e-Banking 8-Bit OTP MCU

HT82R732

Revision : 1.00 Date : October 31, 2013

www.holtek.com

Table of Contents

Features

CPU Features

- Operating voltage: f_{SYS} = 4MHz: 2.2V~5.5V
- Program Memory: 4Kx16
- Data Memory: 576x8
- Up to 1μs instruction cycle with 4MHz system clock at V_{DD}= 3V
- HALT mode and wake-up functions to reduce power consumption
- Oscillator types
  - Internal RC -- HIRC
  - Low frequency internal RC -- LIRC
  - External low frequency crystal -- LXT
- LXT oscillator is implemented by TinyPower™ structure
- Two operational modes: Normal, HALT Mode
- Fully integrated internal 4MHz oscillator requires no external components
- Watchdog Timer function
- LIRC oscillator function for Watchdog Timer
- All instructions executed in one or two instruction cycles
- Table read instructions
- 63 powerful instructions
- 6-level subroutine nesting
- Bit manipulation instruction
- Low voltage reset function
- Low voltage detect function
- 28-pin SSOP, 48-pin LQFP package types

Peripheral Features

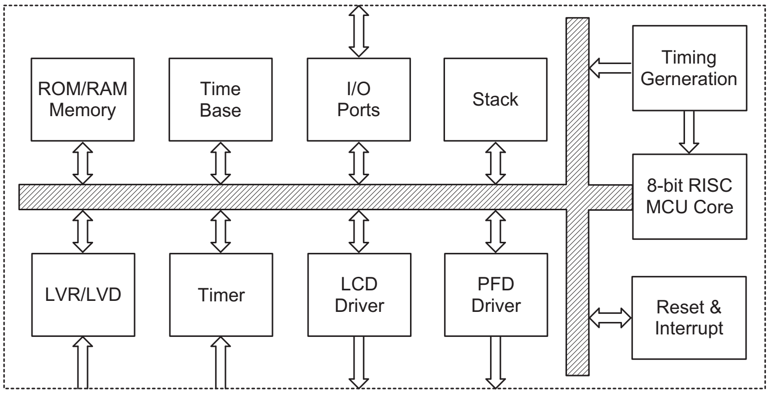
- Up to 18 bidirectional I/O lines
- Internal LCD driver with 24x4 segments
- External interrupt input shared with an I/O line
- Two 8-bit programmable Timer/Event Counter with overflow interrupt and prescaler
- Dual Time-Base functions for generation of fixed time interrupt signals
- Programmable Frequency Divider -- PFD shared with I/O line

General Description

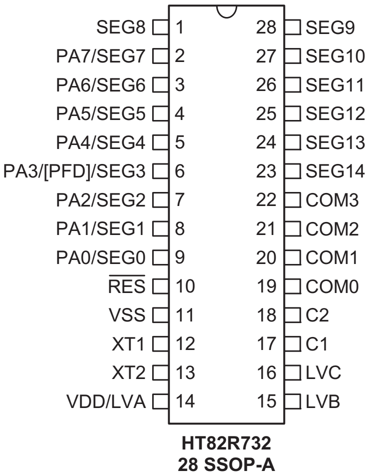
The device is an 8-bit high performance, RISC architecture microcontrollers specifically designed for the LCD applications. The usual Holtek microcontroller features of low power consumption, I/O flexibility, timer functions, oscillator options, HALT and wake-up functions, watchdog timer, low voltage reset, low voltage detect and internal LCD driver, combine to provide the device with a wide range of functional options while still maintaining a high level of cost effectiveness. The fully integrated system oscillator HIRC, which requires no external components and which has three frequency selections, opens up a huge range of new application possibilities for this device, some of which may include measuring scales, electronic meters, gas meters, timers, and many other LCD-based industrial and home appliance applications.

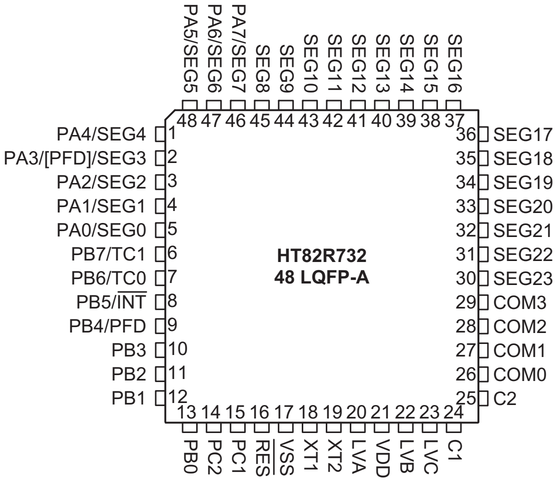
Block Diagram

The following block diagram illustrates the main functional blocks.



Pin Assignment





Pin Description

Pin Name	Function	OPT	I/T	O/T	Description
PA0/SEG0~ PA2/SEG2	PAn	PAPU PAWK	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	SEGn	SEGC	—	CMOS	LCD segment output
PA3/SEG3/PFD	PA3	PAPU PAWK	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	SEGn	SEGC	—	CMOS	LCD segment output
	PFD	—	—	CMOS	PFD output
PA4/SEG4~ PA7/SEG7	PAn	PAPU PAWK	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	SEGn	SEGC	—	CMOS	LCD segment output
PB0~PB3	PBn	PBPU PBWK	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
PB4/PFD	PB4	PBPU PBWK	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	PFD	—	—	CMOS	PFD output
PB5/$\overline{\text{INT}}$	PB5	PBPU PBWK	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	$\overline{\text{INT}}$	—	ST	—	External interrupt input
PB6/TC0	PB6	PBPU PBWK	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	TC0	—	ST	—	External Timer 0 clock input
PB7/TC1	PB7	PBPU PBWK	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	TC1	—	ST	—	External Timer 1 clock input
$\overline{\text{RES}}$	$\overline{\text{RES}}$	—	ST	—	Reset input
PC1~PC2	PCn	PCPU PCWK	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
XT1	LXT	—	LXT	—	LXT pin
XT2	LXT	—	—	LXT	LXT pin
VDD	VDD	—	PWR	—	Power supply
VSS	VSS	—	PWR	—	Ground
LVA,LVB	LVA,LVB	—	PWR	—	LCD power supply
LVC,C1,C2	LVC,C1,C2	—	—	—	LCD voltage pump
SEG8~SEG23	SEGn	—	—	CMOS	LCD segment output
COM0~COM3	COMn	—	—	CMOS	LCD common output

Legend: I/T: Input type
O/T: Output type
OPT: Options selected by register
PWR: Power
ST: Schmitt Trigger input
CMOS: CMOS output;
LXT: Low frequency crystal oscillator

Note: As the device exists in more than one package type the table reflects the situation for the package with the highest pin count. For this reason not all pins will exist on all package types.

Absolute Maximum Ratings

Supply Voltage	V_{SS}–0.3V to V_{SS}+6.0V
Input Voltage	V_{SS}–0.3V to V_{DD}+0.3V
Storage Temperature	–50°C to 125°C
Operating Temperature	–40°C to 85°C
I_{OL} Total	100mA
I_{OH} Total	–100mA
Total Power Dissipation	500mW

Note: These are stress ratings only. Stresses exceeding the range specified under "Absolute Maximum Ratings" may cause substantial damage to 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 (High Frequency Internal RC OSC)	—	f_{SYS}=4MHz	2.2	—	5.5	V
I_{DD}	Operating Current (HIRC OSC, f_S= f_P=f_{RTC} or f_{WDTOSC})	3V	No load, f_{SYS}=4MHz, WDT enable	—	0.8	1.6	mA
		5V		—	1.6	3.2	mA
I_{STB1}	Standby Current (HIRC OSC, f_{SYS}=off, f_S= WDTOSC, f_P= RTCOSC)	3V	No load, system HALT, LCD disable, WDT enable, TBC disable	—	2.0	6.0	μA
		5V		—	6.0	12.0	μA
I_{STB2}	Standby Current (HIRC OSC, f_{SYS}=off, f_S= f_P= RTCOSC)	3V	No load, system HALT, LCD enable, WDT enable, TBC enable, C type, 1/2 or 1/3 Bias	—	1.0	1.5	μA
		5V		—	2.0	4.0	μA
V_{IL1}	Input Low Voltage for I/O, TCn and $\overline{\text{INT}}$	—	—	0	—	$0.3V_{DD}$	V
V_{IH1}	Input High Voltage for I/O, TCn and $\overline{\text{INT}}$	—	—	$0.7V_{DD}$	—	V_{DD}	V
V_{IL2}	Input Low Voltage ($\overline{\text{RES}}$)	—	—	0	—	$0.4V_{DD}$	V
V_{IH2}	Input High Voltage ($\overline{\text{RES}}$)	—	—	$0.9V_{DD}$	—	V_{DD}	V
V_{LVR}	Low Voltage Reset Voltage	—	LVR enable	-5% x Typ.	2.10	+5% x Typ.	V
I_{LVR}	Low Voltage Reset Current	—	LVR enable, LVDEN=0 (Disabled)	—	60	90	μA
V_{LVD1}	Low Voltage Detector Voltage	—	LVDEN=1, V_{LVD}=2.0V	-5% x Typ.	2.0	+5% x Typ.	V
V_{LVD2}			LVDEN=1, V_{LVD}=2.2V		2.2		V
V_{LVD3}			LVDEN=1, V_{LVD}=2.4V		2.4		V
V_{LVD4}			LVDEN=1, V_{LVD}=2.7V		2.7		V
V_{LVD5}			LVDEN=1, V_{LVD}=3.0V		3.0		V
V_{LVD6}			LVDEN=1, V_{LVD}=3.3V		3.3		V
V_{LVD7}			LVDEN=1, V_{LVD}=3.6V		3.6		V
V_{LVD8}			LVDEN=1, V_{LVD}=4.4V		4.0		V
I_{LVD}	Low Voltage Detector Current	—	LVDEN=1 (enabled)	—	70	120	μA

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{OL1}	I/O Port Sink Current	3V	V_{OL}=0.1V_{DD}	4	8	—	mA
		5V		10	20	—	mA
I_{OH1}	I/O Port Source Current	3V	V_{OH}=0.9V_{DD}	–2	–4	—	mA
		5V		–5	–10	—	mA
I_{OL2}	LCD Common and Segment Sink Current	3V	V_{OL}=0.1V_{DD}	210	420	—	μA
		5V		350	700	—	μA
I_{OH2}	LCD Common and Segment Source Current	3V	V_{OH}=0.9V_{DD}	–80	–160	—	μA
		5V		–180	–360	—	μA
R_{PH}	Pull-high Resistance of I/O Ports	3V	—	40	100	200	kΩ
		5V	—	20	50	100	kΩ

A.C. Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
f_{SYS}	System Clock	—	2.2V~5.5V	—	4	—	MHz
f_{HIRC}	System Clock (HIRC)	3V/5V	Ta=25°C	–1%	4	+1%	MHz
		3V/5V	Ta=0~70°C	–5%	4	+5%	MHz
		2.2V~ 3.6V	Ta=0~70°C	–8%	4	+8%	MHz
		3.0V~ 5.5V	Ta=0~70°C	–8%	4	+8%	MHz
		2.2V~ 3.6V	Ta= –40°C~85°C	–12%	4	+12%	MHz
		3.0V~ 5.5V	Ta= –40°C~85°C	–12%	4	+12%	MHz
f_{RTC}	Real Time Clock (32768 Crystal)	—	—	—	32768	—	Hz
f_{TIMER}	Timer0/1 I/P Frequency	—	—	—	—	1	f_{SYS}
t_{WDTOSC}	Watchdog oscillator period	3V	Ta=25°C	45	90	180	μs
		5V		32	65	130	μs
t_{RES}	External Reset Low Pulse Width	—	—	1	—	—	μs
t_{SST}	System start-up timer period (wake-up from HALT)	—	SST=0	—	1024	—	t_{SYS}
			SST=1	—	16	—	t_{SYS}
t_{INT}	Interrupt Pulse Width	—	—	1	—	—	t_{SYS}
t_{LVR}	Low Voltage Width to Reset	—	—	120	240	480	μs
t_{LVDS}	LVDO Stable Time	—	For all VLVD	15	—	—	μs

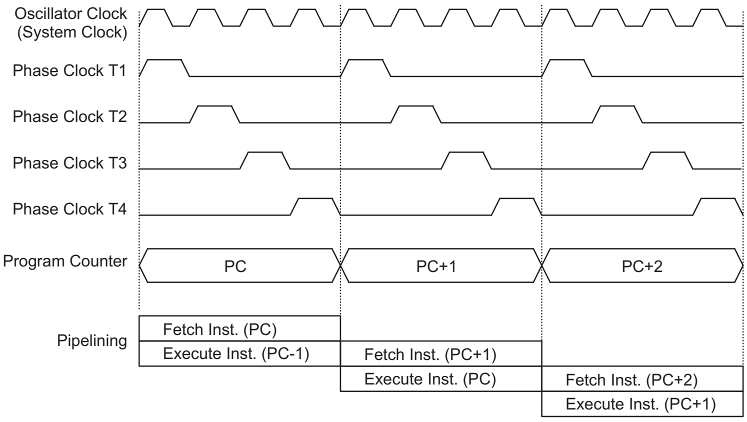
Note: t_{SYS}=1/f_{SYS}

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 control system with maximum reliability and flexibility.

Clocking and Pipelining

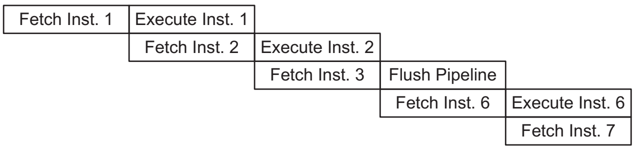
The main system clock, derived from the internal RC, HIRC, oscillator is subdivided into four internally generated non-overlapping clocks, T1~T4. The Program Counter is incremented at the beginning of the T1 clock during which time a new instruction is fetched. The remaining T2~T4 clocks carry out the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



System Clocking and Pipelining

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

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



Instruction Fetching

Program Counter

During program execution, the Program Counter is used to keep track of the address of the next instruction to be executed. It is automatically incremented by one each time an instruction is executed except for instructions, such as "JMP" or "CALL" that demand a jump to a non-consecutive Program Memory address. 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.

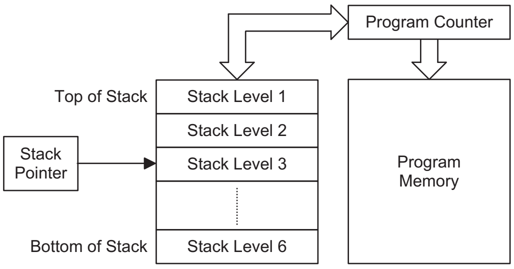
Program Counter	
Program Counter High Byte	**PCL Register**
PC11~PC8	PCL7~PCL0

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

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

Stack

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



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

Arithmetic and Logic Unit – ALU

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

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

Program Memory

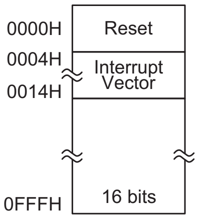
The Program Memory is the location where the user code or program is stored. The device is supplied with One-Time Programmable, OTP, memory where users can program their application code into the device. By using the appropriate programming tools, OTP devices offer users the flexibility to freely develop their applications which may be useful during debug or for products requiring frequent upgrades or program changes.

Structure

The Program Memory has a capacity of 4Kx16. 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 the reset and interrupts. The location 000H 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.



Program Memory Structure

Look-up Table

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

There are three methods to read the Program Memory data by the two table read instructions: ″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.

The three methods are shown as follows:

- Using the instruction ″TABRDC [m]″, where the table location is defined by TBLP in the current page (one page=256 words). Here the control bit of the TBHP function, the TBRC bit in the SYSC register, is disabled (default).
- Using the instruction ″TABRDC [m]″, where the table location is defined by TBLP and TBHP. Here the control bit of the TBHP function is enabled.
- The instruction ″TABRDL [m]″, where the table location is defined by TBLP in the last page (0F00H~0FFFH).

Only the destination of the lower-order byte in the table is well-defined, the other bits of the table word are transferred to the lower portion of TBLH. The Table Higher-order byte register (TBLH) is read only and cannot be restored. The table pointers (TBLP, TBHP) are read/write registers which indicate the table location. Before accessing the table, the location must be placed in the TBLP and TBHP registers (If the software option of the TBHP function is disabled, the value in TBHP has no effect). If the main routine and the ISR (Interrupt Service Routine) both employ the table read instruction, the contents of the TBLH in the main routine is likely to be changed by the table read instruction used in the ISR. As a result errors may occur. In other words, using the table read instruction in the main routine and in the ISR simultaneously should be avoided. However, if the table read instruction has to be applied in both the main routine and the ISR, the interrupt should be disabled prior to the table read instruction. It will not be enabled until the TBLH has been backed up. All table related instructions require two cycles to complete the operation. These areas may function as normal program memory depending on the requirements.

Once the control bit of the TBHP function is enabled, the instruction "TABRDC [m]" reads the Program Memory data as defined by TBLP and TBHP register value. Otherwise, if the control bit of TBHP function is disabled, the instruction "TABRDC [m]" reads the Program Memory data as defined by the TBLP and the current program counter bits.

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

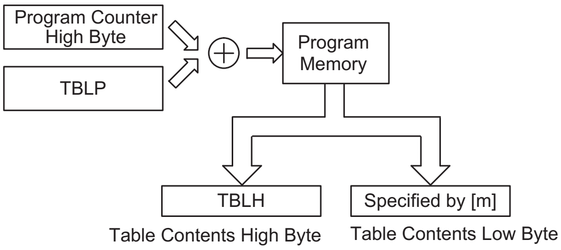


Table Read - TBLP only

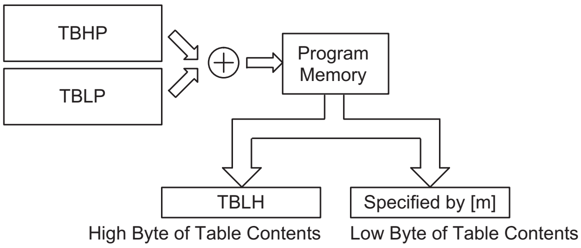


Table Read - TBLP / TBHP

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 when the TBHP control bit is disabled
TBHP register bit3 ~bit0 when TBHP is enabled
@7~@0: Table Pointer TBLP bits

Table Program Example

The accompanying example shows how the table pointer and table data is defined and retrieved from the device. 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 "0F00H" which refers to the start address of the last page within the 4K Program Memory of the device. The table pointer is setup here to have an initial value of "06H". This will ensure that the first data read from the data table will be at the Program Memory address "0F06H" or 6 locations after the start of the last page. Note that the value for the table pointer is referenced to the first address of the present page if the "TABRDC [m]" instruction is being used. The high byte of the table data which in this case is equal to zero will be transferred to the TBLH register automatically when the "TABRDL [m]" instruction is executed.

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

Table Read Program Example

```
tempreg1  db ?     ; temporary register #1
tempreg2  db ?     ; temporary register #2
  :
  :

mov   a,06h        ; initialise table pointer - note that this address is referenced

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 "0F06H" transferred to
                   ; tempreg1 and TBLH

dec   tblp         ; reduce value of table pointer by one

Tabrdl  tempreg2   ; transfers value in table referenced by table pointer
                   ; to tempreg2
                   ; data at prog.memory address "0F05H" 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   300h         ; sets initial address of last page

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 several sections, the first of these is an area of RAM where special function registers are located. These registers have fixed location and are necessary for correct operation of the device.

Many of these registers can be read 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.

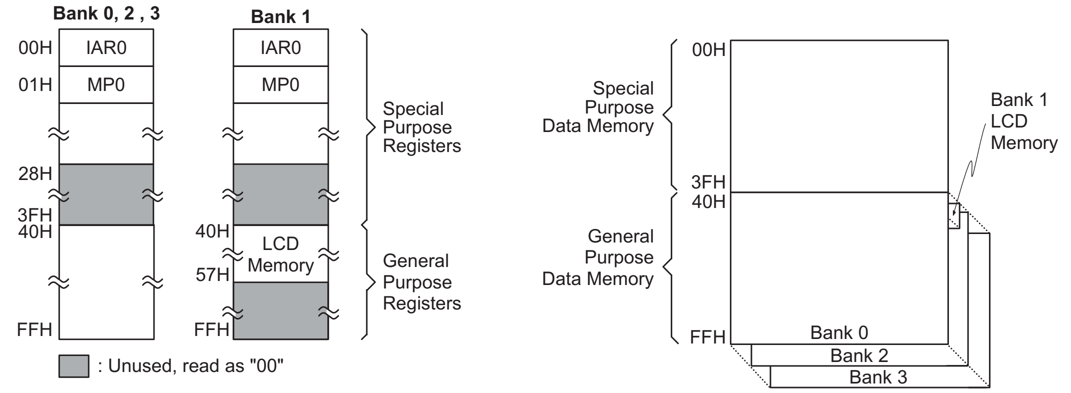
The addresses of the LCD Memory area overlap those in the General Purpose Data Memory area. Switching between the different Data Memory banks is achieved by setting the Bank Pointer to the correct value.

Structure

The Data Memory is subdivided into several banks, all of which are implemented in 8-bit wide RAM. The Data Memory located in Bank 0 is subdivided into two sections, the Special Purpose Data Memory and the General Purpose Data Memory.

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 Memory is mapped into Bank 1. The Banks 2 to 3 contain only General Purpose Data Memory for those devices with large Data Memory capacities. As the Special Purpose Data Memory registers are mapped into all bank areas, they can subsequently be accessed from any bank location.

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. 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.



Note: The 40H~57H of Bank 1 are used for LCD memory.

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.

Special Purpose Data Memory

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

Special Function Registers

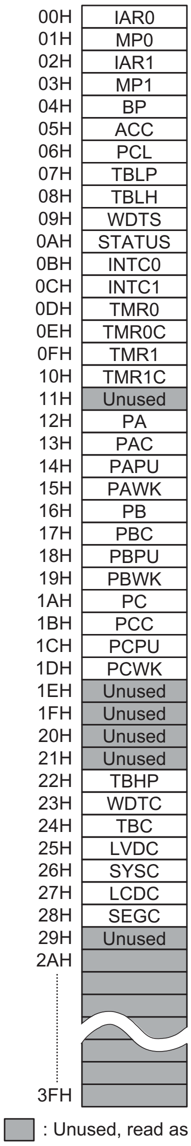
To ensure successful operation of the microcontroller, certain internal registers are implemented in the Data Memory area. These registers ensure correct operation of internal functions such as timers, interrupts, etc., as well as external functions such as I/O data control. The location of these registers within the Data Memory begins at the address ″00H″ and are mapped into both Bank 0 and Bank 1. Any unused Data Memory locations between these special function registers and the point where the General Purpose Memory begins is reserved and attempting to read data from these locations will return a value of ″00H″.

Indirect Addressing Registers – IAR0, IAR1

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

Memory Pointers – MP0, MP1

Two Memory Pointers, known as MP0 and MP1 are provided. These Memory Pointers are physically implemented in the Data Memory and can be manipulated in the same way as normal registers providing a convenient way with which to indirectly 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. The following example shows how to clear a section of four Data Memory locations already defined as locations adres1 to adres4.



00H	IAR0
01H	MP0
02H	IAR1
03H	MP1
04H	BP
05H	ACC
06H	PCL
07H	TBLP
08H	TBLH
09H	WDTS
0AH	STATUS
0BH	INTC0
0CH	INTC1
0DH	TMR0
0EH	TMR0C
0FH	TMR1
10H	TMR1C
11H	Unused
12H	PA
13H	PAC
14H	PAPU
15H	PAWK
16H	PB
17H	PBC
18H	PBPU
19H	PBWK
1AH	PC
1BH	PCC
1CH	PCPU
1DH	PCWK
1EH	Unused
1FH	Unused
20H	Unused
21H	Unused
22H	TBHP
23H	WDTC
24H	TBC
25H	LVDC
26H	SYSC
27H	LCDC
28H	SEGC
29H	Unused
2AH	▨
3FH	▨

▨ : Unused, read as "00"

Special Purpose Data Memory

Indirect Addressing Program Example

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

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

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

      continue:
```

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

Bank Pointer – BP

In the HT82R732 device, the Data Memory is divided into several Banks, known as Bank 0~3. The bit 0~1 of the Bank Pointer register are used to select the required Data Memory bank. Only data in Bank 0 can be directly addressed, as data in Bank 1~3 must be indirectly addressed using Memory Pointer MP1 and Indirect Addressing Register IAR1. Using Memory Pointer MP0 and Indirect Addressing Register IAR0 will always access data from Bank 0, irrespective of the value of the Bank Pointer. Memory Pointer MP1 and Indirect Addressing Register IAR1 can indirectly address data in Bank 0 ~ Bank 3 depending upon the value of the Bank Pointer. The Data Memory is initialised to Bank 0 after a reset, except for the WDT time-out reset in the HALT Mode, in which case, the Data Memory bank remains unaffected.

It should be noted that Special Function Data Memory is not affected by the bank selection, which means that the Special Function Registers can be accessed from within Bank 0 ~ Bank 3. Directly addressing the Data Memory will always result in Bank 0 being accessed irrespective of the value of the Bank Pointer.

- BP Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	DMBP1	DMBP0
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2 unimplemented, read as "0"

Bit 1~0 **DMBP1~DMBP0**: Select Data Memory Banks
00: Bank 0
01: Bank 1 LCD Memory
10: Bank 2
11: Bank 3

Accumulator – ACC

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

Program Counter Low Register – PCL

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

Look-up Table Registers – TBLP, TBHP, TBLH

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

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.

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

- STATUS Register

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

″x″ unknown

Bit 7, 6 Unimplemented, read as ″0″

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

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

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

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

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

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

System Control Registers – SYSC

This register is used to provide control over various internal functions. Some of these include the Wake-up time selection, PFD control and the TBHP register enable control.

- SYSC Register

Bit	7	6	5	4	3	2	1	0
Name	SSTC	TBRC	PFDC1	PFDCS	—	—	—	PFDC0
R/W	R/W	R/W	R/W	R/W	—	—	—	R/W
POR	0	0	0	0	—	—	—	0

Bit 7 **SSTC**: wake up time selection
0: 1024 system clocks
1: 16 system clocks
The SSTC bit is used to select the wake-up time for the system to wake up from the HALT mode.

Bit 6 **TBRC**: TBHP register enable control
0: disable
1: enable

Bit 5, 0 **PFDC1, PFDC0** : PFD or I/O selection
00: I/O
01: I/O
10: PFD output to PB4
11: PFD output to PA3

Bit 4 **PFDCS**: PFD clock source
0: Timer 0
1: Timer 1

Bit 3~1 Unimplemented, read as ″0″

Oscillator

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

System Oscillator Overview

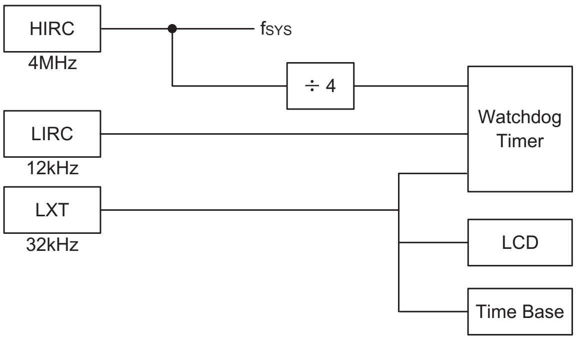
In addition to being the source of the main system clock the oscillators also provide clock sources for the Watchdog Timer and Time Base functions. An external oscillator requiring some external components as well as two fully integrated internal oscillators, requiring no external components, are provided to form a range of both fast and slow system oscillators.

Type	Name	Freq.	Pins
Internal High Speed RC	HIRC	4MHz	—
External Low Speed Crystal	LXT	32768Hz	XT1/XT2
Internal Low Speed RC	LIRC	12kHz	—

The oscillator is connected to the XT1/XT2 pins with TinyPower^{TM} design.

System Clock Configurations

There are three system oscillators, one high speed oscillator and two low speed oscillators. The high speed oscillator is the internal RC oscillator -- HIRC. The two low speed oscillators are the external 32768Hz oscillator -- LXT and the internal 12kHz (V_{DD}=5V) oscillator -- LIRC.

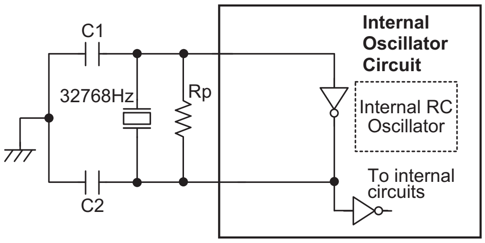


Internal RC Oscillator – HIRC

The internal RC oscillator is a fully integrated system oscillator requiring no external components. The internal RC oscillator has fixed frequency of 4MHz. Device trimming during the manufacturing process and the inclusion of internal frequency compensation circuits are used to ensure that the influence of the power supply voltage, temperature and process variations on the oscillation frequency are minimised. As a result, at a power supply of either 3V or 5V and at a temperature of 25 degrees, the fixed oscillation frequency of 4MHz will have a tolerance within 2%.

External 32768Hz Crystal Oscillator – LXT

When the microcontroller enters the HALT 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 HALT Mode. To do this, another clock, independent of the system clock, must be provided, known as the LXT oscillator. The LXT oscillator is implemented using a 32768Hz crystal connected to pins XT1/XT2. However, for some crystals, to ensure oscillation and accurate frequency generation, it is necessary to add two small value external capacitors, C1 and C2. The exact values of C1 and C2 should be selected in consultation with the crystal or resonator manufacturer's specification. The external parallel feedback resistor, Rp, is required. The low speed oscillator, LXT, is used to provide a clock source to the Watchdog Timer, the LCD driver and the Time Base Interrupts.



Note: 1. Rp, C1 and C2 are required.
2. Although not shown pins have a parasitic capacitance of around 7pF.

External LXT Oscillator – LXT

LXT Oscillator C1 and C2 Values		
Crystal Frequency	C1	C2
32768Hz	8pF	10pF
Note: 1. C1 and C2 values are for guidance only. 2. R_P=5M~10MΩ is recommended.		

32768 Hz Crystal Recommended Capacitor Values

Crystal Specifications

Symbol	Parameter	Min.	Typ.	Max.	Unit
f_O	Nominal Frequency	—	32.768	—	kHz
ESR	Series Resistance	—	50	65	kΩ
C_L	Load Capacitance	—	9	—	pF

Note: 1. It is strongly recommended to use a crystal with load capacitance 9pF.
2. The oscillator selection can be optimized using a high quality resonator with small ESR value. Refer to crystal manufacturer for more details: www.microcrystal.com

Internal Low Speed Oscillator – LIRC

The LIRC is a fully self-contained free running internal RC oscillator with a typical frequency of 12kHz at 5V requiring no external components. Its sole purpose is as one of the clock sources for the Watchdog Timer. It is automatically enabled if the Watchdog Timer selects it as its clock source using the WCS0 and WCS1 bits in the WDTC register. When the device enters the HALT Mode, the LIRC oscillator continues to run if the Watchdog Timer has selected it as its clock source.

Operating Modes

The system can be selected to operate in two modes, namely the Normal mode and HALT mode. In the Normal mode the HIRC oscillator is the system clock source. In the HALT mode the HIRC oscillator stops but the LXT and LIRC oscillators continue to run.

Mode Types and Selection

When the HALT instruction is executed, the device will enter the HALT Mode, the high frequency HIRC oscillator will stop running and the LXT and LIRC oscillator continue to run to provide the clock sources for peripheral function operation.

The accompanying table shows the relationship between the HALT instruction and the high/low frequency oscillators.

Operating Mode	HIRC	LXT	LIRC
Normal	Run	Run	Run
HALT	Stop	Run	Run

Note: The LIRC oscillator is only enabled if the Watchdog Timer selects it as its clock source.

Operating Mode Control

Mode Switching

The device is switched from the normal mode to the HALT mode using the HALT instruction. The HALT instruction forces the system into the HALT Mode. In the HALT mode, the LXT and LIRC oscillators keeps running. When a HALT instruction is executed, the system enters the HALT mode and the following conditions 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 LIRC or LXT 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.
- The Time Base counter will continue counting if the control bit, the TBON bit in the TBC register is set to ″1″.
- The LCD driver will remain active if the control bit, the LCDEN bit in the LCDC register, is set to ″1″.

Standby Current Considerations

As the main reason for entering the HALT 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 minimised.

Special attention must be made to the I/O pins on the device. All high-impedance input pins must be connected to either a fixed high or low level as any floating input pins could create internal oscillations and result in increased current consumption. 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.

If the software options have enabled the Watchdog Timer internal oscillator LIRC, then this will continue to run when in the HALT Mode and will thus consume some power. For power sensitive applications it may be therefore preferable to use the system clock source for the Watchdog Timer. The LXT, if configured for use, will also consume a limited amount of power, as it continues to run when the device enters the HALT Mode.

Wake-up

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

- An external reset
- An external falling edge on PA, PB or PC I/O pins
- 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.

Pins PA[0 :7], PB [0:7], PC[1:2] can be setup via the PAWK, PBWK and PCWK registers to permit a negative transition on the pin to wake-up the system. When an I/O 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 HALT Mode, then any interrupt requests will not generate a wake-up function of the related interrupt will be ignored. No matter what the source of the wake-up event is, once a wake-up event occurs, there will be a time delay before normal program execution resumes. Consult the table for the related time.

Wake-up Source	Oscillator Type
	IRC
External $\overline{RES}$	t_{RSDT} + t_{SST1}
PA, PB, PC Port	t_{SST1}
Interrupt	
WDT Overflow	

Note: 1. t_{RSTD} (reset delay time), t_{SYS} (system clock)
2. t_{RSTD} is power-on delay, typical time=100ms
3. t_{SST1}= 16 or 1024 t_{SYS}, selected by the SSTC bit in the SYSC register.

Wake-up Delay Time

Watchdog Timer

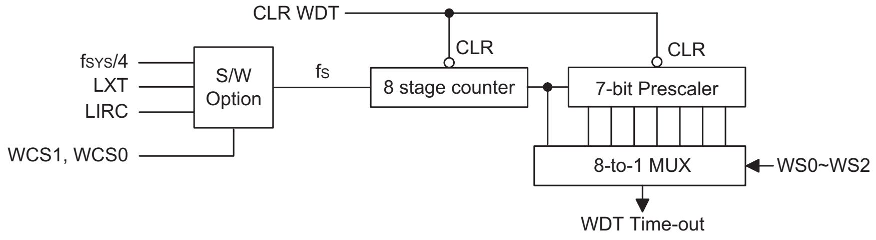
The Watchdog Timer, also known as the WDT, is provided to inhibit program malfunctions caused by the program jumping to unknown locations due to certain uncontrollable external events such as electrical noise.

Watchdog Timer Operation

The Watchdog Timer operates by providing a device reset when the Watchdog Timer counter overflows.

The Watchdog Timer clock can emanate from three different sources, selected by the WCS1 and WCS0 bits in the WDTC register. These are LXT, $f_{SYS}/4$ or LIRC. It is important to note that when the system enters the HALT Mode the instruction clock is stopped, therefore if the WDTC register bits have selected $f_{SYS}/4$ as the Watchdog Timer clock source, the Watchdog Timer will cease to function. For systems that operate in noisy environments, using the LIRC or the LXT as the clock source is therefore the recommended choice. The division ratio of the prescaler is determined by bits 0, 1 and 2 of the WDTS register, known as WS0, WS1 and WS2. If the LIRC clock source is selected and with the WS0, WS1 and WS2 bits of the WDTS register all set high, the prescaler division ratio will be 1:128, which will give a maximum time-out period.

Under normal program operation, a Watchdog Timer time-out will initialise a device reset and set the status bit TO. However, if the system is in the HALT Mode, when a Watchdog Timer time-out occurs, the device will be woken up, the TO bit in the status register will be set and only the Program Counter and Stack Pointer will be reset. Three methods can be adopted to clear the contents of the Watchdog Timer. The first is an external hardware reset, which means a low level on the external reset pin. The second is using the Clear Watchdog Timer software instructions and the third is when a HALT instruction is executed. For a simple execution of "CLR WDT" instruction will clear the Watchdog Timer.



Watchdog Timer

WDTS Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	WS2	WS1	WS0
R/W	—	—	—	—	—	R/W	R/W	R/W
POR	—	—	—	—	—	0	0	0

Bit 7~3 : unimplemented, read as ″0″

Bit 2~0 **WS2, WS1, WS0**: WDT prescaler rate selection
000= 1:1
001= 1:2
010= 1:4
011= 1:8
100= 1:16
101= 1:32
110= 1:64
111= 1:128

WDTC Register

Bit	7	6	5	4	3	2	1	0
Name	WCS1	WCS0	—	—	—	—	—	—
R/W	R/W	R/W	—	—	—	—	—	—
POR	0	0	—	—	—	—	—	—

Bit 7~6 **WCS1**, **WCS0**: WDT clock source selection
00: LIRC (default)
01: $f_{SYS}/4$
10: LXT
11: unimplemented

Bit 5~0 unimplemented, read as ″0″

Low Voltage Detector – LVD

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

LVD Register

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

LVDC Register

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

Bit7 **LVDEN**: LVD function control
0: disable
1: enable

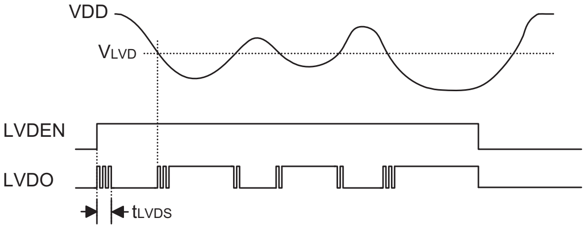
bit6 **LVDO**: LVD output flag
0: no low voltage detect
1: low voltage detect

Bit5~3 **VLVD2~VLVD0**: select LVD voltage
000: 2.0V
001: 2.2V
010: 2.4V
011: 2.7V
100: 3.0V
101: 3.3V
110: 3.6V
111: 4.4V

Bit 2~0 : unimplemented, read as ″0″

LVD Operation

The Low Voltage Detector function operates by comparing the power supply voltage, V_{DD}, with a pre-specified voltage level stored in the LVDC register. This has a range of between 2.0V and 4.4V. When the power supply voltage, V_{DD}, falls below this pre-determined value, the LVDO bit will be set high indicating a low power supply voltage condition. The Low Voltage Detector function is supplied by a reference voltage which will be automatically enabled. When the device is powered down the low voltage detector will remain active if the LVDEN bit is high. After enabling the Low Voltage Detector, a time delay t_{LVDS} should be allowed for the circuitry to stabilise before reading the LVDO bit. Note also that as the V_{DD} voltage may rise and fall rather slowly, at the voltage nears that of V_{LVD}, there may be multiple bit LVDO transitions.



LVD Operation

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.

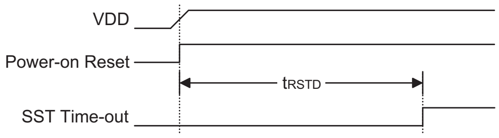
Reset Functions

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.

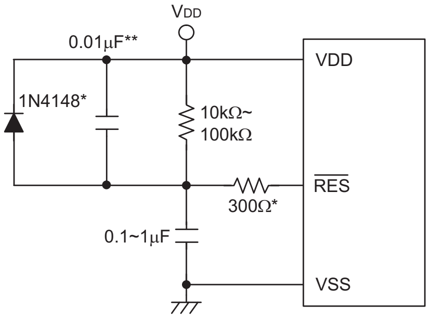


Note: t_{RSTD} is power-on delay, typical time=100ms

Power-On Reset Timing Chart

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

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



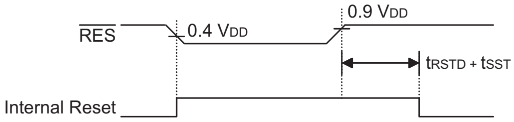
Note: "*" It is recommended that this component is added for added ESD protection
"**" It is recommended that this component is added in environments where power line noise is significant

External $\overline{RES}$ Circuit

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

$\overline{RES}$ Pin Reset

This type of reset occurs when the microcontroller is already running and the $\overline{RES}$ 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.

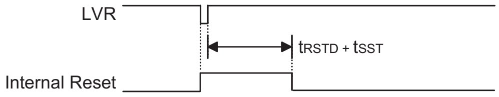


Note: t_{RSTD} is power-on delay, typical time=100ms

$\overline{RES}$ 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. If the supply voltage of the device drops to within a range of 0.9V~V_{LVR} such as might occur when changing the battery, the LVR will automatically reset the device internally. For a valid LVR signal, a low supply voltage, i.e., a voltage in the range between 0.9V~V_{LVR} must exist for a time greater than that specified by t_{LVR} in the A.C. characteristics. If the low supply voltage state does not exceed this value, the LVR will ignore the low supply voltage and will not perform a reset function.

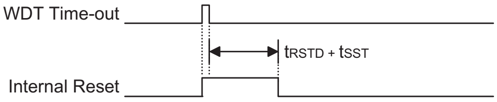


Note: t_{RSTD} is power-on delay, typical time=100ms

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".

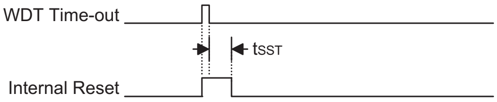


Note: t_{RSTD} is power-on delay, typical time=100ms

WDT Time-out Reset during Normal Operation Timing Chart

Watchdog Time-out Reset during HALT Mode

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



WDT Time-out Reset during HALT 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 HALT function or Watchdog Timer. The reset flags are shown in the table:

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

Note: "u" stands for unchanged

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

Item	Condition After RESET
Program Counter	Reset to zero
Interrupts	All interrupts will be disabled
WDT	Clear after reset, WDT begins counting
Timer/Event Counter	Timer Counter will be turned off
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 executio n 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.

Register	Power-on Reset	$\overline{\text{RES}}$ or LVR Reset	WDT Time-out (Normal Operation)	WDT Time-out (Idle/Sleep)
PCL	0000 0000	0000 0000	0000 0000	0000 0000
MP0	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
MP1	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
BP	–––– ––00	–––– ––00	–––– ––00	–––– ––uu
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLP	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLH	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
WDTS	–––– –000	–––– –000	–––– –000	–––– –uuu
STATUS	––00 xxxx	––uu uuuu	––1u uuuu	––11 uuuu
INTC0	–000 0000	–000 0000	–000 0000	–uuu uuuu
INTC1	––00 ––00	––00 ––00	––00 ––00	––uu ––uu
TMR0	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMR0C	0000 1000	0000 1000	0000 1000	uuuu uuuu
TMR1	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
PAC	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAWK	0000 0000	0000 0000	0000 0000	uuuu uuuu
PAPU	0000 0000	–000 0000	–000 0000	–uuu uuuu
PB	1111 1111	1111 1111	1111 1111	uuuu uuuu
PBC	1111 1111	1111 1111	1111 1111	uuuu uuuu
PBPU	0000 0000	0000 0000	0000 0000	uuuu uuuu
PBWK	0000 0000	0000 0000	0000 0000	uuuu uuuu
PC	–––– –11–	–––– –11–	–––– –11–	–––– –uu–
PCC	–––– –11–	–––– –11–	–––– –11–	–––– –uu–
PCPU	–––– –00–	–––– –00–	–––– –00–	–––– –uu–
PCWK	–––– –00–	–––– –00–	–––– –00–	–––– –uu–
TBHP	–––– xxxx	–––– uuuu	–––– uuuu	–––– uuuu
WDTC	00–– ––––	00–– ––––	00–– ––––	uu–– ––––
TBC	0–11 –111	0–11 –111	0–11 –111	u–uu uuuu
LVDC	0000 0–––	0000 0–––	0000 0–––	uuuu u–––
SYSC	0000 –––0	0000 –––0	0000 –––0	uuuu –––u
LCDC	0––– 0––0	0––– 0––0	0––– 0––0	u––– u––u
SEGC	1111 1111	1111 1111	1111 1111	uuuu uuuu

Note: ″-″ not implemented
″u″ means ″unchanged″
″x″ means ″unknown″

Input/Output Ports

Holtek microcontrollers offer considerable flexibility on their I/O ports. Most pins can have either an input or output designation under user program control. Additionally, as there are pull-high resistors and wake-up software configurations, the user is provided with an I/O structure to meet the needs of a wide range of application possibilities.

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

Pull-high Resistors

Many product applications require pull-high resistors for their switch inputs usually requiring the use of an external resistor. To eliminate the need for these external resistors, when configured as an input have the capability of being connected to an internal pull-high resistor. These pull-high resistors are selectable via a register known as PAPU, PBPU and PCPU located in the Data Memory. The pull-high resistors are implemented using weak PMOS transistors.

Port Wake-up

If the HALT instruction is executed, the device will enter the HALT Mode, where the system clock will stop resulting in power being conserved, 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 PA0~PA7 (or PB0~PB7 or PC1~PC2) pins from high to low. After a HALT instruction forces the microcontroller into entering the HALT Mode, the processor will remain idle or in a low-power state until the logic condition of the selected wake-up pin on Port A (or Port B or Port C) changes from high to low. This function is especially suitable for applications that can be woken up via external switches. Note that pins of the PA, PB and PC can be selected individually to have this wake-up feature using internal registers known as PAWK, PBWK and PCWK, located in the Data Memory.

PAWK, PAC, PAPU, PBWK, PBC, PBPU, PCWK, PCC, PCPU Registers

Register Name	POR	Bit							
		7	6	5	4	3	2	1	0
PAWK	00H	PAWK7	PAWK6	PAWK5	PAWK4	PAWK3	PAWK2	PAWK1	PAWK0
PAC	FFH	PAC7	PAC6	PAC5	PAC4	PAC3	PAC2	PAC1	PAC0
PAPU	00H	PAPU7	PAPU6	PAPU5	PAPU4	PAPU3	PAPU2	PAPU1	PAPU0
PBWK	00H	PBWK7	PBWK6	PBWK5	PBWK4	PBWK3	PBWK2	PBWK1	PBWK0
PBC	3FH	PBC7	PBC6	PBC5	PBC4	PBC3	PBC2	PBC1	PBC0
PBPU	00H	PBPU7	PBPU6	PBPU5	PBPU4	PBPU3	PBPU2	PBPU1	PBPU0
PCWK	00H	—	—	—	—	—	PCWK2	PCWK1	—
PCC	FFH	—	—	—	—	—	PCC2	PCC1	—
PCPU	00H	—	—	—	—	—	PCPU2	PCPU1	—

″—″ Unimplemented, read as ″0″

PAWKn/PBWKn/PCWKn: PA, PB, PC wake-up function enable
0: disable
1: enable

PACn/PBCn/PCCn: I/O type selection
0: output
1: input

PAPUn/PBPUn/PCPUn: Pull-high function enable
0: disable
1: enable

I/O Port Control Registers

Each Port has its own control register, known as PAC, PBC and PCC which control the input/output configuration. With this control register, each I/O pin with or without pull-high resistors can be reconfigured dynamically under software control. 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. The chosen function of the multi-function I/O pins is set by application program control. Note that if the I/O pins are pin shared with multiple functions and which are enabled at the same time, then the pin name located at the right side of the ″/″ symbol has the higher priority in the pin-shared functions.

External Interrupt Input

The external interrupt pin, $\overline{\text{INT}}$, is pin-shared with an I/O pin. To use the pin as an external interrupt input the correct bits in the INTC0 register must be programmed. The pin must also be setup as an input by setting the PBC5 bit in the Port Control Register. A pull-high resistor can also be selected via the appropriate port pull-high resistor register. Note that even if the pin is setup as an external interrupt input the I/O function still remains.

External Timer/Event Counter Input

The Timer/Event Counter pins, TC0 and TC1 are pin-shared with I/O pins. For these shared pins to be used as Timer/Event Counter inputs, the Timer/Event Counter must be configured to be in the Event Counter or Pulse Width Capture Mode. This is achieved by setting the appropriate bits in the Timer/Event Counter Control Register. The pins must also be setup as inputs by setting the appropriate bit in the Port Control Register. Pull-high resistor options can also be selected using the port pull-high resistor registers. Note that even if the pin is setup as an external timer input the I/O function still remains.

PFD Output

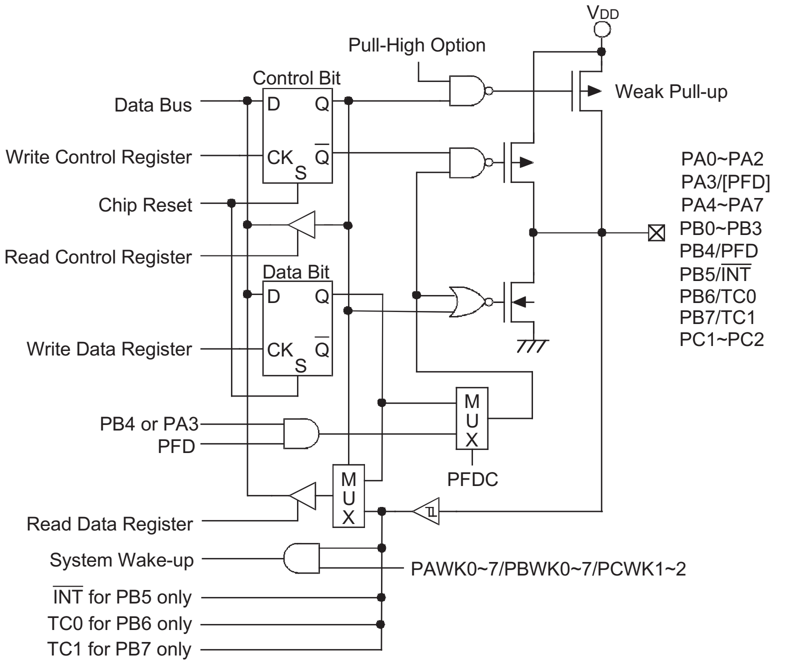
The device contains a PFD function which is pin-shared an I/O pin. The output function of this pin is chosen using the SYSC register. Note that the corresponding bit of the port control register, must setup the pin as an output to enable the PFD output. If the port control register has setup this pin as input, then this pin will function as normal logic input with the usual pull-high selection, even if the PFD function has been selected.

SEG Outputs

Some device SEG pins are shared with I/O pins. The SEG function of these pins is setup using the SEGC register.

I/O Pin Structures

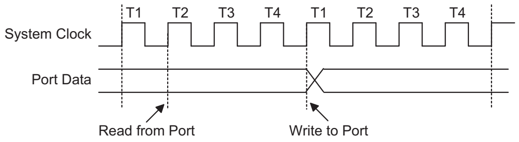
The diagrams illustrate the I/O pin internal structures. As the exact logical construction of the I/O pin may differ from these drawings, they are supplied as a guide only to assist with the functional understanding of the I/O pins.



Generic Input/Output Structure

Programming Considerations

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



Read Modify Write Timing

Pins PA0 to PA7, PB0 to PB7, PC1 to PC2 each has a wake-up function, selected via the PAWK, PBWK and PCWK registers respectively. When the device is in the HALT Mode, various methods are available to wake the device up. One of these is a high to low transition of any of these pins. Single or multiple pins on Port A, Port B or Port C can be setup to have this function.

Timer/Event Counters

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

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

Configuring the Timer/Event Counter Input Clock Source

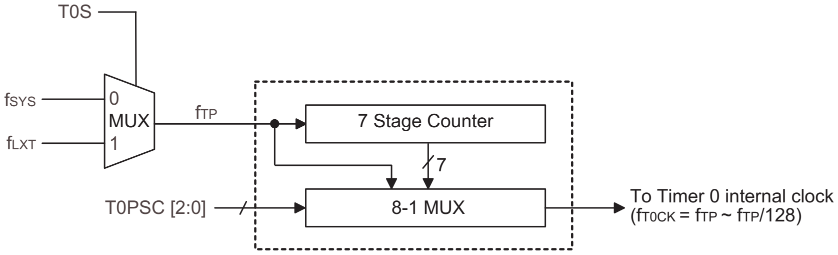
The Timer/Event Counter clock source can originate from various sources, an internal clock or an external pin. The internal clock source is used when the timer is in the timer mode or in the pulse width capture mode. For Timer/Event Counter 0, this internal clock source is first divided by a prescaler, the division ratio of which is conditioned by the Timer Control Register bits T0PSC0~T0PSC2. For Timer/Event Counter 0, the internal clock source can be either f_{SYS} or the LXT Oscillator, the choice of which is determined by the T0S bit in the TMR0C register.

An external clock source is used when the Timer/Event Counter n is in the event counting mode, the clock source being provided on an external timer pin TCn. Depending upon the condition of the TnE bit, each high to low, or low to high transition on the external timer pin will increment the counter by one.

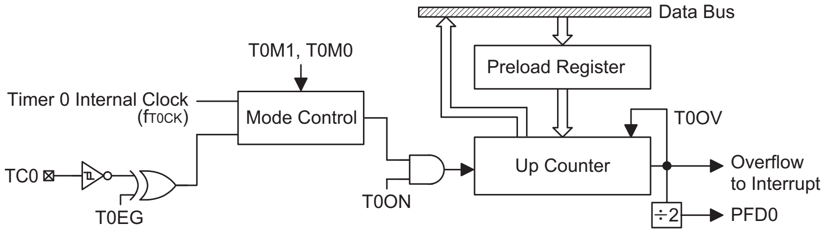
Timer Registers – TMR0, TMR1

The timer registers are special function registers located in the Special Purpose Data Memory and is the place where the actual timer value is stored. These registers are known as TMR0 and TMR1. The value in the timer registers increases by one each time an internal clock pulse is received or an external transition occurs on the external timer pin. The timer will count from the initial value loaded by the preload register to the full count of FFH at which point the timer overflows and an internal interrupt signal is generated. 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, 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.

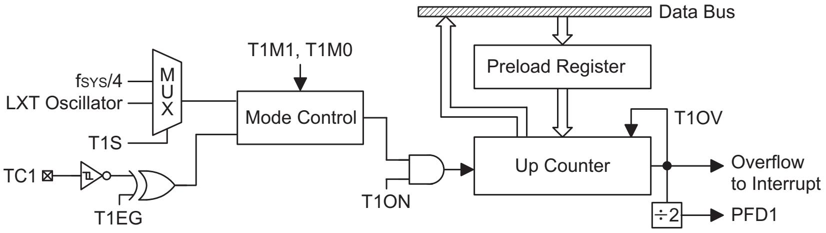
Note that if the Timer/Event Counter is in an OFF condition and data is written to its preload register, this data will be immediately written into the actual counter. However, if the counter is enabled and counting, any new data written into the preload data register during this period will remain in the preload register and will only be written into the actual counter the next time an overflow occurs.



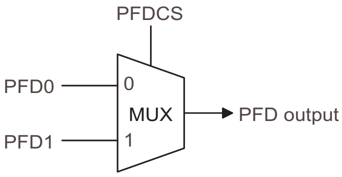
Clock Structure for Timer/Time Base



8-bit Timer/Event Counter 0 Structure



8-bit Timer/Event Counter 1 Structure



Timer Control Registers – TMR0C, TMR1C

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

The Timer Control Register is known as TMRnC. It is the Timer Control Register together with its corresponding Timer Register that controls the full operation of the Timer/Event Counter. Before the timer can be used, it is essential that the Timer Control Register is fully programmed with the right data to ensure its correct operation, a process that is normally carried out during program initialisation.

To choose which of the three modes the timer is to operate in, either in the timer mode, the event counting mode or the pulse width capture mode, bits 7 and 6 of the Timer Control Register, which are known as the bit pair TnM1/TnM0, must be set to the required logic levels. The timer-on bit, which is bit 4 of the Timer Control Register and known as TnON, provides the basic on/off control of the respective timer. Setting the bit high allows the counter to run, clearing the bit stops the counter. Bits 0~2 of the Timer Control Register, TMR0C, determine the division ratio of the input clock prescaler for Timer 0. Note that there is no prescaler selection in the TMR1C register for Timer 1. The prescaler bit settings have no effect if an external clock source is used. If the timer is in the event count or pulse width capture mode, the active transition edge level type is selected by the logic level of bit 3 of the Timer Control Register which is known as TnEG. The TnS bit selects the internal clock source if used.

TMR0C Register

Bit	7	6	5	4	3	2	1	0
Name	T0M1	T0M0	T0S	T0ON	T0EG	T0PSC2	T0PSC1	T0PSC0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	1	0	0	0

Bit 7,6 **T0M1, T0M0**: Timer0 operation mode selection
00: no mode available
01: event counter mode
10: timer mode
11: pulse width capture mode

Bit 5 **T0S**: timer clock source
0: f_{SYS}
1: LXT oscillator
T0S selects the clock source for f_{TP} which is provided for Timer 0 and the Time-Base.

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

Bit 3 **T0EG**:
Event counter active edge selection
0: count on raising edge
1: count on falling edge
Pulse Width Capture active edge selection
0: start counting on falling edge, stop on rasing edge
1: start counting on raising edge, stop on falling edge

Bit 2~0 **T0PSC2, T0PSC1, T0PSC0**: Timer prescaler rate selection
Timer internal clock=
000: f_{TP}
001: $f_{TP}/2$
010: $f_{TP}/4$
011: $f_{TP}/8$
100: $f_{TP}/16$
101: $f_{TP}/32$
110: $f_{TP}/64$
111: $f_{TP}/128$

TMR1C Register

Bit	7	6	5	4	3	2	1	0
Name	T1M1	T1M0	T1S	T1ON	T1EG	—	—	—
R/W	R/W	R/W	R/W	R/W	R/W	—	—	—
POR	0	0	0	0	1	—	—	—

Bit 7,6 **T1M1, T1M0**: Timer 1 operation mode selection
00: no mode available
01: event counter mode
10: timer mode
11: pulse width capture mode

Bit 5 **T1S**: Timer clock source
0: $f_{SYS}/4$
1: LXT oscillator

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

Bit 3 **T1EG**:
Event counter active edge selection
0: count on raising edge
1: count on falling edge
Pulse Width Capture active edge selection
0: start counting on falling edge, stop on rasing edge
1: start counting on raising edge, stop on falling edge

Bit 2~0 unimplemented, read as ″0″

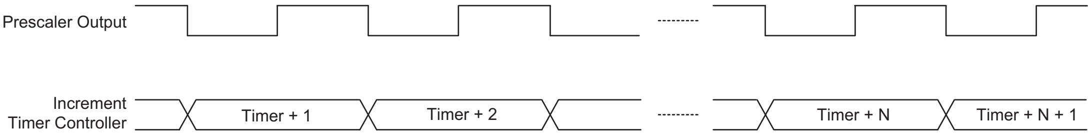
Timer Mode

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

Control Register Operating Mode Select Bits for the Timer Mode

Bit7	Bit6
1	0

In this mode the internal clock is used as the timer clock. The timer input clock source is either f_{SYS} , $f_{SYS}/4$ or the LXT oscillator. However, this timer 0 clock source is further divided by a prescaler, the value of which is determined by the bits T0PSC2~T0PSC0 in the Timer Control Register. The timer-on bit, TnON must be set high to enable the timer to run. Each time an internal clock high to low transition occurs, the timer increments by one; when the timer is full and overflows, an interrupt signal is generated and the timer will reload the value already loaded into the preload register and continue counting. A timer overflow condition and corresponding internal interrupt is one of the wake-up sources, however, the internal interrupts can be disabled by ensuring that the TnE bits of the INTC0 register are reset to zero.



Timer Mode Timing Chart

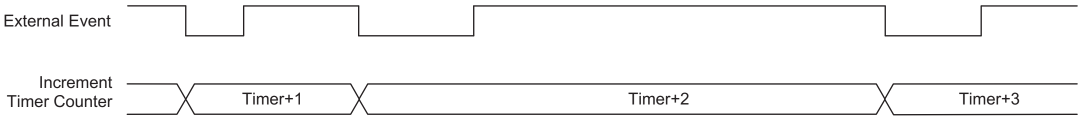
Event Counter Mode

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

Control Register Operating Mode Select Bits for the Event Counter Mode	Bit7	Bit6
	0	1

In this mode, the external timer TCn pin is used as the Timer/Event Counter clock source, however it is not divided by the internal prescaler. After the other bits in the Timer Control Register have been setup, the enable bit TnON, which is bit 4 of the Timer Control Register, can be set high to enable the Timer/Event Counter to run. If the Active Edge Select bit, TnEG, which is bit 3 of the Timer Control Register, is low, the Timer/Event Counter will increment each time the external timer pin receives a low to high transition. If the TnEG is high, the counter will increment each time the external timer pin receives a high to low transition. When it is full and overflows, an interrupt signal is generated and the Timer/Event Counter will reload the value already loaded into the preload register and continue counting. The interrupt can be disabled by ensuring that the Timer/Event Counter Interrupt Enable bit in the corresponding Interrupt Control Register is reset to zero.

As the external timer pin is shared with an I/O pin, to ensure that the pin is configured to operate as an event counter input pin, two things have to happen. The first is to ensure that the Operating Mode Select bits in the Timer Control Register place the Timer/Event Counter in the Event Counting Mode, the second is to ensure that the port control register configures the pin as an input. It should be noted that in the event counting mode, even if the microcontroller is in the Idle/Sleep Mode, the Timer/Event Counter will continue to record externally changing logic events on the timer input TCn pin. As a result when the timer overflows it will generate a timer interrupt and corresponding wake-up source.



Event Counter Mode Timing Chart (TnEG=1)

Pulse Width Capture Mode

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

Control Register Operating Mode Select Bits for the Pulse Width Capture Mode	Bit7	Bit6
	1	1

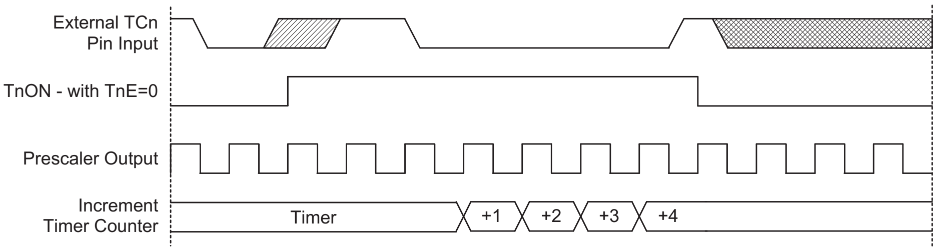
In this mode the internal clock, f_{SYS} , $f_{SYS}/4$ or the LXT, is used as the internal clock for the 8-bit Timer/Event Counter. However, the clock source, f_{SYS}, for the 8-bit timer is further divided by a prescaler, if the Timer 0 is utilised, the value of which is determined by the Prescaler Rate Select bits T0PSC2~T0PSC0, which are bits 2~0 in the Timer Control Register 0. After the other bits in the Timer Control Register have been setup, the enable bit TnON, which is bit 4 of the Timer Control Register, can be set high to enable the Timer/Event Counter, however it will not actually start counting until an active edge is received on the external timer pin.

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

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

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

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



Prescaler Output is sampled at every falling edge of T1.

Pulse Width Capture Mode Timing Chart (TnE=0)

Prescaler

Bits T0PSC0~T0PSC2 of the TMR0C register can be used to define a division ratio for the internal clock source of the Timer/Event Counter enabling longer time out periods to be setup.

PFD Function

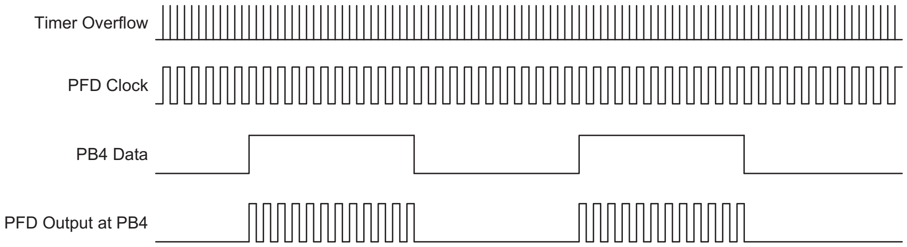
The Programmable Frequency Divider provides a means of producing a variable frequency output suitable for applications, such as piezo-buzzer driving or other interfaces requiring a precise frequency generator.

As the pin is shared with I/O pin, the function is selected using the SYSC register. The Timer/Event Counter overflow signal is the clock source for the PFD function, which is controlled by PFDCS bit in the SYSC. For applicable devices the clock source can come from either Timer/Event Counter 0 or Timer/Event Counter 1. The output frequency is controlled by loading the required values into the timer prescaler and timer registers to give the required division ratio. The counter will begin to count-up from this preload register value until full, at which point an overflow signal is generated, causing both the PFD output to change state. The counter will then be automatically reloaded with the preload register value and continue counting-up.

If the SYSC register has selected the PFD function, then for the PFD output to operate, it is essential for the Port control register, to setup the PFD pin as output. If only one pin is setup as an output, the other pin can still be used as a normal data input pin. However, if both pins are setup as inputs then the

PFD will not function. The bit PB4 or PA3 must be set high to activate the PFD. These output data bits can be used as the on/off control bit for the PFD outputs. Note that the PFD outputs will all be low if the output data bit is cleared to zero.

Using this method of frequency generation, and if a crystal oscillator is used for the system clock, very precise values of frequency can be generated.



PFD Function

Note: The PFD is pin-shared with PA3 as well. If this PA3 pn-shared functon is selected, the designer should refer the PB4 settings for PA3.

I/O Interfacing

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

Programming Considerations

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

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

When the Timer/Event Counter overflows, its corresponding interrupt request flag in the interrupt control register will be set. If the Timer/Event Counter interrupt is enabled this will in turn generate an interrupt signal. However irrespective of whether the interrupts are enabled or not, a Timer/Event Counter overflow will also generate a wake-up signal if the device is in a HALT 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 HALT 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 HALT Mode.

Timer Program Example

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

Timer Programming Example

```
org    04h               ; external interrupt vector

org    08h               ; Timer Counter 0 interrupt vector
jmp    tmr0int           ; jump here when Timer 0 overflows
:      :
org    20h               ; main program
:      :
;internal Timer 0 interrupt routine
tmr0int:
:
; Timer 0 main program placed here
:

:
begin:
;setup Timer 0 registers
mov    a,09bh            ; setup Timer 0 preload value
mov    tmr0,a
mov    a,081h            ; setup Timer 0 control register
mov    tmr0c,a           ; timer mode and prescaler set to /2
;setup interrupt register
mov    a,00dh            ; enable master interrupt and both timer interrupts
mov    intc0,a
:      :
set tmr0c.4              ; start Timer 0
:      :
```

Interrupts

Interrupts are an important part of any microcontroller system. When an external event or an internal function such as a Timer/Event Counter or Time Base 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 a single external interrupt and multiple internal interrupts. The external interrupt is controlled by the action of the external interrupt pin, while the internal interrupts are generated by the various functions such as Timer/Event Counters and Time Base.

Interrupt Registers

Overall interrupt control, which means interrupt enabling and request flag setting, is controlled by using two registers, INTC0 and INTC1. By controlling the appropriate enable bits in this registers 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 control bit if cleared to zero will disable all interrupts.

INTC0 Register

Bit	7	6	5	4	3	2	1	0
Name	—	T1F	T0F	INTF	T1E	T0E	INTE	EMI
R/W	—	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	0	0	0	0	0	0	0

Bit 7 unimplemented, read as ″0″

Bit 6 **T1F**: Timer/Event Counter 1 interrupt request flag
0: inactive
1: active

Bit 5 **T0F**: Timer/Event Counter 0 interrupt request flag
0: inactive
1: active

Bit 4 **INTF**: External interrupt request flag
0: inactive
1: active

Bit 3 **T1E**: Timer/Event Counter 1 interrupt enable
0: disable
1: enable

Bit 2 **T0E**: Timer/Event Counter 0 interrupt enable
0: disable
1: enable

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

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

INTC1 Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	TB1F	TB0F	—	—	TB1E	TB0E
R/W	—	—	R/W	R/W	—	—	R/W	R/W
POR	—	—	0	0	—	—	0	0

Bit 7~6 unimplemented, read as ″0″

Bit 5 **TB1F**: Time Base 1 interrupt request flag
0: no request
1: interrupt request

Bit 4 **TB0F**: Time Base 0 interrupt request flag
0: no request
1: interrupt request

Bit 3~2 unimplemented, read as ″0″

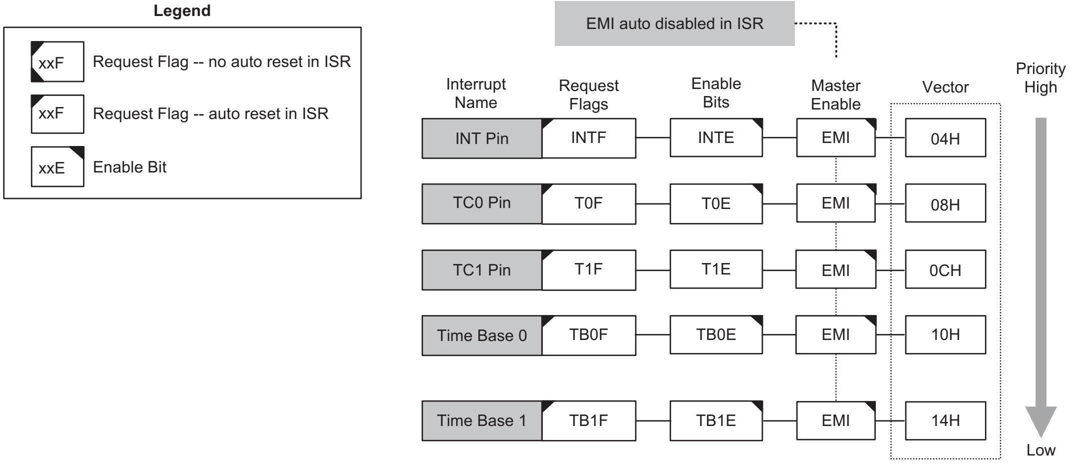
Bit 1 **TB1E**: Time Base 1 interrupt control
0: disable
1: enable

Bit 0 **TB0E**: Time Base 0 interrupt control
0: disable
1: enable

Interrupt Operation

A Timer/Event Counter overflow, an active edge on the external interrupt pin, or a Time Base event 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 instruction, 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.

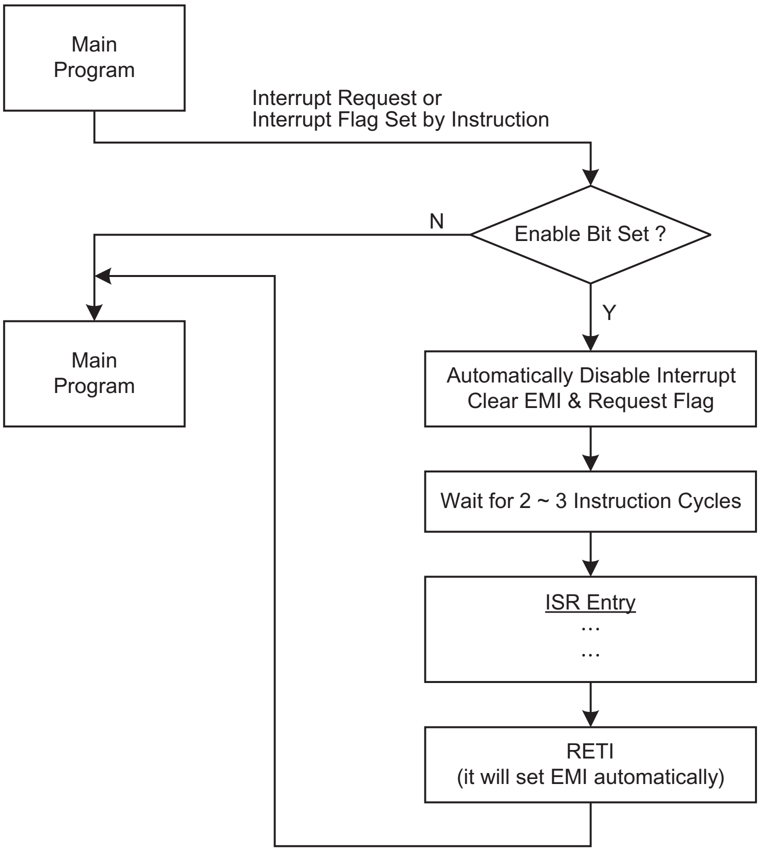
The various interrupt enable bits, together with their associated request flags, are shown in the following diagram with their order of priority.



Interrupt Scheme

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.

When an interrupt request is generated, it takes 2 or 3 instruction cycle before the program jumps to the interrupt vector. If the device is in the Sleep or Idle Mode and is woken up by an interrupt request, then it will take 3 cycles before the program jumps to the interrupt vector.


Main Program
Interrupt Request or
Interrupt Flag Set by Instruction
N
Enable Bit Set ?
Y
Main Program
Automatically Disable Interrupt
Clear EMI & Request Flag
Wait for 2 ~ 3 Instruction Cycles
ISR Entry
…
…
RETI
(it will set EMI automatically)

Interrupt Flow

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	Vector
External Interrupt	1	04H
Timer/Event Counter 0 Overflow	2	08H
Timer/Event Counter 1 Overflow	3	0CH
Time Base 0 Overflow	4	10H
Time Base 1 Overflow	5	14H

In cases where both external and internal interrupts are enabled and where an external and internal interrupt occurs simultaneously, the external interrupt will always have priority and will therefore be serviced first. Suitable masking of the individual interrupts using the interrupt registers can prevent simultaneous occurrences.

External Interrupt

The external interrupt pin is pin-shared with the I/O pin PB5 and can only be configured as an external interrupt pin if the corresponding external interrupt enable bit in the INTC0 register has been set. The pin must also be setup as an input by setting the corresponding PBC.5 bit in the port control register. When the interrupt is enabled, the stack is not full and a transition appears on the external interrupt pin, a subroutine call to the external interrupt vector at location 04H, will take place. When the interrupt is serviced, the external interrupt request flag, INTF, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts. Note that any pull-high resistor connections on this pin will remain valid even if the pin is used as an external interrupt input.

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, TnE, must first be set. An actual Timer/Event Counter interrupt will take place when the Timer/Event Counter request flag, TnF, 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 n overflow occurs, a subroutine call to the relevant timer interrupt vector, will take place. When the interrupt is serviced, the timer interrupt request flag, TnF, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts.

Time Base Interrupts

The function of the Time Base Interrupts is to provide regular time signal in the form of an internal interrupt. They are controlled by the overflow signals from their respective timer functions. When these happens their respective interrupt request flags, TB0F or TB1F will be set. To allow the program to branch to their respective interrupt vector addresses, the global interrupt enable bit, EMI and Time Base enable bits, TB0E or TB1E, must first be set. When the interrupt is enabled, the stack is not full and the Time Base overflows, a subroutine call to their respective vector locations will take place. When the interrupt is serviced, the respective interrupt request flag, TB0F or TB1F, will be automatically reset and 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. Their clock sources originate from the internal clock source f_{TB}. This f_{TB} input clock passes through a divider, the division ratio of which is selected by programming the appropriate bits in the TBC register to obtain longer interrupt periods whose value ranges. The clock source that generates f_{TB}, which in turn controls the Time Base interrupt period, originates from the LXT oscillator, as shown in the System Operating Mode section.

TBC Register

Bit	7	6	5	4	3	2	1	0
Name	TBON	—	TB11	TB10	—	TB02	TB01	TB00
R/W	R/W	—	R/W	R/W	—	R/W	R/W	R/W
POR	0	—	1	1	—	1	1	1

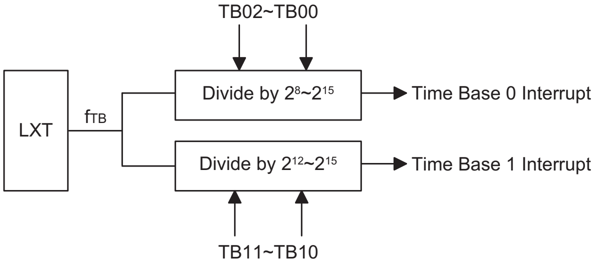
Bit 7 **TBON**: TB0 and TB1 Control
0: disable
1: enable

Bit 6 unimplemented, read as "0"

Bit 5~4 **TB11~TB10**: Select Time Base 1 Time-out Period
00: $4096/f_{TB}$
01: $8192/f_{TB}$
10: $16384/f_{TB}$
11: $32768/f_{TB}$

Bit 3 unimplemented, read as "0"

Bit 2~0 **TB02**~TB00: Select Time Base 0 Time-out Period
000: $256/f_{TB}$
001: $512/f_{TB}$
010: $1024/f_{TB}$
011: $2048/f_{TB}$
100: $4096/f_{TB}$
101: $8192/f_{TB}$
110: $16384/f_{TB}$
111: $32768/f_{TB}$



Time Base Interrupt

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 interrupt register 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 HALT 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.

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 COM and SEG 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. These devices all contain an LCD Driver function, which with their internal LCD signal generating circuitry and various options, 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.

The table shows the range of options available across the device range.

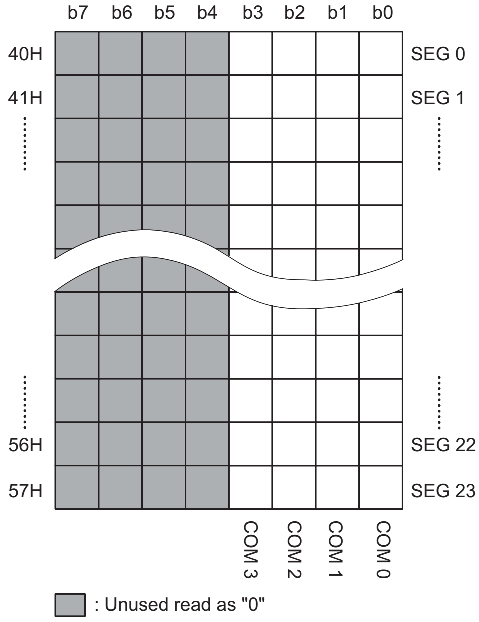
Duty	Driver No.	Bias	Bias Type	Waveform Type
1/4	24x4	1/2 or 1/3	C	A or B

LCD Memory

An area of Data Memory is especially reserved to use for the LCD display data. This data area is known as the LCD Memory. Any data written here will be automatically read by the internal display driver circuits, which will in turn automatically generate the necessary LCD driving signals. Therefore any data written into this Memory will be immediately reflected into the actual display connected to the microcontroller.

As the LCD Memory addresses overlap those of the General Purpose Data Memory, it′s stored in its own independent Bank 1 area. The Data Memory Bank to be used is chosen by using the Bank Pointer, which is a special function register in the Data Memory, with the name, BP. To access the LCD Memory therefore requires first that Bank 1 is selected by writing a value of 01H to the BP register. After this, the 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, address from ″40H″ to ″57H″, will result in operations to the LCD Memory. Directly addressing the Display Memory is not applicable and will result in a data access to the Bank 0 General Purpose Data Memory.

The accompanying LCD Memory Map diagrams shows how the internal LCD Memory is mapped to the Segments and Commons of the display for the devices. LCD Memory Maps for devices with smaller memory capacities can be extrapolated from these diagrams.



LCD Memory Map

LCD Registers

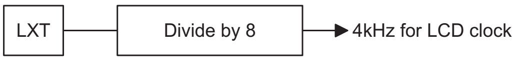
Control Registers in the Data Memory, are used to control the various setup features of the LCD Driver.

There is one control register for the LCD function, LCDC.

Various bits in this registers control functions such as bias type as well as overall LCD enable and disable. The LCDEN bit in the LCDC register, which provides the overall LCD enable/disable function, will be effective when the device is in the Normal or HALT Mode. The LCDWT bit in the same register is used to select whether Type A or Type B LCD control signals are used. The register, SEGC is used to determine if the output function of display pins SEG0~SEG7 are used as segment drivers or I/O functions.

Clock Source

The LCD clock source is the LXT Oscillator divided by 8 to generate an ideal LCD clock source frequency of 4kHz.



LCD Driver Output

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 may 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 is fixed at a value of 1/4 and which equates to a COM number of 4, therefore defines the number of time divisions within each LCD signal frame. Two types of signal generation are also provided, known as Type A and Type B, the required type is selected via the LCDWT bit in the LCDC register. Type B offers lower frequency signals, however lower frequencies may introduce flickering and influence display clarity.

LCDC Register

Bit	7	6	5	4	3	2	1	0
Name	LCDWT	—	—	—	LCDB	—	—	LCDEN
R/W	R/W	—	—	—	R/W	—	—	R/W
POR	0	—	—	—	0	—	—	0

Bit 7 **LCDWT**: LCD Type Control
0: Type A
1: Type B

Bit 6~4 unimplemented, read as "0"

Bit 3 **LCDB**: LCD Bias Control
0: 1/2 Bias
1: 1/3 Bias

Bit 2~1 Unimplemented

Bit 0 **LCDEN**: LCD Enable Control
0: Disable
1: Enable
In the Normal or HALT mode, the LCD on/off function can be controlled by this bit.

SEGC Register

Bit	7	6	5	4	3	2	1	0
Name	SEG7C	SEG6C	SEG5C	SEG4C	SEG3C	SEG2C	SEG1C	SEG0C
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	1	1	1	1	1	1	1	1

Bit 7~0 **SEG7C~SEG0C**: SEG7~SEG0 output or PA7~PA0
0: LCD Segment output
1: I/O

LCD Voltage Source and Biasing

The time and amplitude varying signals generated by the LCD Driver function require the generation of several voltage levels for their operation. The number of voltage levels used by the signal depends upon the value of the LCDB bit in the LCDC register.

The device provides C type biasing with a value of either 1/2 or 1/3. For C type biasing, an external LCD voltage source can be supplied on LVB pin or LVA pin to generate the internal biasing voltages. The C type biasing scheme uses an internal charge pump circuit, which in the case of the 1/3 bias software control option can generate voltages higher than what is supplied on LVB pin. This feature is useful in applications where the microcontroller supply voltage is less than the supply voltage required by the LCD. An additional charge pump capacitor must also be connected between pins C1 and C2 to generate the necessary voltage levels.

For the C type 1/2 bias selection, three voltage levels VSS, VA and VB are utilised. The voltage VA is generated internally and has a value of LVB voltage. The internal VB biasing voltage will have a value equal to LVB voltage x 0.5. For the C type 1/2 bias software control option VC is not used.

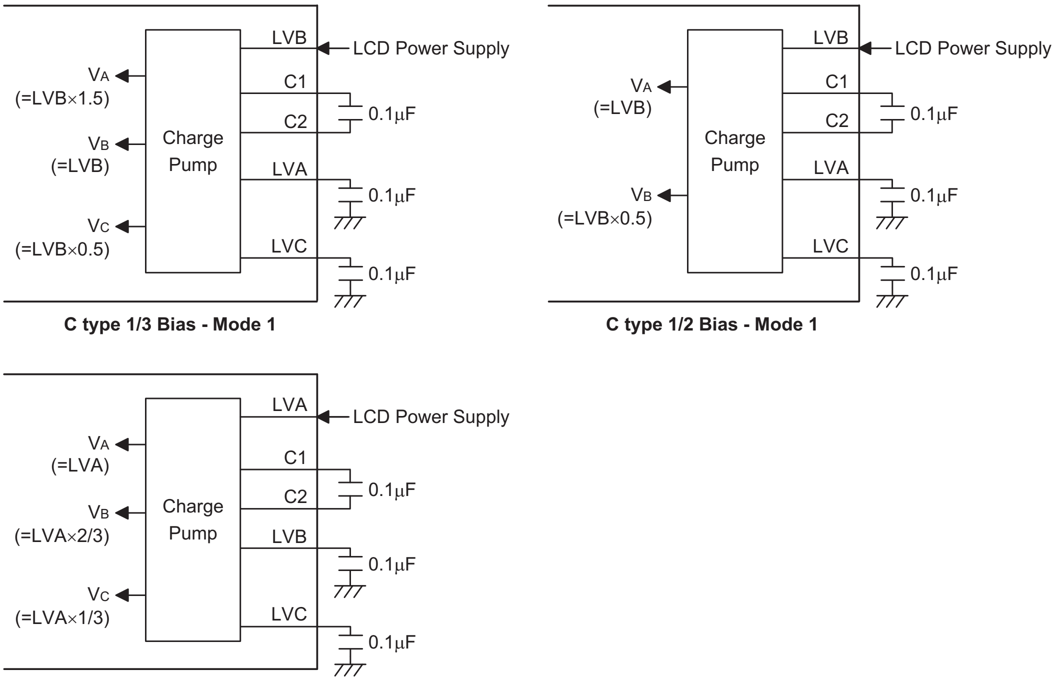
For the C type 1/3 bias selection, four voltage levels VSS, VA, VB and VC are utilised. There are two LCD Power Supply input options, namely mode 1 and mode 2.

In mode 1, the LCD Power Supply is supplied on the LVB pin. The voltage VA is generated internally and has a value of the LVB voltage x1.5, VB will have a value equal to the LVB voltage and VC will have a value equal to the LVB voltage x 0.5. The LVA voltage must be greater than or equal to VDD.

In mode 2, the LCD Power Supply is supplied on the LVA pin. The voltage VA is generated internally and has a value of the LVA voltage. The voltage VB will have a value equal to the LVA voltage x 2/3 and VC will have a value equal to LVA voltage x 1/3.

Note that the 28-pin package type only supports the LCD Power Supply mode 2 function.

The following diagrams illustrate the basic LCD Bias functional block diagrams.



C type 1/3 Bias - Mode 1

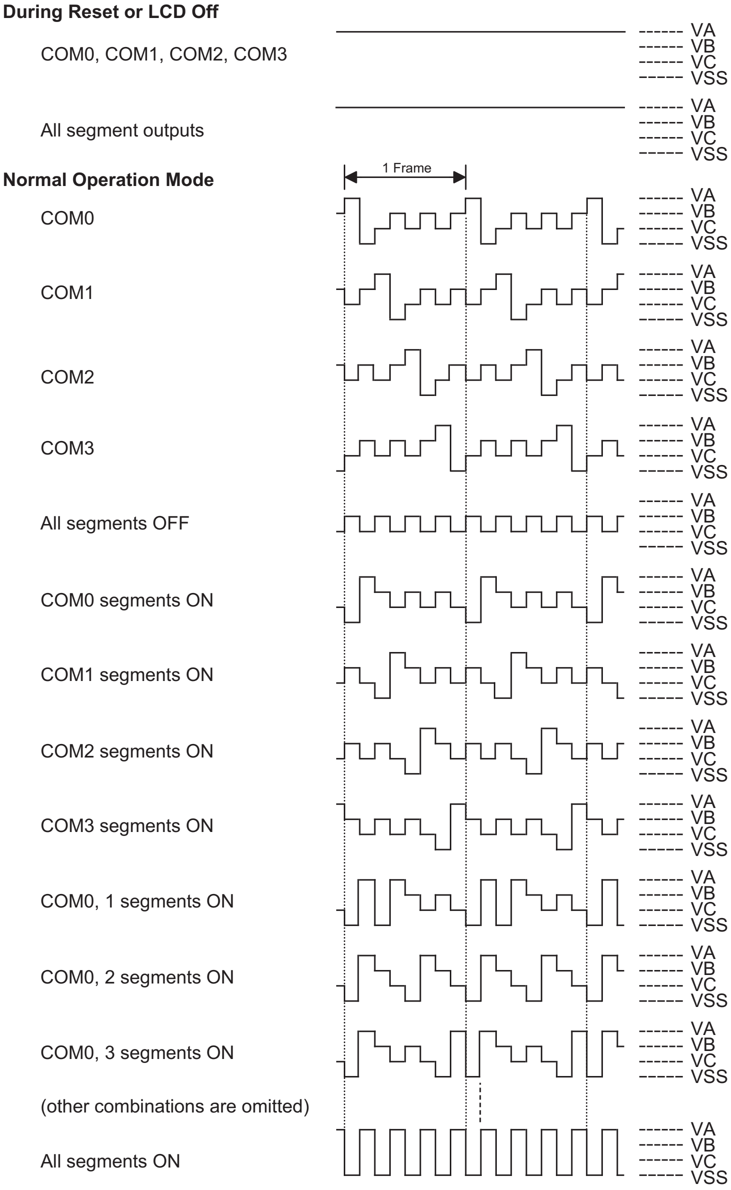
C type 1/2 Bias - Mode 1

C type 1/3 Bias - Mode 2

C Type Bias Voltage Levels

LCD Waveform Timing Diagrams

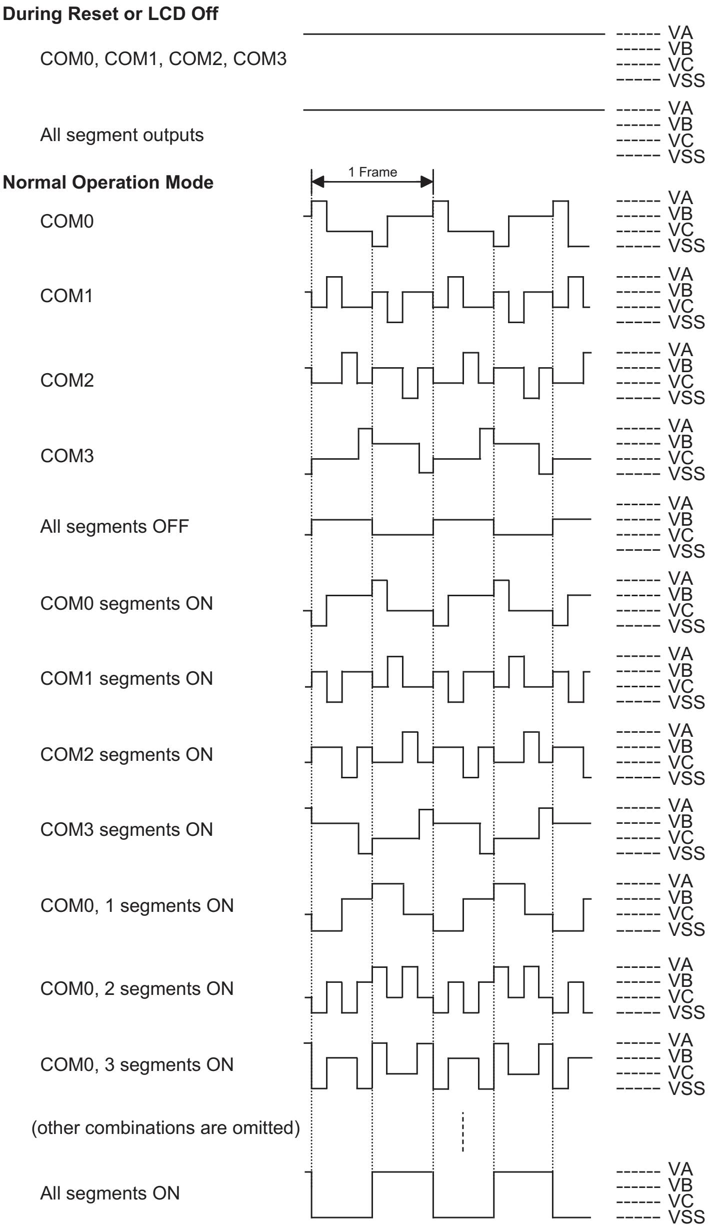
The accompanying timing diagrams depict the display driver signals generated by the microcontroller for various values of duty and bias. The huge range of various permutations only permit a few types to be displayed here.



LCD Driver Output − Type A - 1/4 Duty, 1/3 Bias

Note: 1. LCD power supply mode 1, VA=LVB×1.5, VB=LVB and VC=LVB×1/2.

2. LCD power supply mode 2, VA=LVA, VB=LVA×2/3 and VC=LVA×1/3.



LCD Driver Output – Type B - 1/4 Duty, 1/3 Bias

Note: 1. LCD power supply mode 1, VA=LVB×1.5, VB=LVB and VC=LVB×1/2.

2. LCD power supply mode 2, VA=LVA, VB=LVA×2/3 and VC=LVA×1/3.

Programming Considerations

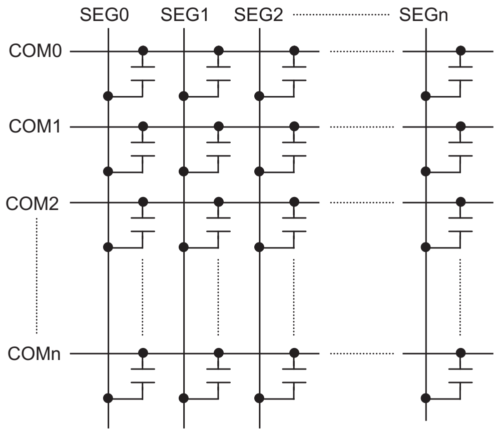
Certain precautions must be taken when programming the LCD. One of these is to ensure that the LCD

Memory is properly initialised 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 display, it is important to initialise 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.

One additional consideration that must be taken into account is what happens when the microcontroller enters the HALT Mode. The LCDEN control bit in the LCDC register permits the display to be powered off to reduce power consumption. If this bit is zero, the driving signals to the display will cease, producing a blank display pattern but reducing any power consumption associated with the LCD.

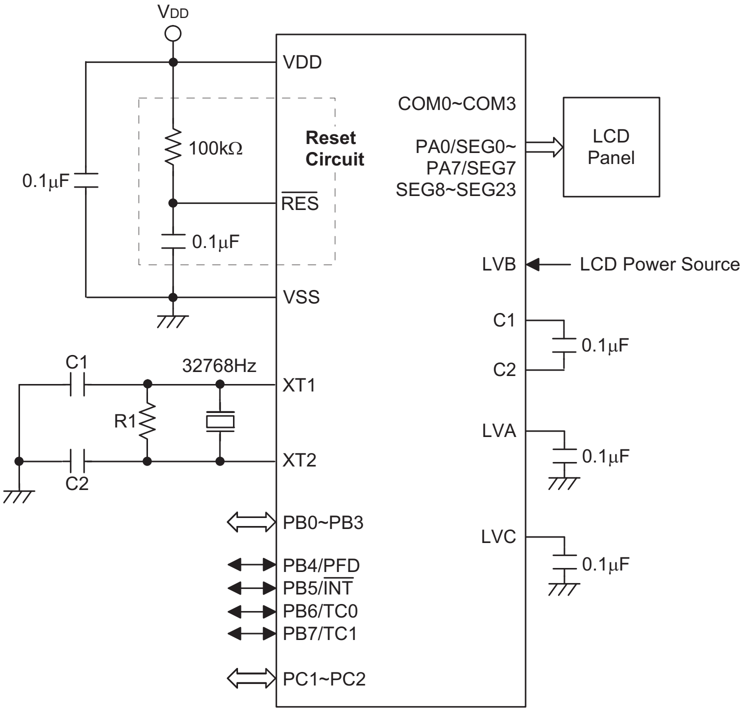
After Power-on, note that as the LCDEN bit will be cleared to zero, the display function will be disabled.



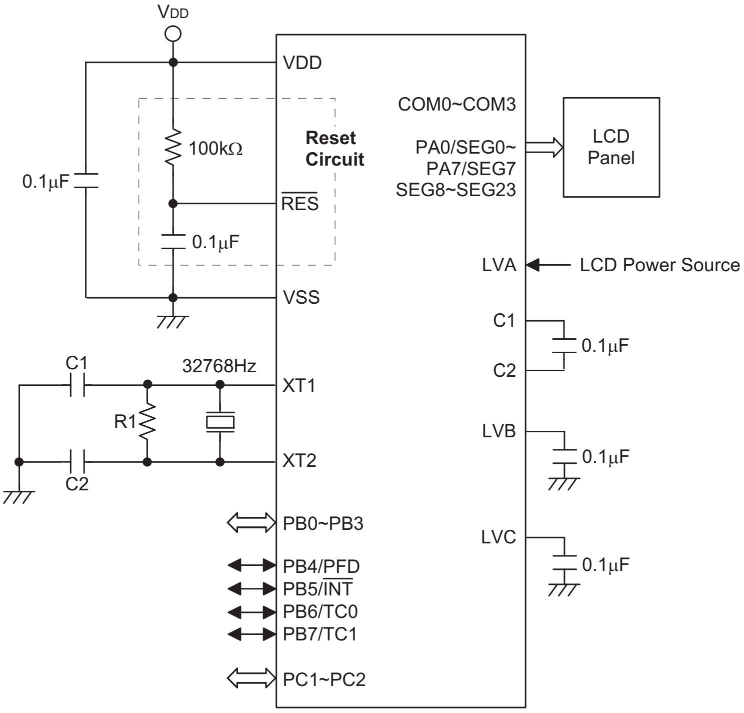
LCD Panel Equivalent Circuit

Application Circuits

LCD Application Mode 1



LCD Application Mode 2



Note: The 28-pin package only provides LCD power supply mode 2 as the LVA pin is internally connected to VDD.

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
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

Mnemonic	Description	Cycles	Flag Affected
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][4]	Read ROM code (locate by TBLP and TBHP) to data memory and TBLH	2^{Note}	None
TABRDC [m][5]	Read ROM code (current page) to data memory and TBLH	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.
4. ″TBHP option″ is enabled by TBRC bit of SYSC register.
5. ″TBHP option″ is disabled by TBRC bit of SYSC register.

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	$[m] \leftarrow \overline{[m]}$
Affected flag(s)	Z
CPLA [m]	Complement Data Memory with result in ACC
Description	Each bit of the specified Data Memory is logically complemented (1′s complement). Bits which previously contained a 1 are changed to 0 and vice versa. The complemented result is stored in the Accumulator and the contents of the Data Memory remain unchanged.
Operation	$ACC \leftarrow \overline{[m]}$
Affected flag(s)	Z
DAA [m]	Decimal-Adjust ACC for addition with result in Data Memory
Description	Convert the contents of the Accumulator value to a BCD (Binary Coded Decimal) value resulting from the previous addition of two BCD variables. If the low nibble is greater than 9 or if AC flag is set, then a value of 6 will be added to the low nibble. Otherwise the low nibble remains unchanged. If the high nibble is greater than 9 or if the C flag is set, then a value of 6 will be added to the high nibble. Essentially, the decimal conversion is performed by adding 00H, 06H, 60H or 66H depending on the Accumulator and flag conditions. Only the C flag may be affected by this instruction which indicates that if the original BCD sum is greater than 100, it allows multiple precision decimal addition.
Operation	[m] ← ACC + 00H or [m] ← ACC + 06H or [m] ← ACC + 60H or [m] ← ACC + 66H
Affected flag(s)	C
DEC [m]	Decrement Data Memory
Description	Data in the specified Data Memory is decremented by 1.
Operation	[m] ← [m] – 1
Affected flag(s)	Z
DECA [m]	Decrement Data Memory with result in ACC
Description	Data in the specified Data Memory is decremented by 1. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.
Operation	ACC ← [m] – 1
Affected flag(s)	Z
HALT	Enter power down mode
Description	This instruction stops the program execution and turns off the system clock. The contents of the Data Memory and registers are retained. The WDT and prescaler are cleared. The power down flag PDF is set and the WDT time-out flag TO is cleared.

Operation	TO ← 0
	PDF ← 1

Affected flag(s)	TO, PDF

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Affected flag(s)	None

RLC [m] Rotate Data Memory left through Carry

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

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

Affected flag(s): C

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

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

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

Affected flag(s): C

RR [m] Rotate Data Memory right

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

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

Affected flag(s): None

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

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

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

Affected flag(s): None

RRC [m] Rotate Data Memory right through Carry

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

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

Affected flag(s): C

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

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

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

Affected flag(s): C

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

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

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

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] ← [m] + 1
Skip if [m] = 0

Affected flag(s) None

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

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

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

Affected flag(s) None

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

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

Operation Skip if [m].i ≠ 0

Affected flag(s) None

SUB A,[m] Subtract Data Memory from ACC

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

Operation ACC ← ACC − [m]

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

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

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

Operation [m] ← ACC − [m]

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

SUB A,x Subtract immediate data from ACC

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

Operation ACC ← ACC − x

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

SWAP [m] Swap nibbles of Data Memory

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

Operation [m].3~[m].0 ↔ [m].7 ~ [m].4

Affected flag(s) None

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

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

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

Affected flag(s) None

SZ [m] Skip if Data Memory is 0

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

Operation Skip if [m] = 0

Affected flag(s) None

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

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

Operation ACC ← [m]
Skip if [m] = 0

Affected flag(s) None

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

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

Operation Skip if [m].i = 0

Affected flag(s) None

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

TABRDC [m] Move the ROM code (locate by TBLP and TBHP) to TBLH and data memory (ROM code TBHP is enabled)

Description The low byte of ROM code addressed by the table pointers (TBLP and TBHP) is moved to the specified data memory and the high byte transferred to TBLH directly.

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

Affected flag(s) None

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

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

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

Affected flag(s) None

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

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

Operation ACC ← ACC ″XOR″ [m]

Affected flag(s) Z

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

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

Operation [m] ← ACC ″XOR″ [m]

Affected flag(s) Z

XOR A,x Logical XOR immediate data to ACC

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

Operation ACC ← ACC ″XOR″ x

Affected flag(s) Z

Package Information

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

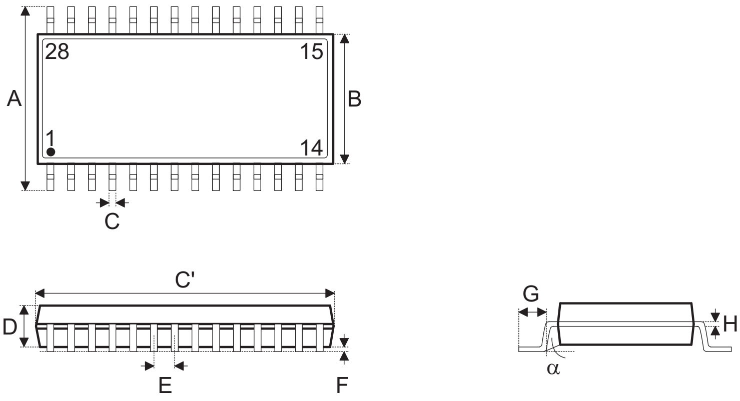
Additional supplementary information with regard to packaging is listed below. Click on the relevant section to be transferred to the relevant website page.

- Package Information (include Outline Dimensions, Product Tape and Reel Specifications)

- The Operation Instruction of Packing Materials

- Carton information

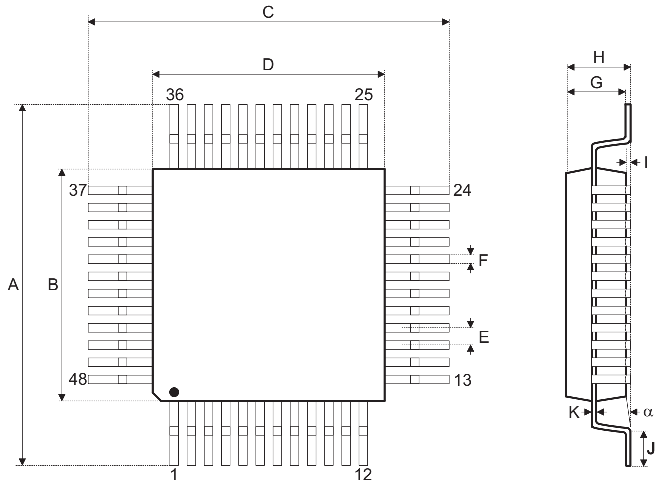
28-pin SSOP (150mil) Outline Dimensions



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

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	—	6.000 BSC	—
B	—	3.900 BSC	—
C	0.20	—	0.30
C′	—	9.900 BSC	—
D	—	—	1.72
E	—	0.635 BSC	—
F	0.10	—	0.25
G	0.41	—	1.27
H	0.10	—	0.25
α	0°	—	8°

48-pin LQFP (7mm×7mm) Outline Dimensions



Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	—	0.354 BSC	—
B	—	0.276 BSC	—
C	—	0.354 BSC	—
D	—	0.276 BSC	—
E	—	0.020 BSC	—
F	0.007	0.009	0.011
G	0.053	0.055	0.057
H	—	—	0.063
I	0.002	—	0.006
J	0.018	0.024	0.030
K	0.004	—	0.008
α	0°	—	7°

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	—	9.0 BSC	—
B	—	7.0 BSC	—
C	—	9.0 BSC	—
D	—	7.0 BSC	—
E	—	0.5 BSC	—
F	0.17	0.22	0.27
G	1.35	1.40	1.45
H	—	—	1.60
I	0.05	—	0.15
J	0.45	0.60	0.75
K	0.09	—	0.20
α	0°	—	7°

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.