-out flag, TO, will be cleared.
- The WDT will be cleared and resume counting if the WDT function is always 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 Modes, there are other considerations which must also be taken into account by the circuit designer if the power consumption is to be minimised. Special attention must be made to the I/O pins on the device. All high-impedance input pins must be connected to either a fixed high or low level as any floating input pins could create internal oscillations and result in increased current consumption. This also applies to the device which has different package types, as there may be unbonded pins. These pins must either be setup as outputs or if setup as inputs must have pull-high resistors connected.

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

In the IDLE1 and IDLE2 Modes 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 device can enter the SLEEP or any IDLE Mode, where the CPU will be switched off. However, when the device is woken up again, it will take a considerable time for the original system oscillator to restart, stabilise and allow normal operation to resume.

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

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

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

If the system is woken up by a WDT overflow, a Watchdog Timer Time-out reset will be initiated and the TO flag will be set to 1. The TO flag is set high if a WDT time-out occurs, and causes a wake-up that only resets the Program Counter and Stack Pointer, the other flags remain in their original status.

Each pin on Port A can be setup using the PAWU register to permit a negative transition on the pin to wake-up the system. When a 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  $2^8$  to  $2^{15}$  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 timeout period as well as the enable/disable and reset MCU operation.

#### • WDTC Register

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

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

01010: Enable

10101: Disable

Other values: Reset MCU

When these bits are changed by the environmental noise or software setting to reset the microcontroller, the reset operation will be activated after a delay time,  $t_{SRESET}$  and the WRF bit in the RSTFC register will be set high.

Bit 2~0    **WS2~WS0**: WDT time-out period selection  
 000:  $2^8/f_{LIRC}$   
 001:  $2^9/f_{LIRC}$   
 010:  $2^{10}/f_{LIRC}$   
 011:  $2^{11}/f_{LIRC}$   
 100:  $2^{12}/f_{LIRC}$   
 101:  $2^{13}/f_{LIRC}$   
 110:  $2^{14}/f_{LIRC}$   
 111:  $2^{15}/f_{LIRC}$

These three bits determine the division ratio of the watchdog timer source clock, which in turn determines the time-out period.

• **RSTFC Register**

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

“x”: unknown

Bit 7~3    Unimplemented, read as “0”  
 Bit 2    **LVRF**: LVR function reset flag  
 Refer to the Low Voltage Reset section.  
 Bit 1    **LRF**: LVR control register software reset flag  
 Refer to the Low Voltage Reset section.  
 Bit 0    **WRF**: WDTC register software reset flag  
 0: Not occurred  
 1: Occurred

This bit is set to 1 by the WDTC register software reset and cleared to zero by the application program. Note that this bit can be cleared to zero only by the application program.

**Watchdog Timer Operation**

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

WE4~WE0 Bits	WDT Function
10101B	Disable
01010B	Enable
Any other values	Reset MCU

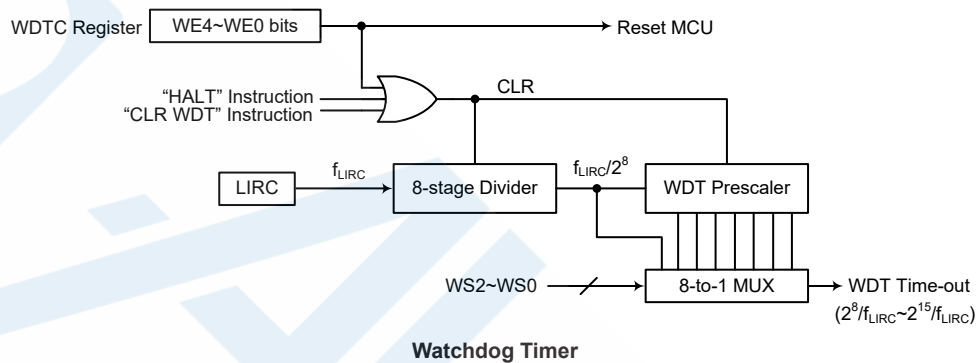
**Watchdog Timer Enable/Disable/Reset Control**

Under normal program operation, a Watchdog Timer time-out will initialise a device reset and set the status bit TO high. 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 high and only the Program Counter and

Stack Pointer will be reset. Three methods can be adopted to clear the contents of the Watchdog Timer. The first is a WDTC software reset, which means a certain value except 01010B and 10101B written into the WE4~WE0 bits, the second is using the Watchdog Timer software clear instruction and the third is via a HALT instruction.

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

The maximum time out period is when the  $2^{15}$  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 ratio.



## Reset and Initialisation

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

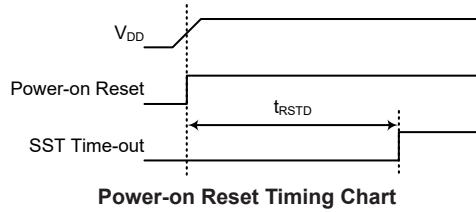
Another reset exists in the form of a Low Voltage Reset, LVR, where a full reset is implemented in situations where the power supply voltage falls below a certain threshold. Another type of reset is when the Watchdog Timer overflows and resets the microcontroller. All types of reset operations result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

### Reset Functions

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

#### Power-on Reset

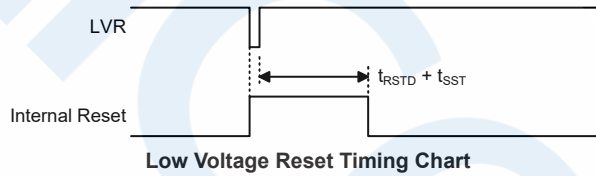
The most fundamental and unavoidable reset is the one that occurs after power is first applied to the microcontroller. As well as ensuring that the Program Memory begins execution from the first memory address, a power-on reset also ensures that certain other registers are preset to known conditions.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



**Low Voltage Reset – LVR**

The microcontroller contains a low voltage reset circuit in order to monitor the supply voltage of the device and provide an MCU reset when the value falls below a certain predefined level. If the supply voltage of the device drops to within a range of  $0.9V \sim V_{LVR}$  such as might occur when changing the battery in battery powered applications, the LVR will automatically reset the device internally and the LVRF bit in the RSTFC register will also be set to 1. For a valid LVR signal, a low supply voltage, i.e., a voltage in the range between  $0.9V \sim V_{LVR}$  must exist for a time greater than that specified by  $t_{LVR}$  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. If the LVS7~LVS0 bits are set to 01011010B, the LVR function is enabled with a fixed LVR voltage of 1.7V. If the LVS7~LVS0 bits are set to 10100101B, the LVR function is disabled. If the LVS7~LVS0 bits are changed to other values by environmental noise, the LVR will reset the device after a delay time,  $t_{SRESET}$ . When this happens, the LRF bit in the RSTFC register will be set high. After power on the register will have the value of 01011010B.

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



**• LVRC Register**

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

Bit 7~0 **LVS7~LVS0**: LVR voltage select

01011010: 1.7V

10100101: LVR disable

Other values: Generates a MCU reset – register is reset to POR value

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

Any register value, other than 01011010B and 10100101B values, will also result in the generation of an MCU reset. The reset operation will be activated after a delay time,  $t_{SRESET}$ . However, in this situation the register contents will be reset to the POR value.

• **VBGC Register**

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

Bit 7~4 Unimplemented, read as “0”

Bit 3 **VBGEN**: Bandgap Buffer Control  
 0: Disable  
 1: Enable

Note that the Bandgap circuit is enabled when the LVR function is enabled or when the VBGEN bit is set to 1.

Bit 2~0 Unimplemented, read as “0”

• **RSTFC Register**

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

“x”: unknown

Bit 7~3 Unimplemented, read as “0”

Bit 2 **LVRF**: LVR function reset flag  
 0: Not occurred  
 1: Occurred

This bit is set to 1 when an actual Low Voltage Reset situation condition occurs. This bit can only be cleared to zero by the application program.

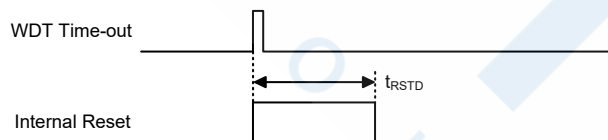
Bit 1 **LRF**: LVR control register software reset flag  
 0: Not occurred  
 1: Occurred

This bit is set to 1 by the LVRC control register contains any undefined LVR voltage register values. This in effect acts like a software-reset function. Note that this bit can only be cleared to zero by the application program.

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

**Watchdog Time-out Reset during Normal Operation**

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

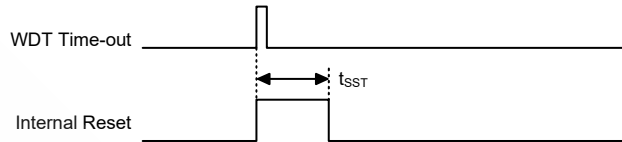


**WDT Time-out Reset during Normal Operation Timing Chart**



### Watchdog Time-out Reset during SLEEP or IDLE Mode

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



**WDT Time-out Reset during SLEEP or IDLE Mode Timing Chart**

### Reset Initial Conditions

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

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

“u” stands for unchanged

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

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

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

Register	Power On Reset	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)
IAR0	x xxx x xxx	u u u u u u u u	u u u u u u u u
MP0	x xxx x xxx	u u u u u u u u	u u u u u u u u
IAR1	x xxx x xxx	u u u u u u u u	u u u u u u u u
MP1	x xxx x xxx	u u u u u u u u	u u u u u u u u
BP	- - - - - 0	- - - - - 0	- - - - - u
ACC	x xxx x xxx	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
TBLP	x xxx x xxx	u u u u u u u u	u u u u u u u u



Register	Power On Reset	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)
TBLH	-xxx xxxx	-uuu uuuu	-uuu uuuu
TBHP	---- -xxx	---- -uuu	---- -uuu
STATUS	--00 xxxx	--1u uuuu	--11 uuuu
SCC	000- --00	000- --00	uuu- --uu
HIRCC	---- --01	---- --01	---- --uu
LVRC	0101 1010	0101 1010	uuuu uuuu
INTEG	---- --00	---- --00	---- --uu
RSTFC	---- -x00	---- -uuu	---- -uuu
INTC0	-000 0000	-000 0000	-uuu uuuu
INTC1	0000 0000	0000 0000	uuuu uuuu
INTC2	---0 ---0	---0 ---0	---u ---u
PA	1111 1111	1111 1111	uuuu uuuu
PAC	1111 1111	1111 1111	uuuu uuuu
PAPU	0000 0000	0000 0000	uuuu uuuu
PAWU	0000 0000	0000 0000	uuuu uuuu
PB	---1 1111	---1 1111	---u uuuu
PBC	---1 1111	---1 1111	---u uuuu
PBPU	---0 0000	---0 0000	---u uuuu
SLEDC	0000 0000	0000 0000	uuuu uuuu
VBGC	---- 0---	---- 0---	---- u---
WDTC	0101 0011	0101 0011	uuuu uuuu
PSC0R	---- --00	---- --00	---- --uu
TB0C	0--- -000	0--- -000	u--- -uuu
PSC1R	---- --00	---- --00	---- --uu
TB1C	0--- -000	0--- -000	u--- -uuu
EEA	---0 0000	---0 0000	---u uuuu
EED	0000 0000	0000 0000	uuuu uuuu
PAS0	0000 0000	0000 0000	uuuu uuuu
PAS1	0000 --00	0000 --00	uuuu --uu
PBS0	0000 0000	0000 0000	uuuu uuuu
PBS1	---- --00	---- --00	---- --uu
TKTMR	0000 0000	0000 0000	uuuu uuuu
TKC0	-000 0-00	-000 0-00	-uuu u-uu
TKC1	---- --11	---- --11	---- --uu
TK16DL	0000 0000	0000 0000	uuuu uuuu
TK16DH	0000 0000	0000 0000	uuuu uuuu
TKM0C0	0000 0000	0000 0000	uuuu uuuu
TKM0C1	0-00 --00	0-00 --00	u-uu --uu
TKM016DL	0000 0000	0000 0000	uuuu uuuu
TKM016DH	0000 0000	0000 0000	uuuu uuuu
TKM0ROL	0000 0000	0000 0000	uuuu uuuu
TKM0ROH	---- --00	---- --00	---- --uu
CTMC0	0000 0000	0000 0000	uuuu uuuu
CTMC1	0000 0000	0000 0000	uuuu uuuu
CTMDL	0000 0000	0000 0000	uuuu uuuu
CTMDH	---- --00	---- --00	---- --uu
CTMAL	0000 0000	0000 0000	uuuu uuuu
CTMAH	---- --00	---- --00	---- --uu
ISGENC	0--- --00	0--- --00	u--- --uu

Register	Power On Reset	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)
ISGDATA0	---0 0000	---0 0000	---u uuuu
ISGDATA1	---0 0000	---0 0000	---u uuuu
LVPUC	---- - - -0	---- - - -0	---- - - -u
OPSWA	0000 0000	0000 0000	uuuu uuuu
OPSWB	0000 0000	0000 0000	uuuu uuuu
OPSWC	---- 0000	---- 0000	---- uuuu
OPDC	---- -000	---- -000	---- -uuu
OPDACAL	0010 0000	0010 0000	uuuu uuuu
SADC0	0000 0000	0000 0000	uuuu uuuu
SADC1	0000 0000	0000 0000	uuuu uuuu
SADOH	xxxx xxxx	xxxx xxxx	uuuu uuuu (ADRF=0)
			---- - - uu (ADRF=1)
SADOL	xx - - - - -	xx - - - - -	uu - - - - - (ADRF=0)
			uuuu uuuu (ADRF=1)
ORMC	0000 0000	0000 0000	0000 0000
EEC	0 - - - 0000	0 - - - 0000	u - - - uuuu

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

## Input/Output Ports

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

The device provides bidirectional input/output lines labeled with 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	PA6	PA5	PA4	PA3	PA2	PA1	PA0
PAC	PAC7	PAC6	PAC5	PAC4	PAC3	PAC2	PAC1	PAC0
PAPU	PAPU7	PAPU6	PAPU5	PAPU4	PAPU3	PAPU2	PAPU1	PAPU0
PAWU	PAWU7	PAWU6	PAWU5	PAWU4	PAWU3	PAWU2	PAWU1	PAWU0
PB	—	—	—	PB4	PB3	PB2	PB1	PB0
PBC	—	—	—	PBC4	PBC3	PBC2	PBC1	PBC0
PBPU	—	—	—	PBPU4	PBPU3	PBPU2	PBPU1	PBPU0
LVPUC	—	—	—	—	—	—	—	LVPU

“—”: Unimplemented, read as “0”

### I/O Logic Function Register List

## Pull-high Resistors

Many product applications require pull-high resistors for their switch inputs usually requiring the use of an external resistor. To eliminate the need for these external resistors, all I/O pins, when configured as a digital input have the capability of being connected to an internal pull-high resistor. These pull-high resistors are selected using the PxPU and LVPUC registers, and are implemented using weak PMOS transistors. The PxPU register is used to determine whether the pull-high function is enabled or not while the LVPUC register is used to select the pull-high resistor value for low voltage power supply applications.

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

### • PxPU Register

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

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

0: Disable

1: Enable

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

### • LVPUC Register

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

Bit 7~1 Unimplemented, read as “0”

Bit 0 **LVPU:** Pull-high resistor selection for low voltage power supply

0: All pin pull-high resistors are 60kΩ @ 3V

1: All pin pull-high resistors are 15kΩ @ 3V

The LVPU bit is used to select the pull-high resistor value for low voltage power supply applications. Not that this bit is only available when the corresponding pin pull-high function is enabled. If the pull-high function is disabled, the LVPU bit has no effect on selecting the pull-high resistor value.

## 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

Bit 7~0      **PAWU7~PAWU0:** PA7~PA0 wake-up function control  
 0: Disable  
 1: Enable

**I/O Port Control Registers**

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

• **PxC Register**

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

**PxCn:** I/O Port x Pin type selection  
 0: Output  
 1: Input

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

**I/O Port Source Current Control**

Each pin in this device can be configured with different output source current which is selected by the corresponding source current selection bits. Users should refer to the Input/Output Characteristics section to obtain the exact value for different applications.

• **SLEDC Register**

Bit	7	6	5	4	3	2	1	0
Name	SLEDC7	SLEDC6	SLEDC5	SLEDC4	SLEDC3	SLEDC2	SLEDC1	SLEDC0
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      **SLEDC7~SLEDC6:** PA7~PA6 source current selection  
 00: Source current = Level 0 (Min.)  
 01: Source current = Level 1  
 10: Source current = Level 2  
 11: Source current = Level 3 (Max.)

- Bit 5~4     **SLEDC5~SLEDC4:** PA5~PA4 source current selection  
 00: Source current = Level 0 (Min.)  
 01: Source current = Level 1  
 10: Source current = Level 2  
 11: Source current = Level 3 (Max.)
- Bit 3~2     **SLEDC3~SLEDC2:** PA3~PA2 source current selection  
 00: Source current = Level 0 (Min.)  
 01: Source current = Level 1  
 10: Source current = Level 2  
 11: Source current = Level 3 (Max.)
- Bit 1~0     **SLEDC1~SLEDC0:** PA1~PA0 source current selection  
 00: Source current = Level 0 (Min.)  
 01: Source current = Level 1  
 10: Source current = Level 2  
 11: Source current = Level 3 (Max.)

### Pin-shared Functions

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

#### Pin-shared Function Selection Registers

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

The most important point to note is to make sure that the desired pin-shared function is properly selected and also deselected. For most pin-shared functions, to select the desired pin-shared function, the pin-shared function should first be correctly selected using the corresponding pin-shared control register. After that the corresponding peripheral functional setting should be configured and then the peripheral function can be enabled. However, a special point must be noted for some digital input pins, such as INT, CTCK, etc, which share the same pin-shared control configuration with their corresponding general purpose I/O functions when setting the relevant pin-shared control bit fields. To select these pin functions, in addition to the necessary pin-shared control and peripheral functional setup aforementioned, they 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	PAS07	PAS06	PAS05	PAS04	PAS03	PAS02	PAS01	PAS00
PAS1	PAS17	PAS16	PAS15	PAS14	—	—	PAS11	PAS10
PBS0	PBS07	PBS06	PBS05	PBS04	PBS03	PBS02	PBS01	PBS00
PBS1	—	—	—	—	—	—	PBS11	PBS10

**Pin-shared Function Selection Register List**

• **PAS0 Register**

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

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

00: PA3  
01: OPDAP  
10: PA3  
11: PA3

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

00: PA2  
01: OPDAN  
10: VBAT  
11: PA2

Note: since the VBAT pin has pin map design, it is recommended to use PA4 pin first and avoid using PA2/PA4 pins simultaneously.

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

00: PA1  
01: OPDAO  
10: AN1  
11: CTP

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

00: PA0/INT  
01: AN0  
10: CTP  
11: PA0/INT

• **PAS1 Register**

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

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

00: PA7  
01: AN3  
10: PA7  
11: PA7

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

00: PA6  
01: OPDAP  
10: AN2  
11: CTPB

Bit 3~2 Unimplemented, read as “0”

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

00: PA4  
01: VBAT  
10: PA4  
11: PA4

Note: since the VBAT pin has pin map design, it is recommended to use PA4 pin first and avoid using PA2/PA4 pins simultaneously.

• **PBS0 Register**

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

Bit 7~6 **PBS07~PBS06:** PB3 Pin-shared function selection

00: PB3  
 01: KEY2  
 10: PB3  
 11: PB3

Bit 5~4 **PBS05~PBS04:** PB2 Pin-shared function selection

00: PB2  
 01: KEY1  
 10: VREF  
 11: PB2

Bit 3~2 **PBS03~PBS02:** PB1 Pin-shared function selection

00: PB1  
 01: CTPB  
 10: PB1  
 11: PB1

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

00: PB0  
 01: CTP  
 10: PB0  
 11: PB0

• **PBS1 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	PBS11	PBS10
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

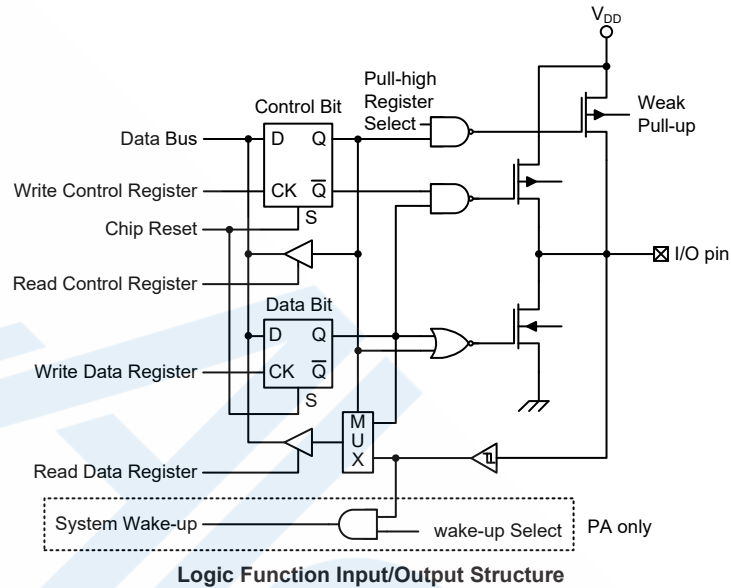
Bit 7~2 Unimplemented, read as “0”

Bit 1~0 **PBS11~PBS10:** PB4 Pin-shared function selection

00: PB4  
 01: CTP  
 10: PB4  
 11: PB4

### I/O Pin Structures

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



### Programming Considerations

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

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



## Timer Modules – TM

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

### Introduction

The device contain a Compact Type TM, with its individual reference name, CTM. The main features of CTM are summarised in the accompanying table.

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

**TM Function Summary**

### TM Operation

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

### TM Clock Source

The clock source which drives the main counter in the TM can originate from various sources. The selection of the required clock source is implemented using the CTCK2~CTCK0 bits in the CTM control registers. The clock source can be a ratio of the system clock  $f_{SYS}$  or the internal high clock  $f_H$ , the  $f_{SUB}$  clock source or the external CTCK pin. The CTCK pin clock source is used to allow an external signal to drive the TM as an external clock source or for event counting.

### TM Interrupts

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

### TM External Pins

The TM has one input pin with the label CTCK. The CTM input pin CTCK, is essentially a clock source for the TM and is selected using the CTCK2~CTCK0 bits in the CTMC0 register. This external TM input pin allows an external clock source to drive the internal TM. The TM input pin can be chosen to have either a rising or falling active edge.

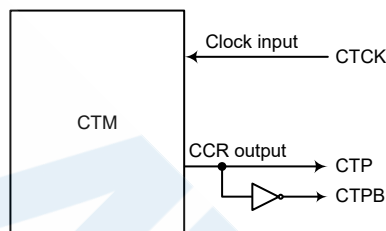
The TM has two output pins, CTP and CTPB. The CTPB pin outputs the inverted signal of the CTP. When the TM is in the Compare Match Output Mode, these pins can be controlled by the TM to

switch to a high or low level or to toggle when a compare match situation occurs. The external CTP and CTPB output pins are also the pins where the TM generates the PWM output waveform.

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

CTM	
Input	Output
CTCK	CTP, CTPB

TM External Pins

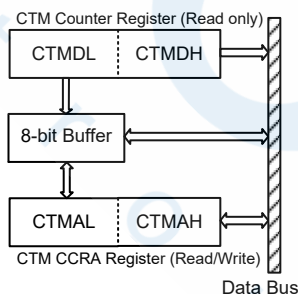


CTM Function Pin Block Diagram

### Programming Considerations

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

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



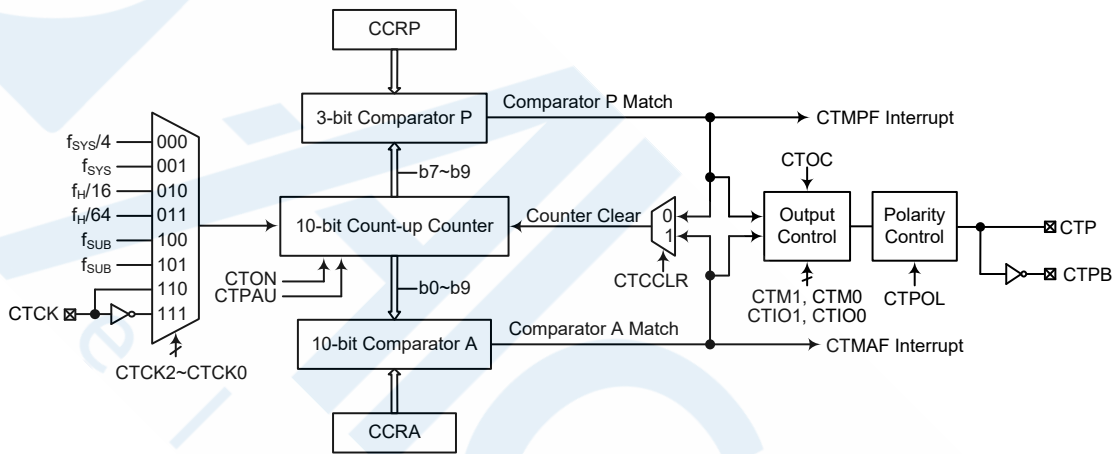
The following steps show the read and write procedures:

- Writing Data to CCRA
  - ♦ Step 1. Write data to Low Byte CTMAL
    - Note that here data is only written to the 8-bit buffer.
  - ♦ Step 2. Write data to High Byte CTMAH
    - Here data is written directly to the high byte registers and simultaneously data is latched from the 8-bit buffer to the Low Byte registers.

- Reading Data from the Counter Registers and CCRA
  - ♦ Step 1. Read data from the High Byte CTMDH, CTMAH
    - Here data is read directly from the High Byte registers and simultaneously data is latched from the Low Byte register into the 8-bit buffer.
  - ♦ Step 2. Read data from the Low Byte CTMDL, CTMAL
    - This step reads data from the 8-bit buffer.

## Compact Type TM – CTM

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



Note: 1. As the CTM external pins are pin-shared with other functions, the relevant pin-shared control bits should be properly configured before using these pins, except the CTCK pin which is selected only when the CTM counter clock is set to come from the CTCK pin by configuring the related CTM control register.  
 2. The CTPB is the inverted signal of the CTP.

**10-bit Compact Type TM Block Diagram**

### Compact Type TM Operation

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

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

### Compact Type TM Register Description

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

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

**10-bit Compact Type TM Register List**

#### • CTMC0 Register

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

**Bit 7**      **CTPAU:** CTM counter pause control  
0: Run  
1: Pause

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

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

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

**Bit 3**      **CTON:** CTM counter on/off control  
0: Off  
1: On

This bit controls the overall on/off function of the CTM. Setting the bit high enables the counter to run, clearing the bit to 0 disables the CTM. Clearing this bit to zero will stop the counter from counting and turn off the CTM which will reduce its power consumption. When the bit changes state from low to high the internal counter value will be reset to zero, however when the bit changes from high to low, the internal counter will retain its residual value until the bit returns high again.

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

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

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

• **CTMC1 Register**

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

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

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

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

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

In the Compare Match Output Mode, the CTIO1 and CTIO0 bits determine how the CTM output pin changes state when a compare match occurs from the Comparator A. The CTM output pin can be setup to switch high, switch low or to toggle its present state when a compare match occurs from the Comparator A. When the bits are both zero, then no change will take place on the output. The initial value of the CTM output pin should be setup using the CTOC bit in the CTMC1 register. Note that the output level requested by the CTIO1 and CTIO0 bits must be different from the initial value setup using the CTOC bit otherwise no change will occur on the CTM output pin when a compare match occurs. After the CTM output pin changes state it can be reset to its initial level by changing the level of the CTON bit from low to high.

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

Bit 3

**CTOC:** CTM CTP output control bit

Compare Match Output Mode

0: Initial low

1: Initial high

PWM Output Mode

0: Active low

1: Active high

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

Bit 2

**CTPOL:** CTM CTP output polarity control

0: Non-invert

1: Invert

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

Bit 1

**CTDPX:** CTM PWM period/duty control

0: CCRP - period; CCRA – duty

1: CCRP - duty; CCRA – period

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

Bit 0

**CTCCLR:** CTM counter clear condition selection

0: CTM Comparatror P match

1: CTM Comparatror A match

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

• **CTMDL Register**

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

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

• **CTMDH Register**

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

Bit 7~2      Unimplemented, read as “0”  
 Bit 1~0      **D9~D8**: CTM Counter High Byte Register bit 1 ~ bit 0  
 CTM 10-bit Counter bit 9 ~ bit 8

• **CTMAL Register**

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

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

• **CTMAH Register**

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

Bit 7~2      Unimplemented, read as “0”  
 Bit 1~0      **D9~D8**: CTM CCRA High Byte Register bit 1 ~ bit 0  
 CTM 10-bit CCRA bit 9 ~ bit 8



## Compact Type TM Operating Modes

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

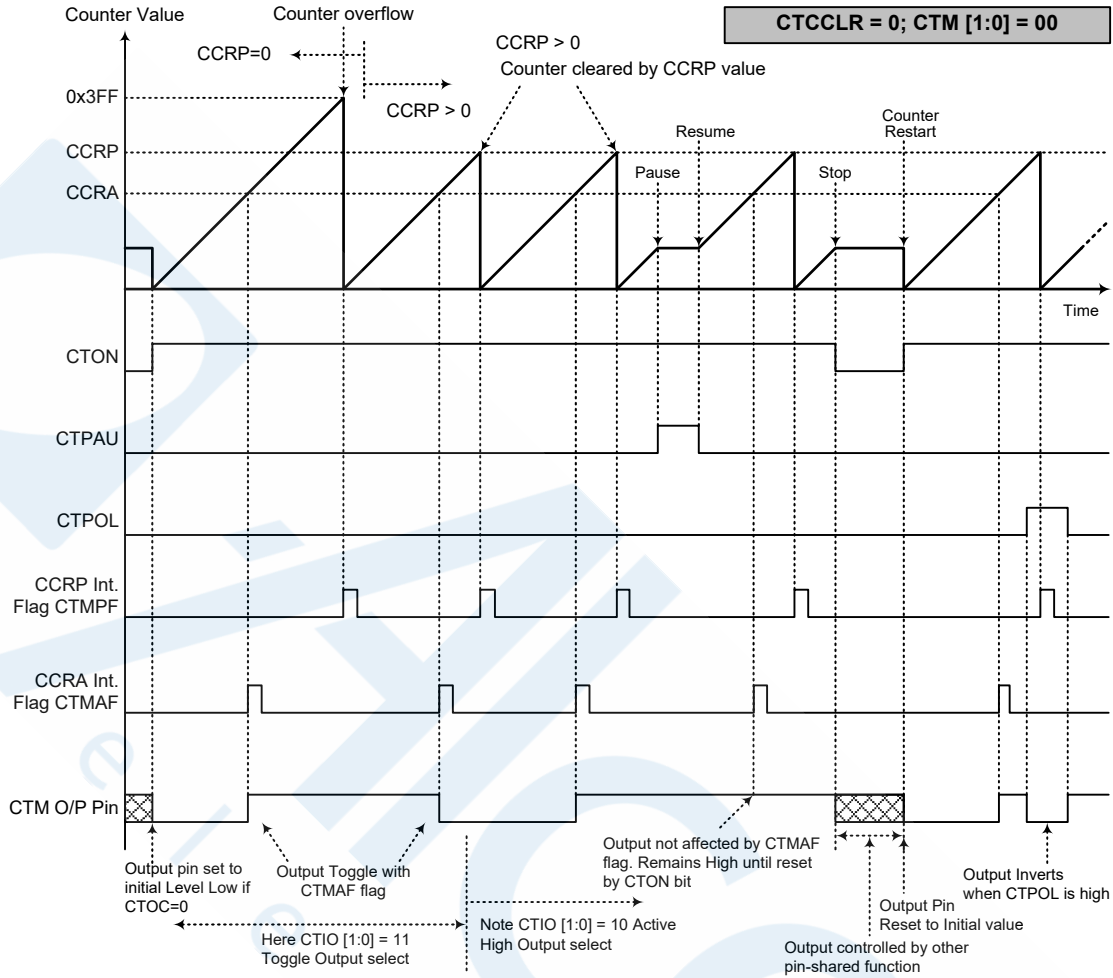
### Compare Match Output Mode

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

If the CTCCLR bit in the CTMC1 register is high, then the counter will be cleared when a compare match occurs from Comparator A. However, here only the CTMAF interrupt request flag will be generated even if the value of the CCRP bits is less than that of the CCRA registers. Therefore, when CTCCLR is high no CTMPF interrupt request flag will be generated. If the CCRA bits are all zero, the counter will overflow when it reaches its maximum 10-bit, 3FF Hex, value, however here the CTMAF interrupt request flag will not be generated.

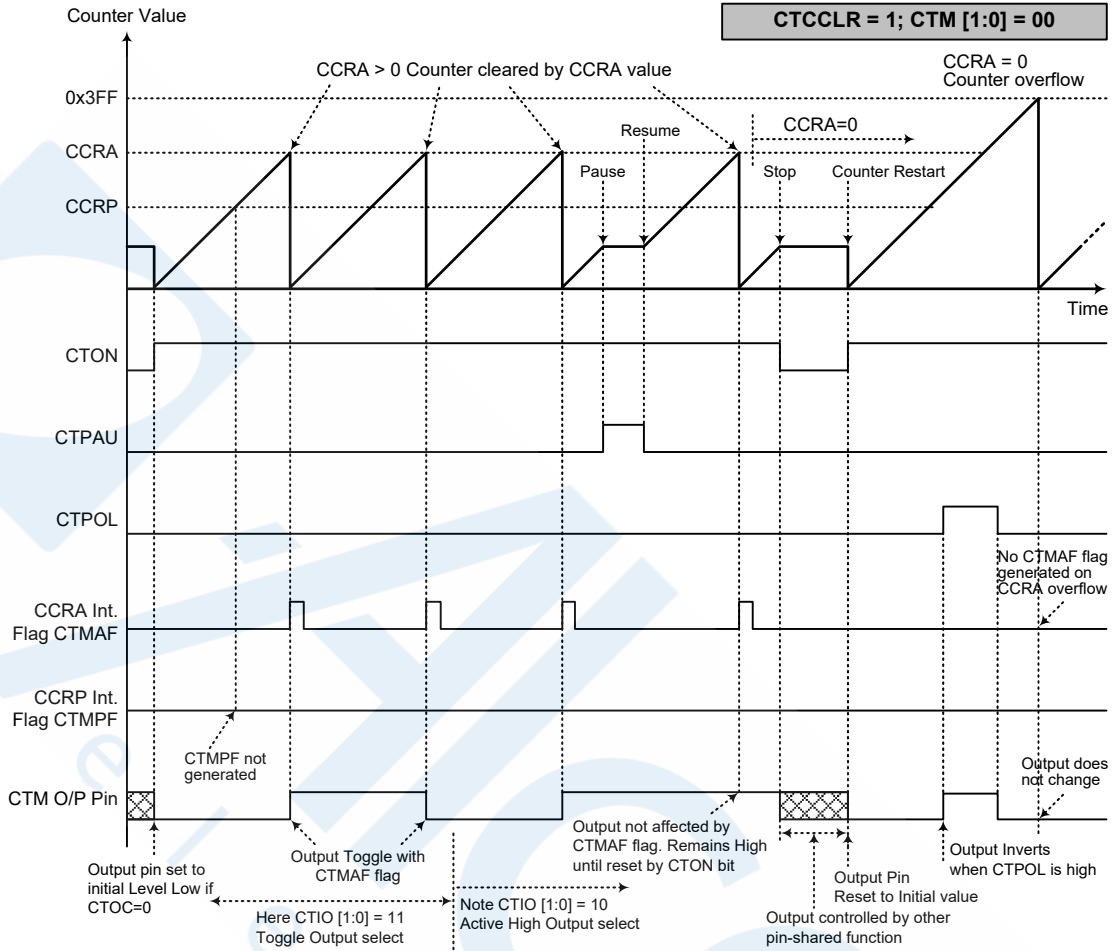
As the name of the mode suggests, after a comparison is made, the CTM output pin will change state. The CTM output pin condition however only changes state when a CTMAF interrupt request flag is generated after a compare match occurs from Comparator A. The CTMPF interrupt request flag, generated from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





**Compare Match Output Mode – CTCCLR=0**

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



**Compare Match Output Mode – CTCCLR=1**

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

**Timer/Counter Mode**

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

**PWM Output Mode**

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

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

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

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

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

If  $f_{SYS}=8\text{MHz}$ , CTM clock source is  $f_{SYS}/4$ , CCRP=2, CCRA=128,

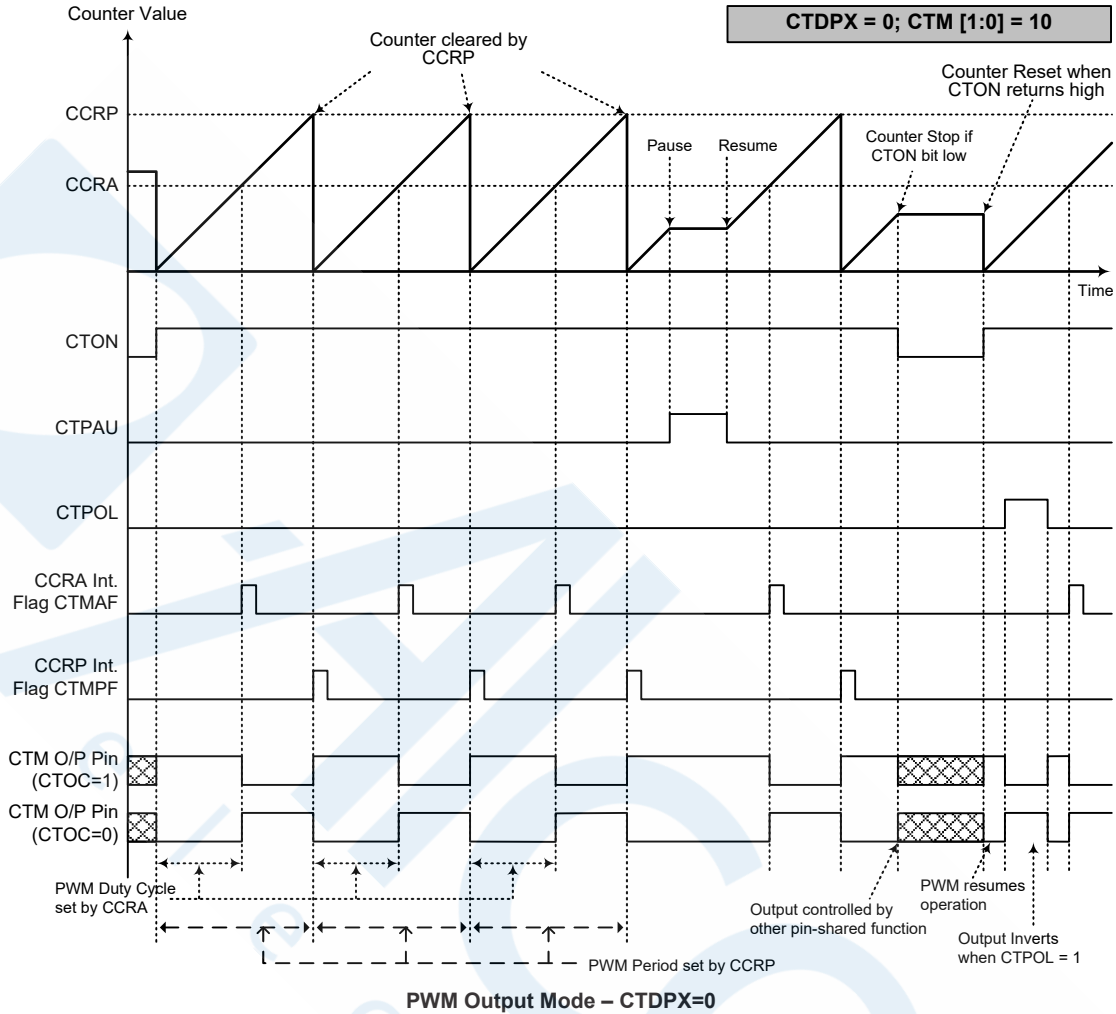
The CTM PWM output frequency= $(f_{SYS}/4)/(2 \times 128)=f_{SYS}/1024=8\text{kHz}$ , duty= $128/(2 \times 128)=50\%$ .

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

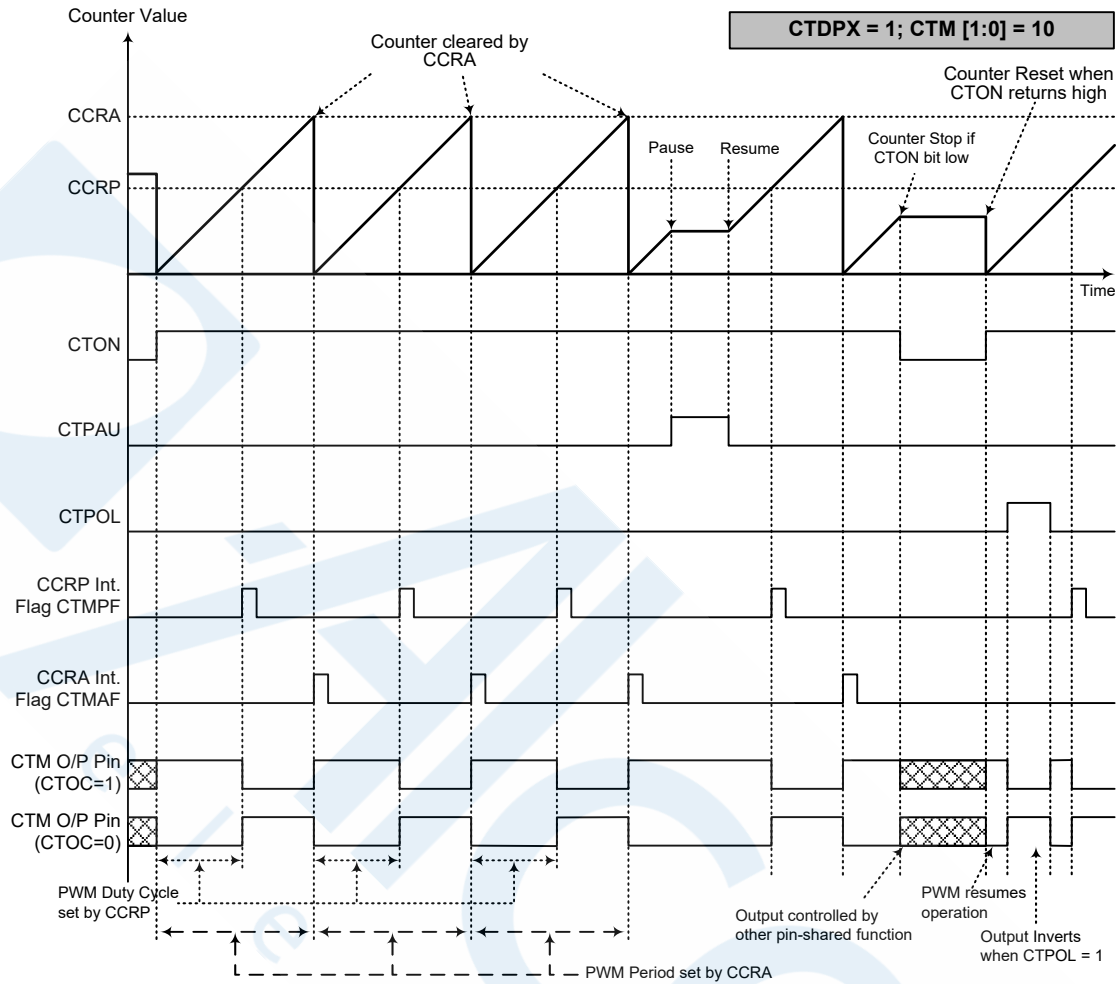
• **10-bit CTM, PWM Output Mode, Edge-aligned Mode, CTDPX=1**

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

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



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

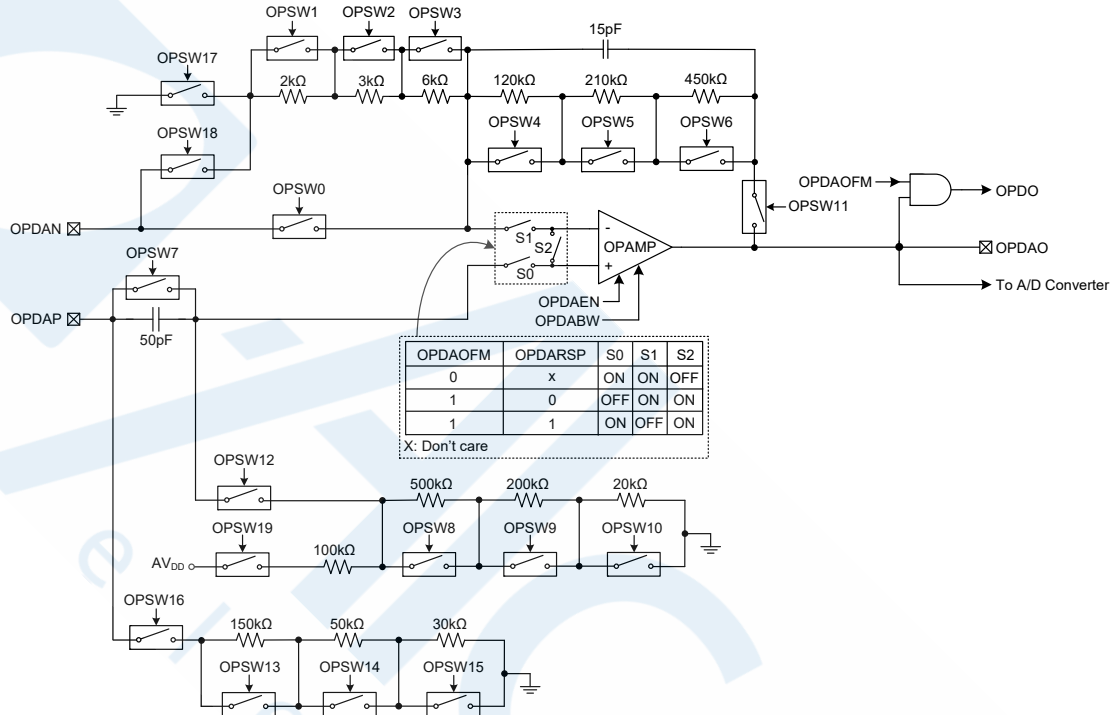


**PWM Output Mode – CTD PX=1**

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

## Proximity Sensing Circuit

The device includes a proximity sensing circuit which has an operational amplifier. The operational amplifier circuit can amplify a small signal with a gain ranging from 1 to 391. Users can choose different combinations according to different applications, no matter inverting or non-inverting amplification. And finally, the amplified analog signal will be converted to digital signal.



Proximity Sensing Circuit Block Diagram

## Proximity Sensing Circuit Operation

The source voltage is input from OPDAN and/or OPDAP. The operational amplifier chooses different amplifier modes through the switches OPSW0~OPSW19. The OPAMP is consist of a PGA function, the PGA gain can be positive or negative determined by the input voltage connecting to the positive input or negative input of PGA. The OPAMP's gain can be 1~391 configured by OPSW1~OPSW6.

The input signal which is amplified by OPAMP can be directly output on the OPDAO pin, and also can be internally connected to the A/D converter selected by setting the relevant register for reading the amplified input voltage.

## Proximity Sensing Circuit Registers

The overall operation of the Proximity Sensing Circuit is controlled using a series of registers. The OPSWA~OPSWC registers are used to control the analog switches. The OPDC register is used to control the operational amplifier, as well as the operational amplifiers' low current/high bandwidth selection. The OPDACAL register is used to control the operational amplifier input offset voltage calibration function.

Register Name	Bit							
	7	6	5	4	3	2	1	0
OPSWA	OPSW7	OPSW6	OPSW5	OPSW4	OPSW3	OPSW2	OPSW1	OPSW0
OPSWB	OPSW15	OPSW14	OPSW13	OPSW12	OPSW11	OPSW10	OPSW9	OPSW8
OPSWC	—	—	—	—	OPSW19	OPSW18	OPSW17	OPSW16
OPDC	—	—	—	—	—	OPDAEN	OPDO	OPDABW
OPDACAL	OPDAOFM	OPDARSP	OPDAOFF5	OPDAOFF4	OPDAOFF3	OPDAOFF2	OPDAOFF1	OPDAOFF0

**Proximity Sensing Circuit Register List**

### • OPSWA Register

Bit	7	6	5	4	3	2	1	0
Name	OPSW7	OPSW6	OPSW5	OPSW4	OPSW3	OPSW2	OPSW1	OPSW0
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     **OPSW7:** OPSW7 switch on/off control  
           0: Off  
           1: On
- Bit 6     **OPSW6:** OPSW6 switch on/off control  
           0: Off  
           1: On
- Bit 5     **OPSW5:** OPSW5 switch on/off control  
           0: Off  
           1: On
- Bit 4     **OPSW4:** OPSW4 switch on/off control  
           0: Off  
           1: On
- Bit 3     **OPSW3:** OPSW3 switch on/off control  
           0: Off  
           1: On
- Bit 2     **OPSW2:** OPSW2 switch on/off control  
           0: Off  
           1: On
- Bit 1     **OPSW1:** OPSW1 switch on/off control  
           0: Off  
           1: On
- Bit 0     **OPSW0:** OPSW0 switch on/off control  
           0: Off  
           1: On

• **OPSWB Register**

Bit	7	6	5	4	3	2	1	0
Name	OPSW15	OPSW14	OPSW13	OPSW12	OPSW11	OPSW10	OPSW9	OPSW8
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      **OPSW15:** OPSW15 switch on/off control  
0: Off  
1: On
- Bit 6      **OPSW14:** OPSW14 switch on/off control  
0: Off  
1: On
- Bit 5      **OPSW13:** OPSW13 switch on/off control  
0: Off  
1: On
- Bit 4      **OPSW12:** OPSW12 switch on/off control  
0: Off  
1: On
- Bit 3      **OPSW11:** OPSW11 switch on/off control  
0: Off  
1: On
- Bit 2      **OPSW10:** OPSW10 switch on/off control  
0: Off  
1: On
- Bit 1      **OPSW9:** OPSW9 switch on/off control  
0: Off  
1: On
- Bit 0      **OPSW8:** OPSW8 switch on/off control  
0: Off  
1: On

• **OPSWC Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	OPSW19	OPSW18	OPSW17	OPSW16
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0

- Bit 7~4    Unimplemented, read as “0”
- Bit 3      **OPSW19:** OPSW19 switch on/off control  
0: Off  
1: On
- Bit 2      **OPSW18:** OPSW18 switch on/off control  
0: Off  
1: On
- Bit 1      **OPSW17:** OPSW17 switch on/off control  
0: Off  
1: On
- Bit 0      **OPSW16:** OPSW16 switch on/off control  
0: Off  
1: On



• **OPDC Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	OPDAEN	OPDO	OPDABW
R/W	—	—	—	—	—	R/W	R	R/W
POR	—	—	—	—	—	0	0	0

- Bit 7~3 Unimplemented, read as “0”
- Bit 2 **OPDAEN**: OPD OPAMP enable or disable control  
 0: Disable – OPDAO high impedance  
 1: Enable
- Bit 1 **OPDO**: OPAMP output under offset calibration
- Bit 0 **OPDABW**: OPD OPAMP low current/high bandwidth selection  
 0: Low current  
 1: High bandwidth

• **OPDACAL Register**

Bit	7	6	5	4	3	2	1	0
Name	OPDAOFM	OPDARSP	OPDAOF5	OPDAOF4	OPDAOF3	OPDAOF2	OPDAOF1	OPDAOF0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	1	0	0	0	0	0

- Bit 7 **OPDAOFM**: OPD OPAMP normal operation or input offset voltage calibration mode selection  
 0: Normal operation  
 1: Offset calibration mode
- Bit 6 **OPDARSP**: OPD OPAMP input offset voltage calibration reference selection  
 0: Select inverting input as the reference voltage input  
 1: Select non-inverting input as the reference voltage input
- Bit 5~0 **OPDAOF5~OPDAOF0**: OPD OPAMP input offset voltage calibration control  
 This 6-bit field is used to perform the OPD OPAMP input offset calibration operation and the value for the OPD OPAMP input offset calibration can be restored into this bit field. More detailed information is described in the “Operational Amplifier Input Offset Calibration” section.

**Offset Calibration Procedure**

To operate in the input offset calibration mode for the Operational Amplifier, the OPDAOFM bit should first be set to “1” followed by the reference input selection by configuring the OPDARSP bit. Note that as the Operational Amplifier inputs are pin-shared with I/O or other functions, before the calibration, they should be configured as the OPD operational amplifier input pin function first by correctly setting the related pin-shared function register.

**Operational Amplifier Input Offset Calibration**

- Step1. Set OPDAOFM=1 and OPDARSP=1, the Operational Amplifier is now under offset calibration mode. To make sure  $V_{OS}$  as minimal as possible after calibration, the input reference voltage in calibration should be the same as input DC operating voltage in normal mode operation.
- Step2. Set OPDAOF[5:0]=000000 and then read OPDO bit.
- Step3. Increase the OPDAOF[5:0] value by 1 and then read the OPDO bit.  
 If the OPDO bit state has not changed, then repeat Step 3 until the OPDO bit state has changed.  
 If the OPDO bit state has changed, record the OPDAOF[5:0] value as  $V_{OS1}$  and then go to Step 4.

Step4. Set OPDAOF[5:0]=111111 then read OPDO bit.

Step5. Decrease the OPDAOF[5:0] value by 1 and then read the OPDO bit.

If the OPDO bit state has not changed, then repeat Step 5 until the OPDO bit state has changed.

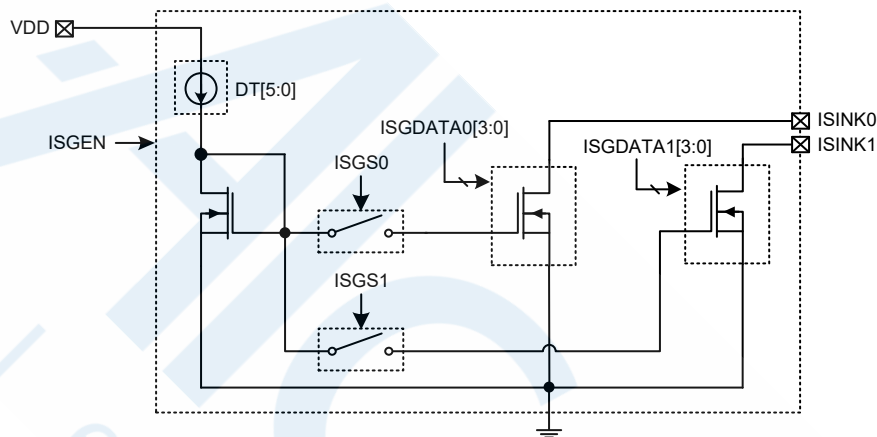
If the OPDO bit state has changed, record the OPDAOF[5:0] value as  $V_{OS2}$  and then go to Step 6.

Step6. Restore the Operational Amplifier input offset calibration value  $V_{OS}$  into the OPDAOF[5:0] bit field. The offset Calibration procedure is now finished.

$$V_{OS} = (V_{OS1} + V_{OS2}) / 2. \text{ If } (V_{OS1} + V_{OS2}) / 2 \text{ is not integral, discard the decimal.}$$

## Sink Current Generator

The sink current source generator could provide constant current no matter what  $V_{ISINK}$  voltage is from 0.7V to 4.5V. The constant current value is controlled by the ISGDATA0/ISGDATA1 register, and the sink current range is 2mA~160mA.



Note: For the 8-pin SOP/16-pin NSOP-A package where ISINK0 and ISINK1 are bounded to the same pin location, or the 16-pin NSOP-B package where these two pins may be externally bounded together by users, when ISGS0/ISGS1=0, the corresponding ISGDATA0/ISGDATA1 register must be cleared to zero to avoid current leakage problem.

Sink Current Generator Block Diagram

## Sink Current Generator Registers

There is a series of registers controlling the overall operation of the Sink Current Generator function.

Register Name	Bit							
	7	6	5	4	3	2	1	0
ISGENC	ISGEN	—	—	—	—	—	ISGS1	ISGS0
ISGDATA0	—	—	—	ISST0	D3	D2	D1	D0
ISGDATA1	—	—	—	ISST1	D3	D2	D1	D0

Sink Current Generator Register List

• **ISGENC Register**

Bit	7	6	5	4	3	2	1	0
Name	ISGEN	—	—	—	—	—	ISGS1	ISGS0
R/W	R/W	—	—	—	—	—	R/W	R/W
POR	0	—	—	—	—	—	0	0

- Bit 7      **ISGEN**: Sink current generator enable control  
 0: Disable  
 1: Enable  
 When the ISGEN bit is cleared to zero to disable the sink current generator, the ISINK0 and ISINK1 pin statuses are  $V_{ISINK0/1} = \text{floating}$ ,  $I_{ISINK0/1} = 0$ .
- Bit 6~2      Unimplemented, read as “0”
- Bit 1      **ISGS1**: ISINK1 switch on/off control  
 0: Off  
 1: On
- Bit 0      **ISGS0**: ISINK0 switch on/off control  
 0: Off  
 1: On

• **ISGDATA0 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	ISST0	D3	D2	D1	D0
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      **ISST0**: Switch first stage or second stage current control  
 0: First stage  
 1: Second stage
- Bit 3~0      **D3~D0**: Sink current control for the ISINK0 pin  
 First stage current value (mA) =  $2 + 2 \times D[3:0]$ , 2mA/step  
 Second stage current value (mA) =  $32 + 8 + 8 \times D[3:0]$ , 8mA/step  
 Maximum current (mA) =  $40\text{mA} + 120\text{mA} = 160\text{mA}$

• **ISGDATA1 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	ISST1	D3	D2	D1	D0
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      **ISST1**: Switch first stage or second stage current control  
 0: First stage  
 1: Second stage
- Bit 3~0      **D3~D0**: Sink current control for the ISINK1 pin  
 First stage current value (mA) =  $2 + 2 \times D[3:0]$ , 2mA/step  
 Second stage current value (mA) =  $32 + 8 + 8 \times D[3:0]$ , 8mA/step  
 Maximum current (mA) =  $40\text{mA} + 120\text{mA} = 160\text{mA}$

## Analog to Digital Converter

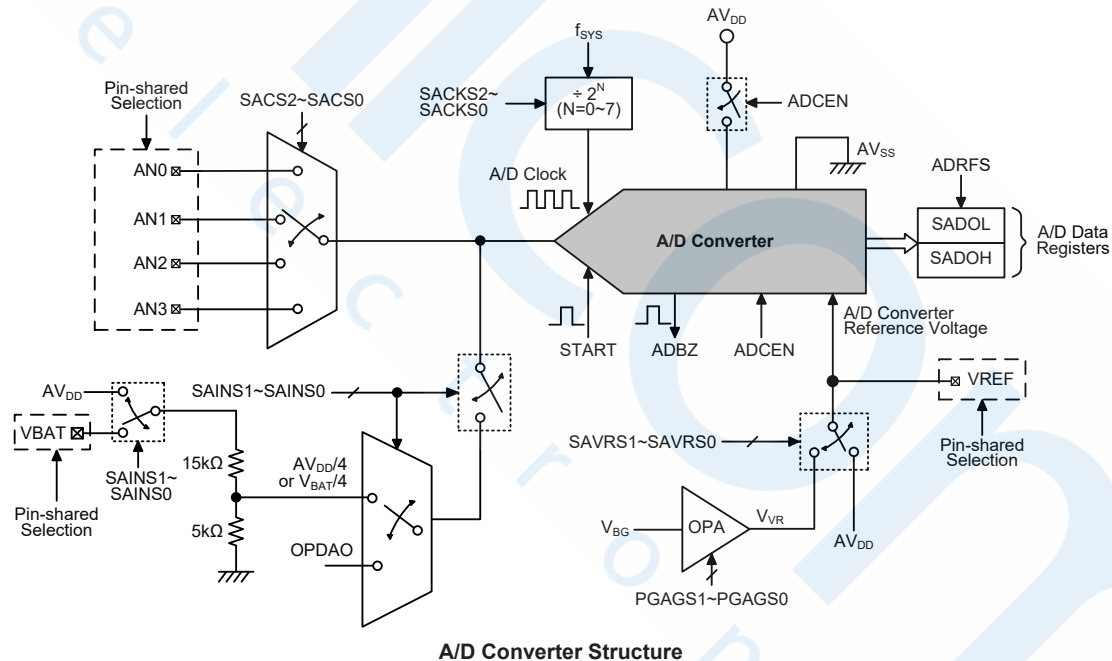
The need to interface to real world analog signals is a common requirement for many electronic systems. However, to properly process these signals by a microcontroller, they must first be converted into digital signals by A/D converters. By integrating the A/D conversion electronic circuitry into the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

### A/D Converter Overview

The device contains a multi-channel 10-bit analog to digital converter which can directly interface to external analog signals, such as that from sensors or other control signals and convert these signals directly into a 10-bit digital value. It also can convert the internal signals, such as the operational amplifier outputs, into a 10-bit digital value. The external or internal analog signal to be converted is determined by the SAINS1~SAINS0 bits together with the SACS2~SACS0 bits. More detailed information about the A/D input signal is described in the “A/D Converter Control Registers” and “A/D Converter Input Signals” sections respectively.

External Input Channels	Internal Signals	Channel Select Bits
4: AN0~AN3	3: AV <sub>DD</sub> /4, V <sub>BAT</sub> /4, OPDAO	SAINS1~SAINS0, SACS2~SACS0

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



**A/D Converter Structure**

### A/D Converter Register Description

Overall operation of the A/D converter is controlled using several registers. A read only register pair exists to store the A/D converter data 10-bit value. The remaining two registers are control registers which setup the operating and control function of the A/D converter.

Register Name	Bit							
	7	6	5	4	3	2	1	0
SADOL (ADRFS=0)	D1	D0	—	—	—	—	—	—
SADOL (ADRFS=1)	D7	D6	D5	D4	D3	D2	D1	D0
SADOH (ADRFS=0)	D9	D8	D7	D6	D5	D4	D3	D2
SADOH (ADRFS=1)	—	—	—	—	—	—	D9	D8
SADC0	START	ADBZ	ADCEN	ADRFS	PGAGS0	SACS2	SACS1	SACS0
SADC1	SAINS1	SAINS0	PGAGS1	SAVRS1	SAVRS0	SACKS2	SACKS1	SACKS0

**A/D Converter Register List**

### A/D Converter Data Registers – SADOL, SADOH

As the device contains an internal 10-bit A/D converter, it requires two data registers to store the converted value. These are a high byte register, known as SADOH, and a low byte register, known as SADOL. After the conversion process takes place, these registers can be directly read by the microcontroller to obtain the digitised conversion value. As only 10 bits of the 16-bit register space is utilised, the format in which the data is stored is controlled by the ADRFS bit in the SADC0 register as shown in the accompanying table. D0~D9 are the A/D conversion result data bits. Any unused bits will be read as zero. Note that A/D Converter data register contents will be unchanged if the A/D converter is disabled.

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

**A/D Data Registers**

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

To control the function and operation of the A/D converter, two control registers known as SADC0 and SADC1 are provided. These 8-bit registers define functions such as the selection of which analog channel is connected to the internal A/D converter, the digitised data format, the A/D clock source as well as controlling the start function and monitoring the A/D converter busy status, etc. As the device contains only one actual analog to digital converter hardware circuit, each of the external or internal analog signal inputs must be routed to the converter. The SACS2~SACS0 bits in the SADC0 register are used to determine which external channel input is selected to be converted. The SAINS1~SAINS0 bits in the SADC1 register are used to determine that the analog signal to be converted comes from the internal analog signal or external analog channel input.

The relevant pin-shared function selection bits determine which pins on I/O Ports are used as analog inputs for the A/D converter input and which pins are not to be used as the A/D converter input. When the pin is selected to be an A/D input, its original function whether it is an I/O or other pin-shared function will be removed. In addition, any internal pull-high resistor connected to the pin will be automatically removed if the pin is selected to be an A/D converter input.

• **SADC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	START	ADBZ	ADCEN	ADRFS	PGAGS0	SACS2	SACS1	SACS0
R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

- Bit 7**     **START:** Start the A/D conversion  
0→1→0: Start  
This bit is used to initiate an A/D conversion process. The bit is normally low but if set high and then cleared low again, the A/D converter will initiate a conversion process.
- Bit 6**     **ADBZ:** A/D converter busy flag  
0: No A/D conversion is in progress  
1: A/D conversion is in progress  
This read only flag is used to indicate whether the A/D conversion is in progress or not. When the START bit is set from low to high and then to low again, the ADBZ flag will be set to 1 to indicate that the A/D conversion is initiated. The ADBZ flag will be cleared to 0 after the A/D conversion is complete.
- Bit 5**     **ADCEN:** A/D converter function enable control  
0: Disable  
1: Enable  
This bit controls the A/D internal function. This bit should be set to one to enable the A/D converter. If the bit is set low, then the A/D converter will be switched off reducing the device power consumption. When the A/D converter function is disabled, the contents of the A/D data register pair known as SADOH and SADOL will be unchanged.
- Bit 4**     **ADRFS:** A/D converter data format selection  
0: A/D converter data format → SADOH = D[9:2]; SADOL = D[1:0]  
1: A/D converter data format → SADOH = D[9:8]; SADOL = D[7:0]  
This bit controls the format of the 10-bit converted A/D value in the two A/D data registers. Details are provided in the A/D data register section.
- Bit 3**     **PGAGS0:** PGA gain selection bit 0  
PGAGS[1:0]=  
00:  $V_{VR}=1.6V$   
01:  $V_{VR}=3.0V$   
10:  $V_{VR}=4.0V$   
11:  $V_{VR}=1.6V$   
Note: The PGAGS1 bit is in the SADC1 register.
- Bit 2~0**   **SACS2~SACS0:** A/D converter external analog channel input selection  
000: AN0  
001: AN1  
010: AN2  
011: AN3  
100~111: Non-existent channel, the input will be floating

• **SADC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	SAINS1	SAINS0	PGAGS1	SAVRS1	SAVRS0	SACKS2	SACKS1	SACKS0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

- Bit 7~6 **SAINS1~SAINS0**: A/D converter input signal selection  
 00: External input – External analog channel input ANn  
 01: Internal input – Proximity Sensing Circuit OPAMP output, OPDAO  
 10: Internal input – A/D converter external input voltage,  $V_{BAT}/4$   
 11: Internal input – Internal A/D converter power supply voltage,  $AV_{DD}/4$

Care must be taken if the SAINS1~SAINS0 bits are set to “01”~“11” to select the internal analog signal to be converted. When the internal analog signal is selected to be converted, the external input pin must never be selected as the A/D input signal by properly setting the SACKS2~SACKS0 bits with a value from “100” to “111”. Otherwise, the external channel input will be connected together with the internal analog signal. This will result in unpredictable situations such as an irreversible damage.

- Bit 5 **PGAGS1**: PGA gain selection bit 1

Details are provided in the SADC0 register.

- Bit 4~3 **SAVRS1~SAVRS0**: A/D converter reference voltage selection

- 00: From external VREF pin  
 01: From internal A/D converter power,  $AV_{DD}$   
 10: From internal A/D converter OPA output voltage,  $V_{VR}$   
 11: From external VREF pin

These bits are used to select the A/D converter reference voltage. Care must be taken if the SAVRS1~SAVRS0 bits are set to “01” or “10” to select the internal A/D converter power or OPA output as the reference voltage source. When the internal A/D converter power or OPA output is selected as the reference voltage, the VREF pin cannot be configured as the reference voltage input by properly configuring the corresponding pin-shared function control bits. Otherwise, the external input voltage on VREF pin will be connected to the internal A/D converter power.

- Bit 2~0 **SACKS2~SACKS0**: A/D conversion clock source selection

- 000:  $f_{SYS}$   
 001:  $f_{SYS}/2$   
 010:  $f_{SYS}/4$   
 011:  $f_{SYS}/8$   
 100:  $f_{SYS}/16$   
 101:  $f_{SYS}/32$   
 110:  $f_{SYS}/64$   
 111:  $f_{SYS}/128$

These three bits are used to select the clock source for the A/D converter.

• **VBGC Register**

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

- Bit 7~4 Unimplemented, read as “0”

- Bit 3 **VBGEN**:  $V_{BG}$  Bandgap reference control

- 0: Disable  
 1: Enable

Note that the Bandgap circuit is enabled when the LVR function is enabled or when the VBGEN bit is set high.

- Bit 2~0 Unimplemented, read as “0”



### A/D Converter Operation

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

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

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

$f_{SYS}$	A/D Clock Period ( $t_{ADCK}$ )							
	SACKS[2:0] = 000 ( $f_{SYS}$ )	SACKS[2:0] = 001 ( $f_{SYS}/2$ )	SACKS[2:0] = 010 ( $f_{SYS}/4$ )	SACKS[2:0] = 011 ( $f_{SYS}/8$ )	SACKS[2:0] = 100 ( $f_{SYS}/16$ )	SACKS[2:0] = 101 ( $f_{SYS}/32$ )	SACKS[2:0] = 110 ( $f_{SYS}/64$ )	SACKS[2:0] = 111 ( $f_{SYS}/128$ )
1MHz	1 $\mu$ s	2 $\mu$ s	4 $\mu$ s	8 $\mu$ s	16 $\mu$ s*	32 $\mu$ s*	64 $\mu$ s*	128 $\mu$ s*
2MHz	500ns	1 $\mu$ s	2 $\mu$ s	4 $\mu$ s	8 $\mu$ s	16 $\mu$ s*	32 $\mu$ s*	64 $\mu$ s*
4MHz	250ns*	500ns	1 $\mu$ s	2 $\mu$ s	4 $\mu$ s	8 $\mu$ s	16 $\mu$ s*	32 $\mu$ s*
8MHz	125ns*	250ns*	500ns	1 $\mu$ s	2 $\mu$ s	4 $\mu$ s	8 $\mu$ s	16 $\mu$ s*

#### A/D Clock Period Examples

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

### A/D Converter Reference Voltage

The reference voltage supply to the A/D converter can be supplied from the power supply  $AV_{DD}$ , or from the A/D converter OPA output  $V_{VR}$ , or from an external reference source supplied on pin VREF. The desired selection is made using the SAVRS1~SAVRS0 bits. When the SAVRS bit field is set to “01”, the A/D converter reference voltage will come from the  $AV_{DD}$ . When the SAVRS bit field is set to “10”, the A/D converter reference voltage will come from the OPA output. Otherwise, the A/D converter reference voltage will come from the VREF pin. As the VREF pin is pin-shared

with other functions, when the VREF pin is selected as the reference voltage supply pin, the VREF pin-shared function control bit should be properly configured to disable other pin function. However, if the internal A/D converter power  $AV_{DD}$  or the OPA output  $V_{VR}$  is selected as the reference voltage, the VREF pin must not be configured as the reference voltage input function to avoid the internal connection between the VREF pin and the internal reference signal.

The analog input values must not be allowed to exceed the value of the selected A/D reference voltage.

SAVRS[1:0]	Reference	Description
00, 11	VREF pin	External A/D converter reference pin VREF
01	$AV_{DD}$	Internal A/D converter power supply voltage
10	$V_{VR}$	Internal A/D converter OPA output voltage

**A/D Converter Reference Voltage Selection**

### A/D Converter Input Signals

All the external A/D analog channel input pins are pin-shared with the I/O pins as well as other functions. The corresponding control bits for each A/D external input pin in the PxS0 and PxS1 registers determine whether the input pins are setup as A/D converter analog input channel or whether they have other functions. If the pin is setup to be as an A/D analog channel input, the original pin functions will be disabled. In this way, pins can be changed under program control to change their function between A/D inputs and other functions. All pull high resistors, which are setup through register programming, will be automatically disconnected if the pins are setup as A/D inputs. Note that it is not necessary to first setup the A/D pin as an input in the port control register to enable the A/D input as when the pin-shared function control bits enable an A/D input, the status of the port control register will be overridden.

If the external channel input is selected to be converted, the SAINS1~SAINS0 bits should be set to “00” and the SACS2~SACS0 bits can determine which external channel is selected. If the internal analog signal is selected to be converted, the SACS2~SACS0 bits must be configured with an appropriate value to switch off the external analog channel input. Otherwise, the internal analog signal will be connected together with the external channel input. This will result in unpredictable situations.

SAINS[1:0]	SACS[2:0]	Input Signals	Description
00	000~011	ANO~AN3	External pin analog input
	100~111	—	Non-existed channel, input is floating
01	100~111	OPDAO	Proximity Sensing Circuit OPAMP output voltage
10	100~111	$V_{BAT}/4$	A/D converter external input $V_{BAT}/4$
11	100~111	$AV_{DD}/4$	A/D converter power supply voltage $AV_{DD}/4$

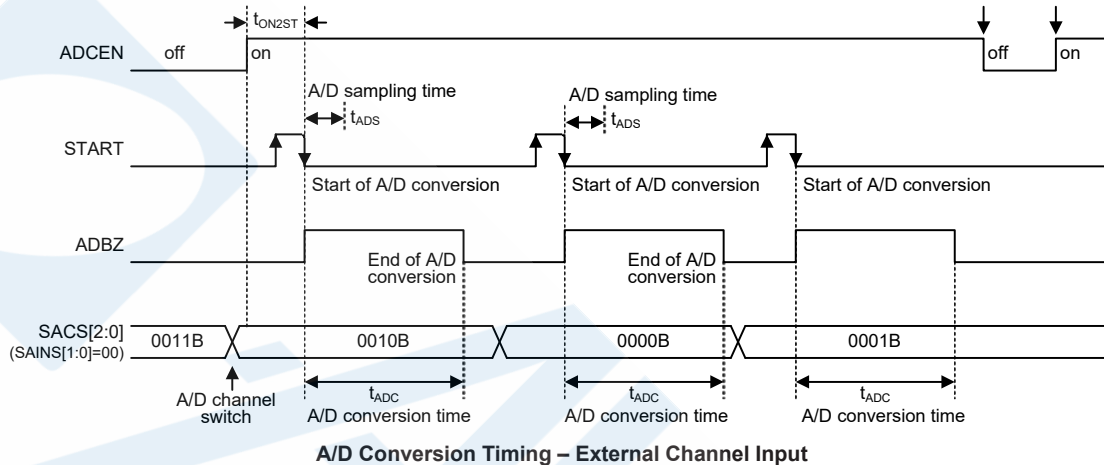
**A/D Converter Input Signal Selection**

### Conversion Rate and Timing Diagram

A complete A/D conversion contains two parts, data sampling and data conversion. The data sampling which is defined as  $t_{ADS}$  takes 4 A/D clock periods and the data conversion takes 10 A/D clock periods. Therefore, a total of 14 A/D clock periods for an external input A/D conversion which is defined as  $t_{ADC}$  are necessary.

$$\text{Maximum single A/D conversion rate} = 1 / (\text{A/D clock period} \times 14)$$

The accompanying diagram shows graphically the various stages involved in an analog to digital conversion process and its associated timing. After an A/D conversion process has been initiated by the application program, the microcontroller internal hardware will begin to carry out the conversion, during which time the program can continue with other functions. The time taken for the A/D conversion is  $14 t_{ADCK}$  where  $t_{ADCK}$  is equal to the A/D clock period.



### Summary of A/D Conversion Steps

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

- Step 1  
Select the required A/D conversion clock by correctly programming bits SACKS2~SACKS0 in the SADC1 register.
- Step 2  
Enable the A/D converter by setting the ADCEN bit in the SADC0 register to 1.
- Step 3  
Select which signal is to be connected to the internal A/D converter by correctly configuring the SAINS1~SAINS0 bits.  
Select the external channel input to be converted, go to Step 4.  
Select the internal analog signal to be converted, go to Step 5.
- Step 4  
If the A/D input signal comes from the external channel input selected by configuring the SAINS1~SAINS0 bits, the corresponding pins should be configured as A/D input function by configuring the relevant pin-shared function control bits. The desired analog channel then should be selected by configuring the SACS2~SACS0 bits. After this step, go to Step 6.
- Step 5  
Before the A/D input signal is selected to come from the internal analog signal by configuring the SAINS1~SAINS0 bits, the corresponding external input must be switched to a non-existent channel input by properly configured the SACS2~SACS0 bits. The desired internal analog signal then can be selected by configuring the SAINS1~SAINS0 bits. After this step, go to Step 6.

- Step 6  
Select the reference voltage source by configuring the SAVRS1~SAVRS0 bits in the SADC1 register. Care should be taken in this step which can refer to the A/D Converter Reference Voltage section for details.
- Step 7  
Select A/D converter output data format by setting the ADRFS bit in the SADC0 register.
- Step 8  
If A/D conversion interrupt is used, the interrupt control registers must be correctly configured to ensure the A/D interrupt function is active. The master interrupt control bit, EMI, and the A/D conversion interrupt control bit, ADE, must both be set high in advance.
- Step 9  
The A/D conversion procedure can now be initialized by setting the START bit from low to high and then low again.
- Step 10  
If A/D conversion is in progress, the ADBZ flag will be set high. After the A/D conversion process is complete, the ADBZ flag will go low and then the output data can be read from SADOH and SADOL registers.

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

### Programming Considerations

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

### A/D Conversion Function

As the device contains a 10-bit A/D converter, its full-scale converted digitised value is equal to 3FFH. Since the full-scale analog input value is equal to the actual A/D converter reference voltage,  $V_{REF}$ , this gives a single bit analog input value of  $V_{REF}$  divided by 1024.

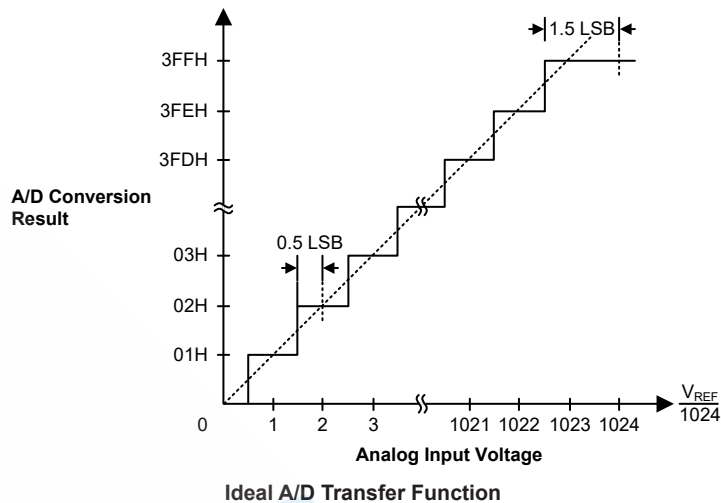
$$1 \text{ LSB} = V_{REF} \div 1024$$

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

$$\text{A/D input voltage} = \text{A/D output digital value} \times (V_{REF} \div 1024)$$

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

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



### A/D Conversion Programming Examples

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

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

```

clr ADE          ; disable ADC interrupt
mov a, 03H
mov SADC1, a     ; select input signal from external channel input, reference voltage
                 ; from AVDD, fsys/8 as A/D clock

mov a, 01h      ; setup PAS0 register to configure pin AN0
mov PAS0, a
mov a, 20h
mov SADC0, a    ; enable A/D converter and connect AN0 channel to A/D converter
:
start_conversion:
clr START      ; high pulse on start bit to initiate conversion
set START     ; reset A/D
clr START     ; start A/D
polling_EOC:
sz ADBZ       ; poll the SADC0 register ADBZ bit to detect end of A/D conversion
jmp polling_EOC ; continue polling
mov a, SADOL  ; read low byte conversion result value
mov SADOL_buffer, a ; save result to user defined register
mov a, SADOH  ; read high byte conversion result value
mov SADOH_buffer, a ; save result to user defined register
:
:
jmp start_conversion ; start next A/D conversion
    
```

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

```
clr ADE          ; disable ADC interrupt
mov a, 03H
mov SADC1, a     ; select input signal from external channel input, reference voltage
                ; from AVDD, fsys/8 as A/D clock
mov a, 01h      ; setup PAS0 register to configure pin AN0
mov PAS0, a
mov a, 20h
mov SADC0, a     ; enable A/D converter and connect AN0 channel to A/D converter
Start_conversion:
clr START       ; high pulse on START bit to initiate conversion
set START       ; reset A/D
clr START       ; start A/D
clr ADF         ; clear ADC interrupt request flag
set ADE         ; enable ADC interrupt
set EMI         ; enable global interrupt
:
:
                ; ADC interrupt service routine
ADC_ISR:
mov acc_stack, a ; save ACC to user defined memory
mov a, STATUS
mov status_stack, a ; save STATUS to user defined memory
:
:
mov a, SADOL    ; read low byte conversion result value
mov SADOL_buffer, a ; save result to user defined register
mov a, SADOH    ; read high byte conversion result value
mov SADOH_buffer, a ; save result to user defined register
:
:
EXIT_INT_ISR:
mov a, status_stack
mov STATUS, a   ; restore STATUS from user defined memory
mov a, acc_stack ; restore ACC from user defined memory
reti
```

## Touch Key Function

The 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 pin-shared selection register bits. Keys are organised into one group, known as Module 0. The module is a fully independent set of two Touch Keys and each key has its own oscillator. The key module contains its own control logic circuits and register set.

Total Key Number	Touch Key	Shared I/O Pin
2	KEY1~KEY2	PB2~PB3

**Touch Key Structure**

### Touch Key Register Definition

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

Register Name	Description
TKTMR	Touch key time slot 8-bit counter preload register
TKC0	Touch key function Control register 0
TKC1	Touch key function Control register 1
TK16DL	Touch key function 16-bit counter low byte
TK16DH	Touch key function 16-bit counter high byte
TKM016DL	Touch key module 0 16-bit C/F counter low byte
TKM016DH	Touch key module 0 16-bit C/F counter high byte
TKM0ROL	Touch key module 0 reference oscillator capacitor selection low byte
TKM0ROH	Touch key module 0 reference oscillator capacitor selection high byte
TKM0C0	Touch key module 0 Control register 0
TKM0C1	Touch key module 0 Control register 1

**Touch Key Function Register Definition**

Register Name	Bit							
	7	6	5	4	3	2	1	0
TKTMR	D7	D6	D5	D4	D3	D2	D1	D0
TKC0	—	TKRCOV	TKST	TKCFOV	TK16OV	—	TK16S1	TK16S0
TKC1	—	—	—	—	—	—	TKFS1	TKFS0
TK16DL	D7	D6	D5	D4	D3	D2	D1	D0
TK16DH	D15	D14	D13	D12	D11	D10	D9	D8
TKM016DL	D7	D6	D5	D4	D3	D2	D1	D0
TKM016DH	D15	D14	D13	D12	D11	D10	D9	D8
TKM0ROL	D7	D6	D5	D4	D3	D2	D1	D0
TKM0ROH	—	—	—	—	—	—	D9	D8
TKM0C0	M0MXS1	M0MXS0	M0DFEN	M0FILEN	M0SOFC	M0SOF2	M0SOF1	M0SOF0
TKM0C1	M0TSS	—	M0ROEN	M0KOEN	—	—	M0K2EN	M0K1EN

**Touch Key Function Register List**



• **TKTMR Register**

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

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

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

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

• **TKC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	TKRCOV	TKST	TKCFOV	TK16OV	—	TK16S1	TK16S0
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 **TKRCOV:** Touch key time slot counter overflow flag

0: No overflow occurs  
 1: Overflow occurs

When this bit is set by key module 0 time slot counter overflow, 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. 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.

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

0: Stopped or no operation  
 0→1: Start detection

In the module, the 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 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 Unimplemented, read as “0”

Bit 1~0 **TK16S1~TK16S0:** Touch key function 16-bit counter clock source select

00:  $f_{\text{SYS}}$   
 01:  $f_{\text{SYS}}/2$   
 10:  $f_{\text{SYS}}/4$   
 11:  $f_{\text{SYS}}/8$



• **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 selection  
 00: 1MHz  
 01: 3MHz  
 10: 7MHz  
 11: 11MHz

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

Register	TK16DH								TK16DL							
	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 cleared to zero.

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

Register	TKM016DH								TKM016DL							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Name	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
R/W	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This register pair is used to store the touch key module 0 16-bit C/F counter value. This 16-bit C/F counter will be stopped and the counter content will be kept unchanged when the touch key time slot counter overflows. This register pair will be cleared to zero when the TKST bit is cleared to zero.

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

Register	TKM0ROH								TKM0ROL							
	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 reference oscillator capacitor value.

The reference oscillator internal capacitor value = (TKM0RO[9:0]×30pF)/1024

• **TKM0C0 Register**

Bit	7	6	5	4	3	2	1	0
Name	M0MXS1	M0MXS0	M0DFEN	M0FILEN	M0SOFC	M0SOF2	M0SOF1	M0SOF0
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 **M0MXS1~M0MXS0**: Multiplexer Key Selection

00: KEY1  
 01: KEY2  
 10: Reserved  
 11: Reserved

Bit 5 **M0DFEN**: Touch key module 0 multi-frequency control

0: Disable  
 1: Enable

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

Bit 4 **M0FILEN**: Touch key module 0 filter function control

0: Disable  
 1: Enable

Bit 3 **M0SOFC**: Touch key module 0 C/F oscillator frequency hopping function control selection

0: Controlled by M0SOF2~M0SOF0  
 1: Controlled by hardware circuit

This bit is used to select the touch key oscillator frequency hopping function control method. When this bit is set to 1, the key oscillator frequency hopping function is controlled by the hardware circuit regardless of the M0SOF2~M0SOF0 bits value.

Bit 2~0 **M0SOF2~M0SOF0**: Touch key module 0 Reference and Key oscillators hopping frequency selection (when M0SOFC=0)

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

These bits are used to select the touch key oscillator frequency for the hopping function. Note that these bits are only available when the M0SOFC bit is cleared to 0.

The frequency mentioned here will be changed when the external or internal capacitor has different values. If the touch key operates at 1MHz frequency, users can adjust the frequency in scale when any other frequency is selected.

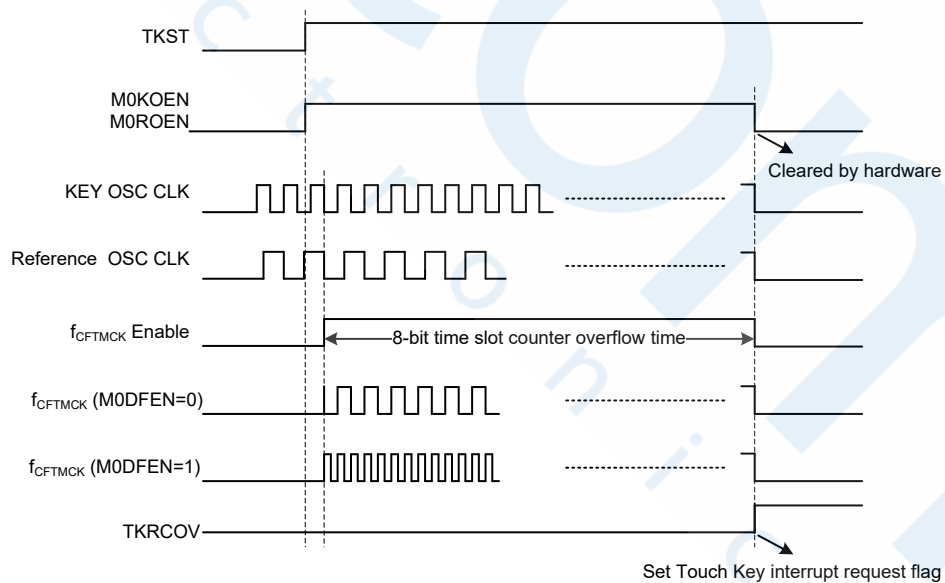
• **TKM0C1 Register**

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

- Bit 7      **M0TSS**: Touch key module 0 time slot counter clock source selection  
0: Touch key module 0 reference oscillator  
1:  $f_{sys}/4$
- Bit 6      Unimplemented, read as “0”
- Bit 5      **M0ROEN**: Touch key module 0 Reference oscillator enable control  
0: Disable  
1: Enable
- Bit 4      **M0KOEN**: Touch key module 0 Key oscillator enable control  
0: Disable  
1: Enable
- Bit 3~2    Unimplemented, read as “0”
- Bit 1      **M0K2EN**: Touch key module 0 KEY2 enable control  
0: Disable  
1: Enable
- Bit 0      **M0K1EN**: Touch key module 0 KEY1 enable control  
0: Disable  
1: Enable

**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 Timing Diagram**

The touch key module 0 contains two touch key inputs, KEY1~KEY2, which are shared with logical I/O pins, and the desired function is selected using the relevant pin-shared control register bits. Each touch key has its own independent sense oscillator. Therefore, there are two sense oscillators within the touch key module 0.

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

The touch key module 0 16-bit C/F counter, touch key function 16-bit counter, 5-bit time slot unit period counter will be automatically cleared when the TKST bit is cleared to zero, but the 8-bit programmable time slot counter will not be cleared. The overflow time is set by 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 module 0 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 MTSS bit in the TKMC1 register. The reference oscillator and key oscillator will be enabled by setting the MROEN bit and MKOEN bits in the TKMC1 register.

When the time slot counter 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.

### **Touch Key Interrupt**

The touch key only has a single interrupt, when the time slot counter in the touch key module 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 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 set, 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 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.

## Interrupts

Interrupts are an important part of any microcontroller system. When an external event or an internal function such as a Timer Module or an A/D converter requires microcontroller attention, their corresponding interrupt will enforce a temporary suspension of the main program allowing the microcontroller to direct attention to their respective needs. The device contains several external interrupt and internal interrupt functions. The external interrupt is generated by the action of the external INT pin, while the internal interrupts are generated by various internal functions including the TM, Time Bases, EEPROM, Touch Key and the A/D converter.

### 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. The registers fall into two categories. The first is the INTC0~INTC2 registers which setup the primary interrupts, the second is the INTEG register which sets the external interrupt trigger edge type.

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

Function	Enable Bit	Request Flag	Notes
Global	EMI	—	—
INT Pin	INTE	INTF	—
Touch Key	TKME	TKMF	—
A/D Converter	ADE	ADF	—
CTM	CTMPE	CTMPF	—
	CTMAE	CTMAF	—
EEPROM	DEE	DEF	—
Time Bases	TBnE	TBnF	n=0~1

**Interrupt Register Bit Naming Conventions**

Register Name	Bit							
	7	6	5	4	3	2	1	0
INTEG	—	—	—	—	—	—	INTS1	INTS0
INTC0	—	ADF	TKMF	INTF	ADE	TKME	INTE	EMI
INTC1	CTMPF	TB1F	TB0F	DEF	CTMPE	TB1E	TB0E	DEE
INTC2	—	—	—	CTMAF	—	—	—	CTMAE

**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	—	ADF	TKMF	INTF	ADE	TKME	INTE	EMI
R/W	—	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	0	0	0	0	0	0	0

- Bit 7 Unimplemented, read as “0”
- Bit 6 **ADF**: A/D Converter interrupt request flag  
0: No request  
1: Interrupt request
- Bit 5 **TKMF**: Touch Key interrupt request flag  
0: No request  
1: Interrupt request
- Bit 4 **INTF**: INT interrupt request flag  
0: No request  
1: Interrupt request
- Bit 3 **ADE**: A/D Converter interrupt control  
0: Disable  
1: Enable
- Bit 2 **TKME**: Touch Key interrupt control  
0: Disable  
1: Enable
- Bit 1 **INTE**: INT interrupt control  
0: Disable  
1: Enable
- Bit 0 **EMI**: Global interrupt control  
0: Disable  
1: Enable

• **INTC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	CTMPF	TB1F	TB0F	DEF	CTMPE	TB1E	TB0E	DEE
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 **CTMPF**: CTM Comparator P match interrupt request flag  
0: No request  
1: Interrupt request
- Bit 6 **TB1F**: Time Base 1 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 **DEF**: Data EEPROM interrupt request flag  
0: No request  
1: Interrupt request
- Bit 3 **CTMPE**: CTM Comparator P match interrupt control  
0: Disable  
1: Enable
- Bit 2 **TB1E**: Time Base 1 interrupt control  
0: Disable  
1: Enable

- Bit 1      **TB0E**: Time Base 0 interrupt control  
0: Disable  
1: Enable
- Bit 0      **DEE**: Data EEPROM interrupt control  
0: Disable  
1: Enable

• **INTC2 Register**

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

- Bit 7~5      Unimplemented, read as “0”
- Bit 4      **CTMAF**: CTM Comparator A match interrupt request flag  
0: No request  
1: Interrupt request
- Bit 3~1      Unimplemented, read as “0”
- Bit 0      **CTMAE**: CTM Comparator A match interrupt control  
0: Disable  
1: Enable

### Interrupt Operation

When the conditions for an interrupt event occur, such as a TM Comparator P, Comparator A match or A/D conversion completion etc., the relevant interrupt request flag will be set. Whether the request flag actually generates a program jump to the relevant interrupt vector is determined by the condition of the interrupt enable bit. If the enable bit is set high, then the program will jump to its relevant vector; if the enable bit is zero then although the interrupt request flag is set an actual interrupt will not be generated and the program will not jump to the relevant interrupt vector. The global interrupt enable bit, if cleared to zero, will disable all interrupts.

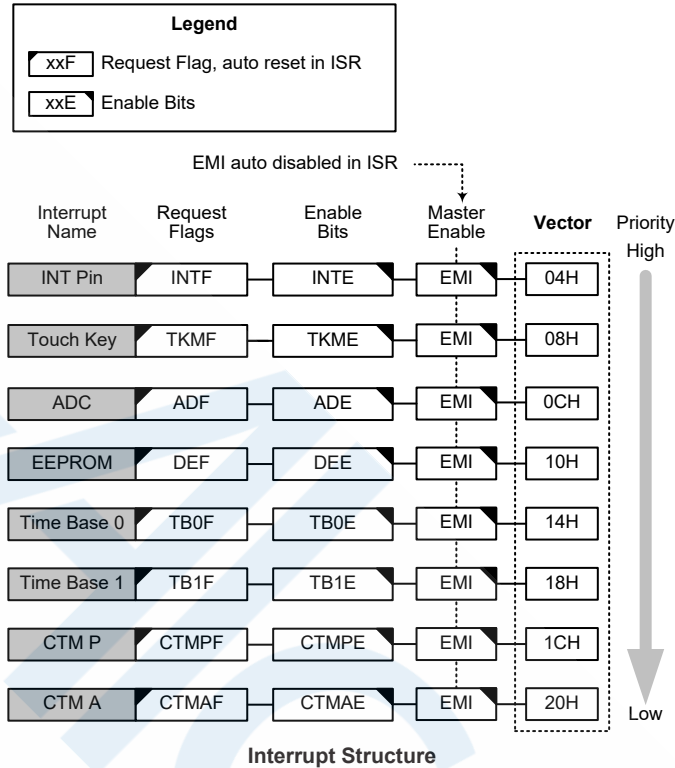
When an interrupt is generated, the Program Counter, which stores the address of the next instruction to be executed, will be transferred onto the stack. The Program Counter will then be loaded with a new address which will be the value of the corresponding interrupt vector. The microcontroller will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

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

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



is applied. All of the interrupt request flags when set will wake-up the device if it is in SLEEP or IDLE Mode, however to prevent a wake-up from occurring the corresponding flag should be set before the device are 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 selection bits, appears on the external interrupt pin. To allow the program to branch to its 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 setup as an input by setting the corresponding bit in the port control register. When the interrupt is enabled, the stack is not full and the correct transition type appears on the external interrupt pin, a subroutine call to the external interrupt vector, will take place. When the interrupt is serviced, the external interrupt request flag, INTF, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts. Note that the 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.



### Touch Key Interrupt

A Touch Key Interrupt request will take place when the Touch Key Interrupt request flag, TKMF, is set, which occurs when the touch key module 0 time slot counter overflows. To allow the program to branch to its interrupt vector address, the global interrupt enable bit, EMI, and the Touch Key interrupt enable bit, TKME, must be first set. When the interrupt is enabled, the stack is not full and the Touch Key module 0 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.

### A/D Converter Interrupt

The A/D Converter Interrupt is controlled by the termination of an A/D conversion process. An A/D Converter Interrupt request will take place when the A/D Converter Interrupt request flag, ADF, is set, which occurs when the A/D conversion process finishes. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and A/D Interrupt enable bit, ADE, must first be set. When the interrupt is enabled, the stack is not full and the A/D conversion process has ended, a subroutine call to the A/D Converter Interrupt vector, will take place. When the interrupt is serviced, the A/D Converter Interrupt flag, ADF, will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts.

### TM Interrupts

The Compact Type TM has two interrupts, one comes from the comparator A match situation and the other comes from the comparator P match situation. The Compact Type TM has two interrupt request flags, CTMAF and CTMPF, and two enable control bits, CTMAE and CTMPE. A TM interrupt request will take place when any of the TM request flags are set, a situation which occurs when a TM comparator P or A match situation happens.

To allow the program to branch to their respective interrupt vector addresses, the global interrupt enable bit, EMI, and the respective TM Interrupt enable bit, CTMAE or CTMPE, must first be set. When the interrupt is enabled, the stack is not full and a TM comparator match situation occurs, a subroutine call to the corresponding TM Interrupt vector location, will take place. When the TM interrupt is serviced, the TM interrupt request flag, CTMAF or CTMPF, will be automatically cleared and the EMI bit will be also automatically cleared to disable other interrupts.

### EEPROM Interrupt

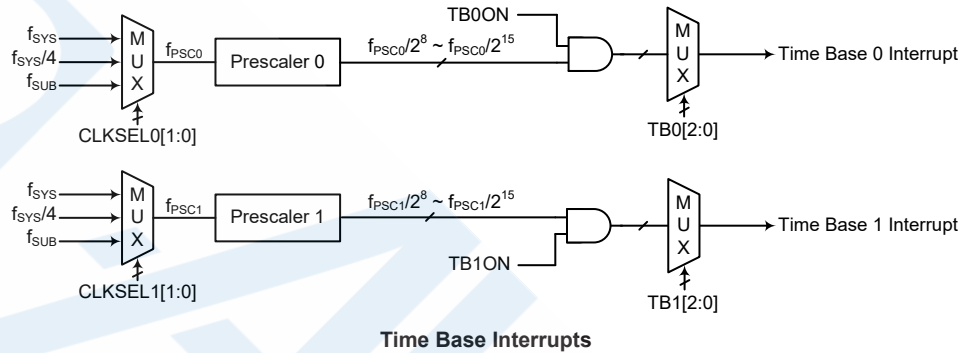
An EEPROM Interrupt request will take place when the EEPROM Interrupt request flag, DEF, is set, which occurs when an EEPROM Write cycle ends. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and the EEPROM Interrupt enable bit, DEE, must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

### Time Base Interrupts

The function of the Time Base Interrupts is to provide regular time signal in the form of an internal interrupt. They are controlled by the overflow signals from their respective timer functions. When these happens their respective interrupt request flags, TB0F or TB1F will be set. To allow the program to branch to their respective interrupt vector addresses, the global interrupt enable bit, EMI and Time Base enable bits, TB0E or TB1E, must first be set. When the interrupt is enabled, the stack is not full and the Time Base overflows, a subroutine call to their respective vector locations will

take place. When the interrupt is serviced, the respective interrupt request flag, TB0F or TB1F, will be automatically reset and the EMI bit will be cleared to disable other interrupts.

The purpose of the Time Base Interrupts is to provide an interrupt signal at fixed time periods. Its clock source,  $f_{PSC0}$  or  $f_{PSC1}$ , originates from the internal clock source  $f_{SYS}$ ,  $f_{SYS}/4$  or  $f_{SUB}$  and then passes through a divider, the division ratio of which is selected by programming the appropriate bits in the TB0C or TB1C 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 PSC0R and PSC1R register respectively.



• **PSCnR Register (n=0~1)**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	CLKSELn1	CLKSELn0
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2 Unimplemented, read as “0”

Bit 1~0 **CLKSELn1~CLKSELn0**: Prescaler n clock source selection

- 00:  $f_{SYS}$
- 01:  $f_{SYS}/4$
- 1x:  $f_{SUB}$

• **TBnC Register (n=0~1)**

Bit	7	6	5	4	3	2	1	0
Name	TBnON	—	—	—	—	TBn2	TBn1	TBn0
R/W	R/W	—	—	—	—	R/W	R/W	R/W
POR	0	—	—	—	—	0	0	0

Bit 7 **TBnON**: Time Base n Control

- 0: Disable
- 1: Enable

Bit 6~3 Unimplemented, read as “0”

Bit 2~0 **TBn2~TBn0**: Time Base n Time-out Period Selection

- 000:  $2^8/f_{PSC}$
- 001:  $2^9/f_{PSC}$
- 010:  $2^{10}/f_{PSC}$
- 011:  $2^{11}/f_{PSC}$
- 100:  $2^{12}/f_{PSC}$
- 101:  $2^{13}/f_{PSC}$
- 110:  $2^{14}/f_{PSC}$
- 111:  $2^{15}/f_{PSC}$

## 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.

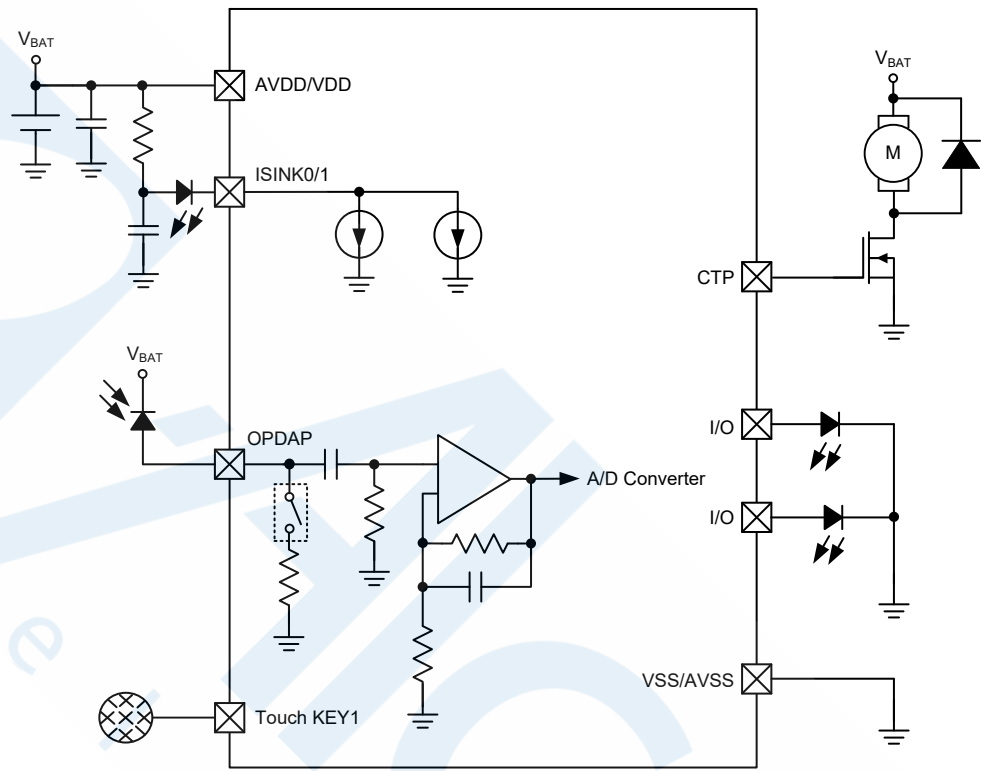
Every interrupt has the capability of waking up the microcontroller when it is in SLEEP or IDLE Mode, the wake up being generated when the interrupt request flag changes from low to high. If it is required to prevent a certain interrupt from waking up the microcontroller then its respective request flag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

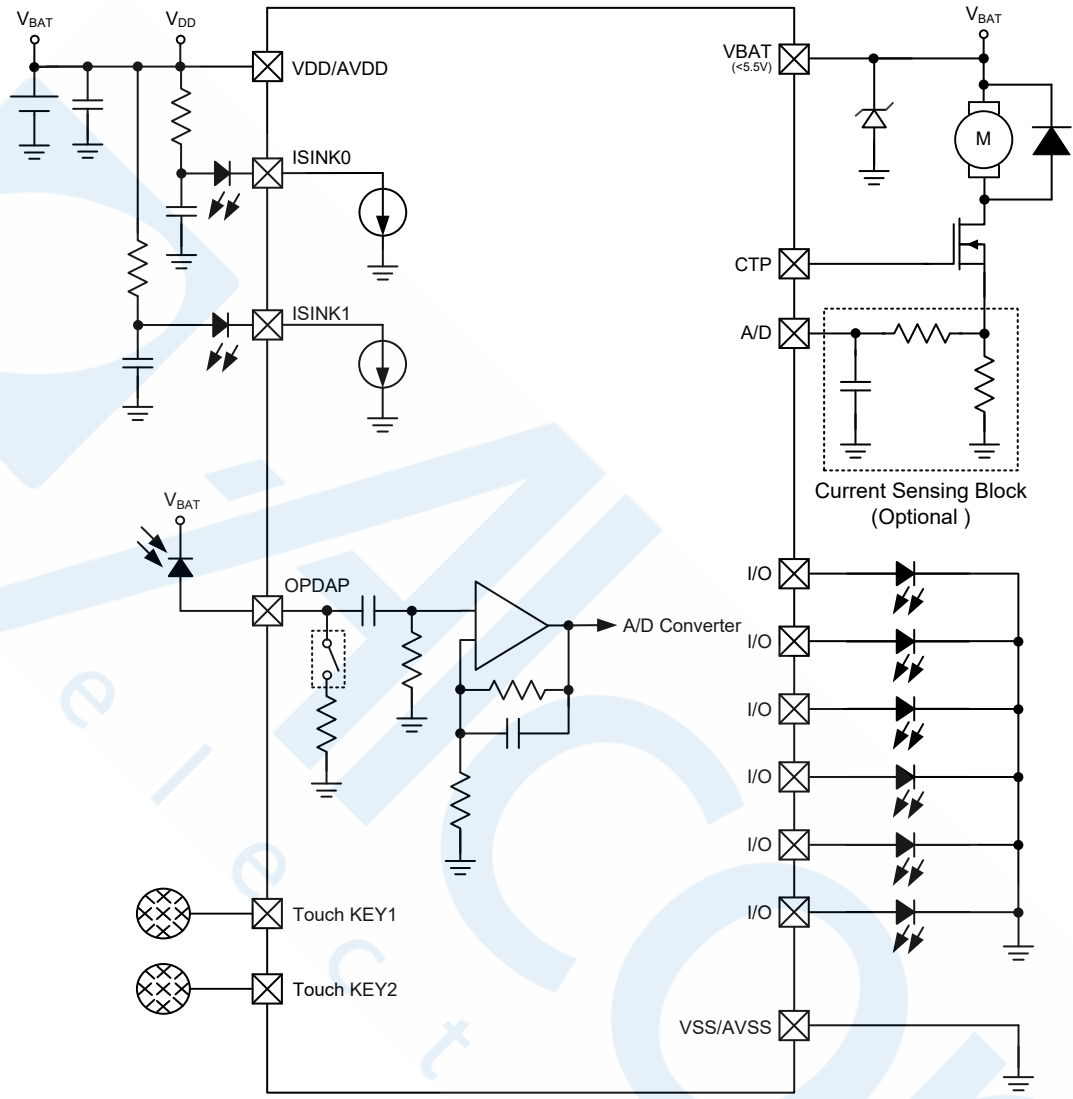
To return from an interrupt subroutine, either a RET or RETI instruction may be executed. The RETI instruction in addition to executing a return to the main program also automatically sets the EMI bit high to allow further interrupts. The RET instruction however only executes a return to the main program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

## Application Circuits

### Application Circuit for 8-pin SOP



Application Circuit for 16-pin NSOP



## 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	$[m] \leftarrow ACC \text{ "AND" } [m]$
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 ← Program Counter + 1 Program Counter ← 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] ← 00H
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 ← 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 TO ← 0 PDF ← 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] ← $\overline{[m]}$
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	ACC ← $\overline{[m]}$
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	$[m] \leftarrow [m] - 1$
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	$ACC \leftarrow [m] - 1$
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	$TO \leftarrow 0$ $PDF \leftarrow 1$
Affected flag(s)	TO, PDF
<b>INC [m]</b>	Increment Data Memory
Description	Data in the specified Data Memory is incremented by 1.
Operation	$[m] \leftarrow [m] + 1$
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	$ACC \leftarrow [m] + 1$
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 ← 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 ← 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] ← 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 ← 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 ← Stack ACC ← 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 ← Stack EMI ← 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) ← [m].i; (i=0~6) [m].0 ← [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	$ACC.(i+1) \leftarrow [m].i; (i=0\sim6)$ $ACC.0 \leftarrow [m].7$
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	$[m].(i+1) \leftarrow [m].i; (i=0\sim6)$ $[m].0 \leftarrow C$ $C \leftarrow [m].7$
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 ← [m].(i+1); (i=0~6) ACC.7 ← C C ← [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 ← ACC – [m] – $\bar{C}$
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] ← 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] ← [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 ← [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] ← 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 ← 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 ← [m].7~[m].4 ACC.7~ACC.4 ← [m].3~[m].0
Affected flag(s)	None
<b>SZ [m]</b>	Skip if Data Memory is 0
Description	If the contents of the specified Data Memory is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.
Operation	Skip if [m]=0
Affected flag(s)	None
<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 ← [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] ← program code (low byte) TBLH ← 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

**XORM A,[m]** Logical XOR ACC to Data Memory  
Description Data in the specified Data Memory and the Accumulator perform a bitwise logical XOR operation. The result is stored in the Data Memory.  
Operation  $[m] \leftarrow \text{ACC} \text{ "XOR" } [m]$   
Affected flag(s) Z

**XOR A,x** Logical XOR immediate data to ACC  
Description Data in the Accumulator and the specified immediate data perform a bitwise logical XOR operation. The result is stored in the Accumulator.  
Operation  $\text{ACC} \leftarrow \text{ACC} \text{ "XOR" } x$   
Affected flag(s) Z

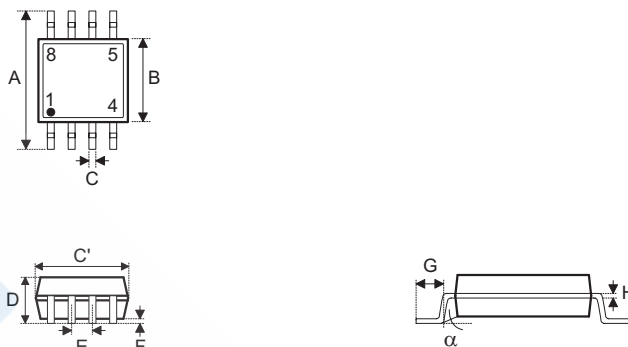
## Package Information

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](#)

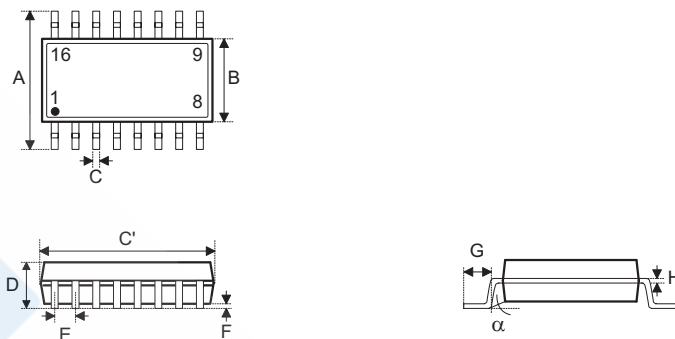
**8-pin SOP (150mil) Outline Dimensions**



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	—	6.0 BSC	—
B	—	3.9 BSC	—
C	0.31	—	0.51
C'	—	4.9 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°

**16-pin NSOP (150mil) Outline Dimensions**



Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	—	0.236 BSC	—
B	—	0.154 BSC	—
C	0.012	—	0.020
C'	—	0.390 BSC	—
D	—	—	0.069
E	—	0.050 BSC	—
F	0.004	—	0.010
G	0.016	—	0.050
H	0.004	—	0.010
α	0°	—	8°

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	—	6.000 BSC	—
B	—	3.900 BSC	—
C	0.31	—	0.51
C'	—	9.900 BSC	—
D	—	—	1.75
E	—	1.270 BSC	—
F	0.10	—	0.25
G	0.40	—	1.27
H	0.10	—	0.25
α	0°	—	8°



Singel 3 | B-2550 Kontich | Belgium | Tel. +32 (0)3 458 30 33

info@alcom.be | www.alcom.be

Rivium 1e straat 52 | 2909 LE Capelle aan den IJssel | The Netherlands

Tel. +31 (0)10 288 25 00 | info@alcom.nl | www.alcom.nl

Copyright© 2021 by HOLTEK SEMICONDUCTOR INC.

The information appearing in this Data Sheet is believed to be accurate at the time of publication. However, Holtek assumes no responsibility arising from the use of the specifications described. The applications mentioned herein are used solely for the purpose of illustration and Holtek makes no warranty or representation that such applications will be suitable without further modification, nor recommends the use of its products for application that may present a risk to human life due to malfunction or otherwise. Holtek's products are not authorized for use as critical components in life support devices or systems. Holtek reserves the right to alter its products without prior notification. For the most up-to-date information, please visit our web site at <http://www.holtek.com>.