A/D Type Recorder 8-Bit Flash MCU

HT83F22

Revision: V.1.00 Date: April 01, 2011

www.holtek.com

Table of Contents

Features

CPU Features

- Operating Voltage:
  f_{SYS}= 8MHz: 2.2V~5.5V
  f_{SYS}= 12MHz: 2.7V~5.5V
  f_{SYS}= 20MHz: 4.5V~5.5V
- Up to 0.2μs instruction cycle with 20MHz system clock at V_{DD}=5V
- Power down and wake-up functions to reduce power consumption
- Five oscillators:
  External Crystal - HXT
  External 32.768kHz Crystal - LXT
  External RC - ERC
  Internal RC - HIRC
  Internal 32kHz RC - LIRC
- Multi-mode operation: NORMAL, SLOW, IDLE and SLEEP
- Fully integrated internal 4MHz, 8MHz and 12MHz oscillator requires no external components
- All instructions executed in one or two instruction cycles
- Table read instructions
- 63 powerful instructions
- Up to 8-level subroutine nesting
- Bit manipulation instruction

Peripheral Features

- Flash Program Memory: 4Kx16
- RAM Data Memory: 384x8
- Watchdog Timer function
- Up to 24 bidirectional I/O lines
- Software controlled 4-SCOM lines LCD driver with1/2 bias
- Multiple pin-shared external interrupts
- Multiple Timer Module for time measurement, input capture, compare match output, PWM output or single pulse output function
- Single 8-bit programmable Timer/Event Counter with overflow interrupt and prescaler
- Two Serial Interface Modules with Dual SPI and I^2C interfaces
- Single Comparator
- Dual Time-Base functions for generation of fixed time interrupt signals
- Up to 8 channel 12-bit resolution A/D converter
- Single channel 12-bit D/A converter
- Output Signal Power Amplifier
- Input Signal Amplifier with Automatic Gain Control - AGC
- Switch Capacitor Filter function
- Low voltage reset function
- Low voltage detect function
- Package types: 48LQFP

General Description

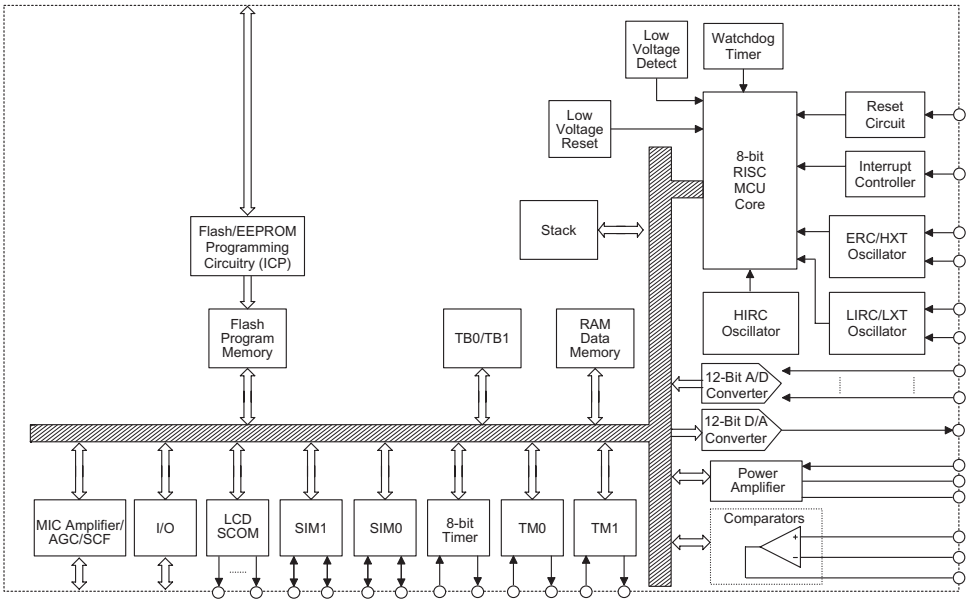
The HT83F22 is an A/D type 8-bit high performance Flash Memory MCU which includes a digital voice recorder, synthesizer and tone generator. It is specifically designed for applications which require multiple I/Os and sound effects, such as voice and melodies. Additionally, it can provide various sampling rates and beats, tone levels, tempos for speech synthesisers and melody generators. Furthermore, with the internal amplifer, auto gain control, SCF filter and the fast 12-bit A/D converter features, this device provides a means of implementing digital audio recorder applications. Offering users the convenience of Flash Memory multi-programming features, this device also includes a wide range of functions and features. Analog features include a multi-channel 12-bit A/D converter and comparator. Multiple and extremely flexible Timer Modules provide timing, pulse generation and PWM generation functions. Communication with the outside world is catered for by including fully integrated SPI or I^2C interface functions, two popular interfaces which provide designers with a means of easy communication with external peripheral hardware. Protective features such as an internal Watchdog Timer, Low Voltage Reset and Low Voltage Detector coupled with excellent noise immunity and ESD protection ensure that reliable operation is maintained in hostile electrical environments.It also includes an integrated high quality, voltage type DAC output, a two-stage amplifier with Auto Gain Control for microphone input signal amplification and a power amplifier for speaker driving. The external interrupt can be triggered with falling edges or both falling and rising edges.

A full choice of HXT, LXT, ERC, HIRC and LIRC oscillator functions are provided including a fully integrated system oscillator which requires no external components for its implementation. The ability to operate and switch dynamically between a range of operating modes using different clock sources gives users the ability to optimise microcontroller operation and minimise power consumption.

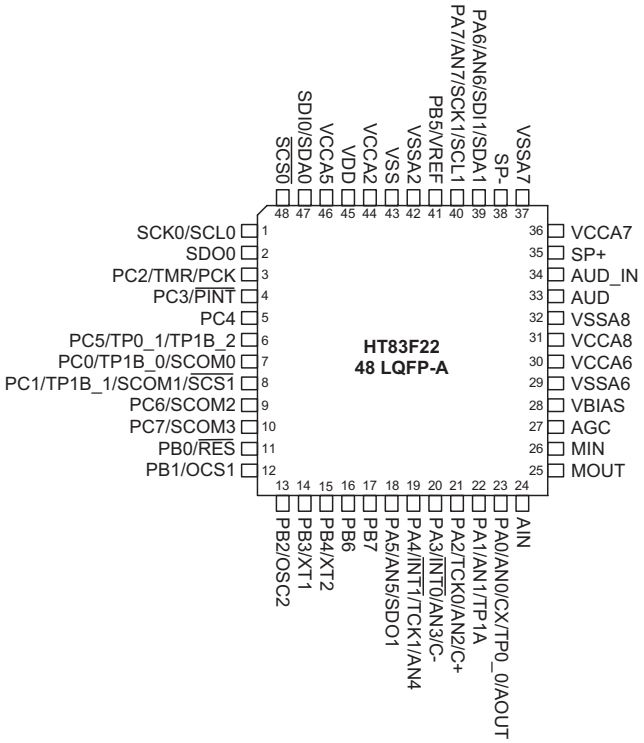
The inclusion of flexible I/O programming features, Time-Base functions along with many other features ensure that the devices will find excellent use in applications such as voice recorder, answering machine, electronic metering, environmental monitoring, handheld instruments, household appliances, electronically controlled tools, motor driving in addition to many others.

This device is fully supported by the Holtek range of fully functional development and programming tools, providing a means for fast and efficient product development cycles.

Block Diagram



Pin Assignment



Pin Description

The pins on this device can be referenced by their Port name, e.g. PA.0, PA.1 etc, which refer to the digital I/O function of the pins. However these Port pins are also shared with other function such as the Analog to Digital Converter, Serial Port pins etc. The function of each pin is listed in the following table, however the details behind how each pin is configured is contained in other sections of the datasheet.

Pin Name	Function	OP	I/T	O/T	Pin-Shared Mapping
PA0~PA7	Port A	PAWU PAPU	ST	CMOS	—
PB0~PB7	Port B	PBPU	ST	CMOS	—
PC0~PC7	Port C	PCPU	ST	CMOS	—
AN0~AN7	ADC input	ACER	AN	—	PA0~PA7
VREF	ADC reference input	ADCR1	AN	—	PB5
C-	Comparator input	CPC	AN	—	PA3
C+	Comparator input		AN	—	PA2
CX	Comparator output		—	CMOS	PA0
TCK0, TCK1, TMR	TC0, TC1,TMR input	—	ST	—	PA2, PA4, PC2
TP0_0, TP0_1	TM0 I/O	TMPC0	ST	CMOS	PA0, PC5
TP1A	TM1 I/O	TMPC0	ST	CMOS	PA1
TP1B_0, TP1B_1, TP1B_2	TM1 I/O	TMPC0	ST	CMOS	PC0, PC1,PC5
INT0, INT1	Ext. Interrupt 0, 1	—	ST	—	PA3, PA4
$\overline{\text{PINT}}$	Peripheral Interrupt	—	ST	—	PC3
PCK	Peripheral Clock output	—	—	CMOS	PC2
SDI1	SPI1 Data input	—	ST	—	PA6
SDO1	SPI1 Data output	—		CMOS	PA5
$\overline{\text{SCS1}}$	SPI1 Slave Select	—	ST	CMOS	PC1
SCK1	SPI1 Serial Clock	—	ST	CMOS	PA7
SCL1	I^2C1 Clock	—	ST	NMOS	PA7
SDA1	I^2C1 Data	—	ST	NMOS	PA6
SDI0	SPI0 Data input	—	ST	—	—
SDO0	SPI0 Data output	—	—	CMOS	—
$\overline{\text{SCS0}}$	SPI0 Slave Select	—	ST	CMOS	—
SCK0	SPI0 Serial Clock	—	ST	CMOS	—
SCL0	I^2C0 Clock	—	ST	NMOS	—
SDA0	I^2C0 Data	—	ST	NMOS	—
SCOM0~SCOM3	SCOM0~SCOM3	SCOMC	—	SCOM	PC0,PC1, PC6,PC7
OSC1	HXT/ERC pin	CO	HXT	—	PB1
OSC2	HXT pin	CO	—	HXT	PB2
XT1	LXT pin	CO	LXT	—	PB3
XT2	LXT pin	CO	—	LXT	PB4
$\overline{\text{RES}}$	Reset input	CO	ST	—	PB0
AUD	DAC output	—	—	AN	—
SP+, SP-	Power amplifier output	—	—	AN	—
AUD_IN	Power amplifier input	—	AN	—	—
VBIAS	AGC and Power amplifier voltage bias reference	—	PWR	—	—

Pin Name	Function	OP	I/T	O/T	Pin-Shared Mapping
MIN	Microphone input	—	AN	—	—
MOUT	Microphone output	—	—	AN	—
AIN	Amplifier input	—	AN	—	—
AOUT	Amplifier output	—	—	AN	PA0
AGC	AGC input	—	AN	—	—
VSS	Ground	—	PWR	—	—
VDD	IO, Core Power	—	PWR	—	—
VCCA2	ADC Power	—	PWR	—	—
VCCA5	SIM0 SPI Power	—	PWR	—	—
VCCA6	AGC Power	—	PWR	—	—
VCCA7	Power Amp Power	—	PWR	—	—
VCCA8	DAC Power	—	PWR	—	—
NC	No connection	—	—	—	—
VSSA2	ADC Ground	—	PWR	—	—
VSSA6	AGC Ground	—	PWR	—	—
VSSA7	Power Amp Ground	—	PWR	—	—
VSSA8	DAC Ground	—	PWR	—	—

Note: I/T: Input type; O/T: Output type

OP: Optional by configuration option (CO) or register option

PWR: Power; CO: Configuration option; ST: Schmitt Trigger input

CMOS: CMOS output; NMOS: NMOS output

AN: Analog input pin

HXT: High frequency crystal oscillator

LXT: Low frequency crystal oscillator

*: VDD is the device power supply while AVDD is the ADC power supply. The AVDD pin is bonded together internally with VDD.

**: VSS is the device ground pin while AVSS is the ADC ground pin. The AVSS pin is bonded together internally with VSS.

As the Pin Description Summary table applies to the package type with the most pins, not all of the above listed pins may be present on package types with smaller numbers of pins.

Absolute Maximum Ratings

Supply Voltage ..... V_{SS}-0.3V to V_{SS}+6.0V

Input Voltage ..... V_{SS}-0.3V to V_{DD}+0.3V

I_{OL} Total 80mA

Total Power Dissipation 500mW

Storage Temperature ..... -50°C to 125°C

Operating Temperature ..... -40°C to 85°C

I_{OH} Tatal ..... -80mA

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

D.C. Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD1}	Operating Voltage (HXT)	—	f_{SYS}=8MHz	2.2	—	5.5	V
			f_{SYS}=10MHz	2.7	—	5.5	V
			f_{SYS}=12MHz	3.3	—	5.5	V
			f_{SYS}=16MHz	4.5	—	5.5	V
V_{DD2}	Operating Voltage (ERC)	—	f_{SYS}=6MHz	2.2	—	5.5	V
			f_{SYS}=8MHz	2.7	—	5.5	V
			f_{SYS}=12MHz	4.5	—	5.5	V
V_{DD3}	Operating Voltage (HIRC)	—	f_{SYS}=8MHz	2.2	—	5.5	V
			f_{SYS}=12MHz	2.7	—	5.5	V
I_{DD1}	Operating Current ($f_{SYS}=f_H$, $f_S=f_{SUB}=f_{LXT}$ or f_{LIRC})	3V	No load, f_H=8MHz, ADC off, WDT enable	—	0.8	1.5	mA
		5V		—	2.5	4	mA
		3V	No load, f_H=10MHz, ADC off, WDT enable	—	1.3	2.0	mA
		5V		—	3.3	5.0	mA
		3V	No load, f_H=12MHz, ADC off, WDT enable	—	1.6	2.4	mA
		5V		—	4.0	6.0	mA
		3V	No load, f_H=16MHz, ADC off, WDT enable	—	2.0	3.0	mA
		5V		—	5.1	7.7	mA
		5V	No load, f_H=20MHz, ADC off, WDT enable	—	6.2	9.3	mA
I_{DD2}	Operating Current (ERC, $f_{SYS}=f_H$, $f_S=f_{SUB}=f_{LXT}$ or f_{LIRC})	3V	No load, f_H=6MHz, ADC off, WDT enable	—	0.9	1.4	mA
		5V		—	2.0	3.0	mA
		3V	No load, f_H=8MHz, ADC off, WDT enable	—	1.3	2.0	mA
		5V		—	2.8	4.5	mA
		3V	No load, f_H=12MHz, ADC off, WDT enable	—	1.9	2.9	mA
		5V		—	4.0	6.0	mA
		5V	No load, f_H=16MHz, ADC off, WDT enable	—	5.0	7.5	mA
I_{DD3}	Operating Current (HIRC, $f_{SYS}=f_H$, $f_S=f_{SUB}=f_{LXT}$ or f_{LIRC})	3V	No load, f_H=4MHz, ADC off, WDT enable	—	0.7	1.1	mA
		5V		—	1.5	2.5	mA
		3V	No load, f_H=8MHz, ADC off, WDT enable	—	1.3	2.0	mA
		5V		—	2.8	4.5	mA
		3V	No load, f_H=12MHz, ADC off, WDT enable	—	2.1	3.2	mA
		5V		—	4.0	6.0	mA
I_{DD4}	Operating Current (LXT, $f_{SYS}=f_L$, $f_S=f_{SUB}=f_{LXT}$ or f_{LIRC})	3V	No load, f_H=12MHz, $f_L=f_H/2$, ADC off, WDT enable	—	0.9	1.4	mA
		5V		—	2.1	3.3	mA
		3V	No load, f_H=12MHz, $f_L=f_H/4$, ADC off, WDT enable	—	0.6	0.9	mA
		5V		—	1.7	2.6	mA
		3V	No load, f_H=12MHz, $f_L=f_H/8$, ADC off, WDT enable	—	0.4	0.6	mA
		5V		—	1.4	2.1	mA
		3V	No load, f_H=12MHz, $f_L=f_H/16$, ADC off, WDT enable	—	0.4	0.6	mA
		5V		—	1.2	1.8	mA
		3V	No load, f_H=12MHz, $f_L=f_H/32$, ADC off, WDT enable	—	0.3	0.5	mA
		5V		—	1.1	1.7	mA
		3V	No load, f_H=12MHz, $f_L=f_H/64$, ADC off, WDT enable	—	0.3	0.5	mA
		5V		—	1.1	1.7	mA
I_{DD5}	Operating Current (LXT, $f_{SYS}=f_L=f_{LXT}$, $f_S=f_{SUB}=f_{LXT}$)	3V	No load, ADC off, WDT enable, LXTLP=0	—	70	110	uA
		5V		—	210	320	uA
		3V	No load, ADC off, WDT enable, LXTLP =1	—	70	110	uA
		5V		—	210	320	uA
I_{DD6}	Operating Current (LIRC OSC, $f_{SYS}=f_L=f_{LIRC}$, $f_S=f_{SUB}=f_{LIRC}$)	3V	No load, ADC off, WDT enable	—	68	100	uA
		5V		—	210	320	uA

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{DD7}	Operating Current (LXT+LIRC, $f_{SYS}=f_L=f_{LXT}$, $f_S=f_{SUB}=f_{LIRC}$)	3V	No load, ADC off, WDT enable, LXTLP =0	—	68	100	uA
		5V		—	210	320	uA
I_{STB1}	Stanby Current (Idle) (HXT, $f_{SYS}=f_H$, $f_S=f_{SUB}=f_{LXT}$ or f_{LIRC})	3V	No load, system HALT, ADC off, WDT enable, f_{SYS}=12MHz	—	0.5	1.0	mA
		5V		—	1.2	2.0	mA
I_{STB2}	Stanby Current (Idle) (HXT, f_{SYS}=off, f_S=T1)	3V	No load, system HALT, ADC off, WDT enable, f_{SYS}=12MHz	—	1.3	3.0	μA
		5V		—	2.2	5.0	μA
I_{STB3}	Stanby Current (Idle) (HXT, f_{SYS}=off, $f_S=f_{SUB}=f_{LXT}$ or f_{LIRC})	3V	No load, system HALT, ADC off, WDT enable, f_{SYS}=12MHz	—	1.3	3.0	μA
		5V		—	2.2	5.0	μA
I_{STB4}	Stanby Current (Idle) (ERC, f_{SYS}=off, $f_S=f_{SUB}=f_{LXT}$)	3V	No load, system HALT, ADC off, WDT enable, f_{SYS}=12MHz	—	1.3	3.0	μA
		5V		—	1.8	5.0	μA
I_{STB5}	Stanby Current (Idle) (HIRC, f_{SYS}=off, $f_S=f_{SUB}=f_{LIRC}$)	3V	No load, system HALT, ADC off, WDT enable, f_{SYS}=12MHz	—	0.9	3.0	μA
		5V		—	1.6	5.0	μA
I_{STB6}	Stanby Current (Idle) (HXT, $f_{SYS}=f_L$, $f_S=f_{SUB}=f_{LXT}$ or f_{LIRC})	3V	No load, system HALT, ADC off, WDT enable, f_{SYS}=12MHz/64	—	0.5	0.8	mA
		5V		—	1.2	1.8	mA
I_{STB7}	Stanby Current (Idle) (HXT, f_{SYS}=off, $f_S=f_{SUB}=f_{LXT}$ or f_{LIRC})	3V	No load, system HALT, ADC off, WDT enable, f_{SYS}=12MHz/64	—	1.3	3.0	μA
		5V		—	2.2	5.0	μA
I_{STB8}	Stanby Current (Idle) (LXT, $f_{SYS}=f_L=f_{LXT}$, $f_S=f_{SUB}=f_{LXT}$)	3V	No load, system HALT, ADC off, WDT enable, f_{SYS}=32768Hz	—	1.9	4.0	μA
		5V		—	3.3	7.0	μA
I_{STB9}	Stanby Current (Idle) (LXT, f_{SYS}=off, f_S=T1)	3V	No load, system HALT, ADC off, WDT enable, f_{SYS}=32768Hz	—	1.3	3.0	μA
		5V		—	2.1	5.0	μA
I_{STB10}	Stanby Current (Idle) (LXT, f_{SYS}=off, $f_S=f_{SUB}=f_{LXT}$)	3V	No load, system HALT, ADC off, WDT enable, f_{SYS}=32768Hz	—	1.3	3.0	μA
		5V		—	2.1	5.0	μA
I_{STB11}	Stanby Current (Idle) (LIRC, f_{SYS}=off, $f_S=f_{SUB}=f_{LIRC}$)	3V	No load, system HALT, ADC off, WDT enable, f_{SYS}=32kHz	—	1.3	3.0	μA
		5V		—	2.2	5.0	μA
I_{STB12}	Stanby Current (Idle) (LXT, f_{SYS}=off, $f_S=f_{SUB}=f_{LIRC}$)	3V	No load, system HALT, ADC off, WDT enable, f_{SYS}=32768Hz	—	1.3	3.0	μA
		5V		—	2.2	5.0	μA
I_{STB13}	Stanby Current (Sleep) (HXT, f_{SYS}=off, $f_S=f_{SUB}=f_{LXT}$ or f_{LIRC})	3V	No load, system HALT, ADC off, WDT disable, f_{SYS}=12MHz	—	0.1	1	μA
		5V		—	0.3	2	μA
I_{STB14}	Stanby Current (Sleep) (HXT, f_{SYS}=off, $f_S=f_{SUB}=f_{LTX}$)	3V	No load, system HALT, ADC off, WDT enable, f_{SYS}=12MHz	—	1.3	5	μA
		5V		—	2.1	10	μA
I_{STB15}	Stanby Current (Sleep) (HXT, f_{SYS}=off, $f_S=f_{SUB}=f_{LIRC}$)	3V	No load, system HALT, ADC off, WDT enable, f_{SYS}=12MHz	—	1.3	5	μA
		5V		—	2.2	10	μA
I_{STB16}	Stanby Current (Sleep) (LXT, f_{SYS}=off, $f_S=f_{SUB}=f_{LXT}$ or f_{LIRC})	3V	No load, system HALT, ADC off, WDT disable, f_{SYS}=32768Hz	—	0.1	1.0	μA
		5V		—	0.3	2.0	μA

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{STB17}	Stanby Current (Sleep) (LXT, f_{SYS}=off, f_S=f_{SUB}=f_{LXT})	3V	No load, system HALT, ADC off, WDT enable, f_{SYS}=32768Hz	—	1.3	3.0	μA
		5V		—	2.2	5.0	
I_{STB18}	Stanby Current (Sleep) (HXT, f_{SYS}=off, f_S=f_{SUB}=f_{LXT} or f_{LIRC})	5V	No load, system HALT, ADC off, WDT disable, f_{SYS}=12MHz, LVR enable and LVDEN=1	—	60	90	μA
V_{IL1}	Input Low Voltage for I/O Ports or input pins except $\overline{RES}$ pin	—	—	0	–	0.3V_{DD}	V
V_{IH1}	Input High Voltage for I/O Ports or input pins except $\overline{RES}$ pin	—	—	0.7V_{DD}	–	V_{DD}	V
V_{IL2}	Input Low Voltage ($\overline{RES}$)	—	—	0	–	0.4V_{DD}	V
V_{IH2}	Input High Voltage ($\overline{RES}$)	—	—	0.9V_{DD}	–	V_{DD}	V
V_{LVR1}	Low Voltage Reset Voltage	—	LVR Enable, 2.1V option	-5%× Typ.	2.1	+5%× Typ.	V
V_{LVR2}			LVR Enable, 2.55V option		2.55		V
V_{LVR3}			LVR Enable, 3.15V option		3.15		V
V_{LVR4}			LVR Enable, 4.2V option		4.2		V
I_{LVR}	Low Voltage Reset Current	3V	LVR Enable, LVDEN=0	—	48	70	μA
		5V		—	55	80	uA
V_{LVD1}	Low Voltage Detector Voltage	—	LVDEN = 1, V_{LVD} = 2.0V	-5%× Typ.	2.0	+5%× Typ.	V
V_{LVD2}			LVDEN = 1, V_{LVD} = 2.2V		2.2		V
V_{LVD3}			LVDEN = 1, V_{LVD} = 2.4V		2.4		V
V_{LVD4}			LVDEN = 1, V_{LVD} = 2.7V		2.7		V
V_{LVD5}			LVDEN = 1, V_{LVD} = 3.0V		3.0		V
V_{LVD6}			LVDEN = 1, V_{LVD} = 3.3V		3.3		V
V_{LVD7}			LVDEN = 1, V_{LVD} = 3.6V		3.6		V
V_{LVD8}			LVDEN = 1, V_{LVD} = 4.4V		4.4		V
I_{LVD1}	Low Voltage Detector Current	3V	LVR disable, LVDEN = 1	—	47	70	μA
I_{LVD2}			LVR enable, LVDEN = 1	—	49	75	μA
I_{LVD1}	Low Voltage Detector Current	5V	LVR disable, LVDEN = 1	—	52	75	μA
I_{LVD2}			LVR enable, LVDEN = 1	—	55	80	μA
I_{OL}	I/O Port Sink Current	3V	V_{OL}=0.1V_{DD}	4	8	—	mA
		5V	V_{OL}=0.1V_{DD}	10	20	—	mA
I_{OH}	I/O Port, Source Current	3V	V_{OH}=0.9V_{DD}	-2	-4	—	mA
		5V	V_{OH}=0.9V_{DD}	-5	-10	—	mA
V_{SCOM}	V_{DD}/2 voltage for LCD COM port (INVEN=0)	—	No load	0.475	0.5	0.525	V_{DD}
R_{PH}	Pull-high Resistance of I/O Ports	3V	—	20	60	100	kΩ
		5V	—	10	30	50	kΩ
AV_{DD}	Analog operating voltage	—	V_{REF}=AV_{DD}	2.7	—	5.5	V
V_{AD}	AD Input Voltage	—	—	0	—	AV_{DD}/ V_{REF}	V
V_{REF}	ADC input reference voltage range	—	AV_{DD}=3V	1.6	—	AV_{DD} + 0.1	V
		—	AV_{DD}=5V	1.6	—	AV_{DD} + 0.1	V
V_{BG}	Bandgap reference with buffer voltage	—	—	-3%× Typ.	1.25	+3%× Typ.	V
I_{BG}	Bandgap reference with buffer driving current	—	V_{BG} is used, LVR disable, LVDEN=0	–	200	260	μA
DNL	Differential non-linearity	3V	V_{REF}=AV_{DD}=V_{DD}, t_{AD} =0.5μs	-3	—	+3	LSB
		5V	V_{REF}=AV_{DD}=V_{DD}, t_{AD} =0.5μs	-3	—	+3	LSB

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
INL	Integral non-linearity	3V	V_{REF}=AV_{DD}=V_{DD}, t_{AD} =0.5μs	-4	—	+4	LSB
		5V	V_{REF}=AV_{DD}=V_{DD}, t_{AD} =0.5μs	-4	—	+4	LSB
I_{ADC}	only ADC Enable, Others Disable	3V	No load (t_{AD} =0.5μs)	—	1.0	2.0	mA
		5V	No load (t_{AD} =0.5μs)	—	1.5	3.0	mA
PO	Internal AMP Output Power	3V	(THD+N)/S<=1%, RL=8_ VIN=1kHz Sinewave	—	90	—	mW
			(THD+N)/S<=10%, RL=8_ VIN=1kHz Sinewave	—	125	—	mW
		5V	(THD+N)/S<=1%, RL=8_ VIN=1kHz Sinewave	—	385	—	mW
			(THD+N)/S<=10%, RL=8_ VIN=1kHz Sinewave	—	490	—	mW
Io	AUD current source	3V	V_{OH}=0.9V_{DD}	—	-3	—	mA
		5V	V_{OH}=0.9V_{DD}	—	-10	—	mA

DAC and Power Amplifier Electrical Characteristics

Ta=25°C

DAC							
Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
THD+N	THD+N Note1	5V	10KΩ Load	—	-55	—	dB
Power Amplifier							
G	Gain	5V	Note2	—	10	—	V/V
THD+N	THD+N Note1	5V	8Ω Load	—	-51	—	dB
		5V	16Ω Load	—	-54	—	dB
POUT	Output Power	5V	8Ω Load, THD=1%	—	710	—	mW
		5V	16Ω Load, THD=1%	—	540	—	mW

Note1:sine wave input@ 1kHz, –6dB

Note2:sine wave input@ 1kHz, 50mVp-p

Automatic Gain Control Electrical Characteristics

Parameter	Symbol	Condition	SPECIFICATION			Unit
			Min.	Typ.	Max.	
Preamp Input Impedance	RMIC	V_{DD}=5V	—	20	—	kΩ
MIC Input Swing Voltage	VMIC	V_{DD}=5V, TA=25°C	1	—	200	mVp-p
Preamp Gain	APRE	V_{DD}=5V, TA=25°C	—	—	22	dB
Gain of Programmable Gain Amplifier	APGA	V_{DD}=5V, TA=25°C RGS='000' RGS='001' RGS='010' RGS='011' RGS='100' RGS='101' RGS='110' RGS='111'	—	1.16 1.56 2.22 2.91 5.09 8.91 13.82 25.97	—	V/V

A.C. Characteristics

Symbol	Parameter	V_{DD}	Condition	Min.	Typ.	Max.	Unit
f_{SYS1}	System clock (HXT)	2.2~5.5V	—	2	—	8	MHz
		2.7~5.5V		2	—	10	MHz
		3.3~5.5V		2	—	12	MHz
		4.5~5.5V		2	—	16	MHz
f_{SYS2}	System clock (ERC)	5V	Ta = 25° C External R_{ERC} = 120 KΩ	-2%×Typ.	8	+2%×Typ.	MHz
f_{SYS3}	System clock (HIRC)	5V	Ta = 25° C	-2%×Typ.	8	+2%×Typ.	MHz
f_{SYS4}	System Clock (LXT)	—	—	—	32768	—	Hz
f_{TIMER}	Timer I/P Frequency (TMR)	2.2~5.5V	—	2	—	8	MHz
		2.7~5.5V		2	—	10	MHz
		3.3~5.5V		2	—	12	MHz
		4.5~5.5V		2	—	16	MHz
f_{LIRC}	System Clock (32K RC)	5V	Ta = 25° C	-10%×Typ.	32	+10%×Typ.	kHz
t_{RES}	External reset low pulse width	—	—	1	—	—	μs
t_{SST}	System start-up timer period (wake-up from HALT)	—	f_{SYS}=HXT or LXT	—	1024	—	t_{SYS}
			f_{SYS}=ERC or HIRC	—	15~16	—	
			f_{SYS}=LIRC	—	1~2	—	
t_{INT}	Interrupt pulse width	—	—	1	—	—	μs
t_{LVR}	Low Voltage Width to Reset	—	—	120	240	480	μs
t_{LVD}	Low Voltage Width to Interrupt	—	—	20	60	120	μs
t_{LVDS}	LVDO stable time	—	For all V_{LVD}, LVR disable	15	—	—	μs
t_{BGS}	VBG turn on stable time	—	—	10	—	—	ms
t_{AD}	A/D Clock Period	2.7~5.5V	—	0.5	—	100	μs
t_{ADC}	AD Conversion Time(Note2)	2.7~5.5V	12 bit ADC	—	16	—	t_{AD}
t_{ON2ST}	ADC on to ADC start	2.7~5.5V	—	2	—	—	μs

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

2. * For f_{ERC}, as the resistor tolerance will influence the frequency a precision resistor is recommended.

3. To maintain the accuracy of the internal HIRC oscillator frequency, a 0.1μF decoupling capacitor should be connected between VDD and VSS and located as close to the device as possible.

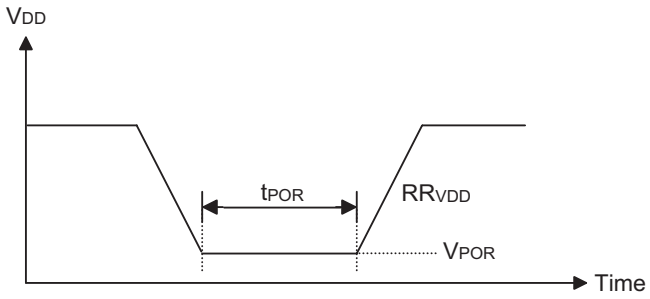
Comparator Electrical Characteristics

Symbol	Parameter	V_{DD}	Condition	Min.	Typ.	Max.	Unit
—	Comparator operating voltage	—	—	2.2	—	5.5	V
—	Comparator operating current	5V	—	—	—	200	μA
—	Comparator power down current	5V	Comparator disable	—	—	0.1	μA
V_{CMPOS}	Comparator input offset voltage	5V	—	-10	—	+10	mV
V_{HYS}	Hysteresis width	5V	—	20	40	60	mV
V_{CM}	Comparator common mode voltage range	—	—	V_{SS}	—	V_{DD}-1.4V	V
A_{OL}	Comparator open loop gain	—	—	60	80	—	dB
t_{PD}	Comparator response time	3V	With 100mV overdrive (Note)	—	200	400	ns
		5V					

Note: Measured with comparator one input pin at V_{CM} = (V_{DD}-1.4)/2 while the other pin input transition from V_{SS} to (V_{CM} +100mV) or from V_{DD} to (V_{CM} -100mV).

Power on Reset (AC+DC) Electrical Characteristic

Symbol	Parameter	V_{DD}	Condition	Min.	Typ.	Max.	Unit
V_{POR}	V_{DD} Start Voltage to ensure Power-on Reset	—	—	—	—	100	mV
$R_{POR\ AC}$	V_{DD} Rise Rate to ensure Power-on Reset	—	—	0.035	—	—	V/ms
t_{POR}	Minimum Time for V_{DD} Stays at V_{POR} to Ensure Power-on Reset	—	—	1	—	—	ms



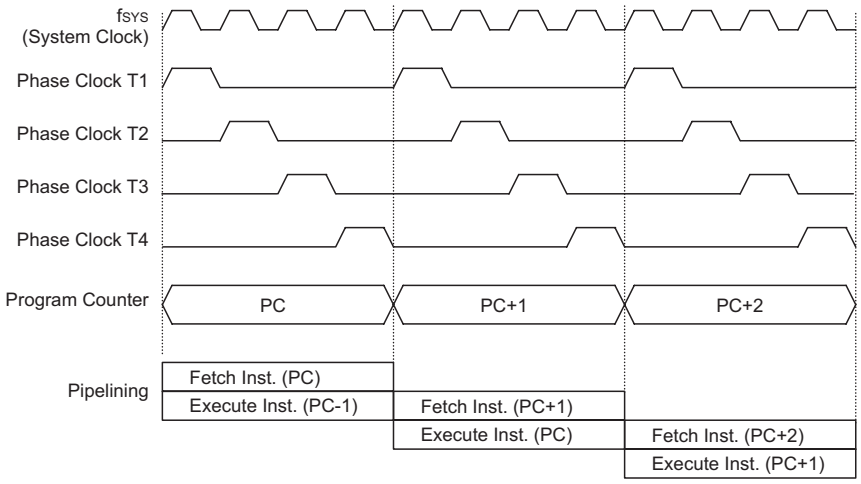
System Architecture

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

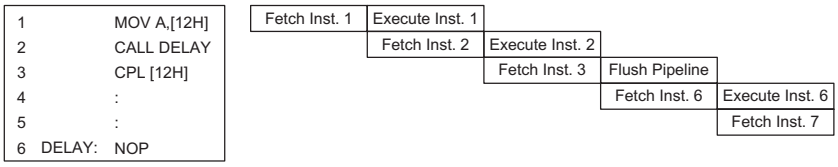
Clocking and Pipelining

The main system clock, derived from either a HXT, LXT, HIRC, LIRC or ERC 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	
Program Counter High Byte	**PCL Register**
PC11~PC8	PCL7~PCL0

Program Counter

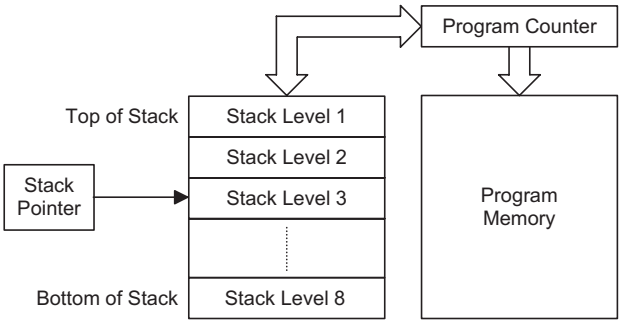
The lower byte of the Program Counter, known as the Program Counter Low register or PCL, is available for program control and is a readable and writeable register. By transferring data directly into this register, a short program jump can be executed directly, however, as only this low byte is available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

Stack

This is a special part of the memory which is used to save the contents of the Program Counter only. The stack has multiple levels depending upon the device 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

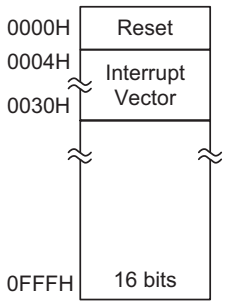
The Program Memory is the location where the user code or program is stored. For this device series the Program Memory is Flash type, which means it can be programmed and re-programmed a large number of times, allowing the user the convenience of code modification on the same device. By using the appropriate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

Structure

The Program Memory has a capacity of 4Kx16 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.

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.



Porgram Memory Structure

Look-up Table

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

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

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

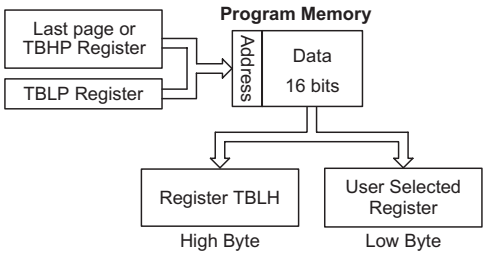


Table Program Example

The following example shows how the table pointer and table data is defined and retrieved from the microcontroller. This example uses raw table data located in the Program Memory which is stored there using the ORG statement. The value at this ORG statement is "F00H" which refers to the start address of the last page within the 4K Program Memory of the HT83F22. The table pointer is setup here to have an initial value of "06H". This will ensure that the first data read from the data table will be at the Program Memory address "F06H" or 6 locations after the start of the last page. Note that the value for the table pointer is referenced to the first address of the present page if the "TABRD [m]" instruction is being used. 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.

In Circuit Programming

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

As an additional convenience, Holtek has provided a means of programming the microcontroller in-circuit using a 5-pin interface. This provides manufacturers with the possibility of manufacturing their circuit boards complete with a programmed or un-programmed microcontroller, and then programming or upgrading the program at a later stage. This enables product manufacturers to easily keep their manufactured products supplied with the latest program releases without removal and re-insertion of the device.

MCU Programming Pins	Function
PA0	Serial Data Input/Output
PA2	Serial Clock
RES	Device Reset
VDD	Power Supply
VSS	Ground

The Program Memory can be programmed serially in-circuit using this 5-wire interface. Data is downloaded and uploaded serially on a single pin with an additional line for the clock. Two additional lines are required for the power supply and one line for the reset. The technical details regarding the in-circuit programming of the devices are beyond the scope of this document and will be supplied in supplementary literature.

During the programming process the RES pin will be held low by the programmer disabling the normal operation of the microcontroller and taking control of the PA0 and PA2 I/O pins for data and clock programming purposes. The user must there take care to ensure that no other outputs are connected to these two pins.

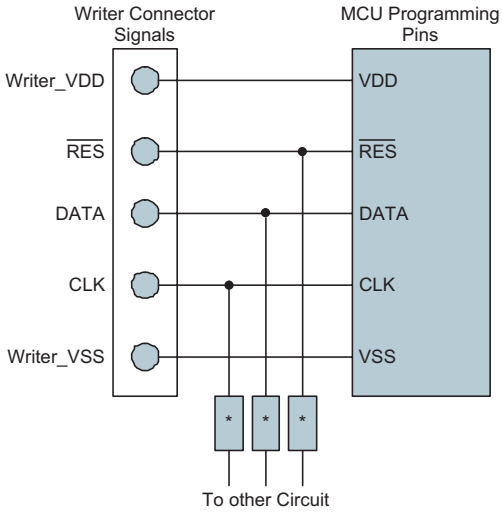
Table Read Program Example

```
tempreg1  db ?       ; temporary register #1
tempreg2  db ?       ; temporary register #2
:
:
mov a,06h            ; initialise low table pointer - note that this address
mov tblp,a           ; is referenced
mov a,0Fh            ; initialise high table pointer
tbhp,a
:
:
tabrd tempreg1       ; transfers value in table referenced by table pointer data at program
                     ; memory address F06H 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 F05H transferred to tempreg2 and TBLH in this
                     ; example the data 1AH is transferred to tempreg1 and data 0FH to
                     ; register tempreg2
:
:
org F00h             ; sets initial address of program memory

dc 00Ah, 00Bh, 00Ch, 00Dh, 00Eh, 00Fh, 01Ah, 01Bh
:
:
```



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

Programmer Pin	MCU Pins
RES	PB0
DATA	PA0
CLK	PA2

Programmer and MCU Pins

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

Divided into two sections, the first of these is an area of RAM, known as the Special Function Data Memory.

Here are located registers which 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.

Capacity	Banks
384x8	0: 80H~FFH 1: 80H~FFH 2: 80H~FFH

Bank 0, 1, 2			
	RAM Mapping		RAM Mapping
00H	IAR0	28H	VOL
01H	MP0	29H	—
02H	IAR1	2AH	SIM0C0
03H	MP1	2BH	SIM0C1
04H	BP	2CH	SIM0D
05H	ACC	2DH	SIM0A/SIM0C2
06H	PCL	2EH	ADRL
07H	TBLP	2FH	ADRH
08H	TBLH	30H	ADCR0
09H	TBHP	31H	ADCR1
0AH	STATUS	32H	ACER
0BH	SMOD	33H	AGCC
0CH	LVDC	34H	CPC
0DH	INTEG	35H	—
0EH	WDTC	36H	SIM1C0
0FH	TBC	37H	SIM1C1
10H	INTC0	38H	SIM1D
11H	INTC1	39H	SIM1A/SIM1C2
12H	INTC2	3AH	TM0C0
13H	INTC3	3BH	TM0C1
14H	MFI0	3CH	TM0DL
15H	MFI1	3DH	TM0DH
16H	MFI2	3EH	TM0AL
17H	—	3FH	TM0AH
18H	PAWU	40H	TMPC0
19H	PAPU	41H	—
1AH	PA	42H	—

Bank 0, 1, 2			
	RAM Mapping		RAM Mapping
1BH	PAC	43H	—
1CH	PBPU	44H	—
1DH	PB	45H	TM1C0
1EH	PBC	46H	TM1C1
1FH	PCPU	47H	TM1C2
20H	PC	48H	TM1DL
21H	PCC	49H	TM1DH
22H	TMRC	4AH	TM1AL
23H	TMR	4BH	TM1AH
24H	—	4CH	TM1BL
25H	MISC	4DH	TM1BH
26H	DAL	4EH	SCOMC
27H	DAH	4FH	—
Bank0: 80H~FFH User data ram: 128*8 bit			

Data Memory

Special Function Register Description

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

Indirect Addressing Registers – IAR0, IAR1

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

Memory Pointers – MP0, MP1

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

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

The Data Memory is divided into several banks. Selecting the required Data Memory area is achieved using the Bank Pointer. The bits 0~1 of the Bank Pointer are used to select Data Memory Banks 0~2.

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

As both the Program Memory and Data Memory share the same Bank Pointer Register, care must be taken during programming.

Bit							
7	6	5	4	3	2	1	0
—	—	—	—	—	—	DMBP1	DMBP0

BP Registers List

- **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**: Select Data Memory Banks
00: Bank 0
01: Bank 1
10: Bank 2
11: Undefined

Accumulator – ACC

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

Program Counter Low Register – PCL

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

Look-up Table Registers – TBLP, TBHP, TBLH

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

Status Register – STATUS

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

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

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

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

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

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

- **STATUS Register**

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

"X" unknown

Bit 7, 6 Unimplemented, read as "0"

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

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

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

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

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

Bit 0 **C**: Carry flag
0: no carry-out
1: an operation results in a carry during an addition operation or if a borrow does not take place during a subtraction operation.
C is also affected by a rotate through carry instruction.

Timer Registers

This device contains one 8-bit Timer whose associated registers are known as TMR which is the location where the associated timer's 8-bit value is located. Its associated control register, known asTMRC, contains the setup information for this timer. This 8-bit timer can be used to provide the clock source for the Switch Capacitor Filter.

Note that all timer registers can be directly written to in order to preload their contents with fixed data to allow different time intervals to be setup.

Oscillator

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

Oscillator Overview

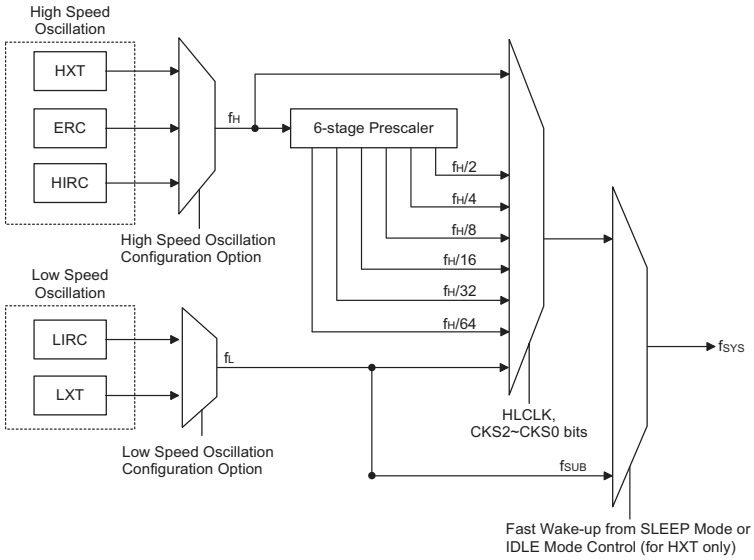
In addition to being the source of the main system clock the oscillators also provide clock sources for the Watchdog Timer and Time Base Interrupts. External oscillators requiring some external components as well as fully integrated internal oscillators, requiring no external components, are provided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

Type	Name	Freq.	Pins
External Crystal	HXT	400kHz~20MHz	OSC1/OSC2
External RC	ERC	8MHz	OSC1
Internal High Speed RC	HIRC	4, 8 or 12MHz	—
External Low Speed Crystal	LXT	32.768kHz	XT1/XT2
Internal Low Speed RC	LIRC	32kHz	—

Oscillator Types

System Clock Configurations

There are five methods of generating the system clock, three high speed oscillators and two low speed oscillators. The high speed oscillators are the external crystal/ceramic oscillator, external RC network oscillator and the internal 4MHz, 8MHz or 12MHz RC oscillator. The two low speed oscillators are the internal 32kHz RC oscillator and the external 32.768kHz crystal oscillator. Selecting whether the low or high speed oscillator is used as the system oscillator is implemented using the HLCLK bit and CKS2 ~ CKS0 bits in the SMOD register and as the system clock can be dynamically selected.

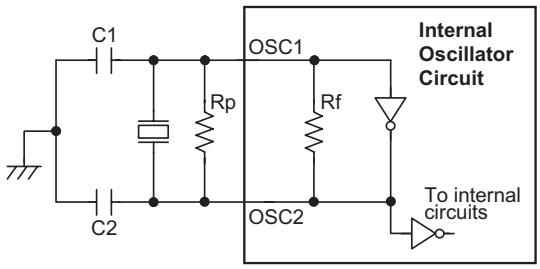


System Clock Configurations

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

External Crystal/ Ceramic Oscillator – HXT

The External Crystal/ Ceramic System Oscillator is one of the high frequency oscillator choices, which is selected via configuration option. For most crystal oscillator configurations, the simple connection of a crystal across OSC1 and OSC2 will create the necessary phase shift and feedback for oscillation, without requiring external capacitors. However, for some crystal types and frequencies, to ensure oscillation, it may be necessary to add two small value capacitors, C1 and C2. Using a ceramic resonator will usually require two small value capacitors, C1 and C2, to be connected as shown for oscillation to occur. The values of C1 and C2 should be selected in consultation with the crystal or resonator manufacturer's specification.



Note: 1. Rp is normally not required. C1 and C2 are required.
2. Although not shown OSC1/OSC2 pins have a parasitic capacitance of around 7pF.

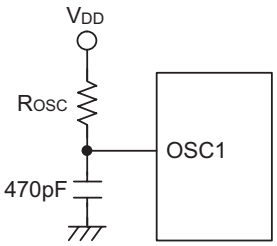
Crystal/Resonator Oscillator — HXT

Crystal Oscillator C1 and C2 Values		
Crystal Frequency	C1	C2
12MHz	0pF	0pF
8MHz	0pF	0pF
4MHz	0pF	0pF
1MHz	100pF	100pF
Note: C1 and C2 values are for guidance only.		

Crystal Recommended Capacitor Values

External RC Oscillator – ERC

Using the ERC oscillator only requires that a resistor, with a value between 56kΩ and 2.4MΩ, is connected between OSC1 and VDD, and a capacitor is connected between OSC1 and ground, providing a low cost oscillator configuration. It is only the external resistor that determines the oscillation frequency; the external capacitor has no influence over the frequency and is connected for stability purposes only. 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. As a resistance/frequency reference point, it can be noted that with an external 120kΩ resistor connected and with a 5V voltage power supply and temperature of 25°C degrees, the oscillator will have a frequency of 8MHz within a tolerance of 2%. Here only the OSC1 pin is used, which is shared with I/O pin PB1, leaving pin PB2 free for use as a normal I/O pin.



External RC Oscillator — ERC

Internal 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 either 4MHz, 8MHz or 12MHz. 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. As a result, at a power supply of either 3V or 5V and at a temperature of 25°C degrees, the fixed oscillation frequency of 4MHz, 8MHz or 12MHz will have a tolerance within 2%. Note that if this internal system clock option is selected, as it requires no external pins for its operation, I/O pins PB1 and PB2 are free for use as normal I/O pins.

External 32.768kHz Crystal Oscillator – LXT

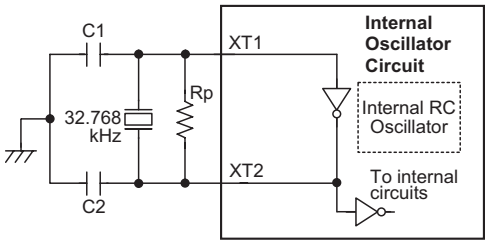
The External 32.768kHz Crystal System Oscillator is one of the low frequency oscillator choices, which is selected via configuration option. This clock source has a fixed frequency of 32.768kHz and requires a 32.768kHz crystal to be connected between pins XT1 and XT2. The external resistor and capacitor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

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

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

Some configuration options determine if the XT1/XT2 pins are used for the LXT oscillator or as I/O pins.

- If the LXT oscillator is not used for any clock source, the XT1/XT2 pins can be used as normal I/O pins.
- If the LXT oscillator is used for any clock source, the 32.768kHz crystal should be connected to the XT1/XT2 pins.



Note: 1. Rp, C1 and C2 are required.
2. Although not shown pins have a parasitic capacitance of around 7pF.

External LXT Oscillator

LXT Oscillator C1 and C2 Values		
Crystal Frequency	C1	C2
32.768kHz	10pF	10pF
Note:1. C1 and C2 values are for guidance only. 2. R_P=5M~10MΩ is recommended.		

32.768kHz Crystal Recommended Capacitor Values

LXT Oscillator Low Power Function

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

LXTLP Bit	LXT Mode
0	Quick Start
1	Low-power

After power on the LXTLP bit will be automatically cleared to zero ensuring that the LXT oscillator is in the Quick Start operating mode. In the Quick Start Mode the LXT oscillator will power up and stabilise quickly. However, after the LXT oscillator has fully powered up it can be placed into the Low-power mode by setting the LXTLP bit high. The oscillator will continue to run but with reduced current consumption, as the higher current consumption is only required during the LXT oscillator start-up. In power sensitive applications, such as battery applications, where power consumption must be kept to a minimum, it is therefore recommended that the application program sets the LXTLP bit high about 2 seconds after power-on.

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

Internal 32kHz Oscillator – LIRC

The Internal 32kHz System Oscillator is one of the low frequency oscillator choices, which is selected via configuration option. It is a fully integrated RC oscillator with a typical frequency of 32kHz at 5V, 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. As a result, at a power supply of 5V and at a temperature of 25°C degrees, the fixed oscillation frequency of 32kHz will have a tolerance within 10%.

Supplementary Oscillators

The low speed oscillators, in addition to providing a system clock source are also used to provide a clock source to two other device functions. These are the Watchdog Timer and the Time Base Interrupts.

Operating Modes and System Clocks

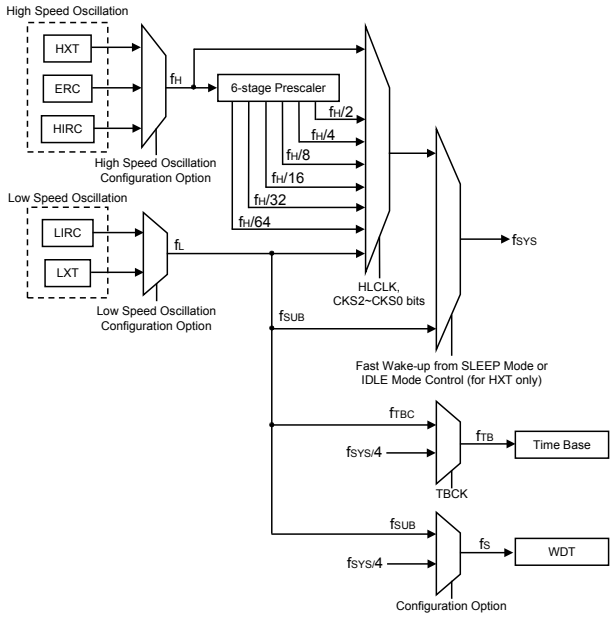
Present day applications require that their microcontrollers have high performance but often still demand that they consume as little power as possible, conflicting requirements that are especially true in battery powered portable applications. The fast clocks required for high performance will by their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

System Clocks

The device has many different clock sources for both the CPU and peripheral function operation. By providing the user with a wide range of clock options using configuration options and 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_L, source, and is selected using the HLCLK bit and CKS2~CKS0 bits in the SMOD register. The high speed system clock can be sourced from either an HXT, ERC or HIRC oscillator, selected via a configuration option. The low speed system clock source can be sourced from internal clock $f_{L.}$ If f_L is selected then it can be sourced by either the LXT or LIRC oscillators, selected via a configuration option. The other choice, which is a divided version of the high speed system oscillator has a range of $f_H/2$~$f_H/64$.

There are two additional internal clocks for the peripheral circuits, the substitute clock, f_{SUB}, and the Time Base clock, f_{TBC}. Each of these internal clocks are sourced by either the LXT or LIRC oscillators, selected via configuration options. The f_{SUB} clock is used to provide a substitute clock for the microcontroller just after a wake-up has occurred to enable faster wake-up times.



System Clock Configurations

Note: When the system clock source f_{SYS} is switched to f_L from f_H, the high speed oscillation will stop to conserve the power. Thus there is no f_H~$f_H/64$ for peripheral circuit to use.

Together with $f_{SYS}/4$ it is also used as one of the clock sources for the Watchdog timer. The f_{TBC} clock is used as a source for the Time Base interrupt functions and for the TMs.

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 NORMAL Mode and SLOW Mode. The remaining four modes, the SLEEP0, SLEEP1, IDLE0 and IDLE1 Mode are used when the microcontroller CPU is switched off to conserve power.

Operation Mode	Description				
	CPU	f_{SYS}	f_{SUB}	f_S	f_{TBC}
NORMAL Mode	On	f_H~ f_H/64	On	On	On
SLOW Mode	On	f_L	On	On	On
IDLE0 Mode	Off	Off	On	On/Off	On
IDLE1 Mode	Off	On	On	On	On
SLEEP0 Mode	Off	Off	Off	Off	Off
SLEEP1 Mode	Off	Off	On	On	Off

- NORMAL Mode
  As the name suggests this is one of the main operating modes where the microcontroller has all of its functions operational and where the system clock is provided by one of the high speed oscillators. This mode operates allowing the microcontroller to operate normally with a clock source will come from one of the high speed oscillators, either the HXT, ERC or HIRC oscillators. 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~LCKS0 and HLCLK bits in the SMOD 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 one of the low speed oscillators, either the LXT or the LIRC. Running the microcontroller in this mode allows it to run with much lower operating currents. In the SLOW Mode, the f_H is off.

- SLEEP0 Mode
  The SLEEP Mode is entered when an HALT instruction is executed and when the IDLEN bit in the SMOD register is low. In the SLEEP0 mode the CPU will be stopped, and the f_{SUB} and f_S clocks will be stopped too, and the Watchdog Timer function is disabled. In this mode, the LVDEN is must set to "0". If the LVDEN is set to "1", it won't enter the SLEEP0 Mode.

- SLEEP1 Mode
  The SLEEP Mode is entered when an HALT instruction is executed and when the IDLEN bit in the SMOD register is low. In the SLEEP1 mode the CPU will be stopped. However the f_{SUB} and f_S clocks will continue to operate if the LVDEN is "1" or the Watchdog Timer function is enabled and if its clock source is chosen via configuration option to come from the f_{SUB}.

- IDLE0 Mode
  The IDLE0 Mode is entered when a HALT instruction is executed and when the IDLEN bit in the SMOD register is high and the FSYSON bit in the WDTC register is low. In the IDLE0 Mode the system oscillator will be inhibited from driving the CPU but some peripheral functions will remain operational such as the Watchdog Timer, TMs and SIM. In the IDLE0 Mode, the system oscillator will be stopped. In the IDLE0 Mode the Watchdog Timer clock, f_S, will either be on or off depending upon the f_S clock source. If the source is f_{SYS}/4 then the f_S clock will be off, and if the source comes from f_{SUB} then f_S will be on.

- IDLE1 Mode
  The IDLE1 Mode is entered when an HALT instruction is executed and when the IDLEN bit in the SMOD register is high and the FSYSON bit in the WDTC register is high. In the IDLE1 Mode the system oscillator will be inhibited from driving the CPU but may continue to provide a clock source to keep some peripheral functions operational such as the Watchdog Timer, TMs, and SIM. In the IDLE1 Mode, the system oscillator will continue to run, and this system oscillator may be high speed or low speed system oscillator. In the IDLE1 Mode the Watchdog Timer clock, f_S, will be on. If the source is f_{SYS}/4 then the f_S clock will be on, and if the source comes from f_{SUB} then f_S will be on.

Control Register

A single register, SMOD, is used for overall control of the internal clocks within the device.

- **SMOD Register**

Bit	7	6	5	4	3	2	1	0
Name	CKS2	CKS1	CKS0	FSTEN	LTO	HTO	IDLEN	HLCLK
R/W	R/W	R/W	R/W	R/W	R	R	R/W	R/W
POR	0	0	0	0	0	0	1	1

Bit 7~5 **CKS2~CKS0**: The system clock selection when HLCLK is "0"
000: f_L (f_{LXT} or f_{LIRC})
001: f_L (f_{LXT} or f_{LIRC})
010: $f_H/64$
011: $f_H/32$
100: $f_H/16$
101: $f_H/8$
110: $f_H/4$
111: $f_H/2$
These three bits are used to select which clock is used as the system clock source. In addition to the system clock source, which can be either the LXT or LIRC, a divided version of the high speed system oscillator can also be chosen as the system clock source.

Bit 4 **FSTEN**: Fast Wake-up Control (only for HXT)
0: Disable
1: Enable
This is the Fast Wake-up Control bit which determines if the f_{SUB} clock source is initially used after the device wakes up. When the bit is high, the f_{SUB} clock source can be used as a temporary system clock to provide a faster wake up time as the f_{SUB} clock is available.

Bit 3 **LTO**: Low speed system oscillator ready flag
0: Not ready
1: Ready
This is the low speed system oscillator ready flag which indicates when the low speed system oscillator is stable after power on reset or a wake-up has occurred. The flag will be low when in the SLEEP0 Mode but after a wake-up has occurred, the flag will change to a high level after 1024 clock cycles if the LXT oscillator is used and 1~2 clock cycles if the LIRC oscillator is used.

Bit 2 **HTO**: High speed system oscillator ready flag
0: Not ready
1: Ready
This is the high speed system oscillator ready flag which indicates when the high speed system oscillator is stable. This flag is cleared to "0" by hardware when the device is powered on and then changes to a high level after the high speed system oscillator is stable. Therefore this flag will always be read as "1" by the application program after device power-on. The flag will be low when in the SLEEP or IDLE0 Mode but after a wake-up has occurred, the flag will change to a high level after 1024 clock cycles if the HXT oscillator is used and after 15~16 clock cycles if the ERC or HIRC oscillator is used.

Bit 1 **IDLEN**: IDLE Mode control
0: Disable
1: Enable

This is the IDLE Mode Control bit and determines what happens when the HALT instruction is executed. If this bit is high, when a HALT instruction is executed the device will enter the IDLE Mode. In the IDLE1 Mode the CPU will stop running but the system clock will continue to keep the peripheral functions operational, if FSYSON bit is high. If FSYSON bit is low, the CPU and the system clock will all stop in IDLE0 mode. If the bit is low the device will enter the SLEEP Mode when a HALT instruction is executed.

Bit 0 **HLCLK**: system clock selection
0: $f_H/2 \sim f_H/64$ or f_L
1: f_H
This bit is used to select if the f_H clock or the $f_H/2 \sim f_H/64$ or f_L clock is used as the system clock. When the bit is high the f_H clock will be selected and if low the $f_H/2 \sim f_H/64$ or f_L clock will be selected. When system clock switches from the f_H clock to the f_L clock and the f_H clock will be automatically switched off to conserve power.

Fast Wake – up

To minimise power consumption the device can enter the SLEEP or IDLE0 Mode, where the system clock source to the device will be stopped. However when the device is woken up again, it can take a considerable time for the original system oscillator to restart, stabilise and allow normal operation to resume. To ensure the device is up and running as fast as possible a Fast Wake-up function is provided, which allows f_{SUB}, namely either the LXT or LIRC oscillator, to act as a temporary clock to first drive the system until the original system oscillator has stabilised. As the clock source for the Fast Wake-up function is f_{SUB}, the Fast Wake-up function is only available in the SLEEP1 and IDLE0 modes. When the device is woken up from the SLEEP0 mode, the Fast Wake-up function has no effect because the f_{SUB} clock is stopped. The Fast Wake-up enable/disable function is controlled using the FSTEN bit in the SMOD register.

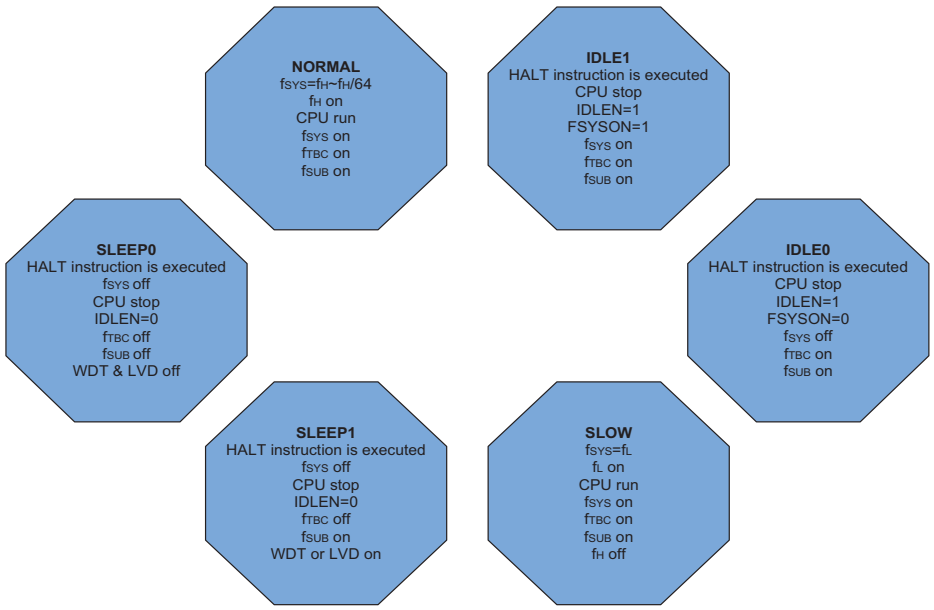
If the HXT oscillator is selected as the NORMAL Mode system clock, and if the Fast Wake-up function is enabled, then it will take one to two t_{SUB} clock cycles of the LIRC or LXT oscillator for the system to wake-up. The system will then initially run under the f_{SUB} clock source until 1024 HXT clock cycles have elapsed, at which point the HTO flag will switch high and the system will switch over to operating from the HXT oscillator.

If the ERC or HIRC oscillators or LIRC oscillator is used as the system oscillator then it will take 15~16 clock cycles of the ERC or HIRC or 1~2 cycles of the LIRC to wake up the system from the SLEEP or IDLE0 Mode. The Fast Wake-up bit, FSTEN will have no effect in these cases.

System Oscillator	FSTEN Bit	Wake-up Time (SLEEP0 Mode)	Wake-up Time (SLEEP1 Mode)	Wake-up Time (IDLE0 Mode)	Wake-up Time (IDLE1 Mode)
HXT	0	1024 HXT cycles	1024 HXT cycles		1~2 HXT cycles
	1	1024 HXT cycles	1~2 f_{SUB} cycles (System runs with f_{SUB} first for 1024 HXT cycles and then switches over to run with the HXT clock)		1~2 HXT cycles
ERC	X	15~16 ERC cycles	15~16 ERC cycles		1~2 ERC cycles
HIRC	X	15~16 HIRC cycles	15~16 HIRC cycles		1~2 HIRC cycles
LIRC	X	1~2 LIRC cycles	1~2 LIRC cycles		1~2 LIRC cycles
LXT	X	1024 LTX cycles	1024 LXT cycles		1~2 LXT cycles

Wake-Up Times

Note that if the Watchdog Timer is disabled, which means that the LXT and LIRC are all both off, then there will be no Fast Wake-up function available when the device wakes-up from the SLEEP0 Mode.



Operating Mode Switching and Wake-up

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.

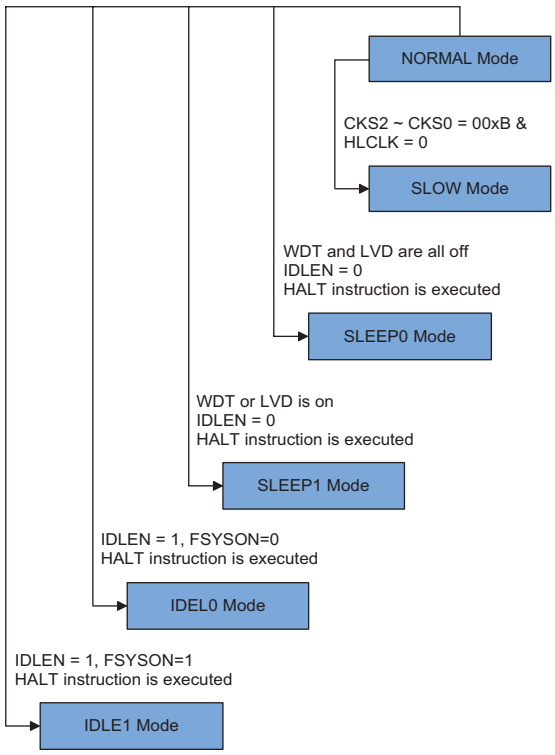
In simple terms, Mode Switching between the NORMAL Mode and SLOW Mode is executed using the HLCLK bit and CKS2~CKS0 bits in the SMOD register while Mode Switching from the NORMAL/SLOW Modes to the SLEEP/IDLE Modes is executed via the HALT instruction. When a HALT instruction is executed, whether the device enters the IDLE Mode or the SLEEP Mode is determined by the condition of the IDLEN bit in the SMOD register and FSYSON in the WDTC register.

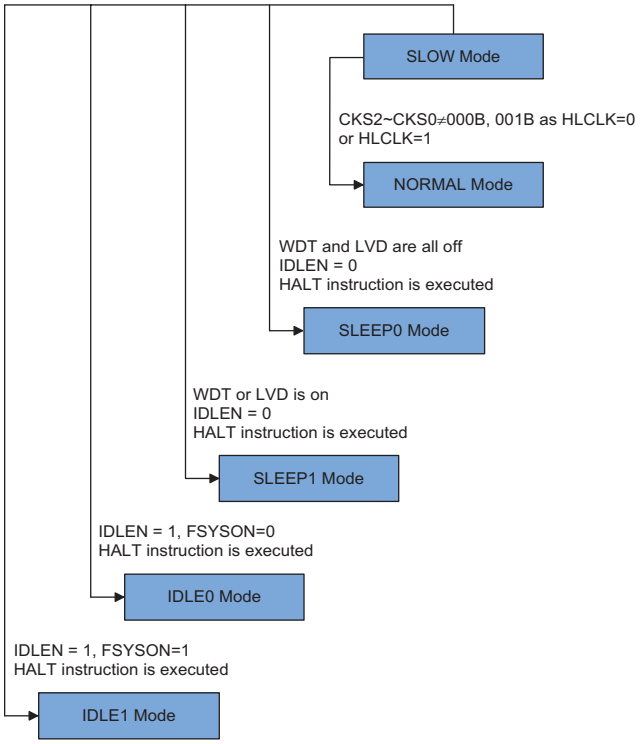
When the HLCLK bit switches to a low level, which implies that clock source is switched from the high speed clock source, f_H, to the clock source, $f_H/2$~$f_H/64$ or fL. If the clock is from the f_L, the high speed clock source will stop running to conserve power. When this happens it must be noted that the $f_H/16$ and $f_H/64$ internal clock sources will also stop running, which may affect the operation of other internal functions such as the TMs and the SIM. The accompanying flowchart shows what happens when the device moves between the various operating modes.

NORMAL Mode to SLOW Mode Switching

When running in the NORMAL Mode, which uses the high speed system oscillator, and therefore consumes more power, the system clock can switch to run in the SLOW Mode by set the HLCLK bit to "0" and set the CKS2~CKS0 bits to "000" or "001" in the SMOD register. This will then use the low speed system oscillator which will consume less power. Users may decide to do this for certain operations which do not require high performance and can subsequently reduce power consumption.

The SLOW Mode is sourced from the LXT or the LIRC oscillators and therefore requires these oscillators to be stable before full mode switching occurs. This is monitored using the LTO bit in the SMOD register.


NORMAL Mode
CKS2 ~ CKS0 = 00xB &
HLCLK = 0
SLOW Mode
WDT and LVD are all off
IDLEN = 0
HALT instruction is executed
SLEEP0 Mode
WDT or LVD is on
IDLEN = 0
HALT instruction is executed
SLEEP1 Mode
IDLEN = 1, FSYSON=0
HALT instruction is executed
IDEL0 Mode
IDLEN = 1, FSYSON=1
HALT instruction is executed
IDLE1 Mode


SLOW Mode
CKS2~CKS0≠000B, 001B as HLCLK=0
or HLCLK=1
NORMAL Mode
WDT and LVD are all off
IDLEN = 0
HALT instruction is executed
SLEEP0 Mode
WDT or LVD is on
IDLEN = 0
HALT instruction is executed
SLEEP1 Mode
IDLEN = 1, FSYSON=0
HALT instruction is executed
IDLE0 Mode
IDLEN = 1, FSYSON=1
HALT instruction is executed
IDLE1 Mode

SLOW Mode to NORMAL Mode Switching

In SLOW Mode the system uses either the LXT or LIRC low speed system oscillator. To switch back to the NORMAL Mode, where the high speed system oscillator is used, the HLCLK bit should be set to "1" or HLCLK bit is "0", but CKS2~CKS0 is set to "010", "011", "100", "101", "110" or "111". As a certain amount of time will be required for the high frequency clock to stabilise, the status of the HTO bit is checked. The amount of time required for high speed system oscillator stabilization depends upon which high speed system oscillator type is used.

Entering the SLEEP0 Mode

There is only one way for the device to enter the SLEEP0 Mode and that is to execute the "HALT" instruction in the application program with the IDLEN bit in SMOD register equal to "0" and the WDT and LVD both off. When this instruction is executed under the conditions described above, the following will occur:

- The system clock, WDT clock and Time Base 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 WDT will be cleared and stopped no matter if the WDT clock source originates from the f_{SUB} clock or from the system clock.
- The I/O ports will maintain their present conditions.
- In the status register, the Power Down flag, PDF, will be set and the Watchdog time-out flag, TO, will be cleared.

Entering the SLEEP1 Mode

There is only one way for the device to enter the SLEEP1 Mode and that is to execute the "HALT" instruction in the application program with the IDLEN bit in SMOD register equal to "0" and the WDT or LVD on. When this instruction is executed under the conditions described above, the following will occur:

- The system clock and Time Base clock will be stopped and the application program will stop at the "HALT" instruction, but the WDT or LVD will remain with the clock source coming from the f_{SUB} clock.
- The Data Memory contents and registers will maintain their present condition.
- The WDT will be cleared and resume counting if the WDT clock source is selected to come from the f_{SUB} clock as the WDT is enabled.
- The I/O ports will maintain their present conditions.
- In the status register, the Power Down flag, PDF, will be set and the Watchdog time-out flag, TO, will be cleared.

Entering the IDLE0 Mode

There is only one way for the device to enter the IDLE0 Mode and that is to execute the "HALT" nstruction in the application program with the IDLEN bit in SMOD register equal to "1" and the FSYSON bit in WDTC register equal to "0". 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, but the Time Base clock and f_{SUB} clock will be on.
- The Data Memory contents and registers will maintain their present condition.
- The WDT will be cleared and resume counting if the WDT clock source is selected to come from the f_{SUB} clock and the WDT is enabled. The WDT will stop if its clock source originates from the system clock.

- The I/O ports will maintain their present conditions.
- In the status register, the Power Down flag, PDF, will be set and the Watchdog time-out flag, TO, will be cleared.

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 the IDLEN bit in SMOD register equal to "1" and the FSYSON bit in WDTC register equal to "1". When this instruction is executed under the with conditions described above, the following will occur:

- The system clock and Time Base clock and f_{SUB} clock will be on and the application program will stop at the "HALT" instruction.
- The Data Memory contents and registers will maintain their present condition.
- The WDT will be cleared and resume counting if the WDT is enabled regardless of the WDT clock source which originates from the f_{SUB} clock or from the system clock.
- The I/O ports will maintain their present conditions.
- In the status register, the Power Down flag, PDF, will be set and the Watchdog time-out flag, TO, will be cleared.

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 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 devices which have different package types, as there may be unbonbed 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 configuration options have enabled the LXT or LIRC oscillator.

In the IDLE1 Mode the system oscillator is on, if the system oscillator is from the high speed system oscillator, the additional standby current will also be perhaps in the order of several hundred micro-amps

Wake-up

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

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

If the system is woken up by an external 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.

Programming Considerations

The HXT and LXT oscillators both use the same SST counter. For example, if the system is woken up from the SLEEP0 Mode and both the HXT and LXT oscillators need to start-up from an off state. The LXT oscillator uses the SST counter after HXT oscillator has finished its SST period.

- If the device is woken up from the SLEEP0 Mode to the NORMAL Mode, the high speed system oscillator needs an SST period. The device will execute first instruction after HTO is "1". At this time, the LXT oscillator may not be stability if f_{SUB} is from LXT oscillator. The same situation occurs in the power-on state. The LXT oscillator is not ready yet when the first instruction is executed.
- If the device is woken up from the SLEEP1 Mode to NORMAL Mode, and the system clock source is from HXT oscillator and FSTEN is "1", the system clock can be switched to the LXT or LIRC oscillator after wake up.
- There are peripheral functions, such as WDT, TMs and SIM, for which the f_{SYS} is used. If the system clock source is switched from f_H to f_L, the clock source to the peripheral functions mentioned above will change accordingly.
- The on/off condition of f_{SUB} and f_S depends upon whether the WDT is enabled or disabled as the WDT clock source is selected from f_{SUB}.

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_S, which is in turn supplied by one of two sources selected by configuration option: f_{SUB} or $f_{SYS}/4$. The f_{SUB} clock can be sourced from either the LXT or LIRC oscillators, again chosen via a configuration option. The Watchdog Timer source clock is then subdivided by a ratio of 2^8 to 2^{15} to give longer timeouts, the actual value being chosen using the WS2~WS0 bits in the WDTC register. The LIRC internal oscillator has an approximate period of 32kHz at a supply voltage of 5V.

However, it should be noted that this specified internal clock period can vary with V_{DD}, temperature and process variations. The LXT oscillator is supplied by an external 32.768kHz crystal. The other Watchdog Timer clock source option is the $f_{SYS}/4$ clock. The Watchdog Timer clock source can originate from its own internal LIRC oscillator, the LXT oscillator or $f_{SYS}/4$. It is divided by a value of 28 to 215, using the WS2~WS0 bits in the WDTC register to obtain the required Watchdog Timer time-out period.

Watchdog Timer Control Register

A single register, WDTC, controls the required timeout period as well as the enable/disable operation. This register together with several configuration options control the overall operation of the Watchdog Timer.

- **WDTC Register**

Bit	7	6	5	4	3	2	1	0
Name	FSYSON	WS2	WS1	WS0	WDTEN3	WDTEN2	WDTEN1	WDTEN0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	1	1	1	1	0	1	0

Bit 7 **FSYSON**: f_{SYS} Control in IDLE Mode
0: Disable
1: Enable

Bit 6 ~ 4 **WS2, WS1, WS0**: WDT time-out period selection
000: $256/f_S$
001: $512/f_S$
010: $1024/f_S$
011: $2048/f_S$
100: $4096/f_S$
101: $8192/f_S$
110: $16384/f_S$
111: $32768/f_S$
These three bits determine the division ratio of the Watchdog Timer source clock, which in turn determines the timeout period.

Bit 3 ~ 0 **WDTEN3, WDTEN2, WDTEN1, WDTEN0** : WDT Software Control
1010: Disable
Other: Enable

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 instructions. If the program malfunctions for whatever reason, jumps to an unkown location, or enters an endless loop, these clear instructions will not be executed in the correct manner, in which case the Watchdog Timer will overflow and reset the device. Some of the Watchdog Timer options, such as enable/disable, clock source selection and clear instruction type are selected using configuration options. In addition to a configuration option to enable/disable the Watchdog Timer, there are also four bits, WDTEN3~WDTEN0, in the WDTC register to offer an additional enable/disable control of the Watchdog Timer. To disable the Watchdog Timer, as well as the configuration option being set to disable, the WDTEN3~WDTEN0 bits must also be set to a specific value of "1010". Any other values for these bits will keep the Watchdog Timer enabled, irrespective of the configuration enable/disable setting. After power on these bits will have the value of 1010. If the Watchdog Timer is used it is recommended that they are set to a value of 0101 for maximum noise immunity. Note that if the Watchdog Timer has been disabled, then any instruction relating to its operation will result in no operation.

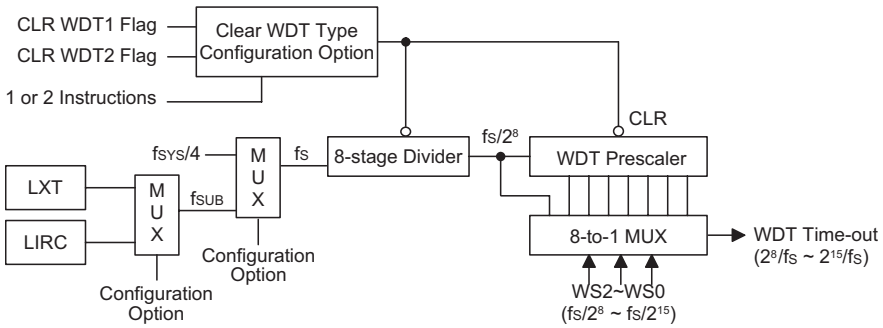
WDT Configuration Option	WDTEN3~WDTEN0 Bits	WDT
WDT Enable	xxxx	Enable
WDT Disable	Except 1010	Enable
WDT Disable	1010	Disable

Watchdog Timer Enable/Disable Control

Under normal program operation, a Watchdog Timer time-out will initialise a device reset and set the status bit TO. However, if the system is in the SLEEP or IDLE Mode, when a Watchdog Timer time-out occurs, the TO bit in the status register will be set and only the Program Counter and Stack Pointer will be reset. Three methods can be adopted to clear the contents of the Watchdog Timer. The first is an external hardware reset, which means a low level on the $\overline{\text{RES}}$ pin, the second is using the Watchdog Timer software clear instructions and the third is via a HALT instruction.

There are two methods of using software instructions to clear the Watchdog Timer, one of which must be chosen by configuration option. The first option is to use the single "CLR WDT" instruction while the second is to use the two commands "CLR WDT1" and "CLR WDT2". For the first option, a simple execution of CLR WDT will clear the WDT while for the second option, both "CLR WDT1" and "CLR WDT2" must both be executed alternately to successfully clear the Watchdog Timer. Note that for this second option, if "CLR WDT1" is used to clear the Watchdog Timer, successive executions of this instruction will have no effect, only the execution of a "CLR WDT2" instruction will clear the Watchdog Timer. Similarly after the "CLR WDT2" instruction has been executed, only a successive "CLR WDT1" instruction can clear the Watchdog Timer.

The maximum time out period is when the 2^{15} division ratio is selected. As an example, with a 32.768kHz LXT oscillator as its source clock, this will give a maximum watchdog period of around 1 second for the 2^{15} division ratio, and a minimum timeout of 7.8ms for the 2^8 division ration. If the $f_{SYS}/4$ clock is used as the Watchdog Timer clock source, it should be noted that when the system enters the SLEEP or IDLE0 Mode, then the instruction clock is stopped and the Watchdog Timer may lose its protecting purposes. For systems that operate in noisy environments, using the f_{SUB} clock source is strongly recommended.



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.

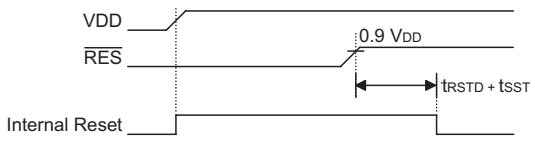
In addition to the power-on reset, situations may arise where it is necessary to forcefully apply a reset condition when the microcontroller is running. One example of this is where after power has been applied and the microcontroller is already running, the $\overline{RES}$ line is forcefully pulled 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 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. Another reset exists in the form of a Low Voltage Reset, LVR, where a full reset, similar to the $\overline{RES}$ reset is implemented in situations where the power supply voltage falls below a certain threshold.

Reset Functions

There are five 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.



Note: t_{RSTD} is power-on delay, typical time=100ms

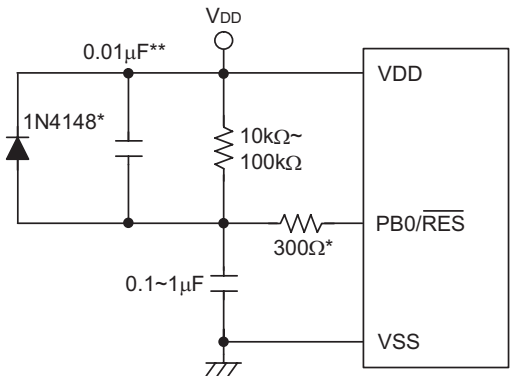
Power-On Reset Timing Chart

- $\overline{RES}$ Pin

  As the reset pin is shared with PB.0, the reset function must be selected using a configuration option. Although the microcontroller has an internal RC reset function, if the VDD 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 $\overline{RES}$ pin, whose additional time delay will ensure that the $\overline{RES}$ 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 $\overline{RES}$ line reaches a certain voltage value, the reset delay time t_{RSTD} is invoked to provide an extra delay time after which the microcontroller will begin normal operation. The abbreviation SST in the figures stands for System Start-up Timer.

  For most applications a resistor connected between VDD and the $\overline{RES}$ pin and a capacitor connected between VSS and the $\overline{RES}$ pin will provide a suitable external reset circuit. Any wiring connected to the $\overline{RES}$ 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.

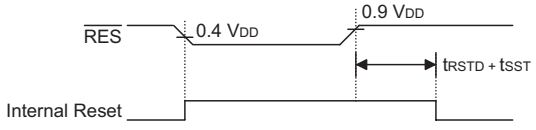


Note: * It is recommended that this component is added for added ESD protection.
** It is recommended that this component is added in environments where power line noise is significant.

External RES Circuit

More information regarding external reset circuits is located in Application Note HA0075E on the Holtek website.

Pulling the $\overline{RES}$ 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.

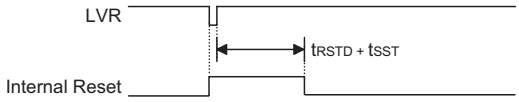


Note: t_{RSTD} is power-on delay, typical time=100ms

$\overline{RES}$ Reset Timing Chart

- Low Voltage Reset — LVR

  The microcontroller contains a low voltage reset circuit in order to monitor the supply voltage of the device, which is selected via a configuration option. If the supply voltage of the device drops to within a range of 0.9V~V_{LVR} such as might occur when changing the battery, the LVR will automatically reset the device internally. The LVR includes the following specifications: For a valid LVR signal, a low voltage, i.e., a voltage in the range between 0.9V~V_{LVR} must exist for greater than the value t_{LVR} specified in the A.C. characteristics. If the low voltage state does not exceed t_{LVR}, the LVR will ignore it and will not perform a reset function. One of a range of specified voltage values for V_{LVR} can be selected using configuration options.

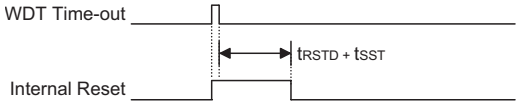


Note: t_{RSTD} is power-on delay, typical time=100ms

Low Voltage Reset Timing Chart

- Watchdog Time-out Reset during Normal Operation

  The Watchdog time-out Reset during normal operation is the same as a hardware $\overline{RES}$ pin reset except that the Watchdog time-out flag TO will be set to "1".

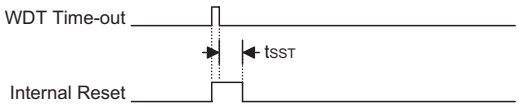


Note: t_{RSTD} is power-on delay, typical time=100ms

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 A.C. Characteristics for t_{SST} details.



Note: The t_{SST} is 15~16 clock cycles if the system clock source is provided by ERC or HIRC. The t_{SST} is 1024 clock for HXT or LXT. The t_{SST} is 1~2 clock for LIRC.

WDT Time-out Reset during SLEEP or IDLE Timing Chart

Reset Initial Conditions

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

TO	PDF	RESET Conditions
0	0	Power-on reset
u	u	$\overline{\text{RES}}$ or LVR reset during NORMAL or SLOW Mode operation
1	u	WDT time-out reset during NORMAL or SLOW Mode operation
1	1	WDT time-out reset during IDLE or SLEEP Mode operation

Note: "u" stands for unchanged

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

Item	Condition After RESET
Program Counter	Reset to zero
Interrupts	All interrupts will be disabled
WDT	Clear after reset, WDT begins counting
Timer/Event Counter	Timer Counter will be turned off
Input/Output Ports	I/O ports will be setup as inputs, and AN0~AN11 in as A/D input pin.
Stack Pointer	Stack Pointer will point to the top of the stack

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

- **HT83F22 Register**

Register	Reset (Power On)	$\overline{\text{RES}}$ or LVR Reset	WDT Time–out (Normal Operation)	WDT Time–out (Idle/Sleep)
MP0	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
MP1	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
BP	---- --00	---- --00	---- --00	---- --00
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
PCL	0000 0000	0000 0000	0000 0000	0000 0000
TBLP	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLH	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBHP	---- xxxx	---- uuuu	---- uuuu	---- uuuu
STATUS	--00 xxxx	--uu uuuu	--1u uuuu	--11 uuuu
SMOD	0000 0011	0000 0011	0000 0011	uuuu uuuu
INTEG	---- 0000	---- 0000	---- 0000	---- uuuu
LVDC	--00 -000	--00 -000	--00 -000	--uu -uuu
INTC0	-000 0000	-000 0000	-000 0000	-uuu uuuu
INTC1	0000 0000	0000 0000	0000 0000	uuuu uuuu
INTC2	0000 0000	0000 0000	0000 0000	uuuu uuuu
INTC3	---0 ---0	---0 ---0	---0 ---0	---u ---u
MFI0	--00 --00	--00 --00	--00 --00	--uu --uu

Register	Reset (Power On)	$\overline{\text{RES}}$ or LVR Reset	WDT Time–out (Normal Operation)	WDT Time–out (Idle/Sleep)
MFI1	–000 –000	–000 –000	–000 –000	–uuu –uuu
MFI2	––00 ––00	––00 ––00	––00 ––00	––uu ––uu
PA	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAC	1111 1111	1111 1111	1111 1111	uuuu uuuu
PB	1111 1111	1111 1111	1111 1111	uuuu uuuu
PBC	1111 1111	1111 1111	1111 1111	uuuu uuuu
PC	1111 1111	1111 1111	1111 1111	uuuu uuuu
PCC	1111 1111	1111 1111	1111 1111	uuuu uuuu
ADRL	xxxx ----	xxxx ----	xxxx ----	(ADRFS=0) uuuu ---- (ADRFS=1) uuuu uuuu
ADRH	xxxx xxxx	xxxx xxxx	xxxx xxxx	(ADRFS=0) uuuu uuuu (ADRFS=1) ---- uuuu
ADCR0	0110 –000	0110 –000	0110 –000	uuuu –uuu
ADCR1	00–0 –000	00–0 –000	00–0 –000	uu–u –uuu
ACER	1111 1111	1111 1111	1111 1111	uuuu uuuu
WDTC	0111 1010	0111 1010	0111 1010	uuuu uuuu
TBC	0011 –111	0011 –111	0011 –111	uuuu –uuu
SIM0C0	1110 000–	1110 000–	1110 000–	uuuu uuu–
SIM0C1	1000 0001	1000 0001	1000 0001	uuuu uuuu
SIM0D	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
SIM0A/SIM0C2	0000 0000	0000 0000	0000 0000	uuuu uuuu
SIM1C0	1110 000–	1110 000–	1110 000–	uuuu uuu–
SIM1C1	1000 0001	1000 0001	1000 0001	uuuu uuuu
SIM1D	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
SIM1A/SIM1C2	0000 0000	0000 0000	0000 0000	uuuu uuuu
PAWU	0000 0000	0000 0000	0000 0000	uuuu uuuu
PAPU	0000 0000	0000 0000	0000 0000	uuuu uuuu
PBPU	0000 0000	0000 0000	0000 0000	uuuu uuuu
PCPU	0000 0000	0000 0000	0000 0000	uuuu uuuu
CPC	1000 0––1	1000 0––1	1000 0––1	uuuu u––u
TMPC0	1001 ––01	1001 ––01	1001 ––01	uuuu ––uu
TM0C0	0000 0000	0000 0000	0000 0000	uuuu uuuu
TM0C1	0000 0000	0000 0000	0000 0000	uuuu uuuu
TM0DL	0000 0000	0000 0000	0000 0000	uuuu uuuu
TM0DH	---- ––00	---- ––00	---- ––00	---- ––uu
TM0AL	0000 0000	0000 0000	0000 0000	uuuu uuuu
TM0AH	---- ––00	---- ––00	---- ––00	---- ––uu
TM1C0	0000 0000	0000 0000	0000 0000	uuuu uuuu
TM1C1	0000 0000	0000 0000	0000 0000	uuuu uuuu
TM1C2	0000 0000	0000 0000	0000 0000	uuuu uuuu
TM1DL	0000 0000	0000 0000	0000 0000	uuuu uuuu
TM1DH	---- ––00	---- ––00	---- ––00	---- ––uu
TM1AL	0000 0000	0000 0000	0000 0000	uuuu uuuu
TM1AH	---- ––00	---- ––00	---- ––00	---- ––uu

Register	Reset (Power On)	$\overline{RES}$ or LVR Reset	WDT Time–out (Normal Operation)	WDT Time–out (Idle/Sleep)
TM1BL	0000 0000	0000 0000	0000 0000	uuuu uuuu
TM1BH	---- --00	---- --00	---- --00	---- --uu
TMR	0000 0000	0000 0000	0000 0000	uuuu uuuu
TMRC	00–0 1000	00–0 1000	00–0 1000	uu–u uuuu
MISC	000– --0–	000– --0–	000– --0–	uuu– --u–
DAL	xxxx ----	uuuu ----	uuuu ----	uuuu ----
DAH	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
VOL	xxx– ----	uuu– ----	uuu– ----	uuu– ----
SCOMC	0000 0000	0000 0000	0000 0000	uuuu uuuu
AGCC	000– –000	000– –000	000– –000	uuu– –uuu

Note: " * " stands for "warm reset",
" – " not implement
" u " stands for "unchanged"
" x " stands for "unknown"

Input/Output Ports

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

The device provides bidirectional input/output lines labeled with port names PA~PC 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.

- **I/O Register List**

Register Name	Bit							
	7	6	5	4	3	2	1	0
PAWU	D7	D6	D5	D4	D3	D2	D1	D0
PAPU	D7	D6	D5	D4	D3	D2	D1	D0
PA	D7	D6	D5	D4	D3	D2	D1	D0
PAC	D7	D6	D5	D4	D3	D2	D1	D0
PBPU	D7	D6	D5	D4	D3	D2	D1	D0
PB	D7	D6	D5	D4	D3	D2	D1	D0
PBC	D7	D6	D5	D4	D3	D2	D1	D0
PCPU	D7	D6	D5	D4	D3	D2	D1	D0
PC	D7	D6	D5	D4	D3	D2	D1	D0
PCC	D7	D6	D5	D4	D3	D2	D1	D0

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 an input have the capability of being connected to an internal pull-high resistor. These pull-high resistors are selected using registers PAPU~PCPU, and are implemented using weak PMOS transistors.

- **PAPU 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

- **PBPU 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

- **PCPU 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 I/O Port bit 7 ~ bit 0 Pull-High Control
0: Disable
1: Enable

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.

- **PAWU 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 **PAWU**: Port A bit 7 ~ bit 0 Wake-up Control
0: Disable
1: Enable

I/O Port Control Registers

Each I/O port has its own control register known as PAC~PCC, to control the input/output configuration. With this control register, each CMOS output or input can be reconfigured dynamically under software control. Each pin of the I/O ports is directly mapped to a bit in its associated port control register. For the I/O pin to function as an input, the corresponding bit of the control register must be written as a "1". This will then allow the logic state of the input pin to be directly read by instructions. When the corresponding bit of the control register is written as a "0",

the I/O pin will be setup as a CMOS output. If the pin is currently setup as an output, instructions can still be used to read the output register. However, it should be noted that the program will in fact only read the status of the output data latch and not the actual logic status of the output pin.

- **PAC 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	1	1	1	1	1	1	1	1

- **PBC 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	1	1	1	1	1	1	1	1

- **PCC 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	1	1	1	1	1	1	1	1

Bit 7~0 I/O Port bit 7 ~ bit 0 Input/Output Control
0: Output
1: Input

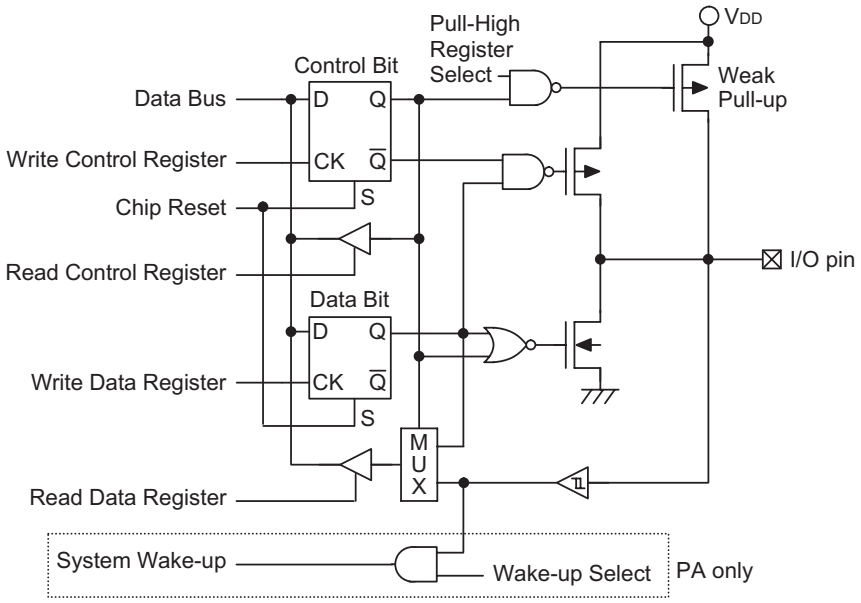
I/O Pin Structures

The accompanying diagrams illustrate the internal structures of some generic I/O pin types. As the exact logical construction of the I/O pin will differ from these drawings, they are supplied as a guide only to assist with the functional understanding of the I/O pins. The wide range of pin-shared structures does not permit all types to be shown.

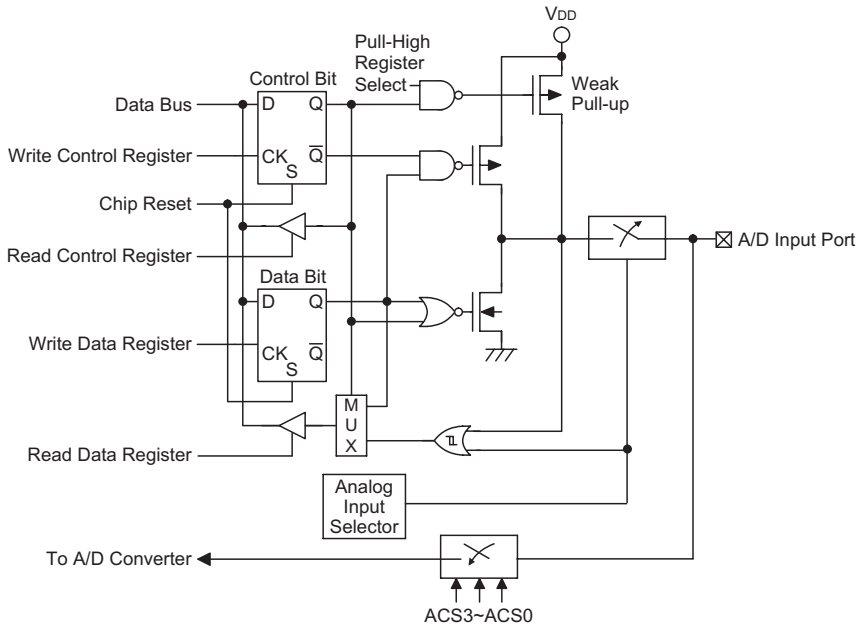
Programming Considerations

Within the user program, one of the first things to consider is port initialisation. After a reset, all of the I/O data and port control registers will be set high. This means that all I/O pins will default to an input state, the level of which depends on the other connected circuitry and whether pull-high selections have been chosen. If the port control registers, PAC~PCC, 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, PA~PC, 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.



Generic Input/Output Structure



A/D Input/Output Structure

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 each device includes several Timer Modules, abbreviated to the name TM. The TMs are multi-purpose timing units and serve to provide operations such as Timer/Counter, 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 either two or three individual interrupts. The addition of input and output pins for each TM ensures that users are provided with timing units with a wide and flexible range of features.

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

Introduction

The device contains two TMs, each TM having a reference name of TM0 and TM1. Each individual TM can be categorised as a certain type, namely Compact Type TM, or Enhanced Type TM. Although similar in nature, the different TM types vary in their feature complexity. The common features to all of the Compact and Enhanced TMs will be described in this section, 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.

Function	CTM	ETM
Timer/Counter	V	V
I/P Capture	—	V
Compare Match Output	V	V
PWM Channels	1	2
Single Pulse Output	—	1
PWM Alignment	Edge	Edge & Centre
PWM Adjustment Period & Duty	Duty or Period	Duty or Period

TM Function Summary

This device contains a Compact Type and an Enhanced Type TM which are shown in the table together with their individual reference name, TM0~TM1.

TM0	TM1
10-bit CTM	10-bit ETM

TM Name/Type Reference

TM Operation

The two different types of TM offer a diverse range of functions, from simple timing operations to PWM signal generation. The key to understand 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 TnCK2~TnCK0 bits in the TM control registers. The clock source can be a ratio of either the system clock f_{SYS} or the internal high clock fH, the fTBC clock source or the external TCKn pin. Note that setting these bits to the value 101 will select a reserved clock input, in effect disconnecting the TM clock source. The TCKn pin clock source is used to allow an external signal to drive the TM as an external clock source or for event counting.

TM Interrupts

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

TM External Pins

Each of the TMs, irrespective of what type, has one TM input pin, with the label TCKn. The TM input pin, is essentially a clock source for the TM and is selected using the TnCK2~TnCK0 bits in the TMnC0 register. This external TM input pin allows an external clock source to drive the internal TM. This external TM input pin is shared with other functions but will be connected to the internal TM if selected using the TnCK2~TnCK0 bits. The TM input pin can be chosen to have either a rising or falling active edge.

The TMs each have one or more output pins with the label TPn. When the TM is in the Compare Match Output Mode, these pins can be controlled by the TM to switch to a high or low level or to toggle when a compare match situation occurs. The external TPn output pin is also the pin where the TM generates the PWM output waveform. As the TM output pins are pin-shared with other function, the TM output function must first be setup using registers. A single bit in one of the registers determines if its associated pin is to be used as an external TM output pin or if it is to have another function. The number of output pins for each TM type and device is different, the details are provided in the accompanying table.

CTM	ETM	Registers
TP0_0, TP0_1	TP1A, TP1B_0, TP1B_1, TP1B_2	TMPC0

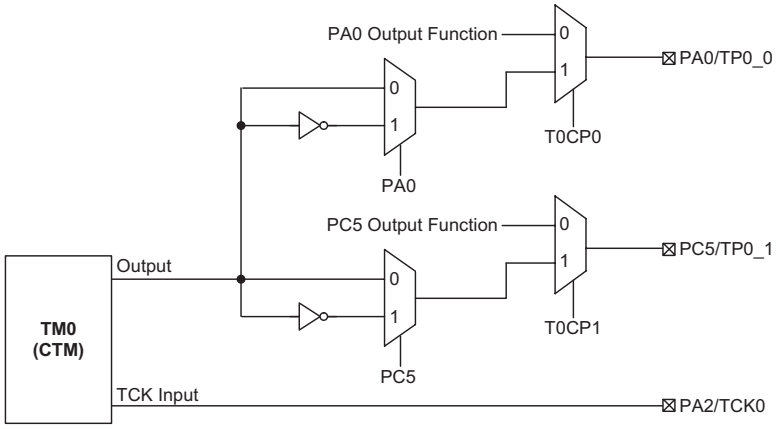
TM Output Pins

TM Input/Output Pin Control Registers

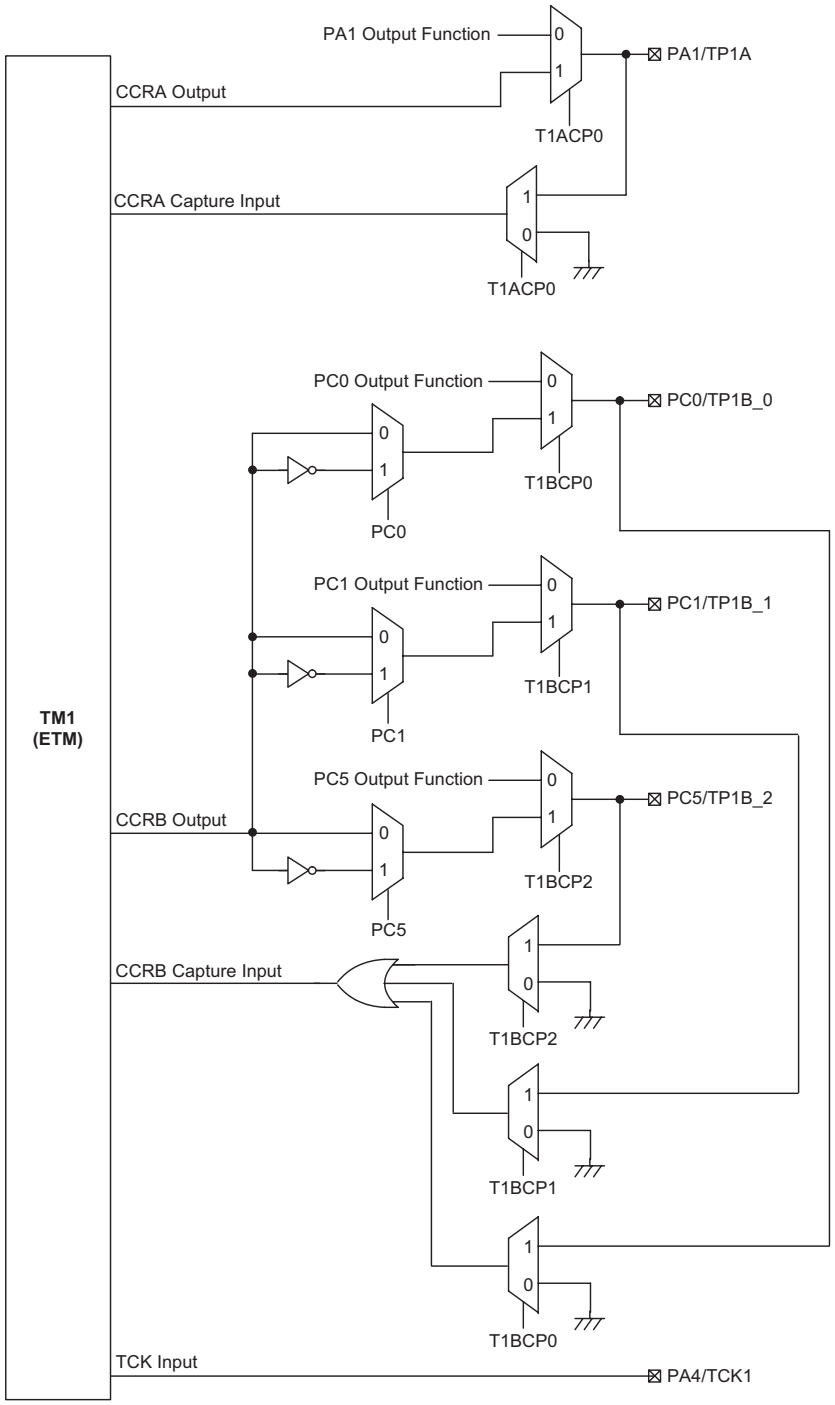
Selecting to have a TM input/output or whether to retain its other shared function, is implemented using one register, with a single bit in each register corresponding to a TM input/output pin. Setting the bit high will setup the corresponding pin as a TM input/output, if reset to zero the pin will retain its original other function.

Registers	Bit							
	7	6	5	4	3	2	1	0
TMPC0	T1ACP0	T1BCP2	T1BCP1	T1BCP0	—	—	T0CP1	T0CP0

TM Input/Output Pin Control Registers List



TM0 Function Pin Control Block Diagram



TM1 Function Pin Control Block Diagram

Bit	7	6	5	4	3	2	1	0
Name	T1ACP0	T1BCP2	T1BCP1	T1BCP0	—	—	T0CP1	T0CP0
R/W	R/W	R/W	R/W	R/W	—	—	R/W	R/W
POR	1	0	0	1	—	—	0	1

Bit 7 **T1ACP0**: TP1A pin Control
0: disable
1: enable

Bit 6 **T1BCP2**: TP1B_2 pin Control
0: disable
1: enable

Bit 5 **T1BCP1**: TP1B_1 pin Control
0: disable
1: enable

Bit 4 **T1BCP0**: TP1B_0 pin Control
0: disable
1: enable

Bit 3~2 Unimplemented, read as "0"

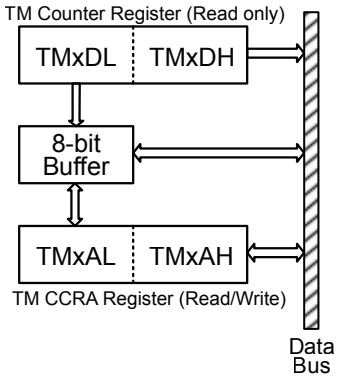
Bit 1 **T0CP1**: TP0_1 pin Control
0: disable
1: enable

Bit 0 **T0CP0**: TP0_0 pin Control
0: disable
1: enable

Programming Considerations

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

As the CCRA register is implemented in the way shown in the following diagram and accessing this register is carried out in a specific way described above, it is recommended to use the "MOV" instruction to access the CCRA low byte register, named TMxAL, in the following access procedures. Accessing the CCRA low byte register without following the access procedures will result in unpredictable value.



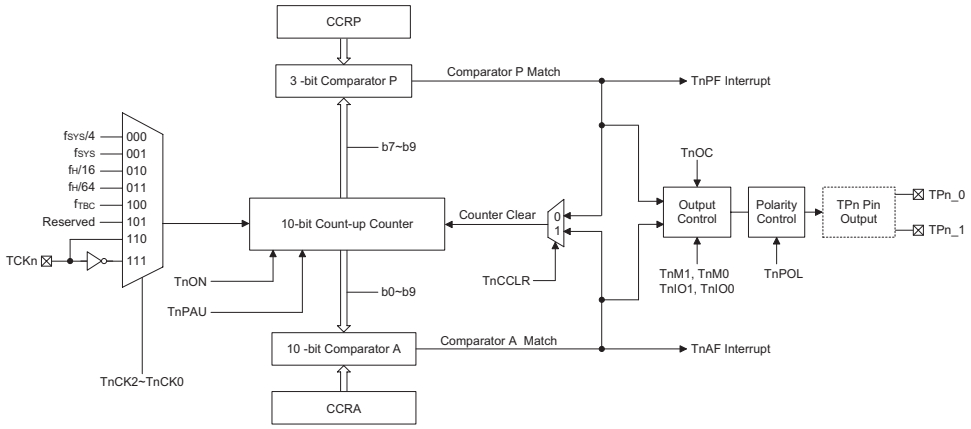
The following steps show the read and write procedures:

- Writing Data to CCRA
  - ♦ Step 1. Write data to Low Byte TMxAL
    - – note that here data is only written to the 8-bit buffer.
  - ♦ Step 2. Write data to High Byte TMxAH
    - – here data is written directly to the high byte registers and simultaneously data is latched from the 8-bit buffer to the Low Byte registers.
- Reading Data from the Counter Registers and CCRA
  - ♦ Step 1. Read data from the High Byte TMxDH, TMxAH
    - – 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 TMxDL, TMxAL
    - – this step reads data from the 8-bit buffer.

Compact Type TM

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

Name	TM No.	TM Input Pin	TM Output Pin
10-bit CTM	0	TCK0	TP0_0, TP0_1



Compact Type TM Block Diagram

Compact TM Operation

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

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

Compact Type TM Register Description

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

Name	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TM0C0	T0PAU	T0CK2	T0CK1	T0CK0	T0ON	T0RP2	T0RP1	T0RP0
TM0C1	T0M1	T0M0	T0IO1	T0IO0	T0OC	T0POL	T0DPX	T0CCLR
TM0DL	D7	D6	D5	D4	D3	D2	D1	D0
TM0DH	—	—	—	—	—	—	D9	D8
TM0AL	D7	D6	D5	D4	D3	D2	D1	D0
TM0AH	—	—	—	—	—	—	D9	D8

Compact TM Register List (if CTM is TM0)

- **TM0DL 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 **TM0DL**: TM0 Counter Low Byte Register bit 7 ~ bit 0
TM0 10-bit Counter bit 7 ~ bit 0

- **TM0DH 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 **TM0DH**: TM0 Counter High Byte Register bit 1 ~ bit 0
TM0 10-bit Counter bit 9 ~ bit 8

- **TM0AL 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 **TM0AL**: TM0 CCRA Low Byte Register bit 7 ~ bit 0
TM0 10-bit CCRA bit 7 ~ bit 0

- **TM0AH 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 **TM0AH**: TM0 CCRA High Byte Register bit 1 ~ bit 0
TM0 10-bit CCRA bit 9 ~ bit 8

• **TM0C0 Register**

Bit	7	6	5	4	3	2	1	0
Name	T0PAU	T0CK2	T0CK1	T0CK0	T0ON	T0RP2	T0RP1	T0RP0
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 **T0PAU**: TM0 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 TM 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 **T0CK2~T0CK0**: Select TM0 Counter clock
000: $f_{SYS}/4$
001: f_{SYS}
010: $f_H/16$
011: $f_H/64$
100: f_{TBC}
101: Reserved
110: TCK0 rising edge clock
111: TCK0 falling edge clock

These three bits are used to select the clock source for the TM. Selecting the Reserved clock input will effectively disable the internal counter. 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_{TBC} are other internal clocks, the details of which can be found in the oscillator section.

Bit 3 **T0ON**: TM0 Counter On/Off Control
0: Off
1: On

This bit controls the overall on/off function of the TM. Setting the bit high enables the counter to run, clearing the bit disables the TM. Clearing this bit to zero will stop the counter from counting and turn off the TM 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. If the TM is in the Compare Match Output Mode then the TM output pin will be reset to its initial condition, as specified by the T0OC bit, when the T0ON bit changes from low to high.

Bit 2~0 **T0RP2~T0RP0**: TM0 CCRP 3-bit register, compared with the TM0 Counter bit 9~bit 7 Comparator P Match Period
000: 1024 TM0 clocks
001: 128 TM0 clocks
010: 256 TM0 clocks
011: 384 TM0 clocks
100: 512 TM0 clocks
101: 640 TM0 clocks
110: 768 TM0 clocks
111: 896 TM0 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 T0CCLR bit is set to zero. Setting the T0CCLR 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.

• **TM0C1 Register**

Bit	7	6	5	4	3	2	1	0
Name	T0M1	T0M0	T0IO1	T0IO0	T0OC	T0POL	T0DPX	T0CCLR
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 **T0M1~T0M0**: Select TM0 Operating Mode
00: Compare Match Output Mode
01: Undefined Mode
10: PWM Mode
11: Timer/Counter Mode

These bits setup the required operating mode for the TM. To ensure reliable operation the TM should be switched off before any changes are made to the T0M1 and T0M0 bits. In the Timer/Counter Mode, the TM output pin control must be disabled.

Bit 5~4 **T0IO1~T0IO0**: Select TP0_0, TP0_1 output function

Compare Match Output Mode
00: No change
01: Output low
10: Output high
11: Toggle output

PWM Mode
00: Force inactive state
01: Force active state
10: PWM output
11: Undefined

Timer/counter Mode
unused

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

In the Compare Match Output Mode, the T0IO1 and T0IO0 bits determine how the TM output pin changes state when a compare match occurs from the Comparator A. The TM output pin can be setup to switch high, switch low or to toggle its present state when a compare match occurs from the Comparator A. When the bits are both zero, then no change will take place on the output. The initial value of the TM output pin should be setup using the T0OC bit in the TM0C1 register. Note that the output level requested by the T0IO1 and T0IO0 bits must be different from the initial value setup using the T0OC bit otherwise no change will occur on the TM output pin when a compare match occurs. After the TM output pin changes state it can be reset to its initial level by changing the level of the T0ON bit from low to high.

Bit 3 **T0OC**: TP0_0, TP0_1 Output control bit

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

PWM Mode
0: Active low
1: Active high

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

Bit 2 **T0POL**: TP0_0, TP0_1 Output polarity Control
0: Non-invert
1: Invert
This bit controls the polarity of the TP0_0 or TP0_1 output pin. When the bit is set high the TM output pin will be inverted and not inverted when the bit is zero. It has no effect if the TM is in the Timer/Counter Mode.

Bit 1 **T0DPX**: TM0 PWM period/duty Control
0: CCRP - period; CCRA - duty
1: CCRP - duty; CCRA - period
This bit, determines which of the CCRA and CCRP registers are used for period and duty control of the PWM waveform.

Bit 0 **T0CCLR**: Select TM0 Counter clear condition
0: TM0 Comparatror P match
1: TM0 Comparatror A match
This bit is used to select the method which clears the counter. Remember that the Compact TM contains two comparators, Comparator A and Comparator P, either of which can be selected to clear the internal counter. With the T0CCLR 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 T0CCLR bit is not used in the PWM Mode.

Compact Type TM Operating Modes

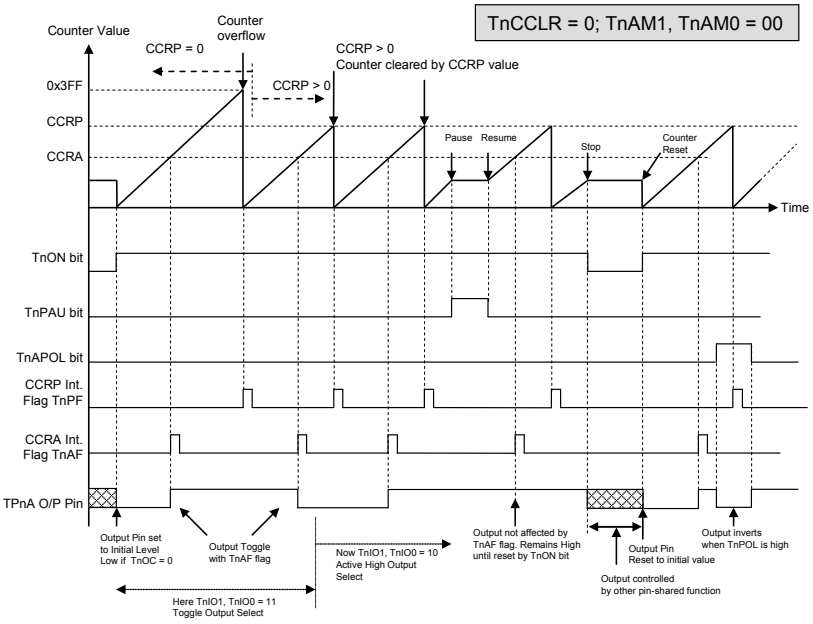
The Compact Type TM can operate in one of three operating modes, Compare Match Output Mode, PWM Mode or Timer/Counter Mode. The operating mode is selected using the TnM1 and TnM0 bits in the TMnC1 register.

Compare Match Output Mode

To select this mode, bits TnM1 and TnM0 in the TMnC1 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 TnCCLR bit is low, there are two ways in which the counter can be cleared. One is when a compare match occurs from Comparator P, the other is when the CCRP bits are all zero which allows the counter to overflow. Here both TnAF and TnPF interrupt request flags for the Comparator A and Comparator P respectively, will both be generated.

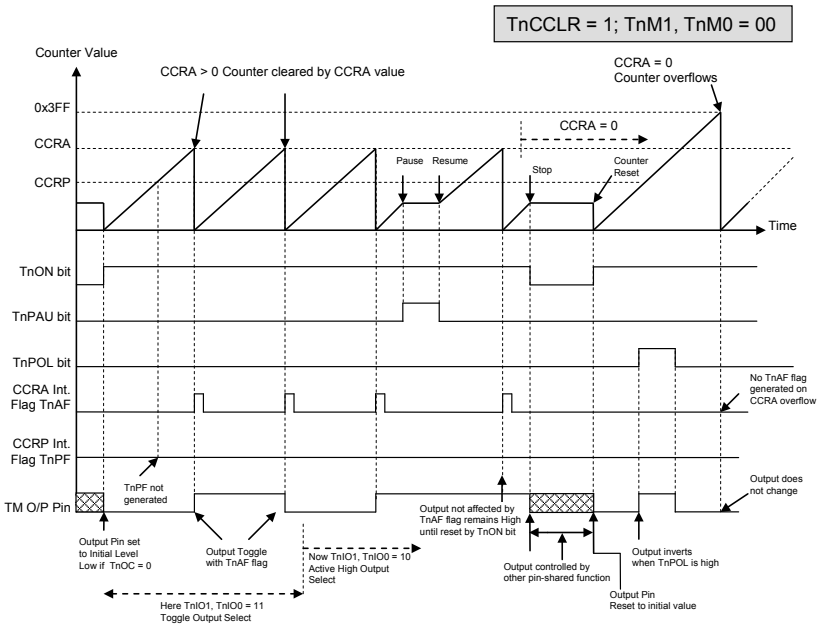
If the TnCCLR bit in the TMnC1 register is high then the counter will be cleared when a compare match occurs from Comparator A. However, here only the TnAF interrupt request flag will be generated even if the value of the CCRP bits is less than that of the CCRA registers. Therefore when TnCCLR is high no TnPF interrupt request flag will be generated. If the CCRA bits are all zero, the counter will overflow when its reaches its maximum 10-bit, 3FF Hex, value, however here the TnAF interrupt request flag will not be generated.

As the name of the mode suggests, after a comparison is made, the TM output pin will change state. The TM output pin condition however only changes state when a TnAF interrupt request flag is generated after a compare match occurs from Comparator A. The TnPF interrupt request flag, generated from a compare match occurs from Comparator P, will have no effect on the TM output pin. The way in which the TM output pin changes state are determined by the condition of the TnIO1 and TnIO0 bits in the TMnC1 register. The TM output pin can be selected using the TnIO1 and TnIO0 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 TM output pin, which is setup after the TnON bit changes from low to high, is setup using the TnOC bit. Note that if the TnIO1 and TnIO0 bits are zero then no pin change will take place.



Compare Match Output Mode — TnCCLR = 0

Note: 1. With TnCCLR = 0 the Comparator P match will clear the counter
2. TM output pin controlled only by TnAF flag
3. Output pin reset to initial state by TnON bit rising edge



Compare Match Output Mode — TnCCLR = 1

Note: 1. With TnCCLR = 1 the Comparator A match will clear the counter
2. TM output pin controlled only by TnAF flag
3.TM output pin reset to initial state by TnON rising edge
4. TnPF flags not generated when TnCCLR = 1

Timer/Counter Mode

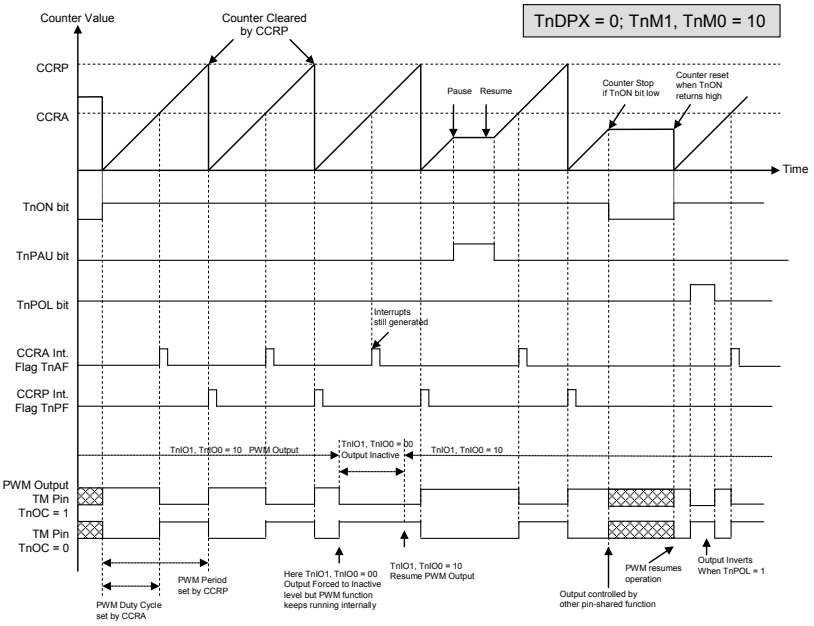
To select this mode, bits TnM1 and TnM0 in the TMnC1 register should be set to 11 respectively. The Timer/Counter Mode operates in an identical way to the Compare Match Output Mode generating the same interrupt flags. The exception is that in the Timer/Counter Mode the TM output pin is not used. Therefore the above description and Timing Diagrams for the Compare Match Output Mode can be used to understand its function. As the TM 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 TnM1 and TnM0 in the TMnC1 register should be set to 10 respectively. The PWM function within the TM 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 TM output pin, a square wave AC waveform can be generated with varying equivalent DC RMS values.

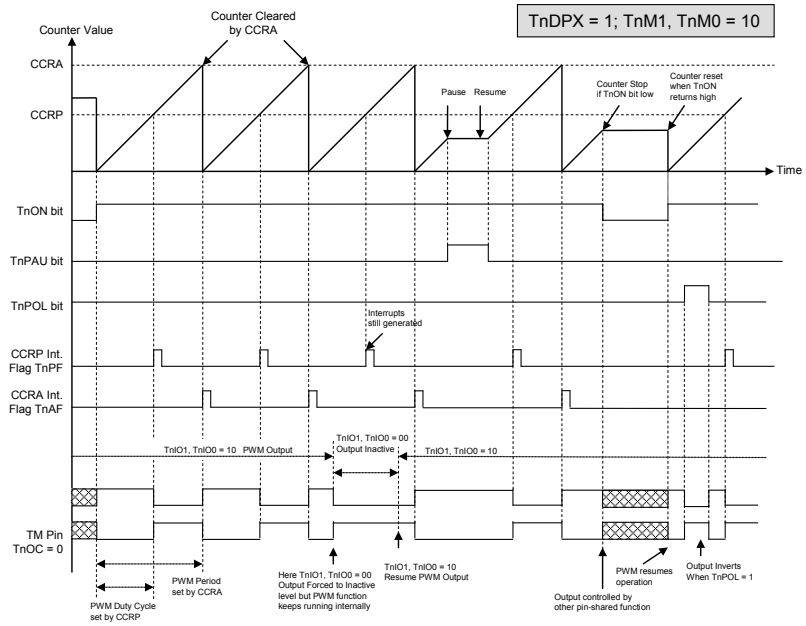
As both the period and duty cycle of the PWM waveform can be controlled, the choice of generated waveform is extremely flexible. In the PWM mode, the TnCCLR bit has no effect on the PWM operation. Both of the CCRA and CCRP registers are used to generate the PWM waveform, one register is used to clear the internal counter and thus control the PWM waveform frequency, while the other one is used to control the duty cycle. Which register is used to control either frequency or duty cycle is determined using the TnDPX bit in the TMnC1 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 TnOC bit in the TMnC1 register is used to select the required polarity of the PWM waveform while the two TnIO1 and TnIO0 bits are used to enable the PWM output or to force the TM output pin to a fixed high or low level. The TnPOL bit is used to reverse the polarity of the PWM output waveform.



PWM Mode — TnDPX = 0

Note: 1. Here TnDPX = 0 - Counter cleared by CCRP
2. Counter Clear sets PWM Period
3. Internal PWM function continues even when TnIO1, TnIO0 = 00 or 01
4. TnCCLR bit has no influence on PWM operation



PWM Mode — TnDPX = 1

Note: 1. Here TnDPX = 1 - Counter cleared by CCRA
2. Counter Clear sets PWM Period
3. Internal PWM function continues even when TnIO1, TnIO0 = 00 or 01
4. TnCCLR bit has no influence on PWM operation

Enhanced Type TM – ETM

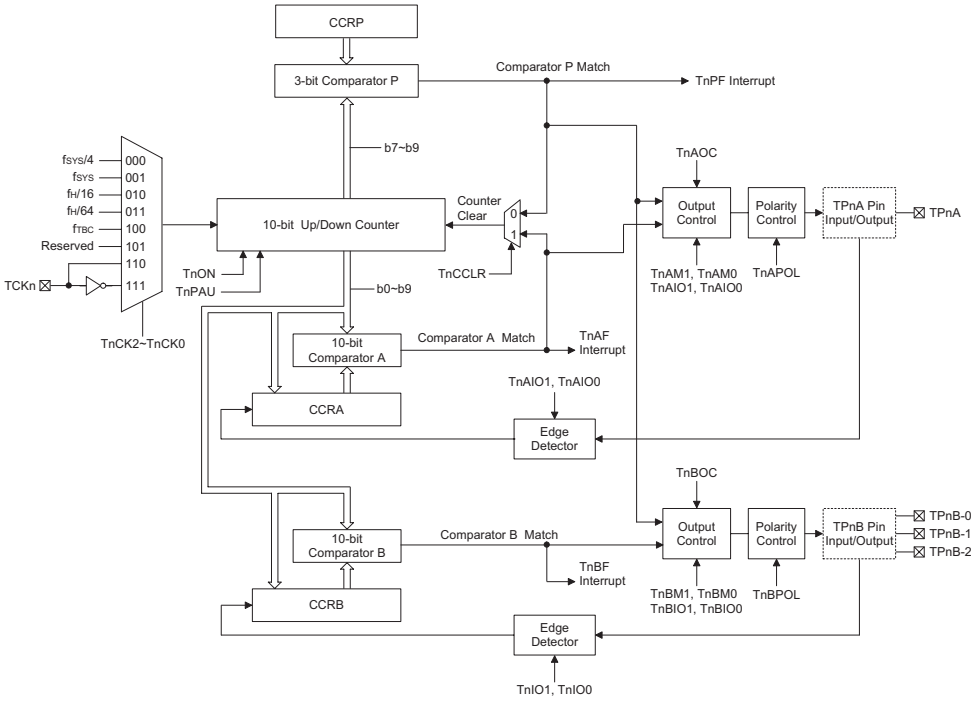
The Enhanced Type TM contains five operating modes, which are Compare Match Output, Timer/Event Counter, Capture Input, Single Pulse Output and PWM Output modes. The Enhanced TM can also be controlled with an external input pin and can drive three or four external output pins.

CTM	Name	TM No.	TM Input Pin	TM Output Pin
HT83F22	10-bit ETM	1	TCK1	TP1A, TP1B_0, TP1B_1, TP1B_2

Enhanced TM Operation

At its core is a 10-bit count-up/count-down counter which is driven by a user selectable internal or external clock source. There are three internal comparators with the names, Comparator A, Comparator B and Comparator P. These comparators will compare the value in the counter with the CCRA, CCRB and CCRP registers. The CCRP comparator is 3-bits wide whose value is compared with the highest 3-bits in the counter while CCRA and CCRB are 10-bits wide and therefore 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 TnON 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 TM interrupt signal will also usually be generated. The Enhanced 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 output pins. All operating setup conditions are selected using relevant internal registers.



Enhanced Type TM Block Diagram

Enhanced Type TM Register Description

Overall operation of the Enhanced 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 CCRB value. The remaining three registers are control registers which setup the different operating and control modes as well as the three CCRP bits.

Name	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TM1C0	T1PAU	T1CK2	T1CK1	T1CK0	T1ON	T1RP2	T1RP1	T1RP0
TM1C1	T1AM1	T1AM0	T1AIO1	T1AIO0	T1AOC	T1APOL	T1CDN	T1CCLR
TM1C2	T1BM1	T1BM0	T1BIO1	T1BIO0	T1BOC	T1BPOL	T1PWM1	T1PWM0
TM1DL	D7	D6	D5	D4	D3	D2	D1	D0
TM1DH	—	—	—	—	—	—	D9	D8
TM1AL	D7	D6	D5	D4	D3	D2	D1	D0
TM1AH	—	—	—	—	—	—	D9	D8
TM1BL	D7	D6	D5	D4	D3	D2	D1	D0
TM1BH	—	—	—	—	—	—	D9	D8

10-bit Enhanced TM Register List (if ETM is TM1)

10-bit Enhanced TM Register List

- **TM1C0 Register – 10-bit ETM**

Bit	7	6	5	4	3	2	1	0
Name	T1PAU	T1CK2	T1CK1	T1CK0	T1ON	T1RP2	T1RP1	T1RP0
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 **T1PAU**: TM1 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 TM 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 **T1CK2~T1CK0**: Select TM1 Counter clock
000: $f_{SYS}/4$
001: f_{SYS}
010: $f_H/16$
011: $f_H/64$
100: f_{TBC}
101: Reserved
110: TCK1 rising edge clock
111: TCK1 falling edge clock

These three bits are used to select the clock source for the TM. Selecting the Reserved clock input will effectively disable the internal counter. 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_{TBC} are other internal clocks, the details of which can be found in the oscillator section.

Bit 3 **T1ON**: TM1 Counter On/Off Control
0: Off
1: On

This bit controls the overall on/off function of the TM. Setting the bit high enables the counter to run, clearing the bit disables the TM. Clearing this bit to zero will stop the counter from counting and turn off the TM 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 TM is in the Compare Match Output Mode then the TM output pin will be reset to its initial condition, as specified by the T1OC bit, when the T1ON bit changes from low to high.

Bit 2~0 **T1RP2~T1RP0**: TM1 CCRP 3-bit register, compared with the TM1 Counter bit 9~bit 7

Comparator P Match Period
000: 1024 TM1 clocks
001: 128 TM1 clocks
010: 256 TM1 clocks
011: 384 TM1 clocks
100: 512 TM1 clocks
101: 640 TM1 clocks
110: 768 TM1 clocks
111: 896 TM1 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 T1CCLR bit is set to zero. Setting the T1CCLR 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.

- **TM1C1 Register – 10-bit ETM**

Bit	7	6	5	4	3	2	1	0
Name	T1AM1	T1AM0	T1AIO1	T1AIO0	T1AOC	T1APOL	T1CDN	T1CCLR
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 **T1AM1~T1AM0**: Select TM1 CCRA Operating Mode
00: Compare Match Output Mode
01: Capture Input Mode
10: PWM Mode or Single Pulse Output Mode
11: Timer/Counter Mode

These bits setup the required operating mode for the TM. To ensure reliable operation the TM should be switched off before any changes are made to the T1AM1 and T1AM0 bits. In the Timer/Counter Mode, the TM output pin control must be disabled.

Bit 5~4 **T1AIO1~T1AIO0**: Select TP1A output function

Compare Match Output Mode
00: No change
01: Output low
10: Output high
11: Toggle output

PWM Mode/ Single Pulse Output Mode
00: Force inactive state
01: Force active state
10: PWM output
11: Single pulse output

Capture Input Mode
00: Input capture at rising edge of TP1A
01: Input capture at falling edge of TP1A
10: Input capture at falling/rising edge of TP1A
11: Input capture disabled

Timer/counter Mode
Unused

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

In the Compare Match Output Mode, the T1AIO1 and T1AIO0 bits determine how the TM output pin changes state when a compare match occurs from the Comparator A. The TM output pin can be setup to switch high, switch low or to toggle its present state when a compare match occurs from the Comparator A. When the bits are both zero, then no change will take place on the output. The initial value of the TM output pin should be setup using the T1AOC bit in the TM1C1 register. Note that the output level requested by the T1AIO1 and T1AIO0 bits must be different from the initial value setup using the T1AOC bit otherwise no change will occur on the TM output pin when a compare match occurs. After the TM output pin changes state it can be reset to its initial level by changing the level of the T1ON bit from low to high.

Bit 3 **T1AOC**: TP1A Output control bit

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

PWM Mode/ Single Pulse Output Mode
0: Active low
1: Active high

This is the output control bit for the TM output pin. Its operation depends upon whether TM is being used in the Compare Match Output Mode or in the PWM Mode/ Single Pulse Output Mode.

It has no effect if the TM is in the Timer/Counter Mode. In the Compare Match Output Mode it determines the logic level of the TM output pin before a compare match occurs. In the PWM Mode it determines if the PWM signal is active high or active low.

Bit 2 **T1APOL**: TP1A Output polarity Control
0: Non-invert
1: Invert

This bit controls the polarity of the TP1A output pin. When the bit is set high the TM output pin will be inverted and not inverted when the bit is zero. It has no effect if the TM is in the Timer/Counter Mode.

Bit 1 **T1CDN**: TM1 Counter count up or down flag
0: Count up
1: Count down

Bit 0 **T1CCLR**: Select TM1 Counter clear condition
0: TM1 Comparator P match
1: TM1 Comparator A match

This bit is used to select the method which clears the counter. Remember that the Enhanced TM contains two comparators, Comparator A and Comparator P, either of which can be selected to clear the internal counter. With the T1CCLR 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 T1CCLR bit is not used in the PWM, Single Pulse or Input Capture Mode.

• **TM1C2 Register – 10-bit ETM**

Bit	7	6	5	4	3	2	1	0
Name	T1BM1	T1BM0	T1BIO1	T1BIO0	T1BOC	T1BPOL	T1PWM1	T1PWM0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 **T1BM1~T1BM0**: Select TM1 CCRB Operating Mode
00: Compare Match Output Mode
01: Capture Input Mode
10: PWM Mode or Single Pulse Output Mode
11: Timer/Counter mode

These bits setup the required operating mode for the TM. To ensure reliable operation the TM should be switched off before any changes are made to the T1BM1 and T1BM0 bits. In the Timer/Counter Mode, the TM output pin control must be disabled.

Bit 5~4 **T1BIO1~T1BIO0**: Select TP1B_0, TP1B_1, TP1B_2 output function

Compare Match Output Mode
00: No change
01: Output low
10: Output high
11: Toggle output

PWM Mode/Single Pulse Output Mode
00: Force inactive state
01: Force active state
10: PWM output
11: Single pulse output

Capture Input Mode
00: Input capture at rising edge of TP1B_0, TP1B_1, TP1B_2
01: Input capture at falling edge of TP1B_0, TP1B_1, TP1B_2
10: Input capture at falling/rising edge of TP1B_0, TP1B_1, TP1B_2
11: Input capture disabled

Timer/counter Mode
Unused

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

In the Compare Match Output Mode, the T1BIO1 and T1BIO0 bits determine how the TM output pin changes state when a compare match occurs from the Comparator A. The TM output pin can be setup to switch high, switch low or to toggle its present state when a compare match occurs from the Comparator A. When the bits are both zero, then no change will take place on the output. The initial value of the TM output pin should be setup using the T1BOC bit in the TM1C2 register. Note that the output level requested by the T1BIO1 and T1BIO0 bits must be different from the initial value setup using the T1BOC bit otherwise no change will occur on the TM output pin when a compare match occurs. After the TM output pin changes state it can be reset to its initial level by changing the level of the T1ON bit from low to high.

Bit 3 **T1BOC**: TP1B_0, TP1B_1, TP1B_2 Output control bit

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

PWM Mode/ Single Pulse Output Mode
0: Active low
1: Active high

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

Bit 2 **T1BPOL**: TP1B_0, TP1B_1, TB1B_2 Output polarity Control
0: Non-invert
1: Invert
This bit controls the polarity of the TP1B_0, TP1B_1, TP1B_2 output pin. When the bit is set high the TM output pin will be inverted and not inverted when the bit is zero. It has no effect if the TM is in the Timer/Counter Mode.

Bit 1~0 **T1PWM1~T1PWM0**: Select PWM Mode
00: Edge aligned
01: Centre aligned, compare match on count up
10: Centre aligned, compare match on count down
11: Centre aligned, compare match on count up or down

- **TM1DL Register – 10-bit ETM**

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 **TM1DL**: TM1 Counter Low Byte Register bit 7~bit 0
TM1 10-bit Counter bit 7~bit 0

- **TM1DH Register – 10-bit ETM**

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 **TM1DH**: TM1 Counter High Byte Register bit 1~bit 0
TM1 10-bit Counter bit 9~bit 8

- **TM1AL Register – 10-bit ETM**

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 **TM1AL**: TM1 CCRA Low Byte Register bit 7~bit 0
TM1 10-bit CCRA bit 7~bit 0

- **TM1AH Register – 10-bit ETM**

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,

Bit 1~0 **TM1AH**: TM1 CCRA High Byte Register bit 1~bit 0
TM1 10-bit CCRA bit 9~bit 8

• **TM1BL Register – 10-bit ETM**

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 **TM1BL**: TM1 CCRB Low Byte Register bit 7~bit 0
TM1 10-bit CCRB bit 7~bit 0

• **TM1BH Register – 10-bit ETM**

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,

Bit 1~0 **TM1BH**: TM1 CCRB High Byte Register bit 1~bit 0
TM1 10-bit CCRB bit 9 ~ bit 8

Enhanced Type TM Operating Modes

The Enhanced 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 TnAM1 and TnAM0 bits in the TMnC1, and the TnBM1 and TnBM0 bits in the TMnC2 register.

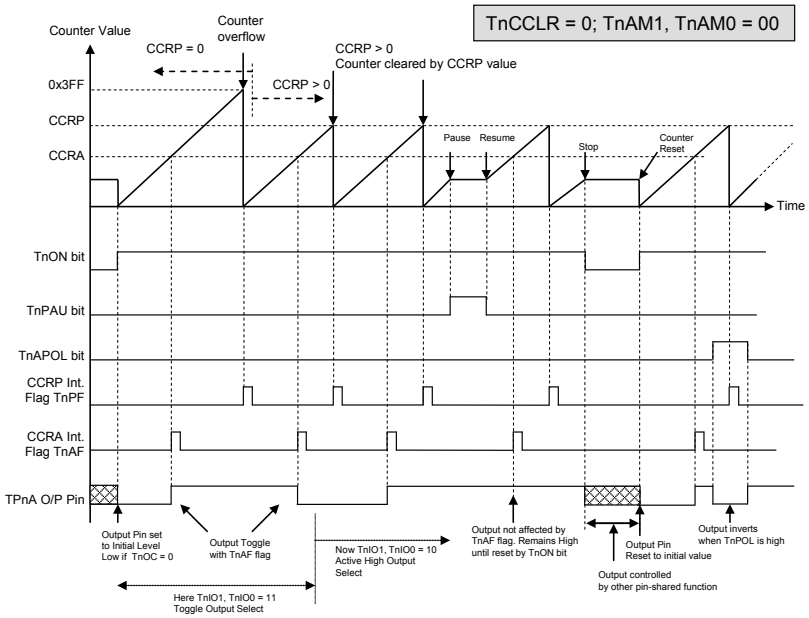
ETM Operating Mode	CCRA Compare Match Output Mode	CCRA Timer/ Counter Mode	CCRA PWM Output Mode	CCRA Single Pulse Output Mode	CCRA Input Capture Mode
CCRB Compare Match Output Mode	√	√	√	—	—
CCRB Timer/ Counter Mode	√	√	√	—	—
CCRB PWM Output Mode	√	√	√	—	—
CCRB Single Pulse Output Mode	—	—	—	√	—
CCRB Input Capture Mode	√	√	√	—	√

Compare Output Mode

To select this mode, bits TnAM1, TnAM0 and TnBM1, TnBM0 in the TMnC1/TMnC2 registers should be all cleared to zero. 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 TnCCLR bit is low, there are two ways in which the counter can be cleared. One is when a compare match occurs from Comparator P, the other is when the CCRP bits are all zero which allows the counter to overflow. Here both the TnAF and TnPF interrupt request flags for Comparator A and Comparator P respectively, will both be generated.

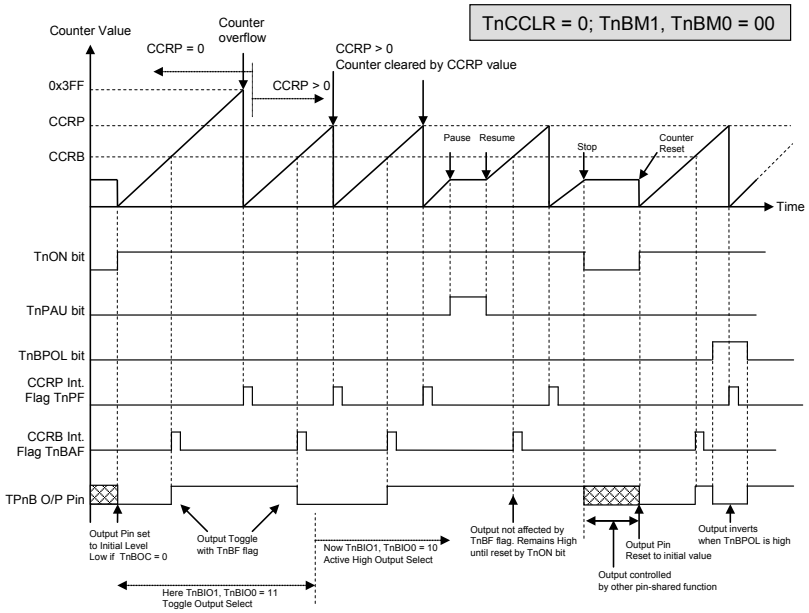
If the TnCCLR bit in the TMnC1 register is high then the counter will be cleared when a compare match occurs from Comparator A. However, here only the TnAF interrupt request flag will be generated even if the value of the CCRP bits is less than that of the CCRA registers. Therefore when TnCCLR is high no TnPF interrupt request flag will be generated.

As the name of the mode suggests, after a comparison is made, the TM output pin, will change state. The TM output pin condition however only changes state when a TnAF or TnBF interrupt request flag is generated after a compare match occurs from Comparator A or Comparator B. The TnPF interrupt request flag, generated from a compare match from Comparator P, will have no effect on the TM output pin. The way in which the TM output pin changes state is determined by the condition of the TnAIO1 and TnAIO0 bits in the TMnC1 register for ETM CCRA, and the TnBIO1 and TnBIO0 bits in the TMnC2 register for ETM CCRB. The TM output pin can be selected using the TnAIO1, TnAIO0 bits (for the TPnA pin) and TnBIO1, TnBIO0 bits (for the TPnB_0, TPnB_1 or TPnB_2 pins) to go high, to go low or to toggle from its present condition when a compare match occurs from Comparator A or a compare match occurs from Comparator B. The initial condition of the TM output pin, which is setup after the TnON bit changes from low to high, is setup using the TnAOC or TnBOC bit for TPnA or TPnB_0, TPnB_1, TPnB_2 output pins. Note that if the TnAIO1,TnAIO0 and TnBIO1, TnBIO0 bits are zero then no pin change will take place.



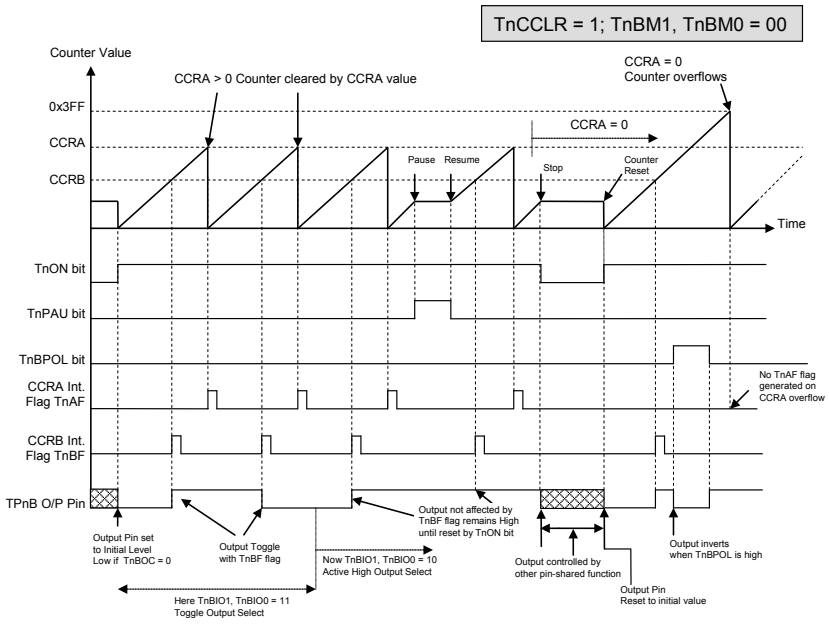
ETM CCRA Compare Match Output Mode — TnCCLR = 0

Note: 1. With TnCCLR = 0 the Comparator P match will clear the counter
2. TPnA output pin controlled only by TnAF flag
3. Output pin reset to initial state by TnON bit rising edge



ETM CCRB Compare Match Output Mode — TnCCLR = 0

Note: 1. With TnCCLR = 0 the Comparator P match will clear the counter
2. TPnB output pin controlled only by TnBF flag
3. Output pin reset to initial state by TnON bit rising edge



ETM CCRB Compare Match Output Mode — TnCCLR = 1

Note: 1. With TnCCLR = 1 the Comparator A match will clear the counter
2. TPnB output pin controlled only by TnBF flag
3. TPnB output pin reset to initial state by TnON rising edge
4. TnPF flags not generated when TnCCLR = 1

Timer/Counter Mode

To select this mode, bits TnAM1, TnAM0 and TnBM1, TnBM0 in the TMnC1 and TMnC2 register should all be set high. 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 TM 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 TM 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, the required bit pairs, TnAM1, TnAM0 and TnBM1, TnBM0 should be set to 10 respectively and also the TnAIO1, TnAIO0 and TnBIO1, TnBIO0 bits should be set to 10 respectively. The PWM function within the TM 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 TM 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 mode, the TnCCLR bit is used to determine in which way the PWM period is controlled. With the TnCCLR bit set high, the PWM period can be finely controlled using the CCRA registers. In this case the CCRB registers are used to set the PWM duty value (for TPnB output pins). The CCRP bits are not used and TPnA output pin is not used. The PWM output can only be generated on the TPnB output pins. With the TnCCLR bit cleared to zero, the PWM period is set using one of the eight values of the three CCRP bits, in multiples of 128. Now both CCRA and CCRB registers can be used to setup different duty cycle values to provide dual PWM outputs on their relative TPnA and TPnB pins.

The TnPWM1 and TnPWM0 bits determine the PWM alignment type, which can be either edge or centre type. In edge alignment, the leading edge of the PWM signals will all be generated concurrently when the counter is reset to zero. With all power currents switching on at the same time, this may give rise to problems in higher power applications. In centre alignment the centre of the PWM active signals will occur sequentially, thus reducing the level of simultaneous power switching currents.

Interrupt flags, one for each of the CCRA, CCRB and CCRP, will be generated when a compare match occurs from either the Comparator A, Comparator B or Comparator P. The TnAOC and TnBOC bits in the TMnC1 and TMnC2 register are used to select the required polarity of the PWM waveform while the two TnAIO1, TnAIO0 and TnBIO1, TnBIO0 bits pairs are used to enable the PWM output or to force the TM output pin to a fixed high or low level. The TnAPOL and TnBPOL bit are used to reverse the polarity of the PWM output waveform.

- **ETM, PWM Mode, Edge-aligned Mode, TnCCLR=0**

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

- **ETM, PWM Mode, Edge-aligned Mode, TnCCLR=1**

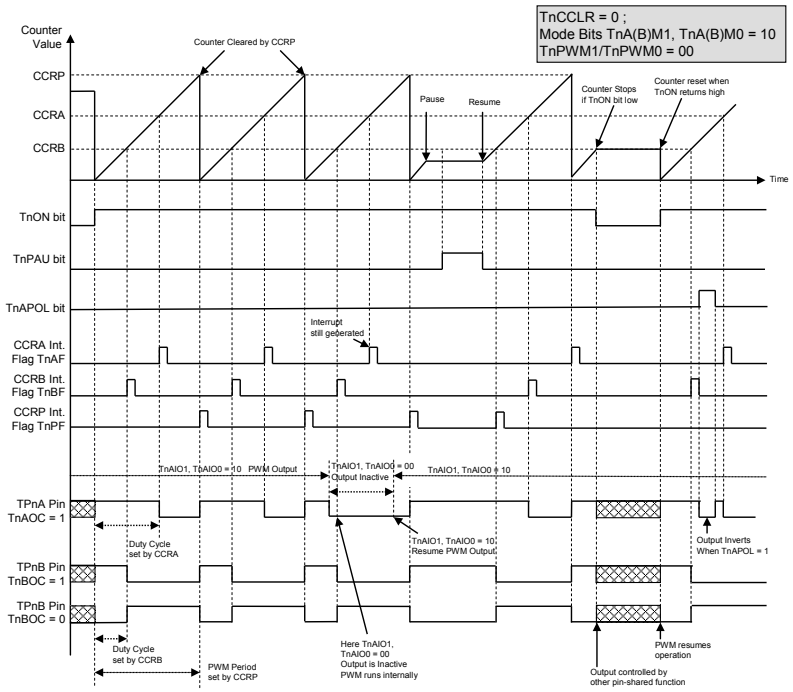
CCRA	1	2	3	511	512	1021	1022	1023
Period	1	2	3	511	512	1021	1022	1023
B Duty	CCRB							

- **ETM, PWM Mode, Center-aligned Mode, TnCCLR=0**

CCRP	001b	010b	011b	100b	101b	110b	111b	000b
Period	256	512	768	1024	1280	1536	1792	2046
A Duty	(CCRAx2) -1							
B Duty	(CCRBx2) -1							

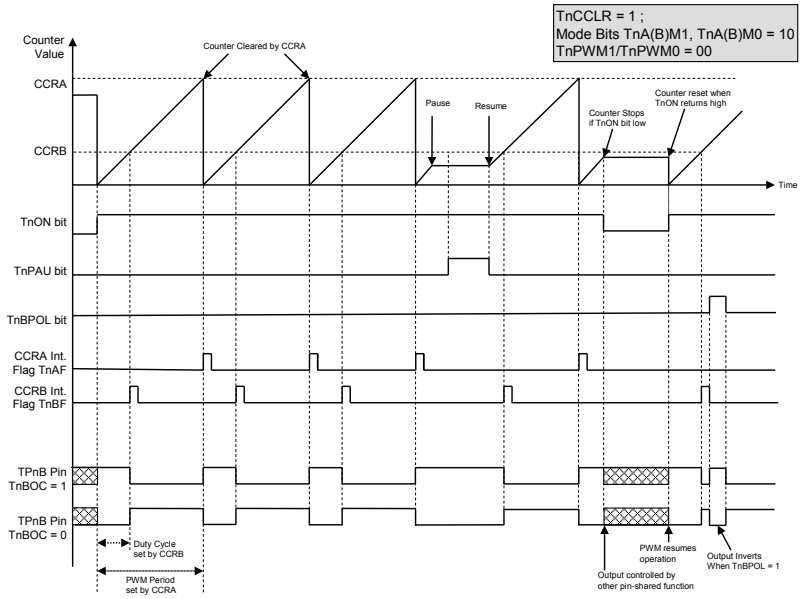
- **ETM, PWM Mode, Center-aligned Mode, TnCCLR=1**

CCRA	1	2	3	511	512	1021	1022	1023
Period	2	4	6	1022	1024	2042	2044	2046
B Duty	(CCRB x 2) x 1							



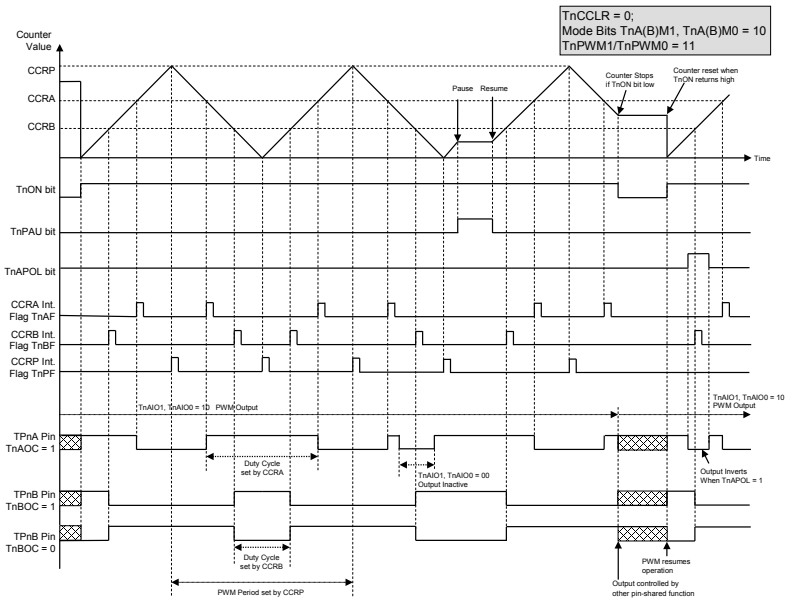
PWM Mode — Edge Aligned

Note: 1. Here TnCCLR = 0 therefore CCRP clears counter and determines PWM period
2. Internal PWM function continues even when TnAIO1, TnAIO0 (or TnBIO1, TnBIO0) = 00 or 01
3. CCRA controls TPnA PWM duty and CCRB controls TPnB PWM duty



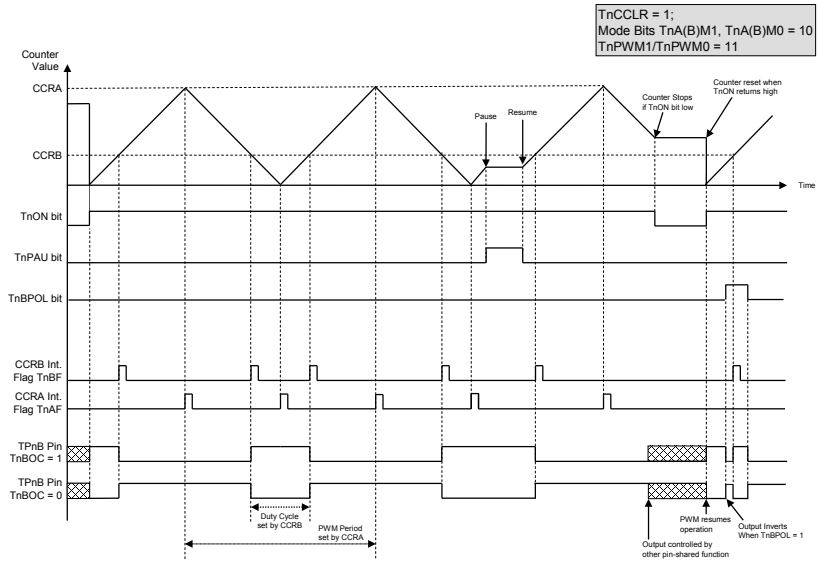
PWM Mode — Edge Aligned

Note: 1. Here TnCCLR = 1 therefore CCRA clears counter and determines PWM period
2. Internal PWM function continues even when TnBIO1, TnBIO0 = 00 or 01
3. CCRA controls TPnB PWM period and CCRB controls TPnB PWM duty



PWM Mode — Centre Aligned

Note: 1. Here TnCCLR = 0 therefore CCRP clears counter and determines PWM period
2. TnPWM1/TnPWM0 = 11 therefore PWM is centre aligned
3. Internal PWM function continues even when TnAIO1, TnAIO0 (or TnBIO1, TnBIO0) = 00 or 01
4. CCRA controls TPnA PWM duty and CCRB controls TPnB PWM duty
5. CCRP will generate an interrupt request when the counter decrements to its zero value.



PWM Mode — Centre Aligned

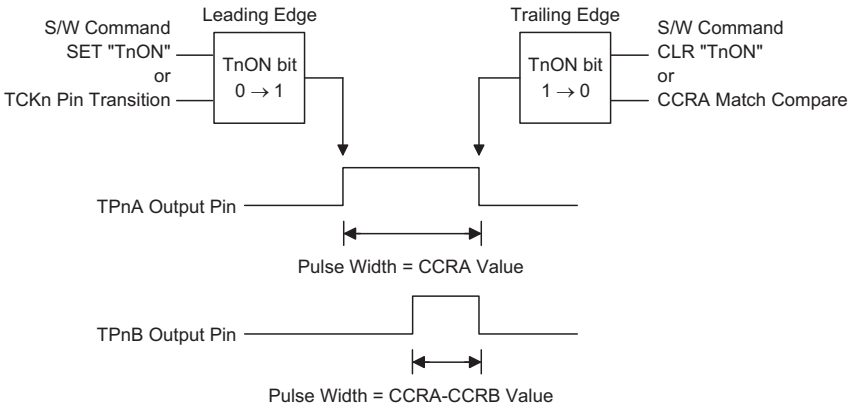
Note: 1. Here TnCCLR = 1 therefore CCRA clears counter and determines PWM period
2. TnPWM1/TnPWM0 = 11 therefore PWM is centre aligned
3. Internal PWM function continues even when TnBIO1, TnBIO0 = 00 or 01
4. CCRA controls the TPnB PWM period and CCRB controls the TPnB PWM duty
5. CCRP will generate an interrupt request when the counter decrements to its zero value.

Single Pulse Output Mode

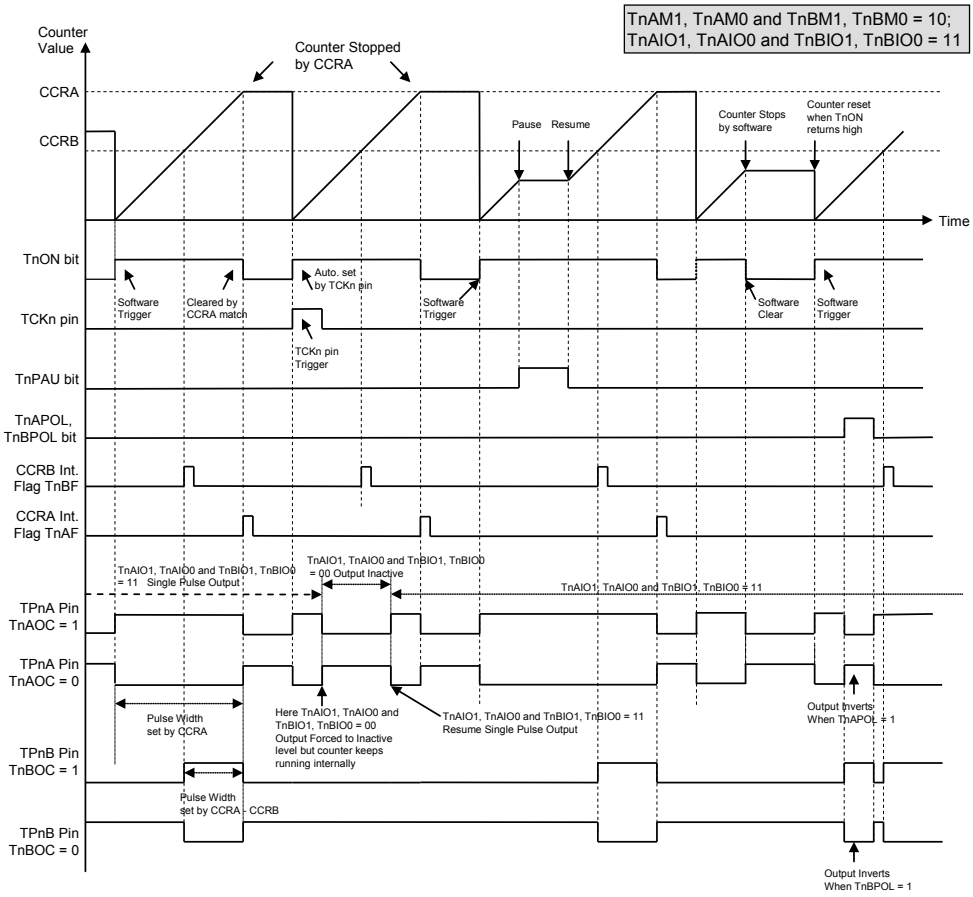
To select this mode, the required bit pairs, TnAM1, TnAM0 and TnBM1, TnBM0 should be set to 10 respectively and also the corresponding TnAIO1, TnAIO0 and TnBIO1, TnBIO0 bits should be set to 11 respectively. The Single Pulse Output Mode, as the name suggests, will generate a single shot pulse on the TM output pin.

The trigger for the pulse TPnA output leading edge is a low to high transition of the TnON bit, which can be implemented using the application program. The trigger for the pulse TPnB output leading edge is a compare match from Comparator B, which can be implemented using the application program. However in the Single Pulse Mode, the TnON bit can also be made to automatically change from low to high using the external TCKn pin, which will in turn initiate the Single Pulse output of TPnA. When the TnON bit transitions to a high level, the counter will start running and the pulse leading edge of TPnA will be generated. The TnON bit should remain high when the pulse is in its active state. The generated pulse trailing edge of TPnA and TPnB will be generated when the TnON 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 TnON bit and thus generate the Single Pulse output trailing edge of TPnA and TPnB. In this way the CCRA value can be used to control the pulse width of TPnA. The CCRA-CCRB value can be used to control the pulse width of TPnB. A compare match from Comparator A and Comparator B will also generate TM interrupts. The counter can only be reset back to zero when the TnON bit changes from low to high when the counter restarts. In the Single Pulse Mode CCRP is not used. The TnCCLR bit is also not used.



Single Pulse Generation



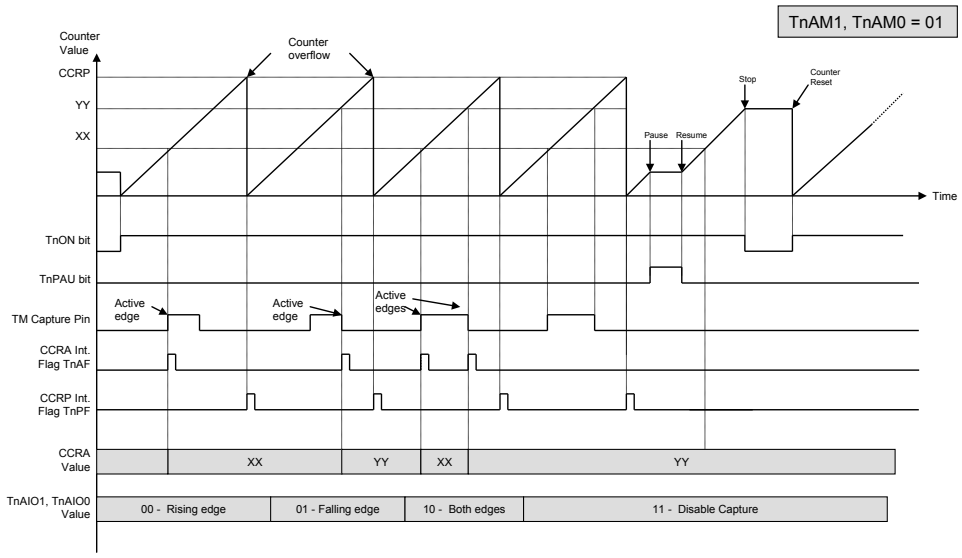
ETM — Single Pulse Mode

Capture Input Mode

To select this mode bits TnAM1, TnAM0 and TnBM1, TnBM0 in the TMnC1 and TMnC2 registers 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 TPnA and TPnB_0, TPnB_1, TPnB_2 pins, whose 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 TnAIO1, TnAIO0 and TnBIO1, TnBIO0 bits in the TMnC1 and TMnC2 registers. The counter is started when the TnON bit changes from low to high which is initiated using the application program.

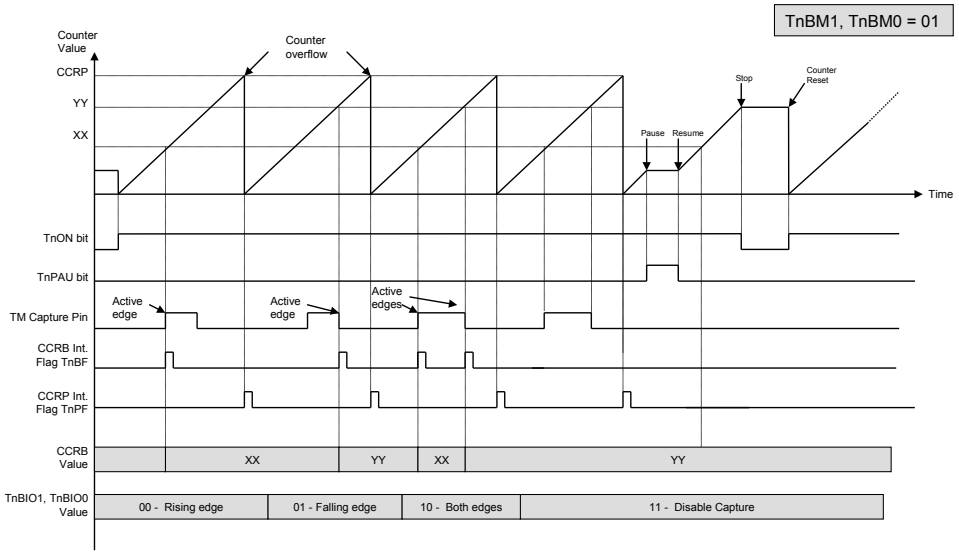
When the required edge transition appears on the TPnA and TPnB_0, TPnB_1, TPnB_2 pins the present value in the counter will be latched into the CCRA and CCRB registers and a TM interrupt generated. Irrespective of what events occur on the TPnA and TPnB_0, TPnB_1, TPnB_2 pins the counter will continue to free run until the TnON 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 TM 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 TnAIO1, TnAIO0 and TnBIO1, TnBIO0 bits can select the active trigger edge on the TPnA and TPnB_0, TPnB_1, TPnB_2 pins to be a rising edge, falling edge or both edge types. If the TnAIO1, TnAIO0 and TnBIO1, TnBIO0 bits are both set high, then no capture operation will take place irrespective of what happens on the TPnA and TPnB_0, TPnB_1, TPnB_2 pins, however it must be noted that the counter will continue to run.

As the TPnA and TPnB_0, TPnB_1, TPnB_2 pins are pin shared with other functions, care must be taken if the TM is in the Capture Input Mode. This is because if the pin is setup as an output, then any transitions on this pin may cause an input capture operation to be executed. The TnCCLR, TnAOC, TnBOC, TnAPOL and TnBPOL bits are not used in this mode.



ETM CCRA Capture Input Mode

Note: 1. TnAM1, TnAM0 = 01 and active edge set by TnAIO1 and TnAIO0 bits
2. TM Capture input pin active edge transfers counter value to CCRA
3. TnCCLR bit not used
4. No output function - TnAOC and TnAPOL bits not used
5. CCRP sets counter maximum value



ETM CCRB Capture Input Mode

Note: 1. TnBM1, TnBM0 = 01 and active edge set by TnBIO1 and TnBIO0 bits
2. TM Capture input pin active edge transfers counter value to CCRB
3. TnCCLR bit not used
4. No output function - TnBOC and TnBPOL bits not used
5. CCRP sets counter maximum value

Timer/Event Counter – TMR

In addition to the Timer Modules, the device provides an additional count-up Timer/Event Counter of 8-bit capacity with the name TMR. It should not be confused with the Timer Modules as its structure and operation is very different from that of the Timer Modules. One specific purpose of this timer is for the SCF clock. As the timer has three different operating modes, it can be configured to operate as a general timer, an external event counter or as a pulse width capture device. The provision of an internal prescaler to the clock circuitry gives added range to the timer.

There are two types of registers related to the Timer/Event Counter. The first is the register that contains the actual value of the timer and into which an initial value can be preloaded. Reading from this register retrieves the contents of the Timer/Event Counter. The second type of associated register is the Timer Control Register which defines the timer options and determines how the timer is to be used. The device can have the timer clock configured to come from the internal clock source. In addition, the timer clock source can also be configured to come from an external timer pin.

Configuring the Timer/Event Counter Input Clock Source

The Timer/Event Counter clock source can originate from various sources, an internal clock or an external pin. The internal clock source is used when the timer is in the timer mode or in the pulse width capture mode. This internal clock source is first divided by a prescaler, the division ratio of which is conditioned by the Timer Control Register bits TPSC0~TPSC2 and the internal clock source is from f_{SYS}.

An external clock source is used when the Timer/Event Counter is in the event counting mode, the clock source being provided on an external timer pin. Depending upon the condition of the TEG bit, each high to low, or low to high transition on the external timer pin will increment the counter by one.

Timer Register – TMR

The timer register is special function register located in the Special Purpose Data Memory and is the place where the actual timer value is stored. This register is known as TMR. The value in the timer register increases by one each time an internal clock pulse is received or an external transition occurs on the external timer pin. The timer will count from the initial value loaded by the preload register to the full count of FFH at which point the timer overflows and an internal interrupt signal is generated. The timer value will then be reset with the initial preload register value and continue counting. Note that to achieve a maximum full range count of FFH, the preload register must first be cleared to all zeros. It should be noted that after power-on, the preload registers will be in an unknown condition.

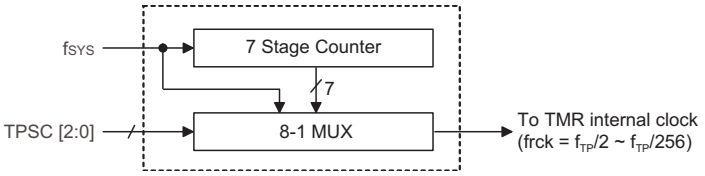
Note that if the Timer/Event Counter is in an OFF condition and data is written to its preload register, this data will be immediately written into the actual counter. However, if the counter is enabled and counting, any new data written into the preload data register during this period will remain in the preload register and will only be written into the actual counter the next time an overflow occurs.

Timer Control Register – TMRC

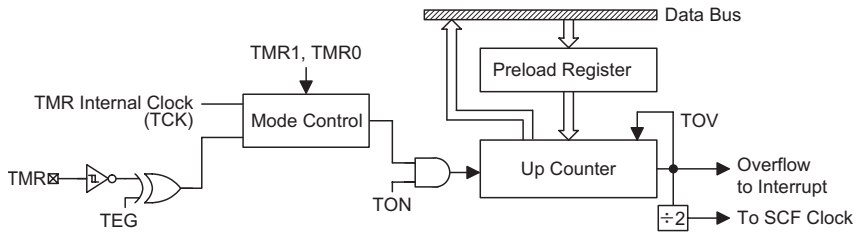
The flexible features of the Holtek microcontroller Timer/Event Counter enable them to operate in three different modes, the options of which are determined by the contents of their respective control register.

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

To choose which of the three modes the timer is to operate in, either in the timer mode, the event counting mode or the pulse width capture mode, bits 7 and 6 of the Timer Control Register, which are known as the bit pair TMR1/TMR0, must be set to the required logic levels. The timer-on bit, which is bit 4 of the Timer Control Register and known as TON, provides the basic on/off control of the respective timer. Setting the bit high allows the counter to run, clearing the bit stops the counter. Bits 0~2 of the Timer Control Register determine the division ratio of the input clock prescaler. The prescaler bit settings have no effect if an external clock source is used. If the timer is in the event count or pulse width capture mode, the active transition edge level type is selected by the logic level of bit 3 of the Timer Control Register which is known as TEG.



Clock Structure for TMR



TMR Structure

• **TMRC Register**

Bit	7	6	5	4	3	2	1	0
Name	TMR1	TMR0	—	TON	TEG	TPSC2	TPSC1	TPSC0
R/W	R/W	R/W	—	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	1	0	0	0

Bit 7, 6 **TMR1, TMR0**: Timer/Event Counter operation mode selection
00: no mode available
01: event counter mode
10: timer mode
11: pulse width capture mode

Bit 5 Unimplemented, read as "0"

Bit 4 **TON**: Timer/Event Counter counting enable
0: disable
1: enable

Bit 3 **TEG**:

Event Counter active edge selection
0: count on rising edge
1: count on falling edge

Pulse Width Capture active edge selection
0: start counting on falling edge, stop on rising edge
1: start counting on rising edge, stop on falling edge

Bit 2~0 **TPSC2, TPSC1, TPSC0**: Timer prescaler rate selection

Timer internal clock =
000: $f_{sys}/2$
001: $f_{sys}/4$
010: $f_{sys}/8$
011: $f_{sys}/16$
100: $f_{sys}/32$
101: $f_{sys}/64$
110: $f_{sys}/128$
111: $f_{sys}/256$

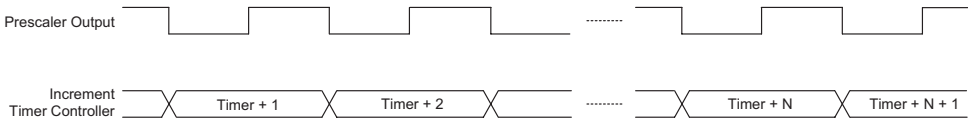
Timer Mode

In this mode, the Timer/Event Counter can be utilised to measure fixed time intervals, providing an internal interrupt signal each time the Timer/Event Counter overflows. To operate in this mode, the Operating Mode Select bit pair, TMR1/TMR0, in the Timer/Event Counter Control Register must be set to the correct value as shown.

Control Register Operating Mode Select Bits for the Timer Mode	Bit7	Bit6
	1	0

In this mode the internal clock is used as the timer clock. The timer input clock source is from f_{SYS}. However, this timer clock source is further divided by a prescaler, the value of which is determined by the bits TPSC2~TPSC0 in the Timer Control Register. The timer-on bit, TON must be set high to enable the timer to run. Each time an internal clock high to low transition occurs, the timer increments by one; when the timer is full and overflows, an interrupt signal is generated and the timer will reload the value already loaded into the preload register and continue counting. A timer overflow condition and corresponding internal interrupt is one of the wake-up sources.

Note that one main reason for TMR is as a clock source for the internal SCF. In order to obtain a precise clock source for the SCF, the Timer/Event Counter interrupt should be disabld to prevent erroneous operation. Refer to the SCF section for more details.



Timer Mode Timing Chart

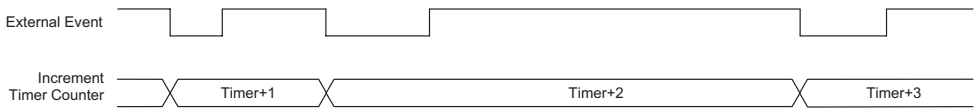
Event Counter Mode

In this mode, a number of externally changing logic events, occurring on the external timer TMR pin, can be recorded by the Timer/Event Counter. To operate in this mode, the Operating Mode Select bit pair, TMR1/TMR0, in the Timer Control Register must be set to the correct value as shown.

Control Register Operating Mode Select Bits for the Event Counter Mode	Bit7	Bit6
	0	1

In this mode, the external timer TMR pin is used as the Timer/Event Counter clock source, however it is not divided by the internal prescaler. After the other bits in the Timer Control Register have been setup, the enable bit TnON, which is bit 4 of the Timer Control Register, can be set high to enable the Timer/Event Counter to run. If the Active Edge Select bit, TEG, which is bit 3 of the Timer Control Register, is low, the Timer/Event Counter will increment each time the external timer pin receives a low to high transition. If the TEG is high, the counter will increment each time the external timer pin receives a high to low transition. When it is full and overflows, an interrupt signal is generated and the Timer/Event Counter will reload the value already loaded into the preload register and continue counting. The interrupt can be disabled by ensuring that the Timer/Event Counter Interrupt Enable bit in the corresponding Interrupt Control Register is reset to zero.

As the external timer pin is shared with an I/O pin, to ensure that the pin is configured to operate as an event counter input pin, two things have to happen. The first is to ensure that the Operating Mode Select bits in the Timer Control Register place the Timer/Event Counter in the Event Counting Mode, the second is to ensure that the port control register configures the pin as an input. It should be noted that in the event counting mode, even if the microcontroller is in the Idle/Sleep Mode, the Timer/Event Counter will continue to record externally changing logic events on the timer input TMR pin. As a result when the timer overflows it will generate a timer interrupt and corresponding wake-up source.



Enent Counter Mode Timing Chart (TEG=1)

Pulse Width Capture Mode

In this mode, the Timer/Event Counter can be utilised to measure the width of external pulses applied to the external timer pin. To operate in this mode, the Operating Mode Select bit pair, TMR1/TMR0, in the Timer Control Register must be set to the correct value as shown.

Control Register Operating Mode	Bit7	Bit6
Select Bits for the Pulse Width Capture Mode	1	1

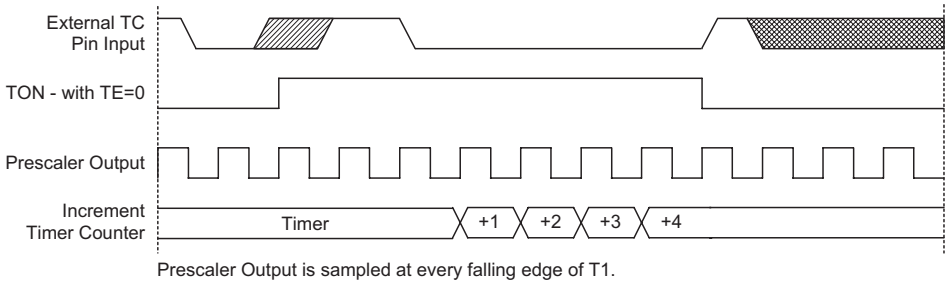
In this mode the internal clock, f_{SYS}, is used as the internal clock for the 8-bit Timer/Event Counter. However, the clock source, f_{SYS}, for the 8-bit timer is further divided by a prescaler, the value of which is determined by the Prescaler Rate Select bits TPSC2~TPSC0, which are bits 2~0 in the Timer Control Register. After the other bits in the Timer Control Register have been setup, the enable bit TON, which is bit 4 of the Timer Control Register, can be set high to enable the Timer/Event Counter, however it will not actually start counting until an active edge is received on the external timer pin.

If the Active Edge Select bit TEG, which is bit 3 of the Timer Control Register, is low, once a high to low transition has been received on the external timer pin, the Timer/Event Counter will start counting until the external timer pin returns to its original high level. At this point the enable bit will be automatically reset to zero and the Timer/Event Counter will stop counting. If the Active Edge Select bit is high, the Timer/Event Counter will begin counting once a low to high transition has been received on the external timer pin and stop counting when the external timer pin returns to its original low level. As before, the enable bit will be automatically reset to zero and the Timer/Event Counter will stop counting. It is important to note that in the pulse width capture Mode, the enable bit is automatically reset to zero when the external control signal on the external timer pin returns to its original level, whereas in the other two modes the enable bit can only be reset to zero under program control.

The residual value in the Timer/Event Counter, which can now be read by the program, therefore represents the length of the pulse received on the TMR pin. As the enable bit has now been reset, any further transitions on the external timer pin will be ignored. The timer cannot begin further pulse width capture until the enable bit is set high again by the program. In this way, single shot pulse measurements can be easily made.

It should be noted that in this mode the Timer/Event Counter is controlled by logical transitions on the external timer pin and not by the logic level. When the Timer/Event Counter is full and overflows, an interrupt signal is generated and the Timer/Event Counter will reload the value already loaded into the preload register and continue counting. The interrupt can be disabled by ensuring that the Timer/Event Counter Interrupt Enable bit in the corresponding Interrupt Control Register is reset to zero.

As the TMR pin is shared with an I/O pin, to ensure that the pin is configured to operate as a pulse width capture pin, two things have to happen. The first is to ensure that the Operating Mode Select bits in the Timer Control Register place the Timer/Event Counter in the pulse width capture Mode, the second is to ensure that the port control register configures the pin as an input.



Pulse Width Capture Mode Timing Chart (TE=0)

Prescaler

Bits TPSC0~TPSC2 of the TMRC register can be used to define a division ratio for the internal clock source of the Timer/Event Counter enabling longer time out periods to be setup.

I/O Interfacing

The Timer/Event Counter, when configured to run in the event counter or pulse width capture mode, requires the use of an external timer pin for its operation. As this pin is a shared pin it must be configured correctly to ensure that it is setup for use as a Timer/Event Counter input pin. This is achieved by ensuring that the mode select bits in the Timer/Event Counter control register select either the event counter or pulse width capture mode. Additionally the corresponding Port Control Register bit must be set high to ensure that the pin is setup as an input. Any pull-high resistor connected to this pin will remain valid even if the pin is used as a Timer/Event Counter input.

Programming Considerations

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

When the Timer/Event Counter is read, or if data is written to the preload register, the clock is inhibited to avoid errors, however as this may result in a counting error, this should be taken into account by the programmer. Care must be taken to ensure that the timers are properly initialized before using them for the first time. The associated timer enable bits in the interrupt control register must be properly set otherwise the internal interrupt associated with the timer will remain inactive. The edge select, timer mode and clock source control bits in timer control register must also be correctly set to ensure the timer is properly configured for the required application. It is also important to ensure that an initial value is first loaded into the timer registers before the timer is switched on; this is because after power-on the initial values of the timer registers are unknown. After the timer has been initialized the timer can be turned on and off by controlling the enable bit in the timer control register.

When the Timer/Event Counter overflows, its corresponding interrupt request flag in the interrupt control register will be set. If the Timer/Event Counter interrupt is enabled this will in turn generate an interrupt signal. However irrespective of whether the interrupts are enabled or not, a Timer/Event Counter overflow will also generate a wake-up signal if the device is in a Power-down condition. This situation may occur if the Timer/Event Counter is in the Event Counting Mode and if the external signal continues to change state. In such a case, the Timer/Event Counter will continue to count these external events and if an overflow occurs the device will be woken up from its Power-down condition. To prevent such a wake-up from occurring, the timer interrupt request flag should first be set high before issuing the "HALT" instruction to enter the Idle/Sleep Mode.

Timer/Event Counter Program Example

The program shows how the Timer/Event Counter registers are setup along with how the interrupts are enabled and managed. Note how the Timer/Event Counter is turned on, by setting bit 4 of the Timer Control Register. The Timer/Event Counter can be turned off in a similar way by clearing the same bit. This example program sets the Timer/Event Counters to be in the timer mode, which uses the internal system clock as their clock source.

Timer/Event Counter Program Example

```
org         04h             ; external interrupt vector
org         30h             ; Timer/Event Counter interrupt vector
jmp         tmrint          ; jump here when TMR overflows
:
:
org         20h             ; main program
:
:
;internal TMR interrupt routine
tmrint:
:
; TMR main program placed here
:
:
begin:
;setup TMR registers
mov         a,09bh          ; setup TMR preload value
mov         tmr,a
mov         a,080h          ; setup TMR control register
mov         tmrc,a          ; timer mode and prescaler set to /2
;setup interrupt register
mov         a,00dh          ; enable master interrupt and both timer interrupts
mov         intc0,a
:
:
set         tmrc.4          ; start TMR
:
```

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 Overview

The devices contains a 8 channels 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 either a 12-bit digital value.

Input Channels	A/D Channel Select Bits	Input Pins
8	ACS4, ACS2~ACS0	AN0~AN7

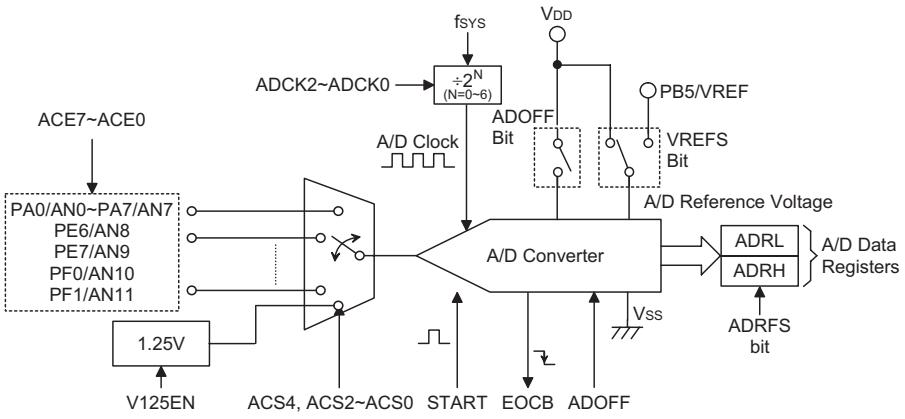
The accompanying block diagram shows the overall internal structure of the A/D converter, together with its associated registers.

A/D Converter Register Description

Overall operation of the A/D converter is controlled using five registers. A read only register pair exists to store the ADC data 12-bit value. The remaining three registers are control registers which setup the operating and control function of the A/D converter.

Register Name	Bit							
	7	6	5	4	3	2	1	0
ADRL(ADRFS=0)	D3	D2	D1	D0	—	—	—	—
ADRL(ADRFS=1)	D7	D6	D5	D4	D3	D2	D1	D0
ADRH(ADRFS=0)	D11	D10	D9	D8	D7	D6	D5	D4
ADRH(ADRFS=1)	—	—	—	—	D11	D10	D9	D8
ADCR0	START	EOCB	ADOFF	ADRFS	—	ACS2	ACS1	ACS0
ADCR1	ACS4	V125EN	—	VREFS	—	ADCK2	ADCK1	ADCK0
ACER	ACE7	ACE6	ACE5	ACE4	ACE3	ACE2	ACE1	ACE0

A/D Converter Register List



A/D Converter Structure

A/D Converter Data Registers – ADRL, ADRH

As the devices contain an internal 12-bit A/D converter, they require two data registers to store the converted value.

These are a high byte register, known as ADRH, and a low byte register, known as ADRL. 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 ADCR0 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.

ADRFS	ADRH								ADRL							
	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 Data Registers

A/D Converter Control Registers – ADCR0, ADCR1, ACER

To control the function and operation of the A/D converter, three control registers known as ADCR0, ADCR1 and ACER are provided. These 8-bit registers define functions such as the selection of which analog channel is connected to the internal A/D converter, the digitised data format, the A/D clock source as well as controlling the start function and monitoring the A/D converter end of conversion status. The ACS2~ACS0 bits in the ADCR0 register and ACS4 bit is the ADCR1 register define the ADC input channel number. As the device contains only one actual analog to digital converter hardware circuit, each of the individual 8 analog inputs must be routed to the converter. It is the function of the ACS4~ACS0 bits to determine which analog channel input pins or internal 1.25V is actually connected to the internal A/D converter.

The ACER control register contains the ACER7~ACER0 bits which determine which pins on Port A 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. Setting the corresponding bit high will select the A/D input function, clearing the bit to zero will select either the I/O or other pin-shared function. When the pin is selected to be an

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

• **ADCR0 Register**

Bit	7	6	5	4	3	2	1	0
Name	START	EOCB	ADOFF	ADRFS	—	ACS2	ACS1	ACS0
R/W	R/W	R	R/W	R/W	—	R/W	R/W	R/W
POR	0	1	1	0	—	0	0	0

Bit 7 **START**: Start the A/D conversion
0→1→0: start
0→1 : reset the A/D converter and set EOCB to "1"
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. When the bit is set high the A/D converter will be reset.

Bit 6 **EOCB**: End of A/D conversion flag
0: A/D conversion ended
1: A/D conversion in progress
This read only flag is used to indicate when an A/D conversion process has completed. When the conversion process is running the bit will be high.

Bit 5 **ADOFF** : ADC module power on/off control bit
0: ADC module power on
1: ADC module power off
This bit controls the power to the A/D internal function. This bit should be cleared to zero to enable the A/D converter. If the bit is set high then the A/D converter will be switched off reducing the device power consumption. As the A/D converter will consume a limited amount of power, even when not executing a conversion, this may be an important consideration in power sensitive battery powered applications.
Note: 1. It is recommended to set ADOFF=1 before entering IDLE/SLEEP Mode for saving power.
2. ADOFF=1 will power down the ADC module.

Bit 4 **ADRFS**: ADC Data Format Control
0: ADC Data MSB is ADRH bit 7, LSB is ADRL bit 4
1: ADC Data MSB is ADRH bit 3, LSB is ADRL bit 0
This bit controls the format of the 12-bit converted A/D value in the two A/D data registers.
Details are provided in the A/D data register section.

Bit 3 unimplemented, read as "0"

Bit 2~0 **ACS2, ACS1, ACS0**: Select A/D channel (when ACS4 is "0")
000: AN0
001: AN1
010: AN2
011: AN3
100: AN4
101: AN5
110: AN6
111: AN7
These are the A/D channel select control bits. As there is only one internal hardware A/D converter each of the eight A/D inputs must be routed to the internal converter using these bits.
If bit ACS4 in the ADCR1 register is set high then the internal 1.25V will be routed to the A/D Converter.

• **ADCR1 Register**

Bit	7	6	5	4	3	2	1	0
Name	ACS4	V125EN	—	VREFS	—	ADCK2	ADCK1	ADCK0
R/W	R/W	R/W	—	R/W	—	R/W	R/W	R/W
POR	0	0	—	0	—	0	0	0

Bit 7 **ACS4**: Selecte Internal 1.25V as ADC input Control
0: Disable
1: Enable
This bit enables 1.25V to be connected to the A/D converter. The V125EN bit must first have been set to enable the bandgap circuit 1.25V voltage to be used by the A/D converter. When the ACS4 bit is set high, the bandgap 1.25V voltage will be routed to the A/D converter and the other A/D input channels disconnected.

Bit 6 **V125EN**: Internal 1.25V Control
0: Disable
1: Enable
This bit controls the internal Bandgap circuit on/off function to the A/D converter. When the bit is set high the bandgap voltage 1.25V can be used by the A/D converter. If 1.25V is not used by the A/D converter and the LVR/LVD function is disabled then the bandgap reference circuit will be automatically switched off to conserve power. When 1.25V is switched on for use by the A/D converter, a time t_{BG} should be allowed for the bandgap circuit to stabilise before implementing an A/D conversion.

Bit 5 unimplemented, read as "0"

Bit 4 **VREFS**: Selecte ADC reference voltage
0: Internal ADC power
1: VREF pin
This bit is used to select the reference voltage for the A/D converter. If the bit is high, then the A/D converter reference voltage is supplied on the external VREF pin. If the pin is low, then the internal reference is used which is taken from the power supply pin VDD.

Bit 3 unimplemented, read as "0"

Bit 2~0 **ADCK2, ADCK1, ADCK0**: Select ADC clock source
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: Undefined
These three bits are used to select the clock source for the A/D converter.

• **ACER Register**

Bit	7	6	5	4	3	2	1	0
Name	ACE7	ACE6	ACE5	ACE4	ACE3	ACE2	ACE1	ACE0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	1	1	1	1	1	1	1	1

Bit 7 **ACE7**: Define PA7 is A/D input or not
0: Not A/D input
1: A/D input, AN7

Bit 6 **ACE6**: Define PA6 is A/D input or not
0: Not A/D input
1: A/D input, AN6

Bit 5 **ACE5**: Define PA5 is A/D input or not
0: Not A/D input
1: A/D input, AN5

Bit 4 **ACE4**: Define PA4 is A/D input or not
0: Not A/D input
1: A/D input, AN4

Bit 3 **ACE3**: Define PA3 is A/D input or not
0: Not A/D input
1: A/D input, AN3

Bit 2 **ACE2**: Define PA2 is A/D input or not
0: Not A/D input
1: A/D input, AN2

Bit 1 **ACE1**: Define PA1 is A/D input or not
0: Not A/D input
1: A/D input, AN1

Bit 0 **ACE0**: Define PA0 is A/D input or not
0: Not A/D input
1: A/D input, AN0

A/D Operation

The START bit in the ADCR0 register is used to start and reset the A/D converter. When the microcontroller sets this bit from low to high and then low again, an analog to digital conversion cycle will be initiated. When the START bit is brought from low to high but not low again, the EOCB bit in the ADCR0 register will be set high and the analog to digital converter will be reset. It is the START bit that is used to control the overall start operation of the internal analog to digital converter.

The EOCB bit in the ADCR0 register is used to indicate when the analog to digital conversion process is complete. This bit will be automatically set to "0" by the microcontroller after a conversion cycle has ended. In addition, the corresponding A/D interrupt request flag will be set in the interrupt control register, and if the interrupts are enabled, an appropriate internal interrupt signal will be generated. This A/D internal interrupt signal will direct the program flow to the associated A/D internal interrupt address for processing. If the A/D internal interrupt is disabled, the microcontroller can be used to poll the EOCB bit in the ADCR0 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 ADCK2~ADCK0 bits in the ADCR1 register.

Although the A/D clock source is determined by the system clocky, f_{SYS}, and by bits ADCK2~ADCK0, there are some limitations on the maximum A/D clock source speed that can be selected. As the minimum value of permissible A/D clock period, tAD, is 0.5s, care must be taken for system clock frequencies equal to or greater than 4MHz. For example, if the system clock operates at a frequency of 4MHz, the ADCK2~ADCK0 bits should not be set to "000". Doing so will give A/D clock periods that are less than the minimum A/D clock period which may result in inaccurate A/D conversion values. Refer to the following table for examples, where values marked with an asterisk * show where, depending upon the device, special care must be taken, as the values may be less than the specified minimum A/D Clock Period.

f_{SYS}	A/D Clock Period (t_{AD})							
	ADCK2, ADCK1, ADCK0 = 000 (f_{SYS})	ADCK2, ADCK1, ADCK0 = 001 (f_{SYS}/2)	ADCK2, ADCK1, ADCK0 = 010 (f_{SYS}/4)	ADCK2, ADCK1, ADCK0 = 011 (f_{SYS}/8)	ADCK2, ADCK1, ADCK0 = 100 (f_{SYS}/16)	ADCK2, ADCK1, ADCK0 = 101 (f_{SYS}/32)	ADCK2, ADCK1, ADCK0 = 110 (f_{SYS}/64)	ADCK2, ADCK1, ADCK0 = 111
1MHz	1us	2 us	4 us	8 us	16 us	32 us	64 us	Undefined
2MHz	500ns	1 us	2 us	4 us	8 us	16 us	32 us	Undefined
4MHz	250ns*	500ns	1 us	2 us	4 us	8 us	16 us	Undefined
8MHz	125ns*	250ns*	500ns	1 us	2 us	4 us	8 us	Undefined
12MHz	83ns*	167ns*	333ns*	667ns	1.33 us	2.67 us	5.33 us	Undefined

A/D Clock Period Examples

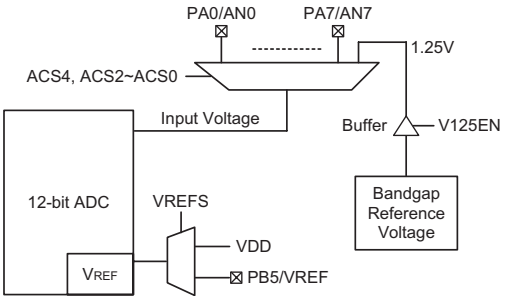
Controlling the power on/off function of the A/D converter circuitry is implemented using the ADOFF bit in the ADCR0 register. This bit must be zero to power on the A/D converter. Even if no pins are selected for use as A/D inputs by clearing the ACE7~ACE0 bits in the ACER register, if the ADOFF bit is zero then some power will still be consumed. In power conscious applications it is therefore recommended that the ADOFF is set high to reduce power consumption when the A/D converter function is not being used.

The reference voltage supply to the A/D Converter can be supplied from either the positive power supply pin, VDD, or from an external reference sources supplied on pin VREF. The desired selection is made using the VREFS bit. As the VREF pin is pin-shared with other functions, when the VREFS bit is set high, the VREF pin function will be selected and the other pin functions will be disabled automatically.

A/D Input Pins

All of the A/D analog input pins are pin-shared with the I/O pins on Port A as well as other functions. The ACE7~ ACE0 bits in the ACER register, determine whether the input pins are setup as A/D converter analog inputs or whether they have other functions. If the ACE7~ ACE0 bits for its corresponding pin is set high then the pin will be setup to be an A/D converter input and the original pin functions disabled. In this way, pins can be changed under program control to change their function between A/D inputs and other functions. All pull-high resistors, which are setup through register programming, will be automatically disconnected if the pins are setup as A/D inputs. Note that it is not necessary to first setup the A/D pin as an input in the PAC port control register to enable the A/D input as when the ACE7~ ACE0 bits enable an A/D input, the status of the port control register will be overridden.

The A/D converter has its own reference voltage pin, VREF, however the reference voltage can also be supplied from the power supply pin, a choice which is made through the VREFS bit in the ADCR1 register. The analog input values must not be allowed to exceed the value of VREF.



A/D Input Structure

Summary of A/D Conversion Steps

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

- Step 1
  Select the required A/D conversion clock by correctly programming bits ADCK2~ADCK0 in the ADCR1 register.

- Step 2
  Enable the A/D by clearing the ADOFF bit in the ADCR0 register to zero.

- Step 3
  Select which channel is to be connected to the internal A/D converter by correctly programming the ACS4~ACS0 bits which are also contained in the ADCR1 and ADCR0 register.

- Step 4
  Select which pins are to be used as A/D inputs and configure them by correctly programming the ACE7~ACE0 bits in the ACER register.

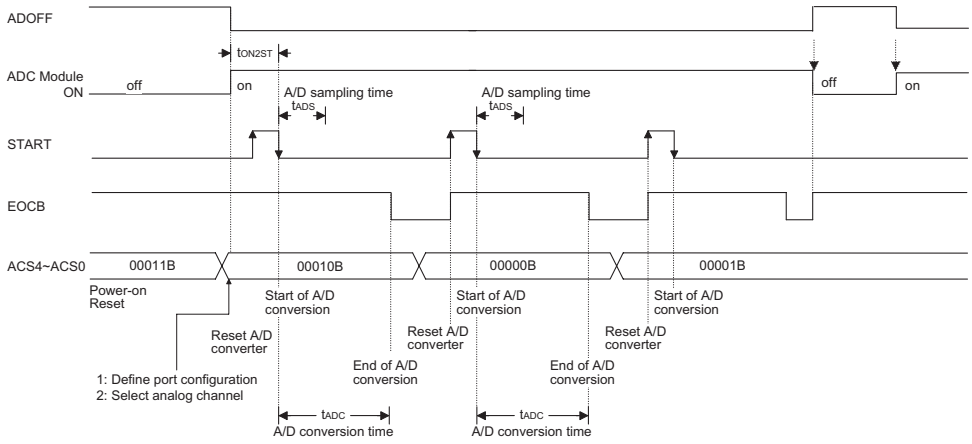
- Step 5
  If the interrupts are to be used, the interrupt control registers must be correctly configured to ensure the A/D converter interrupt function is active. The master interrupt control bit, EMI, and the A/D converter interrupt bit, EADI, must both be set high to do this.

- Step 6
  The analog to digital conversion process can now be initialised by setting the START bit in the ADCR register from low to high and then low again. Note that this bit should have been originally cleared to zero.

- Step 7
  To check when the analog to digital conversion process is complete, the EOCB bit in the ADCR0 register can be polled. The conversion process is complete when this bit goes low. When this occurs the A/D data registers ADRL and ADRH can be read to obtain the conversion value. As an alternative method, if the interrupts are enabled and the stack is not full, the program can wait for an A/D interrupt to occur.

  Note: When checking for the end of the conversion process, if the method of polling the EOCB bit in the ADCR0 register is used, the interrupt enable step above can be omitted.

  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 $16t_{AD}$ where t_{AD} is equal to the A/D clock period.



A/D Conversion Timing

Programming Considerations

During microcontroller operations where the A/D converter is not being used, the A/D internal circuitry can be switched off to reduce power consumption, by setting bit ADOFF high in the ADCR0 register. When this happens, the internal A/D converter circuits will not consume power irrespective of what analog voltage is applied to their input lines. If the A/D converter input lines are used as normal I/Os, then care must be taken as if the input voltage is not at a valid logic level, then this may lead to some increase in power consumption.

A/D Transfer Function

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

$$1\ LSB = (V_{DD} \text{ or } V_{REF}) \div 4096$$

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

$$A/D\ output\ digital\ value \times (V_{DD} \text{ or } V_{REF}) \div 4096$$

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_{DD} or V_{REF} level.

A/D Programming Example

The following two programming examples illustrate how to setup and implement an A/D conversion. In the first example, the method of polling the EOCB bit in the ADCR0 register is used to detect when the conversion cycle is complete, whereas in the second example, the A/D interrupt is used to determine when the conversion is complete.

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

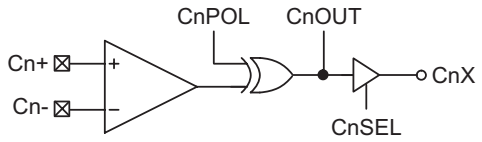
```
    clr     EADI            ; disable ADC interrupt
    mov     a,03H
    mov     ADCR1,a         ; select fSYS/8 as A/D clock and switch off 1.25V
    clr     ADOFF
    mov     a,0Fh           ; setup ACER to configure pins AN0~AN3
    mov     ACER,a
    mov     a,00h
    mov     ADCR0,a         ; enable and connect AN0 channel to A/D converter
    :
start_conversion:
    clr     START           ; high pulse on start bit to initiate conversion
    set     START           ; reset A/D
    clr     START           ; start A/D
polling_EOC:
    sz      EOCB            ; poll the ADCR0 register EOCB bit to detect end
                            ; of A/D conversion
    jmp polling_EOC         ; continue polling
    mov     a,ADRL          ; read low byte conversion result value
    mov     ADRL_buffer,a   ; save result to user defined register
    mov     a,ADRH          ; read high byte conversion result value
    mov     ADRH_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     EADI            ; disable ADC interrupt
    mov     a,03H
    mov     ADCR1,a         ; select fSYS/8 as A/D clock and switch off 1.25V
    Clr     ADOFF
    mov     a,0Fh           ; setup ACER to configure pins AN0~AN3
    mov     ACER,a
    mov     a,00h
    mov     ADCR0,a         ; enable and connect AN0 channel to A/D converter
Start_conversion:
    clr     START           ; high pulse on START bit to initiate conversion
    set     START           ; reset A/D
    clr     START           ; start A/D
    clr     ADF             ; clear ADC interrupt request flag
    set     EADI            ; 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,ADRL          ; read low byte conversion result value
    mov     adrl_buffer,a   ; save result to user defined register
    mov     a,ADRH          ; read high byte conversion result value
    mov     adrh_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
```

Comparator

A single independent analog comparator is contained within this device. This function offers flexibility via its register controlled features such as power-down, polarity select, hysteresis etc. In sharing its pins with normal I/O pins the comparator does not waste precious I/O pins if there functions are otherwise unused.



Comparator

Comparator Operation

The device contains a single comparator which is used to compare two analog voltages and provide an output based on their difference. Full control over the one internal comparator is provided via the control register, CPC. The comparator output is recorded via a bit in their respective control register, but can also be transferred out onto a shared I/O pin. Additional comparator functions include, output polarity, hysteresis functions and power down control.

Any pull-high resistors connected to the shared comparator input pins will be automatically disconnected when the comparator is enabled. As the comparator inputs approach their switching level, some spurious output signals may be generated on the comparator output due to the slow rising or falling nature of the input signals. This can be minimised by selecting the hysteresis function will apply a small amount of positive feedback to the comparator. Ideally the comparator should switch at the point where the positive and negative inputs signals are at the same voltage level, however, unavoidable input offsets introduce some uncertainties here. The hysteresis function, if enabled, also increases the switching offset value.

Comparator Registers

There is one register for overall comparator operation.

- **CPC Register**

Bit	7	6	5	4	3	2	1	0
Name	CSEL	CEN	CPOL	COUT	COS	—	—	CHYEN
R/W	R/W	R/W	R/W	R	R/W	—	—	R/W
POR	1	0	0	0	0	—	—	1

Bit 7 **CSEL**: Select Comparator pins or I/O pins
0: I/O pin select
1: Comparator pin select
This is the Comparator pin or I/O pin select bit. If the bit is high the comparator will be selected and the two comparator input pins will be enabled. As a result, these two pins will lose their I/O pin functions. Any pull-high configuration options associated with the comparator shared pins will also be automatically disconnected.

Bit 6 **CEN**: Comparator On/Off control
0: Off
1: On
This is the Comparator on/off control bit. If the bit is zero the comparator will be switched off and no power consumed even if analog voltages are applied to its inputs. For power sensitive applications this bit should be cleared to zero if the comparator is

not used or before the device enters the SLEEP or IDLE mode.

Bit 5 **CPOL**: Comparator output polarity
0: output not inverted
1: output inverted
This is the comparator polarity bit. If the bit is zero then the COUT bit will reflect the non-inverted output condition of the comparator. If the bit is high the comparator COUT bit will be inverted.

Bit 4 **COUT**: Comparator output bit
CPOL=0
0: C+ < C-
1: C+ > C-
CPOL=1
0: C+ > C-
1: C+ < C-
This bit stores the comparator output bit. The polarity of the bit is determined by the voltages on the comparator inputs and by the condition of the CPOL bit.

Bit 3 **COS**: Output path select
0: CX pin
1: Internal use
This is the comparator output path select control bit. If the bit is set to "0"and the CSEL bit is "1" the comparator output is connected to an external CX pin. If the bit is set to "1" or the CSEL bit is "0" the comparator output signal is only used internally by the device allowing the shared comparator output pin to retain its normal I/O operation.

Bit 2~1 unimplemented, read as "0"

Bit 0 **CHYEN**: Hysteresis Control
0: Off
1: On
This is the hysteresis control bit and if set high will apply a limited amount of hysteresis to the comparator, as specified in the Comparator Electrical Characteristics table. The positive feedback induced by hysteresis reduces the effect of spurious switching near the comparator threshold.

Comparator Interrupt

The comparator also possesses an interrupt function. When the comparator changes state, its relevant interrupt flag will be set, and if the corresponding interrupt enable bit is set, then a jump to its relevant interrupt vector will be executed. Note that it is the changing state of the C0OUT bit and not the output pin which generates an interrupt. If the microcontroller is in the SLEEP or IDLE Mode and the Comparator is enabled, then if the external input lines cause the Comparator output to change state, the resulting generated interrupt flag will also generate a wake-up. If it is required to disable a wake-up from occurring, then the interrupt flag should be first set high before entering the SLEEP or IDLE Mode.

Programming Considerations

If the comparator is enabled, it will remain active when the microcontroller enters the SLEEP or IDLE Mode, however as it will consume a certain amount of power, the user may wish to consider disabling it before the SLEEP or IDLE Mode is entered.

As comparator pins are shared with normal I/O pins the I/O registers for these pins will be read as zero (port control register is "1") or read as port data register value (port control register is "0") if the comparator function is enabled.

Serial Interface Module – SIM

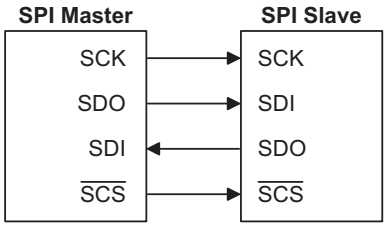
This device contains two Serial Interface Modules, SIM0 and SIM1, which include both the four line SPI interface and the two line I^2C interface types, to allow an easy method of communication with external peripheral hardware. Having relatively simple communication protocols, these serial interface types allow the microcontroller to interface to external SPI or I^2C based hardware such as sensors, Flash or EEPROM memory, etc. Note that the SIM1 interface function must first be selected using a configuration option. As both interface types share the same pins and registers, the choice of whether the SPI or I^2C type is used is made using the SIMn operating mode control bits, named SnSIM2~SnSIM0, in the SIMnC0 register. The index "n", shown as "0" or "1", is used to distinguish these two SIM modules. These pull-high resistors of the SIM1 pin-shared I/O are selected using pull-high control registers, and also if the SIM1 function is enabled.

SPIn Interface

The SPIn interface is often used to communicate with external peripheral devices such as sensors, Flash or EEPROM memory devices etc. Originally developed by Motorola, the four line SPIn interface is a synchronous serial data interface that has a relatively simple communication protocol simplifying the programming requirements when communicating with external hardware devices.

The communication is full duplex and operates as a slave/master type, where the device can be either master or slave. Although the SPIn interface specification can control multiple slave devices from a single master, but this device provided only one SCSn pin. If the master needs to control multiple slave devices from a single master, the master can use I/O pin to select the slave devices.

- SPIn Interface Operation
  The SPIn interface is a full duplex synchronous serial data link. It is a four line interface with pin names SDIn, SDOn, SCKn and SCSn. Pins SDIn and SDOn are the Serial Data Input and Serial Data Output lines, SCKn is the Serial Clock line and SCSn is the Slave Select line. As the SPI1 interface pins are pin-shared with normal I/O pins and with the I^2C1 function pins, the SPI1 interface must first be enabled by selecting the SIM1 enable configuration option and setting the correct bits in the SIM1C0 and SIM1C2 registers. After the SPI1 configuration option has been configured it can also be additionally disabled or enabled using the S1SIMEN bit in the SIM1C0 register. About the SPI0, the interface pins are pin-shared with the I^2C0 function pins, therefore, the SPI0 interface must be enabled by setting the correct bits in the SIM0C0 and SIM0C2 registers.Communication between devices connected to the SPIn interface is carried out in a slave/master mode with all data transfer initiations being implemented by the master. The Master also controls the clock signal. As the device only contains a single SCSn pin only one slave device can be utilized. The SCSn pin is controlled by software, set SnCSEN bit to "1" to enable SCSn pin function, set SnCSEN bit to "0" the SCSn pin will be floating state.



SPI Master/Slave Connection

The SPIn function in this device offers the following features:

- Full duplex synchronous data transfer
- Both Master and Slave modes
- LSB first or MSB first data transmission modes
- Transmission complete flag
- Rising or falling active clock edge
- SnWCOL and SnCSEN bit enabled or disable select

The status of the SPIn interface pins is determined by a number of factors such as whether the device is in the master or slave mode and upon the condition of certain control bits such as SnCSEN and SnSIMEN.

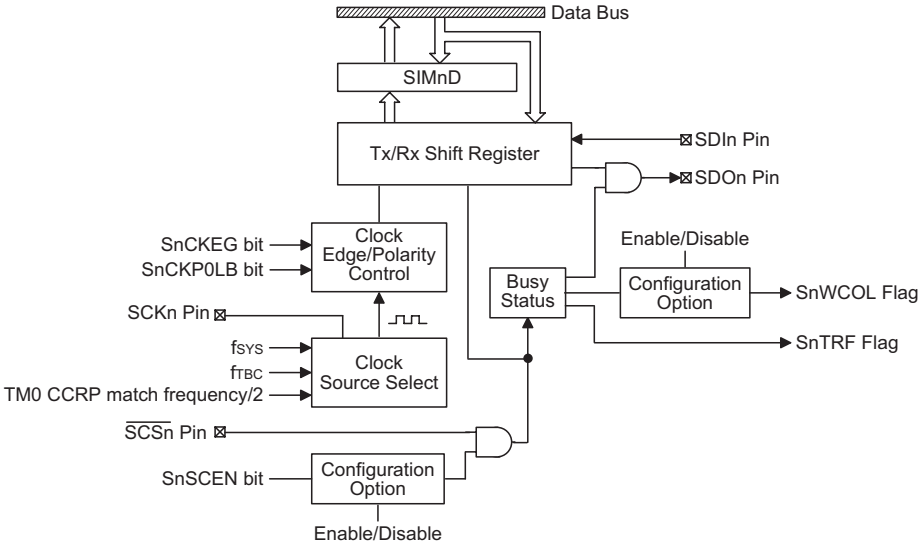
There are several configuration options associated with the SPIn interface. One of these is to enable the SIM1 function which selects the SIM1 pins rather than normal I/O pins. Note that if the configuration option does not select the SIM1 function then the S1SIMEN bit in the SIM1C0 register will have no effect. Another two SPIn configuration options determine if the SnCSEN and SnWCOL bits are to be used.

SPIn Registers

There are several internal registers which control the overall operation of the SPIn interface. These are the SIMnD data register and two registers SIMnC0 and SIMnC2. Note that the SIMnC1 register is only used by the I^2Cn interface.

Register Name	Bit							
	7	6	5	4	3	2	1	0
SIM0C0	S0SIM2	S0SIM1	S0SIM0	—	—	—	S0SIMEN	—
SIM1C0	S1SIM2	S1SIM1	S1SIM0	S1PCKEN	S1PCKP1	S1PCKP0	S1SIMEN	—
SIMnD	SnD7	SnD6	SnD5	SnD4	SnD3	SnD2	SnD1	SnD0
SIMnC2	SnD7	SnD6	SnCKPOLB	SnCKEG	SnMLS	SnCSEN	SnWCOL	SnTRF

Note: n="0" or "1"



SPIn Block Diagram

SIMn Registers List

The SIMnD register is used to store the data being transmitted and received. The same register is used by both the SPIn and I^2Cn functions. Before the device writes data to the SPIn bus, the actual data to be transmitted must be placed in the SIMnD register. After the data is received from the SPIn bus, the device can read it from the SIMnD register. Any transmission or reception of data from the SPIn bus must be made via the SIMnD register.

- **SIMnD Register**

Bit	7	6	5	4	3	2	1	0
Name	SnD7	SnD6	SnD5	SnD4	SnD3	SnD2	SnD1	SnD0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	x	x	x	x	x	x	x	x

"X" unknown

There are also two control registers for the SPIn interface, SIMnC0 and SIMnC2. Note that the SIMnC2 register also has the name SIMnA which is used by the I^2Cn function. The SIMnC1 register is not used by the SPIn function, only by the I^2Cn function. Register SIMnC0 is used to control the enable/disable function and to set the data transmission clock frequency. Although not connected with the SPI function, the SIM1C0 register is also used to control the Peripheral Clock Prescaler. Register SIMnC2 is used for other control functions such as LSB/MSB selection, write collision flag etc.

- **SIM0C0 Register**

Bit	7	6	5	4	3	2	1	0
Name	S0SIM2	S0SIM1	S0SIM0	—	—	—	S0SIMEN	—
R/W	R/W	R/W	R/W	—	—	—	R/W	—
POR	1	1	1	0	0	0	0	—

Bit 7~5 **S0SIM2, S0SIM1, S0SIM0**: SIM0 Operating Mode Control
000: SPI0 master mode; SPI0 clock is $f_{SYS}/4$
001: SPI0 master mode; SPI0 clock is $f_{SYS}/16$
010: SPI0 master mode; SPI0 clock is $f_{SYS}/64$
011: SPI0 master mode; SPI0 clock is f_{TBC}
100: SPI0 master mode; SPI0 clock is TM0 CCRP match frequency/2
101: SPI0 slave mode
110: I^2C0 slave mode
111: Unused mode

These bits setup the overall operating mode of the SIM0 function. As well as selecting if the I^2C0 or SPI0 function, they are used to control the SPI0 Master/Slave selection and the SPI0 Master clock frequency. The SPI0 clock is a function of the system clock but can also be chosen to be sourced from the TM0. If the SPI0 Slave Mode is selected then the clock will be supplied by an external Master device.

Bit 4~2 unimplemented, read as "0"

Bit 1 **S0SIMEN**: SIM0 Control
0: Disable
1: Enable

The bit is the overall on/off control for the SIM0 interface. When the S0SIMEN bit is cleared to zero to disable the SIM0 interface, the SDI0, SDO0, SCK0 and SCS0, or SDA0 and SCL0 lines will be in a floating condition and the SIM0 operating current will be reduced to a minimum value. When the bit is high the SIM0 interface is enabled. The SIM0 configuration option must have first enabled the SIM0 interface for this bit to be effective. If the SIM0 is configured to operate as an SPI0 interface via the S0SIM2~S0SIM0 bits, the contents of the SPI0 control registers will remain at the previous settings when the S0SIMEN bit changes from low to high and should

therefore be first initialised by the application program. If the SIM0 is configured to operate as an I²C0 interface via the S0SIM2~S0SIM0 bits and the S0SIMEN bit changes from low to high, the contents of the I²C0 control bits such as S0HTX and S0TXAK will remain at the previous settings and should therefore be first initialised by the application program while the relevant I²C0 flags such as S0HCF, S0HAAS, S0HBB, S0SRW and S0RXAK will be set to their default states.

Bit 0 unimplemented, read as "0"

- **SIM1C0 Register**

Bit	7	6	5	4	3	2	1	0
Name	S1SIM2	S1SIM1	S1SIM0	PCKEN	PCKP1	PCKP0	S1SIMEN	—
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	—
POR	1	1	1	0	0	0	0	0

Bit 7~5 **S1SIM2, S1SIM1, S1SIM0**: SIM1 Operating Mode Control
000: SPI1 master mode; SPI1 clock is $f_{SYS}/4$
001: SPI1 master mode; SPI1 clock is $f_{SYS}/16$
010: SPI1 master mode; SPI1 clock is $f_{SYS}/64$
011: SPI1 master mode; SPI1 clock is f_{TBC}
100: SPI1 master mode; SPI1 clock is TM0 CCRP match frequency/2
101: SPI1 slave mode
110: I²C1 slave mode
111: Unused mode

These bits setup the overall operating mode of the SIM1 function. As well as selecting if the I²C1 or SPI1 function, they are used to control the SPI1 Master/Slave selection and the SPI1 Master clock frequency. The SPI1 clock is a function of the system clock but can also be chosen to be sourced from the TM0. If the SPI1 Slave Mode is selected then the clock will be supplied by an external Master device.

Bit 4 **PCKEN**: PCK Output Pin Control
0: Disable
1: Enable

Bit 3~2 **PCKP1, PCKP0**: Select PCK output pin frequency
00: f_{SYS}
01: $f_{SYS}/4$
10: $f_{SYS}/8$
11: TM0 CCRP match frequency/2

Bit 1 **S1SIMEN**: SIM1 Control
0: Disable
1: Enable

The bit is the overall on/off control for the SIM1 interface. When the S1SIMEN bit is cleared to zero to disable the SIM1 interface, the SDI1, SDO1, SCK1 and SCS1, or SDA1 and SCL1 lines will be in a floating condition and the SIM1 operating current will be reduced to a minimum value. When the bit is high the SIM1 interface is enabled. The SIM1 configuration option must have first enabled the SIM1 interface for this bit to be effective. If the SIM1 is configured to operate as an SPI1 interface via the S1SIM2~S1SIM0 bits, the contents of the SPI1 control registers will remain at the previous settings when the S1SIMEN bit changes from low to high and should therefore be first initialised by the application program. If the SIM1 is configured to operate as an I²C1 interface via the S1SIM2~S1SIM0 bits and the S1SIMEN bit changes from low to high, the contents of the I²C1 control bits such as S1HTX and S1TXAK will remain at the previous settings and should therefore be first initialised by the application program while the relevant I²C1 flags such as S1HCF, S1HAAS, S1HBB, S1SRW and S1RXAK will be set to their default states.

Bit 0 unimplemented, read as "0"

• **SIMnC2 Register**

Bit	7	6	5	4	3	2	1	0
Name	SnD7	SnD6	SnCKPOLB	SnCKEG	SnMLS	SnCSEN	SnWCOL	SnTRF
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 Undefined bit
This bit can be read or written by user software program.

Bit 5 **SnCKPOLB**: Determines the base condition of the clock line
0: the SCKn line will be high when the clock is inactive
1: the SCKn line will be low when the clock is inactive
The SnCKPOLB bit determines the base condition of the clock line, if the bit is high, then the SCKn line will be low when the clock is inactive. When the SnCKPOLB bit is low, then the SCKn line will be high when the clock is inactive.

Bit 4 **SnCKEG**: Determines SPIn SCKn active clock edge type
SnCKPOLB=0
0: SCKn is high base level and data capture at SCKn rising edge
1: SCKn is high base level and data capture at SCKn falling edge
SnCKPOLB=1
0: SCKn is low base level and data capture at SCKn falling edge
1: SCKn is low base level and data capture at SCKn rising edge
The SnCKEG and SnCKPOLB bits are used to setup the way that the clock signal outputs and inputs data on the SPIn bus. These two bits must be configured before data transfer is executed otherwise an erroneous clock edge may be generated. The SnCKPOLB bit determines the base condition of the clock line, if the bit is high, then the SCKn line will be low when the clock is inactive. When the SnCKPOLB bit is low, then the SCKn line will be high when the clock is inactive.
The SnCKEG bit determines active clock edge type which depends upon the condition of SnCKPOLB bit.

Bit 3 **SnMLS**: SPIn Data shift order
0: LSB
1: MSB
This is the data shift select bit and is used to select how the data is transferred, either MSB or LSB first. Setting the bit high will select MSB first and low for LSB first.

Bit 2 **SnCSEN**: SPIn SCSn pin Control
0: Disable
1: Enable
The SnCSEN bit is used as an enable/disable for the SCSn pin. If this bit is low, then the SCSn pin will be disabled and placed into a floating condition. If the bit is high the SCSn pin will be enabled and used as a select pin.
Note that using the SnCSEN bit can be disabled or enabled via configuration option.

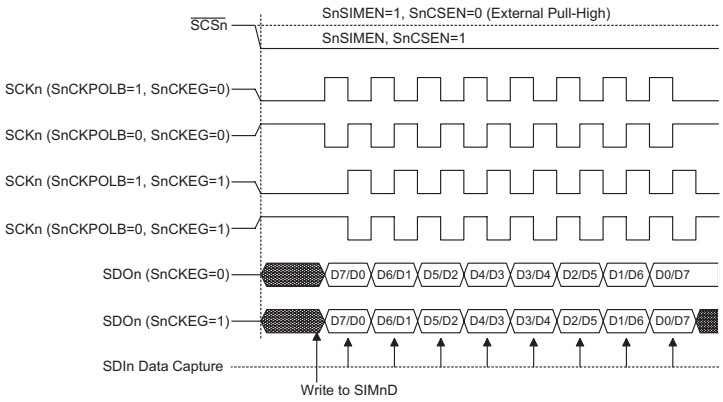
Bit 1 **SnWCOL**: SPIn Write Collision flag
0: No collision
1: Collision
The SnWCOL flag is used to detect if a data collision has occurred. If this bit is high it means that data has been attempted to be written to the SIMnD register during a data transfer operation. This writing operation will be ignored if data is being transferred. The bit can be cleared by the application program. Note that using the SnWCOL bit can be disabled or enabled via configuration option.

Bit 0 **SnTRF**: SPIn Transmit/Receive Complete flag
0: Data is being transferred
1: SPI data transmission is completed
The SnTRF bit is the Transmit/Receive Complete flag and is set “1”automatically when an SPIn data transmission is completed, but must set to “0” by the application program. It can be used to generate an interrupt.

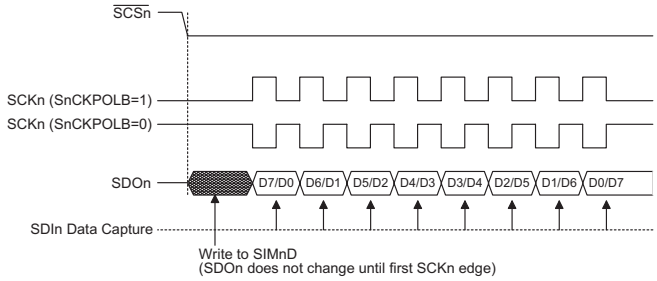
SPI Communication

After the SPIn interface is enabled by setting the SnSIMEN bit high, then in the Master Mode, when data is written to the SIMnD register, transmission/reception will begin simultaneously. When the data transfer is complete, the SnTRF flag will be set automatically, but must be cleared using the application program. In the Slave Mode, when the clock signal from the master has been received, any data in the SIMnD register will be transmitted and any data on the SDIn pin will be shifted into the SIMnD register. The master should output an SCS signal to enable the slave device before a clock signal is provided. The slave data to be transferred should be well prepared at the appropriate moment relative to the SCS signal depending upon the configurations of the SnCKPOLB bit and SnCKEG bit. The accompanying timing diagram shows the relationship between the slave data and SnSCS signal for various configurations of the SnCKPOLB and SnCKEG bits.

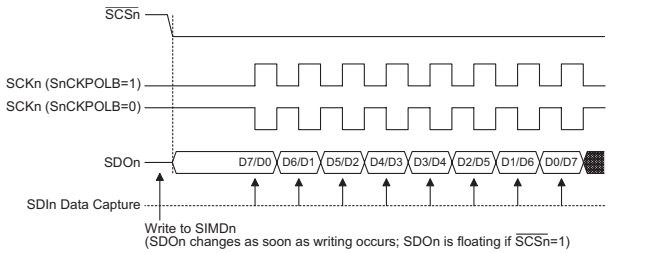
The SPIn will continue to function even in the IDLE Mode.



SPIn Master Mode Timing

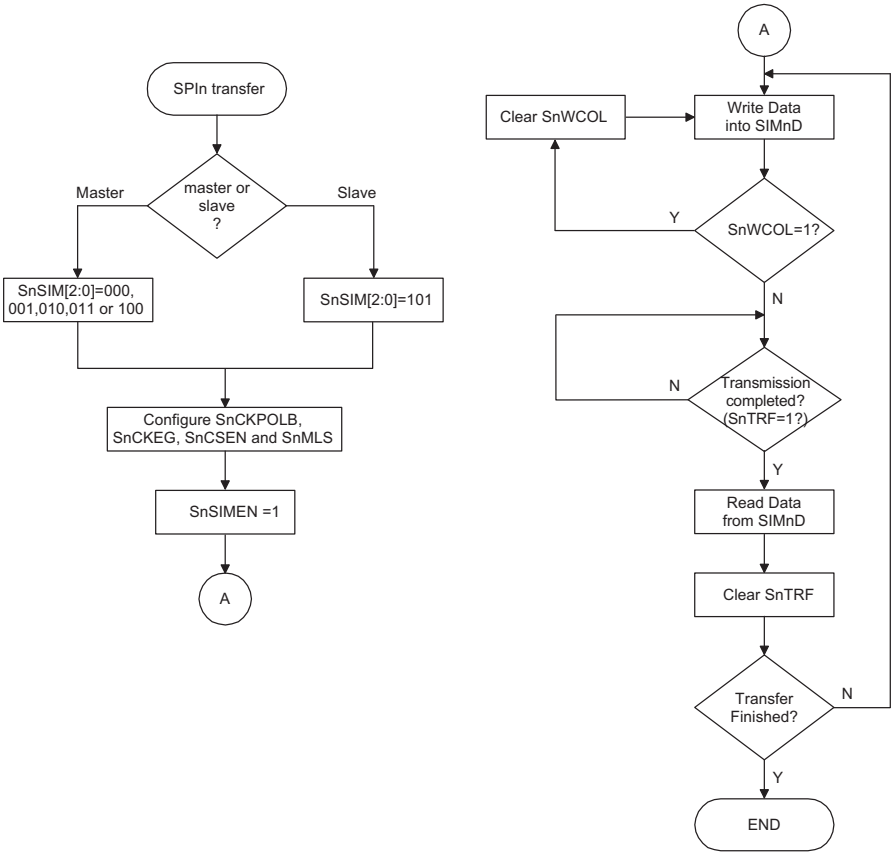


SPIn Slave Mode Timing – SnCKEG=0



Note: For SPIn slave mode, if SnSIMEN=1 and SnCSEN=0, SPIn is always enabled and ignores the $\overline{SCSn}$ level.

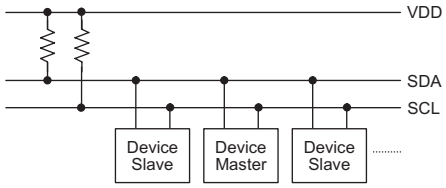
SPIn Slave Mode Timing – SnCKEG=1



SPIn Transfer Control Flowchart

I²C Interface

The I²C interface is used to communicate with external peripheral devices such as sensors, EEPROM memory etc. Originally developed by Philips, it is a two line low speed serial interface for synchronous serial data transfer. The advantage of only two lines for communication, relatively simple communication protocol and the ability to accommodate multiple devices on the same bus has made it an extremely popular interface type for many applications.



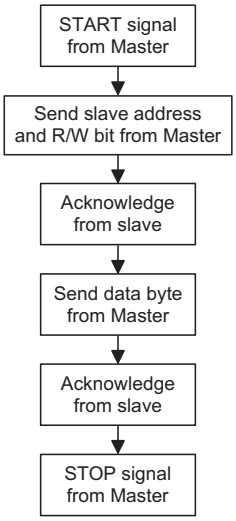
I²C Master Slave Bus Connection

I²C Interface Operation

The I²C serial interface is a two line interface, a serial data line, SDA, and serial clock line, SCL. As many devices may be connected together on the same bus, their outputs are both open drain types. For this reason it is necessary that external pull-high resistors are connected to these outputs. Note that no chip select line exists, as each device on the I²C bus is identified by a unique address which will be transmitted and received on the I²C bus.

When two devices communicate with each other on the bidirectional I²C bus, one is known as the master device and one as the slave device. Both master and slave can transmit and receive data, however, it is the master device that has overall control of the bus. For these devices, which only operates in slave mode, there are two methods of transferring data on the I²C bus, the slave transmit mode and the slave receive mode.

There are several configuration options associated with the I²Cn interface. One of these is to enable the function which selects the SIM1 pins rather than normal I/O pins. Note that if the configuration option does not select the SIM1 function then the S1SIMEN bit in the SIM1C0 register will have no effect. A configuration option exists to allow a clock other than the system clock to drive the I²Cn interface. Another configuration option determines the debounce time of the I²Cn interface. This uses the internal clock to in effect add a debounce time to the external clock to reduce the possibility of glitches on the clock line causing erroneous operation. The debounce time, if selected, can be chosen to be either 1 or 2 system clocks.



I²Cn Registers

There are several control registers associated with the I²Cn bus, SIMnC0, SIMnC1 and SIMnA and data register, SIMnD. The SIMnD register, which is shown in the above SPIn section, is used to store the data being transmitted and received on the I²Cn bus. Before the microcontroller writes data to the I²Cn bus, the actual data to be transmitted must be placed in the SIMnD register. After the data is received from the I²Cn bus, the microcontroller can read it from the SIMnD register. Any transmission or reception of data from the I²Cn bus must be made via the SIMnD register.

Note that the SIMnA register also has the name SIMnC2 which is used by the SPIn function. Bit SnSIMEN and bits SnSIM2~SnSIM0 in register SIMnC0 are used by the I²C interface.

Register Name	Bit							
	7	6	5	4	3	2	1	0
SIM0C0	S0SIM2	S0SIM1	S0SIM0	—	—	—	S0SIMEN	—
SIM1C0	SnSIM2	SnSIM1	SnSIM0	PCKEN	PCKP1	PCKP0	SIMEN	—
SIMnC1	SnHCF	SnHANS	SnHBB	SnHTX	SnTXAK	SnSRW	SnIAMWU	SnRXAK
SIMnD	SnD7	SnD6	SnD5	SnD4	SnD3	SnD2	SnD1	SnD0
SIMnA	SnIICA6	SnIICA5	SnIICA4	SnIICA3	SnIICA2	SnIICA1	SnIICA0	—

Note: n="0" or "1"

I²C Registers List

- **SIM0C0 Register**

Bit	7	6	5	4	3	2	1	0
Name	S0SIM2	S0SIM1	S0SIM0	—	—	—	S0SIMEN	—
R/W	R/W	R/W	R/W	—	—	—	R/W	—
POR	1	1	1	0	0	0	0	—

Bit 7~5 **S0SIM2, S0SIM1, S0SIM0**: SIM0 Operating Mode Control
000: SPI0 master mode; SPI0 clock is $f_{SYS}/4$
001: SPI0 master mode; SPI0 clock is $f_{SYS}/16$
010: SPI0 master mode; SPI0 clock is $f_{SYS}/64$
011: SPI0 master mode; SPI0 clock is f_{TBC}
100: SPI0 master mode; SPI0 clock is TM0 CCRP match frequency/2
101: SPI0 slave mode
110: I²C0 slave mode
111: Unused mode

These bits setup the overall operating mode of the SIM0 function. As well as selecting if the I²C0 or SPI0 function, they are used to control the SPI0 Master/Slave selection and the SPI0 Master clock frequency. The SPI0 clock is a function of the system clock but can also be chosen to be sourced from the TM0. If the SPI0 Slave Mode is selected then the clock will be supplied by an external Master device.

Bit 4~2 unimplemented, read as "0"

Bit 1 **S0SIMEN**: SIM0 Control
0: Disable
1: Enable

The bit is the overall on/off control for the SIM0 interface. When the S0SIMEN bit is cleared to zero to disable the SIM0 interface, the SDI0, SDO0, SCK0 and SCS0, or SDA0 and SCL0 lines will be in a floating condition and the SIM0 operating current will be reduced to a minimum value. When the bit is high the SIM0 interface is enabled. The SIM0 configuration option must have first enabled the SIM0 interface for this bit to be effective. If the SIM0 is configured to operate as an SPI0 interface via the S0SIM2~S0SIM0 bits, the contents of the SPI0 control registers will remain at the previous settings when the S0SIMEN bit changes from low to high and should therefore be first initialised by the application program. If the SIM0 is configured to operate as an I²C0 interface via the S0SIM2~S0SIM0 bits and the S0SIMEN bit changes from low to high, the contents of the I²C0 control bits such as S0HTX and S0TXAK will remain at the previous settings and should therefore be first initialised by the application program while the relevant I²C0 flags such as S0HCF, S0HAAS, S0HBB, S0SRW and S0RXAK will be set to their default states.

Bit 0 unimplemented, read as "0"

• **SIM1C0 Register**

Bit	7	6	5	4	3	2	1	0
Name	S1SIM2	S1SIM1	S1SIM0	PCKEN	PCKP1	PCKP0	S1SIMEN	—
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	—
POR	1	1	1	0	0	0	0	0

Bit 7~5 **S1SIM2, S1SIM1, S1SIM0**: SIM1 Operating Mode Control
000: SPI1 master mode; SPI1 clock is $f_{SYS}/4$
001: SPI1 master mode; SPI1 clock is $f_{SYS}/16$
010: SPI1 master mode; SPI1 clock is $f_{SYS}/64$
011: SPI1 master mode; SPI1 clock is f_{TBC}
100: SPI1 master mode; SPI1 clock is TM0 CCRP match frequency/2
101: SPI1 slave mode
110: I^2C1 slave mode
111: Unused mode
These bits setup the overall operating mode of the SIM1 function. As well as selecting if the I^2C1 or SPI1 function, they are used to control the SPI1 Master/Slave selection and the SPI1 Master clock frequency. The SPI1 clock is a function of the system clock but can also be chosen to be sourced from the TM0. If the SPI1 Slave Mode is selected then the clock will be supplied by an external Master device.

Bit 4 **PCKEN**: PCK Output Pin Control
0: Disable
1: Enable

Bit 3~2 **PCKP1, PCKP0**: Select PCK output pin frequency
00: f_{SYS}
01: $f_{SYS}/4$
10: $f_{SYS}/8$
11: TM0 CCRP match frequency/2

Bit 1 **S1SIMEN**: SIM1 Control
0: Disable
1: Enable
The bit is the overall on/off control for the SIM1 interface. When the S1SIMEN bit is cleared to zero to disable the SIM interface, the SDI1, SDO1, SCK1 and SCS1, or SDA1 and SCL1 lines will be in a floating condition and the SIM1 operating current will be reduced to a minimum value. When the bit is high the SIM1 interface is enabled. The SIM1 configuration option must have first enabled the SIM1 interface for this bit to be effective. If the SIM1 is configured to operate as an SPI1 interface via the S1SIM2~S1SIM0 bits, the contents of the SPI1 control registers will remain at the previous settings when the S1SIMEN bit changes from low to high and should therefore be first initialised by the application program. If the SIM1 is configured to operate as an I^2C1 interface via the S1SIM2~S1SIM0 bits and the S1SIMEN bit changes from low to high, the contents of the I^2C1 control bits such as S1HTX and S1TXAK will remain at the previous settings and should therefore be first initialised by the application program while the relevant I^2C1 flags such as S1HCF, S1HAAS, S1HBB, S1SRW and S1RXAK will be set to their default states.

Bit 0 unimplemented, read as “0”

• **SIMnC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	SnHCF	SnHANS	SnHBB	SnHTX	SnTXAK	SnSRW	SnIAMWU	SnRXAK
R/W	R	R	R	R/W	R/W	R	R/W	R
POR	1	0	0	0	0	0	0	1

Bit 7 **SnHCF**: I^2Cn Bus data transfer completion flag
0: Data is being transferred
1: Completion of an 8-bit data transfer
The SnHCF flag is the data transfer flag. This flag will be zero when data is being transferred. Upon completion of an 8-bit data transfer the flag will go high and an interrupt will be generated.

Bit 6 **SnHAAS**: I^2Cn Bus address match flag
0: Not address match
1: Address match
The SnHASS flag is the address match flag. This flag is used to determine if the slave device address is the same as the master transmit address. If the addresses match then this bit will be high, if there is no match then the flag will be low.

Bit 5 **SnHBB**: I^2Cn Bus busy flag
0: I^2Cn Bus is not busy
1: I^2Cn Bus is busy
The SnHBB flag is the I^2Cn busy flag. This flag will be "1" when the I^2Cn bus is busy which will occur when a START signal is detected. The flag will be set to "0" when the bus is free which will occur when a STOP signal is detected.

Bit 4 **SnHTX**: Select I^2Cn slave device is transmitter or receiver
0: Slave device is the receiver
1: Slave device is the transmitter

Bit 3 **SnTXAK**: I^2Cn Bus transmit acknowledge flag
0: Slave send acknowledge flag
1: Slave do not send acknowledge flag
The SnTXAK bit is the transmit acknowledge flag. After the slave device receipt of 8-bits of data, this bit will be transmitted to the bus on the 9th clock from the slave device. The slave device must always set SnTXAK bit to "0" before further data is received.

Bit 2 **SnSRW**: I^2Cn Slave Read/Write flag
0: Slave device should be in receive mode
1: Slave device should be in transmit mode
The SnSRW flag is the I^2Cn Slave Read/Write flag. This flag determines whether the master device wishes to transmit or receive data from the I^2Cn bus. When the transmitted address and slave address is match, that is when the SnHAAS flag is set high, the slave device will check the SnSRW flag to determine whether it should be in transmit mode or receive mode. If the SnSRW flag is high, the master is requesting to read data from the bus, so the slave device should be in transmit mode. When the SnSRW flag is zero, the master will write data to the bus, therefore the slave device should be in receive mode to read this data.

Bit 1 **SnIAMWU**: I^2Cn Address Match Wake-up Control
0: Disable
1: Enable
This bit should be set to "1" to enable I^2Cn address match wake up from SLEEP or IDLE Mode.

Bit 0 **SnRXAK**: I^2Cn Bus Receive acknowledge flag
0: Slave receive acknowledge flag
1: Slave do not receive acknowledge flag

The SnRXAK flag is the receiver acknowledge flag. When the SnRXAK flag is "0", it means that a acknowledge signal has been received at the 9th clock, after 8 bits of data have been transmitted. When the slave device in the transmit mode, the slave device checks the SnRXAK flag to determine if the master receiver wishes to receive the next byte. The slave transmitter will therefore continue sending out data until the SnRXAK flag is "1". When this occurs, the slave transmitter will release the SDAn line to allow the master to send a STOP signal to release the I²Cn Bus.

The SIMnD register is used to store the data being transmitted and received. The same register is used by both the SPIn and I²Cn functions. Before the device writes data to the SPIn bus, the actual data to be transmitted must be placed in the SIMnD register. After the data is received from the SPIn bus, the device can read it from the SIMnD register. Any transmission or reception of data from the SPIn bus must be made via the SIMnD register.

- **SIMnD Register**

Bit	7	6	5	4	3	2	1	0
Name	SnD7	SnD6	SnD5	SnD4	SnD3	SnD2	SnD1	SnD0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	x	x	x	x	x	x	x	x

"X" unknown

- **SIMnA Register**

Bit	7	6	5	4	3	2	1	0
Name	SnIICA6	SnIICA5	SnIICA4	SnIICA3	SnIICA2	SnIICA1	SnIICA0	—
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	x	x	x	x	x	x	x	0

"X" unknown

Bit 7~1 **IICA6~ IICA0**: I²Cn slave address

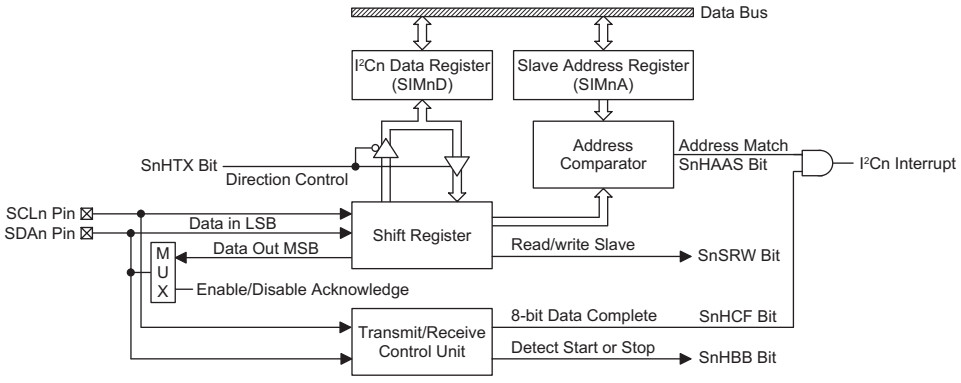
SnIICA6~ SnIICA0 is the I²Cn slave address bit 6~ bit 0.

The SIMnA register is also used by the SPIn interface but has the name SIMnC2. The SIMnA register is the location where the 7-bit slave address of the slave device is stored. Bits 7~ 1 of the SIMnA register define the device slave address. Bit 0 is not defined.

When a master device, which is connected to the I²Cn bus, sends out an address, which matches the slave address in the SIMnA register, the slave device will be selected. Note that the SIMnA register is the same register address as SIMnC2 which is used by the SPIn interface.

Bit 0 Undefined bit

This bit can be read or written by user software program.

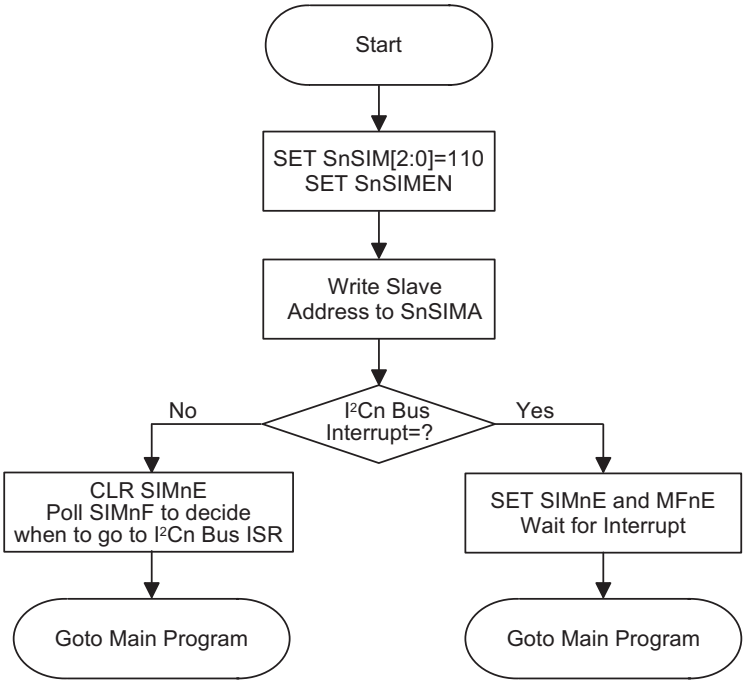


I²Cn Block Diagram

I²Cn Bus Communication

Communication on the I²Cn bus requires four separate steps, a START signal, a slave device address transmission, a data transmission and finally a STOP signal. When a START signal is placed on the I²Cn bus, all devices on the bus will receive this signal and be notified of the imminent arrival of data on the bus. The first seven bits of the data will be the slave address with the first bit being the MSB. If the address of the slave device matches that of the transmitted address, the SnHAAS bit in the SIMnC1 register will be set and an I²C interrupt will be generated. After entering the interrupt service routine, the slave device must first check the condition of the SnHAAS bit to determine whether the interrupt source originates from an address match or from the completion of an 8-bit data transfer. During a data transfer, note that after the 7-bit slave address has been transmitted, the following bit, which is the 8th bit, is the read/write bit whose value will be placed in the SnSRW bit. This bit will be checked by the slave device to determine whether to go into transmit or receive mode. Before any transfer of data to or from the I²C bus, the microcontroller must initialise the bus, the following are steps to achieve this:

- Step 1
  Set the SnSIM2~SnSIM0 and SnSIMEN bits in the SIMnC0 register to "1" to enable the I²C bus.
- Step 2
  Write the slave address of the device to the I²C bus address register SIMnA.
- Step 3
  Set the SIMnE and SIM Muti-Function interrupt enable bit of the interrupt control register to enable the SIMn interrupt and Multi-function interrupt.



I²Cn Bus Initialisation Flow Chart

I²Cn Bus Start Signal

The START signal can only be generated by the master device connected to the I²Cn bus and not by the slave device. This START signal will be detected by all devices connected to the I²Cn bus. When detected, this indicates that the I²Cn bus is busy and therefore the SnHBB bit will be set. A START condition occurs when a high to low transition on the SDAn line takes place when the SCLn line remains high.

Slave Address

The transmission of a START signal by the master will be detected by all devices on the I²Cn bus. To determine which slave device the master wishes to communicate with, the address of the slave device will be sent out immediately following the START signal. All slave devices, after receiving this 7-bit address data, will compare it with their own 7-bit slave address. If the address sent out by the master matches the internal address of the microcontroller slave device, then an internal I²Cn bus interrupt signal will be generated. The next bit following the address, which is the 8th bit, defines the read/write status and will be saved to the SnSRW bit of the SIMnC1 register. The slave device will then transmit an acknowledge bit, which is a low level, as the 9th bit. The slave device will also set the status flag SnHAAS when the addresses match.

As an I²Cn bus interrupt can come from two sources, when the program enters the interrupt subroutine, the SnHAAS bit should be examined to see whether the interrupt source has come from a matching slave address or from the completion of a data byte transfer. When a slave address is matched, the device must be placed in either the transmit mode and then write data to the SIMnD register, or in the receive mode where it must implement a dummy read from the SIMnD register to release the SCLn line.

I²Cn Bus Read/Write Signal

The SnSRW bit in the SIMnC1 register defines whether the slave device wishes to read data from the I²Cn bus or write data to the I²Cn bus. The slave device should examine this bit to determine if it is to be a transmitter or a receiver. If the SnSRW flag is "1" then this indicates that the master device wishes to read data from the I²Cn bus, therefore the slave device must be setup to send data to the I²Cn bus as a transmitter. If the SnSRW flag is "0" then this indicates that the master wishes to send data to the I²Cn bus, therefore the slave device must be setup to read data from the I²Cn bus as a receiver.

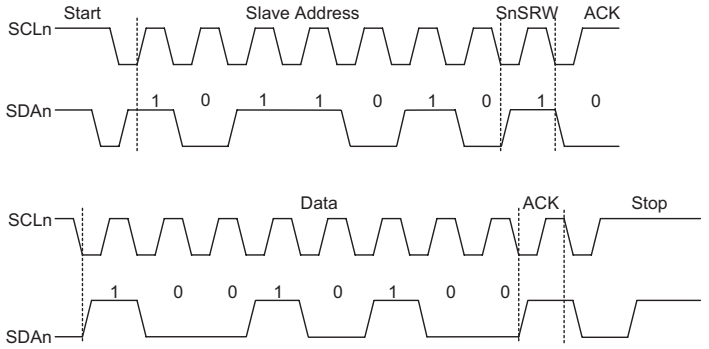
I²Cn Bus Slave Address Acknowledge Signal

After the master has transmitted a calling address, any slave device on the I²Cn bus, whose own internal address matches the calling address, must generate an acknowledge signal. The acknowledge signal will inform the master that a slave device has accepted its calling address. If no acknowledge signal is received by the master then a STOP signal must be transmitted by the master to end the communication. When the HAAS flag is high, the addresses have matched and the slave device must check the SnSRW flag to determine if it is to be a transmitter or a receiver. If the SnSRW flag is high, the slave device should be setup to be a transmitter so the SnHTX bit in the SIMnC1 register should be set to "1". If the SnSRW flag is low, then the microcontroller slave device should be setup as a receiver and the SnHTX bit in the SIMnC1 register should be set to "0".

I²C Bus Data and Acknowledge Signal

The transmitted data is 8-bits wide and is transmitted after the slave device has acknowledged receipt of its slave address. The order of serial bit transmission is the MSB first and the LSB last. After receipt of 8-bits of data, the receiver must transmit an acknowledge signal, level "0", before it can receive the next data byte. If the slave transmitter does not receive an acknowledge bit signal from the master receiver, then the slave transmitter will release the SDAn line to allow the master to send a STOP signal to release the I²Cn Bus. The corresponding data will be stored in the SIMnD register. If setup as a transmitter, the slave device must first write the data to be transmitted into the SIMnD register. If setup as a receiver, the slave device must read the transmitted data from the SIMnD register.

When the slave receiver receives the data byte, it must generate an acknowledge bit, known as SnTXAK, on the 9th clock. The slave device, which is setup as a transmitter will check the SnRXAK bit in the SIMnC1 register to determine if it is to send another data byte, if not then it will release the SDAn line and await the receipt of a STOP signal from the master.

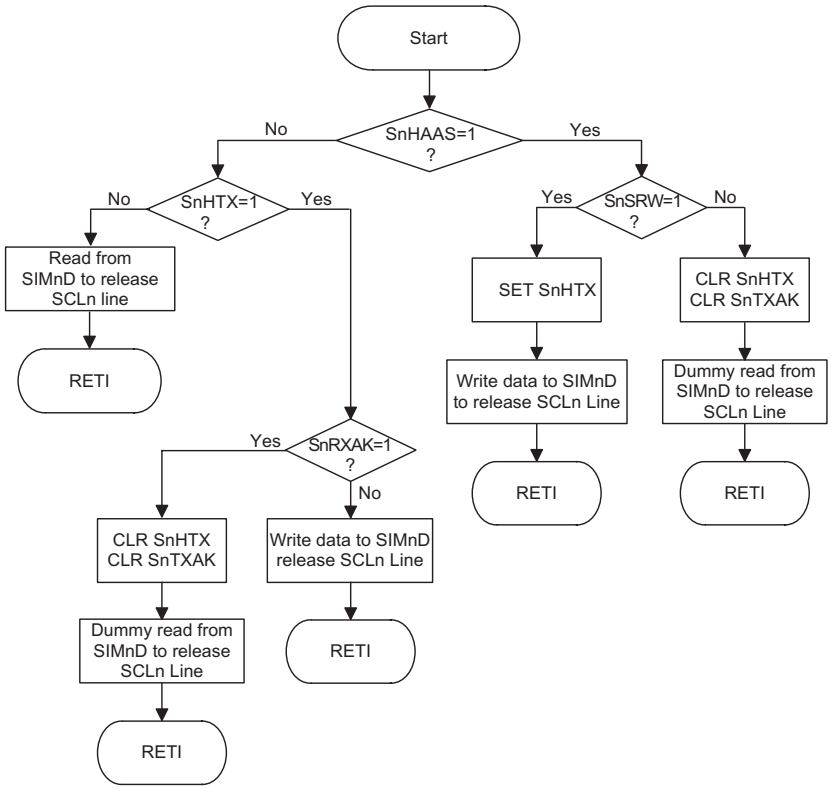


S=Start (1 bit)
SA=Slave Address (7 bits)
SR=SnSRW bit (1 bit)
M=Slave device send acknowledge bit (1 bit)
D=Data (8 bits)
A=ACK (SnRXAK bit for transmitter, SnTXAK bit for receiver 1 bit)
P=Stop (1 bit)

S	SA	SR	M	D	A	D	A	--------	S	SA	SR	M	D	A	D	A	--------	P

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

I²Cn Communication Timing Diagram



I²Cn Bus ISR Flow Chart

Peripheral Clock Output

The Peripheral Clock Output allows the device to supply external hardware with a clock signal synchronised to the microcontroller clock.

Peripheral Clock Operation

As the peripheral clock output pin, PCK, is shared with I/O line, the required pin function is chosen via PCKEN in the SIM1C0 register. The Peripheral Clock function is controlled using the SIM1C0 register. The clock source for the Peripheral Clock Output can originate from either the TM0 CCRP match frequency/2 or a divided ratio of the internal f_{SYS} clock. The PCKEN bit in the SIM1C0 register is the overall on/off control, setting PCKEN bit to "1" enables the Peripheral Clock, setting PCKEN bit to "0" disables it. The required division ratio of the system clock is selected using the PCKP1 and PCKP0 bits in the same register. If the device enters the SLEEP Mode this will disable the Peripheral Clock output.

- **SIM1C0 Register**

Bit	7	6	5	4	3	2	1	0
Name	S1SIM2	S1SIM1	S1SIM0	PCKEN	PCKP1	PCKP0	S1SIMEN	—
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	—
POR	1	1	1	0	0	0	0	—

Bit 7~5 **S1SIM2,S1SIM1, S1SIM0**: SIM1 operating mode control
000: SPI master mode; SPI clock is $f_{SYS}/4$
001: SPI master mode; SPI clock is $f_{SYS}/16$
010: SPI master mode; SPI clock is $f_{SYS}/64$
011: SPI master mode; SPI clock is f_{TBC}
100: SPI master mode; SPI clock is TM0 CCRP match frequency/2
101: SPI slave mode
110: I^2C slave mode
111: Unused mode

These bits setup the overall operating mode of the SIM1 function. As well as selecting if the I^2C1 or SPI1 function, they are used to control the SPI1 Master/Slave selection and the SPI1 Master clock frequency. The SPI1 clock is a function of the system clock but can also be chosen to be sourced from the TM0. If the SPI1 Slave Mode is selected then the clock will be supplied by an external Master device.

Bit 4 **PCKEN**: PCK output pin control
0: Disable
1: Enable

Bit 3~2 **PCKP1, PCKP0**: select PCK output pin frequency
00: f_{SYS}
01: $f_{SYS}/4$
10: $f_{SYS}/8$
11: TM0 CCRP match frequency/2

Bit 1 **S1SIMEN**: SIM1 control
0: Disable
1: Enable

The bit is the overall on/off control for the SIM1 interface. When the S1SIMEN bit is cleared to zero to disable the SIM1 interface, the SDI1, SDO1, SCK1 and SCS1, or SDA1 and SCL1 lines will be in a floating condition and the SIM1 operating current will be reduced to a minimum value. When the bit is high the SIM1 interface is enabled. The SIM1 configuration option must have first enabled the SIM1 interface for this bit to be effective. Note that when the S1SIMEN bit changes from low to high the contents of the SPI1 control registers will be in an unknown condition and should therefore be first initialised by the application program.

Bit 0 unimplemented, read as "0"

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 and PINT pins, while the internal interrupts are generated by various internal functions such as the TMs, Timer, Comparator, Time Base, LVD, SIMs and the A/D converter.

Interrupt Registers

Overall interrupt control, which basically means the setting of request flags when certain microcontroller conditions occur and the setting of interrupt enable bits by the application program, is controlled by a series of registers, located in the Special Purpose Data Memory, as shown in the accompanying table. The number of registers depends upon the device chosen but fall into three categories. The first is the INTC0~INTC3 registers which setup the primary interrupts, the second is the MFI0~MFI3 registers which setup the Multi-function interrupts. Finally there is an INTEG register to setup the external interrupt trigger edge type.

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

Function	Enable Bit	Request Flag	Notes
Global	EMI	—	—
Comparator	CPE	CPF	—
INTn Pin	INTnE	INTnF	n = 0 or 1
A/D Converter	ADE	ADF	—
Multi-function	MFnE	MFnF	n = 0~2
Time Base	TBnE	TBnF	n = 0 or 1
SIM	SIMnE	SIMnF	n = 0 or 1
LVD	LVE	LVF	—
PINT Pin	XPE	XPF	—
TMn0	TnPE	TnPF	n = 0
	TnAE	TnAF	
TMn1	TnPE	TnPF	n = 1
	TnAE	TnAF	
	TnBE	TnBF	

Interrupt Register Bit Naming Conventions

- **Interrupt Register Contents**

Name	Bit							
	7	6	5	4	3	2	1	0
INTEG	—	—	—	—	INT1S1	INT1S0	INT0S1	INT0S0
INTC0	—	CPF	INT1F	INT0F	CPE	INT1E	INT0E	EMI
INTC1	ADF	MF1F	MF0F	SIM0F	ADE	MF1E	MF0E	SIM0E
INTC2	LVF	TB1F	TB0F	MF2F	LVE	TB1E	TB0E	MF2E
INTC3	—	—	—	TF	—	—	—	TE
MFI0	—	—	T0AF	T0PF	—	—	T0AE	T0PE
MFI1	—	T1BF	T1AF	T1PF	—	T1BE	T1AE	T1PE
MFI2	—	—	XPF	SIM1F	—	—	XPE	SIM1E

• **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 edge control for INT1 pin
00: disable
01: rising edge
10: falling edge
11: rising and falling edges

Bit 1~0 **INT0S1, INT0S0**: interrupt edge control 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	—	CPF	INT1F	INT0F	CPE	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 **CPF**: Comparator 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 **CPE**: Comparator 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	ADF	MF1F	MF0F	SIM0F	ADE	MF1E	MF0E	SIM0E
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7 **ADF**: A/D Converter Interrupt Request Flag
0: no request
1: interrupt request

Bit 6 **MF1F**: Multi-function Interrupt 1 Request Flag
0: no request
1: interrupt request

Bit 5 **MF0F**: Multi-function Interrupt 0 Request Flag
0: no request
1: interrupt request

Bit 4 **SIM0F**: SIM 0 Interrupt Request Flag
0: no request
1: interrupt request

Bit 3 **ADE**: A/D Converter Interrupt Control
0: disable
1: enable

Bit 2 **MF1E**: Multi-function Interrupt 1 Control
0: disable
1: enable

Bit 1 **MF0E**: Multi-function Interrupt 0 Control
0: disable
1: enable

Bit 0 **SIM0E**: SIM 0 Interrupt Control
0: Disable
1: Enable

• **INTC2 Register**

Bit	7	6	5	4	3	2	1	0
Name	LVF	TB1F	TB0F	MF2F	LVE	TB1E	TB0E	MF2E
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 **LVF**: LVD Interrupt Request Flag
0: no request
1: interrupt request

Bit 6 **TB1F**: Time Base 1 Interrupt Request Flag
0: no request
1: interrupt request

Bit 5 **TB0F**: Time Base 0 IInterrupt Request Flag
0: no request
1: interrupt request

Bit 4 **MF2F**: Multi-function Interrupt 2 Request Flag
0: no request
1: interrupt request

Bit 3 **LVE**: LVD Interrupt Control
0: disable
1: enable

Bit 2 **TB1E**: Time Base 1 Interrupt Control
0: disable
1: enable

Bit 1 **TB0E**: Time Base 0 Interrupt Control
0: disable
1: enable

Bit 0 **MF2E**: Multi-function Interrupt 2 Control
0: disable
1: enable

• **INTC3 Register**

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

Bit 7~5 unimplemented, read as "0"

Bit 4 **TF**: TMR interrupt request flag
0: no request
1: interrupt request

Bit 3~1 unimplemented, read as “0”

Bit 0 **TE**: TMR interrupt control
0: disable
1: enable

• **MFI0 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	T0AF	T0PF	—	—	T0AE	T0PE
R/W	—	—	R/W	R/W	—	—	R/W	R/W
POR	—	—	0	0	—	—	0	0

Bit 7~6 unimplemented, read as "0"

Bit 5 **T0AF**: TM0 Comparator A match interrupt request flag
0: no request
1: interrupt request

Bit 4 **T0PF**: TM0 Comparator P match interrupt request flag
0: no request
1: interrupt request

Bit 3~2 unimplemented, read as “0”

Bit 1 **T0AE**: TM0 Comparator A match interrupt control
0: disable
1: enable

Bit 0 **T0PE**: TM0 Comparator P match interrupt control
0: disable
1: enable

• **MFI1 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	T1BF	T1AF	T1PF	—	T1BE	T1AE	T1PE
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 **T1BF**: TM1 Comparator B match interrupt request flag
0: no request
1: interrupt request

Bit 5 **T1AF**: TM1 Comparator A match interrupt request flag
0: no request
1: interrupt request

Bit 4 **T1PF**: TM1 Comparator B match interrupt request flag
0: no request
1: interrupt request

Bit 3 unimplemented, read as "0"

Bit 2 **T1BE**: TM1 Comparator P match interrupt control
0: disable
1: enable

Bit 1 **T1AE**: TM1 Comparator A match interrupt control
0: disable
1: enable

Bit 0 **T1PE**: TM1 Comparator P match interrupt control
0: disable
1: enable

• **MFI2 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	XPF	SIM1F	—	—	XPE	SIM1E
R/W	—	—	R/W	R/W	—	—	R/W	R/W
POR	—	—	0	0	—	—	0	0

Bit 7~6 unimplemented, read as "0"

Bit 5 **XPF**: External peripheral interrupt request flag
0: No request
1: Interrupt request

Bit 4 **SIM1F**: SIM1 interrupt request flag
0: No request
1: Interrupt request

Bit 3~2 unimplemented, read as "0"

Bit 1 **XPE**: External Peripheral Interrupt Control
0: Disable
1: Enable

Bit 0 **SIM1E**: SIM1 Interrupt Control
0: Disable
1: Enable

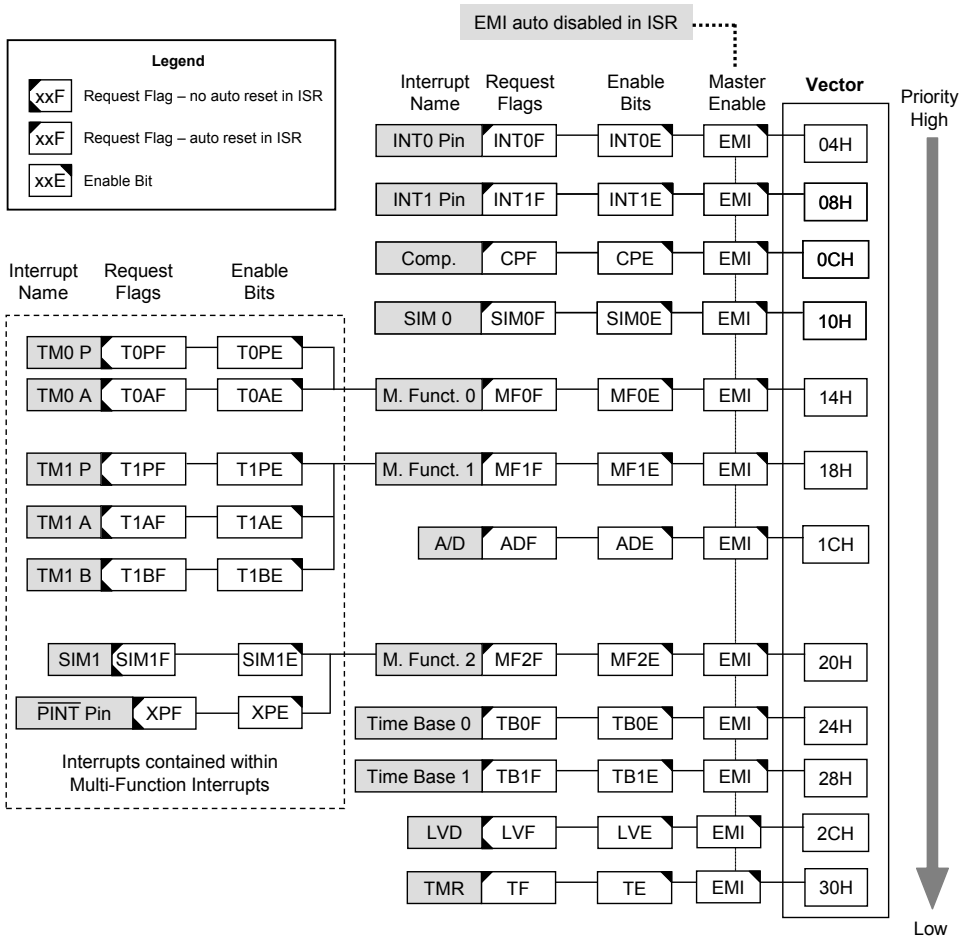
Interrupt Operation

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

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

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

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



Interrupt Structure

External Interrupt

The external interrupts are controlled by signal transitions on the pins INT0 and INT1. An external interrupt request will take place when the external interrupt request flags, INT0F, INT1F, are set, which will occur when a transition, whose type is chosen by the edge select bits, appears on the external interrupt pins. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and respective external interrupt enable bit, INT0E, INT1E, must first be set. Additionally the correct interrupt edge type must be selected using the INTEG register to enable the external interrupt function and to choose the trigger edge type. As the external interrupt pins are pin-shared with I/O pins, they can only be configured as external interrupt pins if their external interrupt enable bit in the corresponding interrupt register has been set. The pin must also be setup as an input by setting the corresponding bit in the port control register. 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.

Comparator Interrupt

The comparator interrupt is controlled by the one internal comparator. A comparator interrupt request will take place when the comparator interrupt request flag, CPF, is set, a situation that will occur when the comparator output changes state. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and comparator interrupt enable bit, CPE, must first be set. When the interrupt is enabled, the stack is not full and the comparator inputs generate a comparator output transition, a subroutine call to the comparator interrupt vector, will take place. When the interrupt is serviced, the external interrupt request flag, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts.

Multi-function Interrupt

Within these devices there are up to three Multi-function interrupts. Unlike the other independent interrupts, these interrupts have no independent source, but rather are formed from other existing interrupt sources, namely the TM Interrupts, SIM1 Interrupt, External Peripheral Interrupt and LVD interrupt.

A Multi-function interrupt request will take place when any of the Multi-function interrupt request flags, MF0F~MF2F are set. The Multi-function interrupt flags will be set when any of their included functions generate an interrupt request flag. To allow the program to branch to its respective interrupt vector address, when the Multi-function interrupt is enabled and the stack is not full, and either one of the interrupts contained within each of Multi-function interrupt occurs, a subroutine call to one of the Multi-function interrupt vectors will take place. When the interrupt is serviced, the related Multi-Function request flag, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts.

However, it must be noted that, although the Multi-function Interrupt flags will be automatically reset when the interrupt is serviced, the request flags from the original source of the Multi-function interrupts, namely the TM Interrupts, SIM1 Interrupt, External Peripheral Interrupt and LVD interrupt will not be automatically reset and must be manually reset by the application program.

A/D Converter Interrupt

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

Timer/Event Counter Interrupt

For a Timer/Event Counter interrupt to occur, the global interrupt enable bit, EMI, and the corresponding timer interrupt enable bit, TE, must first be set. An actual Timer/Event Counter interrupt will take place when the Timer/Event Counter request flag, TF, is set, a situation that will occur when the relevant Timer/Event Counter overflows. When the interrupt is enabled, the stack is not full and a Timer/Event Counter overflow occurs, a subroutine call to the relevant timer interrupt vector, will take place. When the interrupt is serviced, the timer interrupt request flag, TF, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts.

Time Base Interrupts

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

The purpose of the Time Base Interrupt is to provide an interrupt signal at fixed time periods. Their clock sources originate from the internal clock source f_{TB}. This f_{TB} input clock passes through a divider, the division ratio of which is selected by programming the appropriate bits in the TBC register to obtain longer interrupt periods whose value ranges. The clock source that generates f_{TB}, which in turn controls the Time Base interrupt period, can originate from several different sources, as shown in the System Operating Mode section.

- **TBC Register**

Bit	7	6	5	4	3	2	1	0
Name	TBON	TBCK	TB11	TB10	—	TB02	TB01	TB00
R/W	R/W	R/W	R/W	R/W	—	R/W	R/W	R/W
POR	0	0	1	1	0	1	1	1

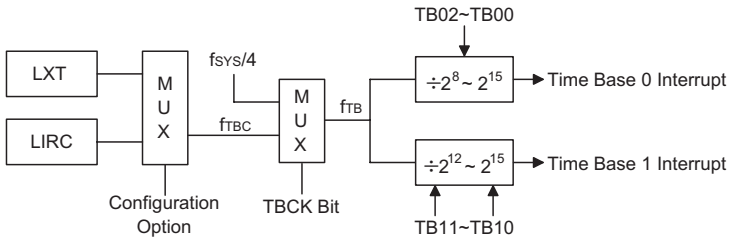
Bit 7 **TBON**: TB0 and TB1 Control
0: Disable
1: Enable

Bit 6 **TBCK**: Select f_{TB} Clock
0: f_{TBC}
1: $f_{SYS}/4$

Bit 5~4 **TB11~TB10**: Select Time Base 1 Time-out Period
00: $4096/f_{TB}$
01: $8192/f_{TB}$
10: $16384/f_{TB}$
11: $32768/f_{TB}$

Bit 3 Unimplemented, read as "0"

Bit 2~0 **TB02~TB00**: Select Time Base 0 Time-out Period
000: $256/f_{TB}$
001: $512/f_{TB}$
010: $1024/f_{TB}$
011: $2048/f_{TB}$
100: $4096/f_{TB}$
101: $8192/f_{TB}$
110: $16384/f_{TB}$
111: $32768/f_{TB}$



Time Base Interrupt

Serial Interface Module Interrupts

There are two Serial Interface Module Interrupts, known as the SIM0 and SIM1 interrupts. The SIM0 interrupt has its specific interrupt vector address and the SIM1 interrupt is contained within the Multi-function Interrupt. A SIMn Interrupt request will take place when the SIMn Interrupt request flag, SIMnF, is set, which occurs when a byte of data has been received or transmitted by the SIMn interface. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and the Serial Interface Interrupt enable bit, SIMnE, must first be set. Note that, for SIM1, the respective muti-function interrupt enable bits should also be set. When the interrupt is enabled, the stack is not full and a byte of data has been transmitted or received by the SIMn interface, a subroutine call to the respective Interrupt vector or Multi-function Interrupt vector, will take place. When the Serial Interface Interrupt is serviced, the EMI bit will be automatically cleared to disable other interrupts, however only the SIM0F flag and the Multi-function interrupt request flag will be also automatically cleared. As the SIM1F flag will not be automatically cleared, it has to be cleared by the application program.

External Peripheral Interrupt

The External Peripheral Interrupt operates in a similar way to the external interrupt and is contained within the Multi-function Interrupt. A Peripheral Interrupt request will take place when the External Peripheral Interrupt request flag, XPF, is set, which occurs when a negative edge transition appears on the PINT pin. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, external peripheral interrupt enable bit, XPE, and associated Multi-function interrupt enable bit, must first be set. When the interrupt is enabled, the stack is not full and a negative transition appears on the External Peripheral Interrupt pin, a subroutine call to the respective Multi-function Interrupt, will take place. When the External Peripheral Interrupt is serviced, the EMI bit will be automatically cleared to disable other interrupts, however only the Multi-function interrupt request flag will be also automatically cleared.

As the XPF flag will not be automatically cleared, it has to be cleared by the application program. The external peripheral interrupt pin is pin-shared with several other pins with different functions. It must therefore be properly configured to enable it to operate as an External Peripheral Interrupt pin.

LVD Interrupt

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

TM Interrupts

The Compact Type TM has two interrupts, while the Enhanced Type TM has three interrupts. All of the TM interrupts are contained within the Multi-function Interrupts. For the Compact Type TM there is two interrupt request flags TnPF and TnAF and two enable bits TnPE and TnAE. For the Enhanced Type TM there are three interrupt request flags TnPF, TnAF and TnBF and three enable bits TnPE, TnAE and TnBE. 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, A or B match situation happens.

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

Interrupt Wake-up Function

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

Programming Considerations

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

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

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

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

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

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

Power Down Mode and Wake-up

Entering the IDLE or SLEEP Mode

There is only one way for the device to enter the SLEEP or IDLE Mode and that is to execute the "HALT" instruction in the application program. When this instruction is executed, 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 WDT will be cleared and resume counting if the WDT clock source is selected to come from the f_{SUB} clock source and the WDT is enabled. The WDT will stop if its clock source originates from the system clock.
- The I/O ports will maintain their present condition.
- In the status register, the Power Down flag, PDF, will be set and the Watchdog time-out flag, TO, will be cleared.

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, 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 devices which have different package types, as there may be unbonbed 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 configuration options have enabled the LIRC oscillator.

Wake-up

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

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

If the system is woken up by an external 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.

Low Voltage Detector – LVD

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

LVD Register

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

- **LVDC Register**

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

Bit 7~6 unimplemented, read as "0"

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

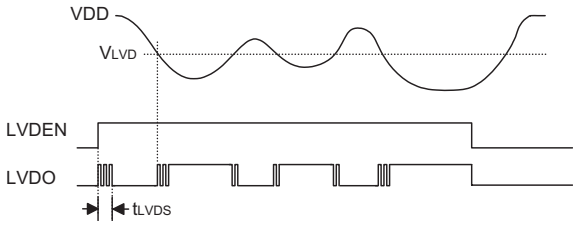
Bit **LVDEN**: Low Voltage Detector Control
0: Disable
1: Enable

Bit 3 unimplemented, read as "0"

Bit 2~0 **VLVD2 ~ VLVD0**: Select LVD Voltage
000: 2.0V
001: 2.2V
010: 2.4V
011: 2.7V
100: 3.0V
101: 3.3V
110: 3.6V
111: 4.4V

LVD Operation

The Low Voltage Detector function operates by comparing the power supply voltage, V_{DD}, with a pre-specified voltage level stored in the LVDC register. This has a range of between 2.0V and 4.4V. When the power supply voltage, V_{DD}, falls below this pre-determined value, the LVDO bit will be set high indicating a low power supply voltage condition. The Low Voltage Detector function is supplied by a reference voltage which will be automatically enabled. When the device is powered down the low voltage detector will remain active if the LVDEN bit is high. After enabling the Low Voltage Detector, a time delay t_{LVDS} should be allowed for the circuitry to stabilise before reading the LVDO bit. Note also that as the V_{DD} voltage may rise and fall rather slowly, at the voltage nears that of V_{LVD}, there may be multiple bit LVDO transitions.



LVD Operation

The Low Voltage Detector also has its own interrupt which is contained within one of the Multi-function interrupts, providing an alternative means of low voltage detection, in addition to polling the LVDO bit. The interrupt will only be generated after a delay of tLVD after the LVDO bit has been set high by a low voltage condition. When the device is powered down the Low Voltage Detector will remain active if the LVDEN bit is high. In this case, the LVF interrupt request flag will be set, causing an interrupt to be generated if V_{DD} falls below the preset LVD voltage. This will cause the device to wake-up from the SLEEP or IDLE Mode, however if the Low Voltage Detector wake up function is not required then the LVF flag should be first set high before the device enters the SLEEP or IDLE Mode.

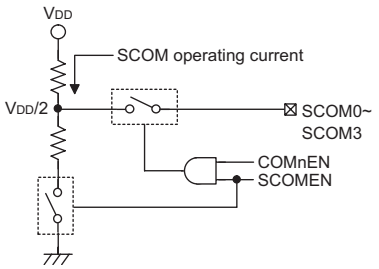
SCOM Function for LCD

The devices have the capability of driving external LCD panels. The common pins for LCD driving, SCOM0~ SCOM3, are pin shared with certain pin on the PC0 ~ PC1, PC6 ~ PC7 port. The LCD signals (COM and SEG) are generated using the application program.

LCD Operation

An external LCD panel can be driven using this device by configuring the PC0 ~ PC1, PC6 ~ PC7 pins as common pins and using other output ports lines as segment pins. The LCD driver function is controlled using the SCOMC register which in addition to controlling the overall on/off function also controls the bias voltage setup function. This enables the LCD COM driver to generate the necessary $V_{DD}/2$ voltage levels for LCD 1/2 bias operation.

The SCOMEN bit in the SCOMC register is the overall master control for the LCD driver, however this bit is used in conjunction with the COMnEN bits to select which Port C pins are used for LCD driving. Note that the Port Control register does not need to first setup the pins as outputs to enable the LCD driver operation.



LCD COM Bias

COMEN	COMnEN	Pin Function	O/P Level
0	X	I/O	0 or 1
1	0	I/O	0 or 1
1	1	SCOMn	$V_{DD}/2$

Output Control

LCD Bias Control

The LCD COM driver enables a range of selections to be provided to suit the requirement of the LCD panel which is being used. The bias resistor choice is implemented using the ISEL1 and ISEL0 bits in the SCOMC register.

Bit	7	6	5	4	3	2	1	0
Name	D7	ISEL1	ISEL0	SCOMEN	COM3EN	COM2EN	COM1EN	COM0EN
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 Reserved Bit
0: Correct level - bit must be reset to zero for correct operation
1: Unpredictable operation - bit must not be set high

Bit 6~5 **ISEL1, ISEL0**: ISEL1 ~ ISEL0: Select SCOM typical bias current (VDD=5V)
00: 25mA
01: 50mA
10: 100mA
11: 200mA

Bit 4 **SCOMEN**: SCOM module Control
0: Disable
1: Enable

Bit 3 **COM3EN**: PC3 or SCOM3 selection
0: GPIO
1: SCOM3

Bit 2 **COM2EN**: PC2 or SCOM2 selection
0: GPIO
1: SCOM2

Bit 1 **COM1EN**: PC1 or SCOM1 selection
0: GPIO
1: SCOM1

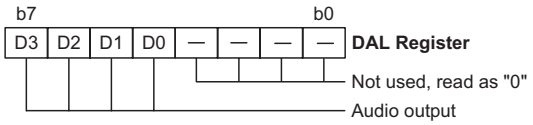
Bit 0 **COM0EN**: PC0 or SCOM0 selection
0: GPIO
1: SCOM0

Digitial to Analog Converter – DAC

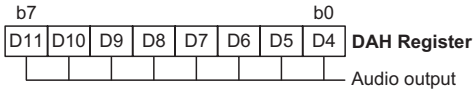
The voice control register controls the DAC circuit. If the DAC circuit is not enabled, any DAH/DAL outputs will be invalid. To write a "1" to the DACEN bit will enable the DAC circuit and output to corresponding I/O, while writing a "0" to the DACEN bit will disable the DAC circuit.

Audio Output and Volume Control – DAL, DAH, VOL, MISC

The audio output is 12-bits wide whose highest 8-bits are written into the DAH register and whose lowest four bits are written into the highest four bits of the DAL register. Bits 0~3 of the DAL register are always read as zero. Note that the 12-bit DAC data format should be unsigned. There are 8 levels of volume which are setup using the VOL register. The highest 3 bits of this register are used for volume control. The MISC register is used to control the DAC, Audio Power Amplifier and mute functions enable or not.



Digital to Analog Data Low Register



Digital to Analog Data High Register

- **VOL Register**

Bit	7	6	5	4	3	2	1	0
Name	VOL2	VOL1	VOL0	—	—	—	—	—
R/W	R/W	R/W	R/W	—	—	—	—	—
POR	0	0	0	—	—	—	—	—

Bit 7~5 **VOL2~VOL0**: DAC volume control data

Bit 4~0 unimplemented, read as "0"

The programmer can change the DAC volume by only writing the volume control data to the Bits 7~5 of the VOL, and Bits 4~0 of the VOL register are always read as zero. Note that the 12-bit DAC data format should be unsigned.

VOL [2:0]	DAC Volume Control
111	High Volume
110	↑
101	
100	
011	
010	
001	↓
000	Low Volume

• **MISC Register**

Bit	7	6	5	4	3	2	1	0
Name	LXTLP	PAEM	PAEN	—	—	—	DACEN	—
R/W	R/W	R/W	R/W	—	—	—	R/W	—
POR	0	0	0	—	—	—	0	—

Bit 7 **LXTLP:** Low Speed Crystal Oscillator power control, described elsewhere
0: disable
1: enable

Bit 6 **PAEM:** Audio mute function control
0: Mute disable
1: Mute enable

Bit 5 **PAEN:** Audio Power Amplifier function control
0: disable
1: enable

Bit 4~2 unimplemented

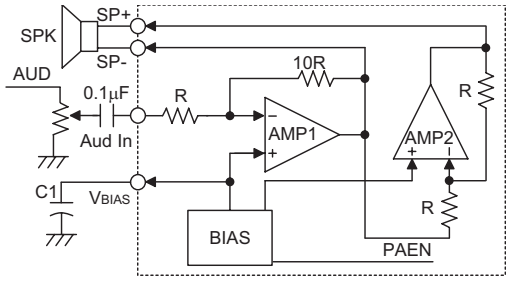
Bit 1 **DACEN:** DAC control
0: disable
1: enable

Bit 0 unimplemented

Audio Power Amplifier

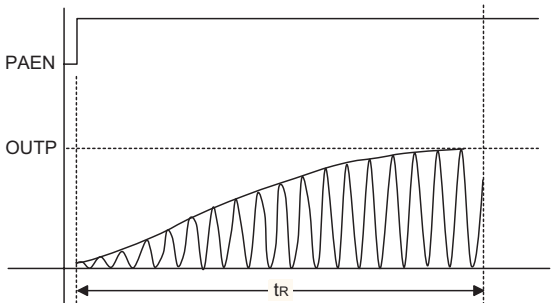
Power Amplifier

This device contains an audio power amplifier which is an integrated class AB monophonic type speaker driver. It has the properties of high S/N ratio, high slew rate, low distortion, large output oltage swing, excellent power supply ripple rejection, low power consumption, low standby current and power off control etc.



Note: Aud In: Audio input
V_{BIAS}: Speaker non-inverting input voltage reference
SP+:Audio Positive output
SP–: Audio Negative output
OUTP Rising Time (t_R)

When the PAEN bit DACC register enables the Power Amplifer, note that it requires a certain time before it can output fully on the OUTP pin. However, this delay time depends on the value of C1. The C1 capacitor is connected between VBIAS and VSS.

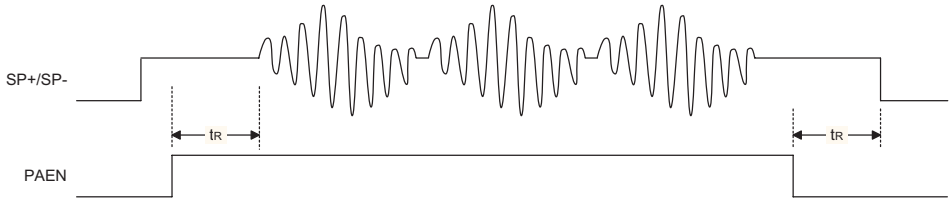


Capacitor / Voltage / t_R	0.1μF	1μF	4.7μF	10μF
2.2V	15ms	30ms	90ms	185ms
3V	15ms	30ms	90ms	185ms
4	15ms	30ms	90ms	185ms

For battery based applications, power consumption is a key issue, therefore the amplifier should be turned off when in the standby state. In order to eliminate any speaker sound bursts while turning the amplifier on, the application circuit, which will incorporate a capacitance value of C1, should be adjusted in accordance with the speaker s audio frequency response. A greater value of C1 will improve the noise burst while turning on the amplifier. The recommended operation sequence is:

- Turn On: audio signal standby (1/2VDD) → enable amplifier → wait t_R for amplifier ready → audio output
- Turn Off: audio signal finished → disable amplifier → wait t_R for amplifier off → audio signal off

If the application is not powered by batteries and there is no problem with amplifier On/Off issues, a capacitor value of 0.1μF for C1 is recommended.

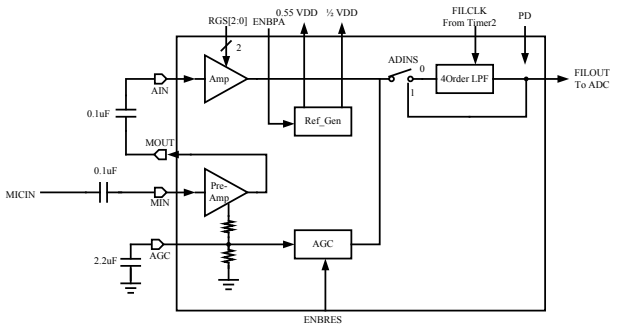


Microphone Amplifier

The device contains a microphone amplifier to enable easy interfacing to external microphones for recorder applications.

Amplifier Overview

The overall amplifier circuit contains several main features, pre-amp, AGC (Automatic Gain Control), Power Amplifier Gain stage and a fourth-order Low Pass SCF (Switched Capacitor Filter). The following block diagram illustrates the main blocks of the microphone input control.



The MIN pin is the amplifier microphone input pin to which external microcphones can be AC-coupled using a series capacitor. The Microphone signal is amplified and output on the MOUT pin. The voltage gain of the Pre-Amp is determined by the voltage level on the AGC pin. An external capacitor ac couples the MOUT output to the AIN pin.

Automatic Gain Control Register – AGCC

The AGCC register controls the overall function of the AGC circuit such as enable/disable, gain control etc.

- **AGCC register**

Bit	7	6	5	4	3	2	1	0
Label	RON	AGCEN	ADINS	—	—	RGS2	RGS1	RGS0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit 7 **RON:** AGC Charge Resistance control
0: off
1: on

Bit 6 **AGCEN**: AGC control
0: disable
1: enable
Note that if the AGC function is enabled, then pin PA0 will become an input pin and will disable any interal pull high options automatically.The AGC function can be disabled to reduce power consumption.

Bit 5 **ADINS**: ADC input control
0: NOT filtered via the SCF
1: filtered via the SCF

Bit 4-3 unimplemented, read as "0".

Bit 2-0 **RGS2~ RGS0:** Amplifier Gain control
000: 1.16
001: 1.56
010: 2.22
011: 2.91
100: 5.09
101: 8.91
110: 13.82
111: 25.97

Automatic Gain Control – AGC

The purpose of the Automatic Gain Control (AGC) is to dynamically adjust the pre-amp gain, and therefore extend the dynamic range of input signals which can be applied the microphone input without distortion. The AGC considerably extends the range of recordable sounds from whispers to loud input voices.

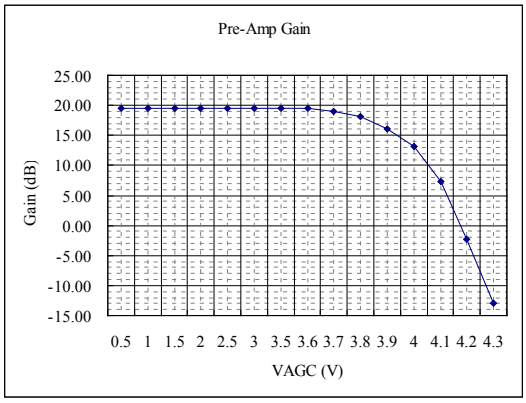
If the input signal to the AGC increases sharply, the AGC output response will be to create an increasing voltage on its output. This increasing output voltage will have the effect of reducing the pre-amp gain thus compensating for the higher input signal.

Similarly, if the input level decreases sharply, the AGC output response will be to create a decreasing voltage on its output. This decreasing output voltage will have the effect of increasing the pre-amp gain thus compensating for the lower input signal.

Peak voltage levels at the Amp output are detected by the AGC circuitry, which charge the external capacitor on the AGC control pin. The internal source resistance of the AGC circuit and external capacitor value determine the attack time of the gain control. The attack time is defined as the time needed for the AGC to respond to a sudden increase of the input signal until the output signal is stable.The Release time is determined by the RC time constant of the external (or internal using register bit RON) resistor and capacitor on the AGC pin. The release time is defined as the time needed to respond to as a sudden decrease of the input signal until the output signal is stable.

The internal two-stage amplifier is used to amplify the microphone input audio signal. This amplified audio signal will be output both to the AOUT pin and to the internal ADC input. The ADINS bit in the AGCC register is used to control the ADC input signal path, to determine if it is to be filtered by the SCF filter or not. In this way, the audio signal can be converted into digital data for recording into the external Flash memory using the SPI interface.

The gain of Pre-Amp amplifier is controlled by the voltage level of AGC pin. The following figure illustrates the relationship between Pre-Amp gain and the voltage level on the VAGC pin.



Two-stage Amplifier

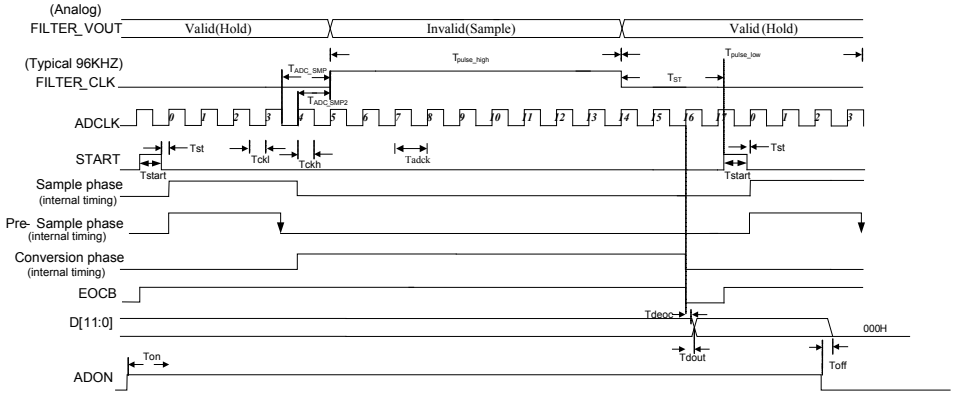
The amplification of the AGC loop is implemented using a two-stage amplifiers, one of which is a pre-amp and Amp. The AGC dynamically monitors the output of the second stage amplifier signal level and sets the gain control voltage of the Pre-Amp accordingly. The Pre-Amp gain will be automatically adjusted to maintain an optimum signal level to the next stage of this AGC loop. The gain of the second amplifer stage is controlled by the RGS2~RGS0 bits in the AGCC register. There are eight amplifier gain levels. Once a specific gain is selected, the AGC will reference itself to this gain level and adjust the Pre-Amp amplifier gain accordingly. Using this method applications can select the most suitable gain levels according to their product requiremets.

Switched Capacitor Filter – SCF

The SCF is used to filter out unwanted high frequency signal. This filter is a fourth order low pass SCF type whose characteristics are selected according to the FICLK frequency. The FILCLK frequency is generated by the Timer/Event Counter. The following table shows the 3-dB breakpoint of the SCF filter with respect to the FILCLK.

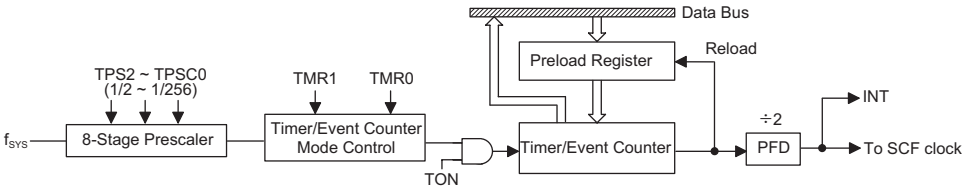
Input Sample Rate (kHz)	3-dB frequency (kHz)	FILCLK (kHz)
6.0	2.6	75
8.0	3.4	96
10.0	4.3	120
12.0	5.2	146
16.0	6.9	195

The SCF output can be sampled by the fast 12-bit A/D converter. In order to prevent unwanted noise, the ADC will start to convert the input signal after skipping a few data samples. The relationship between FILCLK, ADCLK and the start conversion signal is shown in the following diagram.



SCF Clock

The SCF clock source is generated by the Timer/Event Counter. The Timer/Event Counter operation mode should be selected to be the timer mode, therefore the TMR1 and TMR0 bits should be set to 10. The Timer/Event Counter can be used to generate a variable frequency clock according to the desired FILCLK clock source for the SCF filter.A 3-bit prescaler, setup using bits TPS0 ~ TPS2 define the different division ratios of the Timer/Event Counter clock source. Refer to the Timer/Event Counter section of the datasheet for more setup details. Note that the Timer/Event Counter output has its own interrupt source. If the Timer/Event Counter is selected as the SCF clock source, then the interrupt should be disabled to maintain an accurate SCF clock source.



TMR Structure

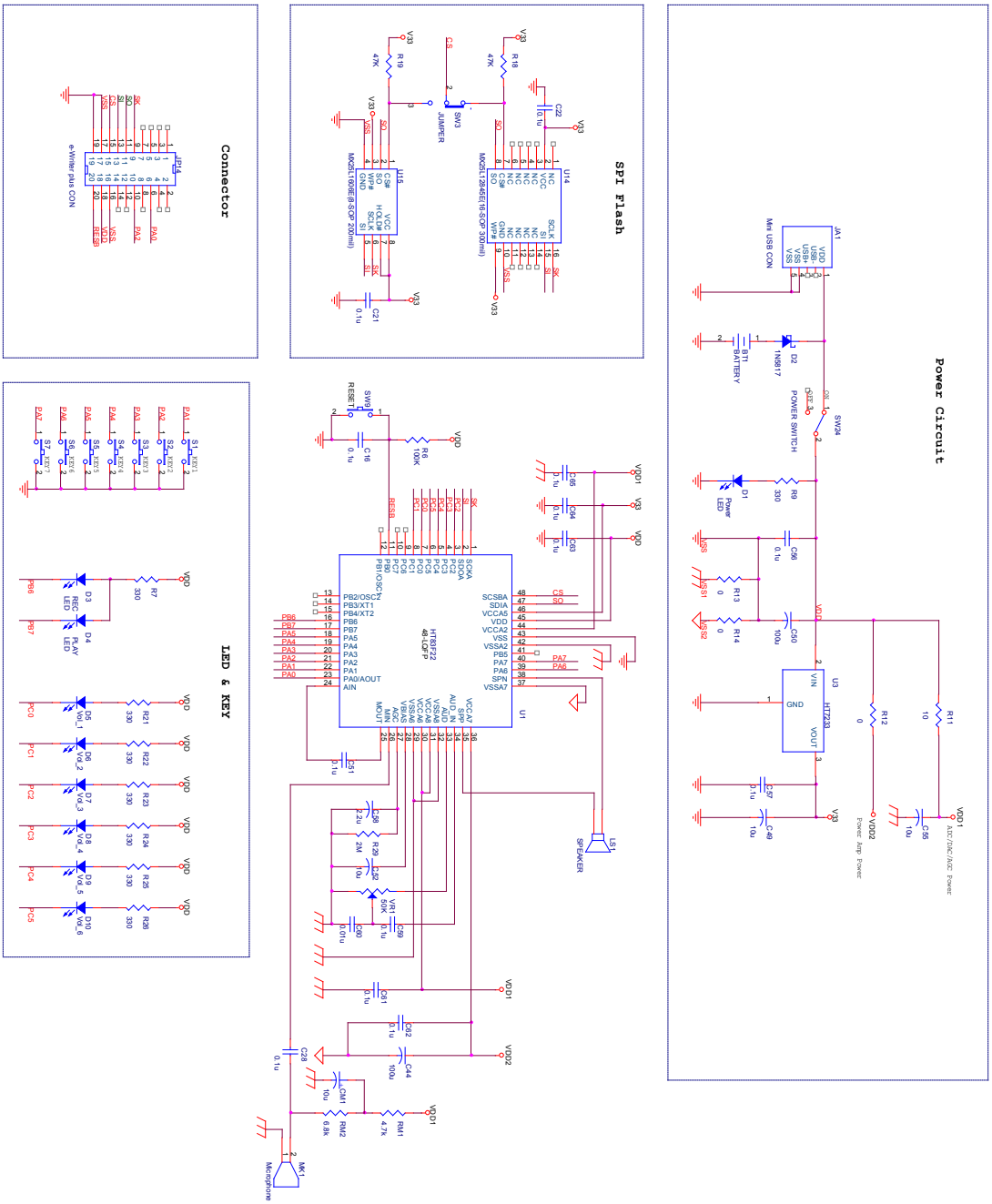
Configuration Options

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. As these options are programmed into the device using the hardware programming tools, once they are selected they cannot be changed later using the application program. All options must be defined for proper system function, the details of which are shown in the table.

No.	Options
Oscillator Options	
1	High Speed System Oscillator Selection - $f_{H:}$ 1. HXT 2. ERC 3. HIRC
2	Low Speed System Oscillator Selection - $f_{L:}$ 1. LXT 2. LIRC
3	f_{SUB} clock source 1. LXT 2. LIRC
4	WDT Clock Selection - $f_{S:}$ 1. f_{SUB} 2. $f_{SYS}/4$
5	HIRC Frequency Selection: 1. 4MHz 2. 8MHz 3. 12MHz
6	f_{TBC} clock source 1. LXT 2. LIRC
Reset Pin Options	
7	PB0/RES Pin Options: 1. RES pin 2. I/O pin
Watchdog Options	
8	Watchdog Timer Function: 1. Enable 2. Disable
9	CLRWDT Instructions Selection: 1. 1 instructions 2. 2 instructions
LVR Options	
10	LVR Function: 1. Enable 2. Disable

No.	Options
11	LVR Voltage Selection: 1. 2.10V 2. 2.55V 3. 3.15V 4. 4.20V
SIM Options	
12	SPI0 - WCOL bit: 1. Enable 2. Disable
13	SPI0 - CSEN bit: 1. Enable 2. Disable
14	I^2C0 - RNIC bit: 1. Enable 2. Disable
15	I^2C0 Debounce Time Selection: 1. No debounce 2. 1 system clock debounce 3. 2 system clock debounce
16	SIM1 Function: 1. Enable 2. Disable
17	SPI1 - WCOL bit: 1. Enable 2. Disable
18	SPI1 - CSEN bit: 1. Enable 2. Disable
19	I^2C1 - RNIC bit: 1. Enable 2. Disable
20	I^2C1 Debounce Time Selection: 1. No debounce 2. 1 system clock debounce 3. 2 system clock debounce

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 microcontrollers, a comprehensive and flexible set of over 60 instructions is provided to enable programmers to implement their application with the minimum of programming overheads.

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

Instruction Timing

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

Moving and Transferring Data

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

Arithmetic Operations

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

Logical and Rotate Operations

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 where 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 setup as a table where data can be directly stored. A set of easy to use instructions provides the means by which this fixed data can be referenced and retrieved from the Program Memory.

Other Operations

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

Instruction Set Summary

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

Table Conventions

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

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

Mnemonic	Description	Cycles	Flag Affected
Data Move			
MOV A,[m]	Move Data Memory to ACC	1	None
MOV [m],A	Move ACC to Data Memory	1Note	None
MOV A,x	Move immediate data to ACC	1	None
Bit Operation			
CLR [m].i	Clear bit of Data Memory	1Note	None
SET [m].i	Set bit of Data Memory	1Note	None
Branch			
JMP addr	Jump unconditionally	2	None
SZ [m]	Skip if Data Memory is zero	1Note	None
SZA [m]	Skip if Data Memory is zero with data movement to ACC	1Note	None
SZ [m].i	Skip if bit i of Data Memory is zero	1Note	None
SNZ [m].i	Skip if bit i of Data Memory is not zero	1Note	None
SIZ [m]	Skip if increment Data Memory is zero	1Note	None
SDZ [m]	Skip if decrement Data Memory is zero	1Note	None
SIZA [m]	Skip if increment Data Memory is zero with result in ACC	1Note	None
SDZA [m]	Skip if decrement Data Memory is zero with result in ACC	1Note	None
CALL addr	Subroutine call	2	None
RET	Return from subroutine	2	None
RET A,x	Return from subroutine and load immediate data to ACC	2	None
RETI	Return from interrupt	2	None
Table Read			
TABRDC [m]	Read table (current page) to TBLH and Data Memory	2Note	None
TABRDL [m]	Read table (last page) to TBLH and Data Memory	2Note	None
Miscellaneous			
NOP	No operation	1	None
CLR [m]	Clear Data Memory	1Note	None
SET [m]	Set Data Memory	1Note	None
CLR WDT	Clear Watchdog Timer	1	TO, PDF
CLR WDT1	Pre-clear Watchdog Timer	1	TO, PDF
CLR WDT2	Pre-clear Watchdog Timer	1	TO, PDF
SWAP [m]	Swap nibbles of Data Memory	1Note	None
SWAPA [m]	Swap nibbles of Data Memory with result in ACC	1	None
HALT	Enter power down mode	1	TO, PDF

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

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

3. For the "CLR WDT1" and "CLR WDT2" instructions the TO and PDF flags may be affected by the execution status. The TO and PDF flags are cleared after both "CLR WDT1" and "CLR WDT2" instructions are consecutively executed. Otherwise the TO and PDF flags remain unchanged.

Instruction Definition

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

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

Operation: ACC ← ACC + [m] + C

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

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

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

Operation: [m] ← ACC + [m] + C

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

ADD A,[m] Add Data Memory to ACC

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

Operation: ACC ← ACC + [m]

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

ADD A,x Add immediate data to ACC

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

Operation: AC ← ACC + x

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

ADDM A,[m] Add ACC to Data Memory

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

Operation: [m] ← ACC + [m]

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

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

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

Operation: ACC ← ACC "AND" [m]

Affected flag(s): Z

AND A,x Logical AND immediate data to ACC

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

Operation: ACC ← ACC "AND"x

Affected flag(s): Z

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

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

Operation: [m] ← ACC "AND" [m]

Affected flag(s): Z

CALL addr	Subroutine call
Description	Unconditionally calls a subroutine at the specified address. The Program Counter then increments by 1 to obtain the address of the next instruction which is then pushed onto the stack. The specified address is then loaded and the program continues execution from this new address. As this instruction requires an additional operation, it is a two cycle instruction.
Operation	Stack ← Program Counter + 1 Program Counter ← addr
Affected flag(s)	None
CLR [m]	Clear Data Memory
Description	Each bit of the specified Data Memory is cleared to 0.
Operation	[m] ← 00H
Affected flag(s)	None
CLR [m].i	Clear bit of Data Memory
Description	Bit i of the specified Data Memory is cleared to 0.
Operation	[m].i ← 0
Affected flag(s)	None
CLR WDT	Clear Watchdog Timer
Description	The TO, PDF flags and the WDT are all cleared.
Operation	WDT cleared TO ← 0 PDF ← 0
Affected flag(s)	TO, PDF
CLR WDT1	Pre-clear Watchdog Timer
Description	The TO, PDF flags and the WDTare all cleared. Note that this instruction works in conjunction with CLR WDT2 and must be executed alternately with CLR WDT2 to have effect. Repetitively executing this instruction without alternately executing CLR WDT2 will have no effect.
Operation	WDT cleared TO ← 0 PDF ← 0
Affected flag(s)	TO, PDF
CLR WDT2	Pre-clear Watchdog Timer
Description	The TO, PDF flags and the WDTare all cleared. Note that this instruction works in conjunction with CLR WDT1 and must be executed alternately with CLR WDT1 to have effect. Repetitively executing this instruction without alternately executing
Operation	WDT cleared TO ← 0 PDF ← 0
Affected flag(s)	TO, PDF

CPL [m] Complement Data Memory

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

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

Affected flag(s) Z

CPLA [m] Complement Data Memory with result in ACC

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

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

Affected flag(s) Z

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

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

Operation $[m] \leftarrow ACC + 00H$ or

$[m] \leftarrow ACC + 06H$ or

$[m] \leftarrow ACC + 60H$ or

$[m] \leftarrow ACC + 66H$

Affected flag(s) C

DEC [m] Decrement Data Memory

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

Operation $[m] \leftarrow [m] - 1$

Affected flag(s) Z

DECA [m] Decrement Data Memory with result in ACC

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

Operation $ACC \leftarrow [m] - 1$

Affected flag(s) Z

HALT	Enter power down mode
Description	This instruction stops the program execution and turns off the system clock. The contents of the Data Memory and registers are retained. The WDT and prescaler are cleared. The power down flag PDF is set and the WDT time-out flag TO is cleared.
Operation	TO ← 0 PDF ← 0
Affected flag(s)	TO, PDF

INC [m]	Increment Data Memory
Description	Data in the specified Data Memory is incremented by 1.
Operation	[m] ← [m]+1
Affected flag(s)	Z

INCA [m]	Increment Data Memory with result in ACC
Description	Data in the specified Data Memory is incremented by 1. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.
Operation	ACC ← [m]+1
Affected flag(s)	Z

JMP addr	Jump unconditionally
Description	The contents of the Program Counter are replaced with the specified address. Program execution then continues from this new address. As this requires the insertion of a dummy instruction while the new address is loaded, it is a two cycle instruction.
Operation	Program Counter ← addr
Affected flag(s)	None

MOV A,[m]	Move Data Memory to ACC
Description	The contents of the specified Data Memory are copied to the Accumulator.
Operation	ACC ← [m]
Affected flag(s)	None

MOV A,x	Move immediate data to ACC
Description	The immediate data specified is loaded into the Accumulator.
Operation	ACC ← x
Affected flag(s)	None

MOV [m],A	Move ACC to Data Memory
Description	The contents of the Accumulator are copied to the specified Data Memory.
Operation	[m] ← ACC
Affected flag(s)	None

NOP	No operation
Description	No operation is performed. Execution continues with the next instruction.
Operation	No operation
Affected flag(s)	None

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

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

Operation: ACC ← ACC "OR" [m]

Affected flag(s): Z

OR A,x Logical OR immediate data to ACC

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

Operation: ACC ← ACC "OR" x

Affected flag(s): Z

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

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

Operation: [m] ← ACC "OR" [m]

Affected flag(s): Z

RET Return from subroutine

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

Operation: Program Counter ← Stack

Affected flag(s): None

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

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

Operation: Program Counter ← Stack
ACC ← x

Affected flag(s): None

RETI Return from interrupt

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

Operation: Program Counter ← Stack
EMI ← 1

Affected flag(s): None

RL [m] Rotate Data Memory left

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

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

Affected flag(s): None

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

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

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

Affected flag(s): None

RLC [m] Rotate Data Memory left through Carry

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

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

Affected flag(s): C

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

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

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

Affected flag(s): C

RR [m] Rotate Data Memory right

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

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

Affected flag(s): None

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

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

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

Affected flag(s): None

RRC [m] Rotate Data Memory right through Carry

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

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

Affected flag(s): C

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

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

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

Affected flag(s): C

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

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

Operation: ACC ← ACC – [m] – $\overline{C}$

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

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

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

Operation: ACC ← ACC – [m] – $\overline{C}$

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

SDZ [m] Skip if decrement Data Memory is 0

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

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

Affected flag(s): None

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

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

Operation:
ACC ← [m] – 1
Skip if ACC = 0

Affected flag(s): None

SET [m]	Set Data Memory
Description	Each bit of the specified Data Memory is set to 1.
Operation	[m] ← FFH
Affected flag(s)	None

SET [m].i	Set bit of Data Memory
Description	Bit i of the specified Data Memory is set to 1.
Operation	[m].1 ← 1
Affected flag(s)	None

SIZ [m]	Skip if increment Data Memory is 0
Description	The contents of the specified Data Memory are first incremented by 1. If the result is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.
Operation	[m] ← [m] + 1 Skip if [m] = 0
Affected flag(s)	None

SIZA [m]	Skip if increment Data Memory is zero with result in ACC
Description	The contents of the specified Data Memory are first incremented by 1. If the result is 0, the following instruction is skipped. The result is stored in the Accumulator but the specified Data Memory contents remain unchanged. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.
Operation	ACC ← [m] + 1 Skip if ACC = 0
Affected flag(s)	None

SNZ [m].i	Skip if bit i of Data Memory is not 0
Description	If bit i of the specified Data Memory is not 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is 0 the program proceeds with the following instruction.
Operation	Skip if [m].i ≠ 0
Affected flag(s)	None

SUB A,[m]	Subtract Data Memory from ACC
Description	The specified Data Memory is subtracted from the contents of the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.
Operation	ACC ← ACC – [m]
Affected flag(s)	OV, Z, AC, C

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

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

Operation [m] ← ACC – [m]

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

SUB A,x Subtract immediate data from ACC

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

Operation ACC ← ACC – x

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

SWAP [m] Swap nibbles of Data Memory

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

Operation [m].3~[m].0↔[m].7 ~ [m].4

Affected flag(s) None

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

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

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

Affected flag(s) None

SZ [m] Skip if Data Memory is 0

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

Operation Skip if [m] = 0

Affected flag(s) None

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

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

Operation ACC ← [m]
Skip if [m] = 0

Affected flag(s) None

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

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

Operation: Skip if [m].i = 0

Affected flag(s): None

TABRDC [m] Read table (current page) to TBLH and Data Memory

Description: The low byte of the program code (current 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

TABRDL [m] Read table (last page) to TBLH and Data Memory

Description: The low byte of the program code (last page) addressed by the table pointer (TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.

Operation: [m] ← program code (low byte)
TBLH ← program code (high byte)

Affected flag(s): None

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

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

Operation: ACC ← ACC "XOR" [m]

Affected flag(s): Z

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

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

Operation: [m] ← ACC "XOR" [m]

Affected flag(s): Z

XOR A,x Logical XOR immediate data to ACC

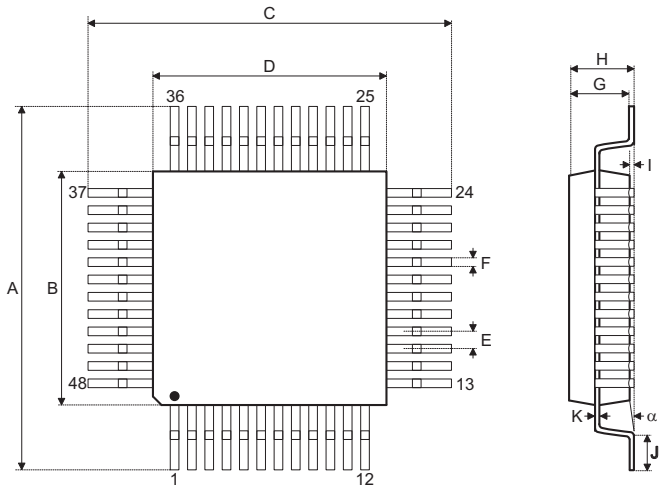
Description: Data in the Accumulator and the specified immediate data perform a bitwise logical XOR operation. The result is stored in the Accumulator.

Operation: ACC ← ACC "XOR"x

Affected flag(s): Z

Package Information

48-pin LQFP (7mmx7mm) Outline Dimensions



Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	0.350	—	0.358
B	0.272	—	0.280
C	0.350	—	0.358
D	0.272	—	0.280
E	—	0.020	—
F	—	0.008	—
G	0.053	—	0.057
H	—	—	0.063
I	—	0.004	—
J	0.018	—	0.030
K	0.004	—	0.008
α	0°	—	7°

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	8.90	—	9.10
B	6.90	—	7.10
C	8.90	—	9.10
D	6.90	—	7.10
E	—	0.50	—
F	—	0.20	—
G	1.35	—	1.45
H	—	—	1.60
I	—	0.10	—
J	0.45	—	0.75
K	0.10	—	0.20
α	0°	—	7°

Holtek Semiconductor Inc. (Headquarters)
No.3, Creation Rd. II, Science Park, Hsinchu, Taiwan
Tel: 886-3-563-1999
Fax: 886-3-563-1189
http://www.holtek.com.tw

Holtek Semiconductor Inc. (Taipei Sales Office)
4F-2, No. 3-2, YuanQu St., Nankang Software Park, Taipei 115, Taiwan
Tel: 886-2-2655-7070
Fax: 886-2-2655-7373
Fax: 886-2-2655-7383 (International sales hotline)

Holtek Semiconductor Inc. (Shenzhen Sales Office)
5F, Unit A, Productivity Building, No.5 Gaoxin M 2nd Road, Nanshan District, Shenzhen, China 518057
Tel: 86-755-8616-9908, 86-755-8616-9308
Fax: 86-755-8616-9722

Holtek Semiconductor (USA), Inc. (North America Sales Office)
46729 Fremont Blvd., Fremont, CA 94538, USA
Tel: 1-510-252-9880
Fax: 1-510-252-9885
http://www.holtek.com