HT49RA1/HT49CA1
Remote Type 8-Bit MCU with LCD

Technical Document

- Tools Information
- FAQs
- Application Note
  - HA0075E MCU Reset and Oscillator Circuits Application Note

Features

- Operating voltage: 2.0V~3.6V
- 12 bidirectional I/O lines, 8 input lines, 8 segment output, PC0 (input/output)/REM
- Two external interrupt inputs shared with an I/O line
- 8-bit programmable Timer/Event Counter with overflow interrupt function
- Single 16-bit programmable timer/event counter with overflow interrupt function
- RC oscillator and 32768Hz crystal oscillator
- LCD driver with 33×2, 33×3 or 32×4 segments (C type only), 8 logical output option for SEG12~SEG19 and 4 I/O port option for SEG0~SEG3 by changing LCDC register
- 4096×15 program memory
- 160×8 data memory
- Real Time Clock – RTC
- 8-bit prescaler for RTC
- One carrier output (1/2 or 1/3 duty)
- Software LCD, RTC control
- Watchdog Timer function
- Power down and wake-up functions to reduce power consumption
- Up to 1μs instruction cycle with 4MHz system clock
- 4-level subroutine nesting
- Bit manipulation instruction
- Table read instructions
- 63 powerful instructions
- All instructions executed in one or two machine cycles
- Low voltage reset/detector function
- 52-pin QFP, 64-pin LQFP packages

General Description

The HT49RA1/HT49CA1 is a Remote Type 8-bit MCU 8-bit high performance RISC architecture microcontroller. With its internal carrier generator and LCD Driver functions the device is is especially suitable for multiple I/O remote control product applications. The usual Holtek MCU features such as power down and wake-up functions, oscillator options, etc. combine to ensure user applications require a minimum of external components.

The benefits of low power consumption, high performance, I/O flexibility and low-cost, provide these devices with the versatility to suit a wide range of application possibilities such as industrial control, consumer products and particularly suitable for use in products such as infrared LCD remote controllers and various, subsystem controllers, etc.

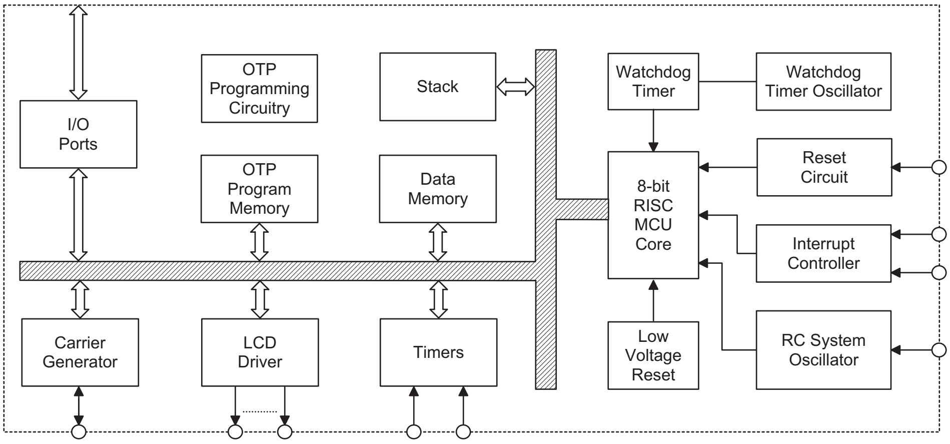
Device Types

Devices which have the letter ″R″ within their part number, indicate that they are OTP devices offering the advantages of easy and effective program updates, using the Holtek range of development and programming tools. These devices provide the designer with the means for fast and low-cost product development cycles. Devices which have the letter ″C″ within their part number indicate that they are mask version devices. These devices offer a complementary device for applications that are at a mature state in their design process and have high volume and low cost demands.

Fully pin and functionally compatible with their OTP sister devices, the mask version devices provide the ideal substitute for products which have gone beyond their development cycle and are facing cost-down demands.

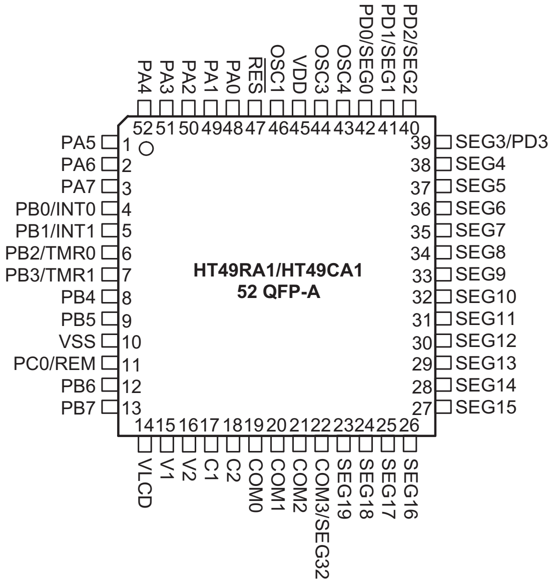
In this datasheet, for convenience, when describing device functions, only the OTP types are mentioned by name, however the same described functions also apply to the Mask type devices.

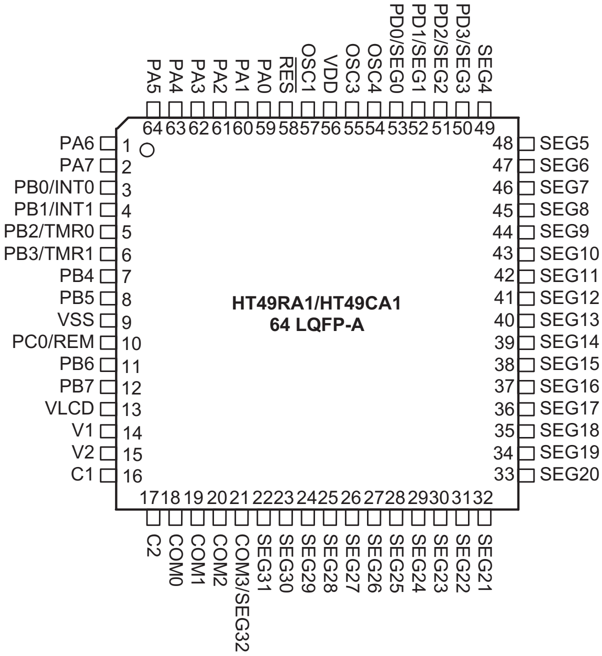
Block Diagram



Note: This block diagram represents the OTP devices, for the mask devices there is no Device Programming Circuitry.

Pin Assignment





Pin Description

Pin Name	I/O	Configuration Option	Description
PA0~PA7	I/O	—	Bidirectional NMOS 8-bit input/output port. Each bit can be chosen as an NMOS output or Schmitt trigger input using software instructions. Pull-high resistors are permanently connected to these pins.
PB0/INT0 PB1/INT1 PB2/TMR0 PB3/TMR1 PB4~PB7	I	Wake-up	8-bit Schmitt trigger input lines with pull-high resistors. Each bit can be configured as a wake-up input via configuration options. Pins PB0, PB1, PB2 and PB3 are pin-shared with INT0, INT1, TMR0 and TMR1 respectively
PC0/REM	I/O	Carrier Output Pull-high	Bidirectional I/O port. PC0 can be configured as a CMOS output pin or carrier output pin using a configuration option.
PD0/SEG0~ PD3/SEG3	I/O	—	Bidirectional NMOS 4-bit input/output port. Each bit can be chosen as an NMOS output or Schmitt trigger input using software instructions. Each pin on this port can be configured either as a segment pin or normal I/O pin using the LCDC register. When used as I/O pins pull-high resistors are permanently connected to these pins.
OSC1	I	—	A resistor is connected between OSC1 and ground to implement the internal system clock.
OSC3 OSC4	I O	—	Real time clock oscillator. OSC3 and OSC4 are connected to a 32768Hz crystal oscillator for timing purpose. It is not used as the system clock. If the RTC is not selected as f_S. then OSC3, OSC4 should be left floating.
VLCD	—	—	LCD power supply. V_{LCD} should be larger than V_{DD} for connect operation i.e. $V_{LCD} \geq V_{DD}$
V1, V2, C1, C2			LCD voltage pump
COM0~COM2 COM3/SEG32	O	1/2, 1/3 or 1/4 Duty	COM0~COM2 are the LCD panel common connections. Pin COM3/SEG32 can be setup as an LCD panel segment or as a common output driver via configuration options.
SEG4~SEG11	O	—	LCD panel segments driver outputs.
SEG12~SEG19	O	SEG12~SEG19 CMOS Output	LCD panel segments driver outputs . SEG12~SEG19 can be setup as LCD segment outputs or as CMOS output via a configuration option.
SEG20~SEG31	O	—	LCD panel segments driver outputs.
$\overline{\text{RES}}$	I	—	Schmitt Trigger reset input. Active low.
VDD	—	—	Positive power supply
VSS	—	—	Negative power supply, ground

Note: Each pin on PB can be programmed through a configuration option to have a wake-up function.

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

I_{OL} Total 150mA

Total Power Dissipation 500mW

Storage Temperature ....–50°C to 125°C

Operating Temperature..........................–40°C to 85°C

I_{OH} Total..–100mA

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

D.C. Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	Operating Voltage	—	—	2.0	—	3.6	V
I_{DD}	Operating Current (RC OSC)	3V	No load, f_{SYS}=4MHz	—	0.7	1.5	mA
I_{STB1}	Standby Current (*f_S=T1)	3V	No load, system HALT, LCD off at HALT	—	0.1	1	μA
I_{STB2}	Standby Current (*f_S=32.768kHz OSC)	3V	No load, system HALT, LCD On at HALT, C type	—	2.5	5	μA
I_{STB3}	Standby Current (*f_S=WDT RC OSC)	3V	No load, system HALT LCD On at HALT, C type	—	2	5	μA
V_{IL1}	Input Low Voltage for I/O Ports, TMR0/TMR1 and INT0/INT1	3V	—	0	—	$0.3V_{DD}$	V
V_{IH1}	Input High Voltage for I/O Ports, TMR0/TMR1 and INT0/INT1	3V	—	$0.7V_{DD}$	—	V_{DD}	V
V_{IL2}	Input Low Voltage ($\overline{RES}$)	3V	—	0	—	$0.4V_{DD}$	V
V_{IH2}	Input High Voltage ($\overline{RES}$)	3V	—	$0.9V_{DD}$	—	V_{DD}	V
I_{OL1}	I/O Port & REM Sink Current	3V	$V_{OL}=0.1V_{DD}$	4	8	—	mA
I_{OH1}	I/O Port & REM Source Current	3V	$V_{OH}=0.9V_{DD}$	–5	–7	—	mA
I_{OL2}	LCD Common and Segment Current	3V	$V_{OL}=0.1V_{DD}$	210	420	—	μA
I_{OH2}	LCD Common and Segment Current	3V	$V_{OH}=0.9V_{DD}$	–80	–160	—	μA
R_{PH}	Pull-high Resistance of I/O Ports	3V	—	100	150	200	kΩ
V_{LVR}	Low Voltage Reset Voltage	—	LVR 2.1V option	1.98	2.10	2.22	V
			LVR 3.15V optio	2.98	3.15	3.32	V
V_{LVD}	Low Voltage Detector Voltage	—	LVD voltage 2.2V option	2.08	2.20	2.32	V
			LVD voltage 3.3V option	3.12	3.30	3.50	V
V_{POR}	VDD Start Voltage to Ensure Power-on Reset	—	—	—	—	100	mV
R_{POR}	VDD Rise Rate to Ensure Power-on Reset	—	—	0.035	—	—	V/ms

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

"*f_S" please refer to WDT clock option

A.C. Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
f_{SYS}	System Clock	2.0V~ 3.6V	4MHz ± 3%, Temp.= 0°C ~ 50°C	—	4000	—	kHz
		3.0V	4MHz ± 2%, Temp.= 25°C	—	4000	—	kHz
f_{RTCOSC}	RTC Frequency	—	—	—	32768	—	Hz
f_{TIMER}	Timer I/P Frequency (TMR0/TMR1)	3V	—	0	—	4000	kHz
t_{WDTOSC}	Watchdog Oscillator Period	3V	—	45	90	180	μs
t_{RES}	External Reset Low Pulse Width	—	—	1	—	—	μs
t_{LVR}	Low Voltage Width to Reset	—	—	0.25	1	2	ms
t_{SST}	System Start-up Timer Period	—	Wake-up from HALT	—	1024	—	$*t_{SYS}$
t_{INT}	Interrupt Pulse Width	—	—	1	—	—	μs

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

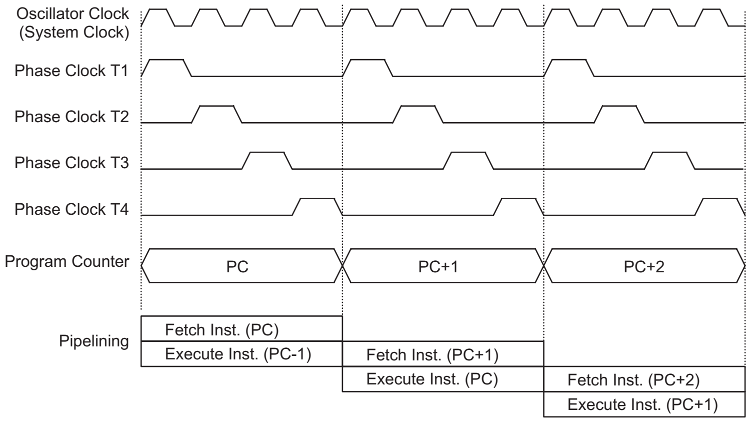
System Architecture

A key factor in the high-performance features of the Holtek range of microcontrollers is attributed to the internal system architecture. The range of devices take advantage of the usual features found within RISC microcontrollers providing increased speed of operation and enhanced performance. The pipelining scheme is implemented in such a way that instruction fetching and instruction execution are overlapped, hence instructions are effectively executed in one cycle, with the exception of branch or call instructions. An 8-bit wide ALU is used in practically all operations of the instruction set. It carries out arithmetic operations, logic operations, rotation, increment, decrement, branch decisions, etc. The internal data path is simplified by moving data through the Accumulator and the ALU. Certain internal registers are implemented in the Data Memory and can be directly or indirectly addressed. The simple addressing methods of these registers along with additional architectural features ensure that a minimum of external components is required to provide a functional I/O with maximum reliability and flexibility. This makes these devices suitable for low-cost, high-volume production for controller applications requiring 4K words of Program Memory and 160 bytes of Data Memory storage.

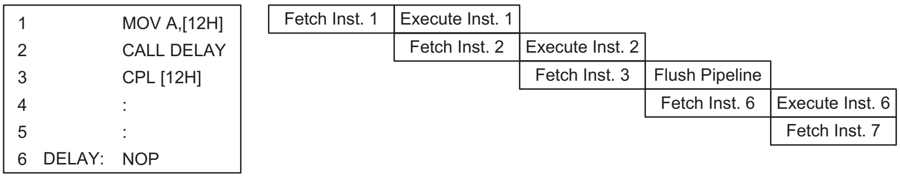
Clocking and Pipelining

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

For instructions involving branches, such as jump or call instructions, two 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.



System Clocking and Pipelining



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

Fetch Inst. 1 | Execute Inst. 1

Fetch Inst. 2 | Execute Inst. 2

Fetch Inst. 3 | Flush Pipeline

Fetch Inst. 6 | Execute Inst. 6

Fetch Inst. 7

Instruction Fetching

Program Counter

During program execution, the Program Counter is used to keep track of the address of the next instruction to be executed. It is automatically incremented by one each time an instruction is executed except for instructions, such as ″JMP″ or ″CALL″ that demand a jump to a non-consecutive Program Memory address. For the Remote Type series of microcontrollers with LCD, note that the Program Counter width varies with the Program Memory capacity depending upon which device is selected. However, it must be noted that only the lower 8 bits, known as the Program Counter Low Register, are directly addressable by user.

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

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 writable register. By transferring data directly into this register, a short program jump can be executed directly, however, as only this low byte is available for manipulation, the jumps are limited to the present page of memory, that is 256 locations. When such program jumps are executed it should also be noted that a dummy cycle will be inserted.

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

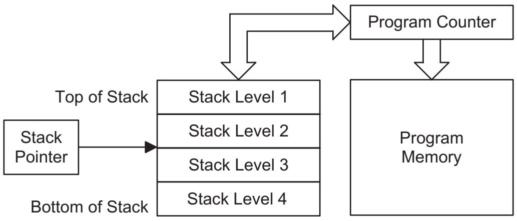
Stack

This is a special part of the memory which is used to save the contents of the Program Counter only. The stack can have 4 levels is neither part of the data nor part of the program space, and is neither readable nor writable. The activated level is indexed by the Stack Pointer, SP, and is neither readable nor writable. 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.

Mode	Program Counter Bits											
	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Initial Reset	0	0	0	0	0	0	0	0	0	0	0	0
External Interrupt 0	0	0	0	0	0	0	0	0	0	1	0	0
External Interrupt 1	0	0	0	0	0	0	0	0	1	0	0	0
Timer/Event Counter 0 Overflow	0	0	0	0	0	0	0	0	1	1	0	0
Timer/Event Counter 1 Overflow	0	0	0	0	0	0	0	1	0	0	0	0
Time Base Interrupt	0	0	0	0	0	0	0	1	0	1	0	0
RTC Interrupt	0	0	0	0	0	0	0	1	1	0	0	0
Skip	Program Counter + 2											
Loading PCL	PC11	PC10	PC9	PC8	@7	@6	@5	@4	@3	@2	@1	@0
Jump, Call Branch	#11	#10	#9	#8	#7	#6	#5	#4	#3	#2	#1	#0
Return from Subroutine	S11	S10	S9	S8	S7	S6	S5	S4	S3	S2	S1	S0

Program Counter

Note: PC11~PC8: Current Program Counter bits
@7~@0: PCL bits
#11~#0: Instruction code address bits
S11~S0: Stack register bits
The Program Counter is 12 bits wide, i.e. from b11~b0.



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

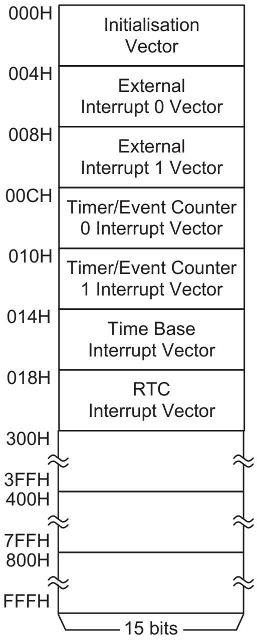
Arithmetic and Logic Unit – ALU

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

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

Program Memory

The Program Memory is the location where the user code or program is stored. For microcontrollers, two types of Program Memory are usually supplied. The first type is the One-Time Programmable, OTP, memory where users can program their application code into the device. Devices with OTP memory are denoted by having an ″R″ within their device name. By using the appropriate programming tools, OTP devices offer users the flexibility to freely develop their applications which may be useful during debug or for products requiring frequent upgrades or program changes. OTP devices are also applicable for use in applications that require low or medium volume production runs. The other type of memory is the mask ROM memory, denoted by having a ″C″ within the device name. These devices offer the most cost effective solutions for high volume products.


000H
Initialisation Vector
004H
External Interrupt 0 Vector
008H
External Interrupt 1 Vector
00CH
Timer/Event Counter 0 Interrupt Vector
010H
Timer/Event Counter 1 Interrupt Vector
014H
Time Base Interrupt Vector
018H
RTC Interrupt Vector
300H
3FFH
400H
7FFH
800H
FFFH
15 bits

Program Memory Structure

Structure

The Program Memory has a capacity of 4K by 15 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 separate table pointer registers.

Special Vectors

Within the Program Memory, certain locations are reserved for special usage such as reset and interrupts.

- Location 000H
  This vector 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.
- Location 004H
  This vector is used by the external interrupt. If the external interrupt pin INT0 on the device receives an active edge, the program will jump to this location and begin execution if the external interrupt is enabled and the stack is not full.
- Location 008H
  This vector is used by the external interrupt. If the external interrupt pin INT1 on the device receives an active edge, the program will jump to this location and begin execution if the external interrupt is enabled and the stack is not full.

- Location 00CH
  This internal vector is used by the Timer/Event Counter 0. If a counter overflow occurs, the program will jump to this location and begin execution if the timer/event counter interrupt is enabled and the stack is not full.
- Location 010H
  This internal vector is used by the Timer/Event Counter 1. If a counter overflow occurs, the program will jump to this location and begin execution if the timer/event counter interrupt is enabled and the stack is not full.
- Location 014H
  This internal vector is used by the Time Base interrupt. If a Time Base interrupt occurs, the program will jump to this location and begin execution if the time base interrupt is enabled and the stack is not full.
- Location 018H
  This internal vector is used by the Real Time Clock interrupt. The program will jump to this location and begin execution when a Real Time Clock interrupt signal is generated if the interrupt is enabled and the stack is not full.

Look-up Table

Any location within the Program Memory can be defined as a look-up table where programmers can store fixed data. To use the look-up table, the table pointer must first be setup by placing the lower order address of the look up data to be retrieved in the table pointer register, TBLP. This register defines the lower 8-bit address of the look-up table.

After setting up the table pointer, the table data can be retrieved from the current Program Memory page or last Program Memory page using the ″TABRDC[m]″ or ″TABRDL [m]″ instructions, respectively. When these instructions are executed, the lower order table byte from the Program Memory will be transferred to the user defined Data Memory register [m] as specified in the instruction. The higher order table data byte from the Program Memory will be transferred to the TBLH special register. Any unused bits in this transferred higher order byte will be read as ″0″.

The following diagram illustrates the addressing/data flow of the look-up table:

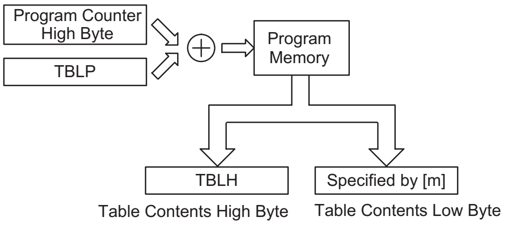


Table Program Example

The following example shows how the table pointer and table data is defined and retrieved from the HT49RA1 microcontroller. This example uses raw table data located in the last page which is stored there using the ORG statement. The value at this ORG statement is ″F00H″ which refers to the start address of the last page within the 4K Program Memory of the HT49RA1 microcontroller. 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 ″TABRDC [m]″ instruction is being used. The high byte of the table data which in this case is equal to zero will be transferred to the TBLH register automatically when the ″TABRDL [m]″ instruction is executed.

Because the TBLH register is a read-only register and cannot be restored, care should be taken to ensure its protection if both the main routine and Interrupt Service Routine use 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.

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

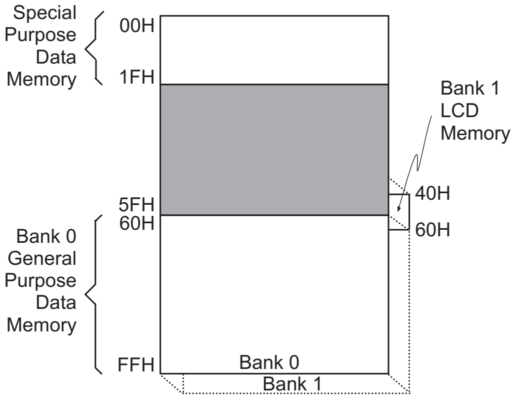
Table Location

Note: PC11~PC8: Current Program Counter bits
@7~@0: Table Pointer TBLP bits
The table address location is 12 bits, i.e. from b11~b0.

```
tempreg1 db    ?        ; temporary register #1
tempreg2 db    ?        ; temporary register #2
    :
    :
mov      a,06h          ; initialise 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 tempregl
                        ; data at prog. 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 prog.memory address "F05H" transferred to
                        ; tempreg2 and TBLH
                        ; in this example the data "1AH" is transferred to
                        ; tempreg1 and data "0FH" to register tempreg2
                        ; the value "00H" will be transferred to the high byte
                        ; register TBLH
         :
         :
org      F00h           ; sets initial address of last page (for HT49RA1)
Dc       00Ah, 00Bh, 00Ch, 00Dh, 00Eh, 00Fh, 01Ah, 01Bh
         :
         :
```

Data Memory

The Data Memory is a volatile area of 8-bit wide RAM internal memory and is the location where temporary information is stored. Divided into three sections, the first of these is an area of RAM where special function registers are located. These registers have fixed locations and are necessary for correct operation of the 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. The third area is reserved for the LCD Memory. This special area of Data Memory is mapped directly to the LCD display so data written into this memory area will directly affect the displayed data. The addresses of the LCD Memory area overlap those in the other Memory areas, switching between the two areas is achieved by setting the Bank Pointer to the correct value.


Special Purpose Data Memory
00H
1FH
Bank 1 LCD Memory
40H
5FH
60H
60H
Bank 0 General Purpose Data Memory
FFH
Bank 0
Bank 1

Data Memory Structure

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

Structure

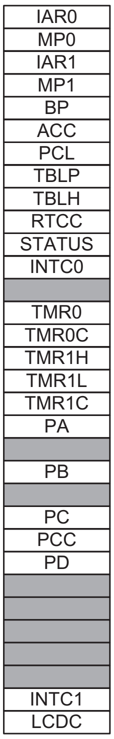
The Special Purpose and General Purpose Data Memory are located at consecutive locations. All are implemented in RAM and are 8 bits wide but the length of each memory section is dictated by the type of microcontroller chosen. The start address of the Data Memory for all devices is the address 00H. Registers which are common to all microcontrollers, such as ACC, PCL, etc., have the same Data Memory address. The LCD Data Memory is mapped into Bank 1 of the Data Memory, however, only the lower four bits are used. The higher four bits, if read by the program will return a zero value. The start of LCD Data Memory for all devices is the address 40H. However, since the LCD Data Memory is located in Bank 1, to access this area the Bank Pointer must first be set to a value of 01H. Note that after power-on the contents of the Data Memory, including the LCD Data Memory, will be in an unknown condition, the programmer must therefore ensure that the Data Memory is properly initialised.

General Purpose Data Memory

All microcontroller programs require an area of read/write memory where temporary data can be stored and retrieved for use later. It is this area of RAM memory that is known as General Purpose Data Memory. This area of Data Memory is fully accessible by the user program for both read and write operations. By using the ″SET [m].i″ and ″CLR [m].i″ instructions individual bits can be set or reset under program control giving the user a large range of flexibility for bit manipulation in the Data Memory. As the General Purpose Data Memory exists in Bank 0, it is necessary to first ensure that the Bank Pointer is set to the correct value before accessing the General Purpose Data Memory. When the Bank Pointer is set to the value 01H, the LCD Memory will be accessed. Bank 1must be addressed indirectly using the Memory Pointer MP1 and the indirect addressing register IAR1. Any direct addressing or any indirect addressing using MP0 and IAR0 will always result in data from Bank 0 being accessed.

Special Purpose Data Memory

This area of Data Memory is where registers, necessary for the correct operation of the microcontroller, are stored. Most of the registers are both readable and 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, any read instruction to these addresses will return the value ″00H″.



IAR0
MP0
IAR1
MP1
BP
ACC
PCL
TBLP
TBLH
RTCC
STATUS
INTC0
▨
TMR0
TMR0C
TMR1H
TMR1L
TMR1C
PA
▨
PB
▨
PC
PCC
PD
▨
▨
▨
▨
▨
INTC1
LCDC

▨ : Unused, read as "00"

Special Purpose Data Memory

LCD Memory

The data to be displayed on the LCD is also stored in an area of fully accessible Data Memory. By writing to this area of RAM, the LCD display output can be directly controlled by the application program. As the LCD Memory exists in Bank 1, but have addresses which map into the Bank 0 Data Memory, it is necessary to first ensure that the Bank Pointer is set to the value 01H before accessing the LCD Memory. The LCD Memory can only be accessed indirectly using the Memory Pointer MP1 and the indirect addressing register IAR1. When the Bank Pointer is set to Bank 1 to access the LCD Data Memory.

Special Function Registers

To ensure successful operation of the microcontroller, certain internal registers are implemented in the Data Memory area. These registers ensure correct operation of internal functions such as timers, interrupts, etc., as well as external functions such as I/O data control. The location of these registers within the Data Memory begins at the address ″00H″. Any unused Data Memory locations between these special function registers and the point where the General Purpose Memory begins is reserved for future expansion purposes, attempting to read data from these locations will return a value of ″00H″.

Indirect Addressing Register – IAR0, IAR1

The IAR0 and IAR1 registers, located at Data Memory addresses ″00H″ and ″02H″, are not physically implemented. These special function registers allows what is known as indirect addressing, which permits data manipulation using Memory Pointers instead of the usual direct memory addressing method where the actual memory address is defined. Any actions on the IAR0 and IAR1 registers will result in corresponding read/write operations to the memory locations specified by the Memory Pointers MP0 and MP1. Reading the IAR0 and IAR1 registers indirectly will return a result of ″00H″ and writing to the register indirectly will result in no operation.

Memory Pointers – MP0, MP1

Two Memory Pointers, known as MP0 and MP1 are provided. These Memory Pointers are physically implemented in the Data Memory and can be manipulated in the same way as normal registers providing a convenient way with which to address and track data. When any operation to the relevant Indirect Addressing Registers is carried out, the actual address that the microcontroller is directed to, is the address specified by the related Memory Pointer. MP0, together with Indirect Addressing Register, IAR0, are used to access data from Bank0, while MP1 and IAR1 are used to access data from Bank0 and Bank1.

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

```
data .section  'data'
adres1    db ?
adres2    db ?
adres3    db ?
adres4    db ?
block     db ?
code .section at 0 'code'
org   00h

start:
    mov  a,04h             ; setup size of block
    mov  block,a
    mov  a,offset adres1   ; Accumulator loaded with first RAM address
    mov  mp0,a             ; setup memory pointer with first RAM address

loop:
    clr  IAR0              ; clear the data at address defined by MP0
    inc  mp0               ; increment memory pointer
    sdz  block             ; check if last memory location has been cleared
    jmp  loop

continue:
```

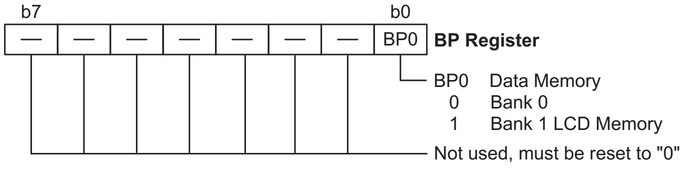
The important point to note here is that in the example shown above, no reference is made to specific RAM addresses.

Bank Pointer – BP

In the Data Memory area it should be noted that both the LCD Memory and the other Data Memory share the same addresses. Therefore when using instructions to access the LCD Memory or the General Purpose Data Memory, it is necessary to ensure that the correct area is selected. The General Purpose is located in Bank 0 while the LCD Memory is located in Bank 1. Selecting the correct Data Memory area is achieved by using the Bank Pointer. If data in Bank 0 is to be accessed then BP should be cleared to zero, while if the LCD Memory is to be accessed, which is located in Bank 1, then BP should be loaded with a value of 01H. It must be noted that data in Bank 1can only be accessed indirectly using the MP1 Memory Pointer and the IAR1 indirect addressing register. Any direct addressing or any indirect addressing using MP0 and IAR0 will always result in data from Bank 0 being accessed. The Data Memory Bank Pointer is initialised to Bank 0 after a reset, except for the WDT time-out reset in the Power Down Mode, in which case, the Data Memory Bank Pointer remains unchanged. It should be noted that the Special Function Data Memory is not affected by the bank selection, which means that the Special Function Registers can be accessed from within either Bank 0 or Bank 1.

Accumulator – ACC

The Accumulator is central to the operation of any 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.



Bank Pointer Register

Program Counter Low Register – PCL

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

Look-up Table Registers – TBLP, TBLH

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

Status Register – STATUS

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

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

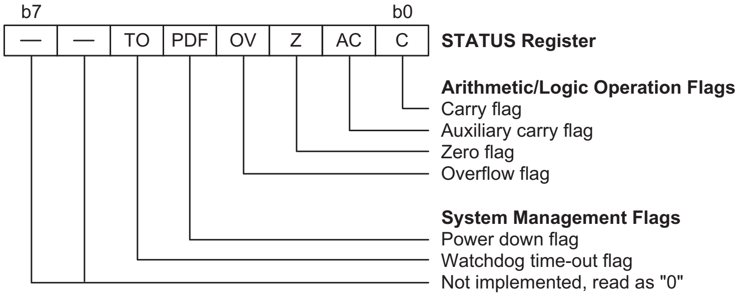
The Z, OV, AC and C flags generally reflect the status of the latest operations.

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

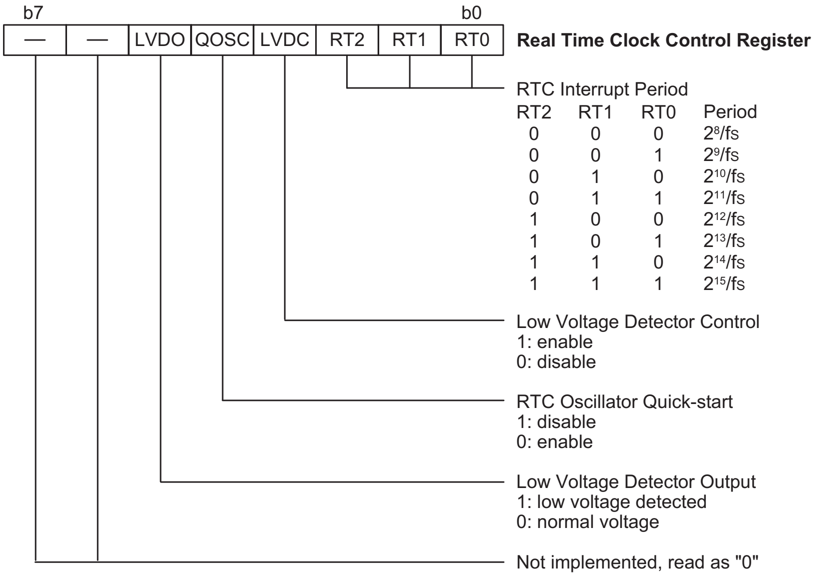
In addition, on entering an interrupt sequence or executing a subroutine call, the status register will not be pushed onto the stack automatically. If the contents of the status registers are important and if the subroutine can corrupt the status register, precautions must be taken to correctly save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

Real Time Clock Control Register – RTCC

The RTCC register controls two internal functions one of which is the Real Time Clock (RTC) interrupt, whose function is to provide an internal interrupt signal at regular fixed intervals. The driving clock for the RTC interrupt comes from the internal clock source, known as f_S, which is then further divided to give longer time values, which in turn generates the interrupt signal. The value of this division ratio is determined by the value programmed into bits 2~0, known as RT2~RT0, of the RTCC register. By writing a value directly into these



Status Register



RT2	RT1	RT0	Period
0	0	0	2^8/fs
0	0	1	2^9/fs
0	1	0	2^{10}/fs
0	1	1	2^{11}/fs
1	0	0	2^{12}/fs
1	0	1	2^{13}/fs
1	1	0	2^{14}/fs
1	1	1	2^{15}/fs

Low Voltage Detector Control
1: enable
0: disable

RTC Oscillator Quick-start
1: disable
0: enable

Low Voltage Detector Output
1: low voltage detected
0: normal voltage

Not implemented, read as "0"

RTCC Register

RTCC register bits, time-out values from $2^8/f_S$ to $2^{15}/f_S$ can be generated. The RTCC register also controls the quick start up function of the RTC oscillator. This oscillator, which has a fixed frequency of 32768Hz, can be made to start up at a quicker rate by setting bit 4, known as the QOSC bit to ″0″. This bit will be set to a ″0″ value when the device is powered on, however, as some extra power is consumed, the QOSC bit should be set to ″1″ after about 2 seconds to reduce power consumption.

Interrupt Control Register – INTC0, INTC1

These 8-bit registers, known as INTC0 and INTC1, control the operation of both the external and internal interrupts. By setting various bits within these registers using standard bit manipulation instructions, the enable/disable function of the external interrupts and each of the internal interrupts can be independently controlled. A master interrupt bit within these registers, the EMI bit, acts like a global enable/disable and is used to set all of the interrupt enable bits on or off. This bit is cleared when an interrupt routine is entered to disable further interrupt and is set by executing the ″RETI″ instruction.

LCDC Register – LCDC

The LCDC register is used as the control register for the LCD panel. The LCDEN bit is the overall on/off control for the LCD driver and can be used to power down the driver and thus used to conserve power when the LCD is not used. As four segment lines are also pin-shared with four Port PD lines, bits SEGPT0~SEGPT3 in the LCDC register are used to determine which function is chosen, either LCD segment line or normal I/O line. This register also contains a bit to control the RTC on/off enable. The RTC on/off control is however also dependent upon which clock is chosen as the internal fs clock source. The accompanying table shows the overall RTC control operation.

LCDEN and RTCEN may decide LCD and RTC On/Off condition on normal operation.

f_S Clock Source	LCD/RTC Control Bits			
	LCDEN, RTCEN=0, 0	LCDEN, RTCEN=0, 1	LCDEN, RTCEN=1, 0	LCDEN, RTCEN=1, 1
f_{SYS}/4	LCD off, RTC off	LCD off, RTC off	LCD on, RTC off	LCD on, RTC off
WDT OSC	LCD off, RTC off	LCD off, RTC off	LCD on, RTC off	LCD on, RTC off
RTC OSC (WDT enable)	LCD off, RTC on	LCD off, RTC on	LCD on, RTC on	LCD on, RTC on
RTC OSC (WDT disable)	LCD off, RTC off	LCD off, RTC on	LCD on, RTC on	LCD on, RTC on

Timer/Event Counter 0/1 Registers – TMR0, TMR0C, TMR1H, TMR1L, TMR1C

All devices possess a single internal 8-bit count-up timer. An associated register known as TMR0 is the location where the timer's 8-bit value is located. This register can also be preloaded with fixed data to allow different time intervals to be setup. An associated control register, known as TMR0C, contains the setup information for this timer, which determines in what mode the timer is to be used as well as containing the timer on/off control function.

All devices possess a single internal 16-bit count-up timer. An associated register known as TMR1H, TMR1L is the location where the timer's 16-bit value is located. This register can also be preloaded with fixed data to allow different time intervals to be setup. An associated control register, known as TMR1C, contains the setup information for this timer, which determines in what mode the timer is to be used as well as containing the timer on/off control function.

Input/Output Ports and Control Registers

Within the area of Special Function Registers, the I/O registers and their associated control registers play a prominent role. All I/O ports have a designated register correspondingly labeled as PA, PB, PC and PD. These labeled I/O registers are mapped to specific addresses within the Data Memory as shown in the Data Memory table, which are used to transfer the appropriate output or input data on that port. One flexible feature of these registers is the ability to directly program single bits using the ″SET [m].i″ and ″CLR [m].i″ instructions. The PC port also has a control register known as PCC, and has the ability to change its I/O pin from output to input and vice versa by manipulating the bit in the register.

Input/Output Ports

Holtek microcontrollers offer considerable flexibility on their I/O ports. Although Port B remains fixed as an input only port, all pins on Port A, Port C and Port D, have the ability to function as either input or output.

The device provides 13 bidirectional input/output lines and 8 input lines. The I/O Ports are known as Port A, Port C and Port D and the input Port is known as Port B. These ports are mapped to the Data Memory with specific addresses as shown in the Special Purpose Data Memory table. The Port A and Port D I/O ports can be used for both input and output operations, however, it must be noted that unlike Port C, they do not have port control registers. Setting up an PA or PD port pin as an input is achieved by first setting its output high which effectively places its NMOS output transistor in a high impedance state allowing the pin to be now used as an input.

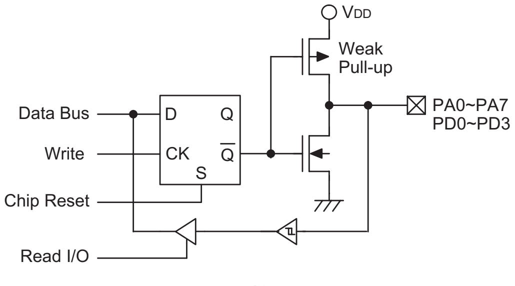
For input operation, these ports are non-latching, which means the inputs must be ready at the T2 rising edge of instruction ″MOV A,[m]″, where m denotes the port address. For output operation, all the data is latched and remains unchanged until the output latch is rewritten.

Pull-high Resistors

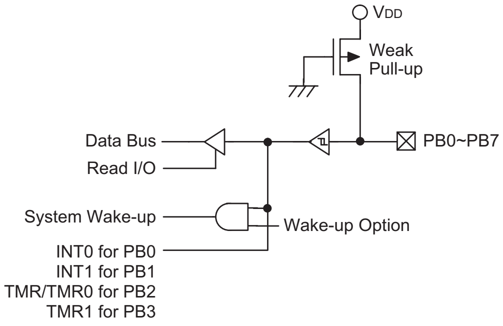
Many product applications require pull-high resistors for their switch inputs usually requiring the use of an external resistor. To eliminate the need for these external resistors, all pins on Port A, Port B and Port D have a permanently connected pull high resistor. The pull high resistor on Port C is chosen via a configuration option. These pull-high resistors are implemented using a weak PMOS transistor.

Port B Wake-up

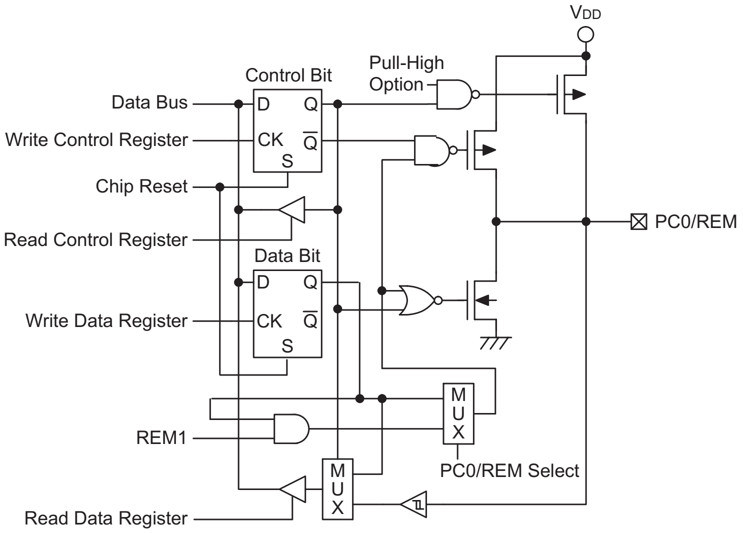
The device has a HALT instruction enabling the microcontroller to enter a Power Down Mode and preserve power, a feature that is important for battery and other low-power applications. Various methods exist to wake-up the microcontroller, one of which is to change the logic condition on one of the Port B pins from high to low. After a ″HALT″ instruction forces the microcontroller into entering a HALT condition, the processor will remain idle or in a low-power state until the logic condition of the selected wake-up pin on Port B changes from high to low. This function is especially suitable for applications that can be woken up via external switches. Note that each pin on Port B can be selected individually to have this wake-up feature.



PA, PD Input/Output Ports



PB Input Port



PC0 Input/Output Port

I/O Port Control Registers

The register PCC is used to control the input/output configuration of port PC. With this control register, this single CMOS output or input with or without pull-high resistor structures can be reconfigured dynamically under software control. The pin of the Port C I/O port is directly mapped to a bit in its associated PCC 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 some pins, the chosen function of the multi-function I/O pins is set by configuration options while for others the function is set by application program control.

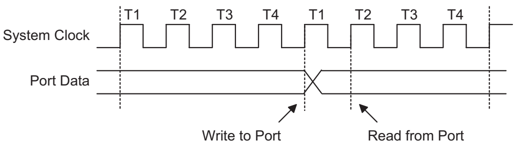
- External Interrupt Input
  The external interrupt pin INT0 or INT1 are pin-shared with the I/O pin PB0 or PB1. For applications not requiring an external interrupt input, the pin-shared external interrupt pin can be used as a normal I/O pin, however to do this, the external interrupt enable bits in the INTC0 register must be disabled.
- External Timer Clock Input
  The external timer pin TMR0 or TMR1 are pin-shared with the I/O pin PB2 or PB3. To configure it to operate as a timer input, the corresponding control bits in the timer control register must be correctly set. For applications that do not require an external timer input, the pin can be used as a normal I/O pin. Note that if used as a normal I/O pin the timer mode control bits in the timer control register must select the timer mode, which has an internal clock source, to prevent the input pin from interfering with the timer operation.

I/O Pin Structures

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

Programming Considerations

Within the application program, one of the first things to consider is port initialization. After a reset,the I/O port registers will be set high. It is important to note that for the NMOS types, when set high the output NMOS transistor will be placed into a high impedance condition, allowing the pin to be used also as an input. The generation of a high level on the NMOS outputs therefore is reliant upon externally connected circuitry and the pull-high resistor.



Read/Write Timing

When using the pin as an output, its logic level can be setup by loading byte wide data into the appropriate port register or by programming individual bits in these registers, 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. However, in the case of NMOS type pins, there are some special considerations that must be noted. In the case of an NMOS pin that is set high by the microcontroller, i.e. placed into a high impedance condition, but driven low by externally connected circuitry, this pin would be read as being in a low condition during the read phase of the ″SET [m].i and ″CLR [m].i″ instructions. When the ensuing write phase occurs, this pin, having been read as being in a low condition during the read phase, would then be consequently erroneously set low. For this reason great care must be taken when using these bit control instructions with NMOS output types.

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

Liquid Crystal Display (LCD) Driver

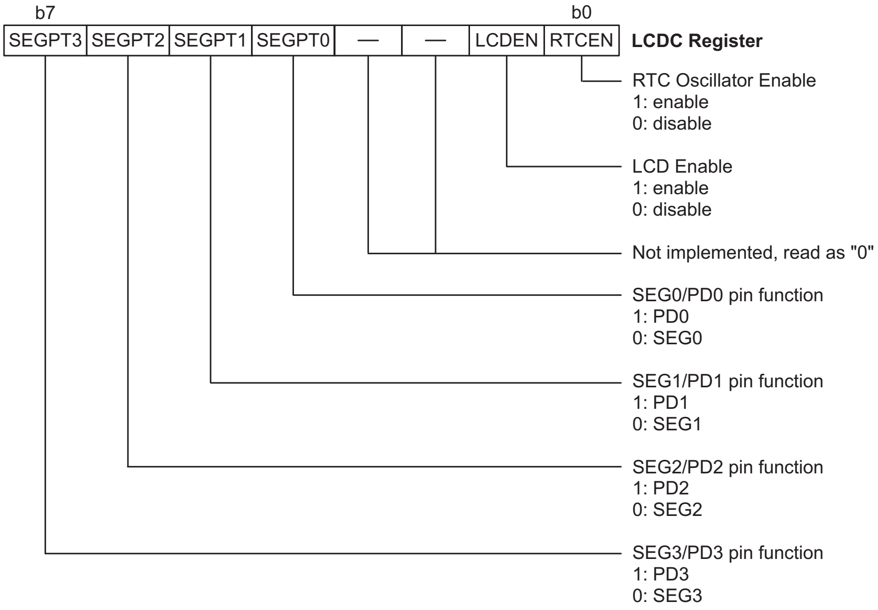
For large volume applications, which incorporate an LCD in their design, the use of a custom display rather than a more expensive character based display reduces costs significantly. However, the corresponding signals required, which vary in both amplitude and time, to drive such a custom display require many special considerations for proper LCD operation to occur. This device includes internal LCD signal generating circuitry and various configuration options, which will automatically generate these time and amplitude varying signals to provide a means of direct driving and easy interfacing to a range of custom LCDs.

LCD Memory

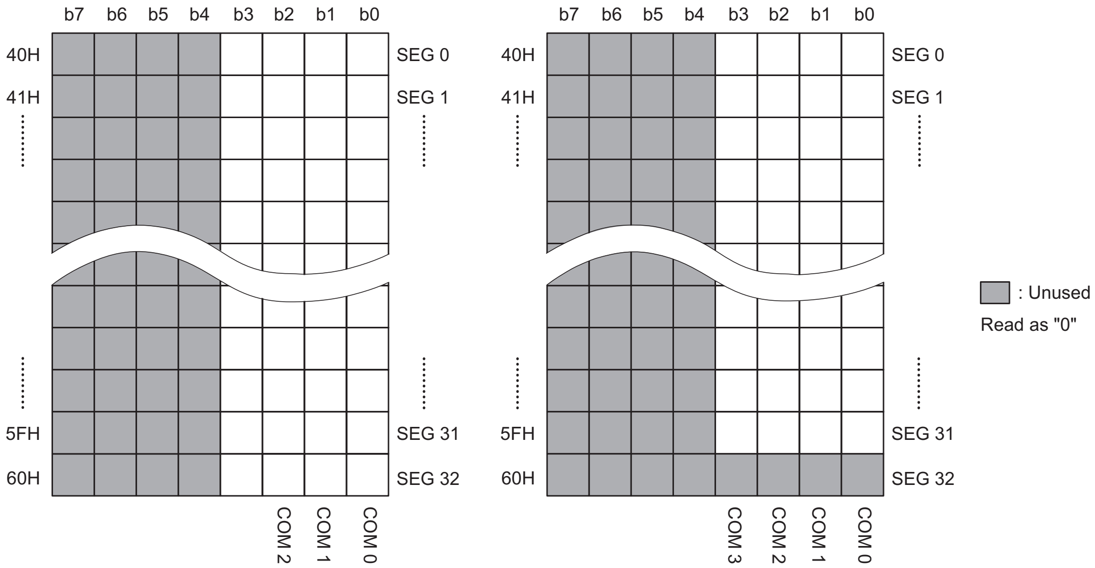
The device provides a specific area of Data Memory for the LCD data. This data area is known as the LCD Memory. Any data written here will be automatically read by the internal LCD driver circuits, which will in turn automatically generate the necessary LCD driving signals. Therefore any data written into the LCD Memory will be immediately reflected into the actual LCD display connected to the microcontroller. The start address of the LCD Memory is 40H, the end address of the LCD Memory is 60H.

As the LCD Data Memory addresses overlap those of the General Purpose Data Memory, the LCD Data Memory is stored in its own memory data bank, which is different from that of the General Purpose Data Memory.

The LCD Data Memory is stored in Bank 1. The Data Memory Bank is chosen by using the Bank Pointer, which is a special function register in the Data Memory, with the name, BP. When the lowest bit of the Bank Pointer have the binary value ″0″, only the General Purpose Data Memory will be accessed, no read or write actions to the LCD Memory will take place. To access the LCD Memory therefore requires first that Bank 1 is selected by setting the lowest bit of the Bank Pointer to



LCD Control Register – LCDC



(1/2 or 1/3 Duty)

(1/4 Duty)

LCD Memory Map

the binary value "1". After this, the LCD Memory can then be accessed by using indirect addressing through the use of Memory Pointer MP1. With Bank 1 selected, then using MP1 to read or write to the memory area, 40H~60H, depending upon which device is chosen, will result in operations to the LCD Memory. Directly addressing the LCD Memory is not applicable and will result in a data access to the Bank 0 General Purpose Data Memory.

The accompanying diagrams show the LCD Memory Map for the 33×2, 33×3 or 32×4 format pixel drive capability. The 4-COM format will be automatically setup if the 1/4 duty configuration option is selected while the 3-COM format will be automatically setup if the 1/2 or 1/3 duty configuration option is selected.

LCD Control Register – LCDC

The device contains a single register known as, LCDC, which is used to control some internal LCD driver functions. The LCDEN bit is the overall on/off control for the LCD driver and can be used to power down the driver and thus used to conserve power when the LCD is not used. As four segment lines are also pin-shared with four Port PD lines, bits SEGPT0~SEGPT3 in the LCDC register are used to determine which function is chosen, either LCD segment line or normal I/O line.

LCD Clock

The LCD clock is driven by the internal clock source f_S, which can originate from either the WDT oscillator, the RTC oscillator or $f_{SYS}/4$, the choice of which is determined by a configuration option. For proper LCD operation, this f_S internal clock source then passes through a divider, to provide an LCD clock source frequency as near as possible to 4kHz.

f_S Clock Source	LCD Clock Selection
WDT Oscillator	WDT/2^2
RTC Oscillator	RTC/2^3
$f_{SYS}/4$	$\frac{f_{SYS}/4}{2^2} \sim \frac{f_{SYS}/4}{2^8}$

LCD Clock Frequency Selection

The available division ratios, however, depends on the clock source that is used for the internal clock source, f_S. If the clock source for f_S originates from the WDT oscillator, then only a fixed division ratio of $f_S/2^2$ is available. If the clock source for f_S originates from the RTC oscillator, then only one division ratio of $f_S/2^3$ is available. However, if the clock source for f_S originates from $f_{SYS}/4$, then a range of LCD clock frequencies are available from $f_S/2^2$ to $f_S/2^8$, the value of which is selected by a further available configuration option. These ratios ensure that for proper LCD operation, a signal frequency as near as possible to 4kHz, can be selected. For an LCD clock frequency of 4kHz, the microcontroller LCD driver circuitry will generate an LCD frame frequency between 55Hz and 62Hz. This is in line with the general LCD operating frequency range which lies between 25Hz and 250Hz. Note that if the selected LCD clock frequency is too high, this will result in a higher than required frame frequency and give rise to higher power consumption while selecting a too low frequency may result in flicker. It is therefore important that if $f_{SYS}/4$ is used as the clock source for f_S, the correct configuration option should be chosen to obtain an LCD clock frequency as close to 4kHz as possible.

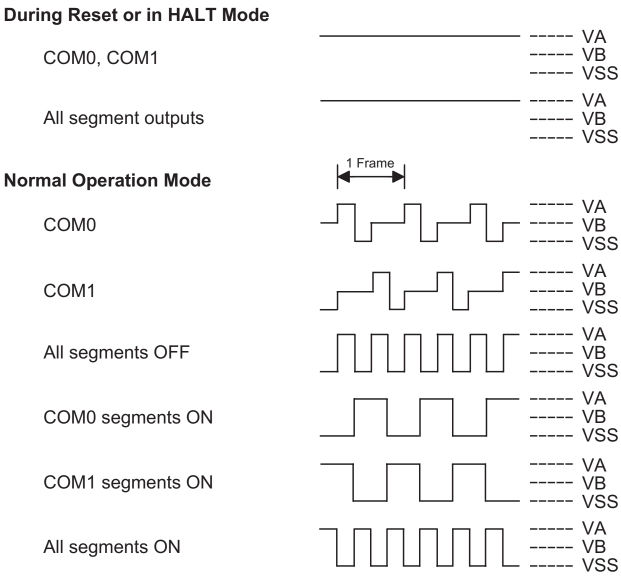
LCD Driver Output

The number of COM and SEG outputs supplied by the LCD driver, as well as its biasing and duty options, are dependent upon the configuration options selected. The accompanying table lists the various options for each of the devices.

Duty	Driver Number	Bias	Bias Type
1/2	33×2	1/2 or 1/3	C type
1/3	33×3	1/2 or 1/3	C type
1/4	32×4	1/2 or 1/3	C type

LCD Driver Outputs, Duty and Bias Options

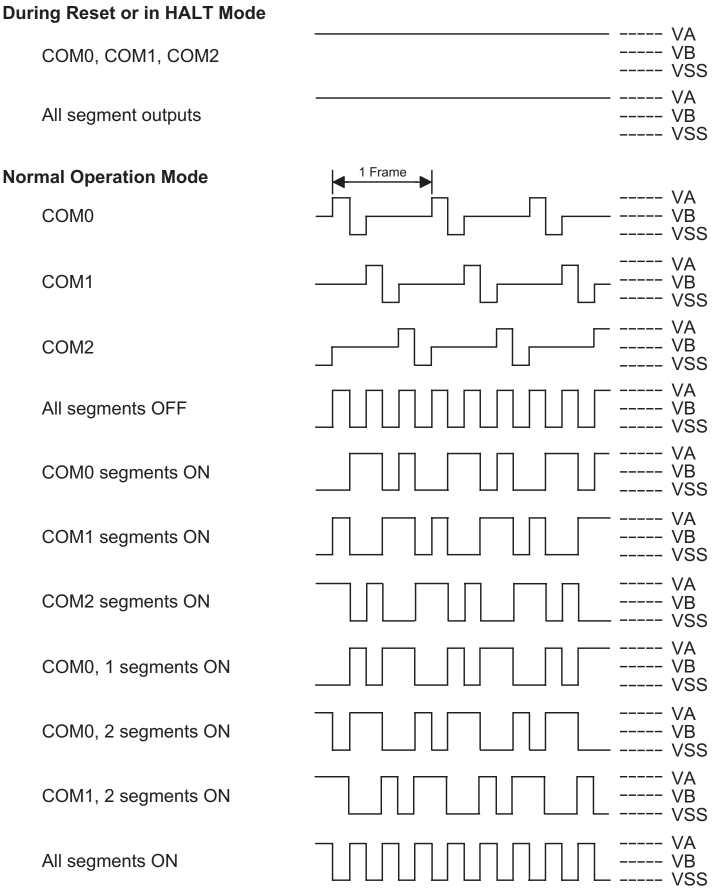
The nature of Liquid Crystal Displays require that only AC voltages can be applied to their pixels as the application of DC voltages to LCD pixels will cause permanent damage. For this reason the relative contrast of an LCD display is controlled by the actual RMS voltage applied to each pixel, which is equal to the RMS value of the voltage on the COM pin minus the voltage applied to the SEG pin. This differential RMS voltage must be greater than the LCD saturation voltage for the pixel to be on and less than the threshold voltage for the pixel to be off. The requirement to limit the DC voltage to zero and to control as many pixels as possible with a minimum number of connections, requires that both a time and amplitude signal is generated and applied to the application LCD. These time and amplitude varying signals are automatically generated by the LCD driver circuits in the microcontroller. What is known as the duty determines the number of common lines used, which are also known as backplanes or COMs. The duty, which is chosen by a configuration option to have a value of 1/2, 1/3 or 1/4 and which equates to a COM number of 2, 3 and 4 respectively, therefore defines the number of time divisions within each LCD signal frame. The accompanying timing diagrams depict the LCD signals generated by the microcontroller for various values of duty and bias.



LCD Driver Output (1/2 Duty, 1/2 Bias)

Note For 1/2 Bias, the VA=VLCD and VB=VLCD×1/2

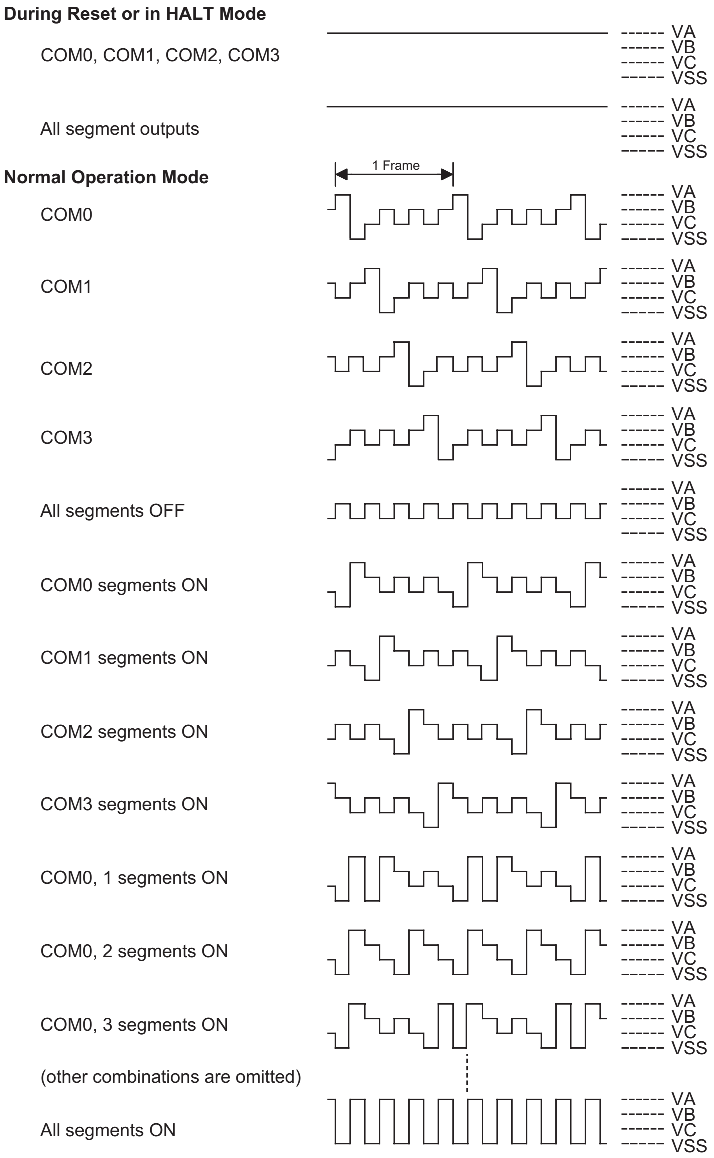
The LCD function can be optioned as on or off during the Power Down Mode by a configuration option.



LCD Driver Output (1/3 Duty, 1/2 Bias)

Note: For 1/2 Bias, the VA=VLCD and VB=VLCD×1/2

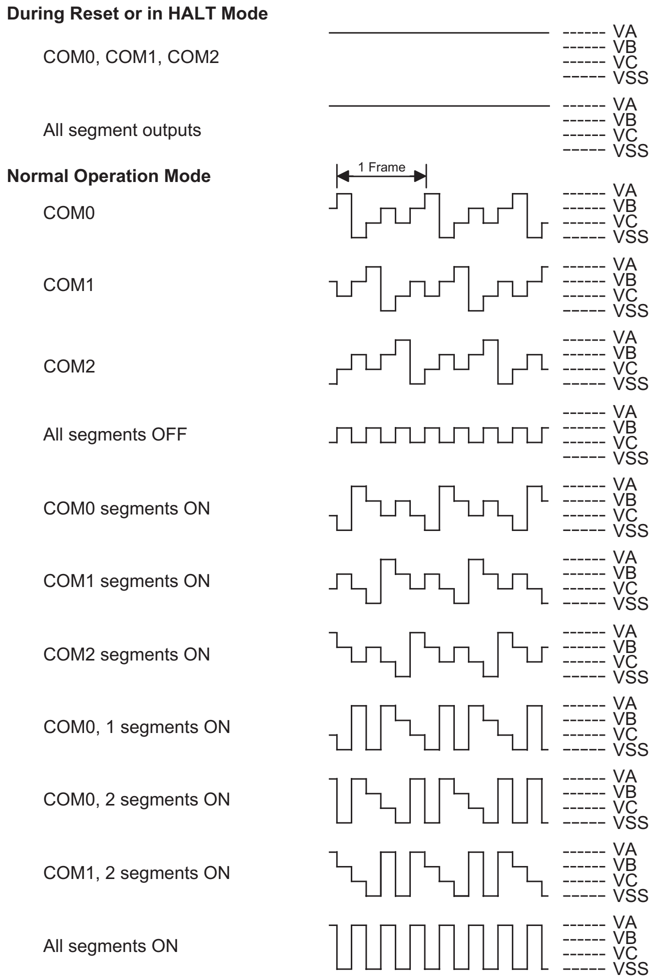
The LCD function can be optioned as on or off during the Power Down Mode by a configuration option.



LCD Driver Output (1/4 Duty, 1/3 Bias)

Note: For 1/3 bias VA=VLCD×1.5, VB=VLCD and VC=VLCD×1/2.

The LCD function can be optioned as on or off during the Power Down Mode by a configuration option.



LCD Driver Output (1/3 Duty, 1/3 Bias)

Note: For 1/3 bias the VA=VLCD×1.5, VB=VLCD and VC=VLCD×1/2.

The LCD function can be optioned as on or off during the Power Down Mode by a configuration option.

LCD Voltage Source and Biasing

The time and amplitude varying LCD signals generated by the microcontroller require the generation of several voltage levels for their operation. The number of voltage levels used by the signal depends upon device and the chosen bias configuration options.

LCD Biasing

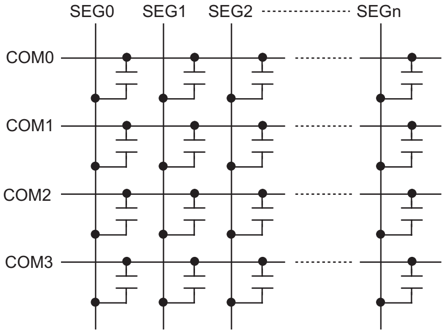
The device has a configuration option to select either 1/2 or 1/3 bias. For the 1/2 bias configuration option, three voltage levels VSS, VA and VB are utilised. VB is generated internally by the microcontroller and will have a value equal to VLCD/2. For the 1/3 bias option, four voltage levels VSS, VA, VB and VC are utilised. An external LCD voltage source is also provided on pin VLCD to generate these voltages. As the C type bias option uses a charge pump circuit, higher voltages than what is provided externally on VLCD can be generated. This feature is useful in applications where the microcontroller supply voltage is less than the supply voltage required by the LCD.

As the LCD driver has a C type bias, a charge-pump capacitor between pins C1 and C2 and filter capacitors on pins V1 and V2 are required to generate the necessary voltage levels.

Programming Considerations

Certain precautions must be taken when programming the LCD. One of these is to ensure that the LCD memory is properly initialized after the microcontroller is powered on. Like the General Purpose Data Memory, the contents of the LCD memory are in an unknown condition after power-on. As the contents of the LCD memory will be mapped into the actual LCD, it is important to initialize this memory area into a known condition soon after applying power to obtain a proper display pattern.

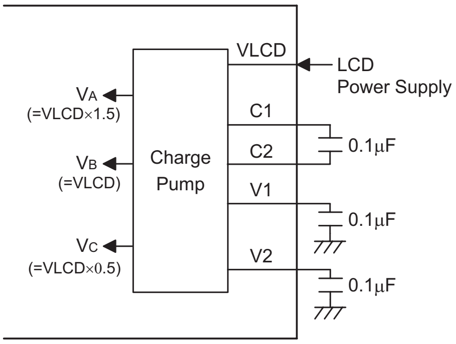
Consideration must also be given to the capacitive load of the actual LCD used in the application. As the load presented to the microcontroller by LCD pixels can be generally modeled as mainly capacitive in nature, it is important that this is not excessive, a point that is particularly true in the case of the COM lines which may be connected to many LCD pixels. The accompanying diagram depicts the equivalent circuit of the LCD.



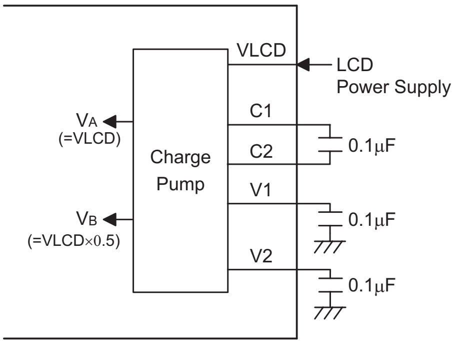
LCD Panel Equivalent Circuit

Setting the correct frequency of the LCD clock is another factor which must be taken into account in user applications. To have the LCDs operate at their best frame frequency, which is normally between 25Hz and 250Hz, it is important to select an appropriate LCD clock frequency configuration option. The correct option should be chosen to ensure that an LCD clock frequency as close to 4kHz as possible is achieved. With such a frequency chosen, the microcontroller internal LCD driver circuits will ensure that the appropriate LCD driving signals are generated to obtain a suitable LCD frame frequency.

Note that as the LCD driver will consume a certain amount of power it can be disabled using the LCDEN bit in the LCDC register. In battery applications where power consumption is an important consideration to prolong battery life, this bit should be used to power down the LCD circuitry to conserve power.



C type 1/3 Bias



C type 1/2 Bias

C Type Bias Voltage Levels

Timer/Event Counters

The provision of timers form an important part of any microcontroller, giving the designer a means of carrying out time related functions. The devices contain one 8-bit and one 16-bit count-up timers. As each timer has three different operating modes, they can be configured to operate as a general timer, an external event counter or as a pulse width measurement device.

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

Configuring the Timer/Event Counter Input Clock Source

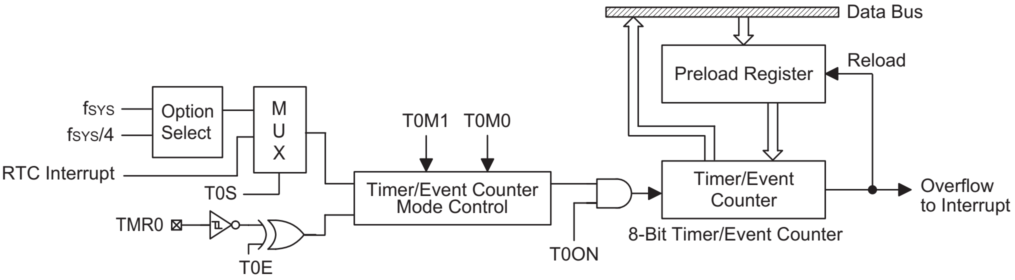
The internal timer′s clock can originate from various sources, depending upon which timer is chosen. The system clock input timer source is used when the timer is in the timer mode or in the pulse width measurement mode.

An external clock source is used when the timer is in the event counting mode, the clock source being provided on an external timer pin TMR0 or TMR1, depending upon which timer is used. Depending upon the condition of the T0E or T1E bit, each high to low, or low to high transition on the external timer pin will increment the counter by one.

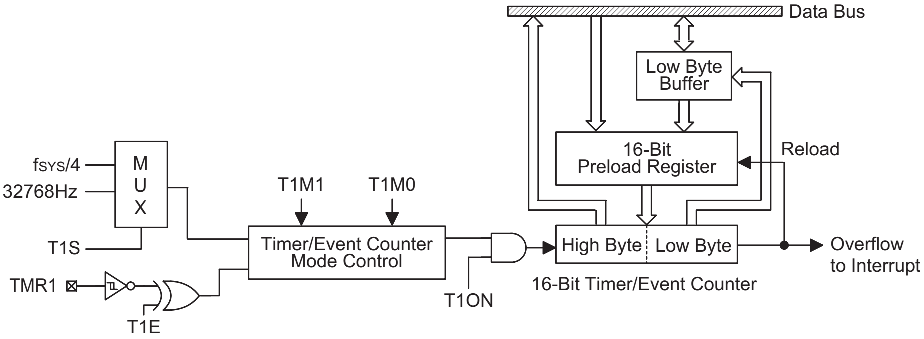
Timer Registers − TMR0, TMR1H, TMR1L

The timer registers are special function registers located in the Special Purpose Data Memory and is the place where the actual timer value is stored. These registers are known as TMR0, TMR1H or TMR1L, depending upon which device is used. The value in the timer registers increases by one each time an internal clock pulse is received or an external transition occurs on the external timer pin. The timer will count from the initial value loaded by the preload register to the full count of FFH or FFFFH at which point the timer overflows and an internal interrupt signal is generated. The timer value will then be reset with the initial preload register value and continue counting.

Note that to achieve a maximum full range count of FFH or FFFFH, the preload register must first be cleared to all zeros. It should be noted that after power-on, the preload registers will be in an unknown condition. Note that if the Timer/Event Counters are in an OFF condition and data is written to their preload registers, this data will be immediately written into the actual counter. However, if the counter is enabled and counting, any new data written into the preload data registers during this period will remain in the preload registers and will only be written into the actual counter the next time an overflow occurs.



Timer/Event Counter 0 Structure



Timer/Event Counter 1 Structure

For the 16-bit Timer/Event Counter which has both low byte and high byte timer registers, accessing these registers is carried out in a specific way. It must be noted when using instructions to preload data into the low byte timer register, namely TMR1L, the data will only be placed in a low byte buffer and not directly into the low byte timer register. The actual transfer of the data into the low byte timer register is only carried out when a write to its associated high byte timer register, namely TMR1H, is executed. On the other hand, using instructions to preload data into the high byte timer register will result in the data being directly written to the high byte timer register. At the same time the data in the low byte buffer will be transferred into its associated low byte timer register. For this reason, the low byte timer register should be written first when preloading data into the 16-bit timer registers. It must also be noted that to read the contents of the low byte timer register, a read to the high byte timer register must be executed first to latch the contents of the low byte timer register into its associated low byte buffer. After this has been done, the low byte timer register can be read in the normal way. Note that reading the low byte timer register will result in reading the previously latched contents of the low byte buffer and not the actual contents of the low byte timer register.

Timer Control Registers – TMR0C, TMR1C

The flexible features of the Holtek microcontroller Timer/Event Counters enable them to operate in three different modes, the options of which are determined by the contents of their respective control register.

It is the Timer Control Register together with its corresponding timer registers that control the full operation of the Timer/Event Counters. Before the timers can be used, it is essential that the appropriate Timer Control Register is fully programmed with the right data to ensure its correct operation, a process that is normally carried out during program initialisation.

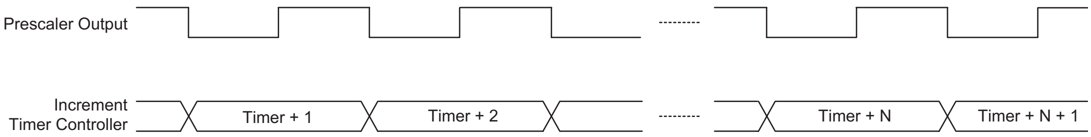
To choose which of the three modes the timer is to operate in, either in the timer mode, the event counting mode or the pulse width measurement mode, bits 7 and 6 of the Timer Control Register, which are known as the bit pair T0M1/T0M0 or T1M1/T1M0 respectively, depending upon which timer is used, must be set to the required logic levels. The timer-on bit, which is bit 4 of the Timer Control Register and known as T0ON or T1ON, depending upon which timer is used, provides the basic on/off control of the respective timer. Setting the bit high allows the counter to run, clearing the bit stops the counter. If the timer is in the event count or pulse width measurement mode, the active transition edge level type is selected by the logic level of bit 3 of the Timer Control Register which is known as T0E or T1E depending upon which timer is used.

Configuring the Timer Mode

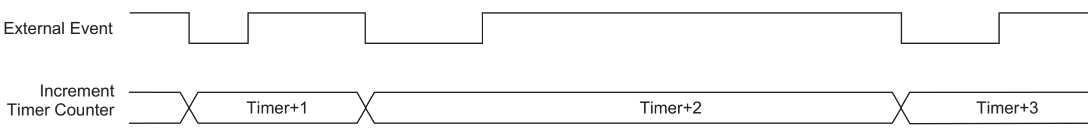
In this mode, the timer can be utilized to measure fixed time intervals, providing an internal interrupt signal each time the counter overflows. To operate in this mode, the bit pair, T0M1/T0M0 or T1M1/T1M0 depending upon which timer is used, must be set to 1 and 0 respectively. In this mode the internal clock is used as the timer clock. The timer-on bit, T0ON or T1ON, depending upon which timer is used, must be set high to enable the timer to run. Each time an internal clock high to low transition occurs, the timer increments by one; when the timer is full and overflows, an interrupt signal is generated and the timer will preload the value already loaded into the preload register and continue counting. A timer overflow condition and corresponding internal interrupt is one of the wake-up sources, however, the internal interrupts can be disabled by ensuring that the ET0I or ET1I bits of the INTC0, INTC1 register are reset to zero.

Configuring the Event Counter Mode

In this mode, a number of externally changing logic events, occurring on the external timer pin, can be recorded by the internal timer. For the timer to operate in the event counting mode, the bit pair, T0M1/T0M0 or T1M1/T1M0 depending upon which timer is used, must be set to 0 and 1 respectively. The timer-on bit T0ON or T1ON depending upon which timer is used, must be set high to enable the timer to count. Depending upon which counter is used, if T0E or T1E is low, the counter will increment each time the external timer pin receives a low

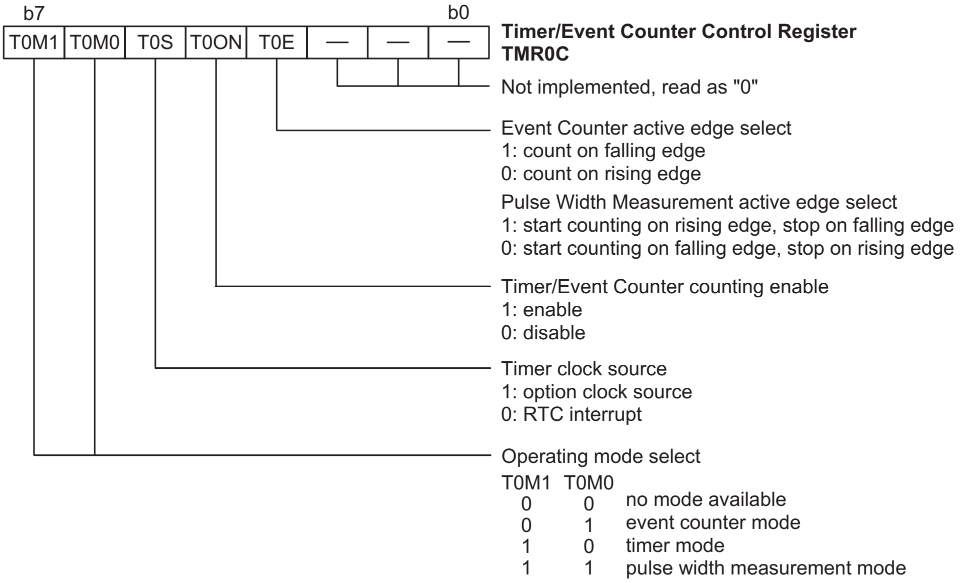


Timer Mode Timing Chart

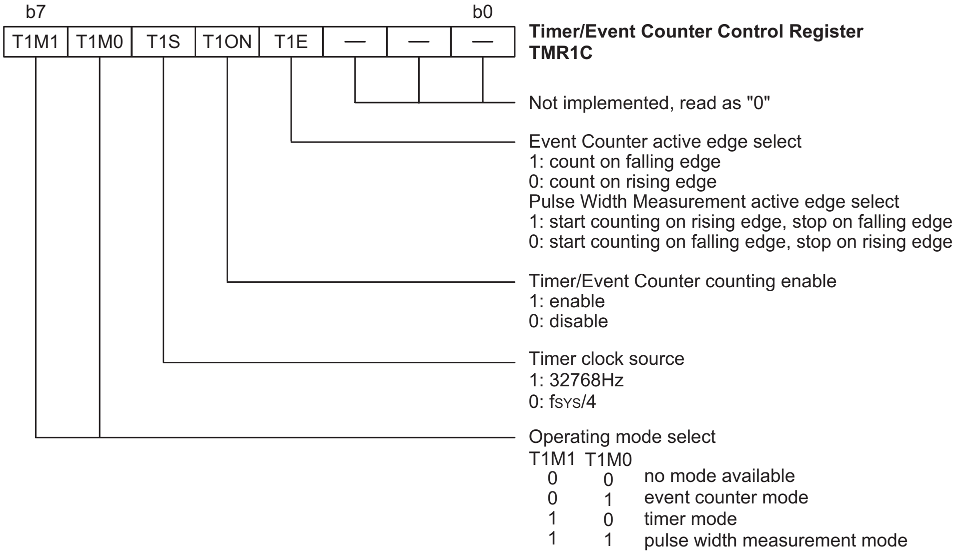


Event Counter Mode Timing Chart

to high transition. If T0E or T1E is high, the counter will increment each time the external timer pin receives a high to low transition. As in the case of the other two modes, when the counter is full, the timer will overflow and generate an internal interrupt signal. The counter will then preload the value already loaded into the preload register. As the external timer pins are pin-shared with other I/O pins, to ensure that the pin is configured to operate as an event counter input pin, two things have to happen. The first is to ensure that the T0M1/T0M0 or T1M1/T1M0 bits place the Timer/Event Counter in the event counting mode, the second is to ensure that the port control register configures the pin as an input. It should be noted that a timer overflow is one of the interrupt and wake-up sources. Note that the timer interrupts can be disabled by ensuring that the ET0I or ET1I bits in the INTC0 or INTC1 register are reset to zero.


b7
b0
T0M1
T0M0
T0S
T0ON
T0E
—
—
—
Timer/Event Counter Control Register
TMR0C
Not implemented, read as "0"
Event Counter active edge select
1: count on falling edge
0: count on rising edge
Pulse Width Measurement active edge select
1: start counting on rising edge, stop on falling edge
0: start counting on falling edge, stop on rising edge
Timer/Event Counter counting enable
1: enable
0: disable
Timer clock source
1: option clock source
0: RTC interrupt
Operating mode select
T0M1 T0M0
0 0 no mode available
0 1 event counter mode
1 0 timer mode
1 1 pulse width measurement mode

Timer/Event Counter 0 Control Register – TMR0C


b7
b0
T1M1
T1M0
T1S
T1ON
T1E
—
—
—
Timer/Event Counter Control Register
TMR1C
Not implemented, read as "0"
Event Counter active edge select
1: count on falling edge
0: count on rising edge
Pulse Width Measurement active edge select
1: start counting on rising edge, stop on falling edge
0: start counting on falling edge, stop on rising edge
Timer/Event Counter counting enable
1: enable
0: disable
Timer clock source
1: 32768Hz
0: fSYS/4
Operating mode select
T1M1 T1M0
0 0 no mode available
0 1 event counter mode
1 0 timer mode
1 1 pulse width measurement mode

Timer/Event Counter 1 Control Register – TMR1C

Configuring the Pulse Width Measurement Mode

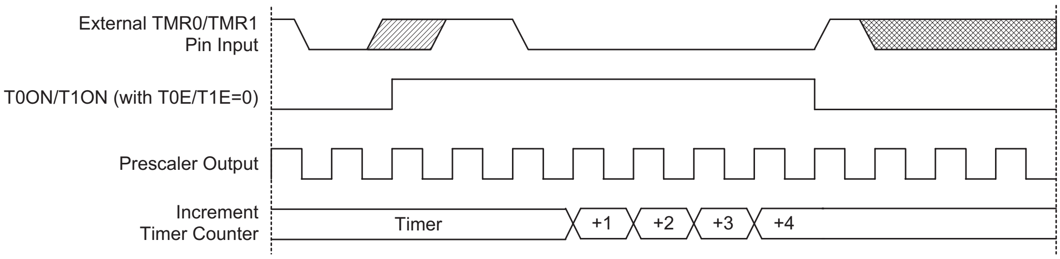
In this mode, the width of external pulses applied to the external timer pin can be measured. In the Pulse Width Measurement Mode the timer clock source is supplied by the internal clock. For the timer to operate in this mode, the bit pair, T0M1/T0M0 or T1M1/T1M0, depending upon which timer is used, must both be set high. Depending upon which counter is used, if the T0E or T1Ebit is low, once a high to low transition has been received on the external timer pin, the timer will start counting until the external timer pin returns to its original high level. At this point the T0ON or T1ON bit, depending upon which counter is used, will be automatically reset to zero and the timer will stop counting. If the T0E or T1E bit is high, the timer will begin counting once a low to high transition has been received on the external timer pin and stop counting when the external timer pin returns to its original low level. As before, the T0ON or T1ON, bit will be automatically reset to zero and the timer will stop counting. It is important to note that in the Pulse Width Measurement Mode, the T0ON or T1ON bit is automatically reset to zero when the external control signal on the external timer pin returns to its original level, whereas in the other two modes the T0ON or T1ON bit can only be reset to zero under program control. The residual value in the timer, which can now be read by the program, therefore represents the length of the pulse received on the external timer pin. As the T0ON or T1ON bit has now been reset, any further transitions on the external timer pin, will be ignored. Not until the T0ON or T1ON bit is again set high by the program can the timer begin further pulse width measurements. In this way, single shot pulse measurements can be easily made. It should be noted that in this mode the counter is controlled by logical transitions on the external timer pin and not by the logic level. As in the case of the other two modes, when the counter is full, the timer will overflow and generate an internal interrupt signal. The counter will also be reset to the value already loaded into the preload register. If the external timer pin is pin-shared with other I/O pins, to ensure that the pin is configured to operate as a pulse width measuring input pin, two things have to happen. The first is to ensure that the T0M1/T0M0 or T1M1/T1M0 bits place the Timer/Event Counter in the pulse width measuring mode, the second is to ensure that the port control register configures the pin as an input. It should be noted that a timer overflow and corresponding timer interrupt is one of the wake-up sources. Note that the timer interrupts can be disabled by ensuring that the ET0I or ET1I bits in the INTC0 or INTC1 register are reset to zero.

I/O Interfacing

The Timer/Event Counter, when configured to run in the event counter or pulse width measurement mode, require the use of the external pin for correct operation. As this pin is a shared pin it must be configured correctly to ensure it is setup for use as a Timer/Event Counter input and not as a normal I/O pin. This is implemented by ensuring that the mode select bits in the Timer/Event Counter control register, select either the event counter or pulse width measurement mode. Additionally the Port Control Register must be set high to ensure that the pin is setup as an input. Any pull-high resistor on this pin will remain valid even if the pin is used as a Timer/Event Counter input.

Programming Considerations

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



Prescaler Output is sampled at every falling edge of T1.

Pulse Width Measure Mode Timing Chart

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

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

Timer Program Example

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

```
org 04h             ; external interrupt vector
reti
org 0Ch             ; Timer/Event Counter 0 interrupt vector
jmp tmrint          ; jump here when Timer overflows
:
org 20h             ; main program
;internal Timer/Event Counter 0 interrupt routine
tmrint:
:
; Timer/Event Counter 0 main program placed here
:
reti
:
:

begin:
;setup Timer 0 registers
mov a,09bh          ; setup Timer 0 preload value
mov tmr0,a;
mov a,080h          ; setup Timer 0 control register
mov tmr0c,a         ; timer mode
; setup interrupt register
mov a,009h          ; enable master interrupt and timer interrupt
mov int0c,a
set tmr0c.4         ; start Timer/Event Counter 0 - note mode bits must be previously setup
```

Carrier Generator

Some remote control transmitter applications require a carrier frequency generator to transmit the remote control signal at the appropriate frequency to the receiving device. These devices include an internal carrier frequency generator for this purpose, the frequency of which is specified by selecting the correct configuration options.

This carrier signal is supplied on the REM pin, which is also pin-shared with PC0. The selection of the required function, whether remote output or CMOS output, is implemented by selecting the required configuration option. If the remote output REM is selected by configuration option, the REM output will be activated if the PC0 data bit in the PC register is set to high. This output data bit is used as the on/off control bit for the REM output. Note that the REM output will be low if the PC0 output data bit is set to zero. However, if the line is configured as a PC0 output pin it will switch to a high level and remain so until the application program resets the pin to a zero. It is therefore important to note that, for OTP devices, if the pin is configured as a PC0 output pin, and a NPN transistor is connected to this output to drive an infrared LED, the LED will be turned on during this power-on reset period. For general purpose remote controller applications, it is therefore recommended that the REM configuration option is selected together with an external NPN transistor to drive the infrared LED.

The clock source for the Carrier Generator is supplied by the system clock divided by 4. By selecting values for ″m″ and ″n″ using configuration options in association with the following equation the required carrier frequency can be generated.

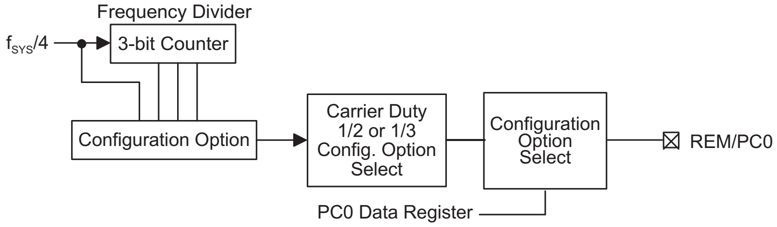
$$\text{Carrier Frequency} = \frac{\text{Clock Source}}{\text{m}\text{x}2^n}$$

The value of ″m″ can be either 2 or 3 while the value of ″n″ can range from 0 to 3, both values are chosen by selecting the required configuration options. If ″m″ is equal to ″2″ the duty cycle of the output waveform will always be equal to 1/2. If ″m″ is equal to ″3″, with the exception of ″n″ being equal to ″0″, the duty cycle can be either 1/2 or 1/3, the actual value of which is determined by configuration options.

$m\times2^n$	Duty Cycle
2, 4, 8, 16	1/2
3	1/3
6, 12, 24	1/2 or 1/3

The following table shows examples of different carrier frequencies:

f_{SYS}	$f_{CARRIER}$	Duty	$m\times2^n$
455kHz	37.92kHz	1/3 only	3
	56.9kHz	1/2 only	2



Carrier Signal Generation

Interrupts

Interrupts are an important part of any microcontroller system. When an external event or an internal function such as a Timer/Event Counter, Time Base or RTC Interrupt 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 two external interrupts and four internal interrupt functions. The external interrupt is controlled by the action of the external INT0, INT1 pin, while the internal interrupts are controlled by the two Timer/Event Counter overflows, the Time Base interrupt and the RTC interrupt.

Interrupt Register

Overall interrupt control, which means interrupt enabling and request flag setting, is controlled by the INTC0 and INTC1 registers, which are located in the Data Memory. By controlling the appropriate enable bits in these register each individual interrupt can be enabled or disabled. Also when an interrupt occurs, the corresponding request flag will be set by the microcontroller. The global enable flag if cleared to zero will disable all interrupts.

Interrupt Operation

A Timer/Event Counter overflow, Time Base or RTC overflow or the external interrupt line being triggered will all generate an interrupt request by setting their corresponding request flag, if their appropriate interrupt enable bit is set. When this happens, 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 statement 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 statement, 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 diagram with their order of priority.

Once an interrupt subroutine is serviced, all the other interrupts will be blocked, as the EMI bit will be cleared automatically. This will prevent any further interrupt nesting from occurring. However, if other interrupt requests occur during this interval, although the interrupt will not be immediately serviced, the request flag will still be recorded. If an interrupt requires immediate servicing while the program is already in another interrupt service routine, the EMI bit should be set after entering the routine, to allow interrupt nesting. If the stack is full, the interrupt request will not be acknowledged, even if the related interrupt is enabled, until the Stack Pointer is decremented. If immediate service is desired, the stack must be prevented from becoming full.

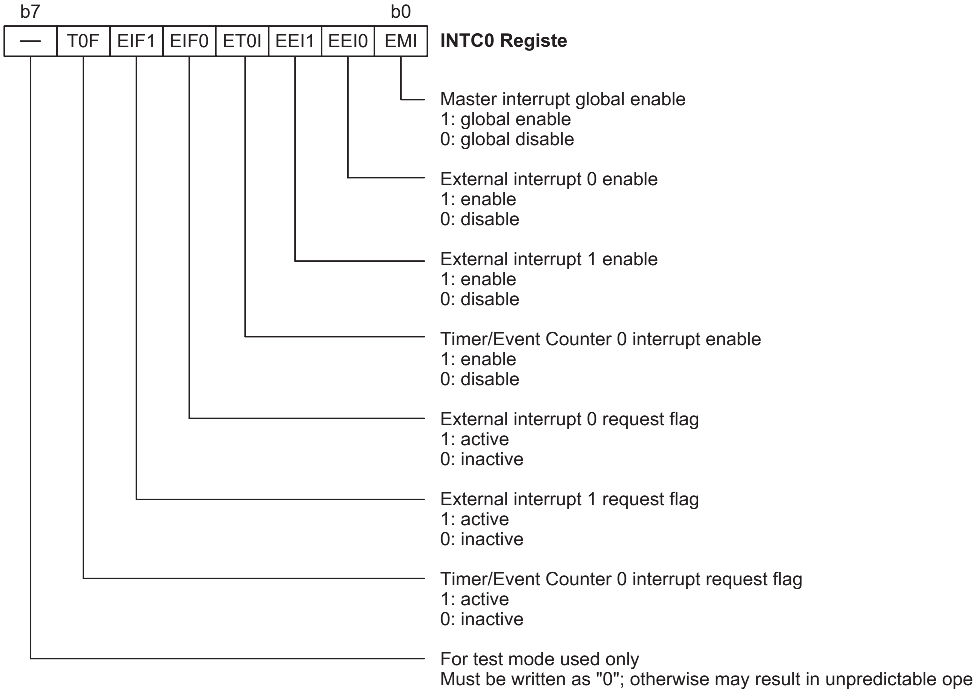
Interrupt Priority

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

Interrupt Source	Priority
External Interrupt 0/1	1/2
Timer/Event Counter 0/1 Overflow	3/4
Time Base Interrupt	5
Real Time Clock Interrupt	6

External Interrupt

For an external interrupt to occur, the global interrupt enable bit, EMI, and external interrupt enable bit, EEI0, EEI1, must first be set. Additionally the correct interrupt edge bit must be selected to enable the external interrupt function and to choose the trigger edge type. An actual external interrupt will take place when the external interrupt request flag, EIF0 or EIF1, is set, a situation that will occur when a transition, whose type is chosen by configuration option appears on the INT0 and, INT1 pins. The external interrupt pins are pin-shared with the I/O pins PB0 and PB1 and can only be configured as an external interrupt pin if the corresponding external interrupt enable bit in the INTC0 register have been set. The pins must also be setup as inputs. 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 relevant external interrupt vectors at locations 04H and 08H, will take place. When the interrupt is serviced, the external interrupt request flag, EIF0, EIF1, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts.



b7							b0	
—	T0F	EIF1	EIF0	ET0I	EEI1	EEI0	EMI	**INTC0 Registe**

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

EEI0: External interrupt 0 enable
1: enable
0: disable

EEI1: External interrupt 1 enable
1: enable
0: disable

ET0I: Timer/Event Counter 0 interrupt enable
1: enable
0: disable

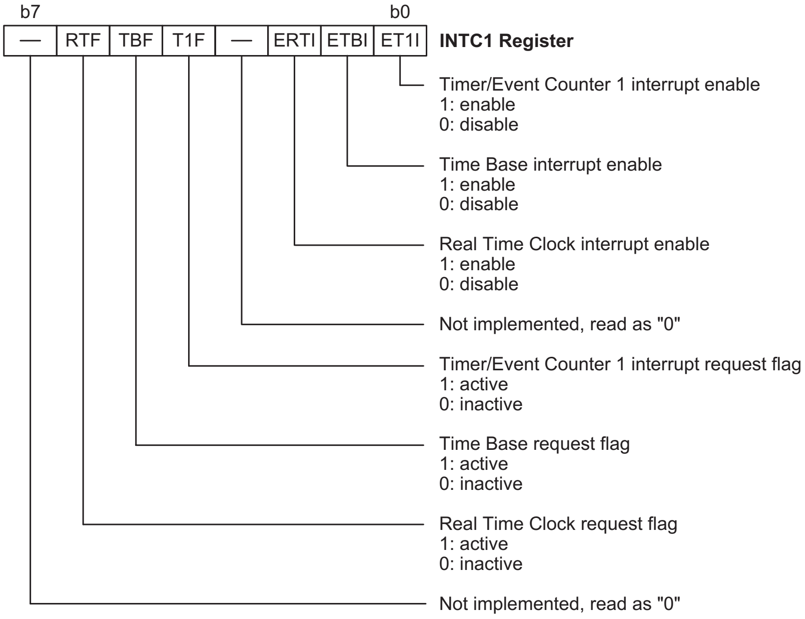
EIF0: External interrupt 0 request flag
1: active
0: inactive

EIF1: External interrupt 1 request flag
1: active
0: inactive

T0F: Timer/Event Counter 0 interrupt request flag
1: active
0: inactive

—: For test mode used only
Must be written as "0"; otherwise may result in unpredictable ope

Interrupt Control Register INTC0



b7							b0	
—	RTF	TBF	T1F	—	ERTI	ETBI	ET1I	**INTC1 Register**

ET1I: Timer/Event Counter 1 interrupt enable
1: enable
0: disable

ETBI: Time Base interrupt enable
1: enable
0: disable

ERTI: Real Time Clock interrupt enable
1: enable
0: disable

—: Not implemented, read as "0"

T1F: Timer/Event Counter 1 interrupt request flag
1: active
0: inactive

TBF: Time Base request flag
1: active
0: inactive

RTF: Real Time Clock request flag
1: active
0: inactive

—: Not implemented, read as "0"

Interrupt Control Register INTC1

Timer/Event Counter Interrupt

For a Timer/Event Counter interrupt to occur, the global interrupt enable bit, EMI, and the corresponding timer interrupt enable bit, ET0I, ET1I must first be set. An actual Timer/Event Counter interrupt will take place when the relevant Timer/Event Counter request flag, T0F, T1F is set, a situation that will occur when the relevant Timer/Event Counter overflows. When the interrupt is enabled, the stack is not full and a Timer/Event Counter overflow occurs, a subroutine call to the timer interrupt vector at location 0CH, 10H, will take place. When the interrupt is serviced, the timer interrupt request flag, T0F, T1F will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts.

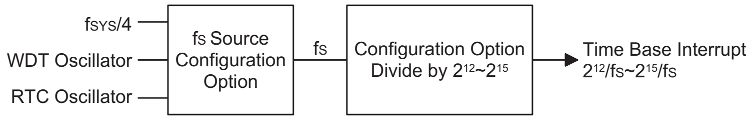
Time Base Interrupt

For a Time Base interrupt to occur the the global interrupt enable bit, EMI, and the corresponding internal interrupt enable bit, which is bit 1 of the INTC1 register, known as ETBI, must be first set. An actual Time Base interrupt will be generated when the Time Base interrupt request flag is set which is bit 5 of the INTC1 register and known as TBF. This will occur when when a time-out signal is generated from the Time Base. When the master interrupt global enable bit is set, the stack is not full and the corresponding Time Base interrupt enable bit is set, an internal Time Base interrupt will be generated when a time-out signal is generated from the Time Base. This will create a subroutine call to location 014H. When a Time Base interrupt occurs, the EMI bit will be cleared to disable other interrupts. The purpose of the Time Base interrupt is to provide an interrupt signal at fixed time periods. The Time Base interrupt clock source originates from the internal clock source f_S. This f_S input clock first passes through a divider, the division ratio of which is selected by configuration options to provide longer Time Base interrupt periods. The Time Base interrupt time-out period ranges from $2^{12}/f_S$~$2^{15}/f_S$. The clock source that generates f_S, which in turn controls the Time Base interrupt period, can originate from three different sources, the RTC oscillator, Watchdog Timer oscillator or the System oscillator/4, the choice of which is determine by the f_S clock source configuration option.

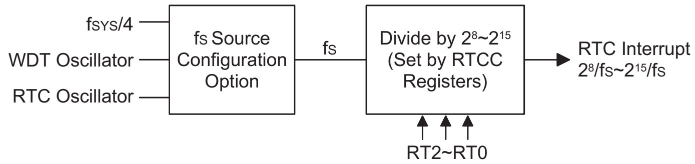
Real Time Clock Interrupt

For a Real Time Clock interrupt to occur the global interrupt enable bit, EMI, and the corresponding internal interrupt enable bit, which is bit 2 of the INTC1 register, known as ERTI, must be first set. An actual Real Time Clock interrupt will be generated when the Real Time Clock interrupt request flag is set which is bit 6 of the INTC1 register and known as RTF. When the master interrupt global enable bit is set, the stack is not full and the corresponding Real Time Clock interrupt enable bit is set, an internal Real Time Clock interrupt will be generated when a time-out signal occurs, a subroutine call to location 018H will be created. When a Real Time interrupt occurs, the EMI bit will be cleared to disable other interrupts.

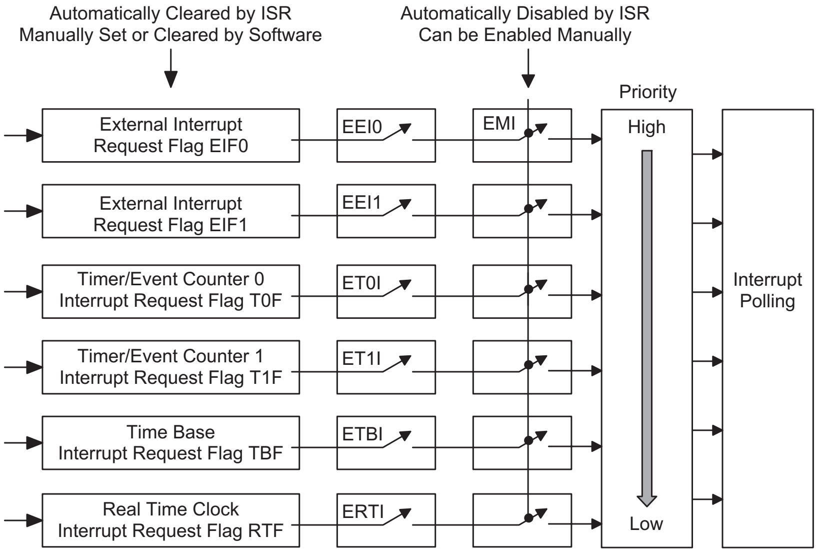
Similar in operation to the Time Base interrupt, the purpose of the RTC interrupt is also to provide an interrupt signal at fixed time periods. The RTC interrupt clock source originates from the internal clock source f_S. This f_S input clock first passes through a divider, the division ratio of which is selected by programming the appropriate bits in the RTCC register to obtain longer RTC interrupt periods whose value ranges from $2^8/f_S$~$2^{15}/f_S$. The clock source that generates f_S, which in turn controls the RTC interrupt period, can originate from three different sources, the RTC oscillator, Watchdog Timer oscillator or the System oscillator/4, the choice of which is determine by the f_S clock source configuration option. Note that if the RTC oscillator is selected as the system clock, then f_S, and correspondingly the RTC interrupt, will also have the RTC oscillator as its clock source.


fSYS/4
WDT Oscillator
RTC Oscillator
fS Source Configuration Option
fS
Configuration Option Divide by 2¹²~2¹⁵
Time Base Interrupt 2¹²/fS~2¹⁵/fS

Time Base Interrupt


fSYS/4
WDT Oscillator
RTC Oscillator
fS Source Configuration Option
fS
Divide by 2⁸~2¹⁵ (Set by RTCC Registers)
RT2~RT0
RTC Interrupt 2⁸/fS~2¹⁵/fS

RTC Interrupt


Automatically Cleared by ISR
Manually Set or Cleared by Software
Automatically Disabled by ISR
Can be Enabled Manually
Priority
External Interrupt Request Flag EIF0
EEI0
EMI
High
External Interrupt Request Flag EIF1
EEI1
Timer/Event Counter 0 Interrupt Request Flag T0F
ET0I
Interrupt Polling
Timer/Event Counter 1 Interrupt Request Flag T1F
ET1I
Time Base Interrupt Request Flag TBF
ETBI
Real Time Clock Interrupt Request Flag RTF
ERTI
Low

Interrupt Structure

Note that the RTC interrupt period is controlled by both configuration options and an internal register RTCC. A configuration option selects the source clock for the internal clock f_S, and the RTCC register bits RT2, RT1 and RT0 select the division ratio. Note that the actual division ratio can be programmed from 2^8 to 2^{15}. For details of the actual RTC interrupt periods, consult the RTCC register section.

Note after a wake-up the system requires 1024 clock cycles to resume normal operation.

Programming Considerations

By disabling the 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 INTC0, INTC1 registers until the corresponding interrupt is serviced or until the request flag is cleared by a software instruction.

It is recommended that programs do not use the ″CALL subroutine″ instruction within the interrupt subroutine. Interrupts often occur in an unpredictable manner or need to be serviced immediately in some applications. 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.

All of these interrupts have the capability of waking up the processor when in the Power Down Mode.

Only the Program Counter is pushed onto the stack. If the contents of the register or status register are altered by the interrupt service program, which may corrupt the desired control sequence, then the contents should be saved in advance.

Reset and Initialisation

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

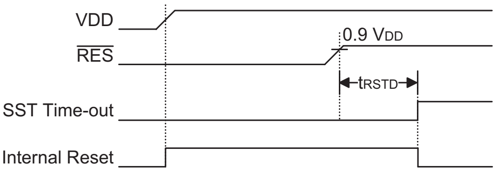
In addition to the power-on reset, situations may arise where it is necessary to forcefully apply a reset condition when the microcontroller is running. One example of this is where after power has been applied and the microcontroller is already running, the $\overline{\text{RES}}$ line is forcefully pulled low. In such a case, known as a normal operation reset, some of the microcontroller registers remain unchanged allowing the microcontroller to proceed with normal operation after the reset line is allowed to return high. Another type of reset is when the Watchdog Timer overflows and resets the microcontroller. All types of reset operations result in different register conditions being setup.

Another reset exists in the form of a Low Voltage Reset, LVR, where a full reset, similar to the $\overline{\text{RES}}$ reset is implemented in situations where the power supply voltage falls below a certain threshold.

Reset Functions

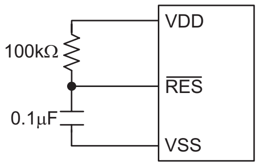
There are five ways in which a microcontroller reset can occur, through events occurring both internally and externally:

- Power-on Reset
  The most fundamental and unavoidable reset is the one that occurs after power is first applied to the microcontroller. As well as ensuring that the Program Memory begins execution from the first memory address, a power-on reset also ensures that certain other registers are preset to known conditions. All the I/O port and port control registers will power up in a high condition ensuring that all pins will be first set to inputs.
  Although the microcontroller has an internal RC reset function, if the VDD power supply rise time is not fast enough or does not stabilise quickly at power-on, the internal reset function may be incapable of providing proper reset operation. For this reason it is recommended that an external RC network is connected to the $\overline{\text{RES}}$ pin, whose additional time delay will ensure that the $\overline{\text{RES}}$ pin remains low for an extended period to allow the power supply to stabilise. During this time delay, normal operation of the microcontroller will be inhibited. After the $\overline{\text{RES}}$ line reaches a certain voltage value, the reset delay time t_{RSTD} is invoked to provide an extra delay time after which the microcontroller will begin normal operation. The abbreviation SST in the figures stands for System Start-up Timer.



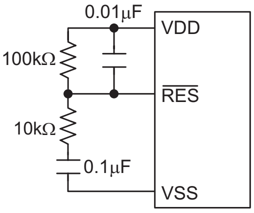
Power-On Reset Timing Chart

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



Basic Reset Circuit

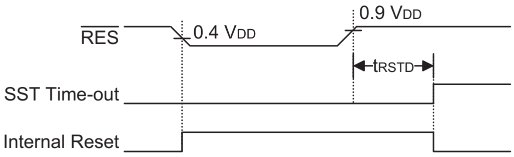
For applications that operate within an environment where more noise is present the Enhanced Reset Circuit shown is recommended.



Enhanced Reset Circuit

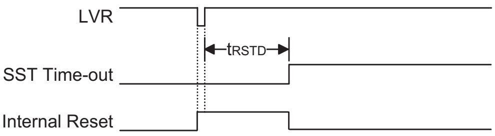
More information regarding external reset circuits is located in Application Note HA0075E on the Holtek website.

- $\overline{\text{RES}}$ Pin Reset
  This type of reset occurs when the microcontroller is already running and the $\overline{\text{RES}}$ pin is forcefully pulled low by external hardware such as an external switch. In this case as in the case of other reset, the Program Counter will reset to zero and program execution initiated from this point.



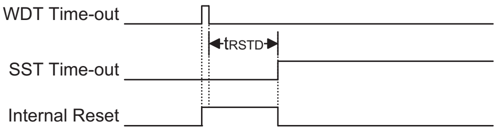
$\overline{\text{RES}}$ Reset Timing Chart

- Low Voltage Reset – LVR
  The microcontroller contains a low voltage reset circuit in order to monitor the supply voltage of the device, which is selected via a configuration option. 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. The LVR includes the following specifications: For a valid LVR signal, a low voltage, i.e., a voltage in the range between 0.9V~V_{LVR} must exist for greater than the value t_{LVR} specified in the A.C. characteristics. If the low voltage state does not exceed 1ms, the LVR will ignore it and will not perform a reset function.



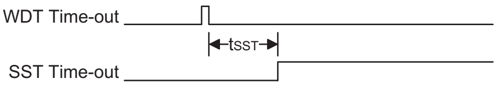
Low Voltage Reset Timing Chart

- Watchdog Time-out Reset during Normal Operation
  The Watchdog time-out Reset during normal operation is the same as a hardware $\overline{\text{RES}}$ pin reset except that the Watchdog time-out flag TO will be set to ″1″.



WDT Time-out Reset during Normal Operation Timing Chart

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



WDT Time-out Reset during Power Down Timing Chart

Reset Initial Conditions

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

TO	PDF	RESET Conditions
0	0	$\overline{\text{RES}}$ reset during power-on
u	u	$\overline{\text{RES}}$ or LVR reset during normal operation
1	u	WDT time-out reset during normal operation
1	1	WDT time-out reset during Power Down

Note: ″u″ stands for unchanged

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

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

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

Register	Reset (Power-on)	$\overline{\text{RES}}$ or LVR Reset	WDT Time-out (Normal Operation)	WDT Time-out (HALT)
MP0	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
MP1	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
BP	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	–xxx xxxx	–uuu uuuu	–uuu uuuu	–uuu uuuu
RTCC	– –00 0111	– –00 0111	– –00 0111	– –uu uuuu
STATUS	– –00 xxxx	– –uu uuuu	– –1u uuuu	– –11 uuuu
INTC0	–000 0000	–000 0000	–000 0000	–uuu uuuu
TMR0	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMR0C	0000 1– – –	0000 1– – –	0000 1– – –	uuuu u– – –
TMR1H	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMR1L	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMR1C	0000 1– – –	0000 1– – –	0000 1– – –	uuuu u– – –
PA	1111 1111	1111 1111	1111 1111	uuuu uuuu
PB	1111 1111	1111 1111	1111 1111	uuuu uuuu
PC	– – – – – – –1	– – – – – – –1	– – – – – – –1	– – – – – – –u
PCC	– – – – – – –1	– – – – – – –1	– – – – – – –1	– – – – – – –u
PD	– – – – 1111	– – – – 1111	– – – – 1111	– – – – uuuu
INTC1	–000 –000	–000 –000	–000 –000	–uuu –uuu
LCDC	0000 – –11	0000 – –11	0000 – –11	0000 – –uu

″u″ stands for unchanged

″x″ stands for unknown

″–″ stands for unimplemented

Oscillator

The one methods of generating the system clock are:

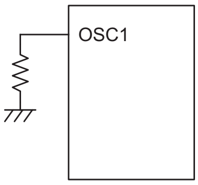
- External RC oscillator

More information regarding the oscillator is located in Application Note HA0075E on the Holtek website.

External RC Oscillator

As an RC oscillator is used, an external resistor between OSC1 and VSS is required whose value should be 12kΩ for a frequency of 4MHz. The RC oscillator provides a ±3% accuracy, the conditions are:

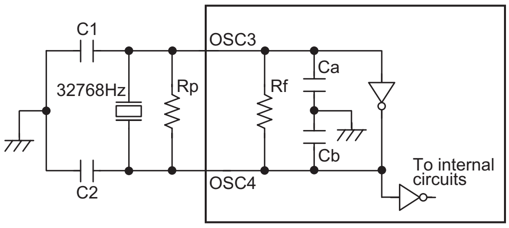
- V_{DD}=2.0V~3.6V
- Temp.= 0°C ~ 50°C
- f_{SYS}=4MHz



RC Oscillator

External RTC Oscillator

When the microcontroller enters the Power Down Mode, the system clock is switched off to stop microcontroller activity and to conserve power. However, in many microcontroller applications it may be necessary to keep the internal timers operational even when the microcontroller is in the Power Down Mode. To do this, another clock, independent of the system clock, must be provided. To do this a configuration option exists for a Real Time Clock - RTC oscillator. Here the OSC3 and OSC4 pins, should be connected to a 32768Hz crystal to implement this internal RTC oscillator. However, for some crystals, to ensure oscillation and accurate frequency generation, it may be necessary to add two small value external capacitors, C1 and C2. The exact values of C1 and C2 should be selected in consultation with the crystal or resonator manufacturer's specification. The external parallel feedback resistor, Rp, is normally not required but in some cases may be needed to assist with oscillation start up.



Note: Rp is normally not required.

External RTC Oscillator

Internal Ca, Cb, Rf Typical Values @ 5V, 25°C		
Ca	**Cb**	**Rf**
TBD	TBD	TBD

RTC Oscillator Internal Component Values

RTC Oscillator C1 and C2 Values			
Crystal Frequency	**C1**	**C2**	**CL**
32768Hz	TBD	TBD	TBD
Note: 1. C1 and C2 values are for guidance only. 2. CL is the crystal manufacturer specified load capacitor value.			

32768 Hz Crystal Recommended Capacitor Values

During power up there is a time delay associated with the RTC oscillator waiting for it to start up. A bit in the RTCC register, known as the QOSC bit, is provided to give a quick start-up function and can be used to minimise this delay. During a power up condition, this bit will be cleared to 0 which will initiate the RTC oscillator quick start-up function. However, as there is additional power consumption associated with this quick start-up function, to reduce power consumption after start up takes place, it is recommended that the application program should set the QOSC bit high about 2 seconds after power on. It should be noted that, no matter what condition the QOSC bit is set to, the RTC oscillator will always function normally, only there is more power consumption associated with the quick start-up function.

Watchdog Timer Oscillator

The WDT oscillator is a fully integrated free running RC oscillator with a typical period of 90μs at 3V, requiring no external components. It is selected via configuration option. If selected, when the device enters the Power Down Mode, the system clock will stop running, however the WDT oscillator will continue to run and keep the watchdog function active. However, as the WDT will consume a certain amount of power when in the Power Down Mode, for low power applications, it may be desirable to disable the WDT oscillator by configuration option.

Power Down Mode and Wake-up

Power Down Mode

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

Entering the Power Down Mode

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

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

Standby Current Considerations

As the main reason for entering the Power Down Mode is to keep the current consumption of the MCU to as low a value as possible, perhaps only in the order of several micro-amps, there are other considerations which must also be taken into account by the circuit designer if the power consumption is to be minimized. Special attention must be made to the I/O pins on the device. All high-impedance input pins must be connected to either a fixed high or low level as any floating input pins could create internal oscillations and result in increased current consumption. This also applies to devices which have different package types, as there may be undonbed pins, which must either be setup as outputs or if setup as inputs must have pull-high resistors connected. Care must also be taken with the loads, which are connected to I/O pins, which are setup as outputs. These should be placed in a condition in which minimum current is drawn or connected only to external circuits that do not draw current, such as other CMOS inputs. Also note that additional standby current will also be required if the configuration options have enabled the Watchdog Timer internal oscillator.

Wake-up

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

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

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

Each pin on Port B can be setup via an individual configuration option to permit a negative transition on the pin to wake-up the system. When a Port B pin wake-up occurs, the program will resume execution at the instruction following the ″HALT″ instruction.

If the system is woken up by an interrupt, then two possible situations may occur. The first is where the related interrupt is disabled or the interrupt is enabled but the stack is full, in which case the program will resume execution at the instruction following the ″HALT″ instruction. In this situation, the interrupt which woke-up the device will not be immediately serviced, but will rather be serviced later when the related interrupt is finally enabled or when a stack level becomes free. The other situation is where the related interrupt is enabled and the stack is not full, in which case the regular interrupt response takes place. If an interrupt request flag is set to ″1″ before entering the Power Down Mode, the wake-up function of the related interrupt will be disabled.

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

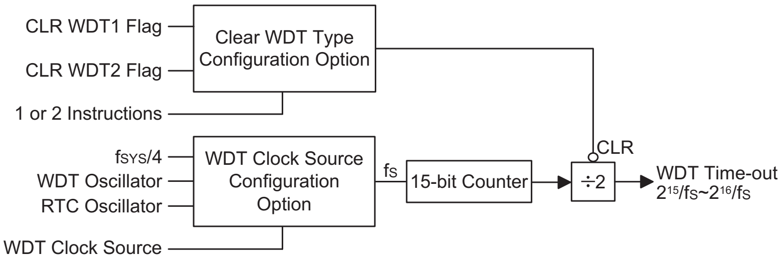
Watchdog Timer

The Watchdog Timer is provided to prevent program malfunctions or sequences from jumping to unknown locations, due to certain uncontrollable external events such as electrical noise. It operates by providing a device reset when the WDT counter overflows. The WDT clock is supplied by one of three sources selected by configuration option: its own self contained dedicated internal RTC oscillator, WDT oscillator or $f_{SYS}/4$. Note that if the WDT configuration option has been disabled, then any instruction relating to its operation will result in no operation.

In the Remote Type with LCD series of microcontrollers, all Watchdog Timer options, such as enable/disable, WDT clock source and clear instruction type all selected through configuration options. There are no internal registers associated with the WDT in the Remote Type MCU with LCD series. One of the WDT clock sources is an internal oscillator which has an approximate period of 90μs at a supply voltage of 3V. However, it should be noted that this specified internal clock period can vary with VDD, temperature and process variations. The other WDT clock source option is the $f_{SYS}/4$ clock. Whether the WDT clock source is its own internal WDT oscillator, or from $f_{SYS}/4$, it is further divided by 16 via an internal 15-bit counter and a clearable single bit counter to give longer Watchdog time-outs. As this ratio is fixed it gives an overall Watchdog Timer time-out value of $2^{15}/f_S$ to $2^{16}/f_S$. As the clear instruction only resets the last stage of the divider chain, for this reason the actual division ratio and corresponding Watchdog Timer time-out can vary by a factor of two. The exact division ratio depends upon the residual value in the Watchdog Timer counter before the clear instruction is executed. It is important to realise that as there are no independent internal registers or configuration options associated with the length of the Watchdog Timer time-out, it is completely dependent upon the frequency of $f_{SYS}/4$, the internal WDT oscillator or RTC oscillator.

If the $f_{SYS}/4$ clock is used as the WDT clock source, it should be noted that when the system enters the Power Down Mode, then the instruction clock is stopped and the WDT will lose its protecting purposes. For systems that operate in noisy environments, using the internal WDT oscillator is strongly recommended.

Under normal program operation, a WDT time-out will initialise a device reset and set the status bit TO. However, if the system is in the Power Down Mode, when a WDT 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 WDT. The first is an external hardware reset, which means a low level on the $\overline{RES}$ pin, the second is using the watchdog software instructions and the third is via a ″HALT″ instruction.



Watchdog Timer

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

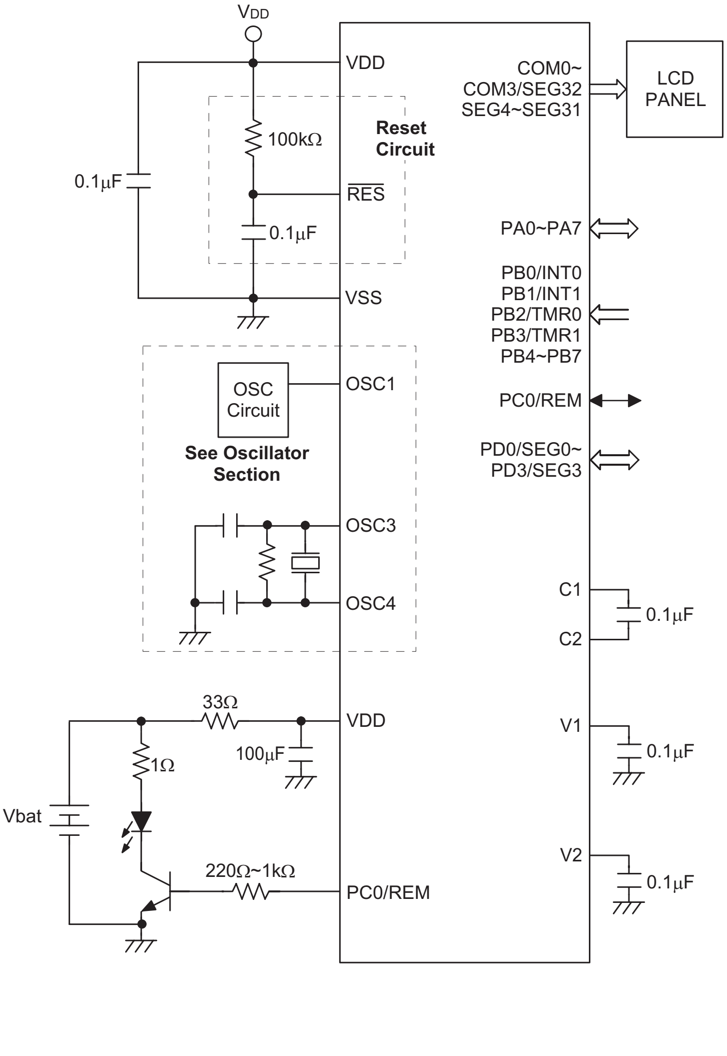
Configuration Options

Configuration options refer to certain options within the MCU that are programmed into the device during the programming process. During the development process, these options are selected using the HT-IDE software development tools. As these options are programmed into the device using the hardware programming tools, once they are selected they cannot be changed later as the application software has no control over the configuration options. All options must be defined for proper system function, the details of which are shown in the table.

Item	Options
I/O Options	
1	PB0~PB7: wake-up enable or disable (bit option)
2	PC0: CMOS output or carrier output (bit option)
3	PC0: Pull-high enable or disable (bit option)
LCD Options	
4	LCD clock: $f_S/2^2$, $f_S/2^3$, $f_S/2^4$, $f_S/2^5$, $f_S/2^6$, $f_S/2^7$, $f_S/2^8$
5	LCD duty: 1/2, 1/3, 1/4
6	LCD bias: 1/2, 1/3
7	LCD segment 12~15 output or CMOS output(Nibble Option)
8	LCD segment 16~19 output or CMOS output(Nibble Option)
Interrupt Options	
9	INT0 function: enable or disable
10	Triggering edge: rising, falling or both
11	INT1 function: enable or disable
12	Triggering edge: rising, falling or both
Oscillator Options	
13	f_S internal clock source: RTC oscillator, WDT oscillator or $f_{SYS}/4$
Timer Options	
14	Timer/Event Counter 0 clock source: f_{SYS} or $f_{SYS}/4$
Time Base Options	
15	Time Base division ratio: $f_S/2^{12}$, $f_S/2^{13}$, $f_S/2^{14}$, $f_S/2^{15}$
Watchdog Options	
16	WDT enable or disable
17	CLRWDT instructions: 1 or 2 instructions

Item	Options
LVD/LVR Options	
18	LVD function: enable or disable
19	LVR function: enable or disable
20	LVR/LVD voltage: 2.1V/2.2V or 3.15V/3.3V
Carrier Options	
21	Carrier duty: 1/2 duty or 1/3 duty
22	Carrier frequency: f_{SYS}/8, f_{SYS}/16, f_{SYS}/32, f_{SYS}/64 for 1/2 duty cycle
23	Carrier frequency: f_{SYS}/12, 1/3 duty cycle
24	Carrier frequency: f_{SYS}/24, f_{SYS}/48, f_{SYS}/96 for 1/2 duty or 1/3 duty cycle

Application Circuits



Instruction Set

Introduction

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

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

Instruction Timing

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

Moving and Transferring Data

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

Arithmetic Operations

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

Logical and Rotate Operations

The 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 where rotate data operations are used is to implement multiplication and division calculations.

Branches and Control Transfer

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

Bit Operations

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

Table Read Operations

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

Other Operations

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

Instruction Set Summary

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

Table conventions:

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

Mnemonic	Description	Cycles	Flag Affected
Arithmetic			
ADD A,[m]	Add Data Memory to ACC	1	Z, C, AC, OV
ADDM A,[m]	Add ACC to Data Memory	1^{Note}	Z, C, AC, OV
ADD A,x	Add immediate data to ACC	1	Z, C, AC, OV
ADC A,[m]	Add Data Memory to ACC with Carry	1	Z, C, AC, OV
ADCM A,[m]	Add ACC to Data memory with Carry	1^{Note}	Z, C, AC, OV
SUB A,x	Subtract immediate data from the ACC	1	Z, C, AC, OV
SUB A,[m]	Subtract Data Memory from ACC	1	Z, C, AC, OV
SUBM A,[m]	Subtract Data Memory from ACC with result in Data Memory	1^{Note}	Z, C, AC, OV
SBC A,[m]	Subtract Data Memory from ACC with Carry	1	Z, C, AC, OV
SBCM A,[m]	Subtract Data Memory from ACC with Carry, result in Data Memory	1^{Note}	Z, C, AC, OV
DAA [m]	Decimal adjust ACC for Addition with result in Data Memory	1^{Note}	C
Logic Operation			
AND A,[m]	Logical AND Data Memory to ACC	1	Z
OR A,[m]	Logical OR Data Memory to ACC	1	Z
XOR A,[m]	Logical XOR Data Memory to ACC	1	Z
ANDM A,[m]	Logical AND ACC to Data Memory	1^{Note}	Z
ORM A,[m]	Logical OR ACC to Data Memory	1^{Note}	Z
XORM A,[m]	Logical XOR ACC to Data Memory	1^{Note}	Z
AND A,x	Logical AND immediate Data to ACC	1	Z
OR A,x	Logical OR immediate Data to ACC	1	Z
XOR A,x	Logical XOR immediate Data to ACC	1	Z
CPL [m]	Complement Data Memory	1^{Note}	Z
CPLA [m]	Complement Data Memory with result in ACC	1	Z
Increment & Decrement			
INCA [m]	Increment Data Memory with result in ACC	1	Z
INC [m]	Increment Data Memory	1^{Note}	Z
DECA [m]	Decrement Data Memory with result in ACC	1	Z
DEC [m]	Decrement Data Memory	1^{Note}	Z

Mnemonic	Description	Cycles	Flag Affected
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
Data Move			
MOV A,[m]	Move Data Memory to ACC	1	None
MOV [m],A	Move ACC to Data Memory	1^{Note}	None
MOV A,x	Move immediate data to ACC	1	None
Bit Operation			
CLR [m].i	Clear bit of Data Memory	1^{Note}	None
SET [m].i	Set bit of Data Memory	1^{Note}	None
Branch			
JMP addr	Jump unconditionally	2	None
SZ [m]	Skip if Data Memory is zero	1^{Note}	None
SZA [m]	Skip if Data Memory is zero with data movement to ACC	1^{note}	None
SZ [m].i	Skip if bit i of Data Memory is zero	1^{Note}	None
SNZ [m].i	Skip if bit i of Data Memory is not zero	1^{Note}	None
SIZ [m]	Skip if increment Data Memory is zero	1^{Note}	None
SDZ [m]	Skip if decrement Data Memory is zero	1^{Note}	None
SIZA [m]	Skip if increment Data Memory is zero with result in ACC	1^{Note}	None
SDZA [m]	Skip if decrement Data Memory is zero with result in ACC	1^{Note}	None
CALL addr	Subroutine call	2	None
RET	Return from subroutine	2	None
RET A,x	Return from subroutine and load immediate data to ACC	2	None
RETI	Return from interrupt	2	None
Table Read			
TABRDC [m]	Read table (current page) to TBLH and Data Memory	2^{Note}	None
TABRDL [m]	Read table (last page) to TBLH and Data Memory	2^{Note}	None
Miscellaneous			
NOP	No operation	1	None
CLR [m]	Clear Data Memory	1^{Note}	None
SET [m]	Set Data Memory	1^{Note}	None
CLR WDT	Clear Watchdog Timer	1	TO, PDF
CLR WDT1	Pre-clear Watchdog Timer	1	TO, PDF
CLR WDT2	Pre-clear Watchdog Timer	1	TO, PDF
SWAP [m]	Swap nibbles of Data Memory	1^{Note}	None
SWAPA [m]	Swap nibbles of Data Memory with result in ACC	1	None
HALT	Enter power down mode	1	TO, PDF

Note:
1. For skip instructions, if the result of the comparison involves a skip then two cycles are required, if no skip takes place only one cycle is required.
2. Any instruction which changes the contents of the PCL will also require 2 cycles for execution.
3. For the ″CLR WDT1″ and ″CLR WDT2″ instructions the TO and PDF flags may be affected by the execution status. The TO and PDF flags are cleared after both ″CLR WDT1″ and ″CLR WDT2″ instructions are consecutively executed. Otherwise the TO and PDF flags remain unchanged.

Instruction Definition

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

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

Operation ACC ← ACC + [m] + C

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

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

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

Operation [m] ← ACC + [m] + C

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

ADD A,[m] Add Data Memory to ACC

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

Operation ACC ← ACC + [m]

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

ADD A,x Add immediate data to ACC

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

Operation ACC ← ACC + x

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

ADDM A,[m] Add ACC to Data Memory

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

Operation [m] ← ACC + [m]

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

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

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

Operation ACC ← ACC ″AND″ [m]

Affected flag(s) Z

AND A,x Logical AND immediate data to ACC

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

Operation ACC ← ACC ″AND″ x

Affected flag(s) Z

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

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

Operation [m] ← ACC ″AND″ [m]

Affected flag(s) Z

CALL addr Subroutine call

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

Operation Stack ← Program Counter + 1
Program Counter ← addr

Affected flag(s) None

CLR [m] Clear Data Memory

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

Operation [m] ← 00H

Affected flag(s) None

CLR [m].i Clear bit of Data Memory

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

Operation [m].i ← 0

Affected flag(s) None

CLR WDT Clear Watchdog Timer

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

Operation WDT cleared
TO ← 0
PDF ← 0

Affected flag(s) TO, PDF

CLR WDT1 Pre-clear Watchdog Timer

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

Operation WDT cleared
TO ← 0
PDF ← 0

Affected flag(s) TO, PDF

CLR WDT2 Pre-clear Watchdog Timer

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

Operation WDT cleared
TO ← 0
PDF ← 0

Affected flag(s) TO, PDF

CPL [m] Complement Data Memory

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

Operation [m] ← $\overline{[m]}$

Affected flag(s) Z

CPLA [m] Complement Data Memory with result in ACC

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

Operation ACC ← $\overline{[m]}$

Affected flag(s) Z

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

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

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

Affected flag(s) C

DEC [m] Decrement Data Memory

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

Operation [m] ← [m] − 1

Affected flag(s) Z

DECA [m] Decrement Data Memory with result in ACC

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

Operation ACC ← [m] − 1

Affected flag(s) Z

HALT Enter power down mode

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

Operation TO ← 0
PDF ← 1

Affected flag(s) TO, PDF

INC [m] Increment Data Memory

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

Operation [m] ← [m] + 1

Affected flag(s) Z

INCA [m] Increment Data Memory with result in ACC

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

Operation ACC ← [m] + 1

Affected flag(s) Z

JMP addr Jump unconditionally

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

Operation Program Counter ← addr

Affected flag(s) None

MOV A,[m] Move Data Memory to ACC

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

Operation ACC ← [m]

Affected flag(s) None

MOV A,x Move immediate data to ACC

Description The immediate data specified is loaded into the Accumulator.

Operation ACC ← x

Affected flag(s) None

MOV [m],A Move ACC to Data Memory

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

Operation [m] ← ACC

Affected flag(s) None

NOP No operation

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

Operation No operation

Affected flag(s) None

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

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

Operation ACC ← ACC ″OR″ [m]

Affected flag(s) Z

OR A,x Logical OR immediate data to ACC

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

Operation ACC ← ACC ″OR″ x

Affected flag(s) Z

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

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

Operation [m] ← ACC ″OR″ [m]

Affected flag(s) Z

RET Return from subroutine

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

Operation Program Counter ← Stack

Affected flag(s) None

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

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

Operation Program Counter ← Stack
ACC ← x

Affected flag(s) None

RETI Return from interrupt

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

Operation Program Counter ← Stack
EMI ← 1

Affected flag(s) None

RL [m] Rotate Data Memory left

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

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

Affected flag(s) None

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

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

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

Affected flag(s) None

RLC [m]
Rotate Data Memory left through Carry

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

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

Affected flag(s)
C

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

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

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

Affected flag(s)
C

RR [m]
Rotate Data Memory right

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

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

Affected flag(s)
None

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

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

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

Affected flag(s)
None

RRC [m]
Rotate Data Memory right through Carry

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

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

Affected flag(s)
C

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

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

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

Affected flag(s)
C

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

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

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

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

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

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

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

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

SDZ [m] Skip if decrement Data Memory is 0

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

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

Affected flag(s) None

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

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

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

Affected flag(s) None

SET [m] Set Data Memory

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

Operation [m] ← FFH

Affected flag(s) None

SET [m].i Set bit of Data Memory

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

Operation [m].i ← 1

Affected flag(s) None

SIZ [m]
Skip if increment Data Memory is 0

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

Operation
$[m] \leftarrow [m] + 1$
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
$ACC \leftarrow [m] + 1$
Skip if $ACC = 0$

Affected flag(s)
None

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

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

Operation
Skip if $[m].i \neq 0$

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
$ACC \leftarrow ACC - [m]$

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

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

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

Operation
$[m] \leftarrow ACC - [m]$

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

SUB A,x
Subtract immediate data from ACC

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

Operation
$ACC \leftarrow ACC - x$

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

SWAP [m]	Swap nibbles of Data Memory
Description	The low-order and high-order nibbles of the specified Data Memory are interchanged.
Operation	[m].3~[m].0 ↔ [m].7 ~ [m].4
Affected flag(s)	None
SWAPA [m]	Swap nibbles of Data Memory with result in ACC
Description	The low-order and high-order nibbles of the specified Data Memory are interchanged. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.
Operation	ACC.3 ~ ACC.0 ← [m].7 ~ [m].4 ACC.7 ~ ACC.4 ← [m].3 ~ [m].0
Affected flag(s)	None
SZ [m]	Skip if Data Memory is 0
Description	If the contents of the specified Data Memory is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.
Operation	Skip if [m] = 0
Affected flag(s)	None
SZA [m]	Skip if Data Memory is 0 with data movement to ACC
Description	The contents of the specified Data Memory are copied to the Accumulator. If the value is zero, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.
Operation	ACC ← [m] Skip if [m] = 0
Affected flag(s)	None
SZ [m].i	Skip if bit i of Data Memory is 0
Description	If bit i of the specified Data Memory is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0, the program proceeds with the following instruction.
Operation	Skip if [m].i = 0
Affected flag(s)	None
TABRDC [m]	Read table (current page) to TBLH and Data Memory
Description	The low byte of the program code (current page) addressed by the table pointer (TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.
Operation	[m] ← program code (low byte) TBLH ← program code (high byte)
Affected flag(s)	None
TABRDL [m]	Read table (last page) to TBLH and Data Memory
Description	The low byte of the program code (last page) addressed by the table pointer (TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.
Operation	[m] ← program code (low byte) TBLH ← program code (high byte)
Affected flag(s)	None

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

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

Operation ACC ← ACC ″XOR″ [m]

Affected flag(s) Z

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

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

Operation [m] ← ACC ″XOR″ [m]

Affected flag(s) Z

XOR A,x Logical XOR immediate data to ACC

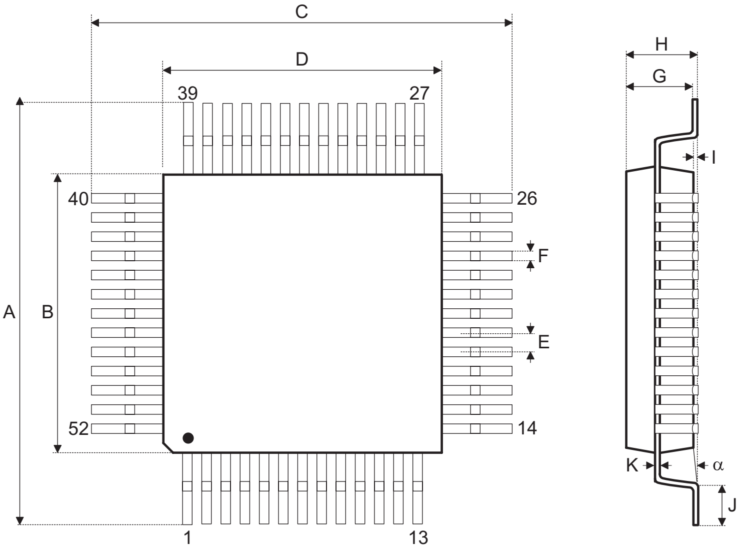
Description Data in the Accumulator and the specified immediate data perform a bitwise logical XOR operation. The result is stored in the Accumulator.

Operation ACC ← ACC ″XOR″ x

Affected flag(s) Z

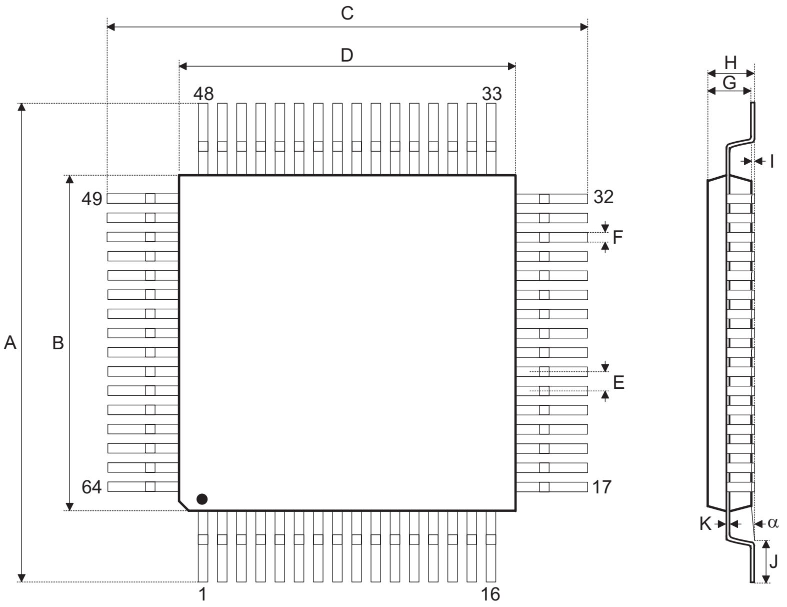
Package Information

52-pin QFP (14×14) Outline Dimensions



Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	17.3	—	17.5
B	13.9	—	14.1
C	17.3	—	17.5
D	13.9	—	14.1
E	—	1	—
F	—	0.4	—
G	2.5	—	3.1
H	—	—	3.4
I	—	0.1	—
J	0.73	—	1.03
K	0.1	—	0.2
α	0°	—	7°

64-pin LQFP (7×7) Outline Dimensions



Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	8.9	—	9.1
B	6.9	—	7.1
C	8.9	—	9.1
D	6.9	—	7.1
E	—	0.4	—
F	0.13		0.23
G	1.35	—	1.45
H	—	—	1.6
I	0.05	—	0.15
J	0.45	—	0.75
K	0.09	—	0.20
α	0°	—	7°

Holtek Semiconductor Inc. (Headquarters)
No.3, Creation Rd. II, Science Park, Hsinchu, Taiwan
Tel: 886-3-563-1999
Fax: 886-3-563-1189
http://www.holtek.com.tw

Holtek Semiconductor Inc. (Taipei Sales Office)
4F-2, No. 3-2, YuanQu St., Nankang Software Park, Taipei 115, Taiwan
Tel: 886-2-2655-7070
Fax: 886-2-2655-7373
Fax: 886-2-2655-7383 (International sales hotline)

Holtek Semiconductor Inc. (Shanghai Sales Office)
7th Floor, Building 2, No.889, Yi Shan Rd., Shanghai, China 200233
Tel: 86-21-6485-5560
Fax: 86-21-6485-0313
http://www.holtek.com.cn

Holtek Semiconductor Inc. (Shenzhen Sales Office)
5F, Unit A, Productivity Building, Gaoxin M 2nd, Middle Zone Of High-Tech Industrial Park, ShenZhen, China 518057
Tel: 86-755-8616-9908, 86-755-8616-9308
Fax: 86-755-8616-9722

Holtek Semiconductor Inc. (Beijing Sales Office)
Suite 1721, Jinyu Tower, A129 West Xuan Wu Men Street, Xicheng District, Beijing, China 100031
Tel: 86-10-6641-0030, 86-10-6641-7751, 86-10-6641-7752
Fax: 86-10-6641-0125

Holtek Semiconductor Inc. (Chengdu Sales Office)
709, Building 3, Champagne Plaza, No.97 Dongda Street, Chengdu, Sichuan, China 610016
Tel: 86-28-6653-6590
Fax: 86-28-6653-6591

Holtek Semiconductor (USA), Inc. (North America Sales Office)
46729 Fremont Blvd., Fremont, CA 94538
Tel: 1-510-252-9880
Fax: 1-510-252-9885
http://www.holtek.com

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