HOLTEK

Ultrasonic Distance Measurement Flash MCU

HT45F39/HT45F391

Revision: V1.30 Date: February 20, 2019

www.holtek.com

Table of Contents

Features

CPU Features

- Internal Shunt Regulator: 5V
- Power Supply input voltage
    - HT45F39: 8V~16V
    - HT45F391: 4.5V~5.5V
- Up to 0.25μs instruction cycle with 16MHz system clock
- Power down and wake-up functions to reduce power consumption
- Oscillator types
    - Internal RC – HIRC
    - Internal 32kHz RC – LIRC
- Operating mode: Normal and Sleep
- Fully integrated internal 32kHz and 16MHz oscillator requires no external components
- All instructions executed in one or two machine Cycles
- Table read instructions
- 63 powerful instructions
- 4-level subroutine nesting
- Bit manipulation instruction

Peripheral Features

- Flash Program Memory: 2K×16
- RAM Data Memory: 160×8
- Watchdog Timer function
- Up to 11 bidirectional I/O lines
- Dual 10-bit Timer Module for time measure, compare match output, PWM output function
- Dual Operational Amplifiers functions
- Up to 8 channel high speed 8-bit ADC
- Switch Capacitor Filter with gain control output to internal high speed 8-bit ADC function
- Auto-Envelope Processor(AEP) for input signal processing
- Special Bus Control Unit interface for Lin Bus like data communication
- Low voltage reset function
- 16-pin NSOP package type

General Description

The series of devices are Flash Memory A/D type 8-bit high performance RISC architecture microcontrolls. Offering users the convenience of Flash Memory multi-programming features, these devices also includes a wide range of functions and features. Other memory include an area of RAM Data Memory.

Analog features include an integrated multi-channel high- speed Analog to Digital Converter, dual Operational Amplifiers, Switch Capacitor Filter with Gain control and one internal 5.0V Shunt Regulator for voltage regulator, communication with the outside world is catered for by including fully integrated Bus Control Unit (BCU) interface, which provide designers with a means of fast and easy communication with external peripheral hardware by Lin Bus like data communication. Protective features such as an internal Watchdog Timer and Low Voltage Reset coupled with excellent noise immunity and ESD protection ensure that reliable operation is maintained in hostile electrical environments.

A full choice of HIRC and LIRC oscillator functions are provided including a fully integrated system oscillator which requires no external components for its implementation. The usual Holtek MCU features such as power down and wake-up functions, oscillator options, etc. combine to ensure user applications require a minimum of external components.

The inclusion of flexible I/O programming features, along with many other features ensure that the devices will find excellent use in applications such as car parking systems, distance control products, measurement equipment and many others.

Selection Table

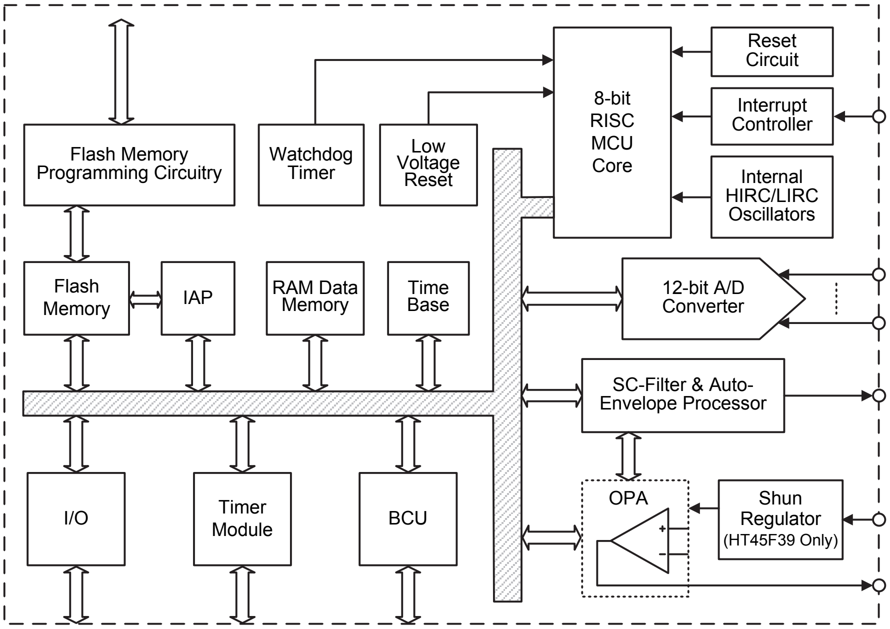
Most features are common to all devices and the main features distinguishing them are the I/O pin count and Level Shift output pins. The following table summarises the main features of each device.

Part No.	VDD	System Clock	Input Voltage	Program Memory	Data Memory	I/O	Timer Module
HT45F39	5.0V	400kHz~16MHz or 32kHz	8V~16V	2K×16	160×8	11	10-bit CTM×2
HT45F391	4.5V~5.5V		—				

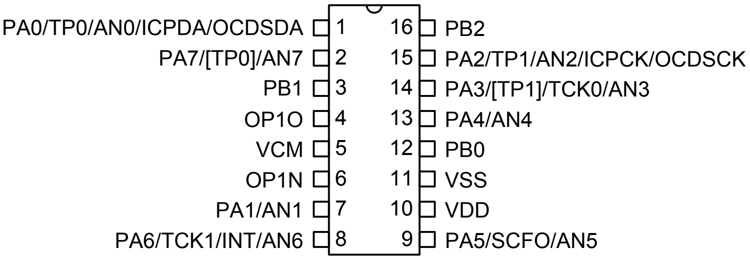
Part No.	A/D	OPA	SCF	AEP	Interface	Stack	Package
HT45F39	8-bit×8	2	1	1	BCU	4	16NSOP
HT45F391							

Block Diagram

The following block diagram illustrates the main functional blocks.



Pin Assignment



HT45F39/HT45F391
16 NSOP-A

Note: If the pin-shared pin functions have multiple outputs, the priority output function is listed from right to left.

Pin Descriptions

Pin Name	Function	OPT	I/T	O/T	Descriptions
PA0/TP0/ AN0/ICPDA/ OCDSDA	PA0	PAPU PAWK	ST	CMOS	General purpose I/O. Register enabled pull-up, wake-up.
	TP0	—	ST	CMOS	Timer0 I/O
	AN0	ADCR ACERL	AN	—	A/D channel 0
	ICPDA	—	ST	CMOS	ICP Data/Address pin
	OCDSDA	—	ST	CMOS	OCDS Data/Address pin, for EV chip only
PA1/AN1	PA1	PAPU PAWK	ST	CMOS	General purpose I/O. Register enabled pull-up, wake-up.
	AN1	ADCR ACERL	AN	—	A/D channel 1
PA2/TP1/ AN2/ICPCK/ OCDSCK	PA2	PAPU PAWK	ST	CMOS	General purpose I/O. Register enabled pull-up, wake-up.
	TP1	—	ST	CMOS	Timer1 I/O
	AN2	ADCR ACERL	AN	—	A/D channel 2
	ICPCK	—	ST	CMOS	ICP Clock pin
	OCDSCK	—	ST	CMOS	OCDS Clock pin, for EV chip only
PA3/[TP1]/ TCK0/AN3	PA3	PAPU PAWK	ST	CMOS	General purpose I/O. Register enabled pull-up, wake-up.
	TP1	—	ST	CMOS	Timer1 I/O
	TCK0	—	ST	—	External Timer0 clock input
	AN3	ADCR ACERL	AN	—	A/D channel 3
PA4/AN4	PA4	PAPU PAWK	ST	CMOS	General purpose I/O. Register enabled pull-up, wake-up.
	AN4	ADCR ACERL	AN	—	A/D channel 4
PA5/SCFO/ AN5	PA5	PAPU PAWK	ST	CMOS	General purpose I/O. Register enabled pull-up, wake-up.
	SCFO	ADCR	AN	—	SCF output pin
	AN5	ADCR ACERL	AN	—	A/D channel 5
PA6/TCK1/ INT/AN6	PA6	PAPU PAWK	ST	CMOS	General purpose I/O. Register enabled pull-up, wake-up.
	TCK1	—	ST	—	External Timer1 clock input
	INT	—	ST	—	External interrupt input
	AN6	ADCR ACERL	AN	—	A/D channel 6
PA7/[TP0]/ AN7	PA7	PAPU PAWK	ST	CMOS	General purpose I/O. Register enabled pull-up, wake-up.
	TP0	—	ST	CMOS	Timer0 I/O
	AN7	ADCR ACERL	AN	—	A/D channel 7
PB0~PB2	PB0~PB2	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up

Pin Name	Function	OPT	I/T	O/T	Descriptions
OP1N	A1N	—	OPAI	—	OPA1 Negative input
OP1O	A1O	—	—	OPAO	OPA1 Output
VCM	VCM	—	AN	—	VCM reference voltage input
VDD	VDD	—	PWR	—	Power supply
VSS	VSS	—	PWR	—	Ground

Note: I/T: Input type
O/T: Output type
OPT: Optional by register option
PWR: Power
ST: Schmitt Trigger input
CMOS: CMOS output;
OPAI: Operational Amplifier input
OPAO: Operational Amplifier output

Absolute Maximum Ratings

MCU Supply Voltage	V_{SS}-0.3V to V_{SS}+6.0V
Input Voltage	V_{SS}-0.3V to V_{DD}+0.3V
I_{OL} Total	150mA
Total Power Dissipation	500mW
Storage Temperature	-50°C to 150°C
Operating Temperature	-40°C to 85°C
I_{OH} Total	-80mA

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

D.C. Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	Operating Voltage (Note)	—	—	4.5	—	5.5	V
I_{DD}	Operating Current	5V	No load,ADC disabled	—	6	9	mA
I_{STB}	Standby Current	5V	No load, all off except WDT enable	—	—	5	μA
V_{IL}	Input Low Voltage for PA, PB, TCKn, TPn, INT	5V	—	0	—	1.5	V
		4.5V~5.5V		0	—	$0.2V_{DD}$	V
V_{IH}	Input High Voltage for PA, PB, TCKn, TPn, INT	5V	—	3.5	—	5	V
		4.5V~5.5V		$0.8V_{DD}$	—	V_{DD}	V
V_{LVR}	Low Voltage Reset	5V	—	-5%	3.15	+5%	V
V_{BG}	Reference Voltage with Buffer Voltage	—	—	-3%	1.04	+3%	V
t_{BGS}	V_{BG} Turn On Stable Time	—	—	—	—	150	μs
I_{BG}	Additional Power Consumption if Reference with Buffer is used	—	—	—	200	300	μA
I_{LVR}	Additional Power Consumption if LVR is Used	5V	LVR enable	—	15	30	μA
I_{OH}	I/O Source Current (PA, PB)	5V	$V_{OH}=0.9V_{DD}$	-5	-10	—	mA
I_{OL}	I/O Sink Current (PA, PB)	5V	$V_{OL}=0.1V_{DD}$	10	20	—	mA
R_{PH}	Pull-high Resistance (I/O)	5V	—	10	30	50	kΩ

Note: For HT45F39 an external resistor should be serially connected between the supply power and VDD pin. Refer to the "Shunt Regulator" chapter for the details.

A.C. Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
f_{SYS}	System Clock	5V	4.5V~5.5V	400	—	16000	kHz
f_{HIRC}	HIRC OSC	4.75V~5.25V	Ta=-20°C~60°C	-2.0%	16	+2.0%	MHz
			Ta=-40°C~85°C	-2.5%	16	+2.5%	MHz
f_{LIRC}	System Clock (LIRC)	5V	Ta=25°C	-10%	32	+10%	kHz
		2.2V~5.5V	Ta=-40°C ~85°C	-50%	32	+60%	kHz
t_{TIMER}	TCKn Input Pin Minimum Pulse Width	—	—	0.3	—	—	μs
t_{INT}	Interrupt Minimum Pulse Width	—	—	10	—	—	μs
t_{LVR}	Low Voltage width to Reset	—	—	120	240	480	μs
t_{SST}	System Start-up Timer Period	—	—	16	—	—	t_{SYS}
t_{RSTD}	System Reset Delay Time (Power On Reset)	—	—	25	50	100	ms
	System Reset Delay Time (Any Reset except Power On Reset)	—	—	8.3	16.7	33.3	ms

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

2. To maintain the accuracy of the internal HIRC oscillator frequency, a 0.1mF decoupling capacitor should be connected between V_{DD} and V_{SS} and located as close to the devices as possible.

A/D Characteristics

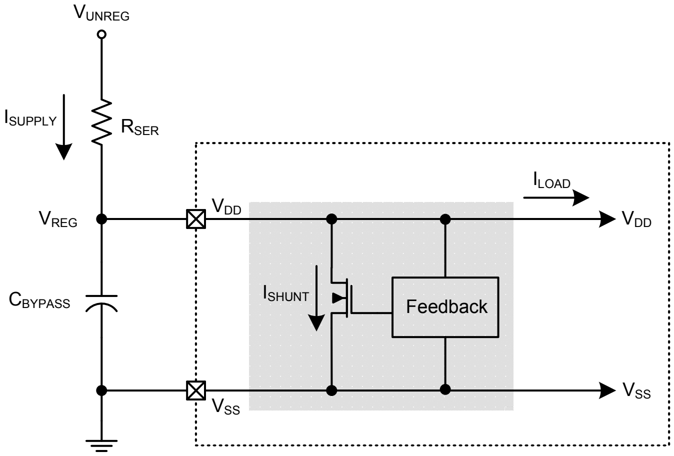
Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
AV_{DD}	Analog Operating Voltage	—	—	4.5	—	5.5	V
V_{AD}	AD Input Voltage	—	—	0	—	AV_{DD}	V
DNL	Differential Non-linearity	5V	AV_{DD}=V_{DD} t_{ADC} =1.25μs	-2	—	+2	LSB
INL	Integral Non-linearity	5V	AV_{DD}=V_{DD} t_{ADC} =1.25μs	-4	—	+4	LSB
I_{ADC}	Additional Power Consumption if ADC is Enabled	5V	No load (t_{ADC} =1.25μs) Only ADC Enabled, Other functions disabled	—	0.6	—	mA
f_{AD}	A/D Clock	4.5~5.5V	—	0.125	—	8	MHz
t_{ADC}	AD Conversion Time	5V	8 bit ADC	—	10	—	t_{AD}
t_{ON2ST}	ADC on to ADC Start	4.5V~5.5V	—	2	—	—	μs

Shunt Regulator Electrical Characteristics – HT45F39 Only

Ta=25°C

Symbol	Parameter	Condition	Min.	Typ.	Max.	Unit
V_{UNREG}	Supply Voltage	—	8	—	16	V
V_{REG1}	Load Voltage – No Load in Room Temperature	V_{UNREG}=8V~16V, I_{SUPPLY}=$I_{SHUNT(Max)}$ @V_{UNREG}=16V MCU in SLEEP mode, no load (1)	-3%	5	+3%	V
V_{REG2}	Load Voltage – Load and Supply Voltage within Specification.	I_{SUPPLY}=$I_{SHUNT(Max)}$ @ V_{UNREG}=16V Select RSER (1) as MCU in SLEEP mode with no load condition (1) V_{UNREG}=8V, I_{LOAD}=$I_{LOAD(Max)}$ (2) V_{UNREG}=16V, I_{LOAD}=0	-5%	5	+5%	V
I_{LOAD}	Load Current (2)	V_{UNREG} =8V~16V, R_{SER}=(16-V_{DD1})/I_{SHUNT}(Max)	0	—	15	mA
$I_{SHUNT(Max)}$	Maximum Shunt Current	V_{UNREG}=16V	80	—	—	mA
$I_{SHUNT(Min)}$	Maximum Static Current	V_{UNREG}=16V	—	—	5	mA
ΔV_{LINR}	Line Regulation	V_{UNREG}=8V to 16V, R_{SER}=(16V–V_{REG1})/$I_{SHUNT(Max)}$ MCU in SLEEP mode, no load	—	—	0.3	%/V
Δ_{VREG_RIPPLE}	Output Voltage Ripple	V_{UNREG}=8V to 16V, R_{SER}=(16V–V_{REG1})/$I_{SHUNT(Max)}$ MCU alternately consumes the average current I_{MCU} and $I_{LOAD(Max)}$. The MCU varies its consumption current once every 0.5ms (3)	—	—	100	mV
Δ_{VLOAD}	Load Regulation	V_{UNREG}=8V~16V/8V, 12V or 16V R_{SER}=(16V–V_{REG1})/$I_{SHUNT(Max)}$ I_{LOAD} = 0mA to $I_{LOAD(Max)}$, C_{BYPASS}=4.7μF, MCU in SLEEP mode	—	—	0.03	%/mA



Note: (1) $R_{SER} = (V_{UNREG} - V_{REG}) / I_{SUPPLY}$

(2) The load current maximum specification value is calculated based on the following formula.

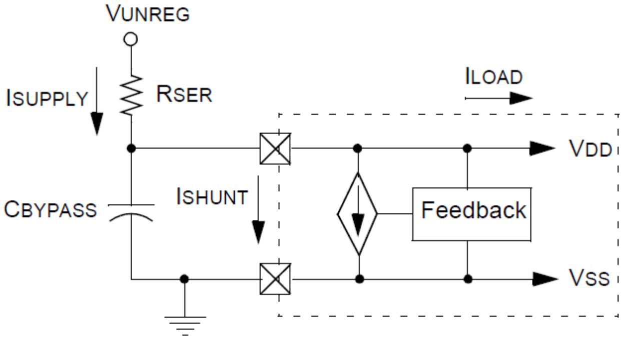
$I_{LOAD(Max)} = I_{SHUNT(Max)} \times (V_{UNREG(Min)} - V_{REG1(Max)}) / (V_{UNREG(Max)} - V_{REG1(Min)}) - I_{STATIC(Measured)}$

$V_{UNREG(MAX)}$, $V_{REG1(Max)}$=Maximum specification value for V_{UNREG} and V_{REG1} respectively

$V_{UNREG(Min)}$, $V_{REG1(Min)}$=Minimum specification value for V_{UNREG} and V_{REG1} respectively

(3) I_{MCU} = MCU current consumption measured when the MCU toggles two I/O pins every 0.5ms with no load.

Shunt Regulator operational block diagram:



$R_{SER} = (V_{UNREG} - V_{DD}) / I_{SUPPLY}$

$I_{SUPPLY} = I_{SHUNT} + I_{LOAD}$

$I_{LOAD} = I_{MCU} = I_{SUPPLY} - I_{SHUNT}$

SCF Electrical Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions	Min	Typ	Max	Unit
SCV_{DD}	Power Supply Voltage	—	4.5	—	5.5	Volt
SCGAIN	Gain Range	—	6	—	30	DB
SCBW	Bandwidth	Connect RL(=2KΩ) to a DC Level(=2.5V)	—	3K	—	Hz
PMRR	Output Range	—	—	4	—	V
AOL	Filter Clock	V_{DD}= all range	—	f_{SYS}/20	—	Hz

Op Amplifier Electrical Characteristics

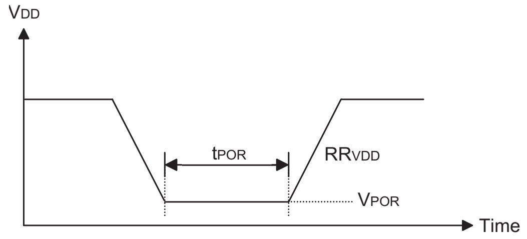
Ta=25°C

Symbol	Parameter	Test Conditions	Min	Typ	Max	Unit
$V_{I(R)}$	Input Common Mode Voltage	—	0	—	V_{DD}-1.4	V
I_{PD}	Power Down Current	—	—	—	0.1	μA
PMRR	Power Mode Rejection Ratio	—	80	—	—	dB
AOL	Open Loop Gain	V_{DD}= all range	100	—	—	dB
CMRR	Common Mode Rejection Ratio	—	80	—	—	dB
UGBW	Unit Gain Band Width	V_{CM}=2.5V	5	—	—	MHz

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



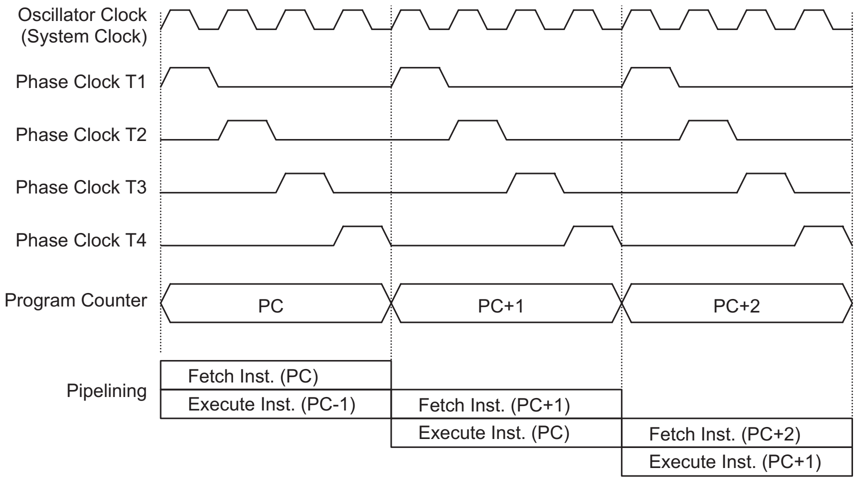
System Architecture

A key factor in the high-performance features of the Holtek range of microcontrollers is attributed to the internal system architecture. The range of devices take advantage of the usual features found within RISC microcontrollers providing increased speed of operation and enhanced performance. The pipelining scheme is implemented in such a way that instruction fetching and instruction execution are overlapped, hence instructions are effectively executed in one cycle, with the exception of branch or call instructions. An 8-bit wide ALU is used in practically all operations of the instruction set. It 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 devices suitable for low-cost, high-volume production for controller applications.

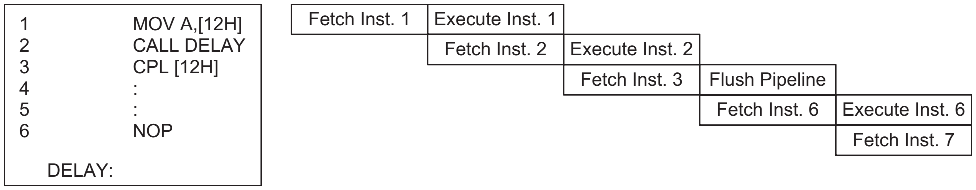
Clocking and Pipelining

The main system clock, derived from 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.

For instructions involving branches, such as jump or call instructions, two instruction cycles are required to complete instruction execution. An extra cycle is required as the program takes one cycle to first obtain the actual jump or call address and then another cycle to actually execute the branch. The requirement for this extra cycle should be taken into account by programmers in timing sensitive applications.



System Clocking and Pipelining



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

  DELAY:
```

Instruction Fetching

Program Counter

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

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

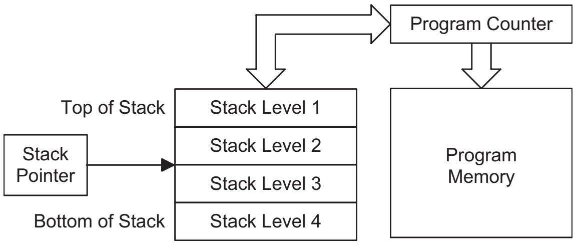
Program Counter	
High Byte	Low Byte (PCL Register)
PC10~PC8	PCL7~PCL0

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

The lower byte of the Program Counter is fully accessible under program control. Manipulating the PCL might cause program branching, so an extra cycle is needed to pre-fetch. Further information on the PCL register can be found in the Special Function Register section.

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 4 levels and is neither part of the Data or Program Memory space, and is neither readable nor writeable. The activated level is indexed by the Stack Pointer, SP, 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.

Arithmetic and Logic Unit – ALU

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

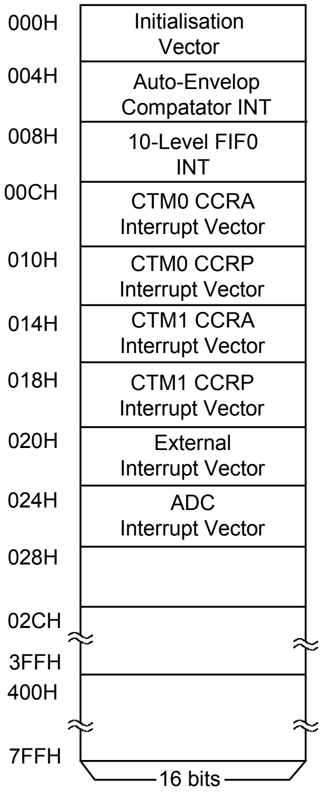
- Arithmetic operations: ADD, ADDM, ADC, ADCM, SUB, SUBM, SBC, SBCM, DAA
- Logic operations: AND, OR, XOR, ANDM, ORM, XORM, CPL, CPLA
- Rotation RRA, RR, RRCA, RRC, RLA, RL, RLCA, RLC
- Increment and Decrement INCA, INC, DECA, DEC
- Branch decision, JMP, SZ, SZA, SNZ, SIZ, SDZ, SIZA, SDZA, CALL, RET,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

Flash Program Memory

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

Structure

The Program Memory has a capacity of 2Kx16. 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 separate table pointer registers.

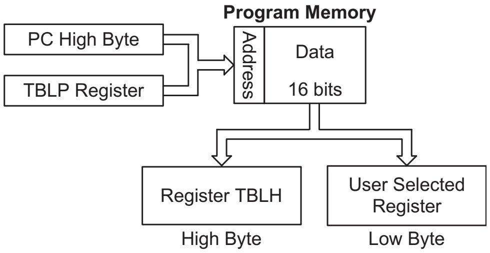

000H
Initialisation Vector
004H
Auto-Envelop Compatator INT
008H
10-Level FIF0 INT
00CH
CTM0 CCRA Interrupt Vector
010H
CTM0 CCRP Interrupt Vector
014H
CTM1 CCRA Interrupt Vector
018H
CTM1 CCRP Interrupt Vector
020H
External Interrupt Vector
024H
ADC Interrupt Vector
028H
02CH
3FFH
400H
7FFH
16 bits

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 lower order address of the look up data to be retrieved in the table pointer register, TBLP. This register defines the lower 8-bit address of the look-up table. After setting up the table pointer, the table data can be retrieved from the current Program Memory page or last Program Memory page using the TABRDC[m] or TABRDL [m] instructions, respectively. When these instructions are 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 following diagram illustrates the addressing/data flow of the look-up table:



Instruction	Table Location Bits										
	b10	B9	B8	B7	B6	B5	B4	B3	B2	B1	B0
TABRDC [m]	PC10	PC9	PC8	@7	@6	@5	@4	@3	@2	@1	@0
TABRDL [m]	1	1	1	@7	@6	@5	@4	@3	@2	@1	@0

Note: PC10~PC8: Current Program Counter bits

@7~@0: Table Pointer TBLP bits

Table Program Example

The accompanying example shows how the table pointer and table data is defined and retrieved from the devices. This example uses raw table data located in the last page which is stored there using the ORG statement. The value at this ORG statement is "0700H" which refers to the start address of the last page within the 2K Program Memory of the devices. The table pointer is setup here to have an initial value of "06H". This will ensure that the first data read from the data table will be at the Program Memory address "0706H" or 6 locations after the start of the last page. Note that the value for the table pointer is referenced to the first address of the present page if the "TABRDC [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 "TABRDL [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 the table read instructions. If using the table read instructions, the Interrupt Service Routines may change the value of 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 table pointer - note that this address
                        ; is referenced to the last page or present page
mov  tblp,a
:
:
tabrdl tempreg1         ; transfers value in table referenced by table pointer
                        ; to tempregl
                        ; data at prog. memory address "0706H" transferred
                        ; to tempreg1 and TBLH
dec  tblp               ; reduce value of table pointer by one
tabrdl tempreg2         ; transfers value in table referenced by table pointer
                        ; to tempreg2
                        ; data at prog.memory address "0705H" transferred to
                        ; tempreg2 and TBLH
                        ; in this example the data "1AH" is transferred to
                        ; tempreg1 and data "0FH" to register tempreg2
                        ; the value "00H" will be transferred to the high byte
:                       ; register TBLH
:
org  0700h              ; sets initial address of last page
dc   00Ah, 00Bh, 00Ch, 00Dh, 00Eh, 00Fh, 01Ah, 01Bh
:
:
```

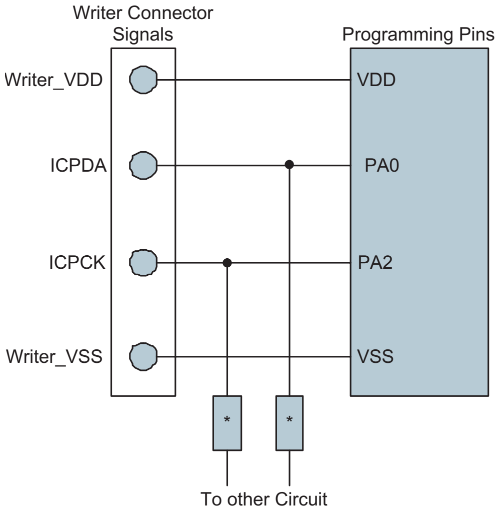
In Circuit Programming – ICP

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 devices.

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

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

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



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

On-Chip Debug Support – OCDS

There is an EV chip named HT45V39/HT45V391 which is used to emulate the real HT45F39/HT45F391 device. The EV chip also provides the "On-Chip Debug" function to debug the real MCU during development process. The real MCU and EV chip are almost functional compatible except the "On-Chip Debug" function and package types. Users can use the EV chip to emulate the real MCU device behaviors by connecting the OCDSDA and OCDSCK pins to the Holtek HT-IDE development tools. The OCDSDA pin is the OCDS Data/Address input/output pin while the OCDSCK pin is the OCDS clock input pin. When users use the EV chip for debugging, the corresponding pin functions shared with the OCDSDA and OCDSCK pins in the real MCU device will have no effect in the EV chip. However, the two OCDS pins which are pin-shared with the ICP programming pins are still used as the Flash Memory programming pins for ICP. For more detailed OCDS information, refer to the corresponding document named "Holtek e-Link for 8-bit MCU OCDS User’s Guide".

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

In Application Programming – IAP

The devices offer IAP function to update data or application program to the flash memory. Users can define any memory location for the IAP, but there are some features which the user must take note of in the IAP.

- Erase page: 256 words/page
- Writing: 4 words/time
- Reading: 1 word/time

IAP Registers

There are two address registers, four 16-bit data registers and two control registers. The control registers are located in Bank0. Read and Write operations to the Flash memory are carried out using 16-bit data operations using the address and data registers and the control registers. Several registers control the overall operation of the internal Flash Program Memory. The address registers are named FARL and FARH, the data registers are named FDnL and FDnH, and the two control register are named FC0 and FC1 respectively. As the FARL, FARH, FDnL and FDnH registers together with the FC0 and FC1 control registers are all located in Bank 0, they can be directly accessed in the same was as any other Special Function Register.

- **IAP Registers List**

Name	Bit							
	7	6	5	4	3	2	1	0
FARL	D7	D6	D5	D4	D3	D2	D1	D0
FARH	—	—	—	—	—	D10	D9	D8
FD0L	D7	D6	D5	D4	D3	D2	D1	D0
FD0H	D15	D14	D13	D12	D11	D10	D9	D8
FD1L	D7	D6	D5	D4	D3	D2	D1	D0
FD1H	D15	D14	D13	D12	D11	D10	D9	D8
FD2L	D7	D6	D5	D4	D3	D2	D1	D0
FD2H	D15	D14	D13	D12	D11	D10	D9	D8
FD3L	D7	D6	D5	D4	D3	D2	D1	D0
FD3H	D15	D14	D13	D12	D11	D10	D9	D8
FC0	CFWEN	FMOD2	FMOD1	FMOD0	FWPEN	FWT	FRDEN	FRD
FC1	D7	D6	D5	D4	D3	D2	D1	D0

- **FARL 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 Flash Program Memory address
Flash Program Memory address bit 7~bit 0

- **FARH Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	D10	D9	D8
R/W	—	—	—	—	—	R/W	R/W	R/W
POR	—	—	—	—	—	0	0	0

Bit 7~3 Reserved, cannot be used

Bit 2~0 Flash Program Memory address
Flash Program Memory address bit 10~bit 8

- **FD0L 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 The first Flash Memory data [7:0]

- **FD0H Register**

Bit	7	6	5	4	3	2	1	0
Name	D15	D14	D13	D12	D11	D10	D9	D8
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 The first Flash Memory data [15:8]

- **FD1L 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 The second Flash Memory data [7:0]

- **FD1H Register**

Bit	7	6	5	4	3	2	1	0
Name	D15	D14	D13	D12	D11	D10	D9	D8
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 The second Flash Memory data [15:8]

- **FD2L 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 The third Flash Memory data [7:0]

- **FD2H Register**

Bit	7	6	5	4	3	2	1	0
Name	D15	D14	D13	D12	D11	D10	D9	D8
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 The third Flash Memory data [15:8]

- **FD3L 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 The fourth Flash Memory data [7:0]

- **FD3H Register**

Bit	7	6	5	4	3	2	1	0
Name	D15	D14	D13	D12	D11	D10	D9	D8
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 The fourth Flash Memory data [15:8]

- **FC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	CFWEN	FMOD2	FMOD1	FMOD0	FWPEN	FWT	FRDEN	FRD
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	1	1	1	0	0	0	0

Bit 7 **CFWEN:** Flash ROM Write control bit
0: Disable
1: Enable

When this bit is cleared to 0 by application program, the Flash memory write function is disabled. Note that writing a "1" into this bit results in no action. This bit is used to indicate that the Flash memory write function status. When this bit is set to 1 by hardware, it means that the Flash memory write function is enabled successfully. Otherwise, the Flash memory write function is disabled as the bit content is zero.

Bit 6~4 **FMOD2~FMOD0:** Flash Program memory operating mode control bits
000: write memory mode
001: Page erase mode
010: Reserved
011 Read memory mode
100: Reserved
101: Reserved
110: FWEN mode - Flash memory write function enable mode
111: Reserved

Bit 3 **FWPEN:** Flash memory Write procedure enable control
0: Disable
1: Enable

When this bit is set to 1 and the FMOD field is set to "110", the IAP controller will execute the "Flash memory write function enable" procedure. Once the Flash memory write function is successfully enabled, it is not necessary to set the FWPEN bit any more.

Bit 2 **FWT:** Flash memory Write Control
0: Write cycle has finished
1: Activate a write cycle

This bit is set by software to activate a write Flash memory cycle and cleared by hardware when the Flash memory write process is completed.

Bit 1 **FRDEN:** Flash Memory Read Enable
0: Disable
1: Enable

This is bit must be set high before a Flash memory read operation is carried out. Clearing this bit to zero will inhibit Flash memory read operations.

Bit 0 **FRD:** Flash memory Read Control
0: Read cycle has finished
1: Activate a read cycle

This bit is set by software to activate a read Flash memory cycle and cleared by hardware when the Flash memory read process is completed.

Note: The FWT, FRDEN and FRD registers can not be set to "1" at the same time with a single instruction.

- **FC1 Register**

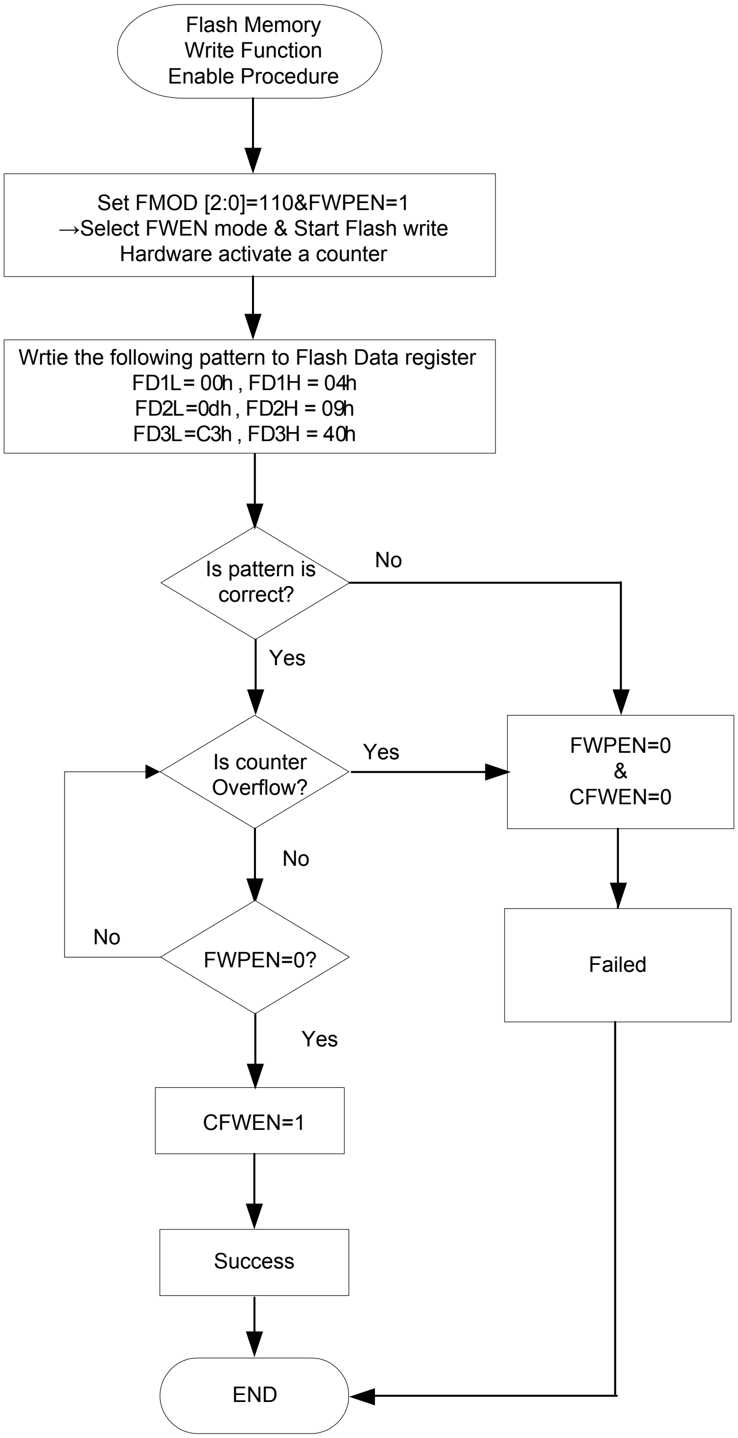
Bit	7	6	5	4	3	2	1	0
Name	D7	D6	D5	D4	D3	D2	D1	D0
R/W	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~0 Software Reset MCU control register
If this register is set to " 55H", it will generate a reset signal to reset MCU.

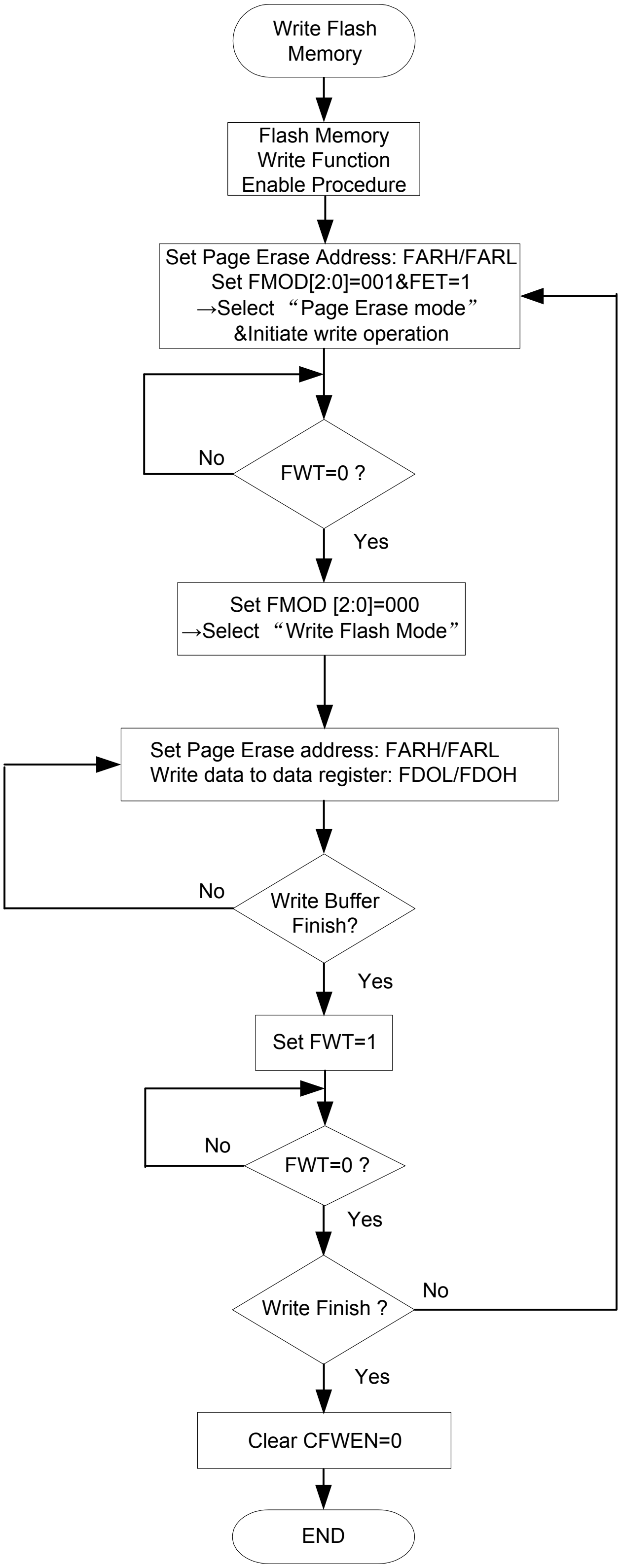
Flash Memory Write Function Enable Procedure

In order to allow users to change the Flash memory data through the IAP control registers, users must first enable the Flash memory write operation by the following procedures:

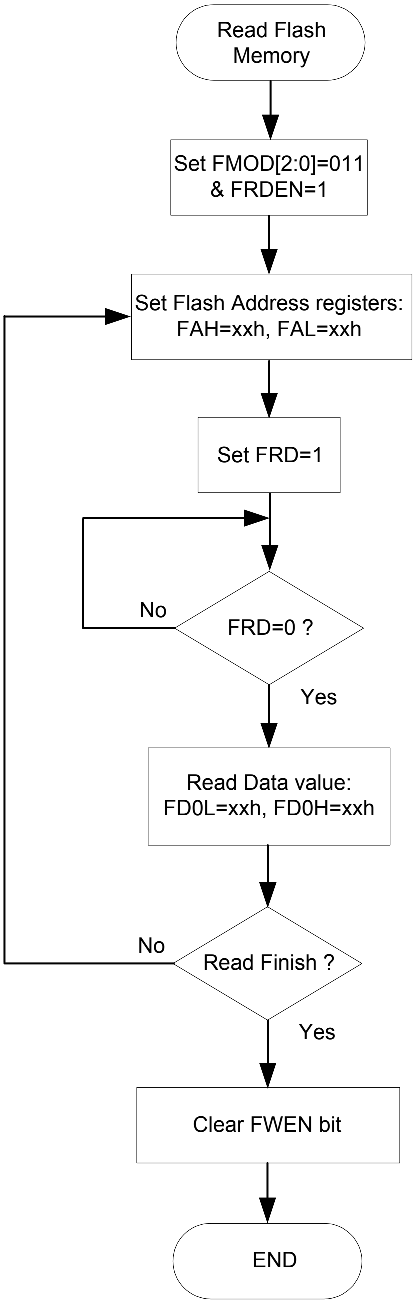
- Write "110" into the FMOD2~FMOD0 bits to select the FWEN mode.
- Set the FWPEN bit to "1". The step 1 and step 2 can be executed simultaneously.
- The pattern data with a sequence of 00H, 04H, 0DH, 09H, C3H and 40H must be written into the FD1L, FD1H, FD2L, FD2H, FD3L and FD3H registers respectively.
- A counter with a time-out period of 300μs will be activated to allow users writing the correct pattern data into the FD1L/FD1H~FD3L/FD3H register pairs. The counter clock is derived from LIRC oscillator.
- If the counter overflows or the pattern data is incorrect, the Flash memory write operation will not be enabled and users must again repeat the above procedure. Then the FWPEN bit will automatically be cleared to 0 by hardware.
- If the pattern data is correct before the counter overflows, the Flash memory write operation will be enabled and the FPWEN bit will automatically be cleared to 0 by hardware. The CFWEN bit will also be set to 1 by hardware to indicate that the Flash memory write operation is successfully enabled.
- Once the Flash memory write operation is enabled, the user can change the Flash ROM data through the Flash control register.
- To disable the Flash memory write operation, the user can clear the CFWEN bit to 0.



Flash Memory Write Function Enable Procedure


Write Flash Memory
Flash Memory Write Function Enable Procedure
Set Page Erase Address: FARH/FARL
Set FMOD[2:0]=001&FET=1
→Select “Page Erase mode”
&Initiate write operation
No
FWT=0 ?
Yes
Set FMOD [2:0]=000
→Select “Write Flash Mode”
Set Page Erase address: FARH/FARL
Write data to data register: FDOL/FDOH
No
Write Buffer Finish?
Yes
Set FWT=1
No
FWT=0 ?
Yes
Write Finish ?
No
Yes
Clear CFWEN=0
END

Write Flash Memory Procedure


Read Flash Memory
Set FMOD[2:0]=011 & FRDEN=1
Set Flash Address registers: FAH=xxh, FAL=xxh
Set FRD=1
No
FRD=0 ?
Yes
Read Data value: FD0L=xxh, FD0H=xxh
No
Read Finish ?
Yes
Clear FWEN bit
END

Read Flash Memory Procedure

Note: When the FWT or FRD bit is set to 1, the MCU is stopped

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 where special function registers are located. These registers have fixed locations and are necessary for correct operation of the devices. 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 reserved for general purpose use. All locations within this area are read and write accessible under program control.

00H	IAR0	30H	FD1L
01H	MP0	31H	FD1H
02H	IAR1	32H	FD2L
03H	MP1	33H	FD2H
04H	Unused	34H	FD3L
05H	ACC	35H	FD3H
06H	PCL	36H	FIFOWR_ADDR
07H	TBLP	37H	FIFORD_ADDR
08H	TBLH	38H	FIFOCTRL
09H	STATUS	39H	Unused
0AH	LVRC	3AH	DVCM
0BH	WDTC	3BH	SUMCMP
0CH	INTC0	3CH	SCFCTL
0DH	INTC1	3DH	AVPCTRL
0EH	INTC2	3EH	ENVCMP
0FH	INTEDGE	3FH	FIFOIN
10H	TM0AL	40H	FIFOOUT
11H	TM0AH	41H	CTRL
12H	TM0DL	42H	Unused
13H	TM0DH	43H	SENCOMM
14H	TM0C0	44H	Unused
15H	Unused	45H	Unused
16H	TM0C1	46H	Unused
17H	TM1AL	47H	Unused
18H	TM1AH	48H	Unused
19H	TM1DL	49H	Unused
1AH	TM1DH	4AH	Unused
1BH	TM1C0	4BH	Unused
1CH	TM1C1	4CH	Unused
1DH	TMPC	4DH	Unused
1EH	PA	4EH	Unused
1FH	PAC	4FH	Unused
20H	PAPU	50H	Unused
21H	PAWK	51H	Unused
22H	PB	52H	Unused
23H	PBC	53H	Unused
24H	PBPU	54H	Unused
25H	Unused	55H	Unused
26H	ADR	56H	Unused
27H	ADCR0	57H	Unused
28H	ADCR1	58H	Unused
29H	ACERL	59H	Unused
2AH	FC0	5AH	Unused
2BH	FC1	5BH	Unused
2CH	FARL	5CH	Unused
2DH	FARH	5DH	Unused
2EH	FD0L	5EH	Unused
2FH	FD0H	5FH	Unused

Special Purpose Data Memory Structure

Special Function Register Description

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

Indirect Addressing Registers – IAR0, IAR1

The Indirect Addressing Registers, IAR0 and IAR1, although having their locations in normal RAM 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 Pointer, MP0 or MP1. Acting as a pair, IAR0 with MP0 and IAR1 with MP1, can together access data from the Data Memory. 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, MP1 together with Indirect Addressing Register, IAR0,IRA1 are used to access data. Note that bit 7 of the Memory Pointers is not required to address the full memory space. When bit 7 of the Memory Pointers for these devices is read, a value of "1" will be returned.

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.

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

Status Register – STATUS

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

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

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

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

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

In addition, on entering an interrupt sequence or executing a subroutine call, the status register will not be pushed onto the stack automatically. If the contents of the status registers are important and if the interrupt routine can change the status register, precautions must be taken to correctly save it. Note that bits 0~3 of the STATUS register are both readable and writeable bits.

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 "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 not 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 on 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.

Oscillators and Operating Modes

Two internal oscillators are provided, one low speed and one high speed. Both are fully integrated and require no external components for their operation.

Internal High Speed 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 16MHz. Device trimming during the manufacturing process and the inclusion of internal frequency compensation circuits are used to ensure that the influence of the power supply voltage, temperature and process variations on the oscillation frequency are minimized. As a result, at a power supply of 5V and at a temperature from -20~60°C degrees, the fixed oscillation frequency 16MHz will have a tolerance within 2% as it requires no external pins for its operation.

Internal Low Speed Oscillator – LIRC

The LIRC is a fully self-contained free running on-chip RC oscillator used as the peripheral function clock source with a typical frequency of 32 kHz at 5V requiring no external components. When the devices enter the SLEEP0/SLEEP1 Mode, the system clock will stop running but the WDT oscillator continues to free-run and to keep the watchdog active.

System Operation Modes

There are two different modes for the microcontroller, each one with its own special characteristics and which can be chosen according to the specific performance and power requirements of the application. There is one mode allowing normal operation of the microcontroller, the NORMAL Mode. The other mode, the SLEEP Mode, is used when the microcontroller CPU is switched off to conserve power.

Operation Mode	Description		
	CPU	HIRC	LIRC
NORMAL Mode	On	On	On
SLEEP Mode	Off	Off	On

NORMAL Mode

As the name suggests this is one of the main operating modes where the microcontroller has all of its functions operational and where the system clock is provided by the HIRC oscillator.

SLEEP Mode

The SLEEP Mode is entered when an HALT instruction is executed. In the SLEEP mode the CPU will be stopped. However, the LIRC clock will continue to operate since the Watchdog Timer function is always enabled.

Entering the SLEEP Mode

There is only one way for the devices to enter the SLEEP Mode and that is to execute the "HALT" instruction in the application program and the WDT on. When this instruction is executed under the conditions described above, the following will occur:

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

Standby Current Considerations

As the main reason for entering the SLEEP Mode is to keep the current consumption of the devices to as low a value as possible, perhaps only in the order of several micro-amps , there are other considerations which must also be taken into account by the circuit designer if the power consumption is to be minimised. Special attention must be made to the I/O pins on the devices. 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.

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

Wake-up

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

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

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

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

Programming Considerations

If the devices are woken up from the SLEEP Mode to the NORMAL Mode, the high speed system oscillator needs an SST period. The device will execute first instruction after TO is "1".

Watchdog Timer

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

Watchdog Timer Clock Source

The Watchdog Timer clock source is provided by the internal clock, which is sourced from the LIRC oscillator. The Watchdog Timer source clock is then subdivided by a ratio of 2^8 to 2^{18} to give longer timeouts, the actual value being chosen using the WS2~WS0 bits in the WDTC register. The LIRC internal oscillator has an approximate period of 32kHz at a supply voltage of 5V.

However, it should be noted that this specified internal clock period can vary with V_{DD}, temperature and process variations.

Watchdog Timer Control Register

A single register, WDTC, controls the required timeout period as well as the enable/disable operation. The WRF software reset flag will be indicated in the CTRL register. This register together with several options control the overall operation of the Watchdog Timer.

WDTC Register

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

Bit 7~3 **WE4~WE0:** WDT function software control
10101 or 01010: WDT Enabled
Other values: Reset MCU

When these bits are changed to any other values due to environmental noise the microcontroller will be reset; this reset operation will be activated after 2~3 LIRC clock cycles and the WRF bit in the CTRL register will be set to 1 to indicate the reset source.

Bit 2 ~ 0 **WS2, WS1, WS0 :** WDT time-out period selection
000: $256/f_{SUB}$
001: $512/f_{SUB}$
010: $1024/f_{SUB}$
011: $2048/f_{SUB}$
100: $4096/f_{SUB}$
101: $8192/f_{SUB}$
110: $16384/f_{SUB}$
111: $32768/f_{SUB}$

These three bits determine the division ratio of the Watchdog Timer source clock, which in turn determines the timeout period.

CTRL Register

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

"x" unknown

Bit 7~3 "—": Unimplemented, read as "0"

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

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

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

Watchdog Timer Operation

The Watchdog Timer clock source is provided by the internal clock, which is sourced from the LIRC oscillator. It is always enable.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 devices. With regard to the Watchdog Timer enable/disable function, there are five bits, WE4~WE0, in the WDTC register to offer additional enable/disable and reset control of the Watchdog Timer.

WDT Function Control

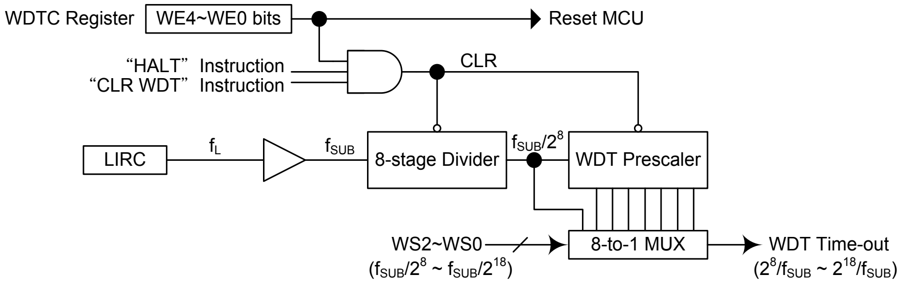
When the WE4~WE0 bit value is equal to 01010B or 10101B, the WDT function is enabled. However, if the WE4~WE0 bits are changed to any other values except 01010B and 10101B, which may be caused by external influences such as environmental noise, it will reset the microcontroller after 2~3 LIRC clock cycles.

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

Watchdog Timer Enable/Disable Control

Under normal program operation, a Watchdog Timer time-out will initialise a device reset and set the status bit TO. However, if the system is in the SLEEP 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 Watchdog Timer contents. The first is a WDT reset, which means a value other than 01010B or 10101B is written into the WE4~WE0 bit locations, the second is to use the Watchdog Timer software clear instructions and the third is via a HALT instruction. There is only one method of using software instruction to clear the Watchdog Timer and that is to use the single "CLR WDT" instruction to clear the WDT.

The maximum time out period is when the 2^{18} division ratio is selected. As an example, with a 32 kHz LIRC oscillator as its source clock, this will give a maximum watchdog period of around 8 seconds for the 2^{18} division ratio, and a minimum timeout of 7.8ms for the 2^{8} division ration.



Watchdog Timer

Reset and Initialisation

A reset function is a fundamental part of any microcontroller ensuring that the devices can be set to some predetermined condition irrespective of outside parameters. A hardware reset will of course be automatically implemented after the devices are powered-on, however there are a number of other hardware and software reset sources that can be implemented dynamically when the devices are running.

Reset Overview

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 instructions commence execution. 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.

The devices provide several reset sources to generate the internal reset signal, providing extended MCU protection. The different types of resets are listed in the accompanying table.

Reset Name	Abbreviation	Indication Bit	Register	Notes
Power-On Reset	POR	—	—	Auto generated at power on
Low-Voltage Reset	LVR	LVRF	CTRL	Low VDD voltage
LVRC Register Setting Sofe Reset	—	LRF	CTRL	Write to LVRC register
Watchdog Reset	WDT	TO	STATUS	WDT time-out
WDTC Register Setting Software Reset	—	WRF	CTRL	Write to WDTC register

Reset Source Summary

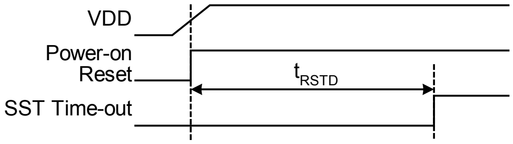
In addition to the power-on reset, situations may arise where it is necessary to forcefully apply a reset condition when the microcontroller is running. Another type of reset is when the Watchdog Timer overflows and resets the microcontroller. All types of reset operations result in different register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

Reset Functions

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

Power-on Reset

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



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

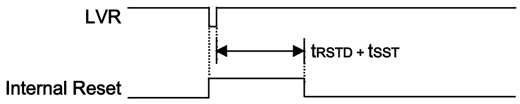
Power-On Reset Timing Chart

Low Voltage Reset – LVR

The microcontroller contains a low voltage reset circuit in order to monitor the supply voltage of the devices and provide an MCU reset should the value fall below a certain predefined level.

- **LVR Operation**

The LVR function is always enabled with a specific LVR voltage V_{LVR}. If the supply voltage of the devices drop to within a range of 0.9V~V_{LVR} such as might occur when changing the battery in battery powered applications, the LVR will automatically reset the devices internally and the LVRF bit in the CTRL register will also be set to 1. For a valid LVR signal, a low supply voltage, i.e., a voltage in the range between 0.9V~VLVR must exist for a time greater than that specified by t_{LVR} in the A.C. characteristics. If the low supply voltage state does not exceed this value, the LVR will ignore the low supply voltage and will not perform a reset function. The actual V_{LVR} value can be selected by the LVS bits in the LVRC register. If the LVS7~LVS0 bits are changed to some different values by environmental noise, the LVR will reset the devices after 2~3 LIRC clock cycles. When this happens, the LRF bit in the CTRL register will be set to 1. After power on the register will have the value of 01010101B. Note that the LVR function will be automatically disabled when the devices enter the power down mode.



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

Low Voltage Reset Timing Chart

- **LVRC Register**

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

Bit 7~0 **LVS7~LVS0:** LVR Voltage control
01010101: 3.15V
00110011: 3.15V
10011001: 3.15V
10101010: 3.15V
Any other value: Generates MCU reset – register is reset to POR value

When an actual low voltage condition occurs, as specified by one of the four defined LVR voltage values above, an MCU reset will be generated. The reset operation will be activated after 2~3 LIRC clock cycles. In this situation the register contents will remain the same after such a reset occurs.

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

- **CTRL Register**

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

Bit 7~3 "—": Unimplemented, read as 0

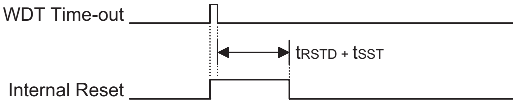
Bit 2 **LVRF:** LVR function reset flag
0: Not occur
1: Occurred
This bit is set to 1 when a specific Low Voltage Reset situation condition occurs. This bit can only be cleared to 0 by the application program.

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

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

Watchdog Time-out Reset during Normal Operation

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

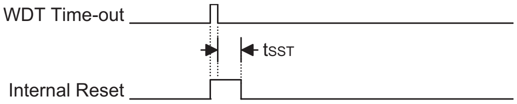


Note: t_{RSTD} is power-on delay, typical time=16.7ms

WDT Time-out Reset during Normal Operation Timing Chart

Watchdog Time-out Reset during SLEEP Mode

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



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

WDTC Register Software Reset

A WDTC software reset will be generated when a value other than "10101" or "01010", exist in the highest five bits of the WDTC register. The WRF bit in the CTRL register will be set high when this occurs, thus indicating the generation of a WDTC software reset.

- **WDTC Register**

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

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

Bit 2 ~ 0 **WS2, WS1, WS0:** WDT time-out period selection
Described elsewhere

Reset Initial Conditions

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

TO	PDF	RESET Conditions
0	0	Power-on reset
u	u	LVR reset during NORMAL
1	u	WDT time-out reset during NORMAL
1	1	WDT time-out reset during SLEEP Mode operation

Note: "u"stands for unchanged

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

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

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

The register states are summarized below:

Register	Power-on Reset	LVR Reset	WDT Time-out (Normal Operation)	WDT Time-Out (SLEEP0/SLEEP1)
MP0	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
MP1	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
PCL	0000 0000	0000 0000	0000 0000	uuuu uuuu
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
LVRC	0101 0101	uuuu uuuu	0101 0101	uuuu uuuu
WDTC	0101 0011	0101 0011	0101 0011	uuuu uuuu
INTC0	-00- -000	-00- -000	-00- -000	-uu- -uuu
INTC1	0000 0000	0000 0000	0000 0000	uuuu uuuu
INTC2	-00- -00-	-00- -00-	-00- -00-	-00- -00-
INTEDGE	---- --00	---- --00	---- --00	---- --uu
TM0AL	0000 0000	0000 0000	0000 0000	uuuu uuuu
TM0AH	---- --00	---- --00	---- --00	---- --uu
TM0DL	0000 0000	0000 0000	0000 0000	uuuu uuuu
TM0DH	---- --00	---- --00	---- --00	---- --uu
TM0C0	0000 0000	0000 0000	0000 0000	uuuu uuuu
TM0C1	0000 0000	0000 0000	0000 0000	uuuu uuuu
TM1AL	0000 0000	0000 0000	0000 0000	uuuu uuuu
TM1AH	---- --00	---- --00	---- --00	---- --uu
TM1DL	0000 0000	0000 0000	0000 0000	uuuu uuuu
TM1DH	---- --00	---- --00	---- --00	---- --uu
TM1C0	0000 0000	0000 0000	0000 0000	uuuu uuuu
TM1C1	0000 0000	0000 0000	0000 0000	uuuu uuuu
TMPC	---- --00	---- --00	---- --00	---- --uu
PA	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAC	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAPU	0000 0000	0000 0000	0000 0000	uuuu uuuu
PAWK	0000 0000	0000 0000	0000 0000	uuuu uuuu
PB	---- -111	---- -111	---- -111	---- -uuu
PBC	---- -111	---- -111	---- -111	---- -uuu
PBPU	---- -000	---- -000	---- -000	---- -uuu
ADR	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
ADCR0	011- 0000	011- 0000	011- 0000	uuu- uuuu
ADCR1	00-- -000	00-- -000	00-- -000	uu-- -uuu
ACERL	1111 1111	1111 1111	1111 1111	uuuu uuuu
FC0	0111 0000	0111 0000	0111 0000	uuuu uuuu
FC1	0000 0000	0000 0000	0000 0000	uuuu uuuu
FARL	0000 0000	0000 0000	0000 0000	uuuu uuuu
FARH	-000 0000	-000 0000	-000 0000	-uuu uuuu

Register	Power-on Reset	LVR Reset	WDT Time-out (Normal Operation)	WDT Time-Out (SLEEP0/SLEEP1)
FD0L	0000 0000	0000 0000	0000 0000	uuuu uuuu
FD0H	0000 0000	0000 0000	0000 0000	uuuu uuuu
FD1L	0000 0000	0000 0000	0000 0000	uuuu uuuu
FD1H	0000 0000	0000 0000	0000 0000	uuuu uuuu
FD2L	0000 0000	0000 0000	0000 0000	uuuu uuuu
FD2H	0000 0000	0000 0000	0000 0000	uuuu uuuu
FD3L	0000 0000	0000 0000	0000 0000	uuuu uuuu
FD3H	0000 0000	0000 0000	0000 0000	uuuu uuuu
FIFOWR_ ADDR	0000 0000	0000 0000	0000 0000	uuuu uuuu
FIFORD_ ADDR	---- ---0	---- ---0	---- ---0	---- ---u
FIFOCTRL	0000 0000	0000 0000	0000 0000	uuuu uuuu
DVCM	0000 0000	0000 0000	0000 0000	uuuu uuuu
SUMCMP	0000 0000	0000 0000	0000 0000	uuuu uuuu
SCFCTL	0-00 0000	0-00 0000	0-00 0000	u-uu uuuu
AVPCTRL	00-0 -000	00-0 -000	00-0 -000	uu-u -uuu
ENVCMP	00—000-	00—000-	00—000-	uu—uuu-
FIFOIN	0000 0000	0000 0000	0000 0000	uuuu uuuu
FIFOOUT	0000 0000	0000 0000	0000 0000	uuuu uuuu
CTRL	---- -000	---- -000	---- -000	---- -000
SENCOMM	---- 0000	---- 0000	---- 0000	---- uuuu

Note:"u" stands for unchanged

"x" stands for unknown

"-" stands for unimplemented

Input/Output Ports

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

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

I/O Register List

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

Pull-high Resistors

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

PAPU Register

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

Bit 7~0 PA bit 7 ~ bit 0 Pull-High Control
0: Disable
1: Enable

PBPU Register

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

Bit 7~3 "—": Unimplemented, read as "0"

Bit 2~0 PB bit 2 ~ bit 0 Pull-High Control
0: Disable
1: Enable

Port A Wake-up

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

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

I/O Port Control Registers

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

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

PAC Register

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

PBC Register

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

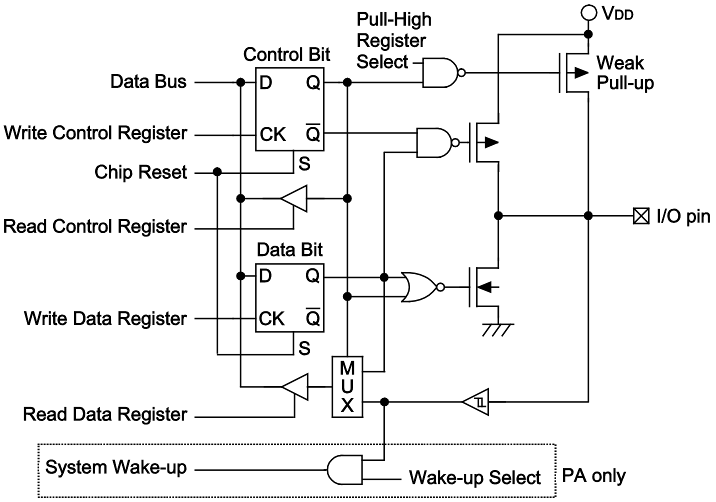
I/O Pin Structures

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

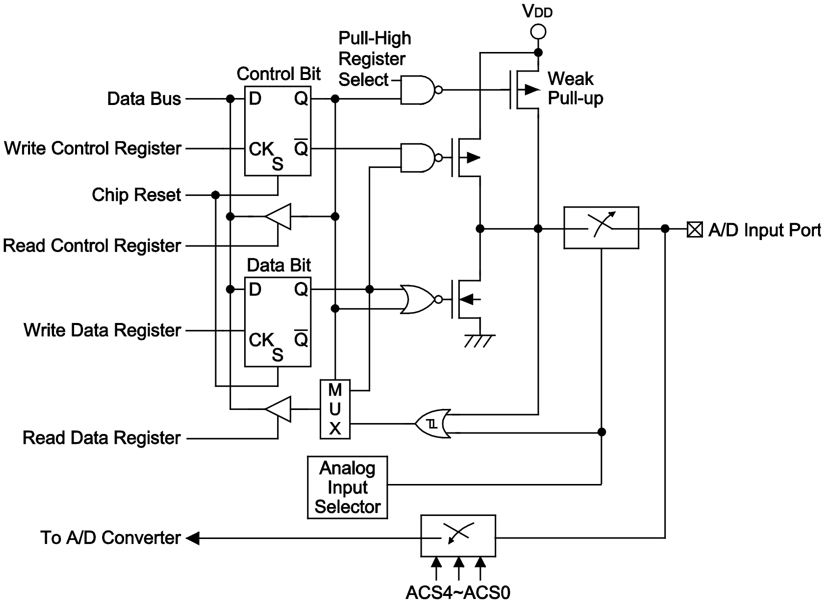
Programming Considerations

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

Port A has the additional capability of providing wake-up functions. When the devices are in the SLEEP Mode, various methods are available to wake the devices 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. In addition, the PORTA pins also provide Open Drain I/O structure options which can be controlled by the specific register.



Generic Input/Output Structure



A/D Input/Output Structure

Timer Modules – TM

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

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

Introduction

The devices contain two TMs having a reference name of TM0 and TM1, which can be categorised as a certain type, namely Compact Type TM. Although similar in nature, the different TM types vary in their feature complexity. The detailed operation regarding each of the TM types will be described in separate sections. The main features and differences between the two types of TMs are summarised in the accompanying table.

Function	CTM
Timer/Counter	V
Compare Match Output	V
PWM Channels	1
PWM Alignment	Edge
PWM Adjustment Period & Duty	Duty or Period

TM Function Summary

TM Operation

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

TM Clock Source

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

TM Interrupts

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

TM External Pins

Each of the TMs has one TM input pin, with the label TCKn. The TM input pin, is essentially a clock source for the TM and is selected using the TnCK2~TnCK0 bits in the TMnC0 register. This external TM input pin allows an external clock source to drive the internal TM. This external TM input pin is shared with other functions but will be connected to the internal TM if selected using the TnCK2~TnCK0 bits. The TM input pin can be chosen to have either a rising or falling active edge. The TMs each have one or more output pins with the label TPn. When the TM is in the Compare Match Output Mode, these pins can be controlled by the TM to switch to a high or low level or to toggle when a compare match situation occurs. The external TPn output pin is also the pin where the TM generates the PWM output waveform. As the TM output pins are pin-shared with other function, the TM output function must first be setup using registers. A single bit in one of the registers determines if its associated pin is to be used as an external TM output pin or if it is to have another function.

Timer Module Pin Control Register – TMPC

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	TP2C	TP1C	TP0C
R/W	—	—	—	—	—	R/W	R/W	R/W
POR	—	—	—	—	—	0	0	0

Bit 7~3 unimplemented, read as "0"

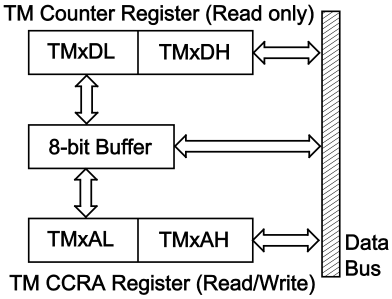
Bit 2 **TP2C:** TP0 and TP1 pin remapping control
0: TP0 on PA0; TP1 on PA2
1: TP0 on PA7; TP1 on PA3

Bit 1 **TP1C:** TP1 pin control
0: disable
1: enable
This bit is used to control the TP1 pin function. If this bit is set to 1 to enable the TP1 pin function but the TM1 is turned off, the TP1 pin function will be disabled, The TP1 pin function will also be disabled if the corresponding analog input function is enabled.

Bit 0 **TP0C:** TP0 pin control
0: disable
1: enable
This bit is used to control the TP0 pin function. If this bit is set to 1 to enable the TP0 pin function but the TM0 is turned off, the TP0 pin function will be disabled, The TP0 pin function will also be disabled if the corresponding analog input function is enabled.

Programming Considerations

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


TM Counter Register (Read only)
TMxDL
TMxDH
8-bit Buffer
TMxAL
TMxAH
Data Bus
TM CCRA Register (Read/Write)

The following steps show the read and write procedures:

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

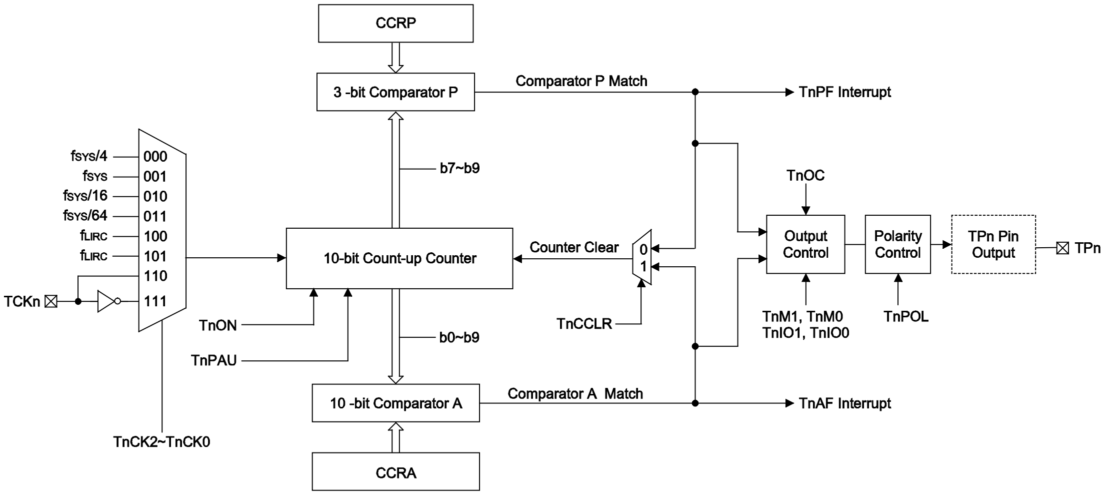
Compact Type TM – CTM

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

TM Type	TM Name	TM Input Pin	TM Output Pin
10-bit CTM	TM0	TCK0	TP0
10-bit CTM	TM1	TCK1	TP1

Compact TM Operation

At its core is a 10-bit count-up counter which is driven by a user selectable internal or external clock source. There are also two internal comparators with the names, Comparator A and Comparator P. These comparators will compare the value in the counter with CCRP and CCRA registers. The CCRP is three bits wide whose value is compared with the highest three bits in the counter while the CCRA is the ten bits and therefore compares with all counter bits. The only way of changing the value of the 10-bit counter using the application program, is to clear the counter by changing the TnON bit from low to high. The counter will also be cleared automatically by a counter overflow or a compare match with one of its associated comparators. When these conditions occur, a TM interrupt signal will also usually be generated. The Compact Type TM can operate in a number of different operational modes, can be driven by different clock sources including an input pin and can also control an output pin. All operating setup conditions are selected using relevant internal registers.



Compact Type TM Block Diagram

Compact Type TM Register Description

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

Name	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TMnC0	TnPAU	TnCK2	TnCK1	TnCK0	TnON	TnRP2	TnRP1	TnRP0
TMnC1	TnM1	TnM0	TnIO1	TnIO0	TnOC	TnPOL	TnDPX	TnCCLR
TMnDL	D7	D6	D5	D4	D3	D2	D1	D0
TMnDH	—	—	—	—	—	—	D9	D8
TMnAL	D7	D6	D5	D4	D3	D2	D1	D0
TMnAH	—	—	—	—	—	—	D9	D8

Compact TM Register List (n=0~1)

TMnDL Register

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

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

TMnDH Register

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

Bit 7~2 Reserved

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

TMnAL Register

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

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

TMnAH Register

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

Bit 7~2 Reserved

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

TMnC0 Register

Bit	7	6	5	4	3	2	1	0
Name	TnPAU	TnCK2	TnCK1	TnCK0	TnON	TnRP2	TnRP1	TnRP0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7 **TnPAU:** TMn Counter Pause Control
0: run
1: pause

The counter can be paused by setting this bit high. Clearing the bit to zero restores normal counter operation. When in a Pause condition the TM will remain powered up and continue to consume power. The counter will retain its residual value when this bit changes from low to high and resume counting from this value when the bit changes to a low value again.

Bit 6~4 **TnCK2~TnCK0:** Select TMn Counter clock
000: $f_{SYS}/4$
001: f_{SYS}
010: $f_{SYS}/16$
011: $f_{SYS}/64$
100: f_{LIRC}
101: f_{LIRC}
110: TCKn rising edge clock
111: TCKn falling edge clock

These three bits are used to select the clock source for the TM. The clock source f_{SYS} is the system clock and the details of which can be found in the oscillator section.

Bit 3 **TnON:** TMn Counter On/Off Control
0: Off
1: On

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

Bit 2~0 **TnRP2~TnRP0:** TMn CCRP 3-bit register, compared with the TMn Counter bit 9~bit 7

Comparator P Match Period
000: 1024 TMn clocks
001: 128 TMn clocks
010: 256 TMn clocks
011: 384 TMn clocks
100: 512 TMn clocks
101: 640 TMn clocks
110: 768 TMn clocks
111: 896 TMn clocks

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

TMnC1 Register

Bit	7	6	5	4	3	2	1	0
Name	TnM1	TnM0	TnIO1	TnIO0	TnOC	TnPOL	TnDPX	TnCCLR
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 **TnM1~TnM0:** Select TMn Operating Mode
00: Compare Match Output Mode
01: Undefined Mode
10: PWM Mode
11: Timer/Counter Mode

These bits setup the required operating mode for the TM. To ensure reliable operation the TM should be switched off before any changes are made to the TnM1 and TnM0 bits. In the Timer/Counter Mode, the TM output must be disabled.

Bit 5~4 **TnIO1~TnIO0:** Select TPn output function Compare Match Output Mode

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

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

Timer/counter Mode
Unused

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

In the Compare Match Output Mode, the TnIO1 and TnIO0 bits determine how the TM output pin changes state when a compare match occurs from the Comparator A. The TM output pin can be setup to switch high, switch low or to toggle its present state when a compare match occurs from the Comparator A. When the bits are both zero, then no change will take place on the output. The initial value of the TM output pin should be setup using the TnOC bit in the TMnC1 register. Note that the output level requested by the TnIO1 and TnIO0 bits must be different from the initial value setup using the TnOC bit otherwise no change will occur on the TM output pin when a compare match occurs. After the TM output pin changes state, it can be reset to its initial level by changing the level of the T0ON bit from low to high.

Bit 3 **TnOC:** Tn Output control bit Compare Match Output Mode

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

PWM Mode
0: Active low
1: Active high

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

Bit 2 **TnPOL:** TPn Output polarity Control
0: Non-invert
1: Invert

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

Bit 1 **TnDPX:** TMn PWM period/duty Control
0: CCRP - period; CCRA - duty
1: CCRP - duty; CCRA - period

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

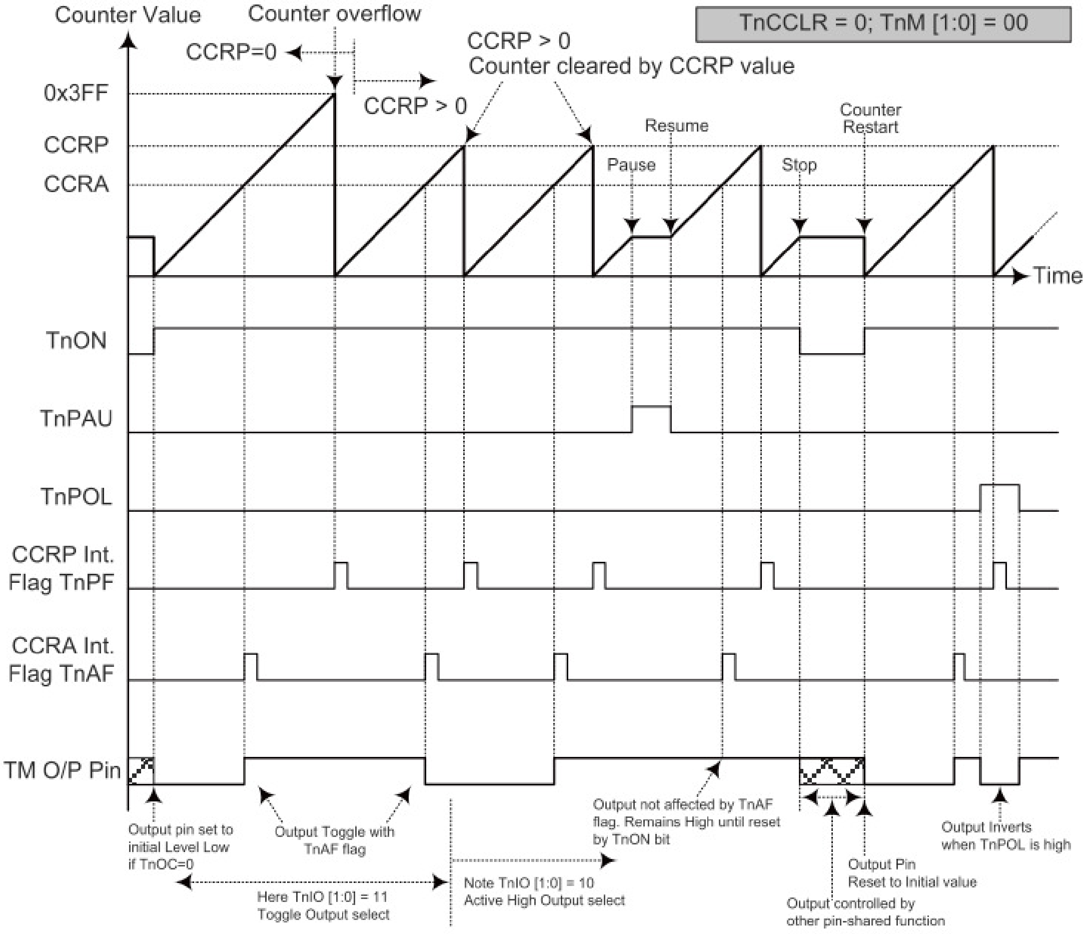
Bit 0 **TnCCLR:** Select TMn Counter clear condition
0: TMn Comparator P match
1: TMn Comparator A match

Compact Type TM Operating Modes

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

Compare Match Output Mode

To select this mode, bits TnM1 and TnM0 in the TMnC1 register, should be set to _00_ respectively. In this mode once the counter is enabled and running it can be cleared by three methods. These are a counter overflow, a compare match from Comparator A and a compare match from Comparator P. When the TnCCLR bit is low, there are two ways in which the counter can be cleared. One is when a compare match occurs from Comparator P, the other is when the CCRP bits are all zero which allows the counter to overflow. Here both TnAF and TnPF interrupt request flags for the Comparator A and Comparator P respectively, will both be generated. If the TnCCLR bit in the TMnC1 register is high then the counter will be cleared when a compare match occurs from Comparator A. However, here only the TnAF interrupt request flag will be generated even if the value of the CCRP bits is less than that of the CCRA registers. Therefore when TnCCLR is high no TnPF interrupt request flag will be generated. If the CCRA bits are all zero, the counter will overflow when its reaches its maximum 10-bit, 3FF Hex, value, however here the TnAF interrupt request flag will not be generated. As the name of the mode suggests, after a comparison is made, the TM output pin will change state. The TM output pin condition however only changes state when a TnAF interrupt request flag is generated after a compare match occurs from Comparator A. The TnPF interrupt request flag, generated from a compare match occurs from Comparator P, will have no effect on the TM output pin. The way in which the TM output pin changes state are determined by the condition of the TnIO1 and TnIO0 bits in the TMnC1 register. The TM output pin can be selected using the TnIO1 and TnIO0 bits to go high, to go low or to toggle from its present condition when a compare match occurs from Comparator A. The initial condition of the TM output pin, which is setup after the TnON bit changes from low to high, is setup using the TnOC bit. Note that if the TnIO1 and TnIO0 bits are zero then no pin change will take place.

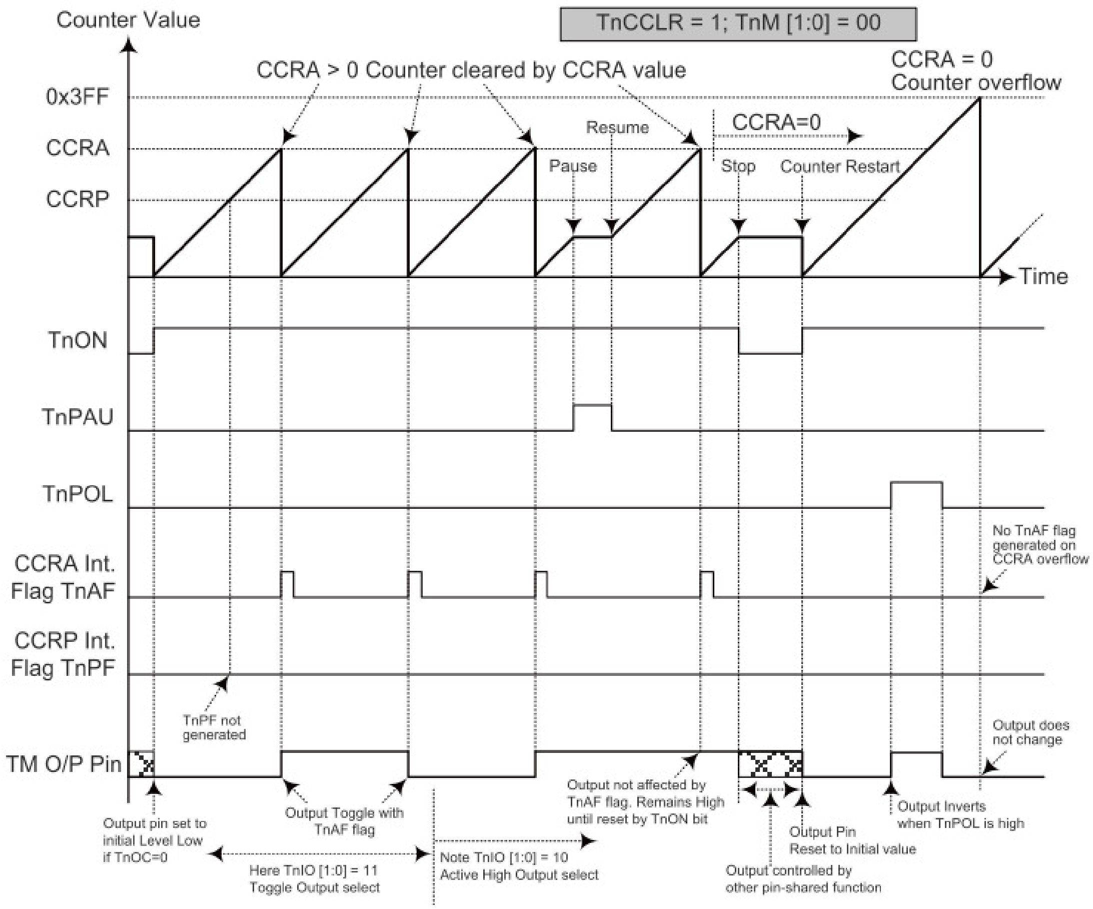

Counter Value
Counter overflow
TnCCLR = 0; TnM [1:0] = 00
CCRP=0
CCRP > 0
CCRP > 0
Counter cleared by CCRP value
0x3FF
CCRP
CCRA
Resume
Pause
Counter Restart
Stop
Time
TnON
TnPAU
TnPOL
CCRP Int. Flag TnPF
CCRA Int. Flag TnAF
TM O/P Pin
Output pin set to initial Level Low if TnOC=0
Output Toggle with TnAF flag
Here TnIO [1:0] = 11 Toggle Output select
Note TnIO [1:0] = 10 Active High Output select
Output not affected by TnAF flag. Remains High until reset by TnON bit
Output Pin Reset to Initial value
Output controlled by other pin-shared function
Output Inverts when TnPOL is high

Compare Match Output Mode – TnCCLR = 0

Note: 1. With TnCCLR=0, a Comparator P match will clear the counter

2. The TM output pin is controlled only by the TnAF flag

3. The output pin is reset to its initial state by a TnON bit rising edge


Counter Value
TnCCLR = 1; TnM [1:0] = 00
CCRA > 0 Counter cleared by CCRA value
CCRA = 0
Counter overflow
0x3FF
CCRA
CCRP
Resume
CCRA=0
Pause
Stop
Counter Restart
Time
TnON
TnPAU
TnPOL
CCRA Int.
Flag TnAF
No TnAF flag generated on CCRA overflow
CCRP Int.
Flag TnPF
TnPF not generated
Output does not change
TM O/P Pin
Output not affected by TnAF flag. Remains High until reset by TnON bit
Output Toggle with TnAF flag
Output Inverts when TnPOL is high
Output Pin Reset to Initial value
Output pin set to initial Level Low if TnOC=0
Here TnIO [1:0] = 11 Toggle Output select
Note TnIO [1:0] = 10 Active High Output select
Output controlled by other pin-shared function

Compare Match Output Mode – TnCCLR = 1

Note: 1. With TnCCLR=1, a Comparator A match will clear the counter

2. The TM output pin is controlled only by the TnAF flag

3. The output pin is reset to its initial state by a TnON bit rising edge

4. The TnPF flag is not generated when TnCCLR=1

Timer/Counter Mode

To select this mode, bits TnM1 and TnM0 in the TMnC1 register should be set to 11 respectively. The Timer/Counter Mode operates in an identical way to the Compare Match Output Mode generating the same interrupt flags. The exception is that in the Timer/Counter Mode the TM output pin is not used. Therefore the above description and Timing Diagrams for the Compare Match Output Mode can be used to understand its function.

PWM Output Mode

To select this mode, bits TnM1 and TnM0 in the TMnC1 register should be set to 10 respectively. The PWM function within the TM is useful for applications which require functions such as motor control, heating control, illumination control etc. By providing a signal of fixed frequency but of varying duty cycle on the TM output pin, a square wave AC waveform can be generated with varying equivalent DC RMS values. As both the period and duty cycle of the PWM waveform can be controlled, the choice of generated waveform is extremely flexible. In the PWM mode, the TnCCLR bit has no effect on the PWM operation. Both of the CCRA and CCRP registers are used to generate the PWM waveform, one register is used to clear the internal counter and thus control the PWM waveform frequency, while the other one is used to control the duty cycle. Which register is used to control either frequency or duty cycle is determined using the TnDPX bit in the TMnC1 register. The PWM waveform frequency and duty cycle can therefore be controlled by the values in the CCRA and CCRP registers. An interrupt flag, one for each of the CCRA and CCRP, will be generated when a compare match occurs from either Comparator A or Comparator P. The TnOC bit in the TMnC1 register is used to select the required polarity of the PWM waveform while the two TnIO1 and TnIO0 bits are used to enable the PWM output or to force the TM output pin to a fixed high or low level. The TnPOL bit is used to reverse the polarity of the PWM output waveform.

CTM, PWM Mode, Edge-aligned Mode, TnDPX=0

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

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

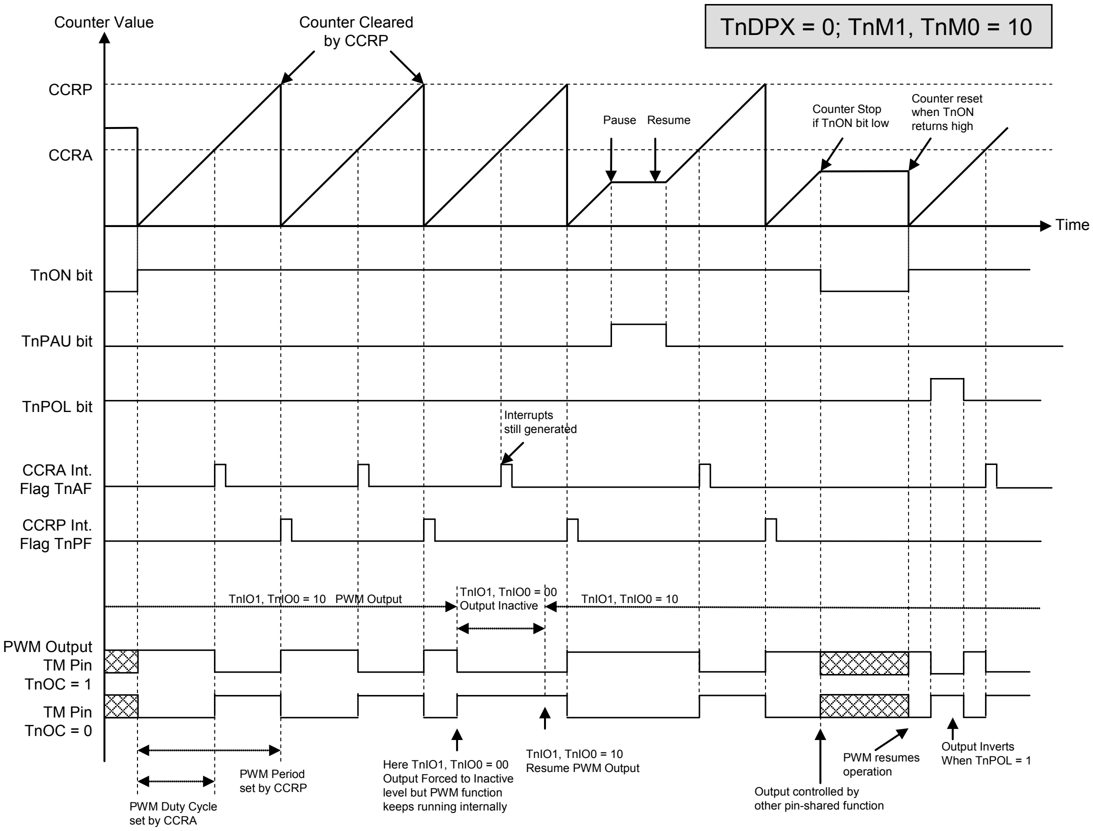
The CTM PWM output frequency = (f_{SYS}/4)/512 = f_{SYS}/2048 = 7.8125kHz, duty = 128/512 = 25%.

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

CTM, PWM Mode, Edge-aligned Mode, TnDPX=1

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

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


TnDPX = 0; TnM1, TnM0 = 10
Counter Value
Counter Cleared by CCRP
CCRP
CCRA
Pause
Resume
Counter Stop if TnON bit low
Counter reset when TnON returns high
Time
TnON bit
TnPAU bit
TnPOL bit
Interrupts still generated
CCRA Int. Flag TnAF
CCRP Int. Flag TnPF
TnIO1, TnIO0 = 10 PWM Output
TnIO1, TnIO0 = 00 Output Inactive
TnIO1, TnIO0 = 10
PWM Output TM Pin TnOC = 1
TM Pin TnOC = 0
PWM Period set by CCRP
PWM Duty Cycle set by CCRA
Here TnIO1, TnIO0 = 00 Output Forced to Inactive level but PWM function keeps running internally
TnIO1, TnIO0 = 10 Resume PWM Output
Output controlled by other pin-shared function
PWM resumes operation
Output Inverts When TnPOL = 1

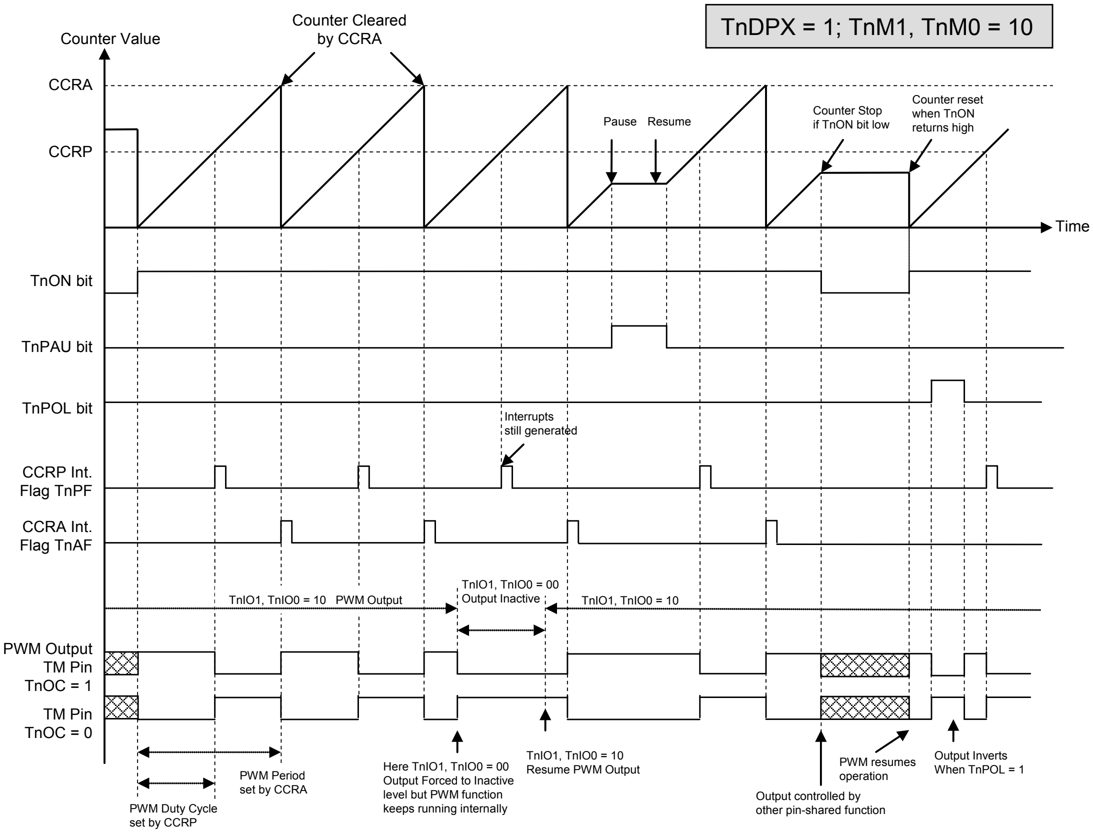
PWM Mode – TnDPX = 0

Note: 1. Here TnDPX=0 – Counter cleared by CCRP

2. A counter clear sets the PWM Period

3. The internal PWM function continues even when TnIO [1:0] = 00 or 01

4. The TnCCLR bit has no influence on PWM operation


TnDPX = 1; TnM1, TnM0 = 10
Counter Value
Counter Cleared by CCRA
CCRA
CCRP
Pause
Resume
Counter Stop if TnON bit low
Counter reset when TnON returns high
Time
TnON bit
TnPAU bit
TnPOL bit
Interrupts still generated
CCRP Int. Flag TnPF
CCRA Int. Flag TnAF
TnIO1, TnIO0 = 10 PWM Output
TnIO1, TnIO0 = 00 Output Inactive
TnIO1, TnIO0 = 10
PWM Output TM Pin TnOC = 1
TM Pin TnOC = 0
PWM Period set by CCRA
PWM Duty Cycle set by CCRP
Here TnIO1, TnIO0 = 00 Output Forced to Inactive level but PWM function keeps running internally
TnIO1, TnIO0 = 10 Resume PWM Output
Output controlled by other pin-shared function
PWM resumes operation
Output Inverts When TnPOL = 1

PWM Mode – TnDPX = 1

Note: 1. Here TnDPX = 1 – Counter cleared by CCRA

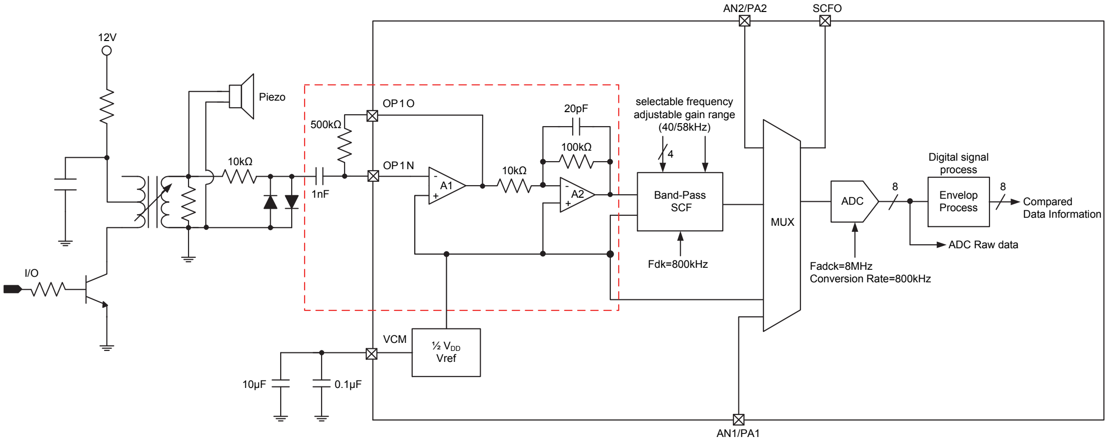
2. A counter clear sets the PWM Period

3. The internal PWM function continues even when TnIO [1:0] = 00 or 01

4. The TnCCLR bit has no influence on PWM operation

Operational Amplifiers

There are two fully integrated Operational Amplifiers in the device, OPA1 and OPA2. These OPAs can be used for signal amplification according to specific user requirements. The OPA1 gain is determined by external resistors and the OPA2 gain has a value of typical value of 7. The internal OPA reference voltage is 1/2 V_{DD} which can be detected by the VCM pin. The following block diagram illustrates the main functional blocks of the OPAs in this device.



Operational Amplifiers Block Diagram

Switch-Capacitor Filter – SCF

The input signal can be amplified by OPA1 and OPA2 before entering the SCF filter. The user can select the center frequency according to the choosing Ultrasonic sensor operating frequency. There are two options: 40kHz and 58kHz. This SCF filter also provides various gain control selections. The adjusted gain range is from 0 to 30dB.

The SCF output can be selected by the control bit, namely SCFOUT, in the SCFCTL register. When the SCFOUT bit is set high and the ACE5 bit is set to select the AN5 ADC input channel, the SCF signal will be output through the SCFO pin.

SCFCTL Register

Bit	7	6	5	4	3	2	1	0
Name	SCBWSL	SCFOUT	SCGAN5	SCGAN4	SCGAN3	SCGAN2	SCGAN1	SCGAN0
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 **SCBWSL :** Bandpass-Filter center frequency switch control bit
0: Fc=40kHz
1: Fc=58kHz

Bit 6 **SCFOUT:** Define pin function as AN5 or SCFO
0: AN5
1: SCFOUT
This bit is only available when the ACE5 bit in the ACERL register is set to 1 to enable the analog pin function.

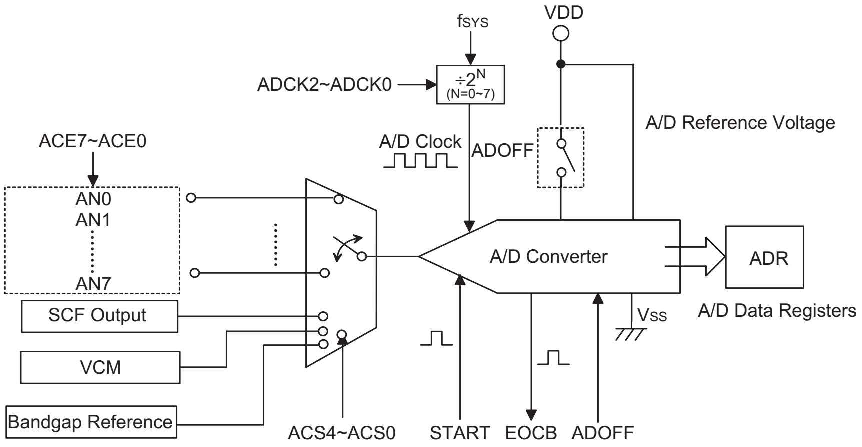
Bit 5~0 **SCGAN5~ SCGAN0 :** Bandpass-Filter gain adjustment bits
Av=Gain [5:0]*0.5 (dB), Gain[5:0] range is selectable from 0~30dB.

A/D Converter

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

A/D Overview

The devices contain a multi-channel analog to digital converter which can directly interface to external analog signals, such as that from sensors or other control signals and convert these signals directly into either a 8-bit digital value.



A/D Converter Structure

A/D Converter Register Description

Overall operation of the A/D converter is controlled using four registers. A read only register pair exists to store the ADC data 8-bit value. The remaining registers are control registers which setup the operating and control function of the A/D converter.

Register Name	Bit							
	7	6	5	4	3	2	1	0
ADR	ADR7	ADR6	ADR5	ADR4	ADR3	ADR2	ADR1	ADR0
ADCR0	Start	EOCB	ADOFF	---	ACS3	ACS2	ACS1	ACS0
ADCR1	ACS4	VBGEN	---	---	---	ADCK2	ADCK1	ADCK0
ACERL	ACE7	ACE6	ACE5	ACE4	ACE3	ACE2	ACE1	ACE0

A/D Converter Register List

A/D Converter Data Register – ADR

As the devices contain an internal 8-bit A/D converter, they require a data register to store the converted value, known as ADR. After the conversion process takes place, the register can be directly read by the microcontroller to obtain the digitised conversion value.

- **ADR Register**

BIT	7	6	5	4	3	2	1	0
NAME	ADR7	ADR6	ADR5	ADR4	ADR3	ADR2	ADR1	ADR0
R/W	R	R	R	R	R	R	R	R
POR	x	x	x	x	x	x	x	x

"x" unknown

Bit 7~0 **ADR7~ADR0:** A/D Conversion data

A/D Converter Conterl Registers – ADCR0, ADCR1, ACERL

To control the function and operation of the A/D converter, three control registers known as ADCR0, ADCR1 and ACERL 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 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 ADCR0 register and ACS4 bit is the ADCR1 register define the ADC input channel number. As the devices contain only one actual analog to digital converter hardware circuit, each of the individual 8 analog inputs must be routed to the converter. It is the function of the ACS4~ACS0 bits to determine which analog channel input pins or internal 1.04V is actually connected to the internal A/D converter.

The ACERL control register contains the ACER7~ACER0 bits which determine which pins on Port A are used as analog inputs for the A/D converter input and which pins are not to be used as the A/D converter input. Setting the corresponding bit high will select the A/D input function, clearing the bit to zero will select either the I/O or other pin-shared function. When the pin is selected to be an A/D input, its original function whether it is an I/O or other pin-shared function will be removed. In addition, any internal pull-high resistors connected to these pins will be automatically removed if the pin is selected to be an A/D input.

ADCR0 Register

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

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

Note that if the Auto-Envelope Processing Unit control bit, namely SAVP, is set by the application, the START control bit is invalid and the A/D conversion function is controlled by Auto-Envelope Processing Unit.

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 is high.

Note that if the Auto-Envelope Processing Unit control bit, namely SAVP, is set by the application, the EOCB bit is invalid and the A/D conversion function is controlled by the Auto-Envelope Processing Unit.

Bit 5 **ADOFF:** ADC Module power on/off control bit
0: Power on
1: Power off

This bit controls the power to the A/D internal function. This bit should be cleared to zero to enable the A/D converter. If the bit is set high then the A/D converter will be switched off reducing the devices power consumption. As the A/D converter will consume a limited amount of power, even when not executing a conversion, this may be an important consideration in power sensitive battery powered applications.

Note: It is recommended to set ADOFF=1 before entering SLEEP0/SLEEP1 Mode for saving power.

ADOFF=1 will power down the ADC module.

Bit 4 Unimplemented, read as "0"

Bit 3~0 **ACS3~ACS0:** Select A/D channel (ACS4=0)
0000 AN0
0001 AN1
0010 AN2
0011 AN3
0100 AN4
0101 AN5
0110 AN6
0111 AN7
1000 SCF Output (MUXIN)
1001 VCM

These are the A/D channel select control bits. As there is only one internal hardware A/D converter each of the eight A/D inputs must be routed to the internal converter using these bits.

ADCR1 Register

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

Bit7 **ACS4:** Selecte Internal 1.04V as ADC input Control
0: Disable
1: Enable

This bit enables 1.04V to be connected to the A/D converter. The VBGEN bit must first have been set to enable the bandgap circuit 1.04V voltage to be used by the A/D converter. When the ACS4 bit is set high, the bandgap 1.04V voltage will be routed to the A/D converter and the other A/D input channels disconnected.

Bit6 **VBGEN:** Internal 1.04V Control
0: Disable
1: Enable

This bit controls the internal Bandgap circuit on/off function to the A/D converter. When the bit is set high the bandgap voltage 1.04V can be used by the A/D converter. If 1.04V is not used by the A/D converter, then the bandgap reference circuit will be automatically switched off to conserve power. When 1.04V is switched on for use by the A/D converter, a time t_{BG} should be allowed for the bandgap circuit to stabilize before implementing an A/D conversion.

Bit 5~3 unimplemented, read as "0"

Bit 2~0 **ADCK2~ADCK0:** Select A/D converter clock source
000: $f_{SYS}/2$
001: $f_{SYS}/4$
010: $f_{SYS}/8$
011: $f_{SYS}/16$
100: $f_{SYS}/32$
101: $f_{SYS}/64$
110: $f_{SYS}/128$
111: Undefined

These three bits are used to select the clock source for the A/D converter.

If the SAVP bit is set high, the ADCK2~ADCK0 bits is invalid and the A/D clock source is fixed at 800kHz, which is controlled by Auto-Envelope Processing Unit.

ACERL register

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

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

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

Bit 5 **ACE5:** Define PA5 is analog function or not
0: Not analog function
1: Analog function, A/D input AN5 or SCF output

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

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

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

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

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

A/D Operation

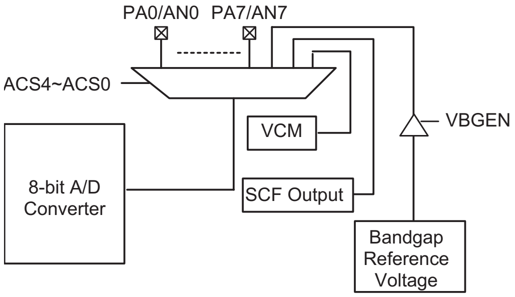
The START bit in the ADCR0 register is used to start and reset the A/D converter. When the microcontroller sets this bit from low to high and then low again, an analog to digital conversion cycle will be initiated. When the START bit is brought from low to high but not low again, the EOCB bit in the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

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

The clock source for the A/D converter, which originates from the system clock f_{SYS}, can be chosen to be either f_{SYS} or a subdivided version of f_{SYS}. The division ratio value is determined by the ADCK2~ADCK0 bits in the ADCR1 register. Controlling the power on/off function of the A/D converter circuitry is implemented using the ADOFF bit in the ADCR0 register. This bit must be zero to power on the A/D converter. Even if no pins are selected for use as A/D inputs by clearing the ACE7~ACE0 bits in the ACERL register, if the ADOFF bit is zero then some power will still be consumed. In power conscious applications it is therefore recommended that the ADOFF is set high to reduce power consumption when the A/D converter function is not being used. The reference voltage supply to the A/D Converter can be supplied from either the positive power supply pin, AVDD.

A/D Input Pins

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



A/D Input Structure

Summary of A/D Conversion Steps

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

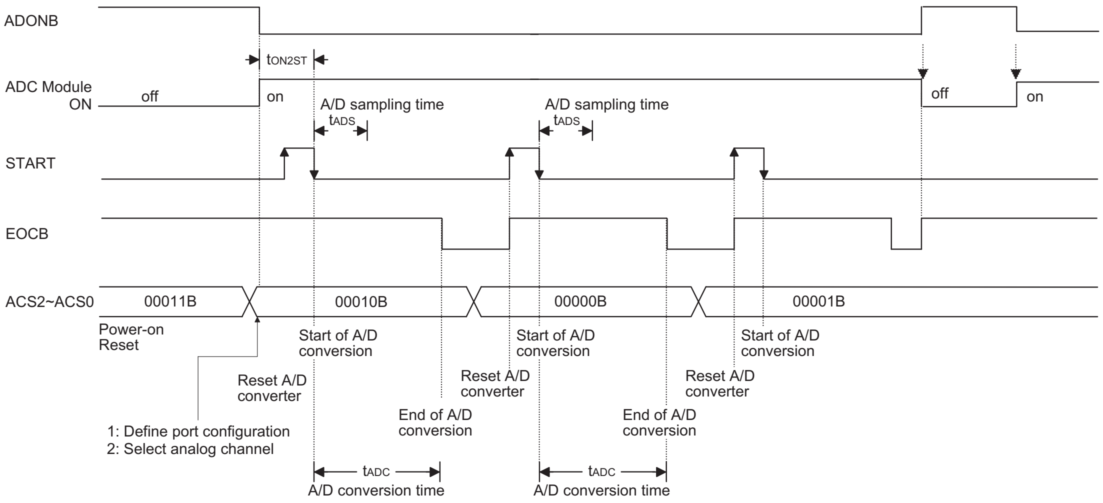
- Step 1
  Select the required A/D conversion clock by correctly programming bits ADCK2~ADCK0 in the ADCR1 register.
- Step 2
  Enable the A/D by clearing the ADOFF bit in the ADCR0 register to zero.
- Step 3
  Select which channel is to be connected to the internal A/D converter by correctly programming the ACS4 ~ACS0 bits which are also contained in the ADCR1 and ADCR0 registers.
- Step 4
  Select which pins are to be used as A/D inputs and configure them by correctly programming the ACE7~ACE0 bits in the ACERL register.
- Step 5
  If the interrupts are to be used, the interrupt control registers must be correctly configured to ensure the A/D converter interrupt function is active. The master interrupt control bit, EMI, and the A/D converter interrupt bit, ADE, must both be set high to do this.
- Step 6
  The analog to digital conversion process can now be initialised by setting the START bit in the ADCR0 register from low to high and then low again. Note that this bit should have been originally cleared to zero.
- Step 7
  To check when the analog to digital conversion process is complete, the EOCB bit in the ADCR0 register can be polled. The conversion process is complete when this bit goes low. When this occurs the A/D data register ADR can be read to obtain the conversion value. As an alternative method, if the interrupts are enabled and the stack is not full, the program can wait for an A/D interrupt to occur.

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

The accompanying diagram shows graphically the various stages involved in an analog to digital conversion process and its associated timing. After an A/D conversion process has been initiated by the application program, the microcontroller internal hardware will begin to carry out the conversion, during which time the program can continue with other functions. The time taken for the A/D conversion is $106t_{ADCK}$ where t_{ADCK} is equal to the A/D clock period.

Programming Considerations

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



A/D Conversion Timing

A/D Programming Examples

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

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

```
clr  ADE               ; disable ADC interrupt
mov  a,03H
mov  ADCR1,a           ; select fSYS/8 as A/D clock and switch off 1.04V
clr   ADOFF
mov  a,0Fh             ; setup ACERL to configure pins AN0~AN3
mov  ACERL,a
mov  a,00h
mov  ADCR0,a           ; enable and connect AN0 channel to A/D converter
:
start_conversion:
clr  START             ; high pulse on start bit to initiate conversion
set  START             ; reset A/D
clr  START             ; start A/D
polling_EOC:
sz   EOCB              ; poll the ADCR0 register EOCB bit to detect end
                       ; of A/D conversion
jmp  polling_EOC       ; continue polling
mov  a,ADR             ; read low byte conversion result value
mov  ADR_buffer,a      ; save result to user defined register
:
:
jmp  start_conversion  ; start next A/D conversion
```

Note: To power off the ADC, it is necessary to set ADOFF as "1".

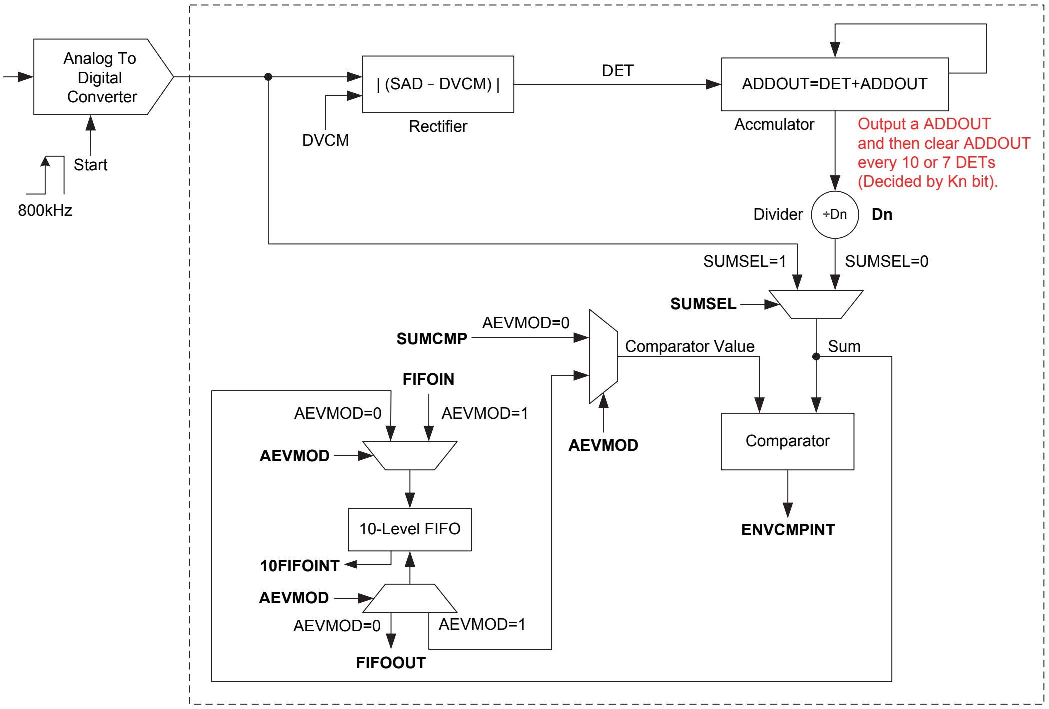
Example: using the interrupt method to detect the end of conversion

```
clr  ADE              ; disable ADC interrupt
mov  a,03H
mov  ADCR1,a          ; select fSYS/8 as A/D clock and switch off 1.04V
clr   ADOFF
mov  a,0Fh            ; setup ACERL to configure pins AN0~AN3
mov  ACERL,a
mov  a,00h
mov  ADCR0,a          ; enable and connect AN0 channel to A/D converter
Start_conversion:
clr  START            ; high pulse on start bit to initiate conversion
set  START            ; reset A/D
clr  START            ; start A/D
clr  ADF              ; clear ADC interrupt request flag
set   ADE             ; enable ADC interrupt
set   EMI             ; enable global interrupt
:
:
;ADC interrupt service routine
; ADC interrupt service routine
ADC_ISR:
mov  acc_stack,a      ; save ACC to user defined memory
mov  a,STATUS
mov  status_stack,a   ; save STATUS to user defined memory
:
:
mov  a,ADR            ; read low byte conversion result value
mov  adr_buffer,a     ; save result to user defined register
:
:
EXIT_INT_ISR:
mov  a,status_stack
mov  STATUS,a         ; restore STATUS from user defined memory
mov  a, acc_stack     ; restore ACC from user defined memory
reti
```

Note: To power off the ADC, it is necessary to set ADOFF as "1".

Auto-Envelope Process Unit

The Auto-Envelop Process (AEP) unit is used to process, transform and compare the input signal data with the selected compare value. The AEP unit is comprised of a Comparator, Rectifier, Accumulator, Divider, Comparator and FIFOs. The Comparator input signal can be selected by the SUMSEL bit in the AVPCTRL register. The user can select the comparing reference level to generate an interrupt. The input ADC converted data will be rectified, accumulated and then divided. The Comparator Value, from the SUMCP register or the internal 10-level FIFO data, can be selected by the AEVMOD bit in the AVPCTRL register . There are several Comparator interrupt conditions, managed by the CMOUT, ENVDGE1 and ENVDGE0 bits in the AVPOCTRL register. When the Comparator input data and the Comparator value matches an interrupt will be generated. The AEP unit is implemented by hardware to implement digital signal processing. It provides an accurate and a fast method to obtain the weight of the input signal for the designer to analyse and judge. The following block diagram illustrates the main functional blocks in the AEP unit in these devices.



Auto Envelop Process Unit

Auto-Envelope Processing Operation

The Auto-Envelope Processing Unit input signal is sourced from the ADC output.

The following points should be noted:

- If the SUM value is over 8 bits, then the SUM value will be set to 0xff automatically.
- The working FIFO has 10 levels, located in the 18-level FIFO. When 10 bytes in the FIFO have been read or written an interrupt will be generated.

There are two operating modes, selected by the AEVMOD bit in the AVPCTRL register.

AVEMOD=0 – Mode0

- The compared data is determined by the data in the SUMCMP register. There is only one compare data in mode 0.
- The data, namely SUM, will be input into the 10 Level FIFO directly. When the 10-Level FIFO interrupt is enabled and the 10th data is received, it will generate an interrupt. Note that the user should fetch the received data within a time of 40 instruction cycles to prevent data from going missing.

The designer can set up various values in the SUMCMP register to form various threshold voltage levels. In this way, the designer can implement an automatic threshold adaptation function.Usually, this mode is used to detect Ultrasonic sensor echos and to calculate the distance for long distance measurements.

AVEMOD=1 – Mode1

- This mode is used to compare the input data and compare it with the data in the 10-level FIFO at a 800KHz clock rate. The user can enter 10 sets of the comparison data into the 10-level FIFO. When the AEP is enabled by the AVP bit, the system hardware will automatically fetch the data from the FIFO to compare with the input signal at a 800KHz clock rate. When 10 sets of data have been fetched from the FIFO, the system will generate an interrupt and the user should then enter another 10 sets of data to the FIFO within a time of 40 instructions cycles.

At the beginning, the designer can use the AEP unit to record the Ultrasonic sensor vibration waveform data using the IAP function. The recorded waveform data can be used as reference data for comparison with the input signal to detect the Echo in the Sensor Vibration range to implement the short range function in this mode.

The compared results can be determined by the options. When the compared data is less than the SUM data, the compare output is "1", and when the comparrd data is greater than the SUM data, the compare output is "0". The compared output result can be used as the source to generate an interrupt, ENVCMPINT. The trigger edge can be selected by the ENVEDGE1 and ENVEDGE0 bits in the ENVCMP register.

Auto Envelop Process Unit Register Description

The Auto Envelop Process Unit, namely AEP, is implemented in hardware. The overall operation of the AEP unit is controlled using nine registers. The DVCM register, the ADC converted data of the VCM, is used to provide the input signal for the Rectifier to implement the full wave rectified function. The SUMCMP register is used to setup the Comparator Value for the Comparator input data. The ENVCMP register is used to setup the ADC sampling phase adjustment and the Comparator Interrupt Enable conditions. The AVPCTRL register is the control register for the AEP unit. These two registers, FIFOOUT and FIFOIN, are assigned for the FIFO input and output data register. The FIFOWR_ADDR and FIFORD_ADDR registers are used to check the current read and write FIFO address. The FIFOCTRL register is the FIFO control register which controls the operation and indicates the status of the FIFO.

Register Name	Bit							
	7	6	5	4	3	2	1	0
DVCM	D7	D6	D5	D4	D3	D2	D1	D0
SUMCMP	D7	D6	D5	D4	D3	D2	D1	D0
ENVCMP	EADADJ1	EADADJ0	—	—	CMPOUT	ENVEDGE1	ENVEDGE0	—
AVPCTRL	Dn	Kn	—	SUMSEL	—	AEVMOD	—	SAVP
FIFOOUT	D7	D6	D5	D4	D3	D2	D1	D0
FIFOIN	D7	D6	D5	D4	D3	D2	D1	D0
FIFOWR_ ADDR	—	—	—	FWR4	FWR3	FWR2	FWR1	FWR0
FIFORD_ ADDR	—	—	—	FRD4	FRD3	FRD2	FRD1	FRD0
FIFOCTRL	—	—	—	—	Empty	Full	FRESET	FSTART

DVCM 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

The VCM voltage is implemented by an internal hardware circuit and whose value is around 1/2 V_{DD}, It is used to provide the internal OPA reference voltage. The designer has to convert the VCM voltage into a digital value using the ADC converter and place it in the DVCM register. The ADC channel has to be setup to be the ACM input using the ACS3~ACS0 bits in the ADCR0 register. When the ACS3~ACS0 bits are set to "1001’, the ADC channel will select the VCM input. The DVCM register, which is the ADC converted VCM data is used to provide the input signal for the Rectifier to implement a full wave rectified function.

SUMCMP Register - SUM Compare value

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

The SUMCMP register is used to setup the Comparator Value for the Comparator input data. When the AEVMOD bit in the AVPCTRL register is "0", the value in the SUMCMP register will be selected to be the Comparator input, namely the Comparator Value.

ENVCMP register

The ENVCMP register is used to adjust the Auto-Envelop ADC sample phase, to setup the Comparator trigger edge and to indicate the Comparator output. The ADC sample delay time is selected by the EADADJ1 and EADADJ0 bits. There are four options for the ADC conversion delay time after the START signal. The CMPOUT bit is used to indicate the Comparator output. When the input data, namely SUM, is less than the Comparator Value, the CMPOUT bit will be "0", otherwise, this bit will be set to ‘1". The user can select the comparator interrupt trigger edge type, rising, falling or dual edge, using the ENVEDGE1 and ENVEDGE0 bits.

- **ENVCMP**

BIT	7	6	5	4	3	2	1	0
NAME	EADADJ1	EADADJ0	—	—	CMPOUT	ENVEDGE1	ENVEDGE0	—
R/W	R/W	R/W	—	—	R	R/W	R/W	—
POR	0	0	—	—	0	0	0	—

Bit7~~6 **EADADJ1~EADADJ0:** Auto-Envelope ADC Sample phase adjustment
00: Starts to implement Auto-Envelope ADC Conversion 2*t_{AD} seconds behind the START signal activation
01: Starts to implement Auto-Envelope ADC Conversion 1*t_{AD} seconds behind the START signal activation
10: Starts to implement Auto-Envelope ADC Conversion once the START signal activation
11: Starts to implement Auto-Envelope ADC Conversion 3*t_{AD} seconds behind the START signal activation

Bit5~4 unimplemented, read as 0

Bit 3 **CMPOUT:** Comparator Output
0:SUM < Comparator Value
1:SUM > Comparator Value

Bit2~1 **ENVEDGE1~ENVEDGE0:** Comparator output interrupt trigger edge selection
00: Positive Edge
01: Negative Edge
10: Dual Edge
11: can not be used

Bit0 unimplemented, read as 0

AVPCTRL Register

The AVPCTRL register is used to control the Auto Envelop Process unit. The Dn bit is used to set up the Divider value, according to whether the Ultrasonic sensor frequency is 40KHz or 58KHz.The reason for this is to allow the accumulated value from the Accumulator be transformed into 8-bit data. When a 40KHz Ultrasonic sensor is implemented, the accumulated value should be divided by 8. When a 58KHz sensor is implemented, the accumulated value should be divided by 4.

The Kn bit is used to select the number of ADC converted data output accumulated times. When the 40KHz frequency sensor is selected, as the ADC sampling rate is 800Khz, there are 20 sets of ADC converted data for a frequency cycle, then the half wave will have 10 sets of ADC converted data. On the other hand, when the 58KHz sensor is selected, the half wave of the 58KHz frequency will have 7 sets of ADC converted data. Therefore, when the 40KHz sensor is selected, the Kn bit must select "0"to accumulate the ADC output data 10 times to get the accumulated weight data as the Comparator input. When the 58KHz sensor is selected, the Kn bit must select "1" to get the proper accumulated weight data.

The SUMSEL bit is used to select the path for the Comparator input. When this bit is set high, the Comparator input will be the ADC converted data. Care should be taken if the ADC converted data output is at a rate of 800KHz, the designer should take care this short data processing time. When this bit is cleared "0", the Comparator input will select the Divider output.

The AEVMOD bit is used to select the Comparator reference data, from the data in the SUMCMP register or the 10-level data in FIFO. When the AEVMOD bit is cleared low, the Comparator reference data is selected as the data in the SUMCMP register. The designer can take advantage of this SUMCMP register to set the various reference data as the comparator input, according to different threshold requirements in the application. When this bit is set high, the 10-level data in the FIFO is selected as the comparator input. The designer should enter the preferred 10-level data to compare with the ADC converted data or the Divider output data, determined by the SUMSEL bit.

As the Comparison is implemented by hardware, the designer should enter the data first before starting this scheme and re-enter the data within a time of 40 instruction cycle as the 10-level data read out interrupt takes place.

The SAVP bit is used to enable the AEP function.

- **AVPCTRL Register (Auto-Envelope Control Register)**

BIT	7	6	5	4	3	2	1	0
NAME	Dn	Kn	—	SUMSEL	—	AEVMOD	—	SAVP
R/W	R/W	R/W	—	R/W	—	R/W	—	R/W
POR	0	0	—	0	—	0	—	0

Bit 7 **Dn:** Divider Value for 40KHz or 58KHz Ultrasonic Sensor
0:4 - 58KHz Ultrasonic sensor
1:8 - 40KHz Ultrasonic sensor

Note that if the Ultrasonic sensor frequency is 58KHz, then this bit must be selected as "0",the accumulated value from the ADDER will be divided by "4"before entering the Comparator. When the Ultrasonic sensor frequency is 40KHz, this be should be selected as "1".

Bit 6 **Kn:** Kn Value
0: Accumulate ADC converted data 10 times to produce an output ADDOUT.
1: Accumulate ADC converted data 7 times to produce an output ADDOUT.

Bit 5 Unimplemented

Bit 4 **SUMSEL:** Comparator input path, SUM, selection
0: SUMSEL=0, from Divider output
1: SUMSEL=1, from ADC converted data output

Bit 3 Unimplemented

Bit 2 **AEVMOD:** Auto-Envelope function Mode selection
0: Mode0
1: Mode1

In Mode 0, the Comapartor Value is from the SUMCMP register and the ADC converted data or the Divider output data will be stored in the FIFO. When the input data has more than 10 sets, it will generate an interrupt.

In Mode 1, the Comparator Value is from the FIFO and the designer should fill in the 10-level FIFO to compare it with the input singal converted data. The Comparator will fetch data in the FIFO at an 800KHz data rate. When a 10-level FIFO interrupt takes place, the dsigner should re-enter 10 sets of data into the FIFO.

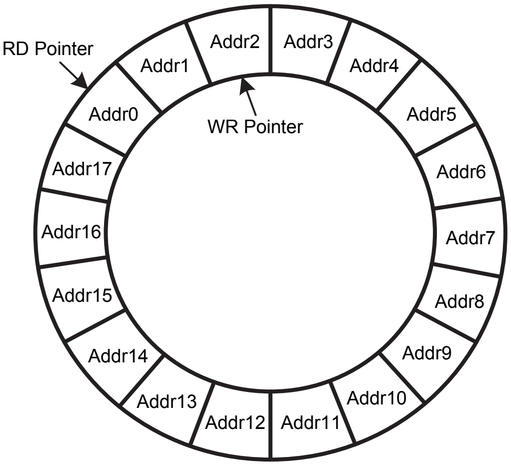
Bit 1 Unimplemented

Bit 0 **SAVP:** Activate Auto-envelope processor
0: Inactivate
1: Activate

When the SAVP bit is activated by the application program, the whole auto-envelope processing function is started. It will activate the ADC and SCF functions simultaneously.

FIFO Operation

These devices provide an 10-level FIFO. When the 10-Level FIFO interrupt is enabled and 10 sets of data are received or transmitted, an interrupt will be generated. There are five registers related to this FIFO operation. The FIFORD_ADDR register is used to indicate the current read address and the FIFOWR_ADDR is used to indicate the current write address in FIFO. The FIFOCTRL register is used to control the FIFO operation and to indicate the FIFO status. The designer can use this register to initialise the FIFO, to start the FIFO operation and to check the FIFIO status. The following operating diagram illustrates the FIFO structure.



FIFO Structure

Note: 1. To read data from the FIFO, the READ address will increase by one automatically until the READ address is equal to the WRITE address.

2. The FIFO is enabled by the FSTART control bit. No matter whether the hardware Auto-Envelop Processing Unit is enabled by the SAVP bit or not, when the AEP is disabled, the FIFO can still be accessed by control registers.

FIFOOUT register

The FIFOOUT register is a read only register. The designer can use this register to read the FIFO data.

- **FIFOOUT (10-Level FIFO Reading Register)**

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

FIFOIN register

The FIFOIN register is a write only register. The designer can use this register to write FIFO data.

- **FIFOIN (10-Level FIFO Writing Register)**

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

FIFOWR_ADDR

BIT	7	6	5	4	3	2	1	0
NAME	—	—	—	FWR4	FWR3	FWR2	FWR1	FWR0
R/W	—	—	—	R	R	R	R	R
POR	—	—	—	0	0	0	0	0

The current write address of the FIFO data. is FIFO addressIt is a read only register.

FIFORD_ADDR

BIT	7	6	5	4	3	2	1	0
NAME	—	—	—	FRD4	FRD3	FRD2	FRD1	FRD0
R/W	—	—	—	R	R	R	R	R
POR	—	—	—	0	0	0	0	0

The current read address of the FIFO data. datathis address It is a read only register.

FIFOCTRL

BIT	7	6	5	4	3	2	1	0
NAME	—	—	—	—	Empty	Full	FRESET	FSTART
R/W	—	—	—	—	R	R	R/W	R/W
POR	—	—	—	—	0	0	0	0

Bit 7~4 Unimplemented

Bit3 Empty
0: Not empty
1: Empty

If the FIFO read address is equal to the FIFO write address by one address, the flag is automatically set to "1" until the user writes a "0" to clear isthe bit using the application program. The bit is only valid if AEVMOD is set to "1".

Bit2 Full
0: Not full
1: Full

If the FIFO write address is behind the FIFO read address by one address, the flag is automatically set to "1" until the user writes a "0" to clear the bit using the application program. The bit is only valid if AEVMOD is set to "0".

Bit1 FRESET

The FRESET bit is used to align the Read address and Write address in the FIFO. When the alignment has been implemented, this bit will be cleared to "0" automatically. Note that this bit is only valid if the FSTART bit is set to "0".

Bit0 FSTART
1: Start to write data into FIFO or read data from the FIFO automatically through the Auto-Envelope processing mechanism using hardware.
0: Stop writing data into the FIFO or reading data from the FIFO through the Auto-envelope processing mechanism. Data can be written to or read from the FIFO manually using the application program.

Note that when the AEVMOD bit is set high, the designer can write the data into FIFO as the Comparator data. When the AEVMOD bit is cleared low, the designer can read the data from FIFO.

Car Parking Sensor Bus Control Unit

These devices provide a special Bus Control Unit (BCU) interface for data communication similar to Lin Bus. The BCU interface is used to communicate with the Host and the devices in a rapid and easy way. With this interface, the Host can transmit commands to all of the devices and manage the Ultrasonic Sensor signal at the same time. With flexible I/O control, the signal process data feedback can be reported to the host at the same time. There are several control bits in the SENCOMM register to control the data bus flow, namely IOC, IOCNT, INSEL1 and INSEL2 respectively. The IOC bit is used to determine the data flow transmissionpaths from PA2 to PA3 and the RX data from PA7 to PA6.

The BCU interface combined with a software application can implement an Ultrasonic Sensor ID Auto-Addressing function.

There are two modes for this interface operation: 5V and 12V, described by the following sections. The major difference is that the application has to handle the 12V higher voltage. The 5V system only needs PA2 and PA7 to implement the TX/RX operations while the 12V system needs PA2, PA7, PA3 and PA6 and transistors for the TX/RX data.

For this functions there is a single control register

SENCOMM Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	IOC	IOCNT	INSEL2	INSEL1
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0

Bit 7~4 : Unimplemented, read as "0"

Bit 3: **IOC:** Internal path PA2 to PA3 and path PA7 to PA6 control bit
0: PA7 to PA6 data path is available
1: PA2 to PA3 data path is available
This bit control is suggested for use in the 12V interface system.

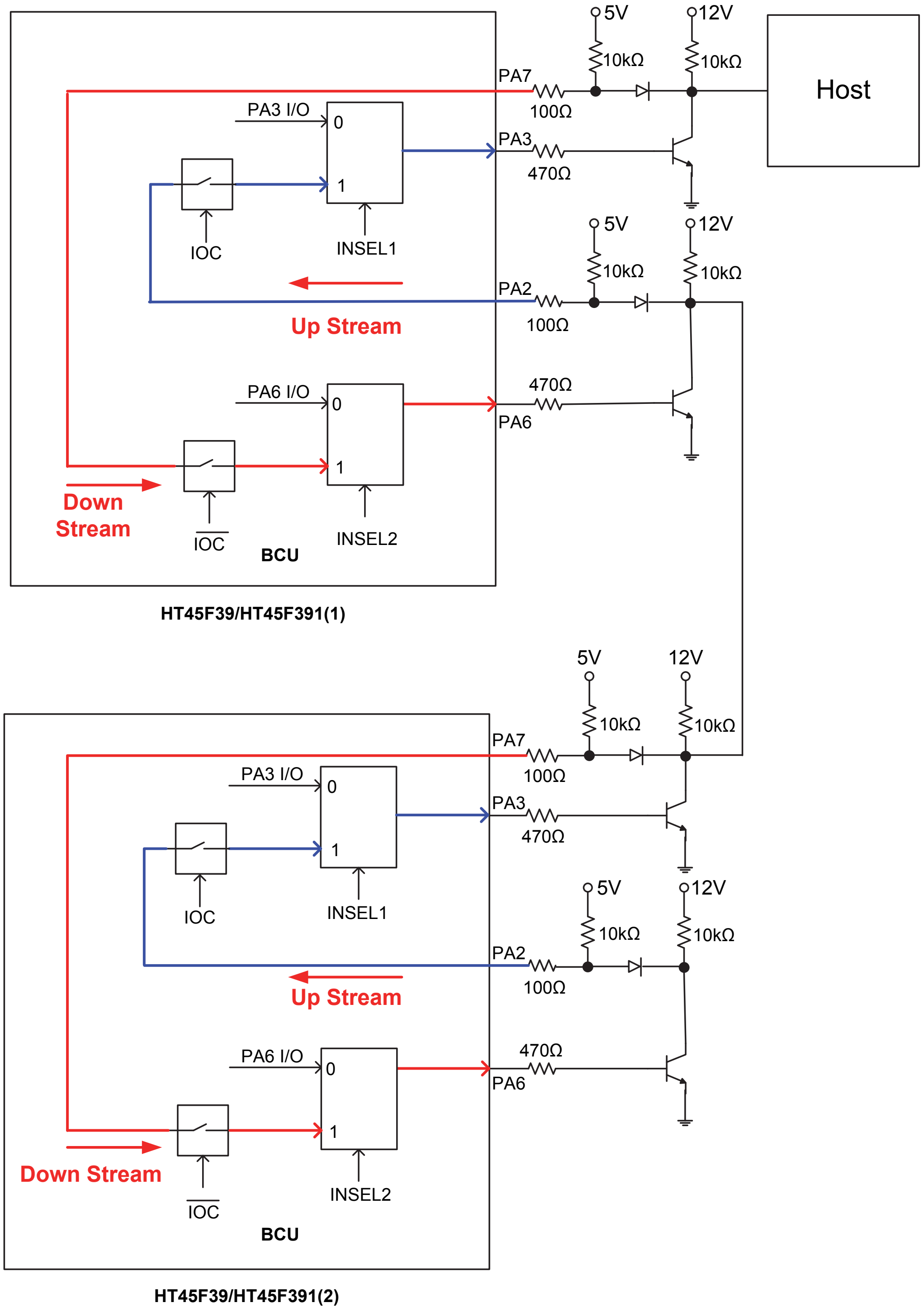
Bit 2 **IOCNT:** Internal path between PA7 and PA2 control bit
0: Disable internal path
1. Enable internal path
Note: This bit is strongly recommended for use with the 5V interface.

Bit 1 **INSEL2:** PA6 as I/O function or internal path PA6 to PA7 control bit
0: PA6 I/O function
1: Internal data path PA7 to PA6 is on when the IOC bit is set to "0".
This bit is used along with the IOC bit to determine the data path PA7 to PA6.

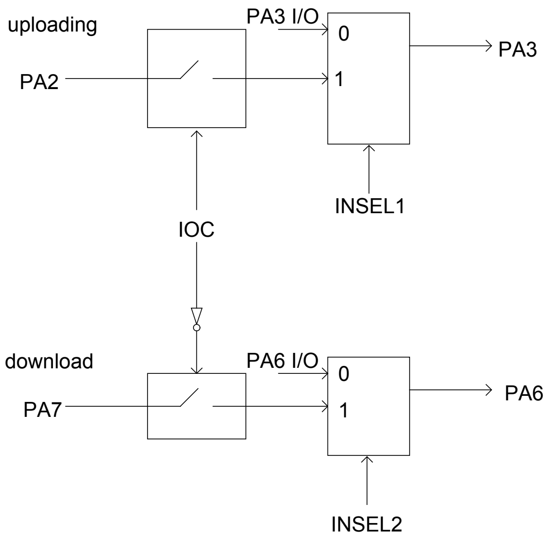
Bit 0 **INSEL1:** PA3 as I/O function or internal path PA2 to PA3 control bit
0: PA3 I/O function
1: Internal data path PA2 to PA3 is on when the IOC bit is set to "1".
This bit is used along with the IOC bit to determine the data path PA2 to PA3.

Bus Control Unit for 12V Interface Application

The following application Block diagram illustrates the 12V interface application. Note that the IOCNT bit is only used for the 5V interface application. In the 12V interface, this bit should be cleared to "0".



12V Interface Application Block Diagram

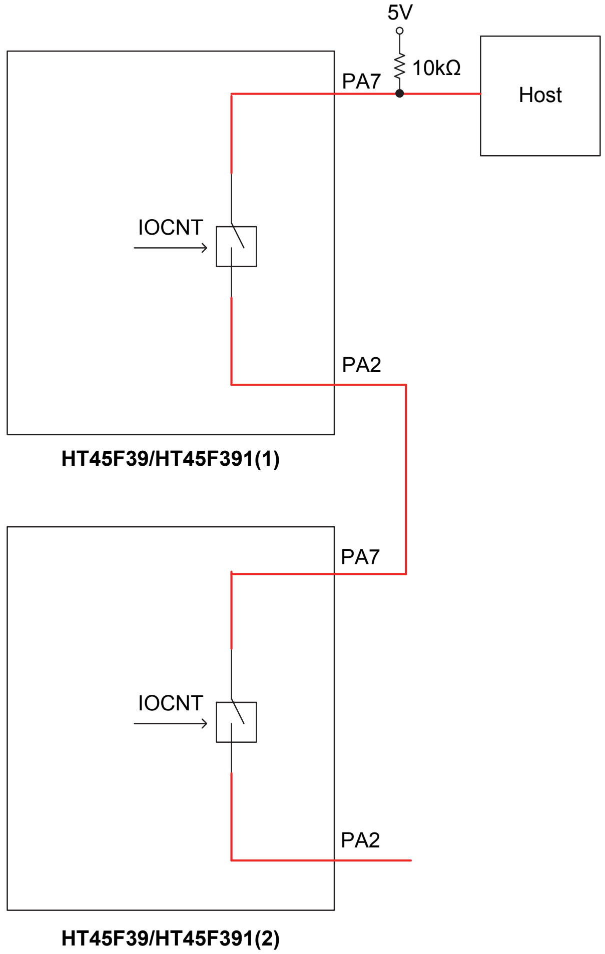


The following is the Auto ID setup procedure for the Ultrasonic Sensor module:

- Before the Ultrasonic Sensor module plug on the Bus, there is no ID in this module.
  The setup for the Bus Control Unit, is as follows, IOC=0,INSEL1=0 and INSEL2=1.
  When the IOC bit is cleared to "0", it will disconnect the PA2 to PA3 path and connect PA7 to the PA6 MUX input. When INSEL1=0, PA3 is selected to have an I/O function. When INSEL2=1, PA6 is selected as a PA7 to PA6 path function.
  Now the HT45F39/HT45F391 begins to send an ID request to the Host through PA3. All the modules which have no ID in them should implement the same setup and send an ID request to the host. PA7 has to be setup as an input pin and PA3 should be setup as an output pin.
- When the HT45F39/HT45F391(1) is connectecd to the BUS, the HOST will receive an ID request from the HT45F39/HT45F391(1) and send a corresponding ID to it. The ID will be received by the PA7 pin.
- When the HT45F39/HT45F391(1) has received the ID, it has to set IOC=1 and INSEL1=1 to set the path, PA2to PA3 for the HT45F39/HT45F391(2).
- When the HT45F39/HT45F391(2) is connected to the BUS, following the step 1 and step 2, an ID will be sent to it by the PA7 input pin from the HOST.
- When the HT45F39/HT45F391(2) as received the ID, it has to set IOC=1 and INSEL1=1 to set the path for another module.
- When all the modules have received their corresponding IDs, then when the HOST sends a command, all the modules, namely HT45F39/HT45F391(n), will receive the command at the same time and check if the command is matched.
- If the ID is matched, the HT45F39/HT45F391 has to set IOC=0 and INSEL1=0, and then send a Response command to the HOST through the PA3 output pin. If the ID is mismatched, the HT45F39/HT45F391 has to set IOC=1 and INSEL1=1 to set the path, PA2 to PA3.
- When the Response command is sent completely, all of the HT45F39/HT45F391 devices in the modules should set IOC=0 and INSEL2=1 to wait for the the next HOST command.

5V Interface Application Bus Control Unit

The following application Block diagram illustrates the 5V interface application. Note that the IOC, INSEL1 and INSEL2 bits are only available for 12V interface application. In the 5V interface, the setup for these bits should be IOC=0, INSEL1=0 and INSEL2=0.



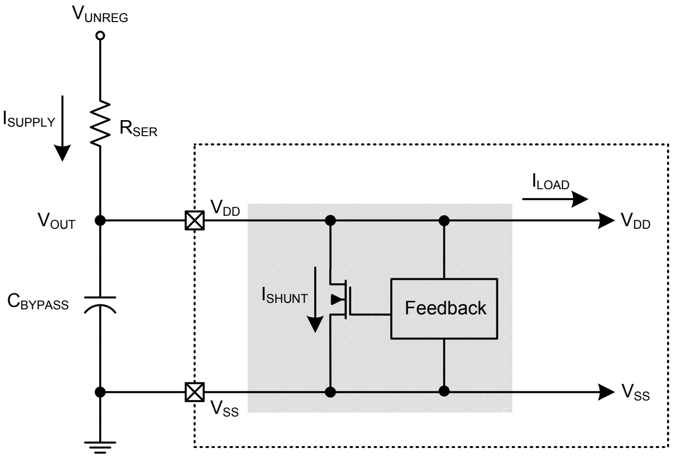
5V Interface Application Block Diagram

The following is the Auto ID Setting procedure for the Ultrasonic Sensor module:

- Before the Ultrasonic Sensor module is connected to the Bus, as there is no ID in this module the setup of the Bus Control Unit has to be IOCNT=0.
  When the IOCNT bit is cleared to "0", it will disconnect the PA2 to PA7 path. Then, the HT45F39/HT45F391 will start to send an ID request to the Host through PA7. All the modules which have no ID in them implement the same setting and send an ID request to the host. The PA7 has to be setup as an output pin to transmit an ID request and setup as input pin when the ID request is transmitted completely.
- When the HT45F39/HT45F391(1) is connected to the BUS, the HOST will receive an ID request from the HT45F39/HT45F391(1) and send a corresponding ID to it. The ID will be received by the PA7 pin.
- When the HT45F39/HT45F391(1) receives the ID, it has to set IOCNT=1 to set the path, PA2to PA7, for the HT45F39/HT45F391(2).
- When the HT45F39/HT45F391(2) is connected to the BUS, following step 1 and step 2, an ID will be sent to it by the PA7 input pin from the HOST.
- When the HT45F39/HT45F391(2) has received the ID, it has to set IOCNT=1, to set the path for another module.
- When all the modules have recevied the corresponding IDs, then when the HOST sends a command, all the modules, namely HT45F39/HT45F391(n), will receive it at the same time and check if the command is matched.
- If the ID is matched, the HT45F39/HT45F391 has to set IOCNT=1 and then send a Response command to the HOST through the PA7 output pin.
- When the Response command is sent completely, all of the HT45F39/HT45F391 devices in the modules should set IOCNT=1 to wait the next command from the HOST.

Shunt Regulator – HT45F39 only

The HT45F39 device includes a fully integrated shunt regulator. The shunt regulator is a common and simple type of regulator whose output is used as the device power supply. The shunt regulator is constructed of a power transistor and a voltage feedback circuit. The power transistor, whose output current is controlled by a voltage feedback circuit, is used to shunt the extra current to maintain a constant regulated voltage on VDD. This power transistor provides a current path from the device supply voltage to ground. An external resistor, RSER, should be properly selected to serially connect VDD to the external power supply. The shunt regulator may not have the expected performance if the RSER resistance is too small or too large. A capacitor is externally connected between VDD and VSS pins to stabilise the regulated voltage. Note that excessive power dissipation in form of heat due to the shunt current through the power transistor should be taken into consideration in the user applications.



Shunt Regulator Block Diagram

Interrupts

Interrupts are an important part of any microcontroller system. When an external event or an internal function such as a Timer Module or an A/D converter requires microcontroller attention, their corresponding interrupt will enforce a temporary suspension of the main program allowing the microcontroller to direct attention to their respective needs. The devices contain one external interrupt and several internal interrupts functions. The external interrupt is generated by the action of the external INT pin, while the internal interrupts are generated by various internal functions such as the TMs, Auto-Envelope Comparator, and the A/D converter.

Interrupt Registers

Overall interrupt control, which basically means the setting of request flags when certain microcontroller conditions occur and the setting of interrupt enable bits by the application program, is controlled by a series of registers, located in the Special Purpose Data Memory, as shown in the accompanying table. The INTC0~INTC2 registers setup the primary interrupts while the INTEG register setup the external interrupt trigger edge type.

INTEDGE Register

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

Bit 7~2 Unimplemented, read as 0

Bit 1~0 **INTS1~INTS0:** Interrupt edge control for INT pin
00: disable
01: rising edge
10: falling edge
11: rising and falling edges

INTC0 Register

Bit	7	6	5	4	3	2	1	0
Name	—	FIFO10F	ENVCMPF	—	—	FIFO10E	ENVCMPE	EMI
R/W	—	R/W	R/W	—	—	R/W	R/W	R/W
POR	—	0	0	—	—	0	0	0

Bit 7 Unimplemented, read as 0

Bit 6 **FIFO10F:** 10-level FIFO Interrupt request flag
0: inactive
1: active

Bit 5 **ENVCMPF:** Auto-Envelope Comparator Request Flag
0: Auto-Envelope Comparator Request Flag does not occur
1: Auto-Envelope Comparator Request Flag occurs

Bit 4~3 Unimplemented, read as 0

Bit 2 **FIFO10E:** 10-level FIFO Interrupt Enable
0: Disable
1: Interrupt event occurs if 10-level FIFO flag is activated

Bit 1 **ENVCMPE:** Auto-Envelope Comparator Interrupt Enable
0: Disable
1: Enable

Bit 0 **EMI:** Master interrupt global enable
0: disable
1: enable

INTC1 Register

Bit	7	6	5	4	3	2	1	0
Name	TP1F	TA1F	TP0F	TA0F	TP1E	TA1E	TP0E	TA0E
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 **TP1F:** CTM1 Comparator P Match interrupt request flag
0: inactive
1: active

Bit 6 **TA1F:** CTM1 Comparator A Match interrupt request flag
0: inactive
1: active

Bit 5 **TP0F:** CTM0 Comparator P Match interrupt request flag
0: inactive
1: active

Bit 4 **TA0F:** CTM0 Comparator A Match interrupt request flag
0: inactive
1: active

Bit 3 **TP1E:** CTM1 Comparator P Match interrupt enable
0: disable
1: enable

Bit 2 **TA1E:** CTM1 Comparator A Match interrupt enable
0: disable
1: enable

Bit 1 **TP0E:** CTM0 Comparator P Match interrupt enable
0: disable
1: enable

Bit 0 **TA0E:** CTM0 Comparator A Match interrupt enable
0: disable
1: enable

INTC2 Register

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

Bit 7 Unimplemented, read as 0

Bit 6 **ADF:** ADC interrupt request flag
0: inactive
1: active

Bit 5 **INTF:** External interrupt request flag
0: inactive
1: active

Bit 4~3 Unimplemented, read as 0

Bit 2 **ADE:** ADC interrupt enable
0: disable
1: enable

Bit 1 **INTE:** External interrupt enable
0: disable
1: enable

Bit 0 Unimplemented, read as 0

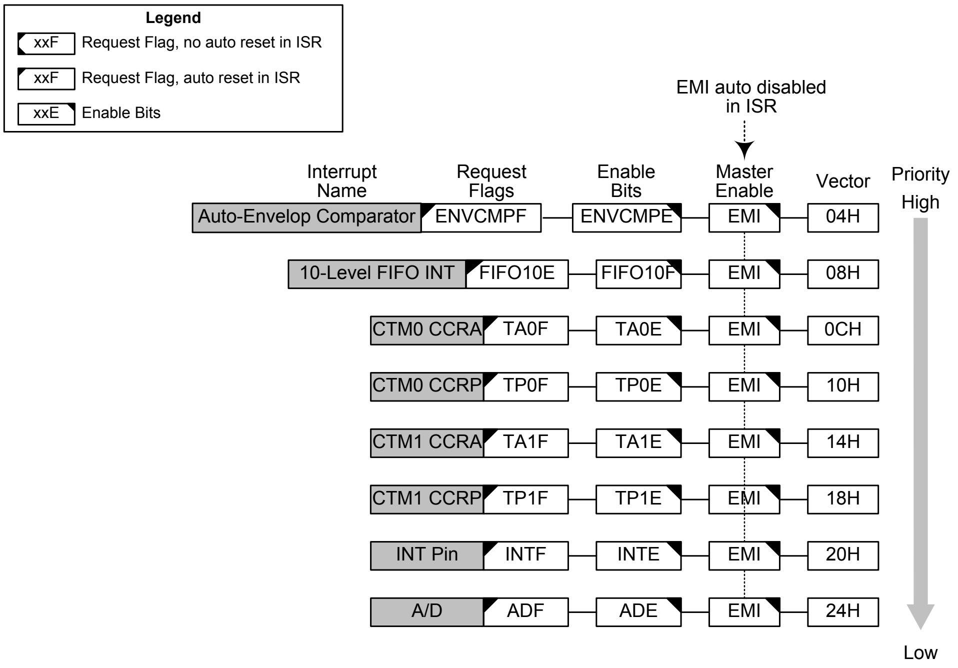
Interrupt Operation

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

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

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

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



Interrupt Structure

External Interrupt

The external interrupt is controlled by signal transition on the pin INT. An external interrupt request will take place when the external interrupt request flag, INTF is set, which will occur when a transition, whose type is chosen by the edge select bits, appears on the external interrupt pin. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and respective external interrupt enable bit INTE, must first be set. Additionally the correct interrupt edge type must be selected using the INTEDEG register to enable the external interrupt function and to choose the trigger edge type. As the external interrupt pin is pin-shared with I/O pin, it can only be configured as external interrupt pin if its external interrupt enable bit in the corresponding interrupt register has been set.

The pin must also be setup as an input by setting the corresponding bit in the port control register. When the interrupt is enabled, the stack is not full and the correct transition type appears on the external interrupt pin, a subroutine call to the external interrupt vector, will take place. When the interrupt is serviced, the external interrupt request flags, INT0F, INT1F, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts. Note that any pull-high resistor selections on the external interrupt pins will remain valid even if the pin is used as an external interrupt input.

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

Auto-Envelope Process Comparator Interrupt

The Auto-Envelope Process comparator interrupt is controlled by the comparing results. The comparing results can be decided by the options. When the comparing data is less than the SUM data, the compare output is "1", and when the comparing data is greater than the SUM data, the compare output is "0". The comparing output result can be used to be the source to generate an interrupt, ENVCMPINT. When the CMPOUT output status changes, its respective interrupt request flag, ENVCMPF will be set. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit EMI and ANP comparator interrupt enable bit ENVCMPE must first be set.

When the interrupt is enabled, the stack is not full and the comparator output status changes, a subroutine call to their respective vector locations will take place. When the interrupt is serviced, the respective interrupt request flag ENVCMPF will be automatically reset and the EMI bit will be cleared to disable other interrupts.

The ENVEDGE1 and ENVEDGE0 bits in ENVCMP register are used to select the type of active edge that will trigger the AEP comparator interrupt. A choice of either rising or falling or both edge types can be chosen to trigger an interrupt.

10-level FIFO Interrupt

The devices contain a 10-level FIFO Interrupt. When the 10 sets data are received or transmitted, the respective interrupt request flag FIFO10F will be set. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit EMI and the 10-level FIFO interrupt enable bit FIFO10E must first be set. When the interrupt is enabled, the stack is not full and the 10 sets data receive or transmit process has ended, a subroutine call to the 10-level FIFO Interrupt vector, will take place. When the interrupt is serviced, the Interrupt flag, FIFO10F, will be automatically cleared. The EMI bit will also be automatically 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.

TM Interrupt

The Compact Type TMs have two interrupts each. For each of the Compact Type TMs there are two interrupt request flags, TPnF and TAnF, and two enable bits TPnE and TAnE. A TM interrupt request will take place when any of the TM request flags is set, a situation which occurs when a TM comparator P or A match situation happens.

To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit EMI and respective TM Interrupt enable bit must first be set. When the interrupt is enabled, the stack is not full and a TM comparator match situation occurs, a subroutine call to the relevant Interrupt vector locations will take place. When the TM interrupt is serviced, the TM interrupt request flags will be automatically cleared and the EMI bit will be automatically cleared to disable other interrupts.

Interrupt Wake-up Function

Each of the interrupt functions has the capability of waking up the microcontroller when in the SLEEP 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 devices are in the SLEEP Mode and its system oscillator stopped, situations such as external edge transition on the external interrupt pin, FIFO 10 sets data are received or transmitted 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 devices enter the SLEEP 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.

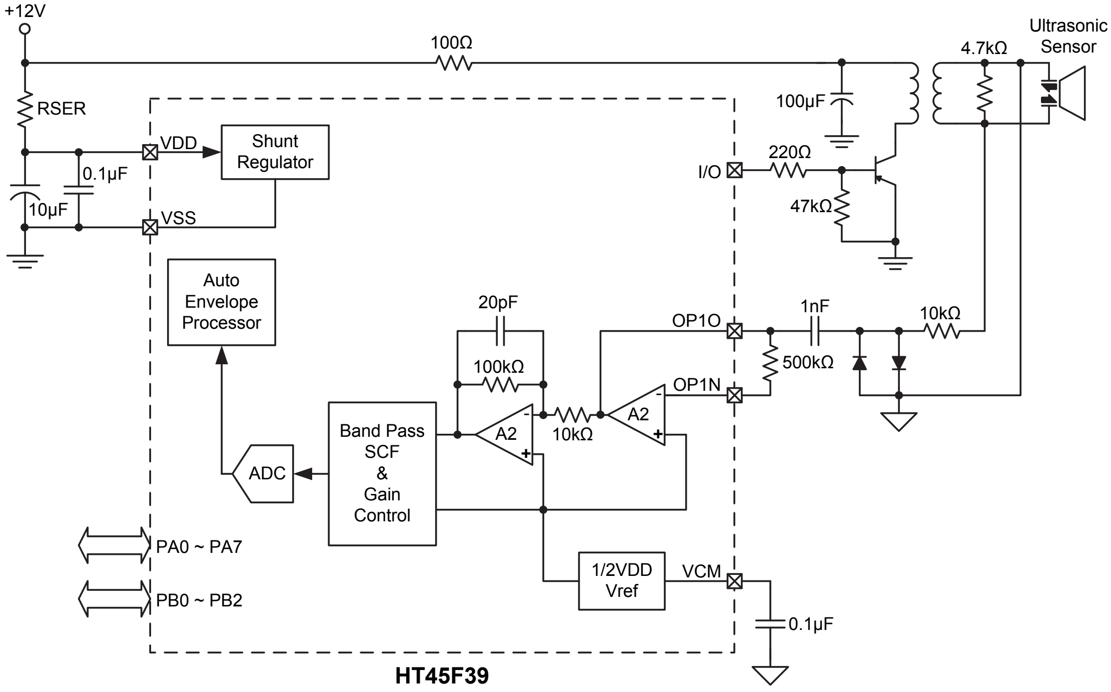
It is recommended that programs do not use the "CALL" instruction within the interrupt service subroutine. Interrupts often occur in an unpredictable manner or need to be serviced immediately. If only one stack is left and the interrupt is not well controlled, the original control sequence will be damaged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

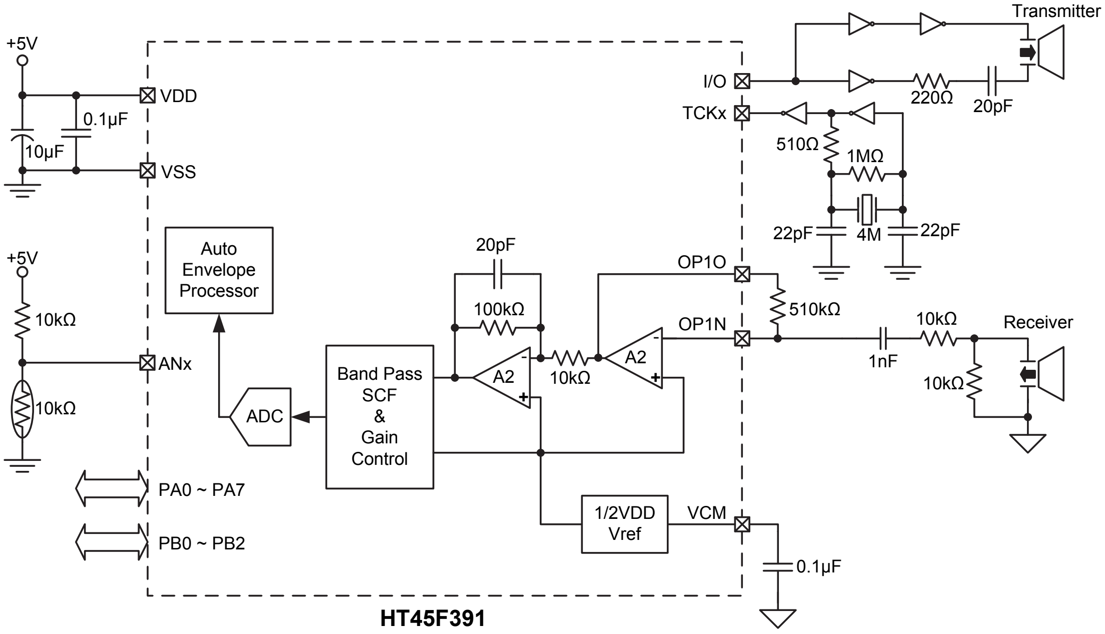
Every interrupt has the capability of waking up the microcontroller when it is in SLEEP Mode, the wake up being generated when the interrupt request flag changes from low to high. If it is required to prevent a certain interrupt from waking up the microcontroller then its respective request flag should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

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

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

Application Circuits





Instruction Set

Introduction

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

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

Instruction Timing

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

Moving and Transferring Data

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

Arithmetic Operations

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

Logical and Rotate Operations

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

Branches and Control Transfer

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

Bit Operations

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

Table Read Operations

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

Other Operations

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

Instruction Set Summary

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

Table Conventions

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

Mnemonic	Description	Cycles	Flag Affected
Arithmetic			
ADD A,[m]	Add Data Memory to ACC	1	Z, C, AC, OV
ADDM A,[m]	Add ACC to Data Memory	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	1^{Note}	None
MOV A,x	Move immediate data to ACC	1	None
Bit Operation			
CLR [m].i	Clear bit of Data Memory	1^{Note}	None
SET [m].i	Set bit of Data Memory	1^{Note}	None
Branch			
JMP addr	Jump unconditionally	2	None
SZ [m]	Skip if Data Memory is zero	1^{Note}	None
SZA [m]	Skip if Data Memory is zero with data movement to ACC	1^{Note}	None
SZ [m].i	Skip if bit i of Data Memory is zero	1^{Note}	None
SNZ [m].i	Skip if bit i of Data Memory is not zero	1^{Note}	None
SIZ [m]	Skip if increment Data Memory is zero	1^{Note}	None
SDZ [m]	Skip if decrement Data Memory is zero	1^{Note}	None
SIZA [m]	Skip if increment Data Memory is zero with result in ACC	1^{Note}	None
SDZA [m]	Skip if decrement Data Memory is zero with result in ACC	1^{Note}	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			
TABRD [m]	Read table to TBLH and Data Memory	2^{Note}	None
TABRDL [m]	Read table (last page) to TBLH and Data Memory	2^{Note}	None
Miscellaneous			
NOP	No operation	1	None
CLR [m]	Clear Data Memory	1^{Note}	None
SET [m]	Set Data Memory	1^{Note}	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	1^{Note}	None
SWAPA [m]	Swap nibbles of Data Memory with result in ACC	1	None
HALT	Enter power down mode	1	TO, PDF

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

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

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

Instruction Definition

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

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

Operation ACC ← ACC + [m] + C

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

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

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

Operation [m] ← ACC + [m] + C

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

ADD A,[m] Add Data Memory to ACC

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

Operation ACC ← ACC + [m]

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

ADD A,x Add immediate data to ACC

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

Operation ACC ← ACC + x

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

ADDM A,[m] Add ACC to Data Memory

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

Operation [m] ← ACC + [m]

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

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

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

Operation ACC ← ACC ″AND″ [m]

Affected flag(s) Z

AND A,x Logical AND immediate data to ACC

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

Operation ACC ← ACC ″AND″ x

Affected flag(s) Z

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

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

Operation [m] ← ACC ″AND″ [m]

Affected flag(s) Z

CALL addr Subroutine call

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

Operation Stack ← Program Counter + 1
Program Counter ← addr

Affected flag(s) None

CLR [m] Clear Data Memory

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

Operation [m] ← 00H

Affected flag(s) None

CLR [m].i Clear bit of Data Memory

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

Operation [m].i ← 0

Affected flag(s) None

CLR WDT Clear Watchdog Timer

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

Operation WDT cleared
TO ← 0
PDF ← 0

Affected flag(s) TO, PDF

CLR WDT1 Pre-clear Watchdog Timer

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

Operation WDT cleared
TO ← 0
PDF ← 0

Affected flag(s) TO, PDF

CLR WDT2 Pre-clear Watchdog Timer

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

Operation WDT cleared
TO ← 0
PDF ← 0

Affected flag(s) TO, PDF

CPL [m] Complement Data Memory

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

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

Affected flag(s) Z

CPLA [m] Complement Data Memory with result in ACC

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

Operation ACC ← $\overline{[m]}$

Affected flag(s) Z

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

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

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

Affected flag(s) C

DEC [m] Decrement Data Memory

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

Operation [m] ← [m] − 1

Affected flag(s) Z

DECA [m] Decrement Data Memory with result in ACC

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

Operation ACC ← [m] − 1

Affected flag(s) Z

HALT Enter power down mode

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

Operation TO ← 0
PDF ← 1

Affected flag(s) TO, PDF

INC [m] Increment Data Memory

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

Operation [m] ← [m] + 1

Affected flag(s) Z

INCA [m] Increment Data Memory with result in ACC

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

Operation ACC ← [m] + 1

Affected flag(s) Z

JMP addr Jump unconditionally

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

Operation Program Counter ← addr

Affected flag(s) None

MOV A,[m] Move Data Memory to ACC

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

Operation ACC ← [m]

Affected flag(s) None

MOV A,x Move immediate data to ACC

Description The immediate data specified is loaded into the Accumulator.

Operation ACC ← x

Affected flag(s) None

MOV [m],A Move ACC to Data Memory

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

Operation [m] ← ACC

Affected flag(s) None

NOP No operation

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

Operation No operation

Affected flag(s) None

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

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

Operation ACC ← ACC ″OR″ [m]

Affected flag(s) Z

OR A,x Logical OR immediate data to ACC

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

Operation ACC ← ACC ″OR″ x

Affected flag(s) Z

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

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

Operation [m] ← ACC ″OR″ [m]

Affected flag(s) Z

RET Return from subroutine

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

Operation Program Counter ← Stack

Affected flag(s) None

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

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

Operation Program Counter ← Stack
ACC ← x

Affected flag(s) None

RETI Return from interrupt

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

Operation Program Counter ← Stack
EMI ← 1

Affected flag(s) None

RL [m] Rotate Data Memory left

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

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

Affected flag(s) None

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

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

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

Affected flag(s) None

RLC [m] Rotate Data Memory left through Carry

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

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

Affected flag(s) C

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

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

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

Affected flag(s) C

RR [m] Rotate Data Memory right

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

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

Affected flag(s) None

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

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

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

Affected flag(s): None

RRC [m] Rotate Data Memory right through Carry

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

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

Affected flag(s): C

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

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

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

Affected flag(s): C

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

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

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

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

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

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

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

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 (current page) to TBLH and Data Memory

Description The low byte of the program code addressed by the table pointer (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

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

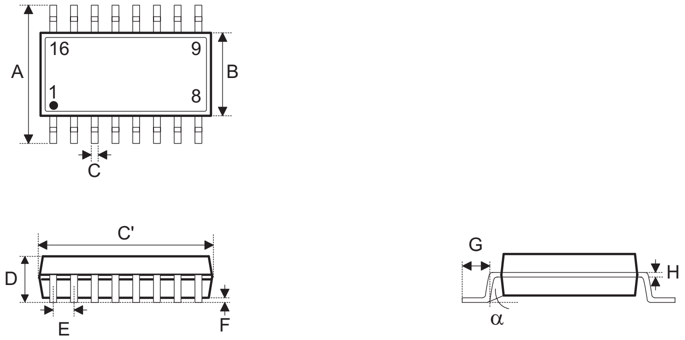
Package Information

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.

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

16-pin NSOP (150mil) Outline Dimensions



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

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	—	6.000 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°

The information appearing in this Data Sheet is believed to be accurate at the time of publication. However, Holtek assumes no responsibility arising from the use of the specifications described. The applications mentioned herein are used solely for the purpose of illustration and Holtek makes no warranty or representation that such applications will be suitable without further modification, nor recommends the use of its products for application that may present a risk to human life due to malfunction or otherwise. Holtek's products are not authorized for use as critical components in life support devices or systems. Holtek reserves the right to alter its products without prior notification. For the most up-to-date information, please visit our web site at http://www.holtek.com.