A/D Flash MCU with EEPROM

HT66F0176

Revision: V1.20 Date: August 28, 2017

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
- Oscillator types:
  - External High Speed Crystal – HXT
  - External 32.768kHz Crystal – LXT
  - Internal High Speed RC – HIRC
  - Internal Low Speed 32kHz RC – LIRC
- Multi-mode operation: FAST, SLOW, IDLE and SLEEP
- Fully integrated internal 8/12/16MHz oscillator requires no external components
- All instructions executed in one or two instruction cycles
- Table read instructions
- 63 powerful instructions
- 8-level subroutine nesting
- Bit manipulation instruction

Peripheral Features

- Flash Program Memory: 2K×16
- RAM Data Memory: 128×8
- True EEPROM Memory: 64×8
- Watchdog Timer function
- 22 bidirectional I/O lines
- Two pin-shared external interrupts
- Dual Timer Modules for time measurement, compare match output or PWM output function
- Serial Interface Module – SIM for SPI or I^2C
- Software controlled 6-SCOM/SSEG and 14-SSEG lines LCD driver with 1/3 bias
- Programmable I/O port source current for LED applications
- Dual Time-Base functions for generation of fixed time interrupt signals
- 8-external channel 12-bit resolution A/D converter
- Fully-duplex Universal Asynchronous Receiver and Transmitter Interface – UART
- Low voltage reset function
- Low voltage detect function
- Package types: 16/20-pin NSOP, 24-pin SOP/SSOP

General Description

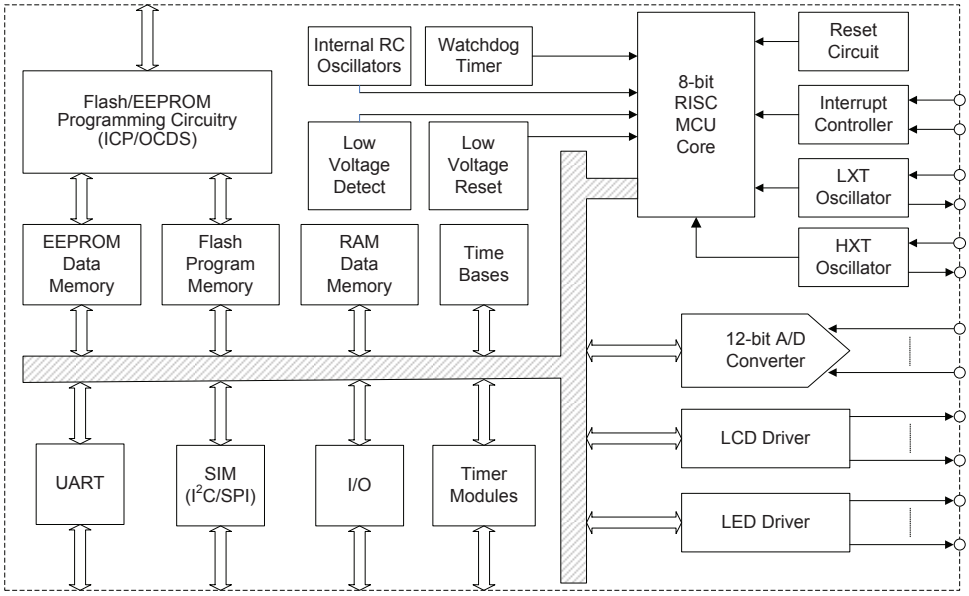
The HT66F0176 device is a Flash Memory A/D type 8-bit high performance RISC architecture microcontroller. Offering users the convenience of Flash Memory multi-programming features, the device also includes a wide range of functions and features. Other memory includes an area of RAM Data Memory as well as an area of true EEPROM memory for storage of non-volatile data such as serial numbers, calibratuib data, etc.

Analog features include a multi-channel 12-bit A/D converter and a comparator functions. Multiple and extremely flexible Timer Modules provide timing, pulse generation and PWM generation functions. Communication with the outside world is catered for by including fully integrated SPI, I^2C, and UART interface functions, popular interfaces which provide designers with a means of easy communication with external peripheral hardware.Protective features such as an internal Watchdog Timer, Low Voltage Reset and Low Voltage Detector coupled with excellent noise immunity and ESD protection ensure that reliable operation is maintained in hostile electrical environments.

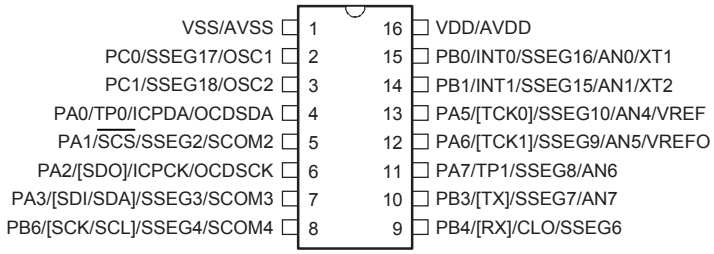
A full choice of external, internal high and low oscillators are provided including a fully integrated system oscillator which requires no external components for its implementation. The ability to operate and switch dynamically between a range of operating modes using different clock sources gives users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

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

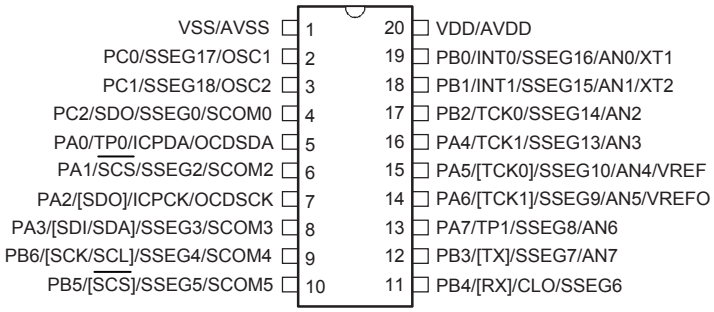
Block Diagram



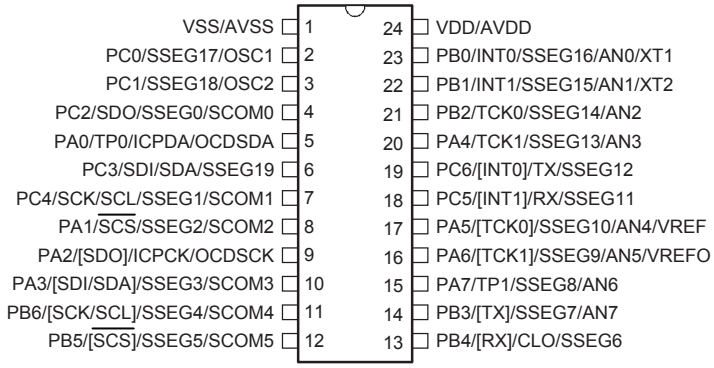
Pin Assignment



HT66F0176/HT66V0176
16 NSOP-A



HT66F0176/HT66V0176
20 NSOP-A



HT66F0176/HT66V0176
24 SOP/SSOP-A

Note: 1. Bracketed pin names indicate non-default pinout remapping locations. The detailed information can be referenced to the relevant chapter.
2. If the pin-shared functions have multiple outputs simultaneously, its pin names at the right side of the "/" sign can be used for higher priority.
3. The OCDSDA and OCDSCK pins are supplied for the OCDS dedicated pins and as such only available for the HT66V0176 device which is the OCDS EV chip for the HT66F0176 device.

Pin Description

The function of each pin is listed in the following table, however the details behind how each pin is configured is contained in other sections of the datasheet.

Pin Name	Function	OPT	I/T	O/T	Description
PA0/TP0/ICPDA/ OCDSDA	PA0	PAPU PAWU	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	TP0	TMPC	—	CMOS	PTM0 output
	ICPDA	—	ST	CMOS	ICP address/data
	OCDSDA	—	ST	CMOS	OCDS address/data, for EV chip only.
PA1/$\overline{SCS}$/SSEG2/ SCOM2	PA1	PAPU PAWU	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	$\overline{SCS}$	IFS0	ST	CMOS	SPI slave select
	SSEG2	SLCDC0 SLCDC1	—	SSEG	LCD segment output
	SCOM2	SLCDC0 SLCDC1	—	SCOM	LCD common output
PA2/SDO/ICPCK/ OCDSCK	PA2	PAPU PAWU	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	SDO	IFS0	—	CMOS	SPI data output
	ICPCK	—	ST	—	ICP clock
	OCDSCK	—	ST	—	OCDS clock pin, for EV chip only
PA3/SDI/SDA/ SSEG3/SCOM3	PA3	PAPU PAWU	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	SDI	IFS0	ST	—	SPI serial data input
	SDA	IFS0	ST	NMOS	I^2C data line
	SSEG3	SLCDC0 SLCDC1	—	SSEG	LCD segment output
	SCOM3	SLCDC0 SLCDC1	—	SCOM	LCD common output
PA4/TCK1/SSEG13/ AN3	PA4	PAPU PAWU	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	TCK1	IFS1	ST	—	PTM1 clock input
	SSEG13	SLCDC2	—	SSEG	LCD segment output
	AN3	ACERL	AN	—	A/D Converter external input
PA5/TCK0/SSEG10/ AN4/VREF	PA5	PAPU PAWU	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	TCK0	IFS1	ST	—	PTM0 clock input
	SSEG10	SLCDC2	—	SSEG	LCD segment output
	AN4	ACERL	AN	—	A/D Converter external input
	VREF	SADC2	AN	—	A/D Converter reference voltage input
PA6/TCK1/SSEG9/ AN5/VREFO	PA6	PAPU PAWU	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	TCK1	IFS1	ST	—	PTM1 clock input
	SSEG9	SLCDC2	—	SSEG	LCD segment output
	AN5	ACERL	AN	—	A/D Converter external input
	VREFO	SADC2	—	AN	A/D Converter reference voltage output
PA7/TP1/SSEG8/ AN6	PA7	PAPU PAWU	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	TP1	TMPC	—	CMOS	PTM1 output
	SSEG8	SLCDC2	—	SSEG	LCD segment output
	AN6	ACERL	AN	—	A/D Converter external input

Pin Name	Function	OPT	I/T	O/T	Description
PB0/INT0/SSEG16/AN0/XT1	PB0	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up
	INT0	IFS0 INTEG INTC0	ST	—	External Interrupt 0 input
	SSEG16	SLCDC3	—	SSEG	LCD segment output
	AN0	ACERL	AN	—	A/D Converter external input
	XT1	CO	LXT	—	LXT oscillator pin
PB1/INT1/SSEG15/AN1/XT2	PB1	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up
	INT1	IFS0 INTEG INTC2	ST	—	External interrupt 1 input
	SSEG15	SLCDC3	—	SSEG	LCD segment output
	AN1	ACERL	AN	—	A/D Converter external input
	XT2	CO	—	LXT	LXT oscillator pin
PB2/TCK0/SSEG14/AN2	PB2	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up
	TCK0	IFS1	ST	—	PTM0 clock input
	SSEG14	SLCDC3	—	SSEG	LCD segment output
	AN2	ACERL	AN	—	A/D Converter external input
PB3/TX/SSEG7/AN7	PB3	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up
	TX	IFS1	—	CMOS	UART TX serial data output
	SSEG7	SLCDC2	—	SSEG	LCD segment output
	AN7	ACERL	AN	—	A/D Converter external input
PB4/RX/CLO/SSEG6	PB4	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up
	RX	IFS1	ST	—	UART RX serial data input
	CLO	TMPC	—	CMOS	System clock output
	SSEG6	SLCDC2	—	SSEG	LCD segment output
PB5/$\overline{SCS}$/SSEG5/SCOM5	PB5	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up
	$\overline{SCS}$	IFS0	ST	CMOS	SPI slave select
	SSEG5	SLCDC1	—	SSEG	LCD segment output
	SCOM5	SLCDC1	—	SCOM	LCD common output
PB6/SCK/SCL/SSEG4/SCOM4	PB6	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up
	SCK	IFS0	ST	CMOS	SPI serial clock
	SCL	IFS0	ST	NMOS	I^2C clock line
	SSEG4	SLCDC1	—	SSEG	LCD segment output
	SCOM4	SLCDC1	—	SCOM	LCD common output
PC0/SSEG17/OSC1	PC0	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up
	SSEG17	SLCDC3	—	SSEG	LCD segment output
	OSC1	CO	HXT	—	HXT oscillator pin
PC1/SSEG18/OSC2	PC1	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up
	SSEG18	SLCDC3	—	SSEG	LCD segment output
	OSC2	CO	—	HXT	HXT oscillator pin
PC2/SDO/SSEG0/SCOM0	PC2	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up
	SDO	IFS0	—	CMOS	SPI data output
	SSEG0	SLCDC0 SLCDC1	—	SSEG	LCD segment output
	SCOM0	SLCDC0 SLCDC1	—	SCOM	LCD common output

Pin Name	Function	OPT	I/T	O/T	Description
PC3/SDI/SDA/ SSEG19	PC3	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up
	SDI	IFS0	ST	—	SPI serial data input
	SDA	IFS0	ST	NMOS	I²C data line
	SSEG19	SLCDC3	—	SSEG	LCD segment output
PC4/SCK/SCL/ SSEG1/SCOM1	PC4	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up
	SCK	IFS0	ST	CMOS	SPI serial clock
	SCL	IFS0	ST	NMOS	I²C clock line
	SSEG1	SLCDC0 SLCDC1	—	SSEG	LCD segment output
	SCOM1	SLCDC0 SLCDC1	—	SCOM	LCD common output
PC5/INT1/RX/ SSEG11	PC5	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up
	INT1	IFS0 INTEG INTC2	ST	—	External interrupt 1 input
	RX	IFS1	ST	—	UART RX serial data input
	SSEG11	SLCDC2	—	SSEG	LCD segment output
PC6/INT0/TX/ SSEG12	PC6	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up
	INT0	IFS0 INTEG INTC0	ST	—	External Interrupt 0 input
	TX	IFS1	—	CMOS	UART TX serial data output
	SSEG12	SLCDC2	—	SSEG	LCD segment output
VDD/AVDD	VDD	—	PWR	—	Digital positive power supply
	AVDD	—	PWR	—	Analog positive power supply
VSS/AVSS	VSS	—	PWR	—	Digital negative power supply,ground
	AVSS	—	PWR	—	Analog negative power supply,ground

Legend: I/T: Input type; O/T: Output type;
OPT: Optional by configuration option (CO) or register option; CO: Configuration option;
ST: Schmitt Trigger input; AN: Analog signal;
CMOS: CMOS output; NMOS: NMOS output;
SSEG: Software controlled LCD SEG; SCOM: Software controlled LCD COM;
HXT: High frequency crystal oscillator; LXT: Low frequency crystal oscillator
PWR: Power
* The AVDD pin is internally bonded together with the VDD pin while the AVSS pin is internally bonded together with the VSS pin.
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 device. Functional operation of the device at other conditions beyond those listed in the specification is not implied and prolonged exposure to extreme conditions may affect device reliability.

D.C. Characteristics

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

Operating Voltage Characteristics

Ta= -40°C~85°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	Operating Voltage – HXT	—	$f_{SYS}=f_{HXT}$=8MHz	2.2	—	5.5	V
			$f_{SYS}=f_{HXT}$=12MHz	2.7	—	5.5	
			$f_{SYS}=f_{HXT}$=16MHz	3.3	—	5.5	
	Operating Voltage – HIRC	—	$f_{SYS}=f_{HIRC}$=8MHz	2.2	—	5.5	V
			$f_{SYS}=f_{HIRC}$=12MHz	2.7	—	5.5	
			$f_{SYS}=f_{HIRC}$=16MHz	3.3	—	5.5	
	Operating Voltage – LXT	—	$f_{SYS}=f_{LXT}$=32.768kHz	2.2	—	5.5	V
	Operating Voltage – LIRC	—	$f_{SYS}=f_{LIRC}$=32kHz	2.2	—	5.5	V

Standby Current Characteristics

Ta=25°C

Symbol	Standby Mode	Test Conditions		Min.	Typ.	Max.	Max.	Unit
		V_{DD}	Conditions				85°C	
I_{STB}	SLEEP Mode	2.2V	WDT off	—	0.2	0.6	0.8	μA
		3V		—	0.2	0.8	1.0	
		5V		—	0.5	1.0	1.2	
		2.2V	WDT on	—	1.2	2.4	2.6	μA
		3V		—	1.5	3	3.2	
		5V		—	2.5	5	5.2	
	IDLE0 Mode	2.2V	f_{SUB} on	—	2.4	4	4.2	μA
		3V		—	3	5	6	
		5V		—	5	10	12	
	IDLE1 Mode – HIRC	2.2V	f_{SUB} on, f_{SYS}=8MHz	—	0.25	0.5	0.8	mA
		3V		—	0.5	1.0	1.2	
		5V		—	1.0	2.0	2.2	
		2.7V	f_{SUB} on, f_{SYS}=12MHz	—	1.0	2.0	2.2	mA
		3V		—	1.2	2.4	2.4	
		5V		—	1.5	3.0	3.2	
		3.3V	f_{SUB} on, f_{SYS}=16MHz	—	1.5	3.0	3.2	mA
		5V		—	2.0	4.0	4.2	
	IDLE1 Mode – HXT	2.2V	f_{SUB} on, f_{SYS}=8MHz	—	0.25	0.5	0.8	mA
		3V		—	0.5	1.0	1.2	
		5V		—	1.0	2.0	2.2	
		2.7V	f_{SUB} on, f_{SYS}=12MHz	—	1.0	2.0	2.2	mA
		3V		—	1.2	2.4	2.4	
		5V		—	1.5	3.0	3.2	
		3.3V	f_{SUB} on, f_{SYS}=16MHz	—	1.5	3.0	3.2	mA
		5V		—	2.0	4.0	4.2	

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

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

Operating Current Characteristics

Ta=25°C

Symbol	Operating Mode	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{DD}	SLOW Mode – LIRC	2.2V	f_{SYS}=32kHz	—	8	16	μA
		3V		—	10	20	
		5V		—	30	50	
	SLOW Mode – LXT	2.2V	f_{SYS}=32768Hz	—	8	16	μA
		3V		—	10	20	
		5V		—	30	50	
	Fast Mode – HIRC	2.2V	f_{SYS}=8MHz	—	0.7	1.0	mA
		3V		—	2.0	2.8	
		5V		—	3.0	4.5	
		2.7V	f_{SYS}=12MHz	—	1.0	1.5	mA
		3V		—	3.0	4.2	
		5V		—	4.5	6.7	
		3.3V	f_{SYS}=16MHz	—	3.0	4.5	mA
		5V		—	6.0	9.0	
	Fast Mode – HXT	2.2V	f_{SYS}=8MHz	—	0.8	1.2	mA
		3V		—	1	1.5	
		5V		—	2.5	4.0	
		2.7V	f_{SYS}=12MHz	—	1.2	2.2	mA
		3V		—	1.5	2.5	
		5V		—	3.5	5.5	
		3.3V	f_{SYS}=16MHz	—	3.2	4.8	mA
		5V		—	4.5	7.0	

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

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

A.C. Characteristics

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

High Speed Internal Oscillator – HIRC – Frequency Accuracy

During the program writing operation the writer will trim the HIRC oscillator at a user selected HIRC frequency and user selected voltage of either 3V or 5V.

8/12/16MHz

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Temp.				
f_{HIRC}	8MHz Writer Trimmed HIRC Frequency	3V/5V	Ta=25°C	-1%	8	+1%	MHz
			Ta= -40°C ~ 85°C	-2%	8	+2%	
		2.2V~5.5V	Ta=25°C	-1.5%	8	+1.5%	
			Ta= -40°C ~ 85°C	-3%	8	+3%	
	12MHz Writer Trimmed HIRC Frequency	3V/5V	Ta=25°C	-1%	12	+1%	MHz
			Ta= -40°C ~ 85°C	-2%	12	+2%	
		2.7V~5.5V	Ta=25°C	-1.5%	12	+1.5%	
			Ta= -40°C ~ 85°C	-3%	12	+3%	
	16MHz Writer Trimmed HIRC Frequency	5V	Ta=25°C	-1%	16	+1%	MHz
			Ta= -40°C ~ 85°C	-2%	16	+2%	
		3.3V~5.5V	Ta=25°C	-1.5%	16	+1.5%	
			Ta= -40°C ~ 85°C	-3%	16	+3%	

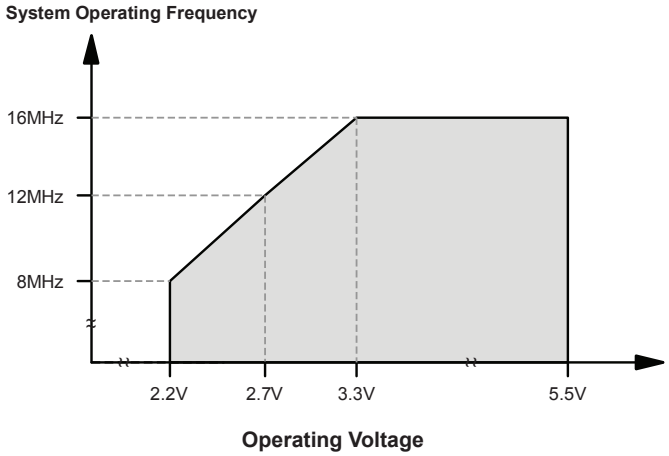
Notes: 1. The 3V/5V values for V_{DD} are provided as these are the two selectable fixed voltages at which the HIRC frequency is trimmed by the writer.
2. The row below the 3V/5V trim voltage row is provided to show the values for the full V_{DD} range operating voltage. It is recommended that the trim voltage is fixed at 3V for application voltage ranges from 2.2V to 3.6V and fixed at 5V for application voltage ranges from 3.3V to 5.5V.
3. The minimum and maximum tolerance values provided in the table are only for the frequency at which the writer trims the HIRC oscillator. After trimming at this chosen specific frequency any change in HIRC oscillator frequency using the oscillator register control bits by the application program will give a frequency tolerance to within ±20%.

Low Speed Internal Oscillator Characteristics – LIRC

Ta=25°C, unless otherwise specified

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Temp.				
f_{LIRC}	Oscillator Frequency	5V	Ta=25°C	-10%	32	+10%	kHz
		2.2V~5.5V	Ta= -40°C ~ 85°C	-50%	32	+60%	

Operating Frequency Characteristic Curves



System Start Up Time Characteristics

Ta= -40°C~85°C

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
t_{SST}	System Start-up Time Wake-up from condition where f_{SYS} is off	$f_{SYS}=f_H \sim f_H/64$, $f_H=f_{HXT}$	128	—	—	t_{HXT}
		$f_{SYS}=f_H \sim f_H/64$, $f_H=f_{HIRC}$	16	—	—	t_{HIRC}
		$f_{SYS}=f_{SUB}=f_{LXT}$	128	—	—	t_{LXT}
		$f_{SYS}=f_{SUB}=f_{LIRC}$	2	—	—	t_{LIRC}
	System Start-up Time Wake-up from condition where f_{SYS} is on.	$f_{SYS}=f_H \sim f_H/64$, $f_H=f_{HXT}$ or f_{HIRC}	2	—	—	t_H
		$f_{SYS}=f_{SUB}=f_{LXT}$ or f_{LIRC}	2	—	—	t_{SUB}
t_{RSTD}	System Reset Delay Time Reset source from Power-on reset or LVR hardware reset	—	25	50	100	ms
	System Reset Delay Time LVRC/WDTC software reset	—				
	System Reset Delay Time Reset source from WDT overflow	—	8.3	16.7	33.3	ms
t_{SRESET}	Minimum software reset width to reset	—	45	90	120	μs

Notes: 1. For the System Start-up time values, whether f_{SYS} is on or off depends upon the mode type and the chosen f_{SYS} system oscillator. Details are provided in the System Operating Modes section.

2. The time units, shown by the symbols t_{HXT}, t_{HIRC} etc. are the inverse of the corresponding frequency values as provided in the frequency tables. For example $t_{HIRC}=1/f_{HIRC}$, $t_{SYS}=1/f_{SYS}$ etc.

3. The System Speed Switch Time is effectively the time taken for the newly activated oscillator to start up.

Input/Output Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{IL}	Input Low Voltage for I/O Ports	5V	—	0	—	1.5	V
		—	—	0	—	$0.2V_{DD}$	
V_{IH}	Input High Voltage for I/O Ports	5V	—	3.5	—	5.0	V
		—	—	$0.8V_{DD}$	—	V_{DD}	
I_{OL}	Sink Current for I/O Pins	3V	$V_{OL}=0.1V_{DD}$	16	32	—	mA
		5V		32	64	—	
I_{OH}	Source Current for I/O Pins	3V	$V_{OH}=0.9V_{DD}$, PxPS[n+1:n]=00, (x=A, B or C, n=0 or 2)	-1.0	-2.0	—	mA
		5V		-2.0	-4.0	—	
		3V	$V_{OH}=0.9V_{DD}$, PxPS[n+1:n]=01, (x=A, B or C, n=0 or 2)	-1.75	-3.5	—	
		5V		-3.5	-7.0	—	
		3V	$V_{OH}=0.9V_{DD}$, PxPS[n+1:n]=10, (x=A, B or C, n=0 or 2)	-2.5	-5.0	—	
		5V		-5.0	-10.0	—	
		3V	$V_{OH}=0.9V_{DD}$, PxPS[n+1:n]=11, (x=A, B or C, n=0 or 2)	-5.5	-11.0	—	
		5V		-11.0	-22.0	—	
R_{PH}	Pull-high Resistance for I/O Ports(Note)	3V	—	20	60	100	kΩ
		5V	—	10	30	50	
t_{TCK}	TM Clock Input Minimum Pulse Width	—	—	0.3	—	—	μs
t_{INT}	Interrupt Input Pin Minimum Pulse Width	—	—	10	—	—	μs

Note: The R_{PH} internal pull high resistance value is calculated by connecting to ground and enabling the input pin with a pull-high resistor and then measuring the input sink current at the specified supply voltage level. Dividing the voltage by this measured current provides the R_{PH} value.

Memory Characteristics

Ta= -40°C~85°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{RW}	V_{DD} for Read / Write	—	—	V_{DDmin}	—	V_{DDmax}	V
Program Flash / Data EEPROM Memory							
t_{DEW}	Erase / Write cycle time	—	—	—	2	4	ms
I_{DDPGM}	Programming / Erase current on V_{DD}	—	—	—	—	5.0	mA
E_P	Cell Endurance	—	—	100K	—	—	E/W
t_{RETD}	ROM Data Retention time	—	Ta=25°C	—	40	—	Year
RAM Data Memory							
V_{DR}	RAM Data Retention voltage	—	Device in SLEEP Mode	1.0	—	—	V

A/D Converter Electrical Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	Operating Voltage	—	—	2.2	—	5.5	V
V_{ADI}	Input Voltage	—	—	0	—	V_{REF}	V
V_{REF}	Reference Voltage	—	—	2	—	V_{DD}	V
DNL	Differential non-linearity	2.2V~2.7V	V_{REF}=V_{DD}, t_{ADCK}=8μs	—	±15	—	LSB
		2.7V~5.5V	V_{REF}=V_{DD}, t_{ADCK}=0.5μs or 10μs	-3	—	+3	LSB
INL	Integral non-linearity	2.2V~2.7V	V_{REF}=V_{DD}, t_{ADCK}=8μs	—	±16	—	LSB
		2.7V~5.5V	V_{REF}=V_{DD}, t_{ADCK}=0.5μs or 10μs	-4	—	+4	LSB
I_{ADC}	Additional Current Consumption for A/D Converter Enable	3V	No load, t_{ADCK} =0.5μs	—	1.0	2.0	mA
		5V	No load, t_{ADCK} =0.5μs	—	1.5	3.0	
t_{ADCK}	Clock Period	2.2V~2.7V	—	8	—	10	μs
		2.7V~5.5V	—	0.5	—	10	μs
t_{ADC}	Conversion Time (A/D Sample and Hold Time)	—	12-bit A/D converter	—	16	—	t_{ADCK}
t_{ON2ST}	A/D Converter On-to-Start Time	—	—	4	—	—	μs

Reference Voltage Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{BG}	Bandgap Reference Voltage	—	—	-3%	1.04	+3%	V
t_{BGS}	V_{BG} Turn on Stable Time	—	No load	—	—	150	μs

LVD/LVR Electrical Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{LVR}	Low Voltage Reset Voltage	—	LVR Enable, voltage select 2.1V	- 5%	2.1	+ 5%	V
			LVR Enable, voltage select 2.55V		2.55		
			LVR Enable, voltage select 3.15V		3.15		
			LVR Enable, voltage select 3.8V		3.8		
V_{LVD}	Low Voltage Detector Voltage	—	LVD Enable, voltage select 2.0V	- 5%	2.0	+ 5%	V
			LVD Enable, voltage select 2.2V		2.2		
			LVD Enable, voltage select 2.4V		2.4		
			LVD Enable, voltage select 2.7V		2.7		
			LVD Enable, voltage select 3.0V		3.0		
			LVD Enable, voltage select 3.3V		3.3		
			LVD Enable, voltage select 3.6V		3.6		
			LVD Enable, voltage select 4.0V		4.0		
t_{LVDS}	LVDO stable time	—	For LVR enable, VBGEN=0, LVD off→on	—	—	15	μs
		—	For LVR disable, VBGEN=0, LVD off→on	—	—	150	
t_{LVR}	Minimum Low Voltage Width to Reset	—	—	120	240	480	μs

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
t_{LVD}	Minimum Low Voltage Width to Interrupt	—	—	60	120	240	μs

Software Controlled LCD Driver Electrical Characteristics

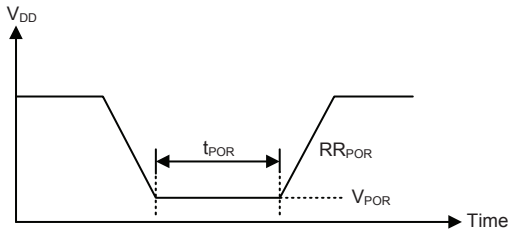
Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{BIAS}	Bias Current	5V	ISEL[1:0]=00	4.2	8.3	13	μA
			ISEL[1:0]=01	8.3	16.7	25	
			ISEL[1:0]=10	25	50	75	
			ISEL[1:0]=11	50	100	150	
V_{LCD_H}	[(2/3) × V_{DD}] voltage for LCD SCOM/ SSEG output	2.2V~5.5V	No load	0.645	0.67	0.695	V_{DD}
V_{LCD_L}	[(1/3) × V_{DD}] voltage for LCD SCOM/ SSEG output	2.2V~5.5V	No load	0.305	0.33	0.355	V_{DD}

Power-on Reset 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

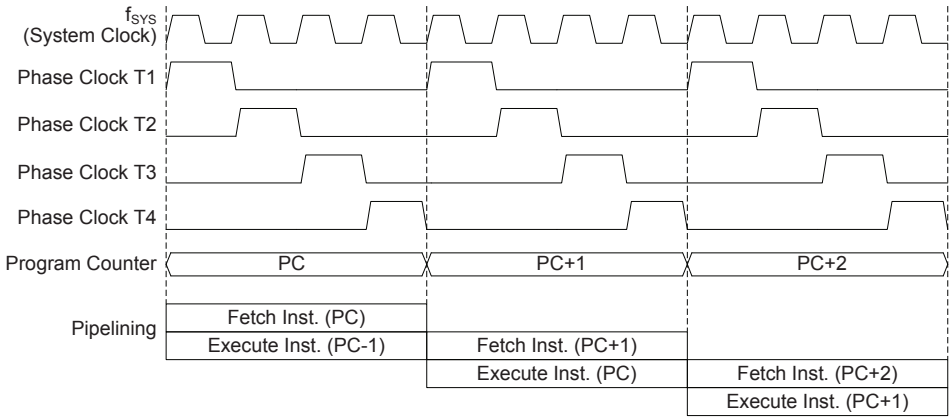


System Architecture

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

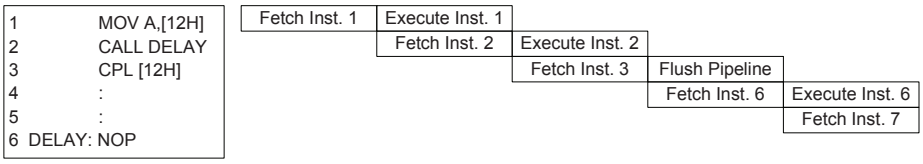
Clocking and Pipelining

The main system clock, derived from either a HXT, LXT, HIRC 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 Clocking 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.

Program Counter	
High Byte	**Low Byte (PCL)**
PC10~PC8	PC7~PC0

Program Counter

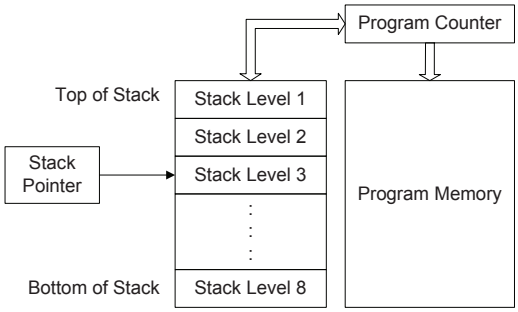
The lower byte of the Program Counter, known as the Program Counter Low register or PCL, is available for program control and is a readable and writeable register. By transferring data directly into this register, a short program jump can be executed directly; however, as only this low byte is available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

Stack

This is a special part of the memory which is used to save the contents of the Program Counter only. The stack is organized into 8 levels and is neither part of the data nor part of the program space, and is neither readable nor writeable. The activated level is indexed by the Stack Pointer, and is neither readable nor writeable. At a subroutine call or interrupt acknowledge signal, the contents of the Program Counter are pushed onto the stack. At the end of a subroutine or an interrupt routine, signaled by a return instruction, RET or RETI, the Program Counter is restored to its previous value from the stack. After a device reset, the Stack Pointer will point to the top of the stack.

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

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



Arithmetic and Logic Unit – ALU

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

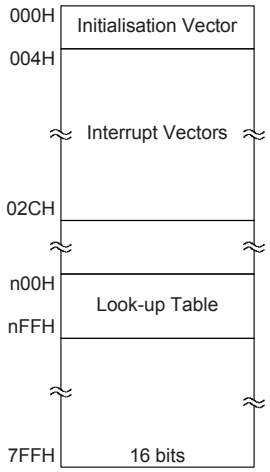
- Arithmetic operations:
  ADD, ADDM, ADC, ADCM, SUB, SUBM, SBC, SBCM, DAA
- Logic operations:
  AND, OR, XOR, ANDM, ORM, XORM, CPL, CPLA
- Rotation:
  RRA, RR, RRCA, RRC, RLA, RL, RLCA, RLC
- Increment and Decrement:
  INCA, INC, DECA, DEC
- Branch decision:
  JMP, SZ, SZA, SNZ, SIZ, SDZ, SIZA, SDZA,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

Flash Program Memory

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

Structure

The Program Memory has a capacity of 2K×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 registers.



Program Memory Structure

Special Vectors

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

Look-up Table

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

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

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

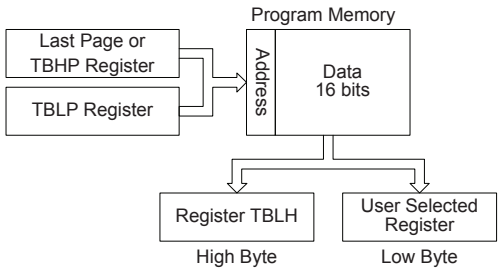


Table Program Example

The accompanying example shows how the table pointer and table data is defined and retrieved from the device. This example uses raw table data located in the last page which is stored there using the ORG statement. The value at this ORG statement is "0700H" which refers to the start address of the last page within the 2K Program Memory of the device. The table pointer low byte register is setup here to have an initial value of "06H". This will ensure that the first data read from the data table will be at the Program Memory address "0706H" or 6 locations after the start of the last page. Note that the value for the table pointer is referenced to the first address of the present page pointed by the TBHP register 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,07h         ; initialise high table pointer
mov  tbhp,a
:
tabrd   tempreg1   ; transfers value in table referenced by table pointer data at program
                   ; memory address "0706H" 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 "0705H" transferred to tempreg2 and TBLH in this
                   ; example the data "1AH" is transferred to tempreg1 and data "0FH" to
                   ; register tempreg2
:
org     0700h      ; sets initial address of program memory
dc      00Ah, 00Bh, 00Ch, 00Dh, 00Eh, 00Fh, 01Ah, 01Bh
:
```

In Circuit Programming – ICP

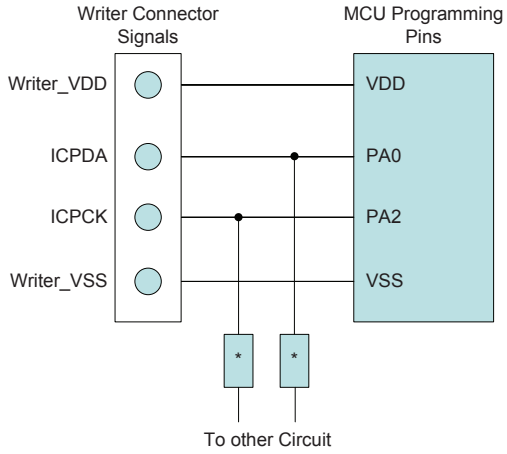
The provision of Flash type Program Memory provides the user with a means of convenient and easy upgrades and modifications to their programs on the same device.

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

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

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

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



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

On-Chip Debug Support – OCDS

There is an EV chip named HT66V0176 which is used to emulate the real MCU device named HT66F0176. The EV chip device also provides the "On-Chip Debug" function to debug the real MCU device during development process. The EV chip and real MCU device, HT66V0176 and HT66F0176, are almost functional compatible except the "On-Chip Debug" function. Users can use the EV chip device to emulate the real MCU device behaviors 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 device for debugging, the corresponding pin functions shared with the OCDSDA and OCDSCK pins in the real 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 more detailed OCDS information, refer to the corresponding document named "Holtek e-Link for 8-bit MCU OCDS User's Guide".

Holtek e-Link Pins	EV Chip OCDS 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
VSS	VSS	Ground

Data Memory

The Data Memory is an 8-bit wide RAM internal memory and is the location where temporary information is stored.

Structure

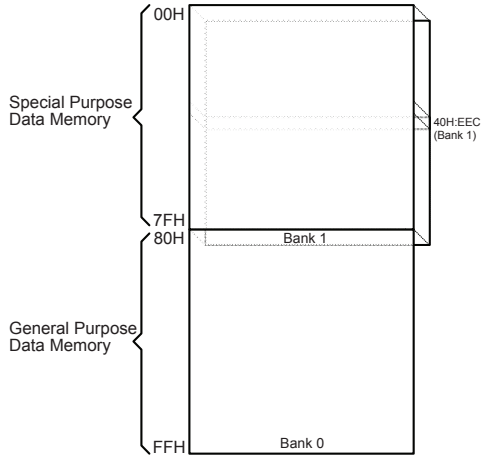
Divided into two parts, the first of these is an area of RAM, known as the Special Function Data Memory. Here are located registers which are necessary for correct operation of the device. Many of these registers can be read from and written to directly under program control, however, some remain protected 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 the device is the address 00H.

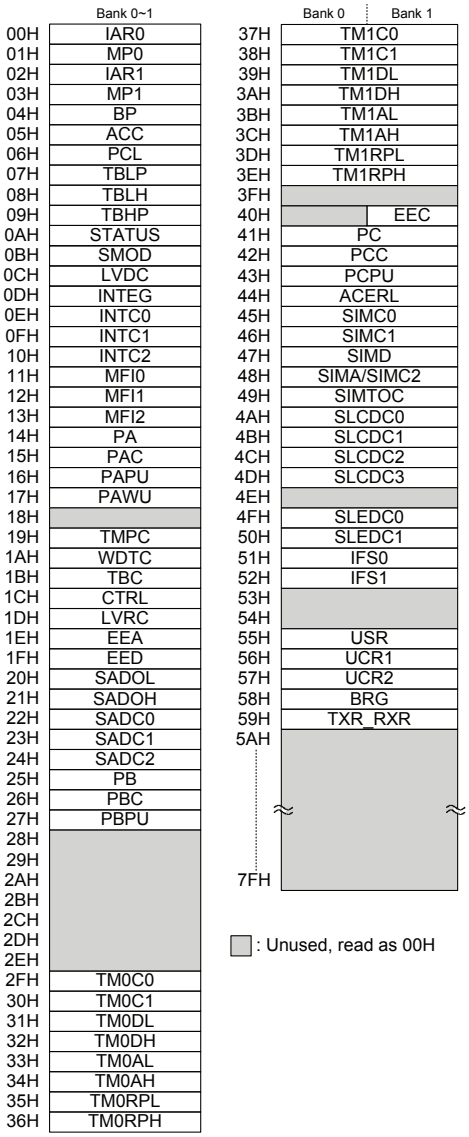
The address range of the Special Purpose Data Memory for the device is from 00H to 7FH while the address range of the General Purpose Data Memory is from 80H to FFH.

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

Data Memory Summary



Data Memory Structure



Special Purpose Data Memory Structure

Special Function Register Description

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

Indirect Addressing Registers – IAR0, IAR1

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

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

Indirect Addressing Program Example

```
data .section 'data'
adres1 db ?
adres2 db ?
adres3 db ?
adres4 db ?
block  db ?
code .section at 0 'code'
org 00h
start:
mov a,04h                ; setup size of block
mov block,a
mov a,offset adres1      ; Accumulator loaded with first RAM address
mov mp0,a                ; setup memory pointer with first RAM address
loop:
clr IAR0                 ; clear the data at address defined by MP0
inc mp0                  ; increment memory pointer
sdz block                ; check if last memory location has been cleared
jmp loop
continue:
     :
```

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

Bank Pointer – BP

For this device, the Data Memory is divided into 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. 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 pointer and indicates the location where the table data is located. Their value must be setup before any table read commands are executed. Their value can be changed, for example using the "INC" or "DEC" instructions, allowing for easy table data pointing and reading. TBLH is the location where the high order byte of the table data is stored after a table read data instruction has been executed. Note that the lower order table data byte is transferred to a user defined location.

Status Register – STATUS

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

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

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

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

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

STATUS Register

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

"x": unknown

Bit 7~6 Unimplemented, read as "0"

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

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

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 substraction

Bit 0 **C**: Carry flag
0: No carry-out
1: An operation results in a carry during an addition operation or if a borrow does not take place during a subtraction operation

The "C" flag is also affected by a rotate through carry instruction.

EEPROM Data Memory

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

EEPROM Data Memory Structure

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

EEPROM Registers

Three registers control the overall operation of the internal EEPROM Data Memory. These are the address registers, EEA, the data register, EED and a single control register, EEC. As the EEA and EED registers are located in bank 0, they can be directly accessed in the same was as any other Special Function Register. The EEC register, however, being located in bank 1, can 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 register must first be set to the value 40H and the Bank Pointer register, BP, set to the value, 01H, before any operations on the EEC register are executed.

Register Name	Bit							
	7	6	5	4	3	2	1	0
EEA	—	—	EEA5	EEA4	EEA3	EEA2	EEA1	EEA0
EED	D7	D6	D5	D4	D3	D2	D1	D0
EEC	—	—	—	—	WREN	WR	RDEN	RD

EEPROM Registers List

EEA Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	EEA5	EEA4	EEA3	EEA2	EEA1	EEA0
R/W	—	—	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	—	0	0	0	0	0	0

Bit 7~6 Unimplemented, read as "0"

Bit 5~0 **EEA5~EEA0**: Data EEPROM address low byte register
Data EEPROM address bit 5 ~ bit 0

EED Register

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

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

EEC Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	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**: Data 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**: Data 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

The EEPROM address of the data to be written must first be placed in the EEA register and the data placed in the EED register. To write data to the EEPROM, the write enable bit, WREN, in the EEC register must first be set high to enable the write function. After this, the WR bit in the EEC register must be immediately set high to initiate a write cycle. These two instructions must be executed 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 device is powered on, the Write Enable bit in the control register will be cleared preventing any write operations. Also at power-on the Bank Pointer register, BP, will be reset to zero, which means that Data Memory bank 0 will be selected. As the EEPROM control register is located in bank 1, this adds a further measure of protection against spurious write operations. During normal program operation, ensuring that the Write Enable bit in the control register is cleared will safeguard against incorrect write operations.

EEPROM Interrupt

The EEPROM write interrupt is generated when an EEPROM write cycle has ended. The EEPROM interrupt must first be enabled by setting the DEE bit in the relevant interrupt register. However, as the EEPROM is contained within a Multi-function Interrupt, the associated multi-function interrupt enable bit must also be set. When an EEPROM write cycle ends, the DEF request flag and its associated multi-function interrupt request flag will both be set. If the global, EEPROM and Multi-function interrupts are enabled and the stack is not full, a jump to the associated Multi-function Interrupt vector will take place. When the interrupt is serviced only the Multi-function interrupt flag will be automatically reset, the EEPROM interrupt flag must be manually reset by the application program.

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 register could be normally cleared to zero as this would inhibit access to bank1 where 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 device should not enter the IDLE or SLEEP mode until the EEPROM read or write operation is totally complete. Otherwise, the EEPROM read or write operation will fail.

Programming Examples

- **Reading data from the EEPROM – polling method**

```
MOV  A, EEPROM_ADRES          ; user defined address
MOV  EEA, A
MOV  A, 40H                   ; setup memory pointer MP1
MOV  MP1, A                   ; MP1 points to EEC register
MOV  A, 01H                   ; setup Bank Pointer BP
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, 40H                   ; setup memory pointer MP1
MOV  MP1, A                   ; MP1 points to EEC register
MOV  A, 01H                   ; setup Bank Pointer BP
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
SET  EMI
BACK:
SZ   IAR1.2                   ; check for write cycle end
JMP  BACK
CLR  IAR1                     ; disable EEPROM write
CLR  BP
```

Oscillators

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

Oscillator Overview

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

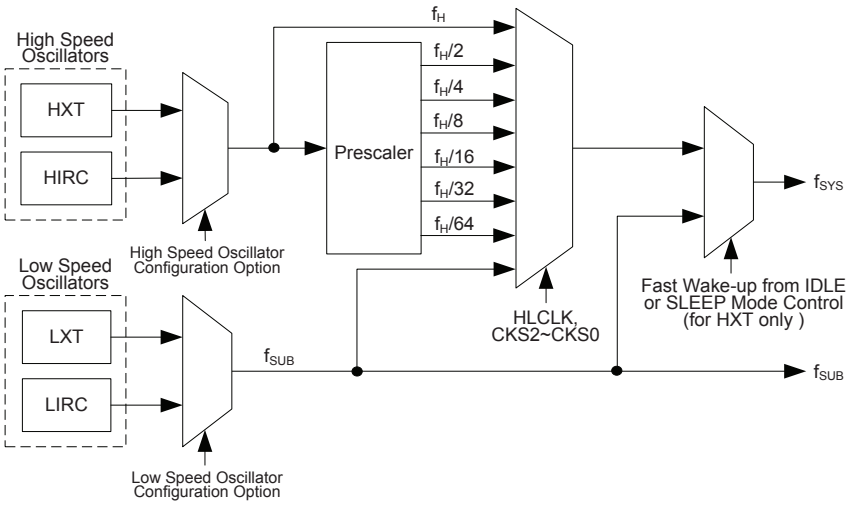
Type	Name	Frequency	Pins
High Speed External Crystal	HXT	400 kHz~20 MHz	OSC1/OSC2
High Speed Internal RC	HIRC	8/12/16 MHz	—
Low Speed External Crystal	LXT	32.768 kHz	XT1/XT2
Low Speed Internal RC	LIRC	32 kHz	—

Oscillator Types

System Clock Configurations

There are four methods of generating the system clock, two high speed oscillators and two low speed oscillators for the device. The high speed oscillator is the external crystal/ceramic oscillator, HXT, and the internal 8/12/16 MHz RC oscillator, HIRC. The two low speed oscillators are the internal 32 kHz RC oscillator, LIRC, and the external 32.768 kHz 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 each of the high and low speed oscillators is chosen via configuration options. The frequency of the slow speed or high speed system clock is also determined using the HLCLK bit and CKS2~CKS0 bits in the SMOD register. Note that two oscillator selections must be made namely one high speed and one low speed system oscillators. It is not possible to choose a no-oscillator selection for either the high or low speed oscillator. The OSC1 and OSC2 pins are used to connect the external components for the external crystal.

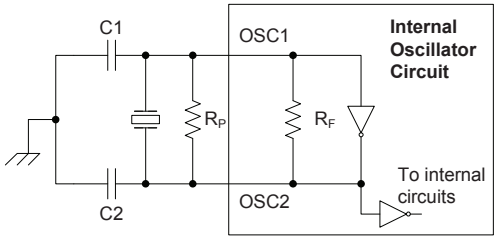


System Clock Configurations

External Crystal/Ceramic Oscillator – HXT

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

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



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

Crystal/Resonator Oscillator

HXT Oscillator C1 and C2 Values		
Crystal Frequency	**C1**	**C2**
16MHz	0 pF	0 pF
12MHz	0 pF	0 pF
8MHz	0 pF	0 pF
4MHz	0 pF	0 pF
1MHz	100 pF	100 pF
Note: C1 and C2 values are for guidance only.		

Crystal Recommended Capacitor Values

Internal High Speed RC Oscillator – HIRC

The internal RC oscillator is a fully integrated system oscillator requiring no external components. The internal RC oscillator has three fixed frequencies of 8MHz, 12MHz and 16MHz. Device trimming during the manufacturing process and the inclusion of internal frequency compensation circuits are used to ensure that the influence of the power supply voltage, temperature and process variations on the oscillation frequency are minimised. As a result, at a power supply of either 3V or 5V and at a temperature of 25°C degrees, the fixed oscillation frequency of 8MHz, 12MHz or 16MHz will have a tolerance within 1%. Note that if this internal system clock option is selected, as it requires no external pins for its operation, I/O pins are free for use as normal I/O pins.

External 32.768 kHz Crystal Oscillator – LXT

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

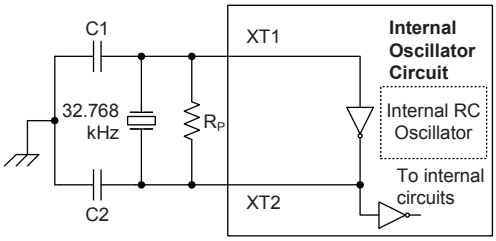
When the microcontroller enters the SLEEP or IDLE Mode, the system clock is switched off to stop microcontroller activity and to conserve power. However, in many microcontroller applications it may be necessary to keep the internal timers operational even when the microcontroller is in the SLEEP or IDLE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

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

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

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

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



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

External LXT Oscillator

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

32.768 kHz 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.

Internal 32kHz Oscillator – LIRC

The Internal 32kHz System Oscillator is one of the low frequency oscillator choices, which is selected via configuration option. It is a fully integrated RC oscillator with a typical frequency of 32kHz from 2.2V to 5.5V, requiring no external components for its implementation. Device trimming during the manufacturing process and the inclusion of internal frequency compensation circuits are used to ensure that the influence of the power supply voltage, temperature and process variations on the oscillation frequency are minimised.

Supplementary Oscillators

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

Operating Modes and System Clocks

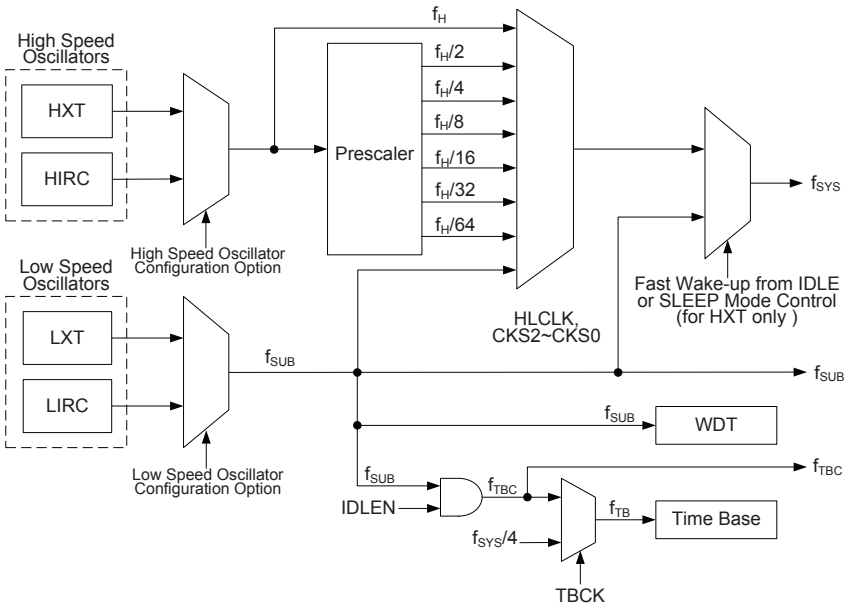
Present day applications require that their microcontrollers have high performance but often still demand that they consume as little power as possible, conflicting requirements that are especially true in battery powered portable applications. The fast clocks required for high performance will by their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

System Clocks

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

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

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



Device Clock Configurations

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

System Operation Modes

There are six different operation modes for the microcontroller, each one with its own special characteristics and which can be chosen according to the specific performance and power requirements of the application. There are two modes allowing normal operation of the microcontroller, the FAST Mode and SLOW Mode. The remaining four modes, the SLEEP0, SLEEP1, IDLE0 and IDLE1 Mode, are used when the microcontroller CPU is switched off to conserve power.

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

FAST Mode

As the name suggests this is one of the main operating modes where the microcontroller has all of its functions operational and where the system clock is provided by one of the high speed oscillators. This mode operates allowing the microcontroller to operate normally with a clock source will come from one of the high speed oscillators, either the HXT or HIRC oscillators. The high speed oscillator will however first be divided by a ratio ranging from 1 to 64, the actual ratio being selected by the CKS2~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 one of the low speed oscillators, either the LXT or the LIRC. Running the microcontroller in this mode allows it to run with much lower operating currents. In the SLOW Mode, the f_H is off.

SLEEP0 Mode

The 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, the f_{SUB} clock will also be stopped and the Watchdog Timer function is disabled. In this mode, the LVDEN must be set to "0". If the LVDEN is set to "1", it won't enter the SLEEP0 Mode.

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} clock will continue to operate if the LVDEN is "1" or 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 CTRL 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 an HALT instruction is executed and when the IDLEN bit in the SMOD register is high and the FSYSON bit in the CTRL 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.

Control Registers

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

SMOD Register

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

Bit 7~5 **CKS2~CKS0**: System clock selection when HLCLK is "0"
000: f_{SUB}
001: f_{SUB}
010: $f_H/64$
011: $f_H/32$
100: $f_H/16$
101: $f_H/8$
110: $f_H/4$
111: $f_H/2$

These three bits are used to select which clock is used as the system clock source. In addition to the system clock source, which can be either the LXT or LIRC, a divided version of the high speed system oscillator can also be chosen as the system clock source.

Bit 4 **FSTEN**: Fast Wake-up Control (only for HXT)
0: Disable
1: Enable

This is the Fast Wake-up Control bit which determines if the f_{SUB} clock source is initially used after the device wakes up. When the bit is high, the f_{SUB} clock source can be used as a temporary system clock to provide a faster wake up time as the f_{SUB} clock is available.

Bit 3 **LTO**: Low speed system oscillator ready flag
0: Not ready
1: Ready

This is the low speed system oscillator ready flag which indicates when the low speed system oscillator is stable after power on reset or a wake-up has occurred. The flag will be low when in the SLEEP0 Mode but after a wake-up has occurred, the flag will change to a high level after 128 clock cycles if the LXT oscillator is used and 1~2 clock cycles if the LIRC oscillator is used.

Bit 2 **HTO**: High speed system oscillator ready flag
0: Not ready
1: Ready

This is the high speed system oscillator ready flag which indicates when the high speed system oscillator is stable. This flag is cleared to "0" by hardware when the device is powered on and then changes to a high level after the high speed system oscillator is stable.

Therefore this flag will always be read as "1" by the application program after device power-on. The flag will be low when in the SLEEP or IDLE0 Mode but after a wake-up has occurred, the flag will change to a high level after 128 clock cycles if the HXT oscillator is used and after 15~16 clock cycles if the HIRC oscillator is used.

Bit 1 **IDLEN**: IDLE mode control
0: Disable
1: Enable
This is the IDLE mode control bit and determines what happens when the HALT instruction is executed. If this bit is high, when a HALT instruction is executed, the device will enter the IDLE mode. In the IDLE mode the CPU will stop running but the system clock will continue to keep the peripheral functions operational, if the FSYSON bit is high. If the 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_{SUB}
1: f_H
This bit is used to select if the f_H clock or the $f_H/2$~$f_H/64$ or f_{SUB} 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_{SUB} clock will be selected. When system clock switches from the f_H clock to the f_{SUB} clock and the f_H clock will be automatically switched off to conserve power.

CTRL Register

Bit	7	6	5	4	3	2	1	0
Name	FSYSON	—	—	—	—	LVRF	LRF	WRF
R/W	R/W	—	—	—	—	R/W	R/W	R/W
POR	0	—	—	—	—	x	0	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 6~3 Unimplemented, read as "0"

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

Bit 1 **LRF**: LVR control register software reset flag
Described elsewhere.

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

Fast Wake-up

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

If the HXT oscillator is selected as the FAST Mode system clock, and if the Fast Wake-up function is enabled, then it will take one to two t_{SUB} clock cycles of the LIRC or LXT oscillator for the system to wake-up. The system will then initially run under the f_{SUB} clock source until 128 HXT clock cycles have elapsed, at which point the HTO flag will switch high and the system will switch over to operating from the HXT oscillator.

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

System Oscillator	FSTEN Bit	Wake-up Time (SLEEP0 Mode)	Wake-up Time (SLEEP1 Mode)	Wake-up Time (IDLE0 Mode)	Wake-up Time (IDLE1 Mode)
HXT	0	128 HXT cycles	128 HXT cycles		1~2 HXT cycles
	1	128 HXT cycles	1~2 f_{SUB} cycles (System runs with f_{SUB} first for 128 HXT cycles and then switches over to run with the HXT clock)		1~2 HXT cycles
HIRC	x	15~16 HIRC cycles	15~16 HIRC cycles		1~2 HIRC cycles
LIRC	x	1~2 LIRC cycles	1~2 LIRC cycles		1~2 LIRC cycles
LXT	x	128 HXT cycles	1~2 LXT cycles		1~2 LXT cycles

"x": don't care

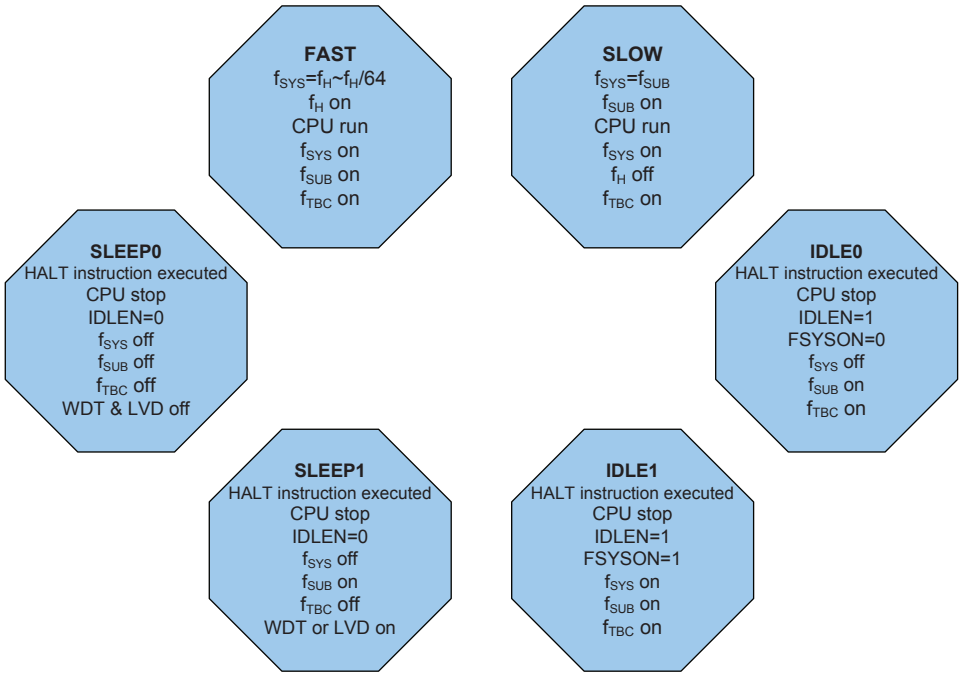
Wake-up Times

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

Operating Mode Switching

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

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

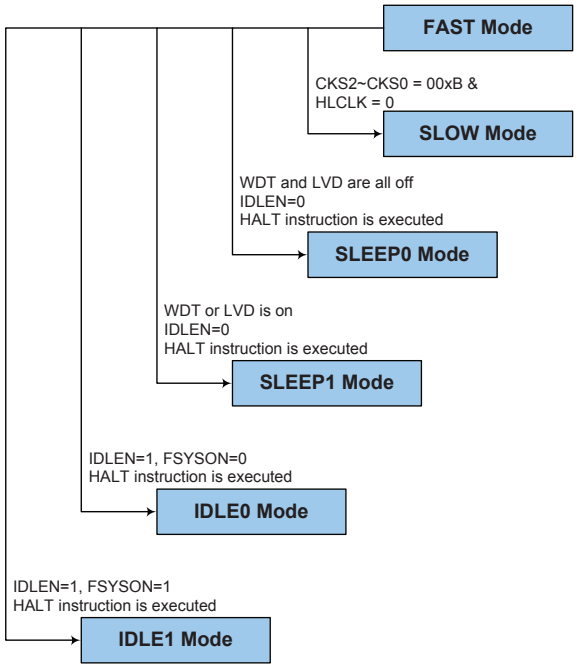


When the HLCLK bit switches to a low level, which implies that clock source is switched from the high speed clock, f_H, to the clock source, $f_H/2 \sim f_H/64$ or f_{SUB}. If the clock is from the f_{SUB}, the high speed clock source will stop running to conserve power. When this happens, it must be 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. The accompamying chart shows what happens when the device moves between the various operating modes.

FAST Mode to SLOW Mode Switching

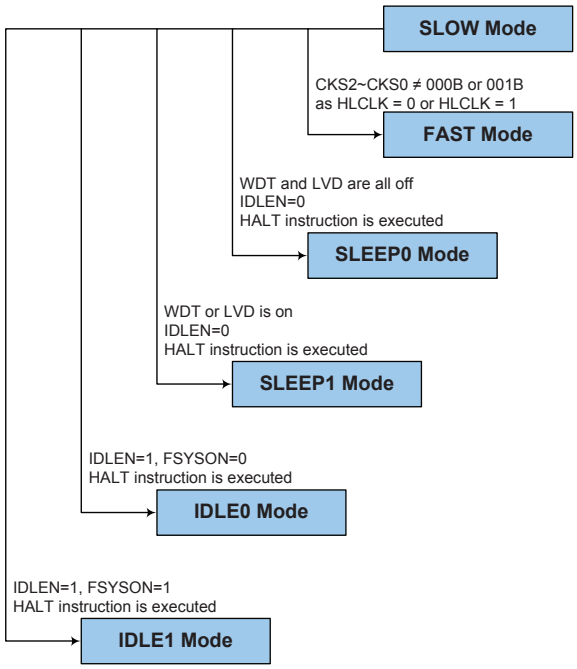
When running in the FAST Mode, which uses the high speed system oscillator, and therefore consumes more power, the system clock can switch to run in the SLOW Mode by set the HLCLK bit to "0" and set the CKS2~CKS0 bits to "000" or "001" in the SMOD register. This will then use the low speed system oscillator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

The SLOW Mode is sourced from the LXT or LIRC oscillator determined by the configuration option 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 FAST Mode Switching

In SLOW mode the system uses either the LXT or LIRC low speed system oscillator. To switch back to the FAST 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 field is set to "010", "011", "100", "101", "110" or "111". As a certain amount of time will be required for high speed system oscillator stabilization depends upon which high speed system oscillator type is used.



Entering the SLEEP0 Mode

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

- The system clock, WDT clock and Time Base clock will be stopped and the application program will stop at the "HALT" instruction.
- The Data Memory contents and registers will maintain their present condition.
- The WDT will be cleared and stopped no matter the WDT clock source orginates from the LXT or LIRC oscillator.
- The I/O ports will maintain their present conditions.
- In the status register, the Power Down flag PDF will be set, and WDT timeout flag TO will be cleared.

Entering the SLEEP1 Mode

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

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

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 the SMOD register equal to "1" and the FSYSON bit in the CTRL 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 f_{TBC} and f_{SUB} clocks 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 function is enabled.
- The I/O ports will maintain their present conditions.
- In the status register, the Power Down flag PDF will be set, and WDT timeout flag TO will be cleared.

Entering the IDLE1 Mode

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

- The system clock, f_{TBC} and f_{SUB} clocks will be on but the application program will stop at the "HALT" instruction.
- The Data Memory contents and registers will maintain their present condition.
- The WDT will be cleared and resume counting if the WDT function is enabled.
- The I/O ports will maintain their present conditions.
- In the status register, the Power Down flag PDF will be set, and WDT timeout flag TO will be cleared.

Standby Current Considerations

As the main reason for entering the SLEEP or IDLE Mode is to keep the current consumption of the device to as low a value as possible, perhaps only in the order of several micro-amps except in the IDLE1 Mode, there are other considerations which must also be taken into account by the circuit designer if the power consumption is to be minimised. Special attention must be made to the I/O pins on the device. All high-impedance input pins must be connected to either a fixed high or low level as any floating input pins could create internal oscillations and result in increased current consumption. This also applies to the device which has different package types, as there may be unbonbed pins. These must either be setup as outputs or if setup as inputs must have pull-high resistors connected.

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

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

Wake-up

To minimise power consumption the device can enter the SLEEP or any IDLE Mode, where the CPU will be switched off. However, when the device is woken up again, it will take a considerable time for the original system oscillator to restart, stablise and allow normal operation to resume.

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

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

When the device executes the "HALT" instruction, it will enter the Power down mode and the PDF flag will be set to 1. The PDF flag will be cleared to 0 if the device experiences a system power-up or executes the clear Watchdog Timer instruction. If the system is woken up by a WDT overflow, a Watchdog Timer reset will be initiated and the TO flag will be set to 1. 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, other flags remain in their original status.

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

Programming Considerations

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

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

Watchdog Timer

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

Watchdog Timer Clock Source

The Watchdog Timer clock source is provided by the internal clock, f_{SUB}, which is in turn supplied by the LIRC or LXT oscillator. The LXT oscillator is supplied by an external 32.768 kHz crystal. The LIRC internal oscillator has an approximate frequency of 32 kHz at a supply voltage of 5V. However, it should be noted that this specified internal clock frequency can vary with V_{DD}, temperature and process variations. The Watchdog Timer source clock is then subdivided by a ratio of 2^8 to 2^{18} to give longer timeouts, the actual value being chosen using the WS2~WS0 bits in the WDTC register.

Watchdog Timer Control Register

A single register, WDTC, controls the required timeout period as well as the enable/disable operation. This register controls 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 enable control
10101: Disabled
01010: Enabled
Other values: Reset MCU
If these bits are changed due to adverse environmental conditions, the microcontroller will be reset. The reset operation will be activated after a delay time,t_{SRESET} and the WRF bit in the CTRL register will be set to 1.

Bit 2~0 **WS2~WS0**: WDT time-out period selection
000: $2^8/f_{SUB}$
001: $2^{10}/f_{SUB}$
010: $2^{12}/f_{SUB}$
011: $2^{14}/f_{SUB}$
100: $2^{15}/f_{SUB}$
101: $2^{16}/f_{SUB}$
110: $2^{17}/f_{SUB}$
111: $2^{18}/f_{SUB}$
These three bits determine the division ratio of the watchdog timer source clock, which in turn determines the time-out period.

CTRL Register

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

"x": unknown

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

Bit 6~3 Unimplemented, read as "0"

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

Bit 1 **LRF**: LVR control register software reset flag
Described elsewhere.

Bit 0 **WRF**: WDT control register software reset flag
0: Not occurred
1: Occurred
This bit is set to 1 by the WDT control register software reset and cleared by the application program. 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 instruction. If the program malfunctions for whatever reason, jumps to an unknown location, or enters an endless loop, the clear instruction will not be executed in the correct manner, in which case the Watchdog Timer will overflow and reset the device. With regard to the Watchdog Timer enable/disable function, there are five bits, WE4~WE0, in the WDTC register to offer the enable/disable control and reset control of the Watchdog Timer. The WDT function will be disabled when the WE4~WE0 bits are set to a value of 10101B while the WDT function will be enabled if the WE4~WE0 bits are equal to 01010B. If the WE4~WE0 bits are set to any other values, except 01010B and 10101B, it will reset the device after a delay time, t_{SRESET}. After power on these bits will have a value of 01010B.

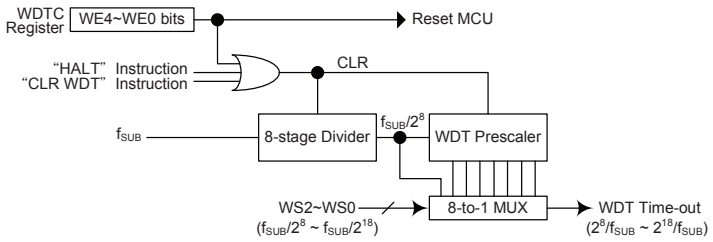
WE4 ~ WE0 Bits	WDT Function
10101B	Disable
01010B	Enable
Any other value	Reset MCU

Watchdog Timer Enable/Disable Control

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

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

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



Watchdog timer

Reset and Initialisation

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

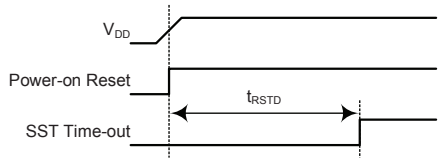
In addition to the power-on reset, another reset exists in the form of a Low Voltage Reset, LVR, where a full reset is implemented in situations where the power supply voltage falls below a certain threshold. Another type of reset is when the Watchdog Timer overflows and resets the microcontroller. All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

Reset Functions

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

Power-on Reset

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

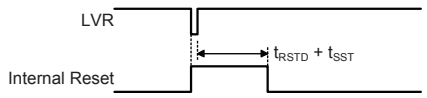


Note: t_{RSTD} is power-on delay with typical time=50 ms

Power-On Reset Timing Chart

Low Voltage Reset – LVR

The microcontroller contains a low voltage reset circuit in order to monitor the supply voltage of the device. The LVR function is always enabled with a specific LVR voltage, V_{LVR}. If the supply voltage of the device drops to within a range of 0.9V~V_{LVR} such as might occur when changing the battery, the LVR will automatically reset the device internally and the LVRF bit in the CTRL register will also be set to 1. For a valid LVR signal, a low supply voltage, i.e., a voltage in the range between 0.9V~ V_{LVR} must exist for a time greater than that specified by t_{LVR} in the LVD/LVR characteristics. If the low supply voltage state does not exceed this value, the LVR will ignore the low supply voltage and will not perform a reset function. The actual V_{LVR} value can be selected by the LVS7~LVS0 bits in the LVRC register. If the LVS7~LVS0 bits have any other value, which may perhaps occur due to adverse environmental conditions such as noise, the LVR will reset the device after a delay time,t_{SRESET}. When this happens, the LRF bit in the CTRL register will be set to 1. After power on the register will have the value of 01010101B. Note that the LVR function will be automatically disabled when the device enters the power down mode.



Note: t_{RSTD} is power-on delay with typical time=50 ms

Low Voltage Reset Timing Chart

- **LVRC Register**

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

Bit 7~0 **LVS7~LVS0**: LVR voltage select
01010101: 2.1V
00110011: 2.55V
10011001: 3.15V
10101010: 3.8V
Other values: Generates a MCU reset – register is reset to POR value

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

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

- **CTRL Register**

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

"x": unknown

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

Bit 6~3 Unimplemented, read as "0"

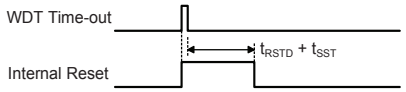
Bit 2 **LVRF**: LVR function reset flag
0: Not occurred
1: Occurred
This bit is set to 1 when a specific low voltage reset condition occurs. Note that this bit can only be cleared to 0 by the application program.

Bit 1 **LRF**: LVR control register software reset flag
0: Not occurred
1: Occurred
This bit is set to 1 by the LVRC control register contains any undefined LVR voltage register values. This in effect acts like a software-reset function. Note that this bit can only be cleared to 0 by the application program.

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

Watchdog Time-out Reset during Normal Operation

The Watchdog time-out Reset during normal operations in the FAST or SLOW mode is the same as the hardware Low Voltage Reset except that the Watchdog time-out flag TO will be set to "1".

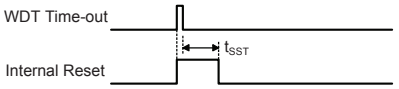


Note: t_{RSTD} is power-on delay with typical time=16.7 ms

WDT Time-out Reset during Normal Operation Timing Chart

Watchdog Time-out Reset during SLEEP or IDLE Mode

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



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

Reset Initial Conditions

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

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

"u" stands for unchanged

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

Item	Reset Function
Program Counter	Reset to zero
Interrupts	All interrupts will be disabled
WDT, Time Bases	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 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	Reset (Power On)	LVR Reset	WDT Time-out (FAST)	WDT Time-out (IDLE/SLEEP)
IAR0	---- ----	---- ----	---- ----	---- ----
MP0	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
IAR1	---- ----	---- ----	---- ----	---- ----
MP1	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
BP	---- --0	---- --0	---- --0	---- ---u
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
PCL	0000 0000	0000 0000	0000 0000	0000 0000
TBLP	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLH	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBHP	---- -xxx	---- -uuu	---- -uuu	---- -uuu
STATUS	--00 xxxx	--uu uuuu	--1u uuuu	--11 uuuu
SMOD	0000 0011	0000 0011	0000 0011	uuuu uuuu
LVDC	--00 -000	--00 -000	--00 -000	--uu -uuu
INTEG	---- 0000	---- 0000	---- 0000	---- uuuu
INTC0	-0-0 0-00	-0-0 0-00	-0-0 0-00	-u-u u-uu
INTC1	0000 0000	0000 0000	0000 0000	uuuu uuuu
INTC2	0000 0000	0000 0000	0000 0000	uuuu uuuu
MFI0	--00 --00	--00 --00	--00 --00	--uu --uu
MFI1	--00 --00	--00 --00	--00 --00	--uu --uu
MFI2	--00 --00	--00 --00	--00 --00	--uu --uu
PA	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAC	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAPU	0000 0000	0000 0000	0000 0000	uuuu uuuu
PAWU	0000 0000	0000 0000	0000 0000	uuuu uuuu
TMPC	0--- --00	0--- --00	0--- --00	u--- --uu
WDTC	0101 0011	0101 0011	0101 0011	uuuu uuuu
TBC	0011 0111	0011 0111	0011 0111	uuuu uuuu
CTRL	0--- -x00	0--- -100	0--- -000	u--- -uuu
LVRC	0101 0101	0101 0101	0101 0101	uuuu uuuu
EEA	--00 0000	--00 0000	--00 0000	--uu uuuu
EED	0000 0000	0000 0000	0000 0000	uuuu uuuu
SADOL	xxxx ----	xxxx ----	xxxx ----	uuuu ---- (ADRFS=0) uuuu uuuu (ADRFS=1)
SADOH	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu (ADRFS=0) ---- uuuu (ADRFS=1)
SADC0	0000 -000	0000 -000	0000 -000	uuuu -uuu
SADC1	000- -000	000- -000	000- -000	uuu- -uuu
SADC2	0000 0000	0000 0000	0000 0000	uuuu uuuu
PB	-111 1111	-111 1111	-111 1111	-uuu uuuu
PBC	-111 1111	-111 1111	-111 1111	-uuu uuuu
PBPU	-000 0000	-000 0000	-000 0000	-uuu uuuu
TM0C0	0000 0---	0000 0---	0000 0---	uuuu u---
TM0C1	0000 0000	0000 0000	0000 0000	uuuu uuuu
TM0DL	0000 0000	0000 0000	0000 0000	uuuu uuuu

Register	Reset (Power On)	LVR Reset	WDT Time-out (FAST)	WDT Time-out (IDLE/SLEEP)
TM0DH	---- --00	---- --00	---- --00	---- --uu
TM0AL	0000 0000	0000 0000	0000 0000	uuuu uuuu
TM0AH	---- --00	---- --00	---- --00	---- --uu
TM0RPL	0000 0000	0000 0000	0000 0000	uuuu uuuu
TM0RPH	---- --00	---- --00	---- --00	---- --uu
TM1C0	0000 0---	0000 0---	0000 0---	uuuu u---
TM1C1	0000 0000	0000 0000	0000 0000	uuuu uuuu
TM1DL	0000 0000	0000 0000	0000 0000	uuuu uuuu
TM1DH	---- --00	---- --00	---- --00	---- --uu
TM1AL	0000 0000	0000 0000	0000 0000	uuuu uuuu
TM1AH	---- --00	---- --00	---- --00	---- --uu
TM1RPL	0000 0000	0000 0000	0000 0000	uuuu uuuu
TM1RPH	---- --00	---- --00	---- --00	---- --uu
PC	-111 1111	-111 1111	-111 1111	-uuu uuuu
PCC	-111 1111	-111 1111	-111 1111	-uuu uuuu
PCPU	-000 0000	-000 0000	-000 0000	-uuu uuuu
ACERL	1111 1111	1111 1111	1111 1111	uuuu uuuu
SIMC0	111- 0000	111- 0000	111- 0000	uuu- uuuu
SIMC1	1000 0001	1000 0001	1000 0001	uuuu uuuu
SIMD	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
SIMA/SIMC2	0000 0000	0000 0000	0000 0000	uuuu uuuu
SIMTOC	0000 0000	0000 0000	0000 0000	uuuu uuuu
SLCDC0	0000 0000	0000 0000	0000 0000	uuuu uuuu
SLCDC1	0000 0000	0000 0000	0000 0000	uuuu uuuu
SLCDC2	0000 0000	0000 0000	0000 0000	uuuu uuuu
SLCDC3	0000 0000	0000 0000	0000 0000	uuuu uuuu
SLEDC0	0101 0101	0101 0101	0101 0101	uuuu uuuu
SLEDC1	---- 0101	---- 0101	---- 0101	---- uuuu
IFS0	--00 0000	--00 0000	--00 0000	--uu uuuu
IFS1	---- 0000	---- 0000	---- 0000	---- uuuu
USR	0000 1011	0000 1011	0000 1011	uuuu uuuu
UCR1	0000 00x0	0000 00x0	0000 00x0	uuuu uuuu
UCR2	0000 0000	0000 0000	0000 0000	uuuu uuuu
BRG	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TXR_RXR	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
EEC	---- 0000	---- 0000	---- 0000	---- uuuu

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

Input/Output Ports

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

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

Register Name	Bit							
	7	6	5	4	3	2	1	0
PAWU	PAWU7	PAWU6	PAWU5	PAWU4	PAWU3	PAWU2	PAWU1	PAWU0
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
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

"–": Unimplemented, read as "0"

I/O Logic Function Registers List

Pull-high Resistors

Many product applications require pull-high resistors for their switch inputs usually requiring the use of an external resistor. To eliminate the need for these external resistors, all I/O pins, when configured as an input have the capability of being connected to an internal pull-high resistor. These pull-high resistors are selected using the relevant pull-high control registers and are implemented using weak PMOS transistors.

PxPU Register

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

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

0: Disable

1: Enable

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

Port A Wake-up

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

PAWU Register

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

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

I/O Port Control Registers

Each Port has its own control register, known as PAC~PCC, which controls the input/output configuration. With this control register, each I/O pin with or without pull-high resistors can be reconfigured dynamically under software control. For the I/O pin to function as an input, the corresponding bit of the control register must be written as a "1". This will then allow the logic state of the input pin to be directly read by instructions. When the corresponding bit of the control register is written as a "0", the I/O pin will be setup as a CMOS output. If the pin is currently setup as an output, instructions can still be used to read the output register.

However, it should be noted that the program will in fact only read the status of the output data latch and not the actual logic status of the output pin.

PxC Register

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

PxCn: I/O Port x Pin type selection

0: Output

1: Input

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

I/O Port Source Current Control

The device supports 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 Input/Output 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	—	—	—	—	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	1	0	1	0	1	0	1

Bit 7~6 **PBPS3~PBPS2**: PB6~PB4 source current selection
00: source current=Level 0 (min.)
01: source current=Level 1
10: source current=Level 2
11: source current=Level 3 (max.)

Bit 5~4 **PBPS1~PBPS0**: 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~PAPS2**: 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~PAPS0**: 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.)

SLEDC1 Register

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.)

Pin-remapping Function

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. Additionally there are two registers, IFS0 and IFS1, to establish certain pin functions.

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. If the pin-shared pin function have multiple outputs simultaneously, its pin names at the right side of the "/" sign can be used for higher priority.

IFS0 Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	SDOPS	SDI_SDAPS	SCK_SCLPS	SCSBPS	INT1PS	INT0PS
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 **SDOPS**: SDO pin-remapping selection
0: SDO on PC2
1: SDO on PA2

Bit 4 **SDI_SDAPS**: SDI/SDA pin-remapping selection
0: SDI/SDA on PC3
1: SDI/SDA on PA3

Bit 3 **SCK_SCLPS**: SCK/SCL pin-remapping selection
0: SCK/SCL on PC4
1: SCK/SCL on PB6

Bit 2 **SCSBPS**: $\overline{\text{SCS}}$ pin-remapping selection
0: $\overline{\text{SCS}}$ on PA1
1: $\overline{\text{SCS}}$ on PB5

Bit 1 **INT1PS**: INT1 pin-remapping selection
0: INT1 on PB1
1: INT1 on PC5

Bit 0 **INT0PS**: INT0 pin-remapping selection
0: INT0 on PB0
1: INT0 on PC6

IFS1 Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	TCK1PS	TCK0PS	TXPS	RXPS
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0

Bit 7~4 Unimplemented, read as "0"

Bit 3 **TCK1PS**: TCK1 pin-remapping selection
0: TCK1 on PA4
1: TCK1 on PA6

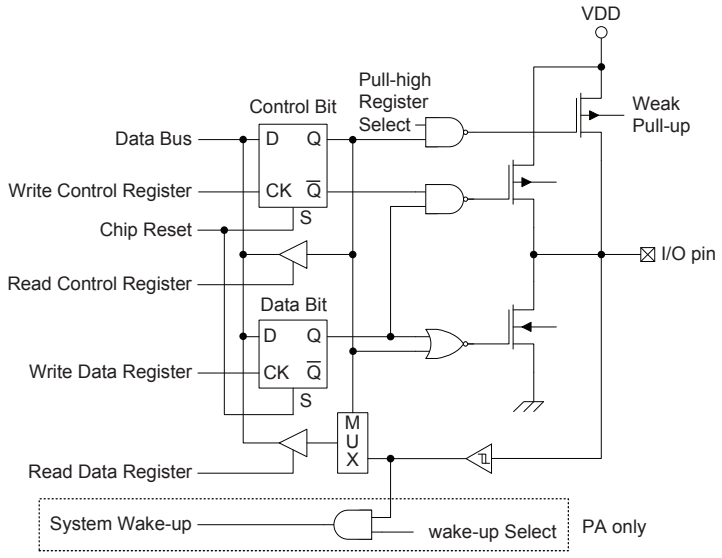
Bit 2 **TCK0PS**: TCK0 pin-remapping selection
0: TCK0 on PB2
1: TCK0 on PA5

Bit 1 **TXPS**: TX pin-remapping selection
0: TX on PC6
1: TX on PB3

Bit 0 **RXPS**: RX pin-remapping selection
0: RX on PC5
1: RX on PB4

I/O Pin Structures

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



Logic Function Input/Output Structure

Programming Considerations

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

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

Timer Modules – TM

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

Introduction

The device contains two TMs categorised as Periodic Type TM having a reference name of TM0 and TM1. The main features of PTM is summarised in the accompanying table.

TM Function	PTM
Timer/Counter	√
Input Capture	√
Compare Match Output	√
PWM Channels	1
Single Pulse Output	1
PWM Alignment	Edge
PWM Adjustment Period & Duty	Duty or Period

TM Function Summary

The device contains a specific number of Periodic Type TM unit which is shown in the table together with their individual reference name, TM0~TM1.

TM0	TM1
10-bit PTM	10-bit PTM

TM Name/Type Reference

TM Operation

The Periodic Type 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 count-up counter whose value is then compared with the value of pre-programmed internal comparators. When the free running count-up counter has the same value as the pre-programmed comparator, known as a compare match situation, a TM interrupt signal will be generated which can clear the counter and perhaps also change the condition of the TM output pin. The internal TM counter is driven by a user selectable clock source, which can be an internal clock or an external pin.

TM Clock Source

The clock source which drives the main counter in each TM can originate from various sources. The selection of the required clock source is implemented using the TnCK2~TnCK0 bits in the TM control registers. The clock source can be a ratio of the system clock, f_{SYS}, or the internal high clock, f_H, the f_{TBC} clock source or the external TCKn pin. The TCKn pin clock source is used to allow an external signal to drive the TM as an external clock source for event counting.

TM Interrupts

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

TM External Pins

The Periodic type TMs each has one TM input pin, with the label TCKn. The TM input pin is essentially a clock source for the TM and is selected using the TnCK2~TnCK0 bits in the TMnC0 register. This external TM input pin allows an external clock source to drive the internal TM. This external TM input pin is shared with other functions but will be connected to the internal TM if selected using the TnCK2~TnCK0 bits. The TM input pin can be chosen to have either a rising or falling active edge. The TCKn pin is also used as the external trigger input pin in single pulse output mode for the PTM.

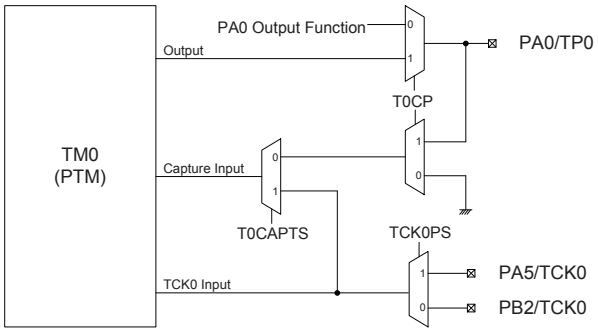
The Periodic type TMs each has one output pin with the label TPn. When the TM is in the Compare Match Output Mode, these pins can be controlled by the TM to switch to a high or low level or to toggle when a compare match situation occurs. The external TPn output pin is also the pin where the TM generates the PWM output waveform. The TPn pin acts as an input when the TM is setup to operate in the Capture Input Mode. As the TPn pins are pin-shared with other functions, the TPn pin function is enabled or disabled according to the internal TM on/off control, operation mode and output control settings. When the corresponding TM configuration selects the TPn pin to be used as an output pin, the associated pin will be setup as an external TM output pin. If the TM configuration selects the TPn pin to be setup as an input pin, the input signal supplied on the associated pin can be derived from an external signal and other pin-shared output function. If the TM configuration determines that the TPn pin function is not used, the associated pin will be controlled by other pin-shared functions. The details of the TPn pin for each TM type and device are provided in the accompanying table.

TM0(PTM)	TM1(PTM)	Register
TCK0; TP0	TCK1; TP1	TMPC

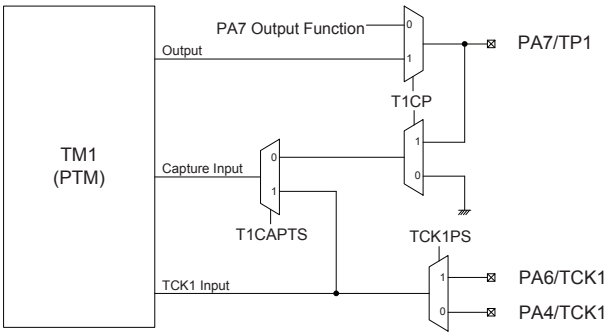
TM External Pins

TM Input/Output Pin Control Register

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



TM0 Function Pin Control Block Diagram



TM1 Function Pin Control Block Diagram

TMPC Register

Bit	7	6	5	4	3	2	1	0
Name	CLOP	—	—	—	—	—	T1CP	T0CP
R/W	R/W	—	—	—	—	—	R/W	R/W
POR	0	—	—	—	—	—	0	0

Bit 7 **CLOP**: CLO pin control
0: Disable
1: Enable

Bit 6~2 Unimplemented, read as "0"

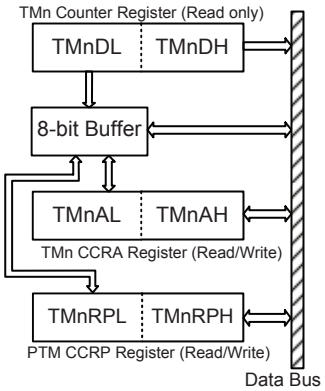
Bit 1 **T1CP**: TP1 pin control
0: Disable
1: Enable

Bit 0 **T0CP**: TP0 pin control
0: Disable
1: Enable

Programming Considerations

The TM Counter Registers and the Capture/Compare CCRA and CCRP registers, being either 10-bit or 16-bit, all have a low and high byte structure. The high bytes can be directly accessed, but as the low bytes can only be accessed via an internal 8-bit buffer, reading or writing to these register pairs must be carried out in a specific way. The important point to note is that data transfer to and from the 8-bit buffer and its related low byte only takes place when a write or read operation to its corresponding high byte is executed.

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



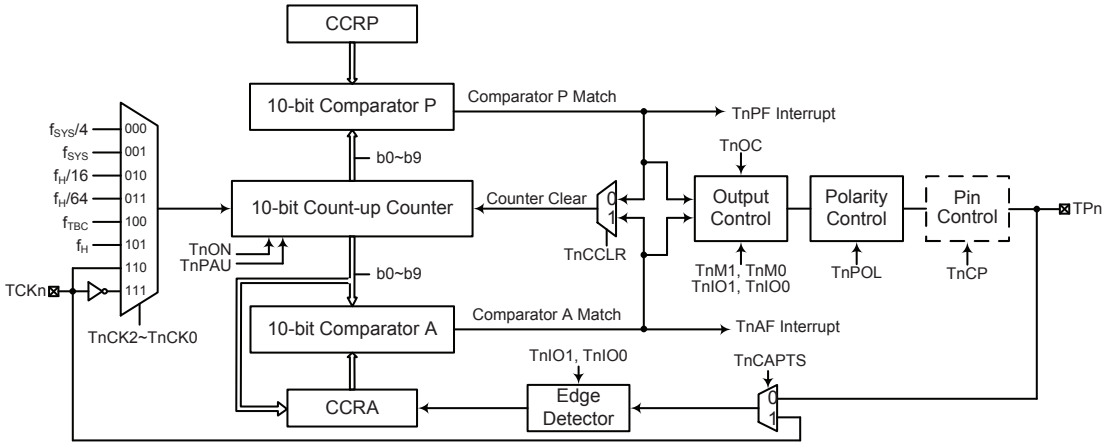
The following steps show the read and write procedures:

- Writing Data to CCRA or CCRP
  - Step 1. Write data to Low Byte TMnAL or TMnRPL
    – note that here data is only written to the 8-bit buffer.
  - Step 2. Write data to High Byte TMnAH or TMnRPH
    – here data is written directly to the high byte registers and simultaneously data is latched from the 8-bit buffer to the Low Byte registers.
- Reading Data from the Counter Registers and CCRA or CCRP
  - Step 1. Read data from the High Byte TMnDH, TMnAH or TMnRPH
    – 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 TMnDL, TMnAL or TMnRPL
    – this step reads data from the 8-bit buffer.

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 one external input pin and can drive one external output pin.

TM Core	TM No.	TM Input Pin	TM Output Pin
10-bit PTM	TM0, TM1	TCK0, TCK1	TP0, TP1



Periodic Type TM Block Diagram (n=0,1)

Note: The TCKn external pins are remapping on different pins, so before using the TMn function, the pin-remapping function register IFS1 must be set properly.

Periodic TM Operation

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

The only way of changing the value of the 10-bit counter using the application program is to clear the counter by changing the TnON bit from low to high. The counter will also be cleared automatically by a counter overflow or a compare match with one of its associated comparators. When these conditions occur, a TM interrupt signal will also usually be generated. The 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 pins. All operating setup conditions are selected using relevant internal registers.

Periodic Type TM Register Description

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

Register Name	Bit							
	7	6	6	4	3	2	1	0
TMnC0	TnPAU	TnCK2	TnCK1	TnCK0	TnON	—	—	—
TMnC1	TnM1	TnM0	TnIO1	TnIO0	TnOC	TnPOL	TnCAPTS	TnCCLR
TMnDL	D7	D6	D5	D4	D3	D2	D1	D0
TMnDH	—	—	—	—	—	—	D9	D8
TMnAL	D7	D6	D5	D4	D3	D2	D1	D0
TMnAH	—	—	—	—	—	—	D9	D8
TMnRPL	TnRP7	TnRP6	TnRP5	TnRP4	TnRP3	TnRP2	TnRP1	TnRP0
TMnRPH	—	—	—	—	—	—	TnRP9	TnRP8

Periodic TM Registers List (n =0,1)

TMnDL 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 TMn Counter Low Byte Register bit 7 ~ bit 0
TMn 10-bit Counter bit 7 ~ bit 0

TMnDH 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 TMn Counter High Byte Register bit 1 ~ bit 0
TMn 10-bit Counter bit 9 ~ bit 8

TMnAL 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 TMn CCRA Low Byte Register bit 7 ~ bit 0
TMn 10-bit CCRA bit 7 ~ bit 0

TMnAH 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 TMn CCRA High Byte Register bit 1 ~ bit 0
TMn 10-bit CCRA bit 9 ~ bit 8

TMnRPL Register

Bit	7	6	5	4	3	2	1	0
Name	TnRP7	TnRP6	TnRP5	TnRP4	TnRP3	TnRP2	TnRP1	TnRP0
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 **TnRP7~TnRP0**: TMn CCRP Low Byte Register bit 7 ~ bit 0
TMn 10-bit CCRP bit 7 ~ bit 0

TMnRPH Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	TnRP9	TnRP8
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2 Unimplemented, read as "0"
Bit 1~0 **TnRP9~TnRP8**: TMn CCRP High Byte Register bit 1 ~ bit 0
TMn 10-bit CCRP bit 9 ~ bit 8

TMnC0 Register

Bit	7	6	5	4	3	2	1	0
Name	TnPAU	TnCK2	TnCK1	TnCK0	TnON	—	—	—
R/W	R/W	R/W	R/W	R/W	R/W	—	—	—
POR	0	0	0	0	0	—	—	—

Bit 7 **TnPAU**: TMn 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 TMn 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 **TnCK2~TnCK0**: Select TMn Counter clock
000: $f_{SYS}/4$
001: f_{SYS}
010: $f_H/16$
011: $f_H/64$
100: f_{TBC}
101: f_H
110: TCKn rising edge clock
111: TCKn falling edge clock
These three bits are used to select the clock source for the TMn. 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 **TnON**: TMn Counter On/Off control
0: Off
1: On
This bit controls the overall on/off function of the TMn. Setting the bit high enables the counter to run while clearing the bit disables the TMn. Clearing this bit to zero will stop the counter from counting and turn off the TMn 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 TMn is in the Compare Match Output Mode then the TMn output pin will be reset to its initial condition, as specified by the TnOC bit, when the TnON bit changes from low to high.

Bit 2~0 Unimplemented, read as "0"

TMnC1 Register

Bit	7	6	5	4	3	2	1	0
Name	TnM1	TnM0	TnIO1	TnIO0	TnOC	TnPOL	TnCAPTS	TnCCLR
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 **TnM1~TnM0**: Select TMn Operating Mode
00: Compare Match Output Mode
01: Capture Input Mode
10: PWM Mode or Single Pulse Output Mode
11: Timer/Counter Mode
These bits setup the required operating mode for the TMn. To ensure reliable operation the TMn should be switched off before any changes are made to the TnM1 and TnM0 bits. In the Timer/Counter Mode, the TMn output pin control will be disabled.

Bit 5~4 **TnIO1~TnIO0**: Select TMn external pin TPn 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 TPn or TCKn
01: Input capture at falling edge of TPn or TCKn
10: Input capture at rising/falling edge of TPn or TCKn
11: Input capture disabled
Timer/Counter Mode
Unused
These two bits are used to determine how the TMn output pin changes state when a certain condition is reached. The function that these bits select depends upon in which mode the TMn is running.

In the Compare Match Output Mode, the TnIO1 and TnIO0 bits determine how the TMn output pin changes state when a compare match occurs from the Comparator A. The TMn 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 TMn output pin should be setup using the TnOC bit in the TMnC1 register. Note that the output level requested by the TnIO1 and TnIO0 bits must be different from the initial value setup using the TnOC bit otherwise no change will occur on the TMn output pin when a compare match occurs. After the TMn output pin changes state, it can be reset to its initial level by changing the level of the TnON bit from low to high.

In the PWM Mode, the TnIO1 and TnIO0 bits determine how the TMn output pin changes state when a certain compare match condition occurs. The TMn output function is modified by changing these two bits. It is necessary to only change the values of the TnIO1 and TnIO0 bits only after the TMn has been switched off. Unpredictable PWM outputs will occur if the TnIO1 and TnIO0 bits are changed when the TMn is running.

Bit 3 **TnOC**: TMn TPn Output control

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

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

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

Bit 2 **TnPOL**: TMn TPn Output polarity control
0: Non-inverted
1: Inverted

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

Bit 1 **TnCAPTS**: TMn Capture Triiger Source selection
0: From TPn pin
1: From TCKn pin

Bit 0 **TnCCLR**: TMn Counter Clear condition selection
0: Comparator P match
1: Comparator A match

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

Periodic Type TM Operation Modes

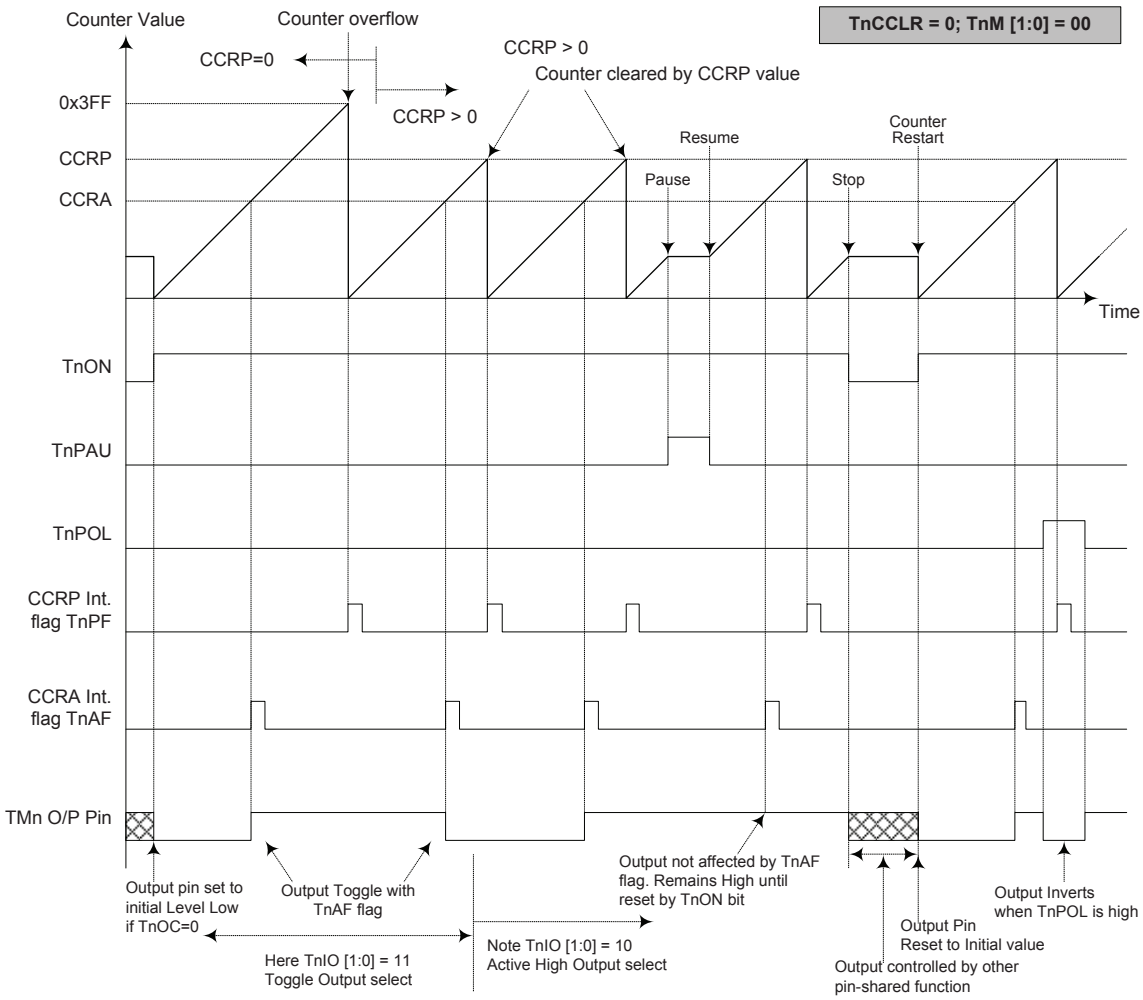
The Periodic Type TM can operate in one of five operating modes, Compare Match Output Mode, PWM Output Mode, Single Pulse Output Mode, Capture Input Mode or Timer/Counter Mode. The operating mode is selected using the TnM1 and TnM0 bits in the TMnC1 register.

Compare Match Output Mode

To select this mode, bits TnM1 and TnM0 in the TMnC1 register, should be set to 00 respectively. In this mode once the counter is enabled and running it can be cleared by three methods. These are a counter overflow, a compare match from Comparator A and a compare match from Comparator P. When the TnCCLR bit is low, there are two ways in which the counter can be cleared. One is when a compare match from Comparator P, the other is when the CCRP bits are all zero which allows the counter to overflow. Here both TnAF and TnPF interrupt request flags for Comparator A and Comparator P respectively, will both be generated.

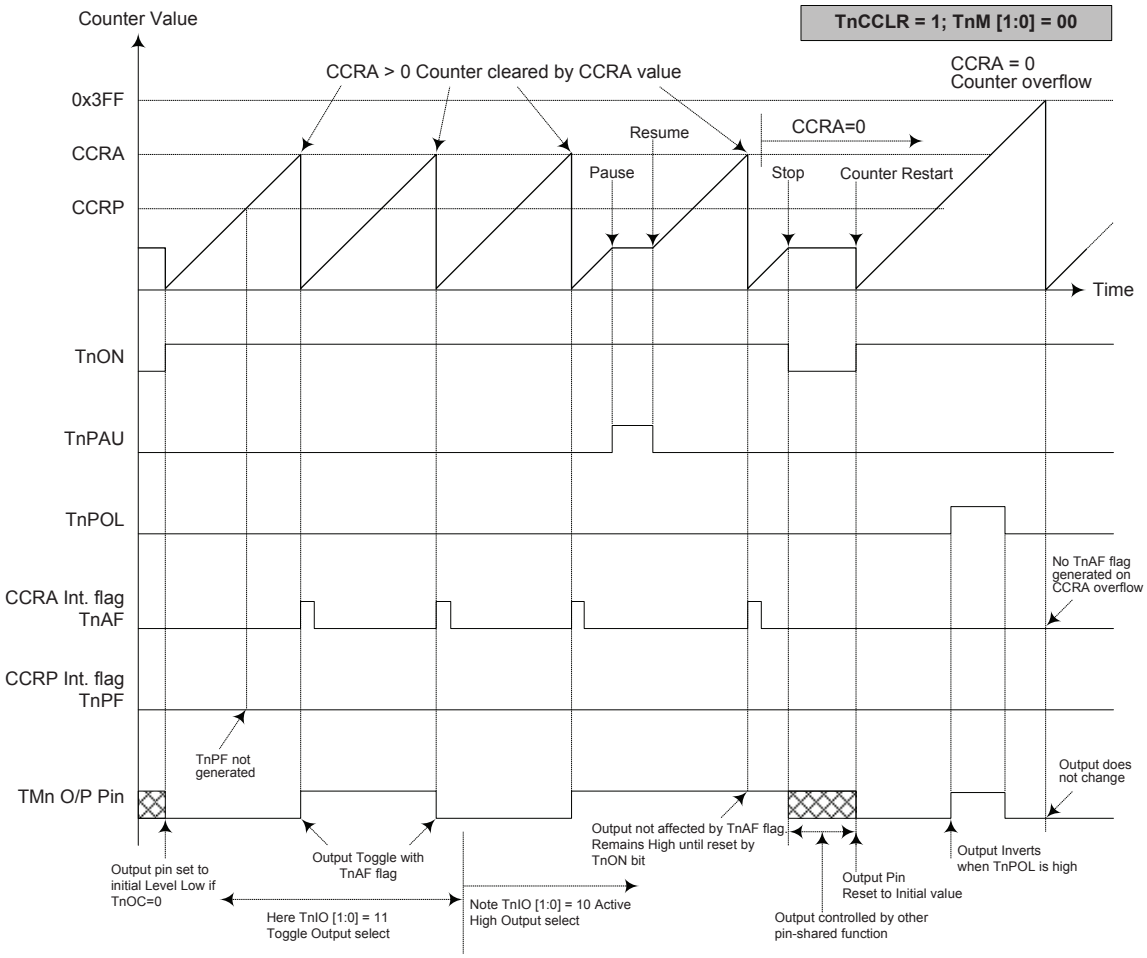
If the TnCCLR bit in the TMnC1 register is high then the counter will be cleared when a compare match occurs from Comparator A. However, here only the TnAF interrupt request flag will be generated even if the value of the CCRP bits is less than that of the CCRA registers. Therefore when TnCCLR is high no TnPF interrupt request flag will be generated. In the Compare Match Output Mode, the CCRA can not be set to "0".

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



Compare Match Output Mode – TnCCLR=0(n=0,1)

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



Compare Match Output Mode – TnCCLR=1(n=0,1)

Note: 1. With TnCCLR=1, a Comparator A match will clear the counter
2. The TMn output pin is controlled only by the TnAF flag
3. The output pin is reset to its initial state by a TnON bit rising edge
4. A TnPF flag is not generated when TnCCLR =1

Timer/Counter Mode

To select this mode, bits TnM1 and TnM0 in the TMnC1 register should be set to 11 respectively. The Timer/Counter Mode operates in an identical way to the Compare Match Output Mode generating the same interrupt flags. The exception is that in the Timer/Counter Mode the TMn 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 TMn output pin is not used in this mode, the pin can be used as a normal I/O pin or other pin-shared function.

PWM Output Mode

To select this mode, bits TnM1 and TnM0 in the TMnC1 register should be set to 10 respectively and also the TnIO1 and TnIO0 bits should be set to 10 respectively. The PWM function within the TMn 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 TMn output pin, a square wave AC waveform can be generated with varying equivalent DC RMS values.

As both the period and duty cycle of the PWM waveform can be controlled, the choice of generated waveform is extremely flexible. In the PWM mode, the TnCCLR bit has no effect as the PWM period. Both of the CCRP and CCRA registers are used to generate the PWM waveform, one register is used to clear the internal counter and thus control the PWM waveform frequency, while the other one is used to control the duty cycle. The PWM waveform frequency and duty cycle can therefore be controlled by the values in the CCRA and CCRP registers.

An interrupt flag, one for each of the CCRA and CCRP, will be generated when a compare match occurs from either Comparator A or Comparator P. The TnOC bit in the TMnC1 register is used to select the required polarity of the PWM waveform while the two TnIO1 and TnIO0 bits are used to enable the PWM output or to force the TMn output pin to a fixed high or low level. The TnPOL bit is used to reverse the polarity of the PWM output waveform.

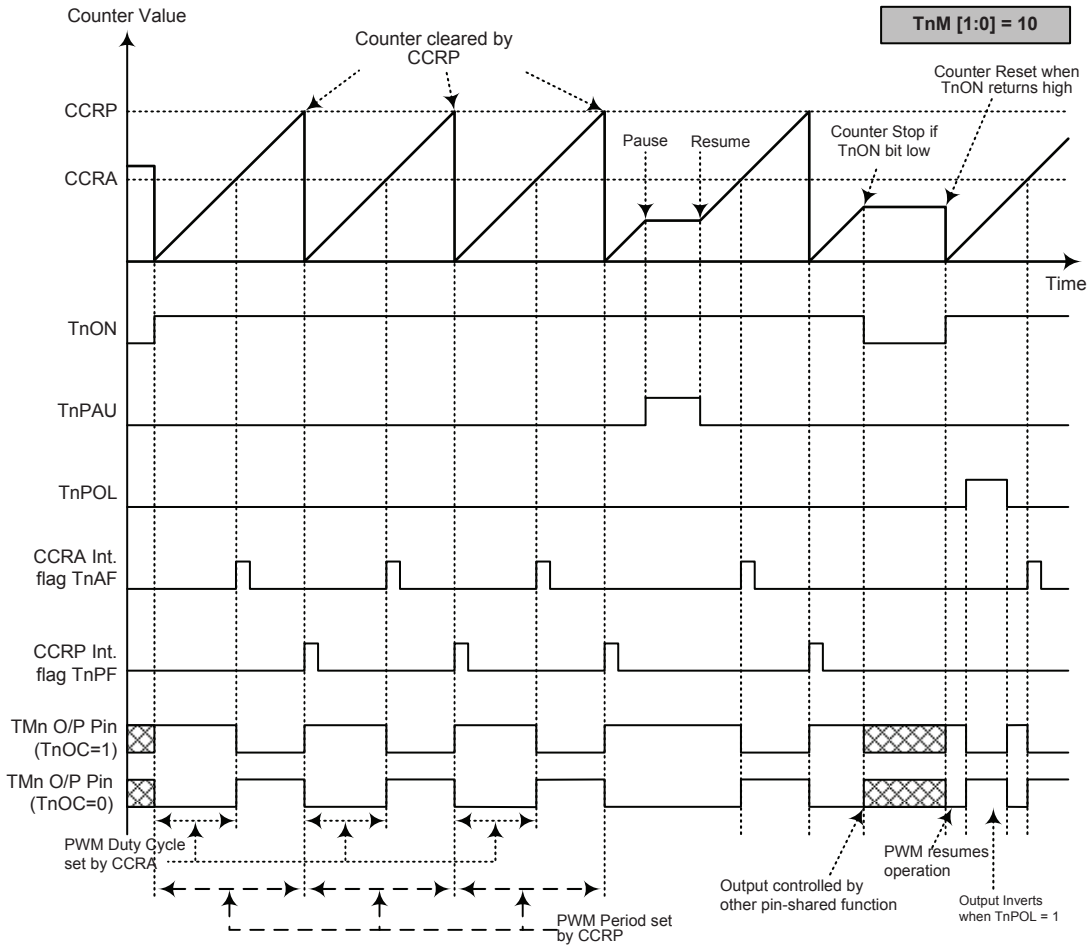
- **10-bit PTM, PWM Mode**

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

If f_{SYS}=16MHz, TM clock source select f_{SYS}/4, CCRP=512 and CCRA=128,

The TMn 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%.



PWM Mode(n=0,1)

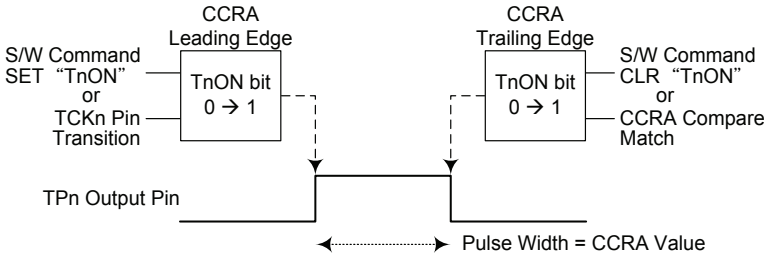
Note: 1. The counter is cleared by CCRP.
2. A counter clear sets the PWM Period
3. The internal PWM function continues running even when TnIO [1:0]=00 or 01
4. The TnCCLR bit has no influence on PWM operation

Single Pulse Output Mode

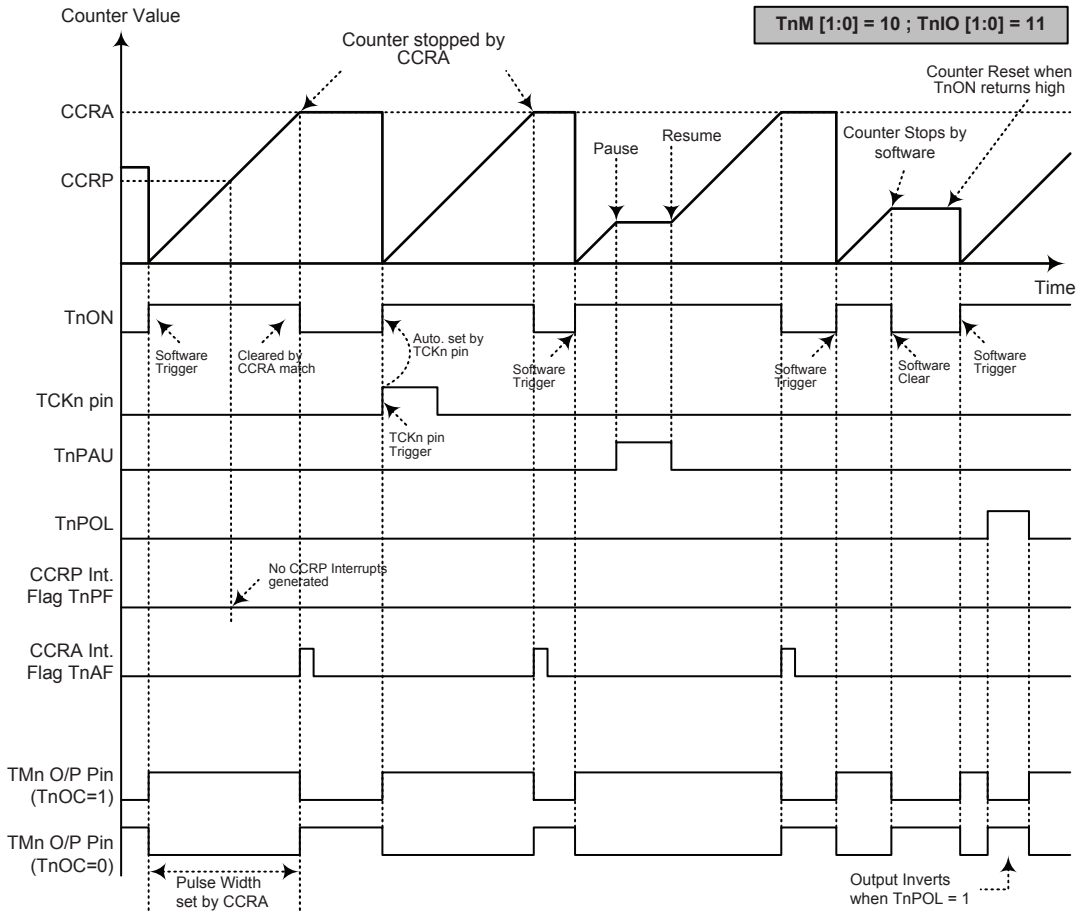
To select this mode, bits TnM1 and TnM0 in the TMnC1 register should be set to 10 respectively and also the TnIO1 and TnIO0 bits should be set to 11 respectively. The Single Pulse Output Mode, as the name suggests, will generate a single shot pulse on the TMn output pin.

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

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



Single Pulse Generation



Single Pulse Mode(n=0,1)

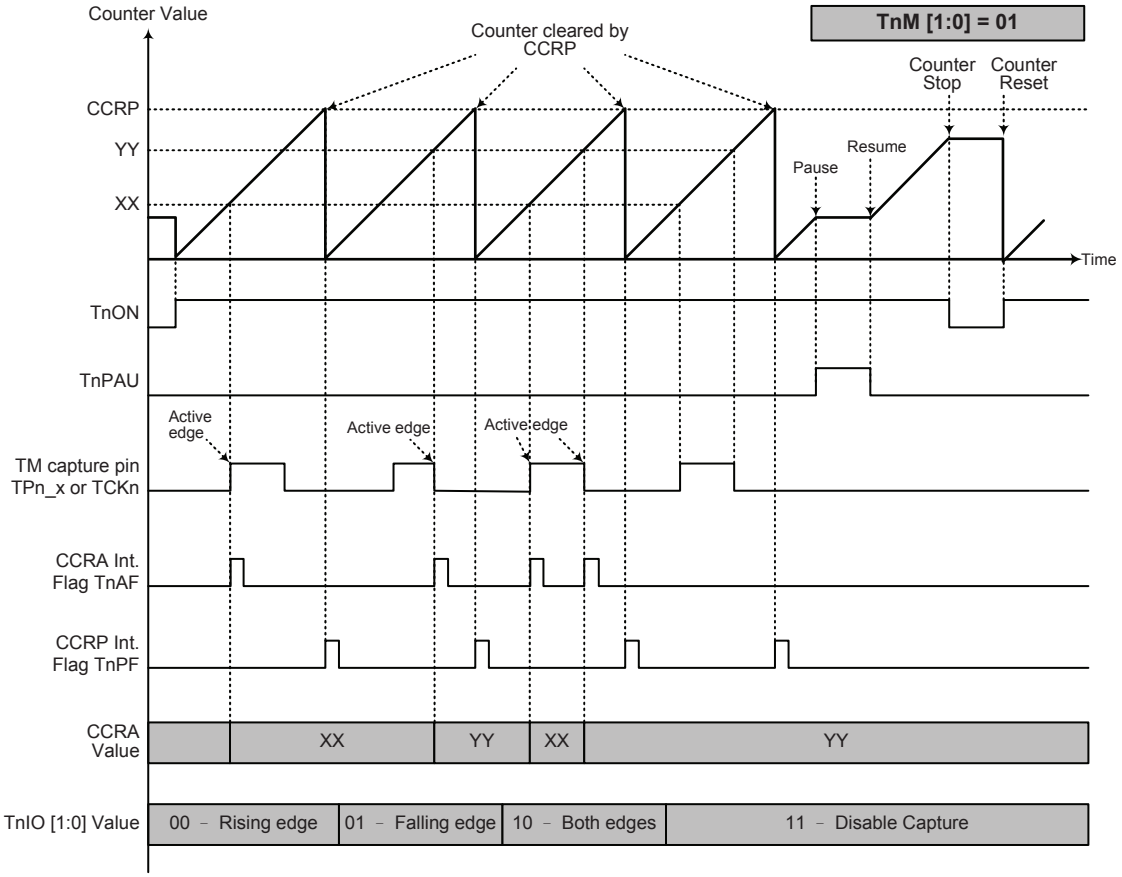
Note: 1. Counter stopped by CCRA
2. CCRP is not used
3. The pulse triggered by the TCKn pin or by setting the TnON bit high
4. A TCKn pin active edge will automatically set the TnON bit high.
5. In the Single Pulse Mode, TnIO [1:0] must be set to "11" and can not be changed.

Capture Input Mode

To select this mode bits TnM1 and TnM0 in the TMnC1 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 TPn or TCKn pin, selected by the TnCAPTS bit in the TMnC1 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 TnIO1 and TnIO0 bits in the TMnC1 register. The counter is started when the TnON bit changes from low to high which is initiated using the application program.

When the required edge transition appears on the TPn or TCKn pin the present value in the counter will be latched into the CCRA registers and a TMn interrupt generated. Irrespective of what events occur on the TPn or TCKn pin the counter will continue to free run until the TnON bit changes from high to low. When a CCRP compare match occurs the counter will reset back to zero; in this way the CCRP value can be used to control the maximum counter value. When a CCRP compare match occurs from Comparator P, a TMn 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 TnIO1 and TnIO0 bits can select the active trigger edge on the TPn or TCKn pin to be a rising edge, falling edge or both edge types. If the TnIO1 and TnIO0 bits are both set high, then no capture operation will take place irrespective of what happens on the TPn or TCKn pin, however it must be noted that the counter will continue to run.

As the TPn or TCKn pin is pin shared with other functions, care must be taken if the TMn is in the Input Capture Mode. This is because if the pin is setup as an output, then any transitions on this pin may cause an input capture operation to be executed. The TnCCLR, TnOC and TnPOL bits are not used in this Mode.



Capture Input Mode (n=0,1)

Note: 1. TnM [1:0]=01 and active edge set by the TnIO [1:0] bits
2. A TMn Capture input pin active edge transfers the counter value to CCRA
3. TnCCLR bit not used
4. No output function – TnOC and TnPOL 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

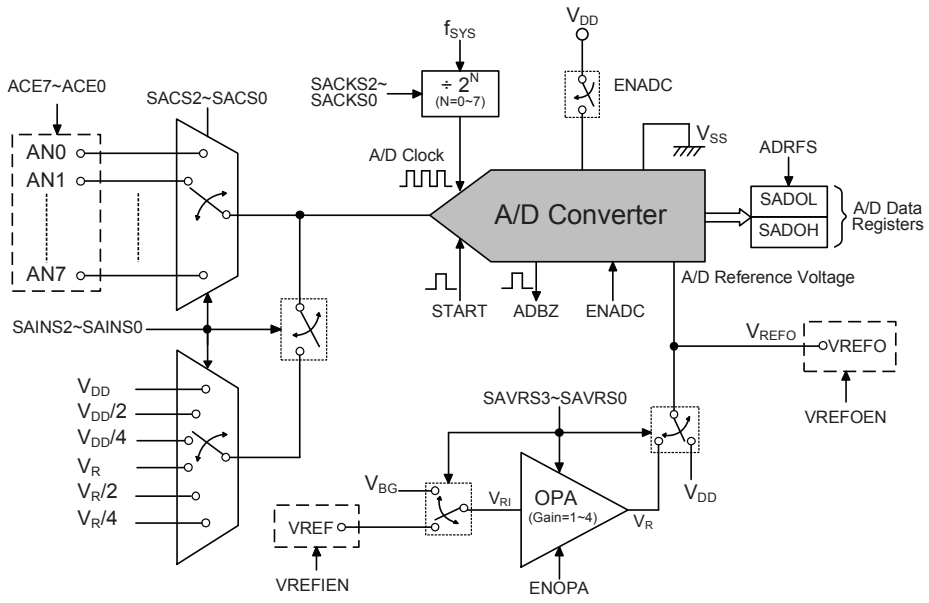
The need to interface to real world analog signals is a common requirement for many electronic systems. However, to properly process these signals by a microcontroller, they must first be converted into digital signals by A/D converters. By integrating the A/D conversion electronic circuitry into the 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 device contains a multi-channel analog to digital converter which can directly interface to external analog signals, such as that from sensors or other control signals and convert these signals directly into a 12-bit digital value. It also can convert the internal signals, such as the Bandgap reference voltage, into a 12-bit digital value. The external or internal analog signal to be converted is determined by the SAINS2~SAINS0 bits together with the SACS2~SACS0 bits. Note that when the external and internal analog signals are simultaneously selected to be converted, the internal analog signal will have the priority. In the meantime the external analog signal will temporarily be switched off until the internal analog signal is deselected. More detailed information about the A/D input signal is described in the "A/D Converter Control Registers" and "A/D Converter Input Signals" sections respectively.

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

External Input Channel	Internal Analog Signals	A/D Signal Select Bits
AN0~AN7	V_{DD}, $V_{DD}/2$, $V_{DD}/4$, V_R, $V_R/2$, $V_R/4$	SAINS2~SAINS0; SACS2~SACS0



A/D Converter Structure

A/D Converter Register Description

Overall operation of the A/D converter is controlled using six registers. A read only register pair exists to store the A/D Converter data 12-bit value. One register, ACERL, is used to configure the external analog input pin function. 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	VREFIEN	VREFOEN	SAVRS3	SAVRS2	SAVRS1	SAVRS0
ACERL	ACE7	ACE6	ACE5	ACE4	ACE3	ACE2	ACE1	ACE0

A/D Converter Registers List

A/D Converter Data Registers – SADOL, SADOH

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

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

A/D Converter Data Registers

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

To control the function and operation of the A/D converter, three 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, the A/D reference voltage as well as controlling the start function and monitoring the A/D converter busy status. As the device contains only one actual analog to digital converter hardware circuit, each of the external and internal analog signals must be routed to the converter. The 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 internal analog signal is selected to be converted, the external channel signal input will automatically be switched off to avoid the signal contention.

The analog input pin function selection bits in the ACERL register determine which pins on I/O Ports are used as external analog channels 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 functions will be removed. In addition, any internal pull-high resistor connected to the pin will be automatically removed if the pin is selected to be an A/D converter input.

- **ACERL Register**

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

Bit 7 **ACE7**: Define PB3 is A/D input or not
0: Not A/D input
1: A/D input, AN7

Bit 6 **ACE6**: Define PA7 is A/D input or not
0: Not A/D input
1: A/D input, AN6

Bit 5 **ACE5**: Define PA6 is A/D input or not
0: Not A/D input
1: A/D input, AN5

Bit 4 **ACE4**: Define PA5 is A/D input or not
0: Not A/D input
1: A/D input, AN4

Bit 3 **ACE3**: Define PA4 is A/D input or not
0: Not A/D input
1: A/D input, AN3

Bit 2 **ACE2**: Define PB2 is A/D input or not
0: Not A/D input
1: A/D input, AN2

Bit 1 **ACE1**: Define PB1 is A/D input or not
0: Not A/D input
1: A/D input, AN1

Bit 0 **ACE0**: Define PB0 is A/D input or not
0: Not A/D input
1: A/D input, AN0

- **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
This bit is used to initiate an A/D conversion process. The bit is normally low but if set high and then cleared low again, the A/D converter will initiate a conversion process.

Bit 6 **ADBZ**: A/D Converter busy flag
0: No A/D conversion is in progress
1: A/D conversion is in progress
This read only flag is used to indicate whether the A/D conversion is in progress or not. When the START bit is set from low to high and then to low again, the ADBZ flag will be set to 1 to indicate that the A/D conversion is initiated. The ADBZ flag will be cleared to 0 after the A/D conversion is complete.

Bit 5 **ENADC**: A/D Converter function enable control
0: Disable
1: Enable
This bit controls the A/D internal function. This bit should be set to one to enable the A/D converter. If the bit is set low, then the A/D converter will be switched off reducing the device power consumption. When the A/D converter function is disabled, the contents of the A/D data register pair, SADOH/SADOL, will keep unchanged.

Bit 4 **ADRFS**: A/D conversion data format select
0: A/D converter data format → SADOH=D [11:4]; SADOL=D [3:0]
1: A/D converter data format → SADOH=D [11:8]; SADOL=D [7:0]
This bit controls the format of the 12-bit converted A/D value in the two A/D data registers. Details are provided in the A/D converter data register section.

Bit 3 Unimplemented, read as "0"

Bit 2~0 **SACS2~SACS0**: A/D converter external analog input channel select
000: AN0
001: AN1
010: AN2
011: AN3
100: AN4
101: AN5
110: AN6
111: 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**: A/D Converter input signal select
000, 100: External signal – External analog channel input
001: Internal signal – Internal A/D converter power supply voltage V_{DD}
010: Internal signal – Internal A/D converter power supply voltage $V_{DD}/2$
011: Internal signal – Internal A/D converter power supply voltage $V_{DD}/4$
101: Internal signal – Internal reference voltage V_R
110: Internal signal – Internal reference voltage $V_R/2$
111: Internal signal – Internal reference voltage $V_R/4$
When the internal analog signal is selected to be converted, the external channel input signal will automatically be switched off regardless of the SACS2~SACS0 bit field value. The internal reference voltage can be derived from various sources selected using the SAVRS3~SAVRS0 bits in the SADC2 register.

Bit 4~3 Unimplemented, read as "0"

Bit 2~0 **SACKS2~SACKS0**: A/D conversion clock source select
000: f_{SYS}
001: $f_{SYS}/2$
010: $f_{SYS}/4$
011: $f_{SYS}/8$
100: $f_{SYS}/16$
101: $f_{SYS}/32$
110: $f_{SYS}/64$
111: $f_{SYS}/128$
These bits are used to select the clock source for the A/D converter.

- **SADC2 Register**

Bit	7	6	5	4	3	2	1	0
Name	ENOPA	VBGEN	VREFIEN	VREFOEN	SAVRS3	SAVRS2	SAVRS1	SAVRS0
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 **ENOPA**: A/D converter OPA function enable control
0: Disable
1: Enable
This bit controls the internal OPA function to provide various reference voltage for the A/D converter. When the bit is set high, the internal reference voltage, V_R, can be used as the internal converted signal or reference voltage by the A/D converter. If the internal reference voltage is not used by the A/D converter, then the OPA function should be properly configured to conserve power.

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

Bit 5 **VREFIEN**: VREF pin selection bit
0: Disable – VREF pin is not selected
1: Enable – VREF pin is selected

Bit 4 **VREFOEN**: VREFO pin selection bit
0: Disable – VREFO pin is not selected
1: Enable – VREFO pin is selected

Bit 3~0 **SAVRS3~SAVRS0**: A/D Converter reference voltage select
0000: V_{DD}
0001: V_{REF}
0010: $V_{REF} \times 2$
0011: $V_{REF} \times 3$
0100: $V_{REF} \times 4$
1001: Reserved, can not be used
1010: $V_{BG} \times 2$
1011: $V_{BG} \times 3$
1100: $V_{BG} \times 4$
Others: V_{DD}
When the A/D converter reference voltage source is selected to derive from the internal V_{BG} voltage, the reference voltage which comes from the external VDD or VREF pin will be automatically switched off.

A/D Converter Input Signals

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

After the desired external pins are selected as the A/D analog channel inputs using the corresponding ACEn bits in the ACERL register, the analog signal to be converted can come from these external channel inputs. If the SAINS2~SAINS0 bits are set to "000" or "100", the external analog channel input will be selected to be converted and the SACS2~SACS0 bits can deternine which external channel is selected to be converted.

If the SAINS2~SAINS0 bits are set to any other values except "000" and "100", one of the internal analog signals can be selected to be converted. The internal analog signals can be derived from the A/D converter supply power, V_{DD}, or internal reference voltage, V_R, with a specific ratio of 1, 1/2 or 1/4. Note that if the internal analog signal is selected to be converted, the external channel signal input will automatically be switched off to avoid the signal contention.

SAINS [2:0]	SACS [2:0]	Input Signals	Description
000, 100	000~111	AN0~AN7	External channel analog input
001	xxx	V_{DD}	A/D converter power supply voltage
010	xxx	$V_{DD}/2$	A/D converter power supply voltage/2
011	xxx	$V_{DD}/4$	A/D converter power supply voltage/4
101	xxx	V_R	Internal reference voltage
110	xxx	$V_R/2$	Internal reference voltage/2
111	xxx	$V_R/4$	Internal reference voltage/4

A/D Converter Input Signal Selection

A/D Converter Reference Voltage

The actual reference voltage, V_{REF}, supplied to the A/D Converter can be derived from the positive power supply pin, V_{DD}, an external reference source supplied on pin VREF or an internal reference source derived from the Bandgap circuit, a choice which is made through the SAVRS3~SAVRS0 bits in the SADC2 register. Then the actual selected reference voltage source can be amplified through a programmable gain amplifier except the voltage sourced from V_{DD}. The OPA gain can be equal to 1, 2, 3 or 4. Note that the desired reference voltage selected using the SAVRS field will be output on the VREFO pin. As the VREF and VREFO pins both are pin-shared with other functions, when the VREF or VREFO pin is selected as the reference voltage pin, the VREF or VREFO selection bit, VREFIEN or VREFOEN, should be properly configured to disable other pin-shared functions before the reference voltage pin function is used. Note that the analog input values must not be allowed to exceed the value of the reference voltage for the A/D converter.

A/D Operation

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

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

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

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

A/D Clock Period Examples

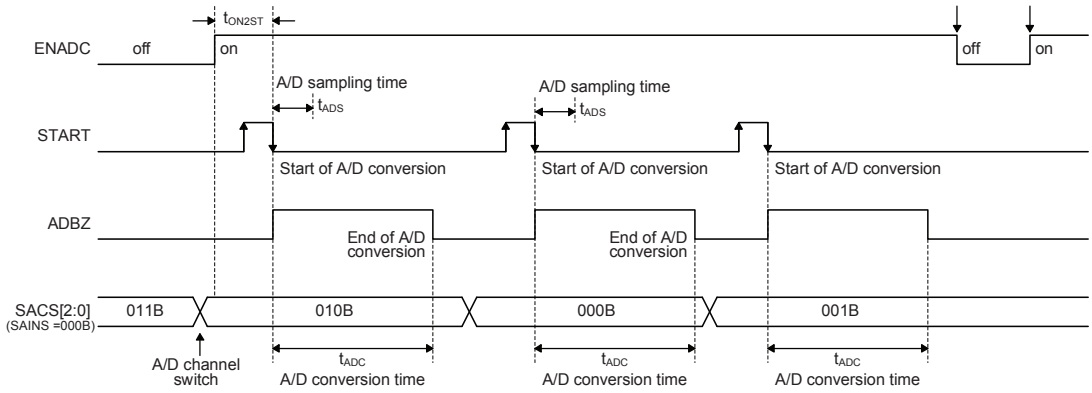
Controlling the power on/off function of the A/D converter circuitry is implemented using the ENADC bit in the SADC0 register. This bit must be set high to power on the A/D converter. When the ENADC bit is set high to power on the A/D converter internal circuitry, a certain delay, as indicated in the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

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 (1)

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



A/D Conversion Example Timing Diagram

Summary of A/D Conversion Steps

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

- Step 1
  Select the required A/D conversion clock by properly programming the SACKS2~SACKS0 bits in the SADC1 register.
- Step 2
  Enable the A/D converter by setting the ENADC bit in the SADC0 register to one.
- Step 3
  Select which signal is to be connected to the internal A/D converter by correctly configuring the SAINS and SACS bit fields
  Select the external channel input to be converted, go to Step 4.
  Select the internal analog signal to be converted, go to Step 5.
- Step 4
  If the A/D input signal comes from the external channel input selecting by configuring the SAINS bit field, the corresponding pins should first be configured as A/D input function by configuring the relevant pin-shared function control bits. The desired analog channel then should be selected by configuring the SACS bit field. After this step, go to Step 6.
- Step 5
  If the A/D input signal is selected to come from the internal analog signal, the SAINS bit field should be properly configured and then the external channel input will automatically be disconnected regardless of the SACS bit field value. After this step, go to Step 6.
- Step 6
  Select the reference voltgage source by configuring the SAVRS3~SAVRS0 bits.
- Step 7
  Select the A/D converter output data format by configuring the ADRFS bit.
- Step 8
  If A/D conversion interrupt is used, the interrupt control registers must be correctly configured to ensure the A/D interrupt function is active. The master interrupt bontrol bit, EMI, and the A/D conversion interrupt control bit, ADE, must both be set high in advance.
- Step 9
  The A/D conversion procedure can now be initialized by setting the START bit from low to high and then low again.
- Step 10
  If A/D conversion is in progress, the ADBZ flag will be set high. After the A/D conversion process is complete, the ADBZ flag will go low and then the output data can be read from SADOH and SADOL registers.

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

Programming Considerations

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

A/D Transfer Function

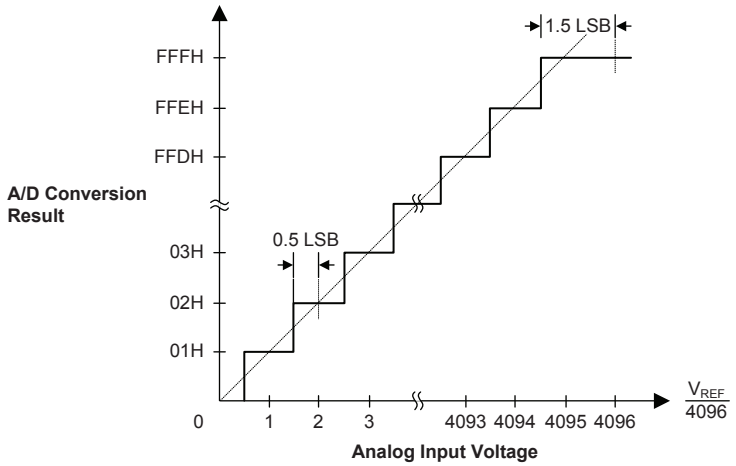
As the device contains a 12-bit A/D converter, its full-scale converted digitised value is equal to FFFH. Since the full-scale analog input value is equal to the actual A/D converter reference voltage, V_{REF}, this gives a single bit analog input value of V_{REF} divided by 4096.

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

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

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

The diagram shows the ideal transfer function between the analog input value and the digitised output value for the A/D converter. Except for the digitised zero value, the subsequent digitised values will change at a point 0.5 LSB below where they would change without the offset, and the last full scale digitised value will change at a point 1.5 LSB below the V_{REF} level. Note that here the V_{REF} voltage is the actual A/D converter reference voltage determined by the SAVRS field.



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 ADBZ polling method to detect the end of conversion

```
clr ADE               ; disable ADC interrupt
mov a,03H
mov SADC1,a           ; select fSYS/8 as A/D clock
set ENADC             ; enable A/D converter
mov a,03H             ; setup ACERL to configure pin AN1 and AN0
mov ACERL,a
mov a,20H
mov SADC0,a           ; enable and connect AN0 channel to A/D converter
mov a,00H             ; select VDD as reference voltage and
mov SADC2,a           ; switch off OPA and Bandgap reference voltage
:
start_conversion:
clr START             ; high pulse on start bit to initiate conversion
set START             ; reset A/D
clr START             ; start A/D
:
polling_EOC:
sz  ADBZ              ; poll the SADC0 register ADBZ bit to detect end of A/D conversion
jmp polling_EOC       ; continue polling
:
mov a,SADOL           ; read low byte conversion result value
mov SADOL_buffer,a    ; save result to user defined register
mov a,SADOH           ; read high byte conversion result value
mov SADOH_buffer,a    ; save result to user defined register
:
jmp start_conversion  ; start next A/D conversion
```

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

```
clr ADE               ; disable ADC interrupt
mov a,03H
mov SADC1,a           ; select fSYS/8 as A/D clock
set ENADC             ; enable A/D converter
mov a,03h             ; setup ACERL to configure pin AN1 and AN0
mov ACERL,a
mov a,20h
mov SADC0,a           ; enable and connect AN0 channel to A/D converter
mov a,00H             ; select VDD as reference voltage and
mov SADC2,a           ; switch off OPA and Bandgap reference voltage
:
Start_conversion:
clr START             ; high pulse on START bit to initiate conversion
set START             ; reset A/D
clr START             ; start A/D
clr ADF               ; clear ADC interrupt request flag
set ADE               ; enable ADC interrupt
set EMI               ; enable global interrupt
:
:
ADC_ISR:              ; ADC interrupt service routine
mov acc_stack,a       ; save ACC to user defined memory
mov a,STATUS
mov status_stack,a    ; save STATUS to user defined memory
:
mov a, SADOL          ; read low byte conversion result value
mov SADOL_buffer,a    ; save result to user defined register
mov a, SADOH          ; read high byte conversion result value
mov SADOH_buffer,a    ; save result to user defined register
:
EXIT_INT_ISR:
mov a,status_stack
mov STATUS,a          ; restore STATUS from user defined memory
mov a,acc_stack       ; restore ACC from user defined memory
reti
```

Serial Interface Module – SIM

The device contains a Serial Interface Module, which includes both the four-line SPI interface or two-line I^2C interface types, to allow an easy method of communication with external peripheral hardware. Having relatively simple communication protocols, these serial interface types allow the microcontroller to interface to external SPI or I^2C based hardware such as sensors, Flash or EEPROM memory, etc. 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-remapping function selection bits. As both interface types share the same pins and registers, the choice of whether the SPI or I^2C type is used is made using the 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 device can be either master or slave. Although the SPI interface specification can control multiple slave devices from a single master, the device provides 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.

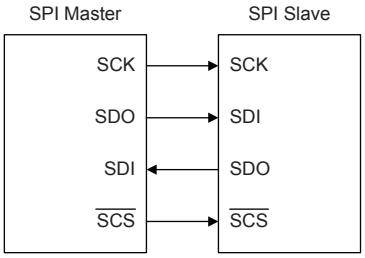
SPI Interface Operation

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 I^2C function pins, the SPI interface pins must first be selected by configuring the pin-remapping 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, set CSEN bit to 0 the $\overline{SCS}$ pin will be floating state.

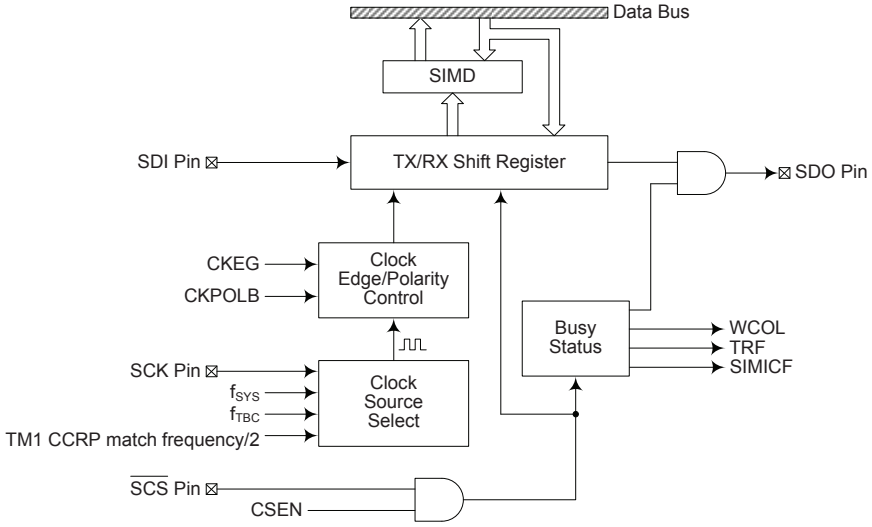
The SPI function in this device offers the following features:

- 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 Master/Slave Connection



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	—	SIMDBNC1	SIMDBNC0	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	—	SIMDBNC1	SIMDBNC0	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 TM1 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 TM1 or f_{TBC}. 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 **SIMDBNC1~SIMDBNC0**: I²C 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 I²C function and the SIM operating current will be reduced to a minimum value. When the bit is high the SIM interface is enabled. The SIM configuration option must have first enabled the SIM interface for this bit to be effective.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 Incomplete Flag
0: SIM incomplete condition not occurred
1: SIM 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

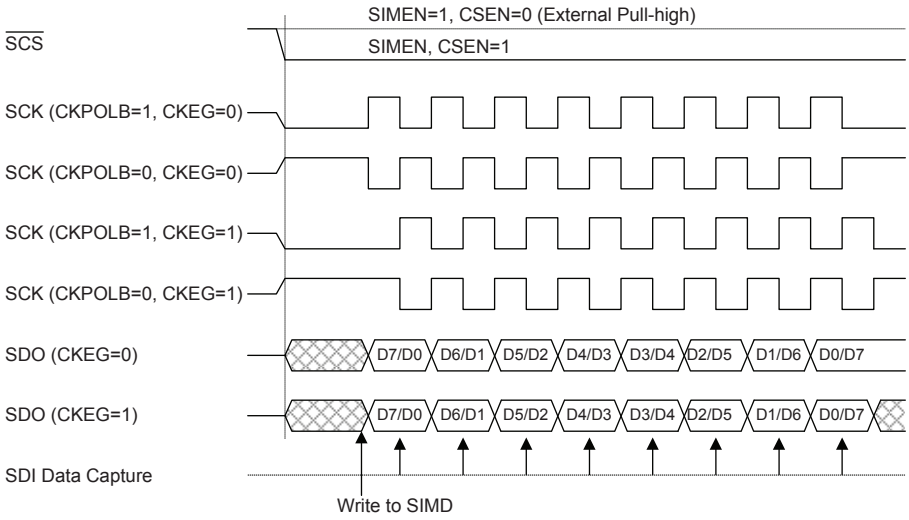
1: SPI data transfer is completed

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.

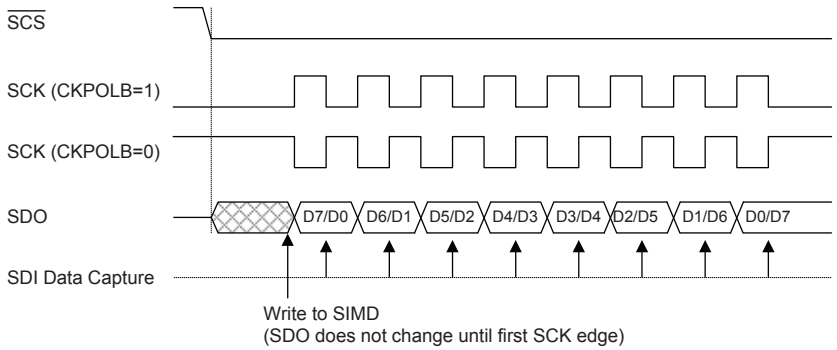
SPI Communication

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 devices 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.

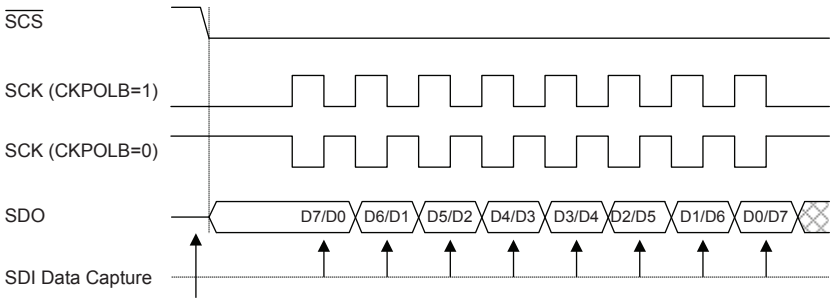
The SPI master mode will continue to function even in the IDLE Mode if the selected SPI clock source is running.



SPI Master Mode Timing

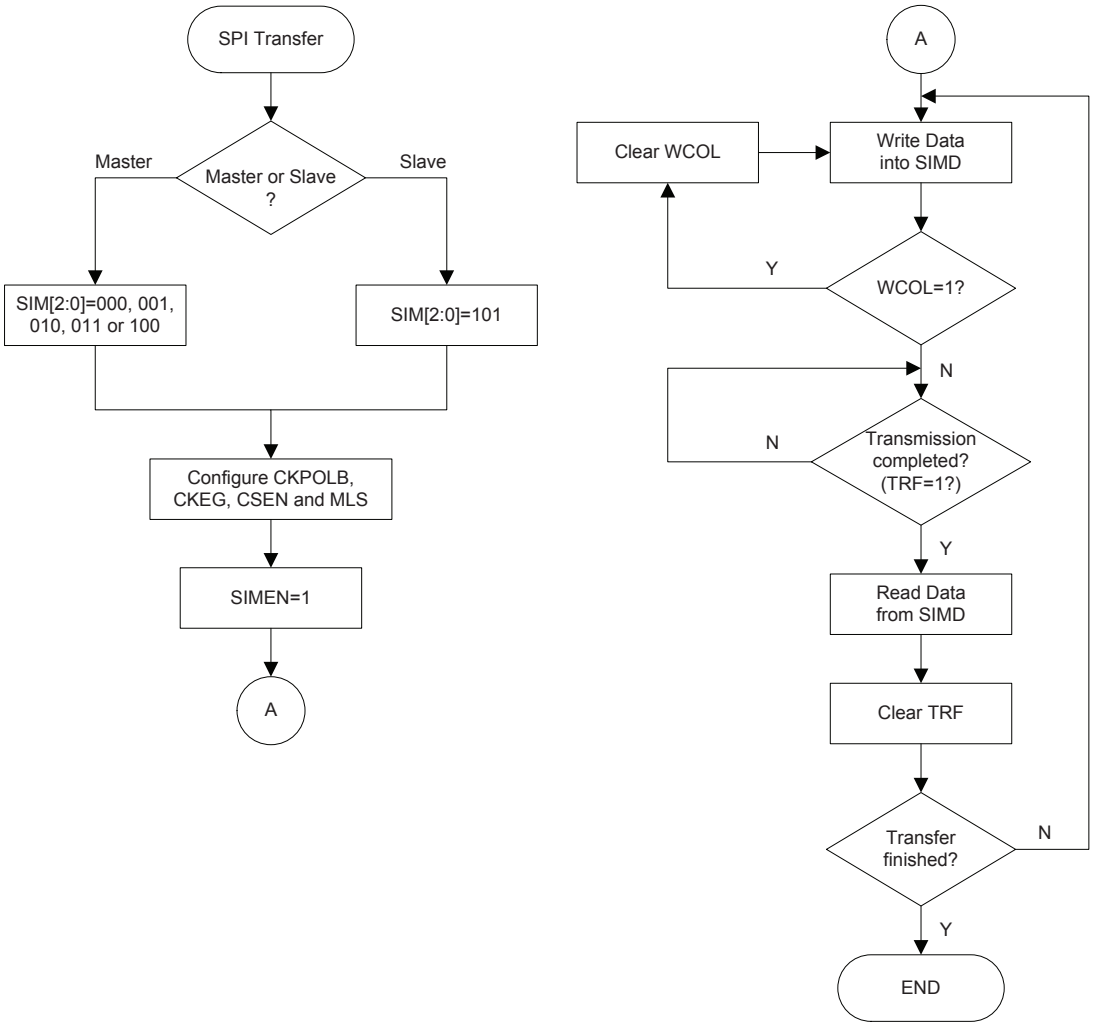


SPI Slave Mode Timing – CKEG=0



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

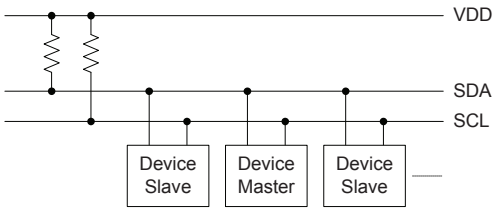
SPI Slave Mode Timing – CKEG=1



SPI Transfer Control Flow Chart

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.

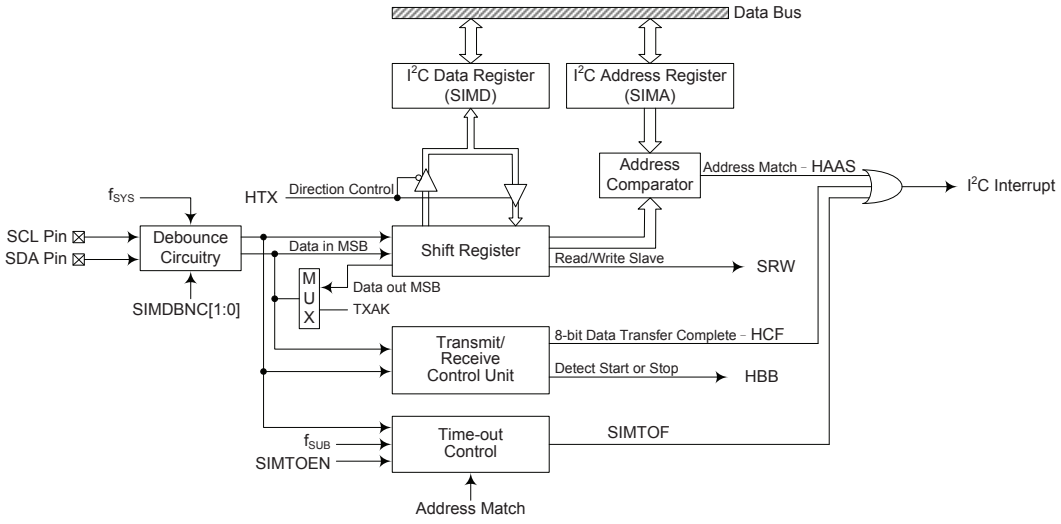


I²C Master Slave Bus Connection

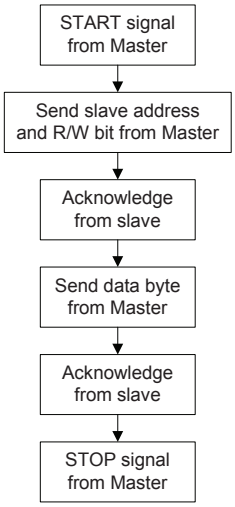
I²C interface Operation

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 the device, which only operates in slave mode, there are two methods of transferring data on the I²C bus, the slave transmit mode and the slave receive mode.



I²C Block Diagram



The SIMDBNC1 and SIMDBNC0 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 SIMTOC, one slave address register, 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. Before the microcontroller 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 microcontroller can read it from the SIMD register. Any transmission or reception of data from the I²C bus must be made via the SIMD register.

Note that the SIMA register also has the name SIMC2 which is used by the SPI function. Bit SIMEN and bits SIM2~SIM0 in register SIMC0 are used by the I²C interface.

Register Name	Bit							
	7	6	5	4	3	2	1	0
SIMC0	SIM2	SIM1	SIM0	—	SIMDBNC1	SIMDBNC0	SIMEN	SIMICF
SIMC1	HCF	HAAS	HBB	HTX	TXAK	SRW	IAMWU	RXAK
SIMA	A6	A5	A4	A3	A2	A1	A0	D0
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	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 7~1 **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 three control registers for the I²C interface, SIMC0, SIMC1 and SIMTOC. 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. The SIMTOC register is used to control the I²C bus time-out function which is described in the I²C Time-out Control section.

- **SIMC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	SIM2	SIM1	SIM0	—	SIMDBNC1	SIMDBNC0	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 TM1 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 TM1 or f_{TBC}. 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 **SIMDBNC1~ SIMDBNC0**: 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. The SIM configuration option must have first enabled the SIM interface for this bit to be effective.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 Incomplete Flag
0: SIM incomplete condition not occurred
1: SIM 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.

- **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.

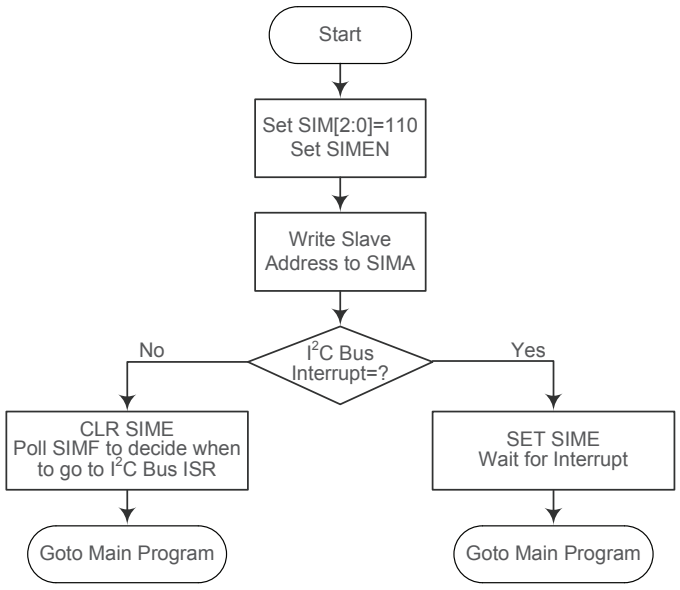
Bit 1 **IAMWU**: I²C Address Match Wake-Up control
0: Disable
1: Enable – must be cleared by the application program after wake-up
This bit should be set to 1 to enable the I²C address match wake up from the SLEEP or IDLE Mode. If the IAMWU bit has been set before entering either the SLEEP or IDLE mode 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
  Set the SIM2~SIM0 bits to "110" and SIMEN bit to "1" in the SIMC0 register to enable the I²C bus.
- Step 2
  Write the slave address of the device to the I²C bus address register SIMA.
- Step 3
  Set the SIME interrupt enable bit of the interrupt control register to enable the SIM interrupt.



I²C Bus Initialisation Flow Chart

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 device 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.

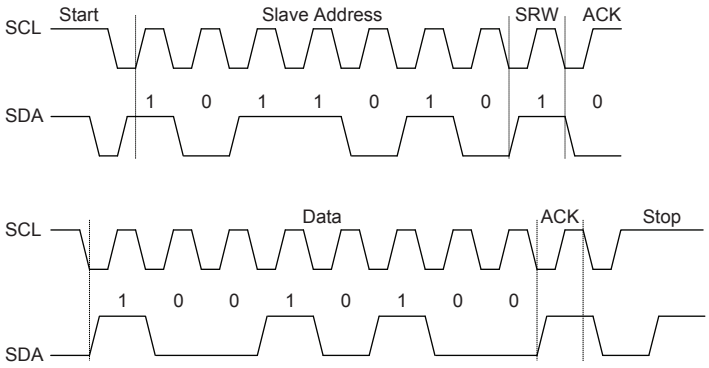
I²C Bus Slave Address Acknowledge Signal

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.

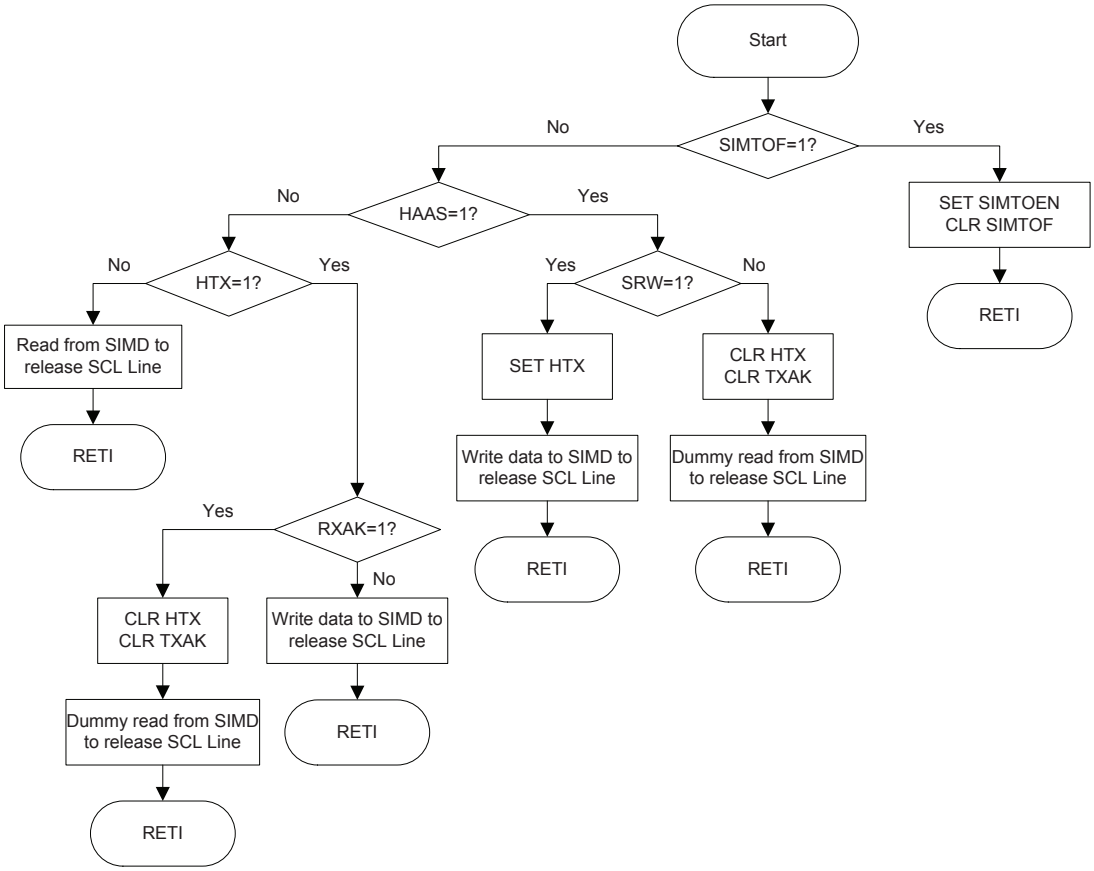


S=Start (1 bit)
SA=Slave Address (7 bits)
SR=SRW bit (1 bit)
M=Slave device send acknowledge bit (1 bit)
D=Data (8 bits)
A=ACK (RXAK bit for transmitter, TXAK bit for receiver, 1 bit)
P=Stop (1 bit)

S	SA	SR	M	D	A	D	A	·····	S	SA	SR	M	D	A	D	A	·····	P

I²C Communication Timing Diagram

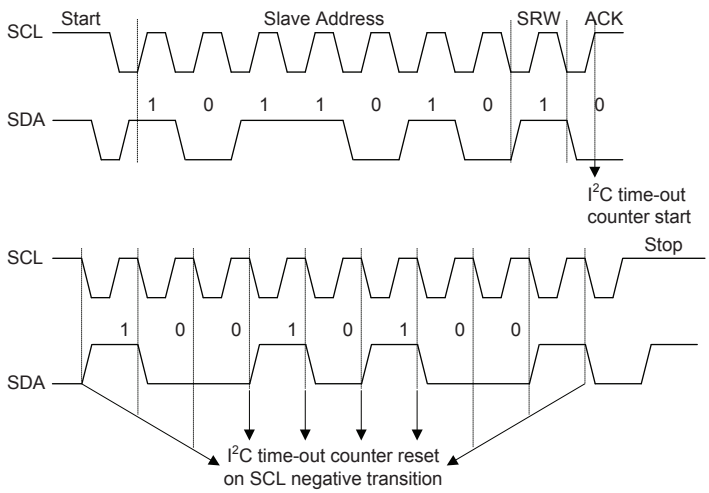
Note: When a slave address is matched, the device must be placed in either the transmit mode and then write data to the SIMD register, or in the receive mode where it must implement a dummy read from the SIMD register to release the SCL line.



I²C Bus 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 function 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 time is equal to (SIMTOS[5:0]+1) × (32/f_{SUB}).

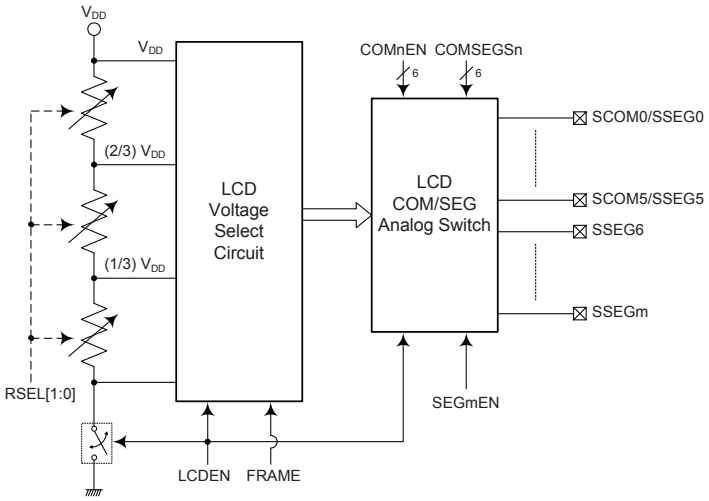
SCOM/SSEG Function for LCD

The device has the capability of driving external LCD panels. The common and segment pins for LCD driving, SCOM0~SCOM5 and SSEG0~SSEG19, are pin-shared with certain pins on the I/O ports. The LCD signals, COM and SEG, are generated using the application program.

LCD Operation

An external LCD panel can be driven using the device by configuring the I/O pins as common pins and and segment pins. The LCD driver function is controlled using the LCD control registers which in addition to controlling the overall on/off function also controls the R-type bias current on the SCOM and SSEG pins. This enables the LCD COM and SEG driver to generate the necessary V_{SS}, $(1/3)V_{DD}$, $(2/3)V_{DD}$ and V_{DD} voltage levels for LCD 1/3 bias operation.

The LCDEN bit in the SLCDC0 register is the overall master control for the LCD driver. This bit is used in conjunction with the COMnEN and SEGmEN bits to select which I/O pins are used for LCD driving. Note that the corresponding Port Control register does not need to first setup the pins as outputs to enable the LCD driver operation.



Software Controlled LCD Driver Structure

LCD Frames

A cyclic LCD waveform includes two frames known as Frame 0 and Frame 1 for which the following offers a functional explanation.

- **Frame 0**

To select Frame 0, clear the FRAME bit in the SLCDC 0 register to 0.

In frame 0, the COM signal output can have a value of V_{DD} or a V_{BIAS} value of $(1/3)\times V_{DD}$. The SEG signal output can have a value of V_{SS} or a V_{BIAS} value of $(2/3)\times V_{DD}$.

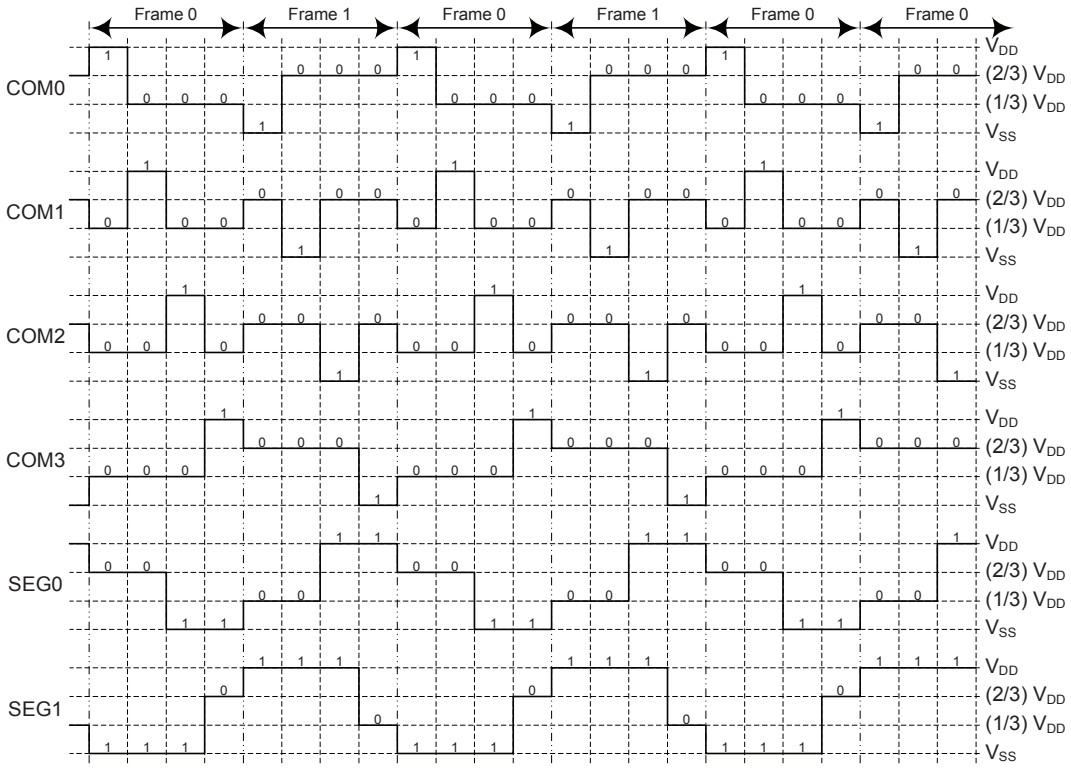
- **Frame 1**

To select Frame 1, set the FRAME bit in the SLCDC0 register to 1.

In frame 1, the COM signal output can have a value of V_{SS} or a V_{BIAS} value of $(2/3)\times V_{DD}$. The SEG signal output can have a value of V_{DD} or a V_{BIAS} value of $(1/3)\times V_{DD}$.

The COMn waveform is controlled by the application program using the FRAME bit in the SLCDC0 register and the corresponding pin-shared I/O data bit for the respective COM pin to determine whether the COMn output has a value of V_{DD}, V_{SS} or V_{BIAS}. The SEGm waveform is controlled in a similar way using the FRAME bit and the corresponding pin-shared I/O data bit for the respective SEG pin to determine whether the SEGm output has a value of V_{DD}, V_{SS} or V_{BIAS}.

The accompanying waveform diagram shows a typical 1/3 bias LCD waveform gemerated using the application program together with the LCD voltage select circuit. Note that the depiction of a "1" in the diagram illustrates an illuminated LCD pixel. The COM signal polarity generated on pins SCOM0~SCOM5, whether "0" or "1", are generated using the corresponding pin-shared I/O data register bit.



Note: The logical values shown in the above diagram are the corresponding pin-shared I/O data bit value.

1/3 Bias LCD Waveform – 4-COM & 2-SEG application

LCD Control Registers

The LCD COM and SEG driver enables a range of selections to be provided to suit the requirement of the LCD panel which is being used. The bias resistor choice is implemented using the RSEL1 and RSEL0 bits in the SLCDC0 register. All COM and SEG pins are pin-shared with I/O pin.

Register Name	Bit							
	7	6	5	4	3	2	1	0
SLCDC0	FRAME	RSEL1	RSEL0	LCDEN	COM3EN	COM2EN	COM1EN	COM0EN
SLCDC1	COM5EN	COM4EN	COMSEGS5	COMSEGS4	COMSEGS3	COMSEGS2	COMSEGS1	COMSEGS0
SLCDC2	SEG13EN	SEG12EN	SEG11EN	SEG10EN	SEG9EN	SEG8EN	SEG7EN	SEG6EN
SLCDC3	—	—	SEG19EN	SEG18EN	SEG17EN	SEG16EN	SEG15EN	SEG14EN

LCD Driver Control Registers List

SLCDC0 Register

Bit	7	6	5	4	3	2	1	0
Name	FRAME	RSEL1	RSEL0	LCDEN	COM3EN	COM2EN	COM1EN	COM0EN
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7 **FRAME**: SCOM/SSEG Output Frame selection
0: Frame 0
1: Frame 1

Bit 6~5 **RSEL1~RSEL0**: Select SCOM/SSEG typical bias current (V_{DD}=5V)
00: 8.3μA
01: 16.7μA
10: 50μA
11: 100μA

Bit 4 **LCDEN**: SCOM/SSEG Module enable control
0: Disable
1: Enable
The SCOMn and SSEGm lines can be enabled using COMnEN and SEGmEN if the LCDEN bit is set to 1. When the LCD bit is cleared to 0, then the SCOMn and SSEGm outputs will be fixed at a V_{SS} level.

Bit 3 **COM3EN**: SCOM3/SSEG3 or other pin function select
0: Other pin-shared functions
1: SCOM3/SSEG3 function

Bit 2 **COM2EN**: SCOM2/SSEG2 or other pin function select
0: Other pin-shared functions
1: SCOM2/SSEG2 function

Bit 1 **COM1EN**: SCOM1/SSEG1 or other pin function select
0: Other pin-shared functions
1: SCOM1/SSEG1 function

Bit 0 **COM0EN**: SCOM0/SSEG0 or other pin function select
0: Other pin-shared functions
1: SCOM0/SSEG0 function

SLCDC1 Register

Bit	7	6	5	4	3	2	1	0
Name	COM5EN	COM4EN	COMSEGS5	COMSEGS4	COMSEGS3	COMSEGS2	COMSEGS1	COMSEGS0
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 **COM5EN**: SCOM5/SSEG5 or other pin function select
0: Other pin-shared functions
1: SCOM5/SSEG5 function

Bit 6 **COM4EN**: SCOM4/SSEG4 or other pin function select
0: Other pin-shared functions
1: SCOM4/SSEG4 function

Bit 5 **COMSEGS5**: SCOM5 or SSEG5 pin function select
0: SCOM5
1: SSEG5

Bit 4 **COMSEGS4**: SCOM4 or SSEG4 pin function select
0: SCOM4
1: SSEG4

Bit 3 **COMSEGS3**: SCOM3 or SSEG3 pin function select
0: SCOM3
1: SSEG3

Bit 2 **COMSEGS2**: SCOM2 or SSEG2 pin function select
0: SCOM2
1: SSEG2

Bit 1 **COMSEGS1**: SCOM1 or SSEG1 pin function select
0: SCOM1
1: SSEG1

Bit 0 **COMSEGS0**: SCOM0 or SSEG0 pin function select
0: SCOM0
1: SSEG0

SLCDC2 Register

Bit	7	6	5	4	3	2	1	0
Name	SEG13EN	SEG12EN	SEG11EN	SEG10EN	SEG9EN	SEG8EN	SEG7EN	SEG6EN
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 **SEG13EN**: SSEG13 pin function select
0: Other pin-shared functions
1: SSEG13 function

Bit 6 **SEG12EN**: SSEG12 pin function select
0: Other pin-shared functions
1: SSEG12 function

Bit 5 **SEG11EN**: SSEG11 pin function select
0: Other pin-shared functions
1: SSEG11 function

Bit 4 **SEG10EN**: SSEG10 pin function select
0: Other pin-shared functions
1: SSEG10 function

Bit 3 **SEG9EN**: SSEG9 pin function select
0: Other pin-shared functions
1: SSEG9 function

Bit 2 **SEG8EN**: SSEG8 pin function select
0: Other pin-shared functions
1: SSEG8 function

Bit 1 **SEG7EN**: SSEG7 pin function select
0: Other pin-shared functions
1: SSEG7 function

Bit 0 **SEG6EN**: SSEG6 pin function select
0: Other pin-shared functions
1: SSEG6 function

SLCDC3 Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	SEG19EN	SEG18EN	SEG17EN	SEG16EN	SEG15EN	SEG14EN
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 **SEG19EN**: SSEG19 pin function select
0: Other pin-shared functions
1: SSEG19 function

Bit 4 **SEG18EN**: SSEG18 pin function select
0: Other pin-shared functions
1: SSEG18 function

Bit 3 **SEG17EN**: SSEG17 pin function select
0: Other pin-shared functions
1: SSEG17 function

Bit 2 **SEG16EN**: SSEG16 pin function select
0: Other pin-shared functions
1: SSEG16 function

Bit 1 **SEG15EN**: SSEG15 pin function select
0: Other pin-shared functions
1: SSEG15 function

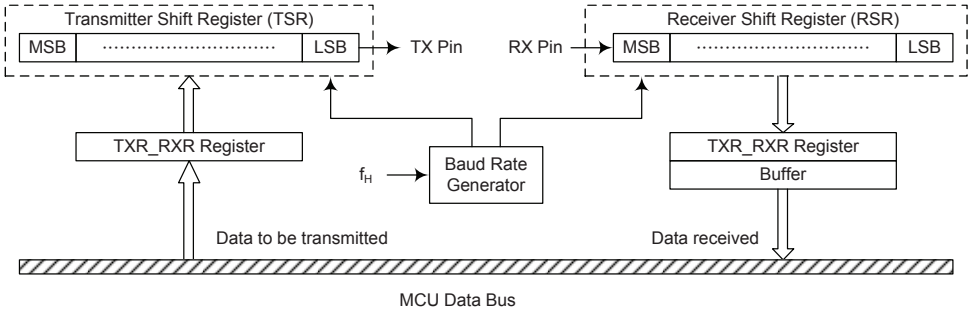
Bit 0 **SEG14EN**: SSEG14 pin function select
0: Other pin-shared functions
1: SSEG14 function

UART Interface

The UART interface module is contained in the device. The 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, asynchronous 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)
- Separately enabled transmitter and receiver
- 2-byte Deep FIFO Receive Data Buffer
- RX pin wake-up function
- Transmit and receive interrupts
- Interrupts can be initialized by the following conditions:
  - Transmitter Empty
  - Transmitter Idle
  - Receiver Full
  - Receiver Overrun
  - Address Mode Detect



UART Data Transfer Block Diagram

UART External Pin

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 respectively the UART transmitter and receiver pins which are pin-shared with I/O or other pin-shared functions. Along with the UARTEN bit, the TXEN and RXEN bits, if set, will automatically setup these I/O 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 used as I/O or other pin-shared functional pin depending upon the pin-shared function priority.

UART Data Transfer Scheme

The above 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_RXR register by the application program. The data will then be transferred to the Transmit Shift Register from where it will be shifted out, LSB first, onto the TX pin at a rate controlled by the Baud Rate Generator. Only the TXR_RXR register is mapped onto the MCU Data Memory, the Transmit Shift Register is not mapped and is therefore inaccessible to the application program.

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

It should be noted that the actual register for data transmission and reception, 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 registers.

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
BRG	BRG7	BRG6	BRG5	BRG4	BRG3	BRG2	BRG1	BRG0
TXR_RXR	TXRX7	TXRX6	TXRX5	TXRX4	TXRX3	TXRX2	TXRX1	TXRX0

UART Status and Control Registers List

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	TXRX7	TXRX6	TXRX5	TXRX4	TXRX3	TXRX2	TXRX1	TXRX0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	x	x	x	x	x	x	x	x

"x": unknown

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

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 and further explanations are 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 TXR_RXR data register.

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

The NF flag is the noise flag. When this read only flag is "0", it indicates no noise condition. When the flag is "1", it indicates that the UART has detected noise on the receiver input. The NF flag is set during the same cycle as the RXIF flag but will not be set in the case of 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 TXR_RXR data register.

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

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

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

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

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

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

Bit 2 **RXIF**: Receive TXR_RXR data register status
0: TXR_RXR data register is empty
1: TXR_RXR data register has available data

The RXIF flag is the receive data register status flag. When this read only flag is "0", it indicates that the TXR_RXR read data register is empty. When the flag is "1", it indicates that the TXR_RXR read data register contains new data. When the contents of the shift register are transferred to the TXR_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 TXR_RXR register, and if the TXR_RXR register has no data available.

Bit 1 **TIDLE**: Transmission status
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_RXR register. The flag is not generated when a data character or a break is queued and ready to be sent.

Bit 0 **TXIF**: Transmit TXR_RXR data register status
0: character is not transferred to the transmit shift register
1: character has transferred to the transmit shift register (TXR_RXR data register is empty)

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

UCR1 Register

The UCR1 register together with the UCR2 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 used as other pin-shared functional pins.
1: Enable UART; TX and RX pins can function as UART pins defined by TXEN and RXEN bits

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 other pin-shared functional pins. 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.

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

This bit is the parity function 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. When this bit is equal to "1", two stop bits format are used. If the bit is equal to "0", then only one stop bit format 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 equal to "0", there are no break characters and the TX pin operats normally. When the bit is equal to "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 transfes 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 transfes are in 8-bit or 9-bit format.

UCR2 Register

The UCR2 register is the second of the UART control registers and serves several purposes. One of its main functions is to control the basic enable/disable operation if 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 function enable and the address detect function 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 enable control
0: UART Transmitter is disabled
1: UART Transmitter is enabled
The TXEN bit 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 other pin-shared functional pin. 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 other pin-shared functional pin.

Bit 6 **RXEN**: UART Receiver enable control
0: UART Receiver is disabled
1: UART Receiver is enabled
The RXEN bit 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 receiver buffers will be reset. In this situation the RX pin will be other pin-shared functional pin. 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 other pin-shared functional pin.

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 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 detection function enable control bit. When this bit is equal to 1, the address detection 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 the BNO bit. If the address bit known as the 8th or 9th bit of the received word is "0" with the address detection function being enabled, an interrupt will not be generated and the received data will be discarded.

Bit 3 **WAKE**: RX pin wake-up UART function enable control
0: RX pin wake-up UART function is disabled
1: RX pin wake-up UART function is enabled
This bit is used to control the wake-up UART function when a falling edge on the RX pin occurs. Note that this bit is only available when the UART clock (f_H) is switched off. There will be no RX pin wake-up UART function if the UART clock (f_H) exists. If the WAKE bit is set to 1 as the UART clock (f_H) is switched off, a UART wake-up request will be initiated when a falling edge on the RX pin occurs. When this request happens and the corresponding interrupt is enabled, an RX pin wake-up UART interrupt will be generated to inform the MCU to wake up the UART function by switching on the UART clock (f_H) via the application program. Otherwise, the UART function can not resume even if there is a falling edge on the RX pin when the WAKE bit is cleared to 0.

Bit 2 **RIE**: Receiver interrupt enable control
0: Receiver related interrupt is disabled
1: Receiver related interrupt is enabled
The bit enables or disables the receiver interrupt. If this bit is equal to 1 and when the receiver overrun flag OERR or received 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
The 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
The 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.

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 BRG register and the second is the value of the BRGH bit within the UCR2 control register. 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)	f_H / [64 (N+1)]	f_H / [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	BRG7	BRG6	BRG5	BRG4	BRG3	BRG2	BRG1	BRG0
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 **BRG7~BRG0**: Baud Rate values

By programming the BRGH bit in the 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 cleared to zero 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=f_H / [64 (N+1)]

Re-arranging this equation gives N=[f_H / (BR×64)] - 1

Giving a value for N=[4000000 / (4800×64)] - 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=4000000 / [64 × (12+1)]=4808

Therefore the error is equal to (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 transm it and receive data is setup using the internal 8-bit baud rate generator, while the data is transmitted and received LSB first. Although the transmitter and receiver of the UART 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 these two pins will be used as I/O or other pin-shared functional pins. 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 enable control, 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.

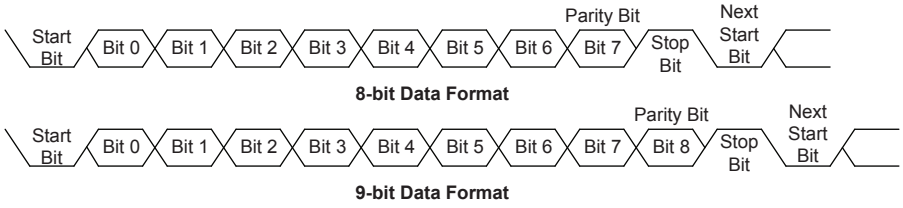
Data, Parity and Stop Bit Selection

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 if odd or even parity. The PREN bit controls the parity on/off function. 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, which is the MSB of the data byte, is used to identify the frame as an address character or data if the address detect function is enabled. The number of stop bits, which can be either one or two, is independent of the data length and is only used for the transmitter. There is only one stop bit for the receiver.

Start Bit	Data Bits	Address Bits	Parity Bit	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.



UART Transmitter

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

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 LSB first. In the transmit mode, the TXR_RXR register forms a buffer between the internal bus and the transmitter shift register. It should be noted that if 9-bit data format has been selected, then the MSB will be taken from the TX8 bit in the UCR1 register. The steps to initiate a data transfer can be summarized as follows:

- Make the correct selection of the BNO, 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_RXR register. Note that this step will clear the TXIF bit.

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

1. A USR register access

2. A TXR_RXR register write execution

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

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

1. A USR register access

2. A TXR_RXR register write execution

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

Transmitting Break

If the TXBRK bit is set, then the break characters will be sent on the next transmission. Break character transmission consists of a start bit, followed by 13xN "0" 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 high 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 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, will be stored in the RX8 bit in the UCR1 register. At the receiver core lies the Receiver Shift Register more 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 TXR_RXR register is a two byte deep FIFO data buffer, where two bytes can be held in the FIFO while the 3rd byte can continue to be received. Note that the application program must ensure that the data is read from TXR_RXR before the 3rd byte has been completely shifted in, otherwise the 3rd 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 the BNO, PRT and PREN bits to define the required 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 RXIFn bit in the UnSR register will be set when the TXR_RXRn register has data available. There will be at most one more character available before an overrun error occurs.
- When the contents of the shift register have been transferred to the TXR_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:

1. A USR register access
2. A TXR_RXR register read execution

Receiving Break

Any break character received by the UART will be managed as a framing error. The receiver will count and expect a certain number of bit times as specified by the values programmed into the BNO plus one stop 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 plus 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. A break is regarded as a character that contains only zeros with the FERR flag being set. If a long break signal has been detected, the receiver will regard it as a data frame including a start bit, data bits and the invalid stop bit and the FERR flag will be set. The receiver must wait for a valid stop bit before looking for the next start bit. The receiver will not make the assumption that the break condition on the line is the next start bit. The break character will be loaded into the buffer and no further data will be received until stop bits are received. It should be noted that the RIDLE read only flag will go high when the stop bits have not yet been received. The reception of a break character on the UART registers will result in the following:

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

Idle Status

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

Receiver Interrupt

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

Managing Receiver Errors

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

Overrun Error – OERR

The TXR_RXR register is composed of a two byte deep FIFO data buffer, where two bytes can be held in the FIFO register, while a 3th byte can continue to be received. Before the 3th byte has been entirely shifted in, the data should be read from the TXR_RXR register. If this is not done, the overrun error flag OERR will be consequently indicated.

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

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

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

Noise Error – NF

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

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

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

Framing Error – FERR

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

Parity Error – PERR

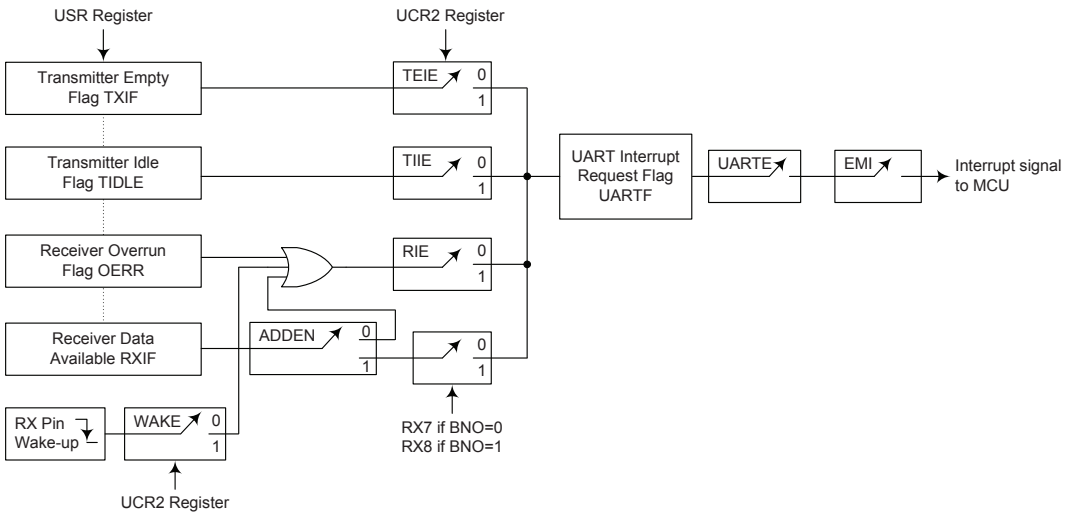
The read only parity error flag, PERR, in the USR register, is set if the parity of the received word is incorrect. This error flag is only applicable if the parity function is enabled, PREN=1, and if the parity type, odd or even, is selected. The read only PERR flag is buffered along with the received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

UART Interrupt Structure

Several individual UART conditions can generate a UART interrupt. When these conditions exist, a low pulse will be generated to get the attention of the microcontroller. These conditions are a transmitter data register empty, transmitter idle, receiver 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 function enable control bit, ADDEN, in the UCR2 register, enables this special function. If this bit is set to 1, 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 equal to 1, then when the 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 of the microcontroller must also be enabled for correct interrupt generation. The highest address bit is the 9^{th} bit if the bit BNO=1 or the 8^{th} bit if the bit BNO=0. If the highest 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 equal to 0, then a Receive Data Available interrupt will be generated each time the RXIF flag is set, irrespective of the data last bit status. The address detection and parity functions are mutually exclusive functions. Therefore, if the address detect function is enabled, then to ensure correct operation, the parity function should be disabled by resetting the parity function 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 UART clock (f_H) is off, the UART will cease to function, all clock sources to the module are shutdown. If the UART clock (f_H) is off while a transmission is still in progress, then the transmission will be paused until the UART clock source derived from the microcontroller is activated. In a similar way, if the MCU enters the Power Down Mode while receiving data, then the reception of data will likewise be paused. When the MCU enters the Power Down 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 Power Down 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 when the UART clock (f_H) is off, then a falling edge on the RX pin will trigger an RX pin wake-up UART interrupt. Note that as it takes certain system clock cycles after a wake-up, before normal microcontroller operation resumes, any data received during this time on the RX 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, UARTE, must be set. If the EMI and UARTE 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.

Low Voltage Detector – LVD

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

LVD Register

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

LVDC Register

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

Bit 7~6 Unimplemented, read as "0"

Bit 5 **LVDO**: LVD output flag
0: No Low Voltage Detected
1: Low Voltage Detected

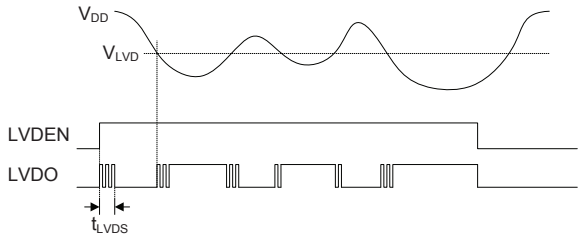
Bit 4 **LVDEN**: Low Voltage Detector Enable control
0: Disable
1: Enable

Bit 3 Unimplemented, read as "0"

Bit 2~0 **VLVD2~VLVD0**: LVD Voltage selection
000: 2.0V
001: 2.2V
010: 2.4V
011: 2.7V
100: 3.0V
101: 3.3V
110: 3.6V
111: 4.0V

LVD Operation

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



LVD Operation

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

Interrupts

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

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

Function	Enable Bit	Request Flag	Notes
Global	EMI	—	—
INTn Pins	INTnE	INTnF	n=0 ~ 1
Multi-function	MFnE	MFnF	n=0 ~ 2
A/D Converter	ADE	ADF	—
Time Base	TBnE	TBnF	n=0 ~ 1
SIM	SIME	SIMF	—
UART	UARTE	UARTF	—
LVD	LVE	LVF	—
EEPROM	DEE	DEF	—
TM	TnPE	TnPF	n=0 ~ 1
	TnAE	TnAF	n=0 ~ 1

Interrupt Register Bit Naming Conventions

Register Name	Bit							
	7	6	5	4	3	2	1	0
INTEG	—	—	—	—	INT1S1	INT1S0	INT0S1	INT0S0
INTC0	—	MF0F	—	INT0F	MF0E	—	INT0E	EMI
INTC1	TB0F	ADF	MF2F	MF1F	TB0E	ADE	MF2E	MF1E
INTC2	UARTF	SIMF	INT1F	TB1F	UARTE	SIME	INT1E	TB1E
MFI0	—	—	T0AF	T0PF	—	—	T0AE	T0PE
MFI1	—	—	T1AF	T1PF	—	—	T1AE	T1PE
MFI2	—	—	DEF	LVF	—	—	DEE	LVE

Interrupt Registers List

INTEG Register

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

Bit 7~4 Unimplemented, read as "0"

Bit 3~2 **INT1S1~INT1S0**: Interrupt edge control for INT1 pin
00: Disable
01: Rising edge
10: Falling edge
11: Rising and falling edges

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

INTC0 Register

Bit	7	6	5	4	3	2	1	0
Name	—	MF0F	—	INT0F	MF0E	—	INT0E	EMI
R/W	—	R/W	—	R/W	R/W	—	R/W	R/W
POR	—	0	—	0	0	—	0	0

Bit 7 Unimplemented, read as "0"

Bit 6 **MF0F**: Multi-function 0 interrupt request flag
0: No request
1: Interrupt request

Bit 5 Unimplemented, read as "0"

Bit 4 **INT0F**: INT0 interrupt request flag
0: No request
1: Interrupt request

Bit 3 **MF0E**: Multi-function 0 interrupt control
0: Disable
1: Enable

Bit 2 Unimplemented, read as "0"

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	TB0F	ADF	MF2F	MF1F	TB0E	ADE	MF2E	MF1E
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7 **TB0F**: Time Base 0 interrupt request flag
0: No request
1: Interrupt request

Bit 6 **ADF**: A/D Converter interrupt request flag
0: No equest
1: Interrupt request

Bit 5 **MF2F**: Multi-function 2 interrupt request flag
0: No equest
1: Interrupt request

Bit 4 **MF1F**: Multi-function 1 interrupt request flag
0: No equest
1: Interrupt request

Bit 3 **TB0E**: Time Base 0 interrupt control
0: Disable
1: Enable

Bit 2 **ADE**: A/D Converter 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

Bit	7	6	5	4	3	2	1	0
Name	UARTF	SIMF	INT1F	TB1F	UARTE	SIME	INT1E	TB1E
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7 **UARTF**: UART interrupt request flag
0: No equest
1: Interrupt Request

Bit 6 **SIMF**: SIM interrupt request flag
0: No equest
1: Interrupt Request

Bit 5 **INT1F**: INT1 interrupt request flag
0: No equest
1: Interrupt Request

Bit 4 **TB1F**: Time Base 1 interrupt request flag
0: No equest
1: Interrupt Request

Bit 3 **UARTE**: UART interrupt control
0: Disable
1: Enable

Bit 2 **SIME**: SIM interrupt control
0: Disable
1: Enable

Bit 1 **INT1E**: INT1 interrupt control
0: Disable
1: Enable

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

MFI0 Register

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

Bit 7~6 Unimplemented, read as "0"

Bit 5 **T0AF**: TM0 Comparator A match Interrupt request flag
0: No equest
1: Interrupt request

Bit 4 **T0PF**: TM0 Comparator P match Interrupt request flag
0: No equest
1: Interrupt request

Bit 3~2 Unimplemented, read as "0"

Bit 1 **T0AE**: TM0 Comparator A match Interrupt control
0: Disable
1: Enable

Bit 0 **T0PE**: TM0 Comparator P match Interrupt control
0: Disable
1: Enable

MFI1 Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	T1AF	T1PF	—	—	T1AE	T1PE
R/W	—	—	R/W	R/W	—	—	R/W	R/W
POR	—	—	0	0	—	—	0	0

Bit 7~6 Unimplemented, read as "0"

Bit 5 **T1AF**: TM1 Comparator A match Interrupt request flag
0: No equest
1: Interrupt request

Bit 4 **T1PF**: TM1 Comparator P match Interrupt request flag
0: No equest
1: Interrupt request

Bit 3~2 Unimplemented, read as "0"

Bit 1 **T1AE**: TM1 Comparator A match Interrupt control
0: Disable
1: Enable

Bit 0 **T1PE**: TM1 Comparator P match Interrupt control
0: Disable
1: Enable

MFI2 Register

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

Bit 7~6 Unimplemented, read as "0"

Bit 5 **DEF**: Data EEPROM Interrupt request flag
0: No equest
1: Interrupt request

Bit 4 **LVF**: LVD Interrupt request flag
0: No equest
1: Interrupt request

Bit 3~2 Unimplemented, read as "0"

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

Bit 0 **LVE**: LVD Interrupt control
0: Disable
1: Enable

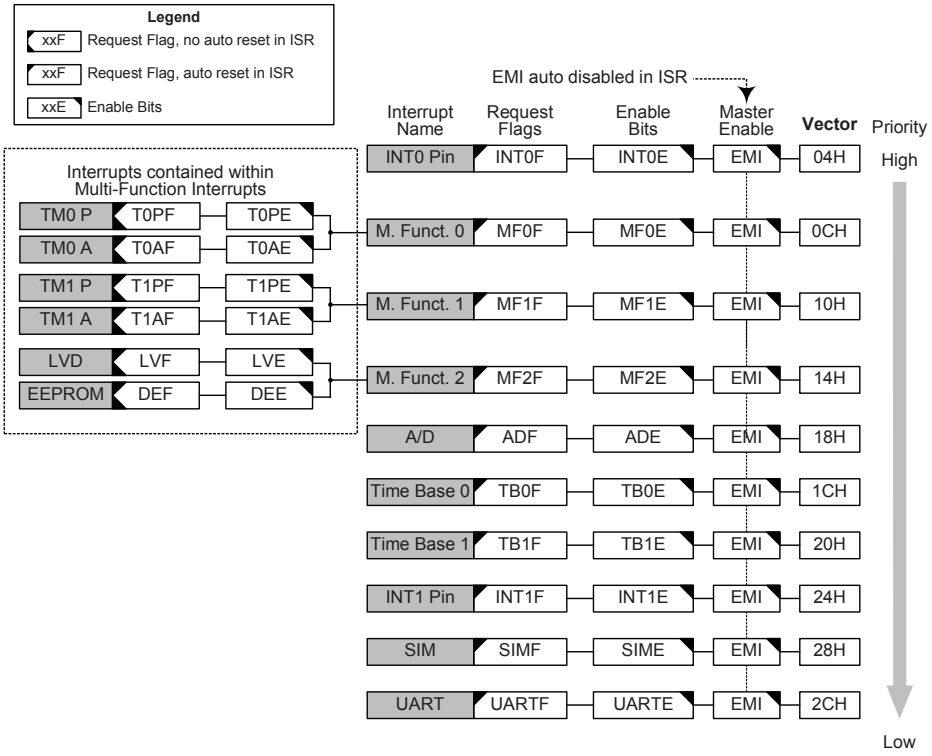
Interrupt Operation

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

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

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

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



Interrupt Scheme

External Interrupt

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

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

Multi-function Interrupt

Within the device 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, LVD interrupt and EEPROM write operation interrupt.

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

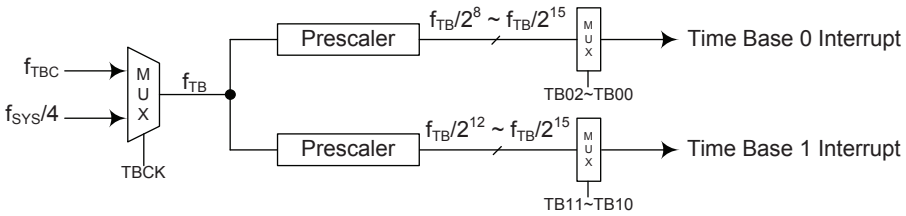
However, it must be noted that, although the Multi-function Interrupt request flags will be automatically reset when the interrupt is serviced, the request flags from the original source of the Multi-function interrupts will not be automatically reset and must be manually reset by the application program.

A/D Converter Interrupt

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

Time Base Interrupt

The function of the Time Base Interrupts is to provide regular time signal in the form of an internal interrupt. They are controlled by the overflow signals from their respective timer functions. When these happens their respective interrupt request flags, TB0F or TB1F will be set. To allow the program to branch to their respective interrupt vector addresses, the global interrupt enable bit, EMI and Time Base enable bits, TB0E or TB1E, must first be set. When the interrupt is enabled, the stack is not full and the Time Base overflows, a subroutine call to their respective vector locations will take place. When the interrupt is serviced, the respective interrupt request flag, TB0F or TB1F, will be automatically reset and the EMI bit will be cleared to disable other interrupts. The purpose of the Time Base Interrupt is to provide an interrupt signal at fixed time periods. Their clock sources originate from the internal clock source f_{TB}. This f_{TB} input clock passes through a divider, the division ratio of which is selected by programming the appropriate bits in the TBC register to obtain longer interrupt periods whose value ranges. The clock source that generates f_{TB}, which in turn controls the Time Base interrupt period, can originate from several different sources, as shown in the System Operating Mode section.



Time Base Interrupts

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**: Time Base function enable control
0: Disable
1: Enable

Bit 6 **TBCK**: Time Base clock source select
0: f_{TBC}
1: $f_{SYS}/4$

Bit 5~4 **TB11~TB10**: Time Base 1 time-out period selection
00: $2^{12}/f_{TB}$
01: $2^{13}/f_{TB}$
10: $2^{14}/f_{TB}$
11: $2^{15}/f_{TB}$

Bit 3 **LXTLP**:LXT Low Power control
0: Disable
1: Enable

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

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, UARTE, 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, UARTF, will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts. However, the USR register flags will only be cleared when certain actions are taken by the UART, the details of which are given in the UART chapter.

LVD Interrupt

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

EEPROM Interrupt

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

TM Interrupt

The Periodic TMs have 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. There are two interrupt request flags and two enable control bits. A TM interrupt request will take place when any of the TM request flags are set, a situation which occurs when a TM comparator P or A match situation happens.

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

Interrupt Wake-up Function

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

Programming Considerations

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

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

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

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

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

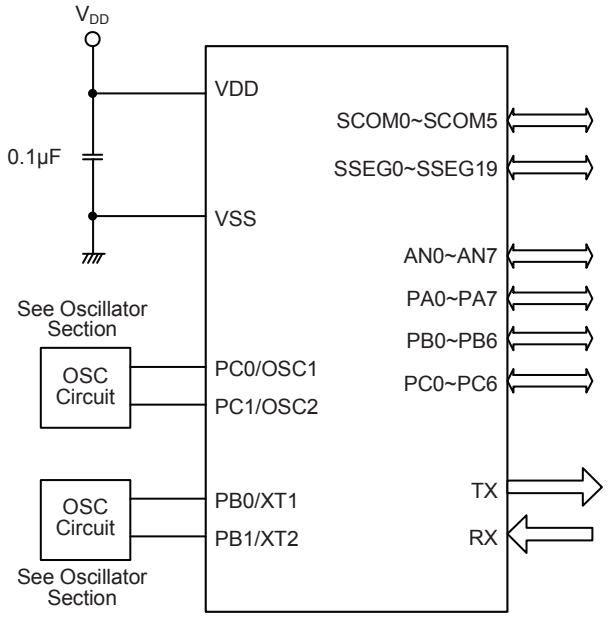
To return from an interrupt subroutine, either a RET or RETI instruction may be executed. The RETI instruction in addition to executing a return to the main program also automatically sets the EMI bit high to allow further interrupts. The RET instruction however only executes a return to the main program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

Configuration Options

Configuration options refer to certain options within the MCU that are programmed into the device during the programming process. During the development process, these options are selected using the HT-IDE software development tools. 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
1	High Speed System Oscillator Selection f_H – HXT or HIRC
2	Low Speed System Oscillator Selection f_{SUB} – LXT or LIRC
3	HIRC Frequency Selection f_{HIRC} – 8MHz, 12MHz or 16MHz

Application Circuits



Instruction Set

Introduction

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

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

Instruction Timing

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

Moving and Transferring Data

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

Arithmetic Operations

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

Logical and Rotate Operation

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

Branches and Control Transfer

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

Bit Operations

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

Table Read Operations

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

Other Operations

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

Instruction Set Summary

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

Table Conventions

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

Mnemonic	Description	Cycles	Flag Affected
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 is rotated right by 1 bit with bit 0 rotated into bit 7. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

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

Affected flag(s): None

RRC [m] Rotate Data Memory right through Carry

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

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

Affected flag(s): C

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

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

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

Affected flag(s): C

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

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

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

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

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

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

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

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

SDZ [m] Skip if decrement Data Memory is 0

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

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

Affected flag(s): None

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

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

Operation ACC ← [m] − 1
Skip if ACC=0

Affected flag(s) None

SET [m] Set Data Memory

Description Each bit of the specified Data Memory is set to 1.

Operation [m] ← FFH

Affected flag(s) None

SET [m].i Set bit of Data Memory

Description Bit i of the specified Data Memory is set to 1.

Operation [m].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

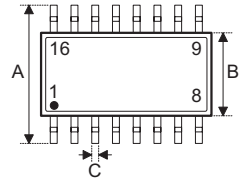
Package Information

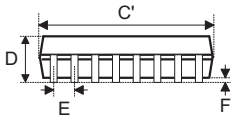
Note that the package information provided here is for consultation purposes only. As this information may be updated at regular intervals users are reminded to consult the Holtek website for the latest version of the Package/Carton Information.

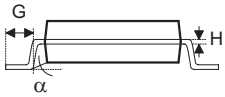
Additional supplementary information with regard to packaging is listed below. Click on the relevant section to be transferred to the relevant website page.

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

16-pin NSOP (150mil) Outline Dimensions



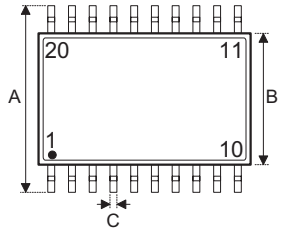


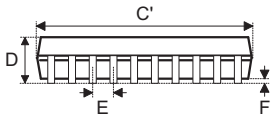


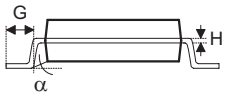
Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	—	0.236 BSC	—
B	—	0.154 BSC	—
C	0.012	—	0.020
C’	—	0.390 BSC	—
D	—	—	0.069
E	—	0.050 BSC	—
F	0.004	—	0.010
G	0.016	—	0.050
H	0.004	—	0.010
α	0°	—	8°

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	—	6 BSC	—
B	—	3.9 BSC	—
C	0.31	—	0.51
C’	—	9.9 BSC	—
D	—	—	1.75
E	—	1.27 BSC	—
F	0.10	—	0.25
G	0.40	—	1.27
H	0.10	—	0.25
α	0°	—	8°

20-pin NSOP (150mil) Outline Dimensions



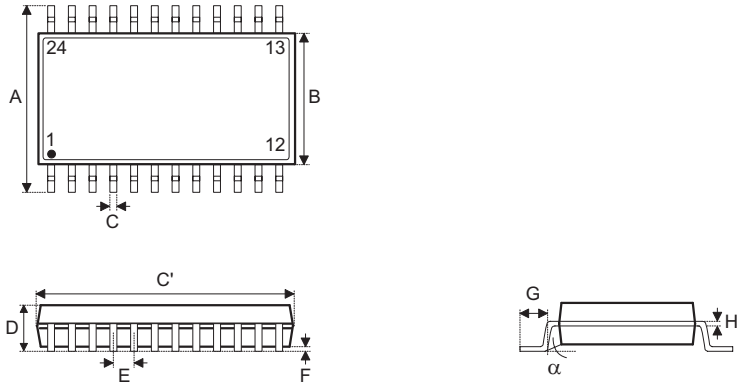




Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	0.228	0.236	0.244
B	0.146	0.154	0.161
C	0.009	—	0.012
C’	0.382	0.390	0.398
D	—	—	0.069
E	—	0.032 BSC	—
F	0.002	—	0.009
G	0.020	—	0.031
H	0.008	—	0.010
α	0°	—	8°

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	5.80	6.00	6.20
B	3.70	3.90	4.10
C	0.23	—	0.30
C’	9.70	9.90	10.10
D	—	—	1.75
E	—	0.80 BSC	—
F	0.05	—	0.23
G	0.50	—	0.80
H	0.21	—	0.25
α	0°	—	8°

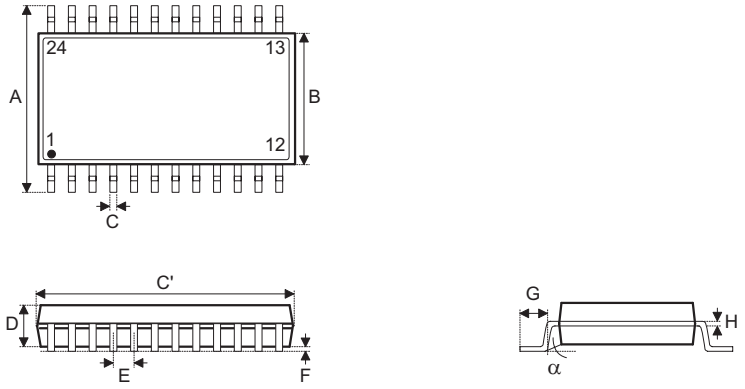
24-pin SOP (300mil) Outline Dimensions



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	—	7.5 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°

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	—	6.000 BSC	—
B	—	3.900 BSC	—
C	0.20	—	0.30
C’	—	8.660 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°