PIR 8-Bit Flash MCU

HT45F0027

Revision: V1.30 Date: February 25, 2019

www.holtek.com

Table of Contents

Features

CPU Features

- Operating Voltage
  - f_{SYS} = 32kHz: 2.2V~5.5V
  - f_{SYS} = 1MHz: 2.2V~5.5V
  - f_{SYS} = 2MHz: 2.2V~5.5V
  - f_{SYS} = 4MHz: 2.2V~5.5V
  - f_{SYS} = 8MHz: 3.3V~5.5V
- TinyPower™ technology for low power operation
- Power down and wake-up functions to reduce power consumption
- Oscillators
  - Internal RC – HIRC
  - Internal 32kHz RC – LIRC
- Multi-mode operation: NORMAL, SLOW, IDLE and SLEEP
- Fully integrated internal 1/2/4/8 MHz oscillator requires no external components
- All instructions executed in one or two instruction cycles
- Table read instructions
- 63 powerful instructions
- 6-level subroutine nesting
- Bit manipulation instruction

Peripheral Features

- Flash Program Memory: 2K×16
- RAM Data Memory: 256×8
- EEPROM Memory: 32×8
- Watchdog Timer function
- 9 bidirectional I/O lines
- Single pin-shared external interrupt
- Two 8-bit programmable Timer/Event Counters with overflow interrupt function
- 6-channel 12-bit resolution A/D converter
- A/D converter auto enable function
- A/D converter lower/upper limit preset
- Serial Interfaces Module - SIM for SPI or I^2C
- Dual Operational Amplifiers functions
- LDO function
- Internal Temperature Sensor function
- Low Voltage Reset function
- Low Voltage Detect function
- Package type: 16-pin NSOP/QFN

General Description

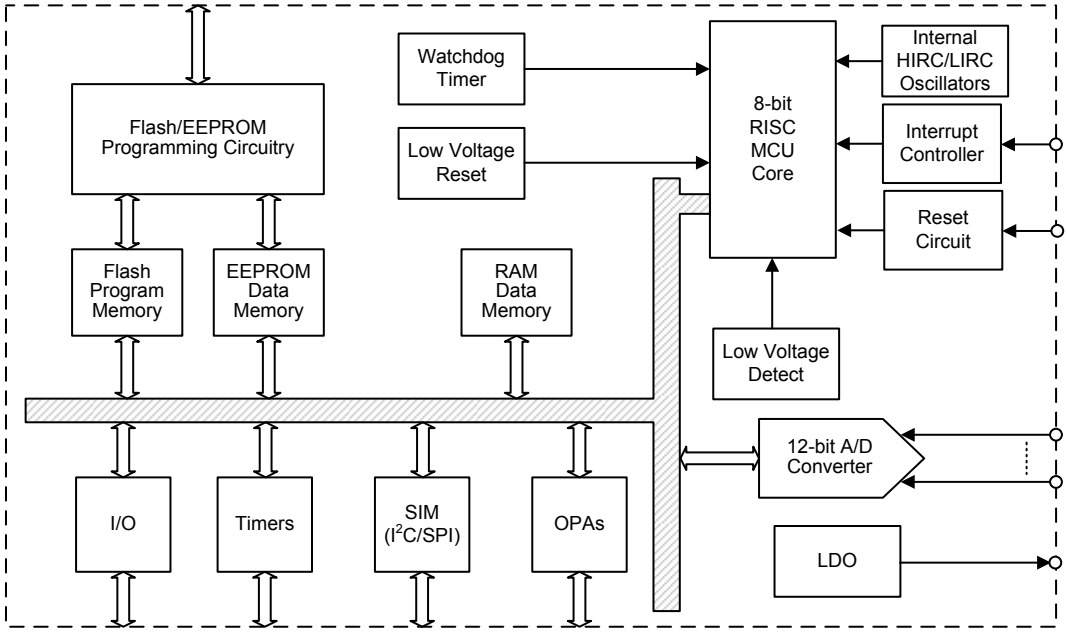
The HT45F0027 is a TinyPower™ A/D type 8-bit high performance RISC architecture microcontroller, designed especially for applications that interface directly to analog signal. The device includes an integrated multi-channel Analog to Digital Converter and low power internal operational amplifiers.

Offering users the convenience of Flash Memory multi-programming features, the device also includes wide range of functions and features. Other memory includes an area of RAM Data Memory as well as an area of EEPROM memory for storage of non-volatile data such as serial numbers, calibration data etc.

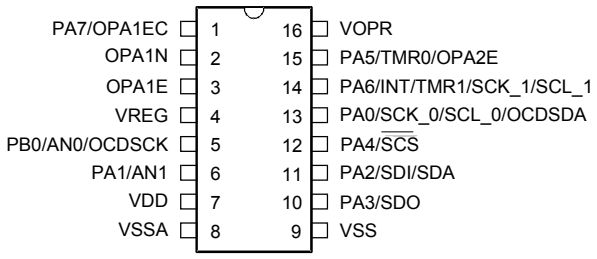
The usual Holtek MCU features such as power down and wake-up functions, oscillator options, etc. combine to ensure user applications require a minimum of external components.

The benefits of an integrated A/D, SPI and I^2C functions, in addition to low power consumption, high performance, I/O flexibility and low-cost, provides the device with the versatility for a wide range of products in the home appliance and industrial application areas. Some of these products could include electronic metering, environmental monitoring, handheld instruments, electronically controlled tools, motor driving etc.

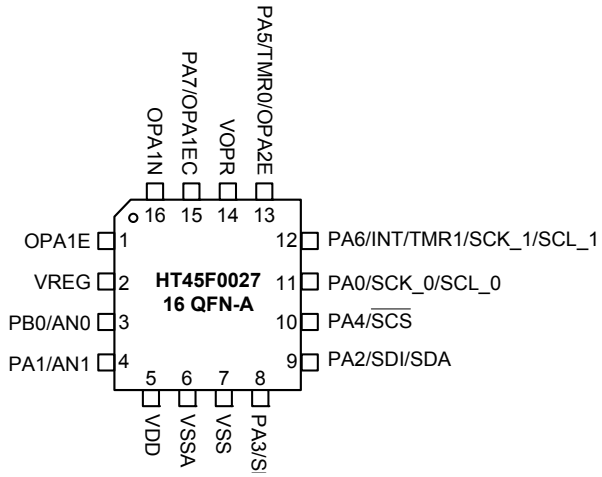
Block Diagram



Pin Assignment



HT45F0027/HT45V0027
16 NSOP-A



Note: 1. If the pin-shared pin functions have multiple outputs simultaneously, its pin names at the right side of the "/" sign can be used for higher priority.
2. The OCDSDA and OCDSCK pins are the OCDS dedicated pins and only available for the HT45V0027 device which is the OCDS EV chip for the HT45F0027 device.

Pin Description

With the exception of the power pins, all pins on the device can be referenced by its Port name, e.g. PA0, PA1 etc., which refer to the digital I/O function of the pins. However these Port pins are also shared with other function such as the Analog to Digital Converter, Timer pins etc. The function of each pin is listed in the following table, however the details behind how each pin is configured is contained in other sections of the datasheet.

Pin Name	Function	OPT	I/T	O/T	Description
PA0/SCK_0/SCL_0/OCDSDA	PA0	PAPU PAWU	ST	CMOS	General purpose I/O. Register enabled pull-high and wake-up.
	SCK	—	ST	—	SPI serial clock
	SCL	—	ST	NMOS	I²C clock
	OCDSDA	—	ST	CMOS	OCDS Address/Data, for EV chip only
PA1/AN1	PA1	PAPU PAWU	ST	CMOS	General purpose I/O. Register enabled pull-high and wake-up.
	AN1	ACER	AN	—	A/D Converter input channel 1

Pin Name	Function	OPT	I/T	O/T	Description
PA2/SDI/SDA	PA2	PAPU PAWU	ST	CMOS	General purpose I/O. Register enabled pull-high and wake-up.
	SDI	—	ST	—	SPI data input
	SDA	—	ST	NMOS	I²C data
PA3/SDO	PA3	PAPU PAWU	ST	CMOS	General purpose I/O. Register enabled pull-high and wake-up.
	SDO	—	—	CMOS	SPI data output
PA4/$\overline{SCS}$	PA4	PAPU PAWU	ST	CMOS	General purpose I/O. Register enabled pull-high and wake-up.
	$\overline{SCS}$	—	ST	—	SPI slave select
PA5/TMR0/OPA2E	PA5	PAPU PAWU	ST	CMOS	General purpose I/O. Register enabled pull-high and wake-up.
	TMR0	TMR0C	ST	—	Timer/Event 0 counter input
	OPA2E	OPAC1	—	AN	OPA2 output
PA6/INT/TMR1/ SCK_1/SCL_1	PA6	PAPU PAWU	ST	CMOS	General purpose I/O. Register enabled pull-high and wake-up.
	INT	—	ST	—	External interrupt input pin
	TMR1	TMR1C	ST	—	Timer/Event 1 counter input
	SCK	—	ST	—	SPI serial clock
	SCL	—	ST	NMOS	I²C clock
PA7/OPA1EC	PA7	PAPU PAWU	ST	CMOS	General purpose I/O. Register enabled pull-high and wake-up.
	OPA1EC	OPAC0	—	AN	OPA1 output via capacitor
PB0/AN0/OCDSCK	PB0	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-high.
	AN0	ACER	AN	—	A/D Converter input channel 0
	OCDSCK	—	ST	—	OCDS clock pin, for EV chip only
VREG	VREG	—	—	—	LDO output voltage
VOPR	VOPR	—	AN	—	OPA1 & OPA2 Internal Reference Voltage
OPA1N	OPA1N	—	AN	—	OPA1 inverting input
OPA1E	OPA1E	—	—	AN	OPA1 output
VDD	VDD	—	PWR	—	Power supply
VSS	VSS	—	PWR	—	Digital Ground
VSSA	VSS	—	PWR	—	Analog Ground

Note: I/T: Input type; O/T: Output type;
OPT: Optional by configuration option (CO) or register option;
PWR: Power; CO: Configuration option;
ST: Schmitt Trigger input; CMOS: CMOS output;
NMOS: NMOS output AN: Analog signal pin;

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 150mA
I_{OH} Total ..... -100mA
Total Power Dissipation 500mW

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

D.C. Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	Operating Voltage (HIRC)	—	f_{SYS}=1MHz	2.2	—	5.5	V
			f_{SYS}=2MHz	2.2	—	5.5	V
			f_{SYS}=4MHz	2.2	—	5.5	V
			f_{SYS}=8MHz	3.3	—	5.5	V
I_{DD}	Operating Current (HIRC)	3.3V	No load, f_{SYS}=f_M=1MHz ADC off, LVR off, OPAs off	—	120	180	μA
		3.3V	No load, f_{SYS}=f_M=2MHz ADC off, LVR off, OPAs off	—	180	260	μA
		3V	No load, f_{SYS}=f_M=4MHz ADC off	—	300	450	μA
		5V		—	600	900	μA
		5V	No load, f_{SYS}=f_M=8MHz ADC off	—	1.3	2.0	mA
		3V	No load, f_{SYS} = f_L, f_S=f_{SUB}=f_{LIRC}, f_H=4MHz ,f_L=f_H/4, ADC off	—	150	250	μA
		5V		—	400	600	μA
		3V	No load, f_{SYS} = f_L, f_S=f_{SUB}=f_{LIRC} f_H=4MHz ,f_L=f_H/2, ADC off	—	250	350	μA
		5V		—	500	750	μA
		3V	No load, f_{SYS} = f_L, f_S=f_{SUB}=f_{LIRC} f_H=8MHz ,f_L=f_H/4, ADC off	—	300	450	μA
		5V		—	680	1020	μA
		3V	No load, f_{SYS} = f_L, f_S=f_{SUB}=f_{LIRC} f_H=8MHz ,f_L=f_H/2, ADC off	—	450	700	μA
		5V		—	900	1400	μA
	Operating Current (LIRC)	3V	No load, WDT off, ADC off	—	10	20	μA
		5V		—	20	35	μA

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{STB}	Standby Current(Sleep) (f_{SYS}, f_{SUB}, f_S, f_{WDT}=off)	3V	No load, System HALT, WDT off	—	0.2	0.8	μA
		5V		—	0.5	1	μA
	Standby Current (Sleep) (f_{SYS} off; f_S on; f_{WDT}=f_{SUB}=LIRC)	3V	No load, System HALT, WDT on	—	2	4	μA
		5V		—	4	6	μA
	Standby Current (Idle) (f_{SYS}, f_{WDT} off; f_S=f_{SUB}=LIRC)	3V	No load, System HALT, WDT off	—	4	6	μA
		5V		—	6	9	μA
	Standby Current (Idle) (f_{SYS} on, f_{SYS} =f_M=4MHz; f_{WDT} off; f_S=f_{SUB}=LIRC)	3V	No load, System HALT, WDT off, SPI or I^2C on	—	150	250	μA
		5V		—	350	660	μA
	Standby Current(Sleep) (f_{SYS}, f_{SUB}, f_S, f_{WDT}=off)	3V	No load, System HALT, WDT off	—	0.1	1	μA
		5V		—	0.2	2	μA
V_{IL}	Input Low Voltage (I/O)	5V	—	0	—	1.5	V
		—	—	0	—	$0.2V_{DD}$	V
V_{IH}	Input High Voltage (I/O)	5V	—	3.5	—	5.0	V
		—	—	$0.8V_{DD}$	—	V_{DD}	V
V_{LVR}	Low Voltage Reset Voltage	—	LVR Enable, 2.1V option	-5%	2.10	+5%	V
			LVR Enable, 2.55V option		2.55		V
			LVR Enable, 3.15V option		3.15		V
			LVR Enable, 3.8V option		3.80		V
V_{LVD}	Low Voltage Detector Voltage	—	LVDEN = 1, V_{LVD} = 2.0V	-5%	2.0	+5%	V
			LVDEN = 1, V_{LVD} = 2.2V		2.2		V
			LVDEN = 1, V_{LVD} = 2.4V		2.4		V
			LVDEN = 1, V_{LVD} = 2.7V		2.7		V
			LVDEN = 1, V_{LVD} = 3.0V		3.0		V
			LVDEN = 1, V_{LVD} = 3.3V		3.3		V
			LVDEN = 1, V_{LVD} = 3.6V		3.6		V
			LVDEN = 1, V_{LVD} = 4.0V		4.0		V
I_{OL}	I/O Port Sink Current	3V	V_{OL}=0.1V_{DD}	6	12	—	mA
		5V		10	25	—	mA
I_{OH}	I/O Port Source Current	3V	V_{OH}=0.9V_{DD}	-2	-4	—	mA
		5V		-5	-8	—	mA
R_{PH}	Pull-high Resistance (I/O)	3V	—	40	60	80	kΩ
		5V		10	30	50	
V_{BG}	LDO Bandgap Reference with Buffer Voltage	—	—	-3%	1.25	+3%	V
I_{LVR}	DC Current when LVR or LVD Turn on	5V	—	—	20	30	μA

A.C. Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Condition				
f_{SYS}	System Clock (HIRC OSC)	—	2.2V~5.5V	1000	—	4000	kHz
		—	3.3V~5.5V	1000	—	8000	kHz
	System Clock (LIRC OSC)	—	2.2V~5.5V	—	32	—	kHz
f_{HIRC}	8MHz HIRC	3.3V	Ta=25°C	-2%	8	+2%	MHz
		3.3V	Ta=-40°C~85°C	-5%	8	+5%	MHz
		2.7V~5.5V	Ta=-40°C~85°C	-10%	8	+10%	MHz
	4MHz HIRC	3.3V	Ta=25°C	-2%	4	+2%	MHz
		3.3V	Ta=-40°C~85°C	-5%	4	+5%	MHz
		2.2V~5.5V	Ta=-40°C~85°C	-10%	4	+10%	MHz
	2MHz HIRC	3.3V	Ta=25°C	-2%	2	+2%	MHz
		3.3V	Ta=-40°C~85°C	-5%	2	+5%	MHz
		2.2V~5.5V	Ta=-40°C~85°C	-10%	2	+10%	MHz
	1MHz HIRC	3.3V	Ta=25°C	-2%	1	+2%	MHz
		3.3V	Ta=-40°C~85°C	-5%	1	+5%	MHz
		2.2V~5.5V	Ta=-40°C~85°C	-10%	1	+10%	MHz
f_{LIRC}	32kHz Internal RC OSC	3.3V	Ta=25°C	-10%	32	+10%	kHz
		2.2V~5.5V	Ta=-40°C~85°C	-60%	32	+60%	kHz
t_{LVR}	Low Voltage Width to Reset	—	—	120	240	480	μs
t_{BGS}	V_{BG} Turn On Stable Time	—	—	—	—	200	μs
t_{LVD}	Low Voltage Width to Interrupt	—	—	1	—	4	t_{SUB}
t_{LVDS}	LVDO Stable Time	5V	LVR disable, LVD enable. VBG is ready.	—	—	100	μs
t_{SST}	System Start-upTimer Period of HIRC	—	Power up or wake-up from HALT (IDLE or SLEEP mode)	—	16	22	t_{SYS}
	System Start-up Timer Period (With Fast Start-up) of LIRC	—	wake-up from Idle mode (f_{SL} = f_{LIRC})	—	1	4	t_{LIRC}
	System Start-up Timer Period (With Fast Start-up) of MCU	—	In HALT mode, when ADC sent a interrupt signal to MCU, the MCU will fast wake-up in 16 HIRC clocks.	—	—	16	t_{SYS}
t_{RSTD}	System Reset Delay Time (Power on Reset, LVR Reset, LVR S/W Reset(LVRC), WDT S/W Reset(WDTC))	—	—	25	50	100	ms
	System Reset Delay Time (WDT Normal Reset)	—	—	8.3	16.7	33.3	ms
t_{INT}	Interrupt Pulse Width	—	—	1	—	—	μs

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

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

OP Amplifier Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
D.C. Characteristic							
V_{DDO}	Operating Voltage	—	—	2.7	—	5.5	V
I_{DDO}	Quiescent Current	5V	No load	—	2	4	μA
I_{OPOS}	Input Offset Current	5V	V_{CM}=1/2V_{DD}, Ta=-40~85°C	—	10	—	nA
V_{OS}	OP Amp Input Offset Voltage	5V	—	-10	—	10	mV
GBW	OP Amp Gain Band Bandwidth	5V	—	2.5	5	—	kHz
PSRR	Power Supply Rejection Ratio	—	—	60	80	—	dB
CMRR	Common Mode Rejection Ratio	—	V_{IN}=(1/2)×V_{REG} or V_{IN}=(2/3)×V_{REG}	60	80	—	dB
A.C. Characteristic							
A_{OL}	Open Loop Gain	—	No load	60	80	—	dB
SR	Slew Rate+, Slew Rate-	—	No load	—	0.01	—	V/μs
t_{PD}	OPA Response Time	5V	No load	—	1.5	2	ms

LDO Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DDIN}	Supply Voltage	—	—	2.7	3.3	5.5	V
V_{DDOUT}	Output Voltage	—	VREG output decided by VSEL fields	-3%	V_{REG}	+3%	V
I_{REF}	Driving Current	—	V_{DDIN}=5V, VCAP=0.1μF	1	—	—	mA
I_{DD}	Current Consumption	5V	After startup, no load, include bandgap consumption	3	—	8	μA

Note:1.This LDO can provide stable power supply for PIR sensor with a 10μF cap.
2. The VREG pin should be connected to 0.1μF for ADC reference voltage and 10μF for PIR sensor.

A/D Converter Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	A/D Converter Operating Voltage	—	—	2.7	5.0	5.5	V
V_{ADI}	A/D Converter Input Voltage	—	—	0	—	AV_{DD}/ V_{REF}	V
V_{REF}	A/D Converter Reference Voltage	3V	—	2	—	AV_{DD}	V
		5V					
I_{ADC}	Additional Power Consumption if A/D Converter is used	3V	ADM=0	—	0.6	1.3	mA
			ADM=1	—	0.7	1.5	mA
		5V	ADM=0	—	1.0	2.0	mA
t_{AD}	A/D Converter Clock Period	2.7~5.5V	—	0.5	—	10	μs
t_{ADC}	A/D Conversion Time	2.7~5.5V	12-bit A/D Converter	16	—	20	t_{AD}
t_{ADS}	A/D Converter Sampling Time	2.7~5.5V	—	—	4	—	t_{AD}
t_{ON2ST}	A/D Converter On-to-Start Time	2.7~5.5V	—	4	—	—	μs
DNL	Differential Non-linearity	3V	V_{REF}=V_{DD} t_{AD} =0.5μs	-3	—	+3	LSB
		5V					
INL	Integral Non-linearity	3V	V_{REF} =V_{DD} t_{AD} =0.5μs	-4	—	+4	LSB
		5V					

Note: ADC conversion time (t_{ADC}) = n (bits ADC) + 4 (sampling time), the conversion for each bit needs one ADC clock (t_{AD}).

Temperature Sensor Characteristics

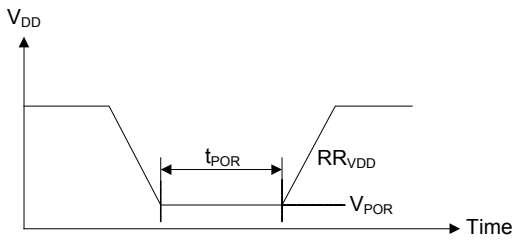
Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	Analog Voltage	—	—	2.7	—	5.5	V
REFO	Bandgap Output Voltage	3V	No load	-3%	1.04	+3%	V
V_{TPS}	Temperature Sensor Voltage	—	bypass pre-buffer	-10%	0.91	+10%	V
T_{slope}	Temperature Sensor Slope	—	Bypass pre-buffer	—	3.12	—	mV/°C

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_{VDD}	V_{DD} Raising Rate to Ensure Power-on Reset	—	—	0.035	—	—	V/ms
t_{POR}	Minimum Time for V_{DD} Stays at V_{POR} to Ensure Power-on Reset	—	—	1	—	—	ms

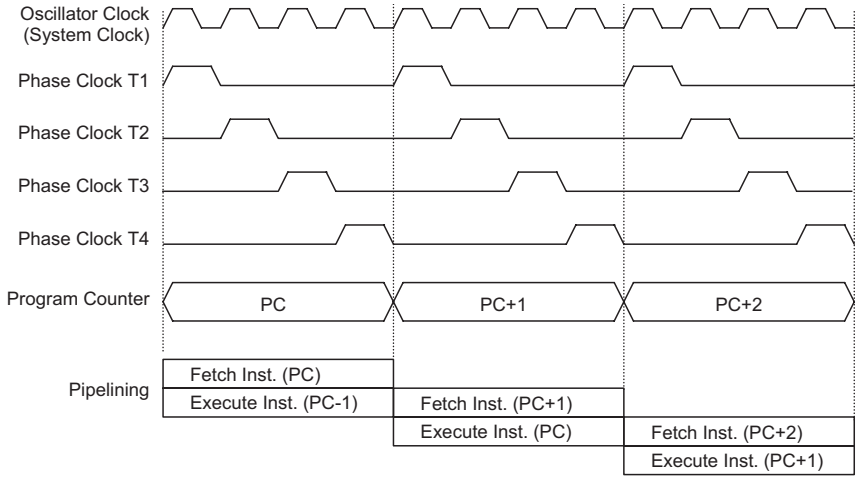


System Architecture

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

Clocking and Pipelining

The main system clock, derived from either a 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.

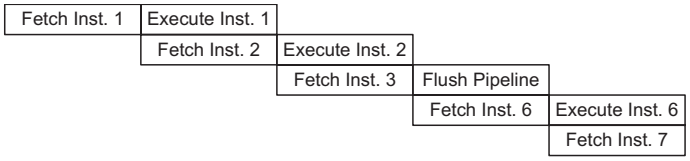


System Clocking and Pipelining

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.

```
1           MOV A,[12H]
2           CALL DELAY
3           CPL [12H]
4           :
5           :
6           NOP

    DELAY:
```



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 demands a jump to a non-consecutive Program Memory address. Only the lower 8 bits, known as the Program Counter Low Register, are directly addressable by the application program.

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

Program Counter	
Program Counter High Byte	**PCL Register**
PC10~PC8	PCL7~PCL0

Program Counter

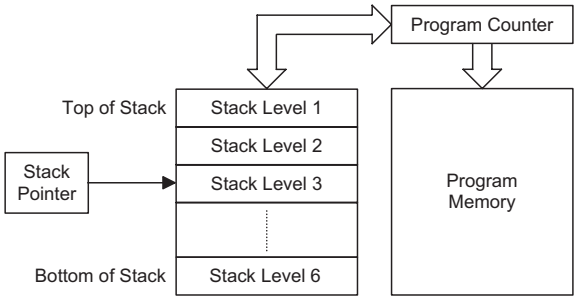
The lower byte of the Program Counter, known as the Program Counter Low register or PCL, is available for program control and is a readable and writeable register. By transferring data directly into this register, a short program jump can be executed directly. However, as only this low byte is available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

Stack

This is a special part of the memory which is used to save the contents of the Program Counter only. The stack is organized into 6 levels and neither part of the data nor part of the program space, and is neither readable nor writeable. The activated level is indexed by the Stack Pointer, and is neither readable nor writeable. At a subroutine call or interrupt acknowledge signal, the contents of the Program Counter are pushed onto the stack. At the end of a subroutine or an interrupt routine, signaled by a return instruction, RET or RETI, the Program Counter is restored to its previous value from the stack. After a device reset, the Stack Pointer will point to the top of the stack.

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

If the stack is overflow, the first Program Counter 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:

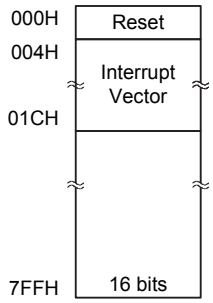
- Arithmetic operations: ADD, ADDM, ADC, ADCM, SUB, SUBM, SBC, SBCM, DAA
- Logic operations: AND, OR, XOR, ANDM, ORM, XORM, CPL, CPLA
- Rotation RRA, RR, RRCA, RRC, RLA, RL, RLCA, RLC
- Increment and Decrement INCA, INC, DECA, DEC
- Branch decision, JMP, SZ, SZA, SNZ, SIZ, SDZ, SIZA, SDZA, CALL, RET, 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 offer users the flexibility to conveniently debug and develop their applications while also offering a means of field programming and updating.

Structure

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



Program Memory Structure

Special Vectors

Within the Program Memory, certain locations are reserved for the reset and interrupts. The location 0000H 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. Any unused bits in this transferred higher order byte will be read as 0.

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

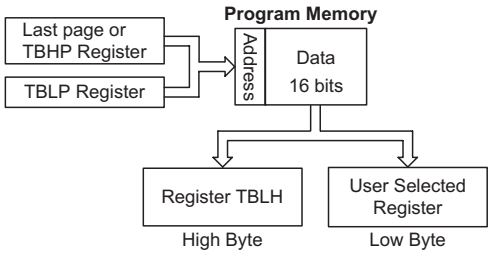


Table Program Example

The following example shows how the table pointer and table data is defined and retrieved from the microcontroller. This example uses raw table data located in the Program Memory which is stored there using the ORG statement. The value at this ORG statement is "0700H" which refers to the start address of the last page within the 2K Program Memory of the microcontroller. 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 page that TBHP pointed 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
:
```

On-Chip Debug Support – OCDS

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

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

RAM Data Memory

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

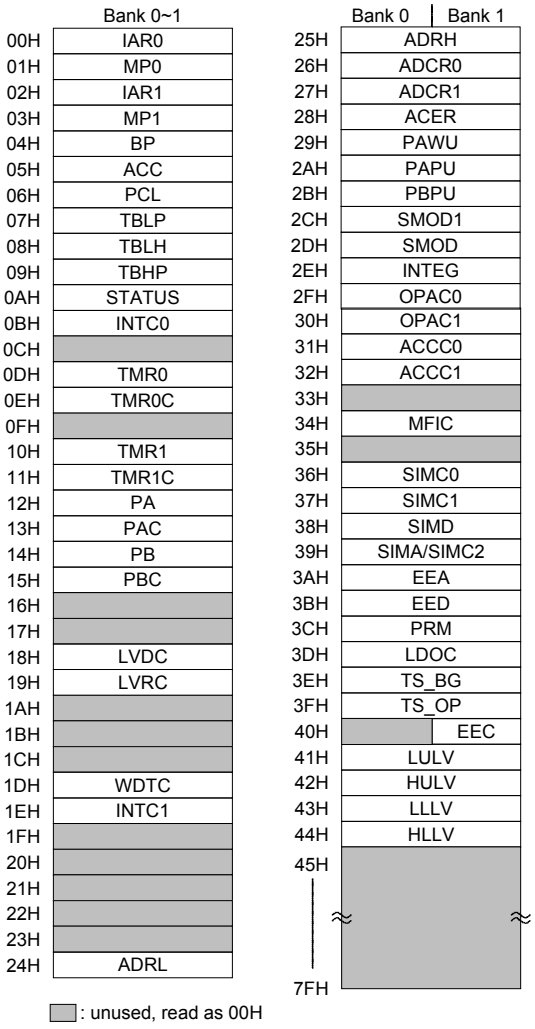
Structure

Divided into two areas, 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.

Capacity	Banks
256 × 8	0: 80H~FFH 1: 80H~FFH

General Purpose Data Memory



Special Purpose Data Memory

Special Function Register Description

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

Indirect Addressing Register – IAR0, IAR1

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

Memory Pointers – MP0, MP1

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

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

Indirect Addressing Program Example

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

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

Bank Pointer – BP

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

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

BP Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	—	BP0
R/W	—	—	—	—	—	—	—	R/W
POR	—	—	—	—	—	—	—	0

Bit 7~1 Unimplemented, read as "0"

Bit 0 **BP0**: Select Data Memory Banks
0: Bank 0
1: Bank 1

Accumulator – ACC

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

Program Counter Low Register – PCL

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

Look-up Table Registers – TBLP, TBHP, TBLH

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

Status Register – STATUS

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

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

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

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

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

STATUS Register

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

"x": unknown

Bit 7~6 Unimplemented, read as "0"

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

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

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

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

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

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

C is also affected by a rotate through carry instruction.

EEPROM Data memory

This 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 32×8 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.

Capacity	Address
32×8	00H~1FH

EEPROM Registers

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

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

Bit 7~5 Unimplemented, read as "0"

Bit 4~0 **D4~D0**: Data EEPROM address
Data EEPROM address bit 4 ~ 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
Data EEPROM data bit 7 ~ bit 0

EEC Register

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

Bit 7~4 Unimplemented, read as "0"

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

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

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

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

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

Reading Data from the EEPROM

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

Writing Data to the EEPROM

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

Write Protection

Protection against inadvertent write operation is provided in several ways. After the 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, 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. More details can be obtained in the Interrupt section.

Programming Considerations

Care must be taken that data is not inadvertently written to the EEPROM. Protection can be enhanced by ensuring that the Write Enable bit is normally cleared to zero when not writing. Also the Bank Pointer could be normally cleared to zero as this would inhibit access to Bank 1 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, 040H              ; setup memory pointer MP1
MOV  MP1, A               ; MP1 points to EEC register
MOV  A, 01H               ; setup Bank Pointer
MOV  BP, A
SET  IAR1.1               ; set RDEN bit, enable read operations
SET  IAR1.0               ; start Read Cycle - set RD bit
BACK:
SZ   IAR1.0               ; check for read cycle end
JMP  BACK
CLR  IAR1                 ; disable EEPROM read/write
CLR  BP
MOV  A, EED               ; move read data to register
MOV  READ_DATA, A
```

- **Writing Data to the EEPROM – polling method**

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

Oscillator

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

Oscillator Overview

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

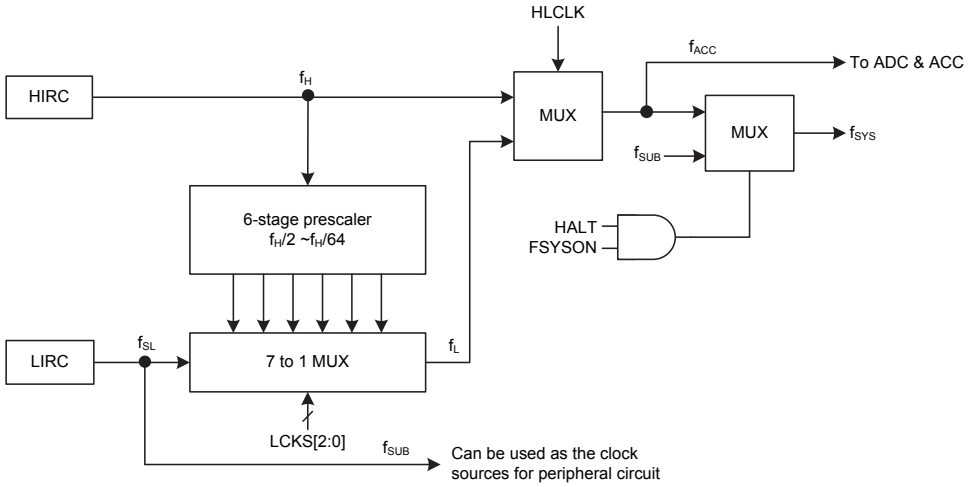
Type	Name	Freq.
Internal High Speed RC	HIRC	1, 2, 4, 8MHz
Internal Low Speed RC	LIRC	32kHz

Oscillator Types

System Clock Configurations

There are two methods of generating the system clock, one high speed oscillator and one low speed oscillator. The high speed oscillator is the internal 1MHz, 2MHz, 4MHz, 8MHz RC oscillator - HIRC. The low speed oscillator is the internal 32kHz RC oscillator – LIRC. Selecting whether the low or high speed oscillator is used as the system oscillator is implemented using the HLCLK bit and LCKS2~LCKS0 bits in the SMOD register and as the system clock can be dynamically selected.

The actual source clock used for the high speed oscillator is chosen via configuration options. The frequency of the slow speed or high speed system clock is also determined using the HLCLK bit and LCKS2~LCKS0 bits in the SMOD register. Note that two oscillator selections must be made namely one high speed and one low speed system oscillator. It is not possible to choose a no-oscillator selection for either the high or low speed oscillator.



System Clock Configurations

Internal RC Oscillator – HIRC

The internal RC oscillator is a fully integrated system oscillator requiring no external components. The internal RC oscillator has four fixed frequencies of 1MHz, 2MHz, 4MHz or 8MHz. Device trimming during the manufacturing process and the inclusion of internal frequency compensation circuits are used to ensure that the influence of the power supply voltage, temperature and process variations on the oscillation frequency are minimised.

Internal 32kHz Oscillator – LIRC

The Internal 32kHz System Oscillator is the low frequency oscillator. It is a fully integrated RC oscillator with a typical frequency of 32kHz at 5V, requiring no external components for its implementation. Device trimming during the manufacturing process and the inclusion of internal frequency compensation circuits are used to ensure that the influence of the power supply voltage, temperature and process variations on the oscillation frequency are minimised.

Supplementary Oscillator

The low speed oscillator, in addition to providing a system clock source is also used to provide a clock source to other device functions.

Operating Modes and System Clocks

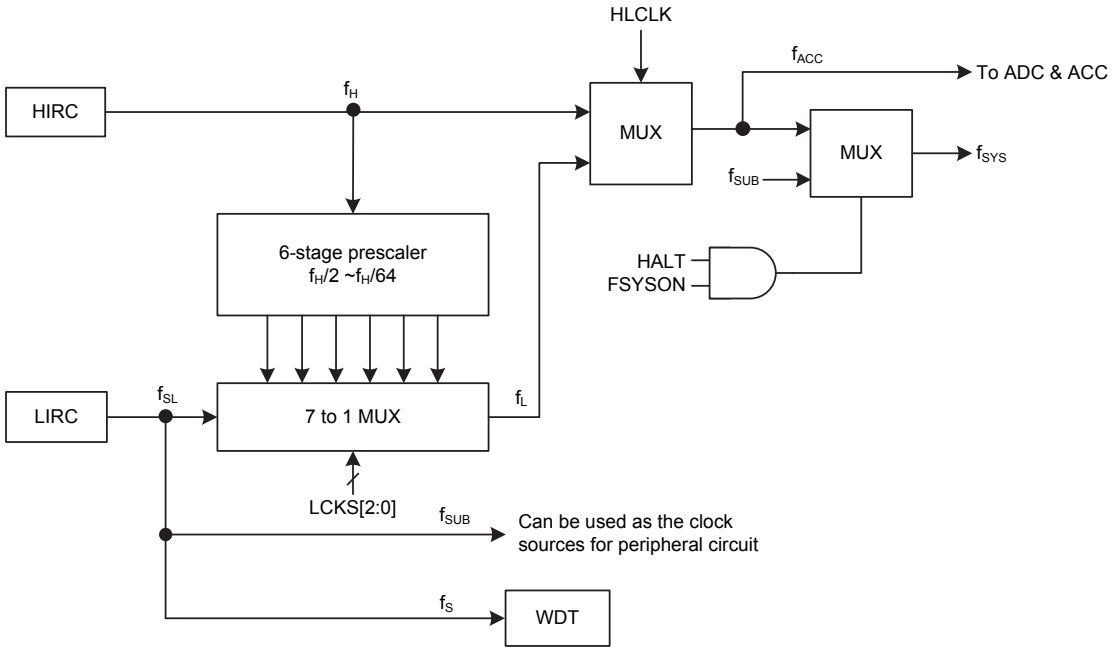
Present day applications require that their microcontrollers have high performance but often still demand that they consume as little power as possible, conflicting requirements that are especially true in battery powered portable applications. The fast clocks required for high performance will by their 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 two different clock sources for both the CPU and peripheral function operation. By providing the user with a wide range of clock options using configuration options and register programming, a clock system can be configured to obtain maximum application performance.

The main system clock, can come from either a high frequency f_H or low frequency f_L source, and is selected using the HLCLK bit and LCKS2~LCKS0 bits in the SMOD register. The high speed system clock can be sourced from HIRC oscillator. The low speed system clock source can be sourced from internal clock f_{SL} which is sourced by LIRC oscillator or 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 f_S. Each of these internal clocks is sourced by the LIRC oscillator. The f_{SUB} clock is used as the clock sources for peripheral circuit.



Note: When the system clock source f_{SYS} is switched to f_L from f_H and f_L comes from f_{SL}, the f_H will stop to conserve the power. Thus there is no $f_H/16$ or $f_H/64$ for peripheral circuit to use.

System Clock Configurations

System Operation Modes

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

Operation Mode	Description				
	CPU	f_{SYS}	f_{SUB}	f_S	f_{ACC} (Note)
NORMAL Mode	ON	f_H	ON	ON	OFF
SLOW Mode	ON	f_{SL}or $f_H/2 \sim f_H/64$	ON	ON	OFF
IDLE0 Mode	OFF	OFF	ON	ON	ON
IDLE1 Mode	OFF	ON	ON	ON	ON
SLEEP0 Mode	OFF	OFF	OFF	OFF	OFF
SLEEP1 Mode	OFF	OFF	ON	ON	ON

Note: f_{ACC} is made for PIR application. When CPU enters HALT state, this clock is supplied to ADC and ACC circuit.

NORMAL Mode

As the name suggests this is one of the main operating modes where the microcontroller has all of its functions operational and where the system clock is provided by the high speed oscillator. This mode operates allowing the microcontroller to operate normally with a clock source will come from the HIRC oscillator.

SLOW Mode

This is also a mode where the microcontroller operates normally although now with a slower speed clock source. The clock source used will be from the LIRC oscillator or a divided version of the high speed system oscillator. The high speed oscillator will however first be divided by a ratio ranging from 2 to 64, the actual ratio being selected by the LCKS2~LCKS0 and HLCLK bits in the SMOD register. Although a high speed oscillator is used, running the microcontroller at a divided clock ratio reduces the operating current.

SLEEP0 Mode

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

SLEEP1 Mode

The SLEEP Mode is entered when an HALT instruction is executed and when the IDLEN bit in the SMOD register is low. In the SLEEP1 mode the CPU will be stopped. However the f_{SUB} ,f_S and f_{ACC} clocks 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 SMOD1 register is low. In the IDLE0 Mode the system oscillator will be inhibited from driving the CPU but some peripheral functions will remain operational such as the Watchdog Timer. 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 SMOD1 register is high. In the IDLE1 Mode the system oscillator will be inhibited from driving the CPU but may continue to provide a clock source to keep some peripheral functions operational such as the Watchdog Timer. 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 Register

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

SMOD Register

Bit	7	6	5	4	3	2	1	0
Name	LCKS2	LCKS1	LCKS0	—	LTO	HTO	IDLEN	HLCLK
R/W	R/W	R/W	R/W	—	R	R	R/W	R/W
POR	0	0	0	—	0	0	1	1

Bit 7~5 **LCKS2~LCKS0**: The low frequency system clock selection when HLCLK is "0"
000: $f_L = fs_L$ (f_{LIRC})
001: $f_L = fs_L$ (f_{LIRC})
010: $f_L = f_H/64$
011: $f_L = f_H/32$
100: $f_L = f_H/16$
101: $f_L = f_H/8$
110: $f_L = f_H/4$
111: $f_L = f_H/2$
These three bits are used to select which clock is used as the low frequency system clock source. In addition to the system clock source, which is the LIRC, a divided version of the high speed system oscillator can also be chosen as the system clock source.

Bit 4 Unimplemented, read as "0"

Bit 3 **LTO**: Low speed system oscillator ready flag
0: Not ready
1: Ready
This is the low speed system oscillator ready flag which indicates when the low speed system oscillator is stable after power on reset or a wake-up has occurred. 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 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 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 IDLE1 Mode the CPU will stop running but the system clock will continue to keep the peripheral functions operational, if FSYSON bit is high. If FSYSON bit is low, the CPU and the system clock will all stop in IDLE0 mode. If the bit is low the device will enter the SLEEP Mode when a HALT instruction is executed.

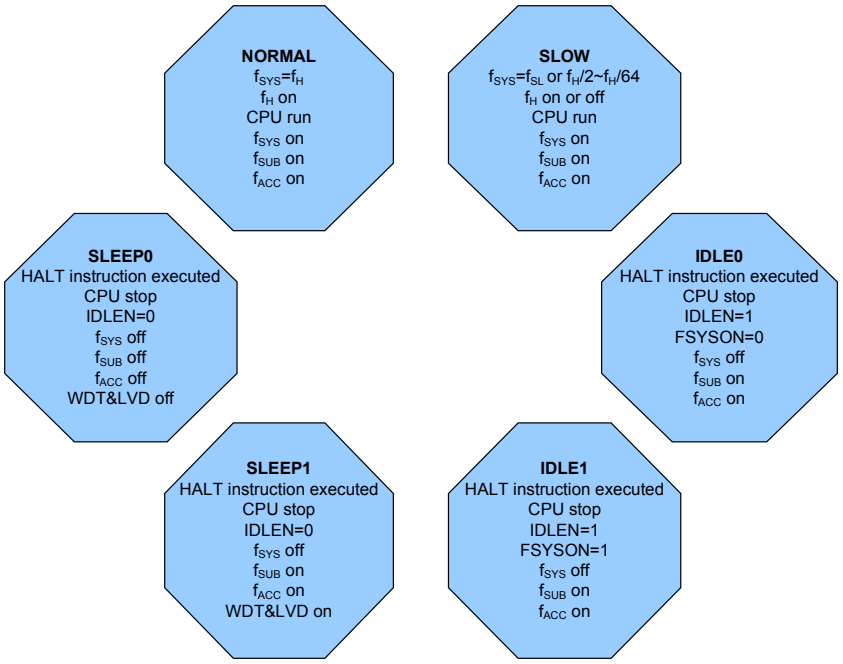
Bit 0 **HLCLK**: system clock selection
0: $f_H/2 \sim f_H/64$ or f_{SL}
1: f_H

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 NORMAL Mode and SLOW Mode is executed using the HLCLK bit and LCKS2~LCKS0 bits in the SMOD register while Mode Switching from the NORMAL/SLOW Modes to the SLEEP/IDLE Modes is executed via the HALT instruction. When a HALT instruction is executed, whether the device enters the IDLE Mode or the SLEEP Mode is determined by the condition of the IDLEN bit in the SMOD register and FSYSON in the SMOD1 register.

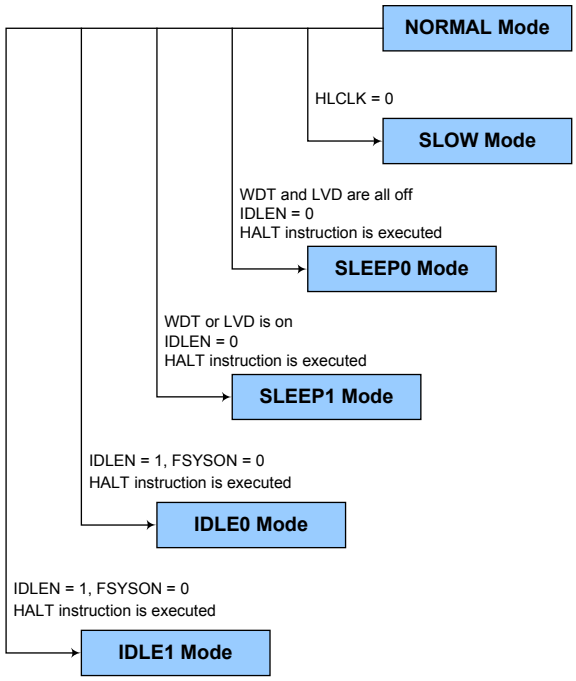
When the HLCLK bit switches to a low level, which implies that clock source is switched from the high speed clock source, f_H, to the clock source, $f_H/2 \sim f_H/64$ or f_{SL}. If the clock is from the f_{SL}, the high speed clock source will stop running to conserve power. When this happens it must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



NORMAL Mode to SLOW Mode Switching

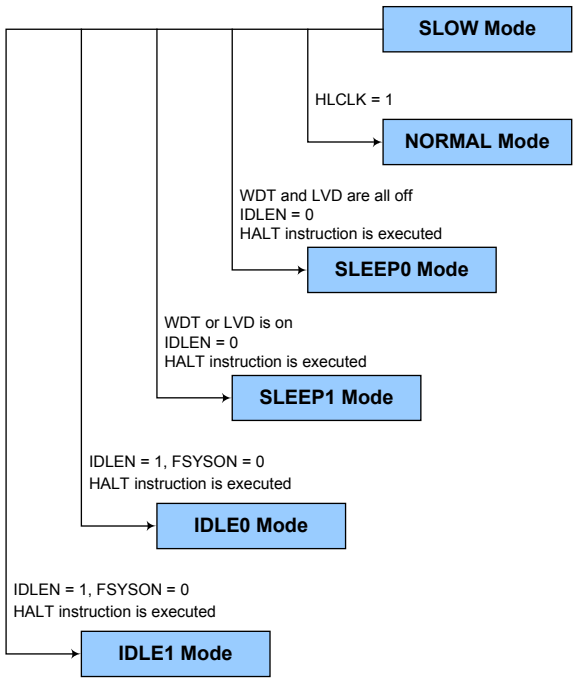
When running in the NORMAL Mode, which uses the high speed system oscillator, and therefore consumes more power, the system clock can switch to run in the SLOW Mode by set the HLCLK bit to "0". This will then use the low speed system oscillator which will consume less power. Users may decide to do this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

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



SLOW Mode to NORMAL Mode Switching

In SLOW Mode the system uses the LIRC low speed system oscillator or the HIRC high speed system oscillator. To switch back to the NORMAL Mode, the HLCLK bit should be set to "1". As a certain amount of time will be required for the high frequency clock to stabilise, the status of the HTO bit is checked.



Entering the SLEEP0 Mode

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

- The system clock, WDT clock will be stopped and the application program will stop at the "HALT" instruction.
- The Data Memory contents and registers will maintain their present condition.
- The WDT will be cleared and stopped.
- The I/O ports will maintain their present conditions.
- In the status register, the Power Down flag, PDF, will be set and the Watchdog time-out flag, TO, will be cleared.

Entering the SLEEP1 Mode

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

- The system clock 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_{SL} clock.
- The Data Memory contents and registers will maintain their present condition.
- The WDT will be cleared and resume counting if the WDT is enabled.
- The I/O ports will maintain their present conditions.
- In the status register, the Power Down flag, PDF, will be set and the Watchdog time-out flag, TO, will be cleared.

Entering the IDLE0 Mode

There is only one way for the device to enter the IDLE0 Mode and that is to execute the "HALT" instruction in the application program with the IDLEN bit in SMOD register equal to "1" and the FSYSON bit in SMOD1 register equal to "0". When this instruction is executed under the conditions described above, the following will occur:

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

Entering the IDLE1 Mode

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

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

Standby Current Considerations

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

Care must also be taken with the loads, which are connected to I/O pins, which are setup as outputs. These should be placed in a condition in which minimum current is drawn or connected only to external circuits that do not draw current, such as other CMOS inputs.

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

Wake-up

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

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

When the device executes the "HALT" instruction, 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.

Watchdog Timer

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

Watchdog Timer Clock Source

The Watchdog Timer clock source is provided by the internal clock, f_S, which is in turn supplied by the LIRC oscillator. The LIRC internal oscillator has an approximate period of 32kHz at a supply voltage of 5V. However, it should be noted that this specified internal clock period can vary with V_{DD}, temperature and process variations. The 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 software control

If the WDT configuration option is selected as "Always Enabled":
10101 or 01010: Enabled
Other: Reset MCU

If the WDT configuration option is selected as "Application Program Enabled":
10101: Disabled
01010: Enabled
Other Values: Reset MCU

When these bits are changed by the environmental noise or software setting to reset the microcontroller, the reset operation will be activated after 2~3 LIRC clock cycles and the WRF bit in the SMOD1 register will be set to 1.

Bit 2~0 **WS2~WS0**: WDT time-out period selection
000: $2^8/f_S$
001: $2^{10}/f_S$
010: $2^{12}/f_S$
011: $2^{14}/f_S$(default)
100: $2^{15}/f_S$
101: $2^{16}/f_S$
110: $2^{17}/f_S$
111: $2^{18}/f_S$

SMOD1 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 occur
1: Occurred
This bit is set to 1 by the WDT Control register software reset and cleared by the application program. Note that this bit can only be cleared to 0 by the application program.

Watchdog Timer Operation

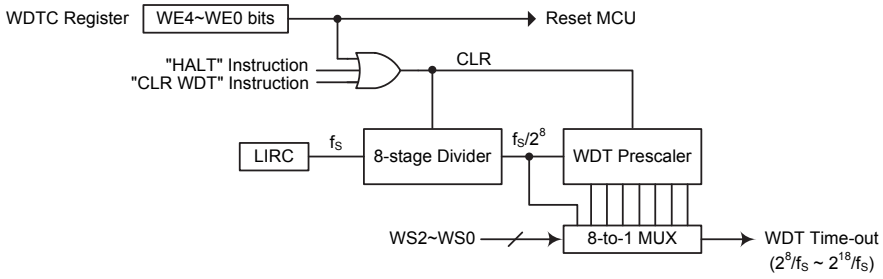
The Watchdog Timer operates by providing a device reset when its timer overflows. This means that in the application program and during normal operation the user has to strategically clear the Watchdog Timer before it overflows to prevent the Watchdog Timer from executing a reset. This is done using the clear watchdog instructions. If the program malfunctions for whatever reason, jumps to an unknown location, or enters an endless loop, these clear instructions will not be executed in the correct manner, in which case the Watchdog Timer will overflow and reset the device. With regard to the Watchdog Timer enable/disable function, there are five bits, WE4~WE0, in the WDTC register to offer additional enable/disable and reset control of the Watchdog Timer. If the WDT configuration option has selected that the WDT function is always enabled, then WE4~WE0 bits still have effect on the WDT function. When the WE4~WE0 bits value are equal to 01010B or 10101B, the WDT function is enabled. However, if the WE4~WE0 bits are changed to any other values except 01010B and 10101B, which could be caused by adverse environmental conditions such as noise, it will reset the microcontroller after 2~3 LIRC clock cycles. If the WDT configuration option has selected that the WDT function is controlled using the application program, then the WDT control register bits, WE4~WE0, are used to enable or disable the Watchdog Timer. In this case the WDT function will be disabled when the WE4~WE0 bits are equal to 10101B and enabled if the WE4~WE0 bits are equal to 01010B. If the WE4~WE0 bits are set to any other values, other than 01010B and 10101B, it will reset the device after 2~3 LIRC clock cycles. After power on these bits will have a value of 01010B.

WDT Configuration Option	WE4 ~ WE0 Bits	WDT Function
Always Enabled	01010B or 10101B	Enable
	Any other value	Reset MCU
Application Program Enabled	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 bit filed, the second is using the Watchdog Timer software clear instructions 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.



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.

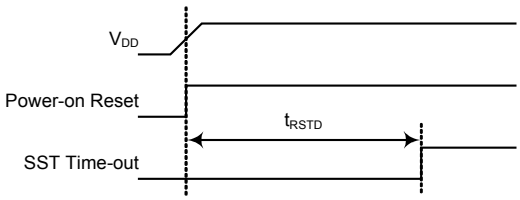
Another type of reset is when the Watchdog Timer overflows and resets. All types of reset operations result in different register conditions being setup. Another reset exists in the form of a Low Voltage Reset, LVR, where a full reset, is implemented in situations where the power supply voltage falls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

Reset Functions

There are several ways in which a reset can occur, each of which will be described as follows.

Power-on Reset

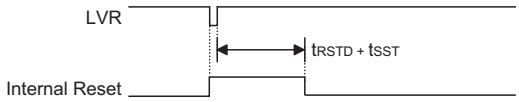
The most fundamental and unavoidable reset is the one that occurs after power is first applied to the microcontroller. As well as ensuring that the Program Memory begins execution from the first memory address, a power-on reset also ensures that certain other registers are preset to known conditions.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



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 SMOD1 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 AC Electrical Characteristics. If the low supply voltage state does not exceed this value, the LVR will ignore the low supply voltage and will not perform a reset function. The actual V_{LVR} value can be selected by the LVS bits in the LVRC register. If the LVS7~LVS0 bits are changed to some certain values by the environmental noise or software setting, the LVR will reset the device after 2~3 LIRC clock cycles. When this happens, the LRF bit in the SMOD1 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.



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	1	0	0	1	1	0

Bit 7~0 **LVS7~LVS0**: LVR voltage select
01010101: 2.1V(default)
00110011: 2.55V
10011001: 3.15V
10101010: 3.8V
Any other value: Generates MCU reset -- register is reset to POR value

When an actual low voltage condition occurs, as specified by one of the four defined LVR voltage values above, an MCU reset will be generated. In this situation the register contents will remain the same after such a reset occurs.

Any register value, other than the four defined LVR values above, will also result in the generation of an MCU reset. The reset operation will be activated after 2~3 LIRC clock cycles. However in this situation the register contents will be reset to the POR value.

- **SMOD1 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"

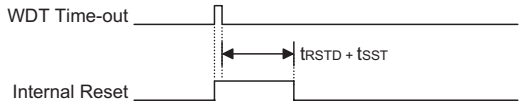
Bit 2 **LVRF**: LVR function reset flag
0: Not occur
1: Occurred
This bit is set to 1 when a specific Low Voltage Reset situation condition occurs. This bit can only be cleared to 0 by the application program.

Bit 1 **LRF**: LVR Control register software reset flag
0: Not occur
1: Occurred
This bit is set to 1 if the LVRC register contains any non-defined LVR voltage register values. This in effect acts like a software-reset function. 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

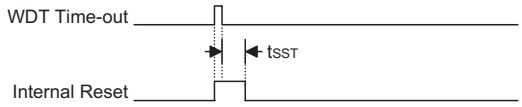
The Watchdog time-out Reset during normal operation is the same as the hardware LVR reset except that the Watchdog time-out flag TO will be set to "1".



WDT Time-out Reset during Normal Operation Timing Chart

Watchdog Time-out Reset during SLEEP or IDLE Mode

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



WDT Time-out Reset during SLEEP or IDLE Timing Chart

Reset Initial Conditions

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

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

Note: "u" stands for unchanged

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

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

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

Register	Reset (Power On)	WDT Time-out (Normal Operation)	LVR Reset	WDT Time-out (HALT)
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	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
STATUS	--00 xxxx	--1u uuuu	--uu uuuu	--11 uuuu
INTC0	-000 0000	-000 0000	-000 0000	-uuu uuuu
TMR0	0000 0000	0000 0000	0000 0000	uuuu uuuu
TMR0C	00-0 1000	00-0 1000	00-0 1000	uu-u uuuu
TMR1	0000 0000	0000 0000	0000 0000	uuuu uuuu
TMR1C	00-0 1000	00-0 1000	00-0 1000	uu-u uuuu
PA	1111 1111	1111 1111	1111 1111	uuuu uuuu

Register	Reset (Power On)	WDT Time-out (Normal Operation)	LVR Reset	WDT Time-out (HALT)
PAC	1111 1111	1111 1111	1111 1111	uuuu uuuu
PB	---- ---1	---- ---1	---- ---1	---- ---u
PBC	---- ---1	---- ---1	---- ---1	---- ---u
LVDC	0000 0000	0000 0000	0000 0000	--uu -uuu
LVRC	0101 0101	0101 0101	0101 0101	Uuuu uuuu
WDTC	0101 0011	0101 0011	0101 0011	uuuu uuuu
INTC1	0000 0000	0000 0000	0000 0000	uuuu uuuu
ADRL	0000 ----	0000 ----	0000 ----	uuuu ----
ADRH	0000 0000	0000 0000	0000 0000	uuuu uuuu
ADCR0	0110 0000	0110 0000	0110 0000	uuuu uuuu
ADCR1	1000 0000	1000 0000	1000 0000	uuuu uuuu
ACER	0--- --11	0--- --11	0--- --11	u--- --uu
PAWU	0000 0000	0000 0000	0000 0000	uuuu uuuu
PAPU	0000 0000	0000 0000	0000 0000	uuuu uuuu
PBPU	---- ---0	---- ---0	---- ---0	---- ---u
SMOD1	0--- -x00	0--- -x00	0--- -x00	u--- -uuu
SMOD	000- 0011	000- 0011	000- 0011	uuu- uuuu
INTEG	0000 0000	0000 0000	0000 0000	---- --uu
OPAC0	0000 -000	0000 -000	0000 -000	uuuu -uuu
OPAC1	--00 0000	--00 0000	--00 0000	--uu uuuu
ACCC0	000- --00	000- --00	000- --00	uuu- --uu
ACCC1	0000 0000	0000 0000	0000 0000	-uuu --uu
MFIC	0000 0000	0000 0000	0000 0000	-u-u –u-u
SIMC0	1110 000-	1110 000-	1110 000-	uuuu uuu-
SIMC1	1000 0001	1000 0001	1000 0001	uuuu uuuu
SIMD	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
SIMA	0000 0000	0000 0000	0000 0000	uuuu uuuu
SIMC2	0000 0000	0000 0000	0000 0000	uuuu uuuu
EEA	---0 0000	---0 0000	---0 0000	---u uuuu
EED	0000 0000	0000 0000	0000 0000	uuuu uuuu
PRM	---- --00	---- --00	---- --00	---- --uu
LDOC	0-00 0000	0-00 0000	0-00 0000	u-uu uuuu
TS_BG	000x xxxx	000x xxxx	000x xxxx	000x xxxx
TS_OP	0xxx xxxx	0xxx xxxx	0xxx xxxx	0xxx xxxx
LULV	0000 ----	0000 ----	0000 ----	uuuu ----
HULV	0000 0000	0000 0000	0000 0000	uuuu uuuu
LLLV	0000 ----	0000 ----	0000 ----	uuuu ----
HLLV	0000 0000	0000 0000	0000 0000	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~PB. These I/O ports are mapped to the RAM Data Memory with specific addresses as shown in the Special Purpose Data Memory table. All of these I/O ports can be used for input and output operations. For input operation, these ports are non-latching, which means the inputs must be ready at the T2 rising edge of instruction "MOV A, [m]", where m denotes the port address. For output operation, all the data is latched and remains unchanged until the output latch is rewritten.

Register Name	Bit							
	7	6	5	4	3	2	1	0
PA	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0
PAC	PAC7	PAC6	PAC5	PAC4	PAC3	PAC2	PAC1	PAC0
PAPU	PAPU7	PAPU6	PAPU5	PAPU4	PAPU3	PAPU2	PAPU1	PAPU0
PAWU	PAWU7	PAWU6	PAWU5	PAWU4	PAWU3	PAWU2	PAWU1	PAWU0
PB	—	—	—	—	—	—	—	PB0
PBC	—	—	—	—	—	—	—	PBC0
PBPU	—	—	—	—	—	—	—	PBPU0

I/O 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 registers PAPU~PBPU and are implemented using weak PMOS transistors.

PAPU Register

Bit	7	6	5	4	3	2	1	0
Name	PAPU7	PAPU6	PAPU5	PAPU4	PAPU3	PAPU2	PAPU1	PAPU0
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 **PAPU7~PAPU0**: Port A bit 7 ~ bit 0 Pull-high Control
0: Disable
1: Enable

PBPU Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	—	PBPU0
R/W	—	—	—	—	—	—	—	R/W
POR	—	—	—	—	—	—	—	0

Bit 7~1 Unimplemented, read as "0"

Bit 0 **PBPU0**: Port B bit 0 Pull-high Control
0: Disable
1: Enable

Port A Wake-up

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

PAWU Register

Bit	7	6	5	4	3	2	1	0
Name	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**: Port A bit 7 ~ bit 0 Wake-up Control
0: Disable
1: Enable

I/O Port Control Registers

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

PAC Register

Bit	7	6	5	4	3	2	1	0
Name	PAC7	PAC6	PAC5	PAC4	PAC3	PAC2	PAC1	PAC0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	1	1	1	1	1	1	1	1

Bit 7~0 **PAC7~PAC0**: Port A bit 7 ~ bit 0 Input/Output Control
0: Output
1: Input

PBC Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	—	PBC0
R/W	—	—	—	—	—	—	—	R/W
POR	—	—	—	—	—	—	—	1

Bit 7~1 Unimplemented, read as "0"

Bit 0 **PBC0**: Port B bit 0 Input/Output Control
0: Output
1: Input

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 is a PRM register to establish certain pin functions. Generally speaking, the analog function has higher priority than the digital function. However, if more than two analog functions are enabled and the analog signal input comes from the same external pin, the analog input will be internally connected to all of these active analog functional modules.

Pin-remapping Registers

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

- **PRM Register**

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

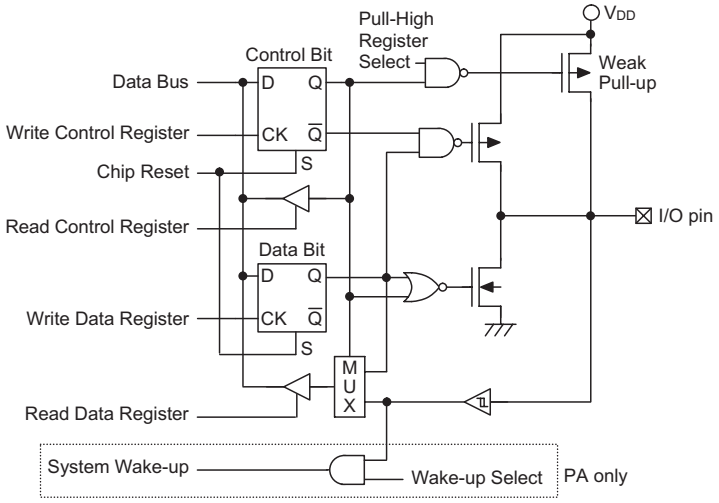
Bit 7~2 Unimplemented, read as "0"

Bit 1 **SCKPS**: SCK Pin Remapping Control
0 : Enable SCK_0 on PA0
1 : Enable SCK_1 on PA6

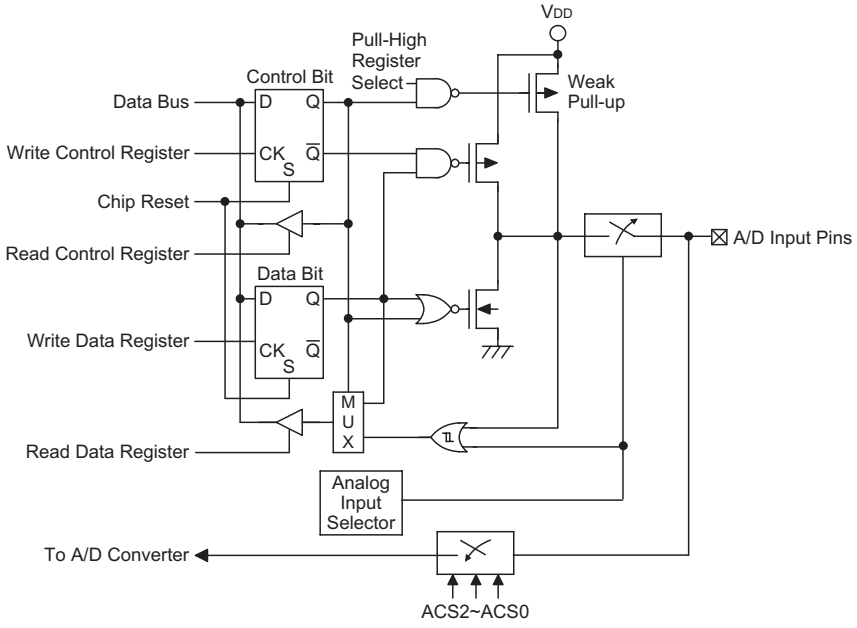
Bit 0 **SCLPS**: SCL Pin Remapping Control
0 : Enable SCL_0 on PA0
1 : Enable SCL_1 on PA6

I/O Pin Structures

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



Generic Input/Output Structure



A/D Input/Output Structure

Programming Considerations

Within the user program, one of the first things to consider is port initialisation. After a reset, all of the I/O data and port control registers will be set high. This means that all I/O pins will default to an input state, the level of which depends on the other connected circuitry and whether pull-high selections have been chosen. If the port control registers, PAC~PBC, are then programmed to setup some pins as outputs, these output pins will have an initial high output value unless the associated port data registers, PA~PB, 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.

The power-on reset condition of the A/D converter control registers ensures that any A/D input pins - which are always shared with other I/O functions - will be setup as analog inputs after a reset. Although these pins will be configured as A/D inputs after a reset, the A/D converter will not be switched on. It is therefore important to note that if it is required to use these pins as I/O digital input pins or as other functions, the A/D converter control registers must be correctly programmed to remove the A/D function. Note also that as the A/D channel is enabled, any internal pull-high resistor connections will be removed.

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.

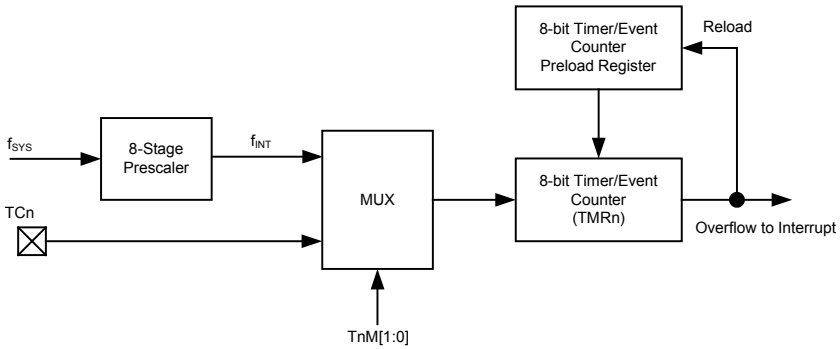
Timer/Event Counters

The provision of timers form an important part of any microcontroller, giving the designer a means of carrying out time related functions. The device contains two count-up timer of 8-bit capacity. As the timers have three different operating modes, they can be configured to operate as a general timer, an external event counter or as a pulse width capture device. The provision of an internal prescaler to the clock circuitry on giving added range to the timers.

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

Configuring the Timer/Event Counter Input Clock Source

The Timer/Event Counter clock source can originate from various sources, an internal clock f_{SYS} or an external pin. The internal clock source is used when the timer is in the timer mode or in the pulse width capture mode, this internal clock source is firstly divided by a prescaler, the division ratio of which is conditioned by the Timer Control Register bits TnPSC2~TnPSC0. An external clock source is used when the timer is in the event counting mode, the clock source being provided on an external timer pin TMRn. Depending upon the condition of the TnE bit, each high to low, or low to high transition on the external timer pin will increment the counter by one.



Timer/Event Counter (n=0 or 1)

Timer Registers – TMR0, TMR1

The timer registers are special function registers located in the Special Purpose Data Memory and is the place where the actual timer value is stored. These registers are known as TMR0 and TMR1. The value in the timer registers increases by one each time an internal clock pulse is received or an external transition occurs on the external timer pin. The timer will count from the initial value loaded by the preload register to the full count of FFH at which point the timer overflows and an internal interrupt signal is generated. Then the timer value will be reset with the initial preload register value and continue counting. Note that to achieve a maximum full range count of FFH, all the preload registers must first be cleared to zero. It should be noted that after power-on, the preload registers will be in an unknown condition. Note that if the Timer/Event Counter is in an OFF condition and data is written to its preload register, this data will be immediately written into the actual counter. However, if the counter is enabled and counting, any new data written into the preload data register during this period will remain in the preload register and will only be written into the actual counter the next time an overflow occurs.

TMRn Register (n=0 or 1)

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

Bit 7~0 **D7~D0**: Timer n values

Timer Control Registers – TMR0C, TMR1C

The flexible features of the Holtek microcontroller Timer/Event Counters enable them to operate in three different modes, the options of which are determined by the contents of their respective control register. The Timer Control Register is known as TMRnC. It is the Timer Control Register together with its corresponding timer registers that control the full operation of the Timer/Event Counter. Before the timer can be used, it is essential that the Timer Control Register is fully programmed with the right data to ensure its correct operation, a process that is normally carried out during program initialisation. To choose which of the three modes the timer is to operate in, either in the timer mode, the event counting mode or the pulse width capture mode, bits 7 and 6 of the Timer Control Register, which are known as the bit pair TnM1/TnM0, must be set to the required logic levels. The timer-on bit, which is bit 4 of the Timer Control Register and known as TnON, provides the basic on/off control of the respective timer. Setting the bit high allows the counter to run. Clearing the bit stops the counter. Bits 0~2 of the Timer Control Register determine the division ratio of the input clock prescaler. The prescaler bit settings have no effect if an external clock source is used. If the timer is in the event count or pulse width capture mode, the active transition edge level type is selected by the logic level of bit 3 of the Timer Control Register which is known as TnE.

TMRnC Register (n=0 or 1)

Bit	7	6	5	4	3	2	1	0
Name	TnM1	TnM0	—	TnON	TnE	TnPSC2	TnPSC1	TnPSC0
R/W	R/W	R/W	—	R/W	R/W	R/W	R/W	R/W
POR	0	0	—	0	1	0	0	0

Bit 7~6 **TnM1, TnM0**: Timer n operation mode selection
00: No mode available
01: Event counter mode(External clock)
10: Timer mode(Internal clock)
11: Pulse width capture mode(External clock)

Bit 5 Unimplemented, read as "0"

Bit 4 **TnON**: Timer/event counter counting enable
0: Disable
1: Enable

Bit 3 **TnE**: Event counter active edge selection
In Event Counter Mode (TnM1,TnM0)=(0,1):
1: Count on falling edge;
0: Count on rising edge
In Pulse Width measurement mode (TnM1,TnM0)=(1,1):
1: Start counting on the rising edge, stop on the falling edge;
0: Start counting on the falling edge, stop on the rising edge

Bit 2~0 **TnPSC2~TnPSC0**: Timer prescaler rate selection
000: $f_{INT} = f_{SYS}$
001: $f_{INT} = f_{SYS}/2$
010: $f_{INT} = f_{SYS}/4$
011: $f_{INT} = f_{SYS}/8$
100: $f_{INT} = f_{SYS}/16$
101: $f_{INT} = f_{SYS}/32$
110: $f_{INT} = f_{SYS}/64$
111: $f_{INT} = f_{SYS}/128$

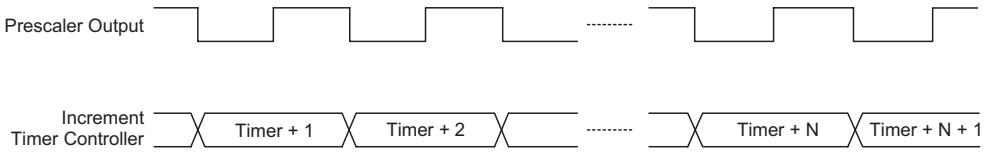
Timer Mode

In this mode, the Timer/Event Counter can be utilised to measure fixed time intervals, providing an internal interrupt signal each time the Timer/Event Counter overflows. To operate in this mode, the Operating Mode Select bit pair, TnM1/TnM0, in the Timer Control Register must be set to the correct value as shown.

Control Register Operating Mode Select Bits for the Timer Mode

Bit7	Bit6
1	0

In this mode the internal clock is used as the timer clock. The timer input clock source is f_{SYS}. However, this timer clock source is further divided by a prescaler, the value of which is determined by the bits TnPSC2~TnPSC0 in the Timer Control Register. The timer-on bit, TnON must be set high to enable the timer to run. Each time an internal clock high to low transition occurs, the timer increments by one. When the timer is full and overflows, an interrupt sigal is generated and the timer will reload the value already loaded into the preload register and continue counting. A timer overflow condition and corresponding internal interrupts are two of the wake-up sources. However, the internal interrupts can be disabled by ensuring that the Timer/Event Counter Interrupt Enable bits in the Interrupt control registers are reset to zero.



Timer Mode Timing Chart

Event Counter Mode

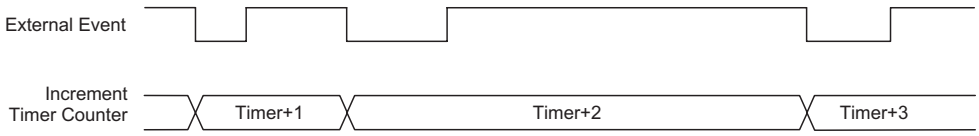
In this mode, a number of externally changing logic events, occurring on the external timer TMRn pin, can be recorded by the Timer/Event Counter. To operate in this mode, the Operating Mode Select bit pair, TnM1/TnM0, in the Timer Control Register must be set to the correct value as shown.

Control Register Operating Mode Select Bits for the Event Counter Mode.

Bit7	Bit6
0	1

In this mode, the external timer pin TMRn, is used as the Timer/Event Counter clock source, however it is not divided by the internal prescaler. After the other bits in the Timer Control Register have been setup, the enable bit TnON, which is bit 4 of the Timer Control Register, can be set high to enable the Timer/Event Counter to run. If the Active Edge Select bit, TnE, which is bit 3 of the Timer Control Register, is low, the Timer/Event Counter will increment each time the external timer pin receives a low to high transition. If the TnE is high, the counter will increment each time the external timer pin receives a high to low transition. When it is full and overflows, an interrupt signal is generated and the Timer/Event Counter will reload the value already loaded into the preload register and continue counting. The interrupt can be disabled by ensuring that the Timer/Event Counter Interrupt Enable bit in the corresponding Interrupt Control Register. It is reset to zero. As the external timer pin is shared with an I/O pin, to ensure that the pin is configured to operate as an event counter input pin, two things have to happen. The first is to ensure that the Operating Mode Select bits in the Timer Control Register place the Timer/Event Counter in the Event Counting Mode. The second is to ensure that the port control register configures the pin as an input. It should

be noted that in the event counting mode, even if the microcontroller is in the Idle/Sleep Mode, the Timer/Event Counter will continue to record externally changing logic events on the timer input TMRn pin. As a result when the timer overflows it will generate a timer interrupt and corresponding wake-up source.



Event Counter Mode Timing Chart (TnE=1)

Pulse Width Capture Mode

In this mode, the Timer/Event Counter can be utilised to measure the width of external pulses applied to the external timer pin. To operate in this mode, the Operating Mode Select bit pair, TnM1/TnM0, in the Timer Control Register must be set to the correct value as shown.

Control Register Operating Mode Select Bits for the Pulse Width Measurement Mode.

Bit7	Bit6
1	1

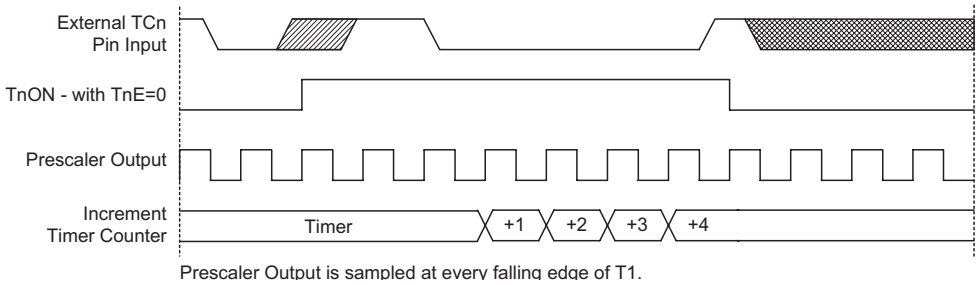
In this mode the internal clock, f_{SYS}, is used as the internal clock for the 8-bit Timer/Event Counter. However, it is further divided by a prescaler, the value of which is determined by the Prescaler Rate Select bits TnPSC2~TnPSC0, which TnON, which is bit 2~0 of the Timer Control Register, can be set high to enable the Timer/Event Counter, however it will not actually start counting until an active edge is received on the external timer pin.

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

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

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

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



Pulse Width Capture Mode Timing Chart (TnE=0)

I/O Interfacing

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

Programming Considerations

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

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

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

Analog to Digital Converter – ADC

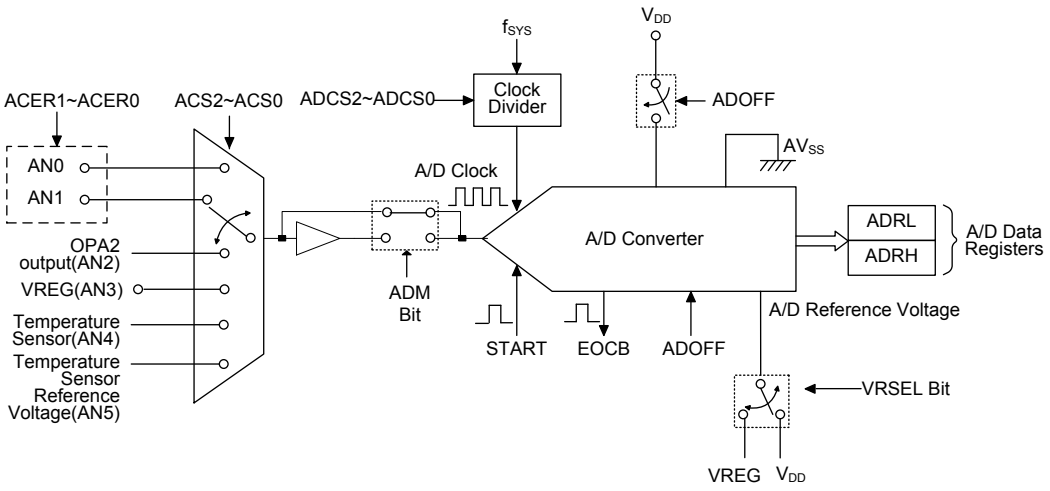
The need to interface to real world analog signals is a common requirement for many electronic systems. However, to properly process these signals by a microcontroller, they must first be converted into digital signals by A/D converters. By integrating the A/D conversion electronic circuitry into the 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.

Input Channels	A/D Channel Select Bits	Input Pins
2+4	ACS2~ACS0	AN0~AN5

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



A/D Converter Structure

A/D Converter Register Description

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

Register Name	Bit							
	7	6	5	4	3	2	1	0
ADCR0	START	EOCB	ADOFF	ADM	BYS	ACS2	ACS1	ACS0
ADCR1	TEST	TSGX	SMP_CK	VRSEL	OPA2V	ADCS2	ADCS1	ADCS0
ACER	TSE	—	—	—	—	—	ACER1	ACER0
ADRL	D3	D2	D1	D0	—	—	—	—
ADRH	D11	D10	D9	D8	D7	D6	D5	D4

A/D Converter Register List

A/D Converter Data Registers – ADRL, ADRH

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 ADRH, and a low byte register, known as ADRL. After the conversion process takes place, these registers can be directly read by the microcontroller to obtain the digitised conversion value. D0~D11 are the A/D conversion result data bits. Any unused bits will be read as zero. Note that the A/D converter data register contents will be cleared to zero if the A/D converter is disabled.

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

A/D Data Registers

ADRL Register

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

Bit 7~4 Lower byte of ADC conversion data

Bit 3~0 Unimplemented, read as "0"

ADRH Register

Bit	7	6	5	4	3	2	1	0
Name	D11	D10	D9	D8	D7	D6	D5	D4
R/W	R	R	R	R	R	R	R	R
POR	0	0	0	0	0	0	0	0

Bit 7~0 Higher byte of ADC conversion data

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

To control the function and operation of the A/D converter, several control registers known as ADCR0, ADCR1 and ACER are provided. These 8-bit registers define functions such as the selection of which analog channel is connected to the internal A/D converter, the digitised data format, the A/D clock source as well as controlling the start function and monitoring the A/D converter 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 ACS2~ACS0 bits in the ADCR0 register are used to determine which external channel input is selected to be converted. As the device contain only one actual analog to digital converter hardware circuit, each of the individual analog inputs, which include external A/D channels and internal outputs, must be routed to the converter. It is the function of the ACS2~ACS0 bits to determine which analog channel input pin is actually connected to the internal A/D converter.

The ACER control register contains the ACER1~ACER0 bits which determine which pins on I/O Port are used as analog inputs for the A/D converter input and which pins are not to be used as the A/D converter input. Setting the corresponding bit high will select the A/D input function, clearing the bit to zero will select either the I/O or other pin-shared function. When the pin is selected to be an A/D input, its original function whether it is an I/O or other pin-shared function will be removed. In addition, any internal pull-high resistors connected to these pins will be automatically removed if the pin is selected to be an A/D input.

ADCR0 Register

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

Bit 7 **START**: Start the A/D conversion
0→1→0: Start A/D conversion
0→1: Reset the A/D converter and set EOCB to 1
This bit is used to initiate an A/D conversion process. The bit is normally low but if set high and then cleared low again, the A/D converter will initiate a conversion process. When the bit is set high the A/D converter will be reset.

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

Bit 5 **ADOFF**: ADC module on/off control bit
0: ADC module is enabled
1: ADC module is disabled
This bit controls the A/D internal function. This bit should be cleared to zero to enable the A/D converter. If the bit is set high, then the A/D converter will be switched off reducing the device power consumption. When the A/D converter function is disabled, the contents of the A/D data register pair, ADRH and ADRL, will be cleared to zero.

Bit 4 **ADM**: A/D Converter mode selection
0: Normal mode (analog input bypass pre-buffer, direct to ADC)
1: High drive mode (analog input through pre-buffer to ADC)
When ADM=1 and ADOFF=0, the analog signal will be through buffer to A/D convertor.

Bit 3 **BYS**: Channel 2 signal bypass selection
0 : Disable
1 : Enable
When this bit is enabled, the channel 2 signal will bypass to pin AN0.

Bit 2~0 **ACS2~ACS0**: A/D converter input channel selection
000: AN0
001: AN1
010: AN2(From OPA2 output)
011: AN3(VREG)
100: AN4(Temperature sensor)
101: AN5(Temperature sensor reference voltage)
others: Undefiend

Note: Temperature sensor reference voltage is for manufacture test, if it is selected, this signal will also output to PB0, therefore, PB0 must be defined as floating,.

ADCR1 Register

Bit	7	6	5	4	3	2	1	0
Name	TEST	TSGX	SMP_CK	VRSEL	OPA2V	ADCS2	ADCS1	ADCS0
R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W
POR	1	0	0	0	0	0	0	0

Bit 7 **TEST**: For test only, read always as "1"

Bit 6 **TSGX**: A/D converter pre-buffer gain control
ACS2~ACS0 = 100
0: 2X
1: 4X
ACS2~ACS0 = other values
0: 1X
1: 1X

Bit 5 **SMP_CK**: A/D conversion sample clock width adjustment
0: 4 clock
1: 2 clock

Bit 4 **VRSEL**: A/D converter reference voltage selection
0: VDD
1: VREG

Bit 3 **OPA2V**: OPA2 power voltage selection
0: VDD
1: LDO

Bit 2~0 **ADCS2~ADCS0**: A/D converter clock source selection
000: $f_{SYS}/2$
001: $f_{SYS}/8$
010: $f_{SYS}/32$
011: Undefined
100: f_{SYS}
101: $f_{SYS}/4$
110: $f_{SYS}/16$
111: Undefined

ACER Register

Bit	7	6	5	4	3	2	1	0
Name	TSE	—	—	—	—	—	ACER1	ACER0
R/W	R/W	—	—	—	—	—	R/W	R/W
POR	0	—	—	—	—	—	1	1

Bit 7 **TSE**: Temperature sensor enable/disable control
0: Disable
1: Enable

Bit 6~2 Unimplemented, read as "0"

Bit 1 **ACER1**: PA1 function selection
0: I/O(PA1)
1: A/D input, AN1

Bit 0 **ACER0**: PB0 function selection
0: I/O(PB0)
1: A/D input, AN0

Temperature Sensor Band-Gap Voltage Adjust Register

The temperature sensor band-gap output can be selected as an A/D converter input signal. During power on these values will be downloaded from the option table.

TS_BG Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	TS_BG4	TS_BG3	TS_BG2	TS_BG1	TS_BG0
R/W	—	—	—	R/W	R/W	R/W	R/W	R/W
POR	—	—	—	x	x	x	x	x

Bit 7~5 Unimplemented, read as "0"

Bit 4~0 Temperature sensor trimming setup value.
(if the adjust temperature sensor band-gap output voltage is 1.04V, adjust temperature sensor output voltage to 0.91V at 25°C and bypass ADC pre-buffer)

A/D Operation

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

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

The clock source for the A/D converter, which originates from the system clock f_{SYS}, can be chosen to be either f_{SYS} or a subdivided version of f_{SYS}. The division ratio value is determined by the ADCS2~ADCS0 bits in the ADCR1 register. Although the A/D clock source is determined by the system clock f_{SYS}, and by bits ADCS2~ADCS0, there are some limitations on the maximum A/D clock source speed that can be selected. As the recommended value of permissible A/D clock period, t_{AD}, is from 0.5μs to 10μs, care must be taken for system clock frequencies. For example, if the system clock operates at a frequency of 4MHz, the ADCS2~ADCS0 bits should not be set to "100". Doing so will give A/D clock periods that are less than the minimum A/D clock period or greater than the maximum A/D clock period which may result in inaccurate A/D conversion values. Refer to the following table for examples, where values marked with an asterisk * show where, depending upon the device, special care must be taken, as the values may be less than the specified minimum A/D Clock Period.

f_{SYS}	A/D Clock Period (t_{AD})						
	ADCS2, ADCS1, ADCS0 =100 (f_{SYS})	ADCS2, ADCS1, ADCS0 =000 (f_{SYS}/2)	ADCS2, ADCS1, ADCS0 =101 (f_{SYS}/4)	ADCS2, ADCS1, ADCS0 =001 (f_{SYS}/8)	ADCS2, ADCS1, ADCS0 =110 (f_{SYS}/16)	ADCS2, ADCS1, ADCS0 =010 (f_{SYS}/32)	ADCS2, ADCS1, ADCS0 =011, 111
1MHz	1μs	2μs	4μs	8μs	16μs*	32μs*	Undefined
2MHz	500ns	1μs	2μs	4μs	8μs	16μs*	Undefined
4MHz	250ns*	500ns	1μs	2μs	4μs	8μs	Undefined
8MHz	125ns*	250ns*	500ns	1μs	2μs	4μs	Undefined

A/D Clock Period Examples

Controlling the power on/off function of the A/D converter circuitry is implemented using the ADOFF bit in the ADCR0 register. This bit must be set high to power on the A/D converter. When the ADOFF bit is set high to power on the A/D converter internal circuitry a certain delay, as indicated in the timing diagram, must be allowed before an A/D conversion is initiated. Even if no pins are selected for use as A/D inputs by configuring the corresponding pin control bits, if the ADOFF bit is high then some power will still be consumed. In power conscious applications it is therefore recommended that the ADOFF is set low to reduce power consumption when the A/D converter function is not being used.

A/D Reference Voltage

The reference voltage supply to the A/D Converter can be supplied from the positive power supply pin, VDD or an external pin VREG. The desired selection is made using the VRSEL bit in the ADCR1 register

A/D Converter Input Signal

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

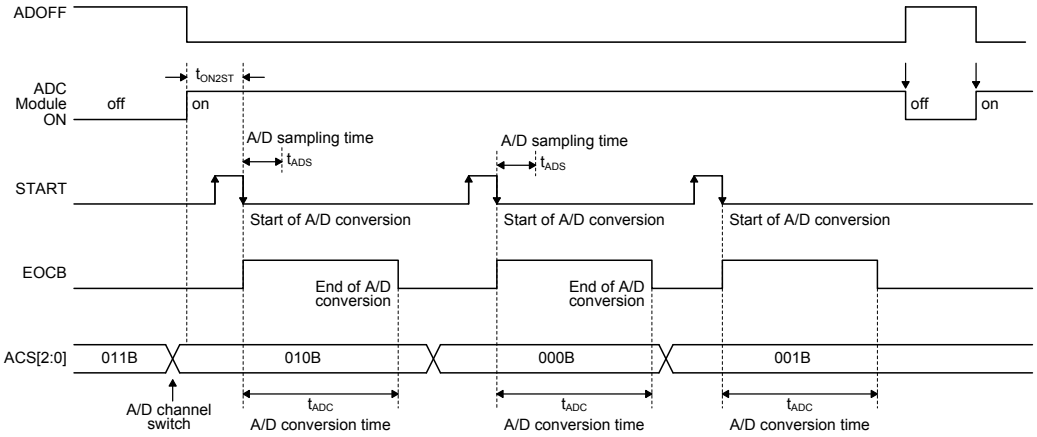
The A/D converter the reference voltage can be supplied from the power supply pin or VREG pin, a choice which is made through the VRSEL bit in the ADCR1 register. The analog input values must not be allowed to exceed the value of V_{REF}.

Conversion Rate and Timing Diagram

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

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

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



A/D Conversion Timing

Summary of A/D Conversion Steps

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

- Step 1
  Select the required A/D conversion clock by correctly programming bits ADCS2~ADCS0 in the ADCR1 register.
- Step 2
  Enable the A/D by clearing the ADOFF bit in the ADCR0 register to zero.
- Step 3
  Select which channel is to be connected to the internal A/D converter by correctly programming the ACS2~ACS0 bits which are also contained in the ADCR0 register
- Step 4
  Select which pins are to be used as A/D inputs and configure them by correctly programming the ACER1~ACER0 bits in the ACER register.
- Step 5
  If the interrupts are to be used, the interrupt control registers must be correctly configured to ensure the A/D converter interrupt function is active. The master interrupt control bit, EMI, Mulit-function interrupt bit, and the A/D converter interrupt bit, ADE, must all be set high to do this.

- Step 6
  The analog to digital conversion process can now be initialised by setting the START bit in the ADCR0 register from low to high and then low again. Note that this bit should have been originally cleared to zero.
- Step 7
  To check when the analog to digital conversion process is complete, the EOCB bit in the ADCR0 register can be polled. The conversion process is complete when this bit goes low. When this occurs the A/D data registers ADRL and ADRH can be read to obtain the conversion value. As an alternative method, if the interrupts are enabled and the stack is not full, the program can wait for an A/D interrupt to occur.

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

Programming Considerations

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

A/D Transfer Function

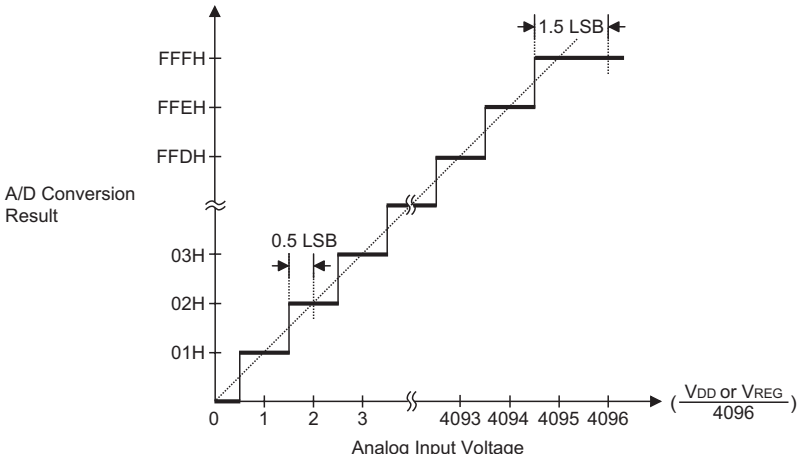
As the device contains a 12-bit A/D converter, its full-scale converted digitised value is equal to FFFH. Since the full-scale analog input value is equal to the V_{DD} or V_{REG} voltage, this gives a single bit analog input value of V_{DD} divided by 4096.

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

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

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

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



Ideal A/D Transfer Function

A/D Programming Examples

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

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

```
clr  ADE              ; disable ADC interrupt
mov  a,01H
mov  ADCR1,a          ; select fSYS/8 as A/D clock and select VDD as ADC reference voltage
set  ADOFF
mov  a,01h            ; setup ACER to configure pin AN0
mov  ACER,a
mov  a,00h
mov  ADCR0,a          ; enable and connect AN0 channel to A/D converter
:
start_conversion:
clr  START            ; high pulse on start bit to initiate conversion
set  START            ; reset A/D
clr  START            ; start A/D
polling_EOC:
sz   EOCB             ; poll the ADCR0 register EOCB bit to detect end of A/D conversion
jmp  polling_EOC      ; continue polling
mov  a,ADRL           ; read low byte conversion result value
mov  ADRL_buffer,a    ; save result to user defined register
mov  a,ADRH           ; read high byte conversion result value
mov  ADRH_buffer,a    ; save result to user defined register
:
jmp  start_conversion ; start next A/D conversion
```

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

```
clr  ADE           ; disable ADC interrupt
mov  a,01H
mov  ADCR1,a       ; select fSYS/8 as A/D clock and select VDD as ADC reference voltage
set  ADOFF
mov  a,01h         ; setup ACER to configure pin AN0
mov  ACER,a
mov  a,00h
mov  ADCR0,a       ; enable and connect AN0 channel to A/D converter
Start_conversion:
clr  START         ; high pulse on START bit to initiate conversion
set  START         ; reset A/D
clr  START         ; start A/D
clr  ADF           ; clear ADC interrupt request flag
set  ADE           ; enable ADC interrupt
set  MFE           ; enable Multi-function interrupt
set  EMI           ; enable global interrupt
:
:
; ADC interrupt service routine
ADC_ISR:
mov  acc_stack,a   ; save ACC to user defined memory
mov  a,STATUS
mov  status_stack,a; save STATUS to user defined memory
:
:
mov  a,ADRL        ; read low byte conversion result value
mov  ADRL_buffer,a ; save result to user defined register
mov  a,ADRH        ; read high byte conversion result value
mov  ADRH_buffer,a ; save result to user defined register
:
:
EXIT_INT_ISR:
mov  a,status_stack
mov  STATUS,a      ; restore STATUS from user defined memory
mov  a,acc_stack   ; restore ACC from user defined memory
reti
```

Auto Conversion Function

The device contains an auto conversion circuit function which is used to enable automatic A/D conversions. This function can be implemented using the WDT counter.

Auto Conversion Operation

The Auto conversion circuit allows the WDT counter to enable the ADC and to compare the ADC conversion value with pre-programmed upper and lower limit values while in the power down state.

The auto A/D conversion time is selected by the user controlled bits, ACCT1~ACCT0, in the ACCC0 register. When the WDT counts up to this period time, it will send a signal to the ACC circuit, to enable the ADC and automatically start a conversion. If the ADC data lies outside the preset lower and upper limit values, the event counter value ECNT2~ECNT0 will be incremented by one. When the ECNT2~ECNT0 value is equal to the number of events as set by bits NOE1~NOE0, the ACC will send an interrupt signal to CPU and the hardware will clear the ECNT2~ECNT0 bits. The CPU will then wake up and execute the ACC interrupt subroutine.

Auto Conversion Registers

To control the operation of the Auto Conversion function, several control registers known as ACCC0, ACCC1, LULV, HULV, LLLV and HLLV are provided. The ACCC0 register is used to enable or disable the auto conversion function and set the auto conversion time. When the WDT counts up to a preset time, the hardware circuit will automatically enable the ADC function. The ACCC1 register is the event counter value register. The remaining four data registers LULV, HULV, LLLV and HLLV are used to store the lower and upper limit values.

ACCC0 Register

Bit	7	6	5	4	3	2	1	0
Name	ACCEN	FH2AD	OP2AD	—	—	—	ACCT1	ACCT0
R/W	R/W	R/W	R/W	—	—	—	R/W	R/W
POR	0	0	0	—	—	—	0	0

Bit 7 **ACCEN**: Auto conversion circuit control bit
0: Disable
1: Enable

Bit 6 **FH2AD**: FH and A/D turn-on timing control
0: Enable
1: Disable

Bit 5 **OP2AD**: OP and A/D turn-on timing control
0: Enable
1: Disable
If FH2AD and OP2AD enable, can improve the performance of Auto Conversion Function.

Bit 4~2 Unimplemented, read as "0"

Bit 1~0 **ACCT1~ACCT0**: Auto A/D conversion time selection
00: 4ms
01: 8ms
10: 16ms
11: 32ms
If the WDT counter value is equal to this period, the hardware circuit will enable the ADC function automatically.

ACCC1 Register

Bit	7	6	5	4	3	2	1	0
Name	—	ECNT2	ECNT1	ECNT0	—	—	NOE1	NOE0
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 6~4 **ECNT2~ ECNT0**: Event counter values
If the ADC data lies outside the lower limitation value and upper limitation value, the event counter values ECNT2~ECNT0 will be incremented by one. When ECNT2~ECNT0 is equal to the number of events values as set by bits NOE1~NOE0, the ACC will send an interrupt signal to the CPU and the hardware will clear the ECNT2~ECNT0 bits.

Bit 3~2 Unimplemented, read as "0"

Bit 1~0 **NOE1~NOE0**: The number of events
00: 1 time
01: 2 times
10: 4 times
11: 7 times

LULV Register

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

Bit 7~4 Lower byte of upper limitation value

Bit 3~0 Unimplemented, read as "0"

HULV Register

Bit	7	6	5	4	3	2	1	0
Name	D11	D10	D9	D8	D7	D6	D5	D4
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 Higher byte of upper limitation value

LLLV Register

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

Bit 7~4 Lower byte of Lower limitation value

Bit 3~0 Unimplemented, read as "0"

HLLV Register

Bit	7	6	5	4	3	2	1	0
Name	D11	D10	D9	D8	D7	D6	D5	D4
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 Higher byte of lower limitation value

Serial Interface Module – SIM

The device contains a Serial Interface Module respectively, which includes both the four line SPI interface or the two line I²C interface types, to allow an easy method of communication with external peripheral hardware. Having relatively simple communication protocols, these serial interface types allow the microcontroller to interface to external SPI or I²C based hardware such as sensors, Flash or EEPROM memory, etc. The SIM interface pins are pin-shared with other I/O pins therefore the SIM interface function must first be selected by setting the SIM enable/disable bit. As both interface types share the same pins and registers, the choice of whether the SPI or I²C type is used is made using the SIM operating mode control bits, named SIM2~SIM0, in the SIMC0 register. These pull-high resistors of the SIM pin-shared I/O are selected using pull-high control registers when the SIM function is enabled.

It is suggested that the user shall not enter the device to HALT status by application program during processing SIM communication.

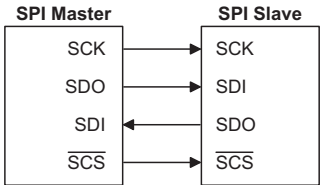
SPI Interface

The SPI interface is often used to communicate with external peripheral device such as sensors, Flash or EEPROM memory device 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 device.

The communication is full duplex and operates as a slave/master type, where the devices can be either master or slave. Although the SPI interface specification can control multiple slave devices from a single master, these devices provide 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 I²C function pins, the SPI interface must first be enabled by setting the correct bits in the SIMC0 and SIMC2 registers. After the desired SPI configuration has been set it can be disabled or enabled using the SIMEN bit in the SIMC0 register. Communication between devices connected to the SPI interface is carried out in a slave/master mode with all data transfer initiations being implemented by the master. The Master also controls the clock signal. As the device only contains a single $\overline{SCS}$ pin only one slave device can be utilized. The $\overline{SCS}$ pin is controlled by software, set CSEN bit to "1" to enable $\overline{SCS}$ pin function, clear CSEN bit, the $\overline{SCS}$ pin will be floating state.

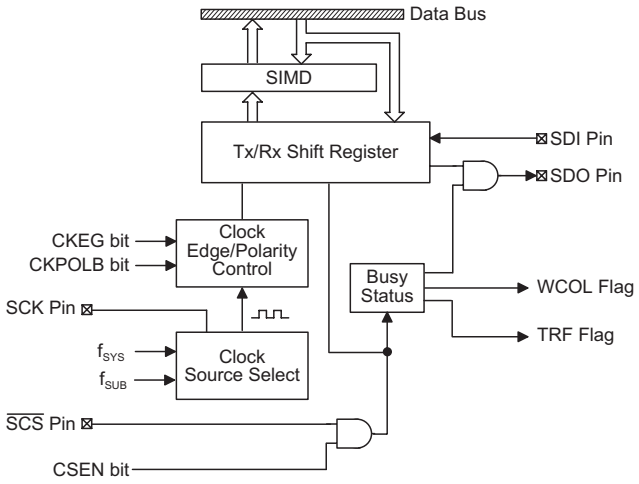


SPI Master/Slave Connection

The SPI function in this device offers the following features:

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

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



SPI Block Diagram

SPI Registers

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

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

SPI Registers List

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.

- **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	—	—	—	SIMEN	—
R/W	R/W	R/W	R/W	—	—	—	R/W	—
POR	1	1	1	—	—	—	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_{SUB}
101: SPI slave mode
110: I²C slave mode
others: Reserved

Bit 4~2 Unimplemented, read as "0"

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

Bit 0 Unimplemented, read as "0"

- **SIMC2 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	CKPOLB	CKEG	MLS	CSEN	WCOL	TRF
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 **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
1: MSB
This is the data shift select bit and is used to select how the data is transferred, either MSB or LSB first. Setting the bit high will select MSB first and low for LSB first.

Bit 2 **CSEN**: SPI $\overline{SCS}$ pin Control
0: Disable
1: Enable
The CSEN bit is used as an enable/disable for the $\overline{SCS}$ pin. If this bit is low then the $\overline{SCS}$ pin will be disabled and placed into a floating condition. If the bit is high the $\overline{SCS}$ pin will be enabled and used as a select pin.

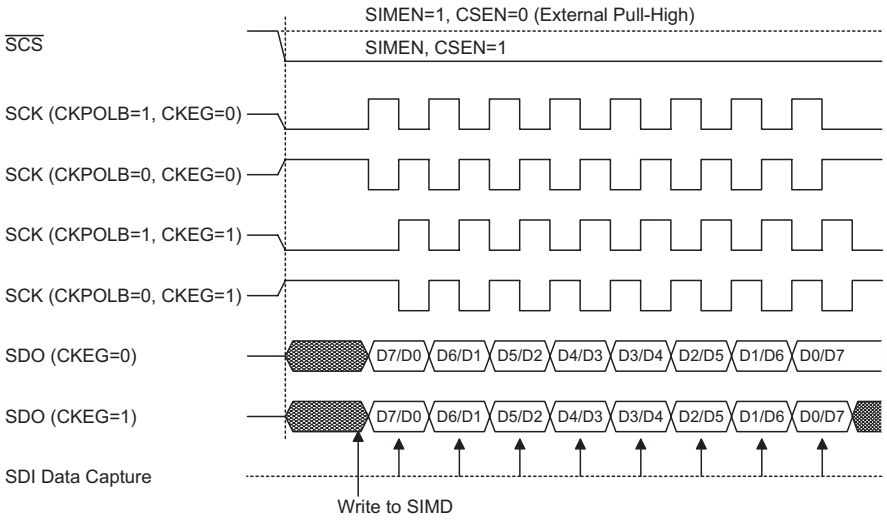
Bit 1 **WCOL**: SPI Write Collision flag
0: No collision
1: Collision
The WCOL flag is used to detect if a data collision has occurred. If this bit is high it means that data has been attempted to be written to the SIMD register during a data transfer operation. This writing operation will be ignored if data is being transferred. The bit can be cleared by the application program.

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

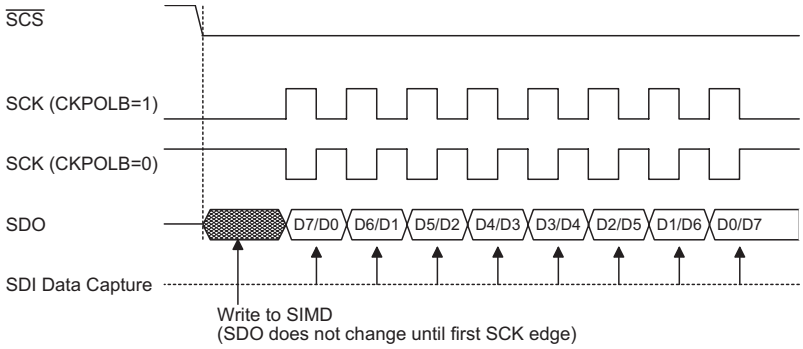
SPI Communication

After the SPI interface is enabled by setting SIMEN bit and the output pins are configured to SPI function, 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 an $\overline{SCS}$ signal to enable the slave device before a clock signal is provided. The slave data to be transferred should be well prepared at the appropriate moment relative to the $\overline{SCS}$ signal depending upon the configurations of the CKPOLB bit and CKEG bit. The accompanying timing diagram shows the relationship between the slave data and $\overline{SCS}$ signal for various configurations of the CKPOLB and CKEG bits.

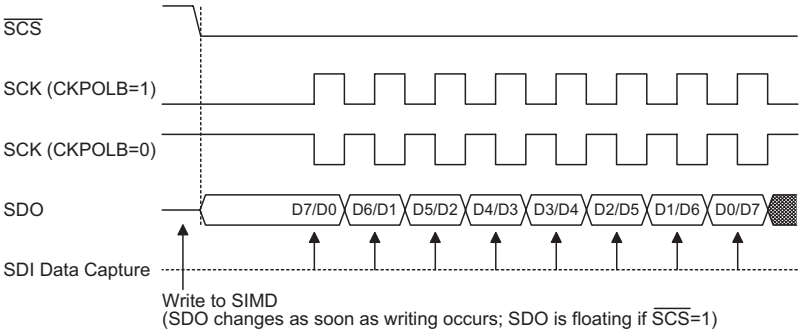
The SPI will continue to function in specific IDLE Modes if the clock source used by the SPI interface is still active.



SPI Master Mode Timing

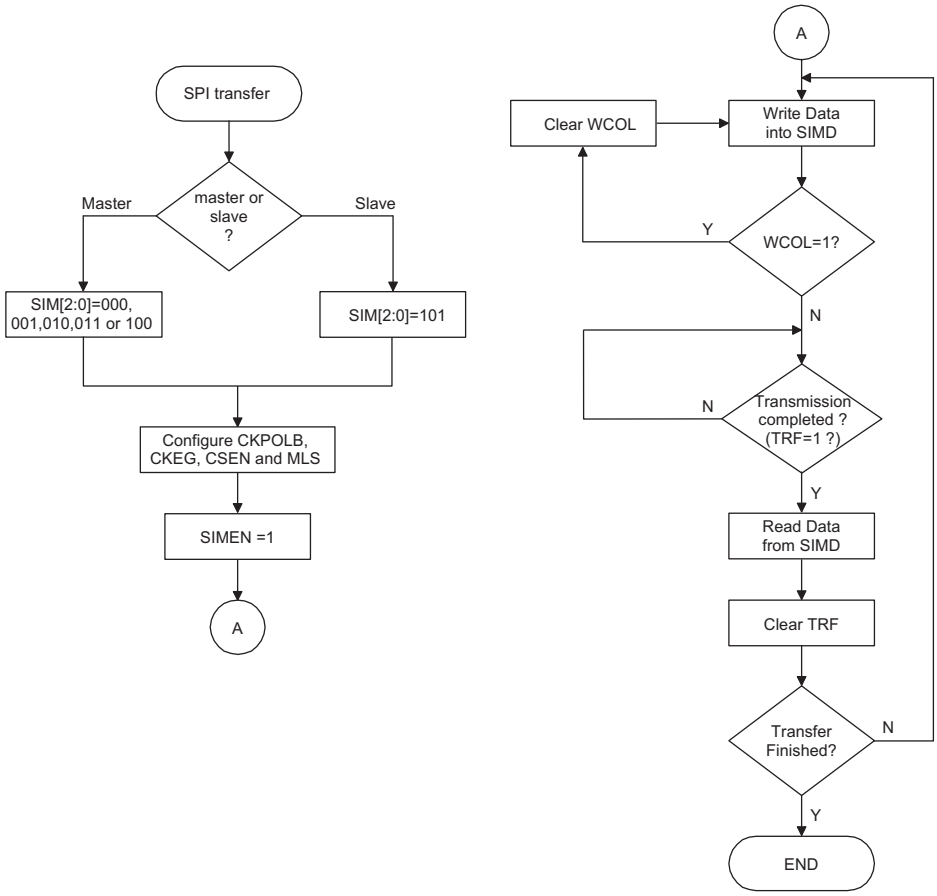


SPI Slave Mode Timing – CKEG=0



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

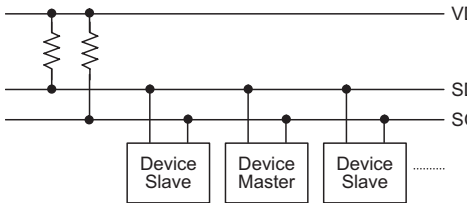
SPI Slave Mode Timing – CKEG=1



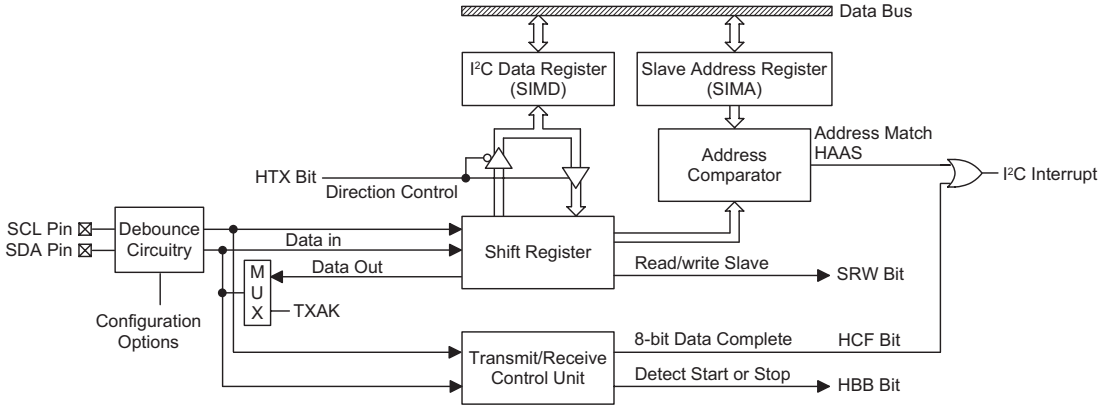
SPI Transfer Control Flowchart

I²C Interface

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



I²C Master/Slave Bus Connection

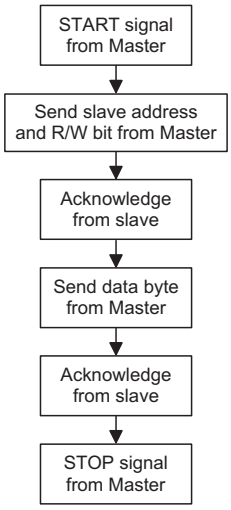


I²C Block Diagram

I²C Interface Operation

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

When two devices communicate with each other on the bidirectional I²C bus, one is known as the master device and one as the slave device. Both master and slave can transmit and receive data, however, it is the master device that has overall control of the bus. For these devices, which only operates in slave mode, there are two methods of transferring data on the I²C bus, the slave transmit mode and the slave receive mode. The pull-up control function pin-shared with SCL/SDA pin is still applicable even if I²C device is activated and the related internal pull-up register could be controlled by its corresponding pull-up control register.



A configuration option determines 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 1 or 2 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 Debounce	—	—
1 system clock debounce	f_{SYS} > 4MHz	f_{SYS} > 10MHz
2 system clock debounce	f_{SYS} > 8MHz	f_{SYS} > 20MHz

I²C Minimum f_{SYS} Frequency

I²C Registers

There are three control registers associated with the I²C bus, SIMC0, SIMC1 and SIMA and one data register, SIMD. The SIMD register, which is shown in the above SPI section, is used to store the data being transmitted and received on the I²C bus. 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. The SIMEN bit, SIM2~SIM0 bits in register SIMC0 are used by the I²C interface.

Register Name	Bit							
	7	6	5	4	3	2	1	0
SIMC0	SIM2	SIM1	SIM0	—	—	—	SIMEN	—
SIMC1	HCF	HAAS	HBB	HTX	TXAK	SRW	RNIC	RXAK
SIMD	D7	D6	D5	D4	D3	D2	D1	D0
SIMA	A6	A5	A4	A3	A2	A1	A0	—

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.

- **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

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 7~0 of the SIMA register define the device slave address.

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.

- **SIMA Register**

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

Bit 7~1 **A6~A0**: I²C slave address
A6~A0 is I²C slave address bit 7~ bit 1.

Bit 0 Unimplemented, read as "0"

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

- **SIMC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	SIM2	SIM1	SIM0	—	—	—	SIMEN	—
R/W	R/W	R/W	R/W	—	—	—	R/W	—
POR	1	1	1	—	—	—	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_{SUB}
101: SPI slave mode
110: I²C slave mode
others: Reserved

Bit 4~2 Unimplemented, read as "0"

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

Bit 0 Unimplemented, read as "0"

- **SIMC1 Register**

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

Bit 7 **HCF**: I^2C 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**: I^2C Bus address match 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**: I^2C Bus busy flag
0: I^2C Bus is not busy
1: I^2C Bus is busy
The HBB flag is the I^2C busy flag. This flag will be "1" when the I^2C 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**: I^2C slave device transmitter/receiver selection
0: Slave device is the receiver
1: Slave device is the transmitter

Bit 3 **TXAK**: I^2C Bus transmit acknowledge flag
0: Slave send acknowledge flag
1: Slave do not send acknowledge flag
The TXAK bit is the transmit acknowledge flag. After the slave device receipt of 8-bit of data, this bit will be transmitted to the bus on the 9th clock from the slave device. The slave device must always clear the TXAK bit to zero 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.

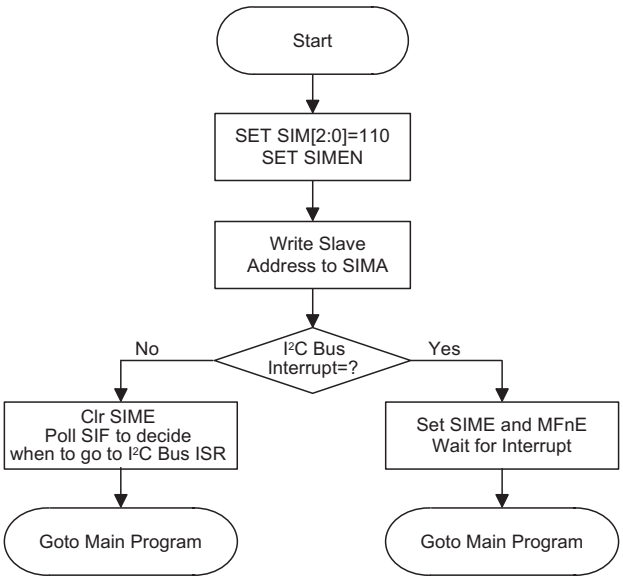
Bit 1 **RNIC**: I²C module clock source selection
0: From internal clock
1: From external clock
The I²C module can run without using internal clock, and generate an interrupt only when an address match occurs if the SIM interrupt is enabled, which can be used in the SLEEP, IDLE Mode and NORMAL Mode.

Bit 0 **RXAK**: I²C Bus Receive acknowledge flag
0: Slave receives acknowledge flag
1: Slave do 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 bit to determine whether the interrupt source originates from an address match or from the completion of an 8-bit data transfer. During a data transfer, note that after the 7-bit slave address has been transmitted, the following bit, which is the 8th bit, is the read/write bit whose value will be placed in the 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 initialize the bus, the following are steps to achieve this:

- Step 1
  Set the SIM2~SIM0 bits and SIMEN bit in the SIMC0 register to "110" and "1" repectively 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 related 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.

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 two sources, when the program enters the interrupt subroutine, the HAAS bit should be examined to see whether the interrupt source has come from a matching slave address or from the completion of a data byte transfer. When a slave address is matched, the device must be placed in either the transmit mode and then write data to the 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.

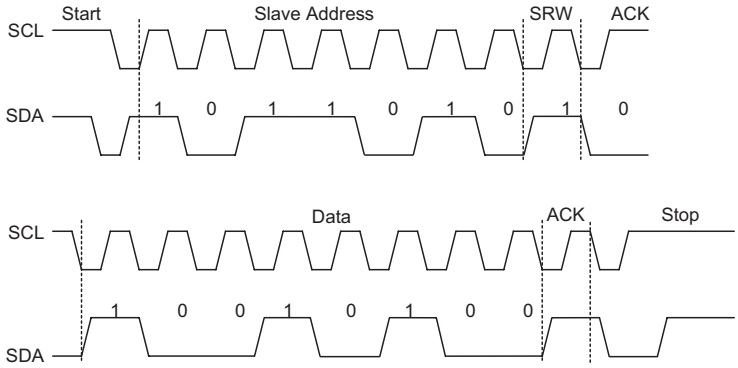
I²C Bus Slave Address Acknowledge Signal

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 cleared to "0".

I²C Bus Data and Acknowledge Signal

The transmitted data is 8-bit 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-bit of data, the receiver must transmit an acknowledge signal, level "0", before it can receive the next data byte. If the slave transmitter does not receive an acknowledge bit signal from the master receiver, then the slave transmitter will release the SDA line to allow the master to send a STOP signal to release the I²C Bus. The corresponding data will be stored in the SIMD register. If setup as a transmitter, the slave device must first write the data to be transmitted into the SIMD register. If setup as a receiver, the slave device must read the transmitted data from the SIMD register.

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

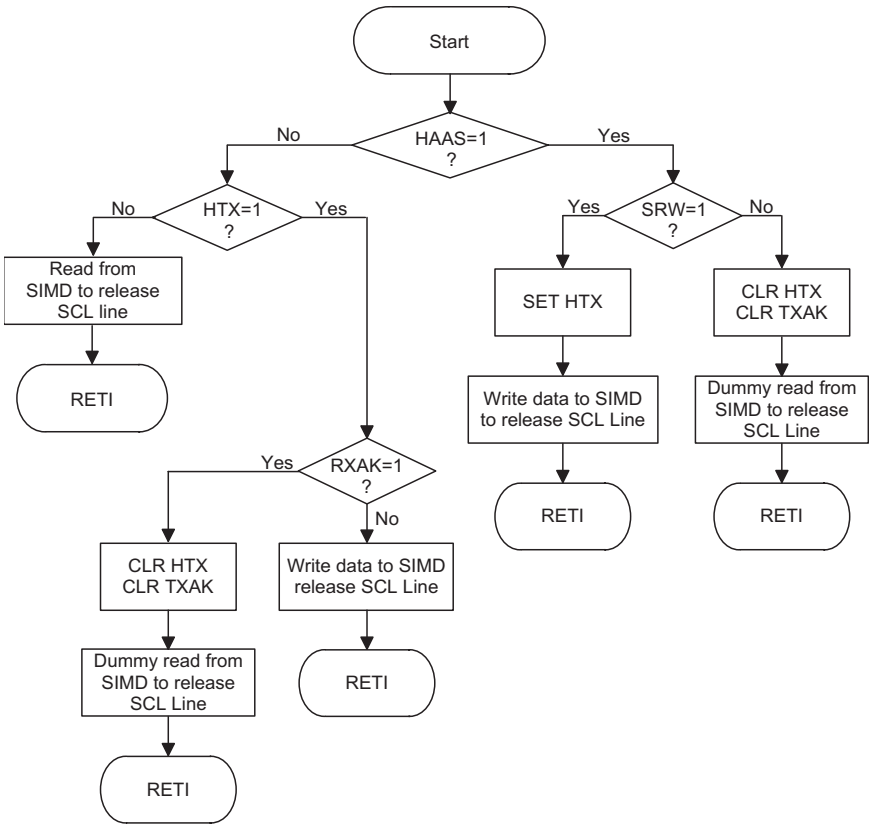


S=Start (1 bit)
SA=Slave Address (7 bits)
SR=SRW bit (1 bit)
M=Slave device send acknowledge bit (1 bit)
D=Data (8 bits)
A=ACK (RXAK bit for transmitter, TXAK bit for receiver 1 bit)
P=Stop (1 bit)

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

Note: *When a slave address is matched, the 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 Communication Timing Diagram



I²C Bus ISR Flow Chart

LDO Function

The device contains a low power voltage regulator implemented in CMOS technology. Using CMOS technology ensures low voltage drop and low quiescent current. The output voltages can range from 2.2V ~ 3.6V, which can be setup by the bits, VSEL3~VSEL0, in the LDOC register.

LDOC Register

Bit	7	6	5	4	3	2	1	0
Name	LDOEN	LDOM	VREGS	VSW	VSEL3	VSEL2	VSEL1	VSEL0
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 **LDOEN**: LDO control bit
0: Disable
1: Enable

Bit 6 **LDOM**: LDO power mode selection
0: Power-saving mode(2/3 bias current)
1: Normal mode

Bit 5 **VREGS**: VREG status when LDOEN=0 and VSW=0
0: Weak pull low
1: Floating

Bit 4 **VSW**: VREG voltage source selection
0: LDO
1: VDD

Bit 3~0 **VSEL3~VSEL0**: LDO output voltage selection (VPSW=1)
0000: 2.2V
0001: 2.3V
0010: 2.4V
0011: 2.5V
0100: 2.6V
0101: 2.7V
0110: 2.8V
0111: 2.9V
1000: 3.0V
1001: 3.1V
1010: 3.2V
1011: 3.3V
1100: 3.4V
1101: 3.5V
1110: 3.6V
1111: 3.6V

Note: When the bit VPSW is 0, the LDO output voltage will not be selected by these four bits, and the voltage will be about 1.25V

Note: 1. When the V_{DD} voltage is stable and after switching on the LDO, the LDO output voltage will be stable after 20 μs.

2. When the user changes the LDO output voltage, the LDO output voltage will be stable after 12 ms.

3. If the LDO output is as the ADC reference voltage, then the VCAP should be connected a 0.1μF capacitor to ground.

4. The LDO input voltage (V_{DD}) must greater 0.1 V than output voltage for obtaining stable output voltage.

Operational Amplifiers

There are two Operational Amplifiers in the device, OPA1 and OPA2. The OPA1 amplifier is setup for a band pass filter application circuit. For PIR applications, it is recommended that the bandwidth is setup from 0.3 to 8Hz. The OPA2 amplifier is a programmable gain amplifier whose gain can be setup to have a range of 32 to 94. This gain is setup using the application software.

The OPA1EC is the reference voltage pin for OPA1. It connects a filter capacitance to stabilize the reference voltage normally. When OPA1SW bit is 1, it is for fast warm-up. Please refer to the following registers for the details.

Operational Amplifier Registers

The Operational Amplifiers are fully under the control of internal registers, OPAC0 and OPAC1. These registers control enable/disable function.

OPAC0 Register

Bit	7	6	5	4	3	2	1	0
Name	OPA2EN	OPA1EN	VPSW	PA7S	—	OPA1SW	VPS1	VPS0
R/W	R/W	R/W	R/W	R/W	—	R/W	R/W	R/W
POR	0	0	0	0	—	0	0	0

Bit 7 **OPA2EN**:OPA2 enable or disable control bit
0: Disable
1: Enable

Bit 6 **OPA1EN**:OPA1 enable or disable control bit
0: Disable
1: Enable

Bit 5 **VPSW**: Switch control bit
0: Open
1: Closed

Bit 4 **PA7S**: OPA1EC/PA7 selection bit
0: PA7
1: OPA1EC

Bit 3 Unimplemented, read as "0"

Bit 2 **OPA1SW**: OPA1 short switch between non-inverting input and output
0: Open
1: Closed

Bit 1 **VPS1**: V_P voltage selection
0: Floating
1: 1/2 V_{REG} when VPSW is set to 1

Bit 0 **VPS0**: V_P voltage selection
0: Floating
1: 2/3 V_{REG} when VPSW is set to 1

Note : VPS0 bit and VPS1 bit can not be 1 at the same time.

OPAC1 Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	PA5S	PGAC4	PGAC3	PGAC2	PGAC1	PGAC0
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 **PA5S**: Port A5 function selection
0: PA5
1: OPA2E

Bit 4~0 **PGAC4~PGAC0**: OPA2 gain control bit
Gain = 32 + (PGAC x 2)

Interrupts

Interrupts are an important part of any microcontroller system. When an external event or an internal function such as a Timer/Event Counter 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 one external interrupt and several internal interrupts functions. The external interrupt is generated by the action of the external INT pin, while the internal interrupts are generated by various internal functions such as Timer/Event Counters, LVD, EEPROM, SIM and the A/D converter.

Interrupt Registers

Overall interrupt control, which basically means the setting of request flags when certain microcontroller conditions occur and the setting of interrupt enable bits by the application program, is controlled by a series of registers, located in the Special Purpose Data Memory, as shown in the accompanying table. The first is the INTC0~INTC1 registers which setup the primary interrupts, the second is the MFIC register which setup the Multi-function interrupts.

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

Function	Enable Bit	Request Flag	Notes
Global	EMI	—	—
INT Pin	INTE	INTF	—
Auto Conversion Circuit	ACCE	ACCF	—
Timer/Event Counter	TnIE	TnF	n=0~1
SIM	SIME	SIMF	—
Multi-function	MFE	MFF	—
A/D Converter	ADE	ADF	—
EEPROM	DEE	DEF	—
LVD	LVE	LVF	—

Interrupt Register Bit Naming Conventions

Register Name	Bit							
	7	6	5	4	3	2	1	0
INTEG	—	—	—	—	—	—	INTS1	INTS0
INTC0	—	T0F	ACCF	INTF	T0IE	ACCE	INTE	EMI
INTC1	LVDF	MFF	SIMF	T1F	LVDE	MFE	SIME	T1IE
MFIC	—	DEF	—	ADF	—	DEE	—	ADE

Interrupt Registers List

INTEG Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	INTS1	INTS0
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2 Unimplemented, read as "0"

Bit 1~0 **INTS1~INTS0**: interrupt edge control for INT pin
00: Disable
01: Rising edge
10: Falling edge
11: Both rising and falling edges

INTC0 Register

Bit	7	6	5	4	3	2	1	0
Name	—	T0F	ACCF	INTF	T0IE	ACCE	INTE	EMI
R/W	—	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	0	0	0	0	0	0	0

Bit 7 Unimplemented, read as "0"

Bit 6 **T0F**: Timer/Event Counter 0 interrupt Request Flag
0: No request
1: Interrupt request

Bit 5 **ACCF**: Auto conversion circuit interrupt request flag
0: No request
1: Interrupt request

Bit 4 **INTF**: External Interrupt Request Flag
0: No request
1: Interrupt request

Bit 3 **T0IE**: Timer/Event Counter 0 interrupt Control
0: Disable
1: Enable

Bit 2 **ACCE**: Auto conversion circuit interrupt control
0: Disable
1: Enable

Bit 1 **INTE**: External 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	LVDF	MFF	SIMF	T1F	LVDE	MFE	SIME	T1IE
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 **LVDF**: LVD interrupt request flag
0: No request
1: Interrupt request

Bit 6 **MFF**: Multi-function Interrupt Request Flag
0: No request
1: Interrupt request

Bit 5 **SIMF**: SPI/I^2C interrupt request flag
0: No request
1: Interrupt request

Bit 4 **T1F**: Timer/Event Counter 1 interrupt request flag
0: No request
1: Interrupt request

Bit 3 **LVDE**: LVD interrupt Control
0: Disable
1: Enable

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

Bit 1 **SIME**: SPI/I^2C Interrupt Control
0: Disable
1: Enable

Bit 0 **T1IE**: Timer/Event Counter 1 Interrupt Control
0: Disable
1: Enable

MFIC Register

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

Bit 7 Unimplemented, read as "0"

Bit 6 **DEF**: Data EEPROM interrupt request flag
0: No request
1: Interrupt request

Bit 5 Unimplemented, read as «0»

Bit 4 **ADF**: A/D converter Interrupt Request Flag
0: No request
1: Interrupt request

Bit 3 Unimplemented, read as «0»

Bit 2 **DEE**: Data EEPROM Interrupt Control
0: Disable
1: Enable

Bit 1 Unimplemented, read as «0»

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

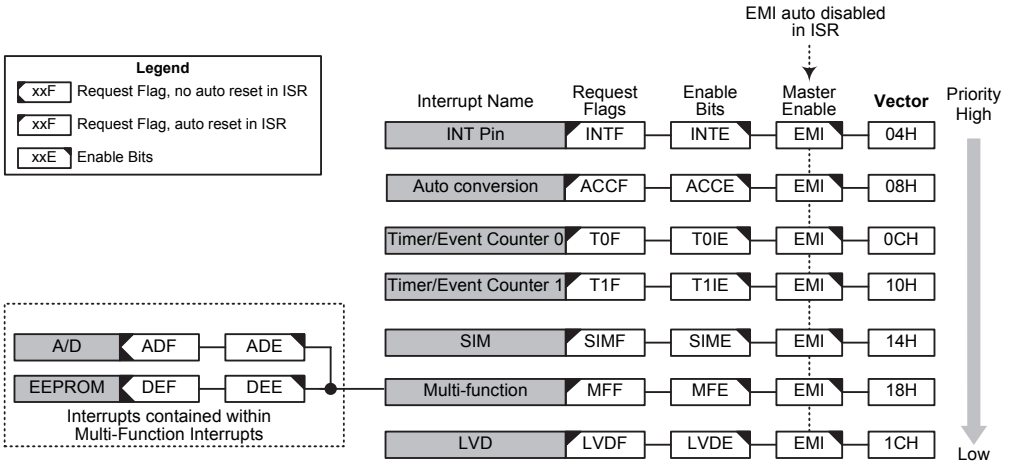
Interrupt Operation

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

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

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

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



Interrupt Structure

External Interrupt

The external interrupt is controlled by signal transitions on the INT pin. An external interrupt request will take place when the external interrupt request flag, INTF, is set, which will occur when a transition, whose type is chosen by the edge select bits, appears on the external interrupt pin. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and respective external interrupt enable bit, INTE, must first be set. Additionally the correct interrupt edge type must be selected using the related register to enable the external interrupt function and to choose the trigger edge type. As the external interrupt pin is pin-shared with I/O pin, it can only be configured as external interrupt pin if the external interrupt enable bit in the corresponding interrupt register has been set. The pin must also be setup as an input by setting the corresponding bit in the port control register. When the interrupt is enabled, the stack is not full and the correct transition type appears on the external interrupt pin, a subroutine call to the external interrupt vector, will take place. When the interrupt is serviced, the external interrupt request flag, INTF, 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 pin will remain valid even if the pin is used as an external interrupt input.

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

Auto conversion circuit Interrupt

The device contains an Auto Conversion Circuit which has its own independent interrupt. The internal Auto Conversion Circuit interrupt is controlled by the termination of an A/D conversion process. An internal Auto Conversion Circuit interrupt will take place when the Auto Conversion Circuit interrupt request flag ACCF, 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 Auto Conversion Circuit interrupt enable bit, ACCE, 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 Auto Conversion Circuit interrupt vector, will take place. When the interrupt is serviced, the Auto Conversion Circuit interrupt flag, ACCF, will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts.

Timer/Event Counter Interrupt

For a Timer/Event Counter interrupt to occur, the global interrupt enable bit, EMI, and the corresponding timer interrupt enable bit, T0IE or T1IE, must first be set. An actual Timer/Event Counter interrupt will take place when the Timer/Event Counter request flag, T0F or T1F, is set, a situation that will occur when the Timer/Event Counter overflows. When the interrupt is enabled, the stack is not full and a Timer/Event Counter overflow occurs, a subroutine call to the timer interrupt vector, will take place. When the interrupt is serviced, the timer interrupt request flag, T0F or T1F, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts.

Serial Interface Module Interrupt

The Serial Interface Module Interrupt is also known as the SIM interrupt. 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 SPI or I^2C interface, or an I^2C address match occurs. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and the SIM Interface Interrupt enable bit, SIME, must first be set. When the interrupt is enabled, the stack is not full and any these conditions are created, a subroutine call to the respective

interrupt vector, will take place. When the SIM Interface Interrupt is serviced, the SIM interrupt request flag, SIMF, will be automatically cleared and the EMI bit will be automatically cleared to disable other interrupts.

Multi-function Interrupt

Within the device there is one Multi-function interrupt. Unlike the other independent interrupts, these interrupts have no independent source, but rather are formed from other existing interrupt sources, namely, A/D converter interrupt and EEPROM interrupt.

A Multi-function interrupt request will take place the Multi-function interrupt request flag, MFF is set. The Multi-function interrupt flag will be set when any of its 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 the Multi-function interrupt occurs, a subroutine call to the Multi-function interrupt vector will take place. When the interrupt is serviced, the related Multi-Function request flag will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts.

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

A/D Converter Interrupt

The A/D converter interrupt is contained within the Multi-function 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, and associated Multi-function interrupt enable bit, 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 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 ADF flag will not be automatically cleared, it has to be cleared by the application program.

EEPROM Interrupt

The EEPROM interrupt is contained within the Multi-function Interrupt. An EEPROM Interrupt request will take place when the EEPROM Interrupt request flag, DEF, is set, which occurs when an EEPROM Write cycle ends. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and EEPROM Interrupt enable bit, DEE, 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 EEPROM Interrupt vector, will take place. When the EEPROM Interrupt is serviced, the EMI bit will be automatically cleared to disable other interrupts, however only the Multi-function interrupt request flag will be also automatically cleared. As the DEF flag will not be automatically cleared, it has to be cleared by the application program.

LVD Interrupt

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

Interrupt Wake-up Function

Each of the interrupt functions has the capability of waking up the microcontroller when in the SLEEP or IDLE Mode. A wake-up is generated when an interrupt request flag changes from low to high and is independent of whether the interrupt is enabled or not. Therefore, even though the device is in the SLEEP or IDLE Mode and its system oscillator stopped, situations such as external edge transitions on the external interrupt 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, MFF, will be automatically cleared, the individual request flag for the function needs to be cleared by the application program.

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

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

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

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

Low Voltage Detector – LVD

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 Detect
1: Low Voltage Detect

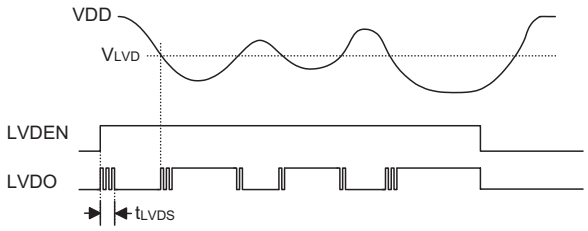
Bit 4 **LVDEN**: Low Voltage Detector Control
0: Disable
1: Enable

Bit 3 Unimplemented, read as «0»

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

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 LVDF 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 LVDF flag should be first set high before the device enters the SLEEP or IDLE Mode.

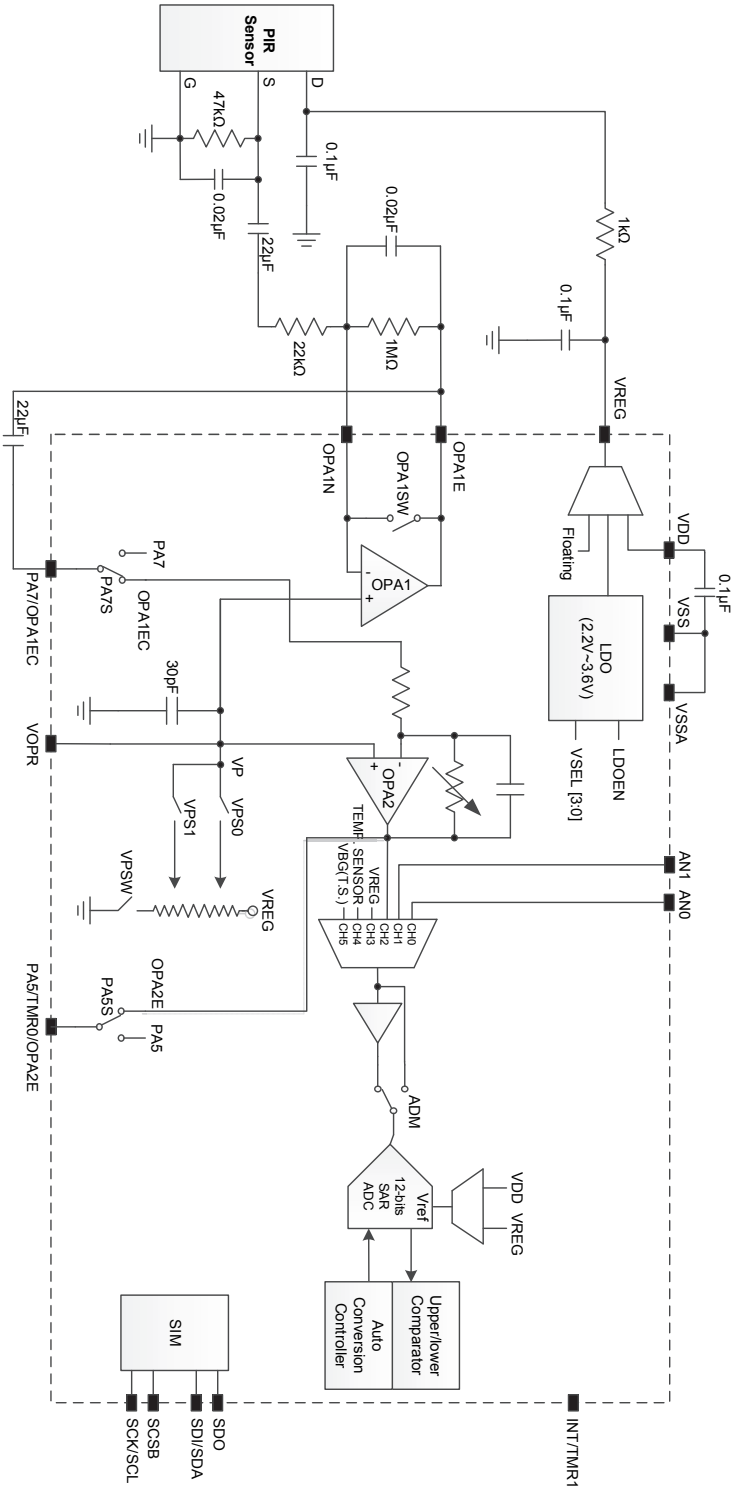
When LVD function is enabled, it is recommenced to clear LVD flag first, and then enables interrupt function to avoid mistake action.

Configuration Options

Configuration options refer to certain options within the MCU that are programmed into the device during the programming process. During the development process, these options are selected using the HT-IDE software development tools. As these options are programmed into the device using the hardware programming tools, once they are selected they cannot be changed later using the application program. All options must be defined for proper system function, the details of which are shown in the table.

No.	Options
Oscillator Options	
1	HIRC Frequency Selection: 1. 1MHz 2. 2MHz 3. 4MHz 4. 8MHz
Watchdog Timer Options	
2	WDT function: Always enable or By S/W control
I²C Option	
3	I²C Debounce Time: no debounce, 1 system clock, 2 system clock
Lock Options	
4	Lock All
	Partial Lock

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

Affected flag(s) Z

CPLA [m] Complement Data Memory with result in ACC

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

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

Affected flag(s) Z

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

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

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

Affected flag(s) C

DEC [m] Decrement Data Memory

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

Operation [m] ← [m] − 1

Affected flag(s) Z

DECA [m] Decrement Data Memory with result in ACC

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

Operation ACC ← [m] − 1

Affected flag(s) Z

HALT Enter power down mode

Description This instruction stops the program execution and turns off the system clock. The contents of the Data Memory and registers are retained. The WDT and prescaler are cleared. The power down flag PDF is set and the WDT time-out flag TO is cleared.

Operation TO ← 0
PDF ← 1

Affected flag(s) TO, PDF

INC [m] Increment Data Memory

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

Operation [m] ← [m] + 1

Affected flag(s) Z

INCA [m] Increment Data Memory with result in ACC

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

Operation ACC ← [m] + 1

Affected flag(s) Z

JMP addr Jump unconditionally

Description The contents of the Program Counter are replaced with the specified address. Program execution then continues from this new address. As this requires the insertion of a dummy instruction while the new address is loaded, it is a two cycle instruction.

Operation Program Counter ← addr

Affected flag(s) None

MOV A,[m] Move Data Memory to ACC

Description The contents of the specified Data Memory are copied to the Accumulator.

Operation ACC ← [m]

Affected flag(s) None

MOV A,x Move immediate data to ACC

Description The immediate data specified is loaded into the Accumulator.

Operation ACC ← x

Affected flag(s) None

MOV [m],A Move ACC to Data Memory

Description The contents of the Accumulator are copied to the specified Data Memory.

Operation [m] ← ACC

Affected flag(s) None

NOP No operation

Description No operation is performed. Execution continues with the next instruction.

Operation No operation

Affected flag(s) None

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

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

Operation ACC ← ACC ″OR″ [m]

Affected flag(s) Z

OR A,x Logical OR immediate data to ACC

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

Operation ACC ← ACC ″OR″ x

Affected flag(s) Z

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

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

Operation [m] ← ACC ″OR″ [m]

Affected flag(s) Z

RET Return from subroutine

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

Operation Program Counter ← Stack

Affected flag(s) None

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

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

Operation Program Counter ← Stack
ACC ← x

Affected flag(s) None

RETI Return from interrupt

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

Operation Program Counter ← Stack
EMI ← 1

Affected flag(s) None

RL [m] Rotate Data Memory left

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

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

Affected flag(s) None

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

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

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

Affected flag(s) None

RLC [m] Rotate Data Memory left through Carry

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

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

Affected flag(s) C

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

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

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

Affected flag(s) C

RR [m] Rotate Data Memory right

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

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

Affected flag(s) None

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

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

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

Affected flag(s): None

RRC [m] Rotate Data Memory right through Carry

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

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

Affected flag(s): C

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

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

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

Affected flag(s): C

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

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

Operation: ACC ← ACC − [m] − C

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

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

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

Operation: [m] ← ACC − [m] − C

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

SDZ [m] Skip if decrement Data Memory is 0

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

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

Affected flag(s): None

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

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

Operation ACC ← [m] − 1
Skip if ACC=0

Affected flag(s) None

SET [m] Set Data Memory

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

Operation [m] ← FFH

Affected flag(s) None

SET [m].i Set bit of Data Memory

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

Operation [m].i ← 1

Affected flag(s) None

SIZ [m] Skip if increment Data Memory is 0

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

Operation [m] ← [m] + 1
Skip if [m]=0

Affected flag(s) None

SIZA [m] Skip if increment Data Memory is zero with result in ACC

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

Operation ACC ← [m] + 1
Skip if ACC=0

Affected flag(s) None

SNZ [m].i Skip if bit i of Data Memory is not 0

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

Operation Skip if [m].i ≠ 0

Affected flag(s) None

SUB A,[m] Subtract Data Memory from ACC

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

Operation ACC ← ACC − [m]

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

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

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

Operation [m] ← ACC − [m]

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

SUB A,x Subtract immediate data from ACC

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

Operation ACC ← ACC − x

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

SWAP [m] Swap nibbles of Data Memory

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

Operation [m].3~[m].0 ↔ [m].7~[m].4

Affected flag(s) None

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

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

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

Affected flag(s) None

SZ [m] Skip if Data Memory is 0

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

Operation Skip if [m]=0

Affected flag(s) None

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

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

Operation ACC ← [m]
Skip if [m]=0

Affected flag(s) None

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

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

Operation Skip if [m].i=0

Affected flag(s) None

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

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

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

Affected flag(s) None

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

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

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

Affected flag(s) None

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

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

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

Affected flag(s) None

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

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

Operation ACC ← ACC ″XOR″ [m]

Affected flag(s) Z

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

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

Operation [m] ← ACC ″XOR″ [m]

Affected flag(s) Z

XOR A,x Logical XOR immediate data to ACC

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

Operation ACC ← ACC ″XOR″ x

Affected flag(s) Z

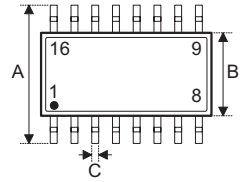
Package Information

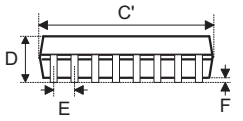
Note that the package information provided here is for consultation purposes only. As this information may be updated at regular intervals users are reminded to consult the Holtek website for the latest version of the package information.

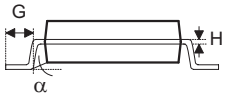
Additional supplementary information with regard to packaging is listed below. Click on the relevant section to be transferred to the relevant website page.

- Further Package Information (include Outline Dimensions, Product Tape and Reel Specifications)
- Packing Meterials Information
- Carton information

16-pin NSOP (150mil) Outline Dimensions



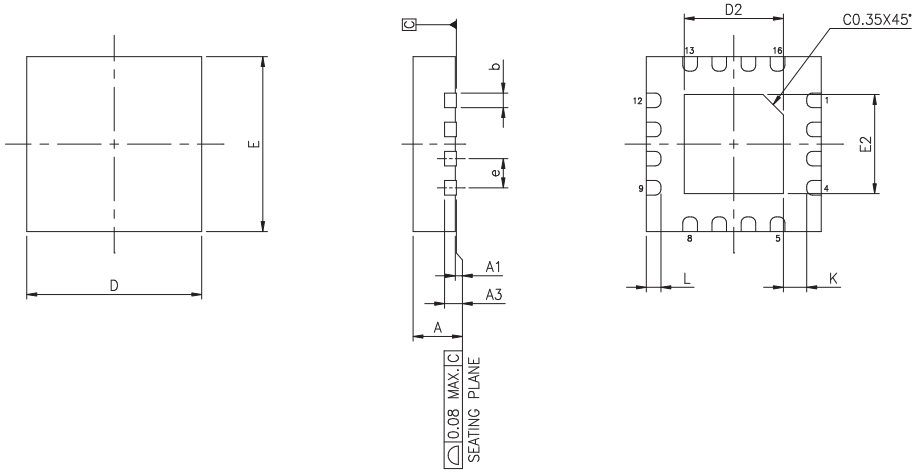




Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	—	0.236 BSC	—
B	—	0.154 BSC	—
C	0.012	—	0.020
C’	—	0.390 BSC	—
D	—	—	0.069
E	—	0.050 BSC	—
F	0.004	—	0.010
G	0.016	—	0.050
H	0.004	—	0.010
α	0°	—	8°

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°

SAW Type 16-pin (3mm×3mm, FP0.25mm) QFN Outline Dimensions



Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	0.028	0.030	0.031
A1	0.000	0.001	0.002
A2	—	0.008 REF	—
b	0.007	0.010	0.012
D	—	0.118 BSC	—
E	—	0.118 BSC	—
e	—	0.020 BSC	—
D2	0.063	0.067	0.069
E2	0.063	0.067	0.069
L	0.008	0.010	0.012
K	0.008	—	—

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	0.700	0.750	0.800
A1	0.000	0.020	0.050
A2	—	0.200 REF	—
b	0.180	0.250	0.300
D	—	3.000 BSC	—
E	—	3.000 BSC	—
e	—	0.50 BSC	—
D2	1.60	1.70	1.75
E2	1.60	1.70	1.75
L	0.20	0.25	0.30
K	0.20	—	—