Induction Cooker Flash MCU

HT45F0004

Revision: V1.50 Date: February 23, 2024

www.holtek.com

Table of Contents

Features

CPU Features

- Operating Voltage
  - f_{SYS}=4MHz: 2.2V~5.5V
  - f_{SYS}=8MHz: 3.3V~5.5V
- Up to 0.5μs instruction cycle with 8MHz system clock at V_{DD}=5V
- Power down and wake-up functions to reduce power consumption
- Two Oscillators
  - High Speed Internal RC – HIRC
  - High Speed External Crystal – HXT
- Fully integrated internal 8 MHz oscillator requires no external components
- All instructions executed in one or two instruction cycles
- Table read instructions
- 63 powerful instructions
- 8-level subroutine nesting
- Bit manipulation instruction

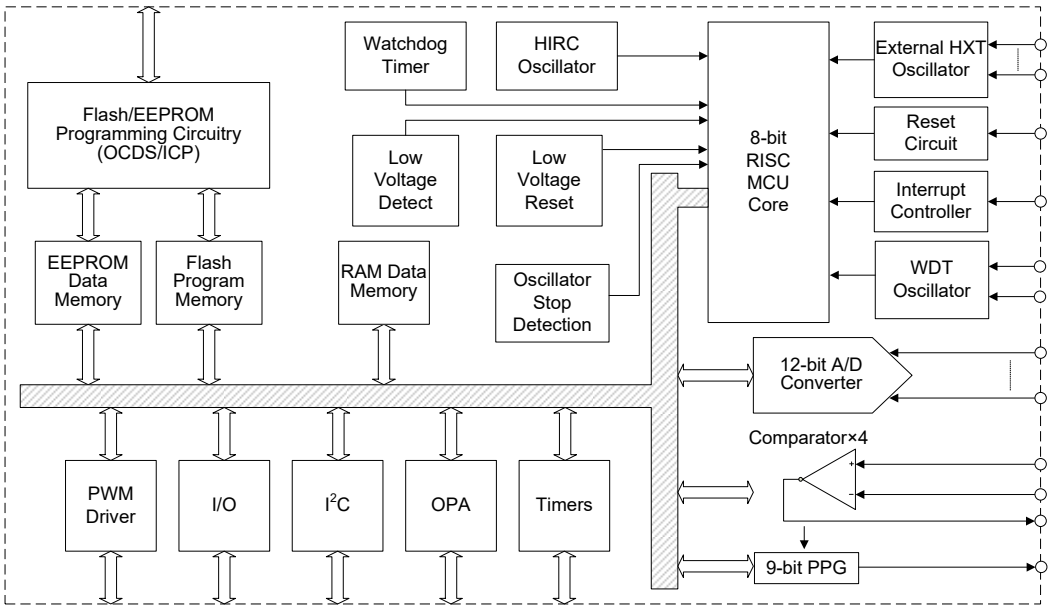
Peripheral Features

- Program Memory: 4K×16
- RAM Data Memory: 208×8
- True EEPROM Memory: 32×8
- Watchdog Timer function
- Oscillator Stop Detection
- Up to 17 bidirectional I/O lines
- 12 channels 12-bit resolution A/D converter
- 9-bit programmable pulse generator
  - Pulse width limit
  - Two PPG preload registers
  - Non-retriggered control
  - Active high or low output (by configuration option)
- Single pin-shared external interrupt
  - Integrated de-bounce circuit for the external interrupt
  - De-bounce time is selected by software options
- Four Comparators
- Operational Amplifier - input voltage offset adjustable by software
- Peripheral clock output and PWM output
- Three 8-bit programmable timer/event counters
  - Timer 0 can be configured to count synchronism pulse number or measure synchronism pulse high or low period
  - Timer 1 can be configured to implement PPG non-retriggered function
- I^2C interface – Slave Mode
- Low Voltage Reset function
- Low Voltage Detect function
- Package type: 16-pin NSOP and 20-pin SOP

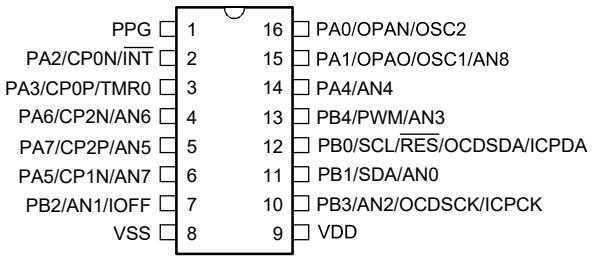
General Description

The device is a Flash Memory A/D type 8-bit high performance RISC architecture microcontroller designed for the A/D applications that interface directly to analog signals, such as those from sensors. The advantages of low power consumption, I/O flexibility, timer functions, oscillator options, multi-channel A/D converter, power down and wake-up functions enhance the versatility of this device to suit for a wide range of A/D application possibilities such as sensor signal processing. The device also provides four comparators, an operational amplifier and a programmable pulse generator – PPG, hence it is particularly for use in products such as induction cooker and home appliances.

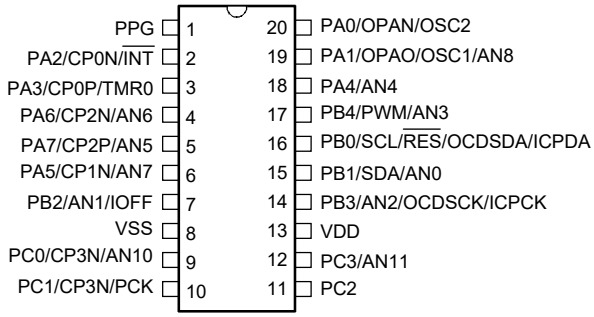
Block Diagram



Pin Assignment



HT45F0004/HT45V0004
16 NSOP-A



HT45F0004/HT45V0004
20 SOP-A

Note: 1. The OCDSDA and OCDSCK pins are the OCDS dedicated pins and only available for the HT45V0004 device which is the OCDS EV chip for the HT45F0004 device.
2. For less pin-count package types there will be unbonded pins which should be properly configured to avoid unwanted current consumption resulting from floating input conditions. Refer to the "Standby Current Considerations" and "Input/Output Ports" sections.

Pin Description

Pin Name	Function	OPT	I/T	O/T	Description
PA0/OPAN/OSC2	PA0	CO	ST	CMOS	General purpose I/O. Pull-high and wake-up functions are selected by configuration options.
	OPAN	CO	OPAI	—	OPA inverting input
	OSC2	CO	—	HXT	HXT pin
PA1/OPAO/OSC1/AN8	PA1	CO	ST	CMOS	General purpose I/O. Pull-high and wake-up functions are selected by configuration options.
	OPAO	CO	—	OPAO	OPA output pin
	OSC1	CO	HXT	—	HXT pin
	AN8	ADCR PCRH	AN	—	A/D Converter analog input
PA2/CP0N/INT	PA2	CO	ST	CMOS	General purpose I/O. Pull-high and wake-up functions are selected by configuration options
	CP0N	CO	CMPI	—	Comparator 0 inverting input pin
	INT	—	ST	—	External interrupt input
PA3/CP0P/TMR0	PA3	CO	ST	CMOS	General purpose I/O. Pull-high and wake-up functions are selected by configuration options
	CP0P	CO CMPSC	CMPI	—	Comparator 0 non-inverting input pin
	TMR0	—	ST	—	External Timer 0 clock input

Pin Name	Function	OPT	I/T	O/T	Description
PA4/AN4	PA4	CO	ST	CMOS	General purpose I/O. Pull-high and wake-up functions are selected by configuration options
	AN4	ADCR PCRL	AN	—	A/D Converter analog input
PA5/CP1N/AN7	PA5	CO	ST	CMOS	General purpose I/O. Pull-high and wake-up functions are selected by configuration options
	CP1N	CO CMPSC	CMPI	—	Comparator 1 inverting input pin
	AN7	ADCR PCRL	AN	—	A/D Converter analog input
PA6/CP2N/AN6	PA6	CO	ST	CMOS	General purpose I/O. Pull-high and wake-up functions are selected by configuration options
	CP2N	CO	CMPI	—	Comparator 2 inverting input pin
	AN6	ADCR PCRL	AN	—	A/D Converter analog input
PA7/CP2P/AN5	PA7	CO	ST	CMOS	General purpose I/O. Pull-high and wake-up functions are selected by configuration options
	CP2P	CO	CMPI	—	Comparator 2 non-inverting input pin
	AN5	ADCR PCRL	AN	—	A/D Converter analog input
PB0/SCL/$\overline{\text{RES}}$/ OCDSDA/ICPDA	PB0	CO	ST	CMOS	General purpose I/O. Pull-high function is selected by configuration options
	SCL	CO	ST	NMOS	I^2C clock line
	$\overline{\text{RES}}$	CO	ST	—	Reset input
	OCDSDA	—	ST	CMOS	OCDS Address/Data, for EV chip only
	ICPDA	—	ST	CMOS	ICP Data/Address
PB1/SDA/AN0	PB1	CO	ST	CMOS	General purpose I/O. Pull-high function is selected by configuration options
	SDA	CO	ST	NMOS	I^2C data line
	AN0	ADCR PCRL	AN	—	A/D Converter analog input
PB2/AN1/IOFF	PB2	CO	ST	CMOS	General purpose I/O. Pull-high function is selected by configuration options
	IOFF	CO	—	CMOS	IOFF output
	AN1	ADCR PCRL	AN	—	A/D Converter analog input
PB3/AN2/OCDSCK/ ICPCK	PB3	PBPU	ST	CMOS	General purpose I/O. Pull-high function is selected by configuration options
	AN2	ADCR PCRL	AN	—	A/D Converter analog input
	OCDSCK	—	ST	—	OCDS Clock pin, for EV chip only
	ICPCK	—	ST	—	ICP clock
PB4/PWM/AN3	PB4	CO	ST	CMOS	General purpose I/O. Pull-high function is selected by configuration options
	PWM	CO	—	CMOS	PWM output
	AN3	ADCR PCRL	AN	—	A/D Converter analog input
PC0/CP3N/AN10	PC0	CO	ST	CMOS	General purpose I/O. Pull-high function is selected by configuration options
	AN10	ADCR PCRH	AN	—	A/D Converter analog input
	CP3N	CO	CMPI	—	Comparator 3 inverting input pin

Pin Name	Function	OPT	I/T	O/T	Description
PC1/CP3N/PCK	PC1	CO	ST	CMOS	General purpose I/O. Pull-high function is selected by configuration options
	PCK	CO	—	CMOS	PCK output
	CP3N	CO	CMPI	—	Comparator 3 inverting input pin
PC2	PC2	CO	ST	CMOS	General purpose I/O. Pull-high function is selected by configuration options
PC3/AN11	PC3	CO	ST	CMOS	General purpose I/O. Pull-high function is selected by configuration options
	AN11	ADCR PCRH	AN	—	A/D Converter analog input
PPG	PPG	CO	—	CMOS	The PPG pin is floating during PPG inactive period, power-on reset, $\overline{RES}$ pin reset, LVR reset and oscillator stop condition. The PPG output active level (active low or active high) can be selected via configuration option.
VDD	VDD	—	PWR	—	Power supply
VSS	VSS	—	PWR	—	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
HXT: High frequency crystal oscillator
OPAI: OPA input
OPAO: OPA output
CMPI: Comparator input

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..... -60°C to 150°C
Operating Temperature..... -40°C to 85°C
I_{OL} Total 80mA
I_{OH} Total..... -80mA
Total Power Dissipation 500mW

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

D.C. Electrical Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	Operating Voltage	—	f_{SYS}=4MHz	2.2	—	5.5	V
		—	f_{SYS}=8MHz	3.3	—	5.5	V
I_{DD1}	Operating Current (Crystal OSC)	3V	No load, f_{SYS}=4MHz ADC Disable	—	0.6	1.5	mA
		5V		—	2.0	4.0	mA
I_{DD2}	Operating Current (Crystal OSC, HIRC OSC)	5V	No load, f_{SYS}=8MHz ADC Disable	—	4	6	mA
I_{STB1}	Standby Current (WDT Enable)	3V	No load, system HALT	—	—	5	μA
		5V		—	—	10	μA
I_{STB2}	Standby Current (WDT Disable)	3V	No load, system HALT	—	—	1	μA
		5V		—	—	2	μA
V_{IL1}	Input Low Voltage for I/O Ports, TMR0 and $\overline{INT}$	—	—	0	—	$0.3V_{DD}$	V
V_{IH1}	Input High Voltage for I/O Ports, TMR0 and $\overline{INT}$	—	—	$0.7V_{DD}$	—	V_{DD}	V
V_{IL2}	Input Low Voltage ($\overline{RES}$)	—	—	0	—	$0.4V_{DD}$	V
V_{IH2}	Input High Voltage ($\overline{RES}$)	—	—	$0.9V_{DD}$	—	V_{DD}	V
V_{LVR1}	Low Voltage Reset Voltage	—	V_{LVR}=2.1V	1.98	2.1	2.22	V
V_{LVR2}		—	V_{LVR}=3.0V	2.83	3.0	3.17	V
V_{LVD}	Low Voltage Detector Voltage	—	—	4.12	4.4	4.7	V
I_{OL1}	I/O Port Sink Current	3V	V_{OL}=0.1V_{DD}	4	8	—	mA
		5V		10	20	—	mA
I_{OH1}	I/O Port Source Current	3V	V_{OL}=0.9V_{DD}	-2	-4	—	mA
		5V		-5	-10	—	mA
R_{PH}	Pull-high Resistance	3V	—	20	60	100	kΩ
		5V	—	10	30	50	kΩ

A.C. Electrical Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Condition				
f_{SYS}	System Clock (Crystal OSC, RC OSC)	2.2V~5.5V	—	400	—	4000	kHz
		3.3V~5.5V	—	400	—	8000	kHz
f_{HIRC}	System Clock (HIRC)	5V	8MHz, Ta=25°C	7760	8000	8240	kHz
	Temperature Drift Percentage Relative to 25 °C	5V	Ta=0~55°C	-2	—	+2	%
f_{TIMER}	Timer Input Frequency (TMRn)	2.2V~5.5V	—	0	—	4000	kHz
		3.3V~5.5V	—	0	—	8000	kHz
t_{WDTOSC}	Watchdog Oscillator Period	3V	—	45	90	180	μs
		5V		32	65	130	
t_{RES}	External Reset Low Pulse Width	—	—	1	—	—	μs
t_{SST}	System Start-up Timer Period	—	Power on or wake-up from power-down mode	—	1024	—	t_{SYS}
t_{INT}	Interrupt Pulse Width	—	No de-bounced	1	—	—	μs
t_{LVR}	Low Voltage Width to Reset	—	—	0.25	1	2	ms
t_{OSTP}	Oscillator Stop to Reset Time	—	—	—	10	—	μs

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Condition				
t_{IIC}	I²C Bus Clock Period	—	Connect to external pull- high resistor 2kΩ	64	—	—	t_{SYS}

Note: 1. $t_{SYS}=1/f_{SYS}$
2. t_{INT} Minimum pulse width is for guaranteed interrupt. It is possible for a smaller pulse width to be recognized.

ADC Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{AD}	A/D Converter Operating Voltage	2.7V~5.5V	—	0	—	V_{DD}	V
DNL	A/D Converter Differential Non-linearity	5V	t_{AD}=0.5μs	-2	—	+2	LSB
INL	A/D Converter Integral Non-Linearity	5V	t_{AD}=0.5μs	-4	—	+4	LSB
I_{ADC}	A/D Converter Additional Current for ADC Enable	3V	No load, t_{AD}=0.5μs	—	0.5	1	mA
		5V	No load, t_{AD}=0.5μs	—	0.6	1.2	mA
t_{AD}	A/D Converter Clock Period	—	—	0.5	—	—	μs
t_{ADC}	A/D Converter Conversion Time	—	—	—	16	—	t_{AD}

Memory Characteristics

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

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{RW}	V_{DD} for Read / Write	—	—	V_{DDmin}	—	V_{DDmax}	V
Flash Program /Data EEPROM Memory							
t_{DEW}	Write Cycle Time – Data EEPROM Memory	—	—	—	4	6	ms
E_P	Program Memory Cell Endurance	—	—	10K	—	—	E/W
	Data EEPROM Memory Cell Endurance	—	—	100K	—	—	E/W
t_{RETD}	ROM Data Retention Time	—	Ta = 25°C	—	40	—	Year
RAM Data Memory							
V_{DR}	RAM Data Retention Voltage	—	Device in SLEEP Mode	1.0	—	—	V

Note: “E/W” means Erase/Write times.

Comparator Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{CP}	Operating Voltage	—	—	3.0	—	5.5	V
V_{R1}	Reference Voltage for Comparator 1	5V	Ta=-40°C~85°C	-5%	$0.6V_{DD}$~$0.775V_{DD}$, $0.025V_{DD}$/step	+5%	V
V_{R2}	Reference Voltage for Comparator 2	5V	Ta=-40°C~85°C	-5%	$0.6V_{DD}$~$0.775V_{DD}$, $0.025V_{DD}$/step	+5%	V
V_{R3}	Reference Voltage for Comparator 2	5V	Ta=-40°C~85°C	-5%	$0.075V_{DD}$~$0.25V_{DD}$, $0.025V_{DD}$/step	+5%	V
V_{R4}	Reference Voltage for Comparator 3	5V	Ta=-40°C~85°C	-5%	$0.6V_{DD}$~$0.775V_{DD}$, $0.025V_{DD}$/step	+5%	V
V_{OS}	Analog Comparator Input Offset Voltage	5V	By calibration	-2	—	+2	mV
			Without calibration. CnCOF[4:0]=10000B	-15	—	+15	mV
V_{CM}	Analog Comparator Common Mode Voltage Range	—	—	0	—	V_{DD}-1.4V	V
t_{PD}	Analog Comparator Response Time	—	Analog comparator hysteresis disable and with 10mV overdrive	—	—	2	μs
V_{HYS}	Analog Comparator Hysteresis Width	5V	Analog comparator hysteresis enable	20	40	60	mV

Operational Amplifier Characteristics

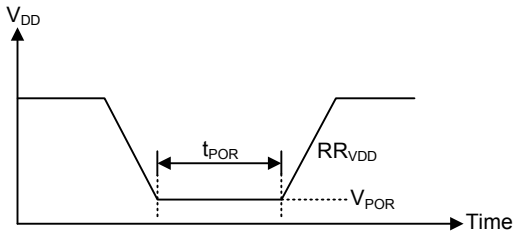
Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{OP}	Operating Voltage	—	—	3.0	—	5.5	V
V_{OPOS1}	Input Offset Voltage	5V	Without calibration, OPAOF[4:0]=10000B	-15	—	+15	mV
V_{OPOS2}	Input Offset Voltage	5V	By calibration	-2	—	+2	mV
V_{CM}	Common Mode Voltage Range	—	—	V_{SS}	—	V_{DD}-1.4V	V
PSRR	Power Supply Rejection Ratio	5V	—	60	80	—	dB
CMRR	Common Mode Rejection Ratio	5V	V_{CM}=0~V_{DD}-1.4V	60	80	—	dB
SR	Slew Rate+, Slew Rate-	5V	No load	0.6	1.8	—	V/μs
GBW	Gain Band Width	5V	R_L=1MΩ, C_L=100pF	0.6	2.2	—	MHz
OPAR1	Typical Value	5V	Ta=25°C	45	60	75	kΩ
	Temperature Drift Percentage Relative to 25°C	5V	Ta=0~50°C	-3	—	+3	%
OPAR2	Typical Value	5V	Ta=25°C	0.75	1	1.25	kΩ
	Temperature Drift Percentage Relative to 25°C	5V	Ta=0~50°C	-3	—	+3	%
OPAR3	Typical Value	5V	Ta=25°C	0.75	1	1.25	kΩ
	Temperature Drift Percentage Relative to 25°C	5V	Ta=0~50°C	-3	—	+3	%
OPAR4	Typical Value	5V	Ta=25°C	—	10	—	kΩ

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

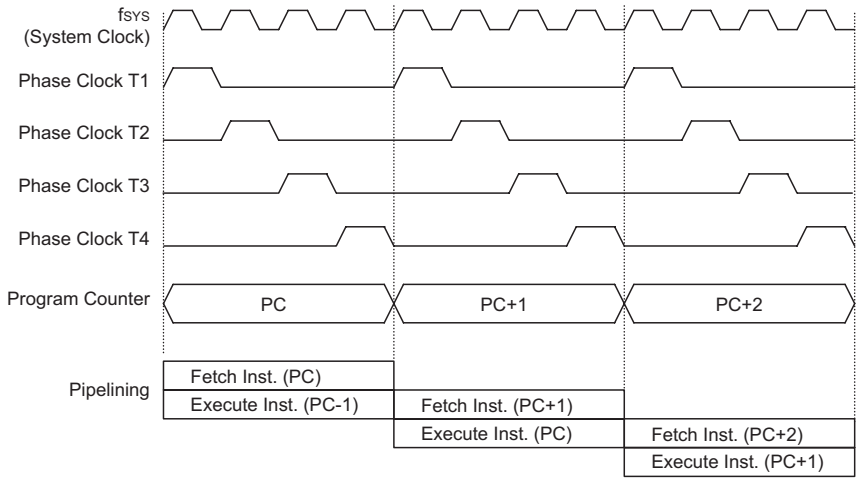


System Architecture

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

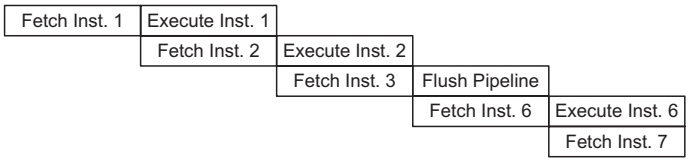
The main system clock, derived from either a Crystal/Resonator or RC oscillator is subdivided into four internally generated non-overlapping clocks, T1~T4. The Program Counter is incremented at the beginning of the T1 clock during which time a new instruction is fetched. The remaining T2~T4 clocks carry 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  DELAY:      NOP
```



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.

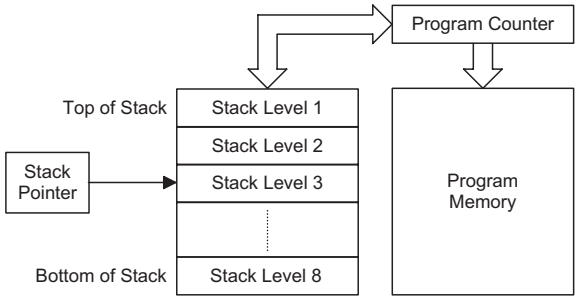
Program Counter	
Program Counter High Byte	**PCL Register**
PC11~PC8	PCL7~PCL0

Program Counter

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

Stack

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



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

Arithmetic and Logic Unit – ALU

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

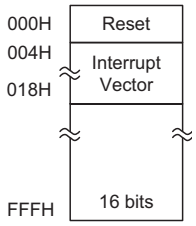
- Arithmetic operations: ADD, ADDM, ADC, ADCM, SUB, SUBM, SBC, SBCM, DAA
- Logic operations: AND, OR, XOR, ANDM, ORM, XORM, CPL, CPLA
- Rotation RRA, RR, RRCA, RRC, RLA, RL, RLCA, RLC
- Increment and Decrement INCA, INC, DECA, DEC
- Branch decision, JMP, SZ, SZA, SNZ, SIZ, SDZ, SIZA, SDZA, CALL, RET,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

Flash Program Memory

The Program Memory is the location where the user code or program is stored. For this device the Program Memory is Flash type, which means it can be programmed and re-programmed a large number of times, allowing the user the convenience of code modification on the same device. By using the appropriate programming tools, the Flash device 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 4K×16 bits. The Program Memory is addressed by the Program Counter and also contains data, table information and interrupt entries. Table data, which can be setup in any location within the Program Memory, is addressed by a separate table pointer register.



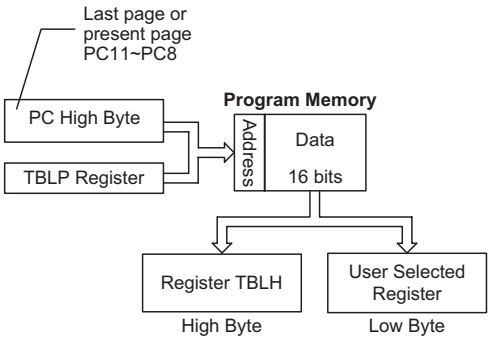
Program Memory Structure

Look-up Table

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



Instruction	Table Location Bits											
	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
TABRDC [m]	PC11	PC10	PC9	PC8	@7	@6	@5	@4	@3	@2	@1	@0
TABRDL [m]	1	1	1	1	@7	@6	@5	@4	@3	@2	@1	@0

Table Location

Note: 1. b11~b0: Table location bits

2. PC11~PC8: Current Program Counter bits

3. @7~@0: Table pointer (TBLP) bits

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 "0F00H" which refers to the start address of the last page within the 4K 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 "0F06H" or 6 locations after the start of the last page. Note that the value for the table pointer is referenced to the first address of the present page if the "TABRD [m]" instruction is being used. The high byte of the table data which in this case is equal to zero will be transferred to the TBLH register automatically when the "TABRDC [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 present page
:
:
tabrdl tempreg1        ; transfers value in table referenced by table pointer data at
                       ; program memory address 0F06H transferred to
                       ; tempreg1 and TBLH
dec tblp               ; reduce value of table pointer by one
tabrdl tempreg2        ; transfers value in table referenced by table pointer data at
                       ; program memory address 0F05H transferred to tempreg2 and TBLH
                       ; in this example the data 1AH is transferred to tempreg1 and data
                       ; 0FH to register tempreg2
:
org 0F00h              ; sets initial address of program memory
dc 00Ah, 00Bh, 00Ch, 00Dh, 00Eh, 00Fh, 01Ah, 01Bh
:
```

In Circuit Programming – ICP

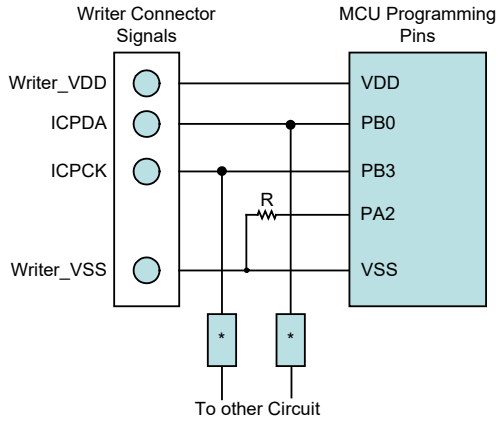
The provision of Flash type Program Memory provides the user with a means of convenient and easy upgrades and modifications to their programs on the same device. As an additional convenience, Holtek has provided a means of programming the microcontroller in-circuit using a 4-pin interface. This provides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

The Holtek Flash MCU to Writer Programming Pin correspondence table is as follows:

Holtek Writer Pins	MCU Programming Pins	Pin Description
ICPDA	PB0	Programming Serial Data/Address
ICPCK	PB3	Programming Clock
VDD	VDD	Power Supply
VSS	VSS	Ground

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

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



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

2. A Resistor with a value less than 1MΩ is recommended to be connected between PA2 and VSS pins during programming.

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 sections, the first of these is an area of RAM, known as the Special Function Data Memory. Here are located registers which are necessary for correct operation of the device. Many of these registers can be read from and written to directly under program control, however, some remain protected from user manipulation. The second area of Data Memory is known as the General Purpose Data Memory, which is reserved for general purpose use. All locations within this area are read and write accessible under program control.

The two sections of Data Memory, the Special Purpose and General Purpose Data Memory are located at consecutive locations. All are implemented in RAM and are 8 bits wide but the length of each memory section is dictated by the type of microcontroller chosen. The start address of the Data Memory for all devices is the address "00H". The overall Data Memory is subdivided into two banks, Bank 0 and Bank 1.

All microcontroller programs require an area of read/write memory where temporary data can be stored and retrieved for use later. It is this area of RAM memory that is known as General Purpose Data Memory. This area of Data Memory is fully accessible by the user program for both read and write operations. By using the "SET [m].i" and "CLR [m].i" instructions individual bits can be set or reset under program control giving the user a large range of flexibility for bit manipulation in the Data Memory.

Capacity	Bank 0	Bank 1
208×8	40H~FFH	40H~4FH

Note: Most of the Data Memory bits can be directly manipulated using the "SET [m].i" and "CLR [m].i" with the exception of a few dedicated bits. The Data Memory can also be accessed through the memory pointer registers.

Special Purpose Data Memory

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

	Bank0	Bank1
00H	IAR0	
01H	MP0	
02H	IAR1	
03H	MP1	
04H	BP	
05H	ACC	
06H	PCL	
07H	TBLP	
08H	TBLH	
09H	Unused	
0AH	STATUS	
0BH	INTC0	
0CH	Unused	
0DH	TMR0	
0EH	TMR0C	
0FH	Unused	
10H	TMR1	
11H	TMR1C	
12H	PA	
13H	PAC	
14H	PB	
15H	PBC	
16H	PC	
17H	PCC	
18H	Unused	
19H	Unused	
1AH	CTRL0	
1BH	CMP0C	
1CH	CMP1C	
1DH	CMP2C	
1EH	INTC1	
1FH	Unused	

	Bank0	Bank1
20H	PPGC	
21H	PPGTA	
22H	PPGTB	
23H	PPGTEX	
24H	ADRL	
25H	ADRH	
26H	ADCR	
27H	ACSR	
28H	CTRL1	
29H	OPAC	
2AH	CTRL2	
2BH	PWLT	
2CH	CTRL3	
2DH	TMR2	
2EH	TMR2C	
2FH	CMPSC	
30H	IICC0	Unused
31H	IICC1	
32H	IICD	
33H	IICA	
34H	IICTOC	
35H	Unused	
36H	PWM	
37H	PWMC	
38H	CMP3C	
39H	CTRL4	
3AH	MFI	
3BH	PCRL	
3CH	PCRH	
3DH	EEA	
3EH	EED	
3FH		EEC

: Unused, read as 00H

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	—	—	—	—	—	—	—	DMBP
R/W	—	—	—	—	—	—	—	R/W
POR	—	—	—	—	—	—	—	0

Bit 7~1 Unimplemented, read as "0"

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

Accumulator – ACC

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

Program Counter Low Register – PCL

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

Look-up Table Registers – TBLP, TBLH

These three special function registers are used to control operation of the look-up table which is stored in the Program Memory. TBLP is the table pointer and indicates the location where the table data is located. Its value must be setup before any table read commands are executed. Its 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.

System Control Registers – CTRL0, CTRL1, CTRL2, CTRL3, CTRL4

These registers are used to provide control over various internal functions. Some of these include the inverting control of the comparator output signal, TMR0 external clock source selection, Peripheral clock prescaler stage, Oscillator stop function control, internal reference voltage selection of the comparator, etc.

- **CTRL0 Register**

Bit	7	6	5	4	3	2	1	0
Name	C2VOINV	TMR0ECS	PCKPSC2	PCKPSC1	PCKPSC0	OSTPC	LVDO	LVDC
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W
POR	0	0	0	0	0	0	0	0

Bit 7 **C2VOINV**: Inverting control of the comparator 2 output signal
0: Non-inverted
1: Inverted

Bit 6 **TMR0ECS**: Select TMR0 external clock source
0: TMR0 pin
1: INT0

Bit 5~3 **PCKPSC2~PCKPSC0**: Peripheral clock prescaler stage
f_{PCK}=
000: $f_{SYS}/4$
001: $f_{SYS}/8$
010: $f_{SYS}/12$
011: $f_{SYS}/16$
100: $f_{SYS}/20$
101: $f_{SYS}/24$
110: $f_{SYS}/1024$
111: $f_{SYS}/2048$

Bit 2 **OSTPC**: Oscillator stop function control
0: Disable
1: Enable

Bit 1 **LVDO**: Low voltage detector output
0: Normal voltage
1: Low voltage detected

Bit 0 **LVDC**: Low voltage detector control
0: Disable
1: Enable

- **CTRL1 Register**

Bit	7	6	5	4	3	2	1	0
Name	INTINV	INTS	DBC5	DBC4	DBC3	DBC2	DBC1	DBC0
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 **INTINV**: Inverting control of the de-bounced external interrupt input signal
0: Non-inverted
1: Inverted

Bit 6 **INTS**: Select external interrupt source
0: $\overline{INT}$
1: Comparator 0 output "C0VO"

Bit 5~0 **DBC5~DBC0**: Select external interrupt input de-bounce time (f_{SYS}=8MHz)
000000: bypass digital de-bounce circuit
000001: 0~1/f_{SYS}, about 0.125μs
000010: 1/f_{SYS}~2/f_{SYS}, about 0.25μs
:
:
101111: 46/f_{SYS}~47/f_{SYS}, about 5.875μs
11xxxx: 47/f_{SYS}~48/f_{SYS}, about 6μs

- **CTRL2 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	CVREF5	CVREF4	CVREF3	CVREF2	CVREF1	CVREF0
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 5~3 **CVREF5~CVREF3**: Select internal reference voltage V_{R2} for comparator 2
000: 0.600V_{DD}
001: 0.625V_{DD}
010: 0.650V_{DD}
011: 0.675V_{DD}
100: 0.700V_{DD}
101: 0.725V_{DD}
110: 0.750V_{DD}
111: 0.775V_{DD}

Bit 2~0 **CVREF2~CVREF0**: Select internal reference voltage V_{R1} for comparator 1
000: 0.600V_{DD}
001: 0.625V_{DD}
010: 0.650V_{DD}
011: 0.675V_{DD}
100: 0.700V_{DD}
101: 0.725V_{DD}
110: 0.750V_{DD}
111: 0.775V_{DD}

- **CTRL3 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	C3VOINV	CVREF11	CVREF10	CVREF9	CVREF8	CVREF7	CVREF6
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 **C3VOINV**: Inverting control of the comparator 3 output signal
0: Non-inverted
1: Inverted

Bit 5~3 **CVREF11~CVREF9**: Select internal reference voltage V_{R4} for comparator 3
000: $0.600V_{DD}$
001: $0.625V_{DD}$
010: $0.650V_{DD}$
011: $0.675V_{DD}$
100: $0.700V_{DD}$
101: $0.725V_{DD}$
110: $0.750V_{DD}$
111: $0.775V_{DD}$

Bit 2~0 **CVREF8~CVREF6**: Select internal reference voltage V_{R3} for comparator 2
000: $0.075V_{DD}$
001: $0.100V_{DD}$
010: $0.125V_{DD}$
011: $0.150V_{DD}$
100: $0.175V_{DD}$
101: $0.200V_{DD}$
110: $0.225V_{DD}$
111: $0.250V_{DD}$

- **CTRL4 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	PPGDL5	PPGDL4	PPGDL3	PPGDL2	PPGDL1	PPGDL0
R/W	—	—	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	—	0	0	0	0	0	0

Bit 7~6 Unimplemented, read as "0"

Bit 5~0 **PPGDL5~PPGDL0**: Select trigger delay for PPG (f_{SYS}=8MHz)
000000: 0μs
000001: 0.125μs
000010: 0.25μs
:
:
101111: 5.875μs
11xxxx: 6μs

Note: 1. Trigger delay means the time of the falling edge of INT0S to PPG trigger signal (INT00) being sent.
2. Falling edge of INT0S during trigger delay period will be ignored.
INT0: Inverted or non-inverted de-bounce signal from $\overline{INT}$ or comparator 0 output "C0VO" by software option;
INT0S: Signal represents single or double falling edge of INT0
INT00: PPG hardware trigger signal

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.

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 3FH in Bank 1, the MP1 Memory Pointer must first be set to the value 3FH and the Bank Pointer register, BP, set to the value, 01H, before any operations on the EEC register are executed.

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

EEPROM Register 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. When an EEPROM write cycle ends, the DEF request flag will be set. If the global, EEPROM interrupts are enabled and the stack is not full, a jump to the associated Interrupt vector will take place. When the EEPROM Interrupt is serviced, the DEF flag will be automatically cleared and the EMI bit will be automatically cleared to disable other interrupts.

Programming Considerations

Care must be taken that data is not inadvertently written to the EEPROM. Protection can be 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 Power Down 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, 03FH              ; 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, 03FH              ; 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
```

Oscillators

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

Oscillator Overview

There are three Oscillators, including an external Crystal/Resonator oscillator, an internal RC oscillator and a WDT oscillator. In addition to being the source of the main system clock the external Crystal/Resonator oscillator and internal RC oscillator also provide clock sources for the Watchdog Timer. The external oscillator requiring some external components as well as the fully integrated internal oscillator, requiring no external components.

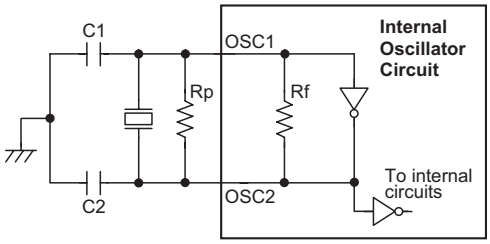
System Clock Configurations

There are two methods of generating the system clock, an external Crystal/Resonator oscillator and an internal RC oscillator. They are selected by the configuration options. No matter what oscillator is selected, their signals can be as the system clock. The system oscillator can be stopped in HALT state and will ignore any external signal to reduce the power consuming.

External High Speed Crystal Oscillator – HXT

The simple connection of a crystal across OSC1 and OSC2 will create the necessary phase shift and feedback for oscillation. However, for some crystals and most resonator types, to ensure oscillation and accurate frequency generation, it is necessary to add two small value external capacitors, C1 and C2. The exact values of C1 and C2 should be selected in consultation with the crystal or resonator manufacturer's specification.

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

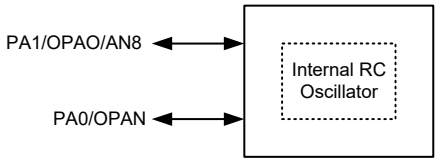


Note: 1. Rp is normally not required. C1 and C2 are required.
2. Although not shown OSC1/OSC2 pins have a parasitic capacitance of around 7pF.

Crystal/Resonator Oscillator – HXT

Internal RC Oscillator – HIRC

The internal RC oscillator is a fully integrated system oscillator requiring no external components. The internal RC oscillator has a fixed frequency of 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. Note that if this internal system clock option is selected, as it requires no external pins for its operation, I/O pins PA0 and PA1 are free for use as normal I/O pins.



Internal RC Oscillator – HIRC

WDT Oscillator – WDTOSC

The WDT oscillator is a free running on-chip RC oscillator, and no external components are required. Even if the system enters the power down mode, the system clock is stopped, but the WDT oscillator still works with a period of approximately 65μs at 5V. The WDT oscillator can be disabled by configuration option to conserve power.

Oscillator Stop Detection

The oscillator stop detection function is used to monitor the oscillator and force the device into the reset state if the oscillator fails. The reset state is maintained until the oscillator is working again. The oscillator stop detection function can be enabled by setting the OSTPC bit in the CTRL0 register. In the power-down mode, the oscillator stop detection function is disabled.

Power Down Mode and Wake-up

Power Down Mode

All of the Holtek microcontrollers have the ability to enter a Power Down Mode. When the device enters this mode, the normal operating current, will be reduced to an extremely low standby current level. This occurs because when the device enters the Power Down Mode, the system oscillator is stopped which reduces the power consumption to extremely low levels, however, as the device maintains its present internal condition, it can be woken up at a later stage and continue running, without requiring a full reset. This feature is extremely important in application areas where the MCU must have its power supply constantly maintained to keep the device in a known condition but where the power supply capacity is limited such as in battery applications.

Entering the Power Down Mode

There is only one way for the device to enter the Power Down Mode and that is to execute the "HALT" instruction in the application program. When this instruction is executed, the following will occur:

- The system oscillator will stop running and the application program will stop at the "HALT" instruction.
- The Data Memory contents and registers will maintain their present condition
- The WDT will be cleared and resume counting if the WDT clock source is selected to come from the WDT oscillator. The WDT will stop if its clock source originates from the system clock.
- The I/O ports will maintain their present condition.
- In the status register, the Power Down flag, PDF, will be set and the Watchdog time-out flag, TO, will be cleared.

Standby Current Considerations

As the main reason for entering the Power Down Mode is to keep the current consumption of the MCU to as low a value as possible, perhaps only in the order of several micro-amps, there are other considerations which must also be taken into account by the circuit designer if the power consumption is to be minimized. 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 undonbed pins, which must either be setup as outputs or if setup as inputs must have pull-high resistors connected. Care must also be taken with the loads, which are connected to I/O pins, which are setup as outputs. These should be placed in a condition in which minimum current is drawn or connected only to external circuits that do not draw current, such as other CMOS inputs. Also note that additional standby current will also be required if the configuration options have enabled the Watchdog Timer internal oscillator.

Wake-up

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

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

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

Each pin on Port A can be setup via an individual configuration option 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 to "1" before entering the Power Down Mode, the wake-up function of the related interrupt will be disabled.

No matter what the source of the wake-up event is, once a wake-up situation occurs, a time period equal to 1024 system clock periods will be required before normal system operation resumes. However, if the wake-up has originated due to an interrupt, the actual interrupt subroutine execution will be delayed by an additional one or more cycles. If the wake-up results in the execution of the next instruction following the "HALT" instruction, this will be executed immediately after the 1024 system clock period delay has ended.

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 WDT oscillator or $f_{SYS}/4$ clock, selected by configuration option. The WDT oscillator has an approximate period of 65μs at a supply voltage of 5V. Note that the oscillation frequency can vary with V_{DD}, temperature and process variations. Another clock source is supplied by $f_{SYS}/4$. The Watchdog Timer source clock is then subdivided by a ratio of 2^{13} to 2^{16} to give longer timeouts, the actual value being chosen using the configuration option.

Note that the LIRC (WDT OSC) is running during MCU normal mode. Once the device entering power down mode, the LIRC will keep running if the WDT is enabled and stop if the WDT is disabled.

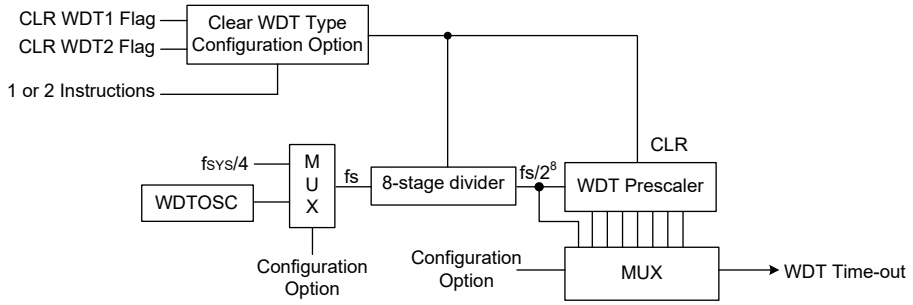
Watchdog Timer Operation

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. Most of the Watchdog Timer options, such as enable/disable, Watchdog Timer clock source and clear instruction type are selected using configuration options. There are not any related register. Note that if the Watchdog Timer configuration option has been disabled, then any instruction relating to its operation will result in no operation.

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 Power Down Mode, when a Watchdog Timer time-out occurs, the TO bit in the status register will be set and only the Program Counter and Stack Pointer will be reset. Three methods can be adopted to clear the contents of the Watchdog Timer. The first is an external hardware reset, which means a low level on the $\overline{RES}$ pin, the second is using the watchdog software instructions and the third is via a "HALT" instruction.

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

The maximum time out period is when the 2^{16} division ratio is selected. As an example, with the WDT oscillator as its source clock, this will give a maximum watchdog period of around 4.7μs for the 2^{16} division ratio. If the device operates in a noisy environment, using the WDT oscillator is strongly recommended, since the system clock will be stopped under the power down mode.



Watchdog Timer

Reset and Initialisation

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

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

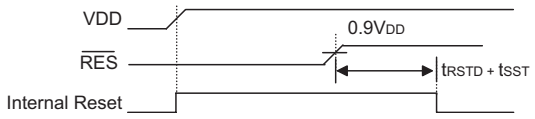
Another type of reset is when the Watchdog Timer overflows and resets. All types of reset operations result in different register conditions being setup. Another reset exists in the form of a Low Voltage Reset, LVR, where a full reset, similar to the $\overline{RES}$ reset is implemented in situations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

Reset Functions

There are five ways in which a 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.



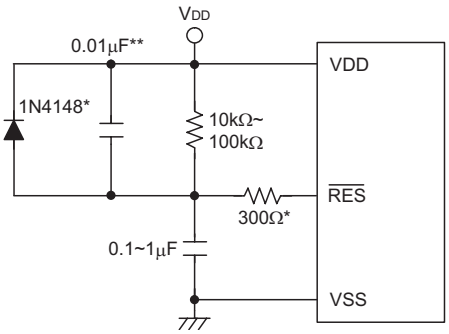
Power-On Reset Timing Chart

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

The $\overline{\text{RES}}$ pin is pin-shared with PB0, the reset function must be selected by the configuration options. Although the microcontroller has an internal RC reset function, if the V_{DD} power supply rise time is not fast enough or does not stabilise quickly at power-on, the internal reset function may be incapable of providing proper reset operation. For this reason it is recommended that an external RC network is connected to the $\overline{\text{RES}}$ pin, whose additional time delay will ensure that the $\overline{\text{RES}}$ pin remains low for an extended period to allow the power supply to stabilise. During this time delay, normal operation of the microcontroller will be inhibited. After the $\overline{\text{RES}}$ line reaches a certain voltage value, the reset delay time t_{RSTD} is invoked to provide an extra delay time after which the microcontroller will begin normal operation. The abbreviation SST in the figures stands for System Start-up Timer.

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

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

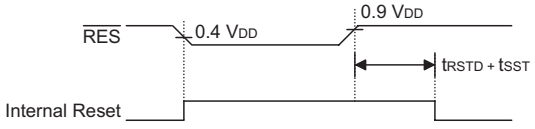


Note: "*" It is recommended that this component is added for added ESD protection.
"**" It is recommended that this component is added in environments where power line noise is significant.

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

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

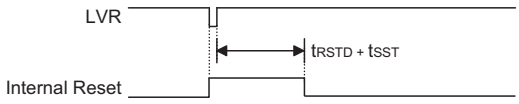
Pulling the $\overline{\text{RES}}$ Pin low using external hardware will also execute devices reset. In this case, as in the case of other resets, the Program Counter will reset to zero and program execution initiated from this point.



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

Low Voltage Reset – LVR

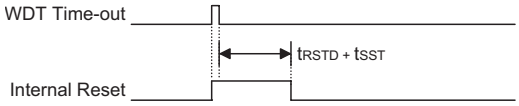
The microcontroller contains a low voltage reset circuit in order to monitor the supply voltage of the device. The LVR function is always enabled with a specific LVR voltage V_{LVR}. If the supply voltage of the device drops to within a range of 0.9V~V_{LVR} such as might occur when changing the battery, the LVR will automatically reset the device internally. 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 2.1V or 3.0V which is selected by the configuration options. Note that the LVR function will be automatically disabled when the device enters the power down mode.



Low Voltage Reset Timing Chart

Watchdog Time-out Reset during Normal Operation

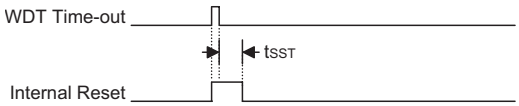
The Watchdog time-out Reset during normal operation is the same as a hardware $\overline{RES}$ pin reset except that the Watchdog time-out flag TO will be set high.



WDT Time-out Reset during Normal Operation Timing Chart

Watchdog Time-out Reset during Power Down Mode

The Watchdog time-out Reset during Power Down 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 Power Down 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 Power Down Mode function or Watchdog Timer. The reset flags are shown in the table:

TO	PDF	Reset Conditions
0	0	Power-on reset
u	u	$\overline{RES}$ or LVR reset during Normal Mode operation
1	u	WDT time-out reset during Normal Mode operation
1	1	WDT time-out reset during Power Down 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	All Timer/Counters will be turned off
PPG Counter	The PPG Counter will be stopped
PPG, IOFF Output	Floating
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)	$\overline{\text{RES}}$ Reset or LVR Reset	WDT Time-out (Normal Operation)	WDT Time-out (HALT)
MP0	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
MP1	xxxx xxxx	uuuu uuuu	uuuu uuuu	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
STATUS	--00 xxxx	--uu uuuu	--1u uuuu	--11 uuuu
INTC0	-000 0000	-000 0000	-000 0000	-uuu uuuu
TMR0	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMR0C	00-0 1000	00-0 1000	00-0 1000	uu-u uuuu
TMR1	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMR1C	00-0 -000	00-0 -000	00-0 -000	uu-u -uuu
PA	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAC	1111 1111	1111 1111	1111 1111	uuuu uuuu
PB	---1 1111	---1 1111	---1 1111	---u uuuu
PBC	---1 1111	---1 1111	---1 1111	---u uuuu
PC	---- 1111	---- 1111	---- 1111	---- uuuu
PCC	---- 1111	---- 1111	---- 1111	---- uuuu
CTRL0	0000 0000	0000 0000	0000 0000	uuuu uuuu
CMP0C	0001 0000	0001 0000	0001 0000	uuuu uuuu
CMP1C	0001 0000	0001 0000	0001 0000	uuuu uuuu
CMP2C	0001 0000	0001 0000	0001 0000	uuuu uuuu
CMP3C	0001 0000	0001 0000	0001 0000	uuuu uuuu
INTC1	0000 0000	0000 0000	0000 0000	uuuu uuuu
MFI	-000 -000	-000 -000	-000 -000	-uuu -uuu
PPGC	0000 --00	0000 --00	0000 --00	uuuu --uu
PPGTA	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
PPGTB	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
PPGTEX	---x ---x	---x ---x	---x ---x	---u ---u
ADRL	xxxx ----	xxxx ----	xxxx ----	uuuu ----
ADRH	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu

Register	Reset (Power On)	$\overline{\text{RES}}$ Reset or LVR Reset	WDT Time-out (Normal Operation)	WDT Time-out (HALT)
ADCR	01-- 0000	01-- 0000	01-- 0000	uu-- uuuu
ACSR	---- --00	---- --00	---- --00	---- --uu
CTRL1	0000 0000	0000 0000	0000 0000	uuuu uuuu
CTRL2	--00 0000	--00 0000	--00 0000	--uu uuuu
CTRL3	-000 0000	-000 0000	-000 0000	-uuu uuuu
OPAC	0001 0000	0001 0000	0001 0000	uuuu uuuu
PWLT	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
CTRL4	--00 0000	--00 0000	--00 0000	--uu uuuu
TMR2	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMR2C	---0 -000	---0 -000	---0 -000	---u -uuu
CMPSC	0000 --00	0000 --00	0000 --00	uuuu --uu
IICC0	---- 000-	---- 000-	---- 000-	---- uuu-
IICC1	1000 0001	1000 0001	1000 0001	uuuu uuuu
IICD	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
IICA	0000 000-	0000 000-	0000 000-	uuuu uuu-
IICTOC	0000 0000	0000 0000	0000 0000	uuuu uuuu
PCRL	0000 0000	0000 0000	0000 0000	uuuu uuuu
PCRH	---- 0000	---- 0000	---- 0000	---- uuuu
PWM	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx
PWMC	---- -000	---- -000	---- -000	---- -uuu
EEA	---0 0000	---0 0000	---0 0000	---u uuuu
EED	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. Most pins can have either an input or output designation under user program control. Additionally, as there are pull-high resistors and wake-up configuration options, the user is provided with an I/O structure to meet the needs of a wide range of application possibilities.

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
PB	—	—	—	PB4	PB3	PB2	PB1	PB0
PBC	—	—	—	PBC4	PBC3	PBC2	PBC1	PBC0
PC	—	—	—	—	PC3	PC2	PC1	PC0
PCC	—	—	—	—	PCC3	PCC2	PCC1	PCC0

I/O Register List

Pull-high Resistors

Many product applications require pull-high resistors for their switch inputs usually requiring the use of an external resistor. To eliminate the need for these external resistors, when configured as an input have the capability of being connected to an internal pull-high resistor. These pull-high resistors are selectable via configuration options. The pull-high resistors are implemented using weak PMOS transistors.

Port A Wake-up

The HALT instruction forces the microcontroller into the Power Down 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. After a HALT instruction forces the microcontroller into entering the Power Down Mode, the processor will remain in a low-power state until the logic condition of the selected wake-up pin on Port A changes 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 by configuration options to have this wake-up feature.

I/O Port Control Registers

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

- **PAC Register**

Bit	7	6	5	4	3	2	1	0
Name	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	—	—	—	PBC4	PBC3	PBC2	PBC1	PBC0
R/W	—	—	—	R/W	R/W	R/W	R/W	R/W
POR	—	—	—	1	1	1	1	1

Bit 7~5 Unimplemented, read as "0"

Bit 4~0 **PBC4~PBC0**: Port B bit 4 ~ bit 0 Input/Output Control
0: Output
1: Input

- **PCC Register**

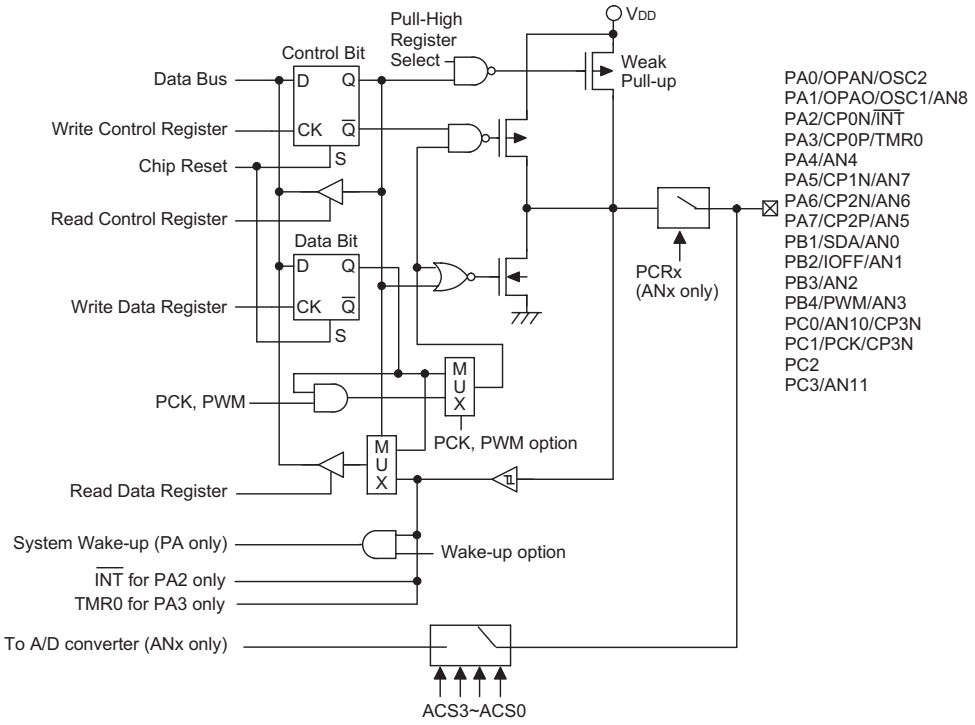
Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	PCC3	PCC2	PCC1	PCC0
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	1	1	1	1

Bit 7~4 Unimplemented, read as "0"

Bit 3~0 **PCC3~PCC0**: Port C bit 3 ~ bit 0 Input/Output Control
0: Output
1: Input

I/O Pin Structures

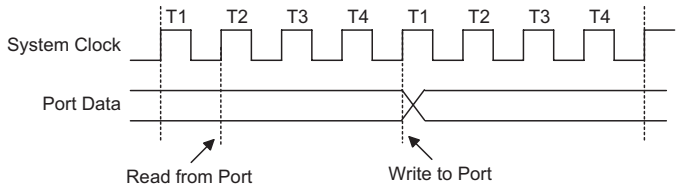
The accompanying diagrams illustrate the internal structures of some generic I/O pin types. As the exact logical construction of the I/O pin will differ from these drawings, they are supplied as a guide only to assist with the functional understanding of the I/O pins.



Generic 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~PCC, are then programmed to setup some pins as outputs, these output pins will have an initial high output value unless the associated port data registers, PA~PC, are first programmed. Selecting which pins are inputs and which are outputs can be achieved byte-wide by loading the correct values into the appropriate port control register or by programming individual bits in the port control register using the "SET [m].i" and "CLR [m].i" instructions. Note that when using these bit control instructions, a read-modify-write operation takes place. The microcontroller must first read in the data on the entire port, modify it to the required new bit values and then rewrite this data back to the output ports.



Read Modify Write Timing

Port A has the additional capability of providing wake-up functions. When the device is in the Power Down 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 three count-up timer of 8-bit capacity. The Timer/Event Counter 0 has three different operating modes, it can be configured to operate as a general timer, an external event counter or as a pulse width capture device. The Timer/Event Counter 1 has only one operating mode – the PPG non-retriggered function. The Timer/Event Counter 2 has only one operating mode – timer mode. The provision of an internal prescaler to the clock circuitry on gives added range to the timers.

There are two types of registers related to the Timer/Event Counters. The first are the registers that contain the actual value of the timer and into which an initial value can be preloaded. Reading from these registers retrieves the contents of the Timer/Event Counter. The second type of associated registers are the Timer Control Registers which define the timer options and determines how the timers are to be used.

Configuring the Timer/Event Counter Input Clock Source

For the Timer/Event Counter 0, the clock source can be from an external source or an internal clock source, when selecting an external source, the choice of which is determined by the TMR0ECS bit in the CTRL0 register, it can be from the TMR0 pin or the INT0 signal. For the Timer/Event Counter 1/2, there is only one clock source, f_{SYS}, which is also divided by a prescaler, the division ratio is conditioned by the Timer Control Register bits TnPSC0~TnPSC2. The external clock input allows the user to count external events, measure time internals or pulse widths. While using the internal clock allows the user to generate an accurate time base.

Timer/Event Counter Registers – TMR0, TMR1, TMR2

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. 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.

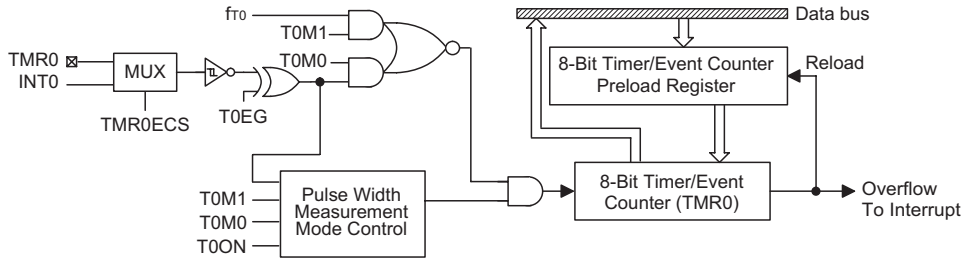
Note that to achieve a maximum full range count of FFH, all the preload registers must first be cleared to zero. It should be noted that after power-on, the preload registers will be in an unknown condition. Note that if the Timer/Event Counter is in an OFF condition and data is written to its preload 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.

Timer Control Registers – TMR0C, TMR1C, TMR2C

The flexible features of the Holtek microcontroller Timer/Event Counters enable them to operate in several 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.

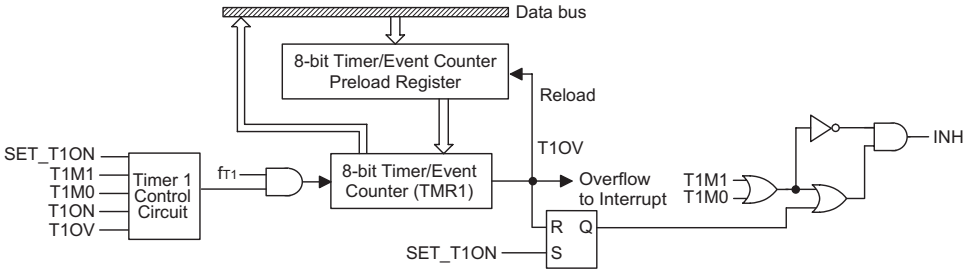
To choose which of the several modes the timer is to operate in, either in the timer mode, the event

counting mode, the pulse width capture mode or the PPG non-retriggered function mode 0, bits 7 and 6 of the Timer Control Register, which are known as the bit pair T0M1/T0M0 or T1M1/T1M0, 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 TMR0C which is known as T0EG.

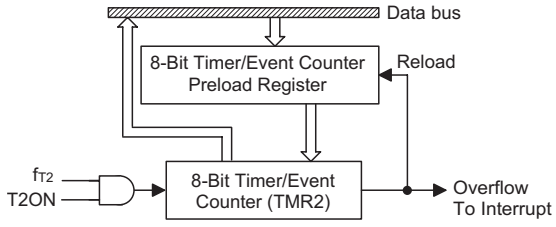


Note: INT0 is inverted or non-inverted de-bounce signal from $\overline{INT}$ pin or comparator 0 output "C0VO" by software option

8-bit Timer/Event Counter 0



8-bit Timer/Event Counter 1



8-bit Timer/Event Counter 2

- **TMR0C Register**

Bit	7	6	5	4	3	2	1	0
Name	T0M1	T0M0	—	T0ON	T0EG	T0PSC2	T0PSC1	T0PSC0
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 **T0M1, T0M0**: Timer 0 operation mode selection
00: No mode available
01: Event counter mode
10: Timer mode
11: Pulse width mearurement mode

Bit 5 Unimplemented, read as "0"

Bit 4 **T0ON**: Timer/event counter counting enable
0: Disable
1: Enable

Bit 3 **T0EG**: Event counter active edge selection
In Event Counter Mode:
0: Count on rising edge
1: Count on falling edge
In Pulse Width measurement mode:
0: Start counting on the falling edge, stop on the rising edge
1: Start counting on the rising edge, stop on the falling edge

Bit 2~0 **T0PSC2~T0PSC0**: Timer prescaler rate selection
Timer internal clock=
000: f_{SYS}
001: $f_{SYS}/2$
010: $f_{SYS}/4$
011: $f_{SYS}/8$
100: $f_{SYS}/16$
101: $f_{SYS}/32$
110: $f_{SYS}/64$
111: $f_{SYS}/128$

- **TMR1C Register**

Bit	7	6	5	4	3	2	1	0
Name	T1M1	T1M0	—	T1ON	—	T1PSC2	T1PSC1	T1PSC0
R/W	R/W	R/W	—	R/W	—	R/W	R/W	R/W
POR	0	0	—	0	—	0	0	0

Bit 7~6 **T1M1, T1M0**: Timer 1 operation mode selection
00: Mode 0 (PPG non-retriggered function)
01: No mode available
10: No mode available
11: No mode available

Bit 5 Unimplemented, read as "0"

Bit 4 **T1ON**: Timer/event counter counting enable
0: Disable
1: Enable

Bit 3 Unimplemented, read as "0"

Bit 2~0 **T1PSC2~T1PSC0**: Timer prescaler rate selection
Timer internal clock f_{T1}=
000: f_{SYS}
001: $f_{SYS}/2$
010: $f_{SYS}/4$
011: $f_{SYS}/8$
100: $f_{SYS}/16$
101: $f_{SYS}/32$
110: $f_{SYS}/64$
111: $f_{SYS}/128$

- **TMR2C Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	T2ON	—	T2PSC2	T2PSC1	T2PSC0
R/W	—	—	—	R/W	—	R/W	R/W	R/W
POR	—	—	—	0	—	0	0	0

Bit 7~5 Unimplemented, read as "0"

Bit 4 **T2ON**: Timer/event counter counting enable
0: Disable
1: Enable

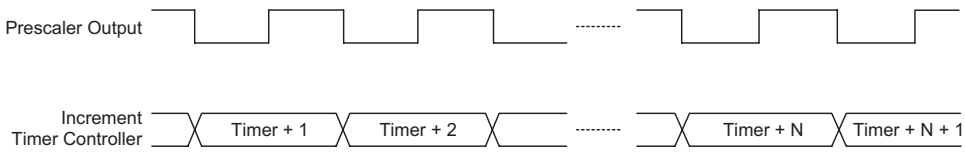
Bit 3 Unimplemented, read as "0"

Bit 2~0 **T2PSC2~T2PSC0**: Timer prescaler rate selection
Timer internal clock f_{T2}=
000: f_{SYS}
001: $f_{SYS}/2$
010: $f_{SYS}/4$
011: $f_{SYS}/8$
100: $f_{SYS}/16$
101: $f_{SYS}/32$
110: $f_{SYS}/64$
111: $f_{SYS}/128$

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, T0M1/T0M0 or T2M1/T2M0, in the Timer Control Register must be set to 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 T0PSC2~T0PSC0 or T2PSC2~T2PSC0 in the Timer Control Register. The timer-on bit, T0ON or T2ON 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.



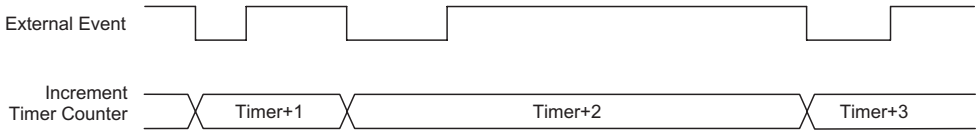
Timer Mode Timing Chart

Event Counter Mode

This mode only exists in the Timer/Event Counter 0. In this mode, a number of externally changing logic events, occurring on the external timer pin TMR0, can be recorded by the Timer/Event Counter 0. To operate in this mode, the Operating Mode Select bit pair, T0M1/T0M0, in the Timer Control Register must be set to 0/1.

In this mode, the external timer pin TMR0 or the INT0 signal can be used as the Timer/Event Counter 0 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 T0ON, 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, T0EG, 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 T0EG 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 0 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 Power Down Mode, the Timer/Event Counter 0 will continue to record externally changing logic events on the timer input TMR0 pin or INT0 signal. As a result when the timer overflows it will generate a timer interrupt and corresponding wake-up source.



Event Counter Mode Timing Chart (T0EG=1)

Pulse Width Measurement Mode

This mode only exists in the Timer/Event Counter 0. 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, T0M1/T0M0, in the Timer Control Register must be set to 1/1.

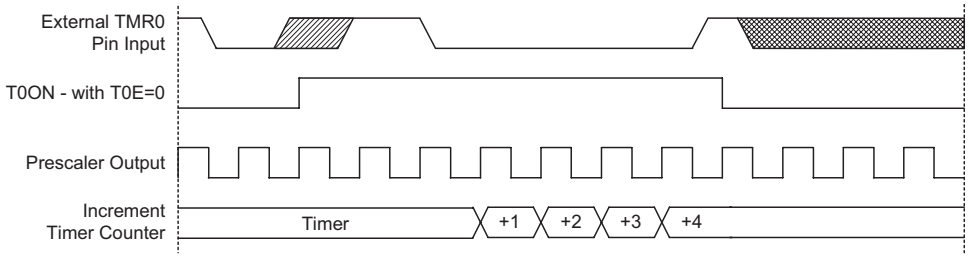
In this mode, the external timer pin TMR0 or the INT0 signal can be used as the Timer/Event Counter clock source, however it is not divided by the internal prescaler. After the other bits in the Timer Control Register have been setup, the enable bit T0ON, which is bit 4 of the Timer Control Register, can be set high to enable the Timer/Event Counter to run. However it will not actually start counting until an active edge is received on the external timer pin.

If the Active Edge Select bit T0EG, 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 TMR0 pin or INT0 signal. 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 TMR0 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.



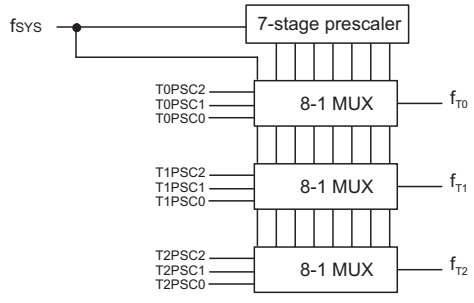
Pulse Width Capture Mode Timing Chart (T0EG=0)

Mode 0

The Timer/Event Counter 1 has a mode 0 for PPG usage. This mode is used to implement the PPG non-retriggered function. In this mode, the timer starts counting when PPG is stopped and stops when overflow. That means the T1ON will be set once PPG stopped and cleared when overflow. Once an overflow occurs, the counter is reloaded from the timer/event counter preload register, and generates an interrupt request flag. 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.

Prescaler

Bits TnPSC0~TnPSC2 of the TMRnC register can be used to define a division ratio for the internal clock source of the Timer/Event Counter enabling longer time out periods to be setup.



Prescaler

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 Power down Mode.

Timer Program Example

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

Timer Programming Example

```
org 0ch               ; external interrupt vector
org 14h               ; Timer Counter 0 interrupt vector
jmp tmr0int           ; jump here when Timer 0 overflows
:
org 20h               ; main program
:
                      ; internal Timer 0 interrupt routine
tmr0int:
:
                      ; Timer 0 main program placed here
begin:
                      ; setup Timer 0 registers
mov a, 09bh           ; setup Timer 0 preload value
mov tmr0, a
mov a, 081h           ; setup Timer 0 control register
mov tmr0c, a          ; timer mode and prescaler set to /2
                      ; setup interrupt register
mov a, 001h           ; enable master interrupt and both timer interrupts
mov intc0, a
mov a, 020h
mov intc1, a
:
set tmr0c.4           ; start Timer 0
:
```

Pulse Width Modulator

The device includes an 8-bit PWM function. Useful for such applications such as motor speed control, the PWM function provides outputs with a fixed frequency but with a duty cycle that can be varied by setting particular values into the corresponding PWM register.

PWM Operation

A single register, known as PWM and located in the Data Memory is assigned to each Pulse Width Modulator channel. It is here that the 8-bit value, which represents the overall duty cycle of one modulation cycle of the output waveform, should be placed. To increase the PWM modulation frequency, each modulation cycle is subdivided into four individual modulation subsections, known as the 6+2 mode. The required mode and the on/off control for each PWM channel is selected using the configuration options. Note that when using the PWM, it is only necessary to write the required value into the PWM register and select the required mode setup and on/off control using the configuration options, the subdivision of the waveform into its sub-modulation cycles is implemented automatically within the microcontroller hardware. The PWM clock source comes from f_{PWM} which is derived from f_{SYS} and selected by PWMPSC2~ PWMPSC0 bits of PWMC register.

This method of dividing the original modulation cycle into a further 4 sub-cycles enable the generation of higher PWM frequencies which allow a wider range of applications to be served. The difference between what is known as the PWM cycle frequency and the PWM modulation frequency should be understood. As the PWM value is 8-bits wide, the overall PWM cycle frequency is $f_{PWM}/256$. The PWM modulation frequency for the 6+2 mode of operation will be $f_{PWM}/64$.

PWM Modulation	PWM Cycle Frequency	PWM Cycle Duty
$f_{PWM}/64$ for (6+2) bits mode	$f_{PWM}/256$	[PWM]/256

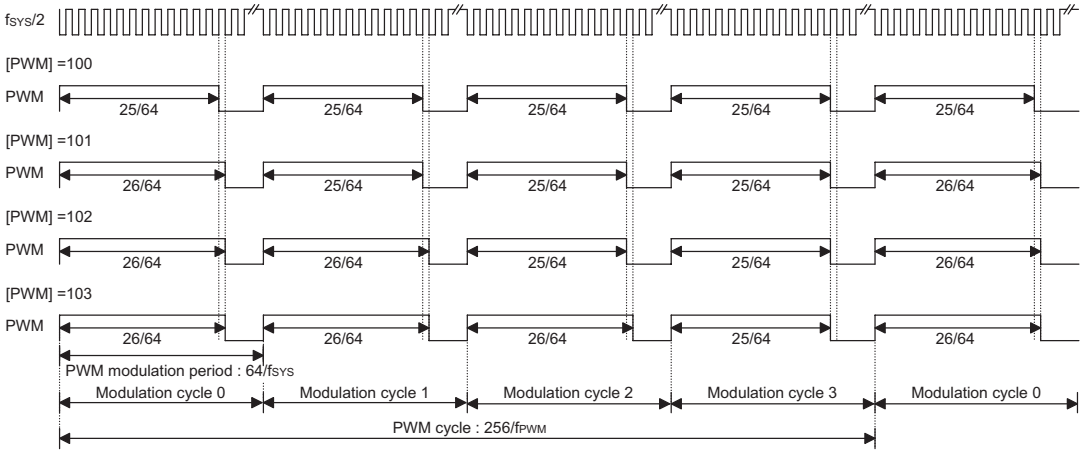
6+2 PWM Mode

Each full PWM cycle, as it is controlled by an 8-bit PWM register, has 256 clock periods. However, in the 6+2 PWM mode, each PWM cycle is subdivided into four individual sub-cycles known as modulation cycle 0 ~ modulation cycle 3, denoted as i in the table. Each one of these four sub-cycles contains 64 clock cycles. In this mode, a modulation frequency increase of four is achieved. The 8-bit PWM register value, which represents the overall duty cycle of the PWM waveform, is divided into two groups. The first group which consists of bit2~bit7 is denoted here as the DC value. The second group which consists of bit0~bit1 is known as the AC value. In the 6+2 PWM mode, the duty cycle value of each of the four modulation sub-cycles is shown in the following table.

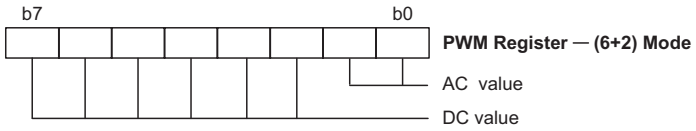
Parameter	AC (0~3)	Duty Cycle
Modulation cycle i (i=0~3)	i<AC	(DC+1)/64
	i≥AC	DC/64

6+2 Mode Modulation Cycle Values

The following diagram illustrates the waveforms associated with the 6+2 mode of PWM operation. It is important to note how the single PWM cycle is subdivided into 4 individual modulation cycles, numbered from 0~3 and how the AC value is related to the PWM value.



6+2 PWM Mode



PWM Register for 6+2 Mode

PWM Output Control

The PWM output is pin-shared with the I/O pin PB4. To operate as a PWM output and not as an I/O pin, the configuration option must be set properly and a zero value must also be written to the bit PBC4 in the I/O port control register to ensure that the corresponding PWM output pin is setup as an output. After these two initial steps have been carried out, and of course after the required PWM value has been written into the PWM register, writing a high value to the PB4 bit in the output data register will enable the PWM data to appear on the pin. Writing a zero value will disable the PWM output function and force the output low. In this way, the Port data output registers can be used as an on/off control for the PWM function. Note that if the configuration options have selected the PWM function, but a high value has been written to its corresponding bit in the PBC control register to configure the pin as an input, then the pin can still function as a normal input line, with pull-high resistor options.

- **PWMC Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	PWMPSC2	PWMPSC1	PWMPSC0
R/W	—	—	—	—	—	R/W	R/W	R/W
POR	—	—	—	—	—	0	0	0

Bit 7~3 Unimplemented, read as "0"

Bit 2~0 **PWMPSC2~PWMPSC0**: Define the prescaler stages

f_{PWM}=

000: f_{SYS}
001: $f_{SYS}/2$
010: $f_{SYS}/4$
011: $f_{SYS}/8$
100: $f_{SYS}/16$
101: $f_{SYS}/32$
110: $f_{SYS}/64$
111: $f_{SYS}/128$

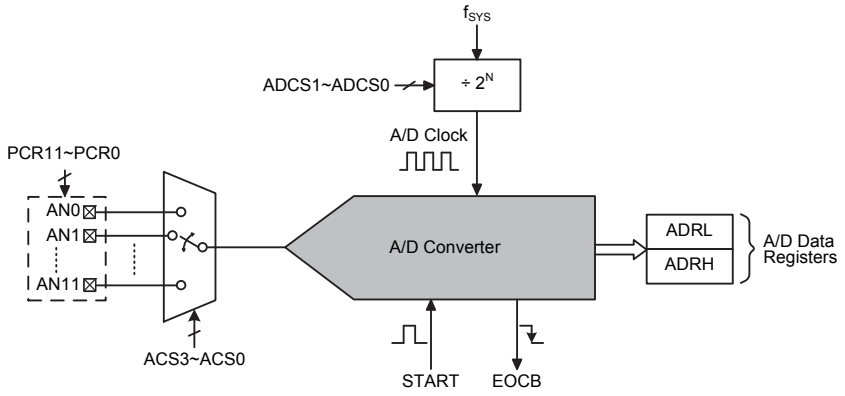
Analog to Digital Converter

The need to interface to real world analog signals is a common requirement for many electronic systems. However, to properly process these signals by a microcontroller, they must first be converted into digital signals by A/D converters. By integrating the A/D conversion electronic circuitry into the microcontroller, the need for external components is reduced significantly with the corresponding follow-on benefits of lower costs and reduced component space requirements.

A/D Overview

The device contains a 12-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 either a 12-bit digital value.

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 Data Registers – ADRL, ADRH

The device, which has an internal 12-bit A/D converter, requires two data registers, 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. Only the high byte register, ADRH, utilises its full 8-bit contents. The low byte register utilises only 4 bit of its 8-bit contents as it contains only the lowest bits of the 12-bit converted value.

In the following table, D0~D11 is the A/D conversion data result bits.

Bit	ADRH								ADRL							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Name	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	—	—	—	—
R/W	R	R	R	R	R	R	R	R	R	R	R	R	—	—	—	—
POR	x	x	x	x	x	x	x	x	x	x	x	x	—	—	—	—

"x" unknown

"—": Unimplemented, read as "0"

D11~D0: ADC conversion data

A/D Converter Control Registers – ADCR, ACSR, PCRL, PCRH

To control the function and operation of the A/D converter, several control registers known as ADCR, ACSR, PCRL and PCRH are provided. These 8-bit registers define functions such as the selection of which analog channel is connected to the internal A/D converter, the digitised data format, the A/D clock source as well as controlling the start function and monitoring the A/D converter end of conversion status.

The ACS3~ACS0 bits in the ADCR register are used to determine which external input is selected to be converted. As the device contains only one actual analog to digital converter circuit, each of the individual 12 analog inputs must be routed to the converter. It is the function of the ACS3~ACS0 bits in the ADCR register to determine which analog channel is actually connected to the internal A/D converter.

The PCRH and PCRL control registers also contain the PCR11~PCR0 bits which determine which pins on PA~PC are used as analog inputs for the A/D converter and which pins are to be used as normal I/O pins.

- **ADCR Register**

Bit	7	6	5	4	3	2	1	0
Name	START	EOCB	—	—	ACS3	ACS2	ACS1	ACS0
R/W	R/W	R	—	—	R/W	R/W	R/W	R/W
POR	0	1	—	—	0	0	0	0

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

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

Bit 5~4 Unimplemented, read as "0"

Bit 3~0 **ACS3~ACS0**: A/D Converter external analog input channel selection
0000: AN0
0001: AN1
0010: AN2
0011: AN3
0100: AN4
0101: AN5
0110: AN6
0111: AN7
1000: AN8
1001: AN9
1010: AN10
1011: AN11
1100~1111: Undefined, cannot be used

- **ACSR Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	ADCS1	ADCS0
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2 Unimplemented, read as "0"

Bit 1~0 **ADCS1~ADCS0**: ADC clock rate selection bit
00: $f_{SYS}/2$
01: $f_{SYS}/8$
10: $f_{SYS}/32$
11: Undefined, cannot be used

- **PCRL Register**

Bit	7	6	5	4	3	2	1	0
Name	PCR7	PCR6	PCR5	PCR4	PCR3	PCR2	PCR1	PCR0
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 **PCR7**: Define PA5 is A/D input or not
0: Not A/D input
1: A/D input, AN7

Bit 6 **PCR6**: Define PA6 is A/D input or not
0: Not A/D input
1: A/D input, AN6

Bit 5 **PCR5**: Define PA7 is A/D input or not
0: Not A/D input
1: A/D input, AN5

Bit 4 **PCR4**: Define PA4 is A/D input or not
0: Not A/D input
1: A/D input, AN4

Bit 3 **PCR3**: Define PB4 is A/D input or not
0: Not A/D input
1: A/D input, AN3

Bit 2 **PCR2**: Define PB3 is A/D input or not
0: Not A/D input
1: A/D input, AN2

Bit 1 **PCR1**: Define PB2 is A/D input or not
0: Not A/D input
1: A/D input, AN1

Bit 0 **PCR0**: Define PB1 is A/D input or not
0: Not A/D input
1: A/D input, AN0

- **PCRH Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	PCR11	PCR10	PCR9	PCR8
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0

Bit 7~4 Unimplemented, read as "0"

Bit 3 **PCR11**: Define PC3 is A/D input or not
0: Not A/D input
1: A/D input, AN11

Bit 2 **PCR10**: Define PC0 is A/D input or not
0: Not A/D input
1: A/D input, AN10

Bit 1 **PCR9**: Internal analog input channel AN9 enable
0: Internal analog input channel AN9 disabled
1: Internal analog input channel AN9 enabled

Bit 0 **PCR8**: Define PA1 is A/D input or not
0: Not A/D input
1: A/D input, AN8

A/D Operation

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

The EOCB bit in the ADCR register is used to indicate when the analog to digital conversion process is complete. This bit will be automatically set to zero 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 ADCR 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 originates from the subdivided version of f_{SYS}. The division ratio value is determined by the ADCS1~ADCS0 bits in the ACSR register.

Although the A/D clock source is determined by the system clock f_{SYS}, and by bits ADCS1~ADCS0, there are some limitations on the maximum A/D clock source speed that can be selected. As the minimum value of permissible A/D clock period, t_{AD}, is 0.5μs, care must be taken for system clock frequencies equal to or greater than 4MHz. For example, if the system clock operates at a frequency of 4MHz, the ADCS1~ADCS0 bits should not be set to "00". Doing so will give A/D clock periods that are less than the minimum A/D clock period which may result in inaccurate A/D conversion values. Refer to the following table for examples, where values marked with an asterisk * show where, depending upon the device, special care must be taken, as the values may be less than the specified minimum A/D Clock Period.

f_{SYS}	A/D Clock Period(t_{AD})			
	ADCS1, ADCS0 =00 (f_{SYS}/2)	ADCS1, ADCS0 =01 (f_{SYS}/8)	ADCS1, ADCS0 =10 (f_{SYS}/32)	ADCS1, ADCS0 =11
1MHz	2μs	8μs	32μs	Undefined
2MHz	1μs	4μs	16μs	Undefined
4MHz	500ns	2μs	8μs	Undefined
8MHz	250ns*	1μs	4μs	Undefined

A/D Clock Period Examples

A/D Input Pins

All of the A/D analog input pins are pin-shared with the I/O pins as well as other functions. The PCR11~PCR0 bits in the PCRH and PCRL register, determine whether the input pins are setup as A/D converter analog inputs or whether they have other functions. 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 configuration options, 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 PA~PC port control register to enable the A/D input as when the PCR11~PCR0 bits enable an A/D input, the status of the port control register will be overridden.

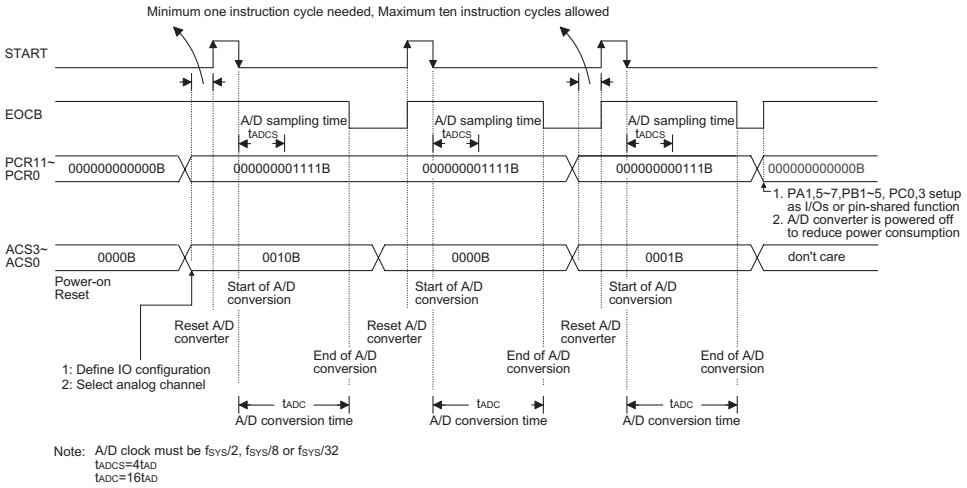
Summary of A/D Conversion Steps

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 ADCS1~ADCS0 in the ACSR register.
- Step 2
  Select which pins are to be used as A/D inputs and configure them as A/D input pins by correctly programming the PCR11~PCR0 bits in the PCRH and PCRL registers.
- Step 3
  Select which channel is to be connected to the internal A/D converter by correctly programming the ACS3~ACS0 bits in the ADCR register.
- Step 4
  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, the INTC0 interrupt control register must be set to "1", the A/D converter interrupt bit, ADE, must also be set to "1".
- Step 5
  The analog to digital conversion process can now be initialised by setting the START bit in the ADCR register from low to high and then low again. Note that this bit should have been originally cleared to zero.
- Step 6
  To check when the analog to digital conversion process is complete, the EOCB bit in the ADCR 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 ADCR register is used, the interrupt enable step above can be omitted.

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



A/D Conversion Timing

Programming Considerations

When programming, the special attention must be given to the PCR11~PCR0 bits in the PCRH and PCRL registers. If these bits are all cleared to zero, no external pins will be selected for use as A/D input pins allowing the pins to be used as normal I/O pins. If the A/D converter is not used, the internal A/D circuitry should be power down. This may be an important consideration in power sensitive applications.

A/D Transfer Function

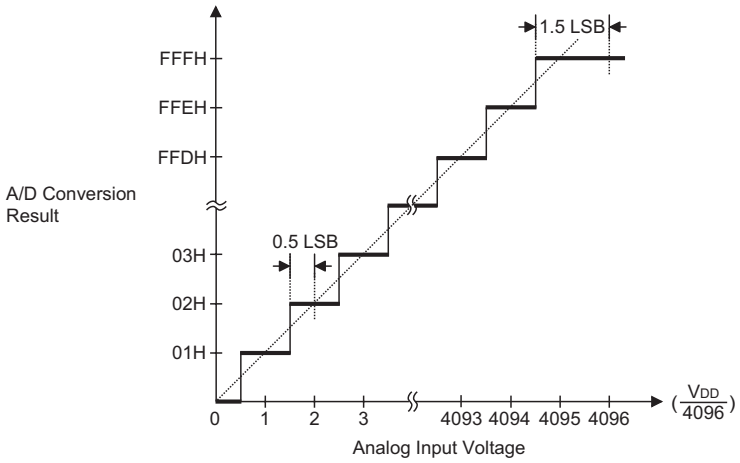
As the device contains a 12-bit A/D converter, its full-scale converted digitised value is equal to FFFH. Since the full-scale analog input value is equal to the V_{DD} voltage, this gives a single bit analog input value of V_{DD} divided by 4096.

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

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

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

The diagram shows the ideal transfer function between the analog input value and the digitised output alue 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 digitized value will change at a point 1.5 LSB below the V_{DD} level.



Ideal A/D Transfer Function

A/D Programming Example

The following two programming examples illustrate how to setup and implement an A/D conversion. In the first example, the method of polling the EOCB bit in the ADCR 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

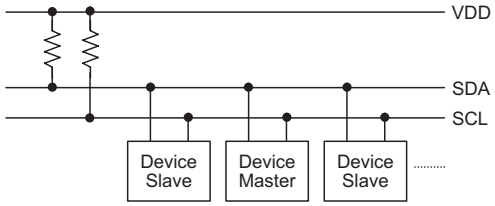
```
clr  ADE              ; disable ADC interrupt
mov  a, 01H
mov  ACSR, a          ; select fSYS/8 as A/D clock
mov  PCRL, a          ; setup PCRL register to configure PB1 as A/D inputs
mov  a, 00h
mov  ADCR, a          ; and select AN0 to be connected to the A/D converter
:
:
start_conversion:
clr  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 ADCR 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  ACSR, a          ; select fSYS/8 as A/D clock
mov  PCRL, a          ; setup PCRL register to configure PB1 as A/D inputs
mov  a, 00h
mov  ADCR, a          ; and select AN0 to be connected to the A/D converter
:
:
Start_conversion:
clr  START            ; high pulse on start bit to initiate conversion
set  START            ; reset A/D
clr  START            ; start A/D
clr  ADF              ; clear ADC interrupt request flag
set  ADE              ; enable ADC interrupt
set  EMI              ; enable global interrupt
:
:
                      ; ADC interrupt service routine
ADC_:
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 _ISR:
mov  a, status_stack
mov  STATUS, a        ; restore STATUS from user defined memory
mov  a, acc_stack     ; restore ACC from user defined memory
clr  ADF              ; clear ADC interrupt flag
reti
```

I²C Interface

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



I²C Master/Slave Bus Connection

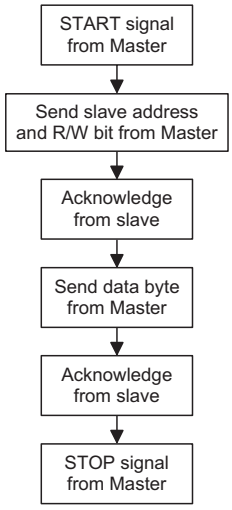
I²C Interface Operation

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

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

It is suggested that the user shall not enter the micro-processor to power down mode by application program during processing I²C communication.

If the pin is configured to SDA or SCL function of I²C interface, the pin is configured to open-collect Input/Output port and its pull-up function can be enabled by configuration options



I²C Registers

There are four control registers associated with the I²C bus, IICC0, IICC1, IICA and IICTOC and one data register, IICD. The IICD register 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 IICD register. After the data is received from the I²C bus, the microcontroller can read it from the IICD register. Any transmission or reception of data from the I²C bus must be made via the IICD register. The IICA register used to save the slave address. The IIC0 register is used to control the I²C interface operations. The IICC1 register is used to indicate the status of the I²C bus. The IICTOC register is used to control the I²C time-out function.

Register Name	Bit							
	7	6	5	4	3	2	1	0
IICC0	—	—	—	—	IICDEB1	IICDEB0	IICEN	—
IICC1	HCF	HAAS	HBB	HTX	TXAK	SRW	IAMWU	RXAK
IICD	D7	D6	D5	D4	D3	D2	D1	D0
IICA	A6	A5	A4	A3	A2	A1	A0	—
IICTOC	IICTOEN	IICTOF	IICTOS5	IICTOS4	IICTOS3	IICTOS2	IICTOS1	IICTOS0

I²C Register List

- **IICC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	IICDEB1	IICDEB0	IICEN	—
R/W	—	—	—	—	R/W	R/W	R/W	—
POR	—	—	—	—	0	0	0	—

Bit 7~4 Unimplemented, read as "0"

Bit 3~2 **IICDEB1~IICDEB0**: I²C Debounce Time Selection
00: Undefined, cannot be used
01: 2 system clock debounce
1x: 4 system clock debounce
If f_{SYS} is come from f_H and ready, or IAMWU=0, the debounce circuit is effect. Otherwise, SCL and SDA will bypass debounce circuit.

Bit 1 **IICEN**: I²C enable
0: Disable
1: Enable

Bit 0 Unimplemented, read as "0"

The I²C function could be turned off or turned on by controlling the bit IICEN. When the I/O ports pin-shared the pins SDA and SCL are chosen to the functions other than SDA and SCL by configuration options and the bit IICEN is zero, I²C function is turned off and its operating current will be reduced to a minimum value. In contrary, I²C function is turned on when the I/O ports pin-shared the pins SDA and SCL are chosen to the pins SDA and SCL by configuration options and the bit IICEN is high.

- **IICC1 Register**

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

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

Bit 6 **HAAS**: I²C Bus 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**: I²C Bus busy flag
0: I²C Bus is not busy
1: I²C Bus is busy
The HBB flag is the I²C busy flag. This flag will be "1" when the I²C bus is busy which will occur when a START signal is detected. The flag will be set to "0" when the bus is free which will occur when a STOP signal is detected.

Bit 4 **HTX**: Select I²C slave device is transmitter or receiver
0: Slave device is the receiver
1: Slave device is the transmitter

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

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

Bit 1 **IAMWU**: I²C address match control
0: Disable
1: Enable
If f_{SYS} is from f_H and ready, then this control bit has no effect, I²C Address Match always can generate interrupt as this interrupt enable bit is set. Otherwise, set IAMWU bit also can generate interrupt when I²C Address Match as this interrupt enable bit is set, but clear IAMWU bit maybe cannot generate interrupt when I²C Address Match even if this interrupt enable bit is set.

Bit 0 **RXAK**: I²C Bus Receive acknowledge flag
0: Slave receive 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.

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

- **IICD Register**

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

"x": unknown

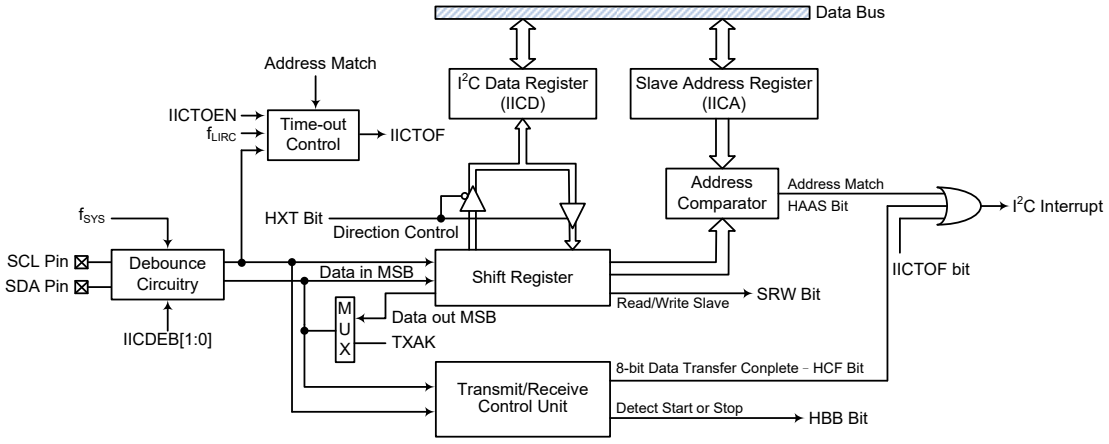
Bit 7~0 **D7~D0**: I²C Data Buffer bit 7~bit 0

- **IICA 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 the I²C slave address bit 6 ~ bit 0.
The IICA register is the location where the 7-bit slave address of the slave device is stored. Bits 7~1 of the IICA register define the device slave address. Bit 0 is not defined. When a master device, which is connected to the I²C bus, sends out an address, which matches the slave address in the IICA register, the slave device will be selected.

Bit 0 Unimplemented, read as "0"
This bit can be read or written by software program.

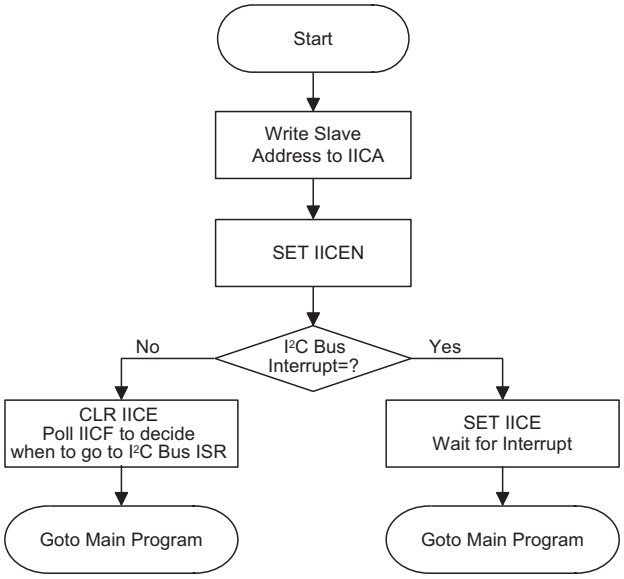


I²C Block Diagram

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 IICC1 register will be set and an I²C interrupt will be generated. After entering the interrupt service routine, the slave device must first check the condition of the HAAS and IICTOF bits to determine whether the interrupt source originates from an address match or from the completion of an 8-bit data transfer or I²C time-out. During a data transfer, note that after the 7-bit slave address has been transmitted, the following bit, which is the 8th bit, is the read/write bit whose value will be placed in the SRW bit. This bit will be checked by the slave device to determine whether to go into transmit or receive mode. Before any transfer of data to or from the I²C bus, the microcontroller must initialise the bus, the following are steps to achieve this:

- Step 1
  Configure the pin-shared I/O ports to I²C function and set IICEN bit in the IICC0 register to enable the I²C bus.
- Step 2
  Write the slave address of the device to the I²C bus address register IICA.
- Step 3
  Set the IICE interrupt enable bit of the interrupt control register to enable the I²C 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 IICC1 register. The slave device will then transmit an acknowledge bit, which is a low level, as the 9th bit. The slave device will also set the status flag HAAS when the addresses match.

As an I²C bus interrupt can come from three sources, when the program enters the interrupt subroutine, the HAAS and IICTOF bits 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 or I²C time-out. When a slave address is matched, the device must be placed in either the transmit mode and then write data to the IICD register, or in the receive mode where it must implement a dummy read from the IICD register to release the SCL line.

I²C Bus Read/Write Signal

The SRW bit in the IICC1 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.

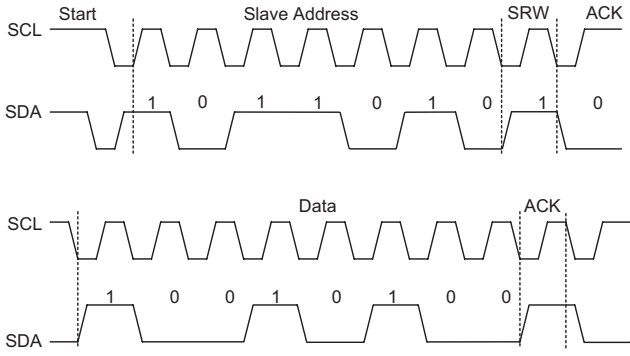
I²C Bus Slave Address Acknowledge Signal

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 IICC1 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 IICC1 register should be set to "0".

I²C Bus Data and Acknowledge Signal

The transmitted data is 8-bits wide and is transmitted after the slave device has acknowledged receipt of its slave address. The order of serial bit transmission is the MSB first and the LSB last. After receipt of 8-bits of data, the receiver must transmit an acknowledge signal, level "0", before it can receive the next data byte. If the slave transmitter does not receive an acknowledge bit signal from the master receiver, then the slave transmitter will release the SDA line to allow the master to send a STOP signal to release the I²C Bus. The corresponding data will be stored in the IICD register. If setup as a transmitter, the slave device must first write the data to be transmitted into the IICD register. If setup as a receiver, the slave device must read the transmitted data from the IICD 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 IICC1 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.

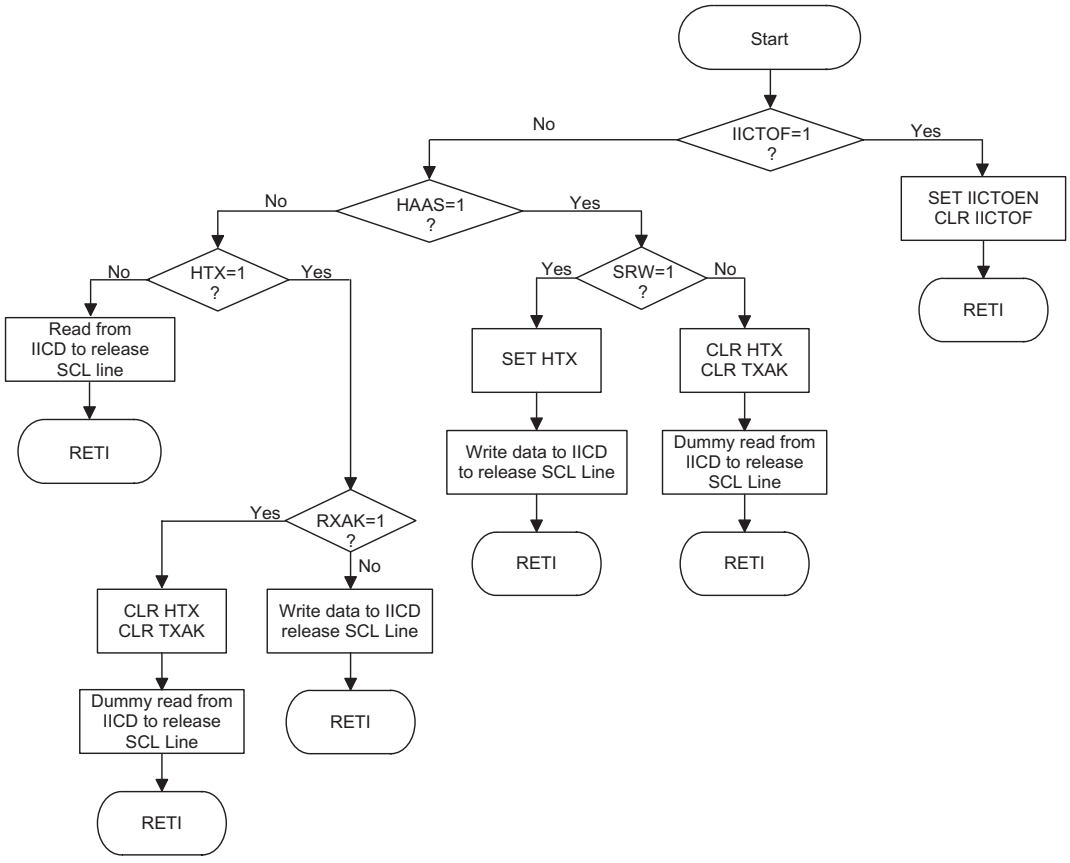


S=Start (1 bit)
SA=Slave Address (7 bits)
SR=SRW bit (1 bit)
M=Slave device send acknowledge bit (1 bit)
D=Data (8 bits)
A=ACK (RXAK bit for transmitter, TXAK bit for receiver 1 bit)
P=Stop (1 bit)

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 IICD register, or in the receive mode where it must implement a dummy read from the IICD register to release the I²C SCL line.

I²C Communication Timing Diagram

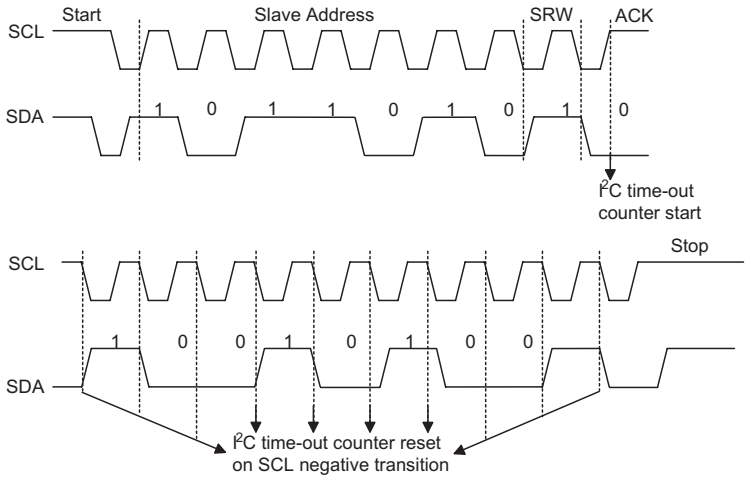


I²C Bus ISR Flow Chart

I²C Time-out Control

In order to reduce the problem of I²C lockup due to reception of erroneous clock sources, a time-out function is provided. If the clock source to the I²C is not received then after a fixed time period, the I²C circuitry and registers will be reset.

The time-out counter starts counting on an I²C bus "START" & "address match" condition, and is cleared by an SCL falling edge. Before the next SCL falling edge arrives, if the time elapsed is greater than the time-out setup by the IICTOC register, then a time-out condition will occur. The time-out function will stop when an I²C "STOP" condition occurs.



I²C time-out timing

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

Register	After I²C Time-out
IICD, IICA, IICC0	No change
IICC1	Reset to POR condition

I²C Registers after Time-out

The IICTOF flag can be cleared by the application program. There are 64 time-out periods which can be selected using bits in the IICTOC register. The time-out time is given by the formula:

$$((1\sim64)\times32) / f_{LIRC}$$

This gives a range of about 1ms to 64ms.

- **IICTOC Register**

Bit	7	6	5	4	3	2	1	0
Name	IICTOEN	IICTOF	IICTOS5	IICTOS4	IICTOS3	IICTOS2	IICTOS1	IICTOS0
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 **IICTOEN**: I²C Time-out Control
0: Disable
1: Enable

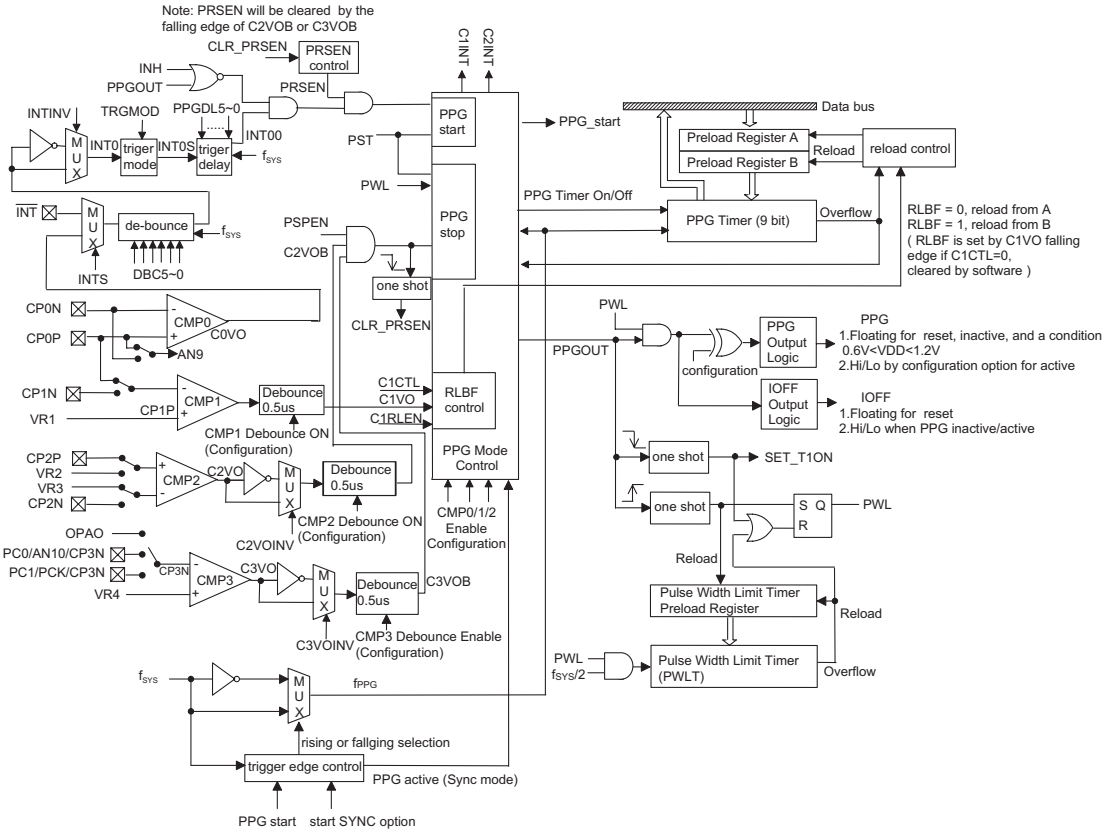
Bit 6 **IICTOF**: Time-out flag (set by time-out and clear by software)
0: No time-out
1: Time-out occurred

Bit 5~0 **IICTOS5~IICTOS0**: Time-out Definition
I^2C time-out clock source is f_{LIRC} /32.
I^2C time-out time is given by: ([IICTOS[5:0]+1)×(32/f_{LIRC})

Programmable Pulse Generator

The device provides one 9-bit PPG output channel. The PPG has a programmable period of 512xT, where T is $1/f_{SYS}$ for an output pulse width. The PPG pulse width could be limited with using pulse width limiter timer.

The PPG detects a trigger input, and outputs a single pulse. The trigger source may come from INT00 falling edge or software trigger bit, which can be configured by software. The PPG can output an active low or active high pulse by setting the configuration option. The PPG output is floating when V_{DD} is between 0.6V and 1.2V, Reset occurs, and the PPG is inactive; an external pull-high or pull-low resistor (depended on polarity configuration option) is need.



PPG Block Diagram

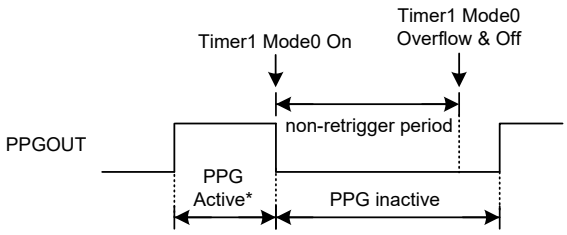
The PPG module consists of one PPG timer, one PPG Mode Control and four comparators. The PPG timer consists of one 9-bit up-counter timer, and two 9-bit preload data registers. The programmable pulse generator (PPG) starts counting at the current contents in the preload register and ends at "1FFH → 000H". A "000H" data write to the PPGTA and PPGTB register yields a pulse width 512xT output. Once an overflow occurs, the counter is reloaded from the PPG timer counter preload register, and generates a signal to stop the PPG timer. The software trigger bit (PST) will be cleared when the PPG timer overflow occurs.

Non-retriggered Function

The PPG unit has non-retriggered function to inhibit further PPG trigger. The PPG will be non-triggered by one of following condition:

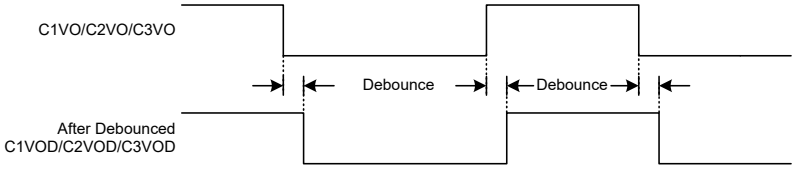
1. PPG is active

2. During the non-retriggered period which starts counting once PPG stopped (Only available by using with mode 0 of Timer/Event Counter 1, the non-trigger period is decided by Timer/Event Counter 1)



Note: * PPG is always non-retrigger if PPG is active.

Debounce for C1VO/C2VO/C3VO



Note: The control bits, C0CMPOP, C1CMPOP, C2CMPOP and C3CMPOP, are used to set output status of C0VO, C1VO, C2VO, and C3VO, which have not been debounced.

Pulse Width Limit Function

The PPG unit has pulse width limit function to stop PPG output. The PPG output will be stopped once the pulse width reaching the limit. This function is implemented by a pulse width limit timer which starts counting once PPG is triggered and stops once overflow or PPG is stopped. The pulse width limit is (256-PWLT)/(f_{SYS}/2), where PWLT is pulse width limit timer register.

To start the PPG operation:

- Enable PPG function by configuration option.
- Set the PPG output active level by configuration option
- Set the PPG timer start counting is synchronized with system clock f_{SYS} or not by configuration option
- Set the PPG input mode selection by the PRSEN bit and PSPEN bit of the PPGC register
- Set the PPG output pulse width. Writing data to PPGTA, PPGTB and PPGTEX registers
- Decide using C1VO falling edge to enable the reload function from preload register B or not by setting the C1RLEN bit of the PPGC register and the C1CTL bit in the CMPSC register

- Decide using the non-retriggered period function or not by using with mode 0 of Timer/Event Counter 1
- Set pulse width limit timer for pulse width limit function by the PWLT register.

When PPG input is triggered by INT00 falling edge transition or triggered by a software bit (PST) being set to "1", the PPG will start counting from the current content of preload register. When PPG input is trigged by a software bit (PST) being cleared to zero, C2VOB falling edge, PPG timer overflow occurs or a pulse width limit condition occurs, the PPG will stop counting.

- **PPGC Register**

Bit	7	6	5	4	3	2	1	0
Name	PST	PRSEN	PSPEN	RLBF	—	—	TRGMOD	C1RLEN
R/W	R/W	R/W	R/W	R/W	—	—	R/W	R/W
POR	0	0	0	0	—	—	0	0

Bit 7 **PST**: PPG software trigger bit
0: Stop PPG
1: Restart PPG

Bit 6 **PRSEN**: Enable or disable restarting the PPG timer using INT00 trigger input
0: Disable
1: Enable
Disable restarting the PPG timer using INT00 trigger input, PPG module output can be restarted by software control bit PST only. Enable restarting the PPG timer using INT00 trigger input, PPG module output can be restarted by INT00 falling edge trigger or software control (PST is set to "1").

Bit 5 **PSPEN**: Enable or disable stopping the PPG timer using the trigger input of C2VOB or C3VOB
0: Disable
1: Enable
The C2VOB or C3VOB signal is the inverted or non-inverted signal from the output of comparator 2 or comparator 3, which is selected by software option
Disable stopping the PPG timer using C2VOB or C3VOB trigger input, PPG module output can be stopped by software control bit PST only. Enable stopping the PPG timer using C2VOB or C3VOB trigger input, PPG module output can be stopped by C2VOB or C3VOB falling edge trigger or software control (PST is set to "0").

Bit 4 **RLBF**: PPG reload control bit
0: From PPGTA
1: From PPGTB

Bit 3~2 Unimplemented, read as "0"

Bit 1 **TRGMOD**: Select single or double falling edge of INT0 as the input of trigger delay circuit which produce INT00
0: Single
1: Double

Bit 0 **C1RLEN**: Enable or disable C1VO falling edge to set RLBF for PPG timer reloads from preload register B
0: Disable
1: Enable
This bit is available when C1CTL=0. CP1P and CP1N are connected to programmable internal reference voltage V_{R1} and PA3/CP0P/TMR0 pin respectively
When C1CTL=0 and C1VO=1, if the C1RLEN is changed from 1 to 0, the RLBF bit will be set. So note that when the C1RLEN is changed from 1 to 0, the RLBF bit should be cleared to 0 by software, otherwise the PPG width will be determined by the PPGTB.

Normally, PPG timer is reloaded from preload register A if RLBF=0. If C1RLEN is set and C1CTL is cleared, the C1VO falling edge caused by comparator output will set RLBF and then PPG timer will be reload from preload register B until RLBF is cleared by software.

The PRSEN is the PPG restarting enable or disable bit using INT00 trigger input. If this bit is enabled, the PPG timer restarting input can be trigger by INT00 falling edge.

The PSPEN is the PPG stopping enable or disable bit using the trigger input of C2VOB or C3VOB. If this bit is enabled, the PPG timer stopping input can be triggered and an IOFF output will active high by the falling edge of C2VOB or C3VOB. The IOFF output is floating during reset, high when PPG is inactive and low when PPG is active. The PRSEN bit will be cleared by the falling edge of C2VOB or C3VOB, no matter the PPG is in active period or not. This will prevent PPG module output be restarted by INT00 falling edge again, only restarted by software control is permitted until PRSEN is set again by software.

The PST is a software trigger bit, if this bit is set to "1", the PPG timer will start counting and this bit will be cleared when the PPG timer overflow occurs or PPG timer stop counting. If this bit is cleared to "0", the PPG timer will stop counting. When the PPG timer is counting and if a falling edge generates from INT00 or a software control bit (PST) is set, the PPG timer counter is not affected, the trigger from INT00 or PST is not useful. The PST can also be used as a status bit of PPG timer output.

The PPG module output pulse active level is decided by configuration option. Another function is provided, which is the PPG timer start counting is synchronized with clock or not, decided by configuration option.

- **PPGTA Register**

Bit	7	6	5	4	3	2	1	0
Name	PGTA7	PGTA6	PGTA5	PGTA4	PGTA3	PGTA2	PGTA1	PGTA0
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

- **PPGTB Register**

Bit	7	6	5	4	3	2	1	0
Name	PGTB7	PGTB6	PGTB5	PGTB4	PGTB3	PGTB2	PGTB1	PGTB0
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

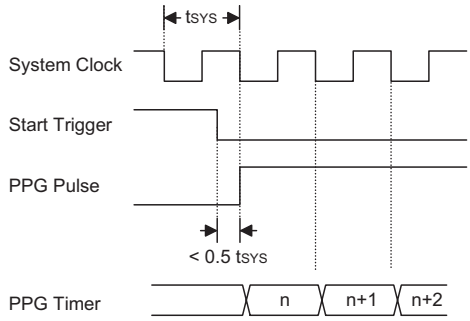
- **PPGTEX Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	PGTB8	—	—	—	PGTA8
R/W	—	—	—	R/W	—	—	—	R/W
POR	—	—	—	x	—	—	—	x

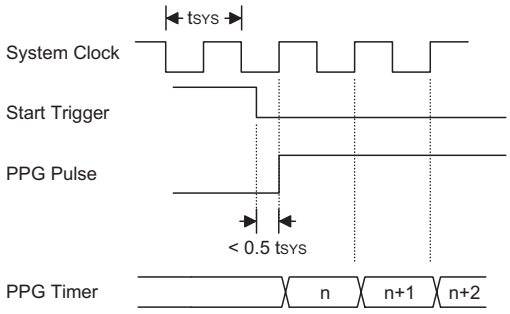
"x": unknown

To control PPG pulse starting delay $\leq 0.5\times(1/f_{SYS})$ when start synchronized with clock is selected, clock (f_{SYS}) edge trigger type (raising or falling), which triggers PPG, varies with next coming clock transition once PPG starts. After PPG starts, the PPG output becomes active and begins to count as soon as first transition (falling or rising) of system clock comes. After first trigger done, the following clock edge trigger type is decided by the first one. For example, once PPG starts and next coming clock transition is falling edge, the PPG will be trigger by falling edge until PPG stops and vice versa.

EX1: Since the first trigger type is falling edge after PPG starts, the PPG timer is triggered by falling edge until PPG stops.



EX2: Since the first trigger type is raising edge after PPG starts, the PPG timer is triggered by raising edge until PPG stops.



Any action causing PPG stop, such as PPG timer overflow, C2VOB falling edge (if PSPEN=1), software stop (PST=1→0) or reaching pulse limit, will cause actions as following:

- PPG timer will be reloaded
- PST is cleared
- PPG is inactive

Comparators

The device includes 4 integrated comparators in PPG module. Either inputs of the comparator 0 can be connected to AN9 selected by C0N2AN9 bit in the CMPSC register. The non-inverting input of the comparator 1 is connected to the programmable internal reference voltage V_{R1}. The inverting input of comparator 1 can be connected to PA5/CP1N/AN7 or PA3/CP0P/TMR0 by the C1CTL bit. The non-inverting input of the comparator 2 can be connected to PA7/CP2P/AN5 or programmable internal reference voltage V_{R2} by configuration options. The inverting input of the comparator 2 can be connected to PA6/CP2N/AN6 or programmable internal reference voltage V_{R3} by configuration options. The non-inverting input of the comparator 3 is connected to programmable internal reference voltage V_{R4}. The inverting input of comparator 3 can be connected to PC0/AN10/CP3N, PC1/PCK/CP3N or OPAO by configuration options

The input offset is adjustable by using a common mode input to calibrate the offset value.

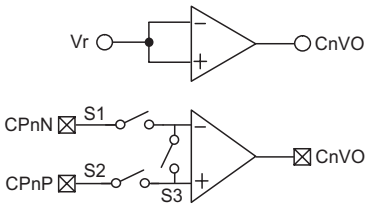
The calibration steps are as the follows:

(1) Setting CnCOFM=1 to offset cancellation mode (S3 is closed)

(2) Setting CnCRS to select which input pin as reference voltage (S1 or S2 is closed)

(3) Adjusting CnCOF0~CnCOF4 until output status is changed

(4) Setting CnCOFM=0 to normal comparator mode



Comparator (n-0~3)

Comparator registers

Full control over the internal comparator is provided via the control registers, CMP0C, CMP1C, CMP2C, CMP3C, and CMPSC. These registers are used to select the input path and the offset voltage cancellation function.

Register Name	Bit							
	7	6	5	4	3	2	1	0
CMP0C	C0CMPOP	C0COFM	C0CRS	C0COF4	C0COF3	C0COF2	C0COF1	C0COF0
CMP1C	C1CMPOP	C1COFM	C1CRS	C1COF4	C1COF3	C1COF2	C1COF1	C1COF0
CMP2C	C2CMPOP	C2COFM	C2CRS	C2COF4	C2COF3	C2COF2	C2COF1	C2COF0
CMP3C	C3CMPOP	C3COFM	C3CRS	C3COF4	C3COF3	C3COF2	C3COF1	C3COF0
CMPSC	C3HYSON	C2HYSON	C1HYSON	C0HYSON	—	—	C1CTL	C0N2AN9

Comparators Register List

- **CMP0C Register**

Bit	7	6	5	4	3	2	1	0
Name	C0CMPOP	C0COFM	C0CRS	C0COF4	C0COF3	C0COF2	C0COF1	C0COF0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	1	0	0	0	0

Bit 7 **C0CMPOP**: Comparator 0 output, positive logic

Bit 6 **C0COFM**: Input offset voltage cancellation mode and comparator mode selection
0: Comparator mode
1: Input offset voltage cancellation mode

Bit 5 **C0CRS**: Comparator 0 input offset voltage cancellation reference selection bit
0: Select CP0N as the reference input
1: Select CP0P as the reference input

Bit 4~0 **C0COF4~C0COF0**: Comparator 0 input offset voltage cancellation control bits

- **CMP1C Register**

Bit	7	6	5	4	3	2	1	0
Name	C1CMPOP	C1COFM	C1CRS	C1COF4	C1COF3	C1COF2	C1COF1	C1COF0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	1	0	0	0	0

Bit 7 **C1CMPOP**: Comparator 1 output, positive logic

Bit 6 **C1COFM**: Input offset voltage cancellation mode and comparator mode selection
0: Comparator mode
1: Input offset voltage cancellation mode

Bit 5 **C1CRS**: Comparator 1 input offset voltage cancellation reference selection bit
0: Select CP1N as the reference input
1: Select CP1P as the reference input

Bit 4~0 **C1COF4~C1COF0**: Comparator 1 input offset voltage cancellation control bits

- **CMP2C Register**

Bit	7	6	5	4	3	2	1	0
Name	C2CMPOP	C2COFM	C2CRS	C2COF4	C2COF3	C2COF2	C2COF1	C2COF0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	1	0	0	0	0

Bit 7 **C2CMPOP**: Comparator 2 output, positive logic

Bit 6 **C2COFM**: Input offset voltage cancellation mode and comparator mode selection
0: Comparator mode
1: Input offset voltage cancellation mode

Bit 5 **C2CRS**: Comparator 2 input offset voltage cancellation reference selection bit
0: Select CP2N as the reference input
1: Select CP2P as the reference input

Bit 4~0 **C2COF4~C2COF0**: Comparator 2 input offset voltage cancellation control bits

- **CMP3C Register**

Bit	7	6	5	4	3	2	1	0
Name	C3CMPOP	C3COFM	C3CRS	C3COF4	C3COF3	C3COF2	C3COF1	C3COF0
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 **C3CMPOP**: Comparator 3 output, positive logic

Bit 6 **C3COFM**: Input offset voltage cancellation mode and comparator mode selection
0: Comparator mode
1: Input offset voltage cancellation mode

Bit 5 **C3CRS**: Comparator 3 input offset voltage cancellation reference selection bit
0: Select CP3N as the reference input
1: Select CP3P as the reference input

Bit 4~0 **C3COF4~C3COF0**: Comparator 3 input offset voltage cancellation control bits

- **CMPSC Register**

Bit	7	6	5	4	3	2	1	0
Name	C3HYSON	C2HYSON	C1HYSON	C0HYSON	—	—	C1CTL	C0N2AN9
R/W	R/W	R/W	R/W	R/W	—	—	R/W	R/W
POR	0	0	0	0	—	—	0	0

Bit 7 **C3HYSON**: Comparator 3 hysteresis control bit
0: Disable
1: Enable

Bit 6 **C2HYSON**: Comparator 2 hysteresis control bit
0: Disable
1: Enable

Bit 5 **C1HYSON**: Comparator 1 hysteresis control bit
0: Disable
1: Enable

Bit 4 **C0HYSON**: Comparator 0 hysteresis control bit
0: Disable
1: Enable

Bit 3~2 Unimplemented, read as "0"

Bit 1 **C1CTL**: Comparator 1 inputs connection control bit
0: The inverting input of comparator 1 is connected to PA3/CP0P/TMR0
1: The inverting input of comparator 1 is connected to PA5/CP1N/AN7

Bit 0 **C0N2AN9**: Comparator 0 inputs connection to AN9
0: CP0P connected to AN9
1: CP0N connected to AN9

Operational Amplifier

The device includes an integrated operational amplifier which is used to amplify the small analog input signal. It can be enabled or disabled by configuration options, it is only available when HIRC is selected in Oscillator configuration.

The non-inverting input can be connected to V_{SS} directly or through a 1kΩ resistor determined by configuration option. The inverting input can be connected to OPAN pin directly or through a 1kΩ resistor determined by configuration option. The output can be connected back to the inverting input through a 60kΩ resistor or not determined by configuration option. The output can be connected to PA1/OPAO/AN8/OSC1 pin or connected to PB3/AN2 pin through a 10kΩ resister by configuration options.

The input offset is adjustable by using a common mode input to calibrate the offset value.

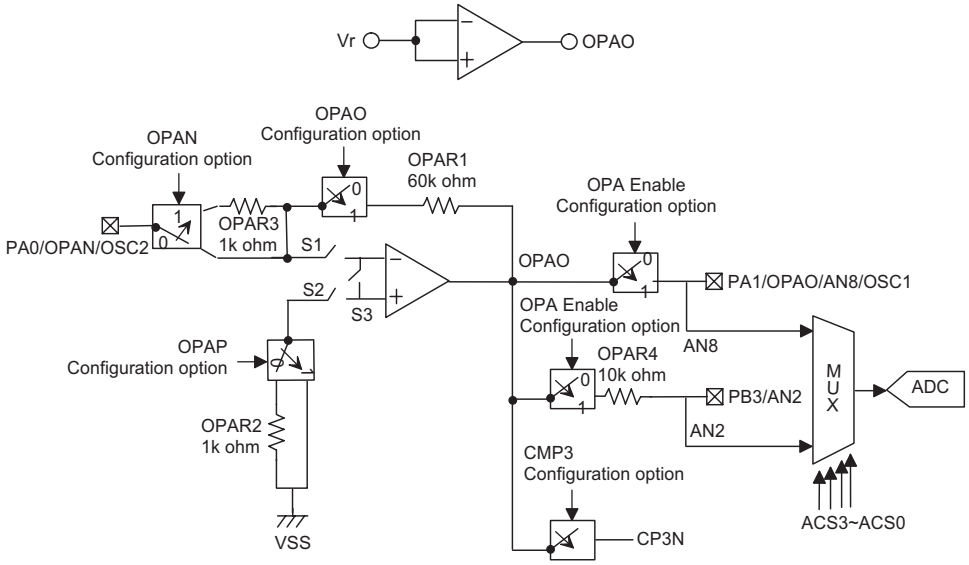
The calibration steps are as the follows:

(1) Setting OPAFM=1 to offset cancellation mode (S3 is closed)

(2) Setting OPARS to select which input pin as reference voltage (S1 or S2 is closed)

(3) Adjusting OPAOF0~OPAOF4 until output status is changed.

(4) Setting OPAOFM=0 to normal operational amplifier mode



Operational Amplifier Register

The overall function is controlled by the OPAC register, it is used to select the operational amplifier input offset voltage cancellation and the input path.

- **OPAC Register**

Bit	7	6	5	4	3	2	1	0
Name	OPAOP	OPAOFM	OPARS	OPAOF4	OPAOF3	OPAOF2	OPAOF1	OPAOF0
R/W	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	1	0	0	0	0

Bit 7 **OPAOP**: Operational amplifier output, positive logic

Bit 6 **OPAOFM**: Input offset voltage cancellation mode and operational amplifier mode selection
0: Operational amplifier mode
1: Input offset voltage cancellation mode

Bit 5 **OPARS**: Operational amplifier input offset voltage cancellation reference selection bit
0: Select OPAN as the reference input
1: Select OPAP as the reference input

Bit 4~0 **OPAOF4~OPAOF0**: Operational amplifier input offset voltage cancellation control bits

Peripheral Clock Output

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

Peripheral Clock Operation

The peripheral clock output pin PCK is pin-shared with the I/O pin PC1. To operate as a peripheral clock output and not as an I/O pin, the configuration option must be set properly and a zero value must also be written to the bit PCC1 in the I/O port control register to ensure that the corresponding peripheral clock output pin is setup as an output. After these two initial steps have been carried out, writing a high value to the PC1 bit in the output data register will enable the PCK output function. Writing a zero value will disable the PCK output function and force the output low. In this way, the Port data output registers can be used as an on/off control for the PCK function. Note that if the configuration options have selected the PCK function, but a high value has been written to its corresponding bit in the PCC control register to configure the pin as an input, then the pin can still function as a normal input line, with pull-high resistor options.

The clock source for the Peripheral Clock Output can originate from the subdivided version of f_{SYS}, The division ratio value is determined by the PCKPSC2~PCKPSC0 bits in the CTRL0 register.

If the device enters the power down mode, this will disable the Peripheral Clock output.

- **CTRL0 Register**

Bit	7	6	5	4	3	2	1	0
Name	C2VOINV	TMR0ECS	PCKPSC2	PCKPSC1	PCKPSC0	OSTPC	LVDO	LVDC
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W
POR	0	0	0	0	0	0	0	0

Bit 7 **C2VOINV**: Inverting control of the comparator 2 output signal
Described elsewhere

Bit 6 **TMR0ECS**: Select TMR0 external clock source
Described elsewhere

Bit 5~3 **PCKPSC2~PCKPSC0**: Peripheral clock prescaler stage
f_{PCK}=
000: $f_{SYS}/4$
001: $f_{SYS}/8$
010: $f_{SYS}/12$
011: $f_{SYS}/16$
100: $f_{SYS}/20$
101: $f_{SYS}/24$
110: $f_{SYS}/1024$
111: $f_{SYS}/2048$

Bit 2 **OSTPC**: Oscillator stop function control
Described elsewhere

Bit 1 **LVDO**: Low voltage detector output
Described elsewhere

Bit 0 **LVDC**: Low voltage detector control
Described elsewhere

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 interrupts and several internal interrupts functions. The external interrupt is generated by the action of the external $\overline{INT}$ pin, while the internal interrupts are generated by various internal functions such as Timer/Event Counters, comparator, LVD, EEPROM, A/D converter and I^2C bus.

Interrupt Registers

All the interrupt enable bits and the request flags are controlled by the INTC0, INTC1 and the MFI registers. By controlling the corresponding interrupt enable bits can enable or disable the interrupts. If an interrupt occurs, the request flag will be set and the EMI bit will be automatically cleared to disable other interrupts.

- **INTC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	INTF/LVDF	CP1F	MFF	INTE/LVDE	CP1E	MFE	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 **INTF/LVDF**: External interrupt or LVD interrupt Request Flag
0: No request
1: Interrupt request
Selected by the configuration options

Bit 5 **CP1F**: Comparator 1 Interrupt Request Flag
0: No request
1: Interrupt request

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

Bit 3 **INTE/LVDE**: External interrupt or LVD interrupt Control
0: Disable
1: Enable
Selected by the configuration options

Bit 2 **CP1E**: Comparator 1 interrupt control
0: Disable
1: Enable

Bit 1 **MFE**: Multi-function 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	DEF	T2F/ADF	T0F	IICF	DEE	T2E/ADE	T0E	IICE
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 **DEF**: EEPROM Interrupt Request Flag
0: No request
1: Interrupt request

Bit 6 **T2F/ADF**: Timer/Event Counter 2 or A/D converter Interrupt Request Flag
0: No request
1: Interrupt request
Selected by the configuration options

Bit 5 **T0F**: Timer/Event Counter 0 interrupt request flag
0: No request
1: Interrupt request

Bit 4 **IICF**: I^2C bus interrupt request flag
0: No request
1: Interrupt request

Bit 3 **DEE**: EEPROM Interrupt Control
0: Disable
1: Enable

Bit 2 **T2E/ADE**: Timer/Event Counter 2 or A/D converter Interrupt Control
0: Disable
1: Enable
Selected by the configuration options

Bit 1 **T0E**: Timer/Event Counter 0 Interrupt Control
0: Disable
1: Enable

Bit 0 **IICE**: I^2C bus Interrupt Control
0: Disable
1: Enable

- **MFI Register**

Bit	7	6	5	4	3	2	1	0
Name	—	T1F	CP2F	CP3F	—	T1E	CP2E	CP3E
R/W	—	R/W	R/W	R/W	—	R/W	R/W	R/W
POR	—	0	0	0	—	0	0	0

Bit 7 Unimplemented, read as “0”

Bit 6 **T1F**: Timer/Event Counter 1 Interrupt Request Flag
0: No request
1: Interrupt request

Bit 5 **CP2F**: Comparator 2 Interrupt Request Flag
0: No request
1: Interrupt request

Bit 4 **CP3F**: Comparator 3 Interrupt Request Flag
0: No request
1: Interrupt request

Bit 3 Unimplemented, read as "0"

Bit 2 **T1E**: Timer/Event Counter 1 Interrupt Control
0: Disable
1: Enable

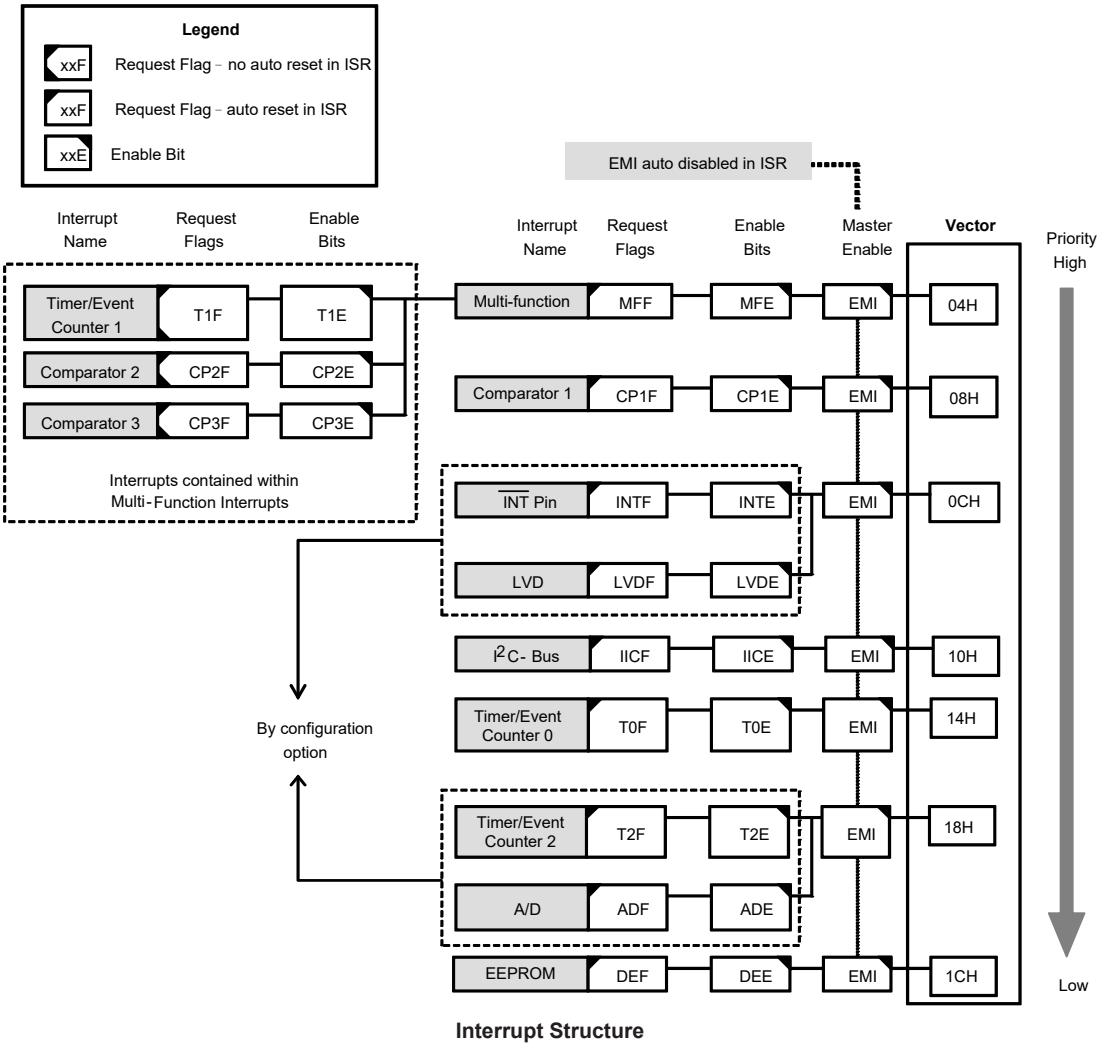
Bit 1 **CP2E**: Comparator 2 Interrupt Control
0: Disable
1: Enable

Bit 0 **CP3E**: Comparator 3 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, the relevant interrupt request flag will be set. 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.



Interrupt Structure

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.

Interrupt Priority

Interrupts, occurring in the interval between the rising edges of two consecutive T2 pulses, will be serviced on the latter of the two T2 pulses, if the corresponding interrupts are enabled. In case of simultaneous requests, the following table shows the priority that is applied.

Interrupt sources	Priority	Vector
Multi-function interrupt	1	04H
Comparator 1 output interrupt	2	08H
External interrupt or LVD interrupt (Selected by configuration option)	3	0CH
I^2C bus interrupt	4	10H
Timer/Event Counter 0 overflow	5	14H
Timer/Event Counter 2 overflow or A/D converter interrupt(Selected by configuration option)	6	18H
EEPROM interrupt	7	1CH

The Timer/Event Counter 1 interrupt, comparator 2 interrupt and the comparator 3 interrupt share the same interrupt vector which is 04H. Each of these interrupts has their own individual interrupt flag but also share the same MFF interrupt flag. The MFF flag will be cleared by hardware once the Multi-function interrupt is serviced, however the individual interrupts that have triggered the Multi-function interrupt need to be cleared by the application program.

External Interrupt

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. The external interrupt is triggered by falling edge transition of INT0 (inverted or non-inverted de-bounce signal from $\overline{INT}$ or comparator 0 output “C0VO” by software option), an external interrupt request will take place when the external interrupt request flag, INTF, is set. As the external interrupt pin is pin-shared with the PA2 pin, it can only be configured as the 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, 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.

Note that the trigger source of 0CH vector can be LVD interrupt or the external interrupt, which is selected by the configuration options. The external interrupt must be first selected by the configuration options before setting the external interrupt function.

LVD Interrupt

An LVD Interrupt request will take place when the LVD Interrupt request flag, LVF, is set, which occurs when the Low Voltage Detector function detects a low power supply voltage. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and Low Voltage Interrupt enable bit, LVE, 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 respective interrupt request flag, LVF, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts.

Note that the trigger source of 0CH vector can be LVD interrupt or the external interrupt, which is selected by the configuration options. The LVD interrupt must be first selected by the configuration options before setting the LVD interrupt function.

EEPROM Interrupt

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

I²C bus Interrupt

An I²C Interrupt request will take place when the I²C Interrupt request flag, IICF, is set, which occurs when a byte of data has been received or transmitted by the I²C interface, I²C address match or I²C time-out. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and the Serial Interface Interrupt enable bit, IICE, must first be set. When the interrupt is enabled, the stack is not full and any of these situations occurs, will take place. When the I²C Interface Interrupt is serviced, the interrupt request flag, IICF, will be automatically reset and the EMI bit will be cleared to disable other interrupts.

A/D Converter Interrupt

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

Note that the trigger source of 18H vector can be Timer/Event Counter 2 interrupt or the A/D converter interrupt, which is selected by the configuration options. The A/D converter interrupt must be first selected by the configuration options before setting the A/D converter interrupt function.

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 Timer/Event Counter 1 interrupt, comparator 2 interrupt and comparator 3 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 Timer/Event Counter 1 interrupt, comparator 2 interrupt and comparator 3 interrupt, will not be automatically reset and must be manually reset by the application program.

Timer/Event Counter Interrupts

There are three Timer/Event Counter interrupts, the Timer/Event Counter interrupt 0/2 is an independent interrupt and the Timer/Event Counter interrupt 1 is contained within the Multi-function Interrupt

For the Timer/Event Counter 0/2 interrupt to occur, the global interrupt enable bit, EMI, and the corresponding timer interrupt enable bit, T0E or T2E, must first be set. An actual Timer/Event Counter interrupt will take place when the Timer/Event Counter request flag, T0F or T2F, is set, a situation that will occur when the relevant Timer/Event Counter overflows. When the interrupt is enabled, the stack is not full and a Timer/Event Counter 0/2 overflow occurs, a subroutine call to the relevant timer interrupt vector, will take place. When the interrupt is serviced, the timer interrupt request flag, T0F or T2F, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts.

Note that the trigger source of 18H vector can be Timer/Event Counter 2 interrupt or the A/D converter interrupt, which is selected by the configuration options. The Timer/Event Counter 2 interrupt must be first selected by the configuration options before setting the Timer/Event Counter 2 interrupt function.

For the Timer/Event Counter 1 interrupt to occur, the global interrupt enable bit, EMI, the relevant Multi-function Interrupt enable bit, MFE and the corresponding timer interrupt enable bit, T1E, must first be set. An actual Timer/Event Counter interrupt will take place when the Timer/Event Counter request flag, T1F, is set, a situation that will occur when the relevant Timer/Event Counter overflows. When the interrupt is enabled, the stack is not full and a Timer/Event Counter 1 overflow occurs, a subroutine call to the relevant timer interrupt vector, will take place. When the interrupt is serviced, the EMI bit will be automatically cleared to disable other interrupts. however only the related MFF flag will be automatically cleared. As the Timer/Event Counter 1 interrupt request flag will not be automatically cleared, it has to be cleared by the application program.

Comparator Interrupt

The comparator interrupt is controlled by three internal comparators. The comparator 1 is an independent interrupt and the comparator 2/3 is contained within the Multi-function Interrupt.

The comparator 1 interrupt request will take place when the comparator 1 interrupt request flag, CP1F is set, a situation that will occur when the comparator output bit changes state. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and comparator interrupt enable bit, CP1E, must first be set. When the interrupt is enabled, the stack is not full and the comparator 1 input generate a comparator output transition, a subroutine call to the comparator interrupt vector, will take place. When the interrupt is serviced, the comparator interrupt request flag, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts.

The comparator 2/3 interrupt request will take place when the comparator 2/3 interrupt request flags, CP2F or CP3F, are set, a situation that will occur when the comparator output bit changes state. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, the relevant Multi-function Interrupt enable bit, MFE and comparator interrupt enable bits, CP2E and CP3E, must first be set. When the interrupt is enabled, the stack is not full and the comparator inputs generate a comparator output transition, a subroutine call to the comparator interrupt vector, will take place. When the interrupt is serviced, the EMI bit will be automatically cleared to disable other interrupts. however only the related MFF flag will be automatically cleared. As the comparator 2/3 interrupt request flag will not be automatically cleared, it has to be cleared by the application program.

Interrupt Wake-up Function

Each of the interrupt functions has the capability of waking up the microcontroller when in the Power down 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 Power down Mode and its system oscillator stopped, situations such as external edge transitions on the external interrupt pin, a low power supply voltage 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 Power down 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.

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 Power down 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 Power down 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.

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

LVD function can be enabled or disabled by configuration option. If the LVD configuration option is enabled, the user can use LVDC bit in system control register 0 to enable/disable the LVD circuit and read the LVD detector status from LVDO bit in system control register 0. The LVD voltage is fixed at 4.4V. The device also provides an interrupt for low voltage detection, see the section of interrupt for the details. In the power down mode, the LVD function will be disabled.

- **CTRL0 Register**

Bit	7	6	5	4	3	2	1	0
Name	C2VOINV	TMR0ECS	PCKPSC2	PCKPSC1	PCKPSC0	OSTPC	LVDO	LVDC
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W
POR	0	0	0	0	0	0	0	0

Bit 7 **C2VOINV**: Inverting control of the comparator 2 output signal
Described elsewhere

Bit 6 **TMR0ECS**: Select TMR0 external clock source
Described elsewhere

Bit 5~3 **PCKPSC2~PCKPSC0**: Peripheral clock prescaler stage
Described elsewhere

Bit 2 **OSTPC**: Oscillator stop function control
Described elsewhere

Bit 1 **LVDO**: Low voltage detector output
0: Normal voltage
1: Low voltage detected

Bit 0 **LVDC**: Low voltage detector control
0: Disable
1: Enable

Configuration Option

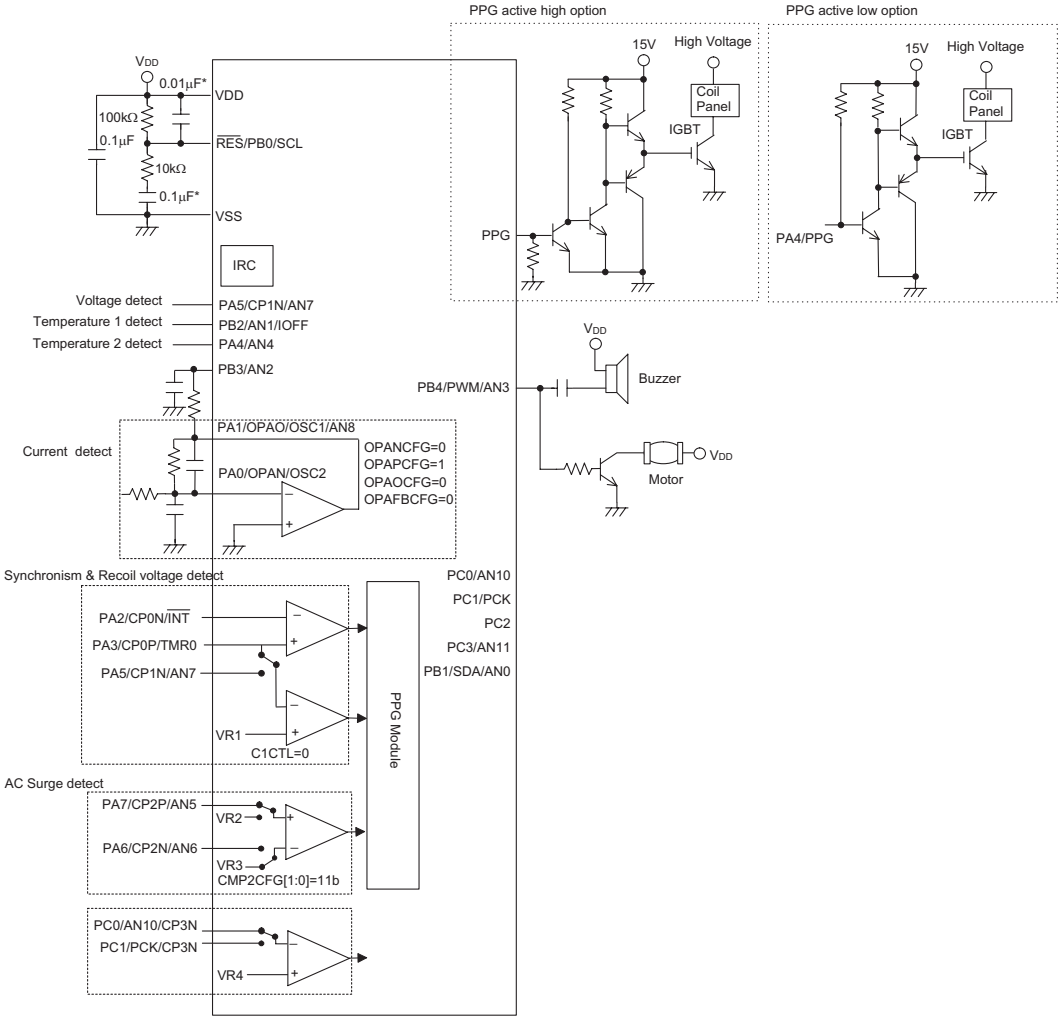
Configuration options refer to certain options within the MCU that are programmed into the device during the programming process. During the development process, these options are selected using the HT-IDE software development tools. As these options are programmed into the device using the hardware 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
Wake-up Option	
1	PA0~PA7 Wake-up Function Control 1. No wake-up 2. wake-up

No.	Options
Pull-high Resistor Option	
2	PA0~PA7 Pull- high Resistor Control 1. No pull-high 2. Pull-high
3	PB0~PB4 Pull- high Resistor Control 1. No pull-high 2. Pull-high
4	PC0~PC3 Pull- high Resistor Control 1. No pull-high 2. Pull-high
I/O or other Options	
5	PB0/$\overline{\text{RES}}$ selection: 1. $\overline{\text{RES}}$ 2. PB0
6	PB0/SCL, PB1/SDA selection: 1. I/O 2. I^2C
7	PB2/IOFF selection: 1. PB2 2. IOFF
8	PB4/PWM selection: 1. PB4 2. PWM
9	PC1/PCK selection: 1. PC1 2. PCK
Oscillator Option	
10	System Oscillator Option 1. HIRC 2. HXT
OPA Option	
11	OPA Function: 1. Disable 2. Enable with output to PA1/OPAO/AN8/OSC1 3. Enable with output through OPAR4 to PB3/AN2
12	OPAN Option: 1. Without OPAR3 2. With OPAR3
13	OPAP Option: 1. Directly to VSS 2. With OPAR2 to VSS
14	OPAO Option: 1. No feedback 2. With OPAR1 feedback
PPG Option	
15	PPG polarity control 1. PPG output active high 2. PPG output active low
16	PPG start count synchronize with clock 1. Synchronized with clock 2. Asynchronous with clock
Comparator Option	
17	Comparator 0 Control 1. Disable 2. Enable

No.	Options
18	Comparator 1 Control 1. Disable 2. Enable
19	Comparator 1 de-bounce Control 1. Disable 2. Enable
20	Comparator 2 Control 1. Disable 2. Enable, CP2N=V_{R3}, CP2P=PA7 3. Enable, CP2N=PA6, CP2P=V_{R2} 4. Enable, CP2N=PA6, CP2P=PA7
21	Comparator 2 de-bounce Control 1. Disable 2. Enable
22	Comparator 3 Control 1. Disable 2. Enable, CP3N=PC0, CP3P=V_{R4} 3. Enable, CP3N=PC1, CP3P=V_{R4} 4. Enable, OPAO connected to CP3N, CP3P=V_{R4}
23	Comparator 3 de-bounce Control 1. Disable 2. Enable
Interrupt Option	
24	Interrupt vector 18H source selection 1. Timer/Event Counter 2 interrupt 2. A/D Converter interrupt
25	Interrupt vector 0CH source selection 1. External interrupt 2. LVD interrupt
WDT Option	
26	WDT control 1. Disable 2. Enable
27	WDT time-out Period Selection 1. $2^{16}/f_S$ 2. $2^{15}/f_S$ 3. $2^{14}/f_S$ 4. $2^{13}/f_S$
28	WDT Clock Source Selection 1. WDTOSC 2. $f_{SYS}/4$
29	Clear WDT Instruction 1. 1 instruction 2. 2 instructions
LVD/LVR Selection	
30	LVD function 1. Disable 2. Enable
31	LVR function 1. Disable 2. Enable
32	LVR Voltage Selection 1. 2.1V 2. 3.0V

Application Circuits



Note: The resistance and capacitance for reset circuit should be designed in such a way as to ensure that the VDD is stable and remains within a valid operating voltage range before bringing $\overline{RES}$ to high.

"*" Make the length of the wiring, which is connected to the $\overline{RES}$ pin as short as possible, to avoid noise interference.

Instruction Set

Introduction

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

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

Instruction Timing

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

Moving and Transferring Data

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

Arithmetic Operations

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

Logical and Rotate Operation

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

Branches and Control Transfer

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

Bit Operations

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

Table Read Operations

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

Other Operations

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

Instruction Set Summary

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

Table Conventions

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

Mnemonic	Description	Cycles	Flag Affected
Arithmetic			
ADD A,[m]	Add Data Memory to ACC	1	Z, C, AC, OV
ADDM A,[m]	Add ACC to Data Memory	1[Note]	Z, C, AC, OV
ADD A,x	Add immediate data to ACC	1	Z, C, AC, OV
ADC A,[m]	Add Data Memory to ACC with Carry	1	Z, C, AC, OV
ADCM A,[m]	Add ACC to Data memory with Carry	1[Note]	Z, C, AC, OV
SUB A,x	Subtract immediate data from the ACC	1	Z, C, AC, OV
SUB A,[m]	Subtract Data Memory from ACC	1	Z, C, AC, OV
SUBM A,[m]	Subtract Data Memory from ACC with result in Data Memory	1[Note]	Z, C, AC, OV
SBC A,[m]	Subtract Data Memory from ACC with Carry	1	Z, C, AC, OV
SBCM A,[m]	Subtract Data Memory from ACC with Carry, result in Data Memory	1[Note]	Z, C, AC, OV
DAA [m]	Decimal adjust ACC for Addition with result in Data Memory	1[Note]	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	1[Note]	Z
ORM A,[m]	Logical OR ACC to Data Memory	1[Note]	Z
XORM A,[m]	Logical XOR ACC to Data Memory	1[Note]	Z
AND A,x	Logical AND immediate Data to ACC	1	Z
OR A,x	Logical OR immediate Data to ACC	1	Z
XOR A,x	Logical XOR immediate Data to ACC	1	Z
CPL [m]	Complement Data Memory	1[Note]	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	1[Note]	Z
DECA [m]	Decrement Data Memory with result in ACC	1	Z
DEC [m]	Decrement Data Memory	1[Note]	Z
Rotate			
RRA [m]	Rotate Data Memory right with result in ACC	1	None
RR [m]	Rotate Data Memory right	1[Note]	None
RRCA [m]	Rotate Data Memory right through Carry with result in ACC	1	C
RRC [m]	Rotate Data Memory right through Carry	1[Note]	C
RLA [m]	Rotate Data Memory left with result in ACC	1	None
RL [m]	Rotate Data Memory left	1[Note]	None
RLCA [m]	Rotate Data Memory left through Carry with result in ACC	1	C
RLC [m]	Rotate Data Memory left through Carry	1[Note]	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 up to two cycles are required, if no skip takes place only one cycle is required.
2. Any instruction which changes the contents of the PCL will also require 2 cycles for execution.
3. For the “CLR WDT1” and “CLR WDT2” instructions the TO and PDF flags may be affected by the execution status. The TO and PDF flags are cleared after both “CLR WDT1” and “CLR WDT2” instructions are consecutively executed. Otherwise the TO and PDF flags remain unchanged.

Instruction Definition

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

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

Operation ACC ← ACC + [m] + C

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

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

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

Operation [m] ← ACC + [m] + C

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

ADD A,[m] Add Data Memory to ACC

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

Operation ACC ← ACC + [m]

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

ADD A,x Add immediate data to ACC

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

Operation ACC ← ACC + x

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

ADDM A,[m] Add ACC to Data Memory

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

Operation [m] ← ACC + [m]

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

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

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

Operation ACC ← ACC "AND" [m]

Affected flag(s) Z

AND A,x Logical AND immediate data to ACC

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

Operation ACC ← ACC "AND" x

Affected flag(s) Z

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

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

Operation [m] ← ACC "AND" [m]

Affected flag(s) Z

CALL addr Subroutine call

Description Unconditionally calls a subroutine at the specified address. The Program Counter then increments by 1 to obtain the address of the next instruction which is then pushed onto the stack. The specified address is then loaded and the program continues execution from this new address. As this instruction requires an additional operation, it is a two cycle instruction.

Operation Stack ← Program Counter + 1
Program Counter ← addr

Affected flag(s) None

CLR [m] Clear Data Memory

Description Each bit of the specified Data Memory is cleared to 0.

Operation [m] ← 00H

Affected flag(s) None

CLR [m].i Clear bit of Data Memory

Description Bit i of the specified Data Memory is cleared to 0.

Operation [m].i ← 0

Affected flag(s) None

CLR WDT Clear Watchdog Timer

Description The TO, PDF flags and the WDT are all cleared.

Operation WDT cleared
TO ← 0
PDF ← 0

Affected flag(s) TO, PDF

CLR WDT1 Pre-clear Watchdog Timer

Description The TO, PDF flags and the WDT are all cleared. Note that this instruction works in conjunction with CLR WDT2 and must be executed alternately with CLR WDT2 to have effect. Repetitively executing this instruction without alternately executing CLR WDT2 will have no effect.

Operation WDT cleared
TO ← 0
PDF ← 0

Affected flag(s) TO, PDF

CLR WDT2 Pre-clear Watchdog Timer

Description The TO, PDF flags and the WDT are all cleared. Note that this instruction works in conjunction with CLR WDT1 and must be executed alternately with CLR WDT1 to have effect. Repetitively executing this instruction without alternately executing CLR WDT1 will have no effect.

Operation WDT cleared
TO ← 0
PDF ← 0

Affected flag(s) TO, PDF

CPL [m] Complement Data Memory

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

Operation [m] ← $\overline{[m]}$

Affected flag(s) Z

CPLA [m] Complement Data Memory with result in ACC

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

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

Affected flag(s) Z

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

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

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

Affected flag(s) C

DEC [m] Decrement Data Memory

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

Operation [m] ← [m] − 1

Affected flag(s) Z

DECA [m] Decrement Data Memory with result in ACC

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

Operation ACC ← [m] − 1

Affected flag(s) Z

HALT Enter power down mode

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

Operation TO ← 0
PDF ← 1

Affected flag(s) TO, PDF

INC [m] Increment Data Memory

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

Operation [m] ← [m] + 1

Affected flag(s) Z

INCA [m] Increment Data Memory with result in ACC

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

Operation ACC ← [m] + 1

Affected flag(s) Z

JMP addr Jump unconditionally

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

Operation Program Counter ← addr

Affected flag(s) None

MOV A,[m] Move Data Memory to ACC

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

Operation ACC ← [m]

Affected flag(s) None

MOV A,x Move immediate data to ACC

Description The immediate data specified is loaded into the Accumulator.

Operation ACC ← x

Affected flag(s) None

MOV [m],A Move ACC to Data Memory

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

Operation [m] ← ACC

Affected flag(s) None

NOP No operation

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

Operation No operation

Affected flag(s) None

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

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

Operation ACC ← ACC ″OR″ [m]

Affected flag(s) Z

OR A,x Logical OR immediate data to ACC

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

Operation ACC ← ACC ″OR″ x

Affected flag(s) Z

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

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

Operation [m] ← ACC ″OR″ [m]

Affected flag(s) Z

RET Return from subroutine

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

Operation Program Counter ← Stack

Affected flag(s) None

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

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

Operation Program Counter ← Stack
ACC ← x

Affected flag(s) None

RETI Return from interrupt

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

Operation Program Counter ← Stack
EMI ← 1

Affected flag(s) None

RL [m] Rotate Data Memory left

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

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

Affected flag(s) None

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

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

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

Affected flag(s) None

RLC [m] Rotate Data Memory left through Carry

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

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

Affected flag(s) C

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

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

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

Affected flag(s) C

RR [m] Rotate Data Memory right

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

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

Affected flag(s) None

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

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

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

Affected flag(s) None

RRC [m] Rotate Data Memory right through Carry

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

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

Affected flag(s) C

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

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

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

Affected flag(s) C

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

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

Operation ACC ← ACC − [m] − C

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

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

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

Operation [m] ← ACC − [m] − C

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

SDZ [m] Skip if decrement Data Memory is 0

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

Operation [m] ← [m] − 1
Skip if [m]=0

Affected flag(s) None

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

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

Operation ACC ← [m] − 1
Skip if ACC=0

Affected flag(s) None

SET [m] Set Data Memory

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

Operation [m] ← FFH

Affected flag(s) None

SET [m].i Set bit of Data Memory

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

Operation [m].i ← 1

Affected flag(s) None

SIZ [m] Skip if increment Data Memory is 0

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

Operation [m] ← [m] + 1
Skip if [m]=0

Affected flag(s) None

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

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

Operation ACC ← [m] + 1
Skip if ACC=0

Affected flag(s) None

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

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

Operation Skip if [m].i ≠ 0

Affected flag(s) None

SUB A,[m] Subtract Data Memory from ACC

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

Operation ACC ← ACC − [m]

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

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

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

Operation [m] ← ACC − [m]

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

SUB A,x Subtract immediate data from ACC

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

Operation ACC ← ACC − x

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

SWAP [m] Swap nibbles of Data Memory

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

Operation [m].3~[m].0 ↔ [m].7~[m].4

Affected flag(s) None

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

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

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

Affected flag(s) None

SZ [m] Skip if Data Memory is 0

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

Operation Skip if [m]=0

Affected flag(s) None

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

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

Operation ACC ← [m]
Skip if [m]=0

Affected flag(s) None

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

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

Operation Skip if [m].i=0

Affected flag(s) None

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

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

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

Affected flag(s) None

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

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

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

Affected flag(s) None

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

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

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

Affected flag(s) None

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

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

Operation ACC ← ACC ″XOR″ [m]

Affected flag(s) Z

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

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

Operation [m] ← ACC ″XOR″ [m]

Affected flag(s) Z

XOR A,x Logical XOR immediate data to ACC

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

Operation ACC ← ACC ″XOR″ x

Affected flag(s) Z

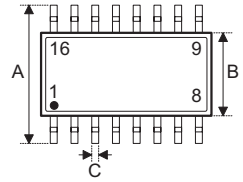
Package Information

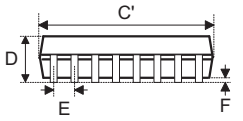
Note that the package information provided here is for consultation purposes only. As this information may be updated at regular intervals users are reminded to consult the Holtek website for the latest version of the Package/Carton Information.

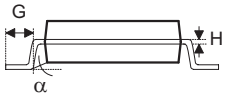
Additional supplementary information with regard to packaging is listed below. Click on the relevant section to be transferred to the relevant website page.

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

16-pin NSOP (150mil) Outline Dimensions



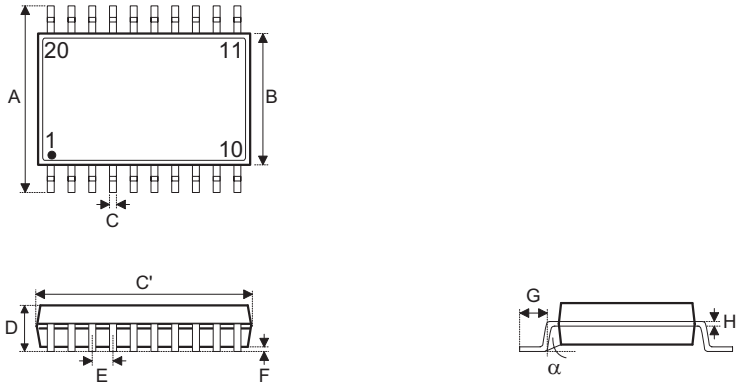




Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A		0.236 BSC	
B		0.154 BSC	
C	0.012	—	0.020
C'		0.390 BSC	
D	—	—	0.069
E		0.050 BSC	
F	0.004	—	0.010
G	0.016	—	0.050
H	0.004	—	0.010
α	0°	—	8°

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A		6.000 BSC	
B		3.900 BSC	
C	0.31	—	0.51
C'		9.900 BSC	
D	—	—	1.75
E		1.270 BSC	
F	0.10	—	0.25
G	0.40	—	1.27
H	0.10	—	0.25
α	0°	—	8°

20-pin SOP (300mil) Outline Dimensions



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

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A		10.30 BSC	
B		7.50 BSC	
C	0.31	—	0.51
C’		12.80 BSC	
D	—	—	2.65
E		1.27 BSC	
F	0.10	—	0.30
G	0.40	—	1.27
H	0.20	—	0.33
α	0°	—	8°

Copyright© 2024 by HOLTEK SEMICONDUCTOR INC. All Rights Reserved.

The information provided in this document has been produced with reasonable care and attention before publication, however, HOLTEK does not guarantee that the information is completely accurate. The information contained in this publication is provided for reference only and may be superseded by updates. HOLTEK disclaims any expressed, implied or statutory warranties, including but not limited to suitability for commercialization, satisfactory quality, specifications, characteristics, functions, fitness for a particular purpose, and non-infringement of any third-party's rights. HOLTEK disclaims all liability arising from the information and its application. In addition, HOLTEK does not recommend the use of HOLTEK's products where there is a risk of personal hazard due to malfunction or other reasons. HOLTEK hereby declares that it does not authorise the use of these products in life-saving, life-sustaining or safety critical components. Any use of HOLTEK's products in life-saving/sustaining or safety applications is entirely at the buyer's risk, and the buyer agrees to defend, indemnify and hold HOLTEK harmless from any damages, claims, suits, or expenses resulting from such use. The information provided in this document, including but not limited to the content, data, examples, materials, graphs, and trademarks, is the intellectual property of HOLTEK (and its licensors, where applicable) and is protected by copyright law and other intellectual property laws. No license, express or implied, to any intellectual property right, is granted by HOLTEK herein. HOLTEK reserves the right to revise the information described in the document at any time without prior notice. For the latest information, please contact us.