A/D Flash MCU with EEPROM

HT66F0042/HT66F0082

Revision: V1.30 Date: December 05, 2016

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} = 16MHz: 3.3V~5.5V
- Up to 0.25μs instruction cycle with 16MHz system clock at V_{DD}=5V
- Power down and wake-up functions to reduce power consumption
- Three Oscillators
  - Internal RC – HIRC
  - Internal 32kHz RC – LIRC
  - External 32.768kHz Crystal – LXT
- Fully intergrated internal 8/12/16MHz oscillator requires no external components
- Multi-mode operation: NORMAL, SLOW, IDLE and SLEEP
- All instructions executed in one or two instruction cycles
- Table read instructions
- 63 powerful instructions
- Up to 6-level subroutine nesting
- Bit manipulation instruction

Peripheral Features

- Flash Program Memory: 2K×15/4K ×16
- RAM Data Memory: 96×8/128×8
- True EEPROM Memory: 32×8/64×8
- Watchdog Timer function
- Up to 26 bidirectional I/O lines
- Software controlled 4-SCOM lines LCD driver with 1/2 bias
- Two pin-shared external interrupts
- Multiple Timer Modules for time measure, compare match output, capture input, PWM output, single pulse output functions
- Dual Time-Base functions for generation of fixed time interrupt signals
- Multi-channel 12-bit resolution A/D converter
- Serial Interface Module - SPI or I^2C
- Full-duplex Universal Asynchronous Receiver and Transmitter Interface – UART (HT66F0082 only)
- Programmable I/O port source current for LED driving applications
- Low voltage reset function
- Flash program memory can be re-programmed up to 100,000 times
- Flash program memory data retention > 10 years
- True EEPROM data memory can be re-programmed up to 1,000,000 times
- True EEPROM data memory data retention > 10 years
- Wide range of available package types

General Description

These devices are Flash Memory type 8-bit high performance RISC architecture microcontrollers. Offering users the convenience of Flash Memory multi-programming features, these devices also include a wide range of functions and features. Other memory includes an area of RAM Data Memory as well as an area of true EEPROM memory for storage of non-volatile data such as serial numbers, calibration data etc.

Analog features include a multi-channel 12-bit A/D converter function. Multiple and extremely flexible Timer Modules provide timing, pulse generation, capture input, compare match output, single pulse output and PWM generation functions. Communication with the outside world is managed by including fully integrated SPI and 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 and Low Voltage Reset coupled with excellent noise immunity and ESD protection ensure that reliable operation is maintained in hostile electrical environments.

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

The SPI, I^2C and UART interfaces offer possibilities for data communication networks between microcontrollers, low-cost data links between PCs and peripheral devices, portable and battery operated device communication, etc.

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 electronic metering, environmental monitoring, handheld instruments, household appliances, electronically controlled tools, motor driving in addition to many others.

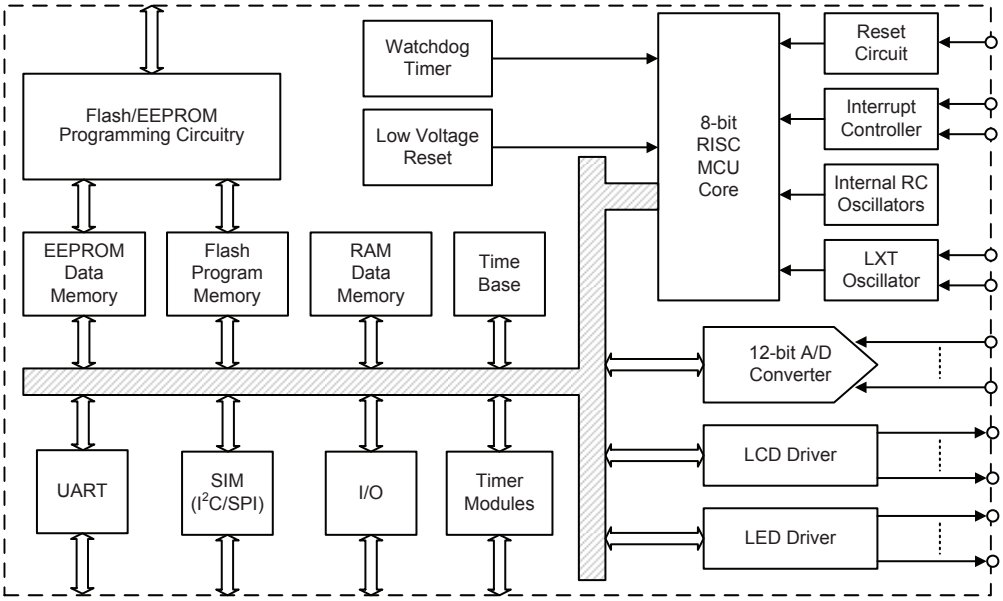
Selection Table

Most features are common to all devices, the main feature distinguishing them are memory capacity, I/O count and package types. The following table summarises the main features of each device.

Part No.	Program Memory	Data Memory	Data EEPROM	I/O	Ext. Int.	A/D Converter
HT66F0042	2K×15	96×8	32×8	22	2	12-bit×8
HT66F0082	4K×16	128×8	64×8	26	2	12-bit×8

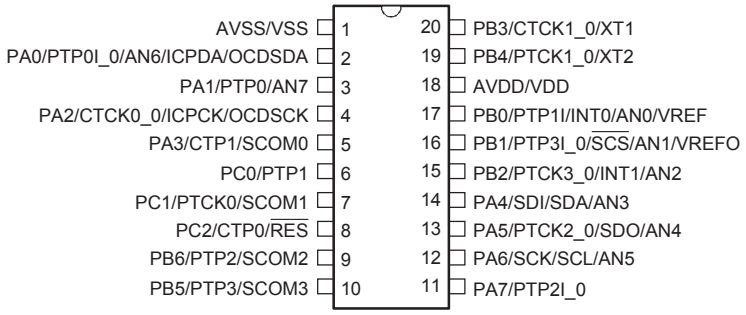
Part No.	Timer Module	Time Base	SIM	UART	Stack	R-Type LCD	Package
HT66F0042	10-bit PTM×4 10-bit CTM×2	2	√	×	6	4SCOM	20SOP/SSOP 24SOP/SSOP
HT66F0082	10-bit PTM×4 10-bit CTM×2	2	√	√	6	4SCOM	24SOP/SSOP 28SOP/SSOP

Block Diagram

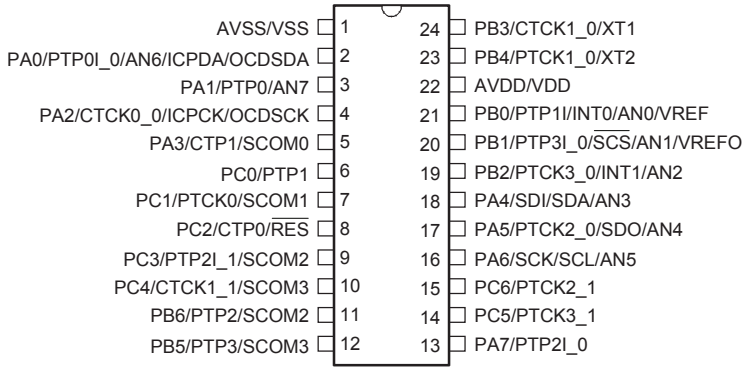


Note: The UART function only exists in the HT66F0082.

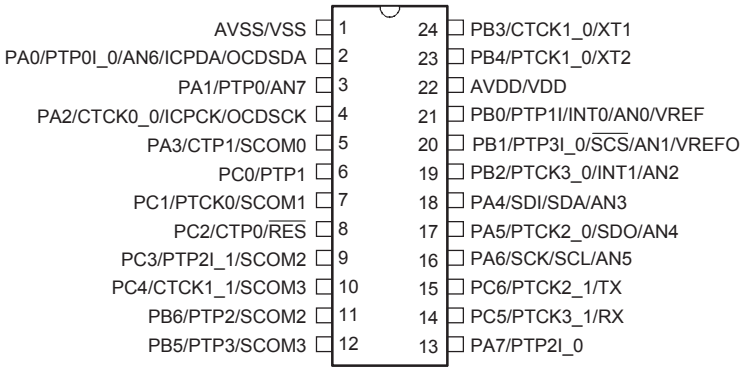
Pin Assignment



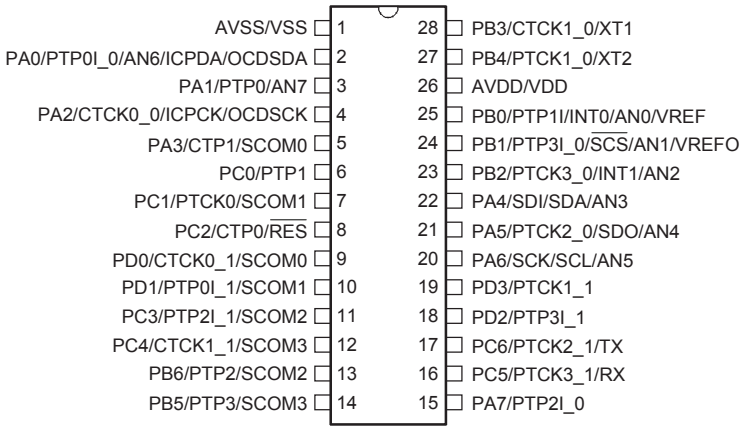
HT66F0042/HT66V0042
20 SOP-A/SSOP-A



HT66F0042/HT66V0042
24 SOP/SSOP-A



HT66F0082/HT66V0082
24 SOP/SSOP-A



HT66F0082/HT66V0082
28 SOP/SSOP-A

Note: 1. If the pin-shared pin functions have multiple outputs simultaneously, the desired pin-shared function is determined by the corresponding software control bits except the functions determined by the configuration options.

2. AVDD&VDD means the VDD and AVDD are the double bonding. VSS&AVSS means the VSS and AVSS are the double bonding.

3. The OCDSDA and OCDSCK pins are the OCDS dedicated pins and only available for the HT66V0042 and HT66V0082 devices, which are the OCDS EV chips for the HT66F0042 and HT66F0082 devices respectively.

Pin Description

With the exception of the power pins and some relevant transformer control pins, all pins on these devices can be referenced by their Port name, e.g. PA0, PA1 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, Timer Module 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.

HT66F0042

Pin Name	Function	OPT	I/T	O/T	Description
PA0/PTP0I_0/AN6/ ICPDA/OCDSDA	PA0	PAWU PAPU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	PTP0I_0	PAS0 PTM0C0	ST	—	PTM0 input
	AN6	PAS0	AN	—	A/D Converter input channel 6
	ICPDA	—	ST	CMOS	ICP Data Line
	OCDSDA	—	ST	CMOS	On Chip Debug System Data Line (OCDS EV only)
PA1/PTP0/AN7	PA1	PAWU PAPU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	PTP0	PAS0	—	CMOS	PTM0 output
	AN7	PAS0	AN	—	A/D Converter input channel 7
PA2/CTCK0_0/ICPCK/ OCDSCK	PA2	PAWU PAPU	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	CTCK0_0	CTM0C0	ST	—	CTM0 clock input
	ICPCK	—	ST	—	ICP Clock Line
	OCDSCK	—	ST	—	On Chip Debug System Clock Line (OCDS EV only)
PA3/CTP1/SCOM0	PA3	PAWU PAPU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	CTP1	PAS0	—	CMOS	CTM1output
	SCOM0	PAS0	—	AN	LCD driver output for LCD panel common
PA4/SDI/SDA/AN3	PA4	PAWU PAPU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	SDI	PAS1 SIMC0	ST	—	SPI serial data input
	SDA	PAS1 SIMC0	ST	CMOS	I^2C data line
	AN3	PAS1	AN	—	A/D Converter input channel 3
PA5/PTCK2_0/SDO/ AN4	PA5	PAWU PAPU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	PTCK2_0	PAS1 PTM2C0 IFS	ST	—	PTM2 clock input
	SDO	PAS1	—	CMOS	SPI serial data output
	AN4	PAS1	AN	—	A/D Converter input channel 4

Pin Name	Function	OPT	I/T	O/T	Description
PA6/SCK/SCL/AN5	PA6	PAWU PAPU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	SCK	PAS1 SIMC0	ST	CMOS	SPI serial clock
	SCL	PAS1 SIMC0	ST	CMOS	I^2C clock line
	AN5	PAS1	AN	—	A/D Converter input channel 5
PA7/PTP2I_0	PA7	PAWU PAPU	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	PTP2I_0	PTM2C0 IFS	ST	—	PTM2 iutput
PB0/PTP1I/INT0/AN0/VREF	PB0	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	PTP1I	PBS0 PTM1C0	ST	—	PTM1 iutput
	INT0	PBS0 INTEG	ST	—	External interrupt input
	AN0	PBS0	AN	—	A/D Converter input channel 0
	VREF	PAS0	AN	—	A/D Converter VREF input
PB1/PTP3I_0/$\overline{SCS}$/AN1/VREFO	PB1	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	PTP3I_0	PBS0 PTM3C0	ST	—	PTM3 iutput
	$\overline{SCS}$	PBS0	ST	CMOS	SPI slave select pin
	AN1	PBS0	AN	—	ADC input channel 1
	VREFO	PBS0	—	AN	A/D Converter reference voltage output
PB2/PTCK3_0/INT1/AN2	PB2	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	PTCK3_0	PBS0 PTM3C0 IFS	ST	—	PTM3 clock input
	INT1	PBS0 INTEG	ST	—	External interrupt input
	AN2	PBS0	AN	—	A/D Converter input channel 2
PB3/CTCK1_0/XT1	PB3	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up
	CTCK1_0	CTM1C0 IFS	ST	—	CTM1 clock input
	XT1	CO	LXT	—	LXT oscillator pin
PB4/PTCK1_0/XT2	PB4	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up
	PTCK1_0	PTM1C0	ST	—	PTM1 clock input
	XT2	CO	—	LXT	LXT oscillator pin
PB5/PTP3/SCOM3	PB5	PBPU PBS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	PTP3	PBS1	—	CMOS	PTM3 output
	SCOM3	PBS1	—	AN	LCD driver output for LCD panel common
PB6/PTP2/SCOM2	PB6	PBPU PBS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	PTP2	PBS1	—	CMOS	PTM2 output
	SCOM2	PBS1	—	AN	LCD driver output for LCD panel common

Pin Name	Function	OPT	I/T	O/T	Description
PC0/PTP1	PC0	PCPU PCS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	PTP1	PCS0	—	CMOS	PTM1 output
PC1/PTCK0/SCOM1	PC1	PCPU PCS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	PTCK0	PCS0 PTM0C0	ST	—	PTM0 clock input
	SCOM1	PCS0	—	AN	LCD driver output for LCD panel common
PC2/CTP0/$\overline{RES}$	PC2	PCPU PCS0 RSTC	ST	CMOS	General purpose I/O. Register enabled pull-up
	CTP0	PCS0 RSTC	—	CMOS	CTM0 output
	$\overline{RES}$	RSTC	ST	—	External reset input
PC3/PTP2I_1/SCOM2	PC3	PCPU PCS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	PTP2I_1	PCS0 PTM2C0 IFS	ST	—	PTM2 input
	SCOM2	PCS0	—	AN	LCD driver output for LCD panel common
PC4/CTCK1_1/ SCOM3	PC4	PCPU PCS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	CTCK1_1	PCS1 CTM1C0 IFS	ST	—	CTM1 clock input
	SCOM3	PCS1	—	AN	LCD driver output for LCD panel common
PC5/PTCK3_1	PC5	PCPU PCS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	PTCK3_1	PCS1 PTM3C0 IFS	ST	—	PTM3 clock input
PC6/PTCK2_1	PC6	PCPU PCS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	PTCK2_1	PCS1 PTM2C0 IFS	ST	—	PTM2 clock input
AVDD/ VDD	VDD	—	PWR	—	Digital positive power supply
	AVDD	—	PWR	—	Analog positive power supply
AVSS/ VSS	VSS	—	PWR	—	Digital negative power supply
	AVSS	—	PWR	—	Analog negative power supply

Note: I/T: Input type
O/T: Output type
OP: Optional by Configuration Option (CO) or register option
PWR: Power
ST: Schmitt Trigger input
CO: Configuration option
CMOS: CMOS output
AN: Analog signal
LXT: Low frequency crystal oscillator
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.

HT66F0082

Pin Name	Function	OPT	I/T	O/T	Description
PA0/PTP0I_0/AN6/ ICPDA/OCDSDA	PA0	PAWU PAPU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	PTP0I_0	PAS0 PTM0C0 IFS	ST	—	PTM0 input
	AN6	PAS0	AN	—	A/D Converter input channel 6
	ICPDA	—	ST	CMOS	ICP Data Line
	OCDSDA	—	ST	CMOS	On Chip Debug System Data Line (OCDS EV only)
PA1/PTP0/AN7	PA1	PAWU PAPU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	PTP0	PAS0	—	CMOS	PTM0 output
	AN7	PAS0	AN	—	A/D Converter input channel 7
PA2/CTCK0_0/ICPCK/ OCDSCK	PA2	PAWU PAPU	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	CTCK0_0	CTM0C0 IFS	ST	—	CTM0 clock input
	ICPCK	—	ST	—	ICP Clock Line
	OCDSCK	—	ST	—	On Chip Debug System Clock Line (OCDS EV only)
PA3/CTP1/SCOM0	PA3	PAWU PAPU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	CTP1	PAS0	—	CMOS	CTM1output
	SCOM0	PAS0	—	AN	LCD driver output for LCD panel common
PA4/SDI/SDA/AN3	PA4	PAWU PAPU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	SDI	PAS1 SIMC0	ST	—	SPI serial data input
	SDA	PAS1 SIMC0	ST	CMOS	I^2C data line
	AN3	PAS1	AN	—	A/D Converter input channel 3
PA5/PTCK2_0/SDO/ AN4	PA5	PAWU PAPU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	PTCK2_0	PAS1 PTM2C0 IFS	ST	—	PTM2 clock input
	SDO	PAS1	—	CMOS	SPI serial data output
	AN4	PAS1	AN	—	A/D Converter input channel 4
PA6/SCK/SCL/AN5	PA6	PAWU PAPU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	SCK	PAS1 SIMC0	ST	CMOS	SPI serial clock
	SCL	PAS1 SIMC0	ST	CMOS	I^2C clock line
	AN5	PAS1	AN	—	A/D Converter input channel 5
PA7/PTP2I_0	PA7	PAWU PAPU	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	PTP2I_0	PTM2C0 IFS	ST	—	PTM2 iutput

Pin Name	Function	OPT	I/T	O/T	Description
PB0/PTP1I/INT0/AN0/VREF	PB0	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	PTP1I	PBS0 PTM1C0	ST	—	PTM1 iutput
	INT0	PBS0 INTEG	ST	—	External interrupt input
	AN0	PBS0	AN	—	A/D Converter input channel 0
	VREF	PAS0	AN	—	A/D Converter VREF input
PB1/PTP3I_0/$\overline{SCS}$/AN1/VREFO	PB1	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	PTP3I_0	PBS0 PTM3C0 IFS	ST	—	PTM3 iutput
	$\overline{SCS}$	PBS0	ST	CMOS	SPI slave select pin
	AN1	PBS0	AN	—	ADC input channel 1
	VREFO	PBS0	—	AN	A/D Converter reference voltage output
PB2/PTCK3_0/INT1/AN2	PB2	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	PTCK3_0	PBS0 PTM3C0 IFS	ST	—	PTM3 clock input
	INT1	PBS0 INTEG	ST	—	External interrupt input
	AN2	PBS0	AN	—	A/D Converter input channel 2
PB3/CTCK1_0/XT1	PB3	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up
	CTCK1_0	CTM1C0 IFS	ST	—	CTM1 clock input
	XT1	CO	LXT	—	LXT oscillator pin
PB4/PTCK1_0/XT2	PB4	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up
	PTCK1_0	PTM1C0 IFS	ST	—	PTM1 clock input
	XT2	CO	—	LXT	LXT oscillator pin
PB5/PTP3/SCOM3	PB5	PBPU PBS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	PTP3	PBS1	—	CMOS	PTM3 output
	SCOM3	PBS1	—	AN	LCD driver output for LCD panel common
PB6/PTP2/SCOM2	PB6	PBPU PBS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	PTP2	PBS1	—	CMOS	PTM2 output
	SCOM2	PBS1	—	AN	LCD driver output for LCD panel common
PC0/PTP1	PC0	PCPU PCS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	PTP1	PCS0	—	CMOS	PTM1 output
PC1/PTCK0/SCOM1	PC1	PCPU PCS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	PTCK0	PCS0 PTM0C0	ST	—	PTM0 clock input
	SCOM1	PCS0	—	AN	LCD driver output for LCD panel common

Pin Name	Function	OPT	I/T	O/T	Description
PC2/CTP0/$\overline{RES}$	PC2	PCPU PCS0 RSTC	ST	CMOS	General purpose I/O. Register enabled pull-up
	CTP0	PCS0 RSTC	—	CMOS	CTM0 output
	$\overline{RES}$	RSTC	ST	—	External reset input
PC3/PTP2I_1/SCOM2	PC3	PCPU PCS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	PTP2I_1	PCS0 PTM2C0 IFS	ST	—	PTM2 input
	SCOM2	PCS0	—	AN	LCD driver output for LCD panel common
PC4/CTCK1_1/ SCOM3	PC4	PCPU PCS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	CTCK1_1	PCS1 CTM1C0 IFS	ST	—	CTM1 clock input
	SCOM3	PCS1	—	AN	LCD driver output for LCD panel common
PC5/PTCK3_1/RX	PC5	PCPU PCS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	PTCK3_1	PCS1 PTM3C0 IFS	ST	—	PTM3 clock input
	RX	PCS1	ST	—	UART receiver pin (only for HT66F0082)
PC6/PTCK2_1/TX	PC6	PCPU PCS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	PTCK2_1	PCS1 PTM2C0 IFS	ST	—	PTM2 clock input
	TX	PCS1	—	CMOS	UART transmitter pin (only for HT66F0082)
PD0/CTCK0_1/ SCOM0	PD0	PDPU PDS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	CTCK0_1	PDS0 CTM0C0 IFS	ST	—	CTM0 clock input
	SCOM0	PDS0	—	AN	LCD driver output for LCD panel common
PD1/PTP0I_1/SCOM1	PD1	PDPU PDS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	PTP0I_1	PDS0 PTM0C0 IFS	ST	—	PTM0 input
	SCOM1	PDS0	—	AN	LCD driver output for LCD panel common
PD2/PTP3I_1	PD2	PDPU	ST	CMOS	General purpose I/O. Register enabled pull-up
	PTP3I_1	PTM3C0 IFS	ST	—	PTM3 input
PD3/PTCK1_1	PC3	PCPU RSTC	ST	CMOS	General purpose I/O. Register enabled pull-up
	PTCK1_1	PTM1C0 IFS	ST	—	PTM1 clock input

Pin Name	Function	OPT	I/T	O/T	Description
AVDD/ VDD	VDD	—	PWR	—	Digital positive power supply
	AVDD	—	PWR	—	Analog positive power supply
AVSS/ VSS	VSS	—	PWR	—	Digital negative power supply
	AVSS	—	PWR	—	Analog negative power supply

Note: I/T: Input type
O/T: Output type
OP: Optional by configuration option(CO) or register option
PWR: Power
ST: Schmitt Trigger input
CO: Configuration option
CMOS: CMOS output
AN: Analog signal
LXT: Low frequency crystal oscillator
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

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

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

I_{OL} Total 80mA

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

Total Power Dissipation 500mW

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

D.C. Characteristics

Ta = 25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	Operating Voltage (HIRC)	—	$f_{SYS}=f_{HIRC}$=8MHz	2.2	—	5.5	V
			$f_{SYS}=f_{HIRC}$=12MHz	2.7	—	5.5	V
			$f_{SYS}=f_{HIRC}$=16MHz	3.3	—	5.5	V
	Operating Voltage (LXT)	—	$f_{SYS}=f_{LXT}$=32768Hz	2.2	—	5.5	V
	Operating Voltage (LIRC)	—	$f_{SYS}=f_{LIRC}$=32kHz	2.2	—	5.5	V
I_{DD}	Operating Current (HIRC)	3V	No load, $f_{SYS}=f_{HIRC}$=8MHz, ADC off, WDT enable, LVR enable	—	1.0	2.0	mA
		5V		—	2.5	4.0	mA
		3V	No load, $f_{SYS}=f_{HIRC}$=12MHz, ADC off, WDT enable, LVR enable	—	1.5	2.5	mA
		5V		—	3.5	5.5	mA
		5V	No load, $f_{SYS}=f_{HIRC}$=16MHz, ADC off, WDT enable, LVR enable	—	4.5	7.0	mA
	Operating Current (LXT)	3V	No load, $f_{SYS}=f_{LXT}$=32768Hz, ADC off, WDT enable, LVR enable, LXTLP=0	—	20	40	μA
		5V		—	40	80	μA
		3V	No load, $f_{SYS}=f_{LXT}$=32768Hz, ADC off, WDT enable, LVR enable, LXTLP=1	—	15	30	μA
		5V		—	30	60	μA
	Operating Current (LIRC)	3V	No load, $f_{SYS}=f_{LICRC}$=32kHz, ADC off, WDT enable, LVR enable	—	15	30	μA
		5V		—	30	60	μA
	Operating Current, Normal Mode, f_H=16MHz (HIRC)	5V	No load, $f_{SYS}=f_H/2$, ADC off, WDT enable, LVR enable	—	3.0	4.0	mA
		5V	No load, $f_{SYS}=f_H/64$, ADC off, WDT enable, LVR enable	—	1.6	2.2	mA
	Operating Current, Normal Mode, f_H=12MHz (HIRC)	3V	No load, $f_{SYS}=f_H/2$, ADC off, WDT enable, LVR enable	—	1.5	2.2	mA
		5V		—	2.2	3.0	mA
		3V	No load, $f_{SYS}=f_H/64$, ADC off, WDT enable, LVR enable	—	0.7	1.2	mA
		5V		—	1.2	1.6	mA
	Operating Current, Normal Mode, f_H=8MHz (HIRC)	3V	No load, $f_{SYS}=f_H/2$, ADC off, WDT enable, LVR enable	—	1.0	1.5	mA
		5V		—	1.5	2.0	mA
		3V	No load, $f_{SYS}=f_H/64$, ADC off, WDT enable, LVR enable	—	0.5	0.8	mA
		5V		—	0.8	1.1	mA

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{STB}	SLEEP0 Mode Standby Current (LIRC and LXT Off)	3V	No load, all peripherals off, WDT off	—	0.2	0.8	μA
		5V		—	0.5	1.0	μA
	SLEEP1 Mode Standby Current (LIRC On)	3V	No load, ADC off, WDT enable, LVR disable	—	1.3	3.0	μA
		5V		—	5.0	10	μA
	SLEEP1 Mode Standby Current (LXT On)	3V	No load, ADC off, WDT enable, LVR disable, LXTLP=0	—	2.0	4.0	μA
		5V		—	4.0	6.0	μA
		3V	No load, ADC off, WDT enable, LVR disable, LXTLP=1	—	1.5	3.0	μA
		5V		—	3.0	5.0	μA
	IDLE0 Mode Standby Current (LIRC On)	3V	No load, ADC off, WDT enable, LVR disable	—	1.3	3.0	μA
		5V		—	5.0	10	μA
	IDLE0 Mode Standby Current (LXT On)	3V	No load, ADC off, WDT enable, LVR disable, LXTLP=0	—	2.0	4.0	μA
		5V		—	4.0	6.0	μA
		3V	No load, ADC off, WDT enable, LVR disable, LXTLP=1	—	1.5	3.0	μA
		5V		—	3.0	5.0	μA
	IDLE1 Mode Standby Current (HIRC On)	3V	No load, ADC off, WDT enable, $f_{SYS}=f_{HIRC}$=8MHz on	—	0.8	1.6	mA
		5V		—	1.0	2.0	mA
		3V	No load, ADC off, WDT enable, $f_{SYS}=f_{HIRC}$=12MHz on	—	1.2	2.4	mA
		5V		—	1.5	3.0	mA
		5V	No load, ADC off, WDT enable, $f_{SYS}=f_{HIRC}$=16MHz on	—	2.0	4.0	mA
V_{IL}	Input Low Voltage for I/O Ports or Input Pins except $\overline{RES}$ Pin	5V	—	0	—	1.5	V
		—	—	0	—	$0.2V_{DD}$	V
	Input Low Voltage for $\overline{RES}$ Pin	—	—	0	—	$0.4V_{DD}$	V
V_{IH}	Input High Voltage for I/O Ports or Input Pins except $\overline{RES}$ Pin	5V	—	3.5	—	5.0	V
		—	—	$0.8V_{DD}$	—	V_{DD}	V
	Input High Voltage for $\overline{RES}$ Pin	—	—	$0.9V_{DD}$	—	V_{DD}	V
I_{OL}	I/O Port Sink Current	3V	$V_{OL}=0.1V_{DD}$	18	36	—	mA
		5V	$V_{OL}=0.1V_{DD}$	40	80	—	mA
I_{OH}	I/O Port, Source Current	3V	$V_{OH}=0.9V_{DD}$, PxPS=00	-1.0	-2.0	—	mA
		5V	$V_{OH}=0.9V_{DD}$, PxPS=00	-2.0	-4.0	—	mA
		3V	$V_{OH}=0.9V_{DD}$, PxPS=01	-1.75	-3.5	—	mA
		5V	$V_{OH}=0.9V_{DD}$, PxPS=01	-3.5	-7.0	—	mA
		3V	$V_{OH}=0.9V_{DD}$, PxPS=10	-2.5	-5.0	—	mA
		5V	$V_{OH}=0.9V_{DD}$, PxPS=10	-5.0	-10	—	mA
		3V	$V_{OH}=0.9V_{DD}$, PxPS=11	-5.5	-11	—	mA
		5V	$V_{OH}=0.9V_{DD}$, PxPS=11	-11	-22	—	mA
R_{PH}	Pull-high Resistance for I/O Ports	3V	—	20	60	100	kΩ
		5V	—	10	30	50	kΩ
I_{OCDS}	Operating Current, Normal Mode, $f_{SYS}=f_H$ (HIRC) (for OCDS EV Testing, Connect to an e-Link)	3V	No load, f_H=8MHz, ADC off, WDT enable	—	1.4	2.0	mA

A.C. Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Condition				
f_{SYS}	System Clock (HIRC)	2.2V~5.5V	$f_{SYS}=f_{HIRC}$=8MHz	—	8	—	MHz
		2.7V~5.5V	$f_{SYS}=f_{HIRC}$=12MHz	—	12	—	MHz
		3.3V~5.5V	$f_{SYS}=f_{HIRC}$=16MHz	—	16	—	MHz
	System Clock (LXT)	2.2V~5.5V	$f_{SYS}=f_{LXT}$=32768Hz	—	32768	—	Hz
	System Clock (LIRC)	2.2V~5.5V	$f_{SYS}=f_{LIRC}$=32kHz	—	32	—	kHz
f_{HIRC}	High Speed Internal RC Oscillator (HIRC Trim @ V_{DD}=3V/5V, Ta=25°C)	2.2V~5.5V	Ta= -40°C to 85°C	-3%	8	+3%	MHz
		2.7V~5.5V		-3%	12	+3%	MHz
		3.3V~5.5V		-3%	16	+3%	MHz
f_{LIRC}	System Clock (32kHz RC Oscillator)	2.2V~5.5V	Ta= -40°C to 85°C	8	32	50	kHz
f_{LXT}	System Clock (32768Hz Crystal Oscillator)	2.2V~5.5V	Ta=25°C	—	32768	—	Hz
t_{TCK}	xTCKn, PTPnI Input Pulse Width	—	—	0.3	—	—	μs
t_{RES}	External Reset Low Pulse Width	—	—	10	—	—	μs
t_{INT}	Interrupt Pulse Width	—	—	0.3	—	—	μs
t_{EERD}	EEPROM Read Time	—	—	—	2	4	t_{SYS}
t_{EEWR}	EEPROM Write Time	—	—	—	2	6.5	ms
t_{SST}	System Start-up Timer Period (Wake-up from HALT, f_{SYS} Off at HALT State, Reset Pin Reset)	—	f_{SYS}=LXT	1024	—	—	t_{SYS}
			f_{SYS}=HIRC	16	—	—	
			f_{SYS}=LIRC	2	—	—	
	System Start-up Timer Period (Wake-up from HALT, f_{SYS} on at HALT State)	—	—	2	—	—	t_{SYS}
t_{RSTD}	System Reset Delay Time (Power On Reset, LVR Reset, WDT S/W Reset (WDTC)	—	—	6.25	50	125	ms
	System Reset Delay Time ($\overline{RES}$ Pin Reset, WDT Normal Reset)	—	—	2.08	16.7	47.9	ms
f_{I2C}	I^2C Standard Mode (100kHz) f_{SYS} Frequency	—	No clock debounce	2	—	—	MHz
		—	2 system clock debounce	4	—	—	MHz
		—	4 system clock debounce	8	—	—	MHz
	I^2C Fast Mode (400kHz) f_{SYS} Frequency	—	No clock debounce	5	—	—	MHz
		—	2 system clock debounce	10	—	—	MHz
		—	4 system clock debounce	20	—	—	MHz

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

2. 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.

ADC Electrical Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
AV_{DD}	A/D Converter Operating Voltage	—	—	2.7	—	5.5	V
V_{ADI}	A/D Converter Input Voltage	—	—	0	—	AV_{DD} /V_{REF}	V
V_{REF}	A/D Converter Reference Voltage	3V	—	2	—	AV_{DD}	V
		5V					
DNL	Differential Non-linearity	2.7V	V_{REF}=AV_{DD}=V_{DD} t_{ADCK} =0.5μs	-3	—	+3	LSB
		3V					
		5V					
INL	Integral Non-linearity	2.7V	V_{REF}=AV_{DD}=V_{DD} t_{ADCK} =0.5μs	-4	—	+4	LSB
		3V					
		5V					
I_{ADC}	Additional Power Consumption if A/D Converter is used	3V	No load (t_{ADCK} =0.5μs)	—	1.0	2.0	mA
		5V	No load (t_{ADCK} =0.5μs)	—	1.5	3.0	mA
t_{ADCK}	A/D Converter Clock Period	2.7V~ 5.5V	—	0.5	—	10	μs
t_{ADC}	A/D Conversion Time (Include Sample and Hold Time)	2.7~ 5.5V	12-bit ADC	16	—	20	t_{ADCK}
t_{ADS}	A/D Converter Sampling Time	2.7V~ 5.5V	—	—	4	—	t_{ADCK}
t_{ON2ST}	A/D Converter On-to-Start Time	2.7V~ 5.5V	—	4	—	—	μs

LVR Electrical Characteristics

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	Operating Voltage	—	—	1.9	—	5.5	V
V_{LVR}	Low Voltage Reset Voltage	—	LVR Enable, 2.1V option	-5%	2.1	+5%	V
BUFO	Reference Output with Buffer	—	T_J = +25°C@3.15V	-5%	1.04	+5%	V
t_{LVR}	Low Voltage Width to Reset	—	—	160	320	640	μs

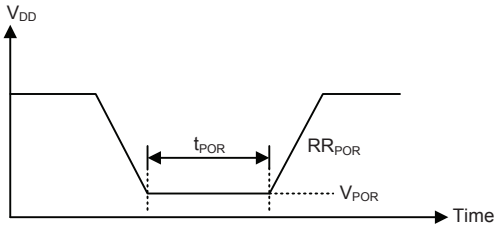
LCD Electrical Characteristics

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{BIAS}	V_{DD}/2 Bias Current for LCD	5V	ISEL[1:0]=00	17.5	25.0	32.5	μA
			ISEL[1:0]=01	35	50	65	μA
			ISEL[1:0]=10	70	100	130	μA
			ISEL[1:0]=11	140	200	260	μA
V_{SCOM}	V_{DD}/2 Voltage for LCD COM Port	2.2V~5.5V	No load	0.475	0.500	0.525	V_{DD}

Power on Reset Electrical Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{POR}	V_{DD} Start Voltage to Ensure Power-on Reset	—	—	—	—	100	mV
RR_{POR}	V_{DD} Rising 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

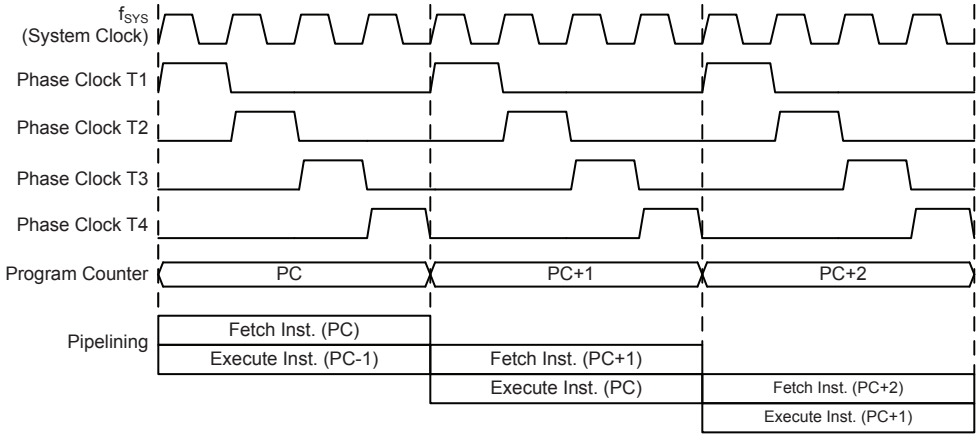


System Architecture

A key factor in the high-performance features of the Holtek range of microcontrollers is attributed to their internal system architecture. The device takes advantage of the usual features found within RISC microcontrollers providing increased speed of operation and Periodic performance. The pipelining scheme is implemented in such a way that instruction fetching and instruction execution are overlapped, hence instructions are effectively executed in one cycle, with the exception of branch or call instructions. 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 these devices suitable for low-cost, high-volume production for controller applications

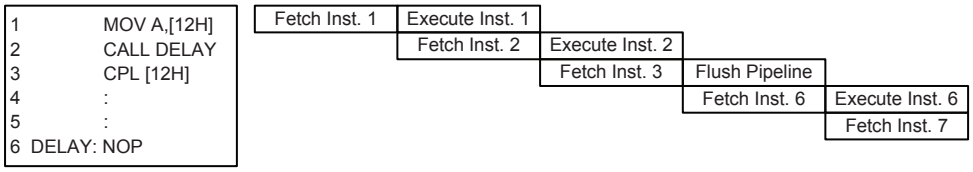
Clocking and Pipelining

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



System Clock and Pipelining

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.



Instruction Fetching

Program Counter

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

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

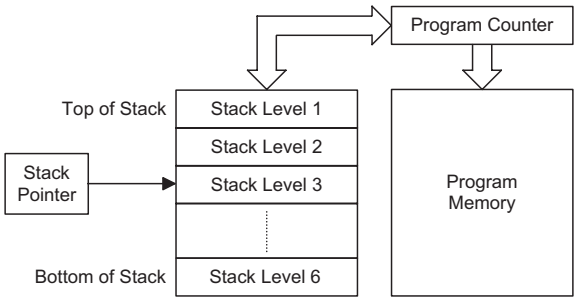
Device	Program Counter	
	Program Counter High byte	PCL Register
HT66F0042	PC10~PC8	PCL7~PCL0
HT66F0082	PC11~PC8	PCL7~PCL0

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

Stack

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

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



Device	Stack Levels
HT66F0042	6
HT66F0082	

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

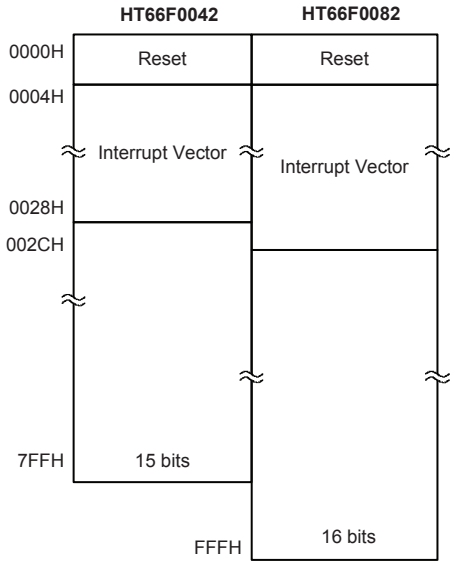
Flash Program Memory

The Program Memory is the location where the user code or program is stored. For these devices the Program Memory are 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, these Flash devices offer users the flexibility to conveniently debug and develop their applications while also offering a means of field programming and updating.

Structure

The Program Memory has a capacity of 2K×15 or 4K×16 bits. The Program Memory is addressed by the Program Counter and also contains data, table information and interrupt entries. Table data, which can be setup in any location within the Program Memory, is addressed by a separate table pointer register.

Device	Program Memory Capacity
HT66F0042	2K×15 bits
HT66F0082	4K×16 bits



Program Memory Structure

Special Vectors

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

Look-up Table

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

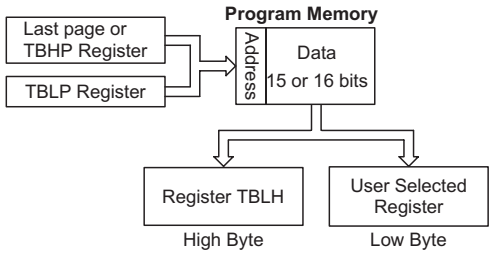


Table Program Example

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 words Program Memory of the HT66F0082 device. 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 specified page that TBHP points to if the "TABRD [m]" instruction is being used. The high byte of the table data which in this case is equal to zero will be transferred to the TBLH register automatically when the "TABRD [m]" instruction is executed.

Because the TBLH register is a read-only register and cannot be restored, care should be taken to ensure its protection if both the main routine and Interrupt Service Routine use table read instructions. If using the table read instructions, the Interrupt Service Routines may change the value of the TBLH and subsequently cause errors if used again by the main routine. As a rule it is recommended that simultaneous use of the table read instructions should be avoided. However, in situations where simultaneous use cannot be avoided, the interrupts should be disabled prior to the execution of any main routine table-read instructions. Note that all table related instructions require two instruction cycles to complete their operation.

Table Read Program Example

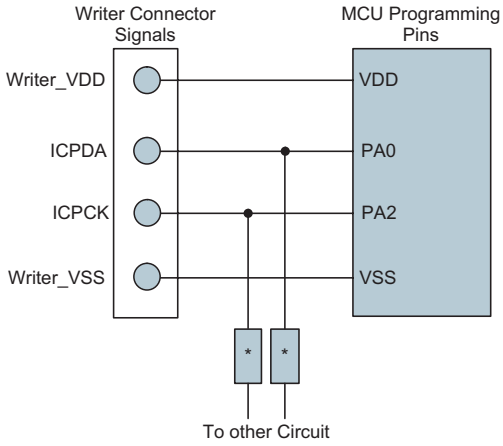
```
tempreg1 db ?      ; temporary register #1
tempreg2 db ?      ; temporary register #2
:
mov a, 06h         ; initialise low table pointer - note that this address is referenced
mov tblp, a        ; to the last page or the page that tbhp pointed
mov a, 0Fh         ; initialise high table pointer
mov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

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 4-pin interface. This provides manufacturers with the possibility of manufacturing their circuit boards complete with a programmed or un-programmed microcontroller, and then programming or upgrading the program at a later stage. This enables product manufacturers to easily keep their manufactured products supplied with the latest program releases without removal and re-insertion of the device.

Holtek Write Pins	MCU Programming Pins	Function
ICPDA	PA0	Programming Serial Data
ICPCK	PA2	Programming Serial Clock
VDD	VDD	Power Supply
VSS	VSS	Ground

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



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

On-Chip Debug Support – OCDS

There are two EV MCU devices, HT66V0042 and HT66V0082, which are used to emulate the HT66F0042 and HT66F0082 MCUs. These EV devices also provide an "On-Chip Debug" function to debug the device during the development process. The EV MCUs and the actual MCU devices are almost functionally compatible except for the "On-Chip Debug" function. Users can use the EV chip device to emulate the real chip device behavior by connecting the OCDSDA and OCDSCK pins to the Holtek HT-IDE development tools. The OCDSDA pin is the OCDS Data/Address input/output pin while the OCDSCK pin is the OCDS clock input pin. When users use the EV chip for debugging, other functions which are shared with the OCDSDA and OCDSCK pins in the actual MCU device will have no effect in the EV chip. However, the two OCDS pins which are pin-shared with the ICP programming pins are still used as the Flash Memory programming pins for ICP. For a more detailed OCDS description, refer to the corresponding document named "Holtek e-Link for 8-bit MCU OCDS User's Guide".

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

RAM 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. The second area of Data Memory is known as the General Purpose Data Memory, which is reserved for general purpose use. All locations within this area are read and write accessible under program control.

The overall Data Memory is subdivided into two banks. The Special Purpose Data Memory registers are accessible in all banks, with the exception of the EEC register at address 40H, which is only accessible in Bank 1. Switching between the different Data Memory banks is achieved by setting the Bank Pointer to the correct value. The start address of the Data Memory for all devices is the address 00H.

Device	Capacity	Bank 0	Bank 1
HT66F0042	96×8	80H~DFH	40H EEC register only
HT66F0082	128×8	80H~FFH	40H EEC register only

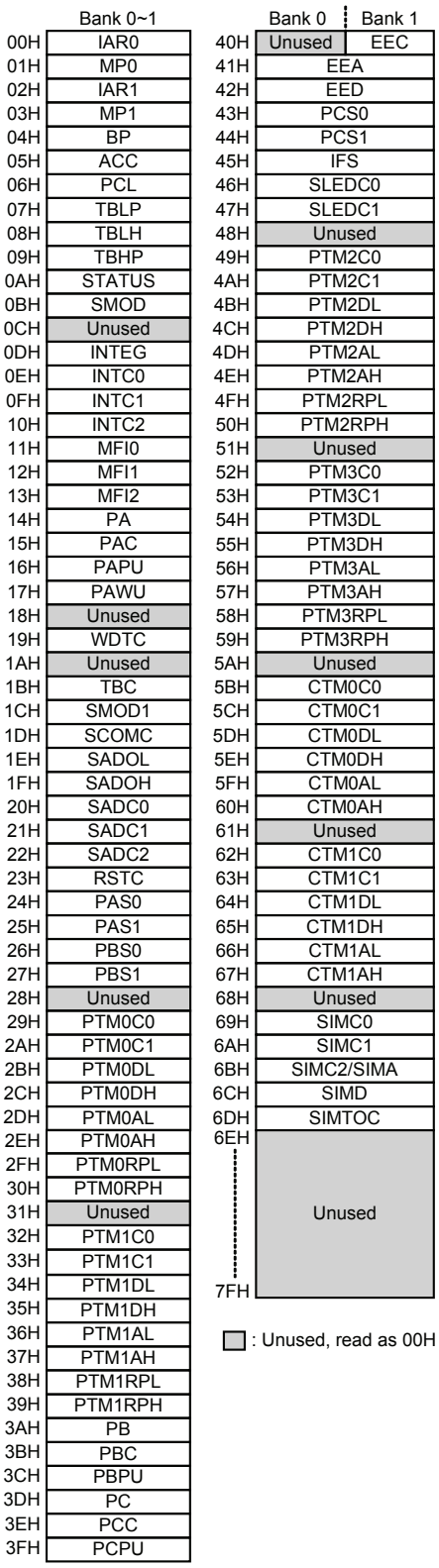
Data Memory Structure

General Purpose Data Memory

There are 96 or 128 bytes of general purpose data memory which are located in Bank 0. All microcontroller programs require an area of read/write memory where temporary data can be stored and retrieved for use later. It is this area of RAM memory that is known as General Purpose Data Memory. This area of Data Memory is fully accessible by the user programing for both reading and writing operations. By using the bit operation instructions individual bits can be set or reset under program control giving the user a large range of flexibility for bit manipulation in the Data Memory.

Special Purpose Data Memory

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



Special Purpose Data Memory Structure – HT66F0042

	Bank 0~1
00H	IAR0
01H	MP0
02H	IAR1
03H	MP1
04H	BP
05H	ACC
06H	PCL
07H	TBLP
08H	TBLH
09H	TBHP
0AH	STATUS
0BH	SMOD
0CH	Unused
0DH	INTEG
0EH	INTC0
0FH	INTC1
10H	INTC2
11H	MFI0
12H	MFI1
13H	MFI2
14H	PA
15H	PAC
16H	PAPU
17H	PAWU
18H	Unused
19H	WDTC
1AH	Unused
1BH	TBC
1CH	SMOD1
1DH	SCOMC
1EH	SADOL
1FH	SADOH
20H	SADC0
21H	SADC1
22H	SADC2
23H	RSTC
24H	PAS0
25H	PAS1
26H	PBS0
27H	PBS1
28H	Unused
29H	PTM0C0
2AH	PTM0C1
2BH	PTM0DL
2CH	PTM0DH
2DH	PTM0AL
2EH	PTM0AH
2FH	PTM0RPL
30H	PTM0RPH
31H	Unused
32H	PTM1C0
33H	PTM1C1
34H	PTM1DL
35H	PTM1DH
36H	PTM1AL
37H	PTM1AH
38H	PTM1RPL
39H	PTM1RPH
3AH	PB
3BH	PBC
3CH	PBPU
3DH	PC
3EH	PCC
3FH	PCPU

	Bank 0	Bank 1
40H	Unused	EEC
41H	EEA	
42H	EED	
43H	PCS0	
44H	PCS1	
45H	IFS	
46H	SLEDC0	
47H	SLEDC1	
48H	Unused	
49H	PTM2C0	
4AH	PTM2C1	
4BH	PTM2DL	
4CH	PTM2DH	
4DH	PTM2AL	
4EH	PTM2AH	
4FH	PTM2RPL	
50H	PTM2RPH	
51H	Unused	
52H	PTM3C0	
53H	PTM3C1	
54H	PTM3DL	
55H	PTM3DH	
56H	PTM3AL	
57H	PTM3AH	
58H	PTM3RPL	
59H	PTM3RPH	
5AH	Unused	
5BH	CTM0C0	
5CH	CTM0C1	
5DH	CTM0DL	
5EH	CTM0DH	
5FH	CTM0AL	
60H	CTM0AH	
61H	Unused	
62H	CTM1C0	
63H	CTM1C1	
64H	CTM1DL	
65H	CTM1DH	
66H	CTM1AL	
67H	CTM1AH	
68H	Unused	
69H	SIMC0	
6AH	SIMC1	
6BH	SIMC2/SIMA	
6CH	SIMD	
6DH	SIMTOC	
6EH	PD	
6FH	PDC	
70H	PDPU	
71H	PDS0	
72H	USR	
73H	UCR1	
74H	UCR2	
75H	TXR_RXR	
76H	BRG	
77H ~ 7FH	Unused	

: Unused, read as 00H

Special Purpose Data Memory Structure – HT66F0082

Special Function Register Description

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

Indirect Addressing Registers – IAR0, IAR1

The Indirect Addressing Registers, IAR0 and IAR1, although having their locations in normal RAM register space, do not actually physically exist as normal registers. The method of indirect addressing for RAM data manipulation uses these Indirect Addressing Registers and Memory Pointers, in contrast to direct memory addressing, where the actual memory address is specified. Actions on the IAR0 and IAR1 registers will result in no actual read or write operation to these registers but rather to the memory location specified by their corresponding Memory Pointers, MP0 or MP1. Acting as a pair, IAR0 and MP0 can together access data 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´
org00h
start:
     mov a,04h            ; setup size of block
     mov block,a
     mov a,offset adres1  ; Accumulator loaded with first RAM address
     mov mp0,a            ; setup memory pointer with first RAM address
loop:
     clr IAR0             ; clear the data at address defined by mp0
     inc mp0              ; increment memory pointer
     sdz block            ; check if last memory location has been cleared
     jmp loop
continue:
```

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

Bank Pointer – BP

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

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 Bank1 must be implemented using Indirect Addressing.

BP Register

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

Bit 7 ~ 1 Unimplemented, read as "0"

Bit 0 **DMBP0**: Select Data Memory Banks
0: Bank 0
1: Bank 1

Accumulator – ACC

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

Program Counter Low Register – PCL

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

Look-up Table Registers – TBLP, TBHP, TBLH

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

Status Register – STATUS

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

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

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

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

In addition, on entering an interrupt sequence or executing a subroutine call, the status register will not be pushed onto the stack automatically. If the contents of the status 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	×	×	×	×

"×" unknown

Bit 7~6 Unimplemented, read as "0"

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

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

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

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

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

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

EEPROM Data Memory

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

EEPROM Data Memory Structure

The EEPROM Data Memory capacity is 32×8 bits or 64×8 bits for this series of devices. Unlike the Program Memory and RAM Data Memory, the EEPROM Data Memory is not directly mapped and is therefore not directly accessible in the same way as the other types of memory. Read and Write operations to the EEPROM are carried out in single byte operations using one address register and one data register in Bank 0 and a single control register in Bank 1.

Device	Capacity
HT66F0042	32×8 bits
HT66F0082	64×8 bits

EEPROM Registers

Three registers control the overall operation of the internal EEPROM Data Memory. These are the address registers, EEA, the data register, EED and a single control register, EEC. As both the EEA and EED registers are located in Bank 0, they can be directly accessed in the same way as any other Special Function Register. The EEC register however, being located in Bank1, cannot be directly addressed directly and can only be read from or written to indirectly using the MP1 Memory Pointer and Indirect Addressing Register, IAR1. Because the EEC control register is located at address 40H in Bank 1, the MP1 Memory Pointer must first be set to the value 40H and the Bank Pointer register, BP, set to the value, 01H, before any operations on the EEC register are executed.

Register Name	Bit							
	7	6	5	4	3	2	1	0
EEA – HT66F0042	—	—	—	D4	D3	D2	D1	D0
EEA – HT66F0082	—	—	D5	D4	D3	D2	D1	D0
EED	D7	D6	D5	D4	D3	D2	D1	D0
EEC	—	—	—	—	WREN	WR	RDEN	RD

EEPROM Control Registers List

EEA Register – HT66F0042

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

Bit 7 ~ 5 Unimplemented, read as "0"

Bit 4 ~ 0 Data EEPROM address
Data EEPROM address bit 4 ~ bit 0

EEA Register – HT66F0082

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

Bit 7 ~ 6 Unimplemented, read as "0"

Bit 5 ~ 0 Data EEPROM address
Data EEPROM address bit 5 ~ bit 0

EED Register

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

Bit 7 ~ 0 Data EEPROM data
Data EEPROM data bit 7 ~ bit 0

EEC Register

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

Bit 7 ~ 4 Unimplemented, read as "0"

Bit 3 **WREN**: Data EEPROM Write Enable
0: Disable
1: Enable
This is the Data EEPROM Write Enable Bit which must be set high before Data EEPROM write operations are carried out. Clearing this bit to zero will inhibit Data EEPROM write operations.

Bit 2 **WR**: EEPROM Write Control
0: Write cycle has finished
1: Activate a write cycle
This is the Data EEPROM Write Control Bit and when set high by the application program will activate a write cycle. This bit will be automatically reset to zero by the hardware after the write cycle has finished. Setting this bit high will have no effect if the WREN has not first been set high.

Bit 1 **RDEN**: Data EEPROM Read Enable
0: Disable
1: Enable
This is the Data EEPROM Read Enable Bit which must be set high before Data EEPROM read operations are carried out. Clearing this bit to zero will inhibit Data EEPROM read operations.

Bit 0 **RD**: EEPROM Read Control
0: Read cycle has finished
1: Activate a read cycle
This is the Data EEPROM Read Control Bit and when set high by the application program will activate a read cycle. This bit will be automatically reset to zero by the hardware after the read cycle has finished. Setting this bit high will have no effect if the RDEN has not first been set high.

Note: The WREN, WR, RDEN and RD can not be set to "1" at the same time in one instruction. The WR and RD can not be set to "1" at the same time.

Reading Data from the EEPROM

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

Writing Data to the EEPROM

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

Write Protection

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

EEPROM Interrupt

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

Programming Considerations

Care must be taken that data is not inadvertently written to the EEPROM. Protection can be Periodic by ensuring that the Write Enable bit is normally cleared to zero when not writing. Also the Bank Pointer could be normally cleared to zero as this would inhibit access to Bank1where the EEPROM control register exist. Although certainly not necessary, consideration might be given in the application program to the checking of the validity of new write data by a simple read back process. When writing data the WR bit must be set high immediately after the WREN bit has been set high, to ensure the write cycle executes correctly. The global interrupt bit EMI should also be cleared before a write cycle is executed and then re-enabled after the write cycle starts. Note that the devices should not enter the IDLE or SLEEP mode until the EEPROM read or write operation is totally complete. Otherwise, the EEPROM read or write operation will fail.

Programming Examples

Reading data from the EEPROM – polling method

```
MOV  A, EEPROM_ADRES          ; user defined address
MOV  EEA, A
MOV  A, 040H                  ; setup memory pointer MP1
MOV  MP1, A                   ; MP1 points to EEC register
MOV  A, 01H                   ; setup Bank Pointer
MOV  BP, A
SET  IAR1.1                   ; set RDEN bit, enable read operations
SET  IAR1.0                   ; start Read Cycle - set RD bit
BACK:
SZ   IAR1.0                   ; check for read cycle end
JMP  BACK
CLR  IAR1                     ; disable EEPROM write
CLR  BP
MOV  A, EED                   ; move read data to register
MOV  READ_DATA, A
```

Writing Data to the EEPROM – polling method

```
MOV  A, EEPROM_ADRES          ; user defined address
MOV  EEA, A
MOV  A, EEPROM_DATA           ; user defined data
MOV  EED, A
MOV  A, 040H                  ; setup memory pointer MP1
MOV  MP1, A                   ; MP1 points to EEC register
MOV  A, 01H                   ; setup Bank Pointer
MOV  BP, A
CLR  EMI
SET  IAR1.3                   ; set WREN bit, enable write operations
SET  IAR1.2                   ; start Write Cycle - set WR bit- executed immediately
                              ; after set WREN bit
SET  EMI
BACK:
SZ   IAR1.2                   ; check for write cycle end
JMP  BACK
CLR  IAR1                     ; disable EEPROM write
CLR  BP
```

Oscillators

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

Oscillator Overview

The devices include two internal oscillators and an external oscillator. In addition to being the source of the main system clock the oscillators also provide clock sources for the Watchdog Timer, Time Bases and TMs. External oscillator 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. The low speed oscillators are selected through the configuration option. 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 devices have the flexibility to optimize the performance/power ratio, a feature especially important in power sensitive portable applications.

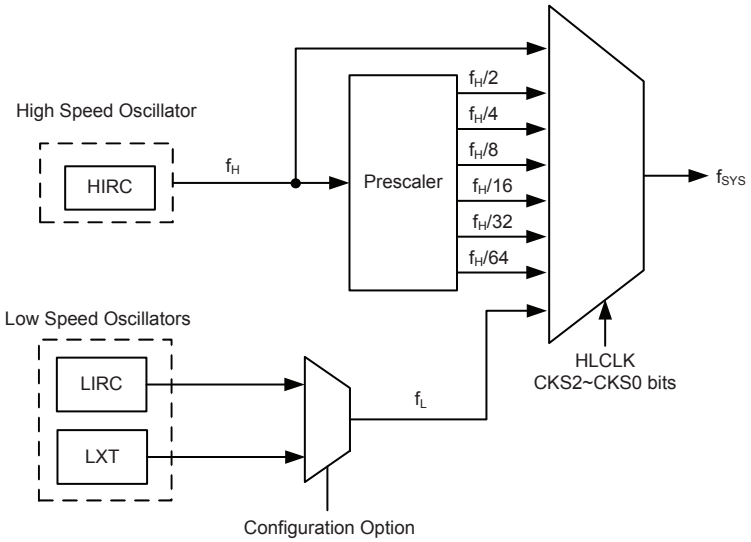
Type	Name	Freq.	Pins
Internal High Speed RC	HIRC	8/12/16MHz	—
Internal Low Speed RC	LIRC	32kHz	—
External Low Speed Crystal	LXT	32.768kHz	XT1/XT2

Oscillator Types

System Clock Configurations

There are three methods of generating the system clock, a high speed oscillator and two low speed oscillators. The high speed oscillator is the internal 8MHz, 12MHz, 16MHz RC oscillator. The two low speed oscillators are the internal 32kHz oscillator, LIRC, and the external 32.768kHz crystal oscillator, LXT. 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.

The actual source clock used for the low speed oscillator is chosen via configuration option. 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.



System Oscillator Configurations

Internal RC Oscillator – HIRC

The internal high speed RC oscillator, HIRC, is a fully integrated system oscillator requiring no external components. The oscillator can be selected to be either 8MHz, 12MHz or 16MHz via the Configuration option. Device trimming during the manufacturing process and the inclusion of internal frequency compensation circuits are used to ensure that the influence of the power supply voltage, temperature and process variations on the oscillation frequency are minimised.

Internal 32kHz Oscillator – LIRC

The Internal 32kHz System Oscillator is one of the low frequency oscillators. It is a fully integrated RC oscillator with a typical frequency of 32kHz at 5V, requiring no external components for its implementation.

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. The 32.768kHz Crystal Oscillator also includes an internal resistor and capacitor, which is also selected via configuration option. Selecting this option may eliminate the need for external components C1, C2 and R_P. In this case it is only required to connect a 32.768kHz crystal to pins XT1 and XT2.

When the configuration selects the external resistor and capacitor, after a 32.768kHz crystal is connected between pins XT1 and XT2, the external resistor and capacitor components may be 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.

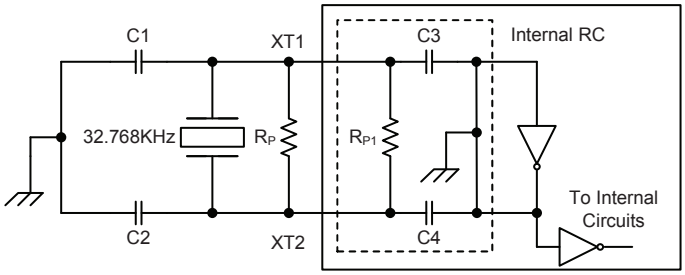
When the microcontroller enters the SLEEP or IDLE Mode, the system clock is switched off to stop microcontroller activity and to conserve power. However, in many microcontroller applications it may be necessary to keep the internal timers operational even when the microcontroller is in the SLEEP or IDLE 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. Note that the wire connected between the 32.768kHz crystal and the XT1/XT2 pins should be kept as short as possible to minimize the stray noise interface.

The configuration option determines if the XT1/XT2 pins are used for the LXT oscillator or as I/O pins or other pin-shared functions.

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

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



Note: 1. When the LXT configuration option selects the External Resistor and Capacitor, then the R_P, C1 and C2 components are required.

2. When the LXT configuration option selects the Integrated Resistor and Capacitor, it is only necessary to connect the 32.768kHz crystal between pins XT1 and XT2

3. 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	8pF	10pF
Note: 1. C1 and C2 values are for guidance only. 2. R_P=5~10MΩ 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 TBC 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.

Supplementary Oscillator

The low speed oscillator, in addition to providing a system clock source is 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

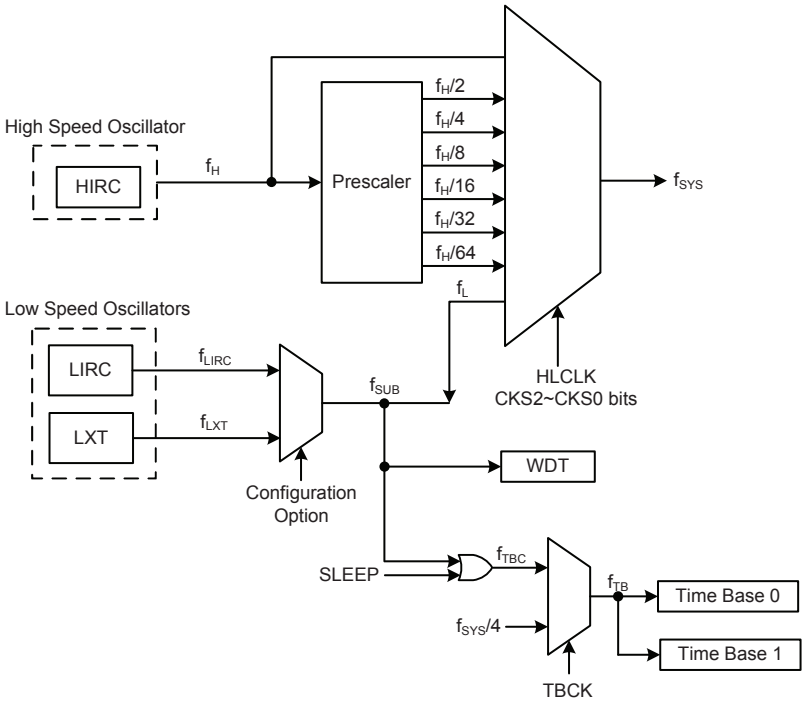
Present day applications require that their microcontrollers have high performance but often still demand that they consume as little power as possible, conflicting requirements that are especially true in battery powered portable applications. The fast clocks required for high performance will by their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

System Clocks

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

The main system clock, can come from either a high frequency, f_H, or low frequency, f_L, source, and is selected using the HLCLK bit and CKS2~CKS0 bits in the SMOD register. The high speed system clock is sourced from the HIRC oscillator. 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 is one additional internal clock for the peripheral circuits, the Time Base clock, f_{TBC}. f_{TBC} is sourced from the LIRC or LXT oscillators. The f_{TBC} clock is used as a source for the Time Base interrupt functions and for the TMs.



System Clock Configurations

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

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 modes are used when the microcontroller CPU is switched off to conserve power.

Operating Mode	Description				
	CPU	f_{SYS}	f_{SUB}	f_{TBC}	f_L
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
IDLE1 mode	Off	On	On	On	On
SLEEP0 mode	Off	Off	Off	Off	Off
SLEEP1 mode	Off	Off	On	Off	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 the high speed oscillator. This mode operates allowing the microcontroller to operate normally with a clock source will come from the high speed oscillator HIRC. The high speed oscillator will however first be divided by a ratio ranging from 1 to 64, the actual ratio being selected by the CKS2~CKS0 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 the low speed oscillator LIRC or LXT. 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 SLEEP0 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} clock will be stopped too, and the Watchdog Timer function is disabled.

SLEEP1 Mode

The SLEEP1 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} clocks will continue to operate if the Watchdog Timer function is enabled.

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 SMOD1 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 and TMs. In the IDLE0 Mode, the system oscillator will be stopped.

IDLE1 Mode

The IDLE1 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 SMOD1 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 and TMs. 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_{SUB}, will be on.

Control Register

The registers, SMOD and SMOD1, are 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	—	LTO	HTO	IDLEN	HLCLK
R/W	R/W	R/W	R/W	—	R	R	R/W	R/W
POR	0	0	0	—	0	0	1	1

Bit 7 ~ 5 **CKS2 ~ CKS0**: The system clock selection when HLCLK is "0"
000: f_L (f_{LIRC} or f_{LXT})
001: f_L (f_{LIRC} or f_{LXT})
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 the LIRC or LXT, a divided version of the high speed system oscillator can also be chosen as the system clock source.

Bit 4 Unimplemented, read as "0"

Bit 3 **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.

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.

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: $f_H/2$ ~ $f_H/64$ or f_L
1: f_H
This bit is used to select if the f_H clock or the $f_H/2$ ~ $f_H/64$ 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 $f_H/2$ ~ $f_H/64$ 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.

SMOD1 Register

Bit	7	6	5	4	3	2	1	0
Name	FSYSON	—	—	—	D3	LVRF	—	WRF
R/W	R/W	—	—	—	R/W	R/W	—	R/W
POR	0	—	—	—	0	x	—	0

"x" unknown

Bit 7 **FSYSON**: f_{SYS} Control in IDLE Mode
0: Disable
1: Enable
This bit is used to control whether the system clock is switched on or not in the IDLE Mode. If this bit is set to "0", the system clock will be switched off in the IDLE Mode. However, the system clock will be switched on in the IDLE Mode when the FSYSON bit is set to "1".

Bit 6~4 Unimplemented, read as 0

Bit 3 **D3**: Reserved bit

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

Bit 1 Unimplemented, read as 0

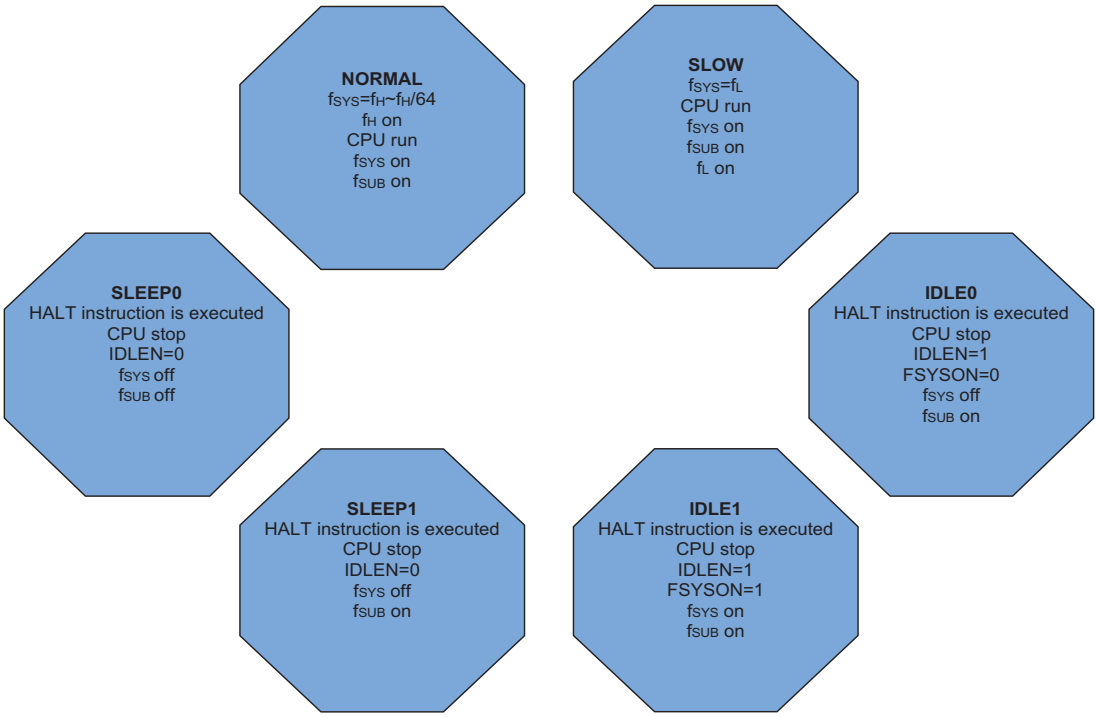
Bit 0 **WRF**: WDT Control register software reset flag
Described elsewhere

Operating Mode Switching

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

In simple terms, Mode Switching between the 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 SMOD1 register.

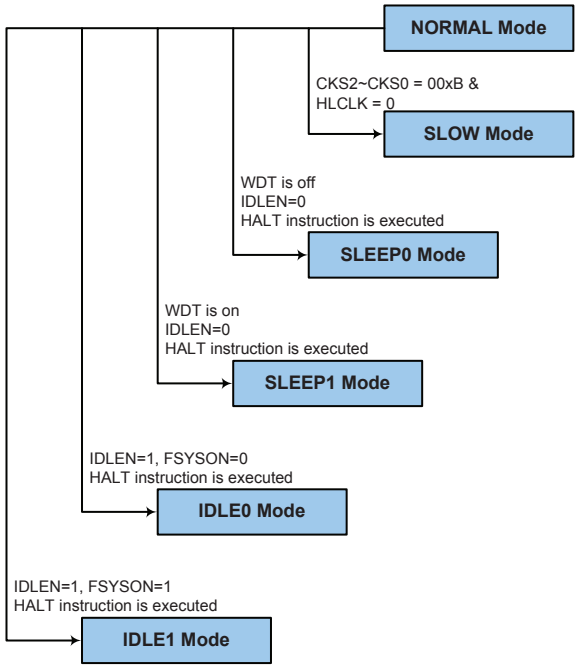
When the HLCLK bit switches to a low level, which implies that clock source is switched from the high speed clock source, f_H, to the clock source, $f_H/2$~$f_H/64$ or f_L. If the clock is from the f_L, the high speed clock source will stop running to conserve power. When this happens it must be noted 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.



NORMAL Mode to SLOW Mode Switching

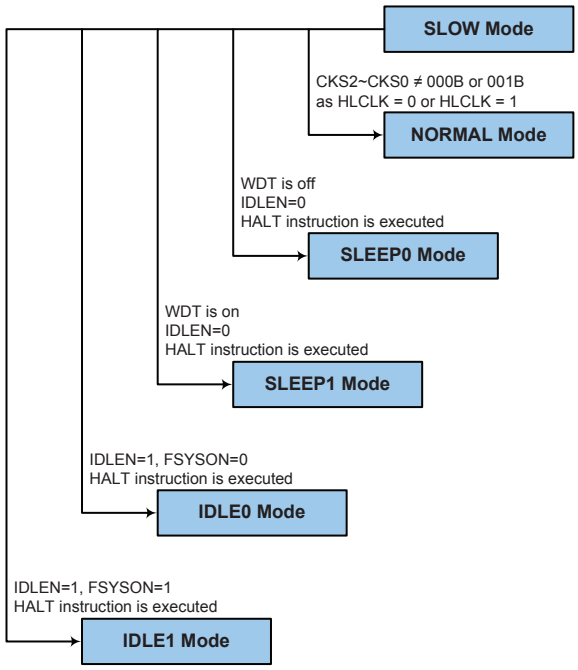
When running in the NORMAL Mode, which uses the high speed system oscillator, and therefore consumes more power, the system clock can switch to run in the SLOW Mode by setting the HLCLK bit to "0" and setting the CKS2~CKS0 bits to "000" or "001" in the SMOD register.This will then use the low speed system 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 LIRC or LXT oscillator and therefore requires this oscillator to be stable before full mode switching occurs. This is monitored using the LTO bit in the SMOD register.



SLOW Mode to NORMAL Mode Switching

In SLOW Mode the system uses LIRC or LXT 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.



Entering the SLEEP0 Mode

There is only one way for the devices 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 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.
- 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 devices 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 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 will remain with the clock source coming from the f_{LIRC} clock.
- The Data Memory contents and registers will maintain their present condition.
- The WDT will be cleared and resume counting if 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" instruction in the application program with the IDLEN bit in SMOD register equal to "1" and the FSYSON bit in SMOD1 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 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 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 IDLE1 Mode

There is only one way for the devices to enter the IDLE1 Mode and that is to execute the "HALT" instruction in the application program with the IDLEN bit in SMOD register equal to "1" and the FSYSON bit in SMOD1 register equal to "1". When this instruction is executed under the conditions described above, the following will occur:

- The system clock and Time Base 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.
- 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 devices 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. 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 devices will experience a full system reset, however, If these devices are woken up by a WDT overflow, a Watchdog Timer reset will be initiated. Although both of these wake-up methods will initiate a reset operation, the actual source of the wake-up can be determined by examining the TO and PDF flags. The PDF flag is cleared by a system power-up or executing the clear Watchdog Timer instructions and is set when executing the "HALT" instruction. The TO flag is set if a WDT time-out occurs, and causes a wake-up that only resets the Program Counter and Stack Pointer, the other flags remain in their original status.

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

Watchdog Timer

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

Watchdog Timer Clock Source

The Watchdog Timer clock source is provided by the internal f_{SUB} clock which is supplied by the LIRC or LXT oscillator. 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 LXT oscillator is supplied by an external 32.768kHz crystal. 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 WDT can be enabled/disabled using the WDTC register.

Watchdog Timer Control Register

A single register, WDTC, controls the required timeout period as well as the enable/disable operation. The WRF software reset flag will be indicated in the SMOD1 register. These registers control the overall operation of the Watchdog Timer.

WDTC Register

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

Bit 7~3 **WE4 ~ WE0**: WDT function software control
10101: WDT disable
01010: WDT enable
Other values: Reset MCU
When these bits are changed to any other values by the environmental noise to reset the microcontroller, the reset operation will be activated after 2~3 LIRC clock cycles and the WRF bit will be set to 1to indicate the reset source.

Bit 2~0 **WS2 ~ WS0**: WDT Time-out period selection
000: $2^8/f_{SUB}$
001: $2^9/f_{SUB}$
010: $2^{10}/f_{SUB}$
011: $2^{11}/f_{SUB}$ (default)
100: $2^{12}/f_{SUB}$
101: $2^{13}/f_{SUB}$
110: $2^{14}/f_{SUB}$
111: $2^{15}/f_{SUB}$
These three bits determine the division ratio of the Watchdog Timer sourece clock, which in turn determines the timeout period.

SMOD1 Register

Bit	7	6	5	4	3	2	1	0
Name	FSYSON	—	—	—	D3	LVRF	—	WRF
R/W	R/W	—	—	—	R/W	R/W	—	R/W
POR	0	—	—	—	0	x	—	0

"x" unknown

Bit 7 **FSYSON**: f_{SYS} Control in IDLE Mode
Described elsewhere

Bit 6~4 Unimplemented, read as 0

Bit 3 **D3**: Reserved bit

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

Bit 1 Unimplemented, read as 0

Bit 0 **WRF**: WDT Control register software reset flag
0: Not occur
1: Occurred
This bit is set to 1 by the WDT Control register software reset and cleared by the application program. Note that this bit can only be cleared to 0 by the application program.

Watchdog Timer Operation

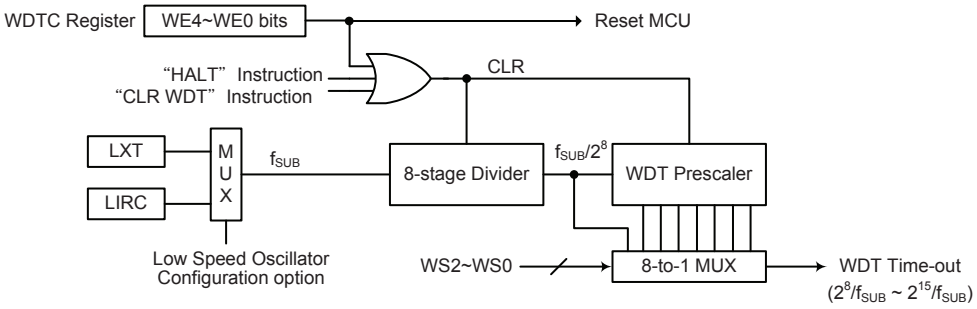
The Watchdog Timer operates by providing a device reset when its timer overflows. This means that in the application program and during normal operation the user has to strategically clear the Watchdog Timer before it overflows to prevent the Watchdog Timer from executing a reset. This is done using the clear watchdog instructions. If the program malfunctions for whatever reason, jumps to an unknown location, or enters an endless loop, the clear WDT instructions will not be executed in the correct manner, in which case the Watchdog Timer will overflow and reset the device. With regard to the Watchdog Timer enable/disable function, there are five bits, WE4~WE0, in the WDTC register to additional enable/disable and reset control of the Watchdog Timer.

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

Watchdog Timer Enable/Disable Control

Under normal program operation, a Watchdog Timer time-out will initialise a device reset and set the status bit TO. However, if the system is in the SLEEP or IDLE Mode, when a Watchdog Timer time-out occurs, the TO bit in the status register will be set and only the Program Counter and Stack Pointer will be reset. Four methods can be adopted to clear the contents of the Watchdog Timer. The first is a WDT reset, which means a value other than 01010B and 10101B is written into the WE4~WE0 bit locations, the second is an external hardware reset, which means a low level on the external reset pin, the third is using the Watchdog Timer software clear instructions and the fourth is via a HALT instruction. There is only one method of using software instruction to clear the Watchdog Timer. That is to use the single "CLR WDT" instruction to clear the WDT.

The maximum time-out period is when the 2^{15} division ratio is selected. As an example, with a 32kHz LIRC oscillator as its source clock, this will give a maximum watchdog period of around 1 second for the 2^{15} division ratio, and a minimum timeout of 7.8ms for the 2^8 division ration.



Watchdog Timer

Reset and Initialisation

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

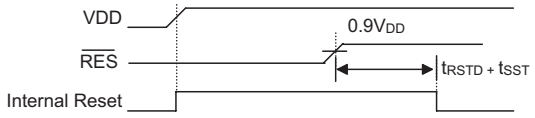
Another type of reset is when the Watchdog Timer overflows and resets the microcontroller. All types of reset operations result in different register conditions being setup. Another reset exists in the form of a Low Voltage Reset, LVR, where a full reset is implemented in situations where the power supply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

Reset Functions

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

Power-on Reset

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

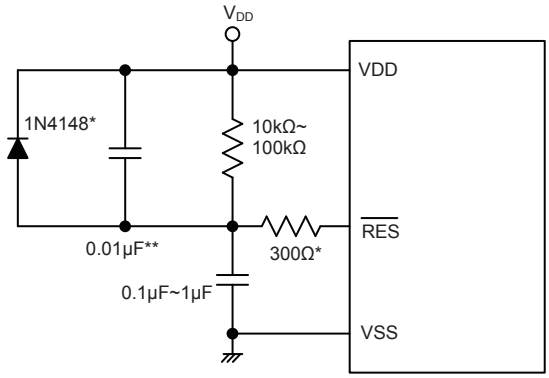


Power-On Reset Timing Chart

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

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

For most applications a resistor connected between V_{DD} and the $\overline{\text{RES}}$ pin and a capacitor connected between VSS and the $\overline{\text{RES}}$ pin will provide a suitable external reset circuit. Any wiring connected to the $\overline{\text{RES}}$ pin should be kept as short as possible to minimize any stray noise interference. For applications that operate within an environment where more noise is present the Enhanced Reset Circuit shown is recommended.



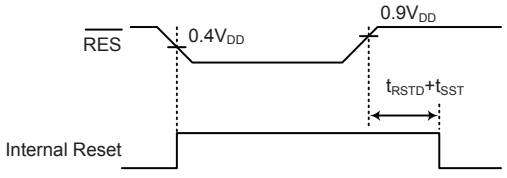
Note: "*" It is recommended that this component is added for added ESD protection

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

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

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

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



$\overline{\text{RES}}$ Reset Timing Chart

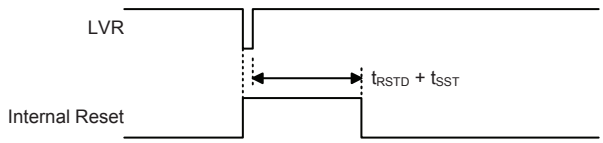
- **RSTC External Reset Register**

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

Bit 7 ~ 0 **RSTC7 ~ RSTC0**: PC2/$\overline{\text{RES}}$ selection
01010101: Configured as PC2 pin or other function
10101010: Configured as $\overline{\text{RES}}$ pin
Other Values: inhibit to use
All reset will reset this register as POR value except WDT time out Hardware warm reset.

Low Voltage Reset – LVR

The devices both contain a low voltage reset circuit in order to monitor the supply voltage of the device and provide an MCU reset should the value fall below a certain predefined level. The LVR function is always enabled during the normal and slow modes with a specific LVR voltage V_{LVR}. 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 devices internally and the LVRF bit in the SMOD1 register will also be set to1. 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 LVR characteristics. If the low voltage state does not exceed this value, the LVR will ignore the low supply voltage and will not perform a reset function. The actual V_{LVR} is 2.1V, the LVR will reset the device after 2~3 LIRC clock cycles. Note that the LVR function will be automatically disabled when the device enters the SLEEP/IDLE mode.



Low Voltage Reset Timing Chart

- **SMOD1 Register**

Bit	7	6	5	4	3	2	1	0
Name	FSYSON	—	—	—	D3	LVRF	—	WRF
R/W	R/W	—	—	—	R/W	R/W	—	R/W
POR	0	—	—	—	0	x	—	0

"x" unknown

Bit 7 **FSYSON**: f_{SYS} Control in IDLE Mode
Describe elsewhere

Bit 6~4 Unimplemented, read as 0

Bit 3 **D3**: Reserved bit

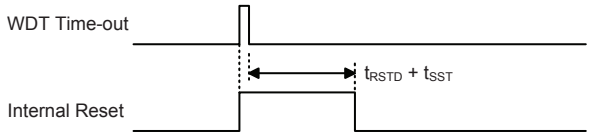
Bit 2 **LVRF**: LVR function reset flag
0: Not active
1: Active
This bit can be clear to "0", but can not be set to "1".

Bit 1 Unimplemented, read as 0

Bit 0 **WRF**: WDT Control register software reset flag
Describe elsewhere

Watchdog Time-out Reset during Normal Operation

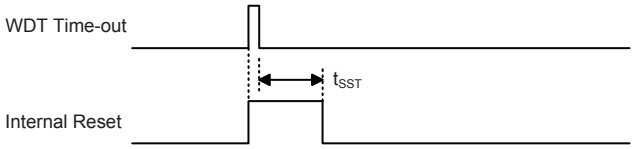
The Watchdog time-out Reset during normal operation is the same as an LVR reset except that the Watchdog time-out flag TO will be set to "1".



WDT Time-out Reset during Normal Operation Timing Chart

Watchdog Time-out Reset during SLEEP or IDLE Mode

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



WDT Time-out Reset during SLEEP or IDLE Timing Chart

Reset Initial Conditions

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

TO	PDF	RESET Conditions
0	0	Power-on reset
u	u	LVR reset during 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 Modules	Timer Modules will be turned off
Input/Output Ports	I/O ports will be setup as inputs
Stack Pointer	Stack Pointer will point to the top of the stack

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

Register	HT66F0042	HT66F0082	Reset (Power On)	WDT Time-out (Normal Operation)	$\overline{RES}$ Reset (Normal Operation)	$\overline{RES}$ Reset (HALT)	WDT Time-out (HALT)*
MP0	●	●	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
MP1	●	●	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
BP	●	●	---- ---0	---- ---0	---- ---0	---- ---0	---- ---u
ACC	●	●	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLP	●	●	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLH	●		-xxx xxxx	-uuu uuuu	-uuu uuuu	-uuu uuuu	-uuu uuuu
		●	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBHP	●		---- -xxx	---- -uuu	---- -uuu	---- -uuu	---- -uuu
		●	---- xxxx	---- uuuu	---- uuuu	---- uuuu	---- uuuu
STATUS	●	●	--00 xxxx	--1u uuuu	--uu uuuu	--01 uuuu	--11 uuuu
SMOD	●	●	000- 0011	000- 0011	000- 0011	000- 0011	uuu- uuuu
INTEG	●	●	---- 0000	---- 0000	---- 0000	---- 0000	---- uuuu
INTC0	●	●	-000 0000	-000 0000	-000 0000	-000 0000	-uuu uuuu
INTC1	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
INTC2	●		-000 -000	-000 -000	-000 -000	-000 -000	-uuu -uuu
		●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
MFI0	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
MFI1	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
MFI2	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PA	●	●	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAC	●	●	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAPU	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PAWU	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
WDTC	●	●	0101 0011	0101 0011	0101 0011	0101 0011	uuuu uuuu
TBC	●	●	0011 0111	0011 0111	0011 0111	0011 0111	uuuu uuuu
SMOD1	●	●	0--- 0x-0	0--- 0x-0	0--- 0x-0	0--- 0x-0	u--- uu-u
SCOMC	●	●	-000 ----	-000 ----	-000 ----	-000 ----	-uuu ----
SADOL (ADRFS=0)	●	●	xxxx ----	xxxx ----	xxxx ----	xxxx ----	uuuu ----
SADOL (ADRFS=1)	●	●	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
SADOH (ADRFS=0)	●	●	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu

Register	HT66F0042	HT66F0082	Reset (Power On)	WDT Time-out (Normal Operation)	$\overline{RES}$ Reset (Normal Operation)	$\overline{RES}$ Reset (HALT)	WDT Time-out (HALT)*
SADOH (ADRFS=1)	●	●	---- xxxx	---- xxxx	---- xxxx	---- xxxx	---- uuuu
SADC0	●	●	0000 -000	0000 -000	0000 -000	0000 -000	uuuu -uuu
SADC1	●	●	000- -000	000- -000	000- -000	000- -000	uuu- -uuu
SADC2	●	●	00-- 0000	00-- 0000	00-- 0000	00-- 0000	uu-- uuuu
RSTC	●	●	0101 0101	0101 0101	0101 0101	0101 0101	uuuu uuuu
PAS0	●	●	00-- 0000	00-- 0000	00-- 0000	00-- 0000	uu-- uuuu
PAS1	●	●	--00 0000	--00 0000	--00 0000	--00 0000	--uu uuuu
PBS0	●	●	--00 0000	--00 0000	--00 0000	--00 0000	--uu uuuu
PBS1	●	●	--00 00--	--00 00--	--00 00--	--00 00--	--uu uu--
PTM0C0	●	●	0000 0---	0000 0---	0000 0---	0000 0---	uuuu u---
PTM0C1	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM0DL	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM0DH	●	●	---- --00	---- --00	---- --00	---- --00	---- --uu
PTM0AL	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM0AH	●	●	---- --00	---- --00	---- --00	---- --00	---- --uu
PTM0RPL	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM0RPH	●	●	---- --00	---- --00	---- --00	---- --00	---- --uu
PTM1C0	●	●	0000 0---	0000 0---	0000 0---	0000 0---	uuuu u---
PTM1C1	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM1DL	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM1DH	●	●	---- --00	---- --00	---- --00	---- --00	---- --uu
PTM1AL	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM1AH	●	●	---- --00	---- --00	---- --00	---- --00	---- --uu
PTM1RPL	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM1RPH	●	●	---- --00	---- --00	---- --00	---- --00	---- --uu
PB	●	●	-111 1111	-111 1111	-111 1111	-111 1111	-uuu uuuu
PBC	●	●	-111 1111	-111 1111	-111 1111	-111 1111	-uuu uuuu
PBPU	●	●	-000 0000	-000 0000	-000 0000	-000 0000	-uuu uuuu
PC	●	●	-111 1111	-111 1111	-111 1111	-111 1111	-uuu uuuu
PCC	●	●	-111 1111	-111 1111	-111 1111	-111 1111	-uuu uuuu
PCPU	●	●	-000 0000	-000 0000	-000 0000	-000 0000	-uuu uuuu
EEA	●		---0 0000	---0 0000	---0 0000	---0 0000	---u uuuu
		●	--00 0000	--00 0000	--00 0000	--00 0000	--uu uuuu
EED	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PCS0	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PCS1	●		---- --00	---- --00	---- --00	---- --00	---- --uu
		●	--00 0000	--00 0000	--00 0000	--00 0000	--uu uuuu
IFS	●		---- 0000	---- 0000	---- 0000	---- 0000	---- uuuu
		●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
SLEDC0	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
SLEDC1	●		---- 0000	---- 0000	---- 0000	---- 0000	---- uuuu
		●	--00 0000	--00 0000	--00 0000	--00 0000	--uu uuuu
PTM2C0	●	●	0000 0---	0000 0---	0000 0---	0000 0---	uuuu u---
PTM2C1	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM2DL	●	●	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu

Register	HT66F0042	HT66F0082	Reset (Power On)	WDT Time-out (Normal Operation)	$\overline{RES}$ Reset (Normal Operation)	$\overline{RES}$ Reset (HALT)	WDT Time-out (HALT)*
PTM2DH	•	•	---- --00	---- --00	---- --00	---- --00	---- --uu
PTM2AL	•	•	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM2AH	•	•	---- --00	---- --00	---- --00	---- --00	---- --uu
PTM2RPL	•	•	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM2RPH	•	•	---- --00	---- --00	---- --00	---- --00	---- --uu
PTM3C0	•	•	0000 0---	0000 0---	0000 0---	0000 0---	uuuu u---
PTM3C1	•	•	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM3DL	•	•	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM3DH	•	•	---- --00	---- --00	---- --00	---- --00	---- --uu
PTM3AL	•	•	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM3AH	•	•	---- --00	---- --00	---- --00	---- --00	---- --uu
PTM3RPL	•	•	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM3RPH	•	•	---- --00	---- --00	---- --00	---- --00	---- --uu
CTM0C0	•	•	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
CTM0C1	•	•	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
CTM0DL	•	•	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
CTM0DH	•	•	---- --00	---- --00	---- --00	---- --00	---- --uu
CTM0AL	•	•	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
CTM0AH	•	•	---- --00	---- --00	---- --00	---- --00	---- --uu
CTM1C0	•	•	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
CTM1C1	•	•	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
CTM1DL	•	•	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
CTM1DH	•	•	---- --00	---- --00	---- --00	---- --00	---- --uu
CTM1AL	•	•	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
CTM1AH	•	•	---- --00	---- --00	---- --00	---- --00	---- --uu
SIMC0	•	•	111- 0000	111- 0000	111- 0000	111- 0000	uuu- uuuu
SIMC1	•	•	1000 0001	1000 0001	1000 0001	1000 0001	uuuu uuuu
SIMC2	•	•	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
SIMA	•	•	0000 000-	0000 000-	0000 000-	0000 000-	uuuu uuu-
SIMD	•	•	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
SIMTOC	•	•	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PD		•	---- 1111	---- 1111	---- 1111	---- 1111	---- uuuu
PDC		•	---- 1111	---- 1111	---- 1111	---- 1111	---- uuuu
PDPU		•	---- 0000	---- 0000	---- 0000	---- 0000	---- uuuu
PDS0		•	---- 0000	---- 0000	---- 0000	---- 0000	---- uuuu
USR		•	0000 1011	0000 1011	0000 1011	0000 1011	uuuu uuuu
UCR1		•	0000 00x0	0000 00x0	0000 00x0	0000 00x0	uuuu uuuu
UCR2		•	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
TXR_RXR		•	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
BRG		•	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
EEC	•	•	---- 0000	---- 0000	---- 0000	---- 0000	---- uuuu

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

Device	Register Name	Bit							
		7	6	5	4	3	2	1	0
HT66F0042 HT66F0082	PA	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0
	PAC	PAC7	PAC6	PAC5	PAC4	PAC3	PAC2	PAC1	PAC0
	PAPU	PAPU7	PAPU6	PAPU5	PAPU4	PAPU3	PAPU2	PAPU1	PAPU0
	PAWU	PAWU7	PAWU6	PAWU5	PAWU4	PAWU3	PAWU2	PAWU1	PAWU0
	PB	—	PB6	PB5	PB4	PB3	PB2	PB1	PB0
	PBC	—	PBC6	PBC5	PBC4	PBC3	PBC2	PBC1	PBC0
	PBPU	—	PBPU6	PBPU5	PBPU4	PBPU3	PBPU2	PBPU1	PBPU0
	PC	—	PC6	PC5	PC4	PC3	PC2	PC1	PC0
	PCC	—	PCC6	PCC5	PCC4	PCC3	PCC2	PCC1	PCC0
	PCPU	—	PCPU6	PCPU5	PCPU4	PCPU3	PCPU2	PCPU1	PCPU0
HT66F0082	PD	—	—	—	—	PD3	PD2	PD1	PD0
	PDC	—	—	—	—	PDC3	PDC2	PDC1	PDC0
	PDPU	—	—	—	—	PDPU3	PDPU2	PDPU1	PDPU0

I/O Register List

"—": Unimplemented, read as "0"

PAWUn: wake-up function control
0: Disable
1: Enable

PAn/PBn/PCn/PDn: I/O Data bit
0: Data 0
1: Data 1

PACn/PBCn/PCCn/PDCn: I/O type selection
0: Output
1: Input

PAPUn/PBPUn/PCPUn/PDPUn: Pull-high function control
0: Disable
1: Enable

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 register PAPU~PDPU, and are implemented using weak PMOS transistors.

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.

I/O Port Control Registers

Each I/O port has its own control register known as PAC~PDC, to control the input/output configuration. With these control registers, 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.

Pin-shared Functions

The flexibility of the microcontroller range is greatly enhanced by the use of pins that have more than one function. Limited numbers of pins can force serious design constraints on designers but by supplying pins with multi-functions, many of these difficulties can be overcome. The way in which the pin function of each pin is selected is different for each function and a priority order is established where more than one pin function is selected simultaneously. Each device includes output function selection registers PxSn and input function selection register IFS for selecting the desired functions of the multi-function pin-shared pins.

Pin-shared Registers

The limited number of supplied pins in a package can impose restrictions on the amount of functions a certain device can contain. However by allowing the same pins to share several different functions and providing a means of function selection, a wide range of different functions can be incorporated into even relatively small package sizes.

Register Name	Bit							
	7	6	5	4	3	2	1	0
PAS0	PAS07	PAS06	—	—	PAS03	PAS02	PAS01	PAS00
PAS1	—	—	PAS15	PAS14	PAS13	PAS12	PAS11	PAS10
PBS0	—	—	PBS05	PBS04	PBS03	PBS02	PBS01	PBS00
PBS1	—	—	PBS15	PBS14	PBS13	PBS12	—	—
PCS0	PCS07	PCS06	PCS05	PCS04	PCS03	PCS02	PCS01	PCS00
PCS1 (HT66F0042)	—	—	—	—	—	—	PCS11	PCS10
PCS1 (HT66F0082)	—	—	PCS15	PCS14	PCS13	PCS12	PCS11	PCS10
PDS0 (HT66F0082)	—	—	—	—	PDS03	PDS02	PDS01	PDS00
IFS (HT66F0042)	—	—	—	—	PTP2IPS	PTCK3PS	PTCK2PS	CTCK1PS
IFS (HT66F0082)	PTP3IPS	PTP0IPS	PTCK1PS	CTCK0PS	PTP2IPS	PTCK3PS	PTCK2PS	CTCK1PS

Pin-shared Function Selection Register List

- **PAS0 Register**

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

Bit 7~6 **PAS07~PAS06**: PA3 Pin-Shared Control Bits
00: PA3
01: CTP1
10: PA3
11: SCOM0

Bit 5~4 Unimplemented, read as "0"

Bit 3~2 **PAS03~PAS02**: PA1Pin-Shared Control Bits
00: PA1
01: PTP0
10: PA1
11: AN7

Bit 1~0 **PAS01~PAS00**: PA0 Pin-Shared Control Bits
00: PA0/PTP0I_0
01: PA0/PTP0I_0
10: PA0/PTP0I_0
11: AN6

- **PAS1 Register**

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

Bit 7~6 Unimplemented, read as "0"

Bit 5~4 **PAS15~PAS14**: PA6 Pin-Shared Control Bits
00: PA6
01: SCK/SCL
10: PA6
11: AN5

Bit 3~2 **PAS13~PAS12**: PA5 Pin-Shared Control Bits
00: PA5/PTCK2_0
01: SDO
10: PA5/PTCK2_0
11: AN4

Bit 1~0 **PAS11~PAS10**: PA4 Pin-Shared Control Bits
00: PA4
01: SDI/SDA
10: PA4
11: AN3

- **PBS0 Register**

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

Bit 7~6 Unimplemented, read as "0"

Bit 5~4 **PBS05~PBS04**: PB2 Pin-Shared Control Bits
00: PB2/PTCK3_0/INT1
01: PB2/PTCK3_0/INT1
10: PB2/PTCK3_0/INT1
11: AN2

Bit 3~2 **PBS03~PBS02**: PB1 Pin-Shared Control Bits
00: PB1/PTP3I_0
01: $\overline{SCS}$
10: VREFO
11: AN1

Bit 1~0 **PBS01~PBS00**: PB0 Pin-Shared Control Bits
00: PB0/PTP1I/INT0
01: PB0/PTP1I/INT0
10: VREF
11: AN0

- **PBS1 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	PBS15	PBS14	PBS13	PBS12	—	—
R/W	—	—	R/W	R/W	R/W	R/W	—	—
POR	—	—	0	0	0	0	—	—

Bit 7~6 Unimplemented, read as "0"

Bit 5~4 **PBS15~PBS14**: PB6 Pin-Shared Control Bits
00: PB6
01: PTP2
10: PB6
11: SCOM2

Bit 3~2 **PBS13~PBS12**: PB5 Pin-Shared Control Bits
00: PB5
01: PTP3
10: PB5
11: SCOM3

Bit 1~0 Unimplemented, read as "0"

- **PCS0 Register**

Bit	7	6	5	4	3	2	1	0
Name	PCS07	PCS06	PCS05	PCS04	PCS03	PCS02	PCS01	PCS00
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 **PCS07~PCS06**: PC3 Pin-Shared Control Bits
00: PC3/PTP2I_1
01: PC3/PTP2I_1
10: PC3/PTP2I_1
11: SCOM2

Bit 5~4 **PCS05~PCS04**: PC2 Pin-Shared Control Bits
00: PC2
01: PC2
10: PC2
11: CTP0
Note: these bits are valid when RSTC[7:0]=01010101

Bit 3~2 **PCS03~PCS02**: PC1 Pin-Shared Control Bits
00: PC1/PTCK0
01: PC1/PTCK0
10: PC1/PTCK0
11: SCOM1

Bit 1~0 **PCS01~PCS00**: PC0 Pin-Shared Control Bits
00: PC0
01: PC0
10: PC0
11: PTP1

- **PCS1 Register (HT66F0042)**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	PCS11	PCS10
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2 Unimplemented, read as "0"

Bit 1~0 **PCS11~PCS10**: PC4 Pin-Shared Control Bits
00: PC4/CTCK1_1
01: PC4/CTCK1_1
10: PC4/CTCK1_1
11: SCOM3

- **PCS1 Register (HT66F0082)**

Bit	7	6	5	4	3	2	1	0
Name	—	—	PCS15	PCS14	PCS13	PCS12	PCS11	PCS10
R/W	—	—	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	—	0	0	0	0	0	0

Bit 7~6 Unimplemented, read as "0"

Bit 5~4 **PCS15~PCS14**: PC6 Pin-Shared Control Bits
00: PC6/PTCK2_1
01: TX
10: PC6/PTCK2_1
11: PC6/PTCK2_1

Bit 3~2 **PCS13~PCS12**: PC5 Pin-Shared Control Bits
00: PC5/PTCK3_1
01: RX
10: PC5/PTCK3_1
11: PC5/PTCK3_1

Bit 1~0 **PCS11~PCS10**: PC4 Pin-Shared Control Bits
00: PC4/CTCK1_1
01: PC4/CTCK1_1
10: PC4/CTCK1_1
11: SCOM3

- **PDS0 Register (HT66F0082)**

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

Bit 7~4 Unimplemented, read as "0"

Bit 3~2 **PDS03~PDS02**: PD1 Pin-Shared Control Bits
00: PD1/PTP0I_1
01: PD1/PTP0I_1
10: PD1/PTP0I_1
11: SCOM1

Bit 1~0 **PDS01~PDS00**: PD0 Pin-Shared Control Bits
00: PD0/CTCK0_1
01: PD0/CTCK0_1
10: PD0/CTCK0_1
11: SCOM0

- **IFS Register (HT66F0042)**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	PTP2IPS	PTCK3PS	PTCK2PS	CTCK1PS
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0

Bit 7~4 Unimplemented, read as "0"

Bit 3 **PTP2IPS**: PTP2I Pin Remapping Control Bit
0: PTP2I_0 on PA7(default)
1: PTP2I_1 on PC3

Bit 2 **PTCK3PS**: PTCK3 Pin Remapping Control Bit
0: PTCK3_0 on PB2(default)
1: PTCK3_1 on PC5

Bit 1 **PTCK2PS**: PTCK2 Pin Remapping Control Bit
0: PTCK2_0 on PA5(default)
1: PTCK2_1 on PC6

Bit 0 **CTCK1PS**: CTCK1 Pin Remapping Control Bit
0: CTCK1_0 on PB3(default)
1: CTCK1_1 on PC4

- **IFS Register (HT66F0082)**

Bit	7	6	5	4	3	2	1	0
Name	PTP3IPS	PTP0IPS	PTCK1PS	CTCK0PS	PTP2IPS	PTCK3PS	PTCK2PS	CTCK1PS
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 **PTP3IPS**: PTP3I Pin Remapping Control Bit
0: PTP3I_0 on PB1(default)
1: PTP3I_1 on PD2

Bit 6 **PTP0IPS**: PTP0I Pin Remapping Control Bit
0: PTP0I_0 on PA0(default)
1: PTP0I_1 on PD1

Bit 5 **PTCK1PS**: PTCK1 Pin Remapping Control Bit
0: PTCK1_0 on PB4(default)
1: PTCK1_1 on PD3

Bit 4 **CTCK0PS**: CTCK0 Pin Remapping Control Bit
0: CTCK0_0 on PA2(default)
1: CTCK0_1 on PD0

Bit 3 **PTP2IPS**: PTP2I Pin Remapping Control Bit
0: PTP2I_0 on PA7(default)
1: PTP2I_1 on PC3

Bit 2 **PTCK3PS**: PTCK3 Pin Remapping Control Bit
0: PTCK3_0 on PB2(default)
1: PTCK3_1 on PC5

Bit 1 **PTCK2PS**: PTCK2 Pin Remapping Control Bit
0: PTCK2_0 on PA5(default)
1: PTCK2_1 on PC6

Bit 0 **CTCK1PS**: CTCK1 Pin Remapping Control Bit
0: CTCK1_0 on PB3(default)
1: CTCK1_1 on PC4

I/O Port Source Current Control

The devices support different source current driving capability for each I/O port. With the corresponding selection register, SLEDC0 and SLEDC1, each I/O port can support four levels of the source current driving capability. Users should refer to the D.C. characteristics section to select the desired source current for different applications.

Register Name	Bit							
	7	6	5	4	3	2	1	0
SLEDC0	PBPS3	PBPS2	PBPS1	PBPS0	PAPS3	PAPS2	PAPS1	PAPS0
SLEDC1 (HT66F0042)	—	—	—	—	PCPS3	PCPS2	PCPS1	PCPS0
SLEDC1 (HT66F0082)	—	—	PDPS1	PDPS0	PCPS3	PCPS2	PCPS1	PCPS0

I/O Port Source Current Control Registers List

SLEDC0 Register

Bit	7	6	5	4	3	2	1	0
Name	PBPS3	PBPS2	PBPS1	PBPS0	PAPS3	PAPS2	PAPS1	PAPS0
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 **PBPS3~PBP2**: PB6~PB4 source current selection
00: Source current = Level 0 (default)
01: Source current = Level 1
10: Source current = Level 2
11: Source current = Level 3 (max.)

Bit 5~4 **PBPS1~PBP0**: PB3~PB0 source current selection
00: Source current = Level 0 (min.)
01: Source current = Level 1
10: Source current = Level 2
11: Source current = Level 3 (max.)

Bit 3~2 **PAPS3~PAP2**: PA7~PA4 source current selection
00: Source current = Level 0 (min.)
01: Source current = Level 1
10: Source current = Level 2
11: Source current = Level 3 (max.)

Bit 1~0 **PAPS1~PAP0**: PA3~PA0 source current selection
00: Source current = Level 0 (min.)
01: Source current = Level 1
10: Source current = Level 2
11: Source current = Level 3 (max.)

Refer to the D.C. Characteristics table for the source current value of each level.

SLEDC1 Register (HT66F0042)

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	PCPS3	PCPS2	PCPS1	PCPS0
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	1	0	1

Bit 7~4 Unimplemented, read as "0"

Bit 3~2 **PCPS3~PCPS2**: PC6~PC4 source current selection
00: Source current = Level 0 (min.)
01: Source current = Level 1
10: Source current = Level 2
11: Source current = Level 3 (max.)

Bit 1~0 **PCPS1~PCPS0**: PC3~PC0 source current selection
00: Source current = Level 0 (min.)
01: Source current = Level 1
10: Source current = Level 2
11: Source current = Level 3 (max.)

Refer to the D.C. Characteristics table for the source current value of each level.

SLEDC1 Register (HT66F0082)

Bit	7	6	5	4	3	2	1	0
Name	—	—	PDPS1	PDPS0	PCPS3	PCPS2	PCPS1	PCPS0
R/W	—	—	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	—	0	1	0	1	0	1

Bit 7~6 Unimplemented, read as "0"

Bit 5~4 **PDPS1~PDPS0**: PD3~PD0 source current selection
00: Source current = Level 0 (min.)
01: Source current = Level 1
10: Source current = Level 2
11: Source current = Level 3 (max.)

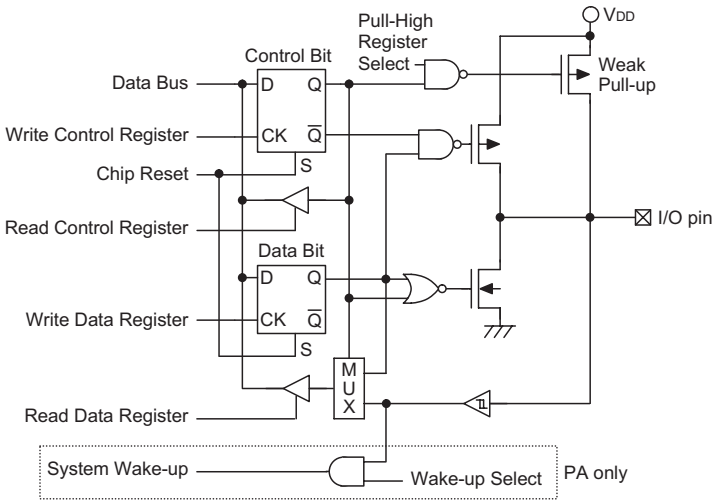
Bit 3~2 **PCPS3~PCPS2**: PC6~PC4 source current selection
00: Source current = Level 0 (min.)
01: Source current = Level 1
10: Source current = Level 2
11: Source current = Level 3 (max.)

Bit 1~0 **PCPS1~PCPS0**: PC3~PC0 source current selection
00: Source current = Level 0 (min.)
01: Source current = Level 1
10: Source current = Level 2
11: Source current = Level 3 (max.)

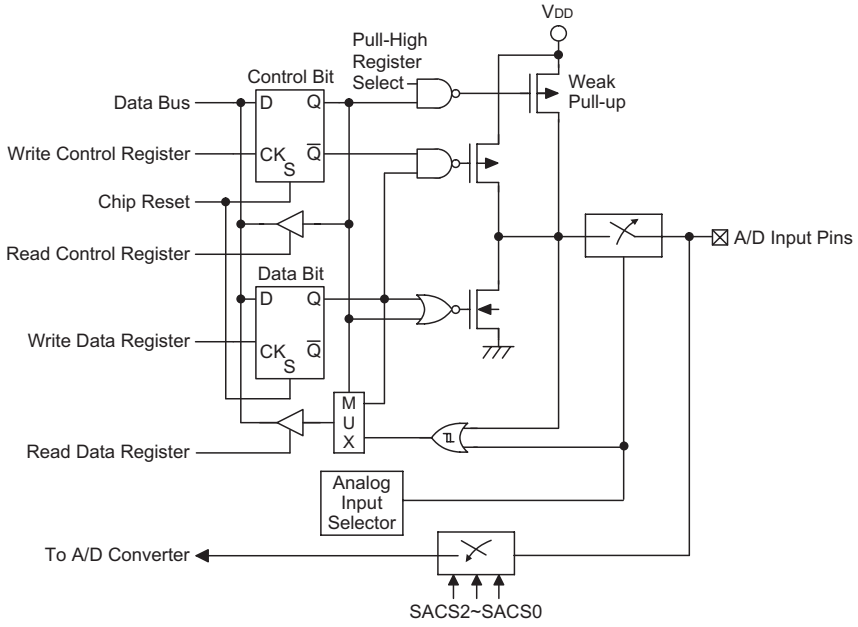
Refer to the D.C. Characteristics table for the source current value of each level.

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.



Generic Input/Output Structure



A/D Input/Output Structure

Programming Considerations

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

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

Timer Modules

One of the most fundamental functions in any microcontroller device is the ability to control and measure time. To implement time related functions the devices include 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 two individual interrupts. The addition of input and output pins for each TM ensures that users are provided with timing units with a wide and flexible range of features.

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

Introduction

Each of the devices contains sixTMs. Each individual TM can be categorised as a certain type, namely Compact Type TM or Periodic Type TM. Although similar in nature, the different TM types vary in their feature complexity. The common features to the Compact and Periodic TMs will be described in this section and the detailed operation will be described in corresponding sections. The main features and differences between the two types of TMs are summarised in the accompanying table.

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

TM Function Summary

CTM	PTM
10-bit CTM0 10-bit CTM1	10-bit PTM0 10-bit PTM1 10-bit PTM2 10-bit PTM3

TM Type Reference

TM Operation

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

TM Clock Source

The clock source which drives the main counter in each TM can originate from various sources. The selection of the required clock source is implemented using the xTnCK2~xTnCK0 bits in the xTMnC0 control registers. The clock source can be a ratio of either the system clock f_{SYS} or the internal high clock f_H, the f_{TBC} clock source or the external xTCKn pin. The xTCKn pin clock source is used to allow an external signal to drive the TM as an external clock source or for event counting.

TM Interrupt

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

TM External Pins

Each of the TMs, irrespective of what type, has two TM input pins, with the label xTCKn and xTPnI. The TM input pin xTCKn, is essentially a clock source for the TM and is selected using the xTnCK2~xTnCK0 bits in the xTMnC0 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 xTnCK2~xTnCK0 bits and the pin-shared register. The TM input pin can be chosen to have either a rising or falling active edge.

The other PTM input pin, PTPnI, is the capture input pin whose active edge can be a rising edge, a falling edge or both rising and falling edges and the active edge transition type is selected using the PTnIO1 and PTnIO0 bits in the PTMnC1 register.

The TMs each have one output pin with the label xTPn. When the TM is in the Compare Match Output Mode, the pin can be controlled by the TM to switch to a high or low level or to toggle when a compare match situation occurs. The external xTPn 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 selected and setup using registers. The bits in one of the pin-shared registers determines if its associated pin is to be used as an external TM output pin or if it is to have another function.

Some TM input pin names have a "_0" or "_1" suffix. Pin names that include a "_1" suffix indicate that the TM input has another remapping pin. The pin remapping function is controlled using the bits in the IFS registers.

Device	Pins	CTM0	CTM1
HT66F0042	Input	CTCK0_0	CTCK1_0/CTCK1_1
	Output	CTP0	CTP1
HT66F0082	Input	CTCK0_0/CTCK0_1	CTCK1_0/CTCK1_1
	Output	CTP0	CTP1

CTMn Input/Output Pins

Device	Pins	PTM0	PTM1	PTM2	PTM3
HT66F0042	Input	PTCK0 PTP0I_0	PTCK1_0 PTP1I	PTCK2_0/PTCK2_1 PTP2I_0/PTP2I_1	PTCK3_0/PTCK3_1 PTP3I_0
	Output	PTP0	PTP1	PTP2	PTP3
HT66F0082	Input	PTCK0 PTP0I_0/PTP0I_1	PTCK1_0/PTCK1_1 PTP1I	PTCK2_0/PTCK2_1 PTP2I_0/PTP2I_1	PTCK3_0/PTCK3_1 PTP3I_0/PTP3I_1
	Output	PTP0	PTP1	PTP2	PTP3

PTMn Input/Output Pins

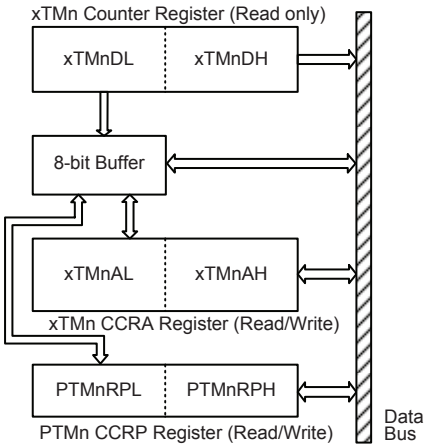
TM Input/Output Pin Control Register

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. Configuring the selection bits correctly will setup the corresponding pin as a TM input/output. The details of the pin-shared function selection are described in the pin-shared function section.

Programming Considerations

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

As the CCRA and CCRP registers are implemented in the way shown in the following diagram and accessing the register is carried out CCRP low byte register using the following access procedures. Accessing the CCRA or CCRP low byte register without following these access procedures will result in unpredictable values.



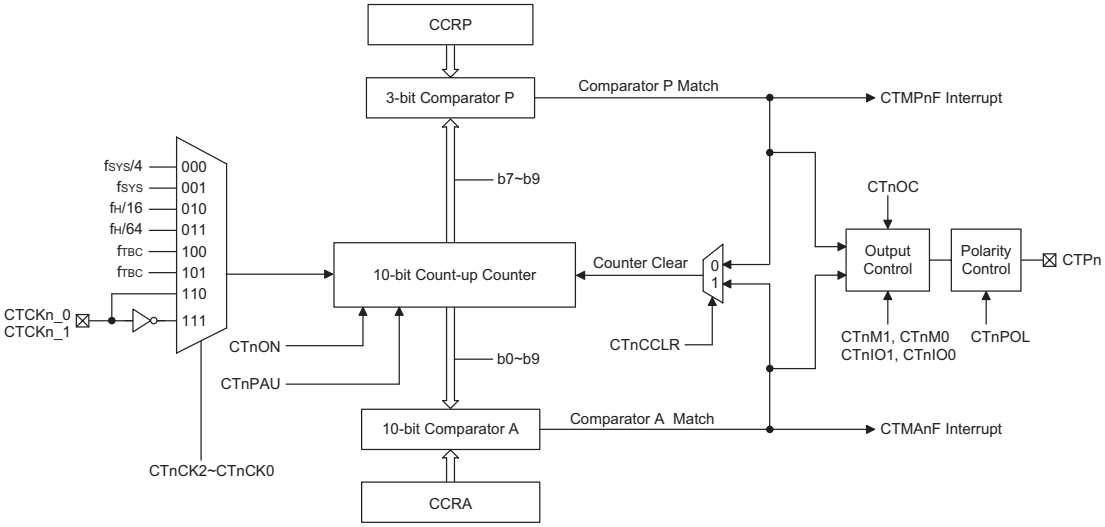
The following steps show the read and write procedures:

- Writing Data to CCRA or PTM CCRP
  - Step 1. Write data to Low Byte CTMnAL, PTMnAL or PTMnRPL
    – note that here data is only written to the 8-bit buffer.
  - Step 2. Write data to High Byte CTMnAH, PTMnAH or PTMnRPH
    – here data is written directly to the high byte registers and simultaneously data is latched from the 8-bit buffer to the Low Byte registers.
- Reading Data from the Counter Registers and CCRA or PTMn CCRP
  - Step 1. Read data from the High Byte CTMnDH, CTMnAH,PTMnDH, PTMnAH or PTMnRPH
    – 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 CTMnDL, CTMnAL, PTMnDL, PTMnAL or PTMnRPL
    – this step reads data from the 8-bit buffer.

Compact Type TM – CTM

Although the simplest form of the three 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 external output pin.

Device	Name	TM Input Pin	TM Output Pin
HT66F0042	10-bit CTM0 10-bit CTM1	CTCK0_0 CTCK1_0/CTCK1_1	CTP0 CTP1
HT66F0082	10-bit CTM0 10-bit CTM1	CTCK0_0/CTCK0_1 CTCK1_0/CTCK1_1	CTP0 CTP1



Compact Type TM Block Diagram (n=0~1)

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 CTnON bit from low to high. The counter will also be cleared automatically by a counter overflow or a compare match with one of its associated comparators. When these conditions occur, a 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 output pins. All operating setup conditions are selected using relevant internal registers.

Compact Type TM Register Description

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

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

Compact TM Register List (n=0~1)

CTMnC0 Register

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

Bit 7 **CTnPAU**: CTMn Counter Pause Control
0: Run
1: Pause

The counter can be paused by setting this bit high. Clearing the bit to zero restores normal counter operation. When in a Pause condition the 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 **CTnCK2 ~ CTnCK0**: Select CTMn Counter clock
000: $f_{SYS}/4$
001: f_{SYS}
010: $f_H/16$
011: $f_H/64$
100: f_{TBC}
101: f_{TBC}
110: CTCKn_0/CTCKn_1 rising edge clock
111: CTCKn_0/CTCKn_1 falling edge clock

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

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

This bit controls the overall on/off function of the CTM. Setting the bit high enables the counter to run, clearing the bit disables the CTM. Clearing this bit to zero will stop the counter from counting and turn off the CTM which will reduce its power consumption. When the bit changes state from low to high the internal counter value will be reset to zero, however when the bit changes from high to low, the internal counter will retain its residual value. If the CTM is in the Compare Match Output Mode or the PWM output Mode then the CTM output pin will be reset to its initial condition, as specified by the CTnOC bit, when the CTnON bit changes from low to high.

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

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

CTMnC1 Register

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

Bit7 ~ 6 **CTnM1 ~ CTnM0**: Select CTMn operating mode
00: Compare match output mode
01: Undefined
10: PWM output mode
11: Timer/Counter Mode

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

Bit5 ~ 4 **CTnIO1 ~ CTnIO0**: Select CTMn function
Compare match output mode
00: No change
01: Output low
10: Output high
11: Toggle output
PWM output mode
00: PWM output inactive state
01: PWM output active state
10: PWM output
11: Undefined
Timer/counter Mode
Unused

These two bits are used to determine how the CTM output pin changes state when a certain condition is reached. The function that these bits select depends upon in which mode the CTM is running. In the Compare Match Output Mode, the CTnIO1 and CTnIO0 bits determine how the CTM output pin changes state when a compare match occurs from the Comparator A. The CTM output pin can be setup to switch high, switch low or to toggle its present state when a compare match occurs from the Comparator A. When the bits are both zero, then no change will take place on the output. The initial value of the CTM output pin should be setup using the CTnOC bit in the CTMnC1 register. Note that the output level requested by the CTnIO1 and CTnIO0 bits must be different from the initial value setup using the CTnOC bit

otherwise no change will occur on the CTM output pin when a compare match occurs. After the CTM output pin changes state it can be reset to its initial level by changing the level of the CTnON bit from low to high. In the PWM Output Mode, the CTnIO1 and CTnIO0 bits determine how the CTM output pin changes state when a certain compare match condition occurs. The PWM output function is modified by changing these two bits. It is necessary to only change the values of the CTnIO1 and CTnIO0 bits only after the CTMn has been switched off. Unpredictable PWM outputs will occur if the CTnIO1 and CTnIO0 bits are changed when the CTM is running.

Bit3 **CTnOC**: CTPn output control

Compare match output mode
0: Initial low
1: Initial high

PWM output mode
0: Active low
1: Active high

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

Bit2 **CTnPOL**: CTPn output polarity control
0: Non-invert
1: Invert

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

Bit1 **CTnDPX**: CTMn PWM period/duty control
0: CCRP - period; CCRA - duty
1: CCRP - duty; CCRA - period

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

Bit0 **CTnCCLR**: Select CTMn counter clear condition
0: CTMn comparator P match
1: CTMn comparator A match

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

CTMnDL Register

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

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

CTMnDH Register

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

Bit 7 ~ 2 Unimplemented, read as "0"

Bit 1 ~ 0 **D9~D8**: CTMn Counter High Byte Register bit 1 ~ bit 0
CTMn 10-bit Counter bit 9 ~ bit 8

CTMnAL Register

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

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

CTMnAH Register

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

Bit 7 ~ 2 Unimplemented, read as "0"

Bit 1 ~ 0 **D9~D8**: CTMn CCRA High Byte Register bit 1 ~ bit 0
CTMn 10-bit CCRA bit 9 ~ bit 8

Compact Type TM Operating Modes

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

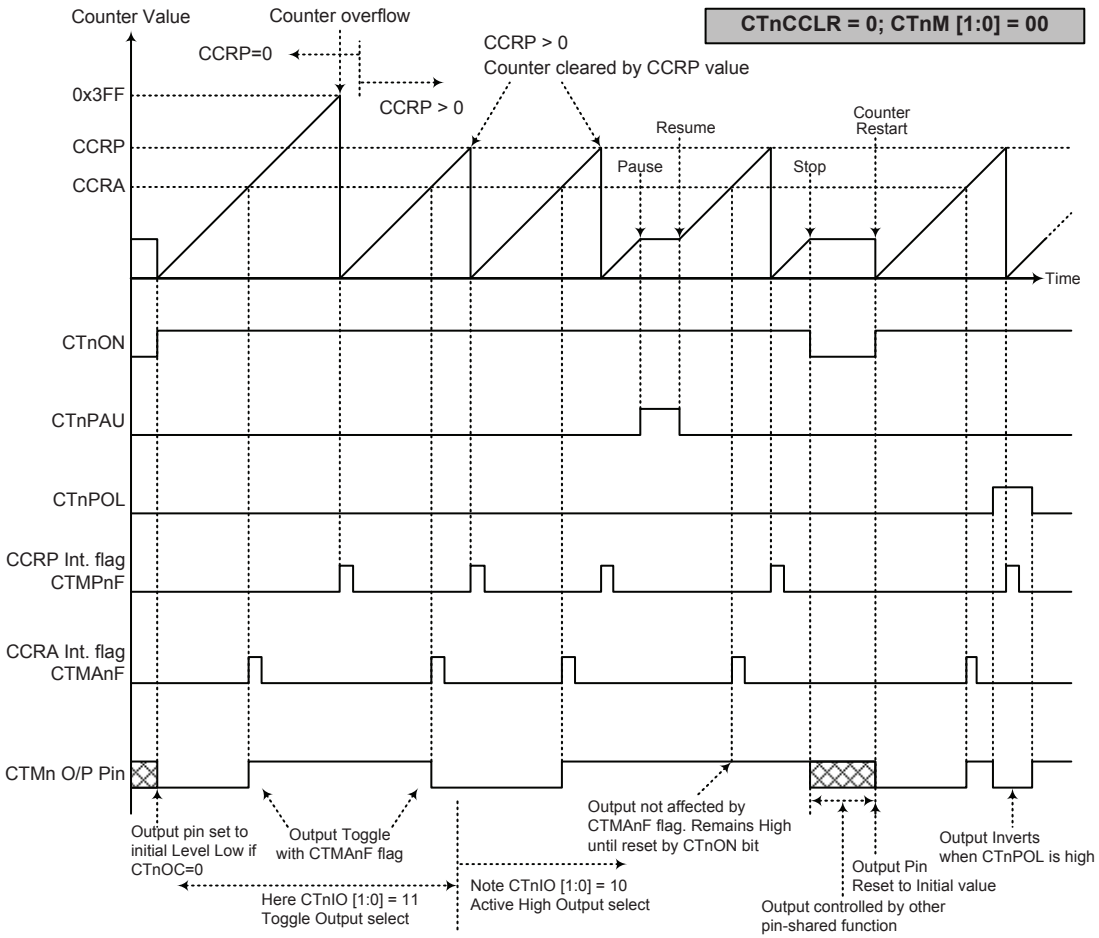
Compare Match Output Mode

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

If the CTnCCLR bit in the CTMnC1 register is high then the counter will be cleared when a compare match occurs from Comparator A. However, here only the CTMAnF interrupt request flag will be generated even if the value of the CCRP bits is less than that of the CCRA registers. Therefore when CTnCCLR is high no CTMPnF interrupt request flag will be generated.

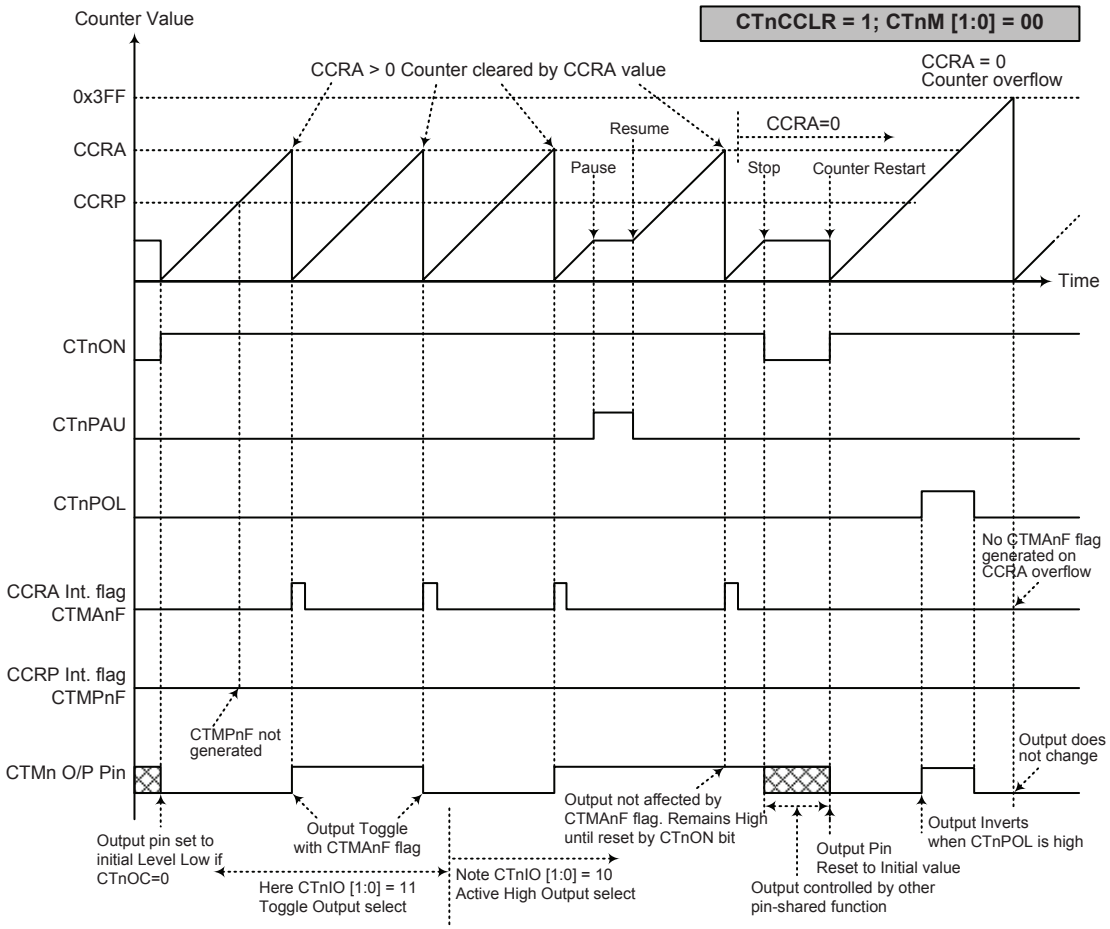
If the CCRA bits are all zero, the counter will overflow when its reaches its maximum 10-bit, 3FF Hex, value, however here the CTMAnF interrupt request flag will not be generated.

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



Compare Match Output Mode – CTnCCLR = 0 (n=0~1)

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



Compare Match Output Mode – CTnCCLR = 1 (n=0~1)

Note: 1. With CTnCCLR = 1, a Comparator A match will clear the counter
2. The CTM output pin controlled only by the CTMAnF flag
3. The output pin reset to initial state by a CTnON rising edge
4. The CTMPnF flag is not generated when CTnCCLR = 1

Timer/Counter Mode

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

PWM Output Mode

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

As both the period and duty cycle of the PWM waveform can be controlled, the choice of generated waveform is extremely flexible. In the PWM Output Mode, the CTnCCLR bit has no effect on the PWM operation. Both of the CCRA and CCRP registers are used to generate the PWM waveform, one register is used to clear the internal counter and thus control the PWM waveform frequency, while the other one is used to control the duty cycle. Which register is used to control either frequency or duty cycle is determined using the CTnDPX bit in the CTMnC1 register. The PWM waveform frequency and duty cycle can therefore be controlled by the values in the CCRA and CCRP registers.

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

- **CTM, PWM Mode, Edge-aligned Mode, CTnDPX=0**

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

If f_{SYS} = 16MHz, CTM clock source is $f_{SYS}/4$, CCRP = 100b, CCRA =128,

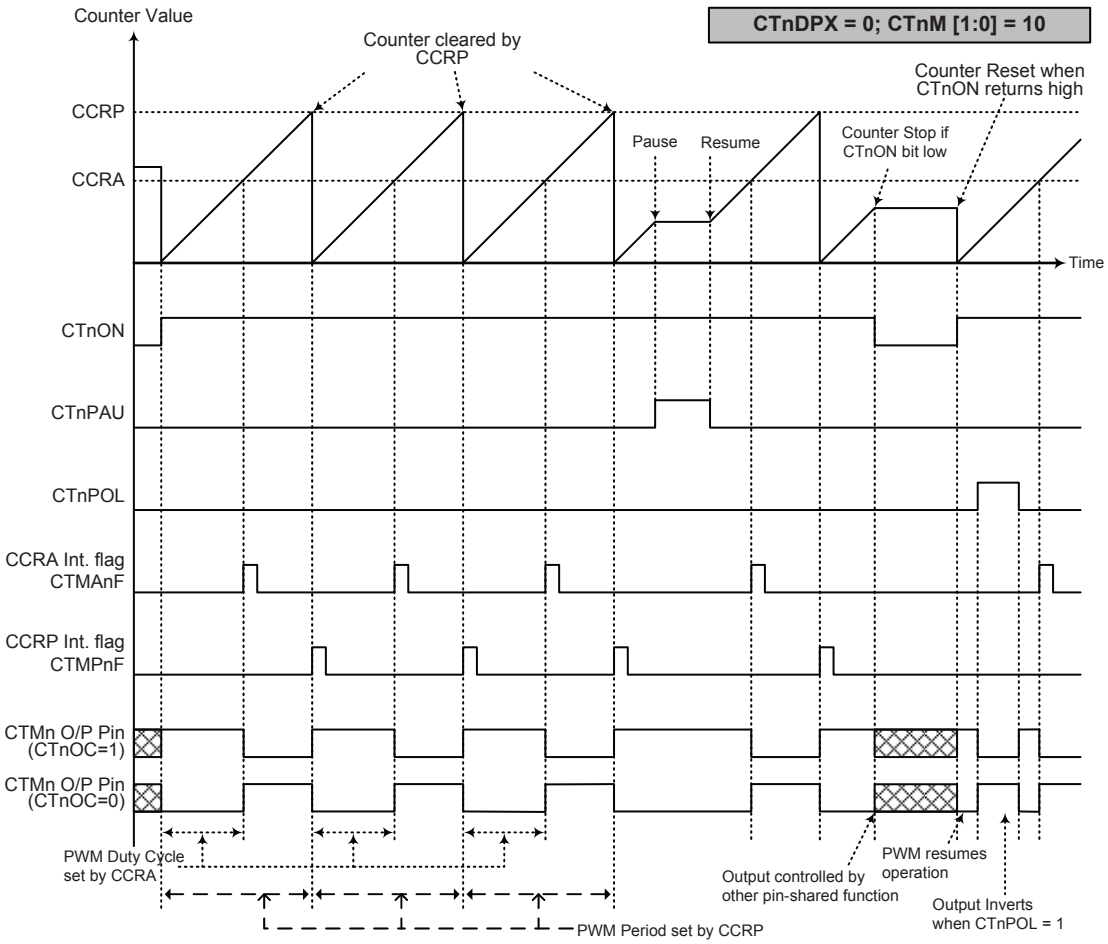
The CTM PWM output frequency = ($f_{SYS}/4$) / 512 = $f_{SYS}/2048$ = 7.8125 kHz, duty = 128/512 = 25%.

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

- **CTM, PWM Mode, Edge-aligned Mode, CTnDPX=1**

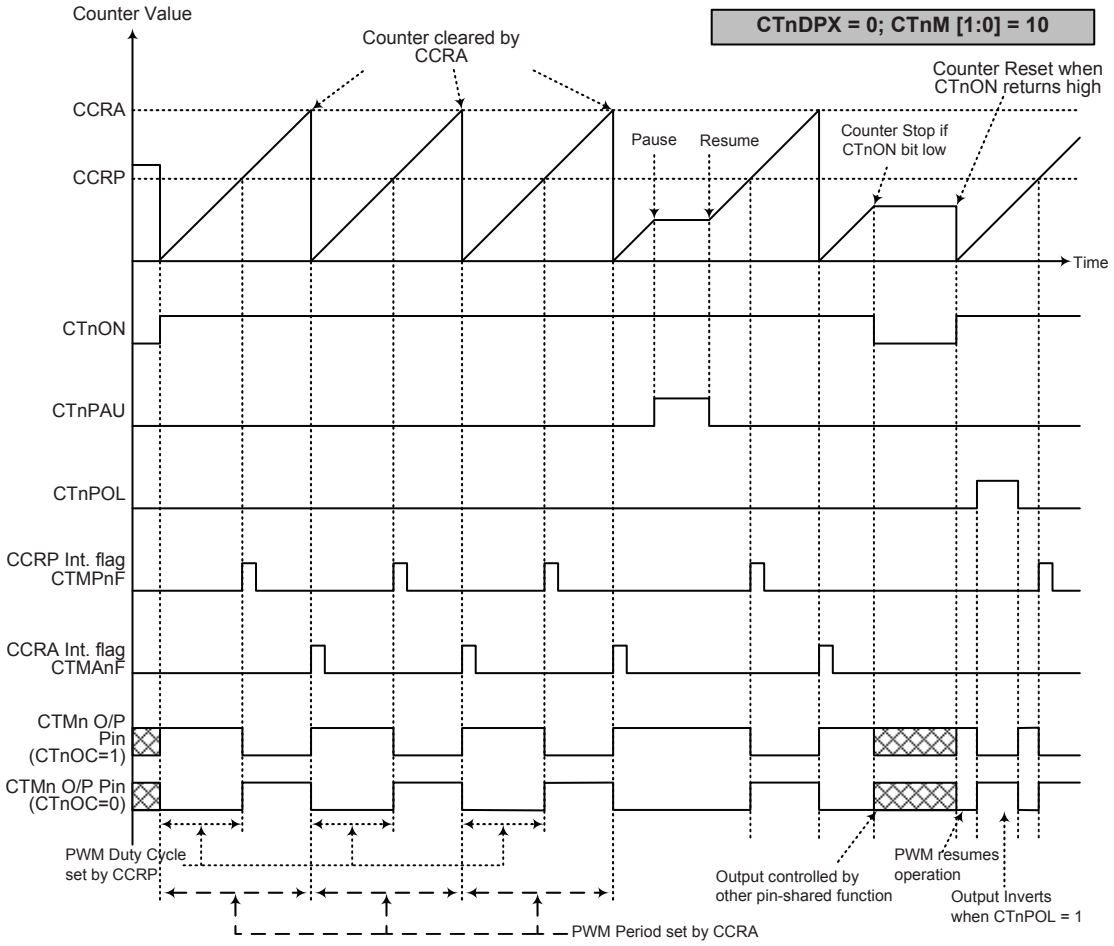
CCRP	001b	010b	011b	100b	101b	110b	111b	000b
Period	CCRA							
Duty	128	256	384	512	640	768	896	1024

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



PWM Mode – CTnDPX = 0 (n=0~1)

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



PWM Mode – CTnDPX = 1 (n=0~1)

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

Periodic Type TM – PTM

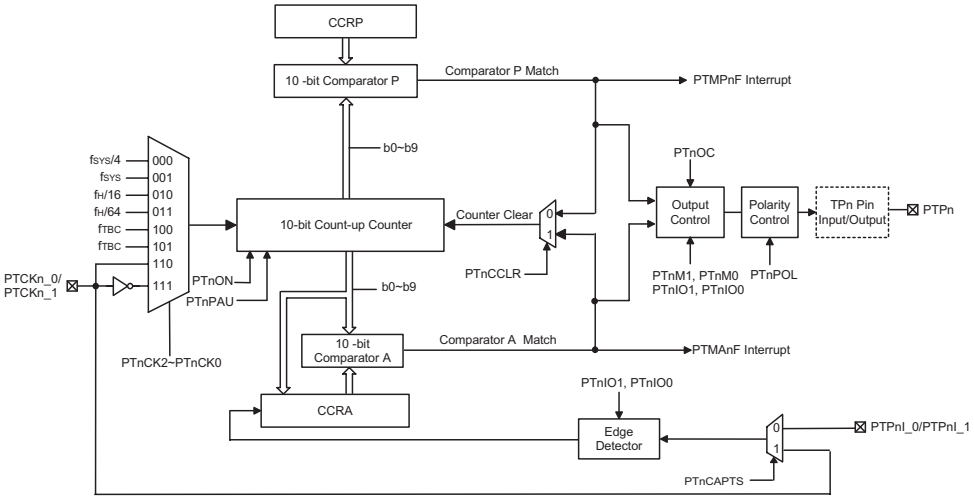
The Periodic Type TM contains five operating modes, which are Compare Match Output, Timer/ Event Counter, Capture Input, Single Pulse Output and PWM Output modes. The Periodic TM can also be controlled with two external input pins and can drive one external output pin.

Device	Name	TM Input Pin	TM Output Pin
HT66F0042	10-bit PTM0 10-bit PTM1 10-bit PTM2 10-bit PTM3	PTCK0, PTP0I_0 PTCK1_0, PTP1I PTCK2_0/PTCK2_1, PTP2I_0/PTP2I_1 PTCK3_0/PTCK3_1, PTP3I_0	PTP0 PTP1 PTP2 PTP3
HT66F0082	10-bit PTM0 10-bit PTM1 10-bit PTM2 10-bit PTM3	PTCK0, PTP0I_0/PTP0I_1 PTCK1_0/PTCK1_1, PTP1I PTCK2_0/PTCK2_1, PTP2I_0/PTP2I_1 PTCK3_0/PTCK3_1, PTP3I_0/PTP3I_1	PTP0 PTP1 PTP2 PTP3

Periodic 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 two internal comparators with the names, Comparator A and Comparator P. These comparators will compare the value in the counter with the CCRA and CCRP registers.

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



Periodic Type TM Block Diagram (n=0~3)

Periodic Type TM Register Description

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

Register Name	Bit							
	7	6	5	4	3	2	1	0
PTMnC0	PTnPAU	PTnCK2	PTnCK1	PTnCK0	PTnON	—	—	—
PTMnC1	PTnM1	PTnM0	PTnIO1	PTnIO0	PTnOC	PTnPOL	PTnCAPTS	PTnCCLR
PTMnDL	D7	D6	D5	D4	D3	D2	D1	D0
PTMnDH	—	—	—	—	—	—	D9	D8
PTMnAL	D7	D6	D5	D4	D3	D2	D1	D0
PTMnAH	—	—	—	—	—	—	D9	D8
PTMnRPL	D7	D6	D5	D4	D3	D2	D1	D0
PTMnRPH	—	—	—	—	—	—	D9	D8

10-bit Periodic TM Register List (n=0~3)

PTMnC0 Register

Bit	7	6	5	4	3	2	1	0
Name	PTnPAU	PTnCK2	PTnCK1	PTnCK0	PTnON	—	—	—
R/W	R/W	R/W	R/W	R/W	R/W	—	—	—
POR	0	0	0	0	0	—	—	—

Bit 7 **PTnPAU**: PTMn 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 **PTnCK2~PTnCK0**: Select PTMn Counter clock
000: $f_{SYS}/4$
001: f_{SYS}
010: $f_H/16$
011: $f_H/64$
100: f_{TBC}
101: f_{TBC}
110: PTCKn_0/PTCKn_1 rising edge clock
111: PTCKn_0/PTCKn_1 falling edge clock
These three bits are used to select the clock source for the PTM. The external pin clock source can be chosen to be active on the rising or falling edge. The clock source f_{SYS} is the system clock, while f_{TBC} is another internal clock, the details of which can be found in the oscillator section.

Bit 3 **PTnON**: PTMn 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 PTM is in the Compare Match Output Mode or the PWM output Mode or Single Pulse Output Mode then the PTM output pin will be reset to its initial condition, as specified by the PTnOC bit, when the PTnON bit changes from low to high.

Bit 2~0 Unimplemented, read as "0"

PTMnC1 Register

Bit	7	6	5	4	3	2	1	0
Name	PTnM1	PTnM0	PTnIO1	PTnIO0	PTnOC	PTnPOL	PTnCAPTS	PTnCCLR
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 **PTnM1~ PTnM0**: Select PTMn Operation 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 PTM. To ensure reliable operation the TM should be switched off before any changes are made to the PTnM1 and PTnM0 bits. In the Timer/Counter Mode, the PTM output pin state is undefined.

Bit 5~4 **PTnIO1~ PTnIO0**: Select PTMn output function
Compare Match Output Mode
00: No change
01: Output low
10: Output high
11: Toggle output
PWM Output Mode /Single Pulse Output Mode
00: PWM Output inactive state
01: PWM Output active state
10: PWM output
11: Single pulse output
Capture Input Mode
00: Input capture at rising edge of the selected PTMn input pin
01: Input capture at falling edge of the selected PTMn input pin
10: Input capture at falling/rising edge of the selected PTMn input pin
11: Input capture disabled
Timer/counter Mode
Unused
These two bits are used to determine how the PTM 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 PTnIO1 and PTnIO0 bits determine how the PTM output pin changes state when a compare match occurs from the Comparator A. The PTM output pin can be setup to switch high, switch low or to toggle its present state when a compare match occurs from the Comparator A. When these bits are both zero, then no change will take place on the output. The initial value of the PTM output pin should be setup using the PTnOC bit. Note that the output level requested by the PTnIO1 and PTnIO0 bits must be different from the initial value setup using the PTnOC bit otherwise no change will occur on the PTM output pin when a compare match occurs. After the PTM output pin changes state, it can be reset to its initial level by changing the level of the PTnON bit from low to high.

In the PWM Mode, the PTnIO1 and PTnIO0 bits determine how the PTM output pin changes state when a certain compare match condition occurs. The PWM output function is modified by changing these two bits. It is necessary to change the values of the PTnIO1 and PTnIO0 bits only after the PTM has been switched off. Unpredictable PWM outputs will occur if the PTnIO1 and PTnIO0 bits are changed when the PTM is running.

Bit 3 **PTnOC**: PTPn Output control bit

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

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

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

Bit 2 **PTnPOL**: PTPn Output polarity Control
0: non-invert
1: invert

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

Bit 1 **PTnCAPTS**: PTMn capture trigger source select
0: From PTPnI_0/PTPnI_1
1: From PTCKn_0/PTCKn_1

Bit 0 **PTnCCLR**: Select PTMn Counter clear condition
0: PTMn Comparatror P match
1: PTMn Comparatror A match

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

PTMnDL 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 **PTMnDL**: PTMn Counter Low Byte Register bit 7 ~ bit 0
PTMn 10-bit Counter bit 7 ~ bit 0

PTMnDH 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 **PTMnDH**: PTMn Counter High Byte Register bit 1 ~ bit 0
PTMn 10-bit Counter bit 9 ~ bit 8

PTMnAL 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 **PTMnAL**: PTMn CCRA Low Byte Register bit 7 ~ bit 0
PTMn 10-bit CCRA bit 7 ~ bit 0

PTMnAH 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 **PTMnAH**: PTMn CCRA High Byte Register bit 1 ~ bit 0
PTMn 10-bit CCRA bit 9 ~ bit 8

PTMnRPL 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 **PTMnRPL**: PTMn CCRP Low Byte Register bit 7 ~ bit 0
PTMn 10-bit CCRP bit 7 ~ bit 0

PTMnRPH 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 **PTMnRPH**: PTMn CCRP High Byte Register bit 1 ~ bit 0
PTMn 10-bit CCRP bit 9 ~ bit 8

Periodic Type TM Operating Modes

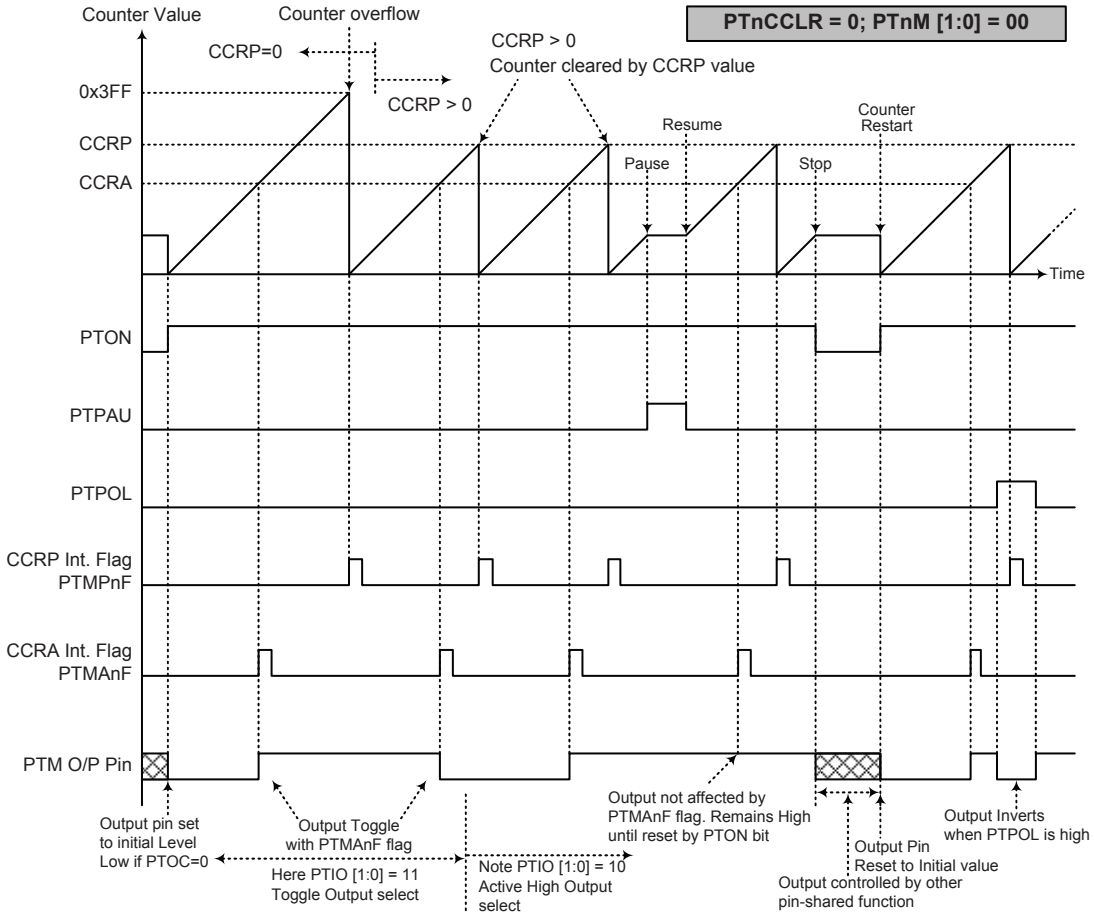
The Periodic Type TM can operate in one of five operating modes, Compare Match Output Mode, PWM Output Mode, Single Pulse Output Mode, Capture Input Mode or Timer/Counter Mode. The operating mode is selected using the PTnM1 and PTnM0 bits in the PTMnC1 register.

Compare Match Output Mode

To select this mode, bits PTnM1 and PTnM0 in the PTMnC1 register, should be all cleared 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 PTnCCLR 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 PTMAnF and PTMPnF interrupt request flags for Comparator Aand Comparator P respectively, will both be generated.

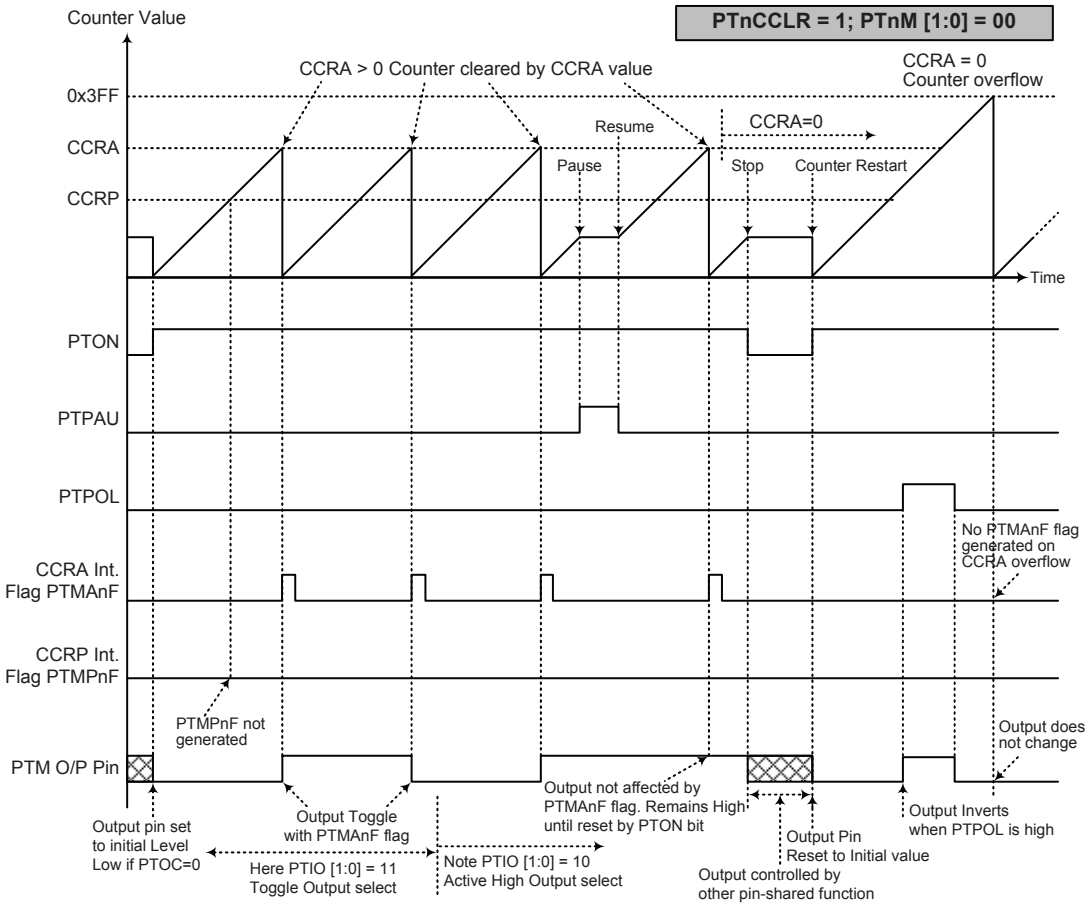
If the PTnCCLR bit in the PTMnC1 register is high then the counter will be cleared when a compare match occurs from Comparator A. However, here only the PTMAnF interrupt request flag will be generated even if the value of the CCRP bits is less than that of the CCRA registers. Therefore when PTnCCLR is high no PTMPnF interrupt request flag will be generated. In the Compare Match Output Mode, the CCRA can not be set to “0”. If the CCRA bits are all zero, the counter will overflow when its reaches its maximum 10-bit, 3FF Hex, value, however here the PTMAnF 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 PTMAnF interrupt request flag is generated after a compare match occurs from Comparator A. The PTMPnF 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 are determined by the condition of the PTnIO1 and PTnIO0 bits in the PTMnC1 register. The TM output pin can be selected using the PTnIO1 and PTnIO0 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 PTnON bit changes from low to high, is setup using the PTnOC bit. Note that if the PTnIO1, PTnIO0 bits are zero then no pin change will take place.



Compare Match Output Mode – PTnCCLR = 0 (n=0~3)

Note: 1. With PTnCCLR = 0 – a Comparator P match will clear the counter
2. The PTM output pin is controlled only by the PTMAnF flag
3. The output pin is reset to initial state by a PTnON bit rising edge



Compare Match Output Mode – PTnCCLR = 1 (n=0~3)

Note: 1. With PTnCCLR = 1 – a Comparator A match will clear the counter
2. The PTM output pin is controlled only by the PTMAnF flag
3. The output pin is reset to initial state by a PTnON rising edge
4. The PTMPnF flag is not generated when PTnCCLR = 1

Timer/Counter Mode

To select this mode, bits PTnM1 and PTnM0 in the PTMnC1 register should all be set to 11 respectively. The Timer/Counter Mode operates in an identical way to the Compare Match Output Mode generating the same interrupt flags. The exception is that in the Timer/Counter Mode the PTM output pin is not used. Therefore the above description and Timing Diagrams for the Compare Match Output Mode can be used to understand its function. As the PTM output pin is not used in this mode, the pin can be used as a normal I/O pin or other pin-shared function.

PWM Output Mode

To select this mode, bits PTnM1 and PTnM0 in the PTMnC1 register should be set to 10 respectively and also the PTnIO1 and PTnIO0 bits should be set to 10 respectively. The PWM function within the PTM is useful for applications which require functions such as motor control, heating control, illumination control etc. By providing a signal of fixed frequency but of varying duty cycle on the 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 PTnCCLR bit has no effect as the PWM period. Both of the CCRP and CCRA registers are used to generate the PWM waveform, CCRP registers are used to clear the internal counter and thus control the PWM waveform frequency, while CCRA registers are used to control the duty cycle. The PWM waveform frequency and duty cycle can therefore be controlled by the values in the CCRA and CCRP registers.

An interrupt flag, one for each of the CCRA and CCRP, will be generated when a compare match occurs from either Comparator A or Comparator P. The PTnOC bit in the PTMnC1 register is used to select the required polarity of the PWM waveform while the PTnIO1 and PTnIO0 bits are used to enable the PWM output or to force the PTM output pin to a fixed high or low level. The PTnPOL bit is used to reverse the polarity of the PWM output waveform.

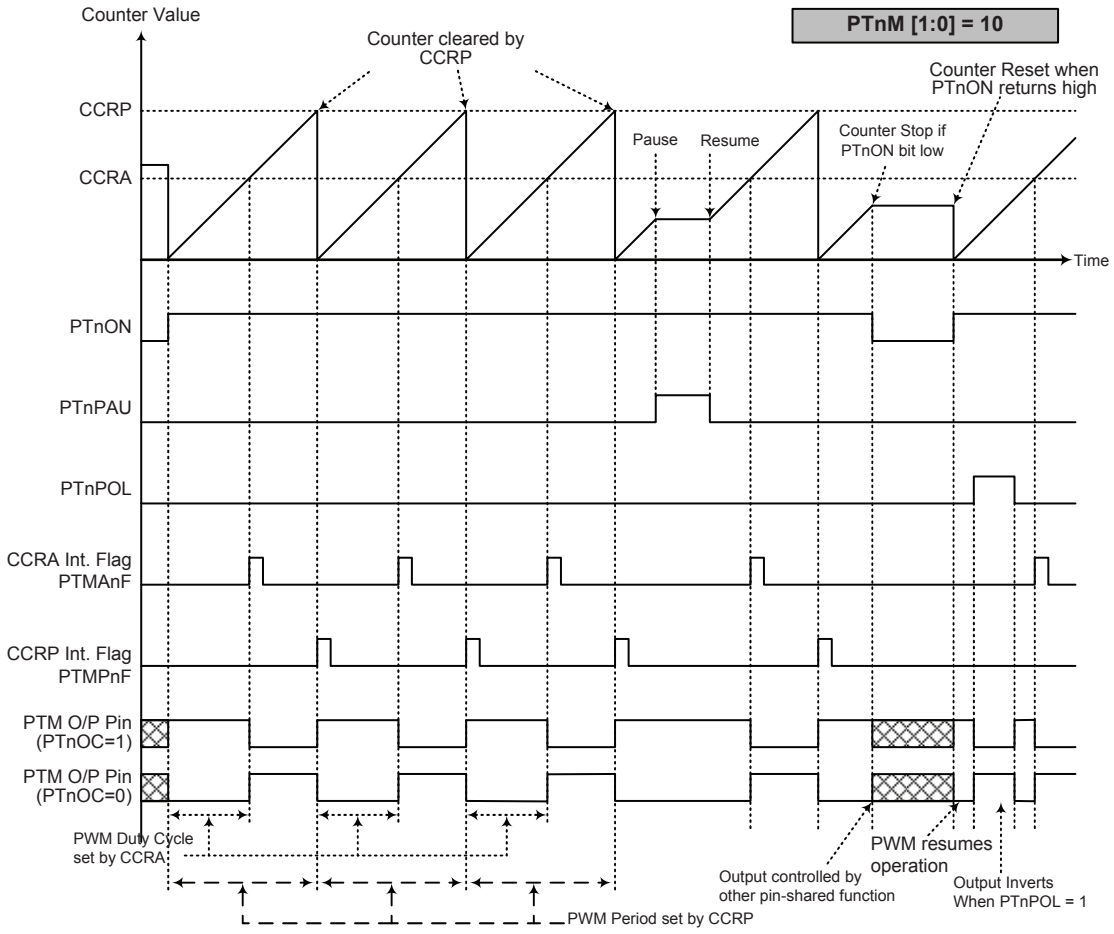
- **10-bit PWM Mode, Edge-aligned Mode**

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

If f_{SYS} = 16MHz, PTM clock source select $f_{SYS}/4$, CCRP = 100b and CCRA = 128,

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

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



PWM Output Mode (n=0~3)

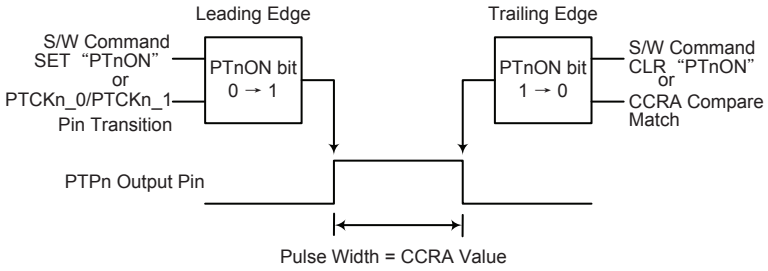
Note: 1. Here Counter cleared by CCRP
2. A counter clear sets the PWM Period
3. The internal PWM function continues running even when PTnIO[1:0] = 00 or 01
4. The PTnCCLR bit has no influence on PWM operation

Single Pulse Output Mode

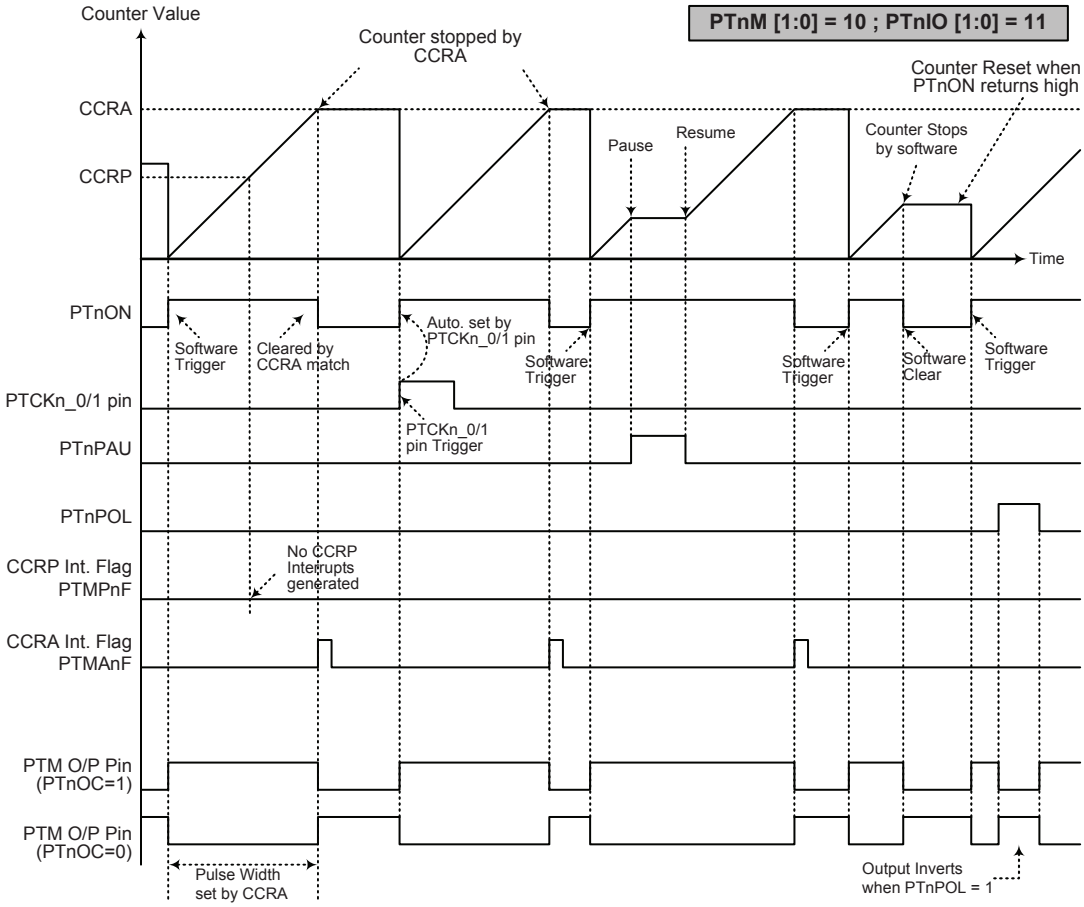
To select this mode, the required bit pairs, PTnM1 and PTnM0 should be set to 10 respectively and also the corresponding PTnIO1 and PTnIO0 bits should be set to 11 respectively. The Single Pulse Output Mode, as the name suggests, will generate a single shot pulse on the TM output pin.

The trigger for the pulse output leading edge is a low to high transition of the PTnON bit, which can be implemented using the application program. However in the Single Pulse Mode, the PTnON bit can also be made to automatically change from low to high using the external PTCKn_0/PTCKn_1 pin, which will in turn initiate the Single Pulse output. When the PTnON bit transitions to a high level, the counter will start running and the pulse leading edge will be generated. The PTnON bit should remain high when the pulse is in its active state. The generated pulse trailing edge will be generated when the PTnON 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 PTnON bit and thus generate the Single Pulse output trailing edge. In this way the CCRA value can be used to control the pulse width. A compare match from Comparator A will also generate PTM interrupts. The counter can only be reset back to zero when the PTnON bit changes from low to high when the counter restarts. In the Single Pulse Mode CCRP is not used. The PTnCCLR bit is not used.



Single Pulse Generation (n=0~3)



Single Pulse Mode (n=0 ~ 3)

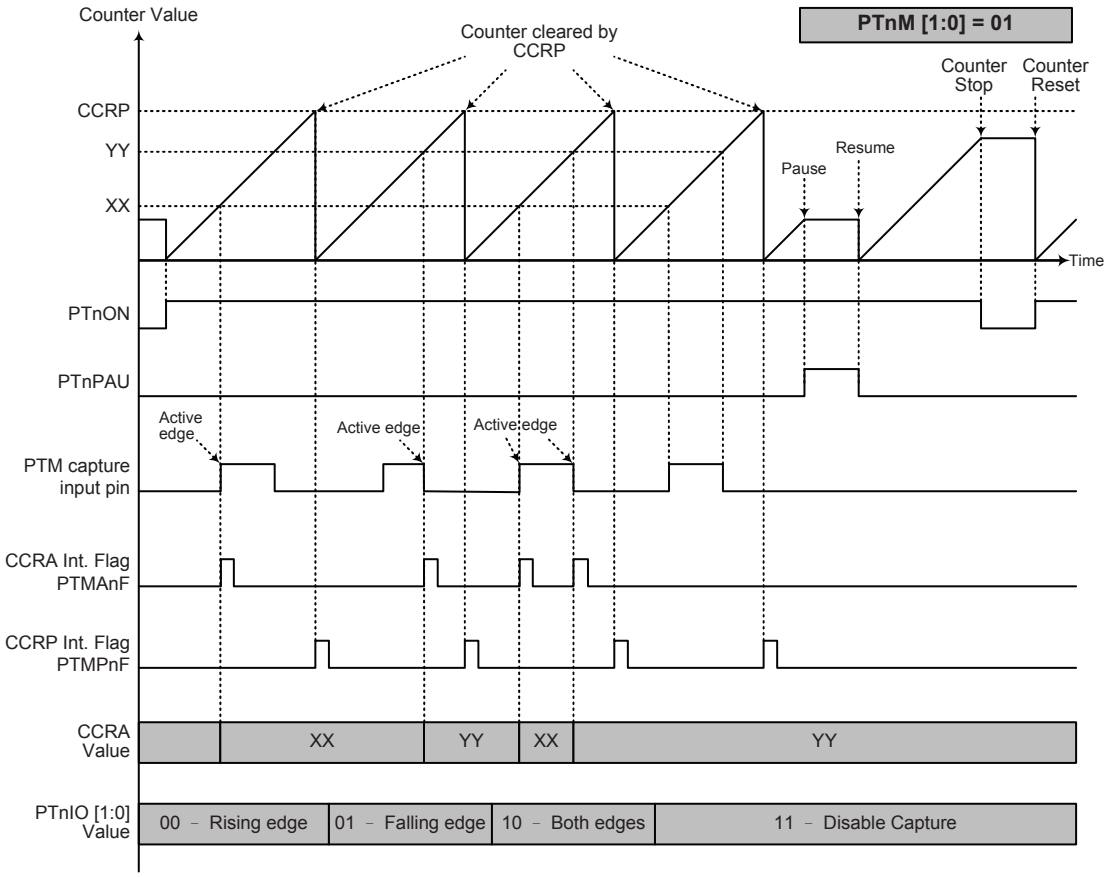
Note: 1. Counter stopped by CCRA
2. CCRP is not used
3. The pulse is triggered by the PTCKn_0/1 pin or by setting the PTnON bit high
4. A PTCKn_0/1 pin active edge will automatically set the PTnON bit high
5. In the Single Pulse Mode, PTnIO [1:0] must be set to “11” and can not be changed.

Capture Input Mode

To select this mode bits PTnM1 and PTnM0 in the PTMnC0 register should be set to 01 respectively. This mode enables external signals to capture and store the present value of the internal counter and can therefore be used for applications such as pulse width measurements. The external signal is supplied on the PTPnI_0/ PTPnI_1 or PTCKn_0/ PTCKn_1 pin, selected by the PTnCAPTS bit in the PTMnC1 register. The input pin active edge can be either a rising edge, a falling edge or both rising and falling edges; the active edge transition type is selected using the PTnIO1 and PTnIO0 bits in the PTMnC1 register. The counter is started when the PTnON bit changes from low to high which is initiated using the application program.

When the required edge transition appears on the selected PTMn input pin the present value in the counter will be latched into the CCRA register and a PTM interrupt generated. Irrespective of what events occur on the selected input pin the counter will continue to free run until the PTnON bit changes from high to low. When a CCRP compare match occurs the counter will reset back to zero; in this way the CCRP value can be used to control the maximum counter value. When a CCRP compare match occurs from Comparator P, a PTM interrupt will also be generated. Counting the number of overflow interrupt signals from the CCRP can be a useful method in measuring long pulse widths. The PTnIO1 and PTnIO0 bits can select the active trigger edge on the input pin to be a rising edge, falling edge or both edge types. If the PTnIO1 and PTnIO0 bits are both set high, then no capture operation will take place irrespective of what happens on the input pin, however it must be noted that the counter will continue to run.

As the PTPnI_0/ PTPnI_1 or PTCKn_0/ PTCKn_1 pin is pin shared with other functions, care must be taken if the PTM 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 PTnCCLR, PTnOC and PTnPOL bits are not used in this Mode.



Capture Input Mode (n=0~3)

Note: 1. PTnM[1:0] = 01 and active edge set by the PTnIO[1:0] bits
2. A PTM Capture input pin active edge transfers counter value to CCRA
3. The PTnCCLR bit is not used
4. No output function – PTnOC and PTnPOL bits are not used
5. CCRP determines the counter value and the counter has a maximum count value when CCRP is equal to zero

Analog to Digital Converter

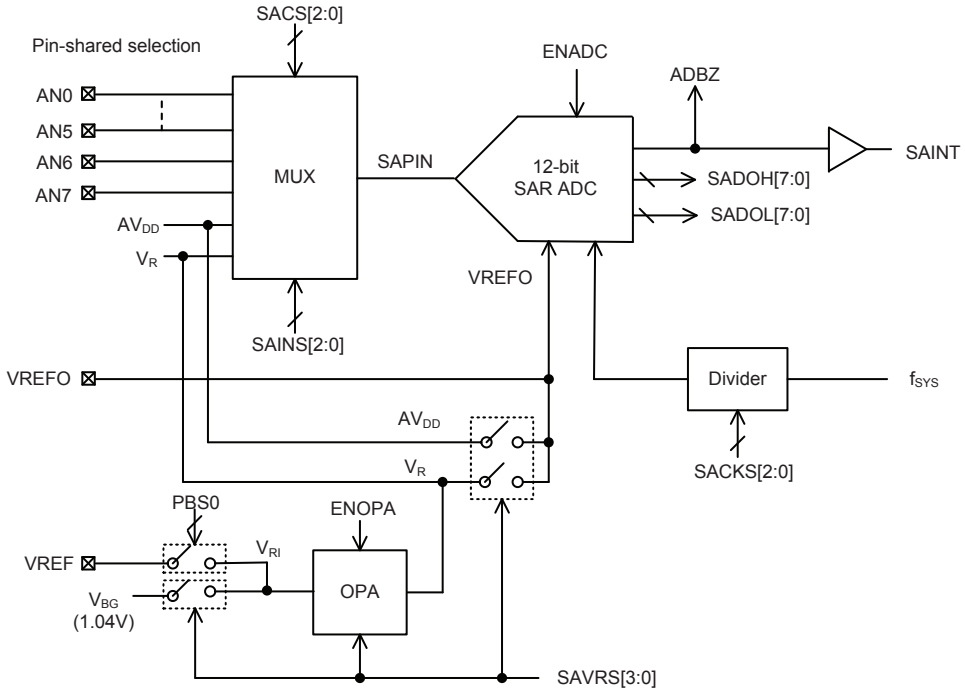
The need to interface to real world analog signals is a common requirement for many electronic systems. However, to properly process these signals by a microcontroller, they must first be converted into digital signals by A/D converters. By integrating the A/D conversion electronic circuitry into the 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 contain an 8-channel analog to digital converter which can directly interface to external analog signals, such as that from sensors or other control signals and convert these signals directly into a 12-bit digital value. The external or internal analog signal to be converted is determined by the SAINS and SACS bit fields. Additionally note that when the internal analog signal is to be converted, the pin-shared control bits should also be properly configured first. More detailed information about the A/D input signal is described in the "A/D Converter Control Registers" and "A/D Converter Input Signal" sections respectively.

Input Channels	A/D Channel Select Bits	Input Pins
8	SAINS2~SAINS0, SACS2~SACS0	AN0~AN7

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



A/D Converter Structure

A/D Converter Register Description

Overall operation of the A/D converter is controlled using 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
SADOL(ADRFS=0)	D3	D2	D1	D0	—	—	—	—
SADOL(ADRFS=1)	D7	D6	D5	D4	D3	D2	D1	D0
SADOH(ADRFS=0)	D11	D10	D9	D8	D7	D6	D5	D4
SADOH(ADRFS=1)	—	—	—	—	D11	D10	D9	D8
SADC0	START	ADBZ	ENADC	ADRFS	—	SACS2	SACS1	SACS0
SADC1	SAINS2	SAINS1	SAINS0	—	—	SACKS2	SACKS1	SACKS0
SADC2	ENOPA	VBGEN	—	—	SAVRS3	SAVRS2	SAVRS1	SAVRS0

A/D Converter Register List

A/D Converter Data Registers – SADOL, SADOH

As the devices contain an internal 12-bit A/D converter, it requires two data registers to store the converted value. These are a high byte register, known as SADOH, and a low byte register, known as SADOL. After the conversion process takes place, these registers can be directly read by the microcontroller to obtain the digitised conversion value. As only 12 bits of the 16-bit register space is utilised, the format in which the data is stored is controlled by the ADRFS bit in the SADC0 register as shown in the accompanying table. D0~D11 are the A/D conversion result data bits. Any unused bits will be read as zero. Note that the A/D converter data register contents will not be cleared to zero if the A/D converter is disabled.

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

A/D Data Registers

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

To control the function and operation of the A/D converter, several control registers known as SADC0, SADC1 and SADC2 are provided. These 8-bit registers define functions such as the selection of which analog channel is connected to the internal A/D converter, the digitised data format, the A/D clock source as well as controlling the start function and monitoring the A/D converter busy status. The SACS2~SACS0 bits in the SADC0 register are used to determine which external channel input is selected to be converted. The SAINS2~SAINS0 bits in the SADC1 register are used to determine that the analog signal to be converted comes from the internal analog signal or external analog channel input. If the SAINS2~SAINS0 bits are set to "000", the external analog channel input is selected to be converted and the SACS2~SACS0 bits can determine which external channel is selected to be converted. If the SAINS2~SAINS0 bits are set to "001~011", the AV_{DD} voltage is selected to be converted. If the SAINS2~SAINS0 bits are set to "101~111", the OPA output voltage is selected to be converted. When V_{REF} or V_{BG} is selected as ADC input or ADC reference voltage, the OPA needs to be enabled by setting ENOPA to 1first.

Note that when the programs select external signal and internal signal as an ADC input signal simultaneously, then the hardware will only choose the internal signal as an ADC input. In addition, if the programs select external reference voltage V_{REF} and the internal reference voltage V_{BG} as ADC reference voltage, then the hardware will only choose the internal reference voltage V_{BG} as an ADC reference voltage input.

Care must be taken when the internal analog signal is selected to be converted. If the internal analog signal is selected to be converted, the selected external input pin determined by the SACS2~SACS0 bits must never be configured as A/D input function by properly setting the relevant pin-shared control bits. Otherwise, the external channel input will be connected together with the internal analog signal. This will result in unpredictable situations such as an irreversible damage.

The pin-shared function control registers, named PAS0, PAS1 and PBS0, contain the corresponding pin-shared selection bits which determine which pins on Port A and Port B are used as analog inputs for the A/D converter input and which pins are not to be used as the A/D converter input. When the pin is selected to be an A/D input, its original function whether it is an I/O or other pin-shared function will be removed. In addition, any internal pull-high resistors connected to these pins will be automatically removed if the pin is selected to be an A/D input.

SADC0 Register

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

Bit 7 **START**: Start the A/D conversion
0→1→0: Start A/D conversion
0→1: Reset the A/D converter and set ADBZ to 0
1→0: Start A/D conversion and set ADBZ to 1

Bit 6 **ADBZ**: ADC busy flag
0: A/D conversion ended or no conversion
1: A/D is busy

Bit 5 **ENADC**: ADC enable/disable control register
0: Disable
1: Enable

Bit 4 **ADRFS**: A/D output data format selection bit
0: ADC output data format → SADOH=D[11:4]; SADOL=D[3:0]
1: ADC output data format → SADOH=D[11:8]; SADOL=D[7:0]

Bit 3~2 Unimplemented, read as "0"

Bit 1~0 **SACS2~SACS0**: ADC input channels selection
000: ADC input channel comes from AN0
001: ADC input channel comes from AN1
010: ADC input channel comes from AN2
011: ADC input channel comes from AN3
100: ADC input channel comes from AN4
101: ADC input channel comes from AN5
110: ADC input channel comes from AN6
111: ADC input channel comes from AN7

SADC1 Register

Bit	7	6	5	4	3	2	1	0
Name	SAINS2	SAINS1	SAINS0	—	—	SACKS2	SACKS1	SACKS0
R/W	R/W	R/W	R/W	—	—	R/W	R/W	R/W
POR	0	0	0	—	—	0	0	0

Bit 7~5 **SAINS2~SAINS0**: Internal ADC input channel selection bit
000: ADC input only comes from external pin
001: ADC input also comes from AV_{DD}
010: ADC input also comes from $AV_{DD}/2$
011: ADC input also comes from $AV_{DD}/4$
101: ADC input also comes from V_R
110: ADC input also comes from $V_R/2$
111: ADC input also comes from $V_R/4$
Other Values: same as 000
Note: V_R is OPA output voltage. V_R can be one of V_{REF} / $V_{REF}\times2$ / $V_{REF}\times3$ / $V_{REF}\times4$ / $V_{BG}\times2$ / $V_{BG}\times3$ / $V_{BG}\times4$.

Bit 4~3 Unimplemented, read as "0"

Bit 2~0 **SACKS2~SACKS0**: ADC clock rate selection bit
000: f_{SYS}
001: f_{SYS} / 2
010: f_{SYS} / 4
011: f_{SYS} / 8
100: f_{SYS} / 16
101: f_{SYS} / 32
110: f_{SYS} / 64
111: f_{SYS} /128

SADC2 Register

Bit	7	6	5	4	3	2	1	0
Name	ENOPA	VBGEN	—	—	SAVRS3	SAVRS2	SAVRS1	SAVRS0
R/W	R/W	R/W	—	—	R/W	R/W	R/W	R/W
POR	0	0	—	—	0	0	0	0

Bit 7 **ENOPA**: OPA enable/disable control register
0: Disable
1: Enable

Bit 6 **VBGEN**: Bandgap buffer disable/enable control bit
0: Disable
1: Enable

Bit 5~4 Unimplemented, read as "0"

Bit 3~0 **SAVRS3~SAVRS0**: ADC reference voltage selection bit
0000: ADC reference voltage comes from AV_{DD}
0001: ADC reference voltage comes from V_{REF}
0010: ADC reference voltage comes from $V_{REF} \times 2$
0011: ADC reference voltage comes from $V_{REF} \times 3$
0100: ADC reference voltage comes from $V_{REF} \times 4$
1001: Inhibit to use
1010: ADC reference voltage comes from $V_{BG} \times 2$
1011: ADC reference voltage comes from $V_{BG} \times 3$
1100: ADC reference voltage comes from $V_{BG} \times 4$
Other Values: same as 0000

Note: 1. When selecting V_{REF}, V_{REF}×2, V_{REF}×3 or V_{REF}×4 as the ADC reference voltage, the pin shared control register should be correctly set to select VREF as the input.
2. V_{BG}=1.04V
3. When SAVRS3=1, then OPA selects V_{BG} as input.
4. If the program selects an external reference voltage V_{REF} and the internal reference voltage V_{BG} as the ADC reference voltage, then the hardware will only choose the internal reference voltage V_{BG} as an ADC reference voltage input.

A/D Operation

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

Although the A/D clock source is determined by the system clock f_{SYS}, and by bits SACK2~SACK0, there are some limitations on the maximum A/D clock source speed that can be selected. As the recommended value of permissible A/D clock period, t_{ADCK}, is from 0.5μs to 10μs, care must be taken for system clock frequencies. For example, if the system clock operates at a frequency of 4MHz, the SACK2~SACK0 bits should not be set to 000B or 11xB. Doing so will give A/D clock periods that are less than the minimum A/D clock period or greater than the maximum A/D clock period which may result in inaccurate A/D conversion values.

Controlling the power on/off function of the A/D converter circuitry is implemented using the ENADC bit in the SADC0 register. This bit must be set high to power on the A/D converter. When the ENADC bit is set high to power on the A/D converter internal circuitry a certain delay, as indicated in the timing diagram, must be allowed before an A/D conversion is initiated. Even if no pins are selected for use as A/D inputs by configuring the corresponding pin-shared control bits, if the ENADC bit is high then some power will still be consumed. In power conscious applications it is therefore recommended that the ENADC is set low 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 internal ADC power or from an external reference sources supplied on pin VREF or V_{BG} voltage. The desired selection is made using the SAVRS3~ SAVRS0 bits. As the VREF pin is pin-shared with other functions, when the VREF pin is selected as the reference voltage supply pin, the VREF pin-shared function control bits should be properly configured to disable other pin functions. When V_{REF} or V_{BG} is selected by ADC input or ADC reference voltage, the OPA needs to be enabled by setting ENOPA=1.

Reference Voltage	SAVRS[3:0]	Description
AV_{DD}	0000	ADC reference voltage comes from AV_{DD}
V_{REF}	0001	ADC reference voltage comes from external V_{REF}
V_{REF}×2	0010	ADC reference voltage comes from external V_{REF}×2
V_{REF}×3	0011	ADC reference voltage comes from external V_{REF}×3
V_{REF}×4	0100	ADC reference voltage comes from external V_{REF}×4
V_{BG}×2	1010	ADC reference voltage comes from V_{BG}×2
V_{BG}×3	1011	ADC reference voltage comes from V_{BG}×3
V_{BG}×4	1100	ADC reference voltage comes from V_{BG}×4

A/D Converter Reference Voltage Selection

A/D Converter Input Signal

All of the A/D analog input pins are pin-shared with the I/O pins on Port A and Port B as well as other functions. The corredponding selection bits for each I/O pin in the PAS0, PAS1 and PBS0 registers, determine whether the input pins are setup as A/D converter analog inputs or whether they have other functions. If the pin-shared function control bits configure its corresponding pin as an A/D analog channel input, the pin will be setup to be an A/D converter external channel 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 and PBC port control register to enable the A/D input as when the pin-shared function control bits enable an A/D input, the status of the port control register will be overridden.

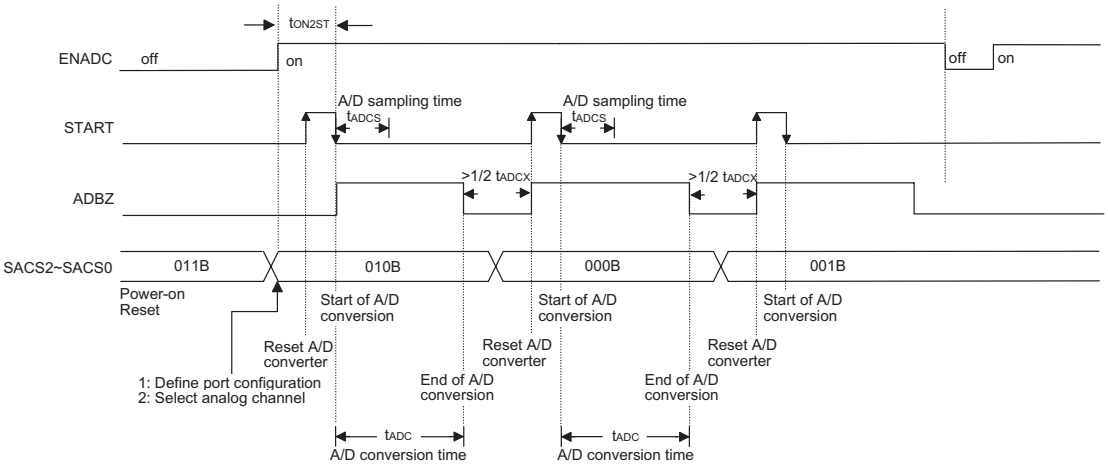
The A/D converter has its own reference voltage pin, VREF, however the reference voltage can also be supplied from the power supply pin, a choice which is made through the SAVRS[3:0] in the SADC2 register. The selected A/D reference voltage can be output on the VREFO pin. The analog input values must not be allowed to exceed the value of V_{REF}.

Conversion Rate and Timing Diagram

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

Maximum single A/D conversion rate = A/D clock period / 16

However, there is a usage limitation on the next A/D conversion after the current conversion is complete. When the current A/D conversion is complete, the converted digital data will be stored in the A/D data register pair and then latched after half an A/D clock cycle. If the START bit is set to 1 in half an A/D clock cycle after the end of A/D conversion, the converted digital data stored in the A/D data register pair will be changed. Therefore, it is recommended to initiate the next A/D conversion after a certain period greater than half an A/D clock cycle at the end of current A/D conversion.



A/D Conversion Timing

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 frequency by SACKS2~ SACKS0
- Step 2
  Enable the ADC by set ENADC=1
- Step 3
  Select which pins will be configure as ADC analog inputs by setting the pin-shared register.
- Step 4
  If input comes from external channel, set SAINS[2:0]=000 and then set SACS bit fields to corresponding input channel
  If input comes from internal input, set SAINS[2:0] to corresponding internal input source
- Step 5
  Select reference voltage comes from external V_{REF}, AV_{DD} or V_{BG} by SAVRS[3:0]
  Note: If select V_{REF} as reference voltage, set PBS0[1:0] = 1, 0
- Step 6
  Select ADC output data format by ADRFS
- Step 7
  If ADC interrupt is used, the interrupt control registers must be correctly configured to ensure the A/D interrupt function is active. The master interrupt control bit, EMI, and the A/D converter interrupt bits, ADE, must both set high in advance.
- Step 8
  The A/D convert procedure can now be initialized by set START from low to high and then low again
- Step 9
  If ADC is under conversion, ADBZ=1. After A/D conversion process is completed, the ADBZ flag will go low, and then output data can be read from SADOH and SADOL registers. If the ADC interrupt is enabled and the stack is not full, data can be acquired by interrupt service program. Another way to get the A/D output data is polling the ADBZ flag.

Programming Considerations

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

A/D Transfer Function

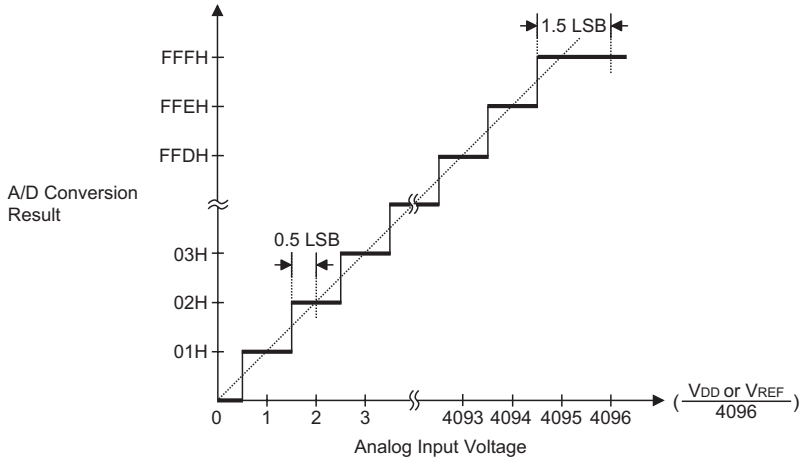
As the devices contain a 12-bit A/D converter, its full-scale converted digitised value is equal to FFFH. Since the full-scale analog input value is equal to the V_{DD} or V_{REF} voltage, this gives a single bit analog input value of V_{DD} or V_{REF} divided by 4096.

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

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

$$\text{A/D input voltage} = \text{A/D output digital value} \times (AV_{DD} \text{ or } V_{REF}) / 4096$$

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.



Ideal A/D Transfer Function

A/D Programming Examples

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

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

```
clr  ADE              ; disable ADC interrupt
mov  a,0BH
mov  SADC1,a          ; select fSYS/8 as A/D clock and switch off the bandgap reference
                      ; voltage
set  ENADC
mov  a,03h            ; setup PBS0 register to configure pin AN0
mov  PBS0,a
mov  a,20h
mov  SADC0,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   ADBZ             ; poll the SADC0 register ADBZ bit to detect end of A/D conversion
jmp  polling_EOC      ; continue polling
mov  a,SADOL          ; read low byte conversion result value
mov  SADOL_buffer,a   ; save result to user defined register
mov  a,SADOH          ; read high byte conversion result value
mov  SADOH_buffer,a   ; save result to user defined register
:
:
jmp  start_conversion ; start next a/d conversion
```

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

```
clr  ADE              ; disable ADC interrupt
mov  a,0BH
mov  SADC1,a          ; select fSYS/8 as A/D clock and switch off the bandgap reference
                      ; voltage
set  ENADC
mov  a,03h            ; setup PBS0 register to configure pin AN0
mov  PBS0,a
mov  a,20h
mov  SADC0,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  ADE              ; enable ADC interrupt
set  EMI              ; enable global interrupt
:
:
                      ; ADC interrupt service routine
ADC_ISR:
mov  acc_stack,a      ; save ACC to user defined memory
mov  a,STATUS
mov  status_stack,a   ; save STATUS to user defined memory
:
:
mov  a,SADOL          ; read low byte conversion result value
mov  SADOL_buffer,a   ; save result to user defined register
mov  a,SADOH          ; read high byte conversion result value
mov  SADOH_buffer,a   ; save result to user defined register
:
:
EXIT_INT_ISR:
mov  a,status_stack
mov  STATUS,a         ; restore STATUS from user defined memory
mov  a,acc_stack      ; restore ACC from user defined memory
reti
```

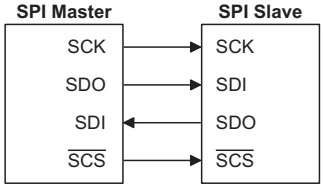
Serial Interface Module – SIM

These devices contain a Serial Interface Module, which includes both the four-line SPI interface or two-line I²C 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 I²C based hardware such as sensors, Flash or EEPROM memory, etc. The SIM interface pins are pin-shared with other I/O pins and therefore the SIM interface functional pins must first be selected using the corresponding pin-shared function selection bits. As both interface types share the same pins and registers, the choice of whether the SPI or I²C type is used is made using the SIM operating mode control bits, named SIM2~SIM0, in the SIMC0 register. These pull-high resistors of the SIM pin-shared I/O pins are selected using pull-high control registers when the SIM function is enabled and the corresponding pins are used as SIM input pins.

SPI Interface

The SPI 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 SPI 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 devices can be either master or slave. Although the SPI interface specification can control multiple slave devices from a single master, these devices provided only one $\overline{SCS}$ 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.



SPI Master/Slave Connection

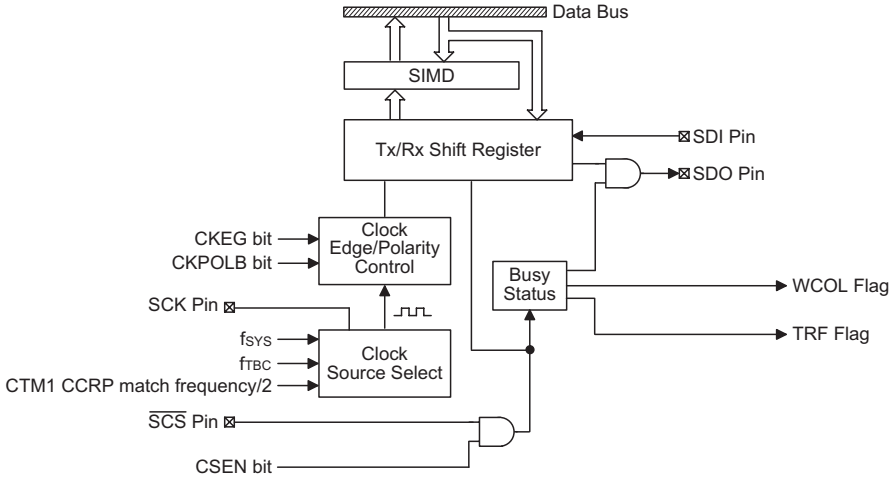
SPI Interface Operation

The SPI interface is a full duplex synchronous serial data link. It is a four line interface with pin names SDI, SDO, SCK and $\overline{SCS}$. Pins SDI and SDO are the Serial Data Input and Serial Data Output lines, SCK is the Serial Clock line and $\overline{SCS}$ is the Slave Select line. As the SPI interface pins are pin-shared with normal I/O pins and with the I²C function pins, the SPI interface pins must first be selected by configuring the pin-shared function selection bits and setting the correct bits in the SIMC0 and SIMC2 registers. After the desired SPI configuration has been set it can be disabled or enabled using the SIMEN bit in the SIMC0 register. Communication between devices connected to the SPI 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 $\overline{SCS}$ pin only one slave device can be utilized. The $\overline{SCS}$ pin is controlled by software, set CSEN bit to 1 to enable $\overline{SCS}$ pin function, clear CSEN bit to 0 the $\overline{SCS}$ pin will be floating state.

The SPI 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

The status of the SPI 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 CSEN and SIMEN.



SPI Block Diagram

SPI Registers

There are three internal registers which control the overall operation of the SPI interface. These are the SIMD data register and two registers SIMC0 and SIMC2. Note that the SIMC1 register is only used by the I²C interface.

Register Name	Bit							
	7	6	5	4	3	2	1	0
SIMC0	SIM2	SIM1	SIM0	—	SIMDEB1	SIMDEB0	SIMEN	SIMICF
SIMC2	D7	D6	CKPOLB	CKEG	MLS	CSEN	WCOL	TRF
SIMD	D7	D6	D5	D4	D3	D2	D1	D0

SPI Registers List

• SIMD Register

The SIMD register is used to store the data being transmitted and received. The same register is used by both the SPI and I²C functions. Before the device writes data to the SPI bus, the actual data to be transmitted must be placed in the SIMD register. After the data is received from the SPI bus, the device can read it from the SIMD register. Any transmission or reception of data from the SPI bus must be made via the SIMD register.

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

"x": unknown

There are also two control registers for the SPI interface, SIMC0 and SIMC2. Note that the SIMC2 register also has the name SIMA which is used by the I²C function. The SIMC1 register is not used by the SPI function, only by the I²C function. Register SIMC0 is used to control the enable/disable function and to set the data transmission clock frequency. Register SIMC2 is used for other control functions such as LSB/MSB selection, write collision flag, etc.

- **SIMC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	SIM2	SIM1	SIM0	—	SIMDEB1	SIMDEB0	SIMEN	SIMICF
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 **SIM2~SIM0**: SIM 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 CTM1 CCRP match frequency/2
101: SPI slave mode
110: I²C slave mode
111: Non SIM function
These bits setup the overall operating mode of the SIM function. As well as selecting if the I²C or SPI function, they are used to control the SPI Master/Slave selection and the SPI Master clock frequency. The SPI clock is a function of the system clock but can also be chosen to be sourced from CTM1. If the SPI Slave Mode is selected then the clock will be supplied by an external Master device.

Bit 4 Unimplemented, read as "0"

Bit 3~2 **SIMDEB1~SIMDEB0**: I²C Debounce Time Selection
Described elsewhere.

Bit 1 **SIMEN**: SIM Enable Control
0: Disable
1: Enable
The bit is the overall on/off control for the SIM interface. When the SIMEN bit is cleared to zero to disable the SIM interface, the SDI, SDO, SCK and $\overline{SCS}$, or SDA and SCL lines will lose their SPI or I²C function and the SIM operating current will be reduced to a minimum value. When the bit is high the SIM interface is enabled. If the SIM is configured to operate as an SPI interface via the SIM2~SIM0 bits, the contents of the SPI control registers will remain at the previous settings when the SIMEN bit changes from low to high and should therefore be first initialised by the application program. If the SIM is configured to operate as an I²C interface via the SIM2~SIM0 bits and the SIMEN bit changes from low to high, the contents of the I²C control bits such as HTX and TXAK will remain at the previous settings and should therefore be first initialised by the application program while the relevant I²C flags such as HCF, HAAS, HBB, SRW and RXAK will be set to their default states.

Bit 0 **SIMICF**: SIM SPI slave mode Incomplete Transfer Flag
0: SIM SPI slave mode incomplete condition not occurred
1: SIM SPI slave mode incomplete condition occured
This bit is only available when the SIM is configured to operate in an SPI slave mode. If the SPI operates in the slave mode with the SIMEN and CSEN bits both being set to 1 but the $\overline{SCS}$ line is pulled high by the external master device before the SPI data transfer is completely finished, the SIMICF bit will be set to 1 together with the TRF bit. When this condition occurs, the corresponding interrupt will occur if the interrupt function is enabled. However, the TRF bit will not be set to 1 if the SIMICF bit is set to 1 by software application program.

- **SIMC2 Register**

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	CKPOLB	CKEG	MLS	CSEN	WCOL	TRF
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 bits

These bits can be read or written by the application program.

Bit 5 **CKPOLB**: SPI clock line base condition selection

0: The SCK line will be high when the clock is inactive.

1: The SCK line will be low when the clock is inactive.

The CKPOLB bit determines the base condition of the clock line, if the bit is high, then the SCK line will be low when the clock is inactive. When the CKPOLB bit is low, then the SCK line will be high when the clock is inactive.

Bit 4 **CKEG**: SPI SCK clock active edge type selection

CKPOLB=0

0: SCK is high base level and data capture at SCK rising edge

1: SCK is high base level and data capture at SCK falling edge

CKPOLB=1

0: SCK is low base level and data capture at SCK falling edge

1: SCK is low base level and data capture at SCK rising edge

The CKEG and CKPOLB bits are used to setup the way that the clock signal outputs and inputs data on the SPI bus. These two bits must be configured before data transfer is executed otherwise an erroneous clock edge may be generated. The CKPOLB bit determines the base condition of the clock line, if the bit is high, then the SCK line will be low when the clock is inactive. When the CKPOLB bit is low, then the SCK line will be high when the clock is inactive. The CKEG bit determines active clock edge type which depends upon the condition of CKPOLB bit.

Bit 3 **MLS**: SPI data shift order

0: LSB first

1: MSB first

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 **CSEN**: SPI $\overline{\text{SCS}}$ pin control

0: Disable

1: Enable

The CSEN bit is used as an enable/disable for the $\overline{\text{SCS}}$ pin. If this bit is low, then the $\overline{\text{SCS}}$ pin will be disabled and placed into I/O pin or other pin-shared functions. If the bit is high, the $\overline{\text{SCS}}$ pin will be enabled and used as a select pin.

Bit 1 **WCOL**: SPI write collision flag

0: No collision

1: Collision

The WCOL flag is used to detect whether a data collision has occurred or not. If this bit is high, it means that data has been attempted to be written to the SIMD register duting a data transfer operation. This writing operation will be ignored if data is being transferred. This bit can be cleared by the application program.

Bit 0 **TRF**: SPI Transmit/Receive complete flag

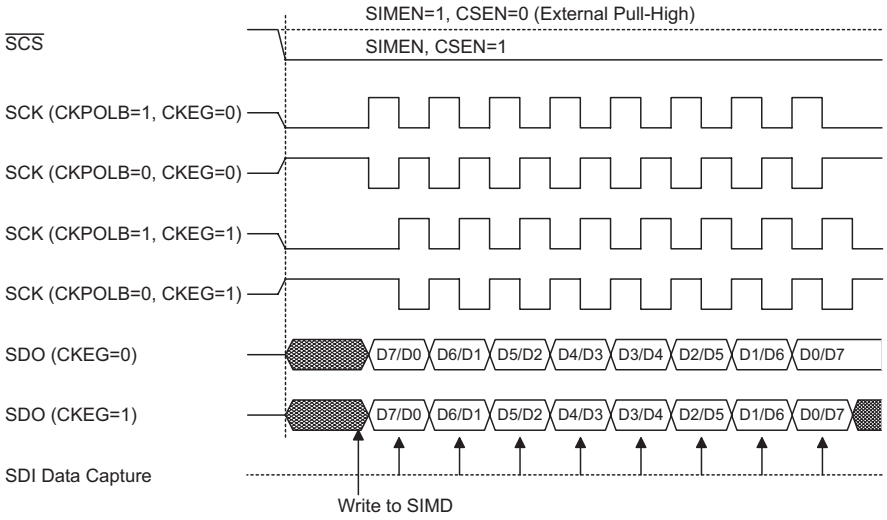
0: SPI data is being transferred

1: SPI data transfer is completed

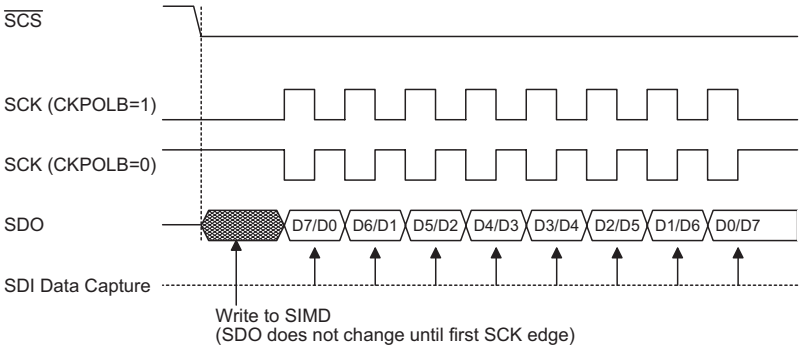
The TRF bit is the Transmit/Receive Complete flag and is set to 1 automatically when an SPI data transfer is completed, but must cleared to 0 by the application program. It can be used to generate an interrupt.

SPI Communication

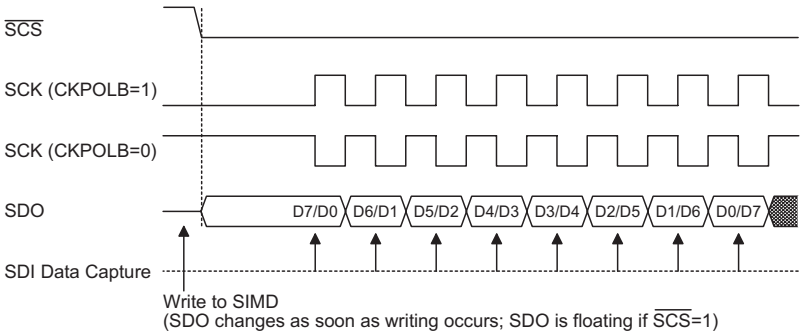
After the SPI interface is enabled by setting the SIMEN bit high, then in the Master Mode, when data is written to the SIMD register, transmission/reception will begin simultaneously. When the data transfer is complete, the TRF flag will be set automatically, but must be cleared using the application program. In the Slave Mode, when the clock signal from the master has been received, any data in the SIMD register will be transmitted and any data on the SDI pin will be shifted into the SIMD register. The master should output a $\overline{SCS}$ 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 $\overline{SCS}$ signal depending upon the configurations of the CKPOLB bit and CKEG bit. The accompanying timing diagram shows the relationship between the slave data and $\overline{SCS}$ signal for various configurations of the CKPOLB and CKEG bits. The SPI will continue to function even in the IDLE Mode.



SPI Master Mode Timing

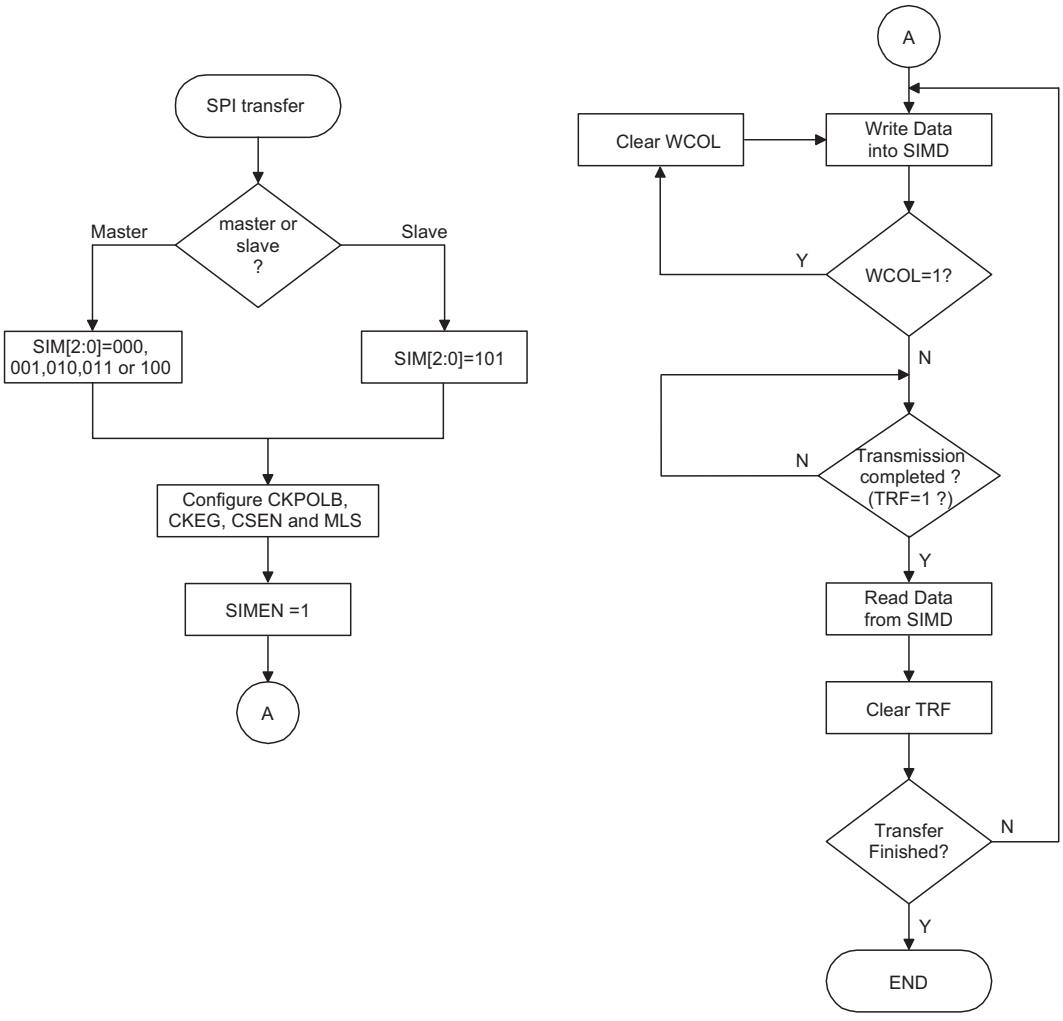


SPI Slave Mode Timing – CKEG=0



Note: For SPI slave mode, if SIMEN=1 and CSEN=0, SPI is always enabled and ignores the $\overline{SCS}$ level.

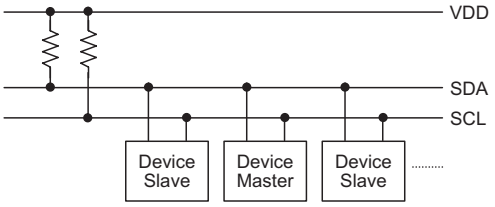
SPI Slave Mode Timing – CKEG=1



SPI 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.

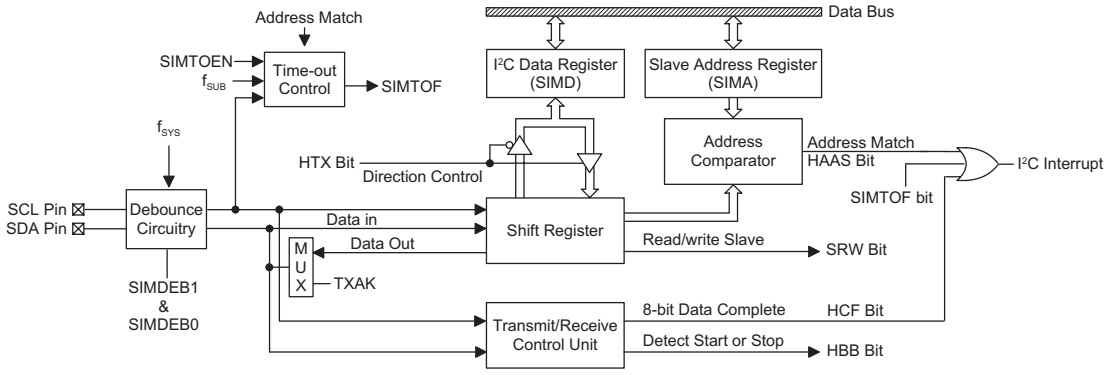


I²C Master Slave Bus Connection

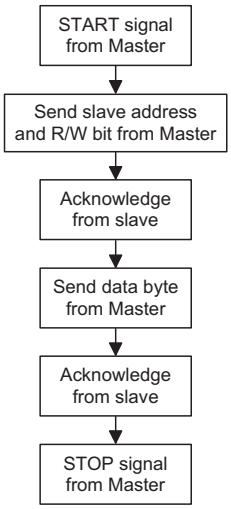
I²C interface Operation

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 operate in slave mode, there are two methods of transferring data on the I²C bus, the slave transmit mode and the slave receive mode.



I²C Block Diagram



The SIMDEB1 and SIMDEB0 bits determine the debounce time of the I²C interface. This uses the system clock to in effect add a debounce time to the external clock to reduce the possibility of glitches on the clock line causing erroneous operation. The debounce time, if selected, can be chosen to be either 2 or 4 system clocks. To achieve the required I²C data transfer speed, there exists a relationship between the system clock, f_{SYS}, and the I²C debounce time. For either the I²C Standard or Fast mode operation, users must take care of the selected system clock frequency and the configured debounce time to match the criterion shown in the following table.

I²C Debounce Time Selection	I²C Standard Mode (100kHz)	I²C Fast Mode (400kHz)
No Devounce	f_{SYS} > 2 MHz	f_{SYS} > 5 MHz
2 system clock debounce	f_{SYS} > 4 MHz	f_{SYS} > 10 MHz
4 system clock debounce	f_{SYS} > 8 MHz	f_{SYS} > 20 MHz

I²C Minimum f_{SYS} Frequency

I²C Registers

There are three control registers associated with the I²C bus, SIMC0, SIMC1 and SIMA, and one data register, SIMD. The SIMD register, which is shown in the above SPI section, is used to store the data being transmitted and received on the I²C bus. Note that the SIMA register also has the name SIMC2 which is used by the SPI function.

Register Name	Bit							
	7	6	5	4	3	2	1	0
SIMC0	SIM2	SIM1	SIM0	—	SIMDEB1	SIMDEB0	SIMEN	SIMICF
SIMC1	HCF	HAAS	HBB	HTX	TXAK	SRW	IAMWU	RXAK
SIMA	A6	A5	A4	A3	A2	A1	A0	—
SIMD	D7	D6	D5	D4	D3	D2	D1	D0
SIMTOC	SIMTOEN	SIMTOF	SIMTOS5	SIMTOS4	SIMTOS3	SIMTOS2	SIMTOS1	SIMTOS0

I²C Registers List

- **SIMD Register**

The SIMD register is used to store the data being transmitted and received. The same register is used by both the SPI and I²C functions. Before the device writes data to the I²C bus, the actual data to be transmitted must be placed in the SIMD register. After the data is received from the I²C bus, the device can read it from the SIMD register. Any transmission or reception of data from the I²C bus must be made via the SIMD register.

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

"x": unknown

- **SIMA Register**

The SIMA register is also used by the SPI interface but has the name SIMC2. The SIMA register is the location where the 7-bit slave address of the slave device is stored. Bits 7~1 of the SIMA register define the device slave address. Bit 0 is not defined.

When a master device, which is connected to the I²C bus, sends out an address, which matches the slave address in the SIMA register, the slave device will be selected. Note that the SIMA register is the same register address as SIMC2 which is used by the SPI interface.

Bit	7	6	5	4	3	2	1	0
Name	A6	A5	A4	A3	A2	A1	A0	—
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	x	x	x	x	x	x	x	x

"x": unknown

Bit 7~1 **A6~A0**: I²C slave address
A6~A0 is the I²C slave address bit 6 ~ bit 0

Bit 0 Undefined bit
The bit can be read or written by the application program.

There are also two control registers for the I²C interface, SIMC0 and SIMC1. The register SIMC0 is used to control the enable/disable function and to set the data transmission clock frequency.The SIMC1 register contains the relevant flags which are used to indicate the I²C communication status.

- **SIMC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	SIM2	SIM1	SIM0	—	SIMDEB1	SIMDEB0	SIMEN	SIMICF
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 **SIM2~SIM0**: SIM 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 CTM1 CCRP match frequency/2
101: SPI slave mode
110: I^2C slave mode
111: Non SIM function
These bits setup the overall operating mode of the SIM function. As well as selecting if the I^2C or SPI function, they are used to control the SPI Master/Slave selection and the SPI Master clock frequency. The SPI clock is a function of the system clock but can also be chosen to be sourced from CTM1. If the SPI Slave Mode is selected then the clock will be supplied by an external Master device.

Bit 4 Unimplemented, read as "0"

Bit 3~2 **SIMDEB1~SIMDEB0**: I^2C Debounce Time Selection
00: No debounce
01: 2 system clock debounce
1x: 4 system clock debounce

Bit 1 **SIMEN**: SIM Enable Control
0: Disable
1: Enable
The bit is the overall on/off control for the SIM interface. When the SIMEN bit is cleared to zero to disable the SIM interface, the SDI, SDO, SCK and $\overline{SCS}$, or SDA and SCL lines will lose their SPI or I^2C function and the SIM operating current will be reduced to a minimum value. When the bit is high the SIM interface is enabled. If the SIM is configured to operate as an SPI interface via the SIM2~SIM0 bits, the contents of the SPI control registers will remain at the previous settings when the SIMEN bit changes from low to high and should therefore be first initialised by the application program. If the SIM is configured to operate as an I^2C interface via the SIM2~SIM0 bits and the SIMEN bit changes from low to high, the contents of the I^2C control bits such as HTX and TXAK will remain at the previous settings and should therefore be first initialised by the application program while the relevant I^2C flags such as HCF, HAAS, HBB, SRW and RXAK will be set to their default states.

Bit 0 **SIMICF**: SIM SPI slave mode Incomplete Transfer Flag
Described elsewhere.

- **SIMC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	HCF	HAAS	HBB	HTX	TXAK	SRW	IAMWU	RXAK
R/W	R	R	R	R/W	R/W	R/W	R/W	R
POR	1	0	0	0	0	0	0	1

Bit 7 **HCF**: I²C Bus data transfer completion flag
0: Data is being transferred
1: Completion of an 8-bit data transfer
The HCF flag is the data transfer flag. This flag will be zero when data is being transferred. Upon completion of an 8-bit data transfer the flag will go high and an interrupt will be generated.

Bit 6 **HAAS**: I²C Bus data transfer completion flag
0: Not address match
1: Address match
The HAAS flag is the address match flag. This flag is used to determine if the slave device address is the same as the master transmit address. If the addresses match then this bit will be high, if there is no match then the flag will be low.

Bit 5 **HBB**: I²C Bus busy flag
0: I²C Bus is not busy
1: I²C Bus is busy
The HBB flag is the I²C busy flag. This flag will be "1" when the I²C 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 **HTX**: I²C slave device transmitter/receiver selection
0: Slave device is the receiver
1: Slave device is the transmitter

Bit 3 **TXAK**: I²C bus transmit acknowledge flag
0: Slave send acknowledge flag
1: Slave does not send acknowledge flag
The TXAK bit is the transmit acknowledge flag. After the slave device receipt of 8-bits of data, this bit will be transmitted to the bus on the 9th clock from the slave device. The slave device must always set TXAK bit to "0" before further data is received.

Bit 2 **SRW**: I²C slave read/write flag
0: Slave device should be in receive mode
1: Slave device should be in transmit mode
The SRW flag is the I²C Slave Read/Write flag. This flag determines whether the master device wishes to transmit or receive data from the I²C bus. When the transmitted address and slave address is match, that is when the HAAS flag is set high, the slave device will check the SRW flag to determine whether it should be in transmit mode or receive mode. If the SRW flag is high, the master is requesting to read data from the bus, so the slave device should be in transmit mode. When the SRW flag is zero, the master will write data to the bus, therefore the slave device should be in receive mode to read this data.

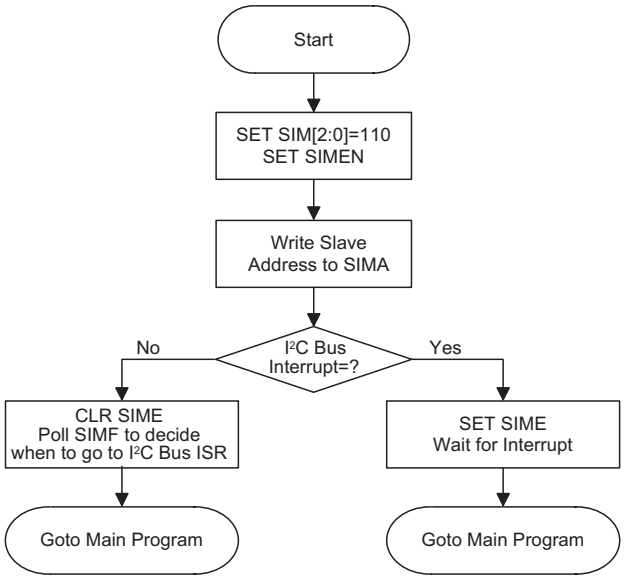
Bit 1 **IAMWU**: I²C Address Match Wake-Up control
0: Disable
1: Enable – must be cleared by the application program after wake-up
This bit should be set to 1 to enable the I²C address match wake up from the SLEEP or IDLE Mode. If the IAMWU bit has been set before entering either the SLEEP or IDLE mode to enable the I²C address match wake up, then this bit must be cleared by the application program after wake-up to ensure correction device operation.

Bit 0 **RXAK**: I²C bus receive acknowledge flag
0: Slave receives acknowledge flag
1: Slave does not receive acknowledge flag
The RXAK flag is the receiver acknowledge flag. When the RXAK flag is "0", it means that a acknowledge signal has been received at the 9th clock, after 8 bits of data have been transmitted. When the slave device in the transmit mode, the slave device checks the RXAK flag to determine if the master receiver wishes to receive the next byte. The slave transmitter will therefore continue sending out data until the RXAK flag is "1". When this occurs, the slave transmitter will release the SDA line to allow the master to send a STOP signal to release the I²C Bus.

I²C Bus Communication

Communication on the I²C 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 I²C bus, all devices on the bus will receive this signal and be notified of the imminent arrival of data on the bus. The first seven bits of the data will be the slave address with the first bit being the MSB. If the address of the slave device matches that of the transmitted address, the HAAS bit in the SIMC1 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 HAAS and SIMTOF bits to determine whether the interrupt source originates from an address match, 8-bit data transfer completion or I²C bus time-out occurrence. During a data transfer, note that after the 7-bit slave address has been transmitted, the following bit, which is the 8th bit, is the read/write bit whose value will be placed in the SRW bit. This bit will be checked by the slave device to determine whether to go into transmit or receive mode. Before any transfer of data to or from the I²C bus, the microcontroller must initialise the bus, the following are steps to achieve this:

- Step 1
  Select the SDA and SCL pin function by pin-shared registers
- Step 2
  Set the SIM2~SIM0 bits to "110" and SIMEN bit to "1" in the SIMC0 register to enable the I²C bus.
- Step 3
  Write the slave address of the device to the I²C bus address register SIMA.
- Step 4
  Set the SIM interrupt enable bit SIME of the interrupt control register to enable the SIM interrupt.



I²C Bus Start Signal

The START signal can only be generated by the master device connected to the I²C bus and not by the slave device. This START signal will be detected by all devices connected to the I²C bus. When detected, this indicates that the I²C bus is busy and therefore the HBB bit will be set. A START condition occurs when a high to low transition on the SDA line takes place when the SCL line remains high.

I²C Slave Address

The transmission of a START signal by the master will be detected by all devices on the I²C bus. To determine which slave device the master wishes to communicate with, the address of the slave device will be sent out immediately following the START signal. All slave devices, after receiving this 7-bit address data, will compare it with their own 7-bit slave address. If the address sent out by the master matches the internal address of the microcontroller slave device, then an internal I²C bus interrupt signal will be generated. The next bit following the address, which is the 8th bit, defines the read/write status and will be saved to the SRW bit of the SIMC1 register. The slave device will then transmit an acknowledge bit, which is a low level, as the 9th bit. The slave device will also set the status flag HAAS when the addresses match.

As an I²C bus interrupt can come from three sources, when the program enters the interrupt subroutine, the HAAS and SIMTOF bits should be examined to see whether the interrupt source has come from a matching slave address, the completion of a data byte transfer or the I²C bus time-out occurrence. When a slave address is matched, the devices must be placed in either the transmit mode and then write data to the SIMD register, or in the receive mode where it must implement a dummy read from the SIMD register to release the SCL line.

I²C Bus Read/Write Signal

The SRW bit in the SIMC1 register defines whether the slave device wishes to read data from the I²C bus or write data to the I²C bus. The slave device should examine this bit to determine if it is to be a transmitter or a receiver. If the SRW flag is “1” then this indicates that the master device wishes to read data from the I²C bus, therefore the slave device must be setup to send data to the I²C bus as a transmitter. If the SRW flag is “0” then this indicates that the master wishes to send data to the I²C bus, therefore the slave device must be setup to read data from the I²C bus as a receiver.

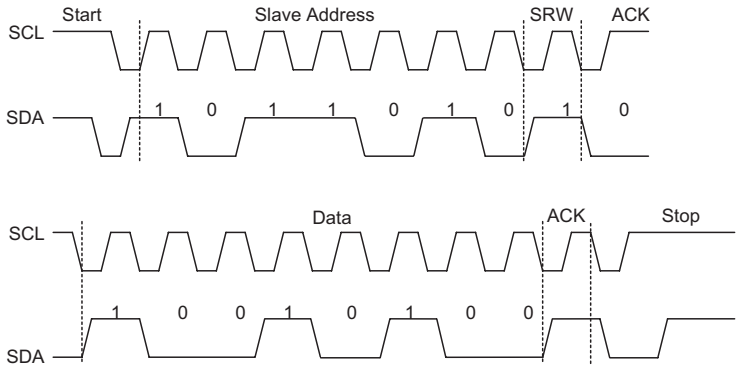
I²C Bus Slave Address Acknowledge Signal

After the master has transmitted a calling address, any slave device on the I²C bus, whose own internal address matches the calling address, must generate an acknowledge signal. The acknowledge signal will inform the master that a slave device has accepted its calling address. If no acknowledge signal is received by the master then a STOP signal must be transmitted by the master to end the communication. When the HAAS flag is high, the addresses have matched and the slave device must check the SRW flag to determine if it is to be a transmitter or a receiver. If the SRW flag is high, the slave device should be setup to be a transmitter so the HTX bit in the SIMC1 register should be set to "1". If the SRW flag is low, then the microcontroller slave device should be setup as a receiver and the HTX bit in the SIMC1 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 SDA line to allow the master to send a STOP signal to release the I²C Bus. The corresponding data will be stored in the SIMD register. If setup as a transmitter, the slave device must first write the data to be transmitted into the SIMD register. If setup as a receiver, the slave device must read the transmitted data from the SIMD register.

When the slave receiver receives the data byte, it must generate an acknowledge bit, known as TXAK, on the 9th clock. The slave device, which is setup as a transmitter will check the RXAK bit in the SIMC1 register to determine if it is to send another data byte, if not then it will release the SDA line and await the receipt of a STOP signal from the master.

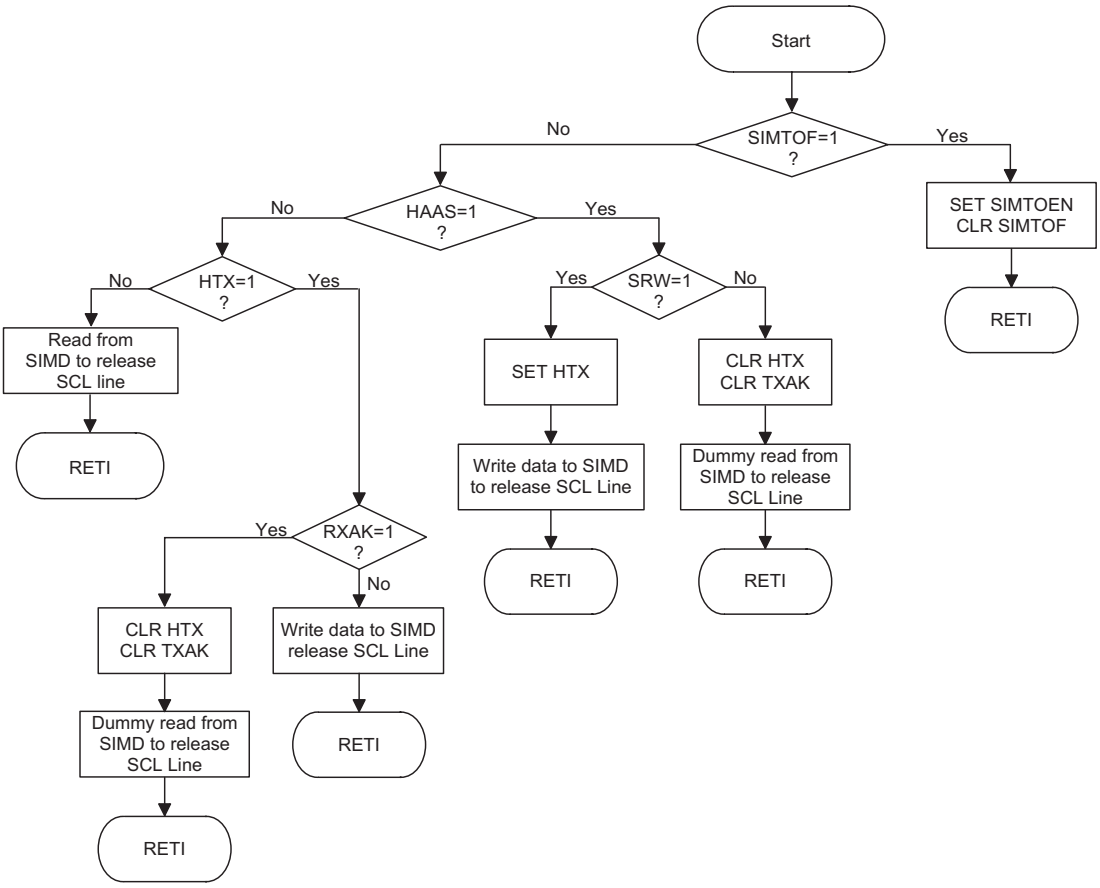


S=Start (1 bit)
SA=Slave Address (7 bits)
SR=SRW bit (1 bit)
M=Slave device send acknowledge bit (1 bit)
D=Data (8 bits)
A=ACK (RXAK bit for transmitter, TXAK bit for receiver 1 bit)
P=Stop (1 bit)

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 devices must be placed in either the transmit mode and then write data to the SIMD register, or in the receive mode where it must implement a dummy read from the SIMD register to release the SCL line.

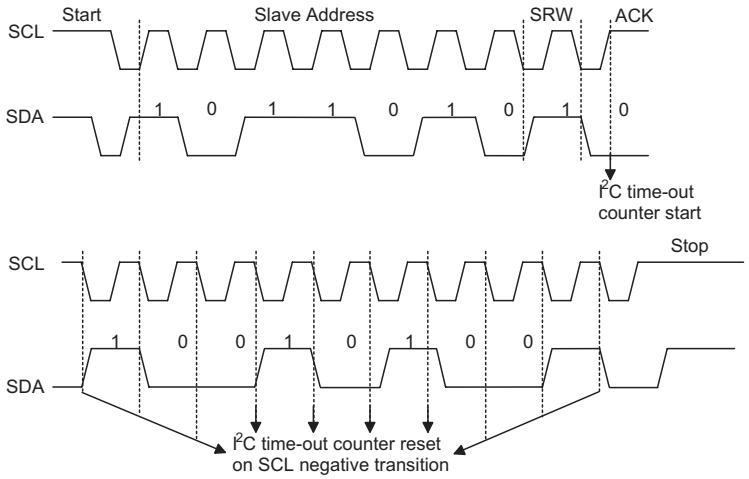
I²C Communication Timing Diagram



I²C Bus ISR Flow Chart

I²C Time-out Control

In order to reduce the I²C lockup problem due to reception of erroneous clock sources, a time-out function is provided. If the clock source connected to the I²C bus is not received for a while, then the I²C circuitry and registers will be reset after a certain time-out period. The time-out counter starts to count on an I²C bus "START" & "address match"condition, and is cleared by an SCL falling edge. Before the next SCL falling edge arrives, if the time elapsed is greater than the time-out period specified by the SIMTOC register, then a time-out condition will occur. The time-out function will stop when an I²C "STOP" condition occurs.



I²C Time-out

When an I²C time-out counter overflow occurs, the counter will stop and the SIMTOEN bit will be cleared to zero and the SIMTOF bit will be set high to indicate that a time-out condition has occurred. The time-out condition will also generate an interrupt which uses the I²C interrrupt vector. When an I²C time-out occurs, the I²C internal circuitry will be reset and the registers will be reset into the following condition:

Register	After I²C Time-out
SIMD, SIMA, SIMC0	No change
SIMC1	Reset to POR condition

I²C Register after Time-out

The SIMTOF flag can be cleared by the application program. There are 64 time-out period selections which can be selected using the SIMTOS bits in the SIMTOC register. The time-out duration is calculated by the formula: ((1~64) × (32/f_{SUB})). This gives a time-out period which ranges from about 1ms to 64ms.

- **SIMTOC Register**

Bit	7	6	5	4	3	2	1	0
Name	SIMTOEN	SIMTOF	SIMTOS5	SIMTOS4	SIMTOS3	SIMTOS2	SIMTOS1	SIMTOS0
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 **SIMTOEN**: SIM I²C Time-out control
0: Disable
1: Enable

Bit 6 **SIMTOF**: SIM I²C Time-out flag
0: No time-out occurred
1: Time-out occurred

Bit 5~0 **SIMTOS5~SIMTOS0**: SIM I²C Time-out period selection
I²C Time-out clock source is $f_{SUB}/32$

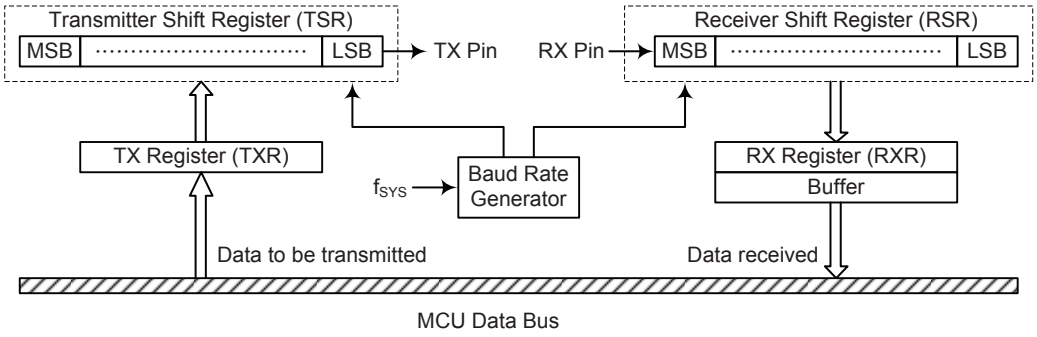
I²C Time-out period is equal to (SIMTOS [5:0] + 1) × $\frac{32}{f_{SUB}}$

UART Interface

The HT66F0082 device contains an integrated full-duplex asynchronous serial communications UART interface that enables communication with external devices that contain a serial interface. The UART function has many features and can transmit and receive data serially by transferring a frame of data with eight or nine data bits per transmission as well as being able to detect errors when the data is overwritten or incorrectly framed. The UART function possesses its own internal interrupt which can be used to indicate when a reception occurs or when a transmission terminates.

The integrated UART function contains the following features:

- Full-duplex, Universal Asynchronous Receiver and Transmitter (UART) communication
- 8 or 9 bits character length
- Even, odd or no parity options
- One or two stop bits
- Baud rate generator with 8-bit prescaler
- Parity, framing, noise and overrun error detection
- Support for interrupt on address detect (last character bit=1)
- Transmitter and receiver enabled independently
- 2-byte Deep FIFO Receive Data Buffer
- Transmit and receive interrupts
- Interrupts can be initialized by the following conditions:
  - Transmitter Empty
  - Transmitter Idle
  - Receiver Full
  - Receiver Overrun
  - Address Mode Detect
  - RX pin wake-up interrupt (RX enable, RX falling edge)



UART Data Transfer Block Diagram

UART External Pin Interface

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

UART Data Transfer Scheme

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

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

It should be noted that the actual register for data transmission and reception, although referred to in the text, and in application programs, as separate TXR and RXR registers, only exists as a single shared register in the Data Memory. This shared register known as the TXR_RXR register is used for both data transmission and data reception.

UART Status and Control Registers

There are five control registers associated with the UART function. The USR, UCR1 and UCR2 registers control the overall function of the UART, while the BRG register controls the Baud rate. The actual data to be transmitted and received on the serial interface is managed through the TXR_RXR data register.

Register Name	Bit							
	7	6	5	4	3	2	1	0
USR	PERR	NF	FERR	OERR	RIDLE	RXIF	TIDLE	TXIF
UCR1	UARTEN	BNO	PREN	PRT	STOPS	TXBRK	RX8	TX8
UCR2	TXEN	RXEN	BRGH	ADDEN	WAKE	RIE	TIIE	TEIE
TXR_RXR	D7	D6	D5	D4	D3	D2	D1	D0
BRG	D7	D6	D5	D4	D3	D2	D1	D0

UART Register List

USR register

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

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

Bit 7 **PERR**: Parity error flag
0: No parity error is detected
1: Parity error is detected

The PERR flag is the parity error flag. When this read only flag is "0", it indicates a parity error has not been detected. When the flag is "1", it indicates that the parity of the received word is incorrect. This error flag is applicable only if Parity mode (odd or even) is selected. The flag can also be cleared by a software sequence which involves a read to the status register USR followed by an access to the RXR data register.

Bit 6 **NF**: Noise flag
0: No noise is detected
1: Noise is detected

The NF flag is the noise flag. When this read only flag is "0", it indicates no noise condition. When the flag is "1", it indicates that the UART has detected noise on the receiver input. The NF flag is set during the same cycle as the RXIF flag but will not be set in the case of as overrun. The NF flag can be cleared by a software sequence which will involve a read to the status register USR followed by an access to the RXR data register.

Bit 5 **FERR**: Framing error flag
0: No framing error is detected
1: Framing error is detected

The FERR flag is the framing error flag. When this read only flag is "0", it indicates that there is no framing error. When the flag is "1", it indicates that a framing error has been detected for the current character. The flag can also be cleared by a software sequence which will involve a read to the status register USR followed by an access to the RXR data register.

Bit 4 **OERR**: Overrun error flag
0: No overrun error is detected
1: Overrun error is detected

The OERR flag is the overrun error flag which indicates when the receiver buffer has overflowed. When this read only flag is "0", it indicates that there is no overrun error. When the flag is "1", it indicates that an overrun error occurs which will inhibit further transfers to the RXR receive data register. The flag is cleared by a software sequence, which is a read to the status register USR followed by an access to the RXR data register.

Bit 3 **RIDLE**: Receiver status
0: Data reception is in progress (data being received)
1: No data reception is in progress (receiver is idle)

The RIDLE flag is the receiver status flag. When this read only flag is "0", it indicates that the receiver is between the initial detection of the start bit and the completion of the stop bit. When the flag is "1", it indicates that the receiver is idle. Between the completion of the stop bit and the detection of the next start bit, the RIDLE bit is "1" indicating that the UART receiver is idle and the RX pin stays in logic high condition.

Bit 2 **RXIF**: Receive RXR data register status
0: RXR data register is empty
1: RXR data register has available data
The RXIF flag is the receive data register status flag. When this read only flag is "0", it indicates that the RXR read data register is empty. When the flag is "1", it indicates that the RXR read data register contains new data. When the contents of the shift register are transferred to the RXR register, an interrupt is generated if RIE=1 in the UCR2 register. If one or more errors are detected in the received word, the appropriate receive-related flags NF, FERR, and/or PERR are set within the same clock cycle. The RXIF flag is cleared when the USR register is read with RXIF set, followed by a read from the RXR register, and if the RXR register has no data available.

Bit 1 **TIDLE**: Transmission idle
0: Data transmission is in progress (data being transmitted)
1: No data transmission is in progress (transmitter is idle)
The TIDLE flag is known as the transmission complete flag. When this read only flag is "0", it indicates that a transmission is in progress. This flag will be set to "1" when the TXIF flag is "1" and when there is no transmit data or break character being transmitted. When TIDLE is equal to "1", the TX pin becomes idle with the pin state in logic high condition. The TIDLE flag is cleared by reading the USR register with TIDLE set and then writing to the TXR register. The flag is not generated when a data character or a break is queued and ready to be sent.

Bit 0 **TXIF**: Transmit TXR data register status
0: Character is not transferred to the transmit shift register
1: Character has transferred to the transmit shift register (TXR data register is empty)
The TXIF flag is the transmit data register empty flag. When this read only flag is "0", it indicates that the character is not transferred to the transmitter shift register. When the flag is "1", it indicates that the transmitter shift register has received a character from the TXR data register. The TXIF flag is cleared by reading the UART status register (USR) with TXIF set and then writing to the TXR data register. Note that when the TXEN bit is set, the TXIF flag bit will also be set since the transmit data register is not yet full.

UCR1 register

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

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

"x" unknown

Bit 7 **UARTEN**: UART function enable control
0: Disable UART. TX and RX pins are in a floating state
1: Enable UART. TX and RX pins function as UART pins
The UARTEN bit is the UART enable bit. When this bit is equal to "0", the UART will be disabled and the RX pin as well as the TX pin will be in a floating state. When the bit is equal to "1", the UART will be enabled and the TX and RX pins will function as defined by the TXEN and RXEN enable control bits.
When the UART is disabled, it will empty the buffer so any character remaining in the buffer will be discarded. In addition, the value of the baud rate counter will be reset. If the UART is disabled, all error and status flags will be reset. Also the TXEN, RXEN, TXBRK, RXIF, OERR, FERR, PERR and NF bits will be cleared, while the TIDLE, TXIF and RIDLE bits will be set. Other control bits in UCR1, UCR2 and BRG registers will remain unaffected. If the UART is active and the UARTEN bit is

cleared, all pending transmissions and receptions will be terminated and the module will be reset as defined above. When the UART is re-enabled, it will restart in the same configuration.

Bit 6 **BNO**: Number of data transfer bits selection
0: 8-bit data transfer
1: 9-bit data transfer

This bit is used to select the data length format, which can have a choice of either 8-bit or 9-bit format. When this bit is equal to "1", a 9-bit data length format will be selected. If the bit is equal to "0", then an 8-bit data length format will be selected. If 9-bit data length format is selected, then bits RX8 and TX8 will be used to store the 9th bit of the received and transmitted data respectively.

Note: 1. If BNO=1 (9-bit data transfer), parity function is enabled, the 9th bit of data is the parity bit which will not be transferred to RX8.
2. If BNO=0 (8-bit data transfer), parity function is enabled, the 8th bit of data is the parity bit which will not be transferred to RX7.

Bit 5 **PREN**: Parity function enable control
0: Parity function is disabled
1: Parity function is enabled

This is the parity enable bit. When this bit is equal to "1", the parity function will be enabled. If the bit is equal to "0", then the parity function will be disabled.

Bit 4 **PRT**: Parity type selection bit
0: Even parity for parity generator
1: Odd parity for parity generator

This bit is the parity type selection bit. When this bit is equal to "1", odd parity type will be selected. If the bit is equal to "0", then even parity type will be selected.

Bit 3 **STOPS**: Number of stop bits selection
0: One stop bit format is used
1: Two stop bits format is used

This bit determines if one or two stop bits are to be used for the TX pin. When this bit is equal to "1", two stop bits are used. If this bit is equal to "0", then only one stop bit is used.

Bit 2 **TXBRK**: Transmit break character
0: No break character is transmitted
1: Break characters transmit

The TXBRK bit is the Transmit Break Character bit. When this bit is "0", there are no break characters and the TX pin operates normally. When the bit is "1", there are transmit break characters and the transmitter will send logic zeros. When this bit is equal to "1", after the buffered data has been transmitted, the transmitter output is held low for a minimum of a 13-bit length and until the TXBRK bit is reset.

Bit 1 **RX8**: Receive data bit 8 for 9-bit data transfer format (read only)

This bit is only used if 9-bit data transfers are used, in which case this bit location will store the 9th bit of the received data known as RX8. The BNO bit is used to determine whether data transfers are in 8-bit or 9-bit format.

Bit 0 **TX8**: Transmit data bit 8 for 9-bit data transfer format (write only)

This bit is only used if 9-bit data transfers are used, in which case this bit location will store the 9th bit of the transmitted data known as TX8. The BNO bit is used to determine whether data transfers are in 8-bit or 9-bit format.

UCR2 register

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

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

Bit 7 **TXEN**: UART Transmitter enabled control
0: UART transmitter is disabled
1: UART transmitter is enabled
The bit named TXEN is the Transmitter Enable Bit. When this bit is equal to "0", the transmitter will be disabled with any pending data transmissions being aborted. In addition the buffers will be reset. In this situation the TX pin will be set in a floating state.
If the TXEN bit is equal to "1" and the UARTEN bit is also equal to "1", the transmitter will be enabled and the TX pin will be controlled by the UART. Clearing the TXEN bit during a transmission will cause the data transmission to be aborted and will reset the transmitter. If this situation occurs, the TX pin will be set in a floating state.

Bit 6 **RXEN**: UART Receiver enabled control
0: UART receiver is disabled
1: UART receiver is enabled
The bit named RXEN is the Receiver Enable Bit. When this bit is equal to "0", the receiver will be disabled with any pending data receptions being aborted. In addition the receive buffers will be reset. In this situation the RX pin will be set in a floating state. If the RXEN bit is equal to "1" and the UARTEN bit is also equal to "1", the receiver will be enabled and the RX pin will be controlled by the UART. Clearing the RXEN bit during a reception will cause the data reception to be aborted and will reset the receiver. If this situation occurs, the RX pin will be set in a floating state.

Bit 5 **BRGH**: Baud Rate speed selection
0: Low speed baud rate
1: High speed baud rate
The bit named BRGH selects the high or low speed mode of the Baud Rate Generator. This bit, together with the value placed in the baud rate register BRG, controls the Baud Rate of the UART. If this bit is equal to "1", the high speed mode is selected. If the bit is equal to "0", the low speed mode is selected.

Bit 4 **ADDEN**: Address detect function enable control
0: Address detection function is disabled
1: Address detection function is enabled
The bit named ADDEN is the address detect function enable control bit. When this bit is equal to "1", the address detect function is enabled. When it occurs, if the 8th bit, which corresponds to RX7 if BNO=0 or the 9th bit, which corresponds to RX8 if BNO=1, has a value of "1", then the received word will be identified as an address, rather than data. If the corresponding interrupt is enabled, an interrupt request will be generated each time the received word has the address bit set, which is the 8th or 9th bit depending on the value of BNO. If the address bit known as the 8th or 9th bit of the received word is "0" with the address detect function being enabled, an interrupt will not be generated and the received data will be discarded.

Bit 3 **WAKE**: RX pin falling edge wake-up function enable control
0: RX pin wake-up function is disabled
1: RX pin wake-up function is enabled
This bit enables or disables the receiver wake-up function. If this bit is equal to "1" and the device is in the IDLE0 or SLEEP mode, a falling edge on the RX input pin will wake-up the device. If this bit is equal to "0" and the device is in the IDLE or SLEEP mode, any edge transitions on the RX pin will not wake-up the device.

Bit 2 **RIE**: Receiver interrupt enable control
0: Receiver related interrupt is disabled
1: Receiver related interrupt is enabled
This bit enables or disables the receiver interrupt. If this bit is equal to "1" and when the receiver overrun flag OERR or receive data available flag RXIF is set, the UART interrupt request flag will be set. If this bit is equal to "0", the UART interrupt request flag will not be influenced by the condition of the OERR or RXIF flags.

Bit 1 **TIIE**: Transmitter Idle interrupt enable control
0: Transmitter idle interrupt is disabled
1: Transmitter idle interrupt is enabled
This bit enables or disables the transmitter idle interrupt. If this bit is equal to "1" and when the transmitter idle flag TIDLE is set, due to a transmitter idle condition, the UART interrupt request flag will be set. If this bit is equal to "0", the UART interrupt request flag will not be influenced by the condition of the TIDLE flag.

Bit 0 **TEIE**: Transmitter Empty interrupt enable control
0: Transmitter empty interrupt is disabled
1: Transmitter empty interrupt is enabled
This bit enables or disables the transmitter empty interrupt. If this bit is equal to "1" and when the transmitter empty flag TXIF is set, due to a transmitter empty condition, the UART interrupt request flag will be set. If this bit is equal to "0", the UART interrupt request flag will not be influenced by the condition of the TXIF flag.

TXR_RXR register

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

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

"x" unknown

Bit 7~0 **D7~D0**: UART Transmit/Receive Data bit 7 ~ bit 0

Baud Rate Generator

To setup the speed of the serial data communication, the UART function contains its own dedicated baud rate generator. The baud rate is controlled by its own internal free running 8-bit timer, the period of which is determined by two factors. The first of these is the value placed in the baud rate register BRG and the second is the value of the BRGH bit with the control register UCR2. The BRGH bit decides if the baud rate generator is to be used in a high speed mode or low speed mode, which in turn determines the formula that is used to calculate the baud rate. The value in the BRG register, N, which is used in the following baud rate calculation formula determines the division factor. Note that N is the decimal value placed in the BRG register and has a range of between 0 and 255.

UCR2 BRGH Bit	0	1
Baud Rate (BR)	$\frac{f_{SYS}}{[64(N+1)]}$	$\frac{f_{SYS}}{[16(N+1)]}$

By programming the BRGH bit which allows selection of the related formula and programming the required value in the BRG register, the required baud rate can be setup. Note that because the actual baud rate is determined using a discrete value, N, placed in the BRG register, there will be an error associated between the actual and requested value. The following example shows how the BRG register value N and the error value can be calculated.

BRG Register

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

"x" unknown

Bit 7~0 **D7~D0**: Baud Rate values
By programming the BRGH bit in UCR2 Register which allows selection of the related formula described above and programming the required value in the BRG register, the required baud rate can be setup.

Calculating the Baud Rate and error values

For a clock frequency of 4MHz, and with BRGH set to "0" determine the BRG register value N, the actual baud rate and the error value for a desired baud rate of 4800.

From the above table the desired baud rate $BR = \frac{f_{SYS}}{[64(N+1)]}$

Re-arranging this equation gives $N = \frac{f_{SYS}}{(BR \times 64)} - 1$

Giving a value for $N = \frac{4000000}{(4800 \times 64)} - 1 = 12.0208$

To obtain the closest value, a decimal value of 12 should be placed into the BRG register. This gives an actual or calculated baud rate value of $BR = \frac{4000000}{[64(12+1)]} = 4808$

Therefore the error is equal to $\frac{4808-4800}{4800} = 0.16\%$

UART Setup and Control

For data transfer, the UART function utilizes a non-return-to-zero, more commonly known as NRZ, format. This is composed of one start bit, eight or nine data bits, and one or two stop bits. Parity is supported by the UART hardware, and can be setup to be even, odd or no parity. For the most common data format, 8 data bits along with no parity and one stop bit, denoted as 8, N, 1, is used as the default setting, which is the setting at power-on. The number of data bits and stop bits, along with the parity, are setup by programming the corresponding BNO, PRT, PREN, and STOPS bits in the UCR1 register. The baud rate used to transmit and receive data is setup using the internal 8-bit baud rate generator, while the data is transmitted and received LSB first. Although the UART transmitter and receiver are functionally independent, they both use the same data format and baud rate. In all cases stop bits will be used for data transmission.

Enabling/disabling the UART interface

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

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

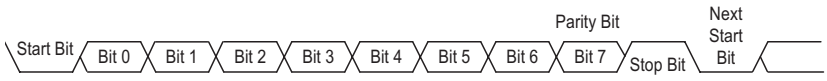
Data, parity and stop bit selection

The format of the data to be transferred is composed of various factors such as data bit length, parity on/off, parity type, address bits and the number of stop bits. These factors are determined by the setup of various bits within the UCR1 register. The BNO bit controls the number of data bits which can be set to either 8 or 9, the PRT bit controls the choice of odd or even parity, the PREN bit controls the parity on/off function and the STOPS bit decides whether one or two stop bits are to be used. The following table shows various formats for data transmission. The address bit identifies the frame as an address character. The number of stop bits, which can be either one or two, is independent of the data length and are only to be used for Transmitter. There is only one stop bit for Receiver.

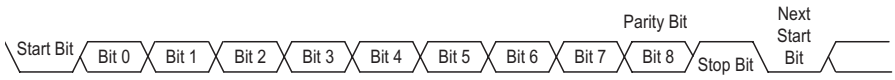
Start Bit	Data Bits	Address Bits	Parity Bits	Stop Bit
Example of 8-bit Data Formats				
1	8	0	0	1
1	7	0	1	1
1	7	1	0	1
Example of 9-bit Data Formats				
1	9	0	0	1
1	8	0	1	1
1	8	1	0	1

Transmitter Receiver Data Format

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



8-Bit Data Format



9-Bit Data Format

UART Transmitter

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

Transmitting data

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

- Make the correct selection of the BNO, PRT, PREN and STOPS bits to define the required word length, parity type and number of stop bits.
- Setup the BRG register to select the desired baud rate.
- Set the TXEN bit to ensure that the UART transmitter is enabled and the TX pin is used as a UART transmitter pin.
- Access the USR register and write the data that is to be transmitted into the TXR register. Note that this step will clear the TXIF bit.

This sequence of events can now be repeated to send additional data. It should be noted that when TXIF is “0”, data will be inhibited from being written to the TXR register. Clearing the TXIF flag is always achieved using the following software sequence:

- A USR register access
- A TXR register write execution

The read-only TXIF flag is set by the UART hardware and if set indicates that the TXR register is empty and that other data can now be written into the TXR register without overwriting the previous data. If the TEIE bit is set then the TXIF flag will generate an interrupt. During a data transmission, a write instruction to the TXR register will place the data into the TXR register, which will be copied to the shift register at the end of the present transmission. When there is no data transmission in progress, a write instruction to the TXR register will place the data directly into the shift register, resulting in the commencement of data transmission, and the TXIF bit being immediately set. When a frame transmission is complete, which happens after stop bits are sent or after the break frame, the TIDLE bit will be set. To clear the TIDLE bit the following software sequence is used:

- A USR register access
- A TXR register write execution

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

Transmitting break

If the TXBRK bit is set then break characters will be sent on the next transmission. Break character transmission consists of a start bit, followed by 13×N "0" bits and stop bits, where N=1, 2, etc. If a break character is to be transmitted then the TXBRK bit must be first set by the application program and then cleared to generate the stop bits. Transmitting a break character will not generate a transmit interrupt. Note that a break condition length is at least 13 bits long. If the TXBRK bit is continually kept at a logic high level then the transmitter circuitry will transmit continuous break characters. After the application program has cleared the TXBRK bit, the transmitter will finish transmitting the last break character and subsequently send out one or two stop bits. The automatic logic highs at the end of the last break character will ensure that the start bit of the next frame is recognized.

UART Receiver

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

Receiving data

When the UART receiver is receiving data, the data is serially shifted in on the external RX input pin to the shift register, with the least significant bit LSB first. The RXR register is a two byte deep FIFO data buffer, where two bytes can be held in the FIFO while a third byte can continue to be received. Note that the application program must ensure that the data is read from RXR before the third byte has been completely shifted in, otherwise this third byte will be discarded and an overrun error OERR will be subsequently indicated. The steps to initiate a data transfer can be summarized as follows:

- Make the correct selection of BNO, PRT and PREN bits to define the word length and parity type.
- Setup the BRG register to select the desired baud rate.
- Set the RXEN bit to ensure that the UART receiver is enabled and the RX pin is used as a UART receiver pin.

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

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

- The RXIF bit in the USR register will be set when RXR register has data available, at least one character can be read.
- When the contents of the shift register have been transferred to the RXR register and if the RIE bit is set, then an interrupt will be generated.
- If during reception, a frame error, noise error, parity error, or an overrun error has been detected, then the error flags can be set.

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

- A USR register access
- An RXR register read execution

Receiving break

Any break character received by the UART will be managed as a framing error. The receiver will count and expect a certain number of bit times as specified by the values programmed into the BNO and one STOPS bit. If the break is much longer than 13 bit times, the reception will be considered as complete after the number of bit times specified by BNO and one STOP bit. The RXIF bit is set, FERR is set, zeros are loaded into the receive data register, interrupts are generated if appropriate and the RIDLE bit is set. If a long break signal has been detected and the receiver has received a start bit, the data bits and the invalid stop bit, which sets the FERR flag, the receiver must wait for a valid stop bit before looking for the next start bit. The receiver will not make the assumption that the break condition on the line is the next start bit. A break is regarded as a character that contains only zeros with the FERR flag set. The break character will be loaded into the buffer and no further data will be received until stop bits are received. It should be noted that the RIDLE read only flag will go high when the stop bits have not yet been received. The reception of a break character on the UART registers will result in the following:

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

Idle status

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

Receiver interrupt

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

Managing Receiver Errors

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

Overrun Error – OERR flag

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

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

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

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

Noise Error – NF

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

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

Note that the NF flag is reset by a USR register read operation followed by an RXR register read operation.

Framing Error – FERR

The read only framing error flag, FERR, in the USR register, is set if a zero is detected instead of stop bits. If two stop bits are selected, only the first stop bit is detected, it must be high. If the first stop bit is low, the FERR flag will be set. The FERR flag is buffered along with the received data and is cleared on any reset.

Parity Error – PERR

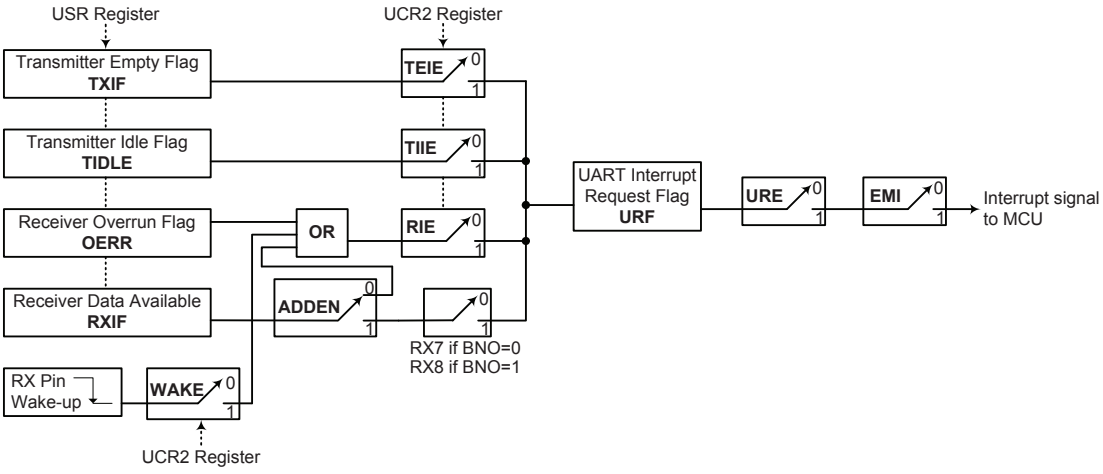
The read only parity error flag, PERR, in the USR register, is set if the parity of the received word is incorrect. This error flag is only applicable if the parity is enabled, PREN bit is “1”, and if the parity type, odd or even is selected. The read only PERR flag is buffered along with the received data bytes. It is cleared on any reset. It should be noted that the FERR and PERR flags are buffered along with the corresponding word and should be read before reading the data word.

UART Module Interrupt Structure

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

The address detect condition, which is also a UART interrupt source, does not have an associated flag, but will generate a UART interrupt when an address detect condition occurs if its function is enabled by setting the ADDEN bit in the UCR2 register. An RX pin wake-up, which is also a UART interrupt source, does not have an associated flag, but will generate a UART interrupt if the microcontroller is woken up from IDLE0 or SLEEP mode by a falling edge on the RX pin, if the WAKE and RIE bits in the UCR2 register are set. Note that in the event of an RX wake-up interrupt occurring, there will be a certain period of delay, commonly known as the System Start-up Time, for the oscillator to restart and stabilize before the system resumes normal operation.

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



UART Interrupt Structure

Address Detect Mode

Setting the Address Detect Mode bit, ADDEN, in the UCR2 register, enables this special mode. If this bit is enabled then an additional qualifier will be placed on the generation of a Receiver Data Available interrupt, which is requested by the RXIF flag. If the ADDEN bit is "1", then when data is available, an interrupt will only be generated, if the highest received bit has a high value. Note that the related interrupt enable control bit and the EMI bit must also be enabled for correct interrupt generation. This highest address bit is the 9th bit if BNO bit is "1" or the 8th bit if BNO bit is "0". If this bit is high, then the received word will be defined as an address rather than data. A Data Available interrupt will be generated every time the last bit of the received word is set. If the ADDEN bit is "0", then a Receiver Data Available interrupt will be generated each time the RXIF flag is set, irrespective of the data last bit status. The address detect mode and parity enable are mutually exclusive functions. Therefore if the address detect mode is enabled, then to ensure correct operation, the parity function should be disabled by resetting the parity enable bit PREN to zero.

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

ADDEN Bit Function

UART Power Down and Wake-up

When the the device system clock is switched off, the UART will cease to function. If the device executes the "HALT" instruction and switches off the system clock while a transmission is still in progress, then the transmission will be paused until the UART clock source derived from the microcontroller is activated. In a similar way, if the device executes the "HALT" instruction and switches off the system clock while receiving data, then the reception of data will likewise be paused. When the device enters the IDLE or SLEEP Mode, note that the USR, UCR1, UCR2, transmit and receive registers, as well as the BRG register will not be affected. It is recommended to make sure first that the UART data transmission or reception has been finished before the microcontroller enters the IDLE or SLEEP mode.

The UART function contains a receiver RX pin wake-up function, which is enabled or disabled by the WAKE bit in the UCR2 register. If this bit, along with the UART enable bit, UARTEN, the receiver enable bit, RXEN and the receiver interrupt bit, RIE, are all set before the device enters the IDLE0 or SLEEP Mode, then a falling edge on the RX pin will wake up the device from the IDLE0 or SLEEP Mode. Note that as it takes certain system clock cycles after a wake-up, before normal microcontroller operation resumes, any data received during this time on the RX pin will be ignored.

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

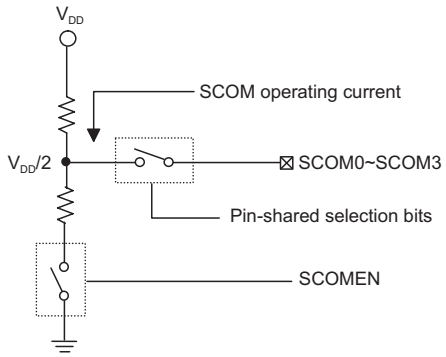
LCD SCOM function

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 I/O ports. The LCD signals are generated using the application program.

LCD Operation

An external LCD panel can be driven using this device by configuring the I/O pins as common 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. The LCD SCOMn pin is selected to be used for LCD driving by the corresponding pin-shared function selection bits. Note that the Port Control register does not need to first setup the pins as outputs to enable the LCD driver operation.



LCD Bias Current Control

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

SCOMC Register

Bit	7	6	5	4	3	2	1	0
Name	—	ISEL1	ISEL0	SCOMEN	—	—	—	—
R/W	—	R/W	R/W	R/W	—	—	—	—
POR	—	0	0	0	—	—	—	—

Bit 7 Unimplemented, read as "0"

Bit 6~5 **ISEL1~ISEL0**: Select resistor for R type LCD bias current (V_{DD}=5V)
00: 2×100kΩ (1/2 Bias), I_{BIAS} = 25μA
01: 2×50kΩ (1/2 Bias), I_{BIAS} = 50μA
10: 2×25kΩ (1/2 Bias), I_{BIAS} = 100μA
11: 2×12.5kΩ (1/2 Bias), I_{BIAS} = 200μA

Bit 4 **SCOMEN**: LCD control bit
0: Disable
1: Enable
When SCOMEN is set, it will turn on the DC path of resistor to generate 1/2 V_{DD} bias voltage.

Bit 3~0 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 devices contain several external interrupts and internal interrupts functions. The external interrupt is generated by the action of the external INT0~INT1 pins, while the internal interrupts are generated by various internal functions such as the TMs, Time Base, EEPROM, SIM, UART and the A/D converter.

Interrupt Registers

Overall interrupt control, which basically means the setting of request flags when certain microcontroller conditions occur and the setting of interrupt enable bits by the application program, is controlled by a series of registers, located in the Special Purpose Data Memory, as shown in the accompanying table. The number of registers depends upon the device chosen but fall into three categories. The first is the INTC0~INTC2 registers which setup the primary interrupts, the second is the MFI0~MFI2 registers which setup the Multi-function interrupts. Finally there is an INTEG register to setup the external interrupt trigger edge type.

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

Function	Enable Bit	Request Flag	Notes
Global	EMI	—	—
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
EEPROM	DEE	DEF	—
CTM	CTMAnE	CTMAnF	n=0 or 1
	CTMPnE	CTMPnF	
PTM	PTMAnE	PTMAnF	n=0~3
	PTMPnE	PTMPnF	
SIM	SIME	SIMF	—
UART	URE	URF	—

Interrupt Register Bit Naming Conventions

Register Name	Bit							
	7	6	5	4	3	2	1	0
INTEG	—	—	—	—	INT1S1	INT1S0	INT0S1	INT0S0
INTC0	—	TB1F	TB0F	INT0F	TB1E	TB0E	INT0E	EMI
INTC1	INT1F	ADF	DEF	MF0F	INT1E	ADE	DEE	MF0E
INTC2 (HT66F0042)	—	SIMF	MF2F	MF1F	—	SIME	MF2E	MF1E
INTC2 (HT66F0082)	URF	SIMF	MF2F	MF1F	URE	SIME	MF2E	MF1E
MFI0	PTMA1F	PTMP1F	PTMA0F	PTMP0F	PTMA1E	PTMP1E	PTMA0E	PTMP0E
MFI1	PTMA3F	PTMP3F	PTMA2F	PTMP2F	PTMA3E	PTMP3E	PTMA2E	PTMP2E
MFI2	CTMA1F	CTMP1F	CTMA0F	CTMP0F	CTMA1E	CTMP1E	CTMA0E	CTMP0E

Interrupt Register Contents

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**: INT1 interrupt active edge selection
00: Disable
01: Rising Edge
10: Falling Edge
11: Dual Edges

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

INTC0 Register

Bit	7	6	5	4	3	2	1	0
Name	—	TB1F	TB0F	INT0F	TB1E	TB0E	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 **TB1F** : Time Base 1 Interrupt Request Flag
0: No request
1: Interrupt request

Bit 5 **TB0F**: Time Base 0 Interrupt Request Flag
0: No request
1: Interrupt request

Bit 4 **INT0F**: INT0 Interrupt Request Flag
0: No request
1: Interrupt request

Bit 3 **TB1E** : Time Base 1 Interrupt Control
0: Disable
1: Enable

Bit 2 **TB0E**: Time Base 0 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	INT1F	ADF	DEF	MF0F	INT1E	ADE	DEE	MF0E
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 **INT1F**: INT1 Interrupt Request Flag
0: No request
1: Interrupt request

Bit 6 **ADF**: A/D Converter Interrupt Request Flag
0: No request
1: Interrupt request

Bit 5 **DEF**: Data EEPROM Interrupt Request Flag
0: No request
1: Interrupt request

Bit 4 **MF0F**: Multi-function 0 Interrupt Request Flag
0: No request
1: Interrupt request

Bit 3 **INT1E**: INT1 Interrupt Control
0: Disable
1: Enable

Bit 2 **ADE**: A/D Converter Interrupt Control
0: Disable
1: Enable

Bit 1 **DEE**: Data EEPROM Interrupt Control
0: Disable
1: Enable

Bit 0 **MF0E**: Multi-function 0 Interrupt Control
0: Disable
1: Enable

INTC2 Register (HT66F0042)

Bit	7	6	5	4	3	2	1	0
Name	—	SIMF	MF2F	MF1F	—	SIME	MF2E	MF1E
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 **SIMF**: SIM Interrupt Request Flag
0: No request
1: Interrupt request

Bit 5 **MF2F**: Multi-function 2 Interrupt Request Flag
0: No request
1: Interrupt request

Bit 4 **MF1F**: Multi-function 1 Interrupt Request Flag
0: No request
1: Interrupt request

Bit 3 Unimplemented, read as "0"

Bit 2 **SIME**: SIM Interrupt Control
0: Disable
1: Enable

Bit 1 **MF2E**: Multi-function 2 Interrupt Control
0: Disable
1: Enable

Bit 0 **MF1E**: Multi-function 1 Interrupt Control
0: Disable
1: Enable

INTC2 Register (HT66F0082)

Bit	7	6	5	4	3	2	1	0
Name	URF	SIMF	MF2F	MF1F	URE	SIME	MF2E	MF1E
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7 **URF**: UART Interrupt Request Flag
0: No request
1: Interrupt request

Bit 6 **SIMF**: SIM Interrupt Request Flag
0: No request
1: Interrupt request

Bit 5 **MF2F**: Multi-function 2 Interrupt Request Flag
0: No request
1: Interrupt request

Bit 4 **MF1F**: Multi-function 1 Interrupt Request Flag
0: No request
1: Interrupt request

Bit 3 **URE**: UART Interrupt Control
0: Disable
1: Enable

Bit 2 **SIME**: SIM Interrupt Control
0: Disable
1: Enable

Bit 1 **MF2E**: Multi-function 2 Interrupt Control
0: Disable
1: Enable

Bit 0 **MF1E**: Multi-function 1 Interrupt Control
0: Disable
1: Enable

MFI0 Register

Bit	7	6	5	4	3	2	1	0
Name	PTMA1F	PTMP1F	PTMA0F	PTMP0F	PTMA1E	PTMP1E	PTMA0E	PTMP0E
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 **PTMA1F**: PTM1 Comparator A match Interrupt Request Flag
0: No request
1: Interrupt request

Bit 6 **PTMP1F**: PTM1 Comparator P match Interrupt Request Flag
0: No request
1: Interrupt request

Bit 5 **PTMA0F**: PTM0 Comparator A match Interrupt Request Flag
0: No request
1: Interrupt request

Bit 4 **PTMP0F**: PTM0 Comparator P match Interrupt Request Flag
0: No request
1: Interrupt request

Bit 3 **PTMA1E**: PTM1 Comparator A match Interrupt Control
0: Disable
1: Enable

Bit 2 **PTMP1E**: PTM1 Comparator P match Interrupt Control
0: Disable
1: Enable

Bit 1 **PTMA0E**: PTM0 Comparator A match Interrupt Control
0: Disable
1: Enable

Bit 0 **PTMP0E**: PTM0 Comparator P match Interrupt Control
0: Disable
1: Enable

MFI1 Register

Bit	7	6	5	4	3	2	1	0
Name	PTMA3F	PTMP3F	PTMA2F	PTMP2F	PTMA3E	PTMP3E	PTMA2E	PTMP2E
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 **PTMA3F**: PTM3 Comparator A match Interrupt Request Flag
0: No request
1: Interrupt request

Bit 6 **PTMP3F**: PTM3 Comparator P match Interrupt Request Flag
0: No request
1: Interrupt request

Bit 5 **PTMA2F**: PTM2 Comparator A match Interrupt Request Flag
0: No request
1: Interrupt request

Bit 4 **PTMP2F**: PTM2 Comparator P match Interrupt Request Flag
0: No request
1: Interrupt request

Bit 3 **PTMA3E**: PTM3 Comparator A match Interrupt Control
0: Disable
1: Enable

Bit 2 **PTMP3E**: PTM3 Comparator P match Interrupt Control
0: Disable
1: Enable

Bit 1 **PTMA2E**: PTM2 Comparator A match Interrupt Control
0: Disable
1: Enable

Bit 0 **PTMP2E**: PTM2 Comparator P match Interrupt Control
0: Disable
1: Enable

MFI2 Register

Bit	7	6	5	4	3	2	1	0
Name	CTMA1F	CTMP1F	CTMA0F	CTMP0F	CTMA1E	CTMP1E	CTMA0E	CTMP0E
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 **CTMA1F**: CTM1 Comparator A match Interrupt Request Flag
0: No request
1: Interrupt request

Bit 6 **CTMP1F**: CTM1 Comparator P match Interrupt Request Flag
0: No request
1: Interrupt request

Bit 5 **CTMA0F**: CTM0 Comparator A match Interrupt Request Flag
0: No request
1: Interrupt request

Bit 4 **CTMP0F**: CTM0 Comparator P match Interrupt Request Flag
0: No request
1: Interrupt request

Bit 3 **CTMA1E**: CTM1 Comparator A match Interrupt Control
0: Disable
1: Enable

Bit 2 **CTMP1E**: CTM1 Comparator P match Interrupt Control
0: Disable
1: Enable

Bit 1 **CTMA0E**: CTM0 Comparator A match Interrupt Control
0: Disable
1: Enable

Bit 0 **CTMP0E**: CTM0 Comparator P match Interrupt Control
0: Disable
1: Enable

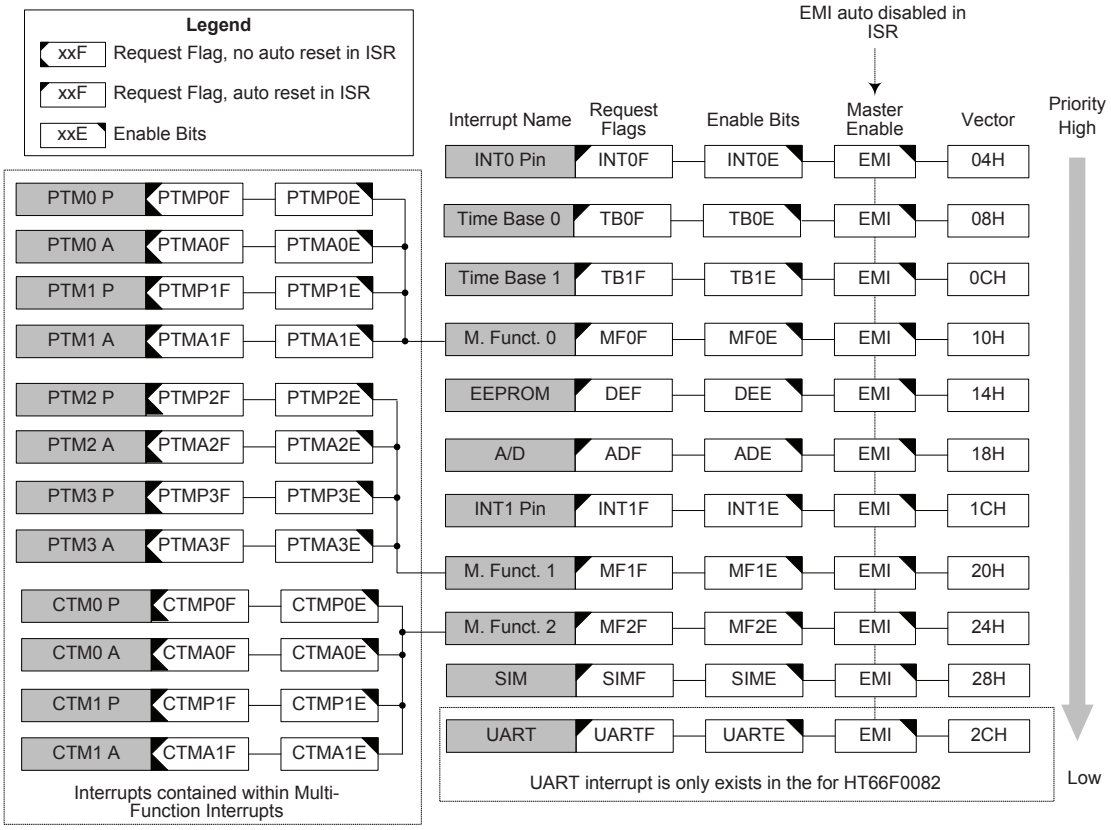
Interrupt Operation

When the conditions for an interrupt event occur, such as a TM Comparator P or Comparator A 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 interrupt is controlled by signal transitions on the pins INT0~INT1. An external interrupt request will take place when the external interrupt request flag, INTnF, is set, which will occur when a transition, whose type is chosen by the edge select bits, appears on the external interrupt pin. To allow the program to branch to the interrupt vector address, the global interrupt enable bit, EMI, and the external interrupt enable bit, INTnE, 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 by setting the pin-shared registers. The pin must also be setup as an input by setting the corresponding bit in the port control register. When the interrupt is enabled, the stack is not full and the correct transition type appears on the external interrupt pin, a subroutine call to the external interrupt vector, will take place. When the interrupt is serviced, the external interrupt request flag, INTnF, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts. Note that the pull-high resistor selection on the external interrupt pin will remain valid even if the pin is used as an external interrupt input.

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

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.

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, will not be automatically reset and must be manually reset by the application program.

A/D Converter Interrupt

The devices contain an A/D converter which has its own independent 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.

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}, the clock of which can be selected between the $f_{SYS}/4$ and f_{TBC}. 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	LXTLP	TB02	TB01	TB00
R/W	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

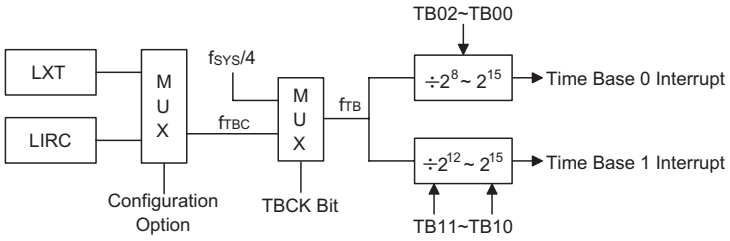
Bit 7 **TBON**: TB0 and TB1 Control bit
0: Disable
1: Enable

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 **LXTLP**: LXT low power control
0: Disable
1: Enable

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

EEPROM Interrupt

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

TM Interrupts

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

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

Serial Interface Module Interrupt

The Serial Interface Module Interrupt, also known as the SIM interrupt, is controlled by the SPI or I^2C data transfer. A SIM Interrupt request will take place when the SIM Interrupt request flag, SIMF, is set, which occurs when a byte of data has been received or transmitted by the SIM interface, an I^2C slave address match or I^2C bus time-out occurrence. 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, SIME, must first be set. When the interrupt is enabled, the stack is not full and any of the above described situations occurs, a subroutine call to the respective SIM Interrupt vector, will take place. When the Serial Interface Interrupt is serviced, the EMI bit will be automatically cleared to disable other interrupts. The SIMF flag will also be automatically cleared.

UART Transfer Interrupt

The UART Transfer Interrupt is controlled by several UART transfer conditions. When one of these conditions occurs, an interrupt pulse will be generated to get the attention of the microcontroller. These conditions are a transmitter data register empty, transmitter idle, receiver data available, receiver overrun, address detect and an RX pin wake-up. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and UART Interrupt enable bit, URE, must first be set. When the interrupt is enabled, the stack is not full and any of the conditions described above occurs, a subroutine call to the UART Interrupt vector, will take place. When the interrupt is serviced, the UART Interrupt flag, URF, will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts.

Interrupt Wake-up Function

Each of the interrupt functions has the capability of waking up the microcontroller when in the SLEEP or IDLE Mode. A wake-up is generated when an interrupt request flag changes from low to high and is independent of whether the interrupt is enabled or not. Therefore, even though the device is in the SLEEP or IDLE Mode and its system oscillator stopped, situations such as external edge transitions on the external interrupt pin, 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.

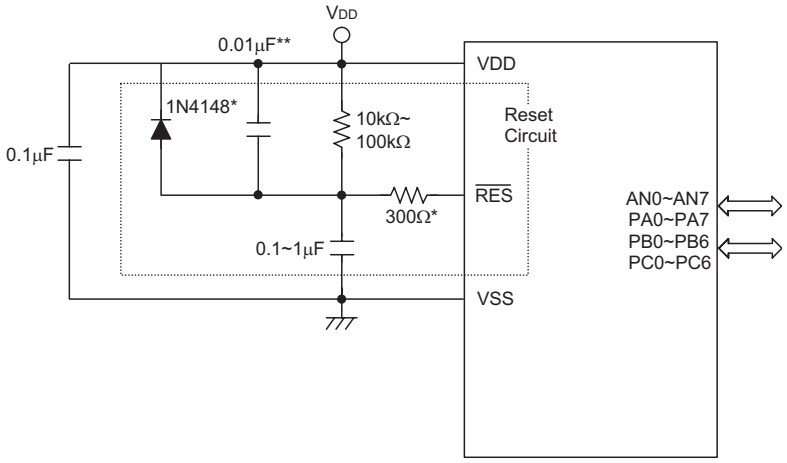
To return from an interrupt subroutine, either a RET or RETI instruction may be executed. The RETI instruction in addition to executing a return to the main program also automatically sets the EMI bit high to allow further interrupts. The RET instruction however only executes a return to the main program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

Configuration Options

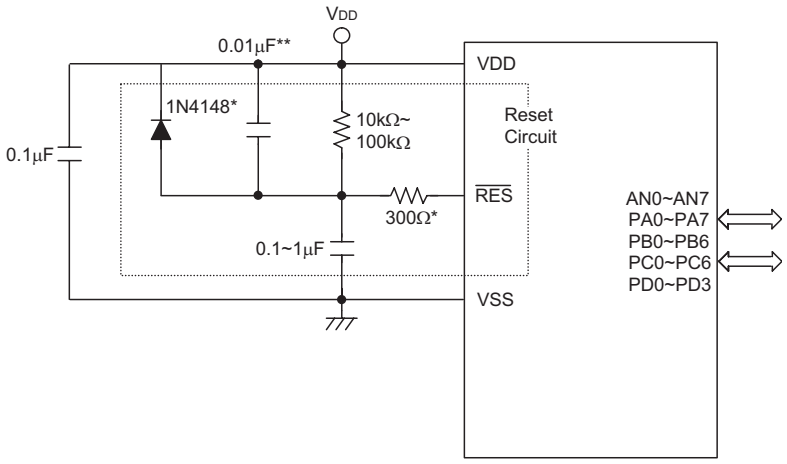
Configuration options refer to certain options within the MCU that are programmed into the device during the programming process. During the development process, these options are selected using the HT-IDE software development tools. As these options are programmed into the device using the hardware 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	Low Speed System Oscillator Selection: LXT LIRC
2	HIRC Frequency Selection: 8MHz 12MHz 16MHz
3	LXT Option: Integrated Resistor and Capacitor External Resistor and Capacitor

Application Circuits



HT66F0042



HT66F0082

Instruction Set

Introduction

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

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

Instruction Timing

Most instructions are implemented within one instruction cycle. The exceptions to this are branch, call, or table read instructions where two instruction cycles are required. One instruction cycle is equal to 4 system clock cycles, therefore in the case of an 8MHz system oscillator, most instructions would be implemented within 0.5μs 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 several 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 such as INC, INCA, DEC and DECA provide a simple means of increasing or decreasing by a value of one of the values in the destination specified.

Logical and Rotate Operation

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

Branches and Control Transfer

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

Bit Operations

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

Table Read Operations

Data storage is normally implemented by using registers. However, when working with large amounts of fixed data, the volume involved often makes it inconvenient to store the fixed data in the Data Memory. To overcome this problem, Holtek microcontrollers allow an area of Program Memory to be 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 Operation			
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 Operation			
TABRD [m]	Read table (specific page) to TBLH and Data Memory	2Note	None
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 ACC ← 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 bit wise 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 WDT are 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 WDT are 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 CLR WDT1 will have no effect.

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] ← $\overline{[m]}$

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] ← ACC + 00H or
[m] ← ACC + 06H or
[m] ← ACC + 60H or
[m] ← ACC + 66H

Affected flag(s) C

DEC [m] Decrement Data Memory

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

Operation [m] ← [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 ← [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 ← 1

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] − 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: [m] ← ACC − [m] − 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].i ← 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

TABRD [m] Read table (specific page) to TBLH and Data Memory

Description The low byte of the program code (specific page) addressed by the table pointer pair (TBHP and 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

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

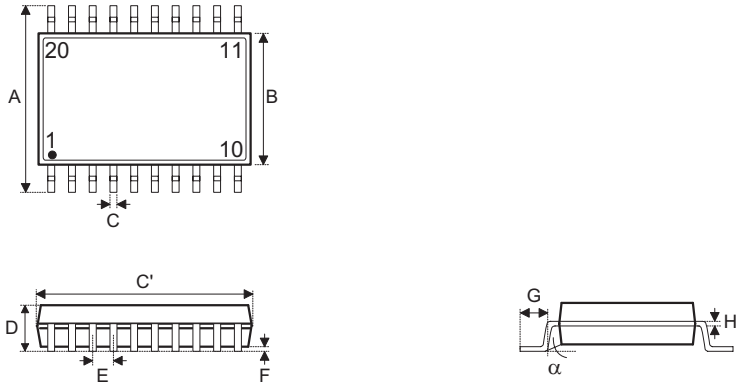
Package Information

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

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

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

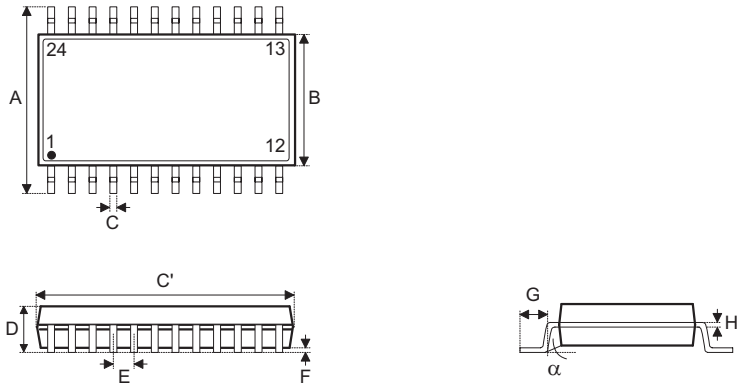
20-pin SOP (300mil) Outline Dimensions



Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	—	0.406 BSC	—
B	—	0.295 BSC	—
C	0.012	—	0.020
C’	—	0.504 BSC	—
D	—	—	0.104
E	—	0.050 BSC	—
F	0.004	—	0.012
G	0.016	—	0.050
H	0.008	—	0.013
α	0°	—	8°

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	—	10.30 BSC	—
B	—	7.5 BSC	—
C	0.31	—	0.51
C’	—	12.8 BSC	—
D	—	—	2.65
E	—	1.27 BSC	—
F	0.10	—	0.30
G	0.40	—	1.27
H	0.20	—	0.33
α	0°	—	8°

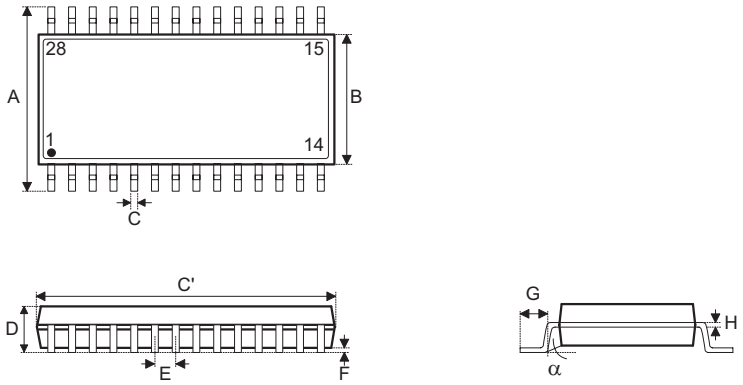
24-pin SOP (300mil) Outline Dimensions



Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	—	0.406 BSC	—
B	—	0.295 BSC	—
C	0.012	—	0.020
C’	—	0.606 BSC	—
D	—	—	0.104
E	—	0.050 BSC	—
F	0.004	—	0.012
G	0.016	—	0.050
H	0.008	—	0.013
α	0°	—	8°

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	—	10.30 BSC	—
B	—	7.50 BSC	—
C	0.31	—	0.51
C’	—	15.4 BSC	—
D	—	—	2.65
E	—	1.27 BSC	—
F	0.10	—	0.30
G	0.40	—	1.27
H	0.20	—	0.33
α	0°	—	8°

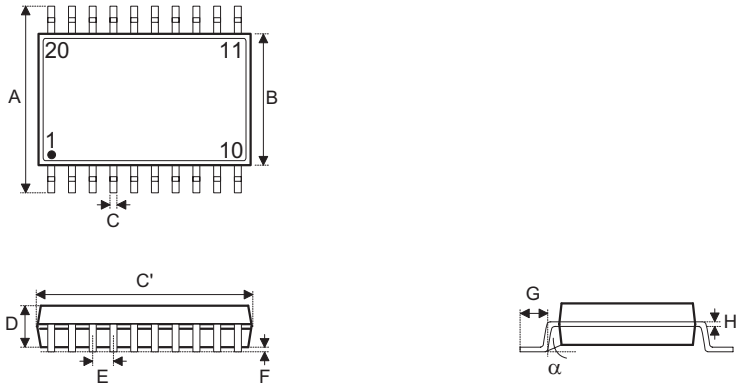
28-pin SOP (300mil) Outline Dimensions



Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	—	0.406 BSC	—
B	—	0.295 BSC	—
C	0.012	—	0.020
C'	—	0.705 BSC	—
D	—	—	0.104
E	—	0.050 BSC	—
F	0.004	—	0.012
G	0.016	—	0.050
H	0.008	—	0.013
α	0°	—	8°

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	—	10.30 BSC	—
B	—	7.50 BSC	—
C	0.31	—	0.51
C'	—	17.90 BSC	—
D	—	—	2.65
E	—	1.27 BSC	—
F	0.10	—	0.30
G	0.40	—	1.27
H	0.20	—	0.33
α	0°	—	8°

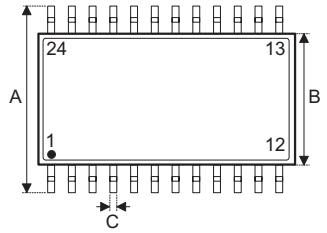
20-pin SSOP (150mil) Outline Dimensions

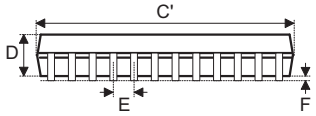


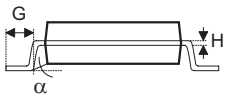
Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	—	0.236 BSC	—
B	—	0.154 BSC	—
C	0.008	—	0.012
C'	—	0.341 BSC	—
D	—	—	0.069
E	—	0.025 BSC	—
F	0.004	—	0.010
G	0.016	—	0.050
H	0.004	—	0.010
α	0°	—	8°

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	—	6.0 BSC	—
B	—	3.9 BSC	—
C	0.20	—	0.30
C'	—	8.66 BSC	—
D	—	—	1.75
E	—	0.635 BSC	—
F	0.10	—	0.25
G	0.41	—	1.27
H	0.10	—	0.25
α	0°	—	8°

24-pin SSOP (150mil) Outline Dimensions



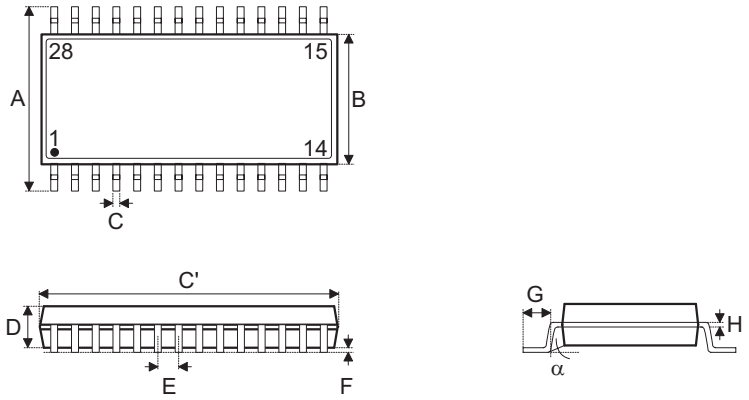




Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	—	0.236 BSC	—
B	—	0.154 BSC	—
C	0.008	—	0.012
C’	—	0.341 BSC	—
D	—	—	0.069
E	—	0.025 BSC	—
F	0.004	—	0.010
G	0.016	—	0.050
H	0.004	—	0.010
α	0°	—	8°

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	—	6.0 BSC	—
B	—	3.9 BSC	—
C	0.20	—	0.30
C’	—	8.66 BSC	—
D	—	—	1.75
E	—	0.635 BSC	—
F	0.10	—	0.25
G	0.41	—	1.27
H	0.10	—	0.25
α	0°	—	8°

28-pin SSOP (150mil) Outline Dimensions



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

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	—	6.0 BSC	—
B	—	3.9 BSC	—
C	0.20	—	0.30
C'	—	9.9 BSC	—
D	—	—	1.75
E	—	0.635 BSC	—
F	0.10	—	0.25
G	0.41	—	1.27
H	0.10	—	0.25
α	0°	—	8°