SPWM Inverter Output-end Flash MCU

HT45F5V

Revision: V1.10 Date: July 09, 2020

www.holtek.com

Features

CPU Features

- Operating Voltage: f_{SYS}=20MHz: 4.0V~5.5V
- Up to 0.2μs instruction cycle with 20MHz system clock at V_{DD}=5V
- Power down and wake-up functions to reduce power consumption
- Three oscillators
  - External Crystal – HXT
  - Internal 16MHz RC – HIRC
  - Internal 32kHz RC – LIRC
- Multi-mode operation: NORMAL, SLOW, IDLE and SLEEP
- Fully integrated internal 16MHz oscillator requires no external components
- All instructions executed in one or two instruction cycles
- Table read instructions
- 115 powerful instructions
- 6-level subroutine nesting
- Bit manipulation instruction

Peripheral Features

- Flash Program Memory: 4K×16
- RAM Data Memory: 256×8
- EEPROM Memory: 64×8
- Watchdog Timer function
- 24 bidirectional I/O lines
- Single pin-shared external interrupt
- Multiple Timer Module for time measure, input capture, compare match output, PWM output or single pulse output function (10-bit CTM×2, 16-bit STM×1)
- Dual Time-Base functions for generation of fixed time interrupt signals
- Multi-channel 12-bit resolution A/D converter
- Over Current Protection function
- Over Voltage Protection function
- Sine-Wave PWM generator function
- AC Zero Cross Detector function
- Low voltage reset function
- Low voltage detect function
- UART module for full duplex asynchronous communication
- Package types: 24-pin/28-pin SSOP

General Description

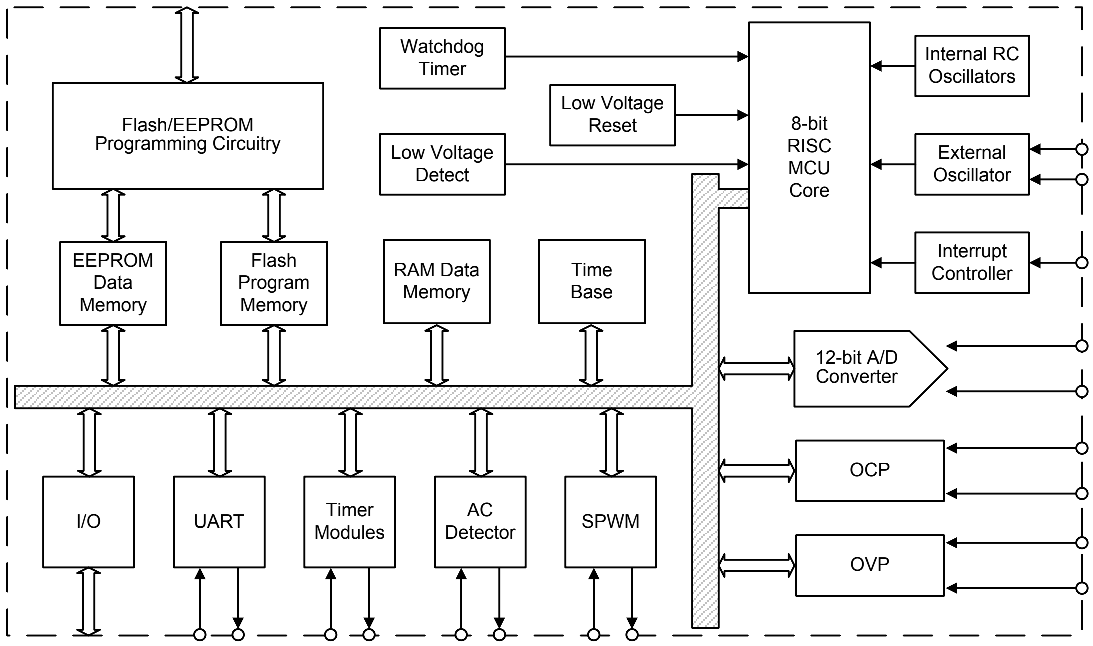
The HT45F5V is an ASSP MCU specifically designed for SPWM inverter output-end applications. Offering users the convenience of Flash Memory multi-programming features, the device also includes a wide range of functions and features. Other memory includes an area of RAM Data Memory as well as an area of EEPROM memory for storage of non-volatile data such as serial numbers, calibration data etc.

Analog features include a multi-channel 12-bit A/D converter, a sine wave PWM generator, an AC power detector. Multiple and extremely flexible Timer Modules provide timing, pulse generation and PWM generation functions. A UART module is also contained in the device. It can support applications such as data communication networks between microcontrollers, low-cost data links between PCs and peripheral devices, portable and battery operated device communication, etc. Protective features such as an internal Watchdog Timer, Low Voltage Reset and Low Voltage Detector coupled with excellent noise immunity and ESD protection ensure that reliable operation is maintained in hostile electrical environments. With regard to power, the protection functions also include Over Current Protection and Over Voltage Protection.

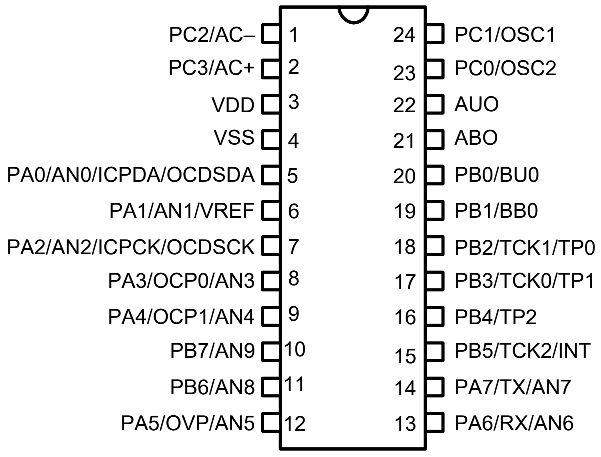
A full choice of external and internal oscillator functions are provided including a fully integrated system oscillator which requires no external components for its implementation. The ability to operate and switch dynamically between a range of operating modes using different clock sources gives users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

The inclusion of flexible I/O programming features, Time-Base functions along with many other features ensure that the device will find excellent use in SPWM inverter output-end application area.

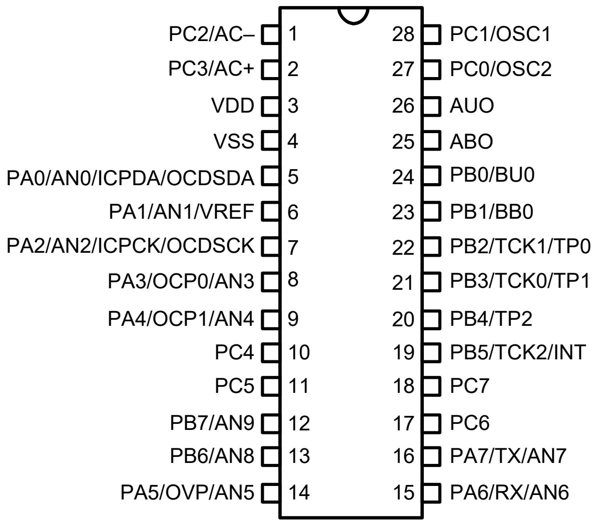
Block Diagram



Pin Assignment



HT45F5V
24 SSOP-A



HT45F5V
28 SSOP-A

Pin Description

Pin Name	Function	OP	I/T	O/T	Description
PA0/AN0/ICPDA/ OCDSDA	PA0	PAWU PAPU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	AN0	ADCR0 PAS0	AN	—	ADC input channel
	ICPDA	—	ST	CMOS	ICP address/data line
	OCDSDA	—	ST	CMOS	OCDS address/data line, for EV chip only.
PA1/AN1/VREF	PA1	PAWU PAPU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	AN1	ADCR0 PAS0	AN	—	ADC input channel
	VREF	ADCR1 PAS0	AN	—	ADC reference voltage input
PA2/AN2/ICPCK/ OCDSCK	PA2	PAWU PAPU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	AN2	ADCR0 PAS0	AN	—	ADC input channel
	ICPCK	—	ST	—	ICP clock line
	OCDSCK	—	ST	—	OCDS clock line, for EV chip only.
PA3/OCP0/AN3	PA3	PAWU PAPU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	OCP0	PAS0	AN	—	OCP input
	AN3	ADCR0 PAS0	AN	—	ADC input channel
PA4/OCP1/AN4	PA4	PAWU PAPU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	OCP1	PAS1	AN	—	OCP input
	AN4	ADCR0 PAS1	AN	—	ADC input channel
PA5/OVP/AN5	PA5	PAWU PAPU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	OVP	PAS1	AN	—	OVP input
	AN5	ADCR0 PAS1	AN	—	ADC input channel
PA6/RX/AN6	PA6	PAWU PAPU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	RX	PAS1	ST	—	UART data received input
	AN6	ADCR0 PAS1	AN	—	ADC input channel
PA7/TX/AN7	PA7	PAWU PAPU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	TX	PAS1	—	CMOS	UART data transmission output
	AN7	ADCR0 PAS1	AN	—	ADC input channel

Pin Name	Function	OP	I/T	O/T	Description
PB0/BUO	PB0	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-up.
	BUO	PBS0	—	CMOS	Sine-wave generator PWM output
PB1/BBO	PB1	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-up.
	BBO	PBS0	—	CMOS	Sine-wave generator PWM output
PB2/TCK1/TP0	PB2	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-up.
	TCK1	—	ST	—	TM clock input
	TP0	PBS0	ST	CMOS	TM I/O
PB3/TCK0/TP1	PB3	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-up.
	TCK0	—	ST	—	TM clock input
	TP1	PBS0	ST	CMOS	TM I/O
PB4/TP2	PB4	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-up.
	TP2	PBS0	ST	CMOS	TM I/O
PB5/TCK2/INT	PB5	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up.
	TCK2	—	ST	—	TM clock input
	INT	—	ST	—	External interrupt
PB6/AN8	PB6	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-up.
	AN8	ADCR0 PBS0	AN	—	ADC input channel
PB7AN9	PB7	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-up.
	AN9	ADCR0 PBS0	AN	—	ADC input channel
PC0/OSC2	PC0	PCPU PCS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	OSC2	PCS0	—	HXT	Oscillator output
PC1/OSC1	PC1	PCPU PCS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	OSC1	PCS0	HXT	—	Oscillator input
PC2/AC-	PC2	PCPU PCS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	AC-	PCS0	AN	—	AC detect input (-)
PC3/AC+	PC3	PCPU PCS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	AC+	PCS0	AN	—	AC detect input (+)
PC4~PC7	PC4~ PC7	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up
AUO	AUO	—	—	CMOS	Sine-wave generator PWM output
ABO	ABO	—	—	CMOS	Sine-wave generator PWM output
VDD	VDD	—	PWR	—	Digital positive power supply.
VSS	VSS	—	PWR	—	Digital negative power supply.

Note: I/T: Input type; O/T: Output type;
OP: Optional by register option; PWR: Power;
ST: Schmitt Trigger input; CMOS: CMOS output;
AN: Analog input pin; HXT: High frequency crystal oscillator

Absolute Maximum Ratings

Supply Voltage	V_{SS}−0.3V to V_{SS}+6.0V
Input Voltage	V_{SS}−0.3V to V_{DD}+0.3V
Storage Temperature	-50˚C to 125˚C
Operating Temperature	-40˚C to 85˚C

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

D.C. Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	Operating voltage	—	f_{SYS}=20MHz	4.0	—	5.5	V
I_{DD1}	Operating Current, Normal Mode, f_{SYS}=f_H	5	No load, f_H=20MHz, ADC off, WDT enable (HXT)	—	4.0	6.0	mA
			No load, f_H=16MHz, ADC off, WDT enable (HIRC)	—	2.8	4.2	
I_{DD2}	Operating Current, Slow Mode, f_{SYS}= f_{SUB}=LIRC	5	No load, f_{SYS}=LIRC, ADC off, WDT enable	—	30	50	μA
I_{IDLE0}	IDLE0 Mode Standby Current (LIRC on)	5	No load, ADC off, WDT enable, LVR disable	—	2.5	5.0	μA
I_{IDLE1}	IDLE1 Mode Standby Current	5	No load, ADC off, WDT enable, f_{SYS}=20MHz on (HXT)	—	2.2	3.3	mA
			No load, ADC off, WDT enable, f_{SYS}=16MHz on (HIRC)	—	1.4	2.1	
I_{SLEEP}	SLEEP Mode Standby Current (LIRC on)	5	No load, ADC off, WDT enable, LVR disable	—	2.5	5.0	μA
V_{IL}	Input Low Voltage for PA, PB, INT, TPn	5	—	0	—	1.5	V
		—		0	—	0.2V_{DD}	V
V_{IH}	Input High Voltage for PA, PB, INT, TPn	5	—	3.5	—	5	V
		—		0.8V_{DD}	—	V_{DD}	V
V_{LVR}	Low Voltage Reset Voltage	—	LVR Enable, 3.8V option	-5%	3.8	+5%	V
V_{LVD}	LVD Voltage Level	—	LVDEN=1, V_{LVD}=2.0V	-5%	2.0	+5%	V
		—	LVDEN=1, V_{LVD}=2.2V	-5%	2.2	+5%	V
		—	LVDEN=1, V_{LVD}=2.4V	-5%	2.4	+5%	V
		—	LVDEN=1, V_{LVD}=2.7V	-5%	2.7	+5%	V
		—	LVDEN=1, V_{LVD}=3.0V	-5%	3.0	+5%	V
		—	LVDEN=1, V_{LVD}=3.3V	-5%	3.3	+5%	V
		—	LVDEN=1, V_{LVD}=3.6V	-5%	3.6	+5%	V
		—	LVDEN=1, V_{LVD}=4.0V	-5%	4.0	+5%	V
I_{OH}	I/O source current (PA, PB)	5	V_{OH}=0.9V_{DD}	-5	-10	—	mA
I_{OL}	I/O sink current (PA, PB)	5	V_{OL}=0.1V_{DD}	10	20	—	mA
R_{PH}	Pull-high resistance	5	—	10	30	50	kΩ

A.C. Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
f_{SYS}	System clock	4.0~5.5V	—	—	—	20	MHz
f_{LIRC}	System Clock (LIRC)	5V	Ta = 25°C	-10%	32	+10%	kHz
		2.2V~5.5V	Ta = -40°C to 85°C	-30%	32	+60%	kHz
f_{HIRC}	System clock (HIRC)	4.5V~5.5V	Ta = 0°C to 85°C	-4%	16	+4%	MHz
t_{TIMER}	TCKn, TPn Input Pulse Width	—	—	0.3	—	—	μS
t_{INT}	Interrupt Pulse Width	—	—	10	—	—	μS
t_{EERD}	EEPROM Read Time	5V	—	—	2	4	t_{SYS}
t_{EEWR}	EEPROM Write Time	5V	—	—	2	4	ms
t_{SST}	System Start-up Timer Period (Power On Reset)	5V	—	—	128	—	t_{SYS}
	Slow → Normal mode (f_H=HXT, f_L=LIRC)	5V	—	—	1024	—	
	Normal → Slow mode (f_H=HXT, f_L=LIRC)	5V	—	—	2	—	
	System Start-up Timer Period (Wake-up from HALT, f_{SYS} off at HALT state	5V	f_{SYS} = HXT	—	128	—	
		5V	f_{SYS} = LIRC	—	2	—	
	System Start-up Timer Period (Wake-up from HALT, f_{SYS} on at HALT state)	5V	—	—	2	—	
t_{RSTD}	System Reset Delay Time (Power On Reset, LVR reset WDTC/ SW reset)	5V	—	25	50	100	ms
	System Reset Delay Time (WDT time-out reset)	5V	—	8.3	16.7	33.3	ms

A/D Converter Characteristics

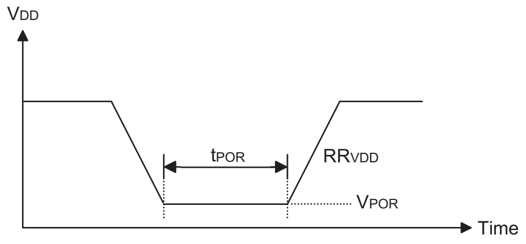
Ta=25°C

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

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	—	—	10	—	—	μs

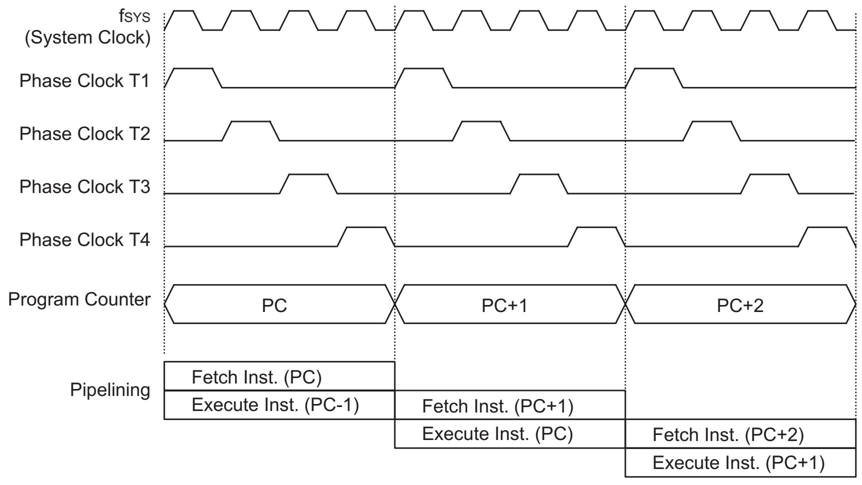


System Architecture

A key factor in the high-performance features of the Holtek range of microcontrollers is attributed to their internal system architecture. The range of 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 instruction set operations, which carries out arithmetic operations, logic operations, rotation, increment, decrement, branch decisions, etc. The internal data path is simplified by moving data through the Accumulator and the ALU. Certain internal registers are implemented in the Data Memory and can be directly or indirectly addressed. The simple addressing methods of these registers along with additional architectural features ensure that a minimum of external components is required to provide a functional I/O and A/D control system with maximum reliability and flexibility. This makes the devices suitable for low-cost, high-volume production for controller applications.

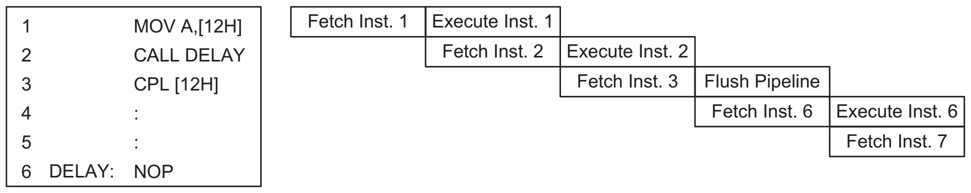
Clocking and Pipelining

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



System Clocking and Pipelining

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



Instruction Fetching

Program Counter

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

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

Device	Program Counter	
	Program Counter High Byte	PCL Register
HT45F5V	PC11~PC8	PCL7~PCL0

Program Counter

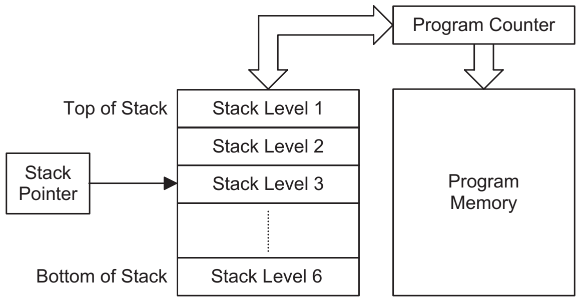
The lower byte of the Program Counter, known as the Program Counter Low register or PCL, is available for program control and is a readable and writeable register. By transferring data directly into this register, a short program jump can be executed directly. However, as only this low byte is available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

Stack

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

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

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



Arithmetic and Logic Unit – ALU

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

- Arithmetic operations, ADD, ADDM, ADC, ADCM, SUB, SUBM, SBC, SBCM, DAA, LADD, LADDM, LADC, LADCM, LSUB, LSUBM, LSBC, LSBCM, LDAA
- Logic operations, AND, OR, XOR, ANDM, ORM, XORM, CPL, CPLA, LAND, LANDM, LOR, LORM, LXOR, LXORM, LCPL, LCPLA
- Rotation, RRA, RR, RRCA, RRC, RLA, RL, RLCA, RLC, LRR, LRRA, LRRCA, LRRC, LRLA, LRLCA, LRLC
- Increment and Decrement, INCA, INC, DECA, DEC, LINCA, LINC, LDECA, LDEC
- Branch decision, JMP, SZ, SZA, SNZ, SIZ, SDZ, SIZA, SDZA, CALL, RET, RETI, LSNZ, LSZ, LSZA, LSIZ, LSDZ, LSIZA, LSDZA

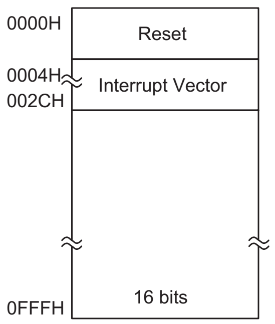
Flash Program Memory

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

Device	Capacity
HT45F5V	4K×16



Program Memory Structure

Special Vectors

Within the Program Memory, certain locations are reserved for the reset and interrupts. The location 0000H is reserved for use by the device reset for program initialisation. After a device reset is initiated, the program will jump to this location and begin execution.

Look-up Table

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

After setting up the table pointer, the table data can be retrieved from the Program Memory using the "TABRD [m]" or "TABRDL [m]" instructions respectively when the memory [m] is located in current page. If the memory [m] is located in other pages, the table data can be retrieved from the Program Memory using the "LTABRD [m]" or "LTABRDL [m]" instructions respectively. When the instruction is executed, the lower order table byte from the Program Memory will be transferred to the user defined Data Memory register [m] as specified in the instruction. The higher order table data byte from the Program Memory will be transferred to the TBLH special register. Any unused bits in this transferred higher order byte will be read as 0.

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

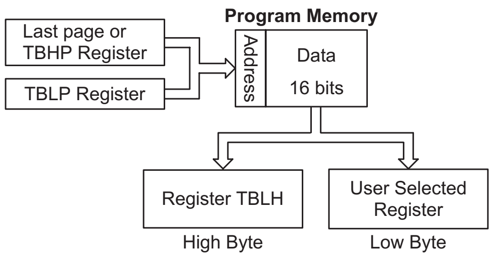


Table Program Example

The following example shows how the table pointer and table data is defined and retrieved from the microcontroller. This example uses raw table data located in the Program Memory which is stored there using the ORG statement. The value at this ORG statement is "F00H" which refers to the start address of the last page within the 4K Program Memory of the device. The table pointer is setup here to have an initial value of "06H". This will ensure that the first data read from the data table will be at the Program Memory address "F06H" or 6 locations after the start of the last page. Note that the value for the table pointer is referenced to the first address of the present page if the "TABRD [m]" or "LTABRD [m]" instruction is being used. The high byte of the table data which in this case is equal to zero will be transferred to the TBLH register automatically when the "TABRD [m]" or "LTABRD [m]" instruction is executed.

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

Table Read Program Example

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

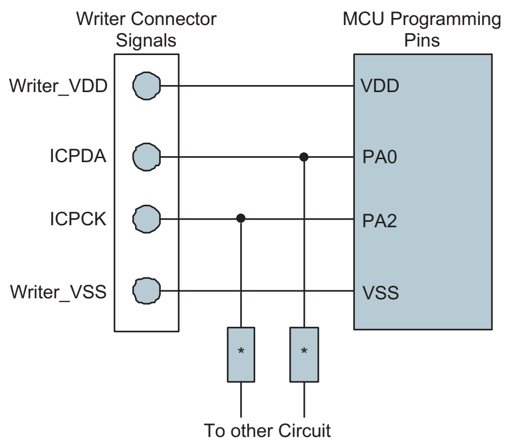
In-Circuit Programming – ICP

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

Holtek Writer Pins	MCU Programming Pins	Pin Description
ICPDA	PA0	Serial Data/Address input/output
ICPCK	PA2	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 one line for the reset. 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 HT45V5V which is used to emulate the HT45F5V device. The EV chip device also provides the "On-Chip Debug" function to debug the real MCU device during development process. The EV chip and real MCU devices are almost functional compatible except the "On-Chip Debug" function and package types. Users can use the EV chip device to emulate the real MCU device behaviors by connecting the OCDSDA and OCDSCK pins to the Holtek HT-IDE development tools. The OCDSDA pin is the OCDS Data/Address input/output pin while the OCDSCK pin is the OCDS clock input pin. When users use the EV chip for debugging, the corresponding pin functions shared with the OCDSDA and OCDSCK pins in the 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
VSS	VSS	Ground

RAM Data Memory

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

Divided into two types, 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 device. Many of these registers can be read from and written to directly under program control, however, some remain protected from user manipulation. The second area of Data Memory is reserved for general purpose use. All locations within this area are read and write accessible under program control.

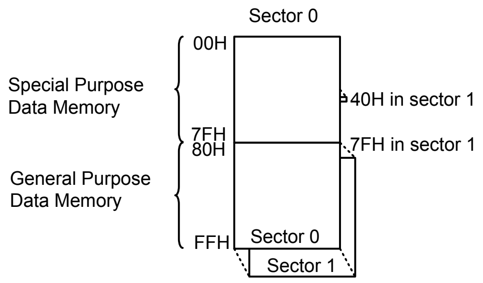
Switching between the different Data Memory sectors is achieved by setting the Memory Pointers to the correct value.

Structure

The Data Memory is subdivided into two sectors, all of which are implemented in 8-bit wide RAM. Each of the Data Memory Sector is categorized into two types, the special Purpose Data Memory and the General Purpose Data Memory.

The start address of the Special Purpose Data Memory is the address 00H while the start address of the General Purpose Data Memory is the address 80H. The Special Purpose Data Memory registers are only accessible in sector 0, with the exception of EEC register at address 40H, which is only accessible in Sector 1.

Device	Capacity	Sectors
HT45F5V	General Purpose: 256×8	0: 80H~FFH 1: 80H~FFH



Data Memory Structure

General Purpose Data Memory

There are 256 bytes of general purpose memory which are arranged in 80H~FFH of Sector 0, Sector 1 separately. All microcontroller programs require an area of read/write memory where temporary data can be stored and retrieved for use later. It is this area of RAM memory that is known as General Purpose Data Memory. The general purpose data memory is fully accessible by the user program for both read and writing operations. By using the "SET [m].i" and "CLR [m].i" instructions individual bits can be set or reset under program control giving the user a large range of flexibility for bit manipulation in the Data Memory.

Special Purpose Data Memory

This area of Data Memory is where registers, necessary for the correct operation of the microcontroller, are stored. They are overlapped in any sector. Most of the registers are both readable and writable but some are protected and are readable only, the details of which are located under the relevant Special Function Register section. Note that for locations that are unused before 80H, any read instruction to these addresses will return the value "00H".

	Sector 0	Sector 1		Sector 0	Sector 1		Sector 0	Sector 1		Sector 0	Sector 1
00H	IAR0	Unused	20H	TBC0	Unused	40H	Unused	EEC	60H	SPWMC0	Unused
01H	MP0	Unused	21H	TBC1	Unused	41H	EEA	Unused	61H	SPWMC1	Unused
02H	IAR1	Unused	22H	PSCR	Unused	42H	EED	Unused	62H	SPWML	Unused
03H	MP1L	Unused	23H	WDTC	Unused	43H	TM0C0	Unused	63H	SPWMH	Unused
04H	MP1H	Unused	24H	LVDC	Unused	44H	TM0C1	Unused	64H	SFIFOL	Unused
05H	ACC	Unused	25H	Unused	Unused	45H	TM0DL	Unused	65H	SFIFOH	Unused
06H	PCL	Unused	26H	RSTC	Unused	46H	TM0DH	Unused	66H	MTF	Unused
07H	TBLP	Unused	27H	Unused	Unused	47H	TM0AL	Unused	67H	SPWMDT	Unused
08H	TBLH	Unused	28H	ADRL	Unused	48H	TM0AH	Unused	68H	ACDC	Unused
09H	TBHP	Unused	29H	ADRH	Unused	49H	TM1C0	Unused	69H	ACDOFF	Unused
0AH	STATUS	Unused	2AH	ADCR0	Unused	4AH	TM1C1	Unused	6AH	OCP0C0	Unused
0BH	Unused	Unused	2BH	ADCR1	Unused	4BH	TM1DL	Unused	6BH	OCP0C1	Unused
0CH	IAR2	Unused	2CH	SCC	Unused	4CH	TM1DH	Unused	6CH	OCP0REF	Unused
0DH	MP2L	Unused	2DH	HXTC	Unused	4DH	TM1AL	Unused	6DH	OCP0ACAL	Unused
0EH	MP2H	Unused	2EH	HIRCC	Unused	4EH	TM1AH	Unused	6EH	OCP0CCAL	Unused
0FH	Unused	Unused	2FH	Unused	Unused	4FH	TM2C0	Unused	6FH	OCP1C0	Unused
10H	INTC0	Unused	30H	INTEG	Unused	50H	TM2C1	Unused	70H	OCP1C1	Unused
11H	Unused	Unused	31H	INTC1	Unused	51H	TM2DL	Unused	71H	OCP1REF	Unused
12H	PA	Unused	32H	INTC2	Unused	52H	TM2DH	Unused	72H	OCP1ACAL	Unused
13H	PAC	Unused	33H	MFI0	Unused	53H	TM2AL	Unused	73H	OCP1CCAL	Unused
14H	PAPU	Unused	34H	MFI1	Unused	54H	TM2AH	Unused	74H	OVPC	Unused
15H	PAWU	Unused	35H	MFI2	Unused	55H	TM2RP	Unused	75H	OVPREF	Unused
16H	Unused	Unused	36H	Unused	Unused	56H	Unused	Unused	76H	OVPCCAL	Unused
17H	RSTFC	Unused	37H	Unused	Unused	57H	Unused	Unused	77H	Unused	Unused
18H	Unused	Unused	38H	Unused	Unused	58H	USR	Unused	78H	Unused	Unused
19H	Unused	Unused	39H	Unused	Unused	59H	UCR1	Unused	79H	Unused	Unused
1AH	PB	Unused	3AH	PAS0	Unused	5AH	UCR2	Unused	7AH	Unused	Unused
1BH	PBC	Unused	3BH	PAS1	Unused	5BH	BRG	Unused	7BH	Unused	Unused
1CH	PBPU	Unused	3CH	PBS0	Unused	5CH	TXR_RXR	Unused	7CH	Unused	Unused
1DH	PC	Unused	3DH	IFS0	Unused	5DH	Unused	Unused	7DH	Unused	Unused
1EH	PCC	Unused	3EH	PCS0	Unused	5EH	Unused	Unused	7EH	Unused	Unused
1FH	PCPU	Unused	3FH	Unused	Unused	5FH	Unused	Unused	7FH	Unused	Unused

▨ : Unused, read as 00H

Special Purpose Data Memory

Special Function Register Description

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

Indirect Addressing Register – IAR0, IAR1, IAR2

The Indirect Addressing Registers, IAR0, IAR1 and IAR2, 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, IAR1 and IAR2 registers will result in no actual read or write operation to these registers but rather to the memory location specified by their corresponding Memory Pointers, MP0, MP1L/MP1H or MP2L/MP2H. Acting as a pair, IAR0 and MP0 can together access data only from Sector 0 while the IAR1 register together with MP1L/MP1H register pair and IAR2 register together with MP2L/MP2H register pair can access data from any Data Memory sector. 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, MP1L, MP1H, MP2L, MP2H

Five Memory Pointers, known as MP0, MP1L, MP1H, MP2L and MP2H, are provided. These Memory Pointers are physically implemented in the Data Memory and can be manipulated in the same way as normal registers providing a convenient way with which to address and track data. When any operation to the relevant Indirect Addressing Registers is carried out, the actual address that the microcontroller is directed to is the address specified by the related Memory Pointer. MP0, together with Indirect Addressing Register, IAR0, are used to access data from Sector 0, while MP1L/MP1H together with IAR1 and MP2L/MP2H together with IAR2 are used to access data from all data sectors according to the corresponding MP1H or MP2H register. Direct Addressing can be used in all data sectors using the corresponding instruction which can address all available data memory space.

The following example shows how to clear a sector 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, TBHP, TBLH

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

Status Register – STATUS

This 8-bit register contains the SC flag, CZ flag, 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, C, SC and CZ flags generally reflect the status of the latest operations.

- SC is the result of the "XOR" operation which is performed by the OV flag and the MSB of the current instruction operation result.
- CZ is the operational result of different flags for different instructions. Refer to register definitions for more details.
- C is set if an operation results in a carry during an addition operation or if a borrow does not take place during a subtraction operation; otherwise C is cleared. C is also affected by a rotate through carry instruction.
- AC is set if an operation results in a carry out of the low nibbles in addition, or no borrow from the high nibble into the low nibble in subtraction; otherwise AC is cleared.
- Z is set if the result of an arithmetic or logical operation is zero; otherwise Z is cleared.
- OV is set if an operation results in a carry into the highest-order bit but not a carry out of the highest-order bit, or vice versa; otherwise OV is cleared.
- PDF is cleared by a system power-up or executing the "CLR WDT" instruction. PDF is set by executing the "HALT" instruction.
- TO is cleared by a system power-up or executing the "CLR WDT" or "HALT" instruction. TO is set by a WDT time-out.

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

STATUS Register

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

"x" unknown

Bit 7 **SC**: The result of the "XOR" operation which is performed by the OV flag and the MSB of the instruction operation result.

Bit 6 **CZ**: The operational result of different flags for different instructions.
For SUB/SUBM/LSUB/LSUBM instructions, the CZ flag is equal to the Z flag.
For SBC/ SBCM/ LSBC/ LSBCM instructions, the CZ flag is the "AND" operation result which is performed by the previous operation CZ flag and current operation zero flag.
For other instructions, the CZ flag will not be affected.

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

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

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

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

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

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

EEPROM Data memory

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

EEPROM Data Memory Structure

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

EEPROM Registers

Three registers control the overall operation of the internal EEPROM Data Memory. These are the address register, EEA, the data register, EED and a single control register, EEC. As both the EEA and EED registers are located in Sector 0, they can be directly accessed in the same was as any other Special Function Register. The EEC register however, being located in Sector1, cannot be directly addressed directly and can only be read from or written to indirectly using the MP1L/ MP1H Memory Pointer and Indirect Addressing Register, IAR1. Because the EEC control register is located at address 40H in Sector 1, the MP1L Memory Pointer low byte must first be set to the value 40H and the MP1H Memory Pointer high byte set to the value 01H before any operations on the EEC register are executed.

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

EEPROM Register List

EEA Register

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

Bit 7~6	Unimplemented, read as "0"
Bit 5~0	**D5~D0**: Data EEPROM address Data EEPROM address bit 5~bit 0

EED Register

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

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

EEC Register

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

Bit 7~4 Unimplemented, read as "0"

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

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

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

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

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

Reading Data from the EEPROM

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

Writing Data to the EEPROM

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

Write Protection

Protection against inadvertent write operation is provided in several ways. After the device is powered-on the Write Enable bit in the control register will be cleared preventing any write operations. Also at power-on MP1L/MP1H and MP2L/MP2H will be reset to zero, which means that Data Memory Sector 0 will be selected. As the EEPROM control register is located in Sector 1, this adds a further measure of protection against spurious write operations. During normal program operation, ensuring that the Write Enable bit in the control register is cleared will safeguard against incorrect write operations.

EEPROM Interrupt

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

Programming Considerations

Care must be taken that data is not inadvertently written to the EEPROM. Protection can be enhanced by ensuring that the Write Enable bit is normally cleared to zero when not writing. Also the Memory Pointer high byte, MP1H or MP2H, could be normally cleared to zero as this would inhibit access to Sector 1 where the EEPROM control register exist. Although certainly not necessary, consideration might be given in the application program to the checking of the validity of new write data by a simple read back process. When writing data the WR bit must be set high immediately after the WREN bit has been set high, to ensure the write cycle executes correctly. The global interrupt bit EMI should also be cleared before a write cycle is executed and then re-enabled after the write cycle starts. Note that the device should not enter the IDLE or SLEEP mode until the EEPROM read or write operation is totally complete. Otherwise, the EEPROM read or write operation will fail.

Programming Examples

Reading data from the EEPROM – polling method

```
MOV A, EEPROM_ADRES       ; user defined address
MOV EEA, A
MOV A, 040H               ; setup memory pointer MP1L
MOV MP1L, A               ; MP1 points to EEC register
MOV A, 01H                ; setup memory pointer MP1H
MOV MP1H, A
SET IAR1.1                ; set RDEN bit, enable read operations
SET IAR1.0                ; start Read Cycle - set RD bit
BACK:
SZ IAR1.0                 ; check for read cycle end
JMP BACK
CLR IAR1                  ; disable EEPROM read/write
CLR MP1H
MOV A, EED                ; move read data to register
MOV READ_DATA, A
```

Writing Data to the EEPROM – polling method

```
MOV A, EEPROM_ADRES       ; user defined address
MOV EEA, A
MOV A, EEPROM_DATA        ; user defined data
MOV EED, A
MOV A, 040H               ; setup memory pointer MP1L
MOV MP1L, A               ; MP1 points to EEC register
MOV A, 01H                ; setup memory pointer MP1H
MOV MP1H, A
CLR EMI
SET IAR1.3                ; set WREN bit, enable write operations
SET IAR1.2                ; start Write Cycle - set WR bit – executed immediately after
                          ; set WREN bit
SET EMI
BACK:
SZ IAR1.2                 ; check for write cycle end
JMP BACK
CLR IAR1                  ; disable EEPROM read/write
CLR MP1H
```

Oscillator

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

Oscillator Overview

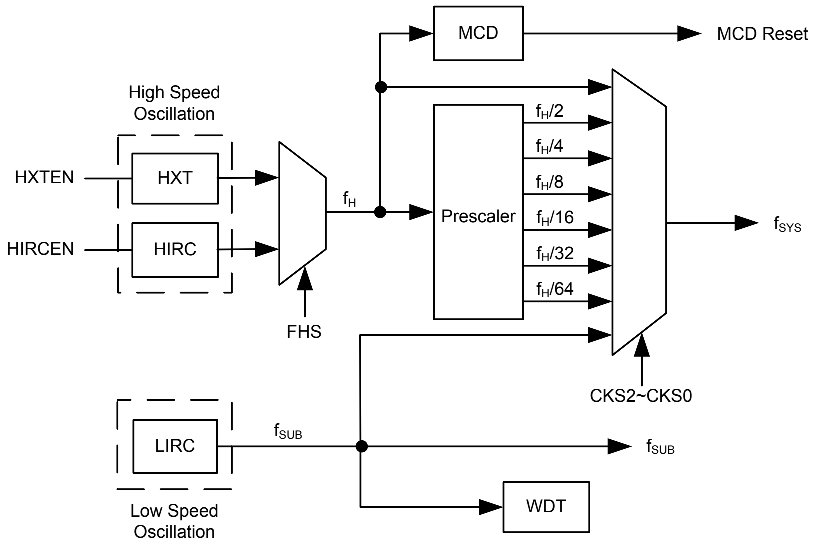
In addition to being the source of the main system clock the oscillators also provide clock sources for the Watchdog Timer and Time Base Interrupts. External oscillators requiring some external components as well as fully integrated internal oscillators, requiring no external components, are provided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

Type	Name	Freq.	Pins
External High Speed Crystal	HXT	20MHz	OSC1/OSC2
Internal High Speed RC	HIRC	16MHz	—
Internal Low Speed RC	LIRC	32kHz	—

Oscillator Types

System Clock Configurations

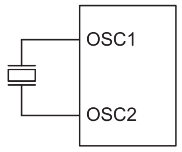
There are three methods of generating the system clock, two high speed oscillator and one low speed oscillators. The high speed oscillators are the internal 16MHz RC oscillator – HIRC and the external high speed oscillator – HXT. The low speed oscillator is the internal 32kHz RC oscillator – LIRC. Selecting whether the low or high speed oscillator is used as the system oscillator is implemented using the CKS2~CKS0 bits in the SCC register and as the system clock can be dynamically selected. It is not possible to choose a no-oscillator selection for either the high or low speed oscillator.



System Clock Configurations

External Crystal/Ceramic Oscillator – HXT

The External Crystal/Ceramic System Oscillator is one of the high frequency oscillator choices, which is selected via the FHS bit in SCC register. This oscillator has a fixed frequency of 20MHz. The HXT with crystal connected between OSC1 and OSC2 will create the necessary oscillation.



Crystal/Resonator Oscillator – HXT

Internal RC Oscillator – HIRC

The internal RC oscillator is a fully integrated system oscillator requiring no external components. The internal RC oscillator has a fixed frequency of 16MHz. Device trimming during the manufacturing process and the inclusion of internal frequency compensation circuits are used to ensure that the influence of the power supply voltage, temperature and process variations on the oscillation frequency are minimised. Note that if this internal system clock option is selected, as it requires no external pins for its operation, I/O pins are free for use as normal I/O pins.

Internal 32kHz Oscillator – LIRC

The Internal 32kHz Oscillator is a fully integrated RC oscillator with a typical frequency of 32kHz at 5V, requiring no external components for its implementation. Device trimming during the manufacturing process and the inclusion of internal frequency compensation circuits are used to ensure that the influence of the power supply voltage, temperature and process variations on the oscillation frequency are minimised. As a result, at a power supply of 5V and at a temperature of 25°C degrees, the fixed oscillation frequency of 32kHz will have a tolerance within 10%.

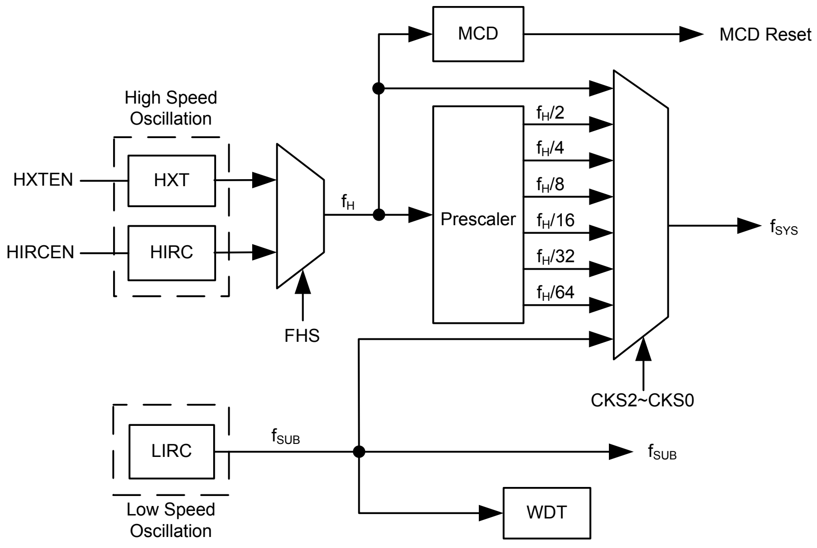
Operating Modes and System Clocks

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

System Clocks

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

The main system clock, can come from either a high frequency f_H or low frequency f_{SUB} source, and is selected using the CKS2~CKS0 bits in the SCC register. The high speed system clock can be sourced either from the HIRC or the HXT oscillator by configuring the FHS bit. The low speed system clock source is sourced from the LIRC oscillator. The other choice, which is a divided version of the high speed system oscillator has a range of $f_H/2$~$f_H/64$.



System Clock Configuration

Note: When the system clock source f_{SYS} is switched to f_{SUB} from f_H, the high speed oscillation will stop to conserve the power or continue to oscillator to provide the clock source, f_H~$f_H/64$, for peripheral circuit to use, which is determined by configuring the corresponding high speed oscillator enable control bit.

System Operation Modes

There are six different modes of operation for the microcontroller, each one with its own special characteristics and which can be chosen according to the specific performance and power requirements of the application. There are two modes allowing normal operation of the microcontroller, the NORMAL Mode and SLOW Mode. The remaining four modes, the SLEEP, IDLE0, IDLE1 and IDLE2 Mode are used when the microcontroller CPU is switched off to conserve power.

Operation Mode	CPU	Register Setting			f_{SYS}	f_H	f_{SUB}
		FHIDEN	FSIDEN	CKS2~CKS0			
NORMAL	On	×	×	000~110	f_H~f_H/64	On	On
SLOW	On	×	×	111	f_{SUB}	On/Off [1]	On
IDLE0	Off	0	1	000~110	Off	Off	On
				111	On		
IDLE1	Off	1	1	×	On	On	On
IDLE2	Off	1	0	000~110	On	On	Off
				111	Off		
SLEEP	Off	0	0	×	Off	Off	Off

Note: 1. The f_H clock will be switched on or off by configuring the corresponding oscillator enable bit in the SLOW mode.

2. If the f_{SUB} clock is derived from the LIRC oscillator, the f_{SUB} clock will be switched on or off which is controlled by the WDT function being enabled or disabled.

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 one of the high speed oscillators. This mode operates allowing the microcontroller to operate normally with a clock source will come from one of the high speed oscillators, HIRC or HXT. The high speed oscillator will however first be divided by a ratio ranging from 1 to 64, the actual ratio being selected by the CKS2~CKS0 bits in the SCC register. Although a high speed oscillator is used, running the microcontroller at a divided clock ratio reduces the operating current.

SLOW Mode

This is also a mode where the microcontroller operates normally although now with a slower speed clock source f_{SUB}. The clock source is derived from the low speed oscillator LIRC. Running the microcontroller in this mode allows it to run with much lower operating currents.

SLEEP Mode

The SLEEP Mode is entered when an HALT instruction is executed and when the FHIDEN and FSIDEN bits are both low. In the SLEEP mode the CPU will be stopped, the f_{SUB} clock will also be off. However, the f_{SUB} clock derived from the f_{LIRC} clock will be on if the Watchdog Timer function is enabled.

IDLE0 Mode

The IDLE0 Mode is entered when a HALT instruction is executed and when the FSIDEN bit in the SCC register is high and the FHIDEN bit in the SCC register is low. In the IDLE0 Mode the CPU will be switched off but the low speed oscillator will be turned on to drive some peripheral functions.

IDLE1 Mode

The IDLE1 Mode is entered when an HALT instruction is executed and when the FHIDEN and FSIDEN bits in the SCC register are both high. In the IDLE1 Mode the CPU will be switched off but both the high and low speed oscillator will be turned on to provide a clock source to keep some peripheral functions operational.

IDLE2 Mode

The IDLE2 Mode is entered when an HALT instruction is executed and when the FHIDEN bit in the SCC register is high and the FSIDEN bit in the SCC register is low. In the IDLE2 Mode the CPU will be switched off but the high speed oscillator will be turned on to provide a clock source to keep some peripheral functions operational.

Missing Clock Detector Function – MCD

There is a Missing Clock Detector, MCD, in this device. The MCD is used to detect the high speed oscillator operation when the corresponding oscillator is enabled. If the oscillator is enabled and no clock cycle is detected by the MCD in certain period of time, it means that the oscillator does not oscillate successfully and then the MCD will generate a signal to reset the microcontroller.

Control Register

The register, SCC, is used for overall control of the internal clocks within the device. The register HXTC is used for HXT oscillator control while the HIRCC register is for the HIRC oscillator control.

SCC Register

Bit	7	6	5	4	3	2	1	0
Name	CKS2	CKS1	CKS0	—	FHS	—	FHIDEN	FSIDEN
R/W	R/W	R/W	R/W	—	R/W	—	R/W	R/W
POR	0	0	0	—	0	—	0	0

Bit 7~5 **CKS2~CKS0**: The system clock selection
000: f_H
001: $f_H/2$
010: $f_H/4$
011: $f_H/8$
100: $f_H/16$
101: $f_H/32$
110: $f_H/64$
111: f_{SUB}
These three bits are used to select which clock is used as the system clock source. In addition to the system clock source directly derived from f_H or f_{SUB}, a divided version of the high speed system oscillator can also be chosen as the system clock source.

Bit 4,2 Unimplemented, read as "0"

Bit 3 **FHS**: High Frequency Clock selection
0: HIRC
1: HXT

Bit 1 **FHIDEN**: High frequency oscillator control when CPU is switched off
0: Disable
1: Enable

Bit 0 **FSIDEN**: Low frequency oscillator control when CPU is switched off
0: Disable
1: Enable
This LIRC is used to control whether the low speed oscillator is activated or stopped when the CPU is switched off by executing an HALT instruction. When the LIRC oscillator is selected to be the low speed oscillator, the LIRC oscillator will also be controlled by this bit together with the WDT function enable control bit. When this bit is cleared to 0, but the WDT function is enabled, the LIRC oscillator will be enabled.

HXTC Register

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

Bit 7~2 Unimplemented, read as "0"

Bit 1 **HXTF**: HXT clock stable flag
0: Unstable
1: Stable
This bit is used to indicate whether the HXT oscillator is stable or not. When bit HXTEN is set high to enable the HXT oscillator, this HXTF bit will be cleared to“0” and then will be set high after the HXT clock is stable.

Bit 0 **HXTEN**: HXT oscillator enable control.
0: Disable
1: Enable

HIRCC Register

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

Bit 7~2 Unimplemented, read as "0"

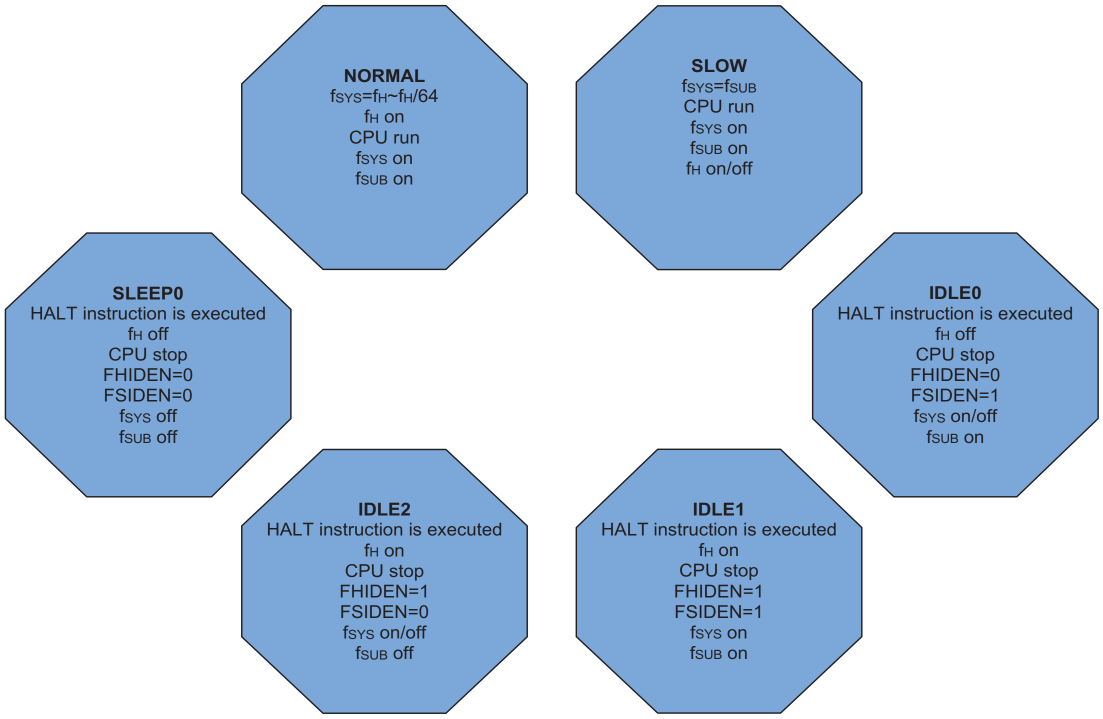
Bit 1 **HIRCF**: HIRC clock stable flag
0: Unstable
1: Stable
This bit is used to indicate whether the HIRC oscillator is stable. When bit HIRCEN is set high to enable the HIRC oscillator, this bit will be cleared to“0” and then will be set high after HIRC clock is stable. The HIRC stable time will spend 16 clocks when bit HIRCEN is enabled.

Bit 0 **HIRCEN**: HIRC oscillator enable control.
0: Disable
1: Enable

Operating Mode Switching

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

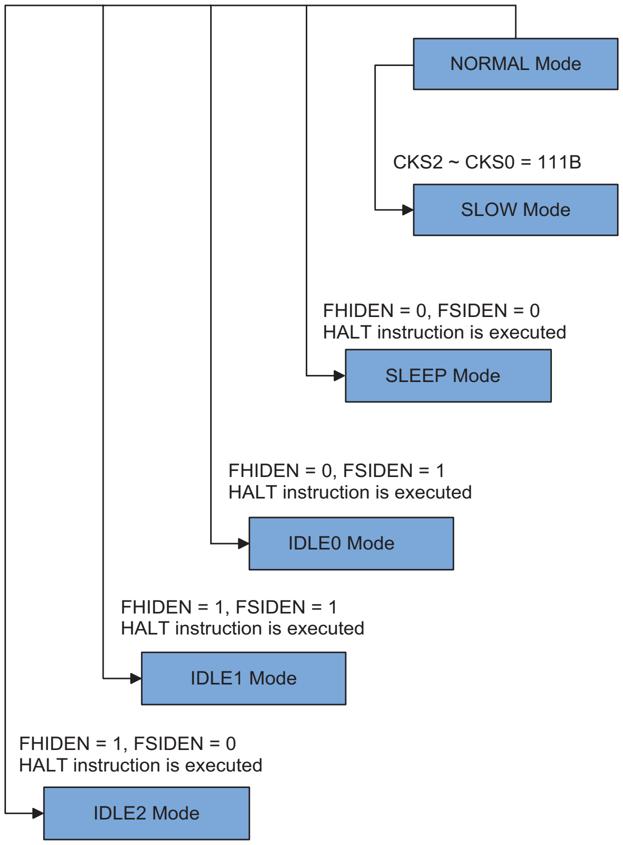
In simple terms, Mode Switching between the NORMAL Mode and SLOW Mode is executed using the CKS2~CKS0 bits in the SCC register while Mode Switching from the NORMAL/SLOW Modes to the IDLE Modes is executed via the HALT instruction. When a HALT instruction is executed, whether the device enters the IDLE0 Mode, IDLE1 Mode, IDLE2 Mode or the SLEEP Mode is determined by the condition of the FHIDEN and FSIDEN bits in the SCC register.



NORMAL Mode to SLOW Mode Switching

When running in the NORMAL Mode, which uses the high speed system oscillator, and therefore consumes more power, the system clock can switch to run in the SLOW Mode by set the CKS2~CKS0 bits to “111”in the SCC register. This will then use the low speed system oscillator which will consume less power. Users may decide to do this for certain operations which do not require high performance and can subsequently reduce power consumption.

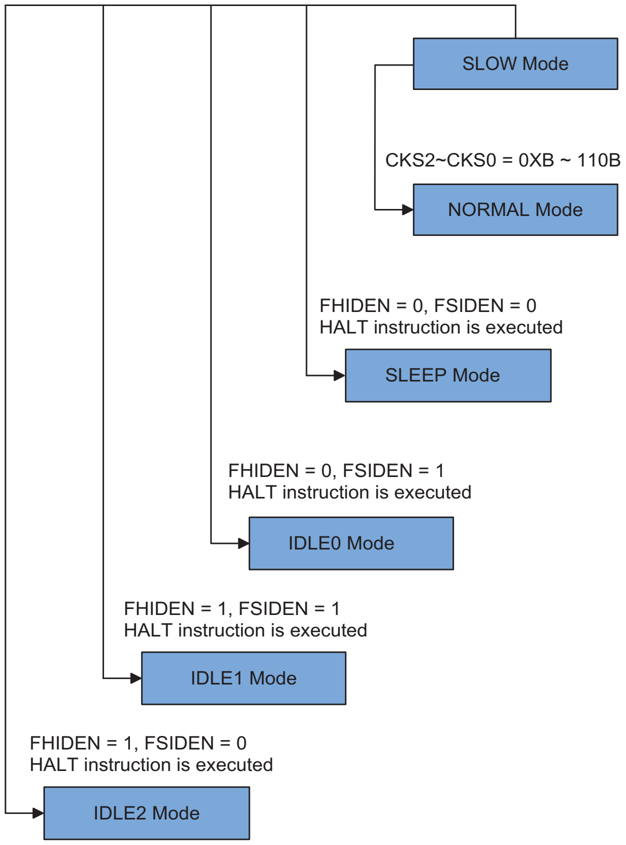
The SLOW Mode is sourced from the LIRC oscillator and therefore requires the oscillator to be stable before full mode switching occurs.



SLOW Mode to NORMAL Mode Switching

In SLOW mode the system clock is from f_{SUB}. When system clock is switched back to the NORMAL mode from f_{SUB}, the CKS [2:0] bits should be set to "000"~"110", and then the system clock will be switched to f_H~f_H/64.

However, if the f_H clock is not used in the SLOW mode, it will take some time to re-oscillate and stabilize. This is monitored using the HXTF/HIRCF bit in the HXTC/HIRCC register. The amount of time required for high speed system oscillator stabilization depends upon which high speed system oscillator type is used.



Entering the SLEEP Mode

There is only one way for the device to enter the SLEEP Mode and that is to execute the "HALT" instruction in the application program with the FHIDEN and FLIDEN bit in SCC register equal to "0". When this instruction is executed under the conditions described above, the following will occur:

- The system clock will be stopped and the application program will stop at the "HALT" instruction.
- The Data Memory contents and registers will maintain their present condition.
- 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.
- The WDT will be cleared and resume counting if the WDT function is enabled.

Entering the IDLE0 Mode

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

- The f_H clock will be off and the f_{SUB} clock will be on and the application program will stop at the "HALT" instruction.
- The Data Memory contents and registers will maintain their present condition.
- The 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.
- The WDT will be cleared and resume counting if the WDT function is enabled.

Entering the IDLE1 Mode

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

- The f_H and the f_{SUB} clock will be on and the application program will stop at the "HALT" instruction.
- The Data Memory contents and registers will maintain their present condition.
- The 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.
- The WDT will be cleared and resume counting if the WDT function is enabled.

Entering the IDLE2 Mode

There is only one way for the device to enter the IDLE2 Mode and that is to execute the “HALT” instruction in the application program with the FHIDEN bit in SCC register equal to “1” and the FSIDEN bit in SCC register equal to “0”. When this instruction is executed under the conditions described above, the following will occur:

- The f_H clock will be on while the f_{SUB} clock will be off and the application program will stop at the "HALT" instruction.
- The Data Memory contents and registers will maintain their present condition.
- 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.
- The WDT will be cleared and resume counting if the WDT function is enabled.

Standby Current Considerations

As the main reason for entering the SLEEP or IDLE Mode is to keep the current consumption of the device to as low a value as possible, perhaps only in the order of several micro-amps except in the IDLE1 and IDLE2 Mode, there are other considerations which must also be taken into account by the circuit designer if the power consumption is to be minimised. Special attention must be made to the I/O pins on the device. All high-impedance input pins must be connected to either a fixed high or low level as any floating input pins could create internal oscillations and result in increased current consumption. This also applies to devices which have different package types, as there may be unbonded pins. These must either be setup as outputs or if setup as inputs must have pull-high resistors connected.

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

In the IDLE1 and IDLE2 Mode the high speed oscillator is on, if the system oscillator is from the high speed oscillator, the additional standby current will also be perhaps in the order of several hundred micro-amps.

Wake-up

To minimise power consumption the device can enter the SLEEP or IDLE0~IDLE2 Mode, where the CPU will be switched off. However, when the device is woken up again, it can take a considerable time for the original system oscillator to restart, stabilise and allow normal operation to resume.

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

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

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

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

Watchdog Timer

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

Watchdog Timer Clock Source

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

Watchdog Timer Control Register

A single register, WDTC, controls the required timeout period as well as the WDT software reset operation as the WDT function is always enabled.

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
01010/10101: Enable
Others: Reset MCU
When these bits are changed by the environmental noise to reset the microcontroller, the reset operation will be activated after 2~3 LIRC clock cycles and the WRF bit in the RSTFC register will be set to 1.

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

Watchdog Timer Operation

The Watchdog Timer operates by providing a device reset when its timer overflows. This means that in the application program and during normal operation the user has to strategically clear the Watchdog Timer before it overflows to prevent the Watchdog Timer from executing a reset. This is done using the clear watchdog instructions. If the program malfunctions for whatever reason, jumps to an unknown location, or enters an endless loop, these clear instructions will not be executed in the correct manner, in which case the Watchdog Timer will overflow and reset the device. With regard to the Watchdog Timer enable/disable function, there are five bits, WE4~WE0, in the WDTC register to offer enable/disable control and reset control of the Watchdog Timer. The WDT function will be enabled if the WE4~WE0 bits value is equal to 01010B or 10101B. If the WE4~WE0 bits are set to any other values by the environmental noise or software setting, except 01010B and 10101B, it will reset the device after 2~3 f_{LIRC} clock cycles. After power on these bits will have the value of 01010B.

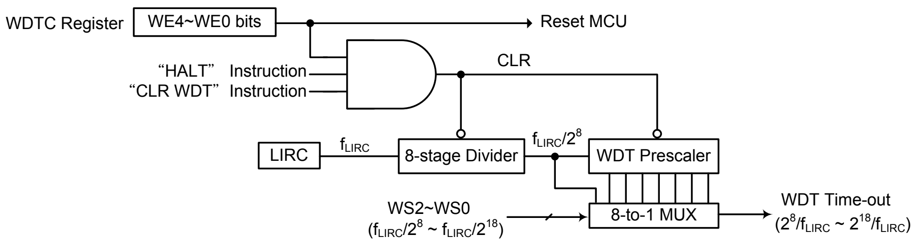
WE4~WE0 Bits	WDT Function
01010B/10101B	Enable
Any other value	Reset MCU

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

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

The maximum time out period is when the 2^{18} division ratio is selected. As an example, with a 32kHz LIRC oscillator as its source clock, this will give a maximum watchdog period of around 8 second 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 device can be set to some predetermined condition irrespective of outside parameters. The most important reset condition is after power is first applied to the microcontroller. In this case, internal circuitry will ensure that the microcontroller, after a short delay, will be in a well defined state and ready to execute the first program instruction. After this power-on reset, certain important internal registers will be set to defined states before the program commences. One of these registers is the Program Counter, which will be reset to zero forcing the microcontroller to begin program execution from the lowest Program Memory address.

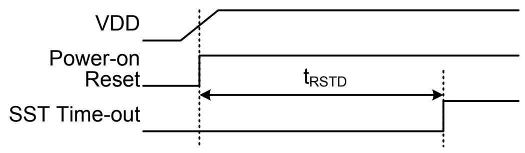
Another type of reset is when the Watchdog Timer overflows and resets. All types of reset operations result in different register conditions being setup. Another reset exists in the form of a Low Voltage Reset, LVR, where a full reset is implemented in situations where the power supply voltage falls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

Reset Functions

There are several ways in which a 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 I/O ports will be first set to inputs.

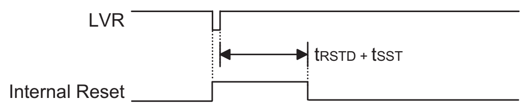


Note: t_{RSTD} is power-on delay, typical time=50ms

Power-On Reset Timing Chart

Low Voltage Reset – LVR

The microcontroller contains a low voltage reset circuit in order to monitor the supply voltage of the device. The LVR function is always enabled with a fixed LVR voltage of 3.8V. If the supply voltage of the device drops to within a range of 0.9V~V_{LVR} such as might occur when changing the battery, the LVR will automatically reset the device internally and the LVRF bit in the RSTFC register will also be set to 1. For a valid LVR signal, a low supply voltage, i.e., a voltage in the range between 0.9V~V_{LVR} must exist for a time greater than that specified by t_{LVR} in the 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. Note that the LVR function will be automatically disabled when the device enters the power down mode.



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

Low Voltage Reset Timing Chart

• **RSTFC Register**

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

"x" unknown

Bit 7~4 Unimplemented, read as "0"

Bit 3 **ERSTF**: Reset caused by RSTF [7:0] setting
0: Not occur
1: Occurred
This bit is set to 1 if the RSTC register contains any undefined RSTC values. This bit can only be cleared to 0 by application program.

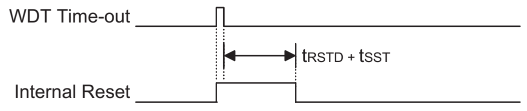
Bit 2 **LVRF**: LVR function reset flag
0: Not occur
1: Occurred
This bit is set to 1 when a low voltage reset situation occurs. This bit can only be cleared to 0 by application program.

Bit 1 Unimplemented, read as "0"

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

Watchdog Time-out Reset during Normal Operation

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

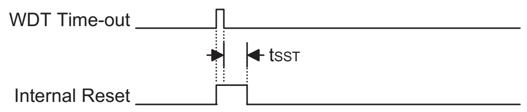


Note: t_{RSTD} is power-on delay, typical time=16.7ms

WDT Time-out Reset during Normal Operation Timing Chart

Watchdog Time-out Reset during SLEEP or IDLE Mode

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



Note: The t_{SST} is 15~16 clock cycles if the system clock source is provided by HIRC.
The t_{SST} is 128 clock for HXT. The t_{SST} is 1~2 clock for LIRC.

WDT Time-out Reset during Sleep or IDLE Timing Chart

Reset Initial Conditions

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

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

Note: "u" stands for unchanged

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

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

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

Register	Power On Reset	LVR Reset	WDT Time-out (Normal Operation)	WDT Time-out (HALT)
MP0	0000 0000	0000 0000	0000 0000	uuuu uuuu
MP1L	0000 0000	0000 0000	0000 0000	uuuu uuuu
MP1H	0000 0000	0000 0000	0000 0000	uuuu uuuu
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
PCL	0000 0000	0000 0000	0000 0000	0000 0000
TBLP	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLH	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBHP	---- xxxx	---- uuuu	---- uuuu	---- uuuu
STATUS	xx00 xxxx	xx0u uuuu	xx1u uuuu	uu11 uuuu
MP2L	0000 0000	0000 0000	0000 0000	uuuu uuuu
MP2H	0000 0000	0000 0000	0000 0000	uuuu uuuu
INTC0	-000 0000	-000 0000	-000 0000	-uuu uuuu
PA	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAC	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAPU	0000 0000	0000 0000	0000 0000	uuuu uuuu
PAWU	0000 0000	0000 0000	0000 0000	uuuu uuuu
RSTFC	---- 0x-0	---- uu-u	---- uu-u	---- uu-u
PB	1111 1111	1111 1111	1111 1111	uuuu uuuu
PBC	1111 1111	1111 1111	1111 1111	uuuu uuuu

Register	Power On Reset	LVR Reset	WDT Time-out (Normal Operation)	WDT Time-out (HALT)
PBPU	0000 0000	0000 0000	0000 0000	uuuu uuuu
PC	1111 1111	1111 1111	1111 1111	uuuu uuuu
PCC	1111 1111	1111 1111	1111 1111	uuuu uuuu
PCPU	0000 0000	0000 0000	0000 0000	uuuu uuuu
TBC0	0--- -000	0--- -000	0--- -000	u--- -uuu
TBC1	0--- -000	0--- -000	0--- -000	u--- -uuu
PSCR	---- --00	---- --00	---- --00	---- --uu
WDTC	0101 0011	0101 0011	0101 0011	uuuu uuuu
LVDC	--00 0000	--00 0000	--00 0000	--uu uuuu
RSTC	0101 0101	0101 0101	0101 0101	uuuu uuuu
ADRL (ADRFS=0)	xxxx ----	xxxx ----	xxxx ----	uuuu ----
ADRL (ADRFS=1)	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
ADRH (ADRFS=0)	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
ADRH (ADRFS=1)	---- xxxx	---- xxxx	---- xxxx	---- uuuu
ADCR0	0100 0000	0100 0000	0100 0000	uuuu uuuu
ADCR1	0000 0000	0000 0000	0000 0000	uuuu uuuu
SCC	000- 0-00	000- 0-00	000- 0-00	uuu- u-uu
HXTC	---- --00	---- --00	---- --00	---- --00
HIRCC	---- --01	---- --01	---- --01	---- --01
INTEG	---- 0000	---- 0000	---- 0000	---- 0000
INTC1	0000 0000	0000 0000	0000 0000	uuuu uuuu
INTC2	0000 0000	0000 0000	0000 0000	uuuu uuuu
MFI0	0000 0000	0000 0000	0000 0000	uuuu uuuu
MFI1	--00 --00	--00 --00	--00 --00	--uu --uu
MFI2	0000 0000	0000 0000	0000 0000	uuuu uuuu
PAS0	0000 0000	0000 0000	0000 0000	uuuu uuuu
PAS1	0000 0000	0000 0000	0000 0000	uuuu uuuu
PBS0	00-0 0000	00-0 0000	00-0 0000	uu-u uuuu
PCS0	---- 0000	---- 0000	---- 0000	---- uuuu
EEA	--00 0000	--00 0000	--00 0000	--uu uuuu
EED	0000 0000	0000 0000	0000 0000	uuuu uuuu
EEC	---- 0000	---- 0000	---- 0000	---- uuuu
TM0C0	0000 0000	0000 0000	0000 0000	uuuu uuuu
TM0C1	0000 0000	0000 0000	0000 0000	uuuu uuuu
TM0DL	0000 0000	0000 0000	0000 0000	uuuu uuuu
TM0DH	---- --00	---- --00	---- --00	---- --uu
TM0AL	0000 0000	0000 0000	0000 0000	uuuu uuuu
TM0AH	---- --00	---- --00	---- --00	---- --uu
TM1C0	0000 0000	0000 0000	0000 0000	uuuu uuuu
TM1C1	0000 0000	0000 0000	0000 0000	uuuu uuuu
TM1DL	0000 0000	0000 0000	0000 0000	uuuu uuuu
TM1DH	---- --00	---- --00	---- --00	---- --uu
TM1AL	0000 0000	0000 0000	0000 0000	uuuu uuuu
TM1AH	---- --00	---- --00	---- --00	---- --uu
TM2C0	0000 0---	0000 0---	0000 0---	uuuu u---
TM2C1	0000 0000	0000 0000	0000 0000	uuuu uuuu
TM2DL	0000 0000	0000 0000	0000 0000	uuuu uuuu

Register	Power On Reset	LVR Reset	WDT Time-out (Normal Operation)	WDT Time-out (HALT)
TM2DH	0000 0000	0000 0000	0000 0000	uuuu uuuu
TM2AL	0000 0000	0000 0000	0000 0000	uuuu uuuu
TM2AH	0000 0000	0000 0000	0000 0000	uuuu uuuu
TM2RP	0000 0000	0000 0000	0000 0000	uuuu uuuu
USR	0000 1011	0000 1011	0000 1011	uuuu uuuu
UCR1	0000 00x0	0000 00x0	0000 00x0	uuuu uuuu
UCR2	0000 0000	0000 0000	0000 0000	uuuu uuuu
BRG	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TXR_RXR	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
SPWMC0	1000 -100	1000 -100	1000 -100	uuuu -uuu
SPWMC1	111- 0000	111- 0000	111- 0000	uuu- uuuu
SPWML	0000 0000	0000 0000	0000 0000	uuuu uuuu
SPWMH	---- 0000	---- 0000	---- 0000	---- uuuu
SFIFOL	0000 0000	0000 0000	0000 0000	uuuu uuuu
SFIFOH	0--- 0000	0--- 0000	0--- 0000	u--- uuuu
MTF	0000 0000	0000 0000	0000 0000	uuuu uuuu
SPWMDT	---- 0000	---- 0000	---- 0000	---- uuuu
ACDC	0000 0--0	0000 0--0	0000 0--0	uuuu u--u
ACDOFF	0000 0000	0000 0000	0000 0000	uuuu uuuu
OCP0C0	00-- ----	00-- ----	00-- ----	uu-- ----
OCP0C1	0-00 0000	0-00 0000	0-00 0000	0-00 0000
OCP0REF	0000 0000	0000 0000	0000 0000	uuuu uuuu
OCP0ACAL	0010 0000	0010 0000	0010 0000	uuuu uuuu
OCP0CCAL	0001 0000	0001 0000	0001 0000	uuuu uuuu
OCP1C0	00-- ----	00-- ----	00-- ----	uu-- ----
OCP1C1	0-00 0000	0-00 0000	0-00 0000	0-00 0000
OCP1REF	0000 0000	0000 0000	0000 0000	uuuu uuuu
OCP1ACAL	0010 0000	0010 0000	0010 0000	uuuu uuuu
OCP1CCAL	0001 0000	0001 0000	0001 0000	uuuu uuuu
OVPC	0--0 -000	0--0 -000	0--0 -000	u--u -uuu
OVPREF	0000 0000	0000 0000	0000 0000	uuuu uuuu
OVPCCAL	0001 0000	0001 0000	0001 0000	uuuu uuuu

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

Input/Output Ports

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

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

Register Name	Bit							
	7	6	5	4	3	2	1	0
PA	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0
PAC	PAC7	PAC6	PAC5	PAC4	PAC3	PAC2	PAC1	PAC0
PAPU	PAPU7	PAPU6	PAPU5	PAPU4	PAPU3	PAPU2	PAPU1	PAPU0
PAWU	PAWU7	PAWU6	PAWU5	PAWU4	PAWU3	PAWU2	PAWU1	PAWU0
PB	PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0
PBC	PBC7	PBC6	PBC5	PBC4	PBC3	PBC2	PBC1	PBC0
PBPU	PBPU7	PBPU6	PBPU5	PBPU4	PBPU3	PBPU2	PBPU1	PBPU0
PC	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0
PCC	PCC7	PCC6	PCC5	PCC4	PCC3	PCC2	PCC1	PCC0
PCPU	PCPU7	PCPU6	PCPU5	PCPU4	PCPU3	PCPU2	PCPU1	PCPU0

I/O Register List

"—": Unimplemented, read as "0"

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

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

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

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

Pull-high Resistors

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

Port A Wake-up

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

I/O Port Control Registers

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

Pin-shared Functions

The flexibility of the microcontroller range is greatly enhanced by the use of pins that have more than one function. Limited numbers of pins can force serious design constraints on designers but by supplying pins with multi-functions, many of these difficulties can be overcome. For these pins, the chosen function of the multi-function I/O pins is selected by a series of registers via the application program control.

Register Name	Bit							
	7	6	5	4	3	2	1	0
PAS0	PAS07	PAS06	PAS05	PAS04	PAS03	PAS02	PAS01	PAS00
PAS1	PAS17	PAS16	PAS15	PAS14	PAS13	PAS12	PAS11	PAS10
PBS0	PBS07	PBS06	—	PBS04	PBS03	PBS02	PBS01	PBS00
PCS0	—	—	—	—	PCS03	PCS02	PCS01	PCS00
IFS0	—	—	—	—	—	—	—	IFS00

Pin-shared Function Selection Register List

PAS0 Register

Bit	7	6	5	4	3	2	1	0
Name	PAS07	PAS06	PAS05	PAS04	PAS03	PAS02	PAS01	PAS00
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 **PAS07~PAS06**: PortA.3 function selection
00: PA3
01: PA3
10: OCP0
11: AN3

Bit 5~4 **PAS05~PAS04**: PortA.2 function selection
00: PA2
01: PA2
10: PA2
11: AN2

Bit 3~2 **PAS03~PAS02**: PortA.1 function selection
00: PA1
01: PA1
10: VREF
11: AN1

Bit 1~0 **PAS01~PAS00**: PortA.0 function selection
00: PA0
01: PA0
10: PA0
11: AN0

PAS1 Register

Bit	7	6	5	4	3	2	1	0
Name	PAS17	PAS16	PAS15	PAS14	PAS13	PAS12	PAS11	PAS10
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 **PAS17~PAS16**: PortA.7 function selection
00: PA7
01: TX
10: PA7
11: AN7

Bit 5~4 **PAS15~PAS14**: PortA.6 function selection
00: PA6
01: RX
10: PA6
11: AN6

Bit 3~2 **PAS13~PAS12**: PortA.5 function selection
00: PA5
01: PA5
10: OVP
11: AN5

Bit 1~0 **PAS11~PAS10**: PortA.4 function selection
00: PA4
01: PA4
10: OCP1
11: AN4

PBS0 Register

Bit	7	6	5	4	3	2	1	0
Name	PBS07	PBS06	—	PBS04	PBS03	PBS02	PBS01	PBS00
R/W	R/W	R/W	—	R/W	R/W	R/W	R/W	R/W
POR	0	0	—	0	0	0	0	0

Bit 7 **PBS07**: PortB.7 function selection
0: PB7
1: AN9

Bit 6 **PBS06**: PortB.6 function selection
0: PB6
1: AN8

Bit 5 Unimplemented, read as "0"

Bit 4 **PBS04**: PortB.4 function selection
0: PB4/TP2_I (TP2_I is for TM2 capture input)
1: TP2_O (TP2_O is for TM2 capture output)

Bit 3 **PBS03**: PortB.3 function selection
0: PB3
1: TP1

Bit 2 **PBS02**: PortB.2 function selection
0: PB2
1: TP0

Bit 1 **PBS01**: PortB.1 function selection
0: PB1
1: BBO

Bit 0 **PBS00**: PortB.0 function selection
0: PB0
1: BUO

PCS0 Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	PCS03	PCS02	PCS01	PCS00
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0

Bit 7~4 Unimplemented, read as "0"

Bit 3 **PCS03**: PortC.3 function selection
0: PC3
1: AC+

Bit 2 **PCS02**: PortC.2 function selection
0: PC2
1: AC-

Bit 1 **PCS01**: PortC.1 function selection
0: PC1
1: OSC1

Bit 0 **PCS00**: PortC.0 function selection
0: PC0
1: OSC2

IFS0 Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	—	IFS00
R/W	—	—	—	—	—	—	—	R/W
POR	—	—	—	—	—	—	—	0

Bit 7~1 Unimplemented, read as "0"

Bit 0 **IFS00**: TM2 capture input source selection
0: TP2 pin
1: ACDO

RSTC Register

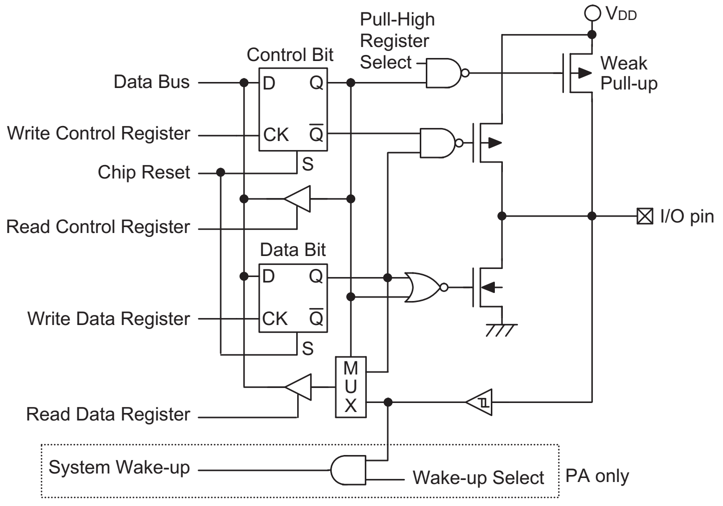
Bit	7	6	5	4	3	2	1	0
Name	RST7	RST6	RST5	RST4	RST3	RST2	RST1	RST0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	1	0	1	0	1	0	1

Bit 7~0 **RST7~RST0**: PortA.2 configuration
01010101B/10101010B: configured as GPIO, PA2
Other values: MCU reset

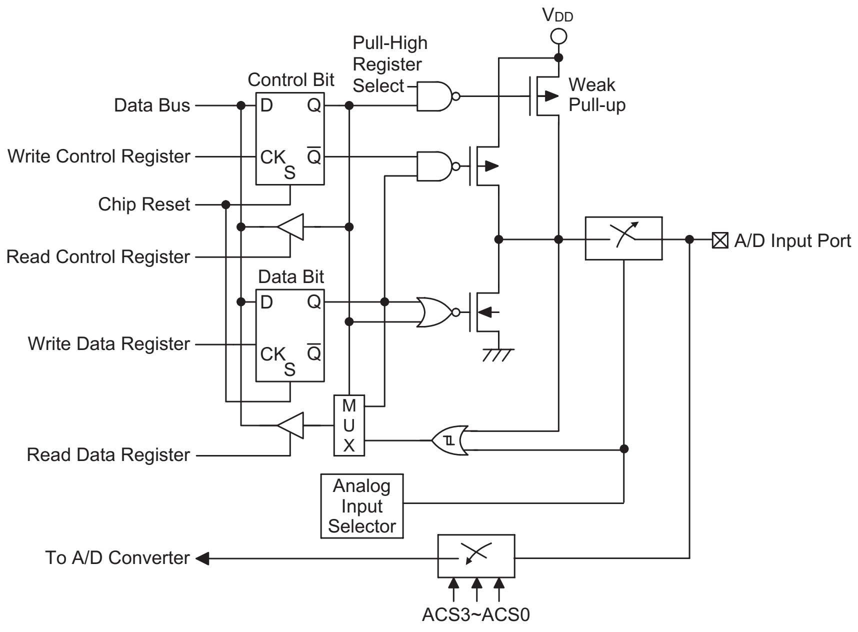
If these bits are changed to any other values except 01010101B and 10101010B, the microcontroller will be reset. The reset operation will be activated after 2~3 LIRC clock cycles and the ERSTF flag will be set.

I/O Pin Structures

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



Generic Input/Output Structure



A/D Input/Output Structure

Programming Considerations

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

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

Timer Modules – TM

One of the most fundamental functions in any microcontroller device is the ability to control and measure time. To implement time related functions 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

These devices contain three TMs having a reference name of TM0, TM1 and TM2. Each individual TM can be categorised as a certain type, namely Compact Type TM or Standard Type TM. Although similar in nature, the different TM types vary in their feature complexity. The common features to all of the Compact and Standard TMs will be described in this section, 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	STM
Timer/Counter	√	√
I/P Capture	—	√
Compare Match Output	√	√
PWM Channels	1	1
Single Pulse Output	—	1
PWM Alignment	Edge	Edge
PWM Adjustment Period & Duty	Duty or Period	Duty or Period

TM Function Summary

This chip contains a specific number of either Compact Type and Standard Type TM units which are shown in the table together with their individual reference names, TM0~TM2.

TM0	TM1	TM2
10-bit CTM	10-bit CTM	16-bit STM

TM Name/Type Reference

TM Operation

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

TM Clock Source

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

TM Interrupts

The Compact Type and Standard Type TMs each 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.

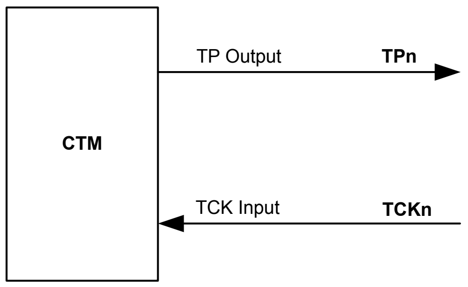
TM External Pins

Each of the TMs, irrespective of what type, 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.

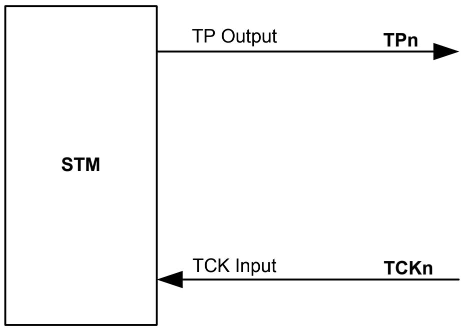
The TMs each have one output pin with the label TPn. When the TM is in the Compare Match Output Mode, these pins can be controlled by the TM to switch to a high or low level or to toggle when a compare match situation occurs. The external TPn output pin is also the pin where the TM generates the PWM output waveform. 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.

TM Input/Output Pin Control Registers

Selecting to have a TM input/output or whether to retain its other shared functions is implemented using one register with a single bit in each register corresponding to a TM input/output pin. Setting the bit high will setup the corresponding pin as a TM input/output if reset to zero the pin will retain its original other functions.



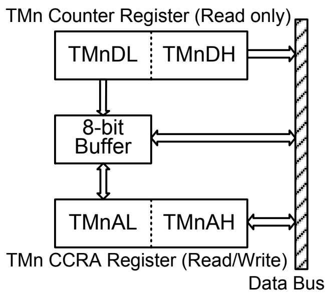
CTM Function Pin Control Block Diagram (n=0 or 1)



STM Function Pin Control Block Diagram (n=2)

Programming Considerations

The TM Counter Registers and the Capture/Compare CCRA registers, being 10-bit or 16-bit, all have a low and high byte structure. The high bytes can be directly accessed, but as the low bytes can only be accessed via an internal 8-bit buffer, reading or writing to these register pairs must be carried out in a specific way. The important point to note is that data transfer to and from the 8-bit buffer and its related low byte only takes place when a write or read operation to its corresponding high byte is executed. As the CCRA registers are implemented in the way shown in the following diagram and accessing the register is carried out in a specific way described above, it is recommended to use the "MOV" instruction to access the CCRA low byte registers, named TMnAL, using the following access procedures. Accessing the CCRA low byte register without following these access procedures will result in unpredictable values.


TMn Counter Register (Read only)
TMnDL
TMnDH
8-bit Buffer
TMnAL
TMnAH
TMn CCRA Register (Read/Write)
Data Bus

The following steps show the read and write procedures:

- Writing Data to CCRA
  - Step 1. Write data to Low Byte TMnAL
    – Note that here data is only written to the 8-bit buffer.
  - Step 2. Write data to High Byte TMnAH
    – Here data is written directly to the high byte registers and simultaneously data is latched from the 8-bit buffer to the Low Byte registers.
- Reading Data from the Counter Registers and CCRA
  - Step 1. Read data from the High Byte TMnDH or TMnAH
    – Here data is read directly from the High Byte registers and simultaneously data is latched from the Low Byte register into the 8-bit buffer.
  - Step 2. Read data from the Low Byte TMnDL or TMnAL
    – This step reads data from the 8-bit buffer.

Compact Type TM – CTM

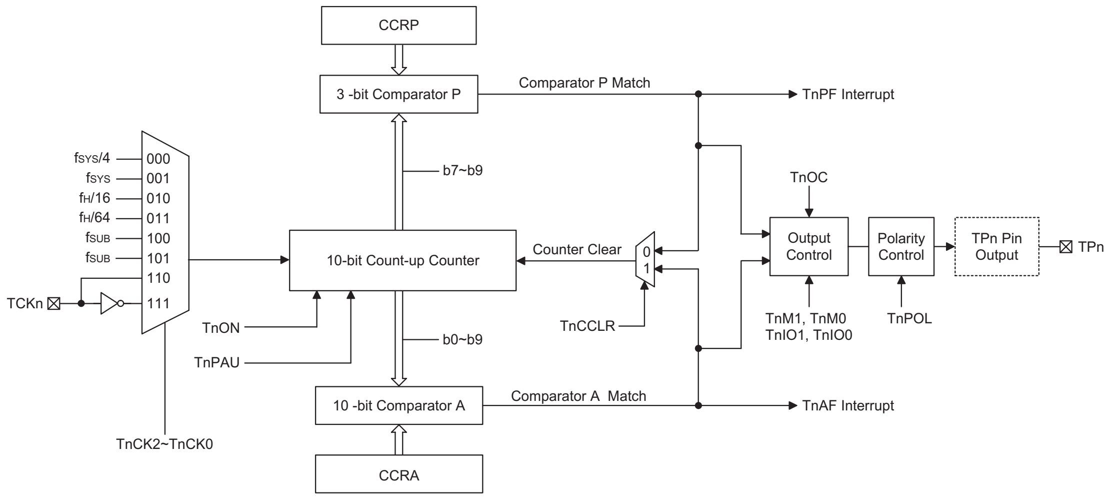
Although the simplest form of the three TM types, the Compact TM type still contains three operating modes, which are Compare Match Output, Timer/Event Counter and PWM Output modes. The Compact TM can also be controlled with an external input pin and can drive one external output pin.

Name	TM No.	TM Input Pin	TM Output Pin
10-bit CTM	0/1	TCK0/TCK1	TP0/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 (n=0, 1)

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 Registers List (n=0, 1)

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: $f_H/16$
011: $f_H/64$
100: f_{SUB}
101: f_{SUB}
110: TCKn rising edge clock
111: TCKn falling edge clock
These three bits are used to select the clock source for the TM. The external pin clock source can be chosen to be active on the rising or falling edge. The clock source f_{SYS} is the system clock, while f_H and f_{SUB} are other internal clocks, the details of which can be found in the oscillator section.

Bit 3 **TnON**: TMn Counter On/Off Control
0: Off
1: On
This bit controls the overall on/off function of the 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
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 pin state is undefined.

Bit 5~4 **TnIO1, TnIO0**: Select TPn output function

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

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

Timer/Counter Mode
Unused

These two bits are used to determine how the 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 TnON bit from low to high.

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

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

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 TPn 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 Comparatror P match
1: TMn Comparatror A match

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

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

TMnDH Register

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

Bit 7~2 Unimplemented, read as "0"

Bit 1~0 **D9~D8**: 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 **D7~D0**: TMn CCRA Low Byte Register bit 7~bit 0
TMn 10-bit CCRA bit 7~bit 0

TMnAH Register

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

Bit 7~2 Unimplemented, read as "0"

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

Compact Type TM Operating Modes

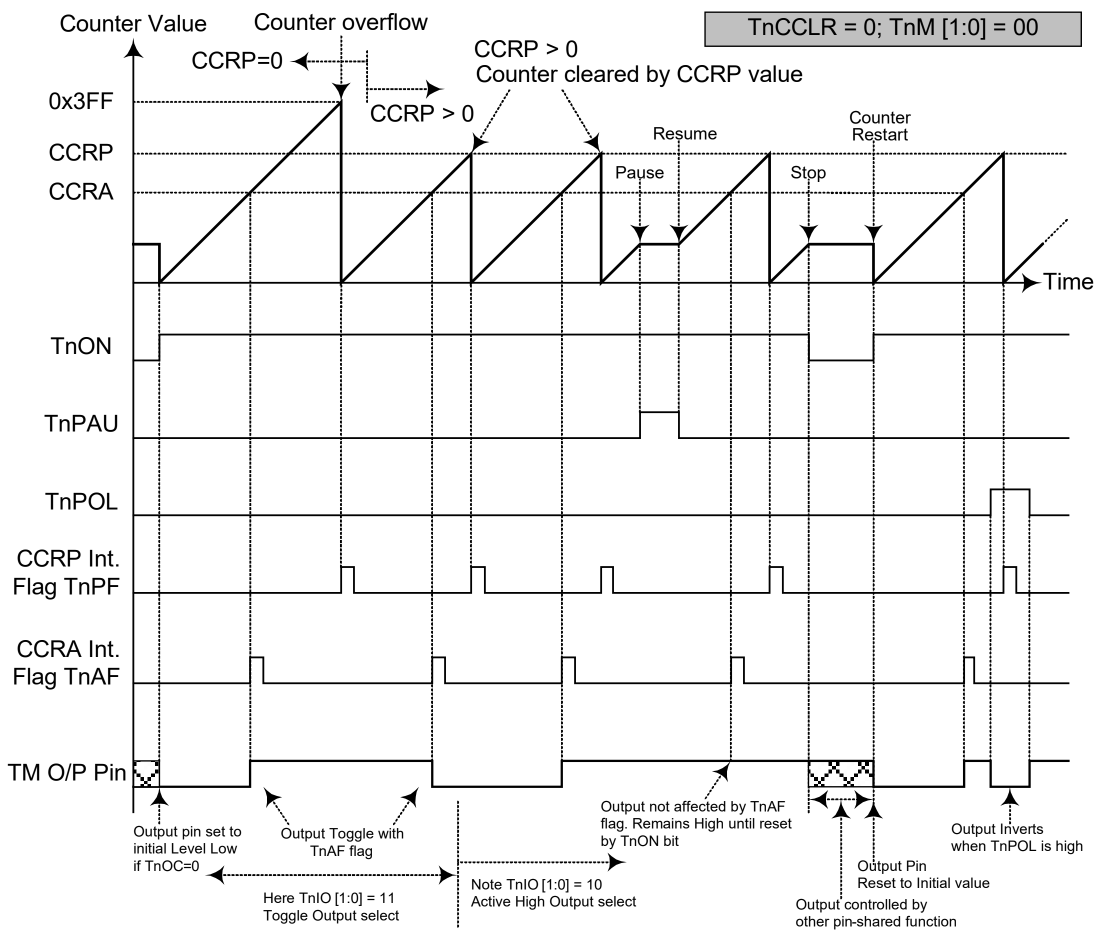
The Compact Type TM can operate in one of three operating modes, Compare Match Output Mode, PWM Mode or Timer/Counter Mode. The operating mode is selected using the TnM1 and TnM0 bits in the TMnC1 register.

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.

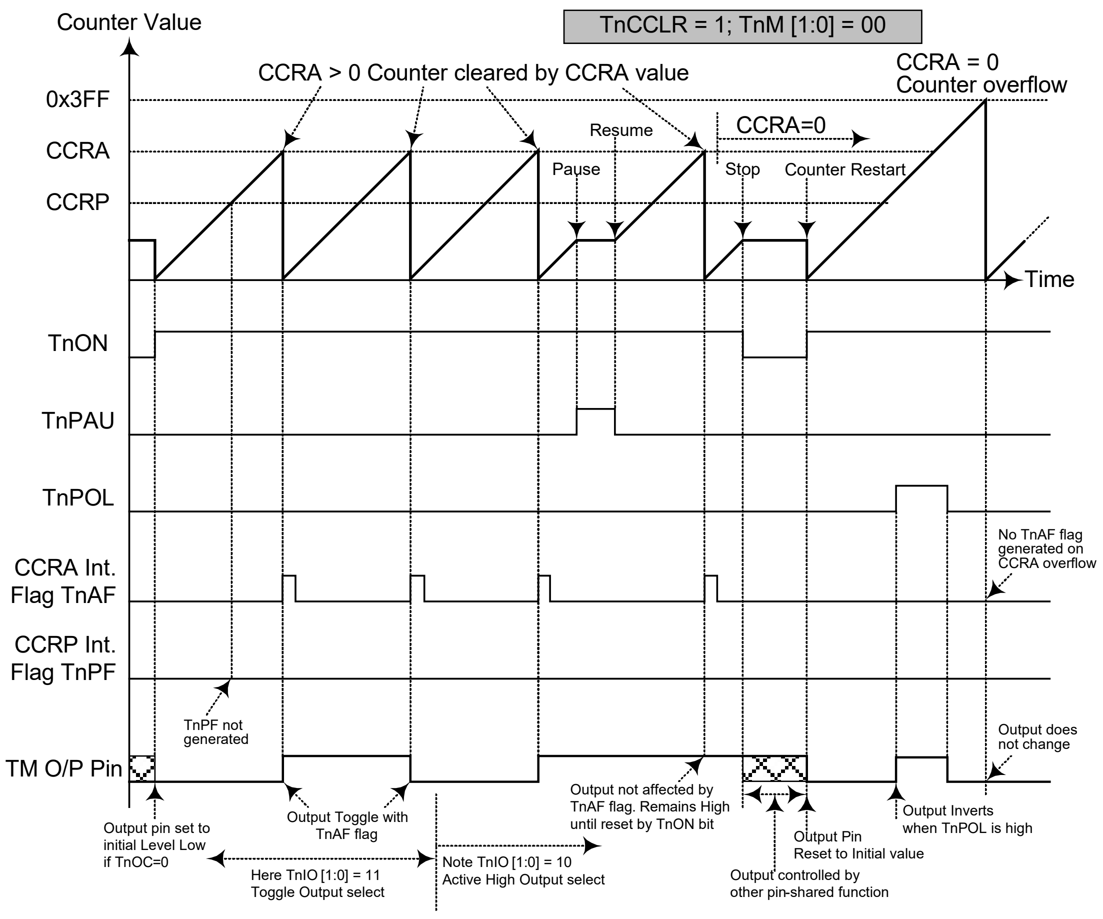
If the TnCCLR bit in the TMnC1 register is high then the counter will be cleared when a compare match occurs from Comparator A. However, here only the TnAF interrupt request flag will be generated even if the value of the CCRP bits is less than that of the CCRA registers. Therefore when TnCCLR is high no TnPF interrupt request flag will be generated. If the CCRA bits are all zero, the counter will overflow when it 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.


TnCCLR = 0; TnM [1:0] = 00
Counter Value
Counter overflow
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 (n=0, 1)

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
CCRP Int. Flag TnPF
No TnAF flag generated on CCRA overflow
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 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 Pin Reset to Initial value
Output controlled by other pin-shared function
Output Inverts when TnPOL is high

Compare Match Output Mode – TnCCLR = 1 (n=0, 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. As the TM output pin is not used in this mode, the pin can be used as a normal I/O pin or other pin-shared function.

PWM Output Mode

To select this mode, bits TnM1 and TnM0 in the TMnC1 register should be set to 10 respectively. 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} = 16 MHz, TM clock source is $f_{SYS}/4$, CCRP = 100b and CCRA = 128,

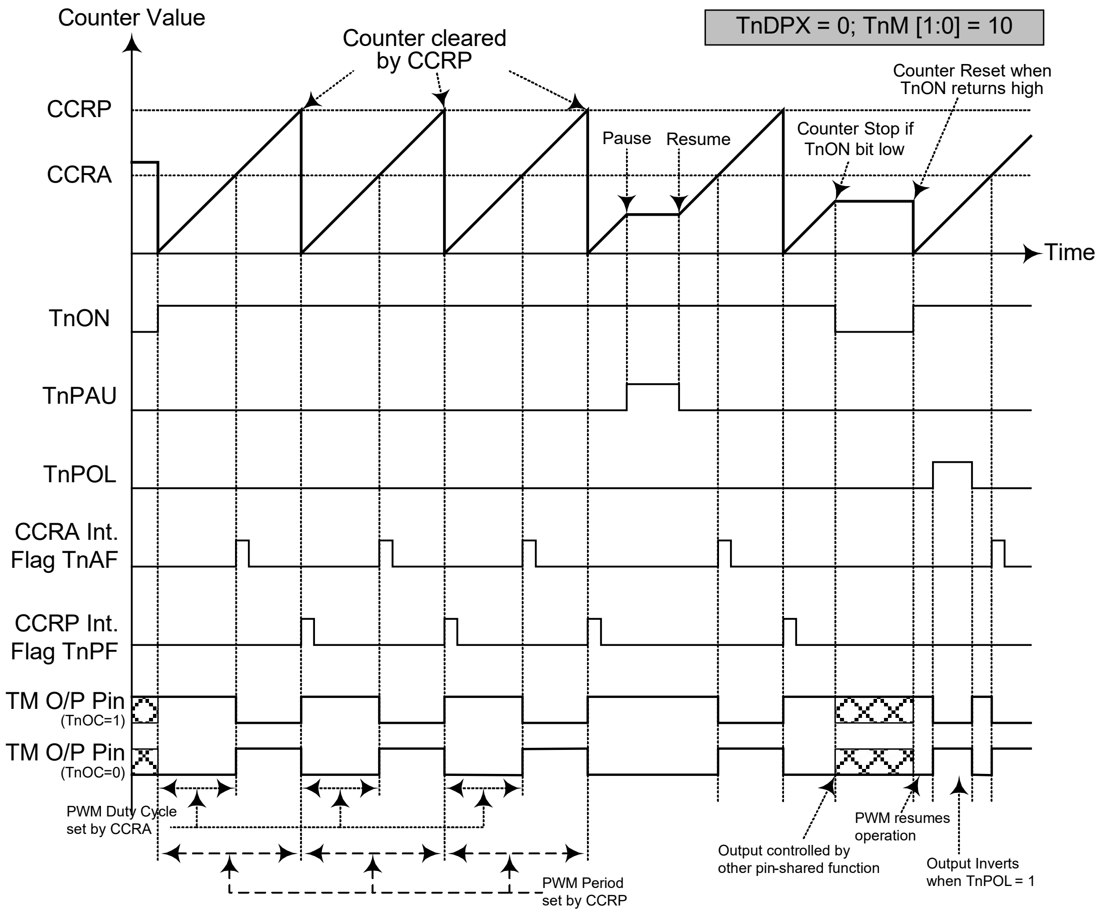
The CTM PWM output frequency = $(f_{SYS}/4)/512 = f_{SYS}/2048$ = 7.8125kHz, duty = 128/512 = 25%.

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

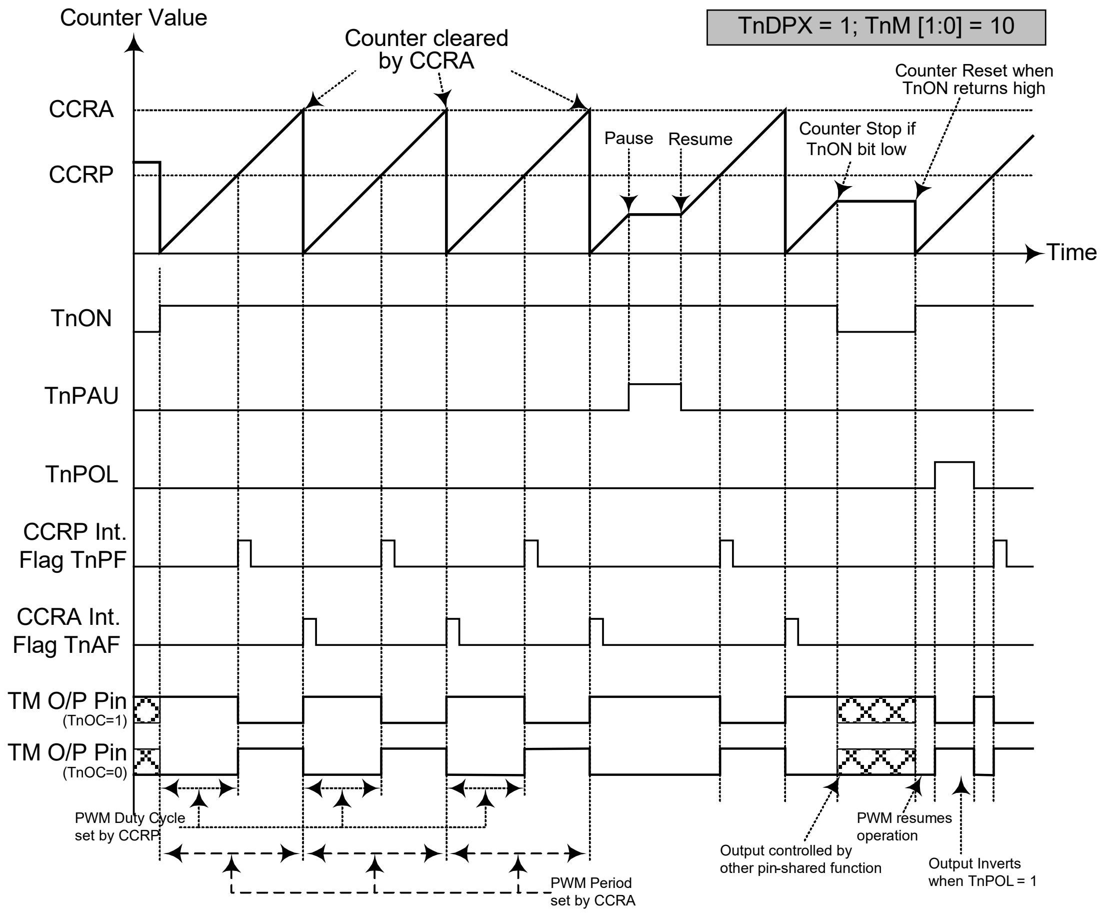
CCRP	001b	010b	011b	100b	101b	110b	111b	000b
Period	CCRA							
Duty	128	256	384	512	640	768	896	1024

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.



PWM Mode – TnDPX = 0 (n=0, 1)

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



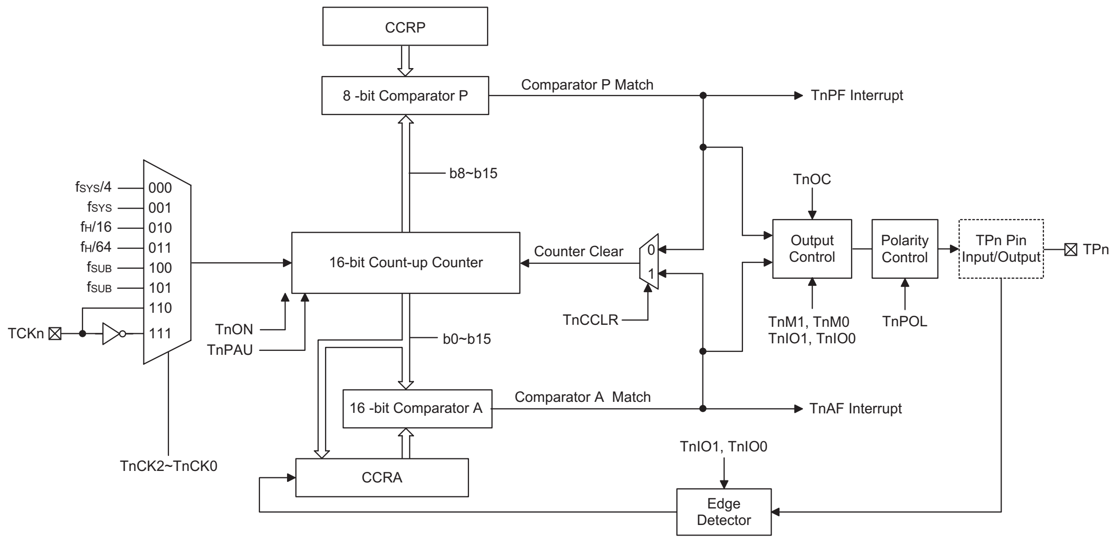
PWM Mode – TnDPX = 1 (n=0, 1)

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

Standard Type TM – STM

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

TM Type	TM Name	TM Input Pin	TM Output Pin
16-bit STM	TM2	TCK2	TP2



Standard Type TM Block Diagram (n=2)

Standard TM Operation

At its core is a 16-bit count-up counter which is driven by a user selectable internal 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 8-bit wide whose value is compared with the highest 8 bits in the counter while the CCRA is 16 bits and therefore compares with all counter bits.

The only way of changing the value of the 16-bit counter using the application program, is to clear the counter by changing the T2ON 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 Standard 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.

Standard Type TM Register Description

Overall operation of the Standard TM is controlled using series of registers. A read only register pair exists to store the internal counter 16-bit value, while a read/write register pair exists to store the internal 16-bit CCRA value. The remaining two registers are control registers which setup the different operating and control modes as well as eight CCRP bits.

Name	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TM2C0	T2PAU	T2CK2	T2CK1	T2CK0	T2ON	—	—	—
TM2C1	T2M1	T2M0	T2IO1	T2IO0	T2OC	T2POL	T2DPX	T2CCLR
TM2DL	D7	D6	D5	D4	D3	D2	D1	D0
TM2DH	D15	D14	D13	D12	D11	D10	D9	D8
TM2AL	D7	D6	D5	D4	D3	D2	D1	D0
TM2AH	D15	D14	D13	D12	D11	D10	D9	D8
TM2RP	D7	D6	D5	D4	D3	D2	D1	D0

16-bit Standard TM Register List

TM2C0 Register

Bit	7	6	5	4	3	2	1	0
Name	T2PAU	T2CK2	T2CK1	T2CK0	T2ON	—	—	—
R/W	R/W	R/W	R/W	R/W	R/W	—	—	—
POR	0	0	0	0	0	—	—	—

Bit 7 **T2PAU**: TM2 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 **T2CK2~T2CK0**: Select TM2 Counter clock
000: $f_{SYS}/4$
001: f_{SYS}
010: $f_H/16$
011: $f_H/64$
100: f_{SUB}
101: f_{SUB}
110: TCK2 rising edge clock
111: TCK2 falling edge clock
These three bits are used to select the clock source for the TM. The external pin clock source can be chosen to be active on the rising or falling edge. The clock source f_{SYS} is the system clock, while f_H and f_{SUB} are other internal clocks, the details of which can be found in the oscillator section.

Bit 3 **T2ON**: TM2 Counter On/Off Control
0: Off
1: On

This bit controls the overall on/off function of the TM. Setting the bit high enables the counter to run, clearing the bit disables the TM. Clearing this bit to zero will stop the counter from counting and turn off the TM which will reduce its power consumption. When the bit changes state from low to high the internal counter value will be reset to zero, however when the bit changes from high to low, the internal counter will retain its residual value until the bit returns high again.

If the TM is in the Compare Match Output Mode then the TM output pin will be reset to its initial condition, as specified by the T2OC bit, when the T2ON bit changes from low to high.

Bit 2~0 Unimplemented, read as "0"

TM2C1 Register

Bit	7	6	5	4	3	2	1	0
Name	T2M1	T2M0	T2IO1	T2IO0	T2OC	T2POL	T2DPX	T2CCLR
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 **T2M1~T2M0**: Select TM2 Operating Mode
00: Compare Match Output Mode
01: Capture Input Mode
10: PWM Mode or Single Pulse Output Mode
11: Timer/Counter Mode

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

Bit 5~4 **T2IO1~T2IO0**: Select TM2 output function

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

PWM Mode/Single Pulse Output Mode
00: PWM output inactive state
01: PWM output active state
10: PWM output
11: Single pulse output

Capture Input Mode
00: Input capture at rising edge of TP2
01: Input capture at falling edge of TP2
10: Input capture at falling/rising edge of TP2
11: Input capture disabled

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 T2IO1~T2IO0 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 T2IO1~T2IO0 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 T2OC bit. Note that the output level requested by the T2IO1~T2IO0 bits must be different from the initial value setup using the T2OC 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 T2ON bit from low to high.

In the PWM Mode, the T2IO1 and T2IO0 bits determine how the TM output pin changes state when a certain compare match condition occurs. The PWM output function is modified by changing these two bits. It is necessary to change the values of the T2IO1 and T2IO0 bits only after the TM has been switched off. Unpredictable PWM outputs will occur if the T2IO1 and T2IO0 bits are changed when the TM is running.

Bit 3 **T2OC**: TM2 Output control bit

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

PWM Mode/ Single Pulse Output 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/ Single Pulse Output 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 **T2POL**: TM2 Output polarity Control
0: Non-invert
1: Invert

This bit controls the polarity of the TM 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 **T2DPX**: TM2 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 **T2CCLR**: Select TM2 Counter clear condition
0: TM Comparator P match
1: TM Comparator A match

This bit is used to select the method which clears the counter. Remember that the Standard TM contains two comparators, Comparator A and Comparator P, either of which can be selected to clear the internal counter. With the T2CCLR bit set high, the counter will be cleared when a compare match occurs from the Comparator A. When the bit is low, the counter will be cleared when a compare match occurs from the Comparator P or with a counter overflow. A counter overflow clearing method can only be implemented if the CCRP bits are all cleared to zero. The T2CCLR bit is not used in the PWM, Single Pulse or Input Capture Mode.

TM2DL Register

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

Bit 7~0 **D7~D0**: TM2 Counter Low Byte Register bit 7~bit 0
TM2 16-bit Counter bit 7~bit 0

TM2DH Register

Bit	7	6	5	4	3	2	1	0
Name	D15	D14	D13	D12	D11	D10	D9	D8
R/W	R	R	R	R	R	R	R	R
POR	0	0	0	0	0	0	0	0

Bit 7~0 **D15~D8**: TM2 Counter High Byte Register bit 7~bit 0
TM2 16-bit Counter bit 15~bit 8

TM2AL Register

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

Bit 7~0 **D7~D0**: TM2 CCRA Low Byte Register bit 7~bit 0
TM2 16-bit CCRA bit 7~bit 0

TM2AH 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 **D15~D8**: TM2 CCRA High Byte Register bit 7~bit 0
TM2 16-bit CCRA bit 15~bit 8

TM2RP 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 **TM2RP**: TM2 CCRP High Byte Register bit 7~bit 0

TM2 CCRP 8-bit register, compared with the TM2 Counter bit 15~bit 8. Comparator P Match Period
0: 65536 TM2 clocks
1~255: 256 × (1~255) TM2 clocks

These eight bits are used to setup the value on the internal CCRP 8-bit register, which are then compared with the internal counter's highest eight bits. The result of this comparison can be selected to clear the internal counter if the T2CCLR bit is set to zero. Setting the T2CCLR 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 eight counter bits, the compare values exist in 256 clock cycle multiples. Clearing all eight bits to zero is in effect allowing the counter to overflow at its maximum value.

Standard Type TM Operating Modes

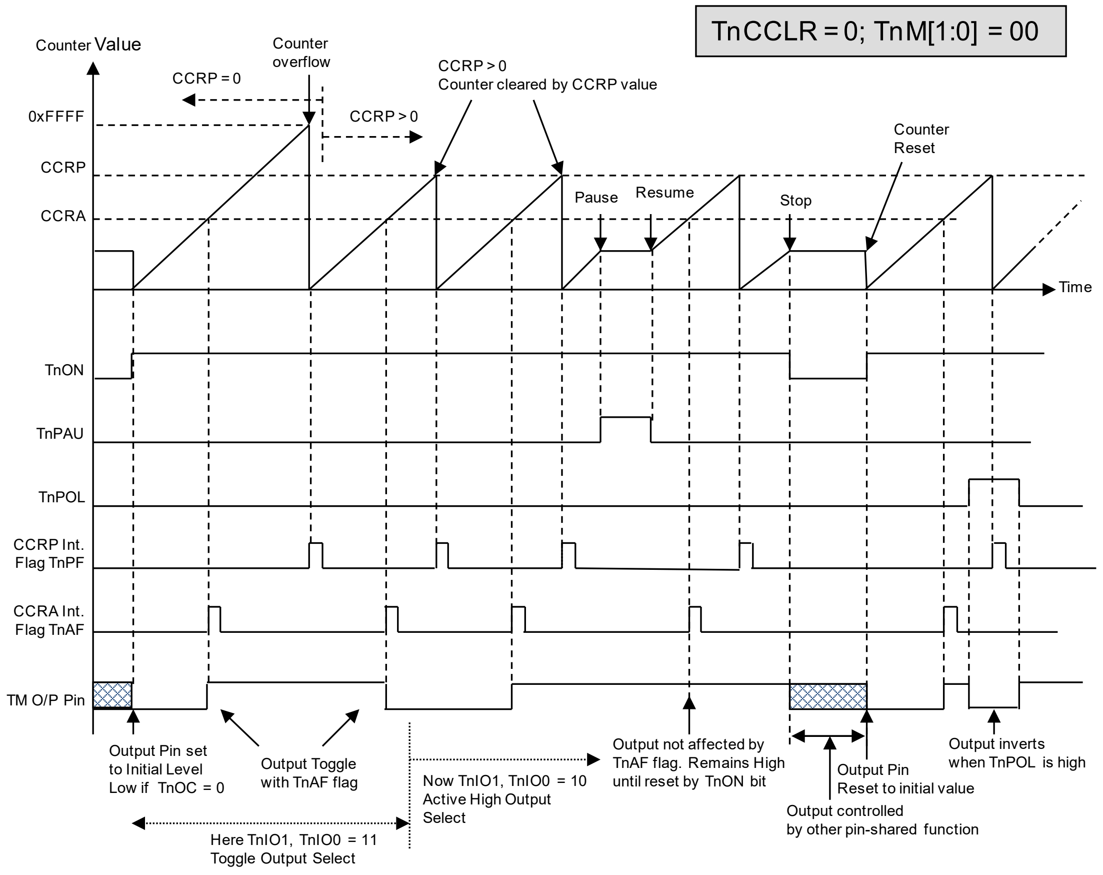
The Standard Type TM can operate in one of five operating modes, Compare Match Output Mode, PWM Output Mode, Single Pulse Output Mode, Capture Input Mode or Timer/Counter Mode. The operating mode is selected using the T2M1 and T2M0 bits in the TM2C1 register.

Compare Match Output Mode

To select this mode, bits T2M1 and T2M0 in the TM2C1 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 T2CCLR bit is low, there are two ways in which the counter can be cleared. One is when a compare match from Comparator P, the other is when the CCRP bits are all zero which allows the counter to overflow. Here both T2AF and T2PF interrupt request flags for Comparator A and Comparator P respectively, will both be generated.

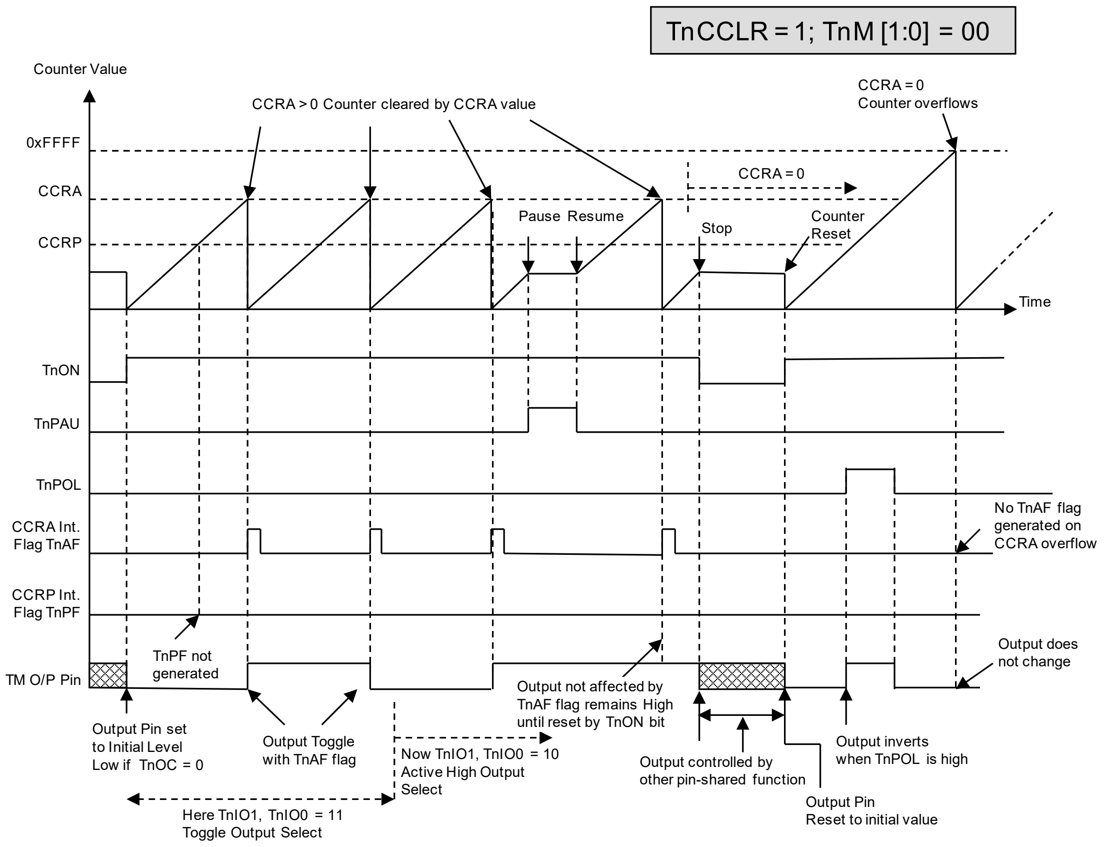
If the T2CCLR bit in the TM2C1 register is high then the counter will be cleared when a compare match occurs from Comparator A. However, here only the T2AF interrupt request flag will be generated even if the value of the CCRP bits is less than that of the CCRA registers. Therefore when T2CCLR is high no T2PF interrupt request flag will be generated. In the Compare Match Output Mode, the CCRA can not be set to "0".

As the name of the mode suggests, after a comparison is made, the TM output pin, will change state. The TM output pin condition however only changes state when a T2AF interrupt request flag is generated after a compare match occurs from Comparator A. The T2PF 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 T2IO1 and T2IO0 bits in the TM2C1 register. The TM output pin can be selected using the T2IO1 and T2IO0 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 T2ON bit changes from low to high, is setup using the T2OC bit. Note that if the T2IO1 and T2IO0 bits are zero then no pin change will take place.


TnCCLR = 0; TnM[1:0] = 00
Counter Value
Counter overflow
CCRP = 0
CCRP > 0
CCRP > 0
Counter cleared by CCRP value
0xFFFF
CCRP
CCRA
Pause
Resume
Stop
Counter Reset
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
Now TnIO1, TnIO0 = 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
Here TnIO1, TnIO0 = 11 Toggle Output Select

Compare Match Output Mode – TnCCLR = 0 (n=2)

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



Compare Match Output Mode – TnCCLR = 1 (n=2)

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

Timer/Counter Mode

To select this mode, bits T2M1 and T2M0 in the TM2C1 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. As the TM output pin is not used in this mode, the pin can be used as a normal I/O pin or other pin-shared function.

PWM Output Mode

To select this mode, bits T2M1 and T2M0 in the TM2C1 register should be set to 10 respectively and also the T2IO1 and T2IO0 bits 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 T2CCLR bit has no effect as the PWM period. 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 T2DPX bit in the TM2C1 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 T2OC bit In the TM2C1 register is used to select the required polarity of the PWM waveform while the two T2IO1 and T2IO0 bits are used to enable the PWM output or to force the TM output pin to a fixed high or low level. The T2POL bit is used to reverse the polarity of the PWM output waveform.

16-bit STM, PWM Mode, Edge-aligned Mode, T2DPX=0

CCRP	1~255	0
Period	CCRP×256	65536
Duty	CCRA	

If f_{SYS} = 16MHz, TM clock source is $f_{SYS}/4$, CCRP = 2 and CCRA =128,

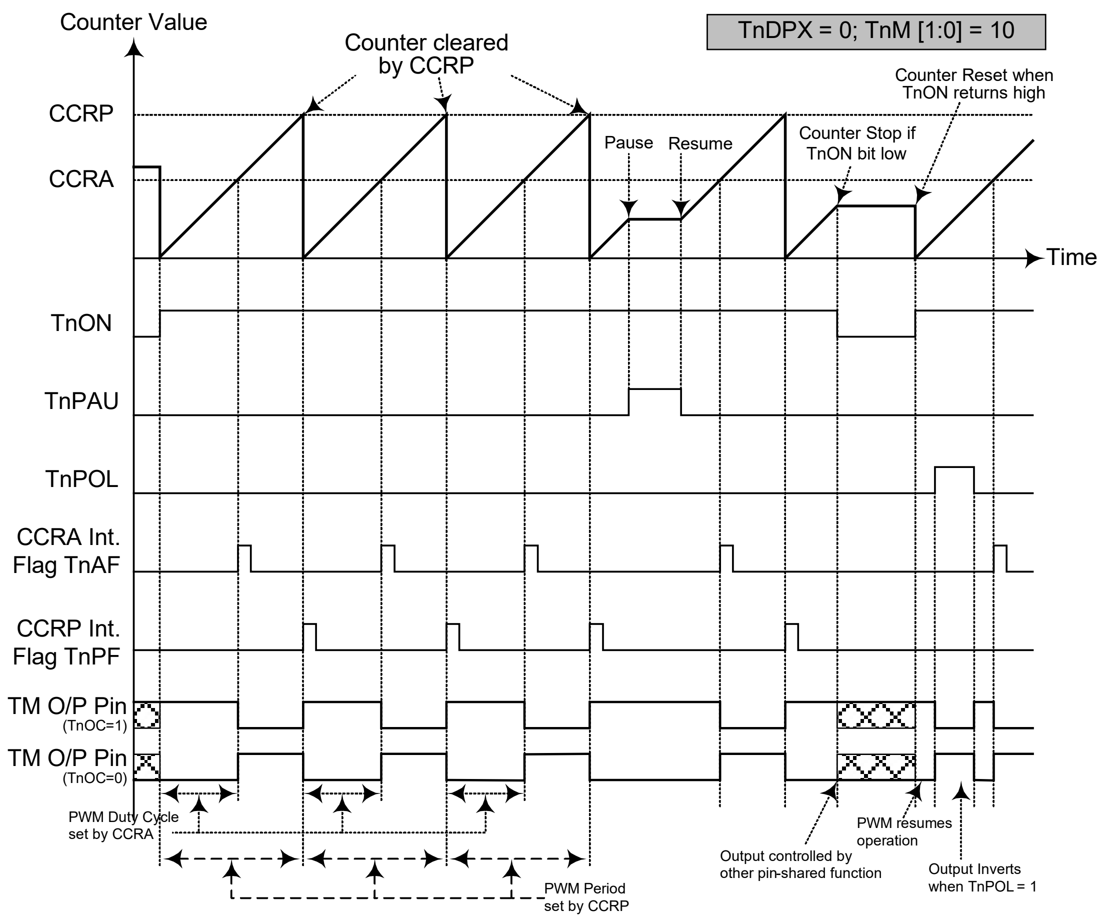
The STM PWM output frequency = $(f_{SYS}/4)$ / 512 = $f_{SYS}/2048$ = 7.8125kHz, duty = 128/512 = 25%.

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%.

16-bit STM, PWM Mode, Edge-aligned Mode, T2DPX=1

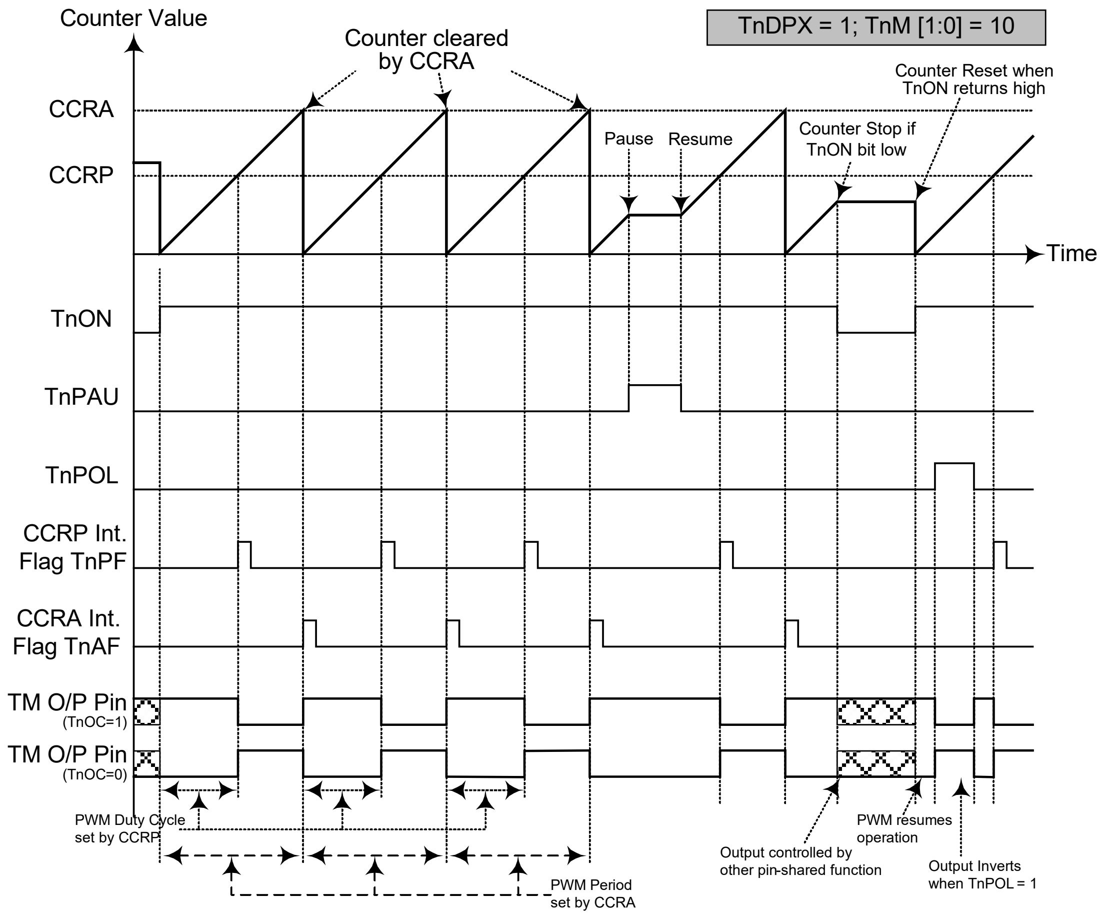
CCRP	1~255	0
Period	CCRA	
Duty	CCRP×256	65536

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×256) except when the CCRP value is equal to 0.



PWM Mode – TnDPX=0 (n=2)

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



PWM Mode—TnDPX=1 (n=2)

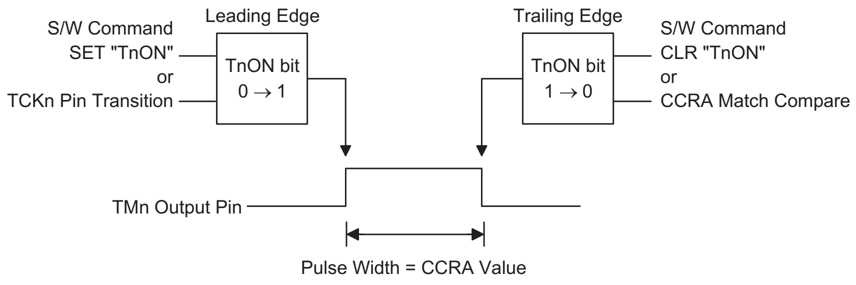
Note: 1. Here TnDPX = 1 - Counter cleared by CCRA
2. A counter clear sets PWM Period
3. The internal PWM function continues even when TnIO[1:0] = 00 or 01
4. The TnCCLR bit has no influence on PWM operation

Single Pulse Mode

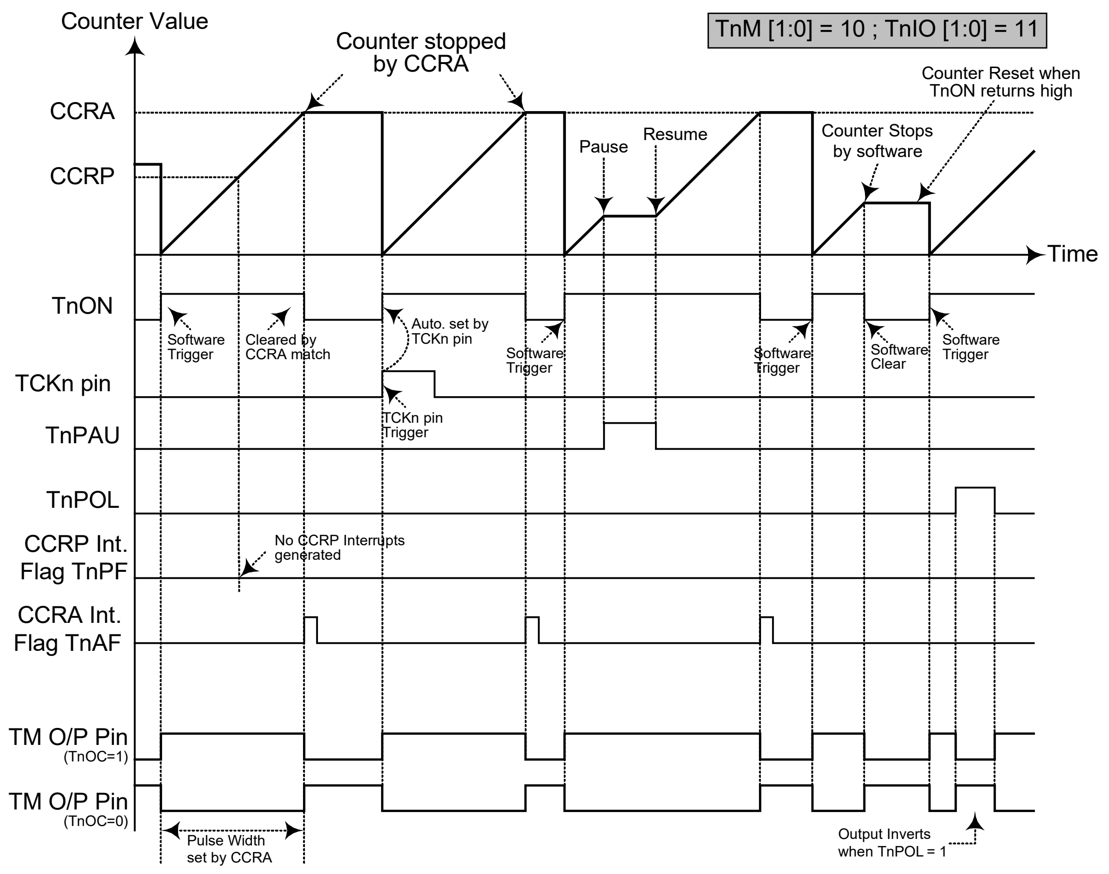
To select this mode, bits T2M1 and T2M0 in the TM2C1 register should be set to 10 respectively and also the T2IO1 and T2IO0 bits should be set to 11 respectively. The Single Pulse Output Mode, as the name suggests, will generate a single shot pulse on the TM output pin.

The trigger for the pulse output leading edge is a low to high transition of the T2ON bit, which can be implemented using the application program. However in the Single Pulse Mode, the T2ON bit can also be made to automatically change from low to high using the external TCK2 pin, which will in turn initiate the Single Pulse output.

When the T2ON bit transitions to a high level, the counter will start running and the pulse leading edge will be generated. The T2ON bit should remain high when the pulse is in its active state. The generated pulse trailing edge will be generated when the T2ON bit is cleared to zero, which can be implemented using the application program or when a compare match occurs from Comparator A.



Single Pulse Generation (n=2)



Single Pulse Mode (n=2)

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

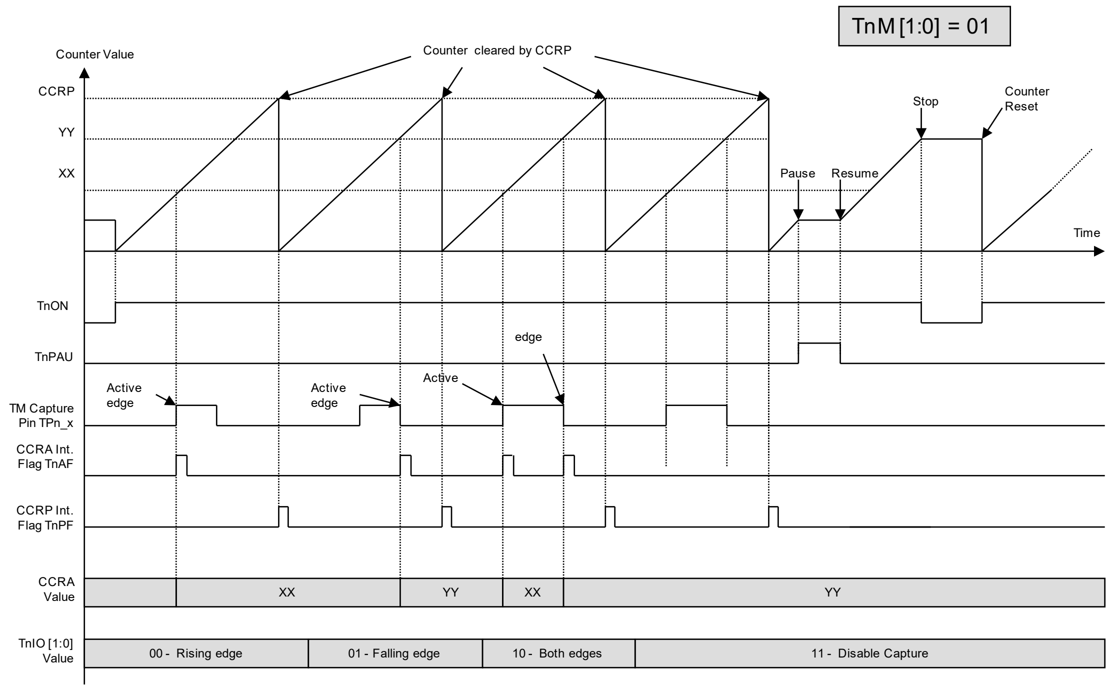
However a compare match from Comparator A will also automatically clear the T2ON bit and thus generate the Single Pulse output trailing edge. In this way the CCRA value can be used to control the pulse width. A compare match from Comparator A will also generate a TM interrupt. The counter can only be reset back to zero when the T2ON bit changes from low to high when the counter restarts. In the Single Pulse Mode CCRP is not used. The T2CCLR and T2DPX bits are not used in this Mode.

Capture Input Mode

To select this mode bits T2M1 and T2M0 in the TM2C1 register should be set to 01 respectively. This mode enables external signals to capture and store the present value of the internal counter and can therefore be used for applications such as pulse width measurements. The external signal is supplied on the TP2 pin, whose active edge can be either a rising edge, a falling edge or both rising and falling edges; the active edge transition type is selected using the T2IO1 and T2IO0 bits in the TM2C1 register. The counter is started when the T2ON bit changes from low to high which is initiated using the application program.

When the required edge transition appears on the TP2 pin the present value in the counter will be latched into the CCRA registers and a TM interrupt generated. Irrespective of what events occur on the TP2 pin the counter will continue to free run until the T2ON bit changes from high to low. When a CCRP compare match occurs the counter will reset back to zero; in this way the CCRP value can be used to control the maximum counter value. When a CCRP compare match occurs from Comparator P, a TM interrupt will also be generated. Counting the number of overflow interrupt signals from the CCRP can be a useful method in measuring long pulse widths. The T2IO1 and T2IO0 bits can select the active trigger edge on the TP2 pin to be a rising edge, falling edge or both edge types. If the T2IO1 and T2IO0 bits are both set high, then no capture operation will take place irrespective of what happens on the TP2 pin, however it must be noted that the counter will continue to run.

As the TP2 pin is pin shared with other functions, care must be taken if the TM is in the Input Capture Mode. This is because if the pin is setup as an output, then any transitions on this pin may cause an input capture operation to be executed. The T2CCLR and T2DPX bits are not used in this Mode.



Capture Input Mode (n=2)

Note: 1. TnM[1:0] = 01 and active edge set by the TnIO[1:0] bits
2. A TM Capture input pin active edge transfers the counter value to CCRA
3. The TnCCLR bit is not used
4. No output function - TnOC and TnPOL bits are not used
5. CCRP determines the counter value and the counter has a maximum count value when CCRP is equal to zero.

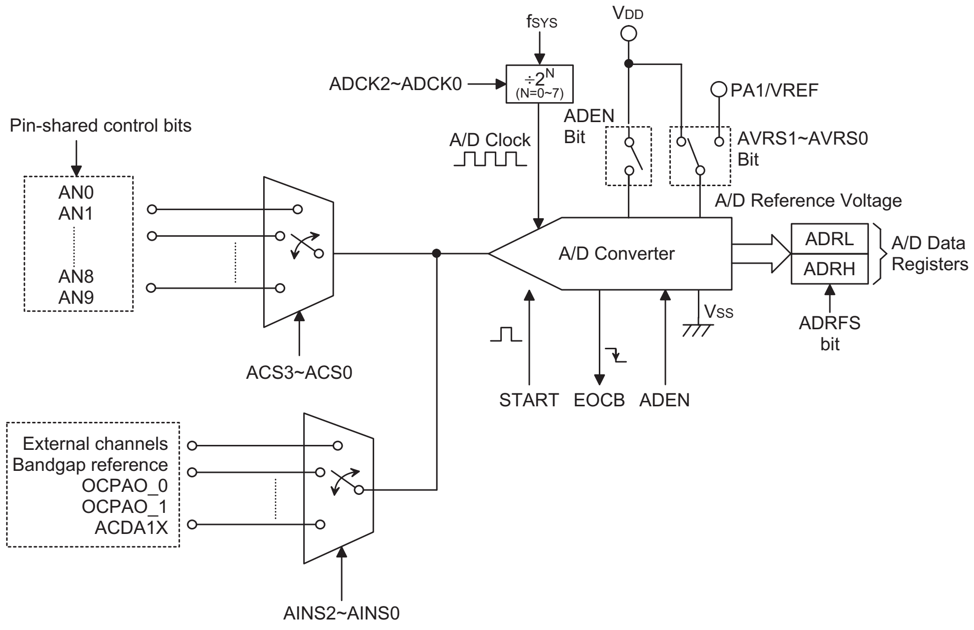
Analog to Digital Converter

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

A/D Overview

The 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 12-bit digital value. The external or internal analog signal to be converted is determined by the AINS2~AINS0 bits together with the ACS3~ACS0 bits. Note that when the internal analog signal is to be converted, the pin-shared control bits should also be properly configured except the AINS[2:0] and ACS[3:0] bit fields. More detailed information about the A/D input signal is described in the "A/D Converter Control Registers" and "A/D Converter Input Signal" sections respectively.

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



A/D Converter Structure

A/D Converter Register Description

Overall operation of the A/D converter is controlled using four registers. A read only register pair exists to store the ADC data 12-bit value. The remaining two 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
ADRL(ADRFS=0)	D3	D2	D1	D0	—	—	—	—
ADRL(ADRFS=1)	D7	D6	D5	D4	D3	D2	D1	D0
ADRH(ADRFS=0)	D11	D10	D9	D8	D7	D6	D5	D4
ADRH(ADRFS=1)	—	—	—	—	D11	D10	D9	D8
ADCR0	START	EOCB	ADEN	ADRFS	ACS3	ACS2	ACS1	ACS0
ADCR1	AINS2	AINS1	AINS0	AVRS1	AVRS0	ADCK2	ADCK1	ADCK0

A/D Converter Register List

A/D Converter Data Registers – ADRL, ADRH

The device, which has an internal 12-bit A/D converter, requires two data registers, a high byte register, known as ADRH, and a low byte register, known as ADRL. After the conversion process takes place, these registers can be directly read by the microcontroller to obtain the digitised conversion value. As only 12 bits of the 16-bit register space is utilised, the format in which the data is stored is controlled by the ADRFS bit in the ADCR0 register as shown in the accompanying table. D0~D11 are the A/D conversion result data bits. Any unused bits will be read as zero.

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

A/D Data Registers

A/D Converter Control Registers – ADCR0, ADCR1, PAS0, PAS1, PBS0

To control the function and operation of the A/D converter, several control registers known as ADCR0 and ADCR1 are provided. These 8-bit registers define functions such as the selection of which analog channel is connected to the internal A/D converter, the digitised data format, the A/D clock source as well as controlling the start function and monitoring the A/D converter end of conversion status. The ACS3~ACS0 bits in the ADCR0 register are used to determine which external input is selected to be converted. The AINS2~AINS0 bits in the ADCR1 register are used to determine that the analog signal to be converted comes from the internal analog signal or external analog channel input. If the AINS2~AINS0 bits are set to “000”or “101”~“111”, the external analog channel input is selected to be converted and the ACS3~ACS0 bits can determine which external channel is selected to be converted. If the AINS2~AINS0 bits are other value except “000” and “101”~“111”, one of the internal signals is selected to be converted. Care must be taken when the internal analog signal is selected to be converted. If the internal analog signal is selected to be converted, the selected external input determined by the ACS3~ACS0 bits must be configured to a floating state by properly setting the ACS3~ACS0 bits to a value from 1010~1111. Otherwise, the external channel input will be connected together with the internal analog signal. This will result in unpredictable situations such as an irreversible damage.

The pin-shared function control registers, named PAS0 and PAS1, contain the corresponding pin-shared selection 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. 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.

AINS [2:0]	ACS [3:0]	Input Signals	Description
000, 101~111	0000~1001	AN0~AN9	External pin analog input
	1010~1111	—	Floating, no channel is selected.
001	1010~1111	VBG	Internal Bandgap reference voltage
010	1010~1111	OCPAO_0	The amplified OCPI0 input voltage
011	1010~1111	OCPAO_1	The amplified OCPI1 input voltage
100	1010~1111	ACDA1X	The AC power detected result

A/D Converter Input Signal Selection

ADCR0 Register

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

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

Bit 5 **ADEN**: ADC module power on/off control bit
0: ADC module power off
1: ADC module power on
This bit controls the power to the A/D internal function. This bit should be set high to enable the A/D converter. If the bit is cleared to zero then the A/D converter will be switched off reducing the device 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: 1. It is recommended to set ADEN=0 before entering IDLE/SLEEP Mode for saving power.
2. ADEN=0 will power down the ADC module.

Bit 4 **ADRFS**: A/D data format control bit
0: ADC Data MSB is ADRH bit 7, LSB is ADRL bit 4
1: ADC Data MSB is ADRH bit 3, LSB is ADRL bit 0

This bit controls the format of the 12-bit converted A/D value in the two A/D data registers. Details are provided in the A/D data register section.

Bit 3~0 **ACS3~ACS0**: A/D Converter external analog input channel selection
0000: AN0
0001: AN1
0010: AN2
0011: AN3
0100: AN4
0101: AN5
0110: AN6
0111: AN7
1000: AN8
1001: AN9
1010~1111: Floating, no channel is selected.

ADCR1 Register

Bit	7	6	5	4	3	2	1	0
Name	AINS2	AINS1	AINS0	AVRS1	AVRS0	ADCK2	ADCK1	ADCK0
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 7~5 **AINS2~AINS0**: A/D Converter input signal selection
000: External source – External analog channel input
001: V_{BG}
010: OCPAO_0
011: OCPAO_1
100: ACDA1X
101~111: External source – External analog channel input

Bit 4~3 **AVRS1~AVRS0**: A/D Converter reference voltage selection
00: From VREF pin
01: From internal ADC power
1x: From VREF pin

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

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.

Although the A/D clock source is determined by the system clock f_{SYS}, and by bits ADCK2~ADCK0, there are some limitations on the A/D clock source speed range that can be selected. As the recommended range of permissible A/D clock period, t_{AD}, is from 0.5μs to 10μs, care must be taken for selected system clock frequencies. For example, if the system clock operates at a frequency of 4MHz, the ADCK2~ADCK0 bits should not be set to 000B or 110B. Doing so will give A/D clock periods that are less than the minimum A/D clock period or greater than the maximum A/D clock period which may result in inaccurate A/D conversion values.

Refer to the following table for examples, where values marked with an asterisk * show where, depending upon the device, special care must be taken, as the values may be less than the specified minimum A/D Clock Period.

f_{SYS}	A/D Clock Period (t_{AD})							
	ADCK2, ADCK1, ADCK0 =000 (f_{SYS})	ADCK2, ADCK1, ADCK0 =001 (f_{SYS}/2)	ADCK2, ADCK1, ADCK0 =010 (f_{SYS}/4)	ADCK2, ADCK1, ADCK0 =011 (f_{SYS}/8)	ADCK2, ADCK1, ADCK0 =100 (f_{SYS}/16)	ADCK2, ADCK1, ADCK0 =101 (f_{SYS}/32)	ADCK2, ADCK1, ADCK0 =110 (f_{SYS}/64)	ADCK2, ADCK1, ADCK0 =111
1MHz	1μs	2μs	4μs	8μs	16μs*	32μs*	64μs*	Undefined
2MHz	500ns	1μs	2μs	4μs	8μs	16μs*	32μs*	Undefined
4MHz	250ns*	500ns	1μs	2μs	4μs	8μs	16μs*	Undefined
8MHz	125ns*	250ns*	500ns	1μs	2μs	4μs	8μs	Undefined

A/D Clock Period Examples

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

The reference voltage supply to the A/D Converter can be supplied from either the internal ADC power or from an external reference sources supplied on pin VREF. The desired selection is made using the AVRS1 and AVRS0 bits. When the AVRS bit field is set to "01", the A/D converter reference voltage will come from the internal ADC power. Otherwise, if the AVRS bit field is set to any other value except "01", the A/D converter reference voltage will come from the VREF pin. As the VREF pin is pin-shared with other functions, when the VREF pin is selected as the reference voltage supply pin, the VREF pin-shared function control bits should be properly configured to disable other pin functions.

AVRS [1:0]	Reference Voltage	Description
00	VREF	A/D Converter Reference voltage comes from VREF pin
01	Internal ADC power	A/D Converter Reference voltage comes from internal ADC power
1x	VREF	A/D Converter Reference voltage comes from VREF pin

A/D Converter Reference Voltage Selection

A/D Converter Input Signal

All of the A/D analog input pins are pin-shared with the I/O pins on Port A as well as other functions. The corresponding selection bits for each I/O pin in the PAS0~PAS1 and PBS0 registers, determine whether the input pins are setup as A/D converter analog inputs or whether they have other functions. If the pin-shared function control bits configure its corresponding pin as an A/D analog channel input, the pin will be setup to be an A/D converter external channel input and the original pin functions will be disabled. In this way, pins can be changed under program control to change their function between A/D inputs and other functions. All pull-high resistors, which are setup through register programming, will be automatically disconnected if the pins are setup as A/D inputs. Note that it is not necessary to first setup the A/D pin as an input in the PAC port control register to enable the A/D input as when the pin-shared function control bits enable an A/D input, the status of the port control register will be overridden.

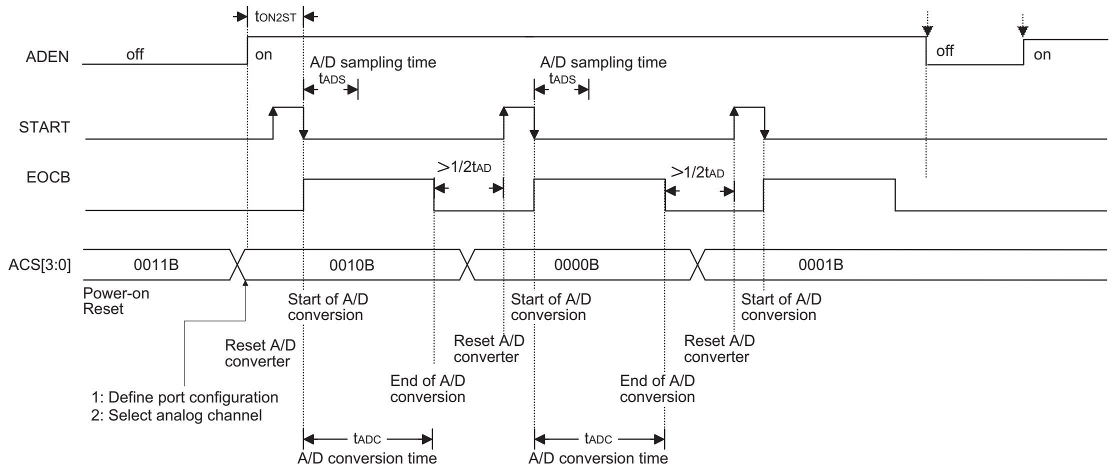
There are also some internal analog signals which can be connected to the A/D converter as the analog input signal by configuring the AINS2~AINS0 bits. If the internal analog signal is selected to be converted, the selected external input determined by the ACS3~ACS0 bits must be configured to a floating state by properly setting the ACS3~ACS0 bits to a value from 1010~1111.

Conversion Rate and Timing Diagram

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

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

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



A/D Conversion Timing

Summary of A/D Conversion Steps

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

- Step 1
  Select the required A/D conversion clock by correctly programming the ADCK2~ADCK0 bits in the ADCR1 register.
- Step 2
  Enable the A/D Converter by setting the ADEN bit to 1.
- Step 3
  Select which signal is to be connected to the internal A/D converter by correctly programming the AINS2~AINS0 bits.
  Select the external channel input to be converted, go to Step 4.
  Select the internal analog signal to be converted, go to Step 5.
- Step 4
  If the A/D input signal comes from the external channel input selecting by configuring the AINS bit field, the desired analog channel then should be selected by configuring the ACS bit field. The corresponding pins should be configured as A/D input function by configuring the relevant pin-shared function control bits. The After this step, go to Step 6.
- Step 5
  Before the A/D input signal is selected to come from the internal analog signal by configuring the AINS bit field, the corresponding pins must never be configured as A/D input function by configuring the relevant pin-shared function control bits. The desired internal analog signal then can be selected by configuring the AINS bit field. After this step, go to Step 6.
- Step 6
  Select the reference voltage source by configuring the AVRS1~AVRS0 bits.
- Step 7
  Select A/D Converter output data format by setting the ADRFS bit.
- Step 8
  If A/D Converter interrupt is used, the interrupt control registers must be correctly configured to ensure the A/D interrupt function is active. The master interrupt control bit, EMI, and the A/D converter interrupt bits, ADE, must both set high in advance.

- Step 9
  The A/D convert procedure can now be initialized by setting the START bit from low to high and then low again.
- Step 10
  If A/D Converter is under conversion, the EOCB flag will be set high. After the A/D conversion process is completed, the EOCB flag will go low, and then the output data can be read from ADRH and ADRL registers. If the ADC interrupt is enabled and the stack is not full, data can be acquired by interrupt service program. Another way to get the A/D output data is polling the EOCB flag.

Programming Considerations

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

A/D Transfer Function

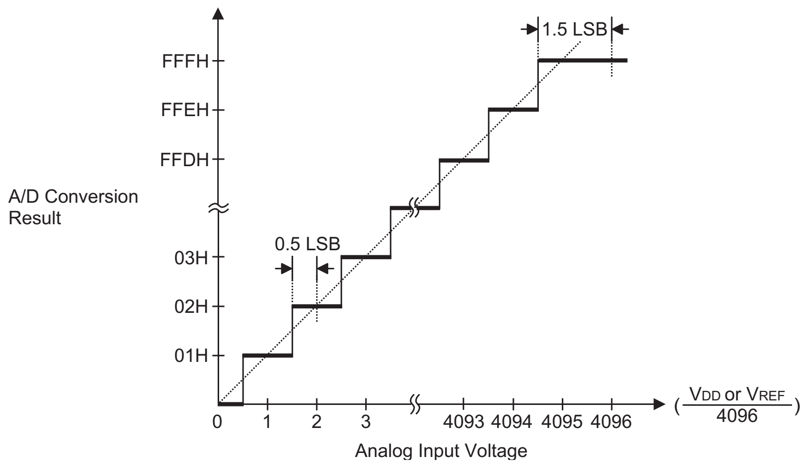
As the devices contain a 12-bit A/D converter, its full-scale converted digitised value is equal to FFFH. Since the full-scale analog input value is equal to the V_{DD} or V_{REF} voltage, this gives a single bit analog input value of V_{DD} or V_{REF} divided by 4096.

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

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

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

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



Ideal A/D Transfer Function

A/D Programming Examples

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

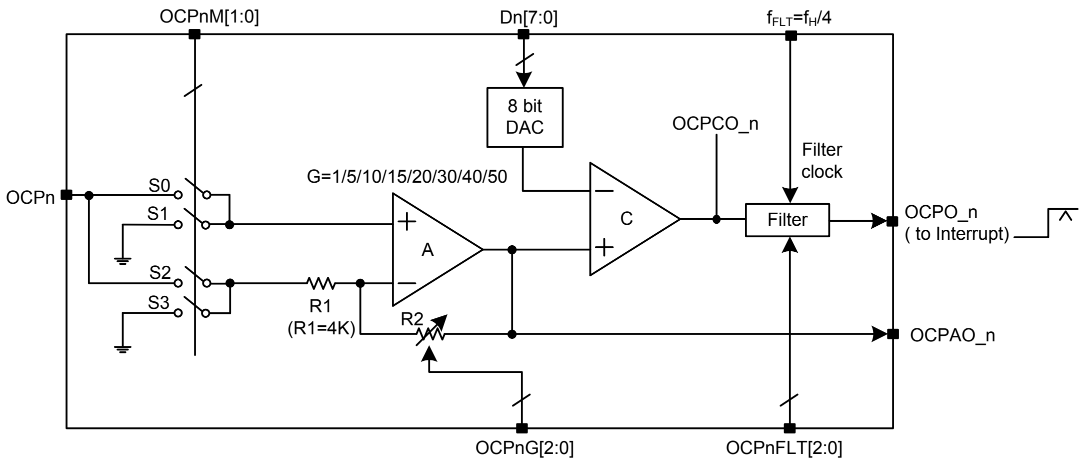
```
clr  ADE              ; disable ADC interrupt
mov  a,0BH
mov  ADCR1,a          ; select fSYS/8 as A/D clock and select the external input signals
                      ; voltage
set  ADEN
mov  a,03h            ; setup PAS0 to configure pin AN0
mov  PAS0,a
mov  a,20h
mov  ADCR0,a          ; enable and connect AN0 channel to A/D converter
:
start_conversion:
clr  START            ; high pulse on start bit to initiate conversion
set  START            ; reset A/D
clr  START            ; start A/D
polling_EOC:
sz   EOCB             ; poll the ADCR0 register EOCB bit to detect end of A/D conversion
jmp  polling_EOC      ; continue polling
mov  a,ADRL           ; read low byte conversion result value
mov  ADRL_buffer,a    ; save result to user defined register
mov  a, ADRH          ; read high byte conversion result value
mov  ADRH_buffer,a    ; save result to user defined register
:
:
jmp  start_conversion  ; start next a/d conversion
```

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

```
clr  ADE              ; disable ADC interrupt
mov  a,0BH
mov  ADCR1,a          ; select fSYS/8 as A/D clock and select the external input signals
                      ; voltage
set  ADEN
mov  a,03h            ; setup PAS0 to configure pin AN0
mov  PAS0,a
mov  a,20h
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_ISR:
mov  acc_stack,a      ; save ACC to user defined memory
mov  a,STATUS
mov  status_stack,a   ; save STATUS to user defined memory
:
:
mov  a,ADRL           ; read low byte conversion result value
mov  ADRL_buffer,a    ; save result to user defined register
mov  a,ADRH           ; read high byte conversion result value
mov  ADRH_buffer,a    ; save result to user defined register
:
:
EXIT_INT_ISR:
mov  a,status_stack
mov  STATUS,a         ; restore STATUS from user defined memory
mov  a,acc_stack      ; restore ACC from user defined memory
reti
```

Over Current Protection Function

OCP is an abbreviation for Over Current Protection. The OCP detects an input voltage which is proportional to the monitored source current. If the input voltage is larger than the reference voltage set by the DAC, the OCP will generate an output signal to indicate that the source current is over the specification.



Over Current Protection Circuit (n=0, 1)

OCP Circuit Operation

The source voltage is input from the OCPn. Four switches S0~S3 consist of a mode select function. An operational amplifier and two resistors form a PGA function. The PGA gain can be positive or negative determined by the input voltage connected to the positive input or negative input of the PGA. The DAC is used to generate a reference voltage. The comparator compares the reference voltage and the amplified output voltage OCPAO_n to produce OCPCO_n. Finally OCPCO_n is filtered to generate the OCP output OCPO_n for OCP interrupt trigger. It is the debounced version of OCPCO_n which are used to indicate whether the source current is over the specification or not.

Note that the filter clock; f_{FLT} is the HXT clock. The amplified output voltage also can be read out by means of another ADC from OCPAO_n. The DAC output voltage is controlled by the OCPnREF register and the DAC output is defined as

DAC V_{OUT} = (DAC V_{REF}/256) × OCPnREF[7:0]..(1)

OCP Register Description

The OCPnC0 and OCPnC1 registers are the OCPn control registers which control the OCPn operation mode, PGA and filter functions. The OCPnREF register is used to provide the reference voltages for the over current protection. OCPnACAL and OCPnCCAL are used to cancel out the operational amplifier and comparator input offset.

OCPnC0 Register

Bit	7	6	5	4	3	2	1	0
Name	OCPnM1	OCPnM0	—	—	—	—	—	—
R/W	R/W	R/W	—	—	—	—	—	—
POR	0	0	—	—	—	—	—	—

Bit 7~6 **OCPnM1~OCPnM0**: Mode selection
00: OCPn function disable, S1, S3 on, S0, S2 off
01: Non-Inverter mode, S0, S3 on, S1, S2 off
10: Inverter mode, S1, S2 on, S0, S3 off
11: Calibration mode, S1, S3 on, S0, S2 off
Note: Disable means OPA, CMP, DAC, Filter all off & comparator output=low & OCPAO_n=low.

Bit 5~0 Unimplemented, read as "0"

OCPnC1 Register

Bit	7	6	5	4	3	2	1	0
Name	OCPnO	—	OCPnG2	OCPnG1	OCPnG0	OCPnFLT2	OCPnFLT1	OCPnFLT0
R/W	R	—	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	—	0	0	0	0	0	0

Bit 7 **OCPnO**: OCPn output (read only)

Bit 6 Unimplemented, read as "0"

Bit 5~3 **OCPnG2~OCPnG0**: OPAn Gain
000: 1
001: 5
010: 10
011: 15
100: 20
101: 30
110: 40
111: 50

bit 2~0 **OCPnFLT2~OCPnFLT0**: Demodulator Filter Selection
000: 0 t_{FLT} (without filter)
001: 1~2 × t_{FLT}
010: 3~4 × t_{FLT}
011: 7~8 × t_{FLT}
100: 15~16 × t_{FLT}
101: 31~32 × t_{FLT}
110: 63~64 × t_{FLT}
111: 127~128 × t_{FLT}
Note: $f_{FLT}=f_H/4$, $t_{FLT}=1/f_{FLT}$

OCPnREF 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 **OCPnREF**: Select reference voltage for over current protection
Reference voltage = (DAC reference voltage /256) × (N), N=OCPnREF[7:0]

OCPnACAL Register – OPA Calibration Register

Bit	7	6	5	4	3	2	1	0
Name	OCPnAOFM	OCPnARS	OCPnAOF5	OCPnAOF4	OCPnAOF3	OCPnAOF2	OCPnAOF1	OCPAnOF0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit7 **OCPnAOFM**: Input offset voltage cancellation mode or normal operating mode selection
0: Normal operating mode
1: Input offset voltage cancellation mode

Bit6 **OCPnARS**: Input offset voltage cancellation reference selection bit
0: Select negative input as the reference input
1: Select positive input as the reference input

Bit 5~0 **OCPnAOF5~OCPnAOF0**: Input offset voltage calibration control

OCPnCCAL Register – Comparator Calibration Register

Bit	7	6	5	4	3	2	1	0
Name	OCPnAXCX	OCPnCOFM	OCPnCRS	OCPnCOF4	OCPnCOF3	OCPnCOF2	OCPnCOF1	OCPnCOF0
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

Bit7 **OCPnAXCX**: OPA/Comparator output for calibration; positive logic (read only)
If OCPnAOFM=1, this bit is calibration output for OPA
If OCPnCOFM=1, this bit is calibration output for comparator
Note: OCPnAOFM & OCPnCOFM can't be 1 simultaneously.

Bit6 **OCPnCOFM**: Input offset voltage cancellation mode or normal operating mode selection
0: Normal operating mode
1: Input offset voltage cancellation mode

Bit5 **OCPnCRS**: Input offset voltage cancellation reference selection bit
0: Select negative input as the reference input
1: Select positive input as the reference input

Bit 4~0 **OCPnCOF4~OCPnCOF0**: Input offset voltage calibration control

Input Voltage Range

The input voltage can be positive or negative, which together with the PGA operating mode provides for a more flexible application.

(1) If $V_{IN} > 0$ and the PGA operates in the non-inverting mode, the output voltage of the PGA is

$VO_{PGA} = (1+R_2/R_1) \times V_{IN}$ .... (2)

(2) When the PGA operates in the non-inverter mode, it also provides a unity gain buffer function.

If OCPnM[1:0]=01 and OCPnG[2:0]=000, the PGA gain will be 1 and is configured as an unity gain buffer. The switches S2 and S3 will be off internally and the output voltage of the PGA is

$VO_{PGA} = V_{IN}$... (3)

(3) If $0 > V_{IN} > -0.7V$ and the PGA operates in inverter mode, the output voltage of the PGA is

$VO_{PGA} = -(R_2/R_1) \times V_{IN}$ .... (4)

Note: if V_{IN} is negative, it should not be lower than -0.7V to avoid leakage current.

Offset Calibration

The OCP circuit has 4 operating modes controlled by OCPnM1~OCPnM0. One of these modes is the calibration mode. In the calibration mode, the OP and comparator offset can be calibrated.

OPAMP calibration:

Step1: Set OCPnM[1:0] =11, OCPnAOFM=1, the OCPn is now in the OPAMP calibration mode

Step2: Set OCPnAOF5~OCPnAOF0 =000000 then read the OCPnAXCX bit status

Step3: Let OCPnAOF=OCPnAOF+1 then read the OCPnAXCX bit status; if OCPnAXCX is changed, record the OCPnAOF[5:0] data as VOS1

Step4: Set OCPnAOF [5:0] 111111 then read the OCPnAXCX bit status

Step5: Let OCPnAOF=OCPnAOF-1 then read the OCPnAXCX bit status; if OCPnAXCX is changed, record the OCPnAOF[5:0] data as VOS2

Step6: Restore VOS = (VOS1 + VOS2)/2 to the OCPnAOF[5:0] bits. The calibration is now finished.

Comparator calibration:

Step1: Set OCPnM[1:0] =11, OCPnCOFM=1, the OCP is now in the comparator calibration mode

Step2: Set OCPnCOF [4:0] =00000 then read the OCPnAXCX bit status

Step3: Let OCPnCOF=OCPnCOF+1 then read the OCPnAXCX bit status; if OCPnAXCX is changed, record the OCPnCOF [4:0] data as VOS1

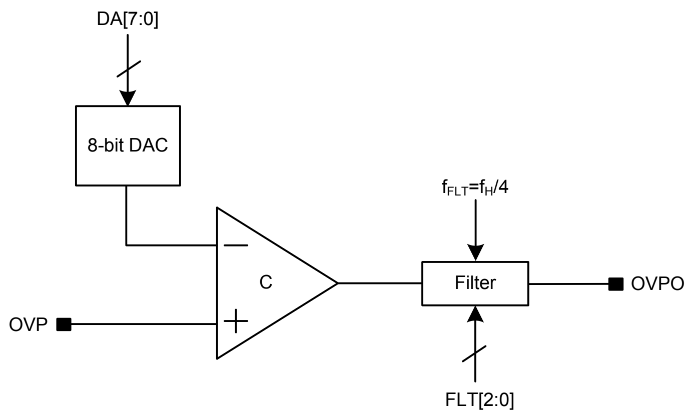
Step4: Set OCPnCOF [4:0] =11111 then read the OCPnAXCX bit status

Step5: Let OCPnCOF=OCPnCOF-1 then read the OCPnAXCX bit status; if OCPnAXCX data is changed, record the OCPnCOF [4:0] data as VOS2.

Step6: Restore VOS = (VOS1 + VOS2)/2 to the OCPnCOF [4:0] bits. The calibration is now finished.

Over Voltage Protection Function

OVP is an abbreviation for Over Voltage Protection. The OVP detects an input voltage which is proportional to the monitored source voltage. If the input voltage is larger than the reference voltage set by the DAC, the OVP will generate an output signal to indicate that the source voltage is over the specification.



Over Voltage Protection Circuit

OVP Circuit Operation

The source voltage is input from OVP and then connected to one input of comparator. The DAC is used to generate a reference voltage. The comparator compares the reference voltage with the input voltage to produce the OVPO signal.

OVP Register Description

The OVPC register is the OVP control register which setup the OVP enable/disable and filter functions. The OVPREF register is used to provide the reference voltages for the over voltage protection. OVPCCAL is used to control the comparator input offset.

OVPC Register

Bit	7	6	5	4	3	2	1	0
Name	OVPEN	—	—	OVPO	—	OVPFLT2	OVPFLT1	OVPFLT0
R/W	R/W	—	—	R	—	R/W	R/W	R/W
POR	0	—	—	0	—	0	0	0

Bit 7 **OVPEN**: OVP Enable Control
0: OVP (Comparator + DAC) disable
1: OVP (Comparator + DAC) enable

Bit 6~5 Unimplemented, read as "0"

Bit 4 **OVPO**: OVP Output Bit, read only

Bit 3 Unimplemented, read as "0"

bit 2~0 **OVPFLT2~OVPFLT0**: Filter Selection
000: 0 t_{FLT} (without filter)
001: 1~2 × t_{FLT}
010: 3~4 × t_{FLT}
011: 7~8 × t_{FLT}
100: 15~16 × t_{FLT}
101: 31~32 × t_{FLT}
110: 63~64 × t_{FLT}
111: 127~128 × t_{FLT}
Note: $f_{FLT}= f_H/4$, $t_{FLT} =1/f_{FLT}$

OVPREF 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	**OVPREF**: Select reference voltage for over voltage protection Reference voltage= (DAC reference voltage /256) × (N), N=OVPREF[7:0]

OVPCCAL Register – Comparator Calibration Register

Bit	7	6	5	4	3	2	1	0
Name	OVPCX	OVPCOFM	OVPCRS	OVPCOF4	OVPCOF3	OVPCOF2	OVPCOF1	OVPCOF0
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

Bit7	**OVPCX**: comparator output; positive logic (read only)
Bit6	**OVPCOFM**: Input offset voltage cancellation mode or normal operating mode selection 0: Normal operating mode 1: Input offset voltage cancellation mode
Bit5	**OVPCRS**: Input offset voltage cancellation reference selection bit 0: Select negative input as the reference input 1: Select positive input as the reference input
Bit 4~0	**OVPCOF4~OVPCOF0**: Input offset voltage calibration control

Offset Calibration

The OVP circuit has a comparator. The comparator offset can be calibrated.

Comparator calibration:

Step1: Set OVPCOFM=1, the OVP is now in the comparator calibration mode

Step2: Set OVPCOF [4:0] =00000 then read the OVPCX bit status

Step3: Let OVPCOF=OVPCOF+1 then read the OVPCX bit status; if OVPCX is changed, record the VOPCOF[4:0] data as VOS1

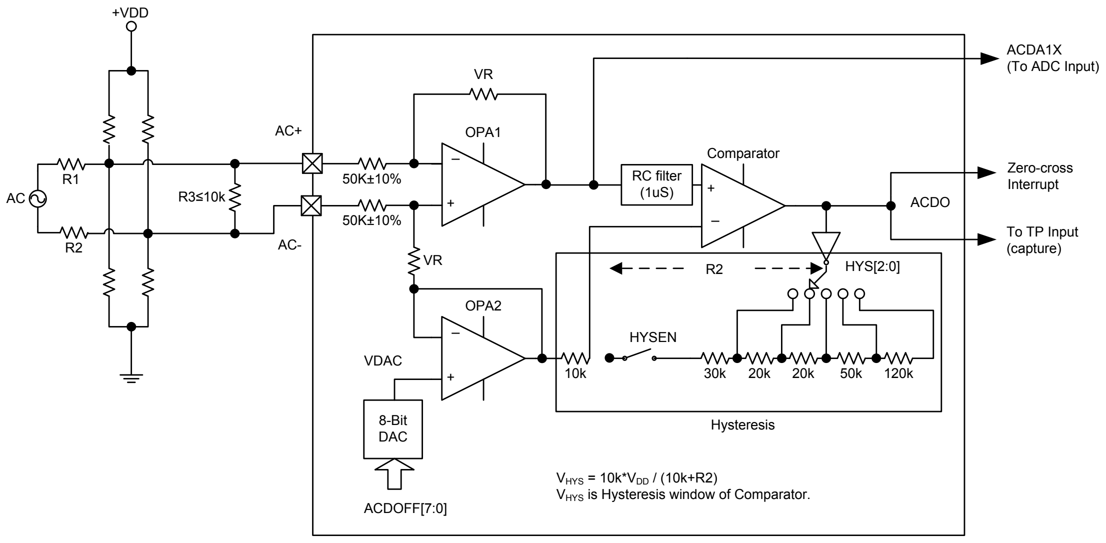
Step4: Set OVPCOF [4:0] =11111 then read the OVPCX bit status

Step5: Let OVPCOF=OVPCOF-1 then read the OVPCX bit status; if OVPCX data is changed, record the VOPCOF[4:0] data as VOS2.

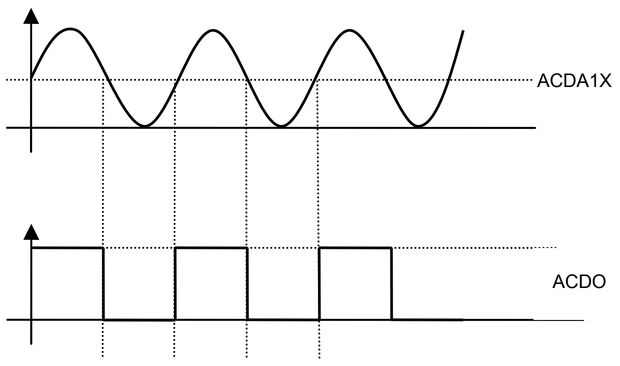
Step6: Restore VOS = (VOS1 + VOS2)/2 to the VOPCOF[4:0] bits. The calibration is now finished.

AC Detect Circuitry

The AC Detect Circuitry can detect whether the AC power is being normally supplied to the application. The AC power supply is connected to the AC+ and AC- pins and the input AC voltage is 220V. The AC power detected result ACDA1X is connected to the A/D converter input to measure the voltage. The signal, ACDO which is obtained by comparing the ACDA1X signal with a programmable reference voltage, is then sent to the interrupt circuitry and the internal Standard Type TM capture input, TP2. Also the ACDA1X and ACDO signals are bonded to the external pins for the application reference and measurement. Note that the operational amplifiers and comparator in the AC detect circuitry are all constructed using P-type input structures and the hysteresis voltage ranges from 50mV to 100mV.



AC Detect Circuitry Diagram



AC Detect Output Signal

AC Detect Registers

The AC Detect circuitry function is controlled by corresponding registers described in the accompanying sections.

Register Name	Bit								
	7	6	5	4	3	2	1	0	
ACDC	ACDEN	HYSEN	HYS2	HYS1	HYS0	—	—	VR	—
ACDOFF	D7	D6	D5	D4	D3	D2	D1	D0	—

AC Detect Circuitry Registers List

ACDC Register – AC Detect Control Register

Bit	7	6	5	4	3	2	1	0
Name	ACDEN	HYSEN	HYS2	HYS1	HYS0	—	—	VR
R/W	R/W	R/W	R/W	R/W	R/W	—	—	R/W
POR	0	0	0	0	0	—	—	0

Bit 7 **ACDEN**: AC detection enable control
0: Disable
1: Enable

Bit 6 **HYSEN**: Hysteresis voltage enable control
0: Disable
1: Enable

Bit 5~3 **HYS2~HYS0**: Hysteresis window selection
000: 30K
001: 50K
010: 70K
011: 120K
1xx: 240K

Bit 2~1 Unimplemented, read as "0"

Bit 0 **VR**: AC Detect Circuitry VR value selection
0: 50kΩ
1: 100kΩ

ACDOFF Register – AC Detect Offset Voltage

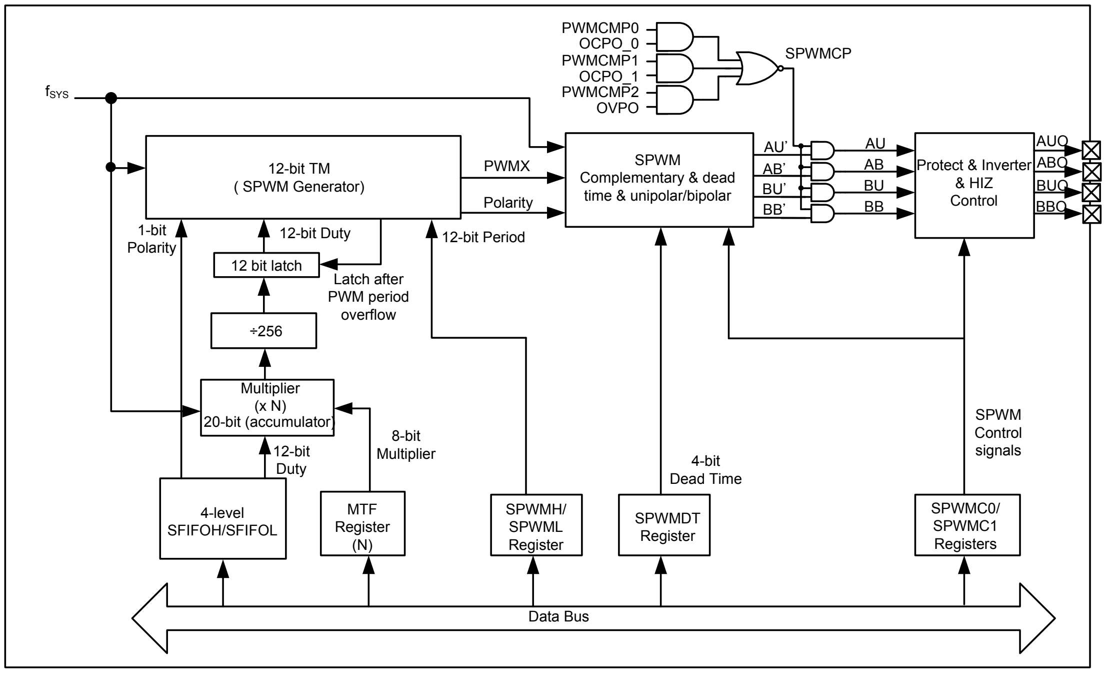
Bit	7	6	5	4	3	2	1	0
Name	D7	D6	D5	D4	D3	D2	D1	D0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~0 **D7~D0**: AC Offset DAC data bits

$$\text{DAC output} = \frac{(D7 \sim D0) \times V_{DD}}{256}$$

Sine-Wave PWM Generator

The sine-wave PWM generator includes a 12-bit Sine-PWM TM, a PWM complementary circuitry, a dead time insertion together with a complementary output pair for protection and inverter control. The AUO, ABO, BUO and BBO signals output transitions will result in a different current direction passing through the external MOSFETs to generate the sine wave signal.



Sine-Wave PWM Generator Block Diagram

Sine-Wave PWM Registers

Overall operation of the Sine-wave PWM generator is controlled using eight registers. A PWM period register pair, named as SPWMH and SPWML, exists to store the desired 12-bit PWM period value. The PWM duty value is stored in a 12-bit SFIFOL/SFIFOH register pair which is constructed in a 4-level FIFO structure. The dead time duration is determined by the SPWMDT register. The registers, SPWMC0/SPWMC1, are used for PWM setup, complementary output control, protecting and inverting control. The remaining register, MTF, is used for setting the multiplier factor value. When writing to the SPWMH/SPWML register pair and the SFIFOH/SFIFOL register pair, data must first be written to the corresponding low byte register before writing data to the corresponding high byte register.

Register Name	Bit							
	7	6	5	4	3	2	1	0
SPWMH	—	—	—	—	D11	D10	D9	D8
SPWML	D7	D6	D5	D4	D3	D2	D1	D0
SFIFOH	SPWMPOL	—	—	—	SFIFO11	SFIFO10	SFIFO9	SFIFO8
SFIFOL	SFIFO7	SFIFO6	SFIFO5	SFIFO4	SFIFO3	SFIFO2	SFIFO1	SFIFO0
SPWMC0	SPWMT	SPWMON	SEN	HIZ	—	PRTEN	DTYVAL	PRST
SPWMC1	PWMCMP2	PWMCMP1	PWMCMP0	—	BBINV	BUINV	ABINV	AUINV
MTF	D7	D6	D5	D4	D3	D2	D1	D0
SPWMDT	—	—	—	—	DT3	DT2	DT1	DT0

Sine-Wave PWM Generator Registers List

SPWMDT Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	DT3	DT2	DT1	DT0
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0

Bit 7~4 Unimplemented, read as "0"

Bit 3~0 **DT3~DT0**: SPWM Dead Time control bits

Dead Time = $\frac{(DT3 \sim DT0) \times 2}{f_{SYS}}$, where f_{SYS} is the system frequency.

SPWMH, SPWML Registers

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

"—": Unimplemented, read as "0"

D11~D0: Sine-PWM 12-bit period bits

TM output period = $\frac{SPWM\ [11:0] + 1}{f_{SYS}}$, where f_{SYS} is the system frequency and the SPWM [11:0] is the SPWM register content, must be greater than 7.

The Sine-PWM 12-bit TM is driven by the system clock and the output period is determined by these SPWMH/SPWML registers. Users must first write the low byte data into the SPWML register before writing the high byte data into the SPWMH register.

MTF 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

D7~D0: Multiplier factor bits

N = D[7:0]

SFIFOH, SFIFOL Registers

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

SPWMPOL: Sine-wave Polarity indication bit
0: negative half-wave
1: positive half-wave

"—": Unimplemented, read as "0"

D11~D0: Sine-PWM 12-bit duty bits

TM output duty = $\frac{\text{SFIFO [11:0]} \times \text{(MTF [7:0]/256)}}{f_{SYS}}$, where f_{SYS} is the system frequency, the SFIFO[11:0] is the SFIFO register content and the MTF[7:0] is the MTF register content.

The Sine-PWM 12-bit TM is driven by the system clock and the output duty is determined by these SFIFOH/SFIFOL and MTF registers. When write to the SFIFOH/SFIFOL registers, users must first write the low byte data into the SFIFOL register before writing the high byte data into the SFIFOH register. The 12-bit SFIFO register pair is constructed using a 4-level FIFO. Therefore, there must be 4 bytes of data written into the SFIFO registers in each write operation. When the 4th data byte is uploaded into the Sine-PWM TM, the SFIFO controller will generate an interrupt to inform the MCU to write the next 4 bytes of data into the SFIFO register pair.

SPWMC0 Register

Bit	7	6	5	4	3	2	1	0
Name	SPWMT	SPWMON	SEN	HIZ	—	PRTEN	DTYVAL	PRST
R/W	R/W	R/W	R/W	R/W	—	R/W	R/W	R/W
POR	1	0	0	0	—	1	0	0

Bit 7 **SPWMT**: Sine-PWM type selection
0: Bipolar mode
1: Unipolar mode

Bit 6 **SPWMON**: Sine-PWM function enable control
0: Disable
1: Enable

When the bit is cleared to 0, the Sine-PWM 12-bit TM, the Complementary and Dead Time Insertion circuitry will be turned off and the AU'/AB'BU'/BB' signals are forced to a low state. However, this bit has no effect on the Sine-PWM Protecting and Inverting circuitry.

Bit 5 **SEN**: Sine-PWM activation and output enable control
0: Disable
1: Enable

When the bit is cleared to 0, the Sine-PWM complementary signals, AU'/AB'/BU'/BB', will stay in their current state while the status of the complementary output pair, AUO/ABO/BUO/BBO, are determined by the HIZ, PWMCMP0~PWMCMP2, AUINV/ ABINV/ BUINV/ BBINV control bits.

When the bit is set to 1, the Sine-PWM function will be activated and the 12-bit period data bits will be uploaded into the Sine-PWM 12-bit TM followed by the 12-bit duty data bits. Then the period and duty comparators will start to compare the counter value with the duty and period data.

Bit 4 **HIZ**: AUO/ABO/BUO/BBO high impedance control
0: AUO/ABO/BUO/BBO are controlled by the AUINV/ABINV/BUINV/BBINV bits when the SEN bit is cleared to 0
1: AUO/ABO/BUO/BBO are all in a high impedance state when the SEN bit is set high

Bit 3 Unimplemented, read as "0"

Bit 2 **PRTEN**: Sine-PWM protecting control
0: Disable
1: Enable
The protection control avoids AUO&ABO or BUO&BBO to be active level simultaneously. (active level depends on AUINV/ABINV/BUINV/BBINV, if the INV bit is 0, active level is 1; if the INV bit is 1, active level is 0).

Bit 1 **DTYVAL**: decide how duty value gets
0: Duty comes from the calculating result of SFIFO[11:0] × N/256 (N=MTF register value)
1: Duty comes directly from SFIFO[11:0] value without × N/256

Bit 0 **PRST**: FIFO read/write point clear control bit
0: No action
1: Clear read/write point (RP/WP) as 0

SPWMC1 Register

Bit	7	6	5	4	3	2	1	0
Name	PWMCMP2	PWMCMP1	PWMCMP0	—	BBINV	BUINV	ABINV	AUINV
R/W	R/W	R/W	R/W	—	R/W	R/W	R/W	R/W
POR	1	1	1	—	0	0	0	0

Bit 7 **PWMCMP2**: Enable to set AU, AB, BU, & BB as low immediately once OVPO is high
0: Disable
1: Enable

Bit 6 **PWMCMP1**: Enable to set AU, AB, BU, & BB as low once OCPO_1 is high
0: Disable
1: Enable

Bit 5 **PWMCMP0**: Enable to set AU, AB, BU, & BB as low once OCPO_0 is high
0: Disable
1: Enable

Bit 4 Unimplemented, read as "0"

Bit 3 **BBINV**: BBO invert control
0: Non-Inverted
1: Inverted

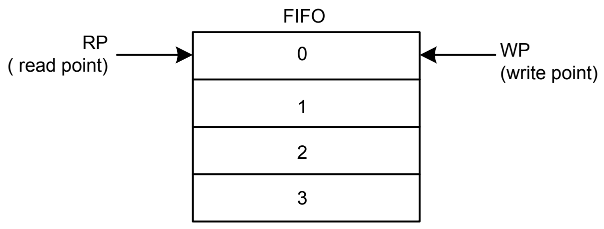
Bit 2 **BUINV**: BUO invert control
0: Non-Inverted
1: Inverted

Bit 1 **ABINV**: ABO invert control
0: Non-Inverted
1: Inverted

Bit 0 **AUINV**: AUO invert control
0: Non-Inverted
1: Inverted

FIFO Operation

The SFIFO register pair is organized into 12 bits by 4 levels and written-only. The data written into the SFIFOH/SFIFOL registers must have 4 bytes for each write operation subroutine. The 4 bytes of data will be sequentially uploaded into the duty shadow register in the SPWM 12-bit TM whenever a TM output period is complete. When the 4th byte data in the SFIFO register pair is loaded into the SPWM 12-bit TM duty shadow register, the SFIFO controller will generate an interrupt to inform the microcontroller that the next 4 bytes of data should be written into the SFIFO register pair.



4-Level FIFO

Note: 1. RP increases by 1 If FIFO is read after SPWM period overflow (RP=0→1→2→3→0→1→2→3 …)

2. No matter SPWMON=0/1, WP increases by 1 If FIFO is written (WP=0→1→2→3→0→1→2→3 …)

3. SPWM Interrupt occurs once the 4th data (RP=3) has been read out

4. PRST=1 to clear RP&WP=0

5. 8 system clocks time is bided for multiplier option, every time FIFO data is read out

FIFO operation for suggestion:

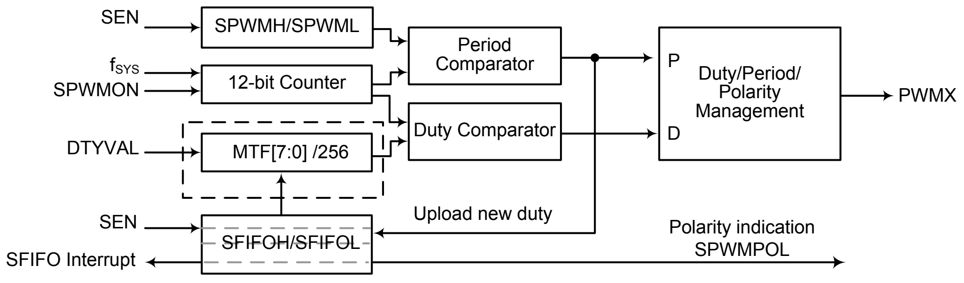
0. SPWMON=0
1. Initial point: PRST=1, then PRST=0 (RP=0, WP=0)
2. Write 4 byte data Into FIFO (WP=0 after 4 bytes of data written with sequence of 0, 1, 2, 3, 0)
3. SPWMON=1
4. Wait for SPWM Interrupt & write 4 byte data after SPWM Interrupt occurs
5. Go to step 4 until SPWMON=0

Also note that:

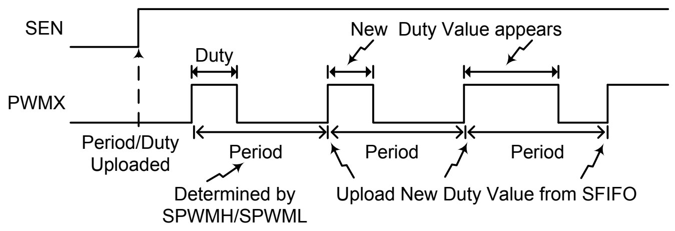
1. After the duty data is read, the data will be updated immediately once the next period starts.
2. SPWM duty Multiplier operation requires at least 8 clocks, so the period must be greater than 10 system clocks.

Sine-PWM TM Operation

The 12-bit Sine-PWM TM is driven by the system clock and can generate a PWM signal, PWMX, with a variable duty and period cycles by configuring the 12-bit SFIFOH/SFIFOL and SPWMH/SPWML registers. The PWMX signal period is depended upon the sine wave sampling rate and period is determined by the SPWMH and SPWML register pair. The PWMX signal duty is determined by the SFIFOH/SFIFOL registers content. There is also an 8-bit multiplier, the multiplier factor determined by the MTF register, can be used for adjusting the PWM signal's duty cycle, thereby adjusting the output sine-wave's amplitude. So the duty value can be selected from the calculating result of SFIFO[11:0]×N/256 or from directly from SFIFO[11:0] values without ×N/256. There is also a SPWMPOL bit for the Sine-wave Polarity indication.



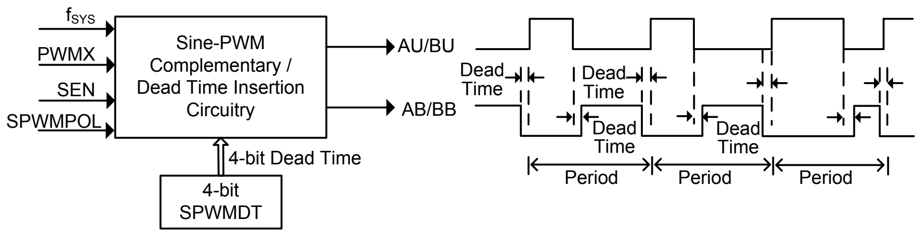
12-bit SPWM TM Block Diagram



12-bit SPWM Waveform

Complementary Operation and Dead Time Insertion

The complementary control circuitry can derive the complementary PWM output pair signals from the PWMX signal. After the complementary PWM output pair signals, AU and AB or BU and BB, are generated, the dead time can be inserted to avoid the complementary signals from both being in an active state. The dead time duration is determined using the 4-bit SPWMDT register.

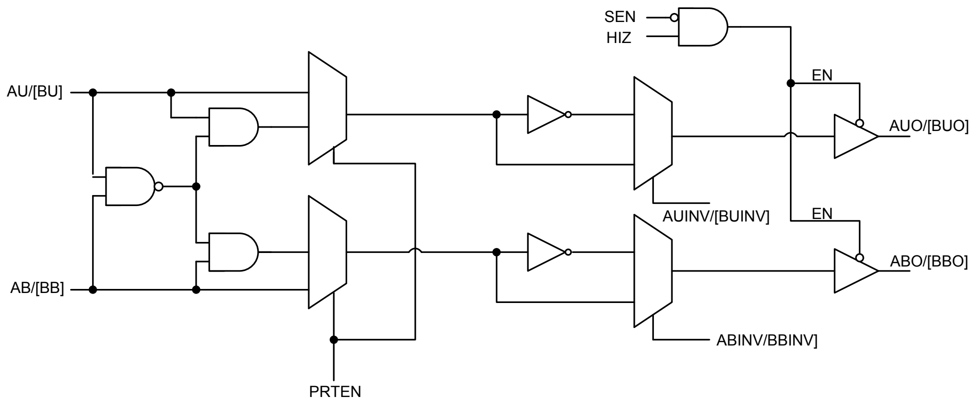


Sine-PWM Complementary/Dead Time Insertion Diagram

Protection and Inverting Control

Although a dead time has been inserted into the Sine-PWM complementary pair signals, these two signals may both be in a high state resulting from the some unpredictable reasons, such as software malfunctions or electrical noise. The device provides a protection function to force the two signals to be in a low state when the AU and AB or BU and BB signals both are in an active state. The inverting control circuitry determines whether the signals are inverted or not using corresponding inverting control bits.

The device has over current protection and over voltage protection functions which are described in the OCP and OVP sections. Once an Over-current or Over-voltage condition occurs, the OVPO, OCPO_1 or OCPO_0 will be set high, immediately the AU, AB, BU and BB signals will be low.



Note: AU/AB or BU/BB use the same circuit as above.

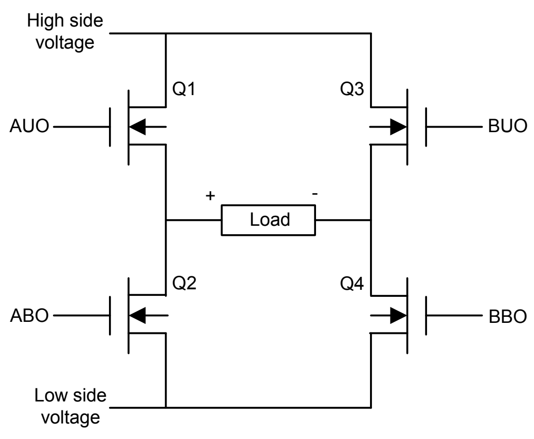
Sine-PWM Protection and Inverter Control Diagram

Sine-PWM Drive Control Mode

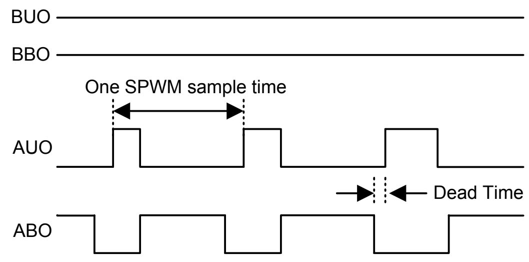
This device provides unipolar and bipolar SPWM drive control mode. Users can do appropriate signal switching by setting the corresponding registers.

	Unipolar Mode, SPWMT=1		Bipolar Mode, SPWMT=0	
	Positive half-cycle SPWMPOL=1	**Negative half-cycle SPWMPOL=0**	**Positive half-cycle SPWMPOL=1**	**Negative half-cycle SPWMPOL=0**
AU	PWM	Low	PWM	$\overline{\text{AB}}$ with Dead Time
AB	$\overline{\text{AU}}$ with Dead Time	High	$\overline{\text{AU}}$ with Dead Time	PWM
BU	Low	PWM	Disable	Disable
BB	High	$\overline{\text{BU}}$ with DT	Disable	Disable

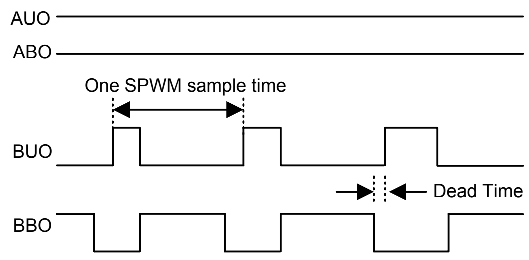
Unipolar SPWM drive control circuitry



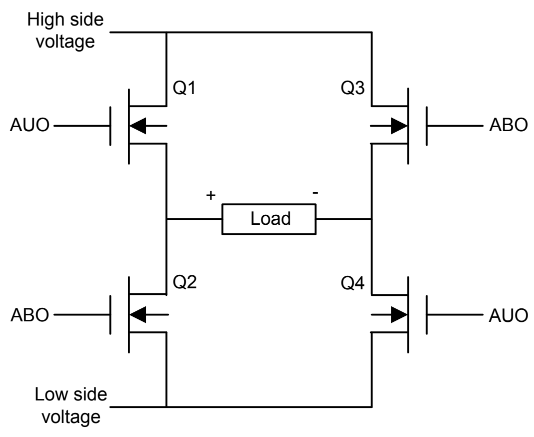
Unipolar mode, positive half-cycle



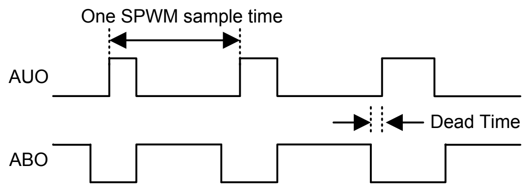
Unipolar mode, negative half-cycle



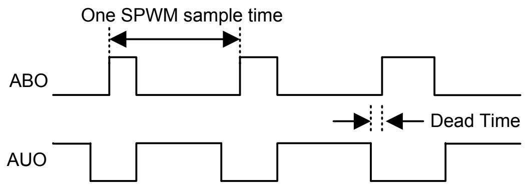
Bipolar SPWM drive control circuitry



Bipolar mode, positive half-cycle



Bipolar mode, negative half-cycle



UART Module Serial Interface

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

The integrated UART function contains the following features:

- Full-duplex, Universal Asynchronous Receiver and Transmitter (UART) communication
- 8 or 9 bits character length
- Even, odd or no parity options
- One or two stop bits
- Baud rate generator with 8-bit prescaler
- Parity, framing, noise and overrun error detection
- Support for interrupt on address detect (last character bit=1)
- Transmitter and receiver enabled independently
- 2-byte Deep FIFO Receive Data Buffer
- Transmit and receive interrupts
- Transmit and Receive Multiple Interrupt Generation Sources
  - Transmitter Empty
  - Transmitter Idle
  - Receiver Full
  - Receiver Overrun
  - Address Mode Detect
  - RX pin Wake-up

UART External Interface

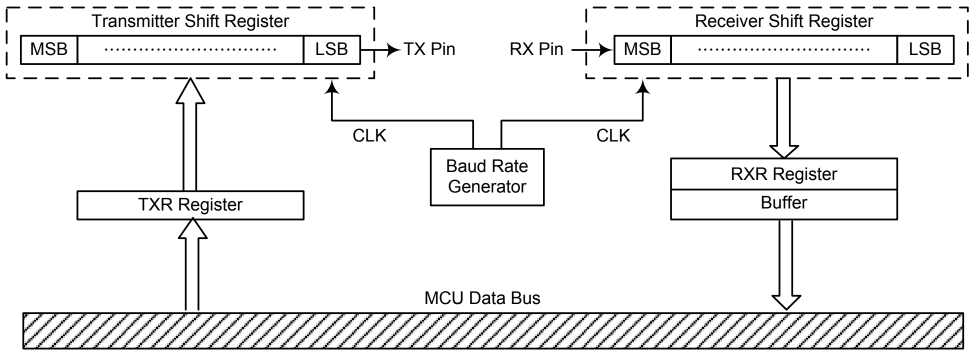
To communicate with an external serial interface, the internal UART has two external pins known as TX and RX. The TX pin is the UART transmitter pin, which can be used as a general purpose I/O or other pin-shared functional pin if the pin is not configured as a UART transmitter, which occurs when the TXEN bit in the UCR2 control register is equal to zero. Similarly, the RX pin is the UART receiver pin, which can also be used as a general purpose I/O or other pin-shared functional pin, if the pin is not configured as a receiver, which occurs if the RXEN bit in the UCR2 register is equal to zero. Along with the UARTEN bit, the TXEN and RXEN bits, if set, will automatically setup these I/O or other pin-shared functional pins to their respective TX output and RX input conditions and disable any pull-high resistor option which may exist on the RX pin.

UART Data Transfer Scheme

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

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

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



UART Data Transfer Scheme

UART Status and Control Registers

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

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
USR	PERR	NF	FERR	OERR	RIDLE	RXIF	TIDLE	TXIF
UCR1	UARTEN	BNO	PREN	PRT	STOPS	TXBRK	RX8	TX8
UCR2	TXEN	RXEN	BRGH	ADDEN	WAKE	RIE	TIIE	TEIE
TXR_RXR	TXRX7	TXRX6	TXRX5	TXRX4	TXRX3	TXRX2	TXRX1	TXRX0
BRG	BRG7	BRG6	BRG5	BRG4	BRG3	BRG2	BRG1	BRG0

UART Register Summary

USR register

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

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

Bit 7 **PERR**: Parity error flag
0: No parity error is detected
1: Parity error is detected
The PERR flag is the parity error flag. When this read only flag is “0”, it indicates a parity error has not been detected. When the flag is “1”, it indicates that the parity of the received word is incorrect. This error flag is applicable only if Parity mode (odd or even) is selected. The flag can also be cleared by a software sequence which involves a read to the status register USR followed by an access to the RXR data register.

Bit 6 **NF**: Noise flag
0: No noise is detected
1: Noise is detected
The NF flag is the noise flag. When this read only flag is "0", it indicates no noise condition. When the flag is "1", it indicates that the UART has detected noise on the receiver input. The NF flag is set during the same cycle as the RXIF flag but will not be set in the case of as overrun. The NF flag can be cleared by a software sequence which will involve a read to the status register USR followed by an access to the RXR data register.

Bit 5 **FERR**: Framing error flag
0: No framing error is detected
1: Framing error is detected
The FERR flag is the framing error flag. When this read only flag is “0”, it indicates that there is no framing error. When the flag is “1”, it indicates that a framing error has been detected for the current character. The flag can also be cleared by a software sequence which will involve a read to the status register USR followed by an access to the RXR data register.

Bit 4 **OERR**: Overrun error flag
0: No overrun error is detected
1: Overrun error is detected
The OERR flag is the overrun error flag which indicates when the receiver buffer has overflowed. When this read only flag is “0”, it indicates that there is no overrun error. When the flag is “1”, it indicates that an overrun error occurs which will inhibit further transfers to the RXR receive data register. The flag is cleared by a software sequence, which is a read to the status register USR followed by an access to the RXR data register.

Bit 3 **RIDLE**: Receiver status
0: Data reception is in progress (data being received)
1: No data reception is in progress (receiver is idle)
The RIDLE flag is the receiver status flag. When this read only flag is “0”, it indicates that the receiver is between the initial detection of the start bit and the completion of the stop bit. When the flag is “1”, it indicates that the receiver is idle. Between the completion of the stop bit and the detection of the next start bit, the RIDLE bit is “1” indicating that the UART receiver is idle and the RX pin stays in logic high condition.

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

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

Bit 1 **TIDLE**: Transmission idle
0: Data transmission is in progress (data being transmitted)
1: No data transmission is in progress (transmitter is idle)

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

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

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

UCR1 register

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

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

"x" unknown

Bit 7 **UARTEN**: UART function enable control
0: Disable UART. TX and RX pins are as I/O or other pin-shared functional pins
1: Enable UART. TX and RX pins function as UART pins

The UARTEN bit is the UART enable bit. When this bit is equal to "0", the UART will be disabled and the RX pin as well as the TX pin will be as General Purpose I/O or other pin-shared functional pins. When the bit is equal to "1", the UART will be enabled and the TX and RX pins will function as defined by the TXEN and RXEN enable control bits.

When the UART is disabled, it will empty the buffer so any character remaining in the buffer will be discarded. In addition, the value of the baud rate counter will be reset. If the UART is disabled, all error and status flags will be reset. Also the TXEN, RXEN, TXBRK, RXIF, OERR, FERR, PERR and NF bits will be cleared, while the TIDLE, TXIF and RIDLE bits will be set. Other control bits in UCR1, UCR2 and BRG registers will remain unaffected. If the UART is active and the UARTEN bit is cleared, all pending transmissions and receptions will be terminated and the module will be reset as defined above. When the UART is re-enabled, it will restart in the same configuration.

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

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

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

This is the parity enable bit. When this bit is equal to "1", the parity function will be enabled. If the bit is equal to "0", then the parity function will be disabled. Replace the most significant bit position with a parity bit.

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

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

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

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

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

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

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

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

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

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

UCR2 register

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

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

Bit 7 **TXEN**: UART Transmitter enabled control
0: UART transmitter is disabled
1: UART transmitter is enabled

The bit named TXEN is the Transmitter Enable Bit. When this bit is equal to "0", the transmitter will be disabled with any pending data transmissions being aborted. In addition the buffers will be reset. In this situation the TX pin will be used as an I/O or other pin-shared functional pin.

If the TXEN bit is equal to "1" and the UARTEN bit is also equal to "1", the transmitter will be enabled and the TX pin will be controlled by the UART. Clearing the TXEN bit during a transmission will cause the data transmission to be aborted and will reset the transmitter. If this situation occurs, the TX pin will be used as an I/O or other pin-shared functional pin.

Bit 6 **RXEN**: UART Receiver enabled control
0: UART receiver is disabled
1: UART receiver is enabled

The bit named RXEN is the Receiver Enable Bit. When this bit is equal to "0", the receiver will be disabled with any pending data receptions being aborted. In addition the receive buffers will be reset. In this situation the RX pin will be used as an I/O or other pin-shared functional pin. If the RXEN bit is equal to "1" and the UARTEN bit is also equal to "1", the receiver will be enabled and the RX pin will be controlled by the UART. Clearing the RXEN bit during a reception will cause the data reception to be aborted and will reset the receiver. If this situation occurs, the RX pin will be used as an I/O or other pin-shared functional pin.

Bit 5 **BRGH**: Baud Rate speed selection
0: Low speed baud rate
1: High speed baud rate

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

Bit 4 **ADDEN**: Address detect function enable control
0: Address detect function is disabled
1: Address detect function is enabled

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

Bit 3 **WAKE**: RX pin falling edge wake-up function enable control
0: RX pin wake-up function is disabled
1: RX pin wake-up function is enabled

This bit enables or disables the receiver wake-up function. If this bit is equal to "1" and the MCU is in IDLE or SLEEP mode, a falling edge on the RX input pin will wake-up the device. Please reference the UART RX pin wake-up functions in different operating mode for the detail. If this bit is equal to "0" and the MCU is in IDLE or SLEEP mode, any edge transitions on the RX pin will not wake-up the device.

Bit 2 **RIE**: Receiver interrupt enable control
0: Receiver related interrupt is disabled
1: Receiver related interrupt is enabled

This bit enables or disables the receiver interrupt. If this bit is equal to "1" and when the receiver overrun flag OERR or receive data available flag RXIF is set, the UART interrupt request flag will be set. If this bit is equal to "0", the UART interrupt request flag will not be influenced by the condition of the OERR or RXIF flags.

Bit 1 **TIIE**: Transmitter Idle interrupt enable control
0: Transmitter idle interrupt is disabled
1: Transmitter idle interrupt is enabled

This bit enables or disables the transmitter idle interrupt. If this bit is equal to "1" and when the transmitter idle flag TIDLE is set, due to a transmitter idle condition, the UART interrupt request flag will be set. If this bit is equal to "0", the UART interrupt request flag will not be influenced by the condition of the TIDLE flag.

Bit 0 **TEIE**: Transmitter Empty interrupt enable control
0: Transmitter empty interrupt is disabled
1: Transmitter empty interrupt is enabled

This bit enables or disables the transmitter empty interrupt. If this bit is equal to "1" and when the transmitter empty flag TXIF is set, due to a transmitter empty condition, the UART interrupt request flag will be set. If this bit is equal to "0", the UART interrupt request flag will not be influenced by the condition of the TXIF flag.

TXR_RXR register

Bit	7	6	5	4	3	2	1	0
Name	TXRX7	TXRX6	TXRX5	TXRX4	TXRX3	TXRX2	TXRX1	TXRX0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	x	x	x	x	x	x	x	x

"x" unknown

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

BRG Register

Bit	7	6	5	4	3	2	1	0
Name	BRG7	BRG6	BRG5	BRG4	BRG3	BRG2	BRG1	BRG0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	x	x	x	x	x	x	x	x

"x" unknown

Bit 7~0 **BRG7~BRG0**: Baud Rate values

By programming the BRGH bit in UCR2 Register which allows selection of the related formula described above and programming the required value in the BRG register, the required baud rate can be setup.

Note: Baud rate= $f_{SYS}/[64\times(N+1)]$ if BRGH=0.

Baud rate= $f_{SYS}/[16\times(N+1)]$ if BRGH=1.

Baud Rate Generator

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

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

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

Calculating the register and error values

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

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

Re-arranging this equation gives N = [f_{SYS} / (BR×64)] - 1

Giving a value for N = [4000000 / (4800×64)] - 1 = 12.0208

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

Therefore the error is equal to (4808 - 4800) / 4800 = 0.16%

The following tables show actual values of baud rate and error values for the two values of BRGH.

Baud Rate K/BPS	Baud Rates for BRGH=0					
	f_{SYS} = 16 MHz			f_{SYS} = 20 MHz		
	BRG	Kbaud	Error (%)	BRG	Kbaud	Error (%)
0.3	—	—	—	—	—	—
1.2	207	1.202	0.16	259	1.202	0.16
2.4	103	2.404	0.16	129	2.404	0.16
4.8	51	4.808	0.16	64	4.808	0.16
9.6	25	9.615	0.16	32	9.470	-1.36
19.2	12	19.231	0.16	15	19.531	1.73
38.4	6	35.714	-6.99	7	39.063	1.73
57.6	3	62.5	8.51	4	62.5	8.51
115.2	1	125	8.51	2	104.17	-9.58
250	0	250	0	0	312.5	25

Baud Rates and Error Values for BRGH = 0

Baud Rate K/BPS	Baud Rates for BRGH=1					
	f_{SYS} = 16 MHz			f_{SYS} = 20 MHz		
	BRG	Kbaud	Error (%)	BRG	Kbaud	Error (%)
0.3	—	—	—	—	—	—
1.2	—	—	—	—	—	—
2.4	—	—	—	—	—	—
4.8	207	4.808	0.16	—	—	—
9.6	103	9.615	0.16	129	9.615	0.16
19.2	51	19.231	0.16	64	19.231	0.16
38.4	25	38.462	0.16	32	37.879	-1.36
57.6	16	58.824	2.12	21	56.818	-1.35
115.2	8	111.11	-3.55	10	113.636	-1.36
250	3	250	0	4	250	0

Baud Rates and Error Values for BRGH = 1

UART Setup and Control

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

Enabling/disabling the UART

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

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

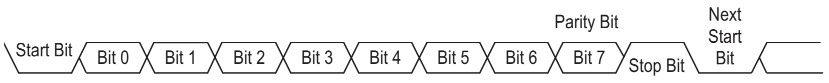
Data, parity and stop bit selection

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

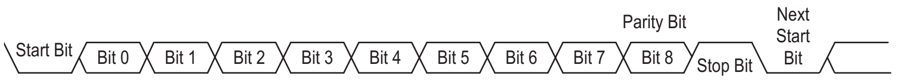
Start Bit	Data Bits	Address Bits	Parity Bits	Stop Bit
Example of 8-bit Data Formats				
1	8	0	0	1
1	7	0	1	1
1	7	1	0	1
Example of 9-bit Data Formats				
1	9	0	0	1
1	8	0	1	1
1	8	1	0	1

Transmitter Receiver Data Format

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



8-Bit Data Format



9-Bit Data Format

UART transmitter

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

Transmitting data

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

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

It should be noted that when TXIF=0, data will be inhibited from being written to the TXR register. Clearing the TXIF flag is always achieved using the following software sequence:

1. A USR register access
2. A TXR register write execution

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

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

1. A USR register access
2. A TXR register write execution

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

Transmit break

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

UART receiver

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

Receiving data

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

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

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

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

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

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

1. A USR register access
2. An RXR register read execution

Receive break

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

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

Idle status

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

Receiver interrupt

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

Managing receiver errors

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

Overrun Error – OERR flag

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

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

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

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

Noise Error – NF Flag

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

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

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

Framing Error – FERR Flag

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

Parity Error – PERR Flag

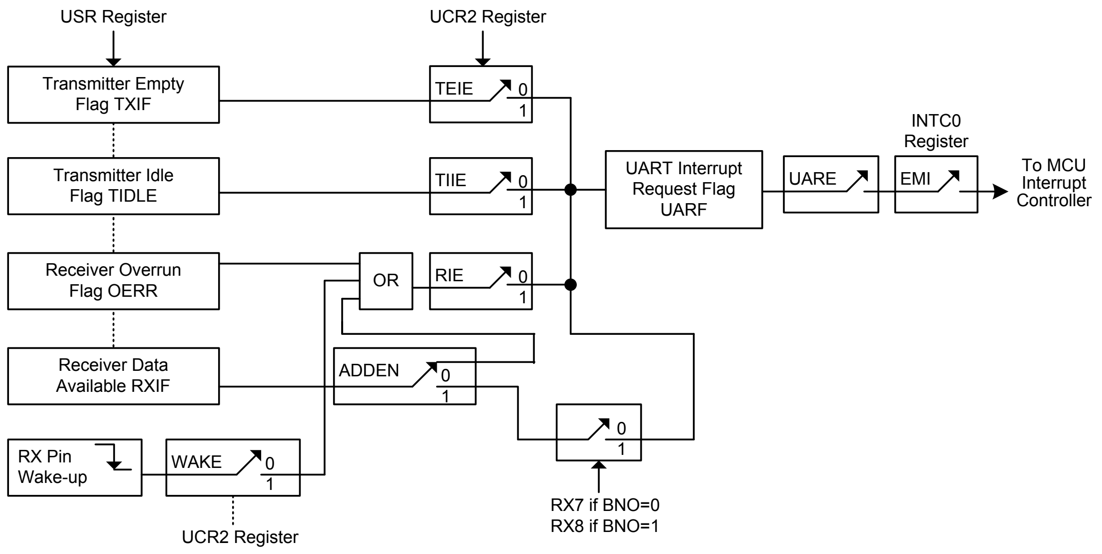
The read only parity error flag, PERR, in the USR register, is set if the parity of the received word is incorrect. This error flag is only applicable if the parity is enabled, PREN = 1, and if the parity type, odd or even is selected. The read only PERR flag is buffered along with the received data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

UART Module Interrupt Structure

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

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

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


USR Register
UCR2 Register
Transmitter Empty Flag TXIF
Transmitter Idle Flag TIDLE
Receiver Overrun Flag OERR
Receiver Data Available RXIF
RX Pin Wake-up
WAKE
UCR2 Register
OR
TEIE
TIIE
RIE
ADDEN
0
1
RX7 if BNO=0
RX8 if BNO=1
UART Interrupt Request Flag UARF
UARE
INTC0 Register
EMI
To MCU Interrupt Controller

UART Interrupt Scheme

Address detect mode

Setting the Address Detect Mode bit, ADDEN, in the UCR2 register, enables this special mode. If this bit is enabled then an additional qualifier will be placed on the generation of a Receiver Data Available interrupt, which is requested by the RXIF flag. If the ADDEN bit is enabled, then when data is available, an interrupt will only be generated, if the highest received bit has a high value. Note that the MFE, URE and EMI interrupt enable bits must also be enabled for correct interrupt generation. This highest address bit is the 9th bit if BNO=1 or the 8th bit if BNO=0. If this bit is high, then the received word will be defined as an address rather than data. A Data Available interrupt will be generated every time the last bit of the received word is set. If the ADDEN bit is not enabled, then a Receiver Data Available interrupt will be generated each time the RXIF flag is set, irrespective of the data last bit status. The address detect mode and parity enable are mutually exclusive functions. Therefore if the address detect mode is enabled, then to ensure correct operation, the parity function should be disabled by resetting the parity enable bit to zero.

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

ADDEN Bit Function

UART Module Power Down and Wake-up

When the f_{SYS} is off, the UART will cease to function. All clock sources to the module are shutdown. If the f_{SYS} is off while a transmission is still in progress, then the transmission will be paused until the UART clock source derived from the microcontroller is activated. In a similar way, if the MCU enters the Power Down Mode while receiving data, then the reception of data will likewise be paused. When the MCU enters the Power Down Mode, note that the USR, UCR1, UCR2, transmit and receive registers, as well as the BRG register will not be affected. It is recommended to make sure first that the UART data transmission or reception has been finished before the microcontroller enters the Power Down mode.

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

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

Below table illustrates the UART RX wake-up functions in different operating mode.

Operation Mode	Description			RX wake-up function
	CPU	f_H	f_{SUB}	
IDLE0 Mode	Off(HALT)	Off	On	When the UCR2.2(RIE)=1, UCR2.3(WAKE)=1 and the CPU is entered in IDLE0 mode, a falling edge on the RX pin will turn on the f_{SYS} if the f_{SYS} is off in this mode, and wake-up CPU.
IDLE1 Mode	Off(HALT)	On	On	When the UCR2.2(RIE)=1, UCR2.3(WAKE)=1 and the CPU is entered in IDLE1 mode: 1. If the UART is not transfer and a falling edge occurred on the RX pin, the f_{SYS} is kept on running and CPU is still off. If the UART transmission is on going, CPU will wake up in the end of transfer. 2. If the UART transmission is on going, the CPU will wake up in the end of transfer. Note: If RIE=0, WAKE=1 and the UART transmission is on going, the CPU will not wake up in the end of receive.
IDLE2 Mode	Off(HALT)	On	On	When the UCR2.2(RIE)=1, UCR2.3(WAKE)=1 and the CPU is entered in IDLE1 mode: 1. If the UART is not transfer and a falling edge occurred on the RX pin, this will turn on f_{SYS} if the f_{SYS} is off in this mode and CPU is still off. If the UART transmission is on going, CPU will wake up in the end of transfer. 2. If the UART transmission is on going, the CPU will wake up in the end of transfer. Note: If RIE=0, WAKE=1 and the UART transmission is on going, the CPU will not wake up in the end of receive.
SLEEP Mode	Off	Off	On	When the UCR2.2(RIE)=1, UCR2.3(WAKE)=1 and the CPU is entered in SLEEP mode, a falling edge on the RX pin will turn on f_{SYS} and wake-up CPU.

Interrupts

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

Interrupt Registers

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

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

Function	Enable Bit	Request Flag	Notes
Global	EMI	—	—
INT0 Pin	INT0E	INT0F	—
AC Detector	ZXF	ZXE	—
Sine-PWM Generator	SFEF	SFEE	—
OCPn	OCPnF	OCPnE	n=0 or 1
OVP	OVPE	OVPF	—
A/D Converter	ADE	ADF	—
UART	UARE	UARF	—
Multi-function	MFnE	MFnF	n=0~2
Time Base	TBnE	TBnF	n=0 or 1
LVD	LVE	LVF	—
EEPROM	DEE	DEF	—
TM	TnPE	TnPF	n=0~2
	TnAE	TnAF	n=0~2

Interrupt Register Bit Naming Conventions

Register Name	Bit							
	7	6	5	4	3	2	1	0
INTEG	—	—	—	—	ZXS1	ZXS0	INT0S1	INT0S0
INTC0	—	OCP0F	SFEF	ZXF	OCP0E	SFEE	ZXE	EMI
INTC1	ADF	INT0F	OVPF	OCP1F	ADE	INT0E	OVPE	OCP1E
INTC2	MF2F	MF1F	MF0F	UARF	MF2E	MF1E	MF0E	UARE
MFI0	T1AF	T1PF	T0AF	T0PF	T1AE	T1PE	T0AE	T0PE
MFI1	—	—	T2AF	T2PF	—	—	T2AE	T2PE
MFI2	TB1F	TB0F	DEF	LVF	TB1E	TB0E	DEE	LVE

Interrupt Register Contents

INTEG Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	ZXS1	ZXS0	INT0S1	INT0S0
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0

Bit 7~4 Unimplemented, read as "0"

Bit 3~2 **ZXS1~ZXS0**: Interrupt edge control for AC zero-cross detect
00: Disable
01: Rising edge
10: Falling edge
11: Both rising and falling edges

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

INTC0 Register

Bit	7	6	5	4	3	2	1	0
Name	—	OCP0F	SFEF	ZXF	OCP0E	SFEE	ZXE	EMI
R/W	—	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	0	0	0	0	0	0	0

Bit 7 Unimplemented, read as “0”

Bit 6 **OCP0F**: OCP0 Interrupt Request Flag
0: No request
1: Interrupt request

Bit 5 **SFEF**: SPWM FIFO empty Interrupt Request Flag
0: No request
1: Interrupt request

Bit 4 **ZXF**: AC zero detection Interrupt Request Flag
0: No request
1: Interrupt request

Bit 3 **OCP0E**: OCP0 Interrupt Control
0: Disable
1: Enable

Bit 2 **SFEE**: SPWM FIFO Empty Interrupt Control
0: Disable
1: Enable

Bit 1 **ZXE**: AC zero detection Interrupt Control
0: Disable
1: Enable

Bit 0 **EMI**: Global Interrupt Control
0: Disable
1: Enable

INTC1 Register

Bit	7	6	5	4	3	2	1	0
Name	ADF	INT0F	OVPF	OCP1F	ADE	INT0E	OVPE	OCP1E
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 **ADF**: A/D Converter Interrupt Request Flag
0: No request
1: Interrupt request

Bit 6 **INT0F**: External Interrupt 0 Request Flag
0: No request
1: Interrupt request

Bit 5 **OVPF**: Over Voltage Protect Interrupt Request Flag
0: No request
1: Interrupt request

Bit 4 **OCP1F**: OCP1 Interrupt Request Flag
0: No request
1: Interrupt request

Bit 3 **ADE**: A/D Converter Interrupt Control
0: Disable
1: Enable

Bit 2 **INT0E**: External Interrupt 0 Control
0: Disable
1: Enable

Bit 1 **OVPE**: Over Voltage Protection Interrupt Control
0: Disable
1: Enable

Bit 0 **OCP1E**: OCP1 Interrupt Control
0: Disable
1: Enable

INTC2 Register

Bit	7	6	5	4	3	2	1	0
Name	MF2F	MF1F	MF0F	UARF	MF2E	MF1E	MF0E	UARE
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 **MF2F**: Multi-function Interrupt 2 Request Flag
0: No request
1: Interrupt request

Bit 6 **MF1F**: Multi-function Interrupt 1 Request Flag
0: No request
1: Interrupt request

Bit 5 **MF0F**: Multi-function Interrupt 0 Request Flag
0: No request
1: Interrupt request

Bit 4 **UARF**: UART interrupt request flag
0: No request
1: Interrupt request

Bit 3 **MF2E**: Multi-function 2 Interrupt Control
0: Disable
1: Enable

Bit 2 **MF1E**: Multi-function 1 Interrupt Control
0: Disable
1: Enable

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

Bit 0 **UARE**: UART interrupt control
0: Disable
1: Enable

MFI0 Register

Bit	7	6	5	4	3	2	1	0
Name	T1AF	T1PF	T0AF	T0PF	T1AE	T1PE	T0AE	T0PE
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 **T1AF**: TM1 Comparator A match interrupt request flag
0: No request
1: Interrupt request

Bit 6 **T1PF**: TM1 Comparator P match interrupt request flag
0: No request
1: Interrupt request

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

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

Bit 3 **T1AE**: TM1 Comparator A match interrupt control
0: Disable
1: Enable

Bit 2 **T1PE**: TM1 Comparator P match interrupt control
0: Disable
1: Enable

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

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

MFI1 Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	T2AF	T2PF	—	—	T2AE	T2PE
R/W	—	—	R/W	R/W	—	—	R/W	R/W
POR	—	—	0	0	—	—	0	0

Bit 7~6 Unimplemented, read as “0”

Bit 5 **T2AF**: TM2 Comparator A match interrupt request flag
0: No request
1: Interrupt request

Bit 4 **T2PF**: TM2 Comparator P match interrupt request flag
0: No request
1: Interrupt request

Bit 3~2 Unimplemented, read as “0”

Bit 1 **T2AE**: TM2 Comparator A match interrupt control
0: Disable
1: Enable

Bit 0 **T2PE**: TM2 Comparator P match interrupt control
0: Disable
1: Enable

MFI2 Register

Bit	7	6	5	4	3	2	1	0
Name	TB1F	TB0F	DEF	LVF	TB1E	TB0E	DEE	LVE
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 **TB1F**: Time Base 1 Interrupt Request Flag
0: No request
1: Interrupt request

Bit 6 **TB0F**: Time Base 0 Interrupt Request Flag
0: No request
1: Interrupt request

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

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

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

Bit 2 **TB0E**: Time Base 0 Interrupt Control
0: Disable
1: Enable

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

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

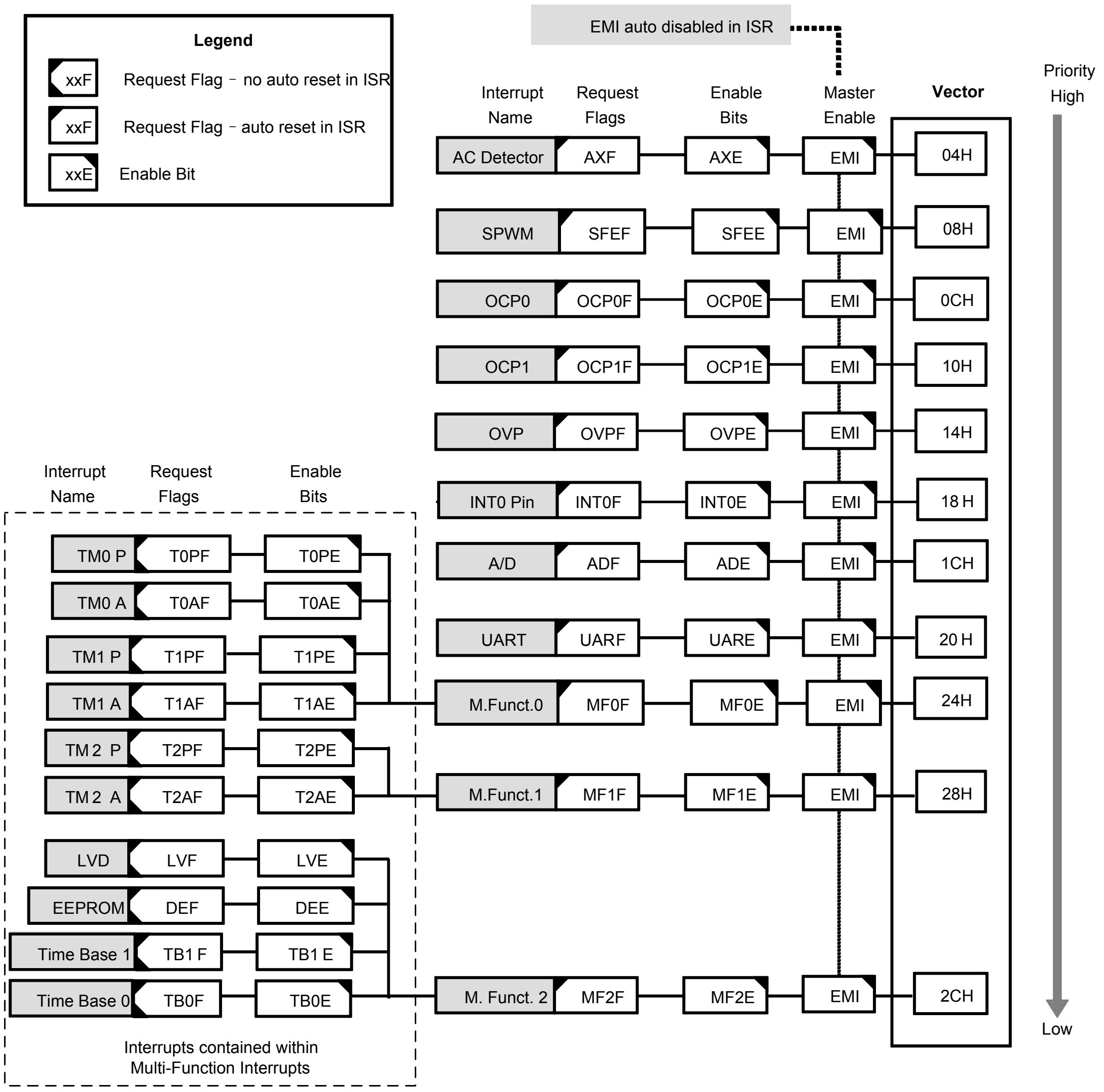
Interrupt Operation

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

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

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

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



Interrupt Structure

External Interrupt

The external interrupt is controlled by signal transitions on the INT0 pins. An external interrupt request will take place when the external interrupt request flag, INT0F, 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, INT0E, must first be set. Additionally the correct interrupt edge type must be selected using the INTEG register to enable the external interrupt function and to choose the trigger edge type. As the external interrupt pin is pin-shared with I/O pin, it can only be configured as external interrupt pin if the external interrupt enable bit in the corresponding interrupt register has been set. The pin must also be setup as an input by setting the corresponding bit in the port control register. When the interrupt is enabled, the stack is not full and the correct transition type appears on the external interrupt pin, a subroutine call to the external interrupt vector will take place. When the interrupt is serviced, the external interrupt request flag, INT0F, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts. Note that any pull-high resistor selections on the external interrupt pin will remain valid even if the pin is used as an external interrupt input.

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

Multi-function Interrupt

Within these devices there are up to three Multi-function interrupts. Unlike the other independent interrupts, these interrupts have no independent source, but rather are formed from other existing interrupt sources, namely the TM Interrupts, LVD interrupt, EEPROM interrupt and Time Base interrupt.

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

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

A/D Converter Interrupt

The A/D Converter Interrupt is 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.

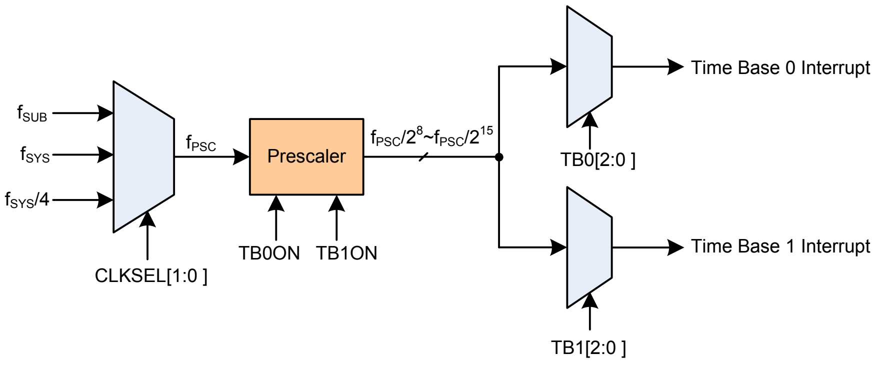
UART Interrupt

Several individual UART conditions can generate a UART interrupt. When these conditions exist, a low pulse will be generated to get the attention of the microcontroller. These conditions are a transmitter data register empty, transmitter idle, receiver data available, receiver overrun, address detect and an RX pin wake-up. To allow the program to branch to the respective interrupt vector addresses, the global interrupt enable bit, EMI, and UART interrupt enable bit, UARE, must first be set. When the interrupt is enabled, the stack is not full and any of these conditions are created, a subroutine call to the UART Interrupt vector will take place. When the interrupt is serviced, the UART Interrupt flag, UARF, will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts. However, the USR register flags will be cleared automatically when certain actions are taken by the UART, the details of which are given in the UART section.

Time Base Interrupt

The function of the Time Base Interrupt is to provide regular time signal in the form of an internal interrupt. They are controlled by the overflow signals from their respective internal timer functions. When these happen their respective interrupt request flag, TBnF, will be set. To allow the program to branch to its respective interrupt vector addresses, the global interrupt enable bit, EMI and Time Base enable bit, TBnE, and associated Multi-function interrupt enable bit must first be set. When the interrupt is enabled, the stack is not full and the Time Base overflows, a subroutine call to the Multi-function interrupt vector location will take place. When the interrupt is serviced, the EMI bit will be automatically cleared to disable other interrupts, however only the Multi-function interrupt request flag will be also automatically cleared. As the TBnF flag will not be automatically cleared, it has to be cleared by the application program.

The purpose of the Time Base Interrupt is to provide an interrupt signal at fixed time periods. Their clock source originates from the internal clock source $f_{SYS}/4$, f_{SYS} or f_{SUB} and then passes through a divider, the division ratio of which is selected by programming the appropriate bits in the TBC0 and TBC1 registers to obtain longer interrupt periods whose value ranges. The clock source which in turn controls the Time Base interrupt period is selected using the CLKSEL1 and CLKSEL0 bits in the PSCR register.



Time Base Interrupt

PSCR Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	CLKSEL1	CLKSEL0
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2 Unimplemented, read as “0”

Bit 1~0 **CLKSEL1~CLKSEL0**: Prescaler clock source selection – f_{PSC}
00: f_{SYS}
01: $f_{SYS}/4$
1x: f_{SUB}

TBC0 Register

Bit	7	6	5	4	3	2	1	0
Name	TB0EN	—	—	—	—	TB02	TB01	TB00
R/W	R/W	—	—	—	—	R/W	R/W	R/W
POR	0	—	—	—	—	0	0	0

Bit 7 **TB0EN**: Time Base 0 Enable/Disable Control
0: Disable
1: Enable

Bit 6~3 Unimplemented, read as “0”

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

TBC1 Register

Bit	7	6	5	4	3	2	1	0
Name	TB1EN	—	—	—	—	TB12	TB11	TB10
R/W	R/W	—	—	—	—	R/W	R/W	R/W
POR	0	—	—	—	—	0	0	0

Bit 7 **TB1EN**: Time Base 1 Enable/Disable Control
0: Disable
1: Enable

Bit 6~3 Unimplemented, read as “0”

Bit 2~0 **TB12~TB10**: Time Base 1 Time-out Period
000: $2^8/f_{PSC}$
001: $2^9/f_{PSC}$
010: $2^{10}/f_{PSC}$
011: $2^{11}/f_{PSC}$
100: $2^{12}/f_{PSC}$
101: $2^{13}/f_{PSC}$
110: $2^{14}/f_{PSC}$
111: $2^{15}/f_{PSC}$

EEPROM Interrupt

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

LVD Interrupt

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

Sine-PWM SFIFO Interrupt

The Sine-PWM SFIFO Empty Interrupt is provided to inform the MCU to write the next 4 bytes of Sine-PWM duty data into the SFIFO register pairs. The interrupt is activated by the SFIFO empty signal. When this happens, the interrupt request flag SFEF will be set. To allow the program to branch its respective interrupt vector address, the global interrupt enable bit, EMI, and the Sine-PWM SFIFO Empty interrupt enable bit, SFEE, must first be set. When the interrupt is enabled, the stack is not full and the SFIFO is empty, a subroutine call to its respective vector location will take place. When the interrupt is serviced, the respective interrupt request flag, SFEF, will be automatically reset and the EMI bit will be cleared to disable other interrupts.

AC Zero Detect Interrupt

The AC Zero Detect Interrupt is provided to inform the MCU that the AC input signal voltage is lower than the specific AC offset specified by a 8-bit AC offset DAC. The AC input signal and the 8-bit offset DAC output are connected to a comparator and are compared with each other. When this comparator output changes state from low to high, its output can activate an AC Zero interrupt. When this happens, the interrupt request flag ZXF will be set. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and the AC Zero interrupt enable bit, ZXE, must first be set. When the interrupt is enabled, the stack is not full and the comparator output changes state, a subroutine call to its respective vector location will take place. When the interrupt is serviced, the respective interrupt request flag will be automatically reset and the EMI bit will be cleared to disable other interrupts.

Over Current Protection Interrupt

The OCPn Interrupt is controlled by detecting the OCPn input current. An OCPn Interrupt request will take place when the OCPn Interrupt request flag, OCPnF, is set, which occurs when a large current is detected. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and OCPn Interrupt enable bit, OCPnE, must first be set. When the interrupt is enabled, the stack is not full and a over current is detected, a subroutine call to the OCPn Interrupt vector, will take place. When the interrupt is serviced, the OCPn Interrupt flag, OCPnF, will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts.

Over Voltage Protection Interrupt

The OVP Interrupt is controlled by detecting the input voltage. An OVP Interrupt request will take place when the OVP Interrupt request flag, OVPF, is set, which occurs when an large voltage is detected. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and OVP Interrupt enable bit, OVPE, must first be set. When the interrupt is enabled, the stack is not full and a large voltage is detected, a subroutine call to the OVP Interrupt vector, will take place. When the interrupt is serviced, the OVP Interrupt flag, OVPF, will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts.

TM Interrupts

The Compact and Standard Type TMs have two interrupts each. All of the TM interrupts are contained within the Multi-function Interrupts. For each of the Compact and Standard Type TMs there are two interrupt request flags TnPF and TnAF and two enable bits TnPE and TnAE. A TM interrupt request will take place when any of the TM request flags are set, a situation which occurs when a TM comparator P or A match situation happens.

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

Interrupt Wake-up Function

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

Programming Considerations

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

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

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

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

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

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

Low Voltage Detector – LVD

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

LVD Register

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

LVDC Register

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

Bit 7~6 Unimplemented, read as "0"

Bit 5 **LVDO**: LVD Output Flag
0: No Low Voltage Detect
1: Low Voltage Detect

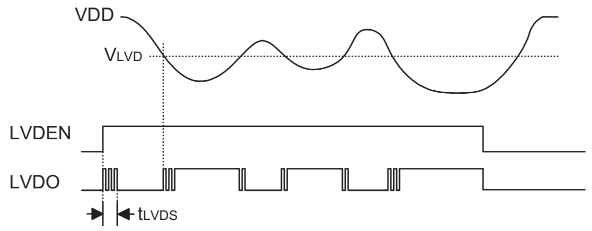
Bit 4 **LVDEN**: Low Voltage Detector Control
0: Disable
1: Enable

Bit 3 **VBGEN**: Bandgap buffer Control
0: Disable
1: Enable

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

LVD Operation

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



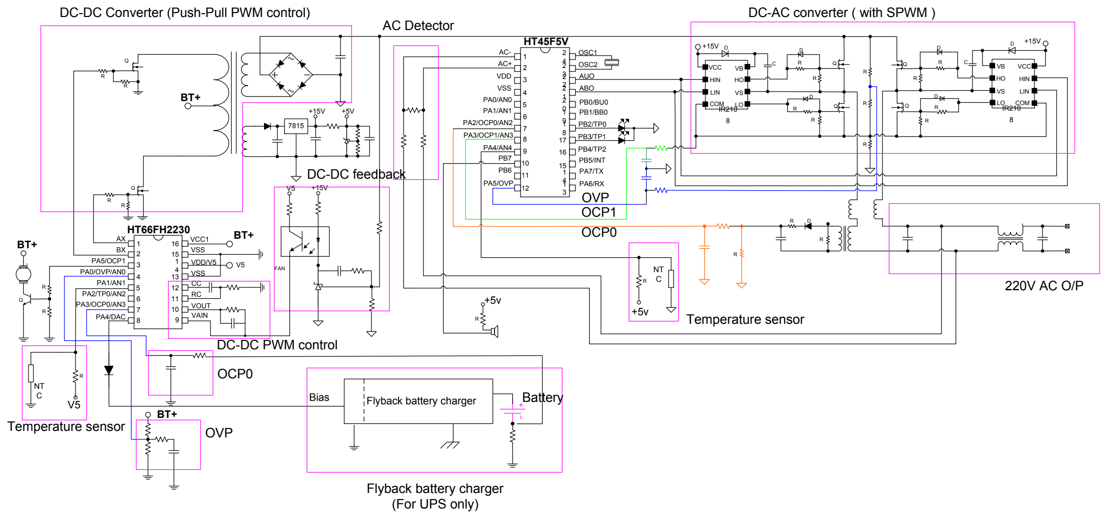
LVD Operation

The Low Voltage Detector also has its own interrupt, providing an alternative means of low voltage detection, in addition to polling the LVDO bit. The interrupt will only be generated after a delay of t_{LVD} after the LVDO bit has been set high by a low voltage condition. When the device is in SLEEP mode the Low Voltage Detector will disable, even if the ENLVD bit is high.

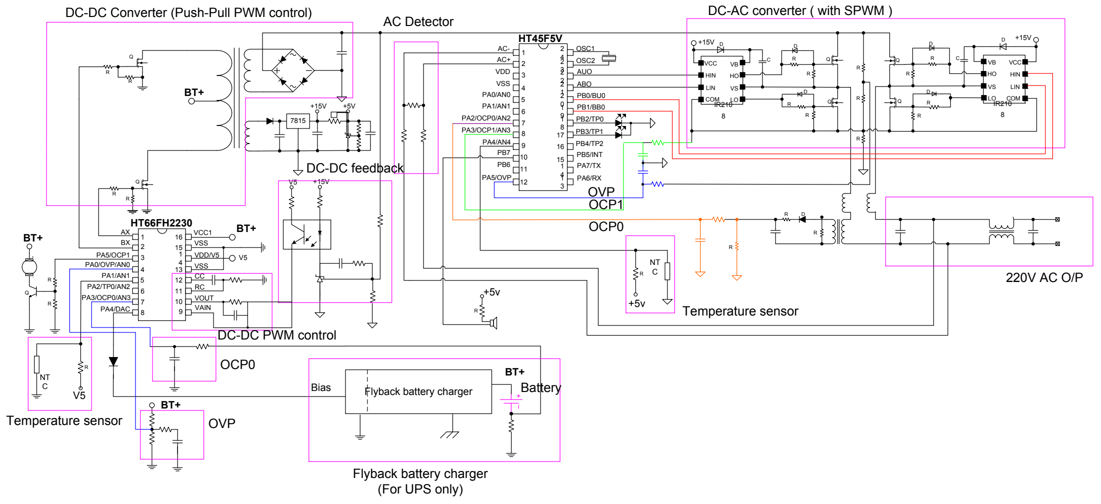
When LVD function is enabled, it is recommenced to clear LVD flag first, and then enables interrupt function to avoid mistake action.

Application Circuits

Bipolar Type



Unipolar mode



Instruction Set

Introduction

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

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

Instruction Timing

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

Moving and Transferring Data

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

Arithmetic Operations

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

Logical and Rotate Operation

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

Branches and Control Transfer

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

Bit Operations

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

Table Read Operations

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

Other Operations

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

Instruction Set Summary

The instructions related to the data memory access in the following table can be used when the desired data memory is located in Data Memory sector 0.

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, SC
ADDM A,[m]	Add ACC to Data Memory	1^{Note}	Z, C, AC, OV, SC
ADD A,x	Add immediate data to ACC	1	Z, C, AC, OV, SC
ADC A,[m]	Add Data Memory to ACC with Carry	1	Z, C, AC, OV, SC
ADCM A,[m]	Add ACC to Data memory with Carry	1^{Note}	Z, C, AC, OV, SC
SUB A,x	Subtract immediate data from the ACC	1	Z, C, AC, OV, SC, CZ
SUB A,[m]	Subtract Data Memory from ACC	1	Z, C, AC, OV, SC, CZ
SUBM A,[m]	Subtract Data Memory from ACC with result in Data Memory	1^{Note}	Z, C, AC, OV, SC, CZ
SBC A,x	Subtract immediate data from ACC with Carry	1	Z, C, AC, OV, SC, CZ
SBC A,[m]	Subtract Data Memory from ACC with Carry	1	Z, C, AC, OV, SC, CZ
SBCM A,[m]	Subtract Data Memory from ACC with Carry, result in Data Memory	1^{Note}	Z, C, AC, OV, SC, CZ
DAA [m]	Decimal adjust ACC for Addition with result in Data Memory	1^{Note}	C
Logic Operation			
AND A,[m]	Logical AND Data Memory to ACC	1	Z
OR A,[m]	Logical OR Data Memory to ACC	1	Z
XOR A,[m]	Logical XOR Data Memory to ACC	1	Z
ANDM A,[m]	Logical AND ACC to Data Memory	1^{Note}	Z
ORM A,[m]	Logical OR ACC to Data Memory	1^{Note}	Z
XORM A,[m]	Logical XOR ACC to Data Memory	1^{Note}	Z
AND A,x	Logical AND immediate Data to ACC	1	Z
OR A,x	Logical OR immediate Data to ACC	1	Z
XOR A,x	Logical XOR immediate Data to ACC	1	Z
CPL [m]	Complement Data Memory	1^{Note}	Z
CPLA [m]	Complement Data Memory with result in ACC	1	Z
Increment & Decrement			
INCA [m]	Increment Data Memory with result in ACC	1	Z
INC [m]	Increment Data Memory	1^{Note}	Z
DECA [m]	Decrement Data Memory with result in ACC	1	Z
DEC [m]	Decrement Data Memory	1^{Note}	Z
Rotate			
RRA [m]	Rotate Data Memory right with result in ACC	1	None
RR [m]	Rotate Data Memory right	1^{Note}	None
RRCA [m]	Rotate Data Memory right through Carry with result in ACC	1	C
RRC [m]	Rotate Data Memory right through Carry	1^{Note}	C
RLA [m]	Rotate Data Memory left with result in ACC	1	None
RL [m]	Rotate Data Memory left	1^{Note}	None
RLCA [m]	Rotate Data Memory left through Carry with result in ACC	1	C
RLC [m]	Rotate Data Memory left through Carry	1^{Note}	C

Mnemonic	Description	Cycles	Flag Affected
Data Move			
MOV A,[m]	Move Data Memory to ACC	1	None
MOV [m],A	Move ACC to Data Memory	1Note	None
MOV A,x	Move immediate data to ACC	1	None
Bit Operation			
CLR [m].i	Clear bit of Data Memory	1Note	None
SET [m].i	Set bit of Data Memory	1Note	None
Branch Operation			
JMP addr	Jump unconditionally	2	None
SZ [m]	Skip if Data Memory is zero	1Note	None
SZA [m]	Skip if Data Memory is zero with data movement to ACC	1Note	None
SZ [m].i	Skip if bit i of Data Memory is zero	1Note	None
SNZ [m]	Skip if Data Memory is not zero	1Note	None
SNZ [m].i	Skip if bit i of Data Memory is not zero	1Note	None
SIZ [m]	Skip if increment Data Memory is zero	1Note	None
SDZ [m]	Skip if decrement Data Memory is zero	1Note	None
SIZA [m]	Skip if increment Data Memory is zero with result in ACC	1Note	None
SDZA [m]	Skip if decrement Data Memory is zero with result in ACC	1Note	None
CALL addr	Subroutine call	2	None
RET	Return from subroutine	2	None
RET A,x	Return from subroutine and load immediate data to ACC	2	None
RETI	Return from interrupt	2	None
Table Read Operation			
TABRD [m]	Read table (specific page) to TBLH and Data Memory	2Note	None
TABRDL [m]	Read table (last page) to TBLH and Data Memory	2Note	None
ITABRD [m]	Increment table pointer TBLP first and Read table (specific page) to TBLH and Data Memory	2Note	None
ITABRDL [m]	Increment table pointer TBLP first and Read table (last page) to TBLH and Data Memory	2Note	None
Miscellaneous			
NOP	No operation	1	None
CLR [m]	Clear Data Memory	1Note	None
SET [m]	Set Data Memory	1Note	None
CLR WDT	Clear Watchdog Timer	1	TO, PDF
SWAP [m]	Swap nibbles of Data Memory	1Note	None
SWAPA [m]	Swap nibbles of Data Memory with result in ACC	1	None
HALT	Enter power down mode	1	TO, PDF

Note: 1. For skip instructions, if the result of the comparison involves a skip then two cycles are required, if no skip takes place only one cycle is required.
2. Any instruction which changes the contents of the PCL will also require 2 cycles for execution.

Extended Instruction Set

The extended instructions are used to support the full range address access for the data memory. When the accessed data memory is located in any data memory sector except sector 0, the extended instruction can be used to directly access the data memory instead of using the indirect addressing access. This can not only reduce the use of Flash memory space but also improve the CPU execution efficiency.

Mnemonic	Description	Cycles	Flag Affected
Arithmetic			
LADD A,[m]	Add Data Memory to ACC	2	Z, C, AC, OV, SC
LADDM A,[m]	Add ACC to Data Memory	2^{Note}	Z, C, AC, OV, SC
LADC A,[m]	Add Data Memory to ACC with Carry	2	Z, C, AC, OV, SC
LADCM A,[m]	Add ACC to Data memory with Carry	2^{Note}	Z, C, AC, OV, SC
LSUB A,[m]	Subtract Data Memory from ACC	2	Z, C, AC, OV, SC, CZ
LSUBM A,[m]	Subtract Data Memory from ACC with result in Data Memory	2^{Note}	Z, C, AC, OV, SC, CZ
LSBC A,[m]	Subtract Data Memory from ACC with Carry	2	Z, C, AC, OV, SC, CZ
LSBCM A,[m]	Subtract Data Memory from ACC with Carry, result in Data Memory	2^{Note}	Z, C, AC, OV, SC, CZ
LDAA [m]	Decimal adjust ACC for Addition with result in Data Memory	2^{Note}	C
Logic Operation			
LAND A,[m]	Logical AND Data Memory to ACC	2	Z
LOR A,[m]	Logical OR Data Memory to ACC	2	Z
LXOR A,[m]	Logical XOR Data Memory to ACC	2	Z
LANDM A,[m]	Logical AND ACC to Data Memory	2^{Note}	Z
LORM A,[m]	Logical OR ACC to Data Memory	2^{Note}	Z
LXORM A,[m]	Logical XOR ACC to Data Memory	2^{Note}	Z
LCPL [m]	Complement Data Memory	2^{Note}	Z
LCPLA [m]	Complement Data Memory with result in ACC	2	Z
Increment & Decrement			
LINCA [m]	Increment Data Memory with result in ACC	2	Z
LINC [m]	Increment Data Memory	2^{Note}	Z
LDECA [m]	Decrement Data Memory with result in ACC	2	Z
LDEC [m]	Decrement Data Memory	2^{Note}	Z
Rotate			
LRRA [m]	Rotate Data Memory right with result in ACC	2	None
LRR [m]	Rotate Data Memory right	2^{Note}	None
LRRCA [m]	Rotate Data Memory right through Carry with result in ACC	2	C
LRRC [m]	Rotate Data Memory right through Carry	2^{Note}	C
LRLA [m]	Rotate Data Memory left with result in ACC	2	None
LRL [m]	Rotate Data Memory left	2^{Note}	None
LRLCA [m]	Rotate Data Memory left through Carry with result in ACC	2	C
LRLC [m]	Rotate Data Memory left through Carry	2^{Note}	C
Data Move			
LMOV A,[m]	Move Data Memory to ACC	2	None
LMOV [m],A	Move ACC to Data Memory	2^{Note}	None
Bit Operation			
LCLR [m].i	Clear bit of Data Memory	2^{Note}	None
LSET [m].i	Set bit of Data Memory	2^{Note}	None

Mnemonic	Description	Cycles	Flag Affected
Branch			
LSZ [m]	Skip if Data Memory is zero	2[Note]	None
LSZA [m]	Skip if Data Memory is zero with data movement to ACC	2[Note]	None
LSNZ [m]	Skip if Data Memory is not zero	2[Note]	None
LSZ [m].i	Skip if bit i of Data Memory is zero	2[Note]	None
LSNZ [m].i	Skip if bit i of Data Memory is not zero	2[Note]	None
LSIZ [m]	Skip if increment Data Memory is zero	2[Note]	None
LSDZ [m]	Skip if decrement Data Memory is zero	2[Note]	None
LSIZA [m]	Skip if increment Data Memory is zero with result in ACC	2[Note]	None
LSDZA [m]	Skip if decrement Data Memory is zero with result in ACC	2[Note]	None
Table Read			
LTABRD [m]	Read table (specific page) to TBLH and Data Memory	3[Note]	None
LTABRDL [m]	Read table (last page) to TBLH and Data Memory	3[Note]	None
LITABRD [m]	Increment table pointer TBLP first and Read table (specific page) to TBLH and Data Memory	3[Note]	None
LITABRDL [m]	Increment table pointer TBLP first and Read table (last page) to TBLH and Data Memory	3[Note]	None
Miscellaneous			
LCLR [m]	Clear Data Memory	2[Note]	None
LSET [m]	Set Data Memory	2[Note]	None
LSWAP [m]	Swap nibbles of Data Memory	2[Note]	None
LSWAPA [m]	Swap nibbles of Data Memory with result in ACC	2	None

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

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

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, SC

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, SC

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, SC

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, SC

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, SC

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

CPL [m]	Complement Data Memory
Description	Each bit of the specified Data Memory is logically complemented (1′s complement). Bits which previously contained a 1 are changed to 0 and vice versa.
Operation	[m] ← $\overline{[m]}$
Affected flag(s)	Z

CPLA [m]	Complement Data Memory with result in ACC
Description	Each bit of the specified Data Memory is logically complemented (1′s complement). Bits which previously contained a 1 are changed to 0 and vice versa. The complemented result is stored in the Accumulator and the contents of the Data Memory remain unchanged.
Operation	ACC ← $\overline{[m]}$
Affected flag(s)	Z

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

DEC [m]	Decrement Data Memory
Description	Data in the specified Data Memory is decremented by 1.
Operation	[m] ← [m] − 1
Affected flag(s)	Z

DECA [m]	Decrement Data Memory with result in ACC
Description	Data in the specified Data Memory is decremented by 1. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.
Operation	ACC ← [m] − 1
Affected flag(s)	Z

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

Operation	TO ← 0
	PDF ← 1

Affected flag(s)	TO, PDF

INC [m]	Increment Data Memory
Description	Data in the specified Data Memory is incremented by 1.
Operation	[m] ← [m] + 1
Affected flag(s)	Z

INCA [m]	Increment Data Memory with result in ACC
Description	Data in the specified Data Memory is incremented by 1. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.
Operation	ACC ← [m] + 1
Affected flag(s)	Z

JMP addr	Jump unconditionally
Description	The contents of the Program Counter are replaced with the specified address. Program execution then continues from this new address. As this requires the insertion of a dummy instruction while the new address is loaded, it is a two cycle instruction.
Operation	Program Counter ← addr
Affected flag(s)	None

MOV A,[m]	Move Data Memory to ACC
Description	The contents of the specified Data Memory are copied to the Accumulator.
Operation	ACC ← [m]
Affected flag(s)	None

MOV A,x	Move immediate data to ACC
Description	The immediate data specified is loaded into the Accumulator.
Operation	ACC ← x
Affected flag(s)	None

MOV [m],A	Move ACC to Data Memory
Description	The contents of the Accumulator are copied to the specified Data Memory.
Operation	[m] ← ACC
Affected flag(s)	None

NOP	No operation
Description	No operation is performed. Execution continues with the next instruction.
Operation	No operation
Affected flag(s)	None

OR A,[m]	Logical OR Data Memory to ACC
Description	Data in the Accumulator and the specified Data Memory perform a bitwise logical OR operation. The result is stored in the Accumulator.
Operation	ACC ← ACC ″OR″ [m]
Affected flag(s)	Z

OR A,x	Logical OR immediate data to ACC
Description	Data in the Accumulator and the specified immediate data perform a bitwise logical OR operation. The result is stored in the Accumulator.
Operation	ACC ← ACC ″OR″ x
Affected flag(s)	Z

ORM A,[m]	Logical OR ACC to Data Memory
Description	Data in the specified Data Memory and the Accumulator perform a bitwise logical OR operation. The result is stored in the Data Memory.
Operation	[m] ← ACC ″OR″ [m]
Affected flag(s)	Z

RET	Return from subroutine
Description	The Program Counter is restored from the stack. Program execution continues at the restored address.
Operation	Program Counter ← Stack
Affected flag(s)	None

RET A,x	Return from subroutine and load immediate data to ACC
Description	The Program Counter is restored from the stack and the Accumulator loaded with the specified immediate data. Program execution continues at the restored address.
Operation	Program Counter ← Stack ACC ← x
Affected flag(s)	None

RETI	Return from interrupt
Description	The Program Counter is restored from the stack and the interrupts are re-enabled by setting the EMI bit. EMI is the master interrupt global enable bit. If an interrupt was pending when the RETI instruction is executed, the pending Interrupt routine will be processed before returning to the main program.
Operation	Program Counter ← Stack EMI ← 1
Affected flag(s)	None

RL [m]	Rotate Data Memory left
Description	The contents of the specified Data Memory are rotated left by 1 bit with bit 7 rotated into bit 0.
Operation	[m].(i+1) ← [m].i; (i=0~6) [m].0 ← [m].7
Affected flag(s)	None

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

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

Affected flag(s)	None

RLC [m]	Rotate Data Memory left through Carry
Description	The contents of the specified Data Memory and the carry flag are rotated left by 1 bit. Bit 7 replaces the Carry bit and the original carry flag is rotated into bit 0.
Operation	[m].(i+1) ← [m].i; (i=0~6) [m].0 ← C C ← [m].7
Affected flag(s)	C

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

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

Affected flag(s)	C

RR [m]	Rotate Data Memory right
Description	The contents of the specified Data Memory are rotated right by 1 bit with bit 0 rotated into bit 7.
Operation	[m].i ← [m].(i+1); (i=0~6) [m].7 ← [m].0
Affected flag(s)	None

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

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

Affected flag(s)	None

RRC [m]	Rotate Data Memory right through Carry
Description	The contents of the specified Data Memory and the carry flag are rotated right by 1 bit. Bit 0 replaces the Carry bit and the original carry flag is rotated into bit 7.
Operation	[m].i ← [m].(i+1); (i=0~6) [m].7 ← C C ← [m].0
Affected flag(s)	C

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

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

Affected flag(s)	C

SBC A,[m]	Subtract Data Memory from ACC with Carry
Description	The contents of the specified Data Memory and the complement of the carry flag are subtracted from the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.
Operation	ACC ← ACC − [m] − $\overline{C}$
Affected flag(s)	OV, Z, AC, C, SC, CZ

SBC A, x	Subtract immediate data from ACC with Carry
Description	The immediate data and the complement of the carry flag are subtracted from the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.
Operation	ACC ← ACC – [m] – $\overline{C}$
Affected flag(s)	OV, Z, AC, C, SC, CZ

SBCM A,[m]	Subtract Data Memory from ACC with Carry and result in Data Memory
Description	The contents of the specified Data Memory and the complement of the carry flag are subtracted from the Accumulator. The result is stored in the Data Memory. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.
Operation	[m] ← ACC − [m] − $\overline{C}$
Affected flag(s)	OV, Z, AC, C, SC, CZ

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 Data Memory is not 0
Description	If 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

SNZ [m]	Skip if Data Memory is not 0
Description	If 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]≠ 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, SC, CZ

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, SC, CZ

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, SC, CZ

SWAP [m]	Swap nibbles of Data Memory
Description	The low-order and high-order nibbles of the specified Data Memory are interchanged.
Operation	[m].3~[m].0 ↔ [m].7~[m].4
Affected flag(s)	None

SWAPA [m]	Swap nibbles of Data Memory with result in ACC
Description	The low-order and high-order nibbles of the specified Data Memory are interchanged. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.
Operation	ACC.3~ACC.0 ← [m].7~[m].4 ACC.7~ACC.4 ← [m].3~[m].0
Affected flag(s)	None

SZ [m]	Skip if Data Memory is 0
Description	If the contents of the specified Data Memory is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.
Operation	Skip if [m]=0
Affected flag(s)	None

SZA [m]	Skip if Data Memory is 0 with data movement to ACC
Description	The contents of the specified Data Memory are copied to the Accumulator. If the value is zero, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.
Operation	ACC ← [m] Skip if [m]=0
Affected flag(s)	None

SZ [m].i	Skip if bit i of Data Memory is 0
Description	If bit i of the specified Data Memory is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0, the program proceeds with the following instruction.
Operation	Skip if [m].i=0
Affected flag(s)	None

TABRD [m]	Read table (specific page) to TBLH and Data Memory
Description	The low byte of the program code (specific page) addressed by the table pointer (TBLP and TBHP) 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

ITABRD [m]	Increment table pointer low byte first and read table (specific page) to TBLH and Data Memory
Description	Increment table pointer low byte, TBLP, first and then the program code (specific page) 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

ITABRDL [m]	Increment table pointer low byte first and read table (last page) to TBLH and Data Memory
Description	Increment table pointer low byte, TBLP, first and then 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

Extended Instruction Definition

The extended instructions are used to directly access the data stored in any data memory sections.

LADC 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, SC

LADCM 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, SC

LADD 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, SC

LADDM 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, SC

LAND 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

LANDM 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

LCLR [m]	Clear Data Memory
Description	Each bit of the specified Data Memory is cleared to 0.
Operation	[m] ← 00H
Affected flag(s)	None

LCLR [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

LCPL [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

LCPLA [m]	Complement Data Memory with result in ACC
Description	Each bit of the specified Data Memory is logically complemented (1′s complement). Bits which previously contained a 1 are changed to 0 and vice versa. The complemented result is stored in the Accumulator and the contents of the Data Memory remain unchanged.
Operation	$ACC \leftarrow \overline{[m]}$
Affected flag(s)	Z

LDAA [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	$[m] \leftarrow ACC + 00H$ or $[m] \leftarrow ACC + 06H$ or $[m] \leftarrow ACC + 60H$ or $[m] \leftarrow ACC + 66H$
Affected flag(s)	C

LDEC [m]	Decrement Data Memory
Description	Data in the specified Data Memory is decremented by 1.
Operation	$[m] \leftarrow [m] - 1$
Affected flag(s)	Z

LDECA [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	$ACC \leftarrow [m] - 1$
Affected flag(s)	Z

LINC [m]	Increment Data Memory
Description	Data in the specified Data Memory is incremented by 1.
Operation	$[m] \leftarrow [m] + 1$
Affected flag(s)	Z

LINCA [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	$ACC \leftarrow [m] + 1$
Affected flag(s)	Z

LMOV 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

LMOV [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

LOR 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

LORM 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

LRL [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

LRLA [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

LRLC [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

LRLCA [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

LRR [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

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

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

Affected flag(s)	None

LRRC [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

LRRCA [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

LSBC A,[m]	Subtract Data Memory from ACC with Carry
Description	The contents of the specified Data Memory and the complement of the carry flag are subtracted from the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.
Operation	ACC ← ACC − [m] − $\overline{C}$
Affected flag(s)	OV, Z, AC, C, SC, CZ

LSBCM A,[m]	Subtract Data Memory from ACC with Carry and result in Data Memory
Description	The contents of the specified Data Memory and the complement of the carry flag are subtracted from the Accumulator. The result is stored in the Data Memory. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.
Operation	[m] ← ACC − [m] − $\overline{C}$
Affected flag(s)	OV, Z, AC, C, SC, CZ

LSDZ [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 three 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

LSDZA [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 three 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

LSET [m]	Set Data Memory
Description	Each bit of the specified Data Memory is set to 1.
Operation	[m] ← FFH
Affected flag(s)	None

LSET [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

LSIZ [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 three 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

LSIZA [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 three 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

LSNZ [m].i	Skip if Data Memory is not 0
Description	If 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 three cycle instruction. If the result is 0 the program proceeds with the following instruction.
Operation	Skip if [m].i ≠ 0
Affected flag(s)	None

LSNZ [m]	Skip if Data Memory is not 0
Description	If the content 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 three cycle instruction. If the result is 0 the program proceeds with the following instruction.
Operation	Skip if [m] ≠ 0
Affected flag(s)	None

LSUB 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, SC, CZ

LSUBM 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, SC, CZ

LSWAP [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

LSWAPA [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

LSZ [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 three cycle instruction. If the result is not 0 the program proceeds with the following instruction.
Operation	Skip if [m]=0
Affected flag(s)	None

LSZA [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 three 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

LSZ [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 three 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

LTABRD [m]	Read table (specific page) to TBLH and Data Memory
Description	The low byte of the program code (specific page) addressed by the table pointer (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

LTABRDL [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

LITABRD [m]	Increment table pointer low byte first and read table (specific page) to TBLH and Data Memory
Description	Increment table pointer low byte, TBLP, first and then the program code (specific page) 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

LITABRDL [m]	Increment table pointer low byte first and read table (last page) to TBLH and Data Memory
Description	Increment table pointer low byte, TBLP, first and then 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

LXOR 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

LXORM 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

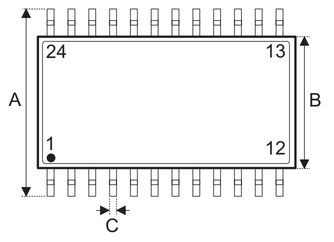
Package Information

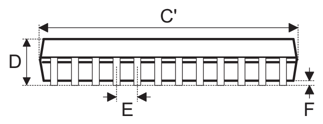
Note that the package information provided here is for consultation purposes only. As this information may be updated at regular intervals users are reminded to consult the Holtek website for the latest version of the Package/Carton Information.

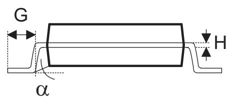
Additional supplementary information with regard to packaging is listed below. Click on the relevant section to be transferred to the relevant website page.

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

24-pin SSOP (300mil) Outline Dimensions



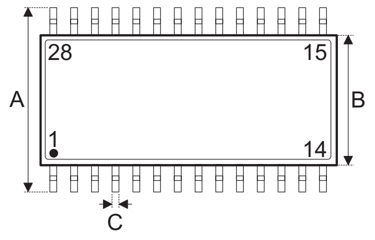


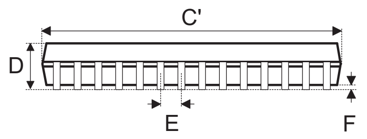


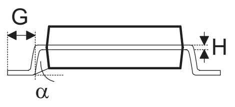
Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	—	0.236 BSC	—
B	—	0.154 BSC	—
C	0.008	—	0.012
C’	—	0.341 BSC	—
D	—	—	0.069
E	—	0.025 BSC	—
F	0.004	—	0.0098
G	0.016	—	0.050
H	0.004	—	0.010
α	0°	—	8°

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

28-pin SSOP (300mil) Outline Dimensions







Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	—	0.236 BSC	—
B	—	0.154 BSC	—
C	0.008	—	0.012
C’	—	0.390 BSC	—
D	—	—	0.069
E	—	0.025 BSC	—
F	0.004	—	0.0098
G	0.016	—	0.050
H	0.004	—	0.010
α	0°	—	8°

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