



**A/D Flash MCU**

**HT66F3126**

Revision: V1.00 Date: November 07, 2025

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

## Features

### CPU Features

- Operating Voltage
  - ♦  $f_{SYS}=4\text{MHz}$ : 1.8V~5.5V
  - ♦  $f_{SYS}=8\text{MHz}$ : 1.8V~5.5V
  - ♦  $f_{SYS}=12\text{MHz}$ : 2.7V~5.5V
  - ♦  $f_{SYS}=16\text{MHz}$ : 3.3V~5.5V
- Up to 0.25 $\mu\text{s}$  instruction cycle with 16MHz system clock at  $V_{DD}=5\text{V}$
- Power down and wake-up functions to reduce power consumption
- Oscillator Types
  - ♦ Internal High Speed 8/12/16MHz RC – HIRC
  - ♦ Internal Low Speed 32kHz RC – LIRC
- Fully integrated internal oscillators require no external components
- Multi-mode operation: FAST, SLOW, IDLE and SLEEP
- All instructions executed in 1~2 instruction cycles
- Table read instructions
- 61 powerful instructions
- 4-level subroutine nesting
- Bit manipulation instruction

### Peripheral Features

- Flash Program Memory: 1K $\times$ 14
- Data Memory: 128 $\times$ 8
- True EEPROM Memory: 64 $\times$ 8
- Watchdog Timer function
- 14 bidirectional I/O lines
- 2 external interrupt lines shared with I/O pins
- Programmable I/O port source current for LED applications
- Two Timer Modules for time measure, input capture, compare match output, PWM output function or single pulse output function
- Dual Time Base functions for generation of fixed time interrupt signals
- 7 external channel 12-bit resolution A/D converter with internal reference voltage  $V_{VR}$
- Low voltage reset function
- Package types: 8-pin SOP, 16-pin NSOP

## General Description

The device is a Flash Memory A/D 8-bit high performance RISC architecture microcontroller, designed for applications that interface directly to analog signals.

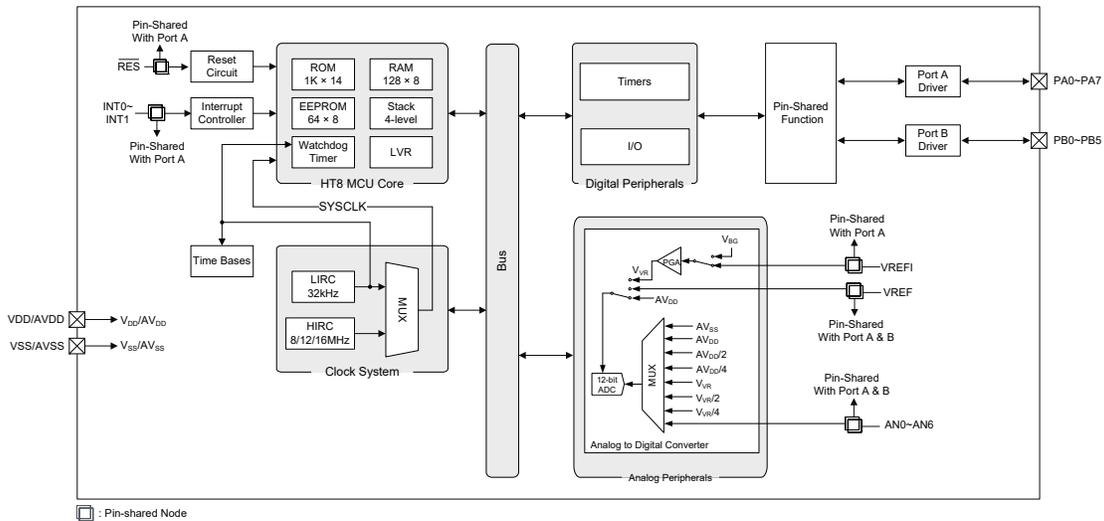
For memory features, the Flash Memory offers users the convenience of Flash Memory multi-programming features. Other memory includes an area of RAM Data Memory as well as an area of true EEPROM memory for storage of non-volatile data such as serial numbers, calibration data etc.

Analog features include a multi-channel 12-bit A/D converter. Two and extremely flexible Timer Modules provide timing, pulse generation and PWM generation functions. Protective features such as an internal Watchdog Timer and Low Voltage Reset coupled with excellent noise immunity and ESD protection ensure that reliable operation is maintained in hostile electrical environments.

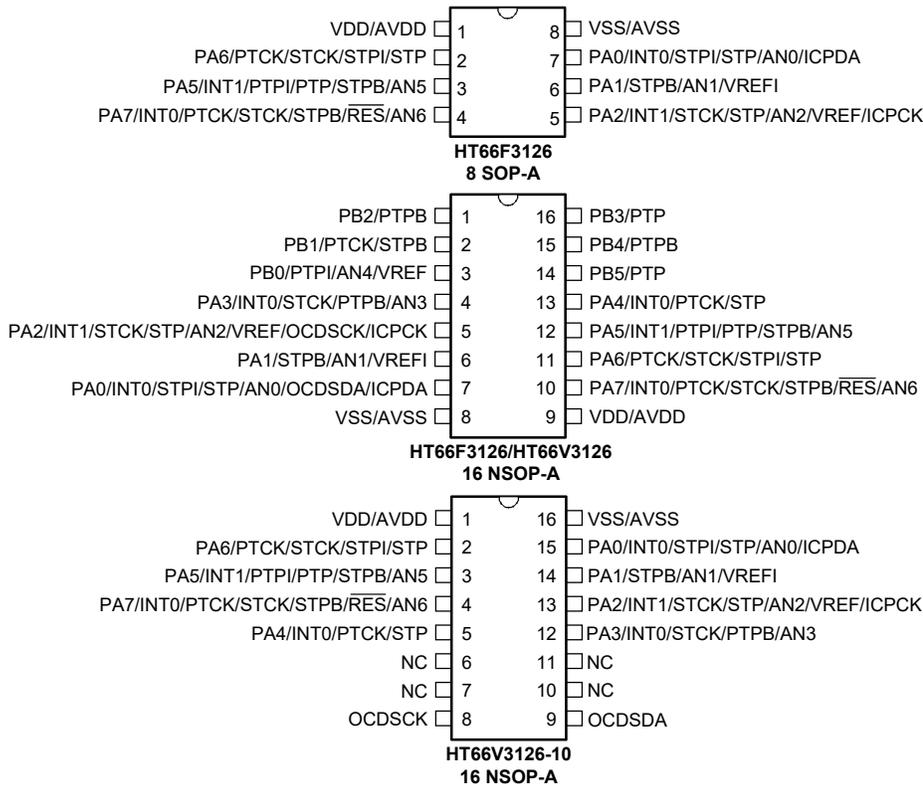
A full choice of internal high and low oscillator functions are provided including two fully integrated system oscillators which require no external components for their implementation. The ability to operate and switch dynamically between a range of operating modes using different clock sources gives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

The inclusion of flexible I/O programming features, Time Base functions along with many other features ensure that the device will find excellent use in applications such as electronic metering, environmental monitoring in addition to many others.

## Block Diagram



## Pin Assignment



- Note: 1. If the pin-shared pin functions have multiple outputs, the desired pin-shared function is determined by the corresponding software control bits.
2. The OCSDA and OCDSCK pins are the OCDS dedicated pins and are only available for the OCDS EV chips for the HT66F3126 device. The HT66V3126 device is the OCDS EV chip for the HT66F3126 device in 16-NSOP package. The HT66V3126-10 device is the OCDS EV chip for the HT66F3126 device in 8-SOP package.
3. For the less pin-count package type there will be unbounded pins which should be properly configured to avoid unwanted power consumption resulting from floating input conditions. Refer to the “Standby Current Considerations” and “Input/Output Ports” sections.

## Pin Descriptions

The function of each pin is listed in the following table, however the details behind how each pin is configured is contained in other sections of the datasheet. As the pin description table shows the situation for the package with the most pins, not all pins in the table will be available on smaller package sizes.

Pin Name	Function	OPT	I/T	O/T	Description
PA0/INT0/STPI/STP/AN0/ OCSDSA/ICPDA	PA0	PAPU PAWU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-high and wake-up
	INT0	PAS0 INTEG INTC0 IFS1	ST	—	External Interrupt
	STPI	PAS0 IFS0	ST	—	STM capture input
	STP	PAS0	—	CMOS	STM output
	AN0	PAS0	AN	—	A/D Converter analog input
	OCSDSA	—	ST	CMOS	OCDS data/address pin, for EV chip only
	ICPDA	—	ST	CMOS	ICP data/address pin
PA1/STPB/AN1/VREFI	PA1	PAPU PAWU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-high and wake-up
	STPB	PAS0	—	CMOS	STM inverted output
	AN1	PAS0	AN	—	A/D Converter analog input
	VREFI	PAS0	AN	—	A/D Converter PGA input
PA2/INT1/STCK/STP/AN2/ VREF/OCDSCK/ICPCK	PA2	PAPU PAWU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-high and wake-up
	INT1	PAS0 INTEG INTC0 IFS1	ST	—	External Interrupt
	STCK	PAS0 IFS0	ST	—	STM clock input
	STP	PAS0	—	CMOS	STM output
	AN2	PAS0	AN	—	A/D Converter analog input
	VREF	PAS0	AN	—	A/D Converter external reference voltage input
	OCDSCK	—	ST	—	OCDS clock pin, for EV chip only
	ICPCK	—	ST	—	ICP clock pin
PA3/INT0/STCK/PTPB/AN3	PA3	PAPU PAWU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-high and wake-up
	INT0	PAS0 INTEG INTC0 IFS1	ST	—	External Interrupt
	STCK	PAS0 IFS0	ST	—	STM clock input
	PTPB	PAS0	—	CMOS	PTM inverted output
	AN3	PAS0	AN	—	A/D Converter analog input

Pin Name	Function	OPT	I/T	O/T	Description
PA4/INT0/PTCK/STP	PA4	PAPU PAWU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-high and wake-up
	INT0	PAS1 INTEG INTC0 IFS1	ST	—	External interrupt
	PTCK	PAS1 IFS0	ST	—	PTM clock input or capture input
	STP	PAS1	—	CMOS	STM output
PA5/INT1/PTPI/PTP/STPB/ AN5	PA5	PAPU PAWU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-high and wake-up
	INT1	PAS1 INTEG INTC0 IFS1	ST	—	External interrupt
	PTPI	PAS1 IFS0	ST	—	PTM capture input
	PTP	PAS1	—	CMOS	PTM output
	STPB	PAS1	—	CMOS	STM inverted output
	AN5	PAS1	AN	—	A/D Converter analog input
PA6/PTCK/STCK/STPI/STP	PA6	PAPU PAWU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-high and wake-up
	PTCK	PAS1 IFS0	ST	—	PTM clock input or capture input
	STCK	PAS1 IFS0	ST	—	STM clock input
	STPI	PAS1 IFS0	ST	—	STM capture input
	STP	PAS1	—	CMOS	STM output
PA7/INT0/PTCK/STCK/ STPB/ $\overline{\text{RES}}$ /AN6	PA7	PAPU PAWU PAS1 RSTC	ST	CMOS	General purpose I/O. Register enabled pull-high and wake-up
	INT0	PAS1 INTEG INTC0 IFS1 RSTC	ST	—	External interrupt
	PTCK	PAS1 IFS0 RSTC	ST	—	PTM clock input or capture input
	STCK	PAS1 IFS0 RSTC	ST	—	STM clock input
	STPB	PAS1 RSTC	—	CMOS	STM inverted output
	$\overline{\text{RES}}$	RSTC	ST	—	Reset pin
	AN6	PAS1 RSTC	AN	—	A/D Converter analog input

Pin Name	Function	OPT	I/T	O/T	Description
PB0/PTPI/AN4/VREF	PB0	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-high
	PTPI	PBS0 IFS0	ST	—	PTM capture input
	AN4	PBS0	AN	—	A/D Converter analog input
	VREF	PBS0	AN	—	A/D Converter external reference voltage input
PB1/PTCK/STPB	PB1	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-high
	PTCK	PBS0 IFS0	ST	—	PTM clock input or capture input
	STPB	PBS0	—	CMOS	STM inverted output
PB2/PTPB	PB2	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-high
	PTPB	PBS0	—	CMOS	PTM inverted output
PB3/PTP	PB3	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-high
	PTP	PBS0	—	CMOS	PTM output
PB4/PTPB	PB4	PBPU PBS1	ST	CMOS	General purpose I/O. Register enabled pull-high
	PTPB	PBS1	—	CMOS	PTM inverted output
PB5/PTP	PB5	PBPU PBS1	ST	CMOS	General purpose I/O. Register enabled pull-high
	PTP	PBS1	—	CMOS	PTM output
VDD/AVDD	VDD	—	PWR	—	Digital positive power supply
	AVDD	—	PWR	—	Analog positive power supply
VSS/AVSS	VSS	—	PWR	—	Digital negative power supply
	AVSS	—	PWR	—	Analog negative power supply

Note: For the HT66V3126-10 device, the OCDSDA and OCDSCK pins are independent.

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

## Absolute Maximum Ratings

Supply Voltage .....	V <sub>SS</sub> -0.3V to 6.0V
Input Voltage .....	V <sub>SS</sub> -0.3V to V <sub>DD</sub> +0.3V
Storage Temperature.....	-60°C to 150°C
Operating Temperature.....	-40°C to 85°C
I <sub>OH</sub> Total .....	-80mA
I <sub>OL</sub> Total .....	80mA
Total Power Dissipation .....	500mW

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

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

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
$V_{DD}$	Operating Voltage – HIRC	CKS[2:0]=001B: $f_H/2$ , $f_H=8\text{MHz}$ , $f_{SYS}=4\text{MHz}$	1.8	—	5.5	V
		$f_{SYS}=8\text{MHz}$	1.8	—	5.5	V
		$f_{SYS}=12\text{MHz}$	2.7	—	5.5	V
		$f_{SYS}=16\text{MHz}$	3.3	—	5.5	V
	Operating Voltage – LIRC	$f_{SYS}=32\text{kHz}$	1.8	—	5.5	V

### Operating Current Characteristics

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

Symbol	Operating Mode	Test Conditions		Min.	Typ.	Max.	Unit
		$V_{DD}$	Conditions				
$I_{DD}$	SLOW Mode – LIRC	1.8V	$f_{SYS}=32\text{kHz}$ , LVR disable	—	11	15	$\mu\text{A}$
		3V		—	12	16	
		5V		—	13	19	
		1.8V	$f_{SYS}=32\text{kHz}$ , LVR enable	—	19	23	$\mu\text{A}$
		3V		—	20	24	
		5V		—	21	27	
	FAST Mode – HIRC	1.8V	$f_{SYS}=8\text{MHz}$	—	360	400	$\mu\text{A}$
		3V		—	510	560	
		5V		—	920	980	
		2.7V	$f_{SYS}=12\text{MHz}$	—	660	720	$\mu\text{A}$
		3V		—	740	830	
		5V		—	1340	1420	
		3.3V	$f_{SYS}=16\text{MHz}$	—	1020	1120	$\mu\text{A}$
		5V		—	1840	1960	

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

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

### Standby Current Characteristics

Ta=25°C, unless otherwise specified

Symbol	Standby Mode	Test Conditions		Min.	Typ.	Max.	Max. @85°C	Unit
		V <sub>DD</sub>	Conditions					
I <sub>STB</sub>	SLEEP Mode	1.8V	WDT off	—	0.19	0.50	2.40	μA
		3V		—	0.20	0.70	2.70	
		5V		—	0.25	0.90	3.60	
		1.8V	WDT on	—	1.0	1.4	3.1	μA
		3V		—	1.1	1.8	3.6	
		5V		—	2.5	3.1	5.6	
	IDLE0 Mode – LIRC	1.8V	f <sub>SUB</sub> on	—	1.3	1.7	3.3	μA
		3V		—	1.6	2.0	3.8	
		5V		—	3.0	3.5	6.0	
	IDLE1 Mode – HIRC	1.8V	f <sub>SUB</sub> on, f <sub>SYS</sub> =8MHz	—	220	240	260	μA
		3V		—	340	360	390	
		5V		—	660	700	740	
		2.7V	f <sub>SUB</sub> on, f <sub>SYS</sub> =12MHz	—	320	340	360	μA
		3V		—	490	520	550	
5V		—		970	1000	1040		
3.3V		f <sub>SUB</sub> on, f <sub>SYS</sub> =16MHz	—	680	720	780	μA	
5V	—		1320	1400	1480			

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

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

## A.C. Characteristics

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

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

During the program writing operation the writer will trim the HIRC oscillator at a user selected HIRC frequency and user selected voltage of 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	-3.5%	8	+3.5%	
			-40°C~85°C	-5%	8	+5%	
		1.8V~5.5V	25°C	-10%	8	+5%	
			-40°C~85°C	-15%	8	+10%	
		2.7V~5.5V	25°C	-2.5%	8	+2.5%	
			-40°C~85°C	-3%	8	+3%	
	12MHz Writer Trimmed HIRC Frequency	3V/5V	25°C	-1%	12	+1%	MHz
			-40°C~85°C	-2%	12	+2%	
		2.7V~5.5V	25°C	-2.5%	12	+2.5%	
			-40°C~85°C	-3%	12	+3%	
16MHz Writer Trimmed HIRC Frequency	5V	25°C	-1%	16	+1%	MHz	
		-40°C~85°C	-2%	16	+2%		
	3.3V~5.5V	25°C	-2.5%	16	+2.5%		
		-40°C~85°C	-3%	16	+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.

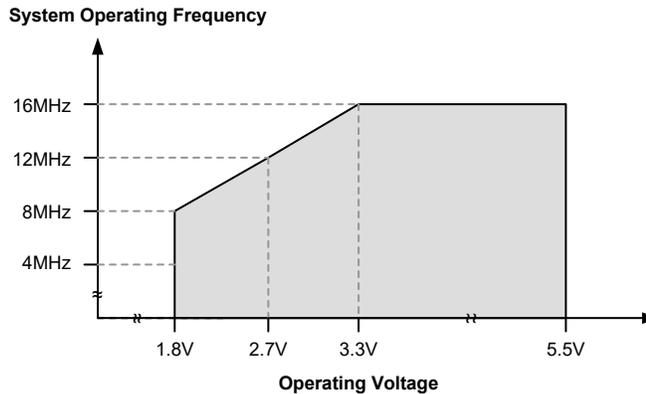
2. The row below the 3V/5V trim voltage row is provided to show the values for the full V<sub>DD</sub> range operating voltage. It is recommended that the trim voltage is fixed at 3V for application voltage ranges from 1.8V to 3.6V and fixed at 5V for application voltage ranges from 3.3V to 5.5V.

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

### 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	3V/5V	25°C	-2%	32	+2%	kHz
		2.2V~5.5V	-40°C~85°C	-15%	32	+15%	
		1.8V~5.5V	-40°C~85°C	-20%	32	+20%	
t <sub>START</sub>	LIRC Start Up Time	—	-40°C~85°C	—	—	100	µs

### Operating Frequency Characteristic Curves



### System Start Up Time Characteristics

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V <sub>DD</sub>	Conditions				
t <sub>SST</sub>	System Start-up Time (Wake-up from Conditions where f <sub>sys</sub> is off)	—	f <sub>sys</sub> =f <sub>H</sub> ~f <sub>H</sub> /64, f <sub>H</sub> =f <sub>HIRC</sub>	—	16	—	t <sub>HIRC</sub>
		—	f <sub>sys</sub> =f <sub>SUB</sub> =f <sub>LIRC</sub>	—	2	—	t <sub>LIRC</sub>
	System Start-up Time (Wake-up from Conditions where f <sub>sys</sub> is on)	—	f <sub>sys</sub> =f <sub>H</sub> ~f <sub>H</sub> /64, f <sub>H</sub> =f <sub>HIRC</sub>	—	2	—	t <sub>H</sub>
		—	f <sub>sys</sub> =f <sub>SUB</sub> =f <sub>LIRC</sub>	—	2	—	t <sub>SUB</sub>
	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>
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/WDTC/RSTC Software Reset)	—	—	—	—	—	
	System Reset Delay Time (Reset Source from WDT Overflow or RES Pin Reset)	—	—	14	16	18	
t <sub>SRESET</sub>	Minimum Software Reset Width to Reset	—	—	45	90	120	μs

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

## Input/Output Characteristics

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V <sub>DD</sub>	Conditions				
V <sub>IL</sub>	Input Low Voltage for I/O Ports	5V	—	0	—	1.5	V
		—	—	0	—	0.2V <sub>DD</sub>	
	Input Low Voltage for $\overline{\text{RES}}$ pin	—	V <sub>DD</sub> ≥2.7V	0	—	0.4V <sub>DD</sub>	V
		—	1.8V≤V <sub>DD</sub> <2.7V	0	—	0.3V <sub>DD</sub>	V
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>	
	Input High Voltage for RES pin	—	—	0.9V <sub>DD</sub>	—	V <sub>DD</sub>	V
I <sub>OL</sub>	Sink Current for I/O Ports	3V	V <sub>OL</sub> =0.1V <sub>DD</sub>	16	32	—	mA
		5V		32	65	—	
I <sub>OH</sub>	Source Current for I/O Ports	3V	V <sub>OH</sub> =0.9V <sub>DD</sub> , SLEDC[m+1:m]=00B (m=0, 2, 4 or 6)	-0.7	-1.5	—	mA
		5V	V <sub>OH</sub> =0.9V <sub>DD</sub> , SLEDC[m+1:m]=00B (m=0, 2, 4 or 6)	-1.5	-2.9	—	mA
		3V	V <sub>OH</sub> =0.9V <sub>DD</sub> , SLEDC[m+1:m]=01B (m=0, 2, 4 or 6)	-1.3	-2.5	—	mA
		5V	V <sub>OH</sub> =0.9V <sub>DD</sub> , SLEDC[m+1:m]=01B (m=0, 2, 4 or 6)	-2.5	-5.1	—	mA
		3V	V <sub>OH</sub> =0.9V <sub>DD</sub> , SLEDC[m+1:m]=10B (m=0, 2, 4 or 6)	-1.8	-3.6	—	mA
		5V	V <sub>OH</sub> =0.9V <sub>DD</sub> , SLEDC[m+1:m]=10B (m=0, 2, 4 or 6)	-3.6	-7.3	—	mA
		3V	V <sub>OH</sub> =0.9V <sub>DD</sub> , SLEDC[m+1:m]=11B (m=0, 2, 4 or 6)	-4	-8	—	mA
		5V	V <sub>OH</sub> =0.9V <sub>DD</sub> , SLEDC[m+1:m]=11B (m=0, 2, 4 or 6)	-8	-16	—	mA
R <sub>PH</sub>	Pull-high Resistance for I/O Ports <sup>(1)</sup>	3V	LVPU=0, P <sub>x</sub> PU=FFH (P <sub>x</sub> : PA or PB)	20	60	100	kΩ
		5V		10	30	50	
		3V	LVPU=1, P <sub>x</sub> PU=FFH (P <sub>x</sub> : PA or PB)	6.67	15	23	kΩ
		5V		3.5	7.5	12	
I <sub>LEAK</sub>	Input leakage current for I/O Ports	5V	V <sub>IN</sub> =V <sub>DD</sub> or V <sub>IN</sub> =V <sub>SS</sub>	—	—	±1	μA
t <sub>INT</sub>	External Interrupt Input Minimum Pulse Width	—	—	10	—	—	μs
t <sub>RES</sub>	External Reset Minimum Low Pulse Width	—	—	10	—	—	μs
t <sub>TPI</sub>	xTPI Input Pin Minimum Pulse Width	—	—	0.3	—	—	μs
t <sub>TCK</sub>	xTCK Input Pin Minimum Pulse Width	—	—	0.3	—	—	μs
f <sub>TMCLK</sub>	xTM Maximum Timer Clock Source Frequency	5V	—	—	—	1	f <sub>sys</sub>
t <sub>CPW</sub>	xTM Minimum Capture Pulse Width	—	—	t <sub>CPW</sub> <sup>(2)</sup>	—	—	μs

Note: 1. 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.

2. For STM

$$t_{CPW} = \max(2 \times t_{TMCLK}, t_{TPI})$$

Ex1: If f<sub>TMCLK</sub>=16MHz, t<sub>TPI</sub>=0.3μs, then t<sub>CPW</sub>=max(0.125μs, 0.3μs)=0.3μs

Ex2: If f<sub>TMCLK</sub>=8MHz, t<sub>TPI</sub>=0.3μs, then t<sub>CPW</sub>=max(0.25μs, 0.3μs)=0.3μs

For PTM

If PTCAPTS=0, then t<sub>CPW</sub>=max(2×t<sub>TMCLK</sub>, t<sub>TPI</sub>)

If PTCAPTS=1, then t<sub>CPW</sub>=max(2×t<sub>TMCLK</sub>, t<sub>TCK</sub>)

Ex1: If PTCAPTS=0, f<sub>TMCLK</sub>=16MHz, t<sub>TPI</sub>=0.3μs, then t<sub>CPW</sub>=max(0.125μs, 0.3μs)=0.3μs

Ex2: If PTCAPTS=1, f<sub>TMCLK</sub>=8MHz, t<sub>TCK</sub>=0.3μs, then t<sub>CPW</sub>=max(0.25μs, 0.3μs)=0.3μs

Where t<sub>TMCLK</sub>=1/f<sub>TMCLK</sub>

## Memory Characteristics

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

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V <sub>DD</sub>	Conditions				
<b>Flash Program Memory</b>							
V <sub>DD</sub>	V <sub>DD</sub> for Read / Write	—	—	1.8	—	5.5	V
t <sub>FER</sub>	Erase Time	—	—	2.273	2.500	2.778	ms
t <sub>FWR</sub>	Write Time	—	—	1.364	1.500	1.667	ms
E <sub>P</sub>	Cell Endurance	—	—	100K	—	—	E/W
t <sub>RETD</sub>	ROM Data Retention Time	—	Ta=25°C	—	10	—	Year
t <sub>ACTV</sub>	ROM Activation Time – Wake-up from IDLE/ SLEEP Mode	—	—	1	—	2	t <sub>LIRC</sub>
<b>Data EEPROM Memory</b>							
V <sub>DD</sub>	V <sub>DD</sub> for Read / Write	—	—	1.8	—	5.5	V
t <sub>EERD</sub>	EEPROM Read Time	—	—	—	—	4	t <sub>sys</sub>
t <sub>EEER</sub>	EEPROM Erase Time (Page Mode)	—	EWERTS=0	—	5.4	6.6	ms
		—	EWERTS=1	—	6.7	8.1	
t <sub>EEWR</sub>	EEPROM Write Time (Byte Mode)	—	EWERTS=0	—	5.4	6.6	ms
		—	EWERTS=1	—	6.7	8.1	
E <sub>P</sub>	Cell Endurance	—	—	100K	—	—	E/W
t <sub>RETD</sub>	Data Retention Time	—	Ta=25°C	—	10	—	Year
<b>RAM Data Memory</b>							
V <sub>DR</sub>	RAM Data Retention Voltage	—	—	1.0	—	—	V

Note: 1. “E/W” means Erase/Write times.

2. The ROM activation time t<sub>ACTV</sub> should be added when calculating the total system start-up time of a wake-up from the IDLE/SLEEP mode.

## 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 (1.7V)	-5%	1.7	+5%	V
I <sub>LVRBG</sub>	Operating Current	3V	LVR enable (1.7V)	—	—	10	μA
		5V	LVR enable (1.7V)	—	10	15	μA
t <sub>LVR</sub>	Minimum Low Voltage Width to Reset	—	TLVR[1:0]=00B	120	240	480	μs
			TLVR[1:0]=01B	0.5	1.0	2.0	
			TLVR[1:0]=10B	1	2	4	
			TLVR[1:0]=11B	2	4	8	
I <sub>LVR</sub>	Additional Current for LVR Enable	5V	—	—	—	14	μA

## Reference Voltage 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%	0.93	+20%	V
t <sub>BGS</sub>	V <sub>BG</sub> Turn-on Stable Time	—	No load	—	—	50	μs
I <sub>BG</sub>	Additional Current for Bandgap Reference Enable	—	VBGEN=1, LVR disable	—	—	2	μA

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

## A/D Converter Electrical Characteristics

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

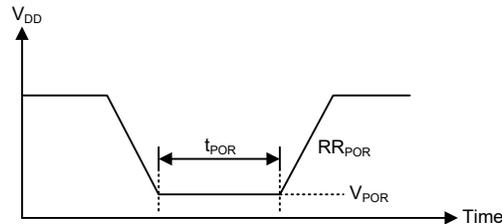
Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V <sub>DD</sub>	Conditions				
AV <sub>DD</sub>	Operating Voltage	—	—	1.8	—	5.5	V
V <sub>ADI</sub>	Input Voltage	—	—	0	—	V <sub>REF</sub>	V
V <sub>REF</sub>	Reference Voltage	—	—	1.8	—	AV <sub>DD</sub>	V
N <sub>R</sub>	Resolution	—	—	—	—	12	Bit
DNL	Differential Non-linearity	1.8V~5.5V	SAINS[3:0]=0000B, SAVRS[1:0]=01B, V <sub>REF</sub> =AV <sub>DD</sub> , t <sub>ADCK</sub> =2μs~10μs	-3	—	+3	LSB
		2.0V~5.5V	SAINS[3:0]=0000B, SAVRS[1:0]=01B, V <sub>REF</sub> =AV <sub>DD</sub> , t <sub>ADCK</sub> =0.5μs				
		2.7V~5.5V	SAINS[3:0]=0000B, SAVRS[1:0]=01B, V <sub>REF</sub> =AV <sub>DD</sub> , t <sub>ADCK</sub> =0.25μs	-3	—	+6	
		2.7V~5.5V	SAINS[3:0]=0000B, SAVRS[1:0]=01B, V <sub>REF</sub> =AV <sub>DD</sub> , t <sub>ADCK</sub> =0.167μs Ta=-10°C~85°C				
		2.7V~5.5V	SAINS[3:0]=0000B, SAVRS[1:0]=01B, V <sub>REF</sub> =AV <sub>DD</sub> , t <sub>ADCK</sub> =0.125μs Ta=-10°C~85°C				
INL	Integral Non-linearity	1.8V~5.5V	SAINS[3:0]=0000B, SAVRS[1:0]=01B, V <sub>REF</sub> =AV <sub>DD</sub> , t <sub>ADCK</sub> =2μs~10μs	-4	—	+4	LSB
		2.0V~5.5V	SAINS[3:0]=0000B, SAVRS[1:0]=01B, V <sub>REF</sub> =AV <sub>DD</sub> , t <sub>ADCK</sub> =0.5μs				
		2.7V~5.5V	SAINS[3:0]=0000B, SAVRS[1:0]=01B, V <sub>REF</sub> =AV <sub>DD</sub> , t <sub>ADCK</sub> =0.25μs	-4	—	+6	
		2.7V~5.5V	SAINS[3:0]=0000B, SAVRS[1:0]=01B, V <sub>REF</sub> =AV <sub>DD</sub> , t <sub>ADCK</sub> =0.167μs Ta=-10°C~85°C				
		2.7V~5.5V	SAINS[3:0]=0000B, SAVRS[1:0]=01B, V <sub>REF</sub> =AV <sub>DD</sub> , t <sub>ADCK</sub> =0.125μs Ta=-10°C~85°C				

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V <sub>DD</sub>	Conditions				
I <sub>ADC</sub>	Additional Current for A/D Converter Enable	1.8V	No load (t <sub>ADCK</sub> =2.0μs)	—	280	400	μA
		3V	No load (t <sub>ADCK</sub> =0.5μs)	—	450	600	
		5V	No load (t <sub>ADCK</sub> =0.5μs)	—	850	1000	
t <sub>ADCK</sub>	Clock Period	—	1.8V≤AV <sub>DD</sub> <2.0V	2.0	—	10.0	μs
			2.0V≤AV <sub>DD</sub> ≤5.5V	0.5	—	10.0	
			2.7V≤AV <sub>DD</sub> ≤5.5V	0.25	—	10.00	
			2.7V≤AV <sub>DD</sub> ≤5.5V Ta=-10°C~85°C	0.167	—	10.000	
			2.7V≤AV <sub>DD</sub> ≤5.5V Ta=-10°C~85°C	0.125	—	10.000	
t <sub>ON2ST</sub>	A/D Converter On-to-Start Time	—	—	4	—	—	μs
t <sub>ADS</sub>	Sampling Time	—	—	—	4	—	t <sub>ADCK</sub>
t <sub>ADC</sub>	Conversion Time (Includes A/D Sample and Hold Time)	—	—	—	16	—	t <sub>ADCK</sub>
I <sub>PGA</sub>	Additional Current for PGA Enable	2.2V	No load, PGAIS=1, PGAGS[1:0]=01	—	250	500	μA
		3V	No load, PGAIS=1, PGAGS[1:0]=01	—	300	600	μA
		5V	No load, PGAIS=1, PGAGS[1:0]=01	—	400	700	μA
V <sub>OR</sub>	PGA Maximum Output Voltage Range	2.2V	—	AV <sub>SS</sub> +0.1	—	AV <sub>DD</sub> -0.1	V
		3V	—	AV <sub>SS</sub> +0.1	—	AV <sub>DD</sub> -0.1	V
		5V	—	AV <sub>SS</sub> +0.1	—	AV <sub>DD</sub> -0.1	V
V <sub>VR</sub>	Fix Voltage Output of PGA	—	Ta=25°C, AV <sub>DD</sub> =2.2V~5.5V, V <sub>RI</sub> =V <sub>BG</sub> (PGAIS=1)	-2.5%	2	+2.5%	V
			Ta=25°C, AV <sub>DD</sub> =3.2V~5.5V, V <sub>RI</sub> =V <sub>BG</sub> (PGAIS=1)	-2.5%	3	+2.5%	
			Ta=25°C, AV <sub>DD</sub> =4.2V~5.5V, V <sub>RI</sub> =V <sub>BG</sub> (PGAIS=1)	-2.5%	4	+2.5%	
			Ta=-40°C~85°C, AV <sub>DD</sub> =2.2V~5.5V, V <sub>RI</sub> =V <sub>BG</sub> (PGAIS=1)	-10%	2	+10%	
			Ta=-40°C~85°C, AV <sub>DD</sub> =3.2V~5.5V, V <sub>RI</sub> =V <sub>BG</sub> (PGAIS=1)	-10%	3	+10%	
			Ta=-40°C~85°C, AV <sub>DD</sub> =4.2V~5.5V, V <sub>RI</sub> =V <sub>BG</sub> (PGAIS=1)	-10%	4	+10%	
V <sub>IR</sub>	PGA Input Voltage Range	3V	Gain=1, PGAIS=0, Relative gain, Gain error<±5%	AV <sub>SS</sub> +0.1	—	AV <sub>DD</sub> -1.4	V
		5V		AV <sub>SS</sub> +0.1	—	AV <sub>DD</sub> -1.4	V
V <sub>OS_PGA</sub>	PGA Input Offset Voltage	3V	—	-15	—	+15	mV
		5V	—	-15	—	+15	

## Power-on Reset Characteristics

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

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



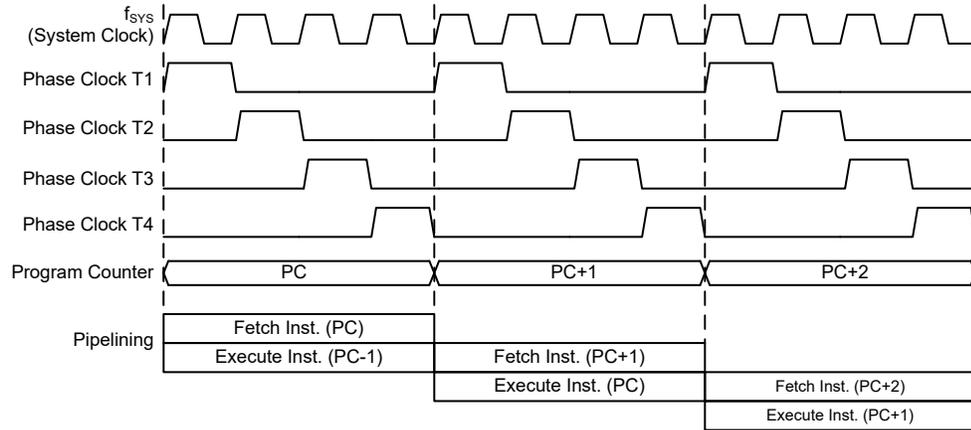
## System Architecture

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

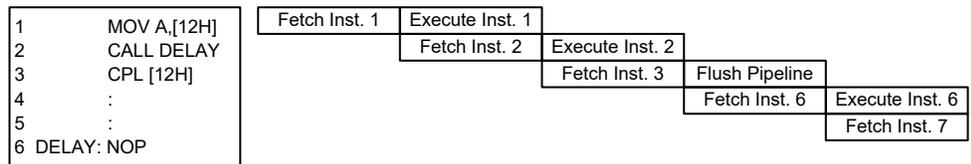
### Clocking and Pipelining

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

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



**System Clocking and Pipelining**



**Instruction Fetching**

### Program Counter

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

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

Program Counter	
High Byte	Low Byte (PCL)
PC9~PC8	PCL7~PCL0

**Program Counter**

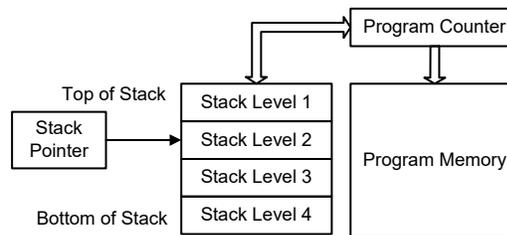
The lower byte of the Program Counter, known as the Program Counter Low register or PCL, is available for program control and is a readable and writable register. By transferring data directly into this register, a short program jump can be executed directly; however, as only this low byte is available 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 4 levels and is neither part of the data nor part of the program space, and is neither readable nor writeable. The activated level is indexed by the Stack Pointer, and is neither readable nor writeable. At a subroutine call or interrupt acknowledge signal, the contents of the Program Counter are pushed onto the stack. At the end of a subroutine or an interrupt routine, signaled by a return instruction, RET or RETI, the Program Counter is restored to its previous value from the stack. After a device reset, the Stack Pointer will point to the top of the stack.

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

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



## Arithmetic and Logic Unit – ALU

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

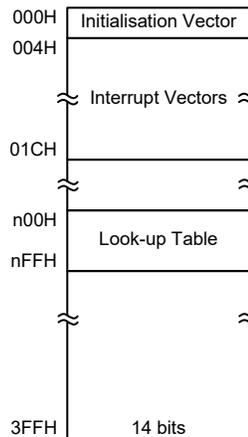
- Arithmetic operations:  
ADD, ADDM, ADC, ADCM, SUB, SUBM, SBC, SBCM, DAA
- Logic operations:  
AND, OR, XOR, ANDM, ORM, XORM, CPL, CPLA
- Rotation:  
RRA, RR, RRCA, RRC, RLA, RL, RLCA, RLC
- Increment and Decrement:  
INCA, INC, DECA, DEC
- Branch decision:  
JMP, SZ, SZA, SNZ, SIZ, SDZ, SIZA, SDZA,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

## Flash Program Memory

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

### Structure

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



**Program Memory Structure**

### Special Vectors

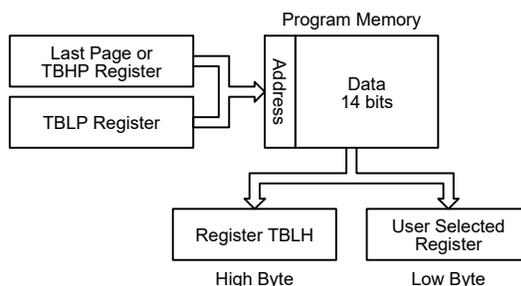
Within the Program Memory, certain locations are reserved for the reset and interrupts. The location 000H is reserved for use by the device reset for program initialisation. After a device reset is initiated, the program will jump to this location and begin execution.

### Look-up Table

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

After setting up the table pointer, the table data can be retrieved from the Program Memory using the “TABRD [m]” or “TABRDL [m]” instruction. When the instruction is executed, the lower order table byte from the Program Memory will be transferred to the user defined Data Memory register [m] as specified in the instruction. The higher order table data byte from the Program Memory will be transferred to the TBLH special register.

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



### Table Program Example

The accompanying example shows how the table pointer and table data are defined and retrieved from the device. This example uses raw table data located in the Program Memory which is stored there using the ORG statement. The value at this ORG statement is “300H” which refers to the start address of the last page within the 1K Program Memory. 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 “306H” or 6 locations after the start of the last page. Note that the value for the table pointer is referenced to the specific address pointed by TBHP and TBLP 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 table pointer - note that this address is referenced
mov tblp,a       ; to the last page or the page that tbhp pointed
mov a,03H         ; initialise high table pointer
mov tbhp,a
:
:
tabrd tempreg1    ; transfers value in table referenced by table pointer data at
                  ; program memory address "306H" transferred to tempreg1 and TBLH
dec tblp          ; reduce value of table pointer by one
tabrd tempreg2    ; transfers value in table referenced by table pointer data at
                  ; program memory address "305H" transferred to tempreg2 and TBLH
                  ; in this example the data "1AH" is transferred to tempreg1 and
                  ; data "0FH" to tempreg2, the value "00H" will be
                  ; transferred to TBLH
:
:
org 300H          ; sets initial address of program memory
dc 00Ah,00Bh,00Ch,00Dh,00Eh,00Fh,01Ah,01Bh
  
```

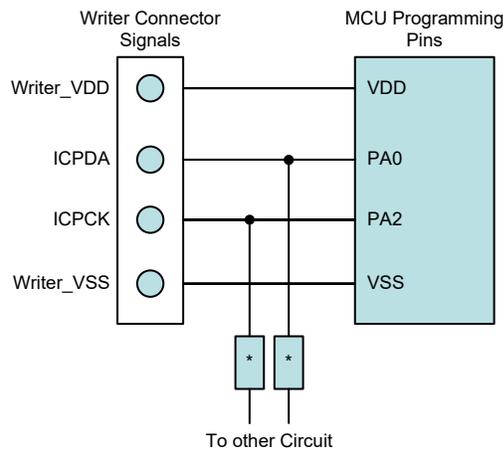
### In Circuit Programming – ICP

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

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

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

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



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

## On-Chip Debug Support – OCDS

There are two EV chips named HT66V3126 and HT66V3126-10 which are used to emulate the HT66F3126 device. Each 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 OCDS function to emulate the real chip device behavior by connecting the OCSDSA and OCDSCK pins to the Holtek HT-IDE development tools. The OCSDSA 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 OCSDSA 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
OCSDSA	OCSDSA	On-Chip Debug Support Data/Address input/output
OCDSCK	OCDSCK	On-Chip Debug Support Clock input
VDD	VDD	Power Supply
VSS	VSS	Ground

## Data Memory

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

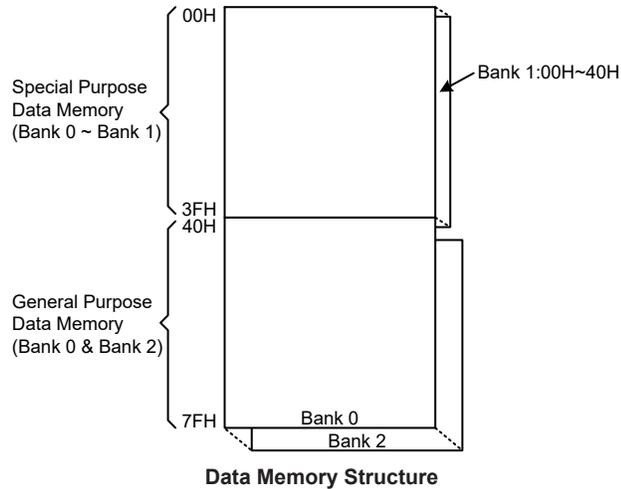
### Structure

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

The Data Memory is subdivided into several banks, all of which are implemented in 8-bit wide Memory. 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 3FH in the Bank 0 and from 00H to 40H in the Bank 1 while the General Purpose Data Memory address range is from 40H to 7FH in the Bank 0 and Bank 2.

Special Purpose Data Memory	General Purpose Data Memory	
Located Banks	Capacity	Bank: Address
Bank 0: 00H~3FH Bank 1: 00H~40H	128×8	0: 40H~7FH 2: 40H~7FH

**Data Memory Summary**



### 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	General Purpose Data Memory	EEC
01H	MP0		41H		
02H	IAR1		42H		
03H	MP1		43H		
04H	BP		44H		
05H	ACC		45H		
06H	PCL		46H		
07H	TBLP		47H		
08H	TBLH		48H		
09H	TBHP		49H		
0AH	STATUS		4AH		
0BH			4BH		
0CH			4CH		
0DH			4DH		
0EH			4EH		
0FH	RSTFC		4FH		
10H			50H		
11H			51H		
12H			52H		
13H	MF11		53H		
14H	PA		54H		
15H	PAC		55H		
16H	PAPU		56H		
17H	PAWU		57H		
18H	LVPUC		58H		
19H	IFS0		59H		
1AH	IFS1		5AH		
1BH	PAS0		5BH		
1CH	PAS1		5CH		
1DH	SLEDC		5DH		
1EH	INTC0		5EH		
1FH	INTC1	5FH			
20H	MF10	60H			
21H	INTEG	61H			
22H	PB	62H			
23H	PBC	63H			
24H	PBPU	64H			
25H	PBS0	65H			
26H	PBS1	66H			
27H	SCC	67H			
28H	HIRCC	68H			
29H	WDTC	69H			
2AH	LVRC	6AH			
2BH	TLVRC	6BH			
2CH	VBGC	6CH			
2DH	RSTC	6DH			
2EH	PSC0R	6EH			
2FH	TB0C	6FH			
30H	PSC1R	70H			
31H	TB1C	71H			
32H	STMC0	PTMC0	72H		
33H	STMC1	PTMC1	73H		
34H	STMDL	PTMDL	74H		
35H	STMDH	PTMDH	75H		
36H	STMAL	PTMAL	76H		
37H	STMAH	PTMAH	77H		
38H	SADC0	PTMRPL	78H		
39H	SADC1	PTMRPH	79H		
3AH	SADC2		7AH		
3BH	SADOL		7BH		
3CH	SADOH		7CH		
3DH	ORMC		7DH		
3EH	EEA		7EH		
3FH	EED		7FH		

□ : Unused, read as 00H

**Special Purpose Data Memory Structure**

## Special Function Register Description

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

### Indirect Addressing Registers – IAR0, IAR1

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

### Bank Pointer – BP

For this device, the Data Memory is divided into three banks, Bank 0, Bank 1 and Bank 2. Selecting the required Data Memory area is achieved using the DMBP1~DMBP0 bits in the BP register.

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

#### • BP Register

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

Bit 7~2 Unimplemented, read as “0”

Bit 1~0 **DMBP1~DMBP0**: Data Memory Bank selection

00: Bank 0

01: Bank 1

10: Bank 2

11: No Bank

### Accumulator – ACC

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

### Program Counter Low Byte Register – PCL

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

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

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

### Option Memory Mapping Register – ORMC

The ORMC register is used to enable Option Memory Mapping function. The Option Memory capacity is 64 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~3FH will be mapped to the Program Memory last page addresses C0H~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.

### 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 content of the status register is important and if the subroutine can corrupt the status register, precautions must be taken to correctly save it.

#### • STATUS Register

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

“x”: unknown

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

## EEPROM Data Memory

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 64×8 bits for this device. Unlike the Program Memory and RAM Data Memory, the EEPROM Data Memory is not directly mapped into memory space and is therefore not directly addressable in the same way as the other types of memory. Read and Write operations to the EEPROM are carried out in the single byte operations using an address and a data register in Bank 0 and a single control register in Bank 1.

### EEPROM Registers

Three registers control the overall operation of the internal EEPROM Data Memory. These are the address register, EEA, the data register, EED and a single control register, EEC. As both the EEA and EED registers are located in Bank 0, they can be directly accessed in the same way as any other Special Function Register. The EEC register, however, being located in Bank 1, can 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	—	—	EEA5	EEA4	EEA3	EEA2	EEA1	EEA0
EED	D7	D6	D5	D4	D3	D2	D1	D0
EEC	EWERTS	EREN	ER	—	WREN	WR	RDEN	RD

EEPROM Register List

#### • EEA Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	EEA5	EEA4	EEA3	EEA2	EEA1	EEA0
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 5~0 **EEA5~EEA0**: Data EEPROM address bit 5 ~ bit 0

#### • EED Register

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

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

**• EEC Register**

Bit	7	6	5	4	3	2	1	0
Name	EWERTS	EREN	ER	—	WREN	WR	RDEN	RD
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     **EWERTS**: Data EEPROM Erase time and Write time select  
0: Erase time is 5.4ms ( $t_{EEER}$ ) / Write time is 5.4ms ( $t_{EEWR}$ )  
1: Erase time is 6.7ms ( $t_{EEER}$ ) / Write time is 6.7ms ( $t_{EEWR}$ )
- Bit 6     **EREN**: Data EEPROM erase enable  
0: Disable  
1: Enable  
This bit is used to enable Data EEPROM erase function and must be set high before Data EEPROM erase operations are carried out. This bit will be automatically reset to zero by the hardware after the erase cycle has finished. Clearing this bit to zero will inhibit data EEPROM erase operations.
- Bit 5     **ER**: Data EEPROM erase control  
0: Erase cycle has finished  
1: Activate an erase cycle  
This is the Data EEPROM erase control bit. When this bit is set high by the application program, an erase cycle will be activated. This bit will be automatically reset to zero by hardware after the erase cycle has finished. Setting this bit high will have no effect if the EREN bit has not first been set high.
- Bit 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. Note that the WREN bit will automatically be cleared to zero after the write operation is finished.
- Bit 2     **WR**: Data EEPROM write control  
0: Write cycle has finished  
1: Activate a write cycle  
This is the Data EEPROM write control bit. When this bit is set high by the application program, a write cycle will be activated. This bit will be automatically reset to zero by 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**: Data EEPROM read control  
0: Read cycle has finished  
1: Activate a read cycle  
This is the Data EEPROM read control bit. When this bit is set high by the application program, a read cycle will be activated. This bit will be automatically reset to zero by 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 EREN, ER, WREN, WR, RDEN and RD cannot be set to “1” at the same time in one instruction.  
2. Ensure that the  $f_{SUB}$  clock is stable before executing the erase or write operation.  
3. Ensure that the erase or write operation is totally complete before changing contents of the EEPROM related registers.

### **Byte Read Operation from the EEPROM**

For a byte read operation the desired EEPROM address should first be placed in the EEA register, as well as the read enable bit, RDEN, in the EEC register should be set high to enable the read function. Then setting the RD bit high will initiate the EEPROM byte read operation. Note that setting only the RD bit high will not initiate a read operation if the RDEN bit is not set high. When the read cycle terminates, the RD bit will automatically be cleared to zero and the EEPROM 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.

### **Page Erase Operation to the EEPROM**

The EEPROM is capable of an 8-byte page erase. For page erase operations the start address of the desired EEPROM page should first be placed in the EEA register. The maximum data length for a page is 8 bytes. The EEPROM address lower 3 bits are invalid in page erase operation. The EREN bit in the EEC register should be set high to enable erase operations and the ER bit must be immediately set high to initiate the EEPROM erase process. These two instructions must be executed in two consecutive instruction cycles to activate an erase operation successfully. The global interrupt enable bit EMI should also first be cleared before implementing an erase operation and then set again after a valid erase activation procedure has completed. Note that setting the ER bit high only will not initiate an erase cycle if the EREN bit is not set.

Note: The above steps must be executed sequentially to successfully complete the page erase operation, refer to the corresponding programming example.

As the EEPROM erase 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 erased from the EEPROM. Detecting when the erase cycle has finished can be implemented either by polling the ER bit in the EEC register or by using the EEPROM interrupt. When the erase cycle terminates, the ER bit will be automatically cleared to zero by the microcontroller, indicating that the page data has been erased. The application program can therefore poll the ER bit to determine when the erase cycle has ended. After the erase operation is finished, the EREN bit will be cleared to zero by hardware. The Data EEPROM erased page content will all be zero after a page erase operation.

### **Byte Write Operation to the EEPROM**

For byte write operations the desired EEPROM address should first be placed in the EEA register, then the data to be written should be placed in the EED register. To write data to the EEPROM, 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 to activate a write operation successfully. The global interrupt bit EMI should also first be cleared before implementing any write operations, and then set high again after a valid write activation procedure has completed. Note that setting the WR bit high only will not initiate a write cycle if the WREN bit is not set.

Note: The above steps must be executed sequentially to successfully complete the byte write operation, refer to the corresponding programming example.

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, indicating 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. After the write operation is finished, the WREN bit will be cleared to zero by hardware. Note that a byte erase operation will automatically be executed before a byte write operation is successfully activated.

### 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 interrupt is generated when an EEPROM erase or write cycle has ended. The EEPROM interrupt must first be enabled by setting the DEE bit in the relevant interrupt register. However as the EEPROM interrupt is contained within a Multi-function Interrupt, the associated multi-function interrupt enable bit must also be set. When an EEPROM erase or write cycle ends, the DEF request flag and its associated multi-function interrupt request flag will both be set. If the global, EEPROM and Multi-function interrupts are enabled and the stack is not full, a jump to the associated Multi-function Interrupt vector will take place. When the interrupt is serviced only the Multi-function interrupt flag will be automatically reset, the EEPROM interrupt flag must be manually reset by the application program. 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. When erasing data the ER bit must be set high immediately after the EREN bit has been set high, to ensure the erase cycle executes correctly. The global interrupt bit EMI should also be cleared before a write or erase cycle is executed and then set again after a valid write or erase activation procedure has completed. Note that the device should not enter the IDLE or SLEEP mode until the EEPROM read/write/erase operation is totally complete. Otherwise, the EEPROM read/write/erase operation will fail.

### Programming Examples

#### Reading a Data Byte from the EEPROM - polling method

```
MOV A, 40H           ; setup memory pointer MP1
MOV MP1, A          ; MP1 points to EEC register
MOV A, 01H          ; setup Bank Pointer BP
MOV BP, A
MOV A, EEPROM_ADRES ; user defined address
MOV EEA, 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 function
CLR BP
MOV A, EED        ; move read data to register
MOV READ_DATA, A
```

#### **Erasing a Data Page to the EEPROM – polling method**

```
MOV A, 40H        ; set memory pointer MP1
MOV MP1, A       ; MP1 points to EEC register
MOV A, 01H       ; set Bank Pointer BP
MOV BP, A
MOV A, EEPROM_ADRES ; user defined address
MOV EEA, A
CLR EMI
SET IAR1.6       ; set EREN bit, enable erase operations
SET IAR1.5       ; start Erase Cycle - set ER bit - executed immediately
                  ; after setting EREN bit

SET EMI
BACK:
SZ IAR1.5        ; check for erase cycle end
JMP BACK
CLR BP
```

#### **Writing a Data Byte to the EEPROM - polling method**

```
MOV A, 40H        ; set memory pointer MP1
MOV MP1, A       ; MP1 points to EEC register
MOV A, 01H       ; set Bank Pointer BP
MOV BP, A
MOV A, EEPROM_ADRES ; user defined address
MOV EEA, A
MOV A, EEPROM_DATA ; user defined data
MOV EED, 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 types offer the user a wide range of functions according to their various application requirements. The flexible features of the oscillator functions ensure that the best optimisation can be achieved in terms of speed and power saving. Oscillator selections and operation are selected through the application program by using some control registers.

### Oscillator Overview

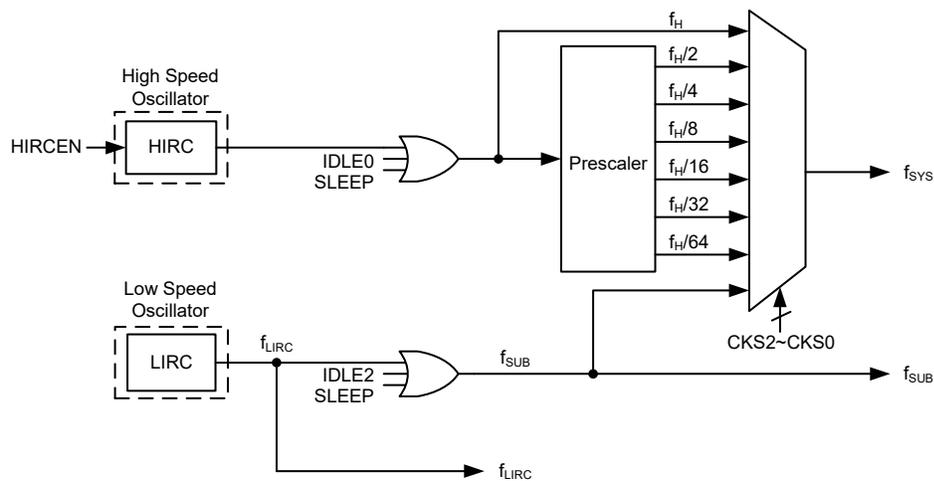
In addition to being the source of the main system clock the oscillators also provide clock sources for the Watchdog Timer and Time Base Interrupts. Fully integrated internal oscillators, requiring no external components, are provided to form a wide range of both fast and slow system oscillators. The higher frequency oscillator provides higher performance but carry with it the disadvantage of higher power requirements, while the opposite is of course true for the lower frequency oscillator. With the capability of dynamically switching between fast and slow system clock, the device has the flexibility to optimize the performance/power ratio, a feature especially important in power sensitive portable applications.

Type	Name	Frequency
Internal High Speed RC	HIRC	8/12/16MHz
Internal Low Speed RC	LIRC	32kHz

Oscillator Types

### System Clock Configurations

There are two oscillator sources, a high speed oscillator and a low speed oscillator. The high speed oscillator is sourced from the internal 8/12/16MHz RC oscillator, HIRC. The low speed oscillator is the internal 32kHz RC oscillator, LIRC. Selecting whether the low or high speed oscillator is used as the system oscillator is implemented using the CKS2~CKS0 bits in the SCC register and as the system clock can be dynamically selected.



System Clock Configurations

### Internal High Speed RC Oscillator – HIRC

The internal RC oscillator is a fully integrated system oscillator requiring no external components. The internal RC oscillator has three fixed frequencies of 8MHz, 12MHz and 16MHz, which are selected by the HIRC1~HIRC0 bits in the HIRCC register. These bits must also be setup to match the selected configuration option frequency to ensure that the HIRC frequency accuracy specified

in the A.C. Characteristics is achieved. Device trimming during the manufacturing process and the inclusion of internal frequency compensation circuits are used to ensure that the influence of the power supply voltage, temperature and process variations on the oscillation frequency are minimised.

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

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

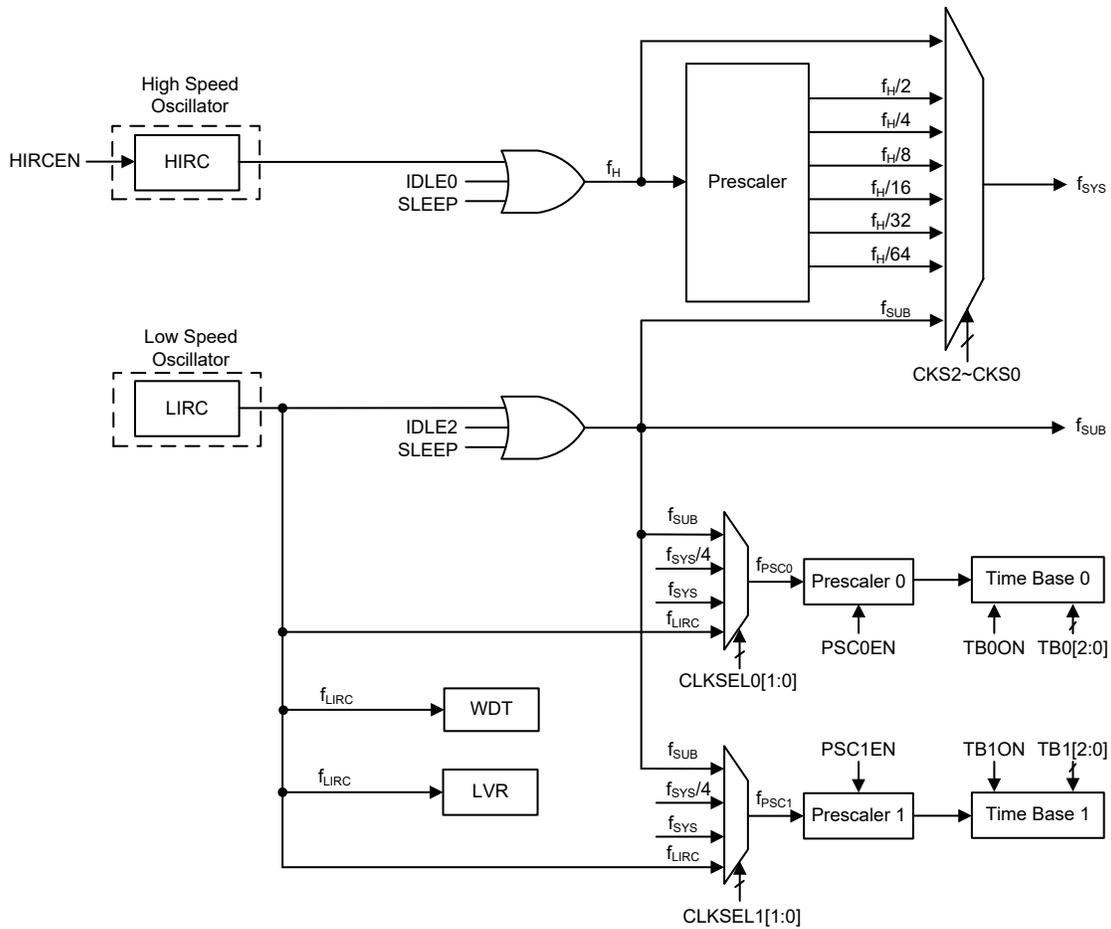
## **Operating Modes and System Clocks**

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

### **System Clocks**

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

The main system clock can come from either a high frequency,  $f_H$ , or low frequency,  $f_{SUB}$ , source, and is selected using the CKS2~CKS0 bits in the SCC register. The high speed system clock is sourced from the HIRC oscillator. The low speed system clock source can sourced from the internal clock  $f_{SUB}$ . If  $f_{SUB}$  is selected then it can be sourced by the LIRC oscillator. The other choice, which is a divided version of the high speed system oscillator has a range of  $f_H/2 \sim f_H/64$ .



### Device Clock Configurations

Note: When the system clock source  $f_{SYS}$  is switched to  $f_{SUB}$  from  $f_H$ , the high speed oscillation can be stopped to conserve the power or continue to oscillate to provide the clock source,  $f_H \sim f_H/64$ , for peripheral circuits to use, which is determined by configuring the corresponding high speed oscillator enable control bit.

### System Operation Modes

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

Operation Mode	CPU	Register Setting			f <sub>sys</sub>	f <sub>H</sub>	f <sub>SUB</sub>	f <sub>LIRC</sub>
		FHIDEN	FSIDEN	CKS2~CKS0				
FAST	On	x	x	000~110	f <sub>H</sub> ~f <sub>H</sub> /64	On	On	On
SLOW	On	x	x	111	f <sub>SUB</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<sub>H</sub> clock will be switched on or off by configuring the corresponding oscillator enable bit in the SLOW mode.  
 2. The f<sub>LIRC</sub> 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 internal high speed oscillator. This mode operates allowing the microcontroller to operate normally with a clock source which will come from the HIRC oscillator. The high speed oscillator will however first be divided by a ratio ranging from 1 to 64, the actual ratio being selected by the CKS2~CKS0 bits in the SCC register. Although a high speed oscillator is used, running the microcontroller at a divided clock ratio reduces the operating current.

### SLOW Mode

This is also a mode where the microcontroller operates normally although now with a slower speed clock source. The clock source used will be from f<sub>SUB</sub>. The f<sub>SUB</sub> clock is derived from the LIRC oscillator.

### SLEEP Mode

The SLEEP Mode is entered when a HALT instruction is executed and when the FHIDEN and FSIDEN bits in the SCC register are both low. In the SLEEP mode the CPU will be stopped. The f<sub>SUB</sub> clock provided to the peripheral function will also be stopped. However, the f<sub>LIRC</sub> clock will 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 on to drive some peripheral functions.

### IDLE1 Mode

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

### Control Registers

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

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

System Operating Mode Control Register List

#### • SCC Register

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

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

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

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

Bit 4~2 Unimplemented, read as “0”

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

0: Disable  
 1: Enable

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

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

0: Disable  
 1: Enable

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

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

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

• **HIRCC Register**

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

Bit 7~4 Unimplemented, read as “0”

Bit 3~2 **HIRC1~HIRC0**: HIRC frequency selection

- 00: 8MHz
- 01: 12MHz
- 10: 16MHz
- 11: 8MHz

When the HIRC oscillator is enabled or the HIRC frequency selection is changed by the application program, the clock frequency will automatically be changed after the HIRCF flag is set to 1.

It is recommended that the HIRC frequency selected by these bits should be the same with the frequency determined by the configuration option to keep the HIRC frequency accuracy specified in the A.C. Characteristics.

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

- 0: Unstable
- 1: 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 or the HIRC frequency selection is changed by application program, the HIRCF bit will first be cleared to 0 and then set to 1 after the HIRC oscillator is stable.

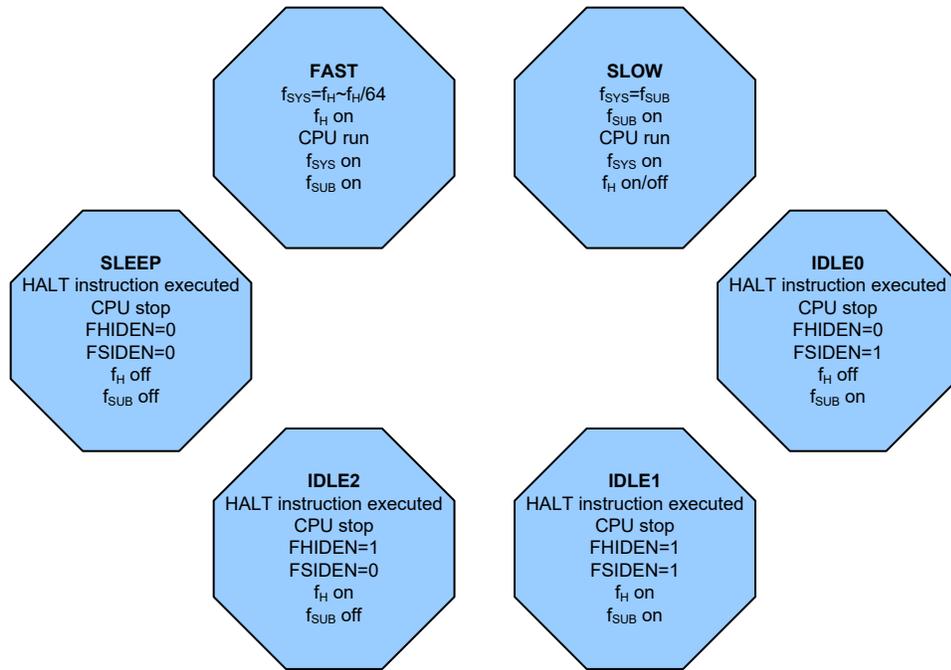
Bit 0 **HIRCEN**: HIRC oscillator enable control

- 0: Disable
- 1: Enable

**Operating Mode Switching**

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

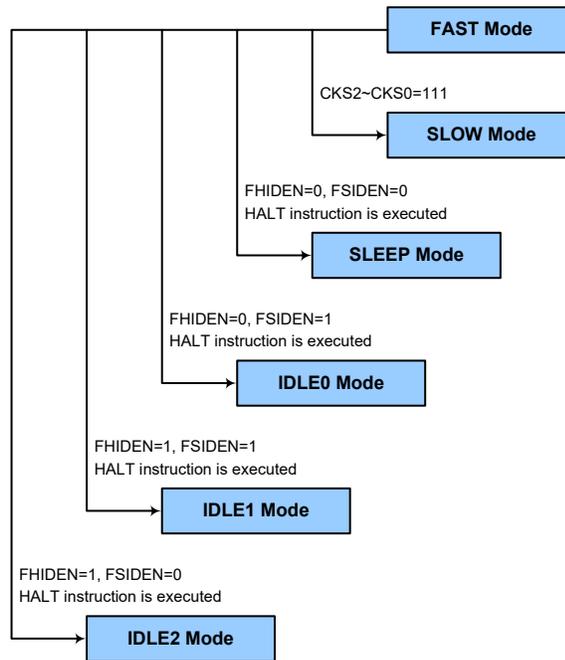
In simple terms, Mode Switching between the FAST Mode and SLOW Mode is executed using the CKS2~CKS0 bits in the SCC register while Mode Switching from the FAST/SLOW Modes to the SLEEP/IDLE Modes is executed via the HALT instruction. When a HALT instruction is executed, whether the device enters the IDLE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



**FAST Mode to SLOW Mode Switching**

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

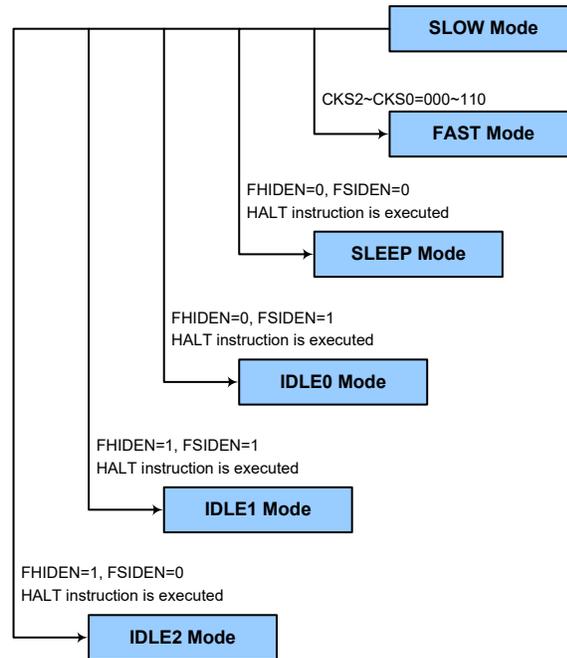
The SLOW Mode is sourced from the LIRC oscillator and therefore requires this oscillator to be stable before full mode switching occurs.



### SLOW Mode to FAST Mode Switching

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

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 WDT timeout 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  $f_{\text{H}}$  clock will be stopped and the application program will stop at the “HALT” instruction, but the  $f_{\text{SUB}}$  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 WDT timeout 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  $f_{\text{H}}$  and  $f_{\text{SUB}}$  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 WDT timeout 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  $f_{\text{H}}$  clock will be on but the  $f_{\text{SUB}}$  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 WDT timeout flag TO will be cleared.
- The WDT will be cleared and resume counting if the WDT function is enabled. If the WDT function is disabled, the WDT will be cleared and then stopped.

### Standby Current Considerations

As the main reason for entering the SLEEP or IDLE Mode is to keep the current consumption of the device to as low a value as possible, perhaps only in the order of several micro-amps except in the IDLE1 and IDLE2 Mode, there are other considerations which must also be taken into account by the circuit designer if the power consumption is to be minimised. Special attention must be made to the I/O pins on the 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 must either be setup as outputs or if setup as inputs must have pull-high resistors connected.

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

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

## **Wake-up**

To minimise power consumption the 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  $\overline{\text{RES}}$  pin reset
- An external falling edge on Port A
- A system interrupt
- A WDT overflow

If the system is woken up by an external  $\overline{\text{RES}}$  pin reset, the device will experience a full system reset, however, if the device is woken up by a WDT overflow, a Watchdog Timer reset will be initiated. Although both of these wake-up methods will initiate a reset operation, the actual source of the wake-up can be determined by examining the TO and PDF flags. The PDF flag is cleared by a system power-up or executing the clear Watchdog Timer instructions and is set when executing the “HALT” instruction. The TO flag is set if a WDT time-out occurs, and causes a wake-up that only resets the Program Counter and Stack Pointer, 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 Port A pin wake-up occurs, the program will resume execution at the instruction following the “HALT” instruction. If the system is woken up by an interrupt, then two possible situations may occur. The first is where the related interrupt is disabled or the interrupt is enabled but the stack is full, in which case the program will resume execution at the instruction following the “HALT” instruction. In this situation, the interrupt which woke up the 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  $2^{18}$  to give longer timeouts, the actual value being chosen using the WS2~WS0 bits in the WDTC register.

### Watchdog Timer Control Register

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

#### • WDTC Register

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

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

10101: Disable

01010: Enable

Other values: Reset MCU

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

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

000:  $2^8/f_{LIRC}$

001:  $2^{10}/f_{LIRC}$

010:  $2^{12}/f_{LIRC}$

011:  $2^{14}/f_{LIRC}$

100:  $2^{15}/f_{LIRC}$

101:  $2^{16}/f_{LIRC}$

110:  $2^{17}/f_{LIRC}$

111:  $2^{18}/f_{LIRC}$

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

#### • RSTFC Register

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

“x”: unknown

Bit 7~4 Unimplemented, read as “0”

Bit 3 **RSTF**: Reset control register software reset flag

Refer to the RES Pin Reset section.

Bit 2 **LVRF**: LVR function reset flag

Refer to the Low Voltage Reset section.

- Bit 1     **LRF:** LVR control register software reset flag  
Refer to the Low Voltage Reset section.
- Bit 0     **WRF:** WDT control register software reset flag  
0: Not occurred  
1: Occurred  
This bit is set to 1 by the WDT control register software reset and cleared by the application program. This bit can only be cleared to zero by 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 Watchdog Timer enable/disable control and the MCU reset. The WDT function will be enabled when the WE4~WE0 bits are set to a value of 01010B while the WDT function will be disabled if the WE4~WE0 bits are equal to 10101B. If the WE4~WE0 bits are set to any other values rather 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.

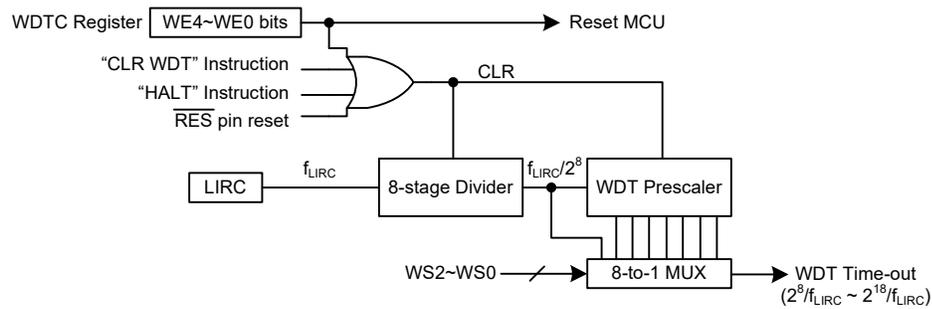
WE4~WE0 Bits	WDT Function
10101B	Disable
01010B	Enable
Any other value	Reset MCU

**Watchdog Timer Function Control**

Under normal program operation, a Watchdog Timer time-out will initialise a device reset and set the status bit TO. However, if the system is in the SLEEP or IDLE Mode, when a Watchdog Timer time-out occurs, the TO bit in the status register will be set and only the Program Counter and Stack Pointer will be reset. Four methods can be adopted to clear the contents of the Watchdog Timer. The first is a WDTC register software reset, which means a certain value except 01010B and 10101B written into the WE4~WE0 bits, the second is using the Watchdog Timer software clear instruction, the third is via a HALT instruction and the fourth is an external hardware reset, which means a low level on the external reset pin if the external reset pin is selected by the RSTC register.

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

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



**Watchdog Timer**

## Reset and Initialisation

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

In addition to the power-on reset, situations may arise where it is necessary to forcefully apply a reset condition when the microcontroller is already running, the  $\overline{\text{RES}}$  line is forcefully pulled low. In such a case, known as a normal operation reset, some of the microcontroller registers remain unchanged allowing the microcontroller to proceed with normal operation after the reset line is allowed to return high.

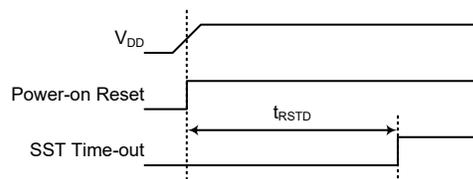
Another reset exists in the form of a Low Voltage Reset, LVR, where a full reset, similar to the  $\overline{\text{RES}}$  reset is implemented in situations where the power supply voltage falls below a certain threshold. Another type of reset is when the Watchdog Timer overflows and resets the microcontroller.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

### Reset Functions

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

#### Power-on Reset

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



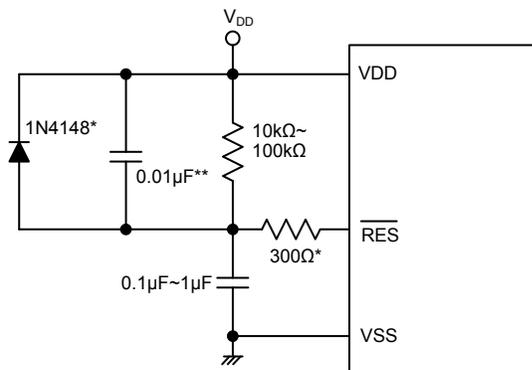
**Power-On Reset Timing Chart**

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

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

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

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

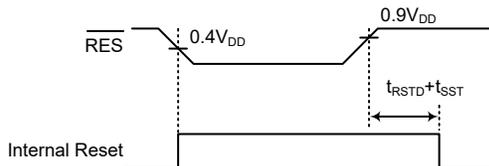


Note: “\*” It is recommended that this component is added for added ESD protection.

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

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

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



**RES Reset Timing Chart**

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

RSTC7~RSTC0 Bits	Reset Function
01010101B	I/O pin or other pin-shared functions
10101010B	$\overline{\text{RES}}$ pin
Any other value	Reset MCU

**Internal Reset Function Control**

**• RSTC Register**

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

Bit 7~0 **RSTC7~RSTC0**: Reset function control  
 01010101: I/O pin or other pin-shared functions  
 10101010:  $\overline{\text{RES}}$  pin  
 Other values: Reset MCU

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

All resets will reset this register to POR value except the WDT time-out hardware warm reset. Note that if the register is set to 10101010 to set the  $\overline{\text{RES}}$  pin, this configuration has higher priority than other related pin-shared controls.

**• RSTFC Register**

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

“x”: unknown

Bit 7~4 Unimplemented, read as “0”

Bit 3 **RSTF**: Reset control register software reset flag  
 0: Not occurred  
 1: Occurred

This bit is set to 1 by the RSTC control register software reset and cleared by the application program. This bit can only be cleared to zero by application program.

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

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

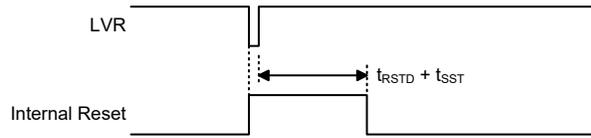
Bit 0 **WRF**: WDT control register software reset flag  
 Refer to the Watchdog Timer Control Register section.

**Low Voltage Reset – LVR**

The microcontroller contains a low voltage reset circuit in order to monitor the supply voltage of the device and provides an MCU reset when the value falls below a certain predefined level.

The LVR function can be enabled or disabled by the LVRC control register. If the supply voltage of the device drops to within a range of  $0.9V \sim V_{\text{LVR}}$  such as might occur when changing the battery in battery powered applications, the LVR will automatically reset the device internally and the LVRF bit in the RSTFC register will also be set to 1. For a valid LVR signal, a low supply voltage, i.e., a voltage in the range between  $0.9V \sim V_{\text{LVR}}$  must exist for a time greater than that specified by  $t_{\text{LVR}}$  in the LVR Electrical Characteristics. If the low supply voltage state does not exceed this value, the LVR will ignore the low supply voltage and will not perform a reset function. The actual  $t_{\text{LVR}}$  value can be selected by the TLVR1~TLVR0 bits in the TLVRC register. If the LVS7~LVS0 bits are set to 01011010B, the LVR function is enabled with a fixed LVR voltage of 1.7V. If the LVS7~LVS0 bits are set to 10100101B, the LVR function is disabled. If the LVS7~LVS0 bits are changed to any other values by environmental noise, the LVR will reset the device after a delay time,  $t_{\text{SRESET}}$ . When this happens, the LRF bit in the RSTFC register will be set to 1. After power on the register will have

the value of 01011010B. Note that the LVR function will be automatically disabled when the device enters the SLEEP or IDLE mode.



**Low Voltage Reset Timing Chart**

• **LVRC Register**

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

Bit 7~0 **LVS7~LVS0**: LVR voltage select  
 01011010: 1.7V  
 10100101: LVR disable

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

When an actual low voltage condition as specified above occurs, 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 two defined register values above, will also result in the generation of an MCU reset. The reset operation will be activated after a delay time,  $t_{SRESET}$ . However in this situation the register contents will be reset to the POR value.

• **TLVRC Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	TLVR1	TLVR0
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	1

Bit 7~2 Unimplemented, read as “0”

Bit 1~0 **TLVR1~TLVR0**: Minimum low voltage width to reset time ( $t_{LVR}$ )  
 00:  $(7\sim8) \times t_{LIRC}$   
 01:  $(31\sim32) \times t_{LIRC}$   
 10:  $(63\sim64) \times t_{LIRC}$   
 11:  $(127\sim128) \times t_{LIRC}$

• **RSTFC Register**

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

“x”: unknown

Bit 7~4 Unimplemented, read as “0”

Bit 3 **RSTF**: Reset control register software reset flag  
 Refer to the  $\overline{RES}$  Pin Reset section.

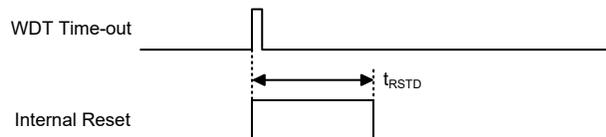
Bit 2 **LVRF**: LVR function reset flag  
 0: Not occurred  
 1: Occurred

This bit is set to 1 when a specific low voltage reset condition occurs. This bit can only be cleared to zero by application program.

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

### Watchdog Time-out Reset during Normal Operation

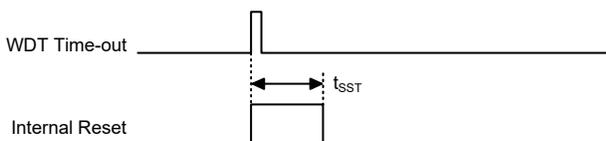
When the Watchdog time-out Reset during normal operations in the FAST or SLOW mode occurs, the Watchdog time-out flag TO will be set to “1”.



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

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

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



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

### Reset Initial Conditions

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

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

“u”: 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 the microcontroller internal registers. Note that where more than one package type exists the table reflect the situation for the larger package type.

Register	Power-On Reset	RES Reset (Normal Operation)	RES Reset (IDLE/SLEEP)	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)
IAR0	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
MP0	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
IAR1	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
MP1	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
BP	---- --00	---- --00	---- --00	---- --00	---- --uu
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
PCL	0000 0000	0000 0000	0000 0000	0000 0000	0000 0000
TBLP	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLH	--xx xxxx	--uu uuuu	--uu uuuu	--uu uuuu	--uu uuuu
TBHP	---- --xx	---- --uu	---- --uu	---- --uu	---- --uu
STATUS	--00 xxxx	--uu uuuu	--01 uuuu	--1u uuuu	--11 uuuu
RSTFC	---- 0x00	---- uuuu	---- uuuu	---- uuuu	---- uuuu
MF11	--00 --00	--00 --00	--00 --00	--00 --00	--uu --uu
PA	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAPU	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PAWU	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
LVPUC	---- ---0	---- ---0	---- ---0	---- ---0	---- ---u
IFS0	-000 -000	-000 -000	-000 -000	-000 -000	-uuu -uuu
IFS1	---- -000	---- -000	---- -000	---- -000	---- -uuu
PAS0	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PAS1	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
SLEDC	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
INTC0	-000 0000	-000 0000	-000 0000	-000 0000	-uuu uuuu
INTC1	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
MF10	-000 -000	-000 -000	-000 -000	-000 -000	-uuu -uuu
INTEG	---- 0000	---- 0000	---- 0000	---- 0000	---- uuuu
PB	--11 1111	--11 1111	--11 1111	--11 1111	--uu uuuu
PBC	--11 1111	--11 1111	--11 1111	--11 1111	--uu uuuu
PBPU	--00 0000	--00 0000	--00 0000	--00 0000	--uu uuuu
PBS0	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PBS1	---- 0000	---- 0000	---- 0000	---- 0000	---- uuuu
SCC	000- --00	000- --00	000- --00	000- --00	uuu- --uu
HIRCC	---- 0001	---- 0001	---- 0001	---- 0001	---- uuuu
WDTC	0101 0011	0101 0011	0101 0011	0101 0011	uuuu uuuu
LVRC	0101 1010	0101 1010	0101 1010	0101 1010	uuuu uuuu
TLVRC	---- --01	---- --01	---- --01	---- --01	---- --uu
VBGC	---- 0---	---- 0---	---- 0---	---- 0---	---- u---
RSTC	0101 0101	0101 0101	0101 0101	0101 0101	uuuu uuuu
PSC0R	---- -000	---- -000	---- -000	---- -000	---- -uuu
TB0C	0--- -000	0--- -000	0--- -000	0--- -000	u--- -uuu
PSC1R	---- -000	---- -000	---- -000	---- -000	---- -uuu

Register	Power-On Reset	RES Reset (Normal Operation)	RES Reset (IDLE/SLEEP)	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)
TB1C	0--- -000	0--- -000	0--- -000	0--- -000	u--- -uuu
STMC0	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
STMC1	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
STMDL	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
STMDH	---- --00	---- --00	---- --00	---- --00	---- --uu
STMAL	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
STMAH	---- --00	---- --00	---- --00	---- --00	---- --uu
SADC0	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
SADC1	0000 -000	0000 -000	0000 -000	0000 -000	uuuu -uuu
SADC2	0--0 0000	0--0 0000	0--0 0000	0--0 0000	u--u uuuu
SADOL	xxxx ----	xxxx ----	xxxx ----	xxxx ----	uuuu ---- (ADRF=0)
					uuuu uuuu (ADRF=1)
SADOH	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu (ADRF=0)
					---- uuuu (ADRF=1)
ORMC	0000 0000	0000 0000	0000 0000	0000 0000	0000 0000
EEA	--00 0000	--00 0000	--00 0000	--00 0000	--uu uuuu
EED	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTMC0	0000 0---	0000 0---	0000 0---	0000 0---	uuuu u---
PTMC1	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTMDL	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTMDH	---- --00	---- --00	---- --00	---- --00	---- --uu
PTMAL	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTMAH	---- --00	---- --00	---- --00	---- --00	---- --uu
PTMRPL	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTMRPH	---- --00	---- --00	---- --00	---- --00	---- --uu
EEC	000- 0000	000- 0000	000- 0000	000- 0000	uuu- uuuu

Note: "u" stands for unchanged  
"x" stands for unknown  
"--" stands for unimplemented

## Input/Output Ports

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

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

Register Name	Bit							
	7	6	5	4	3	2	1	0
PA	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0
PAC	PAC7	PAC6	PAC5	PAC4	PAC3	PAC2	PAC1	PAC0
PAPU	PAPU7	PAPU6	PAPU5	PAPU4	PAPU3	PAPU2	PAPU1	PAPU0
PAWU	PAWU7	PAWU6	PAWU5	PAWU4	PAWU3	PAWU2	PAWU1	PAWU0
PB	—	—	PB5	PB4	PB3	PB2	PB1	PB0
PBC	—	—	PBC5	PBC4	PBC3	PBC2	PBC1	PBC0
PBPU	—	—	PBPU5	PBPU4	PBPU3	PBPU2	PBPU1	PBPU0
LVPUC	—	—	—	—	—	—	—	LVPUC

“—”: 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 LVPUC and PxPU registers, and are implemented using weak PMOS transistors. The PxPU register is used to determine whether the pull-high function is enabled or not while the LVPUC register is used to select the pull-high resistors value for low voltage power supply applications.

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

### • 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” is the Port name which can be A or B. However, the actual available bits for each I/O Port may be different.

**• LVPUC Register**

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

Bit 7~1 Unimplemented, read as “0”

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

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

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

This bit is used to select the pull-high resistor value for low voltage power supply applications. The LVPU bit is only available when the corresponding pin pull-high function is enabled by setting the relevant pull-high control bit high. This bit will have no effect when the pull-high function is disabled.

**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 register 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 pin Wake-up function control

0: Disable

1: Enable

**I/O Port Control Registers**

Each I/O port has its own control register which controls 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” is the Port name which can be A or B. However, the actual available bits for each I/O Port may be different.

**I/O Port Source Current Selection**

The device supports different output source current driving capability for each I/O port. With the SLEDC register, specific I/O port can support four levels of the source current driving capability. These source current selection bits are available only 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 source current for different applications.

• **SLEDC Register**

Bit	7	6	5	4	3	2	1	0
Name	SLEDC7	SLEDC6	SLEDC5	SLEDC4	SLEDC3	SLEDC2	SLEDC1	SLEDC0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 **SLEDC7~SLEDC6:** 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 5~4 **SLEDC5~SLEDC4:** PB3~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.)

Bit 3~2 **SLEDC3~SLEDC2:** PA7~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 **SLEDC1~SLEDC0:** PA3~PA0 Source Current Selection

00: Source current=Level 0 (min.)

01: Source current=Level 1

10: Source current=Level 2

11: Source current=Level 3 (max.)

**Pin-shared Functions**

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

### Pin-shared Function Selection Registers

The limited number of supplied pins in a package can impose restrictions on the amount of functions a certain device can contain. However by allowing the same pins to share several different functions and providing a means of function selection, a wide range of different functions can be incorporated into even relatively small package sizes. The device includes Port “x” output function selection register “n”, labeled as PxSn, and Input Function source pin selection register, labeled as IFSi, 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, xTCK, xTPI, 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 setup as 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
IFS0	—	PTPIPS	PTCKPS1	PTCKPS0	—	STPIPS	STCKPS1	STCKPS0
IFS1	—	—	—	—	—	INT1PS	INT0PS1	INT0PS0
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	—	—	—	—	PBS13	PBS12	PBS11	PBS10

**Pin-shared Function Selection Register List**

#### • IFS0 Register

Bit	7	6	5	4	3	2	1	0
Name	—	PTPIPS	PTCKPS1	PTCKPS0	—	STPIPS	STCKPS1	STCKPS0
R/W	—	R/W	R/W	R/W	—	R/W	R/W	R/W
POR	—	0	0	0	—	0	0	0

Bit 7 Unimplemented, read as “0”

Bit 6 **PTPIPS**: PTPI input source pin selection  
 0: PA5  
 1: PB0

Bit 5~4 **PTCKPS1~PTCKPS0**: PTCK input source pin selection  
 00: PA4  
 01: PA6  
 10: PA7  
 11: PB1

Bit 3 Unimplemented, read as “0”

Bit 2 **STPIPS**: STPI input source pin selection  
 0: PA6  
 1: PA0

Bit 1~0     **STCKPS1~STCKPS0**: STCK input source pin selection  
             00: PA3  
             01: PA2  
             10: PA6  
             11: PA7

• **IFS1 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	INT1PS	INT0PS1	INT0PS0
R/W	—	—	—	—	—	R/W	R/W	R/W
POR	—	—	—	—	—	0	0	0

Bit 7~3     Unimplemented, read as “0”  
 Bit 2       **INT1PS**: INT1 input source pin selection  
             0: PA5  
             1: PA2  
 Bit 1~0     **INT0PS1~INT0PS0**: INT0 input source pin selection  
             00: PA3  
             01: PA4  
             10: PA0  
             11: PA7

• **PAS0 Register**

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

Bit 7~6     **PAS07~PAS06**: PA3 pin-shared function selection  
             00: PA3/INT0/STCK  
             01: PA3/INT0/STCK  
             10: PTPB  
             11: AN3  
 Bit 5~4     **PAS05~PAS04**: PA2 pin-shared function selection  
             00: PA2/INT1/STCK  
             01: STP  
             10: AN2  
             11: VREF  
 Bit 3~2     **PAS03~PAS02**: PA1 pin-shared function selection  
             00: PA1  
             01: STPB  
             10: AN1  
             11: VREFI  
 Bit 1~0     **PAS01~PAS00**: PA0 pin-shared function selection  
             00: PA0/INT0/STPI  
             01: PA0/INT0/STPI  
             10: STP  
             11: AN0

**• 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/INT0/PTCK/STCK/RES

01: PA7/INT0/PTCK/STCK/RES

10: STPB

11: AN6

Note that when the RSTC register has been set to 10101010B to select the  $\overline{\text{RES}}$  pin function, the other pin-shared function selection on the PA7 pin will be invalid.

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

00: PA6/PTCK/STCK/STPI

01: PA6/PTCK/STCK/STPI

10: PA6/PTCK/STCK/STPI

11: STP

Bit 3~2 **PAS13~PAS12:** PA5 pin-shared function selection

00: PA5/INT1/PTPI

01: PTP

10: STPB

11: AN5

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

00: PA4/INT0/PTCK

01: PA4/INT0/PTCK

10: PA4/INT0/PTCK

11: STP

**• 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: PB3

10: PB3

11: PTP

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

00: PB2

01: PB2

10: PB2

11: PTPB

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

00: PB1/PTCK

01: PB1/PTCK

10: PB1/PTCK

11: STPB

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

00: PB0/PTPI

01: PB0/PTPI

10: AN4

11: VREF

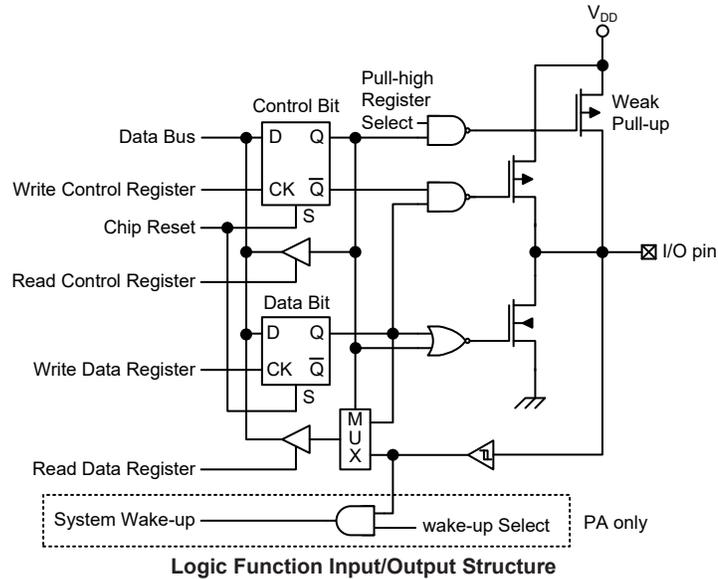
• **PBS1 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	PBS13	PBS12	PBS11	PBS10
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 **PBS13~PBS12**: PB5 pin-shared function selection
  - 00: PB5
  - 01: PB5
  - 10: PB5
  - 11: PTP
- Bit 1~0 **PBS11~PBS10**: PB4 pin-shared function selection
  - 00: PB4
  - 01: PB4
  - 10: PB4
  - 11: PTPB

**I/O Pin Structures**

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



**Programming Considerations**

Within the user program, one of the things first to consider is port initialisation. After a reset, all of the I/O data and port control registers will be set high. This means that all I/O pins will 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 two Timer Modules, generally abbreviated to the name TM. The TMs are multi-purpose timing units and serve to provide operations such as Timer/Counter, Input Capture, Compare Match Output and Single Pulse Output as well as being the functional unit for the generation of PWM signals. Each of the TMs has two interrupts. The addition of input and output pins for each TM ensures that users are provided with timing units with a wide and flexible range of features.

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

### Introduction

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

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

**TM Function Summary**

### TM Operation

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

### TM Clock Source

The clock source which drives the main counter in each TM can originate from various sources. The selection of the required clock source is implemented using the xTCK2~xTCK0 bits in the xTM control registers, where “x” stands for S or P type TM. 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 xTCK pin. The xTCK pin clock source is used to allow an external signal to drive the TM as an external clock source for event counting.

### TM Interrupts

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

### TM External Pins

Each of the TMs, irrespective of what type, has two TM input pins, with the label xTCK and xTPI. The xTM input pin, xTCK, is essentially a clock source for the xTM and is selected using the xTCK2~xTCK0 bits in the xTMC0 register. This external TM input pin allows an external clock source to drive the internal TM. The xTCK input pin can be chosen to have either a rising or falling active edge. The xTCK pin is also used as the external trigger input pin in single pulse output mode for the xTM.

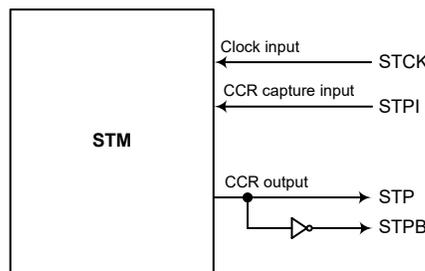
The other xTM input pin, xTPI, is the capture input whose active edge can be a rising edge, a falling edge or both rising and falling edges and the active edge transition type is selected using the xTIO1~xTIO0 bits in the xTMC1 register. There is another capture input, PTCK, for PTM capture input mode, which can be used as the external trigger input source except the PTPI pin.

The TMs each have two output pins, with the label xTP and xTPB. When the TM is in the Compare Match Output Mode, the xTP pin can be controlled by the TM to switch to a high or low level or to toggle when a compare match situation occurs. The xTPB pin outputs the inverted signal of the xTP. The external xTP and xTPB output pins are also the pins where the TM generates the PWM output waveform.

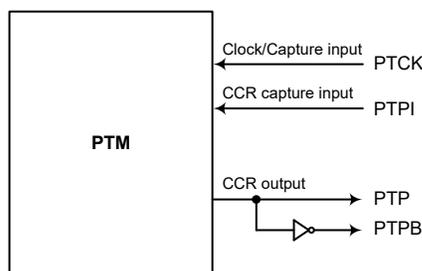
As the TM input and output pins are pin-shared with other functions, the TM input and output functions must first be selected using relevant pin-shared function selection. The details of the pin-shared function selection are described in the pin-shared function section.

STM		PTM	
Input	Output	Input	Output
STCK, STPI	STP, STPB	PTCK, PTPI	PTP, PTPB

**TM External Pins**



**STM Function Pin Block Diagram**

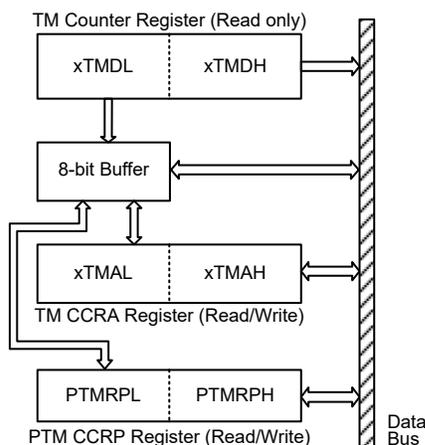


**PTM Function Pin Block Diagram**

### Programming Considerations

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

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



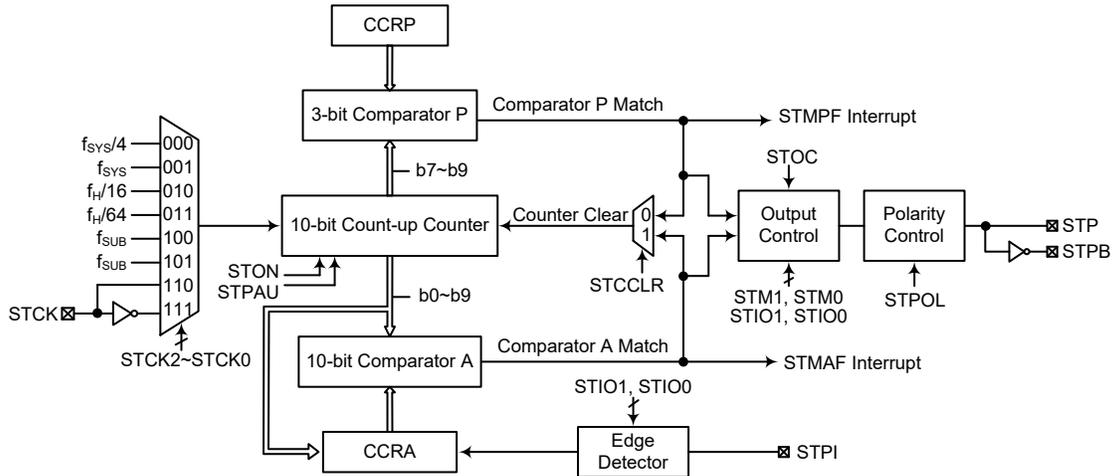
The following steps show the read and write procedures:

- Writing Data to CCRA or CCRP
  - ♦ Step 1. Write data to Low Byte xTMAL or PTMRPL
    - Note that here data is only written to the 8-bit buffer.
  - ♦ Step 2. Write data to High Byte xTMAH 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, CCRA or CCRP
  - ♦ Step 1. Read data from the High Byte xTMDH, xTMAH 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 xTMDL, xTMAL or PTMRPL
  - This step reads data from the 8-bit buffer.

## Standard Type TM – STM

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



Note: The STM external pins are pin-shared with other functions, therefore before using the STM function, the pin-shared function registers must be set properly to enable the STM pin function. The STCK and STPI pins, 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 Type 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 the 10 bits and therefore compares all counter bits.

The only way of changing the value of the 10-bit counter using the application program, is to clear the counter by changing the 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, an 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 two 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 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**: STM counter clock selection

000:  $f_{SYS}/4$   
001:  $f_{SYS}$   
010:  $f_H/16$   
011:  $f_H/64$   
100:  $f_{SUB}$   
101:  $f_{SUB}$   
110: 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, PWM Output Mode or Single Pulse 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. Setting 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**: STM operating mode selection
- 00: Compare Match Output Mode
  - 01: Capture Input Mode
  - 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**: STM external pin function selection
- Compare Match Output Mode
- 00: No change
  - 01: Output low
  - 10: Output high
  - 11: Toggle output
- PWM Output Mode/Single Pulse Output Mode
- 00: PWM output inactive state
  - 01: PWM output active state
  - 10: PWM output
  - 11: Single Pulse Output
- Capture Input Mode
- 00: Input capture at rising edge of STPI
  - 01: Input capture at falling edge of STPI
  - 10: Input capture at rising/falling edge of STPI
  - 11: Input capture disabled
- 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 STM 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 STM output pin when the STON bit changes from low to high.

- Bit 2 **STPOL**: STM STP output polarity control  
     0: Non-inverted  
     1: Inverted

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, Single Pulse Output or Capture Input 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 five operating modes, Compare Match Output Mode, PWM Output Mode, Single Pulse Output Mode, Capture Input Mode or Timer/Counter Mode. The operating mode is selected using the STM1 and STM0 bits in the STMC1 register.

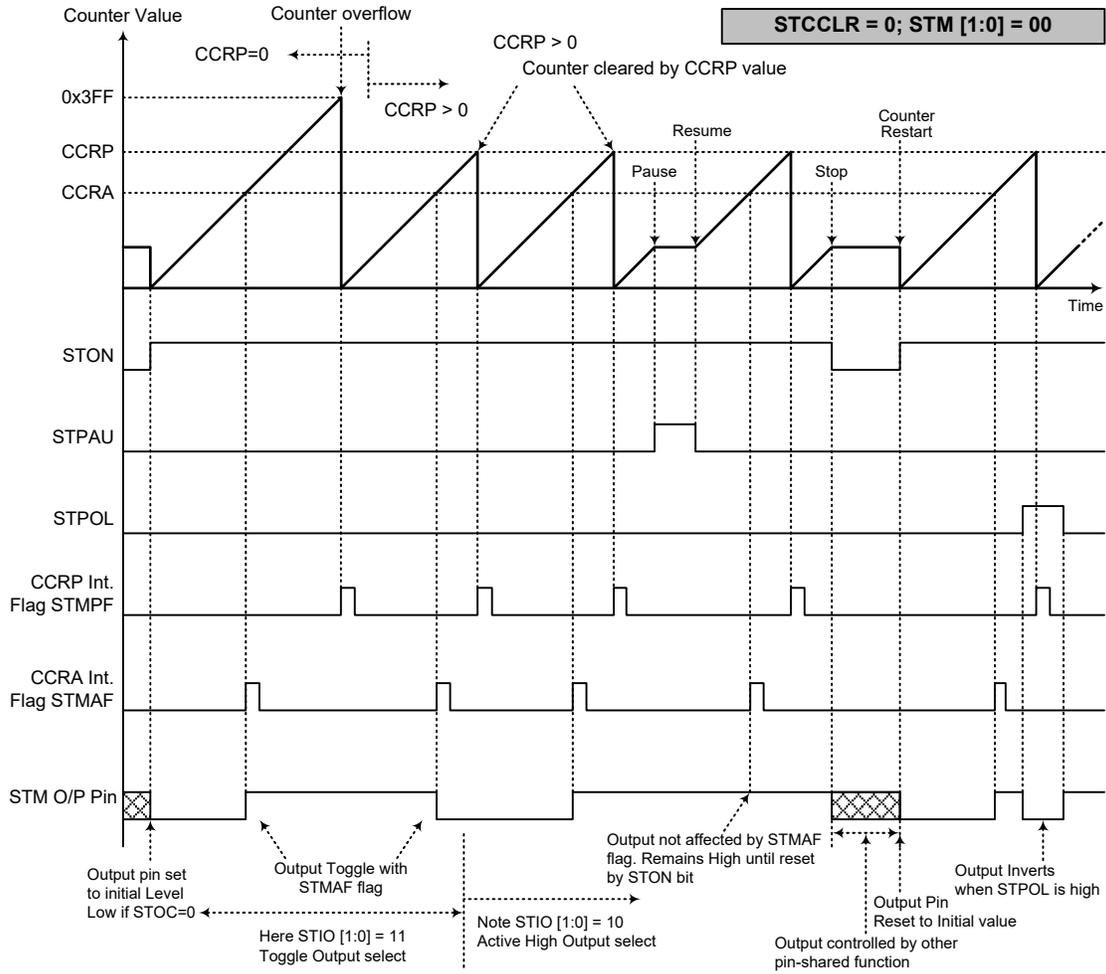
### Compare Match Output Mode

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 cannot be cleared to “0”.

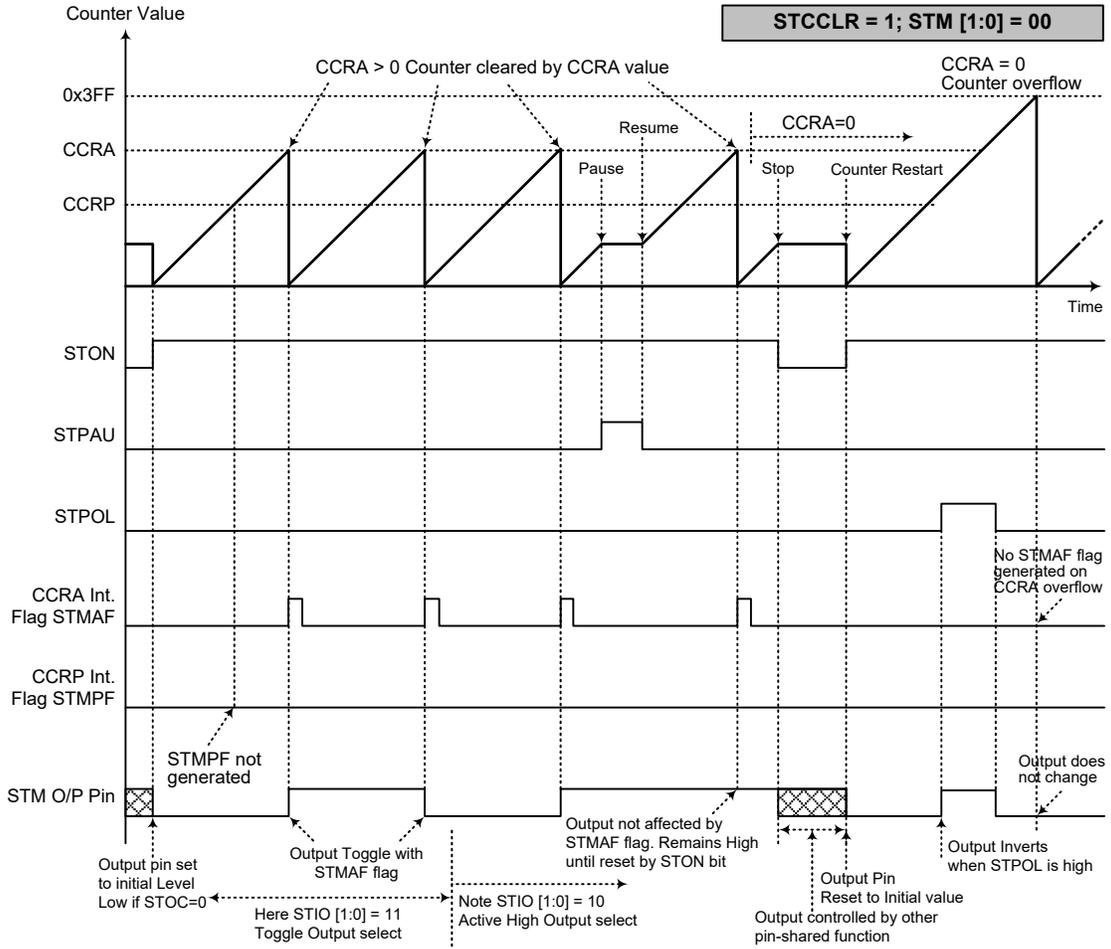
If the CCRA bits are all zero, the counter will overflow when it reaches its maximum 10-bit, 3FF Hex, value, however here the STMAF interrupt request flag will not be generated.

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 an 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.



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

- 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 initial state by an 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 an STON bit rising edge  
 4. An 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. 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 waveform 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 CCRP and CCRA registers are used to generate the PWM waveform, one register is used to clear the internal counter and thus control the PWM waveform frequency, while the other one register 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  $f_{SYS}=16\text{MHz}$ , STM clock source is  $f_{SYS}/4$ ,  $CCRP=4$  and  $CCRA=128$ ,

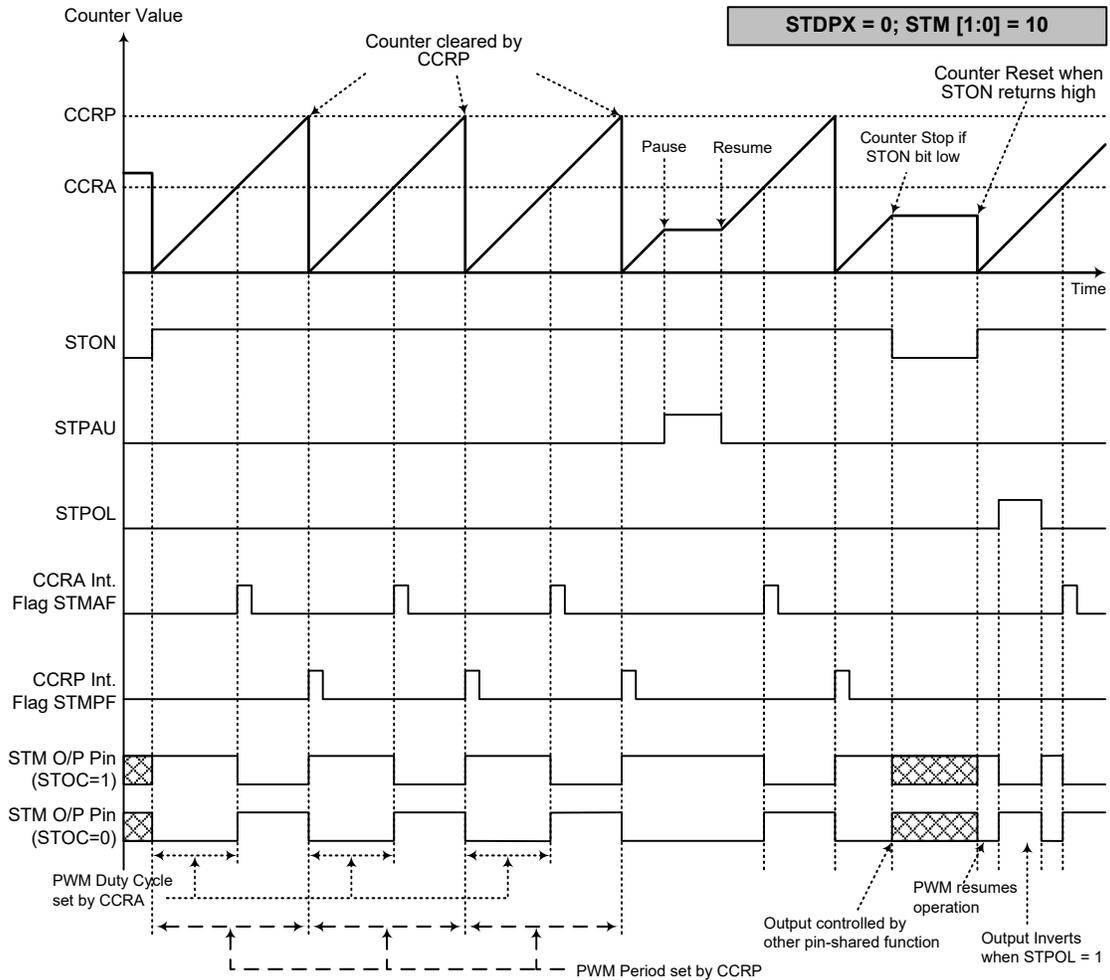
STM PWM output frequency= $(f_{SYS}/4)/(4 \times 128)=f_{SYS}/2048=7.8125\text{kHz}$ , duty= $128/(4 \times 128)=25\%$ .

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**

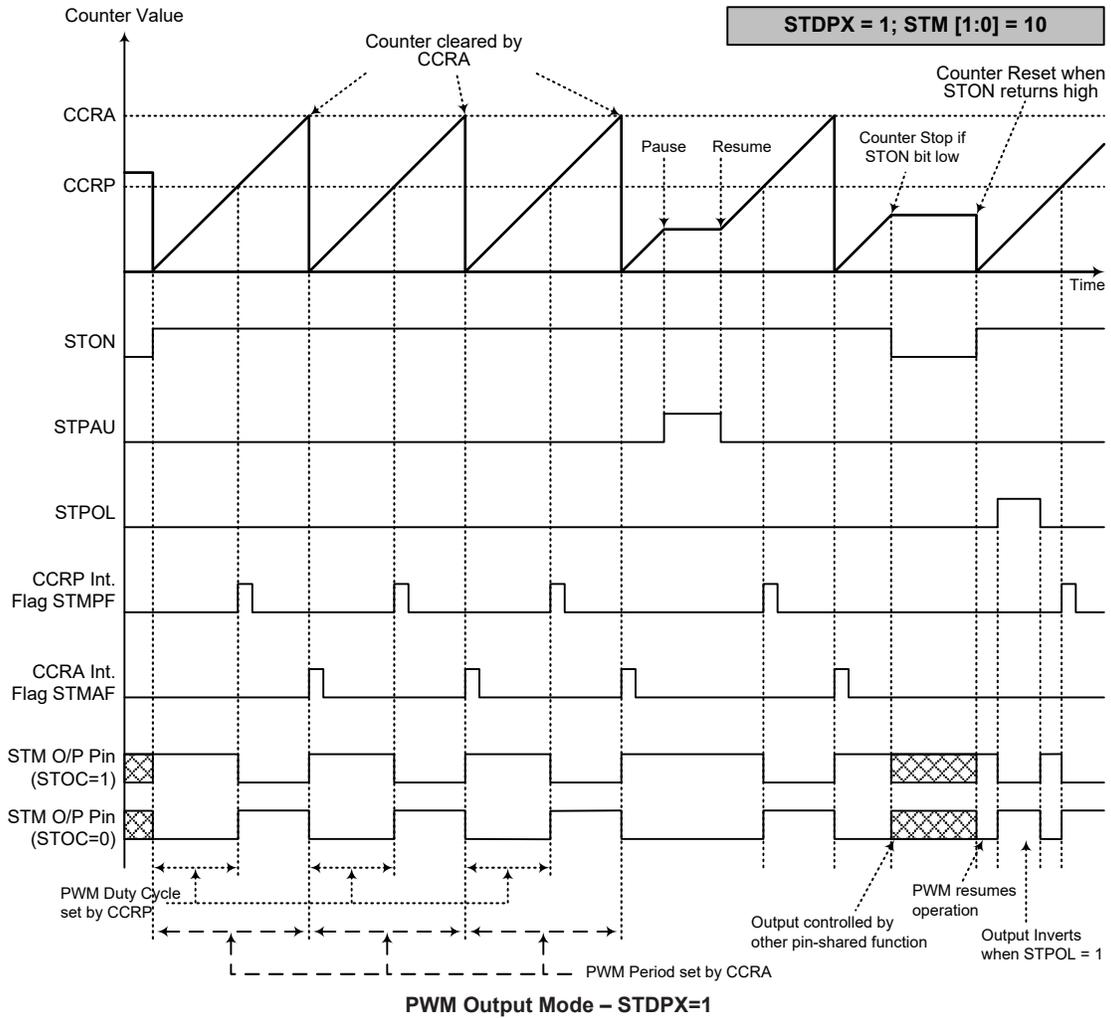
CCRP	1~7	0
Period	CCRA	
Duty	CCRP×128	1024

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.



**PWM Output Mode – STDPX=0**

- 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



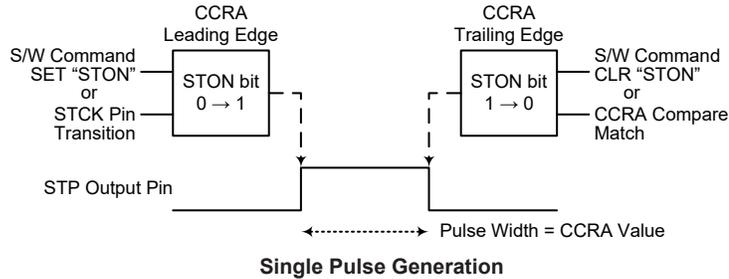
- Note: 1. Here STDPX=1 – Counter cleared by CCRA  
 2. A counter clear sets the PWM Period  
 3. The internal PWM function continues even when STIO[1:0]=00 or 01  
 4. The STCCLR bit has no influence on PWM operation

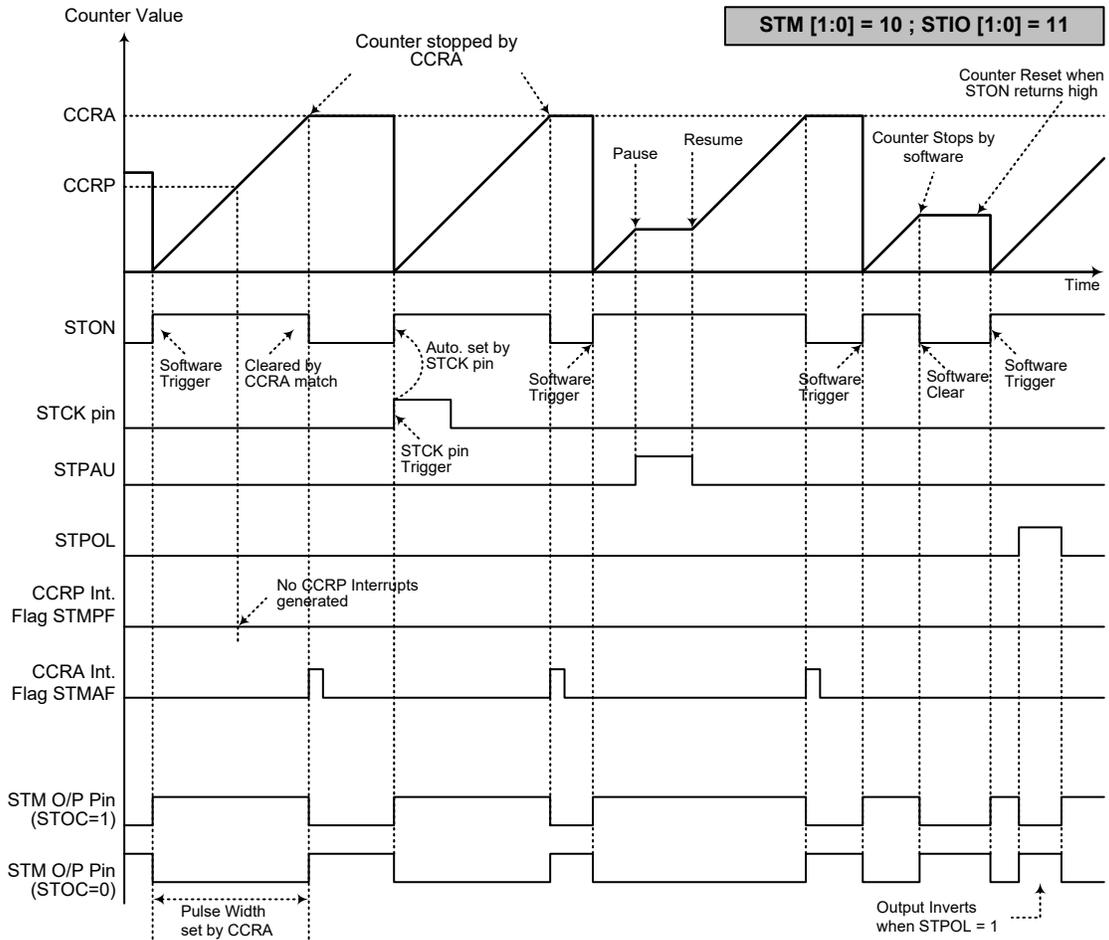
**Single Pulse Output Mode**

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 Output 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 an 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 Output Mode CCRP is not used. The STCCLR and STDPX bits are not used in this mode.





**Single Pulse Output Mode**

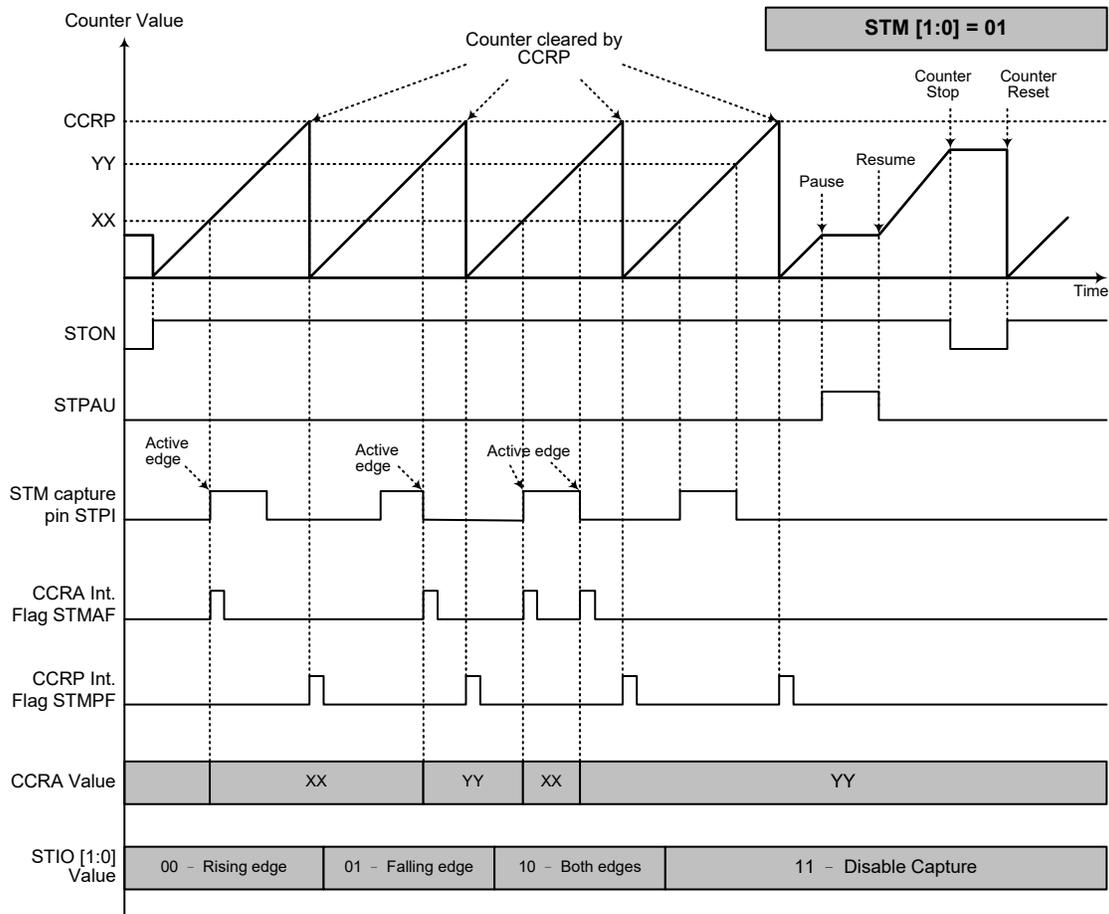
- Note: 1. Counter stopped by CCRA  
 2. CCRP is not used  
 3. The pulse triggered by the STCK pin or by setting the STON bit high  
 4. An STCK pin active edge will automatically set the STON bit high  
 5. In the Single Pulse Output Mode, STIO[1:0] must be set to "11" and cannot be changed

### **Capture Input Mode**

To select this mode bits STM1 and STM0 in the STMC1 register should be set to 01 respectively. This mode enables external signals to capture and store the present value of the internal counter and can therefore be used for applications such as pulse width measurements. The external signal is supplied on the STPI pin, whose active edge can be a rising edge, a falling edge or both rising and falling edges; the active edge transition type is selected using the STIO1 and STIO0 bits in the STMC1 register. The counter is started when the STON bit changes from low to high which is initiated using the application program.

When the required edge transition appears on the STPI pin the present value in the counter will be latched into the CCRA registers and an STM interrupt generated. Irrespective of what events occur on the STPI pin the counter will continue to free run until the STON bit changes from high to low. When a CCRP compare match occurs the counter will reset back to zero; in this way the CCRP value can be used to control the maximum counter value. When a CCRP compare match occurs from Comparator P, an STM interrupt will also be generated. Counting the number of overflow interrupt signals from the CCRP can be a useful method in measuring long pulse widths. The STIO1 and STIO0 bits can select the active trigger edge on the STPI pin to be a rising edge, falling edge or both edge types. If the STIO1 and STIO0 bits are both set high, then no capture operation will take place irrespective of what happens on the STPI pin, however it must be noted that the counter will continue to run. The STCCLR and STDPX bits are not used in this mode.

There are some considerations that should be noted. If the captured pulse width is less than 2 timer clock periods, it may be ignored by hardware. After the counter value is latched to the CCRA registers by an active capture edge, the STMAF flag will be set high after 0.5 timer clock periods. The delay time from the active capture edge received to the action of latching counter value to CCRA registers is less than 1.5 timer clock periods.

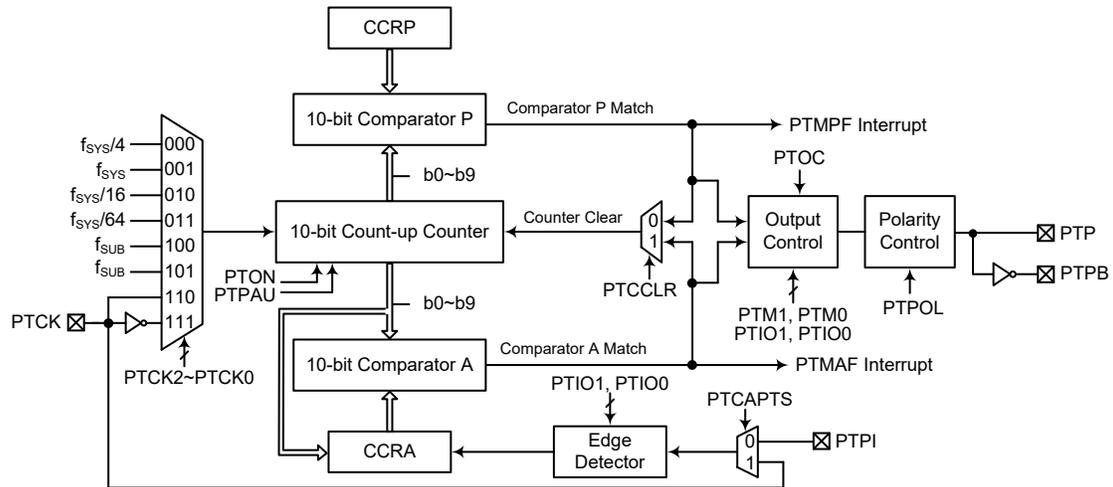


**Capture Input Mode**

- Note: 1. STM[1:0]=01 and active edge set by the STIO[1:0] bits  
 2. An STM Capture input pin active edge transfers the counter value to CCRA  
 3. STCCLR bit not used  
 4. No output function – STOC and STPOL bits are not used  
 5. CCRP determines the counter value and the counter has a maximum count value when CCRP is equal to zero  
 6. The capture input mode cannot be used if the selected STM counter clock is not available

## Periodic Type TM – PTM

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



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

10-bit Periodic Type TM Block Diagram

### Periodic Type TM Operation

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

The only way of changing the value of the 10-bit counter using the application program is to clear the counter by changing the 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 an input pin and can also control two output pins. All operating setup conditions are selected using relevant internal registers.

### Periodic Type TM Register Description

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

Register Name	Bit							
	7	6	5	4	3	2	1	0
PTMC0	PTPAU	PTCK2	PTCK1	PTCK0	PTON	—	—	—
PTMC1	PTM1	PTM0	PTIO1	PTIO0	PTOC	PTPOL	PTCAPTS	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 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**: PTM counter clock selection  
000:  $f_{SYS}/4$   
001:  $f_{SYS}$   
010:  $f_H/16$   
011:  $f_H/64$   
100:  $f_{SUB}$   
101:  $f_{SUB}$   
110: 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  $f_{SYS}$  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 **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, PWM Output Mode or 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	PTCAPTS	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**: PTM operating mode selection  
 00: Compare Match Output Mode  
 01: Capture Input Mode  
 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**: PTM external pin function selection  
 Compare Match Output Mode  
 00: No change  
 01: Output low  
 10: Output high  
 11: Toggle output  
 PWM Output Mode/Single Pulse Output Mode  
 00: PWM output inactive state  
 01: PWM output active state  
 10: PWM output  
 11: Single Pulse Output  
 Capture Input Mode  
 00: Input capture at rising edge of PTPI or PTCK  
 01: Input capture at falling edge of PTPI or PTCK  
 10: Input capture at rising/falling edge of PTPI or PTCK  
 11: Input capture disabled

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 PTM 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  
 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**: PTM PTP output polarity control

- 0: Non-inverted
- 1: Inverted

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 **PTCAPTS**: PTM capture trigger source selection

- 0: From PTPI pin
- 1: From PTCK pin

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, Single Pulse Output or Capture Input 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 Operation Modes**

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

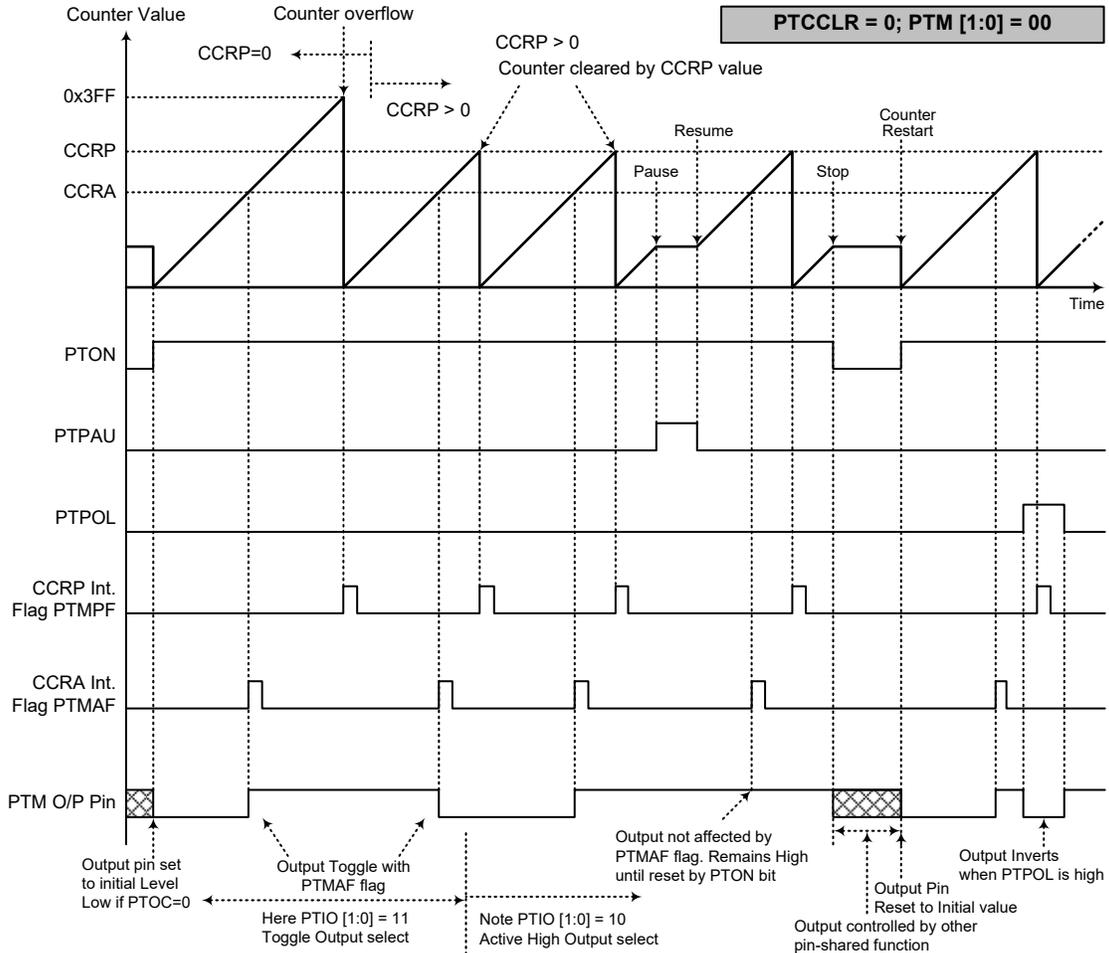
### **Compare Match Output Mode**

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 “0”.

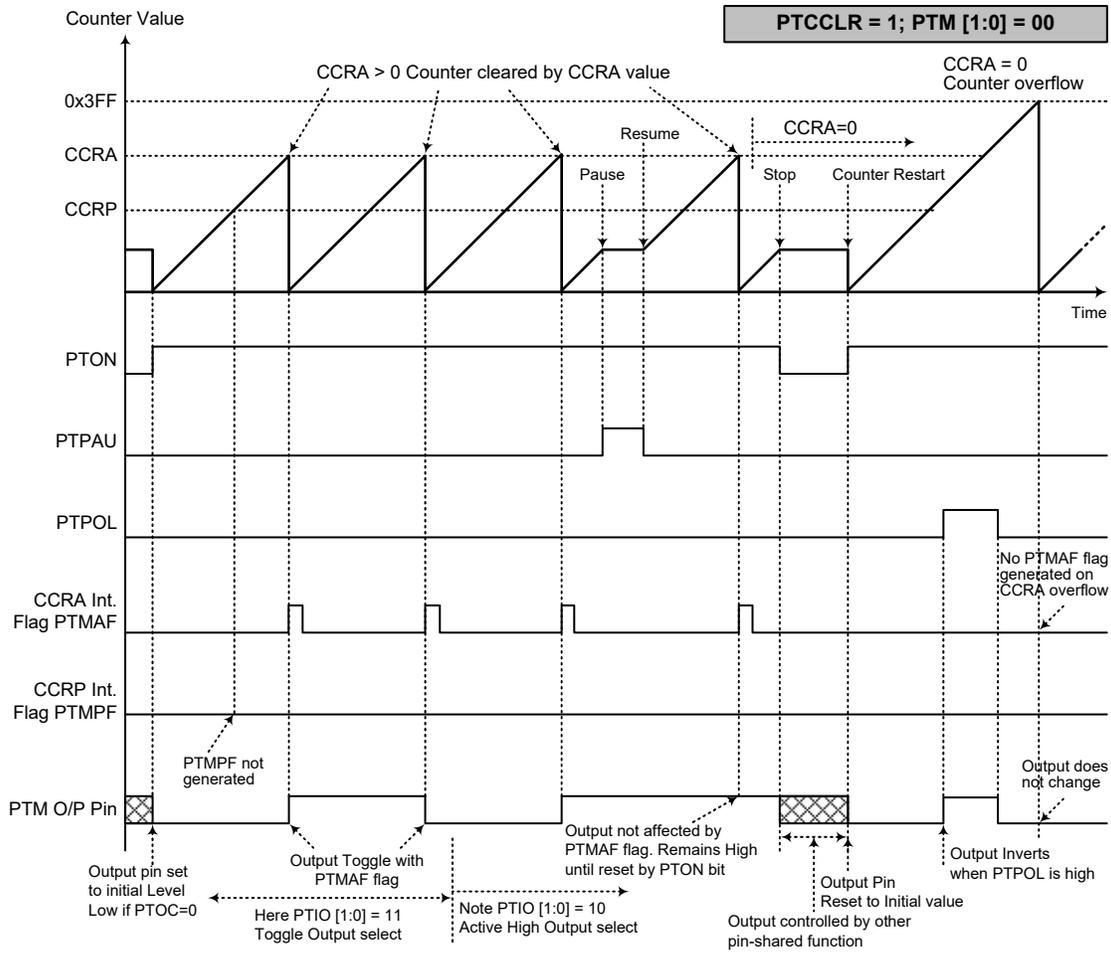
If the CCRA bits are all zero, the counter will overflow when it reaches its maximum 10-bit, 3FF Hex, value, however here the PTMAF interrupt request flag will not be generated.

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 – PTCCCLR=1**

- Note: 1. With PTCCCLR=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 PTCCCLR=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. 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 as the PWM period. Both of the CCRP and CCRA registers are used to generate the PWM waveform, the CCRP register is used to clear the internal counter and thus control the PWM waveform frequency, while the CCRA register is used to control the duty cycle. The PWM waveform frequency and duty cycle can therefore be controlled by the values in the CCRA and CCRP registers.

An interrupt flag, one for each of the CCRA and CCRP, will be generated when a compare match occurs from either Comparator A or Comparator P. The 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.

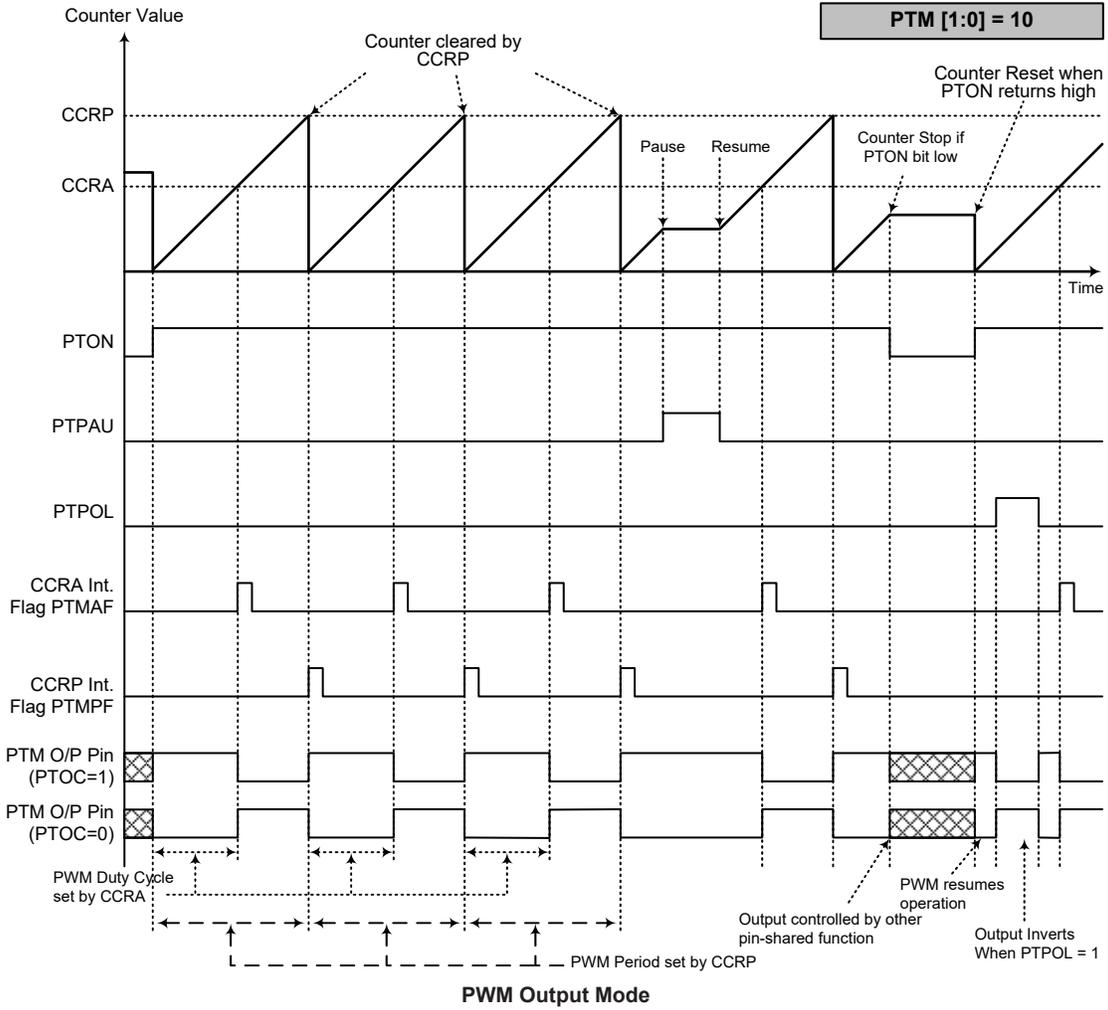
#### • 10-bit PTM, PWM Output Mode, Edge-aligned Mode

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

If  $f_{SYS}=8\text{MHz}$ , PTM clock source select  $f_{SYS}/4$ ,  $\text{CCRP}=512$  and  $\text{CCRA}=128$ ,

PTM 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%.



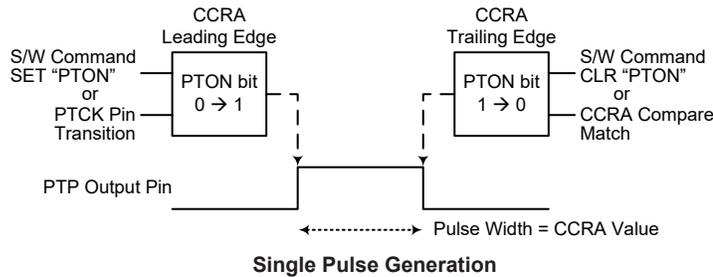
- Note: 1. The counter is cleared by CCRP  
 2. A counter clear sets the PWM Period  
 3. The internal PWM function continues running even when PTIO[1:0]=00 or 01  
 4. The PTCCLR bit has no influence on PWM operation

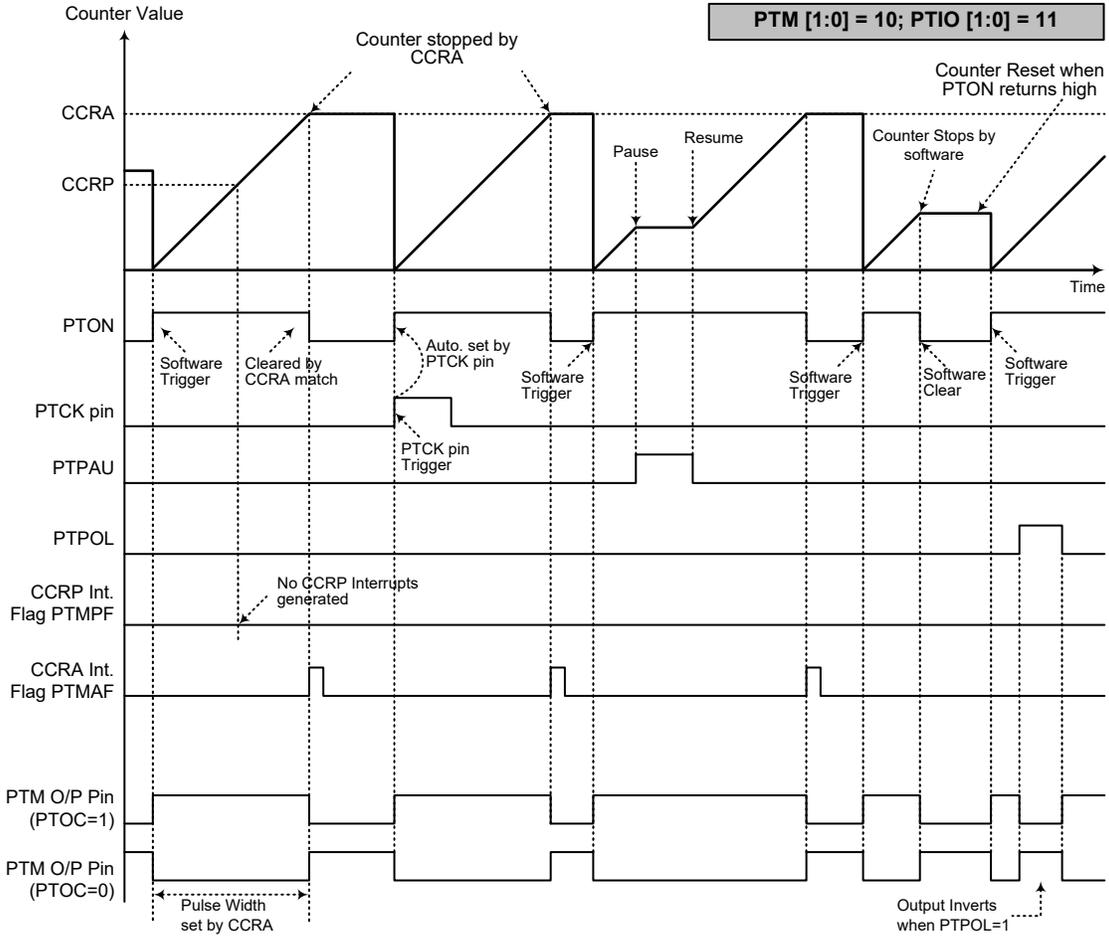
### Single Pulse Output Mode

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 is not used in this mode.





**Single Pulse Output Mode**

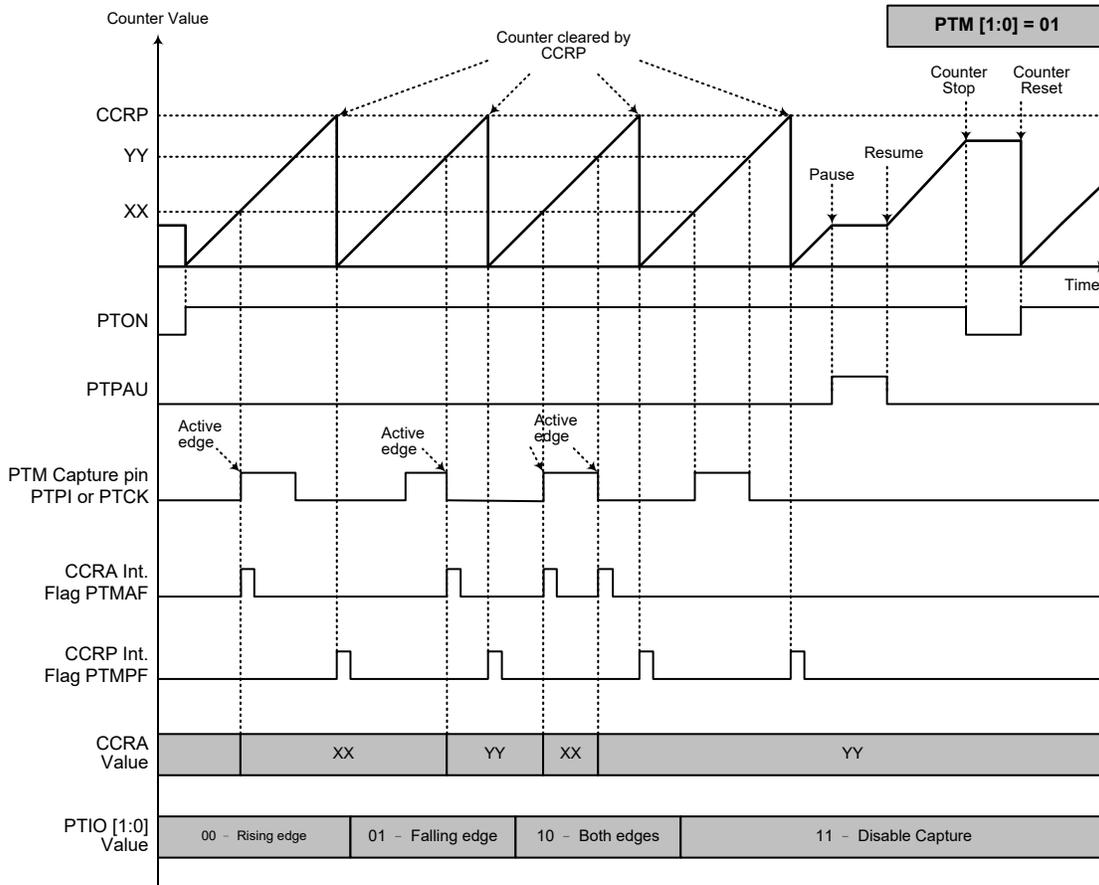
- Note:
1. Counter stopped by CCRA
  2. CCRP is not used
  3. The pulse triggered by the PTCK pin or by setting the PTON bit high
  4. A PTCK pin active edge will automatically set the PTON bit high
  5. In the Single Pulse Output Mode, PTIO[1:0] must be set to "11" and cannot be changed

### **Capture Input Mode**

To select this mode bits PTM1 and PTM0 in the PTMC1 register should be set to 01 respectively. This mode enables external signals to capture and store the present value of the internal counter and can therefore be used for applications such as pulse width measurements. The external signal is supplied on the PTPI or PTCK pin, selected by the PTCAPTS bit in the PTMC1 register. The input pin active edge can be either a rising edge, a falling edge or both rising and falling edges; the active edge transition type is selected using the PTIO1 and PTIO0 bits in the PTMC1 register. The counter is started when the PTON bit changes from low to high which is initiated using the application program.

When the required edge transition appears on the PTPI or PTCK pin the present value in the counter will be latched into the CCRA registers and a PTM interrupt generated. Irrespective of what events occur on the PTPI or PTCK pin the counter will continue to free run until the PTON bit changes from high to low. When a CCRP compare match occurs the counter will reset back to zero; in this way the CCRP value can be used to control the maximum counter value. When a CCRP compare match occurs from Comparator P, a PTM interrupt will also be generated. Counting the number of overflow interrupt signals from the CCRP can be a useful method in measuring long pulse widths. The PTIO1 and PTIO0 bits can select the active trigger edge on the PTPI or PTCK pin to be a rising edge, falling edge or both edge types. If the PTIO1 and PTIO0 bits are both set high, then no capture operation will take place irrespective of what happens on the PTPI or PTCK pin, however it must be noted that the counter will continue to run. The PTCCLR, PTOC and PTPOL bits are not used in this mode.

There are some considerations that should be noted. If PTCK is used as the capture input source, then it cannot be selected as the PTM clock source. If the captured pulse width is less than 2 timer clock periods, it may be ignored by hardware. After the counter value is latched to the CCRA registers by an active capture edge, the PTMAF flag will be set high after 0.5 timer clock periods. The delay time from the active capture edge received to the action of latching counter value to CCRA registers is less than 1.5 timer clock periods.



**Capture Input Mode**

- Note: 1. PTM[1:0]=01 and active edge set by the PTIO[1:0] bits  
 2. A PTM Capture input pin active edge transfers the counter value to CCRA  
 3. PTCCCLR bit not used  
 4. No output function – PTOC and PTPOL bits are not used  
 5. CCRP determines the counter value and the counter has a maximum count value when CCRP is equal to zero  
 6. The capture input mode cannot be used if the selected PTM counter clock is not available

## Analog to Digital Converter

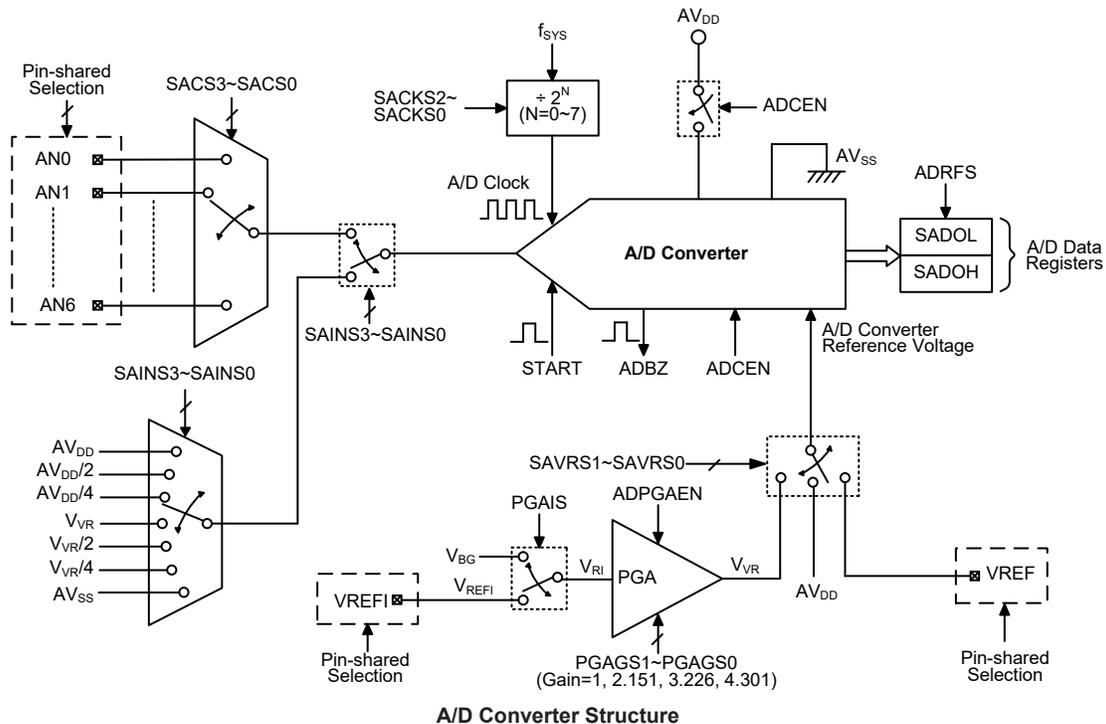
The need to interface to real world analog signals is a common requirement for many electronic systems. However, to properly process these signals by a microcontroller, they must first be converted into digital signals by A/D converters. By integrating the A/D conversion electronic circuitry into the 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, such as the internal A/D converter power supply and the internal PGA output voltage, and convert these signals directly into a 12-bit digital value. The external or internal analog signal to be converted is determined by the SAINS3~SAINS0 bits together with the SACS3~SACS0 bits. When the external analog signal is to be converted, the corresponding pin-shared control bits should first be properly configured and then desired external channel input should be selected using the SAINS3~SAINS0 and SACS3~SACS0 bits. Note that when the internal analog signal is to be converted using the SAINS3~SAINS0 bits, the external channel analog input will be automatically switched off. More detailed information about the A/D converter input signal is described in the “A/D Converter Control Registers” and “A/D Converter Input Signals” sections respectively.

External Input Channels	Internal Analog Signals	A/D Signal Select
AN0~AN6	$AV_{DD}$ , $AV_{DD}/2$ , $AV_{DD}/4$ , $V_{VR}$ , $V_{VR}/2$ , $V_{VR}/4$ , $AV_{SS}$	SAINS3~SAINS0, SACS3~SACS0

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 a series of registers. A read only register pair exists to store the A/D converter data 12-bit value. Three registers, SADC0, SADC1 and SADC2, are the control registers which setup the operating conditions and control function of the A/D converter. The VBGC register contains the VBGEN bit to control the bandgap reference voltage.

Register Name	Bit							
	7	6	5	4	3	2	1	0
SADOL (ADRFS=0)	D3	D2	D1	D0	—	—	—	—
SADOL (ADRFS=1)	D7	D6	D5	D4	D3	D2	D1	D0
SADOH (ADRFS=0)	D11	D10	D9	D8	D7	D6	D5	D4
SADOH (ADRFS=1)	—	—	—	—	D11	D10	D9	D8
SADC0	START	ADBZ	ADCEN	ADRFS	SACS3	SACS2	SACS1	SACS0
SADC1	SAINS3	SAINS2	SAINS1	SAINS0	—	SACKS2	SACKS1	SACKS0
SADC2	ADPGAEN	—	—	PGAIS	SAVRS1	SAVRS0	PGAGS1	PGAGS0
VBGC	—	—	—	—	VBGEN	—	—	—

**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 ADRFS 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 keep unchanged if the A/D converter is disabled.

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

**A/D Converter Data Registers**

### A/D Converter Control Registers – SADC0, SADC1, SADC2

To control the function and operation of the A/D converter, several control registers known as SADC0, SADC1, SADC2 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 converter clock source as well as controlling the start function and monitoring the A/D converter busy status. As the device contains only one actual analog to digital converter hardware circuit, each of the external and internal analog signals must be routed to the converter. The SACS3~SACS0 bits in the SADC0 register are used to determine which external channel input is selected to be converted. The SAINS3~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 A/D converter also contains a programmable gain amplifier, PGA, to generate the A/D converter internal reference voltage. The overall operation of the PGA is controlled using the SADC2 register.

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 converter input, its original function whether it is an I/O or other pin-shared function will be removed. In addition, any internal pull-high resistor connected to the pin will be automatically removed if the pin is selected to be an A/D converter input.

**• SADC0 Register**

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

- Bit 7**     **START:** Start the A/D conversion  
0→1→0: Start 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 high to indicate that the A/D conversion is initiated. The ADBZ flag will be cleared to zero 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 converter internal function. This bit should be set high to enable the A/D converter. If the bit is cleared to zero, then the A/D converter will be switched off reducing the device power consumption. When the A/D converter function is disabled, the contents of the A/D converter data register pair, SADOH and SADOL, will keep unchanged.
- Bit 4**     **ADRFS:** A/D Converter data format control  
0: ADC output data format → SADOH=D[11:4]; SADOL=D[3:0]  
1: ADC output data format → SADOH=D[11:8]; SADOL=D[7:0]  
This bit controls the format of the 12-bit converted A/D converter value in the two A/D converter 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 selection  
0000: AN0  
0001: AN1  
0010: AN2  
0011: AN3  
0100: AN4  
0101: AN5  
0110: AN6  
0111~1111: Non-existed channel, the input will be floating

**• SADC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	SAINS3	SAINS2	SAINS1	SAINS0	—	SACKS2	SACKS1	SACKS0
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 7~4**   **SAINS3~SAINS0:** A/D converter input signal selection  
0000: External signal – External analog channel input, ANn  
0001: Internal signal – Internal A/D converter power supply voltage AV<sub>DD</sub>

- 0010: Internal signal – Internal A/D converter power supply voltage  $AV_{DD}/2$
- 0011: Internal signal – Internal A/D converter power supply voltage  $AV_{DD}/4$
- 0100: External signal – External analog channel input, ANn
- 0101: Internal signal – Internal signal derived from PGA output  $V_{VR}$
- 0110: Internal signal – Internal signal derived from PGA output  $V_{VR}/2$
- 0111: Internal signal – Internal signal derived from PGA output  $V_{VR}/4$
- 10xx: Internal signal – Connected to ground
- 1100~1111: External signal – External analog channel input, ANn

Care must be taken if the SAINS3~SAINS0 bits are set to “0001”~“0011”, “0101”~“10xx” to select the internal analog signal to be converted. When the internal analog signal is selected to be converted, the external channel input signal will automatically be switched off regardless of the SACS3~SACS0 bits value. It will prevent the external channel input from being connected together with the internal analog signal.

- Bit 3 Unimplemented, read as “0”
- Bit 2~0 **SACKS2~SACKS0**: A/D conversion clock source selection
  - 000:  $f_{SYS}$
  - 001:  $f_{SYS}/2$
  - 010:  $f_{SYS}/4$
  - 011:  $f_{SYS}/8$
  - 100:  $f_{SYS}/16$
  - 101:  $f_{SYS}/32$
  - 110:  $f_{SYS}/64$
  - 111:  $f_{SYS}/128$

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

• **SADC2 Register**

Bit	7	6	5	4	3	2	1	0
Name	ADPGAEN	—	—	PGAIS	SAVRS1	SAVRS0	PGAGS1	PGAGS0
R/W	R/W	—	—	R/W	R/W	R/W	R/W	R/W
POR	0	—	—	0	0	0	0	0

- Bit 7 **ADPGAEN**: A/D converter PGA enable/disable control
  - 0: Disable
  - 1: Enable

This bit is used to control the A/D converter internal PGA function. When the PGA output voltage is selected as A/D input or A/D reference voltage, the PGA needs to be enabled by setting this bit high. Otherwise the PGA needs to be disabled by clearing the ADPGAEN bit to zero to conserve power.

- Bit 6~5 Unimplemented, read as “0”
- Bit 4 **PGAIS**: PGA input voltage ( $V_{RI}$ ) selection
  - 0: From VREFI pin
  - 1: From internal reference voltage  $V_{BG}$

This bit is used to select the PGA input voltage source. When the internal reference voltage  $V_{BG}$  is selected as the PGA input voltage, the external reference voltage on the VREFI pin will be automatically switched off. When this bit is set high to select  $V_{BG}$  as PGA input, the internal bandgap reference  $V_{BG}$  should be enabled by setting the VBGEN bit in the VBG register to “1”.

- Bit 3~2 **SAVRS1~SAVRS0**: A/D converter reference voltage selection
  - 00: Internal A/D converter power,  $AV_{DD}$
  - 01: External VREF pin
  - 1x: Internal PGA output voltage,  $V_{VR}$

These bits are used to select the A/D converter reference voltage source. When the internal A/D converter power supply or PGA output voltage is set as the reference voltage, the reference voltage derived from the external VREF pin will be automatically switched off.

Bit 1~0 **PGAGS1~PGAGS0**: PGA gain select

00: Gain=1

01: Gain=2.151 –  $V_{VR}=2V$  as  $V_{RI}=0.93V$

10: Gain=3.226 –  $V_{VR}=3V$  as  $V_{RI}=0.93V$

11: Gain=4.301 –  $V_{VR}=4V$  as  $V_{RI}=0.93V$

These bits are used to select the PGA gain. Note that here the gain is guaranteed only when the PGA input voltage is equal to 0.93V.

### Bandgap Referenc Voltage Control Register – VBGC

A high performance bandgap voltage reference is included in the device. It has an accurate voltage reference output,  $V_{BG}$ , when input supply voltage changes or temperature variates. The VBGC register is used to control the bandgap reference voltage circuit enable or disable.

#### • VBGC Register

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

Bit 7~4 Unimplemented, read as “0”

Bit 3 **VBGEN**: Bandgap reference voltage control

0: Disable

1: Enable

This bit is used to enable/disable the internal Bandgap reference circuit. The internal Bandgap reference circuit should first be enabled before the  $V_{BG}$  voltage is selected to be used. A specific start-up time is necessary for the Bandgap circuit to become stable and accurate. When this bit is cleared to 0, the Bandgap voltage output  $V_{BG}$  is in a low state.

Bit 2~0 Unimplemented, read as “0”

### A/D Converter Reference Voltage

The actual reference voltage supply to the A/D Converter can be supplied from the internal A/D converter power,  $AV_{DD}$ , an external reference source supplied on pin VREF or the internal reference voltage  $V_{VR}$  determined by the SAVRS1~SAVRS0 bits in the SADC2 register. The internal reference voltage  $V_{VR}$  is derived from a programmable gain amplifier, PGA, which is controlled by the ADPGAEN bit in the SADC2 register. The PGA gain can be equal to 1, 2.151, 3.226 or 4.301 and selected using the PGAGS1~PGAGS0 bits in the SADC2 register. The PGA input can come from the external reference input pin, VREFI, or an internal Bandgap reference voltage,  $V_{BG}$ , selected by the PGAIS bit in the SADC2 register. Note that the internal Bandgap reference circuit should first be enabled before the  $V_{BG}$  is selected to be used.

As the VREFI and VREF pin both are pin-shared with other functions, when the VREFI or VREF pin is selected as the reference voltage pin, the VREFI or VREF pin-shared function selection bits should first be properly configured to disable other pin-shared functions. However, if the internal reference signal is selected as the reference source, the external reference voltage input from the VREF or VREFI pin will automatically be switched off by hardware.

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

SAVRS[1:0]	Reference	Description
00	AV <sub>DD</sub>	Internal A/D converter power supply voltage AV <sub>DD</sub>
01	VREF pin	External A/D converter reference pin VREF
10 or 11	V <sub>VR</sub>	Internal A/D converter PGA output voltage

#### A/D Converter Reference Voltage Selection

### A/D Converter Input Signals

All of the external A/D analog input pins are pin-shared with the I/O pins as well as other functions. The corresponding pin-shared function selection bits in the PxS1 and PxS0 registers, determine whether the external input pins are set as A/D converter analog channel inputs or whether they have other functions. If the corresponding pin is setup to be 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 setup through register programming, will be automatically disconnected if the pins are setup as A/D converter inputs. Note that it is not necessary to first setup the A/D pin as an input in the port control register to enable the A/D converter input as when the relevant A/D converter input function selection bits enable an A/D converter input, the status of the port control register will be overridden. If the SAINS3~SAINS0 bits are set to “0000”, “0100” or “1100~1111”, 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 SAINS3~SAINS0 bits are set to other values, the internal analog signal will be selected. If the internal analog signal is selected to be converted, the external input channel will automatically be switched off regardless of the SACS3~SACS0 bits value. It will prevent the external channel input from being connected together with the internal analog signal.

SAINS[3:0]	SACS[3:0]	Input Signals	Description
0000, 0100, 1100~1111	0000~0110	AN0~AN6	External channel analog input ANn
	0111~1111	—	Floating, no external channel is selected
0001	xxxx	AV <sub>DD</sub>	Internal A/D converter power supply voltage AV <sub>DD</sub>
0010	xxxx	AV <sub>DD</sub> /2	Internal A/D converter power supply voltage AV <sub>DD</sub> /2
0011	xxxx	AV <sub>DD</sub> /4	Internal A/D converter power supply voltage AV <sub>DD</sub> /4
0101	xxxx	V <sub>VR</sub>	Internal A/D converter PGA output V <sub>VR</sub>
0110	xxxx	V <sub>VR</sub> /2	Internal A/D converter PGA output V <sub>VR</sub> /2
0111	xxxx	V <sub>VR</sub> /4	Internal A/D converter PGA output V <sub>VR</sub> /4
10xx	xxxx	AV <sub>SS</sub>	Connected to the ground

“x”: Don't care

#### 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 process 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 converter interrupt request flag will be set in the interrupt control register, and if the interrupts are enabled, an appropriate internal interrupt signal will be generated. This A/D converter internal interrupt signal will direct the program flow to the associated A/D converter internal interrupt address for processing. If the A/D converter internal

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

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

#### A/D Conversion Clock Period Examples

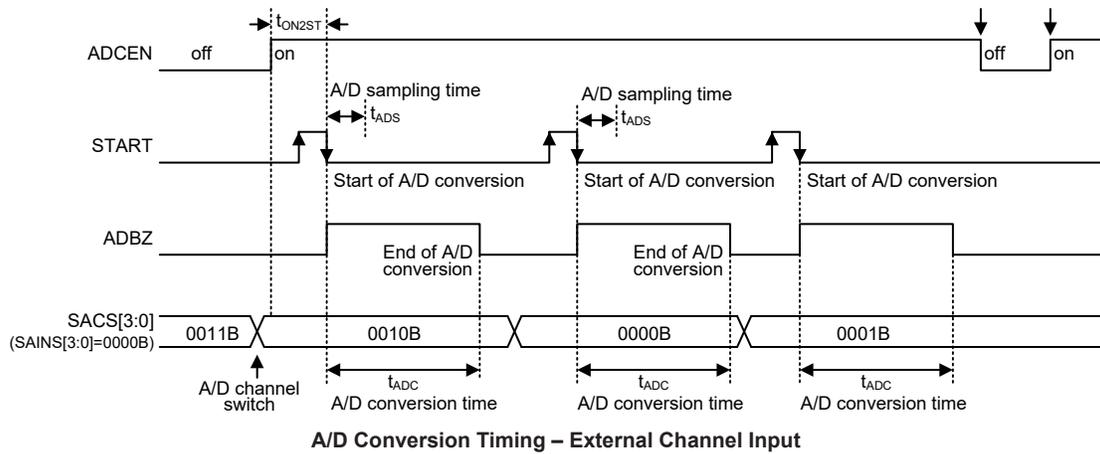
Controlling the power on/off function of the A/D conversion circuitry is implemented using the ADCEN bit in the SADC0 register. This bit must be set high to power on the A/D converter. When the ADCEN bit is set high to power on the A/D conversion internal circuitry, a certain delay as indicated in the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

#### A/D Conversion Rate and Timing Diagram

A complete A/D conversion contains two parts, data sampling and data conversion. The data sampling which is defined as  $t_{ADS}$  takes 4 A/D clock periods and the data conversion takes 12 A/D converter clock periods. Therefore a total of 16 A/D conversion clock periods for an A/D conversion which is defined as  $t_{ADC}$  are necessary.

$$\text{Maximum single A/D conversion rate} = 1/(\text{A/D conversion clock period} \times 16)$$

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



### Summary of A/D Conversion Steps

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

- Step 1  
 Select the required A/D conversion clock by properly programming the SACKS2~SACKS0 bits 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 SAINS3~SAINS0 bits.  
 Select the external channel input to be converted, go to Step 4.  
 Select the internal analog signal to be converted, go to Step 5.
- Step 4  
 If the A/D input signal comes from the external channel input selected by configuring the SAINS3~SAINS0 bits, the corresponding pin should be configured as an A/D input function by configuring the relevant pin-shared function control bits. The desired external channel then should be selected by configuring the SACS3~SACS0 bits. After this step, go to Step 6.
- Step 5  
 If the A/D input signal is selected to come from the internal analog signal by configuring the SAINS3~SAINS0 bits and the external channel analog signal input will be automatically switched off regardless of the SACS3~SACS0 bits value. After this step, go to Step 6.
- Step 6  
 Select the reference voltage source by configuring the SAVRS1~SAVRS0 bits in the SADC2 register. Select the PGA input signal and the desired PGA gain if the PGA output voltage,  $V_{VR}$ , is selected as the A/D converter reference voltage.
- Step 7  
 Select A/D converter output data format by configuring the ADRFS bit in the SADC0 register.
- Step 8  
 If A/D converter interrupt is used, the interrupt control registers must be correctly configured to

ensure the A/D converter interrupt function is active. The master interrupt control bit, EMI, and the A/D converter interrupt bit, ADE, must both set high in advance.

- Step 9

The A/D conversion procedure can now be initialised 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 completed, the ADBZ flag will go low and then output data can be read from the 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 conversion internal circuitry can be switched off to reduce power consumption by setting the ADCEN bit low in the SADC0 register. When this happens, the internal A/D conversion circuits will not consume power irrespective of what analog voltage is applied to their input lines. If the A/D converter input lines are used as normal I/Os, then care must be taken as if the input voltage is not at a valid logic level, then this may lead to some increase in power consumption.

### A/D Conversion Function

As the device contains 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,  $V_{REF}$ , this gives a single bit analog input value of  $V_{REF}$  divided by 4096.

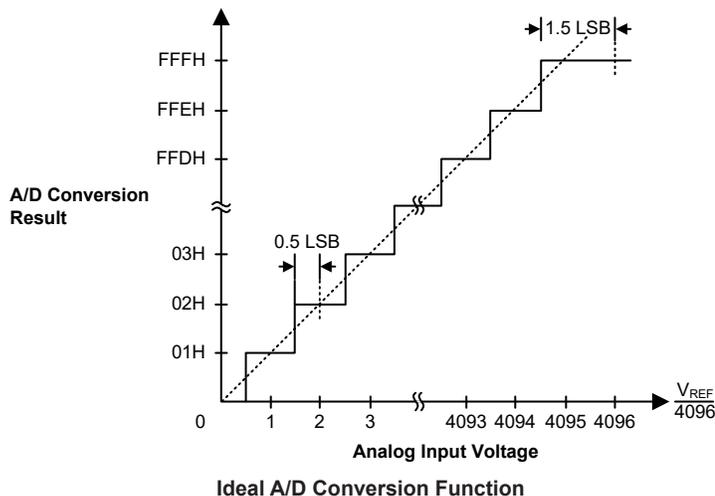
$$1 \text{ LSB} = V_{REF} \div 4096$$

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

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

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  $V_{REF}$  level.

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



## A/D Converter 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 converter interrupt is used to determine when the conversion is complete.

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

```

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

mov a,00H
mov SADC2,a           ; select reference voltage from AVDD
mov a,03h             ; setup PAS0 to configure pin AN0
mov PAS0,a
mov a,20h
mov SADC0,a           ; enable A/D converter and connect AN0 channel to A/D
                    ; converter

:
start_conversion:
clr START             ; high pulse on start bit to initiate conversion
set START             ; reset A/D converter
clr START             ; start A/D conversion
polling_EOC:
sz ADBZ               ; poll the SADC0 register ADBZ bit to detect end of A/D
                    ; conversion

jmp polling_EOC       ; continue polling
mov a,SADOL            ; read low byte conversion result value
mov SADOL_buffer,a    ; save result to user defined register
mov a,SADOH            ; read high byte conversion result value
mov SADOH_buffer,a    ; save result to user defined register
:
jmp start_conversion  ; start next A/D conversion

```

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

```

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

mov a,00H
mov SADC2,a           ; select reference voltage from AVDD
mov a,03h             ; setup PAS0 to configure pin AN0
mov PAS0,a
mov a,20h
mov SADC0,a           ; enable A/D converter and connect AN0 channel to A/D
                    ; converter

Start_conversion:
clr START             ; high pulse on START bit to initiate conversion
set START             ; reset A/D converter
clr START             ; start A/D conversion
clr ADF               ; clear ADC interrupt request flag
set ADE               ; enable ADC interrupt
set EMI               ; enable global interrupt
:
:
; ADC interrupt service routine

```

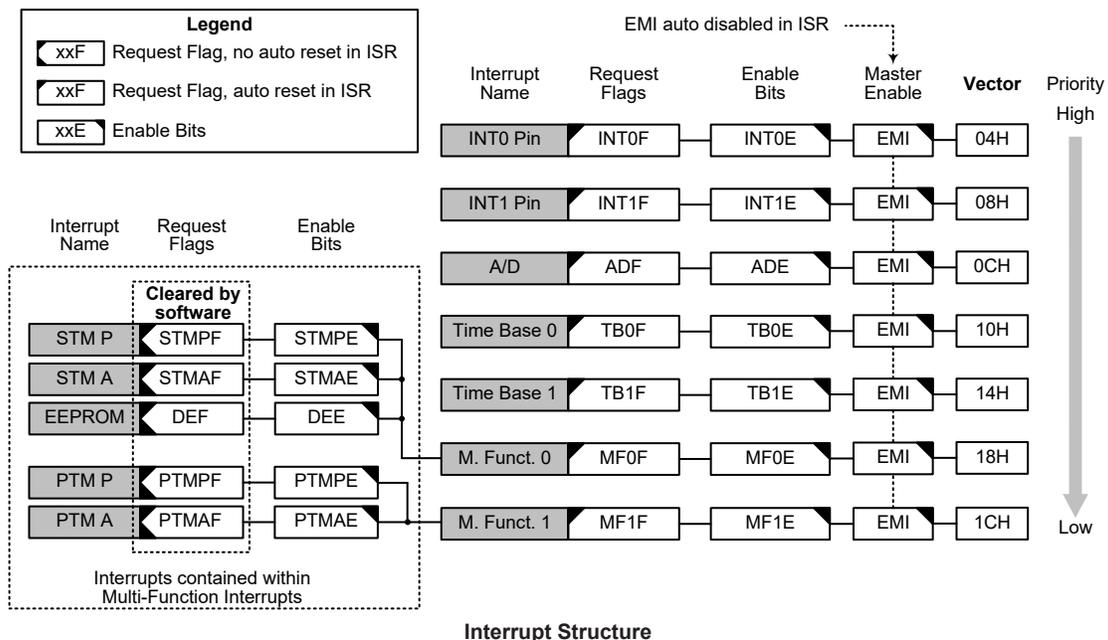
```

ADC_ISR:
mov  acc_stack,a      ; save ACC to user defined memory
mov  a,STATUS
mov  status_stack,a   ; save STATUS to user defined memory
:
:
mov  a,SADOL          ; read low byte conversion result value
mov  SADOL_buffer,a   ; save result to user defined register
mov  a,SADOH          ; read high byte conversion result value
mov  SADOH_buffer,a   ; save result to user defined register
:
:
EXIT_INT_ISR:
mov  a,status_stack
mov  STATUS,a        ; restore STATUS from user defined memory
mov  a,acc_stack     ; restore ACC from user defined memory
reti
    
```

## Interrupts

Interrupts are an important part of any microcontroller system. When an external event or an internal function such as a Timer Module or an A/D converter requires microcontroller attention, their corresponding interrupt will enforce a temporary suspension of the main program allowing the microcontroller to direct attention to their respective needs. The device contains several external interrupt and internal interrupts functions. The external interrupts are generated by the action of the external INT0~INT1 pins, while the internal interrupts are generated by various internal functions such as the TMs, Time Base, EEPROM and the A/D converter, etc.

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.



## Interrupt Registers

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

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

Function	Enable Bit	Request Flag	Notes
Global	EMI	—	—
INTn Pins	INTnE	INTnF	n=0~1
A/D Converter	ADE	ADF	—
Multi-function	MFnE	MFnF	n=0~1
Time Base	TBnE	TBnF	n=0~1
EEPROM	DEE	DEF	—
STM	STMPE	STMPF	—
	STMAE	STMAF	
PTM	PTMPE	PTMPF	—
	PTMAE	PTMAF	

Interrupt Register Bit Naming Conventions

Register Name	Bit							
	7	6	5	4	3	2	1	0
INTEG	—	—	—	—	INT1S1	INT1S0	INT0S1	INT0S0
INTC0	—	ADF	INT1F	INT0F	ADE	INT1E	INT0E	EMI
INTC1	MF1F	MF0F	TB1F	TB0F	MF1E	MF0E	TB1E	TB0E
MFI0	—	DEF	STMAF	STMPF	—	DEE	STMAE	STMPE
MFI1	—	—	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**: Interrupt trigger edge selection for INT1 pin  
 00: Disable  
 01: Rising edge  
 10: Falling edge  
 11: Rising and falling edges

Bit 1~0 **INT0S1~INT0S0**: Interrupt trigger edge selection for INT0 pin  
 00: Disable  
 01: Rising edge  
 10: Falling edge  
 11: Rising and falling edges

**• INTC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	ADF	INT1F	INT0F	ADE	INT1E	INT0E	EMI
R/W	—	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	0	0	0	0	0	0	0

- Bit 7 Unimplemented, read as “0”
- Bit 6 **ADF**: A/D Converter interrupt request flag  
 0: No request  
 1: Interrupt request
- Bit 5 **INT1F**: INT1 interrupt request flag  
 0: No request  
 1: Interrupt request
- Bit 4 **INT0F**: INT0 interrupt request flag  
 0: No request  
 1: Interrupt request
- Bit 3 **ADE**: A/D Converter interrupt control  
 0: Disable  
 1: Enable
- Bit 2 **INT1E**: INT1 interrupt control  
 0: Disable  
 1: Enable
- Bit 1 **INT0E**: INT0 interrupt control  
 0: Disable  
 1: Enable
- Bit 0 **EMI**: Global interrupt control  
 0: Disable  
 1: Enable

**• INTC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	MF1F	MF0F	TB1F	TB0F	MF1E	MF0E	TB1E	TB0E
R/W								
POR	0	0	0	0	0	0	0	0

- Bit 7 **MF1F**: Multi-function 1 interrupt request flag  
 0: No request  
 1: Interrupt request
- Bit 6 **MF0F**: Multi-function 0 interrupt request flag  
 0: No request  
 1: Interrupt request
- Bit 5 **TB1F**: Time Base 1 interrupt request flag  
 0: No request  
 1: Interrupt request
- Bit 4 **TB0F**: Time Base 0 interrupt request flag  
 0: No request  
 1: Interrupt request
- Bit 3 **MF1E**: Multi-function 1 interrupt control  
 0: Disable  
 1: Enable
- Bit 2 **MF0E**: Multi-function 0 interrupt control  
 0: Disable  
 1: Enable

- Bit 1      **TB1E**: Time Base 1 interrupt control  
             0: Disable  
             1: Enable
- Bit 0      **TB0E**: Time Base 0 interrupt control  
             0: Disable  
             1: Enable

• **MF10 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	DEF	STMAF	STMPPF	—	DEE	STMAE	STMPE
R/W	—	R/W	R/W	R/W	—	R/W	R/W	R/W
POR	—	0	0	0	—	0	0	0

- Bit 7      Unimplemented, read as “0”
- Bit 6      **DEF**: Data EEPROM interrupt request flag  
             0: No request  
             1: Interrupt request  
             Note that this bit must be cleared to zero by the application program when the interrupt is serviced.
- Bit 5      **STMAF**: STM Comparator A match interrupt request flag  
             0: No request  
             1: Interrupt request  
             Note that this bit must be cleared to zero by the application program when the interrupt is serviced.
- Bit 4      **STMPPF**: STM Comparator P match interrupt request flag  
             0: No request  
             1: Interrupt request  
             Note that this bit must be cleared to zero by the application program when the interrupt is serviced.
- Bit 3      Unimplemented, read as “0”
- Bit 2      **DEE**: Data EEPROM 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

• **MF11 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  
             Note that this bit must be cleared to zero by the application program when the interrupt is serviced.
- Bit 4      **PTMPF**: PTM Comparator P match interrupt request flag  
             0: No request  
             1: Interrupt request

Note that this bit must be cleared to zero by the application program when the interrupt is serviced.

Bit 3~2	Unimplemented, read as “0”
Bit 1	<b>PTMAE</b> : PTM Comparator A match interrupt control 0: Disable 1: Enable
Bit 0	<b>PTMPE</b> : 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.

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 Interrupt Structure 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 their respective interrupt vector addresses, the global interrupt enable bit, EMI, and respective external interrupt enable bit, INT0E~INT1E, must first be set. Additionally the correct interrupt edge type must be selected using the INTEG register to enable the external interrupt function and to choose the trigger edge type. As the external interrupt pins are pin-shared with I/O pins, they can only be configured as external interrupt pins if

their external interrupt enable bit in the corresponding interrupt register has been set and the external interrupt pin is selected by the corresponding pin-shared function selection bits. The pin must also be setup as an input by setting the corresponding bit in the port control register. When the interrupt is enabled, the stack is not full and the correct transition type appears on the external interrupt pin, a subroutine call to the external interrupt vector, will take place. When the interrupt is serviced, the external interrupt request flags, INT0F~INT1F, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts. Note that any pull-high resistor selections on the external interrupt pins will remain valid even if the pin is used as an external interrupt input.

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

### **A/D Converter Interrupt**

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

### **Multi-function Interrupts**

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

A Multi-function interrupt request will take place when any of the Multi-function interrupt request flags MFnF is 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 their respective interrupt vector addresses, when the Multi-function interrupt is enabled and the stack is not full, and one of the interrupts contained within each of Multi-function interrupt occurs, a subroutine call to the relevant Multi-function interrupt vector will take place. When the interrupt is serviced, the related Multi-Function request flag will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts.

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

### **Timer Module Interrupts**

The Standard and Periodic type TMs each has two interrupts, one comes from the comparator A match situation and the other comes from the comparator P match situation. All of the TM interrupts are contained within the Multi-function Interrupts. For all of the TM Types there are two interrupt request flags and two interrupt enable bits. A TM interrupt request will take place when any of the TM request flags are set, a situation which occurs when a TM comparator P or A match situation happens.

To allow the program to branch to their respective interrupt vector addresses, the global interrupt enable bit, EMI, respective TM Interrupt enable bit, and relevant Multi-function Interrupt enable bit, MFnE, must first be set. When the interrupt is enabled, the stack is not full and a TM comparator

match situation occurs, a subroutine call to the relevant Multi-function Interrupt vector location, 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.

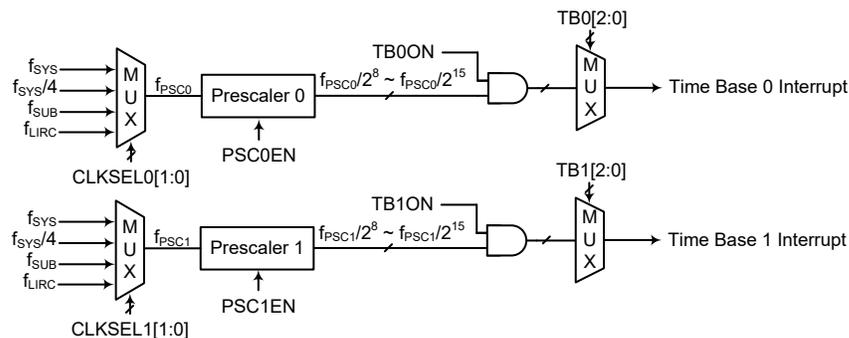
### EEPROM Interrupt

The EEPROM Interrupt is contained within the Multi-function Interrupt. An EEPROM Interrupt request will take place when the EEPROM Interrupt request flag, DEF, is set, which occurs when an EEPROM erase or write cycle ends. To allow the program to branch to its interrupt vector address, the global interrupt enable bit, EMI, EEPROM Interrupt enable bit, DEE, and associated Multi-function interrupt enable bit must first be set. When the interrupt is enabled, the stack is not full and an EEPROM erase or write cycle ends, a subroutine call to the Multi-function Interrupt vector will take place. When the EEPROM Interrupt is serviced, the EMI bit will be automatically cleared to disable other interrupts. However, only the Multi-function interrupt request flag will be automatically cleared. As the DEF flag will not be automatically cleared, it has to be cleared by the application program.

### Time Base Interrupts

The function of the Time Base Interrupts is to provide regular time signals in the form of an internal interrupt. They are controlled by the overflow signals from their respective timer functions. When these happens their respective interrupt request flags, TBOF 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, TBOE 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, TBOF or TB1F, will be automatically cleared, the EMI bit will also be automatically cleared to disable other interrupts.

The purpose of the Time Base Interrupts is to provide an interrupt signal at fixed time periods. Their clock sources,  $f_{PSC0}$  or  $f_{PSC1}$ , originate from the internal clock source  $f_{SYS}$ ,  $f_{SYS}/4$ ,  $f_{SUB}$  or  $f_{LIRC}$  and then pass through a divider, the division ratio of which is selected by programming the appropriate bits in the TB0C and TB1C registers to obtain longer interrupt periods whose value ranges. The clock source that generate  $f_{PSC0}$  or  $f_{PSC1}$ , which in turn controls the Time Base interrupt period, is selected using the CLKSEL0[1:0] and CLKSEL1[1:0] bits in the PSC0R and PSC1R register respectively.



**Time Base Interrupts**

• **PSC0R Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	PSC0EN	CLKSEL01	CLKSEL00
R/W	—	—	—	—	—	R/W	R/W	R/W
POR	—	—	—	—	—	0	0	0

Bit 7~3 Unimplemented, read as “0”

Bit 2 **PSC0EN**: Prescaler 0 clock enable control  
0: Disable  
1: Enable

The PSC0EN bit is the Prescaler 0 clock enable or disable control bit. When the Prescale 0 clock is disabled, it can reduce extra power consumption.

Bit 1~0 **CLKSEL01~CLKSEL00**: Prescaler 0 clock source selection  
00:  $f_{SYS}$   
01:  $f_{SYS}/4$   
10:  $f_{SUB}$   
11:  $f_{LIRC}$

• **PSC1R Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	PSC1EN	CLKSEL11	CLKSEL10
R/W	—	—	—	—	—	R/W	R/W	R/W
POR	—	—	—	—	—	0	0	0

Bit 7~3 Unimplemented, read as “0”

Bit 2 **PSC1EN**: Prescaler 1 clock enable control  
0: Disable  
1: Enable

The PSC1EN bit is the Prescaler 1 clock enable or disable control bit. When the Prescale 1 clock is disabled, it can reduce extra power consumption.

Bit 1~0 **CLKSEL11~CLKSEL10**: Prescaler 1 clock source selection  
00:  $f_{SYS}$   
01:  $f_{SYS}/4$   
10:  $f_{SUB}$   
11:  $f_{LIRC}$

• **TB0C Register**

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

Bit 7 **TB0ON**: Time Base 0 Control  
0: Disable  
1: Enable

Bit 6~3 Unimplemented, read as “0”

Bit 2~0 **TB02~TB00**: Select Time Base 0 Time-out Period  
000:  $2^8/f_{PSC0}$   
001:  $2^9/f_{PSC0}$   
010:  $2^{10}/f_{PSC0}$   
011:  $2^{11}/f_{PSC0}$   
100:  $2^{12}/f_{PSC0}$   
101:  $2^{13}/f_{PSC0}$   
110:  $2^{14}/f_{PSC0}$   
111:  $2^{15}/f_{PSC0}$

**• TB1C Register**

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

Bit 7 **TB1ON**: Time Base 1 Control

0: Disable

1: Enable

Bit 6~3 Unimplemented, read as “0”

Bit 2~0 **TB12~TB10**: Select Time Base 1 Time-out Period

000:  $2^8/f_{PSC1}$

001:  $2^9/f_{PSC1}$

010:  $2^{10}/f_{PSC1}$

011:  $2^{11}/f_{PSC1}$

100:  $2^{12}/f_{PSC1}$

101:  $2^{13}/f_{PSC1}$

110:  $2^{14}/f_{PSC1}$

111:  $2^{15}/f_{PSC1}$

**Interrupt Wake-up Function**

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

**Programming Considerations**

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

Where a certain interrupt is contained within a Multi-function interrupt, then when the interrupt service routine is executed, as only the Multi-function interrupt request flags, MF<sub>n</sub>F, will be automatically cleared, the individual request flag for the function needs to be cleared by the application program.

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

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

As only the Program Counter is pushed onto the stack, then when the interrupt is serviced, if the contents of the accumulator, status register or other registers are altered by the interrupt service

program, their contents should be saved to the memory at the beginning of the interrupt service routine.

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

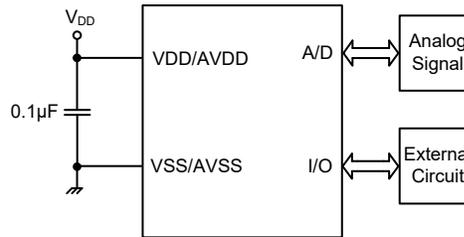
## Configuration Options

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

No.	Options
<b>Oscillator Options</b>	
1	HIRC Frequency Selection – $f_{HIRC}$ : 8MHz, 12MHz or 16MHz

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

## Application Circuits



## Instruction Set

### Introduction

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

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

### Instruction Timing

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

### Moving and Transferring Data

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

### Arithmetic Operations

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

## Logical and Rotate Operation

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

## Branches and Control Transfer

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

## Bit Operations

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

## Table Read Operations

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

## Other Operations

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

## Instruction Set Summary

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

### Table Conventions

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

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

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

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

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

## Instruction Definition

<b>ADC A,[m]</b>	Add Data Memory to ACC with Carry
Description	The contents of the specified Data Memory, Accumulator and the carry flag are added. The result is stored in the Accumulator.
Operation	$ACC \leftarrow ACC + [m] + C$
Affected flag(s)	OV, Z, AC, C
<b>ADCM A,[m]</b>	Add ACC to Data Memory with Carry
Description	The contents of the specified Data Memory, Accumulator and the carry flag are added. The result is stored in the specified Data Memory.
Operation	$[m] \leftarrow ACC + [m] + C$
Affected flag(s)	OV, Z, AC, C
<b>ADD A,[m]</b>	Add Data Memory to ACC
Description	The contents of the specified Data Memory and the Accumulator are added. The result is stored in the Accumulator.
Operation	$ACC \leftarrow ACC + [m]$
Affected flag(s)	OV, Z, AC, C
<b>ADD A,x</b>	Add immediate data to ACC
Description	The contents of the Accumulator and the specified immediate data are added. The result is stored in the Accumulator.
Operation	$ACC \leftarrow ACC + x$
Affected flag(s)	OV, Z, AC, C
<b>ADDM A,[m]</b>	Add ACC to Data Memory
Description	The contents of the specified Data Memory and the Accumulator are added. The result is stored in the specified Data Memory.
Operation	$[m] \leftarrow ACC + [m]$
Affected flag(s)	OV, Z, AC, C
<b>AND A,[m]</b>	Logical AND Data Memory to ACC
Description	Data in the Accumulator and the specified Data Memory perform a bitwise logical AND operation. The result is stored in the Accumulator.
Operation	$ACC \leftarrow ACC \text{ "AND" } [m]$
Affected flag(s)	Z
<b>AND A,x</b>	Logical AND immediate data to ACC
Description	Data in the Accumulator and the specified immediate data perform a bit wise logical AND operation. The result is stored in the Accumulator.
Operation	$ACC \leftarrow ACC \text{ "AND" } x$
Affected flag(s)	Z

<b>ANDM A,[m]</b>	Logical AND ACC to Data Memory
Description	Data in the specified Data Memory and the Accumulator perform a bitwise logical AND operation. The result is stored in the Data Memory.
Operation	$[m] \leftarrow \text{ACC} \text{ "AND" } [m]$
Affected flag(s)	Z
<b>CALL addr</b>	Subroutine call
Description	Unconditionally calls a subroutine at the specified address. The Program Counter then increments by 1 to obtain the address of the next instruction which is then pushed onto the stack. The specified address is then loaded and the program continues execution from this new address. As this instruction requires an additional operation, it is a two cycle instruction.
Operation	Stack $\leftarrow$ Program Counter + 1 Program Counter $\leftarrow$ addr
Affected flag(s)	None
<b>CLR [m]</b>	Clear Data Memory
Description	Each bit of the specified Data Memory is cleared to 0.
Operation	$[m] \leftarrow 00\text{H}$
Affected flag(s)	None
<b>CLR [m].i</b>	Clear bit of Data Memory
Description	Bit i of the specified Data Memory is cleared to 0.
Operation	$[m].i \leftarrow 0$
Affected flag(s)	None
<b>CLR WDT</b>	Clear Watchdog Timer
Description	The TO, PDF flags and the WDT are all cleared.
Operation	WDT cleared TO $\leftarrow$ 0 PDF $\leftarrow$ 0
Affected flag(s)	TO, PDF
<b>CPL [m]</b>	Complement Data Memory
Description	Each bit of the specified Data Memory is logically complemented (1's complement). Bits which previously contained a 1 are changed to 0 and vice versa.
Operation	$[m] \leftarrow \overline{[m]}$
Affected flag(s)	Z
<b>CPLA [m]</b>	Complement Data Memory with result in ACC
Description	Each bit of the specified Data Memory is logically complemented (1's complement). Bits which previously contained a 1 are changed to 0 and vice versa. The complemented result is stored in the Accumulator and the contents of the Data Memory remain unchanged.
Operation	ACC $\leftarrow \overline{[m]}$
Affected flag(s)	Z

<b>DAA [m]</b>	Decimal-Adjust ACC for addition with result in Data Memory
Description	Convert the contents of the Accumulator value to a BCD (Binary Coded Decimal) value resulting from the previous addition of two BCD variables. If the low nibble is greater than 9 or if AC flag is set, then a value of 6 will be added to the low nibble. Otherwise the low nibble remains unchanged. If the high nibble is greater than 9 or if the C flag is set, then a value of 6 will be added to the high nibble. Essentially, the decimal conversion is performed by adding 00H, 06H, 60H or 66H depending on the Accumulator and flag conditions. Only the C flag may be affected by this instruction which indicates that if the original BCD sum is greater than 100, it allows multiple precision decimal addition.
Operation	[m] ← ACC + 00H or [m] ← ACC + 06H or [m] ← ACC + 60H or [m] ← ACC + 66H
Affected flag(s)	C
<b>DEC [m]</b>	Decrement Data Memory
Description	Data in the specified Data Memory is decremented by 1.
Operation	[m] ← [m] - 1
Affected flag(s)	Z
<b>DECA [m]</b>	Decrement Data Memory with result in ACC
Description	Data in the specified Data Memory is decremented by 1. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.
Operation	ACC ← [m] - 1
Affected flag(s)	Z
<b>HALT</b>	Enter power down mode
Description	This instruction stops the program execution and turns off the system clock. The contents of the Data Memory and registers are retained. The WDT and prescaler are cleared. The power down flag PDF is set and the WDT time-out flag TO is cleared.
Operation	TO ← 0 PDF ← 1
Affected flag(s)	TO, PDF
<b>INC [m]</b>	Increment Data Memory
Description	Data in the specified Data Memory is incremented by 1.
Operation	[m] ← [m] + 1
Affected flag(s)	Z
<b>INCA [m]</b>	Increment Data Memory with result in ACC
Description	Data in the specified Data Memory is incremented by 1. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.
Operation	ACC ← [m] + 1
Affected flag(s)	Z

<b>JMP addr</b>	Jump unconditionally
Description	The contents of the Program Counter are replaced with the specified address. Program execution then continues from this new address. As this requires the insertion of a dummy instruction while the new address is loaded, it is a two cycle instruction.
Operation	Program Counter $\leftarrow$ addr
Affected flag(s)	None
<b>MOV A,[m]</b>	Move Data Memory to ACC
Description	The contents of the specified Data Memory are copied to the Accumulator.
Operation	ACC $\leftarrow$ [m]
Affected flag(s)	None
<b>MOV A,x</b>	Move immediate data to ACC
Description	The immediate data specified is loaded into the Accumulator.
Operation	ACC $\leftarrow$ x
Affected flag(s)	None
<b>MOV [m],A</b>	Move ACC to Data Memory
Description	The contents of the Accumulator are copied to the specified Data Memory.
Operation	[m] $\leftarrow$ ACC
Affected flag(s)	None
<b>NOP</b>	No operation
Description	No operation is performed. Execution continues with the next instruction.
Operation	No operation
Affected flag(s)	None
<b>OR A,[m]</b>	Logical OR Data Memory to ACC
Description	Data in the Accumulator and the specified Data Memory perform a bitwise logical OR operation. The result is stored in the Accumulator.
Operation	ACC $\leftarrow$ ACC "OR" [m]
Affected flag(s)	Z
<b>OR A,x</b>	Logical OR immediate data to ACC
Description	Data in the Accumulator and the specified immediate data perform a bitwise logical OR operation. The result is stored in the Accumulator.
Operation	ACC $\leftarrow$ ACC "OR" x
Affected flag(s)	Z
<b>ORM A,[m]</b>	Logical OR ACC to Data Memory
Description	Data in the specified Data Memory and the Accumulator perform a bitwise logical OR operation. The result is stored in the Data Memory.
Operation	[m] $\leftarrow$ ACC "OR" [m]
Affected flag(s)	Z

<b>RET</b>	Return from subroutine
Description	The Program Counter is restored from the stack. Program execution continues at the restored address.
Operation	Program Counter $\leftarrow$ Stack
Affected flag(s)	None
<b>RET A,x</b>	Return from subroutine and load immediate data to ACC
Description	The Program Counter is restored from the stack and the Accumulator loaded with the specified immediate data. Program execution continues at the restored address.
Operation	Program Counter $\leftarrow$ Stack ACC $\leftarrow$ x
Affected flag(s)	None
<b>RETI</b>	Return from interrupt
Description	The Program Counter is restored from the stack and the interrupts are re-enabled by setting the EMI bit. EMI is the master interrupt global enable bit. If an interrupt was pending when the RETI instruction is executed, the pending Interrupt routine will be processed before returning to the main program.
Operation	Program Counter $\leftarrow$ Stack EMI $\leftarrow$ 1
Affected flag(s)	None
<b>RL [m]</b>	Rotate Data Memory left
Description	The contents of the specified Data Memory are rotated left by 1 bit with bit 7 rotated into bit 0.
Operation	[m].(i+1) $\leftarrow$ [m].i; (i=0~6) [m].0 $\leftarrow$ [m].7
Affected flag(s)	None
<b>RLA [m]</b>	Rotate Data Memory left with result in ACC
Description	The contents of the specified Data Memory are rotated left by 1 bit with bit 7 rotated into bit 0. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.
Operation	ACC.(i+1) $\leftarrow$ [m].i; (i=0~6) ACC.0 $\leftarrow$ [m].7
Affected flag(s)	None
<b>RLC [m]</b>	Rotate Data Memory left through Carry
Description	The contents of the specified Data Memory and the carry flag are rotated left by 1 bit. Bit 7 replaces the Carry bit and the original carry flag is rotated into bit 0.
Operation	[m].(i+1) $\leftarrow$ [m].i; (i=0~6) [m].0 $\leftarrow$ C C $\leftarrow$ [m].7
Affected flag(s)	C

<b>RLCA [m]</b>	Rotate Data Memory left through Carry with result in ACC
Description	Data in the specified Data Memory and the carry flag are rotated left by 1 bit. Bit 7 replaces the Carry bit and the original carry flag is rotated into the bit 0. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.
Operation	ACC.(i+1) ← [m].i; (i=0~6) ACC.0 ← C C ← [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 ← [m].(i+1); (i=0~6) [m].7 ← [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 ← [m].(i+1); (i=0~6) ACC.7 ← [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 ← [m].(i+1); (i=0~6) [m].7 ← C C ← [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	$[m] \leftarrow ACC - [m] - \bar{C}$
Affected flag(s)	OV, Z, AC, C
<b>SDZ [m]</b>	Skip if decrement Data Memory is 0
Description	The contents of the specified Data Memory are first decremented by 1. If the result is 0 the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.
Operation	$[m] \leftarrow [m] - 1$ Skip if $[m]=0$
Affected flag(s)	None
<b>SDZA [m]</b>	Skip if decrement Data Memory is zero with result in ACC
Description	The contents of the specified Data Memory are first decremented by 1. If the result is 0, the following instruction is skipped. The result is stored in the Accumulator but the specified Data Memory contents remain unchanged. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0, the program proceeds with the following instruction.
Operation	$ACC \leftarrow [m] - 1$ Skip if $ACC=0$
Affected flag(s)	None
<b>SET [m]</b>	Set Data Memory
Description	Each bit of the specified Data Memory is set to 1.
Operation	$[m] \leftarrow FFH$
Affected flag(s)	None
<b>SET [m].i</b>	Set bit of Data Memory
Description	Bit i of the specified Data Memory is set to 1.
Operation	$[m].i \leftarrow 1$
Affected flag(s)	None
<b>SIZ [m]</b>	Skip if increment Data Memory is 0
Description	The contents of the specified Data Memory are first incremented by 1. If the result is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.
Operation	$[m] \leftarrow [m] + 1$ Skip if $[m]=0$
Affected flag(s)	None

<b>SIZA [m]</b>	Skip if increment Data Memory is zero with result in ACC
Description	The contents of the specified Data Memory are first incremented by 1. If the result is 0, the following instruction is skipped. The result is stored in the Accumulator but the specified Data Memory contents remain unchanged. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.
Operation	$ACC \leftarrow [m] + 1$ Skip if $ACC=0$
Affected flag(s)	None
<b>SNZ [m].i</b>	Skip if bit i of Data Memory is not 0
Description	If bit i of the specified Data Memory is not 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is 0 the program proceeds with the following instruction.
Operation	Skip if $[m].i \neq 0$
Affected flag(s)	None
<b>SUB A,[m]</b>	Subtract Data Memory from ACC
Description	The specified Data Memory is subtracted from the contents of the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.
Operation	$ACC \leftarrow ACC - [m]$
Affected flag(s)	OV, Z, AC, C
<b>SUBM A,[m]</b>	Subtract Data Memory from ACC with result in Data Memory
Description	The specified Data Memory is subtracted from the contents of the Accumulator. The result is stored in the Data Memory. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.
Operation	$[m] \leftarrow ACC - [m]$
Affected flag(s)	OV, Z, AC, C
<b>SUB A,x</b>	Subtract immediate data from ACC
Description	The immediate data specified by the code is subtracted from the contents of the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.
Operation	$ACC \leftarrow ACC - x$
Affected flag(s)	OV, Z, AC, C
<b>SWAP [m]</b>	Swap nibbles of Data Memory
Description	The low-order and high-order nibbles of the specified Data Memory are interchanged.
Operation	$[m].3\sim[m].0 \leftrightarrow [m].7\sim[m].4$
Affected flag(s)	None

<b>SWAPA [m]</b>	Swap nibbles of Data Memory with result in ACC
Description	The low-order and high-order nibbles of the specified Data Memory are interchanged. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.
Operation	ACC.3~ACC.0 ← [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] ← ACC "XOR" [m]
Affected flag(s)	Z
<b>XOR A,x</b>	Logical XOR immediate data to ACC
Description	Data in the Accumulator and the specified immediate data perform a bitwise logical XOR operation. The result is stored in the Accumulator.
Operation	ACC ← ACC "XOR" x
Affected flag(s)	Z

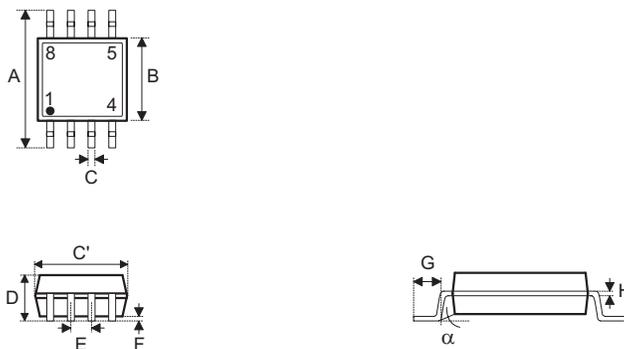
## Package Information

Note that the package information provided here is for consultation purposes only. As this information may be updated at regular intervals users are reminded to consult the [Holtek website](#) for the latest version of the [Package/Carton Information](#).

Additional supplementary information with regard to packaging is listed below. Click on the relevant section to be transferred to the relevant website page.

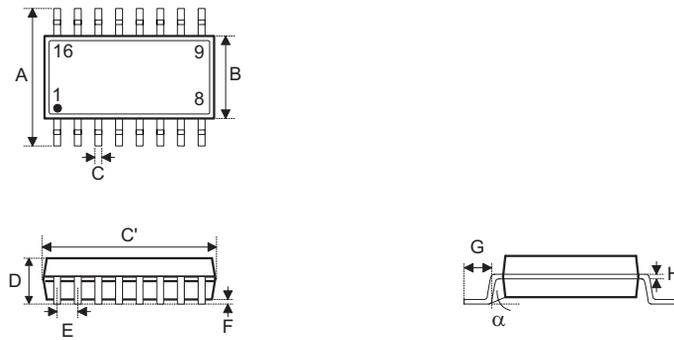
- Package Information (include Outline Dimensions, Product Tape and Reel Specifications)
- The Operation Instruction of Packing Materials
- Carton information

8-pin SOP (150mil) Outline Dimensions



Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A		0.236 BSC	
B		0.154 BSC	
C	0.012	—	0.020
C'		0.193 BSC	
D	—	—	0.069
E		0.050 BSC	
F	0.004	—	0.010
G	0.016	—	0.050
H	0.004	—	0.010
$\alpha$	0°	—	8°

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A		6.00 BSC	
B		3.90 BSC	
C	0.31	—	0.51
C'		4.90 BSC	
D	—	—	1.75
E		1.27 BSC	
F	0.10	—	0.25
G	0.40	—	1.27
H	0.10	—	0.25
$\alpha$	0°	—	8°

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


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

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

Copyright© 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 authorize the use of these products in life-saving, life-sustaining or safety critical components. Any use of HOLTEK's products in life-saving/sustaining or safety applications is entirely at the buyer's risk, and the buyer agrees to defend, indemnify and hold HOLTEK harmless from any damages, claims, suits, or expenses resulting from such use. The information provided in this document, including but not limited to the content, data, examples, materials, graphs, and trademarks, is the intellectual property of HOLTEK (and its licensors, where applicable) and is protected by copyright law and other intellectual property laws. No license, express or implied, to any intellectual property right, is granted by HOLTEK herein. HOLTEK reserves the right to revise the information described in the document at any time without prior notice. For the latest information, please contact us.