



---

**Ultrasonic Atomiser Flash MCU**

**BS45F3843**

Revision: V1.21 Date: February 20, 2025

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

## Features

### CPU Features

- Operating Voltage
  - ♦  $f_{SYS}=8\text{MHz}$ : 2.2V~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
- Oscillators
  - ♦ Internal High Speed 8MHz RC – HIRC
  - ♦ Internal Low Speed 32kHz RC – LIRC
  - ♦ External Low Speed 32.768kHz Crystal – LXT
  - ♦ PTM Dedicated 12/14MHz RC – PTMOSC
- Multi-mode operation: FAST, SLOW, IDLE and SLEEP
- Fully integrated oscillators require no external components
- All instructions executed in one or two instruction cycles
- Table read instructions
- 61 powerful instructions
- 8-level subroutine nesting
- Bit manipulation instruction

### Peripheral Features

- Flash Program Memory: 4K $\times$ 16
- RAM Data Memory: 256 $\times$ 8
- True EEPROM Memory: 32 $\times$ 8
- Watchdog Timer function
- 26 bidirectional I/O lines
- Two pin-shared external interrupts
- Slew Rate Control for PA1 Port Output
- Programmable I/O port source current for LED applications
- Multiple Timer Modules for time measure, compare match output, PWM output function or single pulse output function
  - ♦ 10-bit CTM $\times$ 3
  - ♦ 10-bit STM $\times$ 1
  - ♦ 10-bit PTM $\times$ 1
- Over Current Protection (OCP) Function
- Dual Time-Base functions for generation of fixed time interrupt signals
- 8 external channel 12-bit resolution A/D converter with internal reference voltage  $V_{VR}$
- 8 touch key functions
- Fully-duplex/Half-duplex Universal Asynchronous Receiver and Transmitter Interface – UART
- Low voltage reset function
- Low voltage detect function
- Package Types: 16-pin NSOP, 24/28-pin SSOP

## Development Tools

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

[https://www.holtek.com/page/tool-detail/dev\\_plat/touch/Atomiser\\_Workshop](https://www.holtek.com/page/tool-detail/dev_plat/touch/Atomiser_Workshop)

## General Description

The BS45F3843 is a device dedicated for use in ultrasonic atomiser applications. The application principle for ultrasonic nebulisers is to use electronic high-frequency oscillation and ceramic nebulising chip high-frequency resonance to break up the liquid water molecules thus generating a fine mist without requiring heating or any chemical substances. Compared with the heating nebulisation method, the ultrasonic method can result in 90% energy savings. Additionally, during the nebulisation process, it can release a large number of negative ions which can precipitate smoke and dust particles in air by electrostatic reaction and also can effectively remove formaldehyde, carbon monoxide, bacteria and other harmful substances thus generating cleaner air and reducing the possibility of disease transmission.

The device is a Flash Memory type 8-bit high performance RISC architecture microcontroller containing special internal circuitry for ultrasonic atomiser applications and provides full integrated touch key functions which completely eliminating the need for external components.

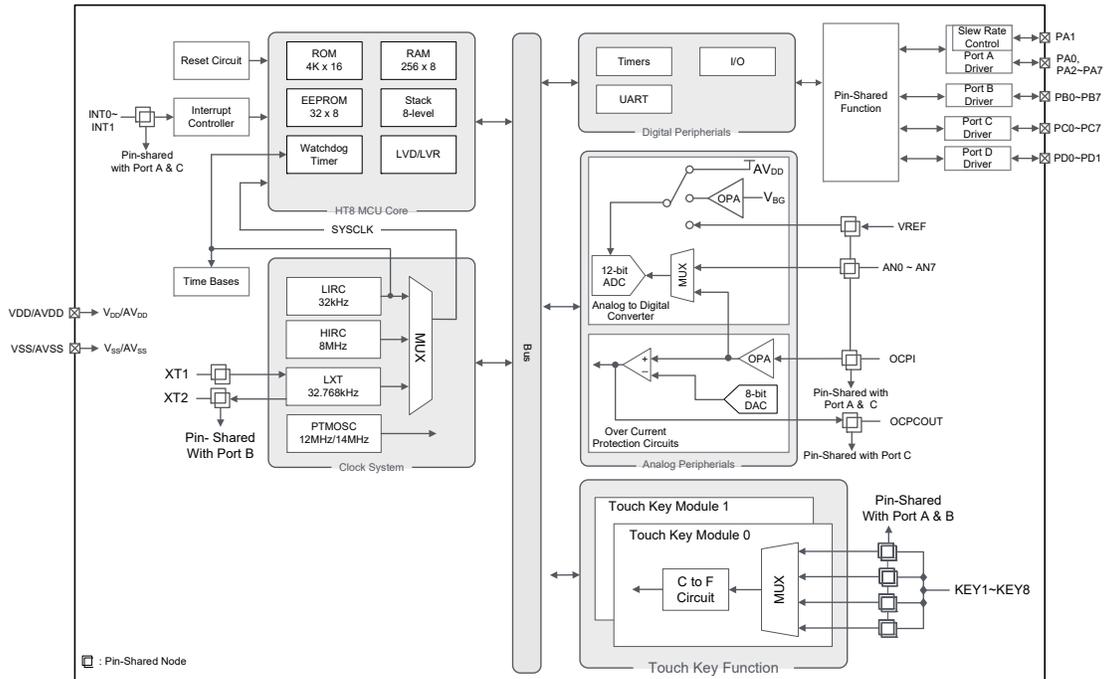
For memory features, the Flash Memory offers users the convenience of multi-programming features. Other memory includes an area of RAM Data Memory as well as an area of True EEPROM memory for storage of non-volatile data such as serial numbers, calibration data etc. Analog features include a multi-channel 12-bit A/D converter and an over current protection function. Multiple and extremely flexible Timer Modules provide timing, pulse generation and PWM generation functions. Communication with the outside world is catered for by including a fully integrated UART interface function, a popular interface which provides designers with a means of easy communication with external peripheral hardware. Protective features such as an internal Watchdog Timer, Low Voltage Reset and Low Voltage Detector coupled with excellent noise immunity and ESD protection ensure that reliable operation is maintained in hostile electrical environments.

The device also includes external low, internal high and low speed oscillators which can be flexibly used for different applications. The ability to operate and switch dynamically between a range of operating modes using different clock sources gives users the ability to optimise microcontroller operation and minimise power consumption. Easy communication with the outside world is provided using the internal I<sup>2</sup>C interface.

While the inclusion of flexible I/O programming features, Time-Base functions, touch keys along with an adjustable ultrasonic nebulizer resonant frequency generator and many other features ensure that the device will find excellent use in different ultrasonic nebuliser applications.

This device can use the nebuliser resonance detector to detect the nebuliser resonant frequency and use the nebulise resonant frequency selection to output PTM resonant frequency for nebuliser control, it can also use the water shortage protection and OCP functions for water shortage detection.

### Block Diagram



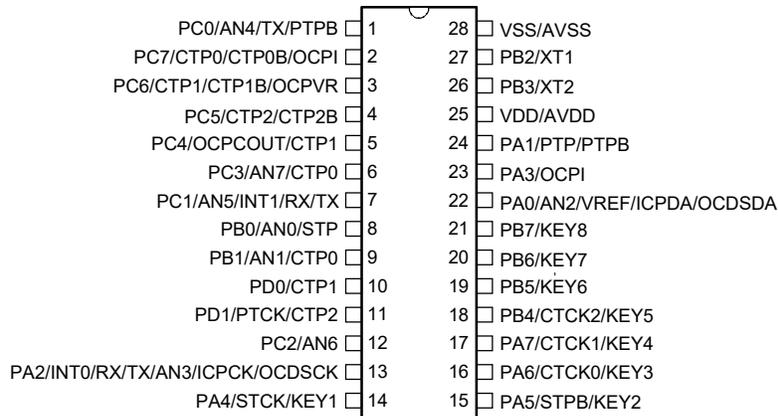
### Pin Assignment

VSS/AVSS	1	16	PB2/XT1
PC5/CTP2/CTP2B	2	15	PB3/XT2
PC4/OCPCOUT/CTP1	3	14	VDD/AVDD
PC3/AN7/CTP0	4	13	PA1/PTP/PTPB
PC1/AN5/INT1/RX/TX	5	12	PA3/OCPI
PA2/INT0/RX/TX/AN3/ICPCK/OCDSCK	6	11	PA0/AN2/VREF/ICPDA/OCSDSA
PA4/STCK/KEY1	7	10	PA7/CTCK1/KEY4
PA5/STPB/KEY2	8	9	PA6/CTCK0/KEY3

**BS45F3843/BS45V3843**  
**16 NSOP-A**

VSS/AVSS	1	24	PB2/XT1
PC0/AN4/TX/PTPB	2	23	PB3/XT2
PC5/CTP2/CTP2B	3	22	VDD/AVDD
PC4/OCPCOUT/CTP1	4	21	PA1/PTP/PTPB
PC3/AN7/CTP0	5	20	PA3/OCPI
PC1/AN5/INT1/RX/TX	6	19	PA0/AN2/VREF/ICPDA/OCSDSA
PB0/AN0/STP	7	18	PB7/KEY8
PB1/AN1/CTP0	8	17	PB6/KEY7
PD0/CTP1	9	16	PB4/CTCK2/KEY5
PD1/PTCK/CTP2	10	15	PA7/CTCK1/KEY4
PA2/INT0/RX/TX/AN3/ICPCK/OCDSCK	11	14	PA6/CTCK0/KEY3
PA4/STCK/KEY1	12	13	PA5/STPB/KEY2

**BS45F3843/BS45V3843**  
**24 SSOP-A**



**BS45F3843/BS45V3843**  
**28 SSOP-A**

- Note: 1. 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.
2. If the pin-shared pin functions have multiple output functions, the desired pin-shared function is determined using corresponding pin-shared software control bits.
3. The OCSDA and OCDSCK pins are supplied for the OCDS dedicated pins and as such only available for the BS45V3843 device which is the OCDS EV chip for the BS45F3843 device.

## Pin Descriptions

Note that the pin description refers to the largest package size, as a result some pins may not exist on smaller package types.

Pin Name	Function	OPT	I/T	O/T	Description
PA0/AN2/VREF/ICPDA/OCSDA	PA0	PAPU PAWU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	AN2	PAS0	AN	—	A/D converter external input channel
	VREF	PAS0	AN	—	A/D Converter external reference voltage input
	ICPDA	—	ST	CMOS	ICP data/address pin
	OCSDA	—	ST	CMOS	OCDS data/address pin, for EV chip only
PA1/PTP/PTPB	PA1	PAPU PAWU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	PTP	PAS0	—	CMOS	PTM output
	PTPB	PAS0	—	CMOS	PTM inverted output
PA2/INT0/RX/TX/AN3/ICPCK/OCDSCK	PA2	PAPU PAWU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	INT0	PAS0 INTEG INTC0	ST	—	External Interrupt 0
	RX/TX	PAS0 IFS	ST	CMOS	UART serial data input in full-duplex communication or UART serial data input / output in Single Wire Mode communication
	AN3	PAS0	AN	—	A/D converter external input channel
	ICPCK	—	ST	—	ICP clock pin
	OCDSCK	—	ST	—	OCDS clock pin, for EV chip only

Pin Name	Function	OPT	I/T	O/T	Description
PA3/OCPI	PA3	PAPU PAWU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	OCPI	PAS0	AN	—	OCP input signal
PA4/STCK/KEY1	PA4	PAPU PAWU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	STCK	PAS1	ST	—	STM clock input
	KEY1	PAS1	AN	—	Touch key input
PA5/STPB/KEY2	PA5	PAPU PAWU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	STPB	PAS1	—	CMOS	STM inverted output
	KEY2	PAS1	AN	—	Touch key input
PA6/CTCK0/KEY3	PA6	PAPU PAWU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	CTCK0	PAS1	ST	—	CTM0 clock input
	KEY3	PAS1	AN	—	Touch key input
PA7/CTCK1/KEY4	PA7	PAPU PAWU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	CTCK1	PAS1	ST	—	CTM1 clock input
	KEY4	PAS1	AN	—	Touch key input
PB0/AN0/STP	PB0	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	AN0	PBS0	AN	—	A/D Converter external input channel 0
	STP	PBS0	—	CMOS	STM output
PB1/AN1/CTP0	PB1	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	AN1	PBS0	AN	—	A/D Converter external input channel 1
	CTP0	PBS0	—	CMOS	CTM0 output
PB2/XT1	PB2	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	XT1	PBS0	LXT	—	LXT oscillator pin
PB3/XT2	PB3	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	XT2	PBS0	—	LXT	LXT oscillator pin
PB4/CTCK2/KEY5	PB4	PBPU PBS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	CTCK2	PBS1	ST	—	CTM2 clock input
	KEY5	PBS1	AN	—	Touch key input
PB5/KEY6	PB5	PBPU PBS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	KEY6	PBS1	AN	—	Touch key input
PB6/KEY7	PB6	PBPU PBS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	KEY7	PBS1	AN	—	Touch key input
PB7/KEY8	PB7	PBPU PBS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	KEY8	PBS1	AN	—	Touch key input

Pin Name	Function	OPT	I/T	O/T	Description
PC0/AN4/TX/PTPB	PC0	PCPU PCS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	AN4	PCS0	AN	—	A/D Converter external input channel 4
	TX	PCS0	—	CMOS	UART serial data output
	PTPB	PCS0	—	CMOS	PTM inverted output
PC1/AN5/INT1/RX/TX	PC1	PCPU PCS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	AN5	PCS0	AN	—	A/D Converter external input channel 5
	INT1	PCS0 INTEG INTC2	ST	—	External Interrupt 1
	RX/TX	PCS0 IFS	ST	CMOS	UART serial data input in full-duplex communication or UART serial data input / output in Single Wire Mode communication
PC2/AN6	PC2	PCPU PCS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	AN6	PCS0	AN	—	A/D Converter external input channel 6
PC3/AN7/CTP0	PC3	PCPU PCS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	AN7	PCS0	AN	—	A/D Converter external input channel 7
	CTP0	PCS0	—	CMOS	CTM0 output
PC4/OCPCOUT/CTP1	PC4	PCPU PCS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	OCPCOUT	PCS1	—	CMOS	OCP Comparator output
	CTP1	PCS1	—	CMOS	CTM1 output
PC5/CTP2/CTP2B	PC5	PCPU PCS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	CTP2	PCS1	—	CMOS	CTM2 output
	CTP2B	PCS1	—	CMOS	CTM2 inverted output
PC6/CTP1/CTP1B/ OCPVR	PC6	PCPU PCS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	CTP1	PCS1	—	CMOS	CTM1 output
	CTP1B	PCS1	—	CMOS	CTM1 inverted output
	OCPVR	PCS1	AN	—	OCP DAC reference voltage input
PC7/CTP0/CTP0B/ OCPI	PC7	PCPU PCS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	CTP0	PCS1	—	CMOS	CTM0 output
	CTP0B	PCS1	—	CMOS	CTM0 inverted output
	OCPI	PCS1	AN	—	OCP input signal
PD0/CTP1	PD0	PDS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	CTP1	PDS0	—	CMOS	CTM1 output
PD1/PTCK/CTP2	PD1	PDP PDS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	PTCK	PDS0	ST	—	PTM clock input
	CTP2	PDS0	—	CMOS	CTM2 output
VDD/AVDD	VDD	—	PWR	—	Digital positive power supply
	AVDD	—	PWR	—	Analog positive power supply
VSS/AVSS	VSS	—	PWR	—	Digital negative power supply, ground
	AVSS	—	PWR	—	Analog negative power supply, ground

Legend: I/T: Input type;

OPT: Optional by register option;

PWR: Power;

CMOS: CMOS output;

O/T: Output type;

ST: Schmitt Trigger input;

AN: Analog signal.

## Absolute Maximum Ratings

Supply Voltage .....	$V_{SS}-0.3V$ to $6.0V$
Input Voltage .....	$V_{SS}-0.3V$ to $V_{DD}+0.3V$
Storage Temperature.....	$-60^{\circ}C$ to $150^{\circ}C$
Operating Temperature.....	$-40^{\circ}C$ to $85^{\circ}C$
$I_{OL}$ 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 this device at other conditions beyond those listed in the specification is not implied and prolonged exposure to extreme conditions may affect device reliability.

## D.C. Characteristics

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

### Operating Voltage Characteristics

$T_a=-40^{\circ}C-85^{\circ}C$

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
$V_{DD}$	Operating Voltage – HIRC	$f_{SYS}=f_{HIRC}=8MHz$	2.2	—	5.5	V
	Operating Voltage – LXT	$f_{SYS}=32.768kHz$	2.2	—	5.5	V
	Operating Voltage – LIRC	$f_{SYS}=f_{LIRC}=32kHz$	2.2	—	5.5	V

### Standby Current Characteristics

$T_a=25^{\circ}C$ , unless otherwise specified

Symbol	Standby Mode	Test Conditions		Min.	Typ.	Max.	Max. @85°C	Unit	
		$V_{DD}$	Conditions						
$I_{STB}$	SLEEP Mode	2.2V	WDT off	—	0.11	0.15	2.00	$\mu A$	
		3V		—	0.11	0.15	2.00		
		5V		—	0.18	0.38	2.90		
		2.2V		WDT on	—	1.2	2.4	2.9	$\mu A$
		3V			—	1.5	3.0	3.6	
		5V			—	3	5	6	
	IDLE0 Mode – LIRC	2.2V	$f_{SUB}$ on	—	2.4	4.0	4.8	$\mu A$	
		3V		—	3	5	6		
		5V		—	5	10	12		
	IDLE0 Mode – LXT	2.2V	$f_{SUB}$ on	—	2.4	4.0	4.8	$\mu A$	
		3V		—	3	5	6		
		5V		—	5	10	12		
IDLE1 Mode – HIRC	2.2V	$f_{SUB}$ on, $f_{SYS}=8MHz$	—	288	400	480	$\mu A$		
	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.

### 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	2.2V	f <sub>sys</sub> =32kHz	—	8	16	μA
		3V		—	10	20	
		5V		—	30	50	
	SLOW Mode – LXT	2.2V	f <sub>sys</sub> =32.768kHz	—	8	16	μA
		3V		—	10	20	
		5V		—	30	50	
	FAST Mode – HIRC	2.2V	f <sub>sys</sub> =8MHz	—	0.6	1.0	mA
		3V		—	0.8	1.2	
		5V		—	1.6	2.4	

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

1. Any digital inputs are 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.

## A.C. Characteristics

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

### Internal High Speed Oscillator – HIRC – Frequency Accuracy

During the program writing operation the writer will trim the HIRC oscillator at a user selected HIRC frequency and user selected voltage of 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	-2%	8	+2%	
		2.2V~5.5V	25°C	-2.5%	8	+2.5%	
			-40°C~85°C	-3%	8	+3%	

- Note: 1. The 3V/5V values for V<sub>DD</sub> are provided as these are the two selectable fixed voltages at which the HIRC frequency is trimmed by the writer.
2. The row below the 3V/5V trim voltage row is provided to show the values for the full V<sub>DD</sub> range operating voltage. It is recommended that the trim voltage is fixed at 3V for application voltage ranges from 2.2V to 3.6V and fixed at 5V for application voltage ranges from 3.3V to 5.5V.

**Internal Low Speed Oscillator Characteristics – LIRC**

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V <sub>DD</sub>	Temp.				
f <sub>LIRC</sub>	LIRC Frequency	2.2V~5.5V	-40°C~85°C	-7%	32	+7%	kHz
t <sub>START</sub>	LIRC Start Up Time	—	-40°C~85°C	—	—	100	µs

**External Low Speed Crystal Oscillator Characteristics – LXT**

T<sub>a</sub>=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V <sub>DD</sub>	Conditions				
f <sub>LXT</sub>	LXT Frequency	2.2V~5.5V	—	—	32.768	—	kHz
Duty Cycle	Duty Cycle	—	—	40	—	60	%
t <sub>START</sub>	LXT Start Up Time	3V	—	—	—	1000	ms
		5V	—	—	—	1000	
R <sub>NEG</sub>	Negative Resistance <sup>(Note)</sup>	2.2V	—	3×ESR	—	—	Ω

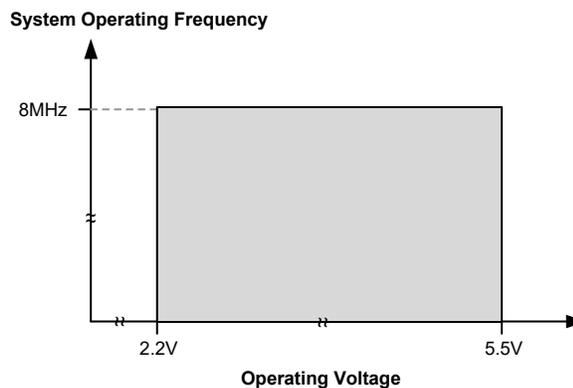
Note: C1, C2 and R<sub>P</sub> are external components. C1=C2=10pF. R<sub>P</sub>=10MΩ. C<sub>L</sub>=7pF, ESR=30kΩ.

**PTM Dedicated Internal RC Oscillator Characteristics**

This 12/14MHz internal RC oscillator is dedicated for providing clocks for the PTM. The writer trim the 12/14MHz oscillator at conditions of V<sub>DD</sub> is 5V and FADJ value is 80H.

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V <sub>DD</sub>	Conditions				
V <sub>DD</sub>	Operating voltage	—	f <sub>PTM</sub> =12MHz	2.7	—	5.5	V
		—	f <sub>PTM</sub> =14MHz	3.0	—	5.5	V
f <sub>PTM</sub>	12MHz Writer Trimmed HIRC Frequency	5V	T <sub>a</sub> =25°C	-2%	12.4	+2%	MHz
		2.7V~5.5V	T <sub>a</sub> =-40°C~85°C	-10%	12.4	+10%	
	14MHz Writer Trimmed HIRC Frequency	5V	T <sub>a</sub> =25°C	-3%	14.4	+3%	MHz
		3.0V~5.5V	T <sub>a</sub> =-40°C~85°C	-10%	14.4	+10%	

**Operating Frequency Characteristic Curves**



### System Start Up Time Characteristics

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V <sub>DD</sub>	Conditions				
t <sub>SST</sub>	System Start-up Time (Wake-up from Condition where f <sub>sys</sub> is off)	—	f <sub>sys</sub> =f <sub>H</sub> ~f <sub>H</sub> /64, f <sub>H</sub> =f <sub>HIRC</sub>	—	16	—	t <sub>HIRC</sub>
			f <sub>sys</sub> =f <sub>SUB</sub> =f <sub>LXT</sub>	—	1024	—	t <sub>LXT</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>LXT</sub> OR f <sub>LIRC</sub>	—	2	—	t <sub>SUB</sub>
	System Speed Switch Time (FAST to Slow Mode or SLOW to FAST Mode)	—	f <sub>HIRC</sub> switches from off → on	—	16	—	t <sub>HIRC</sub>
f <sub>LXT</sub> switches from off → on			—	1024	—	t <sub>LXT</sub>	
t <sub>RSTD</sub>	System Reset Delay Time (Reset Source from Power-on Reset or LVR Hardware Reset)	—	RR <sub>POR</sub> =5V/ms	14	16	18	ms
	System Reset Delay Time (LVRC/WDT Software Reset)	—	—				
	System Reset Delay Time (Reset Source from WDT Overflow pin reset)	—	—				
t <sub>SRESET</sub>	Minimum Software Reset Pulse 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. is 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=25°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 PA0, PA2~PA7, PB2~PB7 Pins	3V	V <sub>OL</sub> =0.1V <sub>DD</sub>	16	32	—	mA
		5V		32	65	—	
	Sink Current for PB0~1, PC0~7, PD0~1 Pins	3V	V <sub>OL</sub> =0.1V <sub>DD</sub> , P <sub>x</sub> NS <sub>n</sub> =0	16	32	—	mA
		3V	V <sub>OL</sub> =0.1V <sub>DD</sub> , P <sub>x</sub> NS <sub>n</sub> =1	25	50	—	
		5V	V <sub>OL</sub> =0.1V <sub>DD</sub> , P <sub>x</sub> NS <sub>n</sub> =0	32	65	—	
			V <sub>OL</sub> =0.1V <sub>DD</sub> , P <sub>x</sub> NS <sub>n</sub> =1	50	100	—	

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V <sub>DD</sub>	Conditions				
I <sub>OH</sub>	Source Current for PA3 Pin	3V	V <sub>OH</sub> =0.9V <sub>DD</sub>	-4	-8	—	mA
		5V		-8	-16	—	
	Source Current for I/O Ports Except PA1, PA3 Pins	3V	V <sub>OH</sub> =0.9V <sub>DD</sub> , SLEDCn[m+1:m]=00, (n=0~2; m=0, 2, 4, 6)	-0.7	-1.5	—	
		5V		-1.5	-2.9	—	
		3V	V <sub>OH</sub> =0.9V <sub>DD</sub> , SLEDCn[m+1:m]=01, (n=0~2; m=0, 2, 4, 6)	-1.3	-2.5	—	
		5V		-2.5	-5.1	—	
		3V	V <sub>OH</sub> =0.9V <sub>DD</sub> , SLEDCn[m+1:m]=10, (n=0~2; m=0, 2, 4, 6)	-1.8	-3.6	—	
		5V		-3.6	-7.3	—	
		3V	V <sub>OH</sub> =0.9V <sub>DD</sub> , SLEDCn[m+1:m]=11, (n=0~2; m=0, 2, 4, 6)	-4	-8	—	
5V	-8	-16		—			
R <sub>PH</sub>	Pull-high Resistance for I/O Ports <sup>(Note)</sup>	3V	—	20	60	100	kΩ
		5V		10	30	50	
I <sub>LEAK</sub>	Input Leakage Current for I/O Ports	3V	V <sub>IN</sub> =V <sub>DD</sub> or V <sub>IN</sub> =V <sub>SS</sub>	—	—	±1	μA
		5V		—	—	±1	
t <sub>TCK</sub>	TM Clock Input Pin Minimum Pulse Width	—	—	0.3	—	—	μs
t <sub>INT</sub>	External Interrupt Input 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.

## Slew Rate Control Characteristics

T<sub>a</sub>=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V <sub>DD</sub>	Conditions				
I <sub>OL</sub>	Sink Current for PA1 Pin with Slew Rate Control	3V	V <sub>OL</sub> =0.2V <sub>DD</sub>	24	60	—	mA
		5V		60	150	—	
I <sub>OH</sub>	Source Current for PA1 Pin with Slew Rate Control	3V	V <sub>OH</sub> =0.8V <sub>DD</sub>	-24	-60	—	mA
		5V		-60	-150	—	
SR <sub>RISE</sub>	Output Rising Edge Slew Rate for PA1 Pin	5V	SLEWC[1:0]=00B 0.5V to 4.5V, C <sub>LOAD</sub> =1000pF	200	—	—	V/μs
		5V	SLEWC[1:0]=01B 0.5V to 4.5V, C <sub>LOAD</sub> =1000pF	—	120	—	V/μs
		5V	SLEWC[1:0]=10B 0.5V to 4.5V, C <sub>LOAD</sub> =1000pF	—	60	—	V/μs
		5V	SLEWC[1:0]=11B 0.5V to 4.5V, C <sub>LOAD</sub> =1000pF	—	45	—	V/μs
SR <sub>FALL</sub>	Output Falling Edge Slew Rate for PA1 Pin	5V	SLEWC[1:0]=00B 4.5V to 0.5V, C <sub>LOAD</sub> =1000pF	200	—	—	V/μs
		5V	SLEWC[1:0]=01B 4.5V to 0.5V, C <sub>LOAD</sub> =1000pF	—	120	—	V/μs
		5V	SLEWC[1:0]=10B 4.5V to 0.5V, C <sub>LOAD</sub> =1000pF	—	60	—	V/μs
		5V	SLEWC[1:0]=11B 4.5V to 0.5V, C <sub>LOAD</sub> =1000pF	—	45	—	V/μs

## Memory 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>RW</sub>	V <sub>DD</sub> for Read / Write	—	—	V <sub>DDmin</sub>	—	V <sub>DDmax</sub>	V
<b>Flash Program / Data EEPROM Memory</b>							
t <sub>DEW</sub>	Erase / Write Cycle Time – Flash Program Memory	—	—	—	2	3	ms
	Erase / Write Cycle Time – Data EEPROM Memory	3.0V ~5.5V	—	—	4	6	ms
t <sub>EERD</sub>	EEPROM 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	—	—	E/W
t <sub>RETD</sub>	ROM Data Retention Time	—	Ta=25°C	—	40	—	Year

Note: “E/W” means Erase/Write times.

## LVD & LVR Electrical Characteristics

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V <sub>DD</sub>	Conditions				
V <sub>LVR</sub>	Low Voltage Reset Voltage	—	LVR enable, 2.1V	-5%	2.1	+5%	V
			LVR enable, 2.55V		2.55		
			LVR enable, 3.15V		3.15		
			LVR enable, 3.8V		3.8		
V <sub>LVD</sub>	Low Voltage Detector Voltage	—	LVD enable, voltage select 2.0V	-5%	2.0	+5%	V
			LVD enable, voltage select 2.2V		2.2		
			LVD enable, voltage select 2.4V		2.4		
			LVD enable, voltage select 2.7V		2.7		
			LVD enable, voltage select 3.0V		3.0		
			LVD enable, voltage select 3.3V		3.3		
			LVD enable, voltage select 3.6V		3.6		
LVD enable, voltage select 4.0V	4.0						
I <sub>LVR/LVD</sub>	Operating Current	3V	LVD enable, LVR enable, V <sub>LVR</sub> =2.1V, V <sub>LVD</sub> =2.2V	—	—	10	μA
		5V		—	8	15	μA
I <sub>LVR</sub>	Additional Current for LVR Enable	—	LVD Disable	—	—	8	μA
t <sub>LVR</sub>	Minimum Low Voltage Width to Reset	—	—	120	240	480	μs
t <sub>LVD</sub>	Minimum Low Voltage Width to Interrupt	—	—	60	120	240	μs
t <sub>LVDS</sub>	LVDO Stable Time	—	For LVR enable, LVD off → on	—	—	18	μs

## Internal Bandgap Reference Voltage Electrical Characteristics

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V <sub>DD</sub>	Conditions				
V <sub>BG</sub>	Bandgap Reference Voltage	—	—	-20%	1.03	+20%	V

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

## A/D Converter Electrical Characteristics

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V <sub>DD</sub>	Conditions				
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	—	—	—	—	12	Bit
DNL	Differential Non-linearity	—	V <sub>REF</sub> =V <sub>DD</sub> , t <sub>ADCK</sub> =0.5μs	-3	—	+3	LSB
INL	Integral Non-linearity	—	V <sub>REF</sub> =V <sub>DD</sub> , t <sub>ADCK</sub> =0.5μs	-4	—	+4	LSB
I <sub>ADC</sub>	Additional Current for A/D Converter Enable	2.2V	No load, t <sub>ADCK</sub> =0.5μs	—	300	420	μA
		3V		—	340	500	μA
		5V		—	500	700	μA
t <sub>ADCK</sub>	A/D Converter Clock Period	—	—	0.5	—	10	μs
t <sub>ON2ST</sub>	A/D Converter On-to-Start Time	—	—	4	—	—	μs
t <sub>ADC</sub>	A/D Conversion Time (Include A/D Sample and Hold Time)	—	—	—	16	—	t <sub>ADCK</sub>
I <sub>OPA</sub>	Additional CURRENT FOR OPA ENABLE	3V	No load	—	390	550	μA
		5V	No load	—	500	650	μA
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
V <sub>VR</sub>	Fix Voltage Output of OPA	—	V <sub>DD</sub> =2.2V~5.5V	-5%	1.6	+5%	V

## Over Current Protection Electrical Characteristics

Ta=-40°C~85°C

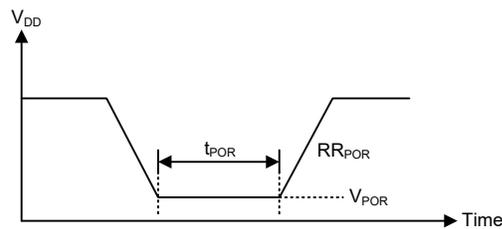
Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V <sub>DD</sub>	Conditions				
I <sub>OCP</sub>	Operating Current	3V	OCPEN[1:0]=01 OCPVRS[1:0]=10	—	300	500	μA
		5V	OCPCHY=1, G[2:0]=000	—	450	600	
V <sub>REF</sub>	DAC Reference voltage	3V	OCPVRS[1:0]=01	2.2	—	V <sub>DD</sub>	V
		5V		2.2	—	V <sub>DD</sub>	V
V <sub>OS_CMP</sub>	Comparator Input Offset Voltage	3V	Without calibration (OCPCOF[4:0]=10000B)	-15	—	15	mV
		5V		-15	—	15	
		3V	With calibration	-2	—	2	mV
		5V		-2	—	2	
V <sub>HYS</sub>	Hysteresis	3V	—	10	40	60	mV
		5V	—	10	40	60	
V <sub>CM_CMP</sub>	Comparator Common Mode Voltage Range	3V	—	V <sub>SS</sub>	—	V <sub>DD</sub> -1.0	V
		5V	—	V <sub>SS</sub>	—	V <sub>DD</sub> -1.0	

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V <sub>DD</sub>	Conditions				
V <sub>OS_OPA</sub>	OPA Input Offset Voltage	3V	Without calibration (OCPOOF[5:0]=100000B)	-15	—	15	mV
		5V		-15	—	15	
		3V	With calibration	-2	—	2	mV
		5V		-2	—	2	
V <sub>CM_OPA</sub>	OPA Common Mode Voltage Range	3V	—	V <sub>SS</sub>	—	V <sub>DD</sub> -1.4	V
		5V	—	V <sub>SS</sub>	—	V <sub>DD</sub> -1.4	
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	
Ga	PGA Gain Accuracy	3V	All gain	-5	—	5	%
		5V		-5	—	5	
DNL	Differential Non-linearity	3V	DAC V <sub>REF</sub> =V <sub>DD</sub>	-1.5	—	1.5	LSB
		5V		-1	—	1	
INL	Integral Non-linearity	3V	DAC V <sub>REF</sub> =V <sub>DD</sub>	-2	—	2	LSB
		5V		-1.5	—	1.5	

### Power on Reset Electrical Characteristics

T<sub>a</sub>=25°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



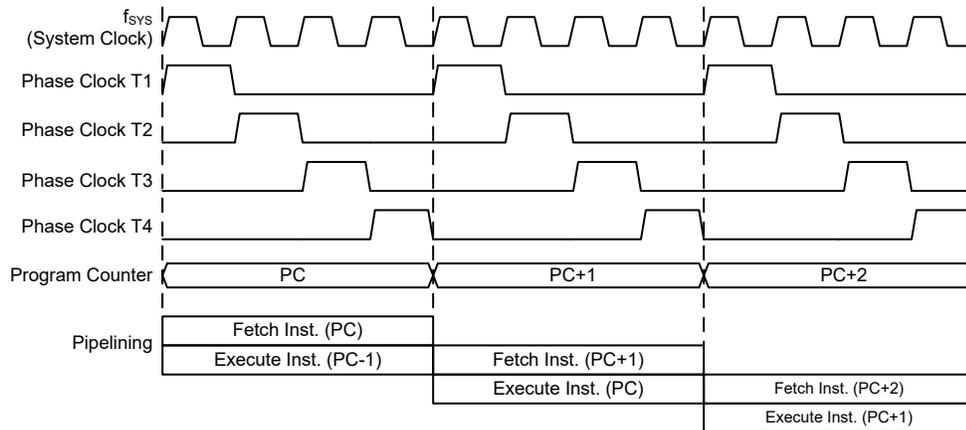
## System Architecture

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 Periodic performance. The pipelining scheme is implemented in such a way that instruction fetching and instruction execution are overlapped, hence instructions are effectively executed in one 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.

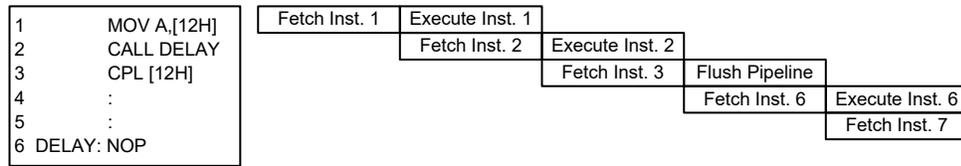
### Clocking and Pipelining

The main system clock, derived from either an HIRC, LXT 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 Clock and Pipelining**



**Instruction Fetching**

**Program Counter**

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

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

Program Counter	
High Byte	Low Byte (PCL Register)
PC11~PC8	PCL7~PCL0

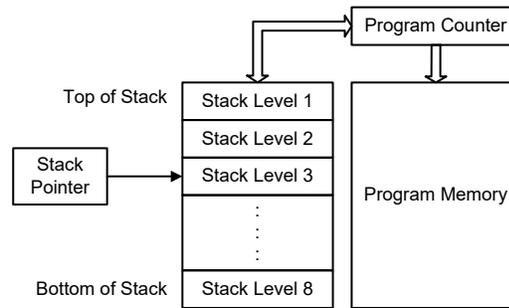
The lower byte of the Program Counter, known as the Program Counter Low register or PCL, is available for program control and is a readable and writeable register. By transferring data directly into this register, a short program jump can be executed directly, however, as only this low byte is available 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 8 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 saved 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:

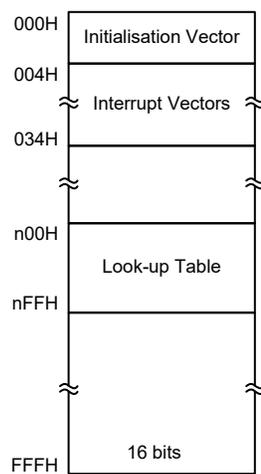
- Arithmetic operations: ADD, ADDM, ADC, ADCM, SUB, SUBM, SBC, SBCM, DAA
- Logic operations: AND, OR, XOR, ANDM, ORM, XORM, CPL, CPLA
- Rotation: RRA, RR, RRCA, RRC, RLA, RL, RLCA, RLC
- Increment and Decrement: INCA, INC, DECA, DEC
- Branch decision: JMP, SZ, SZA, SNZ, SIZ, SDZ, SIZA, SDZA, CALL, RET,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

## Flash Program Memory

The Program Memory is the location where the user code or program is stored. For this 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, this 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 4K×16 bits. The Program Memory is addressed by the Program Counter and also contains data, table information and interrupt entries. Table data, which can be setup in any location within the Program Memory, is addressed by a separate table pointer register.



**Program Memory Structure**

### Special Vectors

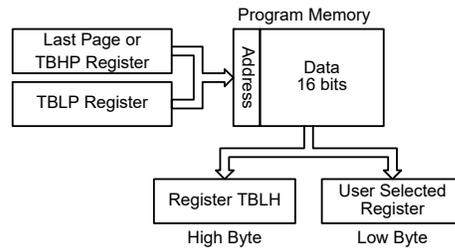
Within the Program Memory, certain locations are reserved for the reset and interrupts. The location 000H is reserved for use by the device reset for program initialisation. After a device reset is initiated, the program will jump to this location and begin execution.

### 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]” instructions, respectively. When the instruction is executed, the lower order table byte from the Program Memory will be transferred to the user defined Data Memory register [m] as specified in the instruction. The higher order table data byte from the Program Memory will be transferred to the TBLH special register.

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



### Table Program Example

The following example shows how the table pointer and table data is defined and retrieved from the microcontroller. This example uses raw table data located in the Program Memory which is stored there using the ORG statement. The value at this ORG statement is “0F00H” which refers to the start address of the last page within the 4K 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 “0F06H” or 6 locations after the start of the last page. Note that the value for the table pointer is referenced to the first address 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 cannot be restored, care should be taken to ensure its protection if both the main routine and Interrupt Service Routine use table read instructions. If using the table read instructions, the Interrupt Service Routines may change the value of the TBLH and subsequently cause errors if used again by the main routine. As a rule, it is recommended that simultaneous use of the table read instructions should be avoided. However, in situations where simultaneous use cannot be avoided, the interrupts should be disabled prior to the execution of any main routine table-read instructions. Note that all table related instructions require two instruction cycles to complete their operation.

### Table Read Program Example

```

tempreg1 db ? ; temporary register #1
tempreg2 db ? ; temporary register #2
:
:
mov a,06h ; initialise low table pointer - note that this address is referenced
mov tblp,a ; to the last page or the page that tbhp pointed
mov a,0Fh ; initialise high table pointer
mov tbhp,a
:
:
tabrd tempreg1 ; transfers value in table referenced by table pointer data at program
; memory address "0F06H" 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 "0F05H" 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 0F00h ; sets initial address of program memory
dc 00Ah, 00Bh, 00Ch, 00Dh, 00Eh, 00Fh, 01Ah, 01Bh
:
:

```

### In Circuit Programming – ICP

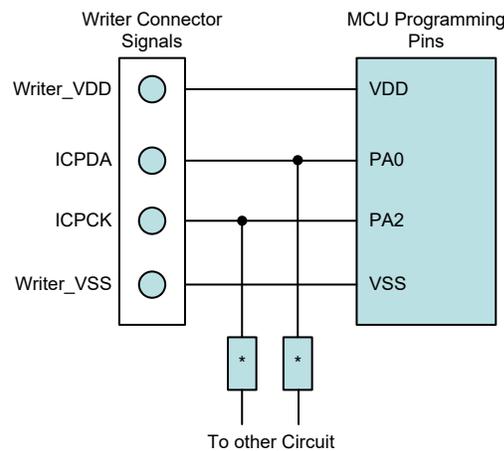
The provision of Flash type Program Memory provides the user with a means of convenient and easy upgrades and modifications to their programs on the same device. As an additional convenience, Holtek has provided a means of programming the microcontroller in-circuit using a 4-pin interface. This provides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

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

Holtek Writer Pins	MCU Programming Pins	Function
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 to ensure that no other outputs are connected to these two pins.



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

### On-Chip Debug Support – OCDS

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
GND	VSS	Ground

## Data Memory

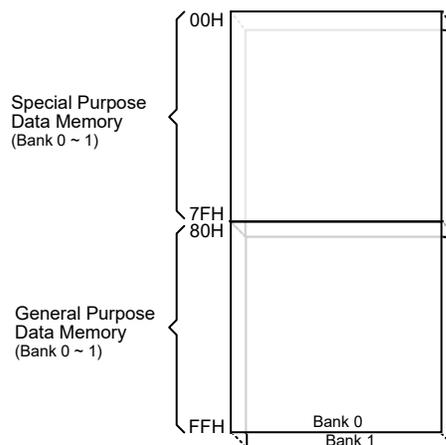
The Data Memory is a volatile area of 8-bit wide RAM internal memory and is the location where temporary information is stored.

### 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, 1	256×8	0: 80H~FFH 1: 80H~FFH



**Data Memory Structure**

### **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	PDS0	
02H	IAR1		42H	CTM0C0	
03H	MP1		43H	CTM0C1	
04H	BP		44H	CTM0DL	
05H	ACC		45H	CTM0DH	
06H	PCL		46H	CTM0AL	
07H	TBLP		47H	CTM0AH	
08H	TBLH		48H	CTM1C0	
09H	TBHP		49H	CTM1C1	
0AH	STATUS		4AH	CTM1DL	
0BH	PSC0R		4BH	CTM1DH	
0CH	PSC1R	IFS	4CH	CTM1AL	
0DH	TB0C	LVRC	4DH	CTM1AH	
0EH	TB1C	LVDC	4EH	CTM2C0	
0FH	RSTFC		4FH	CTM2C1	
10H	SLEWC		50H	CTM2DL	
11H	SCC		51H	CTM2DH	
12H	HIRCC		52H	CTM2AL	
13H	LXTC		53H	CTM2AH	
14H	PA		54H	STMC0	
15H	PAC		55H	STMC1	
16H	PAPU		56H	STMDL	
17H	PAWU		57H	STMDH	
18H	PB		58H	STMAL	
19H	PBC		59H	STMAH	
1AH	PBPU		5AH	PTMC0	
1BH	PBNS		5BH	PTMC1	
1CH	PC		5CH	PTMDL	
1DH	PCC		5DH	PTMDH	
1EH	PCPU		5EH	PTMAL	
1FH	PCNS		5FH	PTMAH	
20H	PD		60H	PTMRPL	
21H	PDC		61H	PTMRPH	
22H	PDPU		62H	USR	
23H	PDNS		63H	UCR1	
24H	SLEDC0		64H	UCR2	
25H	SLEDC1		65H	UCR3	
26H	SLEDC2		66H	BRDH	
27H	SADC0		67H	BRDL	
28H	SADC1		68H	UFCR	
29H	SADOH		69H	TXR_RXR	
2AH	SADOL		6AH	RxCNT	
2BH	INTEG		6BH	FAFHC	
2CH	INTC0		6CH	TKTMR	
2DH	INTC1		6DH	TKC0	
2EH	INTC2		6EH	TKC1	
2FH	INTC3		6FH	TK16DL	
30H	MF10		70H	TK16DH	
31H	MF11		71H	TKM0C0	
32H	MF12		72H	TKM0C1	
33H	EEA		73H	TKM016DL	
34H	EED		74H	TKM016DH	
35H	OCPC0		75H	TKM0ROL	
36H	OCPC1		76H	TKM0ROH	
37H	OCPDA		77H	TKM1C0	
38H	OCPOCAL		78H	TKM1C1	
39H	OCPCCAL		79H	TKM116DL	
3AH	PAS0		7AH	TKM116DH	
3BH	PAS1		7BH	TKM1ROL	
3CH	PBS0		7CH	TKM1ROH	
3DH	PBS1		7DH	WDTC	
3EH	PCS0		7EH	ORMC	
3FH	PCS1		7FH	FADJ	

□ : 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 section. However, several registers require a separate description in this section.

### Indirect Addressing Registers – IAR0, IAR1

The Indirect Addressing Registers, IAR0 and IAR1, although having their locations in normal RAM register space, do not actually physically exist as normal registers. The method of indirect addressing for RAM data manipulation uses these Indirect Addressing Registers and Memory Pointers, in contrast to direct memory addressing, where the actual memory address is specified. Actions on the IAR0 and IAR1 registers will result in no actual read or write operation to these registers but rather to the memory location specified by their corresponding Memory Pointers, MP0 or MP1. Acting as a pair, IAR0 and MP0 can together access data 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 Example

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

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

### Bank Pointer – BP

For this device, the Data Memory is divided into two banks, Bank0 and Bank1. Selecting the required Data Memory area is achieved using the Bank Pointer. Bit 0 of the Bank Pointer is used to select Data Memory Bank 0 or 1.

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

#### • BP Register

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

Bit 7~1 Unimplemented, read as “0”  
 Bit 0 **DMBP0**: Data Memory Bank Selection  
 0: Bank 0  
 1: Bank 1

### 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.

### 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” instruction, allowing for easy table data pointing and reading. TBLH is the location where the high order byte of the table data is stored after a table read data instruction has been executed. Note that the lower order table data byte is transferred to a user defined location.

### Status Register – STATUS

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

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

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

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

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

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

• **STATUS Register**

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

“x”: unknown

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

## 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. Unlike the Program Memory and Data Memory, the EEPROM Data Memory is not directly mapped and is therefore not directly accessible in the same way as the other types of memory. Read and Write operations to the EEPROM are carried out in single byte operations using an address and data register in 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, cannot be directly addressed directly and 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								
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 operation are selected through the relevant control registers.

### Oscillator Overview

In addition to being the source of the main system clock the oscillators also provide clock sources for 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.

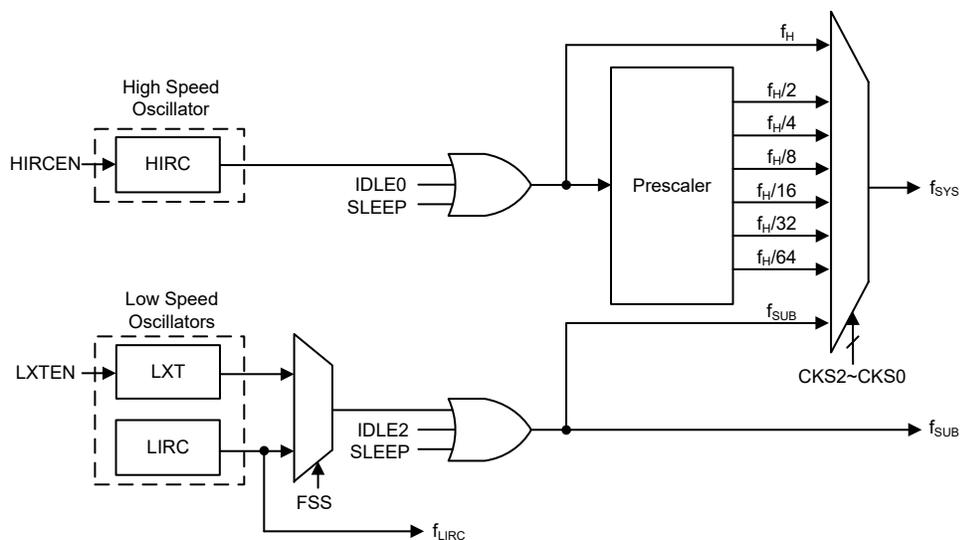
Type	Name	Freq.	Pins
Internal High Speed RC Oscillator	HIRC	8MHz	—
Internal Low Speed RC Oscillator	LIRC	32kHz	—
External Low Speed Crystal	LXT	32.768kHz	XT1/XT2
PTM Dedicated RC Oscillator	PTMOSC	12/14MHz (adjustable)	—

Oscillator Types

### System Clock Configurations

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

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



**System Clock Configurations**

### 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.

### External 32.768kHz Crystal Oscillator – LXT

The External 32.768 kHz Crystal System Oscillator is one of the low frequency oscillator choices, which is selected via a software control bit, FSS. This clock source has a fixed frequency of 32.768kHz and requires a 32.768kHz crystal to be connected between pins XT1 and XT2. The external resistor and capacitor components connected to the 32.768kHz crystal are necessary to provide oscillation. For applications where precise frequencies are essential, these components may be required to provide frequency compensation due to different crystal manufacturing tolerances. After the LXT oscillator is enabled by setting the LXTEN bit to 1, there is a time delay associated with the LXT oscillator waiting for it to start-up.

When the microcontroller enters the SLEEP or IDLE Mode, the system clock is switched off to stop microcontroller activity and to conserve power. However, in many microcontroller applications it may be necessary to keep the internal timers operational even when the microcontroller is in the SLEEP or IDLE Mode. To do this, another clock, independent of the system clock, must be provided.

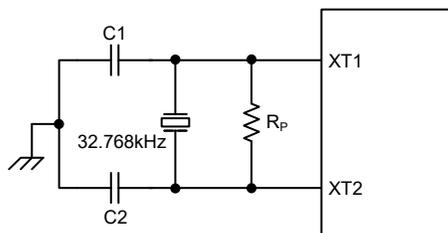
However, for some crystals, to ensure oscillation and accurate frequency generation, it is necessary to add two small value external capacitors, C1 and C2. The exact values of C1 and C2 should be selected in consultation with the crystal or resonator manufacturer's specification. The external parallel feedback resistor,  $R_p$ , is required.

The pin-shared software control bits determine if the XT1/XT2 pins are used for the LXT oscillator or as I/O or other pin-shared functional pins.

- If the LXT oscillator is not used for any clock source, the XT1/XT2 pins can be used as normal I/O or other pin-shared functional pins.

- If the LXT oscillator is used for any clock source, the 32.768kHz crystal should be connected to the XT1/XT2 pins.

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



- Note: 1.  $R_p$ , C1 and C2 are required.  
2. Although not shown XT1/XT2 pins have a parasitic capacitance of around 7pF.

**External LXT Oscillator**

LXT Oscillator C1 and C2 Values		
Crystal Oscillator Frequency	C1	C2
32.768kHz	10pF	10pF
Note: 1. C1 and C2 values are for guidance only. 2. $R_p=5M\sim 10M\Omega$ is recommended.		

**32.768kHz Oscillator Recommended Capacitor Values**

**LXT Oscillator Low Power Function**

The LXT oscillator can function in one of two modes, the Quick Start Mode and the Low Power Mode. The mode selection is executed using the LXTSP bit in the LXTC register.

LXTSP Bit	LXT Operating Mode
0	Low-power
1	Quick Start

When the LXTSP bit is set to high, the LXT Quick Start Mode will be enabled. In the Quick Start Mode the LXT oscillator will power up and stabilise quickly. However, after the LXT oscillator has fully powered up it can be placed into the Low-power mode by clearing the LXTSP bit to zero. The oscillator will continue to run but with reduced current consumption, as the higher current consumption is only required during the LXT oscillator start-up. It is important to note that the LXT operating mode switching must be properly controlled before the LXT oscillator clock is selected as the system clock source. Once the LXT oscillator clock is selected as the system clock source using the CKS bit field and FSS bit in the SCC register, the LXT oscillator operating mode can not be changed.

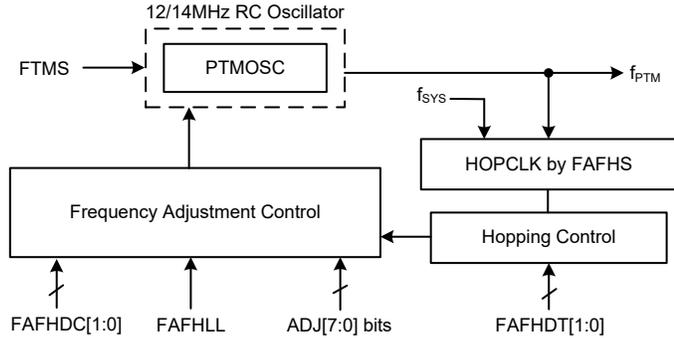
It should be noted that, no matter what condition the LXTSP bit is set to, the LXT oscillator will always function normally. The only difference is that it will take more time to start up if in the Low-power mode.

**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.

**PTM Dedicated RC Oscillator – PTMOSC**

The PTM Dedicated Oscillator is also a fully integrated RC oscillator requiring no external components. This internal oscillator has two frequencies of 12/14MHz which can be fine-adjusted by changing the ADJ[7:0] bits field value. 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.



The ADJ[7:0] bit field in the FADJ register is used to adjust the PTMOSC oscillation frequency. The PTMOSC oscillator is supposed to have two frequencies of 12/14MHz with the default ADJ[7:0] field value, 10000000B. The greater value the ADJ[7:0] field is written, the lower frequency the PTMOSC oscillator has. The PTMOSC oscillator will have a maximum adjusted frequency when the ADJ[7:0] field is set to 00000000B. Note that a certain time delay for the PTMOSC oscillator stabilization should be allowed when the PTMOSC oscillation frequency is changed by configuring the ADJ[7:0] field. The new PTMOSC frequency cannot be used until the updated frequency is stable.

**• FADJ Register**

Bit	7	6	5	4	3	2	1	0
Name	ADJ7	ADJ6	ADJ5	ADJ4	ADJ3	ADJ2	ADJ1	ADJ0
R/W								
POR	1	0	0	0	0	0	0	0

Bit 7~0 **ADJ7~ADJ0**: PTMOSC frequency adjustment control bit 7 ~ bit 0

**• FAFHC Register**

Bit	7	6	5	4	3	2	1	0
Name	FTMEN	FTMS	FAFHS	FAFHLL	FAFHDT1	FAFHDT0	FAFHDC1	FAFHDC0
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 **FTMEN**: PTMOSC oscillator enable control

- 0: Disable
- 1: Enable

The PTMOSC oscillator is fixed to be disabled in the IDLE or SLEEP mode. When the device wakes up, the PTMOSC oscillator will resume normal operation.

Bit 6 **FTMS**: PTMOSC frequency section

- 0: 12MHz
- 1: 14MHz

Before setting this bit, the FAFHDT1~FAFHDT0 bits must be fixed at 00.

- Bit 5      **FAFHS**: Auto Frequency Hopping frequency source clock  
           0:  $f_{SYS}$   
           1:  $f_{PTM}=12/14MHz$   
 Before setting this bit, the FAFHDT1~FAFHDT0 bits must be fixed at 00.
- Bit 4      **FAFHLL**: Auto Frequency Hopping frequency loop level  
           0: 3 level  
           1: 5 level  
 Before setting this bit, the FAFHDT1~FAFHDT0 bits must be fixed at 00.
- Bit 3~2    **FAFHDT1~FAFHDT0**: Auto Frequency Hopping gap time  
           00: AFH mode disable  
           01: AFH mode enable,  $f_{SYS}$  or  $f_{PTM}/256$   
           10: AFH mode enable,  $f_{SYS}$  or  $f_{PTM}/512$   
           11: AFH mode enable,  $f_{SYS}$  or  $f_{PTM}/1024$   
 The AFH mode is fixed to be disabled in the IDLE or SLEEP mode. When the device wakes up, the AFH mode will resume normal operation.
- Bit 1~0    **FAFHDC1~FAFHDC0**: Auto Frequency Hopping range selection  
           00: ADJ+2  
           01: ADJ+4  
           10: ADJ+6  
           11: ADJ+8  
 Before setting this bit, the FAFHDT1~FAFHDT0 bits must be fixed at 00.

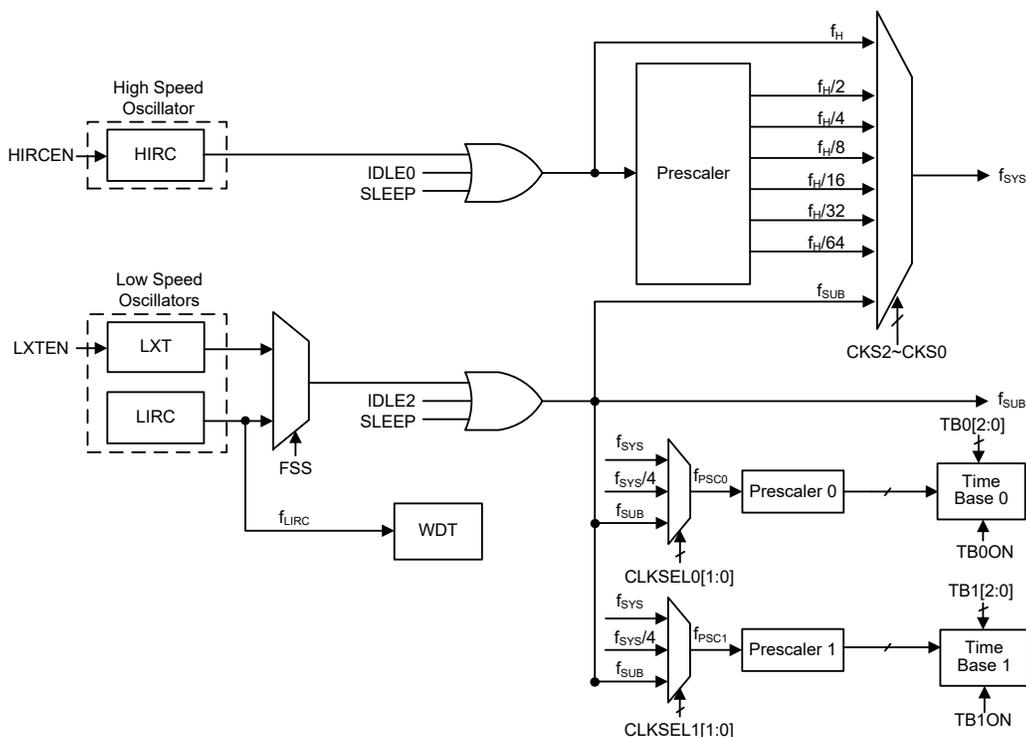
## Operating Modes and System Clocks

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 can be sourced from the internal clock  $f_{SUB}$ . If  $f_{SUB}$  is selected then it can be sourced by either the LXT or the LIRC oscillator, selected via configuring the FSS bit in the SCC register. The other choice, which is a divided version of the high speed system oscillator has a range of  $f_H/2 \sim f_H/64$ .



Note: 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.

### Device Clock Configurations

### 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 or LXT oscillator determined by the FSS bit in the SCC register.

### 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 registers, SCC, HIRCC and LXTC, are used to control the system clock and the corresponding oscillator configurations.

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

**System Operating Mode Control Register List**

• **SCC Register**

Bit	7	6	5	4	3	2	1	0
Name	CKS2	CKS1	CKS0	—	—	FSS	FHIDEN	FSIDEN
R/W	R/W	R/W	R/W	—	—	R/W	R/W	R/W
POR	0	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 4~3 Unimplemented, read as “0”

Bit 2 **FSS**: Low frequency clock selection

0: LIRC  
 1: LXT

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

0: Disable  
 1: Enable

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

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

0: Disable  
 1: Enable

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

• **HIRCC Register**

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

• **LXTC Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	LXTSP	LXTF	LXTEN
R/W	—	—	—	—	—	R/W	R	R/W
POR	—	—	—	—	—	0	0	0

Bit 7~3      Unimplemented, read as “0”

Bit 2      **LXTSP**: LXT Quick Start Control  
0: Disable  
1: Enable

This bit is used to control whether the LXT oscillator is operating in the low power or quick start mode. When the LXTSP bit is set to 1, the LXT oscillator will oscillate quickly but consume more power. If the LXTSP bit is cleared to 0, the LXT oscillator will consume less power but take longer time to stabilise. It is important to note that this bit can not be changed after the LXT oscillator is selected as the system clock source using the CKS2~CKS0 and FSS bits in the SCC register.

Bit 1      **LXTF**: LXT oscillator stable flag  
0: LXT unstable  
1: LXT stable

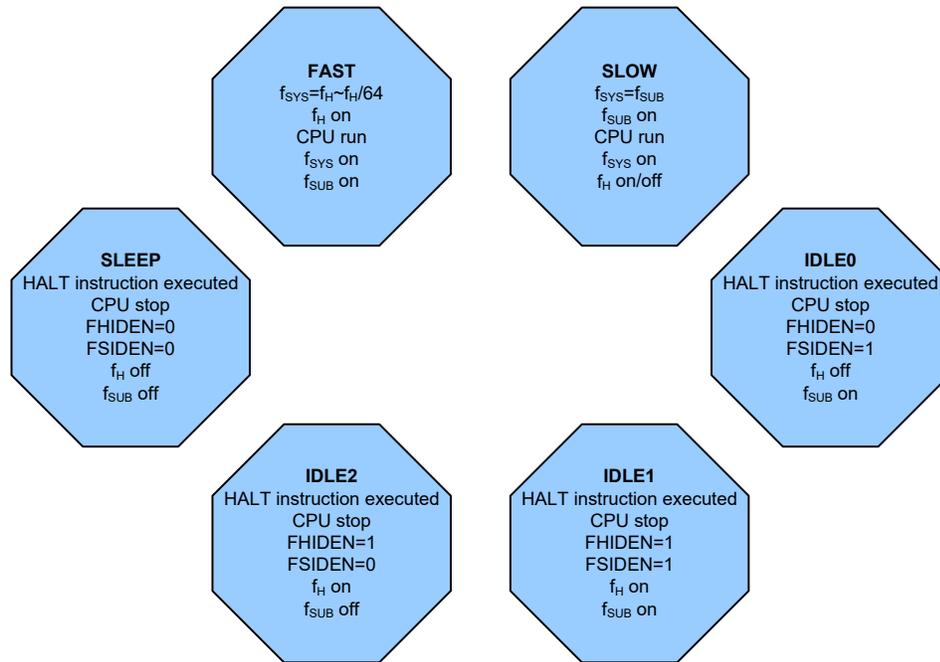
This bit is used to indicate whether the LXT oscillator is stable or not. When the LXTEN bit is set to 1 to enable the LXT oscillator, the LXTF bit will first be cleared to 0 and then set to 1 after the LXT oscillator is stable.

Bit 0      **LXTEN**: LXT 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.

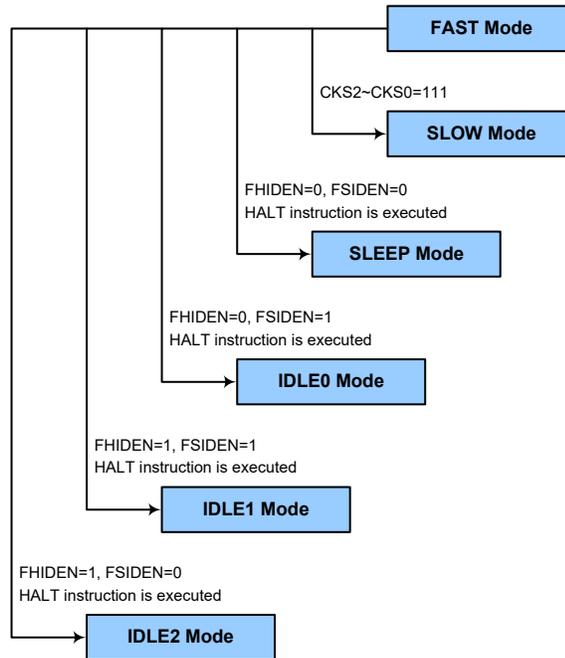
In simple terms, Mode Switching between the FAST Mode and SLOW Mode is executed using the CKS2~CKS0 bits in the SCC register while Mode Switching from the FAST/SLOW Modes to the SLEEP/IDLE Modes is executed via the HALT instruction. When a HALT instruction is executed, whether the device enters the IDLE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

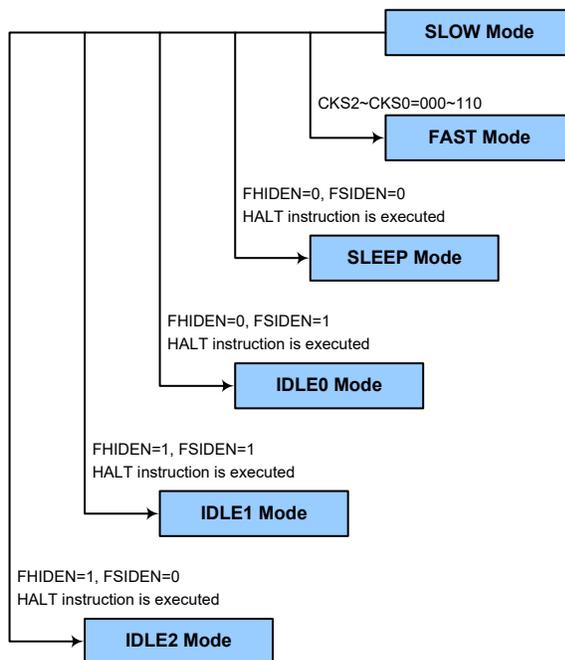
The SLOW Mode is sourced from the LXT or LIRC oscillator determined by the FSS bit in the SCC register and therefore requires the selected oscillator to be stable before full mode switching occurs.



### SLOW Mode to FAST Mode Switching

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

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 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 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 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 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 LXT or LIRC oscillator has 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 WDT 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 enable/disable control

10101: Disable  
 01010: Enable  
 Others: 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.

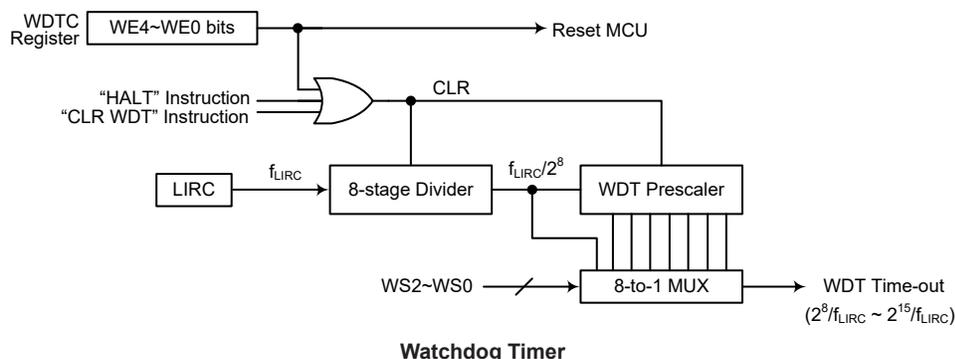
WE4~WE0 Bits	WDT Function
10101B	Disable
01010B	Enable
Any other values	Reset MCU

**Watchdog Timer Enable/Disable 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 ration.



## Reset and Initialisation

A reset function is a fundamental part of any microcontroller ensuring that the device can be set to some predetermined condition irrespective of outside parameters. The most important reset condition is after power is first applied to the 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.

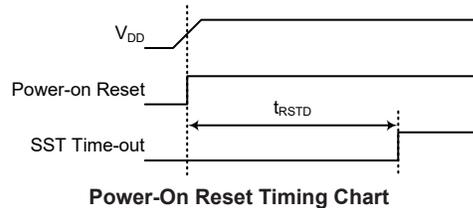
Another reset exists in the form of a Low Voltage Reset, LVR, where a full reset is implemented in situations where the power supply voltage falls below a certain threshold. Another type of reset is when the Watchdog Timer overflows and resets the microcontroller. All types of reset operations result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

### Reset Functions

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

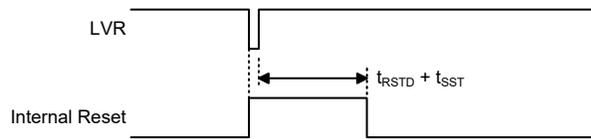
#### Power-on Reset

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. The LVR function is always enabled with a specific LVR voltage  $V_{LVR}$ . If the supply voltage of the device drops to within a range of  $0.9V \sim V_{LVR}$  such as might occur when changing the battery, the LVR will automatically reset the device internally and the LVRF bit in the RSTFC register will also be set high. For a valid LVR signal, a low supply voltage, i.e., a voltage in the range between  $0.9V \sim V_{LVR}$  must exist for a time greater than that specified by  $t_{LVR}$  in the LVD&LVR Electrical Characteristics. If the low supply voltage state does not exceed this value, the LVR will ignore the low supply voltage and will not perform a reset function. The actual  $V_{LVR}$  value can be selected by the LVS7~LVS0 bits in the LVRC register. If the LVS7~LVS0 bits are changed to some certain values by the environmental noise or software setting, 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 01010101B. Note that the LVR function will be automatically disabled when the device enters the IDLE/SLEEP mode.



**Low Voltage Reset Timing Chart**

• **LVRC Register**

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

Bit 7~0 **LVS7~LVS0**: LVR Voltage Select control

01010101: 2.1V

00110011: 2.55V

10011001: 3.15V

10101010: 3.8V

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

When an actual low voltage condition occurs, as specified by one of the four defined LVR voltage values above, an MCU reset will be generated. The reset operation will be activated after the low voltage condition keeps more than a  $t_{LVR}$  time. In this situation the register contents will remain the same after such a reset occurs. Any register value, other than the four defined LVR values above, will also result in the generation of an MCU reset. The reset operation will be activated after a delay time,  $t_{RESET}$ . However in this situation the register contents will be reset to the POR value.

• **RSTFC Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	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

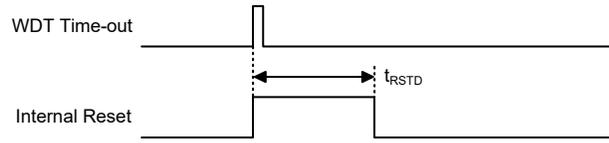
This bit is set to 1 by the LVRC control register contains any undefined LVR voltage register values. This in effect acts like a software-reset function. Note that this bit can only be cleared to zero by the application program.

Bit 0 **WRF**: WDTC register software reset flag

Refer to the Watchdog Timer Control Register section.

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

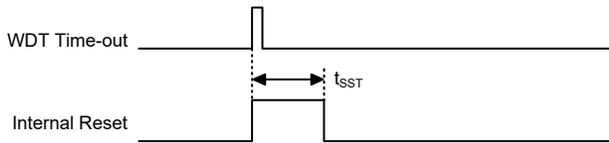
The Watchdog Time-out Reset during normal operation in the FAST or SLOW mode is the same as a LVR reset except that the Watchdog time-out flag TO will be set to “1”.



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

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

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



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

**Reset Initial Conditions**

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

TO	PDF	RESET Conditions
0	0	Power-on reset
u	u	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

Note: “u” stands for unchanged

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

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

The different kinds of resets all affect the internal registers of the microcontroller in different ways. To ensure reliable continuation of normal program execution after a reset occurs, it is important to know what condition the microcontroller is in after a particular reset occurs. The following table describes how each type of reset affects each of the microcontroller internal registers. 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
TBLH	x xxx x xxx	u u u u u u u u	u u u u u u u u
TBHP	- - - - x xxx	- - - - u u u u	- - - - u u u u
STATUS	- - 0 0 x xxx	- - 1 u u u u u	- - 1 1 u u u u
PSC0R	- - - - - - 0 0	- - - - - - 0 0	- - - - - - u u
PSC1R	- - - - - - 0 0	- - - - - - 0 0	- - - - - - u u
TB0C	0 - - - - 0 0 0	0 - - - - 0 0 0	u - - - - u u u
TB1C	0 - - - - 0 0 0	0 - - - - 0 0 0	u - - - - u u u
RSTFC	- - - - - x 0 0	- - - - - u u u	- - - - - u u u
SLEWC	- - - - - - 0 0	- - - - - - 0 0	- - - - - - u u
SCC	0 0 0 - - 0 0 0	0 0 0 - - 0 0 0	u u u - - u u u
HIRCC	- - - - - - 0 1	- - - - - - 0 1	- - - - - - u u
LXTC	- - - - - 0 0 0	- - - - - 0 0 0	- - - - - u u u
PA	1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1	u u u u u u u u
PAC	1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1	u u u u u u u u
PAPU	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	u u u u u u u u
PAWU	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	u u u u u u u u
PB	1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1	u u u u u u u u
PBC	1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1	u u u u u u u u
PBPU	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	u u u u u u u u
PBNS	- - - - - - 0 0	- - - - - - 0 0	- - - - - - u u
PC	1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1	u u u u u u u u
PCC	1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1	u u u u u u u u
PCPU	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	u u u u u u u u
PCNS	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	u u u u u u u u
PD	- - - - - - 1 1	- - - - - - 1 1	- - - - - - u u
PDC	- - - - - - 1 1	- - - - - - 1 1	- - - - - - u u
PDPU	- - - - - - 0 0	- - - - - - 0 0	- - - - - - u u
PDNS	- - - - - - 0 0	- - - - - - 0 0	- - - - - - u u
SLED0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	u u u u u u u u
SLED1	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	u u u u u u u u
SLED2	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	u u u u u u u u
SADC0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	u u u u u u u u
SADC1	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	u u u u u u u u
SADOH	x xxx x xxx	x xxx x xxx	u u u u u u u u (ADRF5=0)
			- - - - u u u u (ADRF5=1)

Register	Power On Reset	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)
SADOL	x x x x - - - -	x x x x - - - -	u u u u - - - - (ADRF5=0)
			u u u u u u u u (ADRF5=1)
INTEG	- - - - 0 0 0 0	- - - - 0 0 0 0	- - - - u u u u
INTC0	- 0 0 0 0 0 0 0 0	- 0 0 0 0 0 0 0 0	- u u u u u u u u
INTC1	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	u u u u u u u u
INTC2	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	u u u u u u u u
INTC3	- - 0 0 - - 0 0	- - 0 0 - - 0 0	- - u u - - u u
MFIO	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	u u u u u u u u
MF11	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	u u u u u u u u
MF12	- - 0 0 - - 0 0	- - 0 0 - - 0 0	- - u u - - u u
EEA	- - - 0 0 0 0 0 0	- - - 0 0 0 0 0 0	- - - u u u u u
EED	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	u u u u u u u u
OCPC0	0 0 0 0 0 - - 0	0 0 0 0 0 - - 0	u u u u u - - u
OCPC1	- - 0 0 0 0 0 0	- - 0 0 0 0 0 0	- - u u u u u u
OCPDA	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	u u u u u u u u
OCPOCAL	0 0 1 0 0 0 0 0	0 0 1 0 0 0 0 0	u u u u u u u u
OCPCCAL	0 0 0 1 0 0 0 0	0 0 0 1 0 0 0 0	u u u u u u u u
PAS0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	u u u u u u u u
PAS1	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	u u u u u u u u
PBS0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	u u u u u u u u
PBS1	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	u u u u u u u u
PCS0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	u u u u u u u u
PCS1	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	u u u u u u u u
PDS0	- - - - 0 0 0 0	- - - - 0 0 0 0	- - - - u u u u
CTM0C0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	u u u u u u u u
CTM0C1	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	u u u u u u u u
CTM0DL	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	u u u u u u u u
CTM0DH	- - - - - 0 0	- - - - - 0 0	- - - - - u u
CTM0AL	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	u u u u u u u u
CTM0AH	- - - - - 0 0	- - - - - 0 0	- - - - - u u
CTM1C0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	u u u u u u u u
CTM1C1	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	u u u u u u u u
CTM1DL	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	u u u u u u u u
CTM1DH	- - - - - 0 0	- - - - - 0 0	- - - - - u u
CTM1AL	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	u u u u u u u u
CTM1AH	- - - - - 0 0	- - - - - 0 0	- - - - - u u
CTM2C0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	u u u u u u u u
CTM2C1	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	u u u u u u u u
CTM2DL	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	u u u u u u u u
CTM2DH	- - - - - 0 0	- - - - - 0 0	- - - - - u u
CTM2AL	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	u u u u u u u u
CTM2AH	- - - - - 0 0	- - - - - 0 0	- - - - - u u
STMC0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	u u u u u u u u
STMC1	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	u u u u u u u u
STMDL	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	u u u u u u u u
STMDH	- - - - - 0 0	- - - - - 0 0	- - - - - u u

Register	Power On Reset	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)
STMAL	0000 0000	0000 0000	uuuu uuuu
STMAH	---- --00	---- --00	---- --uu
PTMC0	0000 0---	0000 0---	uuuu u---
PTMC1	0000 0000	0000 0000	uuuu uuuu
PTMDL	0000 0000	0000 0000	uuuu uuuu
PTMDH	---- --00	---- --00	---- --uu
PTMAL	0000 0000	0000 0000	uuuu uuuu
PTMAH	---- --00	---- --00	---- --uu
PTMRPL	0000 0000	0000 0000	uuuu uuuu
PTMRPH	---- --00	---- --00	---- --uu
USR	0000 1011	0000 1011	uuuu uuuu
UCR1	0000 00x0	0000 00x0	uuuu uuuu
UCR2	0000 0000	0000 0000	uuuu uuuu
UCR3	---- ---0	---- ---0	---- ---u
BRDH	0000 0000	0000 0000	uuuu uuuu
BRDL	0000 0000	0000 0000	uuuu uuuu
UFCR	--00 0000	--00 0000	--uu uuuu
TXR_RXR	xxxx xxxx	xxxx xxxx	uuuu uuuu
RxCNT	---- -000	---- -000	---- -uuu
FAFHC	0000 0000	0000 0000	uuuu uuuu
TKTMR	0000 0000	0000 0000	uuuu uuuu
TKC0	-000 0000	-000 0000	-uuu uuuu
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 0000	0-00 0000	u-uu uuuu
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
TKM1C0	0000 0000	0000 0000	uuuu uuuu
TKM1C1	0-00 0000	0-00 0000	u-uu uuuu
TKM116DL	0000 0000	0000 0000	uuuu uuuu
TKM116DH	0000 0000	0000 0000	uuuu uuuu
TKM1ROL	0000 0000	0000 0000	uuuu uuuu
TKM1ROH	---- --00	---- --00	---- --uu
WDTC	0101 0011	0101 0011	uuuu uuuu
ORMC	0000 0000	0000 0000	0000 0000
FADJ	1000 0000	1000 0000	uuuu uuuu
IFS	---- ---0	---- ---0	---- ---u
LVRC	0101 0101	0101 0101	uuuu uuuu
LVDC	--00 0000	--00 0000	--uu uuuu
EEC	0--- 0000	0--- 0000	u--- uuuu

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

## Input / Output Ports

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

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

Register Name	Bit							
	7	6	5	4	3	2	1	0
PA	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0
PAC	PAC7	PAC6	PAC5	PAC4	PAC3	PAC2	PAC1	PAC0
PAPU	PAPU7	PAPU6	PAPU5	PAPU4	PAPU3	PAPU2	PAPU1	PAPU0
PAWU	PAWU7	PAWU6	PAWU5	PAWU4	PAWU3	PAWU2	PAWU1	PAWU0
PB	PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0
PBC	PBC7	PBC6	PBC5	PBC4	PBC3	PBC2	PBC1	PBC0
PBPU	PBPU7	PBPU6	PBPU5	PBPU4	PBPU3	PBPU2	PBPU1	PBPU0
PC	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0
PCC	PCC7	PCC6	PCC5	PCC4	PCC3	PCC2	PCC1	PCC0
PCPU	PCPU7	PCPU6	PCPU5	PCPU4	PCPU3	PCPU2	PCPU1	PCPU0
PD	—	—	—	—	—	—	PD1	PD0
PDC	—	—	—	—	—	—	PDC1	PDC0
PDPU	—	—	—	—	—	—	PDPU1	PDPU0

“—”: 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 register, and are implemented using weak PMOS transistors.

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

#### • PxPU Register

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

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

0: Disable

1: Enable

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

### Port A Wake-up

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

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

#### • PAWU Register

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

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

#### • PxC Register

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

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

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

### Slew Rate Control

The PA1 port can be setup to have a choice of various slew rate using the SLEWC register. Refer to the Slew Rate Control Characteristics section to obtain the exact value.

• **SLEWC Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	SLEWC1	SLEWC0
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2 Unimplemented, read as “0”

Bit 1~0 **SLEWC1~SLEWC0**: PA1 output slew rate selection

00: Slew rate = Level 0

01: Slew rate = Level 1

10: Slew rate = Level 2

11: Slew rate = Level 3

Note: Users should refer to the Slew Rate Control Characteristics section to obtain the exact value for different applications.

**Source Current Selection**

The device supports different source current driving capability for each I/O port except PA1 and PA3. With the corresponding selection registers, SLEDC0, SLEDC1 and SLEDC2, each I/O port can support four levels of the source current driving capability. Users should refer to the Input/Output Characteristics section to obtain the exact value for different applications.

Register Name	Bit							
	7	6	5	4	3	2	1	0
SLEDC0	SLEDC07	SLEDC06	SLEDC05	SLEDC04	SLEDC03	SLEDC02	SLEDC01	SLEDC00
SLEDC1	SLEDC17	SLEDC16	SLEDC15	SLEDC14	SLEDC13	SLEDC12	SLEDC11	SLEDC10
SLEDC2	SLEDC27	SLEDC26	SLEDC25	SLEDC24	SLEDC23	SLEDC22	SLEDC21	SLEDC20

**I/O Port Source Current Selection Register List**

• **SLEDC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	SLEDC07	SLEDC06	SLEDC05	SLEDC04	SLEDC03	SLEDC02	SLEDC01	SLEDC00
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 **SLEDC07~SLEDC06**: PD1~PD0 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 **SLEDC05~SLEDC04**: 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 3~2 **SLEDC03~SLEDC02**: 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 1~0 **SLEDC01~SLEDC00**: PA2, 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.)

• **SLEDC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	SLEDC17	SLEDC16	SLEDC15	SLEDC14	SLEDC13	SLEDC12	SLEDC11	SLEDC10
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     **SLEDC17~SLEDC16:** PB7~PB6 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     **SLEDC15~SLEDC14:** PB5~PB4 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     **SLEDC13~SLEDC12:** PB3~PB2 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     **SLEDC11~SLEDC10:** PB1~PB0 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.)

• **SLEDC2 Register**

Bit	7	6	5	4	3	2	1	0
Name	SLEDC27	SLEDC26	SLEDC25	SLEDC24	SLEDC23	SLEDC22	SLEDC21	SLEDC20
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     **SLEDC27~SLEDC26:** PC7~PC6 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     **SLEDC25~SLEDC24:** PC5~PC4 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     **SLEDC23~SLEDC22:** PC3~PC2 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     **SLEDC21~SLEDC20:** PC1~PC0 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.)

## I/O Port Sink Current Selection

The device supports different output sink current driving capability for PB0~1, PC0~7 and PD0~1 ports. With the selection registers, PxNS, specific I/O port can support two levels of the sink current driving capability. These sink current selection bits are available when the corresponding pin is configured as a CMOS output. Otherwise, these select bits have no effect. Users should refer to the Input/Output Characteristics section to select the desired output sink current for different applications.

Register Name	Bit							
	7	6	5	4	3	2	1	0
PBNS	—	—	—	—	—	—	PBNS1	PBNS0
PCNS	PCNS7	PCNS6	PCNS5	PCNS4	PCNS3	PCNS2	PCNS1	PCNS0
PDNS	—	—	—	—	—	—	PDNS1	PDNS0

I/O Port Sink Current Selection Register List

### • PBNS Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	PBNS1	PBNS0
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2 Unimplemented, read as “0”

Bit 1 **PBNS1**: PB1 sink current selection (NMOS adjust)  
 0: Sink current = Level 0 (Min.)  
 1: Sink current = Level 1 (Max.)

Bit 0 **PBNS0**: PB0 sink current selection (NMOS adjust)  
 0: Sink current = Level 0 (Min.)  
 1: Sink current = Level 1 (Max.)

### • PCNS Register

Bit	7	6	5	4	3	2	1	0
Name	PCNS7	PCNS6	PCNS5	PCNS4	PCNS3	PCNS2	PCNS1	PCNS0
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 **PCNS7**: PC7 sink current selection (NMOS adjust)  
 0: Sink current = Level 0 (Min.)  
 1: Sink current = Level 1 (Max.)

Bit 6 **PCNS6**: PC6 sink current selection (NMOS adjust)  
 0: Sink current = Level 0 (Min.)  
 1: Sink current = Level 1 (Max.)

Bit 5 **PCNS5**: PC5 sink current selection (NMOS adjust)  
 0: Sink current = Level 0 (Min.)  
 1: Sink current = Level 1 (Max.)

Bit 4 **PCNS4**: PC4 sink current selection (NMOS adjust)  
 0: Sink current = Level 0 (Min.)  
 1: Sink current = Level 1 (Max.)

Bit 3 **PCNS3**: PC3 sink current selection (NMOS adjust)  
 0: Sink current = Level 0 (Min.)  
 1: Sink current = Level 1 (Max.)

- Bit 2      **PCNS2:** PC2 sink current selection (NMOS adjust)  
0: Sink current = Level 0 (Min.)  
1: Sink current = Level 1 (Max.)
- Bit 1      **PCNS1:** PC1 sink current selection (NMOS adjust)  
0: Sink current = Level 0 (Min.)  
1: Sink current = Level 1 (Max.)
- Bit 0      **PCNS0:** PC0 sink current selection (NMOS adjust)  
0: Sink current = Level 0 (Min.)  
1: Sink current = Level 1 (Max.)

• **PDNS Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	PDNS1	PDNS0
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

- Bit 7~2      Unimplemented, read as “0”
- Bit 1      **PDNS1:** PD1 sink current selection (NMOS adjust)  
0: Sink current = Level 0 (Min.)  
1: Sink current = Level 1 (Max.)
- Bit 0      **PDNS0:** PD0 sink current selection (NMOS adjust)  
0: Sink current = Level 0 (Min.)  
1: Sink current = Level 1 (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, and Input Function Selection register, labeled as IFS, which can select the desired functions of the multi-function pin-shared pins.

The most important point to note is to make sure that the desired pin-shared function is properly selected and also deselected. For most pin-shared functions, to select the desired pin-shared function, the pin-shared function should first be correctly selected using the corresponding pin-shared control register. After that the corresponding peripheral functional setting should be configured and then the peripheral function can be enabled. However, a special point must be noted for some digital input pins, such as INTn, xTCKn, etc, which share the same pin-shared control configuration with their corresponding general purpose I/O functions when setting the relevant pin-shared control 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	PAS13	PAS12	PAS11	PAS10
PBS0	PBS07	PBS06	PBS05	PBS04	PBS03	PBS02	PBS01	PBS00
PBS1	PBS17	PBS16	PBS15	PBS14	PBS13	PBS12	PBS11	PBS10
PCS0	PCS07	PCS06	PCS05	PCS04	PCS03	PCS02	PCS01	PCS00
PCS1	PCS17	PCS16	PCS15	PCS14	PCS13	PCS12	PCS11	PCS10
PDS0	—	—	—	—	PDS03	PDS02	PDS01	PDS00
IFS	—	—	—	—	—	—	—	IFS0

**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: OCPI
- 10: PA3
- 11: PA3

Note: If PA3&PC7 are set to the OCPI function at the same time, they are both valid and will be short-circuited internally. Please avoid setting to OCPI at the same time in application.

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

- 00: PA2/INT0
- 01: RX/TX
- 10: AN3
- 11: PA2/INT0

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

- 00: PA1
- 01: PTP
- 10: PTPB
- 11: PA1

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

- 00: PA0
- 01: AN2
- 10: VREF
- 11: PA0

• **PAS1 Register**

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

- Bit 7~6     **PAS17~PAS16:** PA7 Pin-shared function selection  
00: PA7/CTCK1  
01: KEY4  
10: PA7/CTCK1  
11: PA7/CTCK1
- Bit 5~4     **PAS15~PAS14:** PA6 Pin-shared function selection  
00: PA6/CTCK0  
01: KEY3  
10: PA6/CTCK0  
11: PA6/CTCK0
- Bit 3~2     **PAS13~PAS12:** PA5 Pin-shared function selection  
00: PA5  
01: KEY2  
10: STPB  
11: PA5
- Bit 1~0     **PAS11~PAS10:** PA4 Pin-shared function selection  
00: PA4/STCK  
01: KEY1  
10: PA4/STCK  
11: PA4/STCK

• **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: XT2  
10: PB3  
11: PB3
- Bit 5~4     **PBS05~PBS04:** PB2 Pin-shared function selection  
00: PB2  
01: XT1  
10: PB2  
11: PB2
- Bit 3~2     **PBS03~PBS02:** PB1 Pin-shared function selection  
00: PB1  
01: AN1  
10: CTP0  
11: PB1
- Bit 1~0     **PBS01~PBS00:** PB0 Pin-shared function selection  
00: PB0  
01: AN0  
10: STP  
11: PB0

• **PBS1 Register**

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

- Bit 7~6     **PBS17~PBS16:** PB7 Pin-shared function selection  
00: PB7  
01: KEY8  
10: PB7  
11: PB7
- Bit 5~4     **PBS15~PBS14:** PB6 Pin-shared function selection  
00: PB6  
01: KEY7  
10: PB6  
11: PB6
- Bit 3~2     **PBS13~PBS12:** PB5 Pin-shared function selection  
00: PB5  
01: KEY6  
10: PB5  
11: PB5
- Bit 1~0     **PBS11~PBS10:** PB4 Pin-shared function selection  
00: PB4/CTCK2  
01: KEY5  
10: PB4/CTCK2  
11: PB4/CTCK2

• **PCS0 Register**

Bit	7	6	5	4	3	2	1	0
Name	PCS07	PCS06	PCS05	PCS04	PCS03	PCS02	PCS01	PCS00
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     **PCS07~PCS06:** PC3 Pin-shared function selection  
00: PC3  
01: AN7  
10: CTP0  
11: PC3
- Bit 5~4     **PCS05~PCS04:** PC2 Pin-shared function selection  
00: PC2  
01: AN6  
10: PC2  
11: PC2
- Bit 3~2     **PCS03~PCS02:** PC1 Pin-shared function selection  
00: PC1/INT1  
01: AN5  
10: RX/TX  
11: PC1/INT1
- Bit 1~0     **PCS01~PCS00:** PC0 Pin-shared function selection  
00: PC0  
01: AN4  
10: TX  
11: PTPB

• **PCS1 Register**

Bit	7	6	5	4	3	2	1	0
Name	PCS17	PCS16	PCS15	PCS14	PCS13	PCS12	PCS11	PCS10
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     **PCS17~PCS16:** PC7 Pin-shared function selection

00: PC7  
01: CTP0  
10: CTP0B  
11: OCPI

Note: If PA3&PC7 are set to the OCPI function at the same time, they are both valid and will be short-circuited internally. Please avoid setting to OCPI at the same time in application.

Bit 5~4     **PCS15~PCS14:** PC6 Pin-shared function selection

00: PC6  
01: CTP1  
10: CTP1B  
11: OCPVR

Bit 3~2     **PCS13~PCS12:** PC5 Pin-shared function selection

00: PC5  
01: CTP2  
10: CTP2B  
11: PC5

Bit 1~0     **PCS11~PCS10:** PC4 Pin-shared function selection

00: PC4  
01: CTP1  
10: OCPCOUT  
11: PC4

• **PDS0 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	PDS03	PDS02	PDS01	PDS00
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0

bit 7~4     Unimplemented, read as “0”

bit 3~2     **PDS03~PDS02:** PD1 pin-shared function selection

00: PD1/PTCK  
01: CTP2  
10: PD1/PTCK  
11: PD1/PTCK

bit 1~0     **PDS01~PDS00:** PD0 pin-shared function selection

00: PD0  
01: CTP1  
10: PD0  
11: PD0

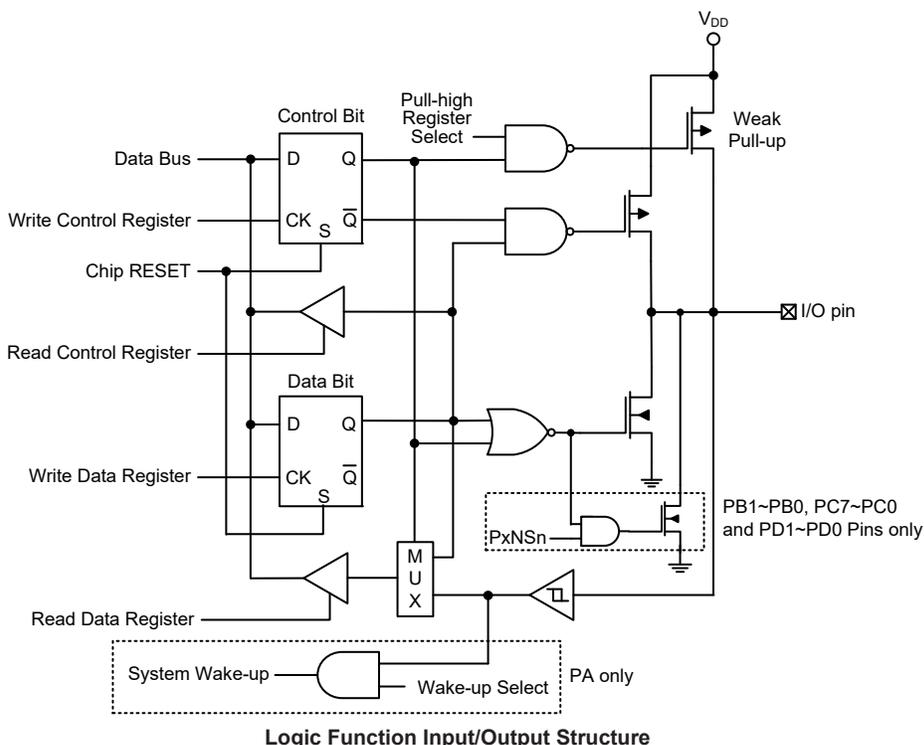
• **IFS Register**

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

Bit 7~1 Unimplemented, read as “0”  
 Bit 0 **IFS0**: RX/TX input source pin selection  
 0: PA2  
 1: PC1

**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 several Timer Modules, abbreviated to the name TM. The TMs are multi-purpose timing units and serve to provide operations such as Timer/Counter, Compare Match Output and Single Pulse Output as well as being the functional unit for the generation of PWM signals. Each of the TMs has two individual interrupts. The addition of input and output pins for each TM ensures that users are provided with timing units with a wide and flexible range of features.

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

### Introduction

The device contains three 10-bit Compact TMs, named CTM0~CTM2, a 10-bit Standard TM named STM and a 10-bit Periodic TM named PTM. Although similar in nature, the different TM types vary in their feature complexity. The common features to the Compact, Standard and Periodic TMs will be described in this section. The detailed operation regarding each of the TM types will be described in corresponding sections. The main features and differences between the three types of TM are summarised in the accompanying table.

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

**TM Function Summary**

### TM Operation

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

### TM Clock Source

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

### TM Interrupts

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

### TM External Pins

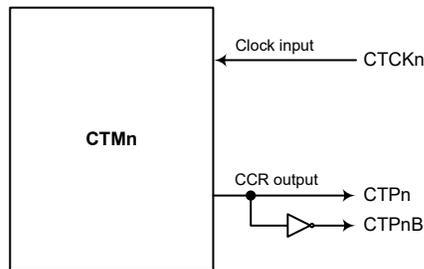
Each of the TMs, irrespective of what type, has one TM input pin, with the label  $xTCKn$ . The  $xTMn$  input pin,  $xTCKn$ , is essentially a clock source for the  $xTM$  and is selected using the  $xTnCK2 \sim xTnCK0$  bits in the  $xTMnC0$  register. This external TM input pin allows an external clock source to drive the internal TM. The  $xTCKn$  input pin can be chosen to have either a rising or falling active edge. The  $STCK$  and  $PTCK$  pins are also used as the external trigger input pin in single pulse output mode for the STM and PTM respectively.

The TMs each have two output pins,  $xTPn$  and  $xTPnB$ . The  $xTPnB$  is the inverted signal of the  $xTPn$  output. 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 output pins are also the pins where the TM generates the PWM output waveform.

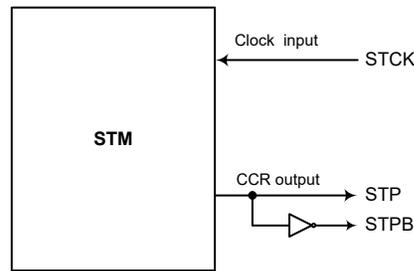
As the TM input/output pins are pin-shared with other functions, the TM input/output function must first be setup using relevant pin-shared function selection registers. The details of the pin-shared function selection are described in the pin-shared function section.

CTMn		STM		PTM	
Input	Output	Input	Output	Input	Output
CTCK0	CTP0, CTP0B	STCK	STP, STPB	PTCK	PTP, PTPB
CTCK1	CTP1, CTP1B				
CTCK2	CTP2, CTP2B				

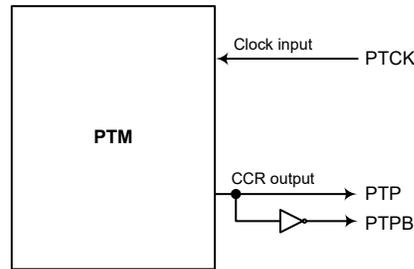
**TM External Pins**



**CTMn Function Pin Block Diagram (n=0~2)**



**STM Function Pin Block Diagram**

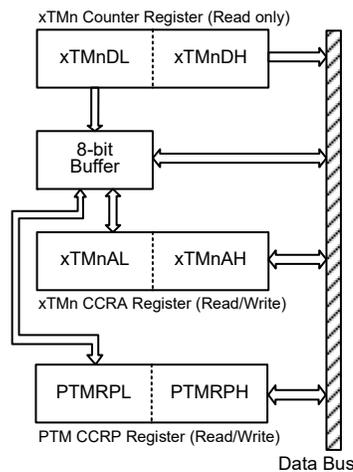


**PTM Function Pin Block Diagram**

**Programming Considerations**

The TM Counter Registers and the Compare CCRA and CCRP registers, being 10-bit, all have a low and high byte structure. The high bytes can be directly accessed, but as the low bytes can only be accessed via an internal 8-bit buffer, reading or writing to these register pairs must be carried out in a specific way. The important point to note is that data transfer to and from the 8-bit buffer and its related low byte only takes place when a write or read operation to its corresponding high byte is executed.

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

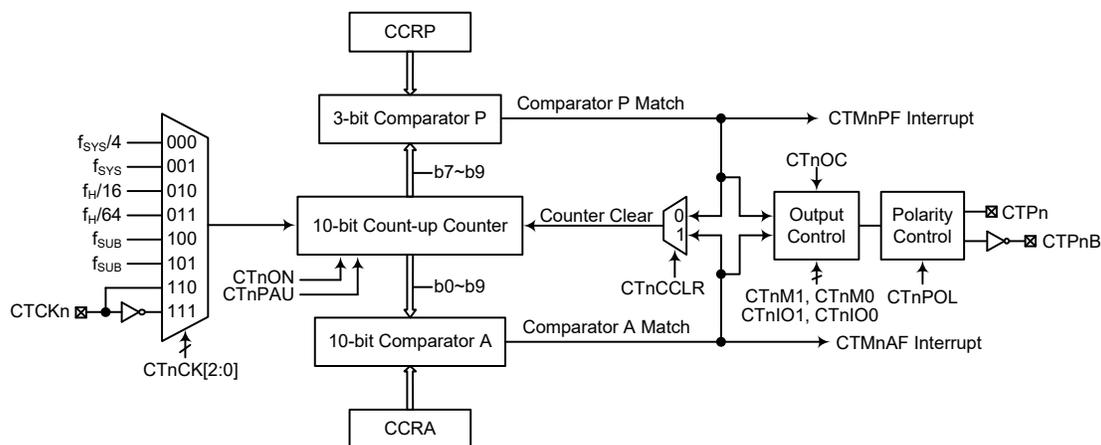


The following steps show the read and write procedures:

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

## Compact Type TM – CTM

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



Note: As the CTMn external pins are pin-shared with other functions, before using the CTMn function, make sure the corresponding pin-shared function registers be set properly.

**Compact Type TM Block Diagram (n=0~2)**

## Compact Type TM Operation

At its core is a 10-bit count-up counter which is driven by a user selectable internal or external clock source. There are also two internal comparators with the names, Comparator A and Comparator P. These comparators will compare the value in the counter with CCRP and CCRA registers. The CCRP is 3-bit wide whose value is compared with the highest three bits in the counter while the CCRA is 10-bit wide 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 CTnON bit from low to high. The counter will also be cleared automatically by a counter overflow or a compare match with one of its associated comparators. When these conditions occur, a CTMn interrupt signal will also usually be generated. The Compact Type TM can operate in a number of different operational modes, can be driven by different clock

sources including an input pin and can also control two output pins. All operating setup conditions are selected using relevant internal registers.

### Compact Type TM Register Description

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

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

**10-bit Compact TM Register List (n=0~2)**

#### • CTMnC0 Register

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

Bit 7 **CTnPAU**: CTMn Counter Pause Control

0: Run  
1: Pause

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

Bit 6~4 **CTnCK2~CTnCK0**: Select CTMn Counter clock

000:  $f_{SYS}/4$   
001:  $f_{SYS}$   
010:  $f_H/16$   
011:  $f_H/64$   
100:  $f_{SUB}$   
101:  $f_{SUB}$   
110: CTCKn rising edge clock  
111: CTCKn falling edge clock

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

Bit 3 **CTnON**: CTMn Counter On/Off Control

0: Off  
1: On

This bit controls the overall on/off function of the CTMn. Setting the bit high enables the counter to run, clearing the bit disables the CTMn. Clearing this bit to zero will stop the counter from counting and turn off the CTMn which will reduce its power consumption. When the bit changes state from low to high the internal counter value

will be reset to zero, however when the bit changes from high to low, the internal counter will retain its residual value until the bit returns high again.

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

- Bit 2~0 **CTnRP2~CTnRP0**: CTMn CCRP 3-bit register, compared with the CTMn Counter bit 9~bit 7 Comparator P Match Period
- 000: 1024 CTMn clocks
  - 001: 128 CTMn clocks
  - 010: 256 CTMn clocks
  - 011: 384 CTMn clocks
  - 100: 512 CTMn clocks
  - 101: 640 CTMn clocks
  - 110: 768 CTMn clocks
  - 111: 896 CTMn clocks

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

• **CTMnC1 Register**

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

- Bit 7~6 **CTnM1~CTnM0**: Select CTMn Operating Mode
- 00: Compare Match Output Mode
  - 01: Undefined
  - 10: PWM Output Mode
  - 11: Timer/Counter Mode

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

- Bit 5~4 **CTnIO1~CTnIO0**: Select CTMn external pin function
- Compare Match Output Mode
    - 00: No change
    - 01: Output low
    - 10: Output high
    - 11: Toggle output
  - PWM Output Mode
    - 00: PWM Output inactive state
    - 01: PWM Output active state
    - 10: PWM Output
    - 11: Undefined
  - Timer/Counter Mode
    - Unused

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

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

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

- Bit 3     **CTnOC:** CTPn 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 CTMn output pin. Its operation depends upon whether CTMn is being used in the Compare Match Output Mode or in the PWM Output Mode. It has no effect if the CTMn is in the Timer/Counter Mode. In the Compare Match Output Mode it determines the logic level of the CTMn output pin before a compare match occurs. In the PWM Output Mode it determines if the PWM signal is active high or active low.

- Bit 2     **CTnPOL:** CTPn Output polarity Control  
           0: Non-invert  
           1: Invert

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

- Bit 1     **CTnDPX:** CTMn PWM 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     **CTnCCLR:** Select CTMn Counter clear condition  
           0: CTMn Comparatror P match  
           1: CTMn 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 CTnCCLR bit set high, the counter will be cleared when a compare match occurs from the Comparator A. When the bit is low, the counter will be cleared when a compare match occurs from the Comparator P or with a counter overflow. A counter overflow clearing method can only be implemented if the CCRP bits are all cleared to zero. The CTnCCLR bit is not used in the PWM Output Mode.

• **CTMnDL Register**

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

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

• **CTMnDH Register**

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

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

• **CTMnAL Register**

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

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

• **CTMnAH Register**

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

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

## Compact Type TM Operating Modes

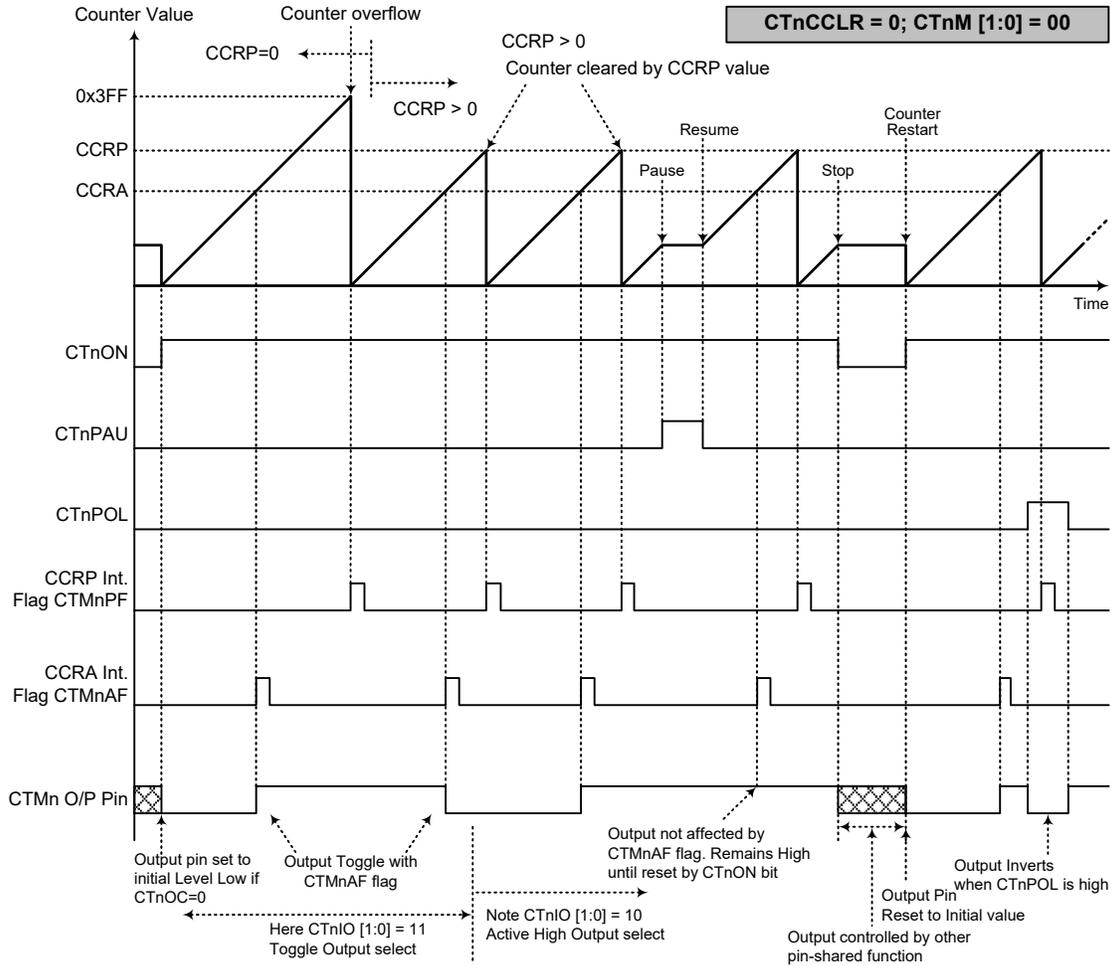
The Compact Type TM can operate in one of three operating modes, Compare Match Output Mode, PWM Output Mode or Timer/Counter Mode. The operating mode is selected using the CTnM1 and CTnM0 bits in the CTMnC1 register.

### Compare Match Output Mode

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

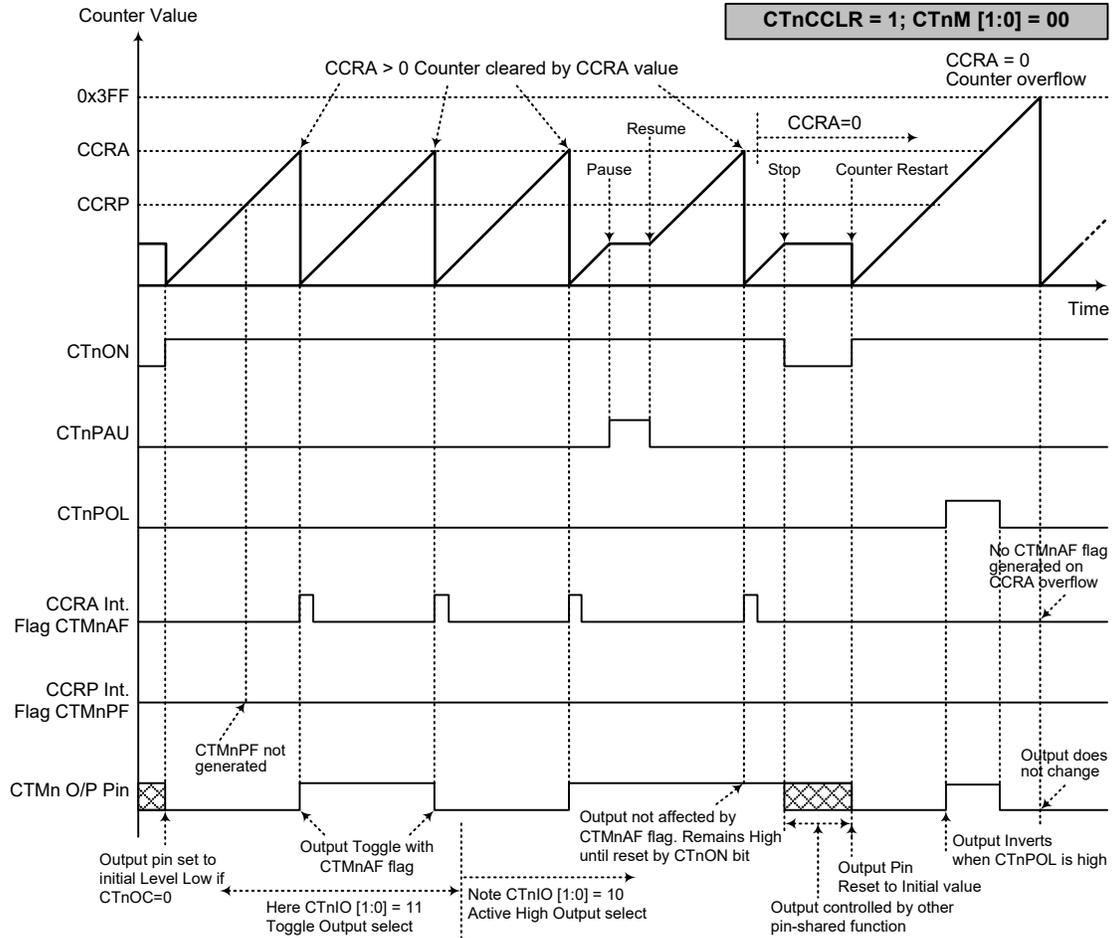
If the CTnCCLR bit in the CTMnC1 register is high then the counter will be cleared when a compare match occurs from Comparator A. However, here only the CTMnAF interrupt request flag will be generated even if the value of the CCRP bits is less than that of the CCRA registers. Therefore when CTnCCLR is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

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



**Compare Match Output Mode – CTnCCLR=0 (n=0~2)**

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



**Compare Match Output Mode – CTnCCR=1 (n=0-2)**

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

**Timer/Counter Mode**

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

**PWM Output Mode**

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

As both the period and duty cycle of the PWM waveform can be controlled, the choice of generated waveform is extremely flexible. In the PWM Output Mode, the CTnCCLR bit has no effect 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 CTnDPX bit in the CTMnC1 register. The PWM waveform frequency and duty cycle can therefore be controlled by the values in the CCRA and CCRP registers.

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

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

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

If  $f_{SYS}=8\text{MHz}$ , CTMn clock source is  $f_{SYS}/4$ , CCRP=100b, CCRA=128,

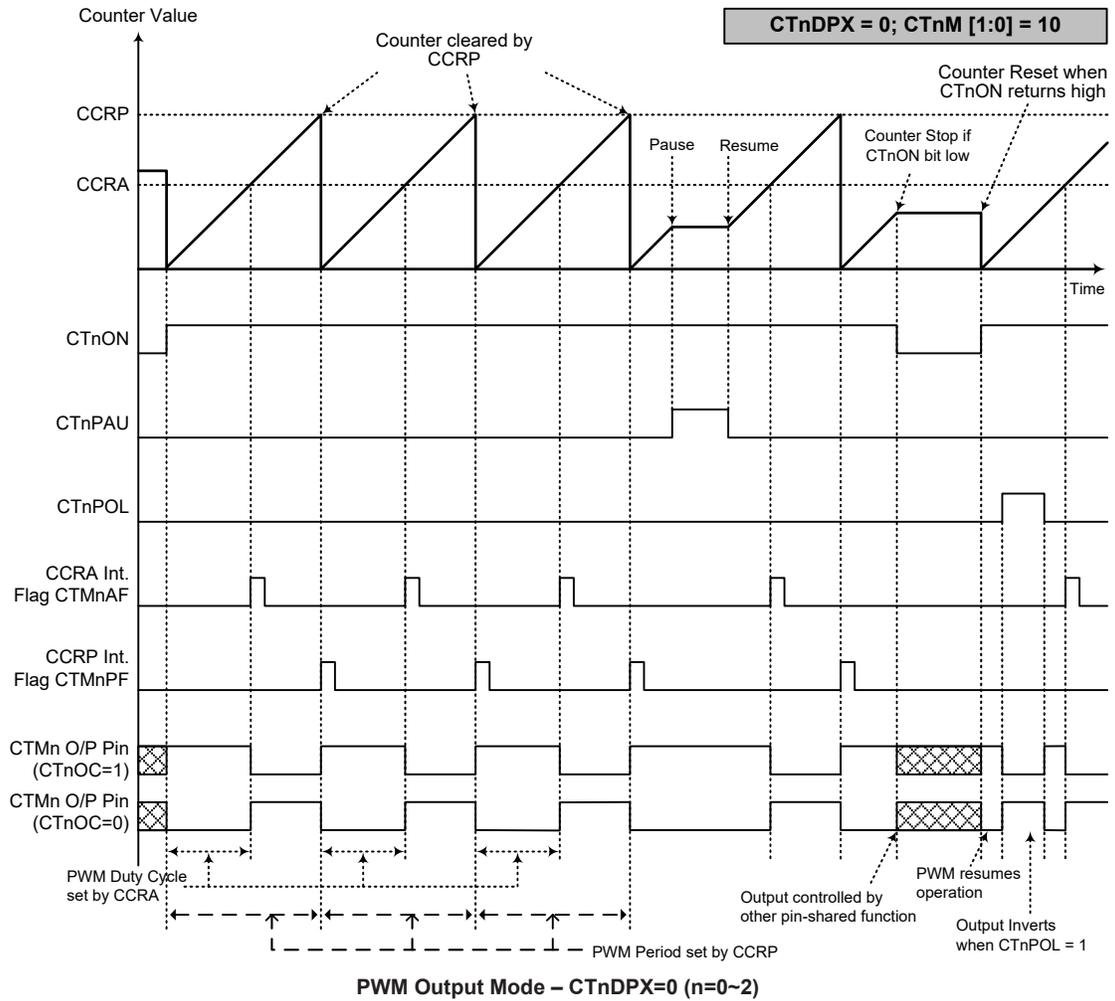
The CTMn PWM output frequency= $(f_{SYS}/4)/512=f_{SYS}/2048=4\text{kHz}$ , duty=128/512=25%.

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

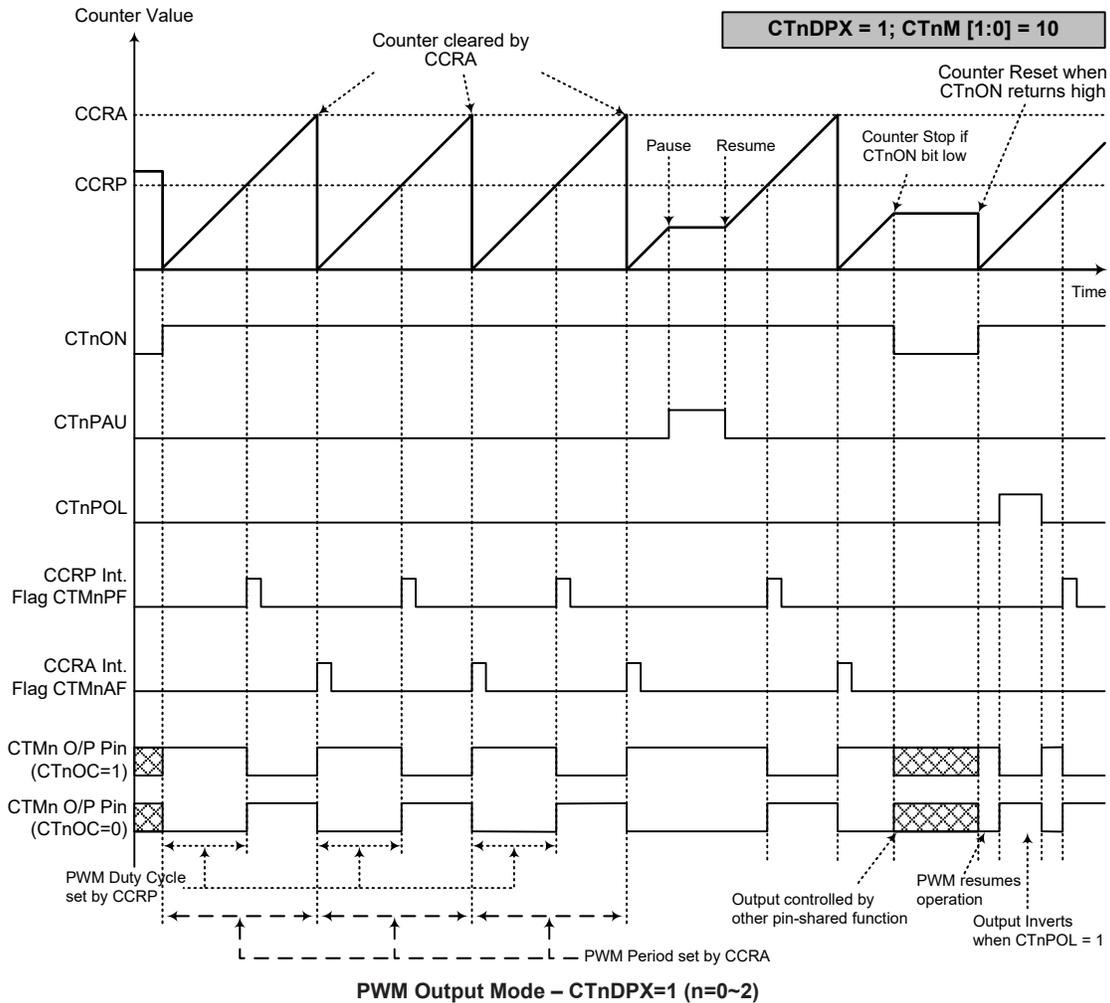
• **10-bit CTMn, PWM Output Mode, Edge-aligned Mode, CTnDPX=1**

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

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



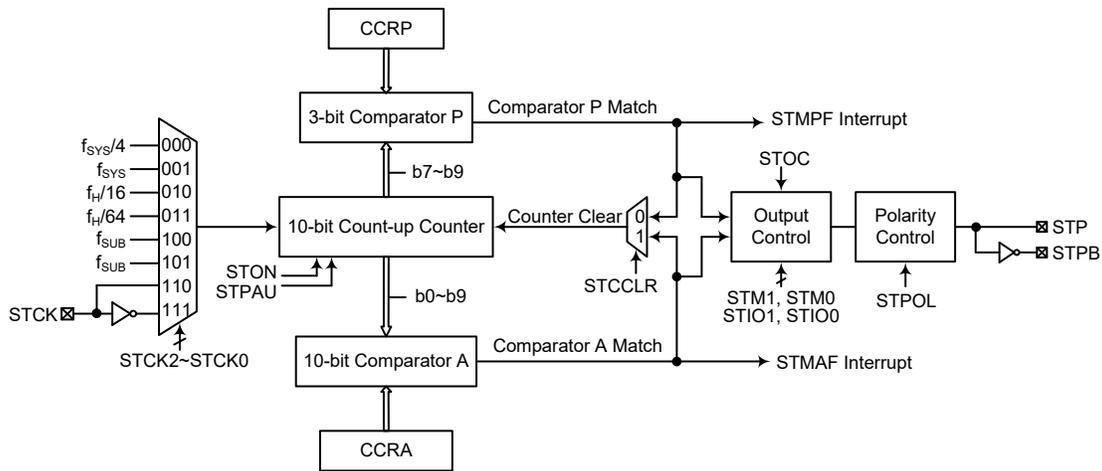
- Note: 1. Here CTnDPX=0 – Counter cleared by CCRP  
 2. A counter clear sets the PWM Period  
 3. The internal PWM function continues running even when CTnIO[1:0]=00 or 01  
 4. The CTnCCLR bit has no influence on PWM operation



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

## Standard Type TM – STM

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



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

10-bit Standard Type TM Block Diagram

### Standard TM Operation

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

The only way of changing the value of the 10-bit counter using the application program, is to clear the counter by changing the STON 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 STM interrupt signal will also usually be generated. The Standard Type TM can operate in a number of different operational modes, can be driven by different clock sources including an input pin and can also control more than one output pins. All operating setup conditions are selected using relevant internal registers.

### Standard Type TM Register Description

Overall operation of the Standard TM is controlled using a series of registers. A read only register pair exists to store the internal counter 10-bit value, while a read/write register pair exists to store the internal 10-bit CCRA value. The remaining two registers are control registers which setup the different operating and control modes as well as the 3-bit CCRP value.

Register Name	Bit							
	7	6	5	4	3	2	1	0
STMC0	STPAU	STCK2	STCK1	STCK0	STON	STRP2	STRP1	STRP0
STMC1	STM1	STM0	STIO1	STIO0	STOC	STPOL	STDPX	STCCLR
STMDL	D7	D6	D5	D4	D3	D2	D1	D0
STMDH	—	—	—	—	—	—	D9	D8
STMAL	D7	D6	D5	D4	D3	D2	D1	D0
STMAH	—	—	—	—	—	—	D9	D8

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

• **STMC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	STPAU	STCK2	STCK1	STCK0	STON	STRP2	STRP1	STRP0
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 STPAU: STM 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 STM 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 STCK2~STCK0: Select STM Counter Clock**  
 000:  $f_{SYS}/4$   
 001:  $f_{SYS}$   
 010:  $f_{H}/16$   
 011:  $f_{H}/64$   
 100:  $f_{SUB}$   
 101:  $f_{SUB}$   
 110: STCK rising edge clock  
 111: STCK falling edge clock

These three bits are used to select the clock source for the STM. 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 STON: STM Counter On/Off Control**  
 0: Off  
 1: On

This bit controls the overall on/off function of the STM. Setting the bit high enables the counter to run while clearing the bit disables the STM. Clearing this bit to zero will stop the counter from counting and turn off the STM 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 STM is in the Compare Match Output Mode then the STM output pin will be reset to its initial condition, as specified by the STOC bit, when the STON bit changes from low to high.

Bit 2~0     **STRP2~STRP0**: STM CCRP 3-bit register, compared with the STM counter bit 9~bit 7  
 Comparator P Match Period=  
     000: 1024 STM clocks  
     001: 128 STM clocks  
     010: 256 STM clocks  
     011: 384 STM clocks  
     100: 512 STM clocks  
     101: 640 STM clocks  
     110: 768 STM clocks  
     111: 896 STM 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 STCCLR bit is set to zero. Clearing the STCCLR 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.

• **STMC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	STM1	STM0	STIO1	STIO0	STOC	STPOL	STDPX	STCCLR
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     **STM1~STM0**: Select STM Operating Mode  
     00: Compare Match Output Mode  
     01: Undefined  
     10: PWM Output Mode or Single Pulse Output Mode  
     11: Timer/Counter Mode

These bits setup the required operating mode for the STM. To ensure reliable operation the STM should be switched off before any changes are made to the STM1 and STM0 bits. In the Timer/Counter Mode, the STM output pin state is undefined.

Bit 5~4     **STIO1~STIO0**: Select STM external pins function  
 Compare Match Output Mode  
     00: No change  
     01: Output low  
     10: Output high  
     11: Toggle output  
 PWM Output Mode/Single Pulse Output Mode  
     00: PWM output inactive state  
     01: PWM output active state  
     10: PWM output  
     11: Single Pulse Output  
 Timer/Counter Mode  
     Unused

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

In the Compare Match Output Mode, the STIO1 and STIO0 bits determine how the STM output pin changes state when a compare match occurs from the Comparator A. The TM 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 STM output pin should be setup using the STOC bit in the STMC1 register. Note that the output level requested by the STIO1 and STIO0 bits must be different from the initial value

setup using the STOC bit otherwise no change will occur on the STM output pin when a compare match occurs. After the STM output pin changes state, it can be reset to its initial level by changing the level of the STON bit from low to high.

In the PWM Output Mode, the STIO1 and STIO0 bits determine how the STM 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 STIO1 and STIO0 bits only after the STM has been switched off. Unpredictable PWM outputs will occur if the STIO1 and STIO0 bits are changed when the STM is running.

Bit 3 **STOC**: STM STP Output control

Compare Match Output Mode

0: Initial low

1: Initial high

PWM Output Mode/Single Pulse Output Mode

0: Active low

1: Active high

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

Bit 2 **STPOL**: STP Output Polarity control

0: Non-invert

1: Invert

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

Bit 1 **STDPX**: STM PWM duty/period control

0: CCRP – period; CCRA – duty

1: CCRP – duty; CCRA – period

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

Bit 0 **STCCLR**: STM Counter Clear condition selection

0: Comparator P match

1: Comparator A match

This bit is used to select the method which clears the counter. Remember that the Standard TM contains two comparators, Comparator A and Comparator P, either of which can be selected to clear the internal counter. With the STCCLR 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 STCCLR bit is not used in the PWM Output or Single Pulse Output Mode.

• **STMDL 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**: STM Counter Low Byte Register bit 7~bit 0  
STM 10-bit Counter bit 7~bit 0

• **STMDH 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**: STM Counter High Byte Register bit 1~bit 0  
STM 10-bit Counter bit 9~bit 8

• **STMAL 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**: STM CCRA Low Byte Register bit 7~bit 0  
STM 10-bit CCRA bit 7~bit 0

• **STMAH 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**: STM CCRA High Byte Register bit 1~bit 0  
STM 10-bit CCRA bit 9~bit 8

## Standard Type TM Operation Modes

The Standard Type TM can operate in one of four operating modes, Compare Match Output Mode, PWM Output Mode, Single Pulse Output Mode or Timer/Counter Mode. The operating mode is selected using the STM1 and STM0 bits in the STMC1 register.

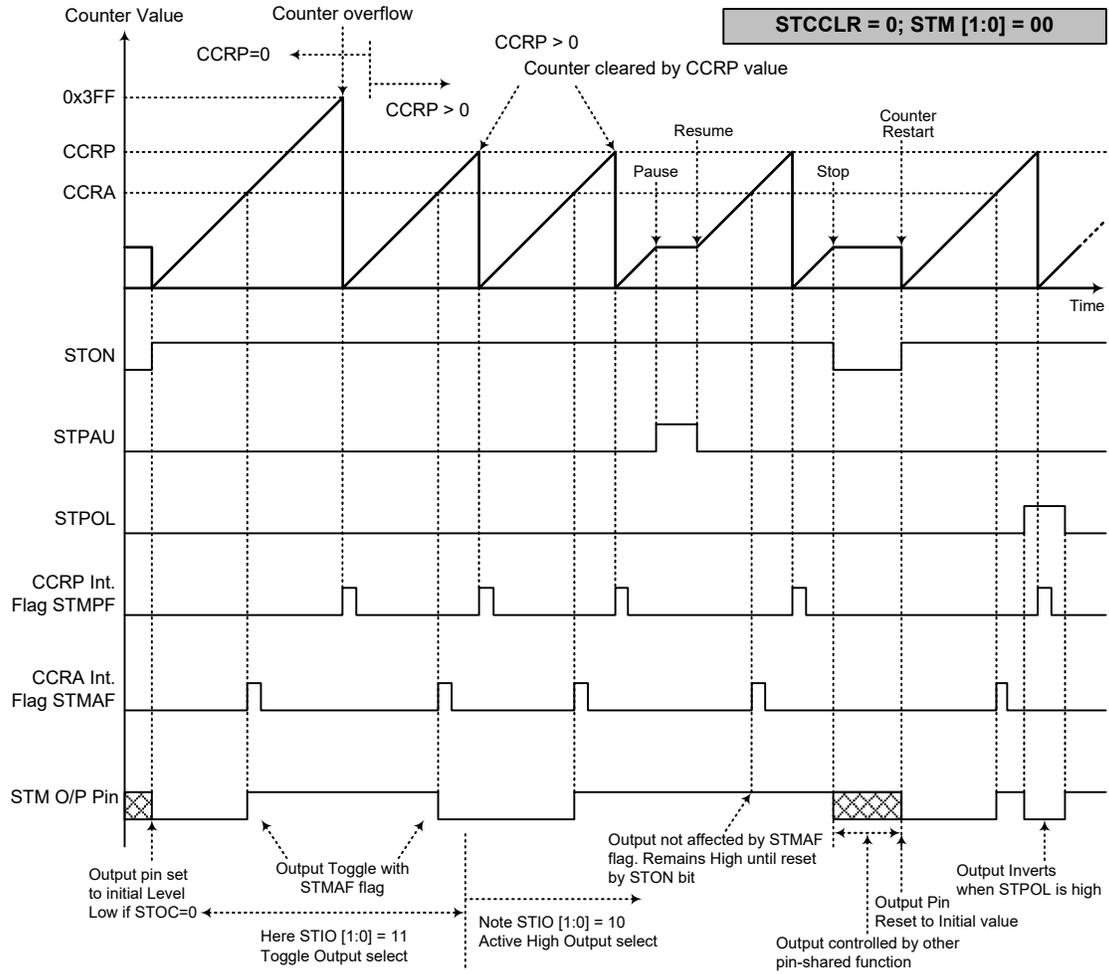
### Compare Match Output Mode

To select this mode, bits STM1 and STM0 in the STMC1 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 STCCLR 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 STMAF and STMPF interrupt request flags for Comparator A and Comparator P respectively, will both be generated.

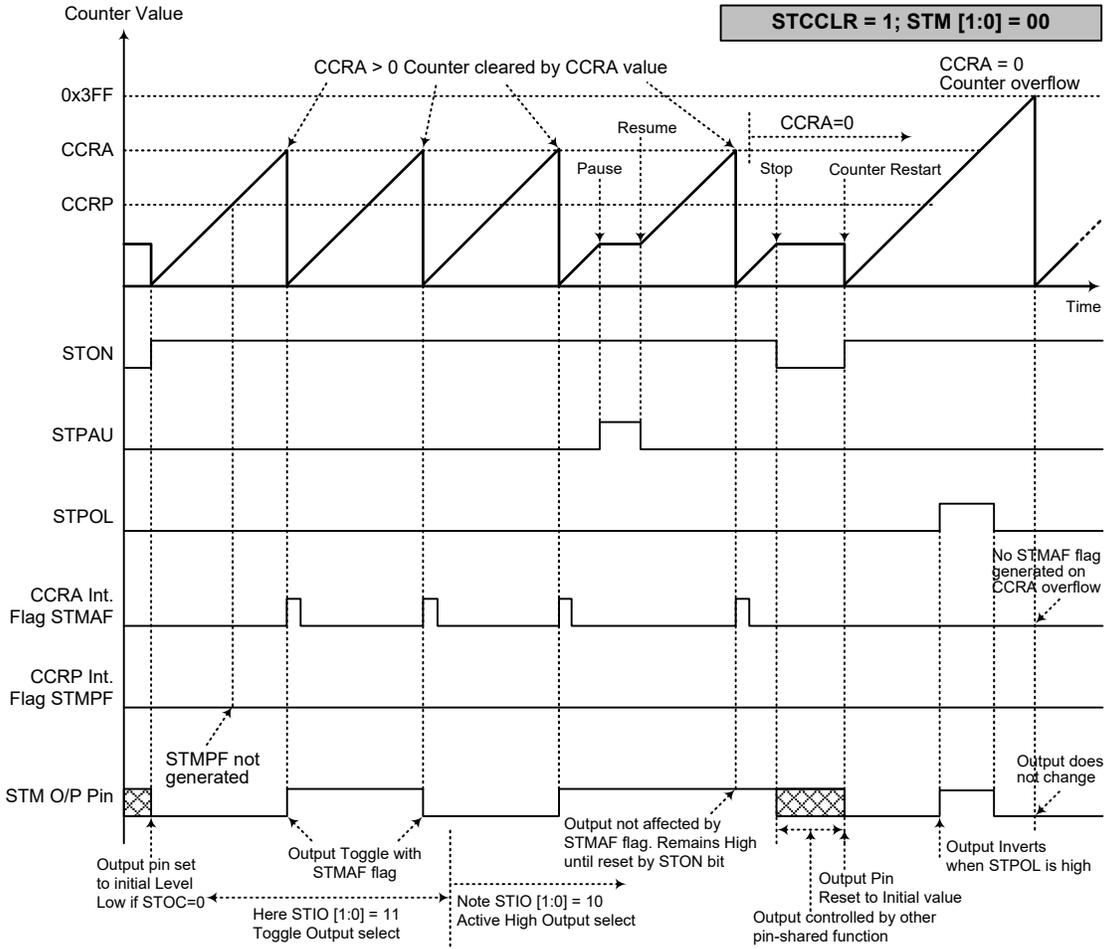
If the STCCLR bit in the STMC1 register is high then the counter will be cleared when a compare match occurs from Comparator A. However, here only the STMAF interrupt request flag will be generated even if the value of the CCRP bits is less than that of the CCRA registers. Therefore when STCCLR is high no STMPF interrupt request flag will be generated. In the Compare Match Output Mode, the CCRA can not be set to “0”.

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

As the name of the mode suggests, after a comparison is made, the STM output pin, will change state. The STM output pin condition however only changes state when a STMAF interrupt request flag is generated after a compare match occurs from Comparator A. The STMPF interrupt request flag, generated from a compare match occurs from Comparator P, will have no effect on the STM output pin. The way in which the STM output pin changes state are determined by the condition of the STIO1 and STIO0 bits in the STMC1 register. The STM output pin can be selected using the STIO1 and STIO0 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 STM output pin, which is setup after the STON bit changes from low to high, is setup using the STOC bit. Note that if the STIO1 and STIO0 bits are zero then no pin change will take place.



- Note: 1. With STCCLR=0 a Comparator P match will clear the counter  
 2. The STM output pin is controlled only by the STMAF flag  
 3. The output pin is reset to its initial state by a STON bit rising edge



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

- Note: 1. With STCCLR=1 a Comparator A match will clear the counter  
 2. The STM output pin is controlled only by the STMAF flag  
 3. The output pin is reset to its initial state by a STON bit rising edge  
 4. A STMPF flag is not generated when STCCLR=1

### Timer/Counter Mode

To select this mode, bits STM1 and STM0 in the STMC1 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 STM 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 STM output pin is not used in this mode, the pin can be used as a normal I/O pin or other pin-shared function.

### PWM Output Mode

To select this mode, bits STM1 and STM0 in the STMC1 register should be set to 10 respectively and also the STIO1 and STIO0 bits should be set to 10 respectively. The PWM function within the STM 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 STM 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 STCCLR bit has no effect as the PWM period. Both of the CCRA and CCRP registers are used to generate the PWM waveform, one register is used to clear the internal counter and thus control the PWM waveform frequency, while the other one is used to control the duty cycle. Which register is used to control either frequency or duty cycle is determined using the STDPX bit in the STMC1 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 STOC bit in the STMC1 register is used to select the required polarity of the PWM waveform while the two STIO1 and STIO0 bits are used to enable the PWM output or to force the STM output pin to a fixed high or low level. The STPOL bit is used to reverse the polarity of the PWM output waveform.

• **10-bit STM, PWM Output Mode, Edge-aligned Mode, STDPX=0**

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

If  $f_{SYS}=4\text{MHz}$ , STM clock source is  $f_{SYS}/4$ , CCRP=4 and CCRA=128,

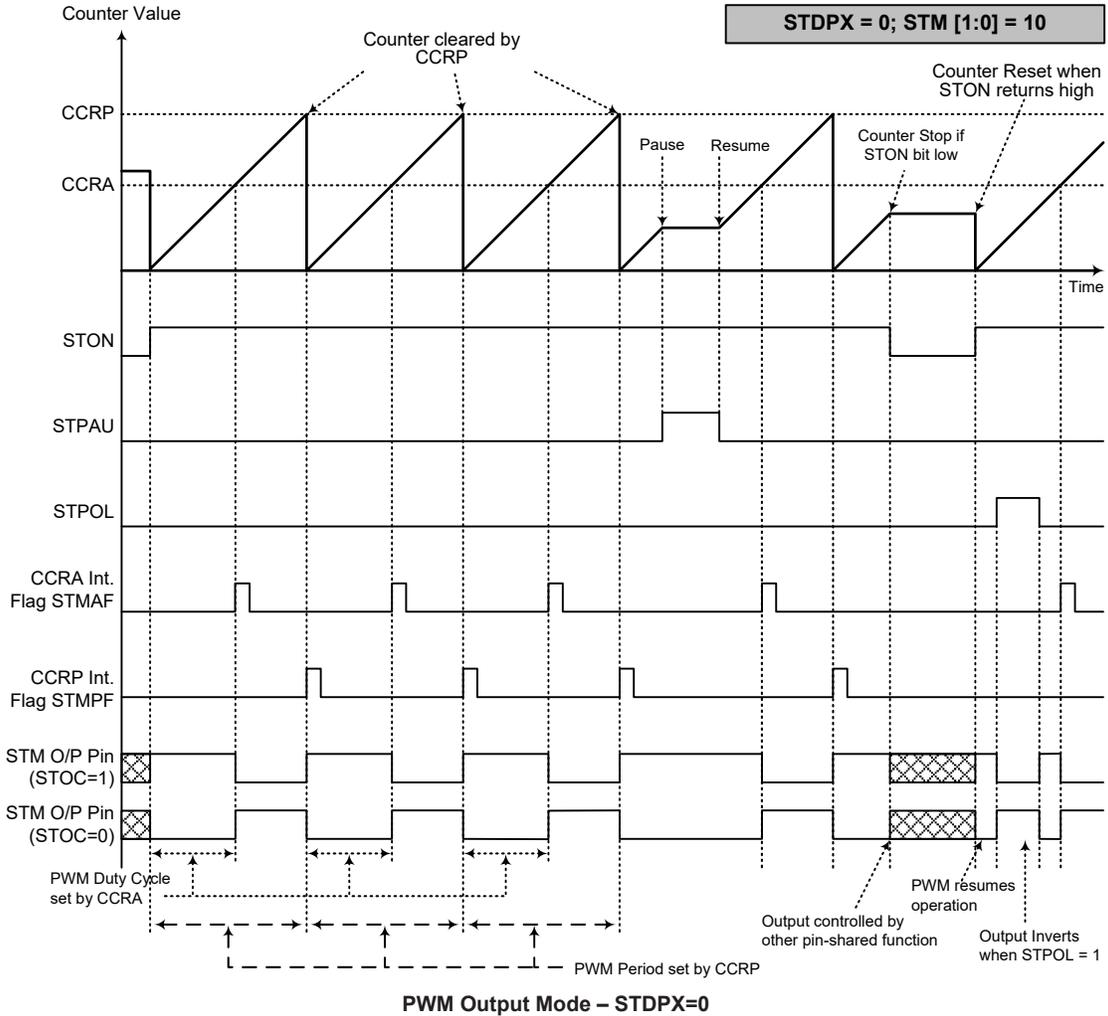
The STM PWM output frequency= $(f_{SYS}/4)/(4\times 128)=f_{SYS}/2048=2\text{kHz}$ , duty= $128/(4\times 128)=25\%$ .

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

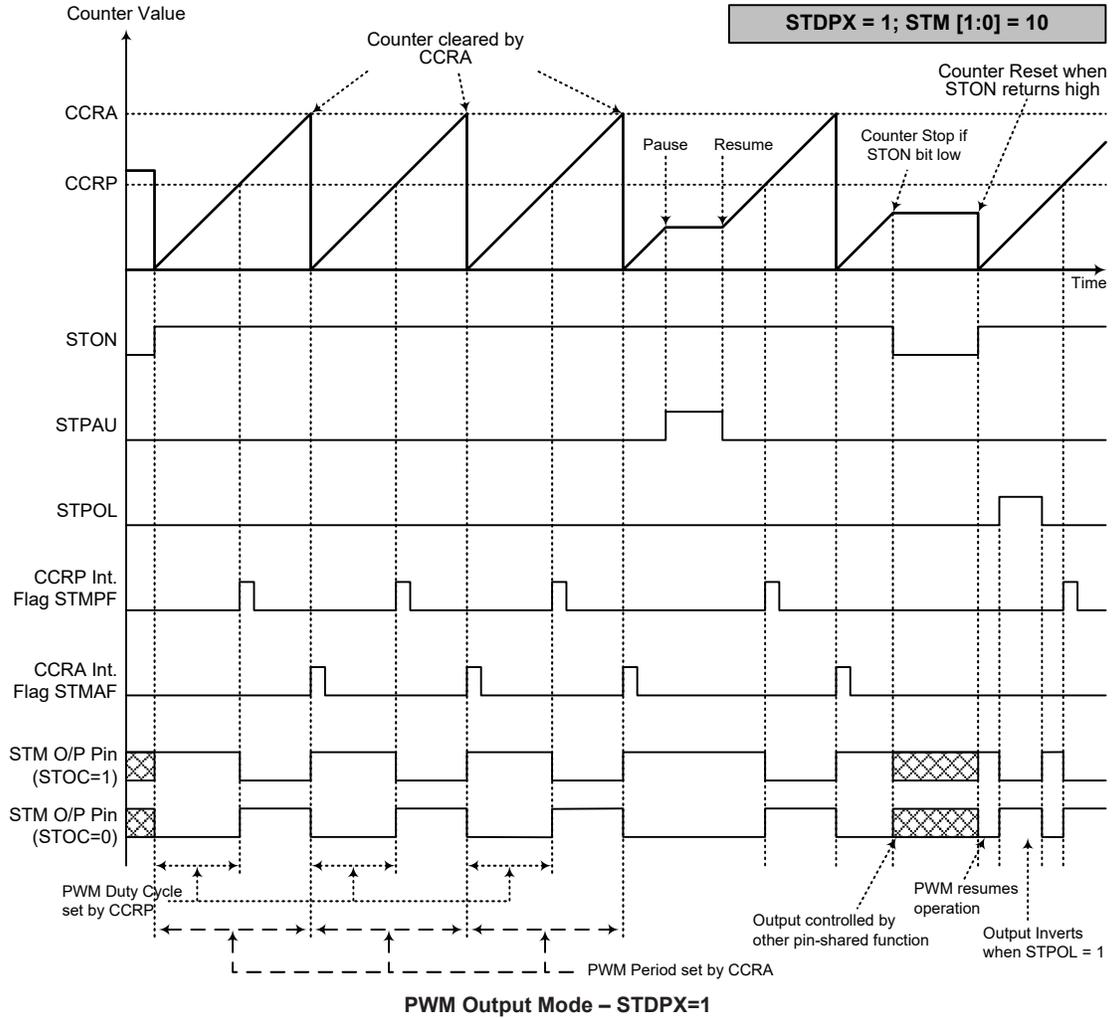
• **10-bit STM, PWM Output Mode, Edge-aligned Mode, STDPX=1**

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

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



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



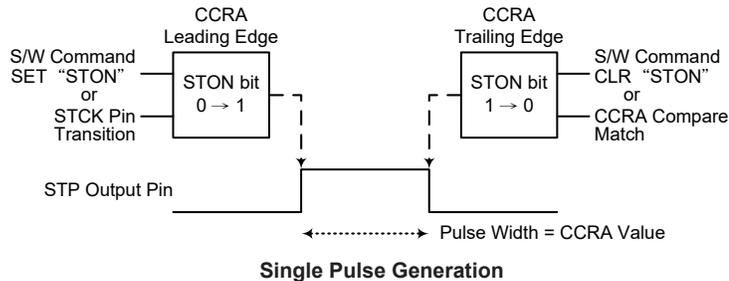
- Note: 1. Here STDPX=1 – Counter cleared by CCRA  
 2. A counter clear sets the PWM Period  
 3. The internal PWM function continues even when STIO[1:0]=00 or 01  
 4. The STCCLR bit has no influence on PWM operation

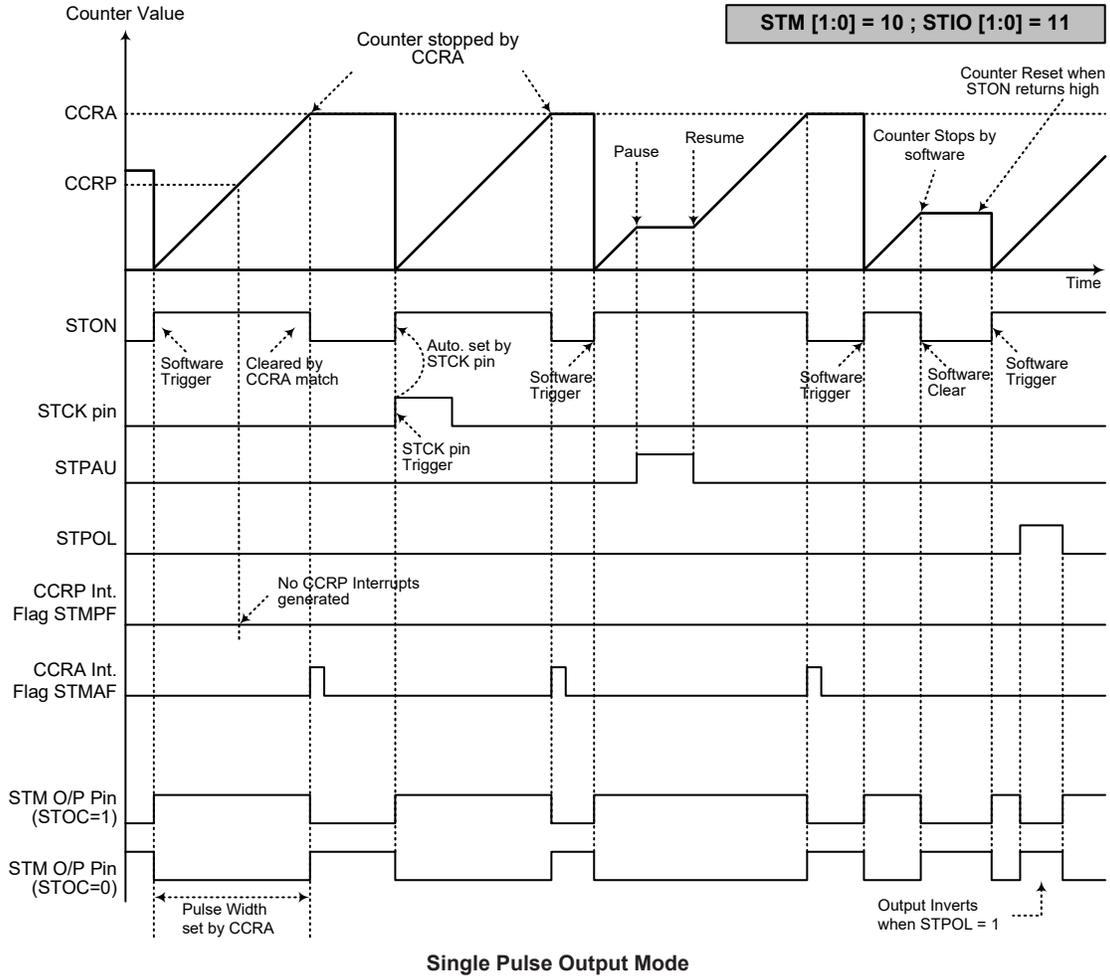
### Single Pulse Output Mode

To select this mode, bits STM1 and STM0 in the STMC1 register should be set to 10 respectively and also the STIO1 and STIO0 bits should be set to 11 respectively. The Single Pulse Output Mode, as the name suggests, will generate a single shot pulse on the STM output pin.

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

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

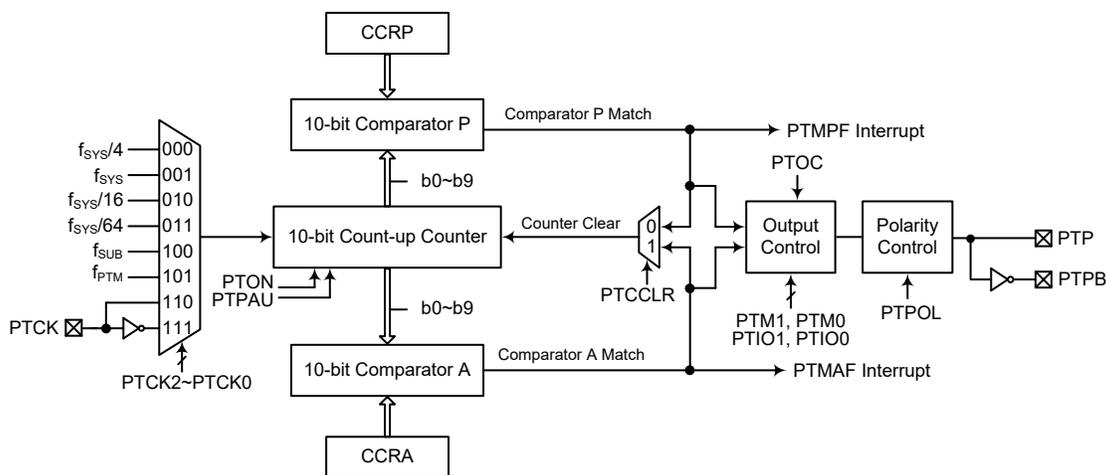




- Note:
1. Counter stopped by CCRA
  2. CCRP is not used
  3. The pulse triggered by the STCK pin or by setting the STON bit high
  4. A STCK pin active edge will automatically set the STON bit high
  5. In the Single Pulse Mode, STIO[1:0] must be set to "11" and can not be changed.

## Periodic Type TM – PTM

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



- Note: 1. The PTM external pins are pin-shared with other functions, so before using the PTM function, ensure that the pin-shared function registers have been set properly to enable the PTM pin function.  
 2. The PTPB is the inverted signal of the PTP.  
 3. A dedicated oscillator which outputs the  $f_{PTM}$  frequency is provided for the PTM clock. Refer to the oscillator section for the oscillator fine-adjustment and enable control.

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

### Periodic TM Operation

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

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

### Periodic Type TM Register Description

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

Register Name	Bit							
	7	6	5	4	3	2	1	0
PTMC0	PTPAU	PTCK2	PTCK1	PTCK0	PTON	—	—	—
PTMC1	PTM1	PTM0	PTIO1	PTIO0	PTOC	PTPOL	D1	PTCCLR
PTMDL	D7	D6	D5	D4	D3	D2	D1	D0
PTMDH	—	—	—	—	—	—	D9	D8
PTMAL	D7	D6	D5	D4	D3	D2	D1	D0
PTMAH	—	—	—	—	—	—	D9	D8
PTMRPL	D7	D6	D5	D4	D3	D2	D1	D0
PTMRPH	—	—	—	—	—	—	D9	D8

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

• **PTMC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	PTPAU	PTCK2	PTCK1	PTCK0	PTON	—	—	—
R/W	R/W	R/W	R/W	R/W	R/W	—	—	—
POR	0	0	0	0	0	—	—	—

Bit 7 **PTPAU**: PTM Counter Pause Control

- 0: Run
- 1: Pause

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

Bit 6~4 **PTCK2~PTCK0**: Select PTM Counter clock

- 000:  $f_{SYS}/4$
- 001:  $f_{SYS}$
- 010:  $f_H/16$
- 011:  $f_H/64$
- 100:  $f_{SUB}$
- 101:  $f_{PTM}$
- 110: PTCK rising edge clock
- 111: PTCK falling edge clock

These three bits are used to select the clock source for the PTM. The external pin clock source can be chosen to be active on the rising or falling edge. The clock source  $f_{PTM}$  is the PTM dedicated oscillator, PTMOSC, output frequency, 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 **PTON**: PTM Counter On/Off Control

- 0: Off
- 1: On

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

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

Bit 2~0 Unimplemented, read as “0”

• **PTMC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	PTM1	PTM0	PTIO1	PTIO0	PTOC	PTPOL	D1	PTCCLR
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 **PTM1~PTM0**: Select PTM Operating Mode  
 00: Compare Match Output Mode  
 01: Undefined  
 10: PWM Output Mode or Single Pulse Output Mode  
 11: Timer/Counter Mode

These bits setup the required operating mode for the PTM. To ensure reliable operation the PTM should be switched off before any changes are made to the PTM1 and PTM0 bits. In the Timer/Counter Mode, the PTM output pin state is undefined.

- Bit 5~4 **PTIO1~PTIO0**: Select PTM external pin function

- Compare Match Output Mode  
 00: No change  
 01: Output low  
 10: Output high  
 11: Toggle output
- PWM Output Mode/Single Pulse Output Mode  
 00: PWM Output inactive state  
 01: PWM Output active state  
 10: PWM output  
 11: Single pulse output

- Timer/Counter Mode  
 Unused

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

In the Compare Match Output Mode, the PTIO1 and PTIO0 bits determine how the PTM output pin changes state when a compare match occurs from the Comparator A. The PTM output pin can be setup to switch high, switch low or to toggle its present state when a compare match occurs from the Comparator A. When the bits are both zero, then no change will take place on the output. The initial value of the PTM output pin should be setup using the PTOC bit in the PTMC1 register. Note that the output level requested by the PTIO1 and PTIO0 bits must be different from the initial value setup using the PTOC bit otherwise no change will occur on the PTM output pin when a compare match occurs. After the PTM output pin changes state, it can be reset to its initial level by changing the level of the PTON bit from low to high.

In the PWM Output Mode, the PTIO1 and PTIO0 bits determine how the PTM output pin changes state when a certain compare match condition occurs. The PWM output function is modified by changing these two bits. It is necessary to only change the values of the PTIO1 and PTIO0 bits only after the PTM has been switched off. Unpredictable PWM outputs will occur if the PTIO1 and PTIO0 bits are changed when the PTM is running.

- Bit 3      **PTOC**: PTM PTP Output control bit  
 Compare Match Output Mode  
     0: Initial low  
     1: Initial high  
 PWM Output Mode/Single Pulse Output Mode  
     0: Active low  
     1: Active high

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

- Bit 2      **PTPOL**: PTP Output polarity control  
     0: Non-invert  
     1: Invert

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

- Bit 1      **D1**: Reserved, must be fixed at “0”

- Bit 0      **PTCCLR**: PTM Counter Clear condition selection  
     0: Comparator P match  
     1: Comparator A match

This bit is used to select the method which clears the counter. Remember that the Periodic TM contains two comparators, Comparator A and Comparator P, either of which can be selected to clear the internal counter. With the PTCCLR 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 PTCCLR bit is not used in the PWM Output Mode or Single Pulse Output Mode.

• **PTMDL 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**: PTM Counter Low Byte Register bit 7 ~ bit 0  
 PTM 10-bit Counter bit 7 ~ bit 0

• **PTMDH 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**: PTM Counter High Byte Register bit 1 ~ bit 0  
 PTM 10-bit Counter bit 9 ~ bit 8

• **PTMAL 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**: PTM CCRA Low Byte Register bit 7 ~ bit 0  
 PTM 10-bit CCRA bit 7 ~ bit 0

• **PTMAH 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**: PTM CCRA High Byte Register bit 1 ~ bit 0  
 PTM 10-bit CCRA bit 9 ~ bit 8

• **PTMRPL 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**: PTM CCRP Low Byte Register bit 7 ~ bit 0  
 PTM 10-bit CCRP bit 7 ~ bit 0

• **PTMRPH 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**: PTM CCRP High Byte Register bit 1 ~ bit 0  
 PTM 10-bit CCRP bit 9 ~ bit 8

## Periodic Type TM Operating Modes

The Periodic Type TM can operate in one of four operating modes, Compare Match Output Mode, PWM Output Mode, Single Pulse Output Mode or Timer/Counter Mode. The operating mode is selected using the PTM1 and PTM0 bits in the PTMC1 register.

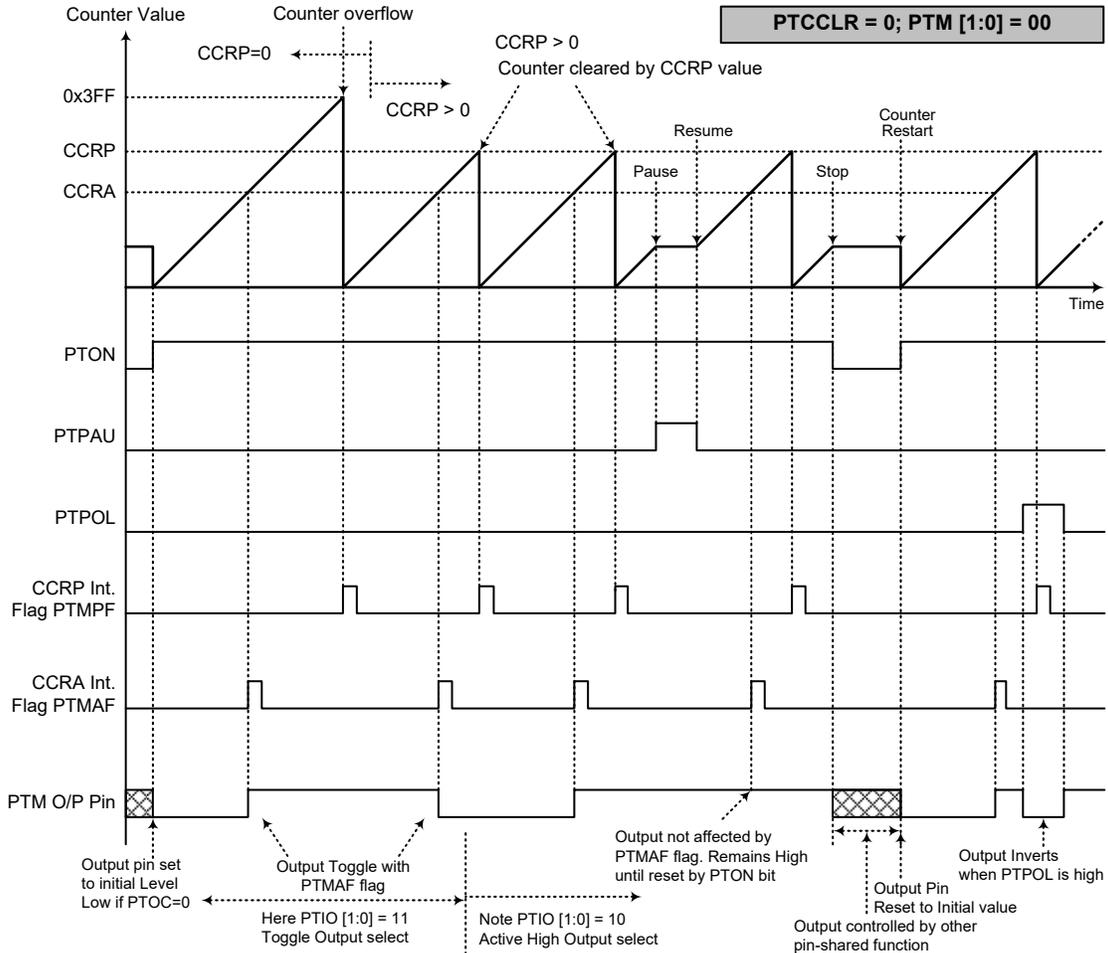
### Compare Match Output Mode

To select this mode, bits PTM1 and PTM0 in the PTMC1 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 PTCCLR 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 PTMAF and PTMPF interrupt request flags for Comparator A and Comparator P respectively, will both be generated.

If the PTCCLR bit in the PTMC1 register is high then the counter will be cleared when a compare match occurs from Comparator A. However, here only the PTMAF interrupt request flag will be generated even if the value of the CCRP bits is less than that of the CCRA registers. Therefore when PTCCLR is high no PTMPF interrupt request flag will be generated. In the Compare Match Output Mode, the CCRA cannot be cleared to zero.

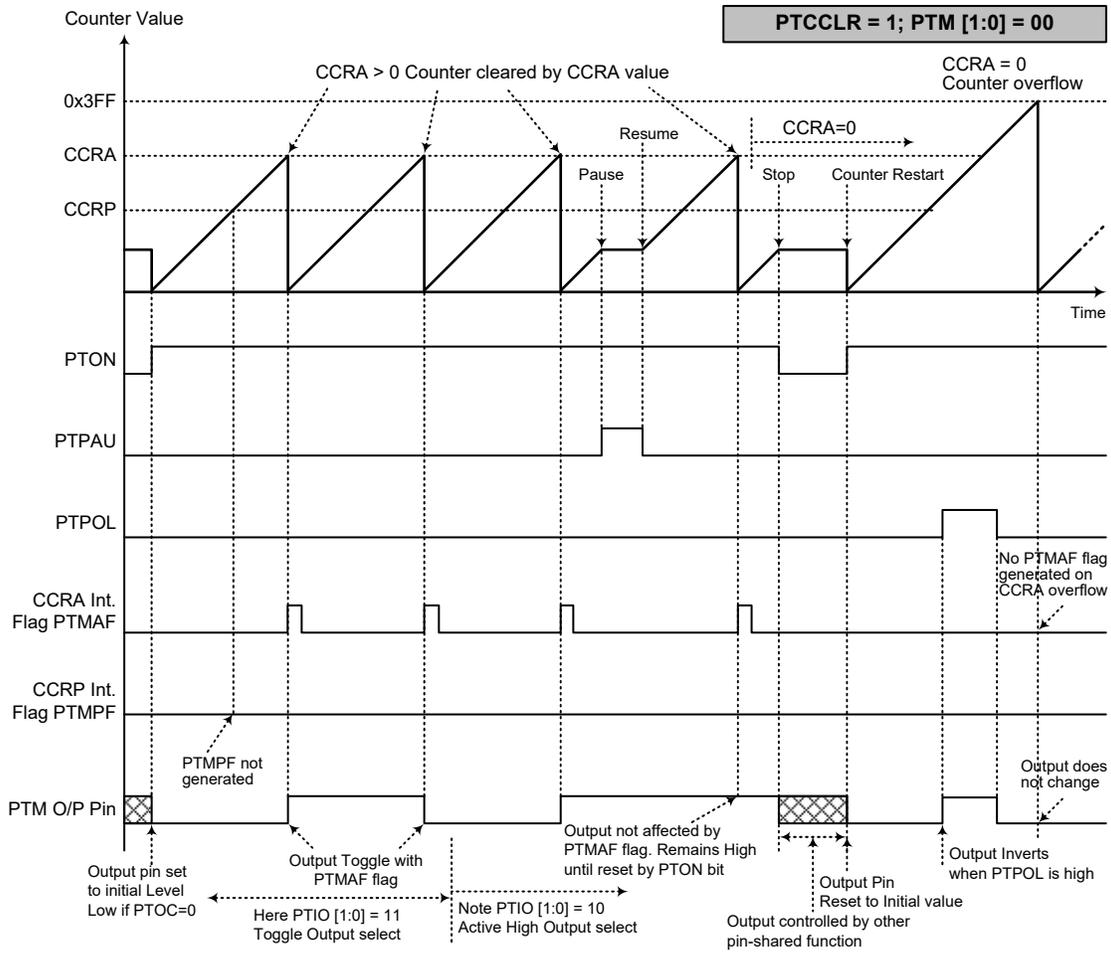
If the CCRA bits are all zero, the counter will overflow when it reaches its maximum 10-bit, 3FF Hex, value, however here the PTMAF interrupt request flag will not be generated.

As the name of the mode suggests, after a comparison is made, the PTM output pin, will change state. The PTM output pin condition however only changes state when a PTMAF interrupt request flag is generated after a compare match occurs from Comparator A. The PTMPF interrupt request flag, generated from a compare match occurs from Comparator P, will have no effect on the PTM output pin. The way in which the PTM output pin changes state are determined by the condition of the PTIO1 and PTIO0 bits in the PTMC1 register. The PTM output pin can be selected using the PTIO1 and PTIO0 bits to go high, to go low or to toggle from its present condition when a compare match occurs from Comparator A. The initial condition of the PTM output pin, which is setup after the PTON bit changes from low to high, is setup using the PTOC bit. Note that if the PTIO1 and PTIO0 bits are zero then no pin change will take place.



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

- Note: 1. With PTCCCLR=0 a Comparator P match will clear the counter  
 2. The PTM output pin is controlled only by the PTMAF flag  
 3. The output pin is reset to its initial state by a PTON bit rising edge



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

- Note: 1. With PTCCLR=1 a Comparator A match will clear the counter  
 2. The PTM output pin is controlled only by the PTMAF flag  
 3. The output pin is reset to its initial state by a PTON bit rising edge  
 4. A PTMPF flag is not generated when PTCCLR=1

**Timer/Counter Mode**

To select this mode, bits PTM1 and PTM0 in the PTMC1 register should be set to 11 respectively. The Timer/Counter Mode operates in an identical way to the Compare Match Output Mode generating the same interrupt flags. The exception is that in the Timer/Counter Mode the PTM output pin is not used. Therefore the above description and Timing Diagrams for the Compare Match Output Mode can be used to understand its function. As the PTM output pin is not used in this mode, the pin can be used as a normal I/O pin or other pin-shared function.

**PWM Output Mode**

To select this mode, bits PTM1 and PTM0 in the PTMC1 register should be set to 10 respectively and also the PTIO1 and PTIO0 bits should be set to 10 respectively. The PWM function within the PTM is useful for applications which require functions such as motor control, heating control, illumination control etc. By providing a signal of fixed frequency but of varying duty cycle on the PTM output pin, a square wave AC waveform can be generated with varying equivalent DC RMS values.

As both the period and duty cycle of the PWM waveform can be controlled, the choice of generated waveform is extremely flexible. In the PWM Output Mode, the PTCCLR bit has no effect on the PWM operation. Both of the CCRA and CCRP registers are used to generate the PWM waveform, the CCRP is used to clear the internal counter and thus control the PWM waveform frequency, while the CCRA is used to control the duty cycle. The PWM waveform frequency and duty cycle can therefore be controlled by the values in the CCRP and CCRA 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 PTOC bit in the PTMC1 register is used to select the required polarity of the PWM waveform while the two PTIO1 and PTIO0 bits are used to enable the PWM output or to force the PTM output pin to a fixed high or low level. The PTPOL bit is used to reverse the polarity of the PWM output waveform.

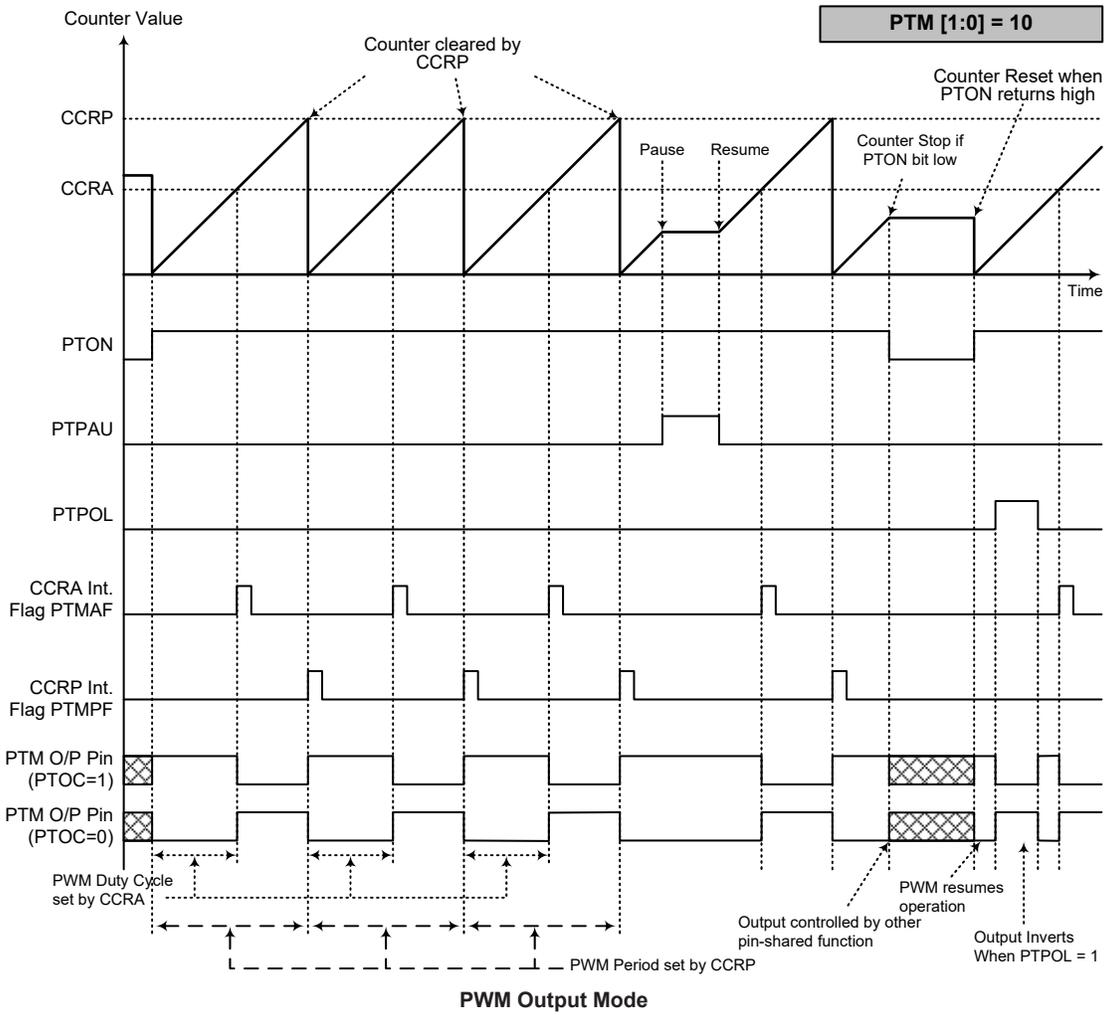
• **10-bit PTM, PWM Output Mode, Edge-aligned Mode**

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

If  $f_{SYS}=16\text{MHz}$ , PTM clock source select  $f_{SYS}/4$ ,  $\text{CCRP}=512$  and  $\text{CCRA}=128$ ,

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

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



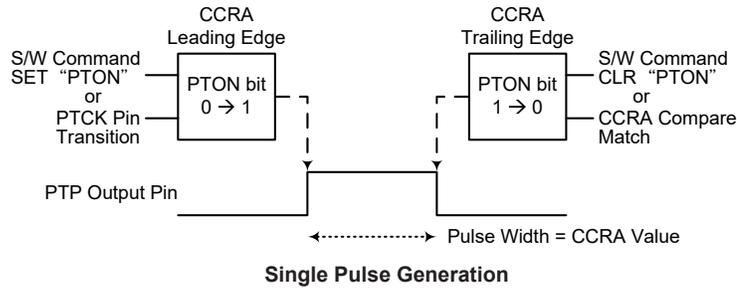
- Note:
1. Counter cleared by CCRP
  2. A counter clear sets PWM Period
  3. The internal PWM function continues running even when PTIO[1:0]=00 or 01
  4. The PTCCLR bit has no influence on PWM operation

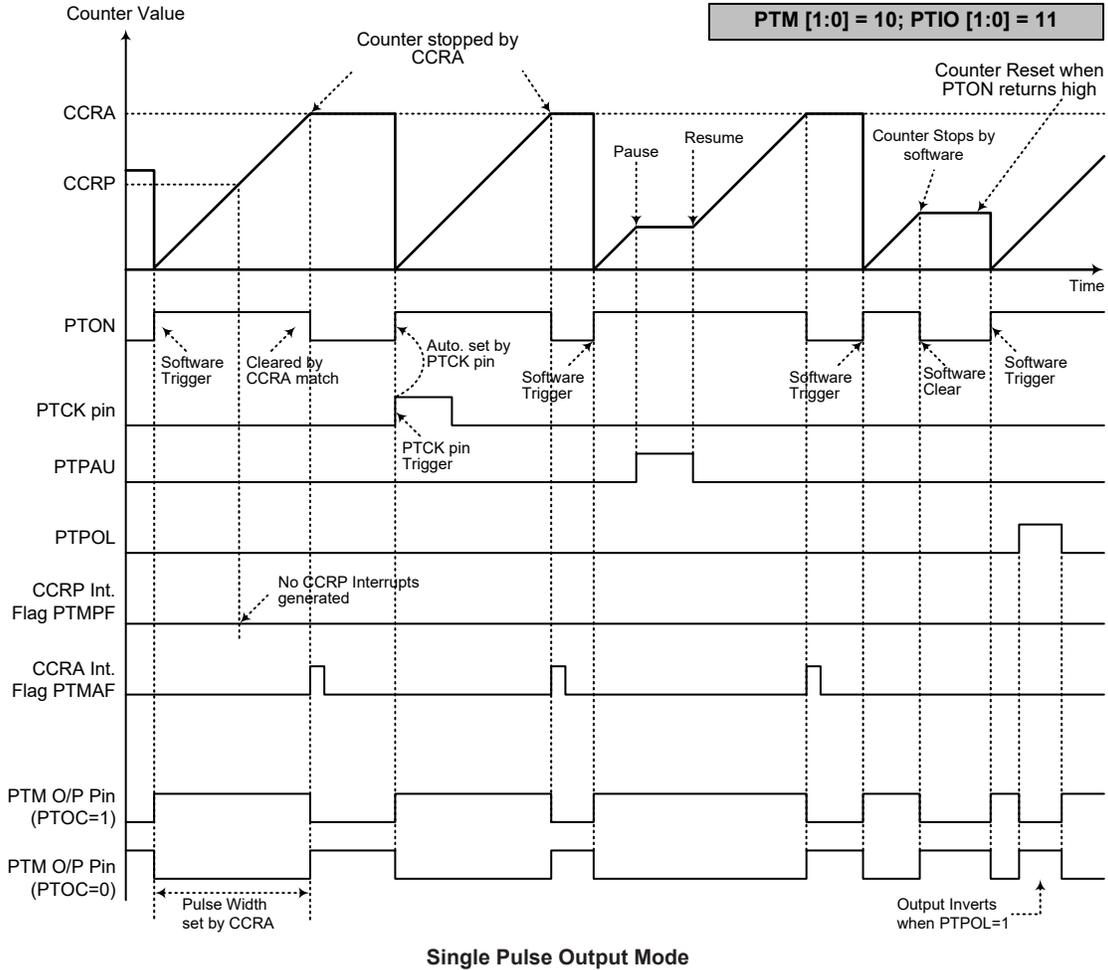
**Single Pulse Output Mode**

To select this mode, bits PTM1 and PTM0 in the PTMC1 register should be set to 10 respectively and also the PTIO1 and PTIO0 bits should be set to 11 respectively. The Single Pulse Output Mode, as the name suggests, will generate a single shot pulse on the PTM output pin.

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

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





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

## Analog to Digital Converter

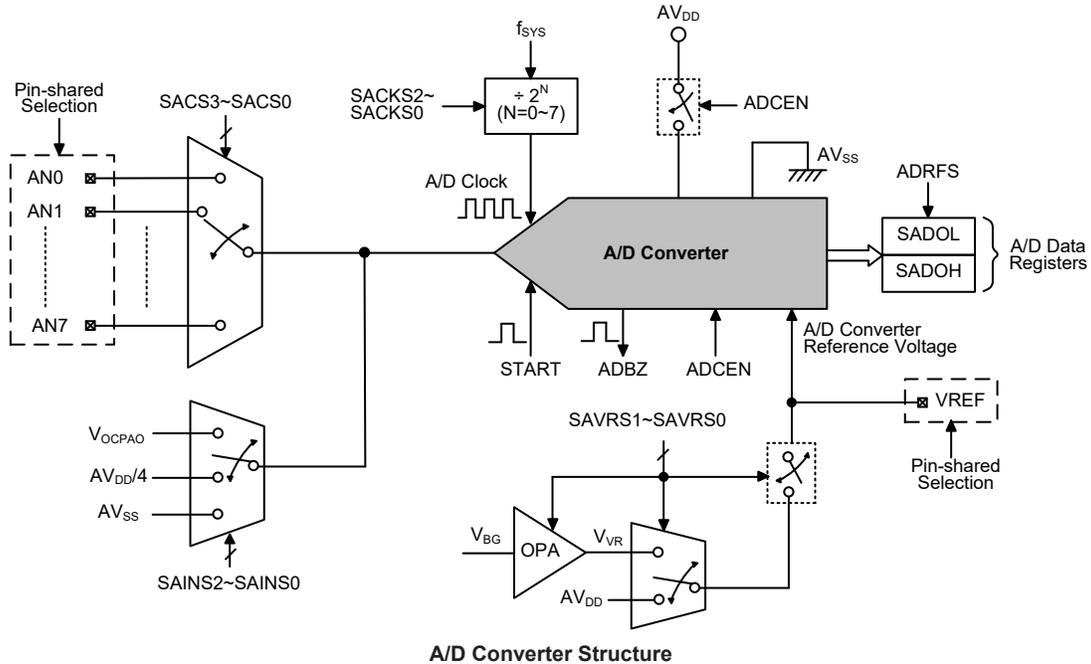
The need to interface to real world analog signals is a common requirement for many electronic systems. However, to properly process these signals by a microcontroller, they must first be converted into digital signals by A/D converters. By integrating the A/D conversion electronic circuitry into the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

### A/D Converter Overview

The device contains a multi-channel analog to digital converter which can directly interface to external analog signals, such as that from sensors or other control signals, and convert these signals directly into a 12-bit digital value. It also can convert the internal signals of the OCPAO signal from the OCP function into a 12-bit digital value. The external or internal analog signal to be converted is determined by the SAINS2~SAINS0 bits together with the SACS3~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 Input Signal	A/D Input Select Bits
AN0~AN7	AV <sub>DD</sub> /4, V <sub>OCPAO</sub> , AV <sub>SS</sub>	SACS3~SACS0 SAINS2~SAINS0

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



### A/D Converter Register Description

Overall operation of the A/D converter is controlled using four registers. A read only register pair exists to store the A/D converter data 12-bit 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 (ADRF5=0)	D3	D2	D1	D0	—	—	—	—
SADOL (ADRF5=1)	D7	D6	D5	D4	D3	D2	D1	D0
SADOH (ADRF5=0)	D11	D10	D9	D8	D7	D6	D5	D4
SADOH (ADRF5=1)	—	—	—	—	D11	D10	D9	D8
SADC0	START	ADBZ	ADCEN	ADRF5	SACS3	SACS2	SACS1	SACS0
SADC1	SAINS2	SAINS1	SAINS0	SAVRS1	SAVRS0	SACKS2	SACKS1	SACKS0

**A/D Converter Register List**

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

As the device contains an internal 12-bit A/D converter, it requires two data registers to store the converted value. These are a high byte register, known as SADOH, and a low byte register, known as SADOL. After the conversion process takes place, these registers can be directly read by the microcontroller to obtain the digitised conversion value. As only 12 bits of the 16-bit register space is utilised, the format in which the data is stored is controlled by the ADRF5 bit in the SADC0 register as shown in the accompanying table. D0~D11 are the A/D conversion result data bits. Any unused bits will be read as zero. Note that the A/D converter data register contents will be unchanged if the A/D converter is disabled.

ADRF5	SADOH								SADOL							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
0	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	0	0	0	0
1	0	0	0	0	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

**A/D Converter Data Registers**

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

To control the function and operation of the A/D converter, three control registers known as SADC0, SADC1 and ACERL are provided. These 8-bit registers define functions such as the selection of which analog channel is connected to the internal A/D converter, the digitised data format, the A/D clock source as well as controlling the start function and monitoring the A/D converter busy status. As each device contains only one actual analog to digital converter hardware circuit, each of the external analog signal inputs must be routed to the converter. The SAINS2~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 SACS3~SACS0 bits in the SADC0 register are used to determine which external channel input is selected to be converted.

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

• **SADC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	START	ADBZ	ADCEN	ADRF5	SACS3	SACS2	SACS1	SACS0
R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

- Bit 7**     **START:** Start the A/D Conversion  
0 → 1 → 0: Start an A/D conversion  
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 high 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, SADOH and SADOL, will be unchanged.
- Bit 4**     **ADRF5:** A/D Converter data format control  
0: A/D converter data format → SADOH=D[11:4]; SADOL=D[3:0]  
1: A/D converter data format → SADOH=D[11:8]; SADOL=D[7:0]  
This bit controls the format of the 12-bit converted A/D value in the two A/D data registers. Details are provided in the A/D converter data register section.
- Bit 3~0**   **SACS3~SACS0:** A/D converter external analog input channel select  
0000: AN0  
0001: AN1  
0010: AN2  
0011: AN3  
0100: AN4  
0101: AN5  
0110: AN6  
0111: AN7  
1000~1111: Undefined, input floating

• **SADC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	SAINS2	SAINS1	SAINS0	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 7~5**   **SAINS2~SAINS0:** A/D converter input signal select  
000: External source – External analog channel input, ANn  
001: Internal source – V<sub>OCPAO</sub>, OCP circuit output  
010~011: Internal source – Connected to ground  
100: Internal source – Internal signal derived from AV<sub>DD</sub>/4  
101~111: External source – External analog channel input, ANn  
Care must be taken if the SAINS2~SAINS0 bits are set to “001”~“100” 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 converter input channel by properly setting the SACS3~SACS0 bits with a value from “1000” to “1111”. 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 4~3    **SAVRS1~SAVRS0:** A/D converter reference voltage selection  
           00: External VREF pin  
           01: Internal A/D converter power supply,  $AV_{DD}$   
           10: Internal OPA output,  $V_{VR}$   
           11: 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 reference voltage sources. In this condition, the VREF pin can not be configured as the reference voltage input by properly configuring the corresponding pin-shared function control bit. Otherwise, the external input voltage on the VREF pin will be connected together with the internal reference voltage. This will result in unpredictable situations.

- Bit 2~0    **SACKS2~SACKS0:** A/D conversion clock source select  
           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 bits are used to select the clock source for the A/D converter.

### A/D Converter Reference Voltage

The reference voltage supply to the A/D converter can be supplied from the internal A/D converter power supply voltage,  $AV_{DD}$ , or internal operational amplifier output voltage,  $V_{VR}$ , 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 power supply voltage,  $AV_{DD}$ . When the SAVRS bit field is set to “10”, the A/D converter reference voltage will come from the internal operational amplifier output voltage,  $V_{VR}$ . Otherwise, if the SAVRS bit field is set to other value except “01” and “10”, 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 functions. However, if the internal reference signal 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 selected reference voltage.

SAVRS[1:0]	Reference Source	Description
00, 11	VREF pin	From external A/D converter reference pin VREF
01	$AV_{DD}$	From internal A/D converter power supply voltage
10	$V_{VR}$	From internal operational amplifier output voltage

**A/D Converter Reference Voltage Selection**

### A/D Converter Input Signals

All the external A/D converter 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 converter external input pin in the pin-shared function selection register determine whether the input pins are set as A/D converter

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

There are three internal analog signal derived from  $AV_{DD}/4$ ,  $AV_{SS}$  and the OCP circuit output can be connected to the A/D converter as the analog input signal. If the SAINS2~SAINS0 bits are set to “000” or “101~111”, the external analog channel input is selected to be converted and the SACS3~SACS0 bits can determine which actual external channel is selected to be converted. If the SAINS2~SAINS0 bits are set to “001”, the OCPAO signal from the OCP function is selected to be converted. If the SAINS2~SAINS0 bits are set to “100”, the  $AV_{DD}/4$  voltage is selected to be converted. Note that if the internal analog signal is selected to be converted, the external input channel determined by the SACS3~SACS0 bits must be switched to a non-existent A/D input channel by properly setting the SACS bit field with a value from “1000” to “1111”.

SAINS [2:0]	SACS [3:0]	Input Signals	Description
000, 101~111	0000~0111	AN0~AN7	External channel analog input ANn
	1000~1111	—	Floating, no external channel is selected
001	1000~1111	$V_{OCPAO}$	OCPAO signal from the OCP function
010~011	1000~1111	$AV_{SS}$	connected to ground
100	1000~1111	$AV_{DD}/4$	Internal signal derived from $AV_{DD}/4$

**A/D Converter Input Signal Selection**

## A/D Converter Operation

The START bit in the SADC0 register is used to start the A/D 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 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~SADCKS0, 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 selected system clock frequencies. For example, if the system clock operates at a frequency of 4MHz, the SACKS2~SACKS0 bits should not be set to 000 or 110 or 111. Doing so will give A/D clock periods that are less than the minimum A/D clock period or greater than the maximum A/D clock period which may result in inaccurate A/D conversion values. Refer to the following table for examples, where values marked with an asterisk

\* show where special care must be taken, as the values may be out of the specified A/D Clock Period range.

f <sub>sys</sub>	A/D Clock Period (t <sub>ADCK</sub> )							
	SACKS[2:0] =000 (f <sub>sys</sub> )	SACKS[2:0] =001 (f <sub>sys</sub> /2)	SACKS[2:0] =010 (f <sub>sys</sub> /4)	SACKS[2:0] =011 (f <sub>sys</sub> /8)	SACKS[2:0] =100 (f <sub>sys</sub> /16)	SACKS[2:0] =101 (f <sub>sys</sub> /32)	SACKS[2:0] =110 (f <sub>sys</sub> /64)	SACKS[2:0] =111 (f <sub>sys</sub> /128)
1MHz	1µs	2µs	4µs	8µs	16µs *	32µs *	64µs *	128µs *
2MHz	500ns	1µs	2µs	4µs	8µs	16µs *	32µs *	64µs *
4MHz	250ns *	500ns	1µs	2µs	4µs	8µs	16µs *	32µs *
8MHz	125ns *	250ns *	500ns	1µs	2µs	4µs	8µs	16µs *

**A/D Clock Period Examples**

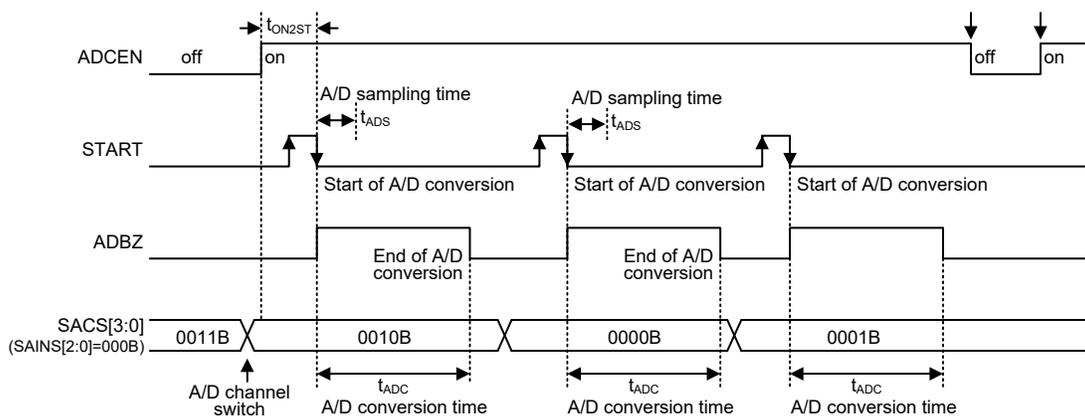
Controlling the power on/off function of the A/D converter circuitry is implemented using the ADCEN bit in the SADC0 register. This bit must be set high to power on the A/D converter. When the ADCEN bit is set high to power on the A/D converter internal circuitry a certain delay, as indicated in the timing diagram, must be allowed before an A/D conversion is initiated. Even if no pins are selected for use as A/D inputs by configuring the corresponding pin-shared control bits, 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.

**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<sub>ADS</sub> takes 4 A/D clock cycles and the data conversion takes 12 A/D clock cycles. Therefore a total of 16 A/D clock cycles for an A/D conversion which is defined as t<sub>ADC</sub> are necessary.

$$\text{Maximum single A/D conversion rate} = \text{A/D clock period} / 16$$

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



**A/D Conversion Timing – External Analog Channel Input**

## 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 SAINS bit field in the SADC1 register.  
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 SAINS bit field, the corresponding pin 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 SACS bit field. 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 SAINS bit field, the corresponding external input pin must be switched to a non-existent channel input by setting the SACS3~SACS0 bits with a value from 1000 to 1111. The desired internal analog signal then can be selected by configuring the SAINS bit field. After this step, go to Step 6.
- Step 6  
Select the reference voltage source by configuring the SAVRS1~SAVRS0 bits in the SADC1 register. If the A/D converter power supply voltage or the operational amplifier output voltage is selected, the external reference input pin function must be disabled by properly configuring the corresponding pin-shared control bits.
- Step 7  
Select A/D converter output data format by setting the ADRFS bit in the SADC0 register.
- Step 8  
If the A/D conversion interrupt is used, the interrupt control registers must be correctly configured to ensure the A/D interrupt function is active. The master interrupt control bit, EMI, and the A/D converter interrupt control bit, ADE, must both be set high in advance.
- Step 9  
The A/D conversion procedure can now be initiated 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 setting bit ADCEN low in the SADC0 register. When this happens, the internal A/D converter circuits will not consume power irrespective of what analog voltage is applied to their input lines. If the A/D converter input lines are used as normal I/Os, then care must be taken as if the input voltage is not at a valid logic level, then this may lead to some increase in power consumption.

## A/D Transfer Function

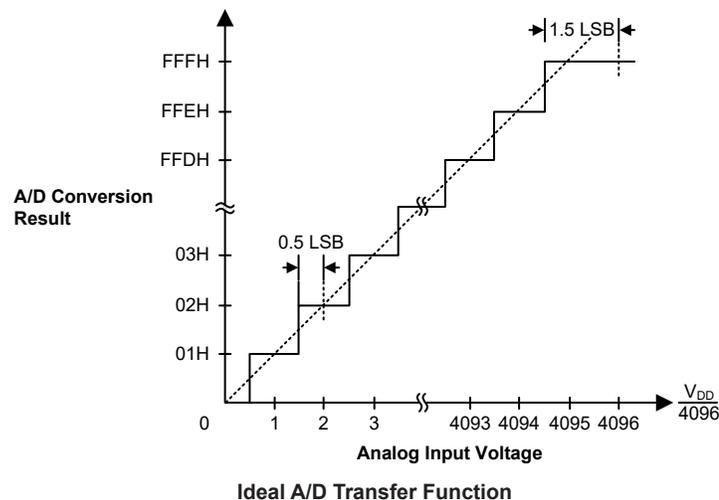
As the devices contain a 12-bit A/D converter, its full-scale converted digitised value is equal to FFFH. Since the full-scale analog input value is equal to the actual A/D converter reference voltage,  $V_{REF}$  voltage, this gives a single bit analog input value  $V_{DD}$  divided by 4096.

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

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

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

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



## A/D Programming Examples

The following two programming examples illustrate how to setup and implement an A/D conversion. In the first example, the method of polling the 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.

### Example1: using an ADBZ polling method to detect the end of conversion

```

clr ADE          ; disable ADC interrupt
mov a,03h        ; select fsys/8 as A/D clock and
mov SADC1,a      ; select external channel input and external reference input
mov a,01h        ; set PBS0 to configure pin AN0
mov PBS0,a
mov a, 02h       ; set PAS1 to configure pin VREF
    
```

```
mov PAS0, a
mov a,20h
mov SADC0,a      ; enable A/D and connect AN0 channel to A/D converter
:
:
start_conversion:
clr START      ; high pulse on start bit to initiate conversion
set START      ; reset A/D
clr START      ; start A/D
polling_EOC:
sz ADBZ        ; poll the SADC0 register ADBZ bit to detect end of A/D conversion
jmp polling_EOC ; continue polling
mov a,SADOL    ; read low byte conversion result value
mov SADOL_buffer,a ; save result to user defined register
mov a,SAD0H    ; read high byte conversion result value
mov SADOH_buffer,a ; save result to user defined register
:
:
jmp start_conversion ; start next A/D conversion
```

**Example2: using the interrupt method to detect the end of conversion**

```
clr ADE        ; disable ADC interrupt
mov a,03h      ; select fsys/8 as A/D clock and
mov SADC1,a    ; select external channel input and external reference input
mov a,01h      ; set PBS0 to configure pin AN0
mov PBS0,a
mov a, 02h     ; set PAS1 to configure pin VREF
mov PAS0, a
mov a,20h
mov SADC0,a    ; enable A/D and connect AN0 channel to A/D converter
:
:
Start_conversion:
clr START      ; high pulse on START bit to initiate conversion
set START      ; reset A/D
clr START      ; start A/D
clr ADF        ; clear ADC interrupt request flag
set ADE        ; enable ADC interrupt
set EMI        ; enable global interrupt
:
:
ADC_ISR:       ; ADC interrupt service routine
mov acc_stack,a ; save ACC to user defined memory
mov a,STATUS
mov status_stack,a ; save STATUS to user defined memory
:
:
mov a, SADOL   ; read low byte conversion result value
mov SADOL_buffer,a ; save result to user defined register
mov a, SADOH   ; read high byte conversion result value
mov SADOH_buffer,a ; save result to user defined register
:
:
EXIT_INT_ISR:
mov a,status_stack
mov STATUS,a   ; restore STATUS from user defined memory
mov a,acc_stack ; restore ACC from user defined memory
reti
```

### Nebulizer Resonance Detector

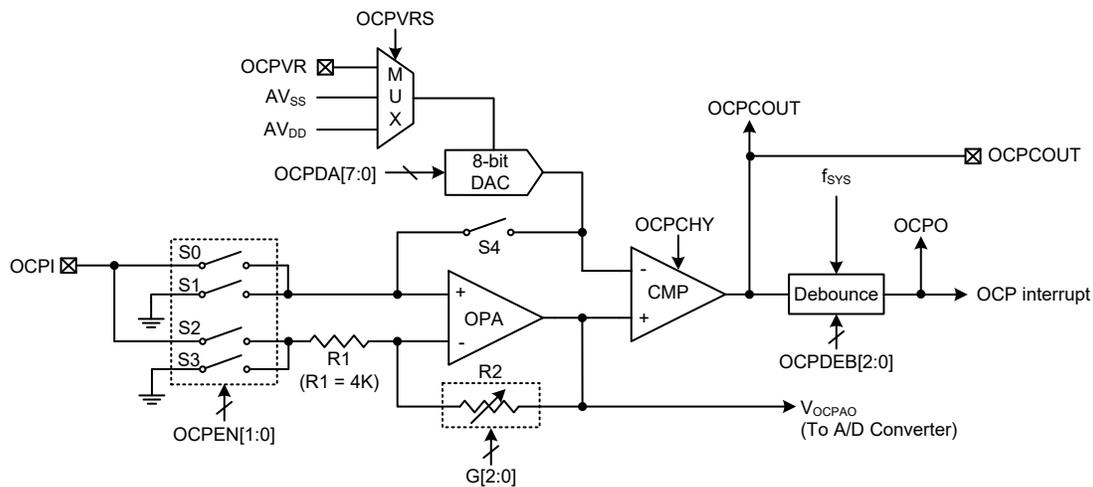
Please refer Holtek Application Notes.

### Water Shortage Protection

Developed by users.

## Over Current Protection – OCP

The device includes an over current protection function which provides a protection mechanism for applications. The current on the OCPI pin is converted to a relevant voltage level according to the current value using the OCP operational amplifier. It is then compared with a reference voltage generated by an 8-bit D/A converter. When an over current event occurs, an OCP interrupt will be generated if the corresponding interrupt control is enabled.



Note: As the OCP function relevant external pins are pin-shared with general I/O or other functions, before using the OCP function, make sure the corresponding pin-shared function registers be set properly.

### Over Current Protection Circuit

### Over Current Protection Operation

The OCP circuit is used to prevent the input current from exceeding a specific level. The current on the OCPI pin is converted to a voltage and then amplified by the OCP operational amplifier with a programmable gain from 1 to 50 selected by the G2~G0 bits in the OCPC1 register. This is known as a Programmable Gain Amplifier or PGA. This PGA can also be configured to operate in the non-inverting, inverting or input offset calibration mode determined by the OCPEN1~OCPEN0 bits in the OCPC0 register. After the current is converted and amplified to a specific voltage level, it will be compared with a reference voltage provided by an 8-bit D/A converter. The 8-bit D/A converter reference voltage can be supplied by AV<sub>DD</sub>, AV<sub>SS</sub> or external OCPVR pin, which is selected by the OCPVRS[1:0] bits in the OCPC0 register. The comparator output, OCPCOUT, will first be filtered with a certain de-bounce time period selected by the OCPDEB2~OCPDEB0 bits in the OCPC1 register. Then a filtered OCP digital comparator output, OCPO, is obtained to indicate whether an over current condition occurs or not. The OCPO bit will be set to 1 if an over current condition occurs. Otherwise, the OCPO bit is zero. Once an over current event occurs, i.e., the converted voltage of the OCP input current is greater than the reference voltage, the corresponding interrupt will be generated if the relevant interrupt control bit is enabled.

Note that the debounce clock,  $f_{DEB}$ , comes from the system clock,  $f_{SYS}$ . The operational amplifier output voltage can be read out by the A/D converter through an A/D internal input channel. The D/A converter output voltage is controlled by the OCPDA register and the D/A converter output is defined as below:

$$DAC V_{OUT} = (D/A \text{ converter reference voltage} / 256) \times OCPDA[7:0]$$

### Over Current Protection Registers

Overall operation of the over current protection is controlled using several registers. The OCPDA register is used to provide the reference voltage for the over current protection circuit. The OCPOCAL and OCPCCAL registers are used to cancel out the operational amplifier and comparator input offset. The OCPC0 and OCPC1 registers are control registers which control the OCP function, D/A converter reference voltage selection, PGA gain selection, comparator de-bounce time together with the hysteresis function.

Register Name	Bit							
	7	6	5	4	3	2	1	0
OCPC0	OCPEN1	OCPEN0	OCPVRS1	OCPVRS0	OCPCHY	—	—	OCPO
OCPC1	—	—	G2	G1	G0	OCPDEB2	OCPDEB1	OCPDEB0
OCPDA	D7	D6	D5	D4	D3	D2	D1	D0
OCPOCAL	OCPOOFM	OCPORSP	OCPOOF5	OCPOOF4	OCPOOF3	OCPOOF2	OCPOOF1	OCPOOF0
OCPCCAL	OCPCOUT	OCPCOFM	OCPCRSF	OCPCOF4	OCPCOF3	OCPCOF2	OCPCOF1	OCPCOF0

**OCPC Register List**

#### • OCPC0 Register

Bit	7	6	5	4	3	2	1	0
Name	OCPEN1	OCPEN0	OCPVRS1	OCPVRS0	OCPCHY	—	—	OCPO
R/W	R/W	R/W	R/W	R/W	R/W	—	—	R
POR	0	0	0	0	0	—	—	0

- Bit 7~6     **OCPEN1~OCPEN0**: OCP function operating mode selection  
 00: OCP function is disabled, S1 and S3 on, S0 and S2 off  
 01: Non-inverting mode, S0 and S3 on, S1 and S2 off  
 10: Inverting mode, S1 and S2 on, S0 and S3 off  
 11: Calibration mode, S1 and S3 on, S0 and S2 off
- Bit 5~4     **OCPVRS1~OCPVRS0**: OCP D/A converter reference voltage selection  
 00: From  $AV_{DD}$   
 01: From OCPVR pin  
 10: From  $AV_{SS}$   
 11: From  $AV_{DD}$
- Bit 3       **OCPCHY**: OCP comparator hysteresis function control  
 0: Disable  
 1: Enable
- Bit 2~1     Unimplemented, read as “0”
- Bit 0       **OCPO**: OCP digital output bit  
 0: No over current situation occurred  
 1: Over current situation occurred

• **OCPC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	G2	G1	G0	OCPDEB2	OCPDEB1	OCPDEB0
R/W	—	—	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	—	0	0	0	0	0	0

Bit 7~6 Unimplemented, read as “0”

Bit 5~3 **G2~G0**: PGA R2/R1 ratio selection

000: Unity gain buffer (non-inverting mode) or R2/R1=1(inverting mode)

001: R2/R1=5

010: R2/R1=10

011: R2/R1=15

100: R2/R1=20

101: R2/R1=30

110: R2/R1=40

111: R2/R1=50

These bits are used to select the R2/R1 ratio to obtain various gain values for inverting and non-inverting mode. The calculating formula of the PGA gain for the inverting and non-inverting mode is described in the “Input Voltage Range” section.

Bit 2~0 **OCPDEB2~OCPDEB0**: OCP output filter debounce time selection

000: Bypass, without debounce

001:  $(1\sim 2) \times t_{DEB}$

010:  $(3\sim 4) \times t_{DEB}$

011:  $(7\sim 8) \times t_{DEB}$

100:  $(15\sim 16) \times t_{DEB}$

101:  $(31\sim 32) \times t_{DEB}$

110:  $(63\sim 64) \times t_{DEB}$

111:  $(127\sim 128) \times t_{DEB}$

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

• **OCPDA 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**: OCP D/A converter output voltage control bits

DAC  $V_{OUT}=(D/A \text{ converter reference voltage}/256) \times \text{OCPDA}[7:0]$

• **OCPOCAL Register**

Bit	7	6	5	4	3	2	1	0
Name	OCPOOFM	OCPORSP	OCPOOF5	OCPOOF4	OCPOOF3	OCPOOF2	OCPOOF1	OCPOOF0
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 **OCPOOFM**: OCP Operational Amplifier Input Offset Calibration Mode enable control

0: Input Offset Calibration Mode disabled

1: Input Offset Calibration Mode enabled

This bit is used to control the OCP operational amplifier input offset Calibration function. The OCPEN1 and OCPEN0 bits must first be set to “11” and then the OCPOOFM bit must be set to 1 followed by the OCPCOFM bit being setting to 0, then the operational amplifier input offset Calibration mode will be enabled. Refer to the “Operational Amplifier Input Offset Calibration” section for the detailed offset Calibration procedures.

- Bit 6      **OCPORSP**: OCP Operational Amplifier Input Offset Voltage Calibration Reference selection  
             0: Select negative input as the reference input  
             1: Select positive input as the reference input
- Bit 5~0    **OCPOOF5~OCPOOF0**: OCP Operational Amplifier Input Offset Voltage Calibration value  
             This 6-bit field is used to perform the operational amplifier input offset Calibration operation and the value for the OCP operational amplifier input offset Calibration can be restored into this bit field. More detailed information is described in the “Operational Amplifier Input Offset Calibration” section.

• **OCPCAL Register**

Bit	7	6	5	4	3	2	1	0
Name	OCPCOUT	OCPCOFM	OCPCRSP	OCPCOF4	OCPCOF3	OCPCOF2	OCPCOF1	OCPCOF0
R/W	R	R/W						
POR	0	0	0	1	0	0	0	0

- Bit 7      **OCPCOUT**: OCP Comparator Output, positive logic (read only)  
             0: Positive input voltage < Negative input voltage  
             1: Positive input voltage > Negative input voltage  
             This bit is used to indicate whether the positive input voltage is greater than the negative input voltage when the OCP operates in the input offset Calibration mode. If the OCPCOUT is set to 1, the positive input voltage is greater than the negative input voltage. Otherwise, the positive input voltage is less than the negative input voltage.
- Bit 6      **OCPCOFM**: OCP Comparator Input Offset Calibration Mode enable control  
             0: Input Offset Calibration Mode disabled  
             1: Input Offset Calibration Mode enabled  
             This bit is used to control the OCP comparator input offset Calibration function. The OCPEN1 and OCPEN0 bits must first be set to “11” and then the OCPCOFM bit must be set to 1 followed by the OCPOOFM bit being setting to 0, then the comparator input offset calibration mode will be enabled. Refer to the “Comparator Input Offset Calibration” section for the detailed offset calibration procedures.
- Bit 5      **OCPCRSP**: OCP Comparator Input Offset Calibration Reference Input select  
             0: Select negative input as the reference input  
             1: Select positive input as the reference input
- Bit 4~0    **OCPCOF4~OCPCOF0**: OCP Comparator Input Offset Calibration value  
             This 5-bit field is used to perform the comparator input offset calibration operation and the value for the OCP comparator input offset calibration can be restored into this bit field. More detailed information is described in the “Comparator Input Offset Calibration” section.

**Input Voltage Range**

Together with different PGA operating modes, the input voltage on the OCP pin can be positive or negative for flexible operation. The PGA output for the positive or negative input voltage is calculated based on different formulas and described by the following.

- For input voltages  $V_{IN} > 0$ , the PGA operates in the non-inverting mode and the PGA output is obtained using the formula below:

$$V_{OUT} = (1 + R2/R1) \times V_{IN}$$

- When the PGA operates in the non-inverting mode by setting the OCPEN[1:0] to “01” with unity gain select by setting the G[2:0] to “000”, the PGA will act as a unit-gain buffer whose output is equal to  $V_{IN}$ .

$$V_{OUT} = V_{IN}$$

- For input voltages  $0 > V_{IN} > -0.2V$ , the PGA operates in the inverting mode and the PGA output is obtained using the formula below. Note that if the input voltage is negative, it cannot be lower than  $-0.2V$  which will result in current leakage.

$$V_{OUT} = -(R2/R1) \times V_{IN}$$

### Input Offset Calibration

The OCP circuit has four operating modes controlled by the OCPEN[1:0] bit field, one of them is calibration mode. In calibration mode, operational amplifier and comparator offset can be calibrated.

#### Operational Amplifier Input Offset Calibration

Step 1. Set OCPEN[1:0]=11, OCPOOFM=1, OCPCOFM=0 and OCPORSP=1, the OCP will operate in the operational amplifier input offset calibration mode.

Step 2. Set OCPOOF[5:0]=000000 and then read the OCPCOUT bit.

Step 3. Increase the OCPOOF[5:0] value by 1 and then read the OCPCOUT bit.

If the OCPCOUT bit state has not changed, then repeat Step 3 until the OCPCOUT bit state has changed.

If the OCPCOUT bit state has changed, record the OCPOOF value as  $V_{OOS1}$  and then go to Step 4.

Step 4. Set OCPOOF[5:0]=111111 and read the OCPCOUT bit.

Step 5. Decrease the OCPOOF[5:0] value by 1 and then read the OCPCOUT bit.

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

If the OCPCOUT bit state has changed, record the OCPOOF value as  $V_{OOS2}$  and then go to Step 6.

Step 6. Restore the operational amplifier input offset calibration value  $V_{OOS}$  into the OCPOOF[5:0] bit field. The offset Calibration procedure is now finished.

$$\text{Where } V_{OOS} = (V_{OOS1} + V_{OOS2}) / 2$$

Note: S4 is off. In this mode, the operational amplifier outputs to OCPCOUT bypassing the comparator.

#### Comparator Input Offset Calibration

Step 1. Set OCPEN[1:0]=11, OCPCOFM=1 and OCPOOFM=0, the OCP will now operate in the comparator input offset calibration mode.

Step 2. Set OCPCOF[4:0]=00000 and read the OCPCOUT bit.

Step 3. Increase the OCPCOF[4:0] value by 1 and then read the OCPCOUT bit.

If the OCPCOUT bit state has not changed, then repeat Step 3 until the OCPCOUT bit state has changed.

If the OCPCOUT bit state has changed, record the OCPCOF value as  $V_{COS1}$  and then go to Step 4.

Step 4. Set OCPCOF[4:0]=11111 and then read the OCPCOUT bit.

Step 5. Decrease the OCPCOF[4:0] value by 1 and then read the OCPCOUT bit.

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

If the OCPCOUT bit state has changed, record the OCPCOF value as  $V_{COS2}$  and then go to Step 6.

Step 6. Restore the comparator input offset calibration value  $V_{COS}$  into the OCPCOF[4:0] bit field.  
 The offset Calibration procedure is now finished.

$$\text{Where } V_{COS} = (V_{COS1} + V_{COS2}) / 2$$

Note: S4 is on and the D/A converter is off. This situation is only available for comparator calibration procedure. In the normal operation mode, S4 is off.

## Touch Key Function

The device provides 8 touch key functions. The touch key function is fully integrated and requires no external components, allowing touch key functions to be implemented by the simple manipulation of internal registers.

### Touch Key Structure

The touch keys are pin-shared with the I/O pins, with the desired function chosen via the corresponding selection register bits. Keys are organised into several groups, with each group known as a module and having a module number, M0 to M1. Each module is a fully independent set of four Touch Keys and each Touch Key has its own oscillator. Each module contains its own control logic circuits and register set. Examination of the register names will reveal the module number it is referring to.

Total Key Number	Touch Key		Shared I/O Pin
8	Mn (n=0~1)	M0 M1	KEY1~KEY4 KEY5~KEY8

**Touch Key Structure**

### Touch Key Registers Description

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

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

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

Register Name	Bit							
	7	6	5	4	3	2	1	0
TKTMR	D7	D6	D5	D4	D3	D2	D1	D0
TKC0	—	TKRCOV	TKST	TKCFOV	TK16OV	TSCS	TK16S1	TK16S0
TKC1	—	—	—	—	—	—	TKFS1	TKFS0
TK16DL	D7	D6	D5	D4	D3	D2	D1	D0
TK16DH	D15	D14	D13	D12	D11	D10	D9	D8
TKMn16DL	D7	D6	D5	D4	D3	D2	D1	D0
TKMn16DH	D15	D14	D13	D12	D11	D10	D9	D8
TKMnROL	D7	D6	D5	D4	D3	D2	D1	D0
TKMnROH	—	—	—	—	—	—	D9	D8
TKMnC0	MnMXS1	MnMXS0	MnDFEN	MnFILEN	MnSOFC	MnSOF2	MnSOF1	MnSOF0
TKMnC1	MnTSS	—	MnROEN	MnKOEN	MnK4EN	MnK3EN	MnK2EN	MnK1EN

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

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

• **TKC0 Register**

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

Bit 7 Unimplemented, read as “0”

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

- 0: No overflow occurs
- 1: Overflow occurs

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

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

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

• **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

• **TK16DL Register**

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

- Bit 7~0    **D7~D0**: Touch key function 16-bit counter low byte contents

• **TK16DH Register**

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

Bit 7~0      **D15~D8**: Touch key function 16-bit counter high byte contents

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.

• **TKMn16DL Register (n=0~1)**

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**: Touch key module n 16-bit C/F counter low byte contents

• **TKMn16DH Register (n=0~1)**

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

Bit 7~0      **D15~D8**: Touch key module n 16-bit C/F counter high byte contents

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

• **TKMnROL Register (n=0~1)**

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

Bit 7~0      **D7~D0**: Touch key module n reference oscillator internal capacitor selection

• **TKMnROH Register (n=0~1)**

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**: Touch key module n reference oscillator internal capacitor selection

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

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

• **TKMnC0 Register (n=0~1)**

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

- Bit 7~6 **MnMXS1~MnMXS0**: Touch key module n multiplexer key selection  
 00: KEY1  
 01: KEY2  
 10: KEY3  
 11: KEY4
- Bit 5 **MnDFEN**: Touch key module n double-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 **MnFILEN**: Touch key module n filter function control  
 0: Disable  
 1: Enable
- Bit 3 **MnSOFC**: Touch key module n C-to-F oscillator frequency hopping function control selection  
 0: Controlled by the MnSOF2~MnSOF0  
 1: Controlled by hardware circuit  
 This bit is used to select the touch key oscillator frequency hopping function control method. When this bit is set to 1, the key oscillator frequency hopping function is controlled by the hardware circuit regardless of the MnSOF2~MnSOF0 bits value.
- Bit 2~0 **MnSOF2~MnSOF0**: Touch key module n Reference and Key oscillators hopping frequency selection  
 000: 1.020MHz  
 001: 1.040MHz  
 010: 1.059MHz  
 011: 1.074MHz  
 100: 1.085MHz  
 101: 1.099MHz  
 110: 1.111MHz  
 111: 1.125MHz  
 These bits are used to select the touch key oscillator frequency for the hopping function. Note that these bits are only available when the MnSOFC bit is cleared to 0.  
 The frequency mentioned here will be changed when the external or internal capacitor is with different values. If the touch key operates at 1MHz frequency, users can adjust the frequency in scale when any other frequency is selected.

• **TKMnC1 Register (n=0~1)**

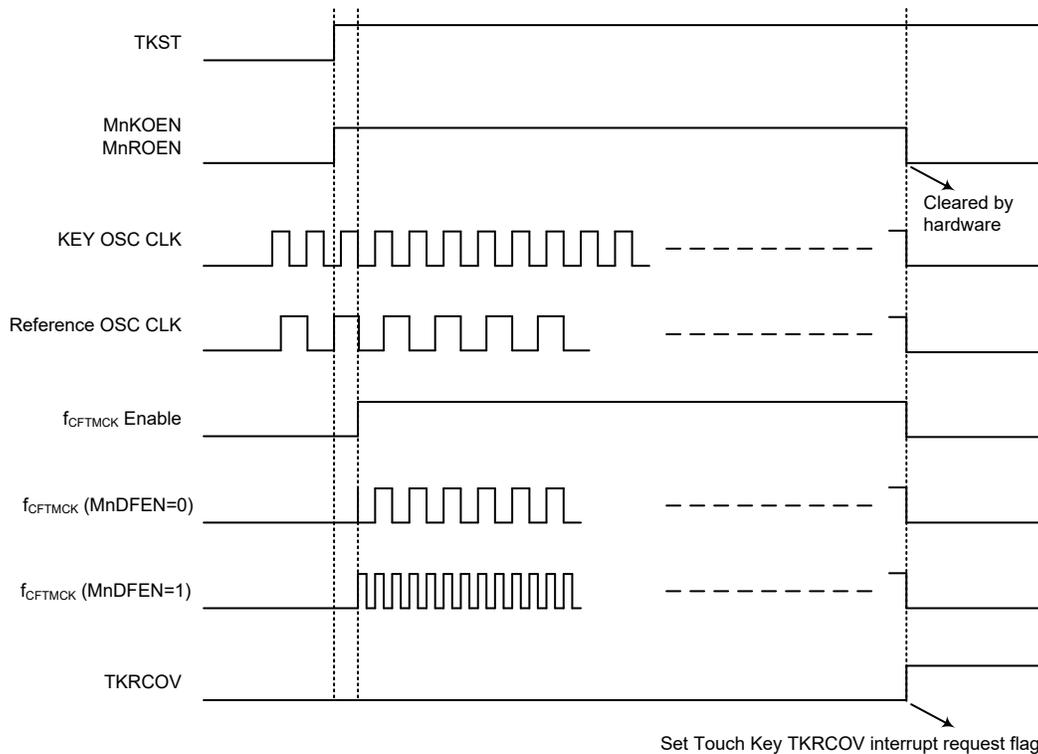
Bit	7	6	5	4	3	2	1	0
Name	MnTSS	—	MnROEN	MnKOEN	MnK4EN	MnK3EN	MnK2EN	MnK1EN
R/W	R/W	—	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	—	0	0	0	0	0	0

- Bit 7 **MnTSS**: Touch key module n time slot counter clock source selection  
 0: Touch key module n reference oscillator  
 1:  $f_{sys}/4$
- Bit 6 Unimplemented, read as “0”

- Bit 5      **MnROEN**: Touch key module n Reference oscillator enable control  
0: Disable  
1: Enable
- Bit 4      **MnKOEN**: Touch key module n Key oscillator enable control  
0: Disable  
1: Enable
- Bit 3      **MnK4EN**: Touch key module n KEY4 enable control  
0: Disable  
1: Enable
- Bit 2      **MnK3EN**: Touch key module n KEY3 enable control  
0: Disable  
1: Enable
- Bit 1      **MnK2EN**: Touch key module n KEY2 enable control  
0: Disable  
1: Enable
- Bit 0      **MnK1EN**: Touch key module n 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 the number of generated clock cycles from the sense oscillator during this fixed time period touch key actions can be determined.



**Touch Key Module Timing Diagram**

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

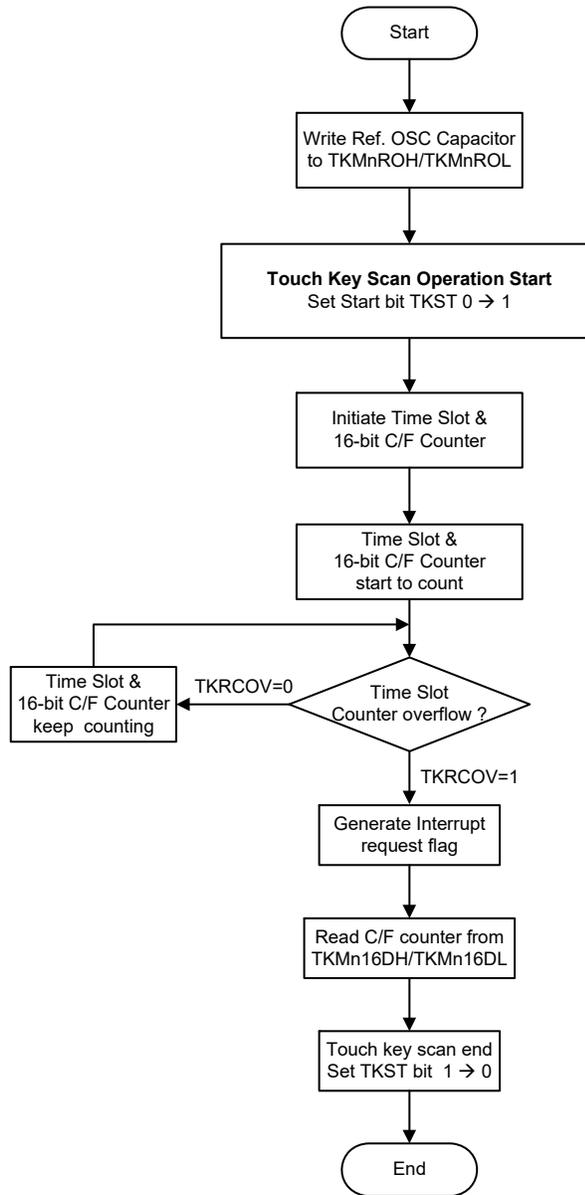
During this reference clock fixed interval, the number of clock cycles generated by the sense oscillator is measured, and it is this value that is used to determine if a touch action has been made or not. At the end of the fixed reference clock time interval a Touch Key interrupt signal will be generated.

Using the TSCS bit in the TKC1 register can select the module 0 time slot counter as the time slot counter for all modules. All modules use the same started signal, TKST, in the TKC0 register. The touch key module 16-bit C/F counter, 16-bit counter, 5-bit time slot unit period counter in the module will be automatically cleared when the TKST bit is cleared to zero, but the 8-bit programmable time slot counter will not be cleared. The overflow time is setup by user. When the TKST bit changes from low to high, the 16-bit C/F counter, 16-bit counter, 5-bit time slot unit period counter and 8-bit time slot timer counter will be automatically switched on.

The key oscillator and reference oscillator in the module will be automatically stopped and the 16-bit C/F counter, 16-bit counter, 5-bit time slot unit period counter and 8-bit time slot timer counter will be automatically switched off when the time slot counter overflows. The clock source for the time slot counter is sourced from the reference oscillator or  $f_{SYS}/4$  which is selected using the MnTSS bit in the TKMC1 register. The reference oscillator and key oscillator will be enabled by setting the MROEN bit and MnKOEN bits in the TKMC1 register.

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.

Touch Key Scan Operation Flowchart



Touch Key Scan Operation Flowchart

## Touch Key Interrupt

The touch key only has single interrupt, when the touch key module time slot counter overflows, an actual touch key interrupt will take place. The touch keys mentioned here are the keys which are enabled. The 16-bit C/F counter, 16-bit counter, 5-bit time slot unit period counter and 8-bit time slot counter in the module will be automatically cleared.

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. More details regarding the touch key interrupts are located in the interrupt section of the datasheet.

## Progrsmming Considerations

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

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

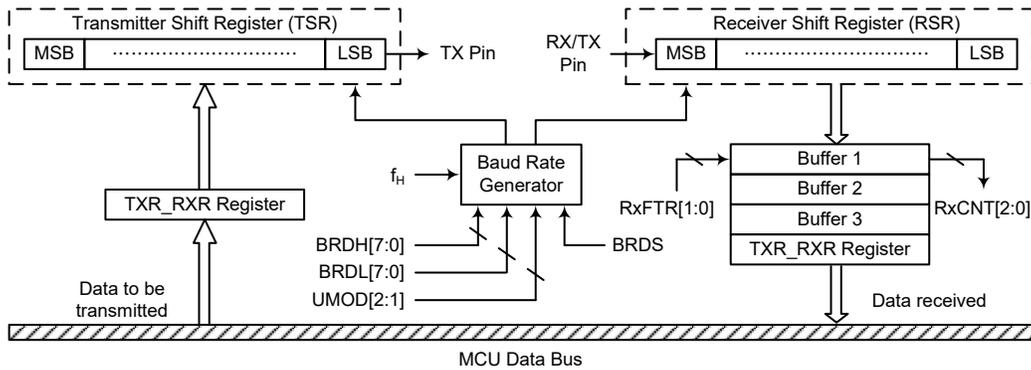
## UART Interface

The device contains an integrated full-duplex or half-duplex asynchronous serial communications UART interface that enables communication with external devices that contain a serial interface. The UART function has many features and can transmit and receive data serially by transferring a frame of data with eight or nine data bits per transmission as well as being able to detect errors when the data is overwritten or incorrectly framed. The UART function possesses its own internal interrupt which can be used to indicate when a reception occurs or when a transmission terminates.

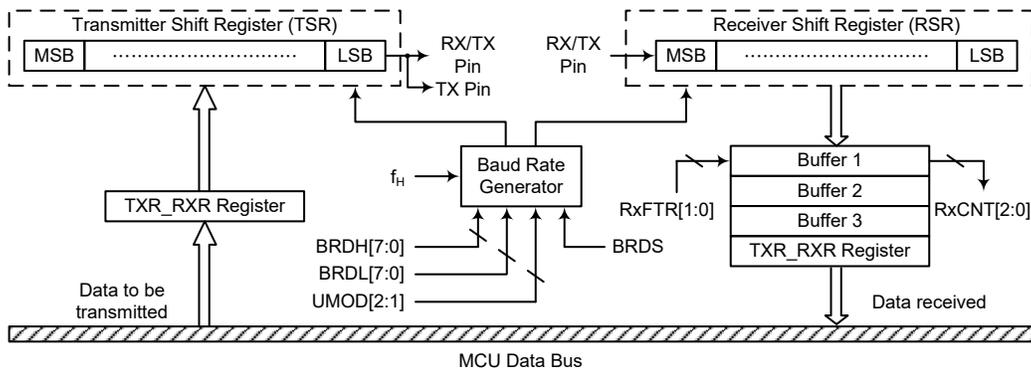
The integrated UART function contains the following features:

- Full-duplex or half-duplex (single wire mode) asynchronous communication
- 8 or 9 bits character length
- Even, odd, mark, space or no parity options
- One or two stop bits configurable for receiver
- Two stop bits for transmitter
- Baud rate generator with 16-bit prescaler
- Parity, framing, noise and overrun error detection
- Support for interrupt on address detect (last character bit=1)
- Separately enabled transmitter and receiver
- 4-byte Deep FIFO Receive Data Buffer
- 1-byte Deep FIFO Transmit Data Buffer
- RX/TX pin wake-up function
- Transmit and receive interrupts

- Interrupts can be triggered by the following conditions:
  - ♦ Transmitter Empty
  - ♦ Transmitter Idle
  - ♦ Receiver reaching FIFO trigger level
  - ♦ Receiver Overrun
  - ♦ Address Mode Detect



**UART Data Transfer Block Diagram – SWM=0**



**UART Data Transfer Block Diagram – SWM=1**

### UART External Pins

To communicate with an external serial interface, the internal UART has two external pins known as TX pin and RX/TX pin. The TX and RX/TX pins are the UART transmitter and receiver pins respectively. The TX and RX/TX pin function should first be selected by the pin-shared function selection register before the UART function is used. Along with the UARTEN bit, the TXEN and RXEN bits, if set, will setup these pins to their respective TX output and RX/TX input conditions and disable any pull-high resistor option which may exist on the TX and RX/TX pins. When the TX or RX/TX pin function is disabled by clearing the UARTEN, TXEN or RXEN bit, the TX or RX/TX pin will be set to a floating state. At this time whether the internal pull-high resistor is connected to the TX or RX/TX pin or not is determined by the corresponding I/O pull-high function control bit.

### UART Single Wire Mode

The UART function also supports a Single Wire Mode communication which is selected using the SWM bit in the UCR3 register. When the SWM bit is set high, the UART function will be in the single wire mode. In the single wire mode, a single RX/TX pin can be used to transmit and receive data depending upon the corresponding control bits. When the RXEN bit is set high, the RX/TX pin

is used as a receiver pin. When the RXEN bit is cleared to zero and the TXEN bit is set high, the RX/TX pin will act as a transmitter pin.

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

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

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

### UART Data Transfer Scheme

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

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

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

### UART Status and Control Registers

There are eight control registers associated with the UART function. The SWM bit in the UCR3 register is used to enable/disable the UART Single Wire Mode. The USR, UCR1, UCR2, UFCR and RxCNT registers control the overall function of the UARTn, while the BRDH and BRDL registers control the Baud rate. The actual data to be transmitted and received on the serial interface is managed through the TXR\_RXRn data register.

Register Name	Bit							
	7	6	5	4	3	2	1	0
USR	PERR	NF	FERR	OERR	RIDLE	RXIF	TIDLE	TXIF
UCR1	UARTEN	BNO	PREN	PRT1	PRT0	TXBRK	RX8	TX8
UCR2	TXEN	RXEN	STOPS	ADDEN	WAKE	RIE	TIIE	TEIE
UCR3	—	—	—	—	—	—	—	SWM
TXR_RXR	TXRX7	TXRX6	TXRX5	TXRX4	TXRX3	TXRX2	TXRX1	TXRX0
BRDH	D7	D6	D5	D4	D3	D2	D1	D0
BRDL	D7	D6	D5	D4	D3	D2	D1	D0
UFCR	—	—	UMOD2	UMOD1	UMOD0	BRDS	RxFTR1	RxFTR0
RxCNT	—	—	—	—	—	D2	D1	D0

**UART Register List**

• **USR Register**

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

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

**Bit 7 PERR:** Parity error flag  
 0: No parity error is detected  
 1: Parity error is detected  
 The PERR flag is the parity error flag. When this read only flag is “0”, it indicates a parity error has not been detected. When the flag is “1”, it indicates that the parity of the received word is incorrect. This error flag is applicable only if the parity is enabled and the parity type (odd, even, mark or space) is selected. The flag can also be cleared to zero by a software sequence which involves a read to the status register USR followed by an access to the TXR\_RXR data register.

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

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

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

**Bit 3 RIDLE:** Receiver status  
 0: Data reception is in progress (Data being received)  
 1: No data reception is in progress (Receiver is idle)  
 The RIDLE flag is the receiver status flag. When this read only flag is “0”, it indicates that the receiver is between the initial detection of the start bit and the completion of the stop bit. When the flag is “1”, it indicates that the receiver is idle. Between the completion of the stop bit and the detection of the next start bit, the RIDLE bit is “1” indicating that the UART receiver is idle and the RX/TX pin stays in logic high condition.

- Bit 2**      **RXIF:** Receive TXR\_RXR data register status  
 0: TXR\_RXR data register is empty  
 1: TXR\_RXR data register has available data and reach Receiver FIFO trigger level  
 The RXIF flag is the receive data register status flag. When this read only flag is “0”, it indicates that the TXR\_RXR read data register is empty. When the flag is “1”, it indicates that the TXR\_RXR read data register contains new data and reaches the Receiver FIFO trigger level. When the contents of the shift register are transferred to the TXR\_RXR register and reach Receiver FIFO trigger level, an interrupt is generated if RIE=1 in the UCR2 register. If one or more errors are detected in the received word, the appropriate receive-related flags NF, FERR, and/or PERR are set within the same clock cycle. The RXIF flag will eventually be cleared to zero when the USR register is read with RXIF set, followed by a read from the TXR\_RXR register, and if the TXR\_RXR register has no more new data available.
- Bit 1**      **TIDLE:** Transmission idle  
 0: Data transmission is in progress (Data being transmitted)  
 1: No data transmission is in progress (Transmitter is idle)  
 The TIDLE flag is known as the transmission complete flag. When this read only flag is “0”, it indicates that a transmission is in progress. This flag will be set high when the TXIF flag is “1” and when there is no transmit data or break character being transmitted. When TIDLE is equal to “1”, the TX pin becomes idle with the pin state in logic high condition. The TIDLE flag is cleared to zero by reading the USR register with TIDLE set and then writing to the TXR\_RXR register. The flag is not generated when a data character or a break is queued and ready to be sent.
- Bit 0**      **TXIF:** Transmit TXR\_RXR data register status  
 0: Character is not transferred to the transmit shift register  
 1: Character has transferred to the transmit shift register (TXR\_RXR data register is empty)  
 The TXIF flag is the transmit data register empty flag. When this read only flag is “0”, it indicates that the character is not transferred to the transmitter shift register. When the flag is “1”, it indicates that the transmitter shift register has received a character from the TXR\_RXR data register. The TXIF flag is cleared to zero by reading the UART status register (USR) with TXIF set and then writing to the TXR\_RXR data register. Note that when the TXEN bit is set, the TXIF flag bit will also be set since the transmit data register is not yet full.

• **UCR1 Register**

The UCR1 register together with the UCR2 and UCR3 register are the three UART control registers that are used to set the various options for the UART function, such as overall on/off control, parity control, data transfer bit length, single wire mode communication etc. Further explanation on each of the bits is given below:

Bit	7	6	5	4	3	2	1	0
Name	UARTEN	BNO	PREN	PRT1	PRT0	TXBRK	RX8	TX8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R	W
POR	0	0	0	0	0	0	x	0

“x”: unknown

- Bit 7**      **UARTEN:** UART function enable control  
 0: Disable UART. TX and RX/TX pins are in a floating state  
 1: Enable UART. TX and RX/TX pins can function as UART pins  
 The UARTEN bit is the UART enable bit. When this bit is equal to “0”, the UART will be disabled and the RX/TX pin as well as the TX pin will be in a floating state. When the bit is equal to “1”, the UART will be enabled and the TX and RX/TX pins will function as defined by the SWM mode selection bit together with the TXEN and RXEN enable control bits.

When the UART is disabled, it will empty the buffer so any character remaining in the buffer will be discarded. In addition, the value of the baud rate counter will be reset. If the UART is disabled, all error and status flags will be reset. Also the TXEN, RXEN, TXBRK, RXIF, OERR, FERR, PERR and NF bits will be cleared to zero, while the TIDLE, TXIF and RIDLE bits will be set high. Other control bits in UCR1, UCR2, UCR3, BRDH and BRDL registers will remain unaffected. If the UART is active and the UARTEN bit is cleared to zero, all pending transmissions and receptions will be terminated and the module will be reset as defined above. When the UART is re-enabled, it will restart in the same configuration.

- Bit 6**     **BNO**: Number of data transfer bits selection  
             0: 8-bit data transfer  
             1: 9-bit data transfer
- This bit is used to select the data length format, which can have a choice of either 8-bit or 9-bit format. When this bit is equal to “1”, a 9-bit data length format will be selected. If the bit is equal to “0”, then an 8-bit data length format will be selected. If 9-bit data length format is selected, then bits RX8 and TX8 will be used to store the 9th bit of the received and transmitted data respectively.
- Note that the 9th bit of data if BNO=1, or the 8th bit of data if BNO=0, which is used as the parity bit, does not transfer to RX8 or TXR\_RXR.7 respectively when the parity function is enabled.
- Bit 5**     **PREN**: Parity function enable control  
             0: Parity function is disabled  
             1: Parity function is enabled
- This is the parity enable bit. When this bit is equal to “1”, the parity function will be enabled. If the bit is equal to “0”, then the parity function will be disabled.
- Bit 4~3**   **PRT1~PRT0**: Parity type selection bit  
             00: Even parity for parity generator  
             01: Odd parity for parity generator  
             10: Mark parity for parity generator  
             11: Space parity for parity generator
- These bits are the parity type selection bits. When these bits are equal to 00b, even parity type will be selected. If these bits are equal to 01b, then odd parity type will be selected. If these bits are equal to 10b, then a 1 (Mark) in the parity bit location will be selected. If these bits are equal to 11b, then a 0 (Space) in the parity bit location will be selected.
- Bit 2**     **TXBRK**: Transmit break character  
             0: No break character is transmitted  
             1: Break characters transmit
- The TXBRK bit is the Transmit Break Character bit. When this bit is “0”, there are no break characters and the TX pin operates normally. When the bit is “1”, there are transmit break characters and the transmitter will send logic zeros. When this bit is equal to “1”, after the buffered data has been transmitted, the transmitter output is held low for a minimum of a 13-bit length and until the TXBRK bit is reset.
- Bit 1**     **RX8**: Receive data bit 8 for 9-bit data transfer format (read only)
- This bit is only used if 9-bit data transfers are used, in which case this bit location will store the 9th bit of the received data known as RX8. The BNO bit is used to determine whether data transfers are in 8-bit or 9-bit format.
- Bit 0**     **TX8**: Transmit data bit 8 for 9-bit data transfer format (write only)
- This bit is only used if 9-bit data transfers are used, in which case this bit location will store the 9th bit of the transmitted data known as TX8. The BNO bit is used to determine whether data transfers are in 8-bit or 9-bit format.

• **UCR2 Register**

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

Bit	7	6	5	4	3	2	1	0
Name	TXEN	RXEN	STOPS	ADDEN	WAKE	RIE	TIIE	TEIE
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7 **TXEN**: UART Transmitter enabled control

- 0: UART transmitter is disabled
- 1: UART transmitter is enabled

The bit named TXEN is the Transmitter Enable Bit. When this bit is equal to “0”, the transmitter will be disabled with any pending data transmissions being aborted. In addition the buffers will be reset. In this situation the TX pin will be in a floating state. If the TXEN bit is equal to “1” and the UARTEN bit is also equal to “1”, the transmitter will be enabled and the TX pin will be controlled by the UART. Clearing the TXEN bit during a transmission will cause the data transmission to be aborted and will reset the transmitter. If this situation occurs, the TX pin will be in a floating state.

Bit 6 **RXEN**: UART Receiver enabled control

- 0: UART receiver is disabled
- 1: UART receiver is enabled

The bit named RXEN is the Receiver Enable Bit. When this bit is equal to “0”, the receiver will be disabled with any pending data receptions being aborted. In addition the receive buffers will be reset. In this situation the RX/TX pin will be in a floating state. If the RXEN bit is equal to “1” and the UARTEN bit is also equal to “1”, the receiver will be enabled and the RX/TX pin will be controlled by the UART. Clearing the RXEN bit during a reception will cause the data reception to be aborted and will reset the receiver. If this situation occurs, the RX/TX pin will be in a floating state.

Bit 5 **STOPS**: Number of Stop bits selection for receiver

- 0: One stop bit format is used
- 1: Two stop bits format is used

This bit determines if one or two stop bits are to be used for receiver. When this bit is equal to “1”, two stop bits are used. If this bit is equal to “0”, then only one stop bit is used. Two stop bits are used for transmitter.

Bit 4 **ADDEN**: Address detect function enable control

- 0: Address detect function is disabled
- 1: Address detect function is enabled

The bit named ADDEN is the address detect function enable control bit. When this bit is equal to “1”, the address detect function is enabled. When it occurs, if the 8th bit, which corresponds to TXRX7 if BNO=0 or the 9th bit, which corresponds to RX8 if BNO=1, has a value of “1”, then the received word will be identified as an address, rather than data. If the corresponding interrupt is enabled, an interrupt request will be generated each time the received word has the address bit set, which is the 8th or 9th bit depending on the value of BNO. If the address bit known as the 8th or 9th bit of the received word is “0” with the address detect function being enabled, an interrupt will not be generated and the received data will be discarded.

Bit 3 **WAKE**: RX/TX pin wake-up UART function enable control

- 0: RX/TX pin wake-up UART function is disabled
- 1: RX/TX pin wake-up UART function is enabled

This bit is used to control the wake-up UART function when a falling edge on the RX/TX pin occurs. Note that this bit is only available when the UART clock ( $f_{H}$ ) is switched off. There will be no RX/TX pin wake-up UART function if the UART clock ( $f_{H}$ ) exists. If the WAKE bit is set to 1 as the UART clock ( $f_{H}$ ) is switched off, a UART wake-up request will be initiated when a falling edge on the RX/TX pin occurs. When this request happens and the corresponding interrupt is enabled, an RX/TX pin wake-up UART interrupt will be generated to inform the MCU to wake up the UART function by switching on the UART clock ( $f_{H}$ ) via the application program. Otherwise, the UART function can not resume even if there is a falling edge on the RX/TX pin when the WAKE bit is cleared to 0.

- Bit 2     **RIE**: Receiver interrupt enable control  
           0: Receiver related interrupt is disabled  
           1: Receiver related interrupt is enabled
- This bit enables or disables the receiver interrupt. If this bit is equal to “1” and when the receiver overrun flag OERR or receive data available flag RXIF is set, the UART interrupt request flag will be set. If this bit is equal to “0”, the UART interrupt request flag will not be influenced by the condition of the OERR or RXIF flags.
- Bit 1     **TIE**: Transmitter Idle interrupt enable control  
           0: Transmitter idle interrupt is disabled  
           1: Transmitter idle interrupt is enabled
- This bit enables or disables the transmitter idle interrupt. If this bit is equal to “1” and when the transmitter idle flag TIDLE is set, due to a transmitter idle condition, the UART interrupt request flag will be set. If this bit is equal to “0”, the UART interrupt request flag will not be influenced by the condition of the TIDLE flag.
- Bit 0     **TEIE**: Transmitter Empty interrupt enable control  
           0: Transmitter empty interrupt is disabled  
           1: Transmitter empty interrupt is enabled
- This bit enables or disables the transmitter empty interrupt. If this bit is equal to “1” and when the transmitter empty flag TXIF is set, due to a transmitter empty condition, the UART interrupt request flag will be set. If this bit is equal to “0”, the UART interrupt request flag will not be influenced by the condition of the TXIF flag.

• **UCR3 Register**

The UCR3 register is used to enable the UART Single Wire Mode communication. As the name suggests in the single wire mode the UART communication can be implemented in one single line, RX/TX, together with the control of the RXEN and TXEN bits in the UCR2 register.

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

- Bit 7~1     Unimplemented, read as “0”
- Bit 0     **SWM**: Single Wire Mode enable control  
           0: Disable, the RX/TX pin is used as UART receiver function only  
           1: Enable, the RX/TX pin can be used as UART receiver or transmitter function controlled by the RXEN and TXEN bits
- Note that when the Single Wire Mode is enabled, if both the RXEN and TXEN bits are high, the RX/TX pin will just be used as UART receiver input.

• **TXR\_RXR Register**

The TXR\_RXR register is the data register which is used to store the data to be transmitted on the TX pin or being received from the RX/TX pin.

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

“x”: Unknown

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

• **BRDH 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**: Baud rate divider high byte

The baud rate divider BRD (BRDH/BRDL) defines the UART clock divider ratio.

$$\text{Baud Rate} = f_{ih} / (\text{BRD} + \text{UMOD} / 8)$$

BRD=16~65535 or 8~65535 depending on BRDS

Note: 1. BRD value should not be set to less than 16 when BRDS=0 or less than 8 when BRDS=1, otherwise errors may occur.

2. The BRDL must be written first and then BRDH, otherwise errors may occur.

3. The BRDH register should not be modified during data transmission process.

• **BRDL 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**: Baud rate divider low byte

The baud rate divider BRD (BRDH/BRDL) defines the UART clock divider ratio.

$$\text{Baud Rate} = f_{ih} / (\text{BRD} + \text{UMOD} / 8)$$

BRD=16~65535 or 8~65535 depending on BRDS

Note: 1. BRD value should not be set to less than 16 when BRDS=0 or less than 8 when BRDS=1, otherwise errors may occur.

2. The BRDL must be written first and then BRDH, otherwise errors may occur.

3. The BRDL register should not be modified during data transmission process.

• **UFCR Register**

The UFCR register is the FIFO control register which is used for UART modulation control, BRD range selection and trigger level selection for RXIF and interrupt.

Bit	7	6	5	4	3	2	1	0
Name	—	—	UMOD2	UMOD1	UMOD0	BRDS	RxFTR1	RxFTR0
R/W	—	—	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	—	0	0	0	0	0	0

Bit 7~6 Unimplemented, read as “0”

Bit 5~3 **UMOD2~UMOD0**: UART Modulation Control bits

The modulation control bits are used to correct the baud rate of the received or transmitted UART signal. These bits determine if the extra UART clock cycle should

be added in a UART bit time. The UMOD2~UMOD0 will be added to internal accumulator for every UART bit time. Until a carry to bit 3, the corresponding UART bit time increases a UART clock cycle.

Bit 2 **BRDS**: BRD range selection  
 0: BRD range is from 16 to 65535  
 1: BRD range is from 8 to 65535

The BRDS is used to control the sampling point in a UARTn bit time. If the BRDS bit is cleared to zero, the sampling point will be  $BRD/2$ ,  $BRD/2+1 \times f_{IH}$ , and  $BRD/2+2 \times f_{IH}$  in a UART bit time. If the BRDS bit is set high, the sampling point will be  $BRD/2-1 \times f_{IH}$ ,  $BRD/2$ , and  $BRD/2+2 \times f_{IH}$  in a UART bit time.

Note that the BRDS bit should not be modified during data transmission process.

Bit 1~0 **RxFTR1~RxFTR0**: Receiver FIFO trigger level (bytes)  
 00: 4 bytes in Receiver FIFO  
 01: 1 or more bytes in Receiver FIFO  
 10: 2 or more bytes in Receiver FIFO  
 11: 3 or more bytes in Receiver FIFO

For the receiver these bits define the number of received data bytes in the Receiver FIFO that will trigger the RXIF bit being set high, an interrupt will also be generated if the RIE bit is enabled. To prevent OERR from being set high, the receiver FIFO trigger level can be set to 2 bytes, avoiding an overrun state that cannot be processed by the program in time when more than 4 data bytes are received.

After the reset the Receiver FIFO is empty.

• **RxCNT Register**

The RxCNT register is the counter used to indicate the number of received data bytes in the Receiver FIFO which have not been read by the MCU. This register is read only.

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

Bit 7~3 Unimplemented, read as “0”

Bit 2~0 **D2~D0**: Receiver FIFO counter

The RxCNT register is the counter used to indicate the number of received data bytes in the Receiver FIFO which is not read by the MCU. When Receiver FIFO receives one byte data, the RxCNT will increase by one; when the MCU reads one byte data from the Receiver FIFO, the RxCNT will decrease by one. If there are 4 bytes of data in the Receiver FIFO, the 5th data will be saved in the shift register. If there is 6th data, the 6th data will be saved in the shift register. But the RxCNT remains the value of 4. The RxCNT will be cleared when reset occurs or UARTEN=1. This register is read only.

**Baud Rate Generator**

To setup the speed of the serial data communication, the UART function contains its own dedicated baud rate generator. The baud rate is controlled by its own internal free running 16-bit timer, the period of which is determined by two factors. The first of these is the value placed in the BRDH/BRDL register and the second is the UART modulation control bits UMOD2~UMOD0. To prevent accumulated error of the receiver baud rate frequency, it is recommended to use two stop bits for resynchronization after each byte is received. If a baud rate BR is required with UART clock  $f_{IH}$ .

$$f_{IH}/BR = \text{Integer Part} + \text{Fractional Part}$$

The integer part is loaded into BRD (BRDH/BRDL). The fractional part is multiplied by 8 and rounded, then loaded into UMOD bit field as following:

$$BRD = \text{TRUNC}(f_H/BR)$$

$$UMOD = \text{ROUND}[\text{MOD}(f_H/BR) \times 8]$$

Therefore, the actual baud rate is as following:

$$\text{Baud rate} = f_H / [BRD + (UMOD/8)]$$

### Calculating the Baud Rate and Error Values

For a clock frequency of 4MHz, determine the BRDH/BRDL register value, the actual baud rate and the error value for a desired baud rate of 230400.

From the above formula, the  $BRD_n = \text{TRUNC}(f_H/BR) = \text{TRUNC}(17.36111) = 17$

The  $UMOD = \text{ROUND}[\text{MOD}(f_H/BR) \times 8] = \text{ROUND}(0.36111 \times 8) = \text{ROUND}(2.88888) = 3$

The actual Baud Rate  $= f_H / [BRD + (UMOD/8)] = 230215.83$

Therefore the error is equal to  $(230215.83 - 230400) / 230400 = -0.08\%$

### Modulation Control Example

To get the best-fitting bit sequence for UART modulation control bits UMOD2~UMOD0, the following algorithm can be used: Firstly, the fractional part of the theoretical division factor is multiplied by 8. Then the product will be rounded and UMOD2~UMOD0 bits will be filled with the rounded value. The UMOD2~UMOD0 will be added to internal accumulator for every UART bit time. Until a carry to bit 3, the corresponding UART bit time increases a UART clock cycle. The following is an example using the fraction 0.36111 previously calculated:  $UMOD[2:0] = \text{ROUND}(0.36111 \times 8) = 011b$ .

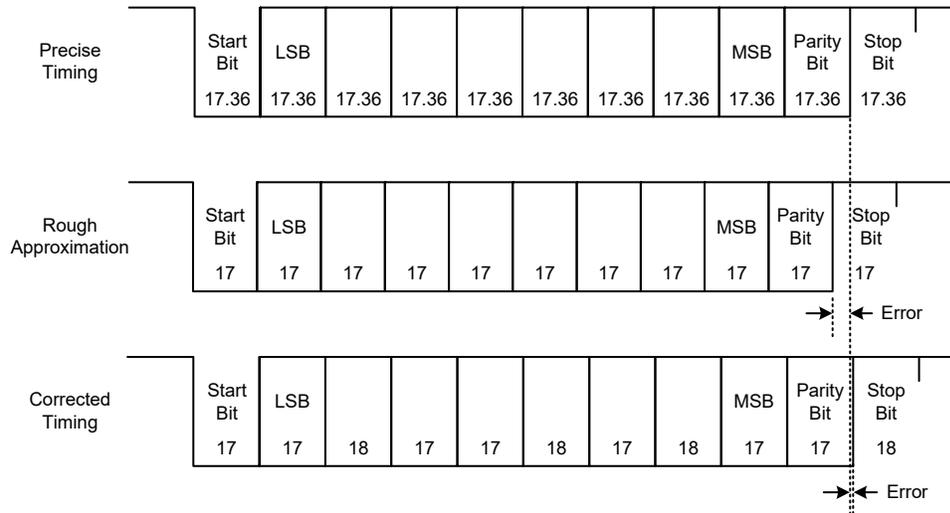
Fraction Addition	Carry to Bit 3	UART Bit Time Sequence	Extra UART Clock Cycle
0000b+0011b=0011b	No	Start bit	No
0011b+0011b=0110b	No	D0	No
0110b+0011b=1001b	Yes	D1	Yes
1001b+0011b=1100b	No	D2	No
1100b+0011b=1111b	No	D3	No
1111b+0011b=0010b	Yes	D4	Yes
0010b+0011b=0101b	No	D5	No
0101b+0011b=1000b	Yes	D6	Yes
1000b+0011b=1011b	No	D7	No
1011b+0011b=1110b	No	Parity bit	No
1110b+0011b=0001b	Yes	Stop bit	Yes

### Baud Rate Correction Example

The following figure presents an example using a baud rate of 230400 generated with UART clock  $f_H$ . The data format for the following figure is: eight data bits, parity enabled, no address bit, two stop bits.

The following figure shows three different frames:

- The upper frame is the correct one, with a bit-length of 17.36  $f_H$  cycles ( $4000000/230400 = 17.36$ ).
- The middle frame uses a rough estimate, with 17  $f_H$  cycles for the bit length.
- The lower frame shows a corrected frame using the best fit for the UART modulation control bits UMOD2~UMOD0.



### UART Setup and Control

For data transfer, the UART function utilizes a non-return-to-zero, more commonly known as NRZ, format. This is composed of one start bit, eight or nine data bits, and one or two stop bits. Parity is supported by the UART hardware, and can be setup to be even, odd, mark, space or no parity. For the most common data format, 8 data bits along with no parity and one stop bit, denoted as 8, N, 1, is used as the default setting, which is the setting at power-on. The number of data bits along with the parity are setup by programming the BNO, PRT1~PRT0 and PREN bits. The transmitter always uses two stop bits while the receiver uses one or two stop bits which is determined by the STOPS bit. The baud rate used to transmit and receive data is setup using the internal 16-bit baud rate generator, while the data is transmitted and received LSB first. Although the UART transmitter and receiver are functionally independent, they both use the same data format and baud rate. In all cases stop bits will be used for data transmission.

#### Enabling/Disabling the UART Interface

The basic on/off function of the internal UART function is controlled using the UARTEN bit in the UCR1 register. If the UARTEN, TXEN and RXEN bits are set, then these two UART pins will act as normal TX output pin and RX/TX input pin respectively. If no data is being transmitted on the TX pin, then it will default to a logic high value.

Clearing the UARTEN bit will disable the TX and RX/TX pins and allow these two pins to be used as normal I/O or other pin-shared functional pins by configuring the corresponding pin-shared control bits. When the UART function is disabled the buffer will be reset to an empty condition, at the same time discarding any remaining residual data. Disabling the UART will also reset the error and status flags with bits TXEN, RXEN, TXBRK, RXIF, OERR, FERR, PERR and NF being cleared while bits TIDLE, TXIF and RIDLE will be set. The remaining control bits in the UCR1, UCR2, UCR3, BRDH and BRDL registers will remain unaffected. If the UARTEN bit in the UCR1 register is cleared while the UART is active, then all pending transmissions and receptions will be immediately suspended and the UART will be reset to a condition as defined above. If the UART is then subsequently re-enabled, it will restart again in the same configuration.

#### Data, Parity and Stop Bit Selection

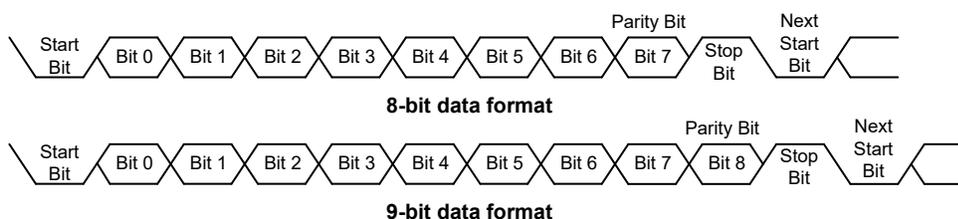
The format of the data to be transferred is composed of various factors such as data bit length, parity on/off, parity type, address bits and the number of stop bits. These factors are determined by the setup of various bits within the UCR1 and UCR2 registers. The BNO bit controls the number of data

bits which can be set to either 8 or 9, the PRT1~PRT0 bits controls the choice of odd, even, mark or space parity, the PREN bit controls the parity on/off function and the STOPS bit decides whether one or two stop bits are to be used for the receiver, while the transmitter always uses two stop bits. The following table shows various formats for data transmission. The address bit, which is the MSB of the data byte, identifies the frame as an address character or data if the address detect function is enabled. The number of stop bits, which can be either one or two, is independent of the data length and is only configurable for the receiver. The transmitter uses two stop bits.

Start Bit	Data Bits	Address Bit	Parity Bit	Stop Bit
<b>Example of 8-bit Data Formats</b>				
1	8	0	0	1 or 2
1	7	0	1	1 or 2
1	7	1	0	1 or 2
<b>Example of 9-bit Data Formats</b>				
1	9	0	0	1 or 2
1	8	0	1	1 or 2
1	8	1	0	1 or 2

**Transmitter Receiver Data Format**

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



## UART Transmitter

Data word lengths of either 8 or 9 bits can be selected by programming the BNO bit in the UCR1 register. When BNO bit is set, the word length will be set to 9 bits. In this case the 9th bit, which is the MSB, needs to be stored in the TX8 bit in the UCR1 register. At the transmitter core lies the Transmitter Shift Register, more commonly known as the TSR, whose data is obtained from the transmit data register, which is known as the TXR\_RXR register. The data to be transmitted is loaded into this TXR\_RXR register by the application program. The TSR register is not written to with new data until the stop bit from the previous transmission has been sent out. As soon as this stop bit has been transmitted, the TSR can then be loaded with new data from the TXR\_RXR register, if it is available. It should be noted that the TSR register, unlike many other registers, is not directly mapped into the Data Memory area and as such is not available to the application program for direct read/write operations. An actual transmission of data will normally be enabled when the TXEN bit is set, but the data will not be transmitted until the TXR\_RXR register has been loaded with data and the baud rate generator has defined a shift clock source. However, the transmission can also be initiated by first loading data into the TXR\_RXR register, after which the TXEN bit can be set. When a transmission of data begins, the TSR is normally empty, in which case a transfer to the TXR\_RXR register will result in an immediate transfer to the TSR. If during a transmission the TXEN bit is cleared, the transmission will immediately cease and the transmitter will be reset. The TX output pin can then be configured as the I/O or other pin-shared function by configuring the corresponding pin-shared control bits.

### Transmitting Data

When the UART is transmitting data, the data is shifted on the TX pin from the shift register, with the least significant bit first. In the transmit mode, the TXR\_RXR register forms a buffer between the internal bus and the transmitter shift register. It should be noted that if 9-bit data format has been selected, then the MSB will be taken from the TX8 bit in the UCR1 register. The steps to initiate a data transfer can be summarized as follows:

- Make the correct selection of the BNO, PRT1~PRT0 and PREN bits to define the required word length and parity type. Two stop bits are used for the transmitter.
- Setup the BRDH and BRDL registers and the UMOD2~UMOD0 bits to select the desired baud rate.
- Set the TXEN bit to ensure that the TX pin is used as a UART transmitter pin.
- Access the USR register and write the data that is to be transmitted into the TXR\_RXR register. Note that this step will clear the TXIF bit.

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

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

1. A USR register access
2. A TXR\_RXR register write execution

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

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

1. A USR register access
2. A TXR\_RXR register write execution

Note that both the TXIF and TIDLE bits are cleared by the same software sequence.

### Transmitting Break

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

## UART Receiver

The UART is capable of receiving word lengths of either 8 or 9 bits. If the BNO bit is set, the word length will be set to 9 bits with the MSB being stored in the RX8 bit of the UCR1 register. At the receiver core lies the Receive Serial Shift Register, commonly known as the RSR. The data which is received on the RX/TX external input pin is sent to the data recovery block. The data recovery block operating speed is 16 times that of the baud rate, while the main receive serial shifter operates at the baud rate. After the RX/TX pin is sampled for the stop bit, the received data in RSR is transferred to the receive data register, if the register is empty. The data which is received on the external RX/TX input pin is sampled three times by a majority detect circuit to determine the logic level that has been placed onto the RX/TX pin. It should be noted that the RSR register, unlike many other registers, is not directly mapped into the Data Memory area and as such is not available to the application program for direct read/write operations.

### Receiving Data

When the UART receiver is receiving data, the data is serially shifted in on the external RX/TX input pin, LSB first. In the read mode, the TXR\_RXR register forms a buffer between the internal bus and the receiver shift register. The TXR\_RXR register is a four byte deep FIFO data buffer, where four bytes can be held in the FIFO while a fifth byte can continue to be received. Note that the application program must ensure that the data is read from TXR\_RXR before the fifth byte has been completely shifted in, otherwise this fifth byte will be discarded and an overrun error OERR will be subsequently indicated. For continuous multi-byte data transmission, it is strongly recommended that the receiver uses two stop bits to avoid a receiving error caused by the accumulated error of the receiver baud rate frequency.

The steps to initiate a data transfer can be summarized as follows:

- Make the correct selection of BNO, PRT1~PRT0, PREN and STOPS bits to define the word length, parity type and number of stop bits.
- Setup the BRDH and BRDL registers and the UMOD2~UMOD0 bits select the desired baud rate.
- Set the RXEN bit to ensure that the RX/TX pin is used as a UART receiver pin.

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

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

- The RXIF bit in the USR register will be set when the TXR\_RXR register has data available, the number of the available data bytes can be checked by polling the RxCNT register content.
- When the contents of the shift register have been transferred to the TXR\_RXR register and reach Receiver FIFO trigger level, then if the RIE bit is set, an interrupt will be generated.
- If during reception, a frame error, noise error, parity error, or an overrun error has been detected, then the error flags can be set.

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

1. A USR register access
2. A TXR\_RXR register read execution

### Receiving Break

Any break character received by the UART will be managed as a framing error. The receiver will count and expect a certain number of bit times as specified by the values programmed into the BNO bit plus one or two stop bits. If the break is much longer than 13 bit times, the reception will be considered as complete after the number of bit times specified by BNO plus one or two stop bits. The RXIF bit is set, FERR is set, zeros are loaded into the receive data register, interrupts are

generated if appropriate and the RIDLE bit is set. A break is regarded as a character that contains only zeros with the FERR flag set. If a long break signal has been detected, the receiver will regard it as a data frame including a start bit, data bits and the invalid stop bit and the FERR flag will be set. The receiver must wait for a valid stop bit before looking for the next start bit. The receiver will not make the assumption that the break condition on the line is the next start bit. The break character will be loaded into the buffer and no further data will be received until one or two stop bits are received. It should be noted that the RIDLE read only flag will go high when the stop bits have not yet been received. The reception of a break character on the UART registers will result in the following:

- The framing error flag, FERR, will be set.
- The receive data register, TXR\_RXR, will be cleared.
- The OERR, NF, PERR, RIDLE or RXIF flags will possibly be set.

### Idle Status

When the receiver is reading data, which means it will be in between the detection of a start bit and the reading of a stop bit, the receiver status flag in the USR register, otherwise known as the RIDLE flag, will have a zero value. In between the reception of a stop bit and the detection of the next start bit, the RIDLE flag will have a high value, which indicates the receiver is in an idle condition.

### Receiver Interrupt

The read only receive interrupt flag RXIF in the USR register is set by an edge generated by the receiver. An interrupt is generated if RIE=1, when a word is transferred from the Receive Shift Register, RSR, to the Receive Data Register, TXR\_RXR. An overrun error can also generate an interrupt if RIE=1.

When a subroutine will be called with an execution time longer than the time for UART to receive five data bytes, if the UART received data could not be read in time during the subroutine execution, clear the RXEN bit to zero in advance to suspend data reception. If the UART interrupt could not be served in time to process the overrun error during the subroutine execution, ensure that both EMI and RXEN bits are disabled during this period, and then enable EMI and RXEN again after the subroutine execution has been completed to continue the UART data reception.

## Managing Receiver Errors

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

### Overrun Error – OERR

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

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

- The OERR flag in the USR register will be set.
- The TXR\_RXR contents will not be lost.
- The shift register will be overwritten.
- An interrupt will be generated if the RIE bit is set.

When the OERR flag is set to “1”, it is necessary to read five data bytes from the four-byte deep receiver FIFO and the shift register immediately to avoid unexpected errors, such as the UART is unable to receive data. If such an error occurs, clear the RXEN bit to “0” then set it to “1” again to continue data reception.

The OERR flag can be cleared by an access to the USR register followed by a read to the TXR\_RXR register.

#### **Noise Error – NF**

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

- The read only noise flag, NF, in the USR register will be set on the rising edge of the RXIF bit.
- Data will be transferred from the Shift register to the TXR\_RXR register.
- No interrupt will be generated. However this bit rises at the same time as the RXIF bit which itself generates an interrupt.

Note that the NF flag is reset by a USR register read operation followed by a TXR\_RXR register read operation.

#### **Framing Error – FERR**

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

#### **Parity Error – PERR**

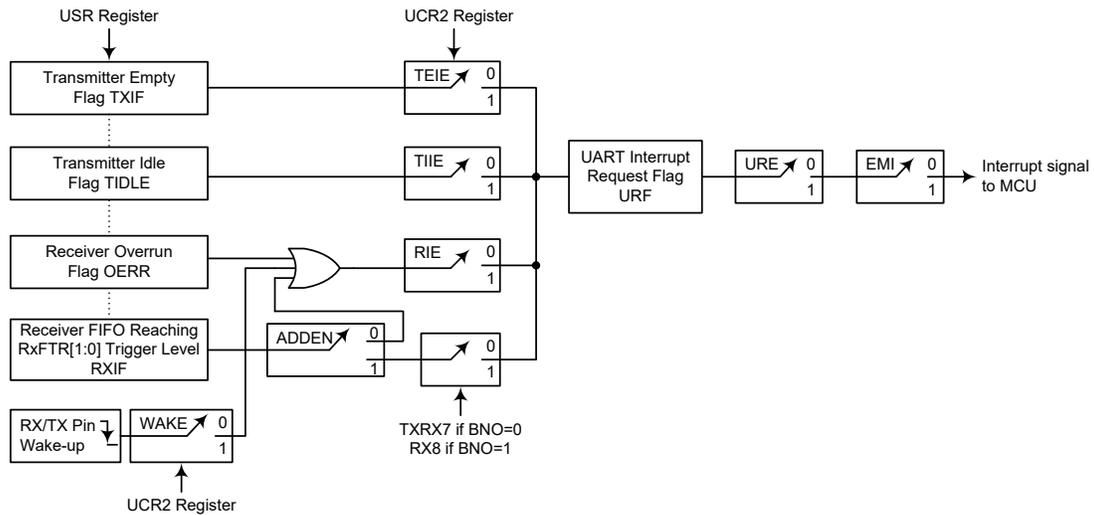
The read only parity error flag, PERR, in the USR register, is set if the parity of the received word is incorrect. This error flag is only applicable if the parity is enabled, PREN=1, and if the parity type, odd or even is selected. The read only PERR flag and the received data will be recorded in the USR and TXR\_RXR registers respectively. It is cleared on any reset, it should be noted that the flags, FERR and PERR, in the USR register should first be read by the application program before reading the data word.

### **UART Interrupt Structure**

Several individual UART conditions can generate a UART interrupt. When these conditions exist, a low pulse will be generated to get the attention of the microcontroller. These conditions are a transmitter data register empty, transmitter idle, receiver reaching FIFO trigger level, receiver overrun, address detect and an RX/TX pin wake-up. When any of these conditions are created, if the global interrupt enable bit and its corresponding interrupt control bit are enabled and the stack is not full, the program will jump to its corresponding interrupt vector where it can be serviced before returning to the main program. Four of these conditions have the corresponding USR register flags which will generate a UART interrupt if its associated interrupt enable control bit in the UCR2 register is set. The two transmitter interrupt conditions have their own corresponding enable control bits, while the two receiver interrupt conditions have a shared enable control bit. These enable bits can be used to mask out individual UART interrupt sources.

The address detect condition, which is also a UART interrupt source, does not have an associated flag, but will generate a UART interrupt when an address detect condition occurs if its function is enabled by setting the ADDEN bit in the UCR2 register. An RX/TX pin wake-up, which is also a UART interrupt source, does not have an associated flag, but will generate a UART interrupt if the UART clock ( $f_{H1}$ ) source is switched off and the WAKE and RIE bits in the UCR2 register are set when a falling edge on the RX/TX pin occurs.

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



**UART Interrupt Structure**

**Address Detect Mode**

Setting the Address Detect Mode bit, ADDEN, in the UCR2 register, enables this special mode. If this bit is enabled then an additional qualifier will be placed on the generation of a Receiver Data Available interrupt, which is requested by the RXIF flag. If the ADDEN bit is enabled, then when data is available, an interrupt will only be generated, if the highest received bit has a high value. Note that the URE and EMI interrupt enable bits must also be enabled for correct interrupt generation. This highest address bit is the 9th bit if BNO=1 or the 8th bit if BNO=0. If this bit is high, then the received word will be defined as an address rather than data. A Data Available interrupt will be generated every time the last bit of the received word is set. If the ADDEN bit is not enabled, then a Receiver Data Available interrupt will be generated each time the RXIF flag is set, irrespective of the data last bit status. The address detect mode and parity enable are mutually exclusive functions. Therefore if the address detect mode is enabled, then to ensure correct operation, the parity function should be disabled by resetting the parity enable bit PREN to zero.

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

**ADDEN Bit Function**

## UART Power Down and Wake-up

When the UART clock,  $f_{H}$ , is switched off, the UART will cease to function. If the MCU switches off the UART clock,  $f_{H}$ , and enters the power down mode while a transmission is still in progress, then the transmission will be paused until the UART clock source derived from the microcontroller is activated. In a similar way, if the MCU switches off the UART clock  $f_{H}$  and enters the IDLE or SLEEP mode by executing the “HALT” instruction while receiving data, then the reception of data will likewise be paused. When the MCU enters the IDLE or SLEEP mode, note that the USR, UCR1, UCR2, UCR3 transmit and receive registers, as well as the BRDH and BRDL registers will not be affected. It is recommended to make sure first that the UART data transmission or reception has been finished before the microcontroller enters the IDLE or SLEEP mode.

The UART function contains a receiver RX/TX pin wake-up function, which is enabled or disabled by the WAKE bit in the UCR2 register. If this bit, along with the UART enable bit, UARTEN, the receiver enable bit, RXEN and the receiver interrupt bit, RIE, are all set when the MCU enters the power down mode with the UART clock  $f_{H}$  being switched off, then a falling edge on the RX/TX pin will trigger an RX/TX pin wake-up UART interrupt. Note that as it takes certain system clock cycles after a wake-up, before normal microcontroller operation resumes, any data received during this time on the RX/TX pin will be ignored.

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

## Interrupts

Interrupts are an important part of any microcontroller system. When an external event or an internal function such as a Timer Module requires microcontroller attention, their corresponding interrupt will enforce a temporary suspension of the main program allowing the microcontroller to direct attention to their respective needs. The device contains several external interrupt and internal interrupt functions. The external interrupt is generated by the action of the external INT0 and INT1 pins, while the internal interrupts are generated by various internal functions such as the Timer Modules (TMs), Over Current Protection function(OCP), Time Base, LVD, EEPROM, Touch Key function, UART 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, as shown in the accompanying table. The interrupt registers fall into three categories. The first is the INTC0~INTC3 registers which setup the primary interrupts, the second is the MFI0~MFI2 registers which setup the Multi-function interrupts. Finally there is an INTEG register to setup the external interrupt trigger edge type.

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

Function	Enable Bit	Request Flag	Notes
Global	EMI	—	—
OCP	OCPE	OCPF	—
INTn Pin	INTnE	INTnF	n=0~1
Multi-function	MFnE	MFnF	n=0~2
Time Base	TBnE	TBnF	n=0~1
LVD	LVE	LVF	—
EEPROM	DEE	DEF	—
Touch Key Module	TKME	TKMF	—
A/D Converter	ADE	ADF	—
UART	URE	URF	—
CTMn	CTMnAE	CTMnAF	n=0~2
	CTMnPE	CTMnPF	
STM	STMAE	STMAF	—
	STMPE	STMPF	
PTM	PTMAE	PTMAF	—
	PTMPE	PTMPF	

**Interrupt Register Bit Naming Conventions**

Register Name	Bit							
	7	6	5	4	3	2	1	0
INTEG	—	—	—	—	INT1S1	INT1S0	INT0S1	INT0S0
INTC0	—	MF0F	INT0F	OCPF	MF0E	INT0E	OCPE	EMI
INTC1	TB0F	DEF	MF2F	MF1F	TB0E	DEE	MF2E	MF1E
INTC2	ADF	TKMF	INT1F	TB1F	ADE	TKME	INT1E	TB1E
INTC3	—	—	LVF	URF	—	—	LVE	URE
MF10	CTM1AF	CTM1PF	CTM0AF	CTM0PF	CTM1AE	CTM1PE	CTM0AE	CTM0PE
MF11	CTM2AF	CTM2PF	STMAF	STMPF	CTM2AE	CTM2PE	STMAE	STMPE
MF12	—	—	PTMAF	PTMPF	—	—	PTMAE	PTMPE

**Interrupt Register List**

• **INTEG Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	INT1S1	INT1S0	INT0S1	INT0S0
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0

Bit 7~4 Unimplemented, read as “0”

Bit 3~2 **INT1S1~INT1S0**: Defines INT1 interrupt active edge  
 00: Disabled  
 01: Rising Edge  
 10: Falling Edge  
 11: Dual Edges

Bit 1~0 **INT0S1~INT0S0**: Defines INT0 interrupt active edge  
 00: Disabled  
 01: Rising Edge  
 10: Falling Edge  
 11: Dual Edges

• **INTC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	MF0F	INT0F	OCPF	MF0E	INT0E	OCPE	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 **MF0F**: Multi-function interrupt 0 request flag  
 0: No request  
 1: Interrupt request
- Bit 5 **INT0F**: INT0 interrupt request flag  
 0: No request  
 1: Interrupt request
- Bit 4 **OCPF**: Over current protection interrupt request flag  
 0: No request  
 1: Interrupt request
- Bit 3 **MF0E**: Multi-function interrupt 0 control  
 0: Disable  
 1: Enable
- Bit 2 **INT0E**: INT0 interrupt control  
 0: Disable  
 1: Enable
- Bit 1 **OCPE**: Over current protection 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	TB0F	DEF	MF2F	MF1F	TB0E	DEE	MF2E	MF1E
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 **TB0F**: Time Base 0 interrupt request flag  
 0: No request  
 1: Interrupt request
- Bit 6 **DEF**: Data EEPROM interrupt request flag  
 0: No request  
 1: Interrupt request
- Bit 5 **MF2F**: Multi-function interrupt 2 request flag  
 0: No request  
 1: Interrupt request
- Bit 4 **MF1F**: Multi-function interrupt 1 request flag  
 0: No request  
 1: Interrupt request
- Bit 3 **TB0E**: Time Base 0 interrupt control  
 0: Disable  
 1: Enable
- Bit 2 **DEE**: Data EEPROM interrupt control  
 0: Disable  
 1: Enable

- Bit 1      **MF2E**: Multi-function interrupt 2 control  
0: Disable  
1: Enable
- Bit 0      **MF1E**: Multi-function interrupt 1 control  
0: Disable  
1: Enable

• **INTC2 Register**

Bit	7	6	5	4	3	2	1	0
Name	ADF	TKMF	INT1F	TB1F	ADE	TKME	INT1E	TB1E
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

- Bit 7      **ADF**: A/D converter interrupt request flag  
0: No request  
1: Interrupt request
- Bit 6      **TKMF**: Touch key module interrupt request flag  
0: No request  
1: Interrupt request
- Bit 5      **INT1F**: INT1 interrupt request flag  
0: No request  
1: Interrupt request
- Bit 4      **TB1F**: Time Base 1 interrupt request flag  
0: No request  
1: Interrupt request
- Bit 3      **ADE**: A/D converter interrupt control  
0: Disable  
1: Enable
- Bit 2      **TKME**: Touch key module interrupt control  
0: Disable  
1: Enable
- Bit 1      **INT1E**: INT1 interrupt control  
0: Disable  
1: Enable
- Bit 0      **TB1E**: Time Base 1 interrupt control  
0: Disable  
1: Enable

• **INTC3 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	LVF	URF	—	—	LVE	URE
R/W	—	—	R/W	R/W	—	—	R/W	R/W
POR	—	—	0	0	—	—	0	0

- Bit 7~6      Unimplemented, read as “0”
- Bit 5      **LVF**: LVD interrupt request flag  
0: No request  
1: Interrupt request
- Bit 4      **URF**: UART transfer interrupt request flag  
0: No request  
1: Interrupt request
- Bit3~2      Unimplemented, read as “0”

- Bit 1      **LVE**: LVD interrupt control  
             0: Disable  
             1: Enable
- Bit 0      **URE**: UART transfer interrupt Control  
             0: Disable  
             1: Enable

• **MF10 Register**

Bit	7	6	5	4	3	2	1	0
Name	CTM1AF	CTM1PF	CTM0AF	CTM0PF	CTM1AE	CTM1PE	CTM0AE	CTM0PE
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

- Bit 7      **CTM1AF**: CTM1 comparator A match interrupt request flag  
             0: No request  
             1: Interrupt request
- Bit 6      **CTM1PF**: CTM1 comparator P match interrupt request flag  
             0: No request  
             1: Interrupt request
- Bit 5      **CTM0AF**: CTM0 comparator A match interrupt request flag  
             0: No request  
             1: Interrupt request
- Bit 4      **CTM0PF**: CTM0 comparator P match interrupt request flag  
             0: No request  
             1: Interrupt request
- Bit 3      **CTM1AE**: CTM1 comparator A match interrupt control  
             0: Disable  
             1: Enable
- Bit 2      **CTM1PE**: CTM1 comparator P match interrupt control  
             0: Disable  
             1: Enable
- Bit 1      **CTM0AE**: CTM0 comparator A match interrupt control  
             0: Disable  
             1: Enable
- Bit 0      **CTM0PE**: CTM0 comparator P match interrupt control  
             0: Disable  
             1: Enable

• **MF11 Register**

Bit	7	6	5	4	3	2	1	0
Name	CTM2AF	CTM2PF	STMAF	STMPF	CTM2AE	CTM2PE	STMAE	STMPE
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      **CTM2AF**: CTM2 comparator A match interrupt request flag  
             0: No request  
             1: Interrupt request
- Bit 6      **CTM2PF**: CTM2 comparator P match interrupt request flag  
             0: No request  
             1: Interrupt request
- Bit 5      **STMAF**: STM comparator A match interrupt request flag  
             0: No request  
             1: Interrupt request

- Bit 4      **STMPF**: STM comparator P match interrupt request flag  
0: No request  
1: Interrupt request
- Bit 3      **CTM2AE**: CTM2 comparator A match interrupt control  
0: Disable  
1: Enable
- Bit 2      **CTM2PE**: CTM2 comparator P match interrupt control  
0: Disable  
1: Enable
- Bit 1      **STMAE**: STM comparator A match interrupt control  
0: Disable  
1: Enable
- Bit 0      **STMPE**: STM comparator P match interrupt control  
0: Disable  
1: Enable

• **MF12 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	PTMAF	PTMPF	—	—	PTMAE	PTMPE
R/W	—	—	R/W	R/W	—	—	R/W	R/W
POR	—	—	0	0	—	—	0	0

- Bit 7~6      Unimplemented, read as “0”
- Bit 5      **PTMAF**: PTM comparator A match interrupt request flag  
0: No request  
1: Interrupt request
- Bit 4      **PTMPF**: PTM comparator P match interrupt request flag  
0: No request  
1: Interrupt request
- Bit 3~2      Unimplemented, read as “0”
- Bit 1      **PTMAE**: PTM comparator A match interrupt control  
0: Disable  
1: Enable
- Bit 0      **PTMPE**: PTM comparator P match interrupt control  
0: Disable  
1: Enable

**Interrupt Operation**

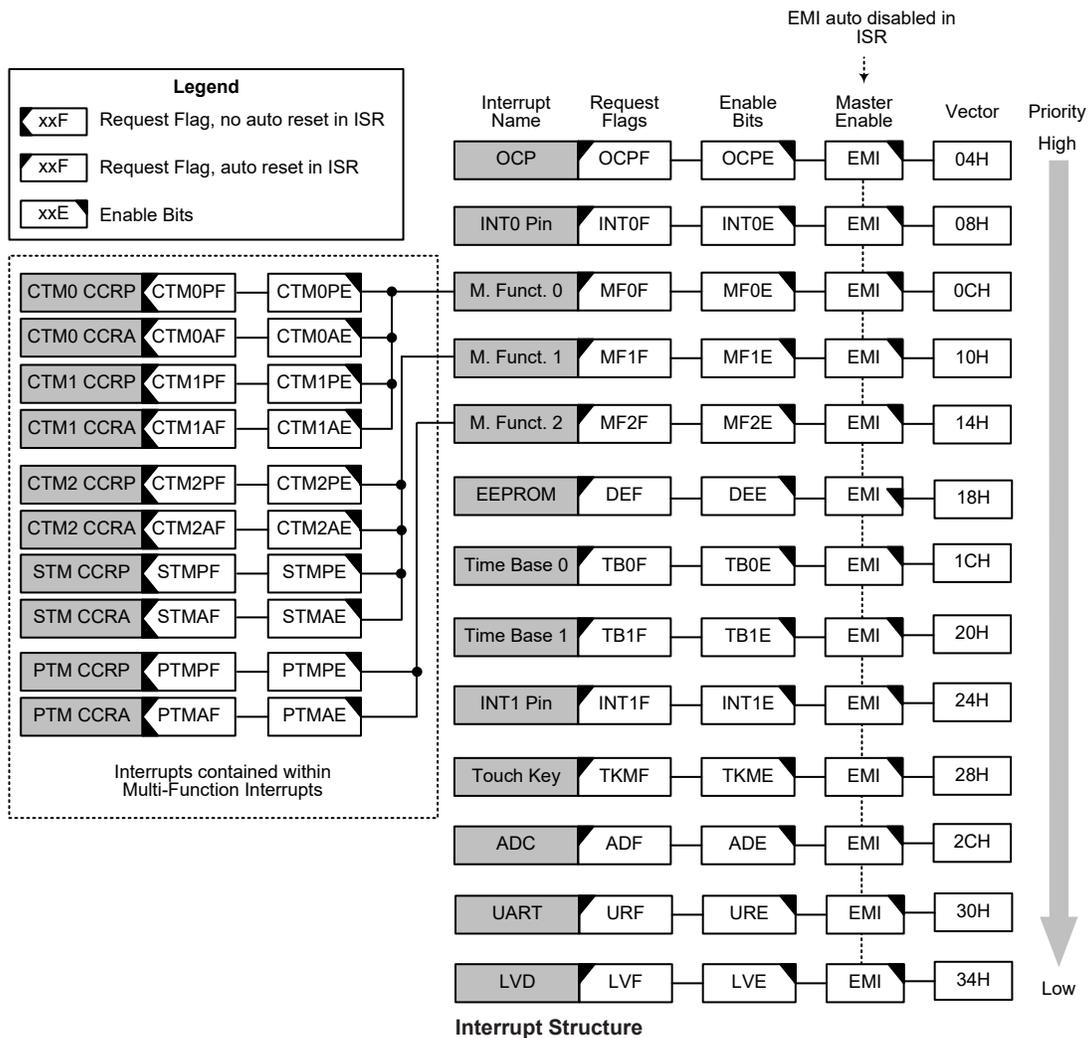
When the conditions for an interrupt event occur, such as a TM Comparator P or Comparator A or A/D conversion completion, etc, the relevant interrupt request flag will be set. Whether the request flag actually generates a program jump to the relevant interrupt vector is determined by the condition of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

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

stack and allows the microcontroller to continue with normal execution at the point where the interrupt occurred.

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

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



## External Interrupts

The external interrupts are controlled by signal transitions on the pins INT0~INT1. An external interrupt request will take place when the external interrupt request flags, INT0F~INT1F, are set, which will occur when a transition, whose type is chosen by the edge select bits, appears on the external interrupt pins. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and respective external interrupt enable bit, INT0E~INT1E, must first be set. Additionally the correct interrupt edge type must be selected using the INTEG register to enable the external interrupt function and to choose the trigger edge type. As the external interrupt pins are pin-shared with I/O pins, they can only be configured as external interrupt pins if their external interrupt enable bit in the corresponding interrupt register has been set. The pin must also be setup as an input by setting the corresponding bit in the port control register as well as the relevant pin-shared function selection bits. When the interrupt is enabled, the stack is not full and the correct transition type appears on the external interrupt pin, a subroutine call to the external interrupt vector, will take place. When the interrupt is serviced, the external interrupt request flags, INT0F~INT1F, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts. Note that any pull-high resistor selections on the external interrupt pins will remain valid even if the pin is used as an external interrupt input.

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

## OCP Interrupt

The OCP Interrupt is controlled by detecting the OCP input current. An OCP Interrupt request will take place when the OCP Interrupt request flag, OCPF, is set, which occurs when the Over Current Protection circuit detects an over current condition. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and OCP Interrupt enable bit, OCPE, must first be set. When the interrupt is enabled, the stack is not full and an over current is detected, a subroutine call to the OCP Interrupt vector, will take place. When the interrupt is serviced, the OCP Interrupt flag, OCPF, will be automatically cleared. 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.

## Multi-function Interrupts

Within the device there are three Multi-function interrupts. Unlike the other independent interrupts, these interrupts have no independent source, but rather are formed from other existing interrupt sources, namely the TM Interrupts.

A Multi-function interrupt request will take place when any of the Multi-function interrupt request flags, MFnF are set. The Multi-function interrupt flags will be set when any of their included functions generate an interrupt request flag. To allow the program to branch to its respective interrupt

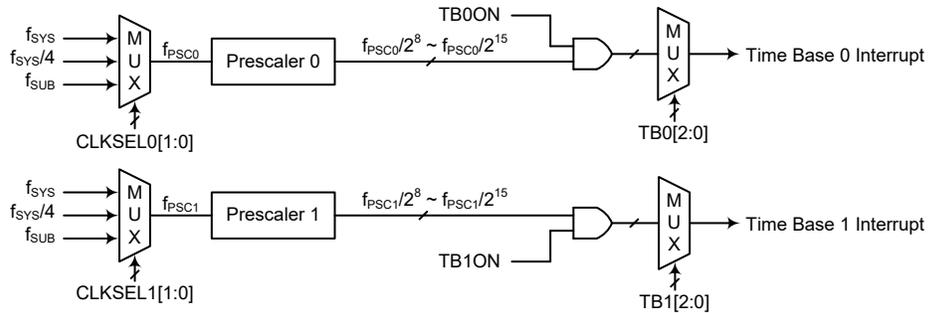
vector address, when the Multi-function interrupt is enabled and the stack is not full, and either one of the interrupts contained within each of Multi-function interrupt occurs, a subroutine call to one of the Multi-function interrupt vectors will take place. When the interrupt is serviced, the related Multi-Function request flag, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts.

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

### 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.



**Time Base Interrupts**

• **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:  $2^{15}/f_{PSC}$

### 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..

### LVD Interrupt

An LVD Interrupt request will take place when the LVD Interrupt request flag, LVF, is set, which occurs when the Low Voltage Detector function detects a low power supply voltage. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and Low Voltage Interrupt enable bit, LVE, and associated Multi-function interrupt enable bit, MF3E, must first be set. When the interrupt is enabled, the stack is not full and a low voltage condition occurs, a subroutine call to the LVD Interrupt vector, will take place. When the Low Voltage Interrupt is serviced, the LVF flag will be automatically cleared and the EMI bit will also be automatically cleared to disable other interrupts.

### Touch Key Module Interrupt

A Touch Key Module Interrupt request will take place when the Touch Key Module Interrupt request flag, TKMF, is set, which occurs when the touch key time slot counter overflows. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and the Touch Key Module Interrupt enable bit, TKME, must first be set. When the interrupt is enabled, the stack is not full and the touch key time slot counter overflows, a subroutine call to the respective Interrupt vector, will take place. When the interrupt is serviced, the Touch Key Module Interrupt flag, TKMF, will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts.

## UART Interrupt

Several individual UART conditions can generate a UART interrupt. When one of these conditions occurs, an interrupt signal will be generated to get the attention of the microcontroller. These conditions are a transmitter data register empty, transmitter idle, receiver reaching FIFO trigger level, receiver overrun, address detect and an RX/TX pin wake-up. To allow the program to branch to the respective interrupt vector addresses, the global interrupt enable bit, EMI, and the UART interrupt enable bit, URE, must first be set. When the interrupt is enabled, the stack is not full and any of these conditions are created, a subroutine call to the UART interrupt vector will take place. When the UART Interrupt is serviced, the UART interrupt request flag, URF, will be automatically cleared and the EMI bit will be automatically cleared to disable other interrupts. However, the USR register flags will only be cleared when certain actions are taken by the UART, the details of which are given in the UART section.

## TM Interrupts

The Compact, Standard and Periodic Type TMs each have two interrupts. All of the TM interrupts are contained within the Multi-function Interrupts. For the different Type TMs there are two interrupt request flags xTMnPF and xTMnAF and two enable bits xTMnPE and xTMnAE. 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 comparator A match situation happens.

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

## Interrupt Wake-up Function

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

## Programming Considerations

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

Where a certain interrupt is contained within a Multi-function interrupt, then when the interrupt service routine is executed, as only the Multi-function interrupt request flags, MF0F~MF2F, will be automatically cleared, the individual request flag for the function needs to be cleared by the application program.

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

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

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

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

## Low Voltage Detector – LVD

The device has a Low Voltage Detector function, also known as LVD. This enables the device to monitor the power supply voltage,  $V_{DD}$ , and provides a warning signal should it fall below a certain level. This function may be especially useful in battery applications where the supply voltage will gradually reduce as the battery ages, as it allows an early warning battery low signal to be generated. The Low Voltage Detector also has the capability of generating an interrupt signal.

### LVD Register

The Low Voltage Detector function is controlled using a single register with the name LVDC. Three bits in this register, VLVD2~VLVD0, are used to select one of eight fixed voltages below which a low voltage condition will be determined. A low voltage condition is indicated when the LVDO bit is set. If the LVDO bit is low, this indicates that the  $V_{DD}$  voltage is above the preset low voltage value. The LVDEN bit is used to control the overall on/off function of the low voltage detector. Setting the bit high will enable the low voltage detector. Clearing the bit to zero will switch off the internal low voltage detector circuits. As the low voltage detector will consume a certain amount of power, it may be desirable to switch off the circuit when not in use, an important consideration in power sensitive battery powered applications.

#### • LVDC Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	LVDO	LVDEN	VBGEN	VLVD2	VLVD1	VLVD0
R/W	—	—	R	R/W	R/W	R/W	R/W	R/W
POR	—	—	0	0	0	0	0	0

Bit 7~6 Unimplemented, read as “0”

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

Bit 4      **LV DEN**: Low Voltage Detector Control

0: Disable  
 1: Enable

Bit 3      **VB GEN**: Bandgap Buffer Control

0: Disable  
 1: Enable

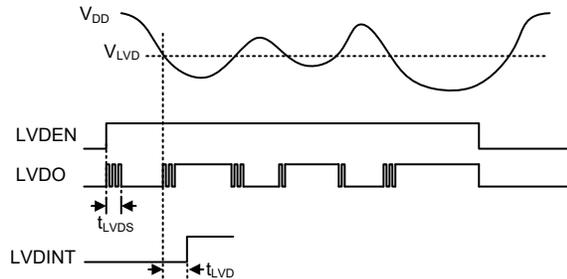
Note that the Bandgap circuit is enabled when the LVD or LVR function is enabled or when the VB GEN bit is set to 1.

Bit 2~0    **VLVD2~VLVD0**: Select LVD Voltage

000: 2.0V  
 001: 2.2V  
 010: 2.4V  
 011: 2.7V  
 100: 3.0V  
 101: 3.3V  
 110: 3.6V  
 111: 4.0V

### LVD Operation

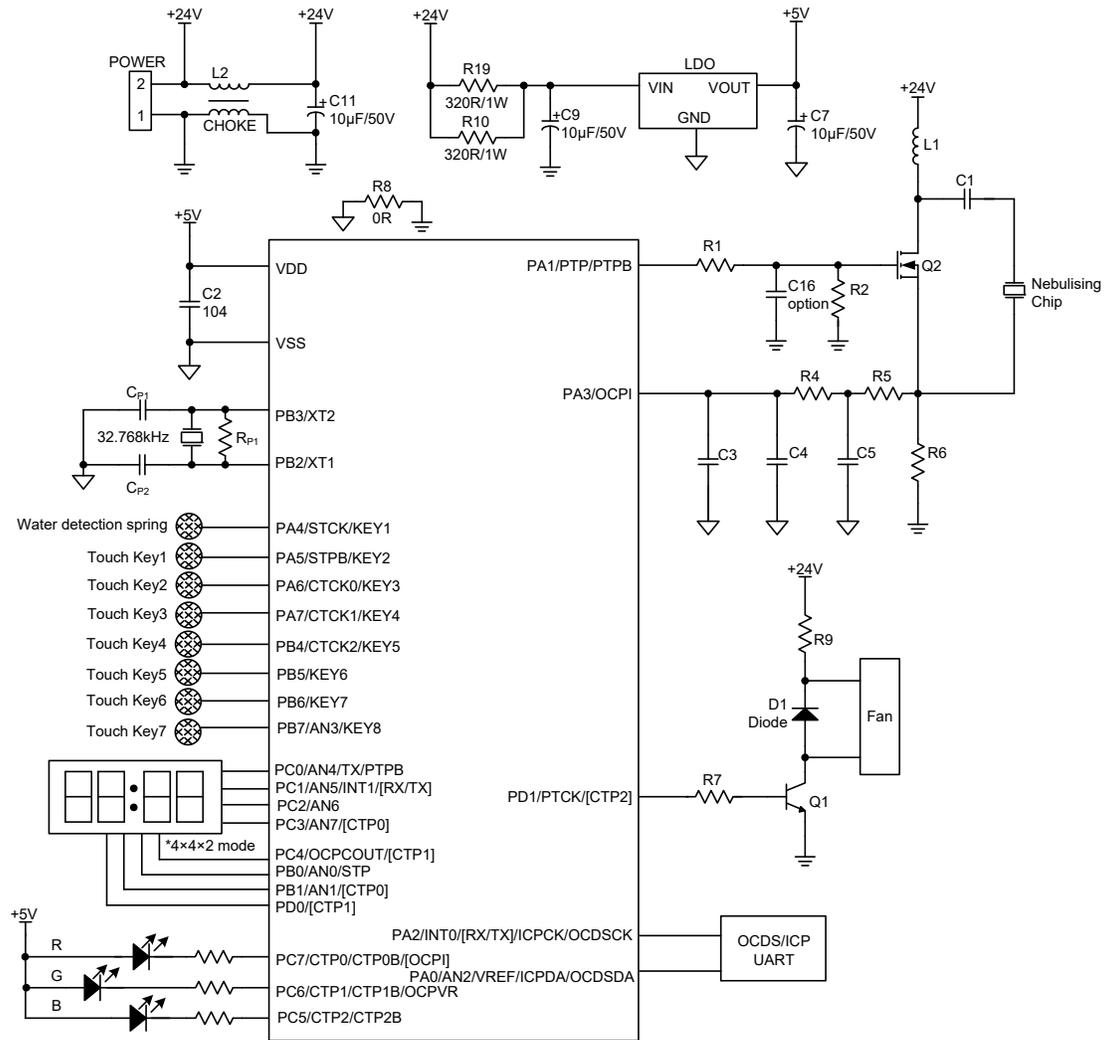
The Low Voltage Detector function operates by comparing the power supply voltage,  $V_{DD}$ , with a pre-specified voltage level stored in the LVDC register. This has a range of between 2.0V and 4.0V. When the power supply voltage,  $V_{DD}$ , falls below this pre-determined value, the LVDO bit will be set high indicating a low power supply voltage condition. The Low Voltage Detector function is supplied by a reference voltage which will be automatically enabled. After enabling the Low Voltage Detector, a time delay  $t_{LVDS}$  should be allowed for the circuitry to stabilise before reading the LVDO bit. Note also that as the  $V_{DD}$  voltage may rise and fall rather slowly, at the voltage nears that of  $V_{LVD}$ , there may be multiple bit LVDO transitions.



**LVD Operation**

The Low Voltage Detector also has an interrupt, providing an alternative means of low voltage detection, in addition to polling the LVDO bit. The interrupt will only be generated after a delay of  $t_{LVD}$  after the LVDO bit has been set high by a low voltage condition. In this case, the LVF interrupt request flag will be set, causing an interrupt to be generated if  $V_{DD}$  falls below the preset LVD voltage. This will cause the device to wake-up from the IDLE Mode, however if the Low Voltage Detector wake up function is not required then the LVF flag should be first set high before the device enters the IDLE Mode.

**Application Circuits**



## Instruction Set

### Introduction

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

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

### Instruction Timing

Most instructions are implemented within one instruction cycle. The exceptions to this are branch, call, or table read instructions where two instruction cycles are required. One instruction cycle is equal to 4 system clock cycles, therefore in the case of an 8MHz system oscillator, most instructions would be implemented within 0.5 $\mu$ 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	The contents of the specified Data Memory are read out and then written to the specified Data Memory again. If the contents of the specified Data Memory is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.
Operation	Skip if [m]=0
Affected flag(s)	None
<b>SZA [m]</b>	Skip if Data Memory is 0 with data movement to ACC
Description	The contents of the specified Data Memory are copied to the Accumulator. If the value is zero, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.
Operation	ACC ← [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

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

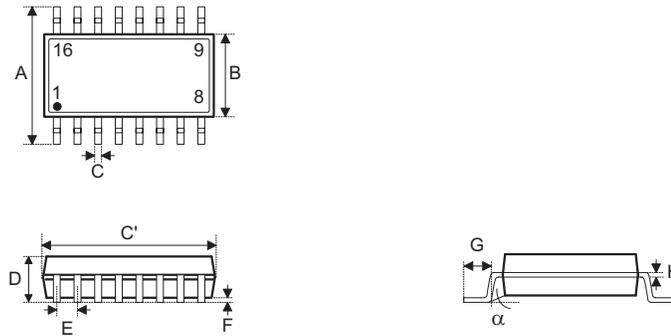
## Package Information

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

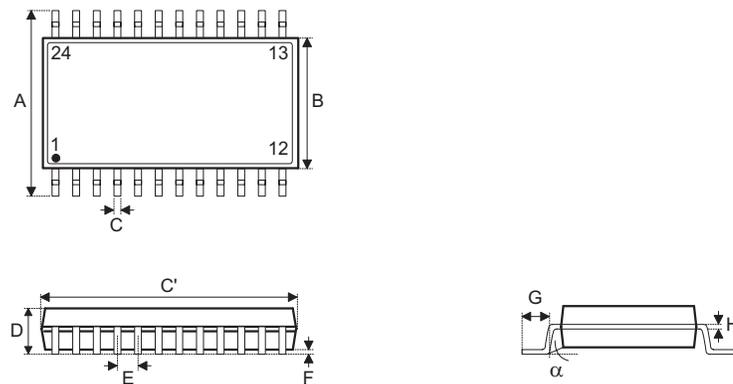
**16-pin NSOP (150mil) Outline Dimensions**



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

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	6.00 BSC		
B	3.90 BSC		
C	0.31	—	0.51
C'	9.90 BSC		
D	—	—	1.75
E	1.27 BSC		
F	0.10	—	0.25
G	0.40	—	1.27
H	0.10	—	0.25
$\alpha$	0°	—	8°

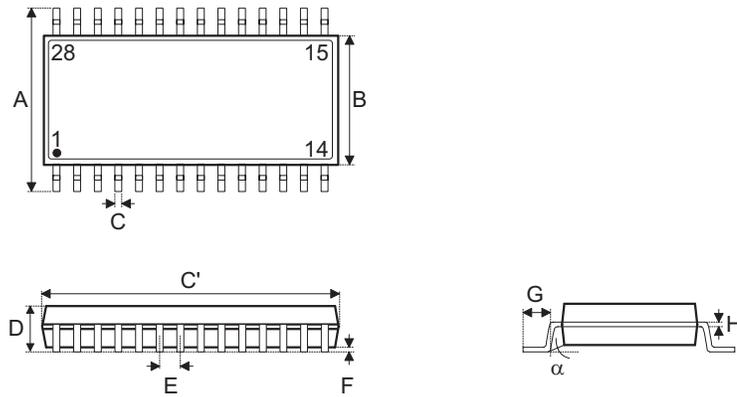
**24-pin SSOP (150mil) Outline Dimensions**



Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	0.236 BSC		
B	0.154 BSC		
C	0.008	—	0.012
C'	0.341 BSC		
D	—	—	0.069
E	0.025 BSC		
F	0.004	—	0.010
G	0.016	—	0.050
H	0.004	—	0.010
α	0°	—	8°

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	6.00 BSC		
B	3.90 BSC		
C	0.20	—	0.30
C'	8.66 BSC		
D	—	—	1.75
E	0.635 BSC		
F	0.10	—	0.25
G	0.41	—	1.27
H	0.10	—	0.25
α	0°	—	8°

**28-pin SSOP (150mil) Outline Dimensions**



Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	0.236 BSC		
B	0.154 BSC		
C	0.008	—	0.012
C'	0.390 BSC		
D	—	—	0.069
E	0.025 BSC		
F	0.004	—	0.010
G	0.016	—	0.050
H	0.004	—	0.010
$\alpha$	0°	—	8°

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	6.00 BSC		
B	3.90 BSC		
C	0.20	—	0.30
C'	9.90 BSC		
D	—	—	1.75
E	0.635 BSC		
F	0.10	—	0.25
G	0.41	—	1.27
H	0.10	—	0.25
$\alpha$	0°	—	8°

Copyright© 2025 by HOLTEK SEMICONDUCTOR INC. All Rights Reserved.

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