I/O Type USB 8-Bit OTP MCU with SPI

HT82A525R

Revision: 1.60 Date: January 22, 2014

www.holtek.com

Table of Contents

Features **4**
General Description **4**
Block Diagram **5**
Pin Assignment **6**
Pin Description **7**
Absolute Maximum Ratings **8**
D.C. Characteristics **9**
A.C. Characteristics **10**
System Architecture **11**
Clocking and Pipelining 11
Program Counter **12**
Stack 12
Arithmetic and Logic Unit – ALU 13
Program Memory **13**
Organisation 13
Special Vectors 14
Data Memory – RAM **16**
Indirect Addressing Register 16
Accumulator – ACC 16
Status Register – STATUS 16
Interrupts 18
Oscillator Configuration **20**
External Crystal Oscillator – HXT 21
Internal RC Oscillator 21
Internal 12kHz Oscillator – WDTOSC 22
Watchdog Timer – WDT **22**
Power Down Operation – HALT **23**
Reset **24**
Timer/Event Counter **27**
Input/Output Ports **29**
Low Voltage Reset – LVR **30**

SPI Serial Interface 30
SPI Interface Communication 30
SPI pin-shared I/O Option Table 31
SPIn Registers 31
SPIn Bus Enable/Disable 32
SPIn Operation 32
SPIn Configuration Options 34
Error Detection 34
Programming Considerations 34
DMA Function 37
Suspend Wake-Up or Remote Wake-Up 40
USB Interface 41
Pulse Width Modulator 47
PWM Registers 47
8+4 PWM Mode Modulation 48
PWM Output Control 48
PWM Programming Example 49
COM Function for LCD 50
LCD Operation 50
LCD Bias Control 51
Options 52
Application Circuits 53
Instruction Set 54
Introduction 54
Instruction Timing 54
Moving and Transferring Data 54
Arithmetic Operations 54
Logical and Rotate Operations 54
Branches and Control Transfer 55
Bit Operations 55
Table Read Operations 55
Other Operations 55
Instruction Set Summary 56
Instruction Definition 58
Package Information 68
24-pin SSOP (150mil) Outline Dimensions 69
SAW Type 32-pin (5mm×5mm) QFN Outline Dimensions 70
48-pin LQFP (7mm×7mm) Outline Dimensions 71

Features

- Operating voltage:
  f_{SYS}=6MHz: 3.3V~5.5V
  f_{SYS}=12MHz: 4.5V~5.5V
- Two oscillators:
  External crystal -- HXT (6MHz or 12 MHz)
  Internal RC -- HIRC (12MHz)
- Fully integrated internal 12MHz oscillator requires no external components
- 4K×15 Program Memory
- 192×8 Data Memory RAM
- USB 2.0 Full Speed Compatible
- Single 16-bit programmable Timer/Event Counters with overflow interrupt
- Single 8-bit programmable Timer/Event Counters with overflow interrupt
- Two SPI interfaces shared with I/O lines
- Support SPI serial DMA or 8-bit parallel DMA
- Support serial clock for peripheral device
- Software controlled 4-COM lines LCD driver with 1/2 bias
- Watchdog Timer function
- Power down and wake-up functions to reduce power consumption
- 3-channel 12-bit PWM output shared with three I/O lines
- Up to 42 bidirectional I/O lines
- Up to 0.33μs instruction cycle with 12MHz system clock at V_{DD}=5V
- 4 endpoints supported (endpoint 0 included)
- Support interrupt, control & bulk transfer
- Total FIFO size is 208 bytes (8,8,64,64×2 for EP0~EP3)
- 6-level subroutine nesting
- Bit manipulation instruction
- Table read instructions
- 63 powerful instructions
- All instructions executed in one or two instruction cycles
- Low voltage reset function (2.2V±0.2V)
- 24-pin SSOP, 32-pin QFN, 48-pin LQFP packages

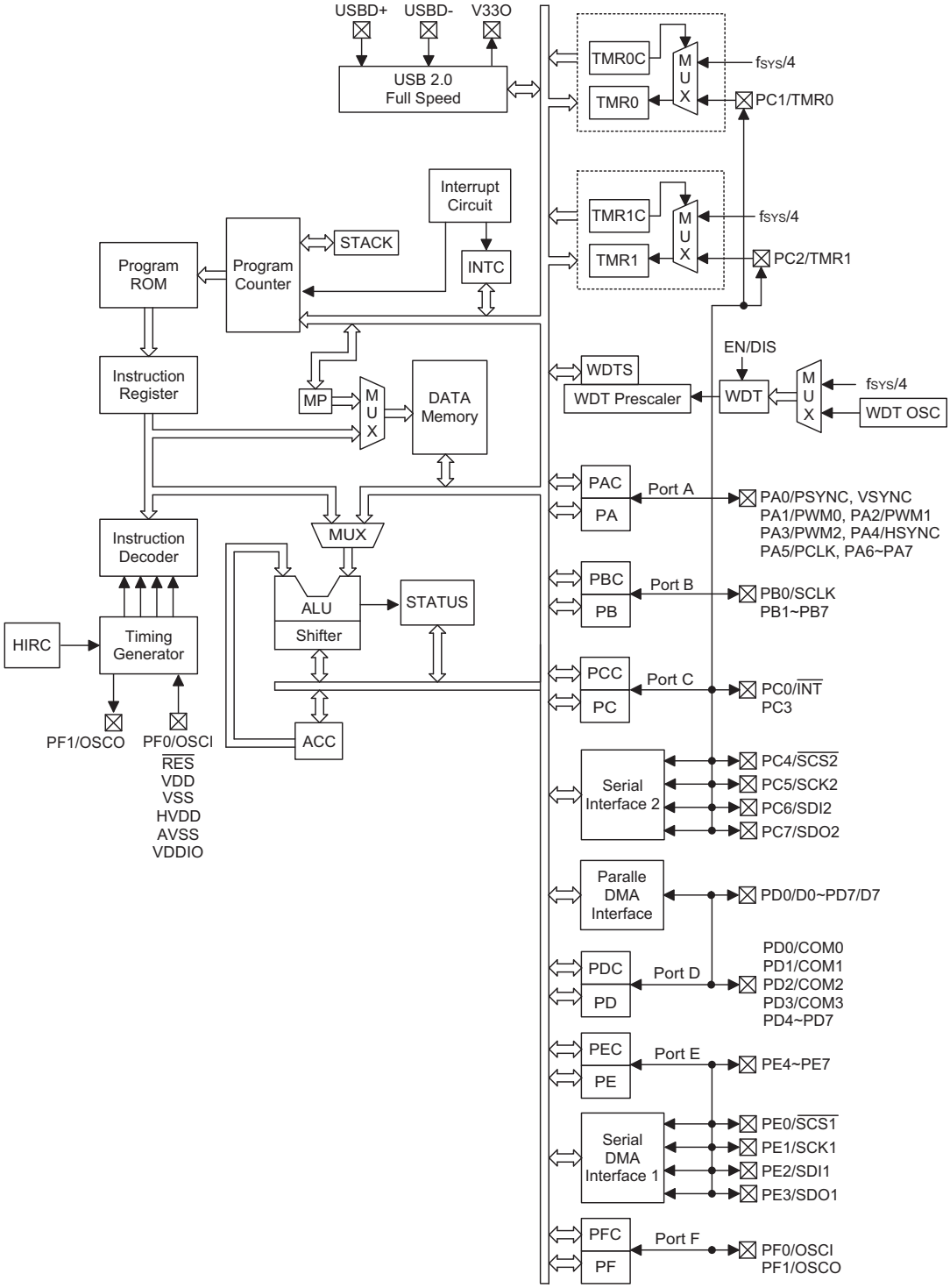
General Description

This HT82A525R is an 8-bit high performance RISC microcontroller designed for USB keyboard mouse and fingerprint product applications.

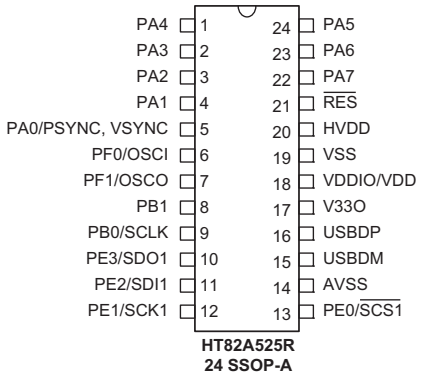
The advantages of low power consumption, I/O flexibility, timer functions, oscillator options, multi-channel Pulse Width Modulation function, Watchdog timer, dual SPI interfaces, SPI serial DMA or 8-bit parallel DMA , Power Down and wake-up functions, combine to provide the device with a huge range of functional options while still maintaining a high level of cost effectiveness. The device includes a fully integrated system oscillator HIRC, which requires no external components. While the DMA function opens up application possibilities such as fingerprint data processing, the device is also suited for a range of other applications such as motor driving, industrial control, consumer products, subsystem controllers, etc.

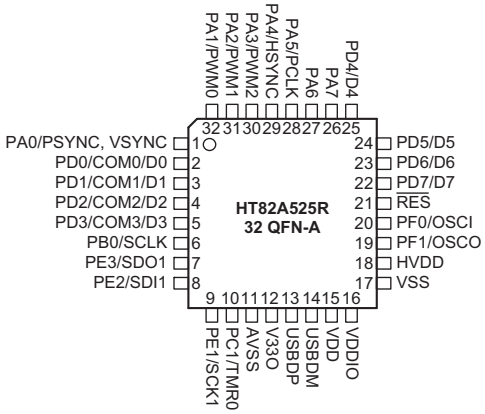
Block Diagram

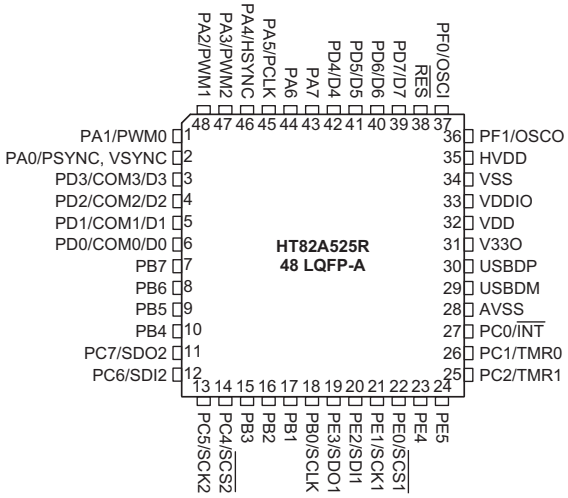
The following block diagram illustrates the main functional blocks.



Pin Assignment







Pin Description

Pin Name	I/O	Options	Description
PA0/PSYNC/VSYNC PA1/PWM0 PA2/PWM1 PA3/PWM2 PA4/HSYNC PA5/PCLK PA6~PA7	I/O	Pull-high (bit option) Wake-up (bit option)	Bidirectional 8-bit input/output port. Each individual pin on this port can be configured as a wake-up input by a configuration option. Software instructions determine if the pin is a CMOS output or Schmitt Trigger input. Configuration options determine which pins on the port have pull-high resistors. The PSYNC/VSYNC is pin shared with PA0 (dependent on PSYNC/VSYNC option). PA1~PA3 are pin-shared with PWM0~PWM2. HSYNC is pin shared with PA4 (dependent on HSYNC option). PCLK is pin shared with PA5 (dependent on PCLK option). The power supply for the I/O pins is sourced from the VDDIO pin.
PB0/SCLK PB1~PB7	I/O	Pull-high (bit option) CMOS/NMOS wake-up (nibble option)	Bidirectional 8-bit input/output port. Each nibble on this port can be configured as a wake-up input by a configuration option. Software instructions determine if the pin is a CMOS output or Schmitt Trigger input. Configuration options determine which pins on the port have pull-high resistors. PB0~PB7 have CMOS/NMOS options. SCLK is pin shared with PB0 (dependent on SCLK option). The power supply for the I/O pins is sourced from the VDDIO pin.
PC0/$\overline{INT}$ PC1/TMR0 PC2/TMR1 PC3 PC4/$\overline{SCS2}$ PC5/SCK2 PC6/SDI2 PC7/SDO2	I/O	Pull-high (nibble option) wake-up (nibble option)	Bidirectional I/O lines. Each nibble on this port can be configured as a wake-up input by a configuration option. Software instructions determine if the pin is a CMOS output or Schmitt Trigger input. Configuration options determine which nibble on the port have pull-high resistors. $\overline{INT}$, TMR0, TMR1 are pin shared with PC0, PC1, PC2 respectively. $\overline{SCS2}$ is pin shared with PC4. SCK2 is pin shared with PC5. SDI2 is pin shared with PC6. SDO2 is pin shared with PC7. The power supply for the I/O pins is sourced from the VDDIO pin.
PD0/COM0/D0 PD1/COM1/D1 PD2/COM2/D2 PD3/COM3/D3 PD4/D4~PD7/D7	I/O	Pull-high (nibble option) wake-up (nibble option)	Bidirectional I/O lines. Each nibble on this port can be configured as a wake-up input by a configuration option. Software instructions determine if the pin is a CMOS output or Schmitt Trigger input. Configuration options determine which nibble on the port have pull-high resistors. PD0~PD3 are pin-shared with COM0~COM3. The parallel DMA data D0~D7 lines are pin shared with PD0~PD7. The power supply for the I/O pins is sourced from the VDDIO pin.
PE0/$\overline{SCS1}$ PE1/SCK1 PE2/SDI1 PE3/SDO1 PE4~PE7	I/O	Pull-high (nibble option) wake-up (nibble option)	Bidirectional I/O lines. Each nibble on this port can be configured as a wake-up input by a configuration option. Software instructions determine if the pin is a CMOS output or Schmitt Trigger input. Configuration options determine which nibble on the port have pull-high resistors. $\overline{SCS1}$ is pin shared with PE0. SCK1 is pin shared with PE1. SDI1 is pin shared with PE2. SDO is pin shared with PE3. The power supply for the I/O pins is sourced from the VDDIO pin.
PF0/OSCI	I/O (PF0)	Pull-high wake-up	Bidirectional I/O line. The pin can be configured as a wake-up input by a configuration option. Software instructions determine if the pin is a CMOS output or Schmitt trigger input. A configuration option determines if the pin has a pull-high resistor. The power supply for the I/O pins is sourced from the VDD pin.
	I(OSCI)	—	External crystal connection

Pin Name	I/O	Options	Description
PF1/OSCO	I/O (PF1)	Pull-high wake-up	Bidirectional I/O line. The pin can be configured as a wake-up input by a configuration option. Software instructions determine if the pin is a CMOS output or Schmitt trigger input. A configuration option determines if the pin has a pull-high resistor. The power supply for the I/O pins is sourced from the VDD pin.
	O(OSCO)	—	External crystal connection
$\overline{RES}$	I	—	Schmitt trigger reset input, active low
VSS	—	—	HIRC and digital negative power supply, ground
AVSS	—	—	Analog negative power supply, ground
VDDIO	—	—	Positive power supply, VDDIO is used for PA, PB, PC, PD, PE, except PF. For 24-pin SSOP package, the VDDIO pin is double bonding to VDD pin.
HVDD	—	—	HIRC Positive power supply
VDD	—	—	Analog and digital positive power supply
V33O	O	—	3.3V regulator output, can be disabled by firmware
USBDP	I/O	—	USB D+ or 3D PS2 data line. USB function is controlled using software control registers. PS2 function is controlled by the SELPS2 bit in USC register.
USBDM	I/O	—	USB D- or 3D PS2 CLK line. USB function is controlled using software control registers. PS2 function is controlled by the SELPS2 bit in USC register.

Absolute Maximum Ratings

Supply Voltage ..... V_{SS}–0.3V to V_{SS}+6.0V
Storage Temperature ..... –50°C to 125°C
Input Voltage ..... V_{SS}–0.3V to V_{DD}+0.3V
Operating Temperature..... –40°C to 85°C

Note: These are stress ratings only. Stresses exceeding the range specified under "Absolute Maximum Ratings" may cause substantial damage to 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	—	f_{SYS}=6MHz	3.3	—	5.5	V
			f_{SYS}=12MHz	4.5	—	5.5	V
V_{DDIO}	Operating Voltage for I/O Ports	—	—	3.3	—	V_{DD}	V
I_{DD1}	Operating Current (6MHz Crystal)	5V	No load, f_{SYS}=6MHz	—	4	8	mA
I_{DD2}	Operating Current (12MHz Crystal and HIRC)	5V	No load, f_{SYS}=12MHz	—	7.5	16	mA
I_{STB1}	Standby Current 1	5V	No load, system HALT, USB mode, LVR disable, WDT disable, Clr V33O [USC.4], Clr PLL [USC.5], Clr SELPS2 [USC.6], Clr USBCKEN [UCC.3], Set SUSP2 [UCC.4], Set RCtrl [UCC.7], Set CLK_adj [PSD.4]	—	—	350	μA
I_{STB2}	Standby Current 2	5V	No load, system HALT, PS2 (I/O) mode, LVR disable, WDT disable, Clr V33O [USC.4], Clr PLL [USC.5], Set SELPS2 [USC.6], Clr USBCKEN [UCC.3], Set SUSP2 [UCC.4], Clr RCtrl [UCC.7], CLR CLK_adj [PSD.4]	—	—	10	μA
V_{IL1}	Input Low Voltage for I/O Ports	—	—	0	—	$0.3V_{DDIO}$	V
V_{IH1}	Input High Voltage for I/O Ports	—	—	$0.7V_{DDIO}$	—	V_{DDIO}	V
V_{IL2}	Input Low Voltage ($\overline{RES}$)	—	—	0	—	$0.4V_{DD}$	V
V_{IH2}	Input High Voltage ($\overline{RES}$)	—	—	$0.9V_{DD}$	—	V_{DD}	V
I_{OL1}	I/O Port Sink Current for PA, PB, PC, PD, PE	—	V_{OL}=$0.1V_{DDIO}$, V_{DDIO}=3.3V	2	5	—	mA
I_{OL2}	I/O Port Sink Current for PF0, PF1	—	V_{OL}=$0.1V_{DD}$, V_{DD}=5V	2	5	—	mA
I_{OH1}	I/O Port Source Current for PA, PB, PC, PD, PE	—	V_{OH}=$0.9V_{DDIO}$, V_{DDIO}=3.3V	–2	–5	—	mA
I_{OH2}	I/O Port Source Current for PF0, PF1	5V	V_{OH}=$0.9V_{DD}$	–2	–5	—	mA
R_{PH1}	Pull-high Resistance for PA, PB, PC, PD, PE	—	V_{DDIO}=3.3V	40	60	80	kΩ
R_{PH2}	Pull-high Resistance for PF0, PF1	5V	—	10	30	50	kΩ
V_{LVR}	Low Voltage Reset Voltage	—	—	2.0	2.2	2.4	V

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{LCD_BIAS}	V_{DDIO}/2 Bias Current for LCD, V_{DDIO}=5V	5V	LCDC. RSEL[1:0]=00	17.5	25.0	32.5	μA
			LCDC. RSEL[1:0]=01	35	50	65	μA
			LCDC. RSEL[1:0]=10	70	100	130	μA
			LCDC. RSEL[1:0]=11	140	200	260	μA
V_{COM}	V_{DDIO}/2 Voltage for LCD COM Port, V_{DDIO}=5V	5V	No load	0.475	0.500	0.525	V_{DDIO}

A.C. Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
f_{SYS1}	System Clock (Crystal OSC)	5V	—	6000	—	12000	kHz
f_{SYS2}	System Clock (HIRC OSC)	4.5V~5.5V	—	11640	12000	12360	kHz
f_{TIMER}	Timer I/P Frequency (TMR0/TMR1)	5V	—	0	—	f_{SYS}	kHz
t_{WDTOSC}	Watchdog Oscillator Period	5V	—	32	65	130	μs
t_{RES}	External Reset Low Pulse Width	—	—	1	—	—	μs
t_{SST}	System Start-up Timer Period	—	Power-up, reset or wake-up from HALT	—	1024	—	*t_{SYS}
t_{INT}	Interrupt Pulse Width	—	—	1	—	—	μs
f_{PCLK}	Parallel Frequency	5V	—	2	—	5	MHz

Note: *t_{SYS}=1/f_{SYS}
*f_{T1}=f_{SYS}/4

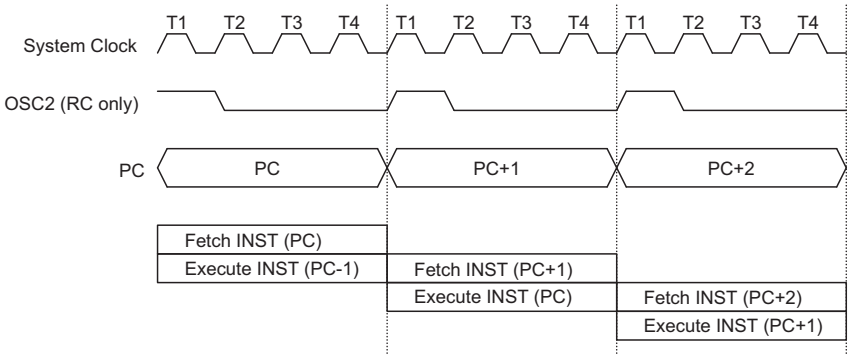
System Architecture

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

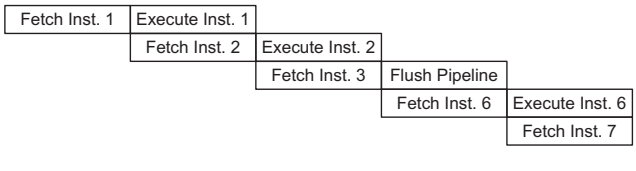
For instructions involving branches, such as jump or call instructions, two machine cycles are required to complete instruction execution. An extra cycle is required as the program takes one cycle to first obtain the actual jump or call address and then another cycle to actually execute the branch. The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



System Clocking and Pipelining

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

DELAY:
```



Instruction Fetching

Program Counter

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

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

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

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

Mode	Program Counter											
	*11	*10	*9	*8	*7	*6	*5	*4	*3	*2	*1	*0
Initial Reset	0	0	0	0	0	0	0	0	0	0	0	0
External Interrupt	0	0	0	0	0	0	0	0	0	1	0	0
Timer/Event Counter 0 Overflow	0	0	0	0	0	0	0	0	1	0	0	0
Timer/Event Counter 1 Overflow	0	0	0	0	0	0	0	0	1	1	0	0
USB Interrupt	0	0	0	0	0	0	0	1	0	0	0	0
Serial Interface 1 Interrupt	0	0	0	0	0	0	0	1	0	1	0	0
Serial Interface 2 Interrupt	0	0	0	0	0	0	0	1	1	0	0	0
Skip	Program Counter+2											
Loading PCL	*11	*10	*9	*8	@7	@6	@5	@4	@3	@2	@1	@0
Jump, Call Branch	#11	#10	#9	#8	#7	#6	#5	#4	#3	#2	#1	#0
Return from Subroutine	S11	S10	S9	S8	S7	S6	S5	S4	S3	S2	S1	S0

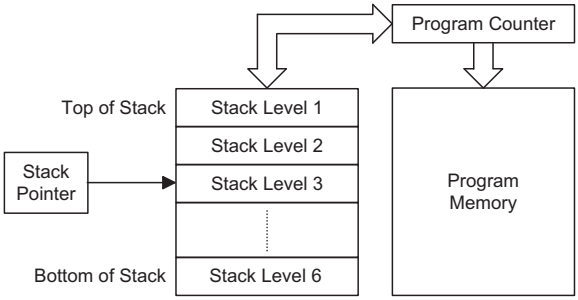
Program Counter

Note: PC11~PC8: Current Program Counter bits
#11~#0: Instruction code address bits
S11~S0: Stack register bits
@7~@0: PCL bits

Stack

This is a special part of the memory which is used to save the contents of the Program Counter only. The stack has 6 levels and is neither part of the data nor part of the program space, and can neither be read from nor written to. The activated level is indexed by the Stack Pointer, SP, which can also neither be read from nor written to. 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 contains 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. OTP devices are also applicable for use in applications that require low or medium volume production runs.

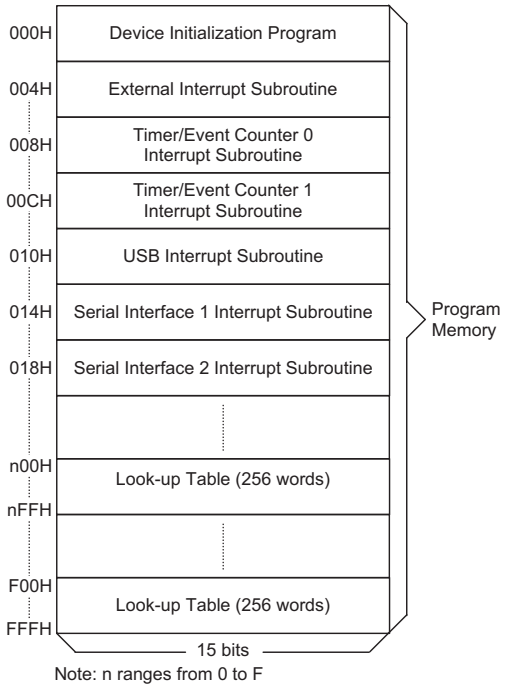
Organisation

The Program Memory has a capacity of 4K by 15 bits. The Program Memory is addressed by the Program Counter and also contains data, table information and interrupt entries. Table data, which can be setup in any location within the Program Memory, is addressed by separate table pointer registers.

Special Vectors

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

- Location 000H
  Location 000H is reserved for program initialization.
  After a chip reset, the program always begins execution at this location.
- Location 004H
  Location 004H is reserved for the external interrupt service program. If the $\overline{INT}$ input pin is activated, and the interrupt is enabled, and the stack is not full, the program begins execution at location 004H.
- Location 008H
  Location 008H is reserved for the Timer/Event Counter 0 interrupt service program. If a timer interrupt results from a Timer/Event Counter 0 overflow, and if the interrupt is enabled and the stack is not full, the program begins execution at location 008H.
- Location 00CH
  Location 00CH is reserved for the Timer/Event Counter 1 interrupt service program. If a timer interrupt results from a Timer/Event Counter 1 overflow, and if the interrupt is enabled and the stack is not full, the program begins execution at location 00CH.
- Location 010H
  Location 010H is reserved for the USB interrupt service program. If the USB interrupt is activated, the interrupt is enabled and the stack is not full, the program begins execution at location 010H.
- Location 014H
  Location 014H is reserved for the Serial Interface 1. When 8 bits data have been received or transmitted successfully from this Serial Interface 1, the related interrupts are enabled, and the stack is not full, the program begins execution at location 014H.



Program Memory

- Location 018H
  Location 018H is reserved for the Serial Interface 2. When 8 bits data have been received or transmitted successfully from Serial Interface 2, the related interrupts are enabled, and the stack is not full, the program begins execution at location 018H.

- Table location
  Any location in the program memory can be used as a look-up tables. There are three methods to read the ROM data using two table read instructions "TABRDC" and "TABRDL", transfer the contents of the lower-order byte to the specified data memory, and the higher-order byte to TBLH (08H).

  The three methods are shown as follows:
  - The instruction "TABRDC [m]" (the current page, one page=256 words), where the table location is defined by TBLP (07H) in the current page. The configuration option, TBHP, is disabled (default).
  - The instructions "TABRDC [m]", where the table location is defined by registers TBLP (07H) and TBHP (01FH). The configuration option, TBHP, is enabled.
  - The instruction "TABRDL [m]", where the table locations is defined by registers TBLP (07H) in the last page (0F00H~0FFFH).

Instruction	Table Location											
	*11	*10	*9	*8	*7	*6	*5	*4	*3	*2	*1	*0
TABRDC [m]	P11	P10	P9	P8	@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: *11~*0: Table location bits P11~P8: Current program counter bits
@7~@0: Table pointer bits

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 pointer (TBLP, TBHP) is a read/write register (07H, 1FH), which indicates the table location. Before accessing the table, the location must be placed in the TBLP and TBHP registers(If the configuration option TBHP 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 TBHP is enabled, the instruction "TABRDC [m]" reads the ROM data as defined by TBLP and TBHP register value. Otherwise, if the configuration option TBHP is disabled, the instruction "TABRDC [m]" reads the ROM data as defined by TBLP and the current program counter bits.

Data Memory – RAM

The data memory (RAM) is designed with 256×8 bits, and is divided into two functional groups, namely; special function registers (64×8 bits) and general purpose data memory (192×8 bits) most of which are readable/writeable, although some are read only.

The unused space before 40H is reserved for future expanded usage and reading these locations will get ″00H″. The general purpose data memory, addressed from 40H to FFH, is used for data and control information under instruction commands. All of the data memory areas can handle arithmetic, logic, increment, decrement and rotate operations directly. Except for some dedicated bits, each bit in the data memory can be set and reset by ″SET [m].i″ and ″CLR [m].i″. They are also indirectly accessible through memory pointer registers (MP0;01H/MP1;03H).

Indirect Addressing Register

Location 00H and 02H are indirect addressing registers that are not physically implemented. Any read/write operation of [00H] and [02H] accesses the RAM pointed to by MP0 (01H) and MP1 (03H) respectively. Reading location 00H or 02H indirectly returns the result 00H. While, writing into it, indirectly leads to no operation.

The function of data movement between two indirect addressing registers is not supported. The memory pointer registers, MP0 and MP1, are both 8-bit registers used to access the RAM by combining corresponding indirect addressing registers.

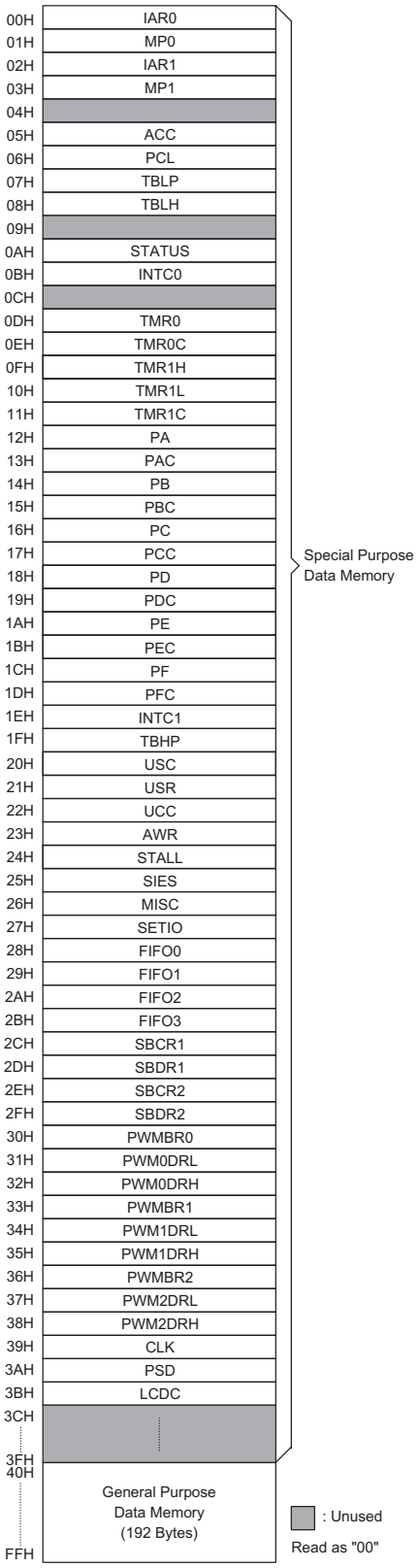
Accumulator – ACC

The accumulator is closely related to ALU operations. It is also mapped to location 05H of the RAM and capable of operating with immediate data. The data movement between two data memory locations must pass through the accumulator.

Status Register – STATUS

The status register (0AH) is 8 bits wide and contains, a carry flag (C), an auxiliary carry flag (AC), a zero flag (Z), an overflow flag (OV), a power down flag (PDF), and a Watchdog time-out flag (TO). It also records the status information and controls the operation sequence. Except for the TO and PDF flags, bits in the status register can be altered by instructions similar to other registers. Data written into the status register does not alter the TO or PDF flags. Operations related to the status register, however, may yield different results from those intended. The TO and PDF flags can only be changed by a Watchdog Timer overflow, chip power-up, or clearing the Watchdog Timer and executing the ″HALT″ instruction.

The Z, OV, AC, and C flags reflect the status of the latest operations. On entering the interrupt sequence or executing a subroutine call, the status register will not be automatically pushed onto the stack. If the contents of the status is important, and if the subroutine is likely to corrupt the status register, the programmer should take precautions and save it properly.


00H IAR0
01H MP0
02H IAR1
03H MP1
04H
05H ACC
06H PCL
07H TBLP
08H TBLH
09H
0AH STATUS
0BH INTC0
0CH
0DH TMR0
0EH TMR0C
0FH TMR1H
10H TMR1L
11H TMR1C
12H PA
13H PAC
14H PB
15H PBC
16H PC
17H PCC
18H PD
19H PDC
1AH PE
1BH PEC
1CH PF
1DH PFC
1EH INTC1
1FH TBHP
20H USC
21H USR
22H UCC
23H AWR
24H STALL
25H SIES
26H MISC
27H SETIO
28H FIFO0
29H FIFO1
2AH FIFO2
2BH FIFO3
2CH SBCR1
2DH SBDR1
2EH SBCR2
2FH SBDR2
30H PWMBR0
31H PWM0DRL
32H PWM0DRH
33H PWMBR1
34H PWM1DRL
35H PWM1DRH
36H PWMBR2
37H PWM2DRL
38H PWM2DRH
39H CLK
3AH PSD
3BH LCDC
3CH
3FH
Special Purpose Data Memory
40H
FFH
General Purpose Data Memory (192 Bytes)
: Unused
Read as "00"

RAM Mapping

Bit No.	Label	Function
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.
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.
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
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.
4	PDF	Power down flag 0: After power up or executing the ″CLR WDT″ instruction 1: By executing the ″HALT″ instruction
5	TO	Watchdog Time-Out flag 0: After power up or executing the ″CLR WDT″ or ″HALT″ instruction 1: A watchdog time-out occurred.
6, 7	—	Unused bit, read as ″0″

Status (0AH) Register

Interrupts

This device provides external interrupts ($\overline{INT}$ pin interrupt), USB interrupt, Serial Interface interrupt and internal timer/event counter interrupts. The Interrupt Control Register0 (INTC0;0BH) and interrupt control register1 (INTC1:1EH) both contain the interrupt control bits that are used to set the enable/disable status and interrupt request flags.

Once an interrupt subroutine is serviced, all the other interrupts will be blocked (by clearing the EMI bit). This scheme may prevent any further interrupt nesting. Other interrupt requests may occur during this interval but only the interrupt request flag is recorded. If a certain interrupt requires servicing within the service routine, the EMI bit and the corresponding bit of the INTC0 or INTC1 may be set 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.

All these kinds of interrupts have a wake-up capability. As an interrupt is serviced, a control transfer occurs by pushing the program counter onto the stack, followed by a branch to a subroutine at specified location in the program memory. Only the program counter is pushed onto the stack. If the contents of the register or status register (STATUS) are altered by the interrupt service program which corrupts the desired control sequence, the contents should be saved in advance.

External interrupts can are triggered by a falling edge transition of $\overline{INT}$), and the related interrupt request flag (EIF; bit4 of the INTC0) is set as well. After the interrupt is enabled, the stack is not full, and the external interrupt is active ($\overline{INT}$ pin), a subroutine call at location 04H occurs. The interrupt flag (EIF) and EMI bits are all cleared to disable other maskable interrupts.

The internal Timer/Event Counter 0 interrupt is initialized by setting the Timer/Event Counter 0 interrupt request flag (bit 5 of the INTC0), caused by a Timer 0 overflow. When the interrupt is enabled, the stack is not full and the T0F bit is set, a subroutine call to location 08H will occur. The related interrupt request flag (T0F) will be reset and the EMI bit cleared to disable further interrupts.

The internal Timer/Event Counter 1 interrupt is initialized by setting the Timer/Event Counter 1 interrupt request flag (bit 6 of the INTC0), caused by a Timer 1 overflow. When the interrupt is enabled, the stack is not full and the T1F is set, a subroutine call to location 0CH will occur. The related interrupt request flag (T1F) will be reset and the EMI bit cleared to disable further interrupts.

USB interrupts are triggered by the following USB events and the related interrupt request flag (USBF; bit 4 of the INTC1) will be set.

- The access of the corresponding USB FIFO from PC
- The USB suspend signal from the PC
- The USB resume signal from the PC
- USB reset signal

Bit No.	Label	Function
0	EMI	Global interrupt control (1: enable; 0: disable)
1	EEI	External interrupt control (1: enable; 0: disable)
2	ET0I	Timer/Event Counter 0 interrupt control (1: enable; 0: disable)
3	ET1I	Timer/Event Counter 1 interrupt control (1: enable; 0: disable)
4	EIF	External interrupt request flag (1: active; 0: inactive)
5	T0F	Timer/Event Counter 0 interrupt request flag (1: active; 0: inactive)
6	T1F	Timer/Event Counter 1 interrupt request flag (1: active; 0: inactive)
7	—	Unused bit, read as ″0″

INTC0 (0BH) Register

Bit No.	Label	Function
0	EUI	USB interrupt control (1: enable; 0: disable)
1	ES1II	Serial Interface 1 interrupt control (1: enable; 0: disable)
2	ES2II	Serial Interface 2 interrupt control (1: enable; 0: disable)
3	—	Unused bit, read as ″0″
4	USBF	USB interrupt request flag (1: active; 0: inactive)
5	SI1F	Serial interface 1 interrupt request flag (1: active; 0: inactive)
6	SI2F	Serial Interface 2 interrupt request flag (1: active; 0: inactive)
7	—	Unused bit, read as ″0″

INTC1 (1EH) Register

When the interrupt is enabled, the stack is not full and the external interrupt is active, a subroutine call to location 10H will occur. The interrupt request flag (USBF) and EMI bits will be cleared to disable other interrupts.

When PC Host access the FIFO of the HT82A525R, the corresponding request bit of USR is set, and a USB interrupt is triggered when the corresponding interrupt is enabled. So user can easily determine which FIFO is accessed. When the interrupt has been served, the corresponding bit should be cleared by firmware. When the HT82A525R receives a USB Suspend signal from the Host PC, the suspend line (bit0 of the USC) of the HT82A525R is set and a USB interrupt is also triggered.

Also when the HT82A525R receives a Resume signal from the Host PC, the resume line (bit3 of the USC) of the HT82A525R is set and a USB interrupt is triggered.

Whenever a USB reset signal is detected, a USB interrupt is triggered.

The serial interface interrupt is indicating by the interrupt flag (SI1F: bit 5 of INTC1 or SI2F: bit 6 of INTC1), that is caused by received or transferred a complete 8-bit data between HT82A525R and external device. The serial interface interrupt is controlled by setting the Serial interface interrupt control bit (ES1II: bit 1 of INTC1 or ES2II: bit2 of INTC1). After the interrupt is enabled (by setting SBEN; bit 4 of SBCR1 or SBCR2), and the stack is not full and the SIF is set, a subroutine call to location 14H or 18H occurs.

During the execution of an interrupt subroutine, other interrupt acknowledge signals are held until the ″RETI″ instruction is executed or the EMI bit and the related interrupt control bit are set to 1 (if the stack is not full). To return from the interrupt subroutine, ″RET″ or ″RETI″ may be invoked. RETI will set the EMI bit to enable an interrupt service, but RET will not.

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 the 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
USB Interrupt	4	10H
Serial Interface 1 Interrupt	5	14H
Serial Interface 2 Interrupt	6	18H

It is recommended that a program does not use the ″CALL subroutine″ 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 enabling the interrupt is not well controlled, the original control sequence will be damaged once the ″CALL″ operates in the interrupt subroutine.

Oscillator Configuration

Various oscillator options offer the user a wide range of functions according to their various application requirements. Two types of system clocks and an internal 12kHz oscillator can be selected while various clock source options for the Watchdog Timer are provided for maximum flexibility. All oscillator options are selected through the configuration options.

The two methods of generating the system clock are:

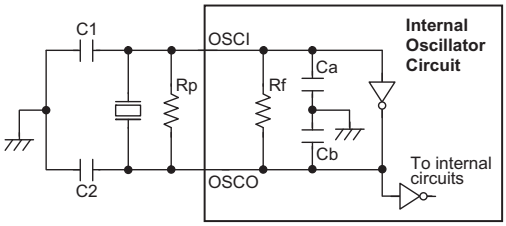
- External crystal oscillator
- Internal RC oscillator

One of these two methods must be selected using the configuration options.

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

External Crystal Oscillator – HXT

The simple connection of a crystal across OSCI and OSCO will create the necessary phase shift and feedback for oscillation, and will normally not require external capacitors. However, for some crystals, to ensure oscillation and accurate frequency generation, it may be necessary to add two small value external capacitors, C1 and C2. The exact values of C1 and C2 should be selected in consultation with the crystal manufacturer's specification. The external parallel feedback resistor, R_P, is normally not required but in some cases may be needed to assist with oscillation start up.



Note: 1. Rp is normally not required.
2. Although not shown OSCI/OSCO pins have a parasitic capacitance of around 7pF.

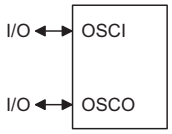
Crystal Oscillator – HXT

Internal Ca, Cb, therefore Typical Values @ 5V, 25°C		
Ca	**Cb**	**Rf**
11~13pF	13~15pF	800kΩ

Oscillator Internal Component Values

Internal RC Oscillator

The internal RC oscillator is a fully integrated system oscillator requiring no external components. The internal RC oscillator has a fixed frequency of 12MHz. Device trimming during the manufacturing process and the inclusion of internal frequency compensation circuits are used to ensure that the influence of the power supply voltage, temperature and process variations on the oscillation frequency are minimised. As a result, at a power supply of 5V and at a temperature of 25°C degrees, the fixed oscillation frequency of 12MHz will have a tolerance within 3%. The HIRC has an Automatic Clock adjust function which can be used to adjust the HIRC frequency and to minimise the frequency deviation caused by temperature variation. This function is controlled by the CLK_adj and CLR_RCP bits in the PSD register. Note that if this internal system clock option is selected, as it requires no external pins for its operation, I/O pins PF0 and PF1 are free for use as normal I/O pins.



Internal RC Oscillator – HIRC

Bit No.	Label	R/W	Function
0	PS2_DAO	R/W	USBDM/DATA pin output control bit 0: output ″low″; 1: output ″high″ This control bit is used to generate the USBDM/DATA pin output signal in the 3D PS2 mouse mode. The default value is ″1″.
1	PS2_CKO	R/W	USBDP/CLK pin output control bit 0: output ″low″; 1: output ″high″ This control bit is used to generate the USBDP/CLK pin output signal in the 3D PS2 mouse mode. The default value is ″1″.
2	—	—	Undefined bit, read as unknown
3	PU	R/W	USBDP and USBDM internal 310kΩ pull-high resistor select bit 0: without pull up resistor; 1: with pull up resistor
4	CLK_adj	R/W	Automatic Clock adjustment control bit 0: disable (default); 1:enable This bit is used to adjust the HIRC mode system clock, to reduce the frequency deviation due to temperature issue. In the Power-down mode, this bit should be clear to reduce power consumption.
5	CLR_RCP	R/W	This bit is must enable and then disable by F/W to clear HIRC initial parameters after power on. 0: disable (default); 1:enable
6~7	—	—	Undefined bit, read as ″0″

PSD (3AH) Register

Internal 12kHz Oscillator – WDTOSC

The low frequency internal 12K RC oscillator (WDTOSC) is a free running on-chip RC oscillator, and no external components are required. Although when the system enters the Halt mode, the system clock stops, the WDT oscillator will still operate if this oscillator is selected. Its period is approximately 83.3μs. The WDT oscillator can also be disabled to conserve power.

Watchdog Timer – WDT

The WDT clock source is implemented by a dedicated RC oscillator (WDTOSC) or the system clock divided by 4 determined by a configuration option. The timer is designed to prevent a software malfunction or sequence from jumping to an unknown location with unpredictable results. The watchdog timer can be disabled by a configuration option. If the watchdog timer is disabled, all executions related to the WDT results in no operation.

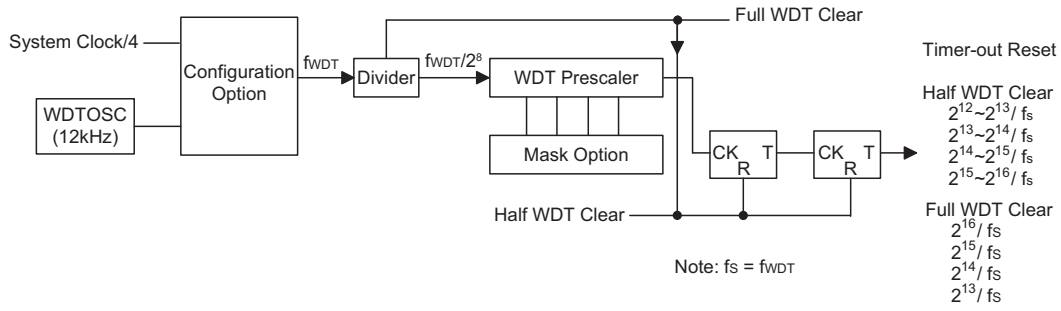
Once the internal WDT oscillator (RC oscillator with a period of 83.3μs , normally at 5V) is selected, it is divided by 2^{12}~2^{16} by configuration option to get the WDT time-out period. The WDT time-out minimum period is about 340ms. This time-out period may vary with temperature, VDD and process variations. By selection from the WDT option, longer time-out periods can be realized. If the WDT time-out is selected as 2^{15}, the maximum time-out period is divided by 2^{15}, about 2.7s.

If the WDT oscillator is disabled, the WDT clock may still come from the instruction clock and operates in the same manner except that in the HALT state the WDT may stop counting and lose its protecting purpose. If the device operates in a noisy environment, using the on-chip RC oscillator (WDTOSC) is strongly recommended, since the HALT instruction will stop the system clock.

The WDT overflow under normal operation will initialize a ″chip reset″ and set the status bit ″TO″. Whereas in the HALT mode, the overflow will initialize a ″warm reset″ and only the Program Counter and Stack Pointer are reset to zero. To clear the contents of WDT, three methods are adopted; external reset (a

low level to $\overline{RES}$), software instructions, or a "HALT" instruction. The software instructions include "CLR WDT" and the other set -- "CLR WDT1" and "CLR WDT2". Of these two types of instruction, only one can be active depending on the option -- "CLR WDT" times selection option. If the "CLR WDT" is selected (i.e. CLR WDT times equal one), any execution of the "CLR WDT" instruction will clear the WDT. In case "CLR WDT1" and "CLR WDT2" are chosen (i.e. "CLRWDT" times equal two), these two instructions must be executed to clear the WDT, otherwise, the WDT may reset the chip due to time-out.

There are two "Timer out WDT clear" modes selected by configuration option. One is "Half WDT clear" and the other is "Full WDT scale". The "Half WDT clear" is only to clear the highest two bits of WDT counter and the "Full WDT clear" is to clear all bits of WDT counter.



Watchdog Timer

Power Down Operation – HALT

The HALT mode is initialized by the "HALT" instruction and results in the following:

- The system oscillator is turned off but the WDT oscillator keeps running (if the WDT oscillator is selected).
- The contents of the on-chip RAM and registers remain unchanged.
- The WDT will be cleared and start recounting (if the WDT clock source is from the WDT oscillator).
- All of the I/O ports maintain their original status.
- The PDF flag is set and the TO flag is cleared.

The system can quit the HALT mode in many ways, by an external reset, an interrupt (except serial interface interrupt and serial interface 2 interrupt), an external falling edge signal on I/O ports or a WDT overflow. An external reset causes a device initialization and the WDT overflow performs a "warm reset". After examining the TO and PDF flags, the cause for a chip reset can be determined. The PDF flag is cleared by a system power-up or by executing the "CLR WDT" instruction and is set when executing the "HALT" instruction. On the other hand, the TO flag is set if the WDT time-out occurs, and causes a wake-up that only resets the Program Counter and Stack Pointer; and leaves the others in their original status.

The Port A wake-up and interrupt methods can be considered as a continuation of normal execution. Each bit in Port A can be independently selected to wake-up the device by option. Awakening from an I/O port stimulus, the program will resume execution of the next instruction. If it awakens from an interrupt, two sequences may occur. If the related interrupt is disabled or the interrupt is enabled but the stack is full, the program will resume execution at the next instruction. But if the interrupt is enabled and the stack is not full, a regular interrupt response takes place. When an interrupt request flag is set to "1" before entering the HALT mode, the wake-up function of the related interrupt will be disabled. If a wake-up event occurs, it takes 1024 f_{SYS} (system clock period) to resume normal operation. In other words, a dummy period is inserted after wake-up. If the wake-up results from an interrupt acknowledge, the actual interrupt subroutine execution is delayed by more than one cycle.

However, if the wake-up results in the next instruction execution, this will be executed immediately after the dummy period is finished.

To minimize power consumption, all the I/O pins should be carefully managed before entering the HALT status.

Reset

There are three ways in which a reset may occur:

- $\overline{\text{RES}}$ reset during normal operation
- $\overline{\text{RES}}$ reset during HALT
- WDT time-out reset during normal operation

The WDT time-out during HALT differs from other chip reset conditions, for it can perform a ″warm reset″ that resets only the program counter and stack pointer, leaving the other circuits in their original state. Some registers remain unaffected during any other reset conditions. Most registers are reset to the ″initial condition″ when the reset conditions are met. Examining the PDF and TO flags, the program can distinguish between different ″chip resets″.

TO	PDF	RESET Conditions
0	0	$\overline{\text{RES}}$ reset during power-up
0	0	$\overline{\text{RES}}$ reset during normal operation
0	0	$\overline{\text{RES}}$ wake-up at HALT
1	u	WDT time-out during normal operation
1	1	WDT wake-up at HALT

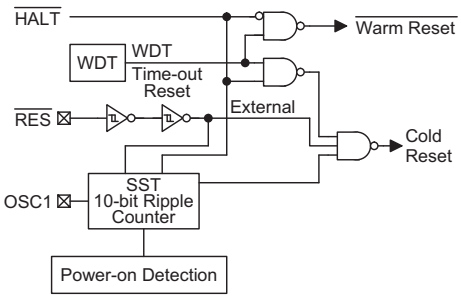
Note: ″u″ stands for ″unchanged″

To guarantee that the system oscillator is started and stabilized, the SST (System Start-up Timer) provides an extra-delay of 1024 system clock pulses when the system awakes from the HALT state or during power up.

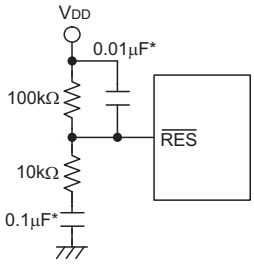
Awaking from the HALT state or system power up, an SST delay is added. An extra SST delay is added during power up period, and any wake-up from HALT may enable only the SST delay.

The functional unit chip reset status are shown below.

Program Counter	000H
Interrupt	Disable
Prescaler, Divider	Cleared
WDT	Clear. After master reset, WDT begins counting
Timer/event Counter	Off
Input/output Ports	Input mode
Stack Pointer	Points to the top of the stack

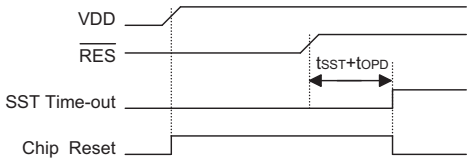


Reset Configuration



Reset Circuit

Note: "*" Make the length of the wiring, which is connected to the RES pin as short as possible, to avoid noise interference.



Reset Timing Chart

The registers states are summarized in the following table.

Register	Reset (Power On)	WDT Time-out (Normal Operation)	RES Reset (Normal Operation)	RES Reset (HALT)	WDT Time-out (HALT)*	USB Reset (Normal)	USB Reset (HALT)
MP0	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
MP1	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
Program Counter	000H	000H	000H	000H	000H	000H	000H
TBLP	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLH	-xxx xxxx	-uuu uuuu	-uuu uuuu	-uuu uuuu	-uuu uuuu	-uuu uuuu	-uuu uuuu
STATUS	--00 xxxx	--1u uuuu	--00 uuuu	--00 uuuu	--11 uuuu	--uu uuuu	--01 uuuu
INTC0	-000 0000	-000 0000	-000 0000	-000 0000	-uuu uuuu	-000 0000	-000 0000
TMR0	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
TMR0C	00-0 1000	00-0 1000	00-0 1000	00-0 1000	uu-u u---	00-0 1000	00-0 1000
TMR1H	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
TMR1L	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
TMR1C	00-0 1---	00-0 1---	00-0 1---	00-0 1---	uu-u u---	00-0 1---	00-0 1---
PA	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu	1111 1111	1111 1111
PAC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu	1111 1111	1111 1111
PB	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu	1111 1111	1111 1111
PBC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu	1111 1111	1111 1111
PC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu	1111 1111	1111 1111
PCC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu	1111 1111	1111 1111
PD	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu	1111 1111	1111 1111

Register	Reset (Power On)	WDT Time-out (Normal Operation)	$\overline{RES}$ Reset (Normal Operation)	$\overline{RES}$ Reset (HALT)	WDT Time-out (HALT)*	USB Reset (Normal)	USB Reset (HALT)
PDC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu	1111 1111	1111 1111
PE	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu	1111 1111	1111 1111
PEC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu	1111 1111	1111 1111
PF	---- --11	---- --11	---- --11	---- --11	---- --uu	---- --11	---- --11
PFC	---- --11	---- --11	---- --11	---- --11	---- --uu	---- --11	---- --11
INTC1	-000 -000	-000 -000	-000 -000	-000 -000	-uuu -uuu	-000 -000	-000 -000
TBHP	---- xxxx	---- uuuu	---- uuuu	---- uuuu	---- uuuu	---- uuuu	---- uuuu
USC	1000 x00x	uuuu xuux	1000 x00x	1000 x00x	uuuu xuux	u-00 0100	u-00 0100
USR	0000 0000	0000 uuuu	0000 0000	0000 0000	0000 uuuu	0000 0000	0000 0000
UCC	-000 0000	-uuu uuuu	-000 0000	-000 0000	-uuu uuuu	-uu0 u000	-uu0 u000
AWR	0000 0000	uuuu uuuu	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
STALL	---- 1110	---- uuuu	---- 1110	---- 1110	---- uuuu	---- 1110	---- 1110
SIES	0x0x x000	uxux xxuu	0x0x x000	0x0x x000	uxux xxuu	0x0x x000	0x0x x000
MISC	0xx- -000	uxx- -uuu	0xx- -000	0xx- -000	uxx- -uuu	000- -000	000- -000
SETIO	---- 1110	---- uuuu	---- 1110	---- 1110	---- uuuu	---- 1110	---- 1110
FIFO0	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	0000 0000	0000 0000
FIFO1	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	0000 0000	0000 0000
FIFO2	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	0000 0000	0000 0000
FIFO3	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	0000 0000	0000 0000
SBCR1	0110 0000	0110 0000	0110 0000	0110 0000	uuuu uuuu	uuuu uuuu	uuuu uuuu
SBDR1	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
SBCR2	0110 0000	0110 0000	0110 0000	0110 0000	uuuu uuuu	uuuu uuuu	uuuu uuuu
SBDR2	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
PWMBR0	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
PWM0DRL	0000 --00	0000 --00	0000 --00	0000 --00	uuuu --uu	0000 --00	0000 --00
PWM0DRH	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
PWMBR1	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
PWM1DRL	0000 --00	0000 --00	0000 --00	0000 --00	uuuu --uu	0000 --00	0000 --00
PWM1DRH	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
PWMBR2	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
PWM2DRL	0000 --00	0000 --00	0000 --00	0000 --00	uuuu --uu	0000 --00	0000 --00
PWM2DRH	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
CLK	---00 0000	--uu uuuu	---00 0000	---00 0000	--uu uuuu	--uu uuuu	--uu uuuu
PSD	--00 0x11	--uu ux11	--00 0x11	--00 0x11	--uu ux11	--uu ux11	--uu ux11
LCDC	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	uuuu uuuu	uuuu uuuu

Note: "*" stands for warm reset
"u" stands for unchanged
"x" stands for unknown

Timer/Event Counter

Two Timer/Event Counters (TMR0, TMR1) are implemented in the microcontroller. The Timer/Event Counter 0 contains a 8-bit programmable count-up counter and the clock may come from an external source or an internal clock source. An internal clock source comes from f_{SYS}. The Timer/Event Counter 1 contains a 16-bit programmable count-up counter and the clock may come from an external source or an internal clock source. An internal clock source comes from $f_{SYS}/4$. The external clock input allows the user to count external events, measure time intervals or pulse widths, or generate an accurate time base.

There are five registers related to the Timer/Event Counter 0; TMR0 (0DH), TMR0C (0EH) and the Timer/Event Counter 1; TMR1H (0FH), TMR1L (10H), TMR1C (11H). For 16bits timer to Write data to TMR1L will only put the written data to an internal lower-order byte buffer (8-bit) and writing TMR1H will transfer the specified data and the contents of the lower-order byte buffer to TMR1H and TMR1L registers. The Timer/Event Counter 1 preload register is changed by each writing TMR1H operations. Reading TMR1H will latch the contents of TMR1H and TMR1L counters to the destination and the lower-order byte buffer, respectively. Reading the TMR1L will read the contents of the lower-order byte buffer. The TMR0C (TMR1C) is the Timer/Event Counter 0 (1) control register, which defines the operating mode, counting enable or disable and an active edge.

The T0M0, T0M1 (TMR0C) and T1M0, T1M1 (TMR1C) bits define the operation mode. The event count mode is used to count external events, which means that the clock source is from an external (TMR0, TMR1) pin. The timer mode functions as a normal timer with the clock source coming from the internal selected clock source. Finally, the pulse width measurement mode can be used to count the high or low level duration of the external signal (TMR0, TMR1), and the counting is based on the internal selected clock source.

In the event count or timer mode, the timer/event counter starts counting at the current contents in the timer/event counter and ends at FFFFH (for 16 bits timer is FFFFH, bit 8 bits timer will be FFH). Once an overflow occurs, the counter is reloaded from the timer/event counter preload register, and generates an interrupt request flag (T0F; bit 5 of the INTC0, T1F; bit 6 of the INTC0).

In the pulse width measurement mode with the values of the T0ON/T1ON and T0E/T1E bits equal to 1, after the TMR0 (TMR1) has received a transient from low to high (or high to low if the T0E/T1E bit is ″0″), it will start counting until the TMR0 (TMR1) returns to the original level and resets the T0ON/T1ON. The measured result remains in the timer/event counter even if the activated transient occurs again. In other words, only 1-cycle measurement can be made until the T0ON/T1ON is set. The cycle measurement will re-function as long as it receives further transient pulse. In this operation mode, the timer/event counter begins counting not according to the logic level but to the transient edges. In the case of counter overflows, the counter is reloaded from the timer/event counter register and issues an interrupt request, as in the other two modes, i.e., event and timer modes.

To enable the counting operation, the Timer ON bit (T0ON: bit 4 of the TMR0C; T10N: bit 4 of the TMR1C) should be set to 1. In the pulse width measurement mode, the T0ON/T1ON is automatically cleared after the measurement cycle is completed. But in the other two modes, the T0ON/T1ON can only be reset by instructions. The overflow of the Timer/Event Counter 0/1 is one of the wake-up sources. No matter what the operation mode is, writing a 0 to ET0I or ET1I disables the related interrupt service.

In the case of timer/event counter OFF condition, writing data to the timer/event counter preload register also reloads that data to the timer/event counter. But if the timer/event counter is turned on, data written to the timer/event counter is kept only in the timer/event counter preload register. The timer/event counter still continues its operation until an overflow occurs.

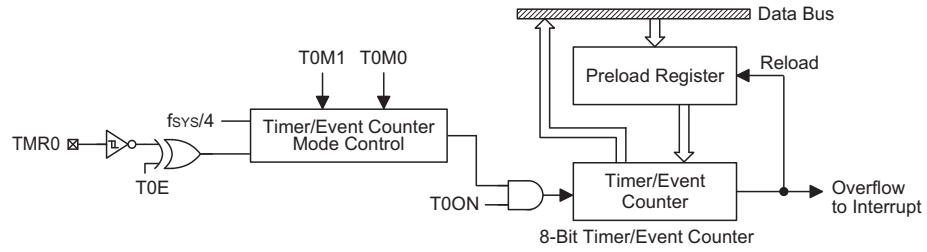
When the timer/event counter (reading TMR0/TMR1) is read, the clock is blocked to avoid errors, as this may results in a counting error. Blocking of the clock should be taken into account by the programmer. It is strongly recommended to load a desired value into the TMR0/TMR1 register first, before turning on the related timer/event counter, for proper operation since the initial value of TMR0/TMR1 is unknown. Due to the timer/event scheme, the programmer should pay special attention on the instruction to enable then disable the timer for the first time, whenever there is a need to use the timer/event function, to avoid unpredictable result. After this procedure, the timer/event function can be operated normally.

Bit No.	Label	Function
0~2	—	Unused bit, read as ″0″
3	T0E	Timer 0 active edge control bit 0: active on low to high 1: active on high to low
4	T0ON	Timer 0 control bit 0: disable 1: enable
5	—	Unused bit, read as ″0″
6 7	T0M0 T0M1	T0M1, T0M0: Timer 0 operating mode control bits 00: Unused 01: Event counter mode (external clock) 10: Timer mode (internal clock) 11: Pulse width measurement mode

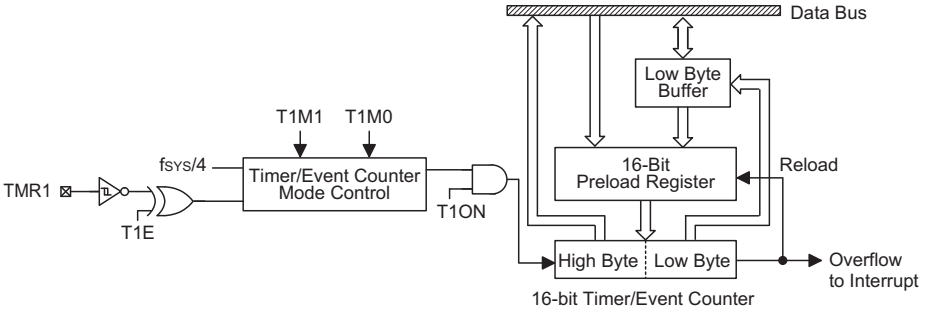
TMR0C (0EH) Register

Bit No.	Label	Function
0~2	—	Unused bit, read as ″0″
3	T1E	Timer 1 active edge control bit 0: active on low to high 1: active on high to low
4	T1ON	Timer 1 control bit 0: disable 1: enable
5	—	Unused bit, read as ″0″
6 7	T1M0 T1M1	T1M1, T1M0: Timer 1 operating mode control bits 00: Unused 01: Event counter mode (external clock) 10: Timer mode (internal clock) 11: Pulse width measurement mode

TMR1C (11H) Register



Timer/Event Counter 0



Timer/Event Counter 1

Input/Output Ports

There are 42 bidirectional input/output lines in the microcontroller, labeled from PA to PF, which are mapped to the data memory of [12H], [14H], [16H], [18H], [1AH] and [1CH] respectively. All of these I/O ports can be used for input and output operations. For input operation, these ports are non-latching, that is, the inputs must be ready at the T2 rising edge of instruction "MOV A,[m]" (m=12H, 14H, 16H, 18H, 1AH or 1CH). For output operation, all the data is latched and remains unchanged until the output latch is rewritten.

Each I/O line has its own control register (PAC, PBC, PCC, PDC, PEC, PFC) to control the input/output configuration. With this control register, CMOS output or Schmitt trigger input with or without pull-high resistor structures can be reconfigured dynamically under software control.

To function as an input, the corresponding latch of the control register must write a "1". The input source also depends on the control register. If the control register bit is "1" the input will read the pad state. If the control register bit is "0" the contents of the latches will move to the internal bus. The latter is possible in the "Read-modify-write" instruction. For output function, CMOS is the only configuration (except PB can be configured as CMOS output or NMOS output). These control registers are mapped to locations 13H, 15H, 17H, 19H, 1BH and 1DH. PA0 is pin-shared with PSYNC or VSYNC signal (dependent on PSYNC/VSYNC option). PB0 is pin-shared with SCLK signal (dependent on SCLK option).

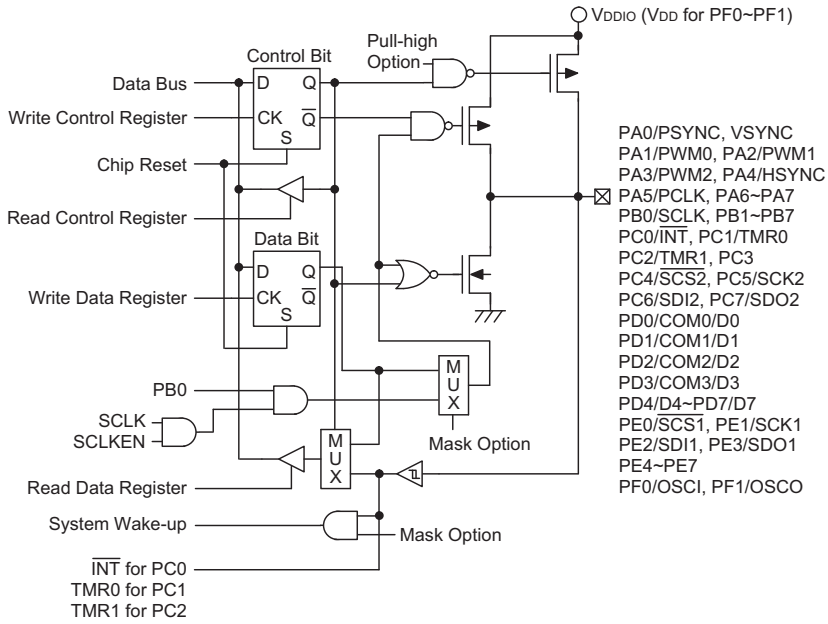
After a chip reset, these input/output lines remain at high levels or floating state (depending on the pull-high options). Each bit of these input/output latches can be set or cleared by "SET [m].i" and "CLR [m].i" (m=12H, 14H, 16H, 18H , 1AH or 1CH) instructions.

Some instructions first input data and then follow the output operations. For example, "SET [m].i", "CLR [m].i", "CPL [m]", "CPLA [m]" read the entire port states into the CPU, execute the defined operations (bit-operation), and then write the results back to the latches or the accumulator.

All the I/O ports have the capability of waking-up the device.

It is recommended that unused or not bonded out I/O lines should be set as output pins by software instruction to avoid consuming power under input floating state.

Note that the I/O lines, from PA to PE, the power supply is from VDDIO pin, except PF, from VDD.



Input/Output Ports

Low Voltage Reset – LVR

The microcontroller provides low voltage reset circuit in order to monitor the supply voltage of the device. If the supply voltage of the device is within the range 0.9V~V_{LVR}, such as changing a battery, the LVR will automatically reset the device internally.

The LVR includes the following specifications:

- The low voltage range (0.9V~V_{LVR}) has to be maintained for over 1ms, otherwise, the LVR will ignore it and do not perform a reset function.
- The LVR uses the "OR" function with the external $\overline{RES}$ signal to perform a chip reset.

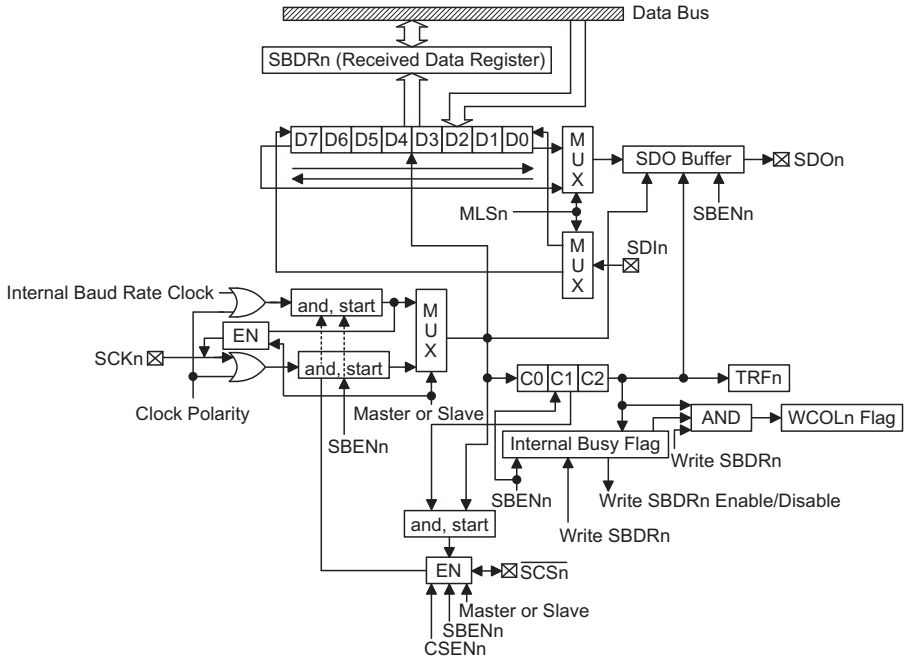
SPI Serial Interface

This device includes two SPI Serial Interfaces, namely SPI1 and SPI2. The SPI interface is a full duplex serial data link, originally designed by Motorola, which allows multiple devices connected to the same SPI bus to communicate with each other. The devices communicate using a master/slave technique where only the single master device can initiate a data transfer. A simple four line signal bus is used for all communication and these pins are shared with normal I/O pins. The SPI function is selected and controlled via configuration options and an SBCR register. The following table shows the SPI pin-shared options.

Note that the SPIn stands for SPI1 and SPI2. The "n", known as "1" or "2", is used to distinguish these two SPIs in the following sections.

SPI Interface Communication

Four lines are used for SPIn communication known as SDIn - Serial Data Input, SDOn - Serial Data Output, SCKn -Serial Clock and $\overline{SCSn}$ - Slave Select. Note that the condition of the Slave Select line is conditioned by the CSENn bit in the SBCRn control register. If the CSENn bit is high then the $\overline{SCSn}$ line is active while if the bit is low then the $\overline{SCSn}$ line will be a normal I/O pin. The following timing diagram depicts the basic timing protocol of the SPIn bus.



Note: n =0 or 1

WCOLn: set by SPIn cleared by users

CSENn: enable/disable chip selection function pin

1. master mode: 1/0 = with/without $\overline{SCSn}$ output function
2. slave mode: 1/0 = with/without $\overline{SCSn}$ input control function

SBENn: enable/disable serial bus (0: initialise all status flags)

1. When SBENn=0, all status flags should be initialised
2. When SBENn=1, all SPI related function pins should stay at floating state

TRFn : data transmitted or received, 0: data is transmitting or still not received

SPIn_CPOL 1/0: clock polarity rising/falling edge : configuration option

SPI Block Diagram

SPI pin-shared I/O Option Table

The following table shows how the SPIn pins are related to the SPIn configuration options and the control bits in the SBCRn register.

Configuration Options		Register Options		Pin-shared I/O Functions		Note
SPI_ENn	SPI_CSENn	SBENn	CSENn	SPIn or I/O	$\overline{SCSn}$ or I/O	
0	x	x	x	I/O	I/O	
1	0	x	x	SPIn	I/O	$\overline{SCSn}$ not floating
1	1	x	0	SPIn	I/O	$\overline{SCSn}$ not floating
1	1	x	1	SPIn	$\overline{SCSn}$	

SPIn Registers

There are several registers associated with the SPIn Interface. These are the SBCRn register which is the control register and the SBDRn which is the data register. The SBCRn register is used to setup the required setup parameters for the SPIn bus and also used to store associated operating flags, while the SBDRn register is used for data storage.

After Power on, the contents of the SBDRn register will be in an unknown condition while the SBCRn register will default to the condition below:

CKSn	M1n	M0n	SBENn	MLSn	CSENn	WCOLn	TRFn
0	1	1	0	0	0	0	0

Note that data written to the SBDRn register will only be written to the TXRXn buffer, whereas data read from the SBDRn register will actual be read from the register.

SPIn Bus Enable/Disable

To enable the SPIn bus, CSENn = 1, $\overline{SCSn}$=0, then wait for data to be written to the SBDRn (TXRXn buffer) register.

For the Master Mode, after data has been written to the (TXRXn buffer) register, then transmission or reception will start automatically. When all the data has been transferred the TRFn bit should be set. For the Slave Mode, when clock pulses are received on SCKn, data in the TXRXn buffer will be shifted out or data on SDIn will be shifted in.

To Disable the SPIn bus SCKn, SDIn, SDOn, $\overline{SCSn}$ will enter I/O mode.

Bit No.	Label	Function
0	TRFn	SPIn Transmit/Receive Complete flag 0: data is being transferred; 1: SPIn data transmission is completed The TRFn bit is the Transmit/Receive Complete flag and is set ″1″ automatically when an SPIn data transmission is completed, but must set to ″0″ by the application program. It can be used to generate an interrupt.
1	WCOLn	SPIn Write Collision flag 0: no collision; 1: collision The WCOLn flag is used to detect if a data collision has occurred. If this bit is high it means that data has been attempted to be written to the SIMDn register during a data transfer operation. This writing operation will be ignored if data is being transferred. The bit can be cleared by the application program. Note that using the WCOLn bit can be disabled or enabled via configuration option.
2	CSENn	SPIn $\overline{SCS}$ pin Control 0: disable; 1: enable The CSENn bit is used as an enable/disable for the $\overline{SCSn}$ pin. If this bit is low, then the SCSBn pin will be disabled and placed into an I/O mode. If the bit is high the SCSBn pin will be enabled and used as a select pin. Note that using the CSENn bit can be disabled or enabled via configuration option.
3	MLSn	SPIn Data shift order 0: LSB; 1: MSB This is the data shift select bit and is used to select how the data is transferred, either MSB or LSB first. Setting the bit high will select MSB first and low for LSB first.
4	SBENn	Serial Bus Control bit 0: disable; 1: enable Depend upon CSENn bit.
5 6	M1n M0n	Master/Slave/Baud rate control bits 00: Master, baud rate is f_{SIO} 01: Master, baud rate is $f_{SIO}/4$ 10: Master, baud rate is $f_{SIO}/16$ 11: Slave mode
7	CKSn	Clock Source Select bit 0: $f_{SIO}=f_{SYS}/4$; 1: $f_{SIO}=f_{SYS}$

Note: The TRFn flag will also generate an SPIn interrupt signal, for more information refer to the interrupt section.

SPIn Interface Control Register

SPIn Operation

All communication is carried out using the 4-line interface for both Master or Slave Mode which is controlled by configuration options and the SBCRn register. The timing diagram shows the basic operation of the bus. The CSENn bit in the SBCRn register and the SPIn_EN configuration option control the overall function of the SPIn interface. Setting these two bits high, will enable the SPIn interface by allowing the $\overline{SCSn}$ line to be active, which can then be used to control the SPIn interface. Meanwhile, the PE0~PE3 or PC4~PC7 will be the SPIn function pins and the corresponding pull-high resistor will be disabled by hardware.

If the CSENn bit is low, the SPIn interface will be disabled and the $\overline{SCSn}$ pins will be setup as normal I/O pins and can therefore not be used for control of the SPIn interface. The SBENn bit in the SBCRn register must also be high which will place the SDIn line in a floating condition and the SDOn line high. If in Master Mode the SCKn line will be either high or low depending upon the clock polarity configuration option. If in Slave Mode the SCKn line will be in a floating condition. If SBENn is low then the bus will be disabled and SCSn, SDIn, SDOn and SCKn will all be in a I/O mode.

In the Master Mode the Master will always generate the clock signal. The clock and data transmission will be initiated after data has been written to the SBDRn register.

In the Slave Mode, the clock signal will be received from an external master device for both data transmission or reception. The following sequences show the order to be followed for data transfer in both Master and Slave Mode.

Master Mode

- Step 1
  Select the clock source using the CKSn bit in the SBCRn control register
- Step 2
  Setup the M0n and M1n bits in the SBCRn control register to select the Master Mode and the required Baud rate. Values of 00, 01 or 10 can be selected.
- Step 3
  Setup the CSENn bit and setup the MLSn bit to choose if the data is MSB or LSB first, this must be same as the Slave device.
- Step 4
  Setup the SBENn bit in the SBCRn control register to enable the SPIn interface.
- Step 5
  For write operations: write the data to the SBDRn register, which will actually place the data into the TXRXn buffer. Then use the SCKn and $\overline{SCSn}$ lines to output the data.
  Goto to step 6.For read operations: the data transferred in on the SDIn line will be stored in the TXRXn buffer until all the data has been received at which point it will be latched into the SBDRn register.
- Step 6
  Check the WCOLn bit, if set high then a collision error has occurred so return to step5. If equal to zero then go to the following step.
- Step 7
  Check the TRFn bit or wait for an SBIn serial bus interrupt.
- Step 8
  Read data from the register.
- Step 9
  Clear TRFn.
- Step10
  Goto step 5.

Slave Mode

- Step 1
  The CKSn bit has a don′t care value in the slave mode.
- Step 2
  Setup the M0n and M1n bits to 11 to select the Slave Mode. The CKSn bit is don′t care.
- Step 3
  Setup the CSENn bit and setup the MLS bit to choose if the data is MSB or LSB first, this must be same as the Master device.
- Step 4
  Setup the SBENn bit in the SBCRn control register to enable the SPIn interface.
- Step 5
  For write operations: write data to the SBCRn register, which will actually place the data into the TXRXn register, then wait for the master clock and $\overline{SCSn}$ signal. After this goto step 6.
  For read operations: the data transferred in on the SDIn line will be stored in the TXRXn buffer until all the data has been received at which point it will be latched into the SBDRn register.
- Step 6
  Check the WCOLn bit, if set high then a collision error has occurred so return to step5. If equal to zero then go to the following step.
- Step 7
  Check the TRFn bit or wait for an SBI serial bus interrupt.
- Step 8
  Read data from the SBDRn register.
- Step 9
  Clear TRFn
- Step10
  Goto step 5

SPIn Configuration Options

Several configuration options exist for the SPIn Interface function which must be setup during device programming. One option is to enable the operation of the WCOLn, write collision bit, in the SBCRn register. Another option exists to select the clock polarity of the SCKn line. A configuration option also exists to disable or enable the operation of the CSENn bit in the SBCRn register. If the configuration option disables the CSENn bit then this bit cannot be used to affect overall control of the SPIn Interface.

Error Detection

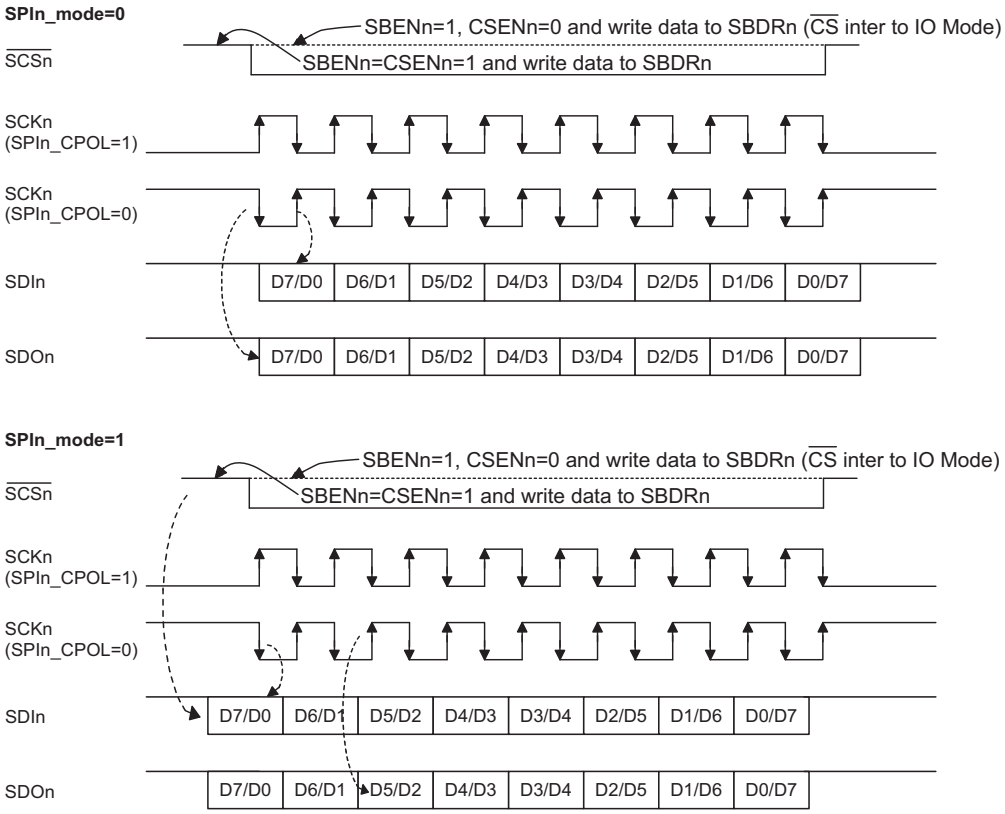
The WCOLn bit in the SBCRn register is provided to indicate errors during data transfer. The bit is set by the Serial Interface but must be cleared by the application program. This bit indicates a data collision has occurred which happens if a write to the SBDRn register takes place during a data transfer operation and will prevent the write operation from continuing. The bit will be set high by the Serial Interface but has to be cleared by the user application program. The overall function of the WCOLn bit can be disabled or enabled by a configuration option.

Programming Considerations

When the device is placed into the Power Down Mode note that data reception and transmission will continue. The TRFn bit is used to generate an interrupt when the data has been transferred or received.

There are two SPIn modes for different data output format, selected by a configuration option. The

following diagram illustrates these two data format, which are dependent on the SPIn_CPOL and SPIn_mode options.

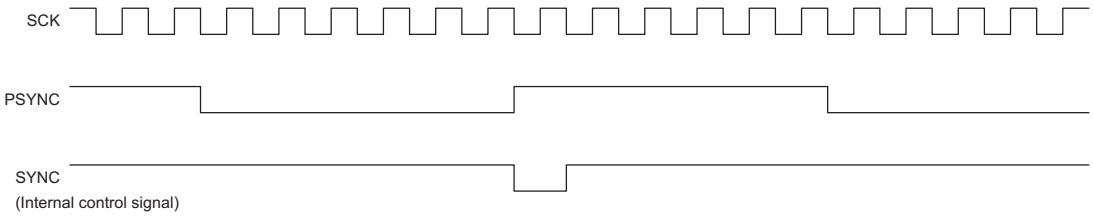


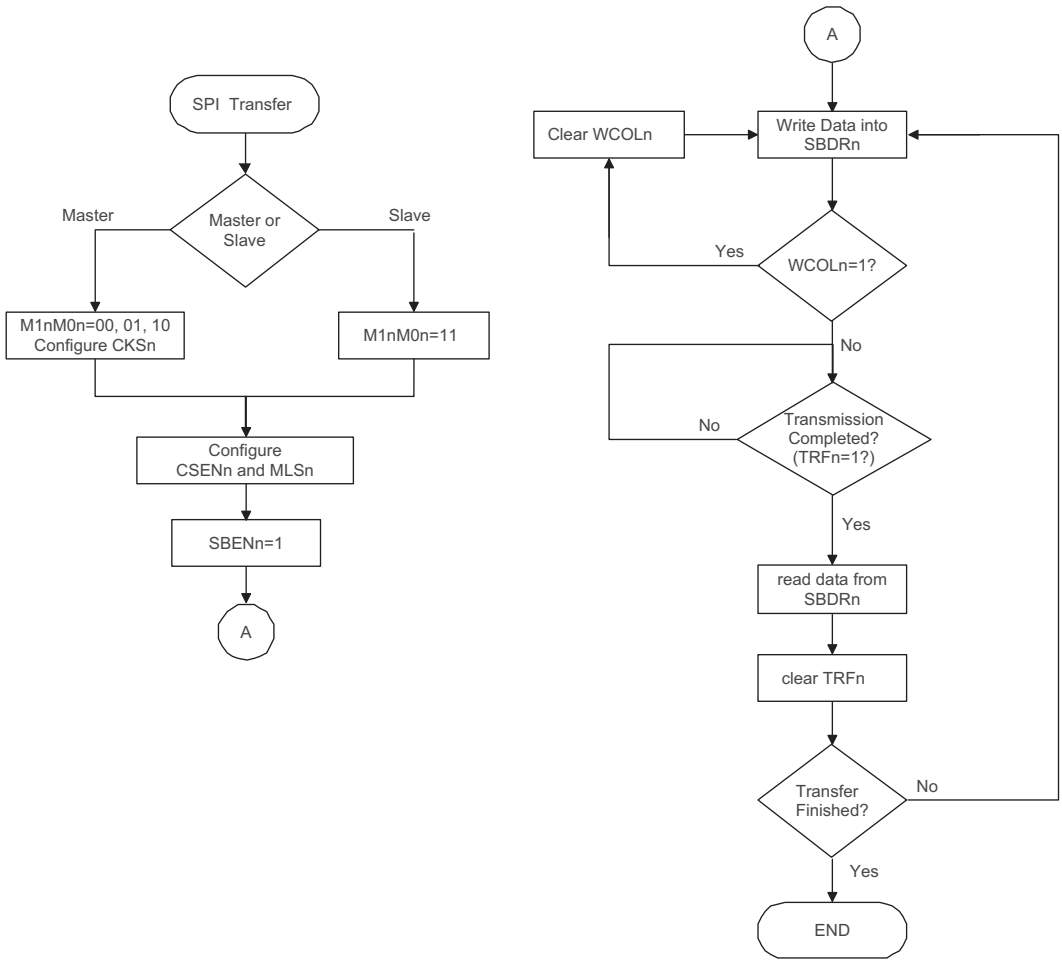
	D7	D6	D5	D4	D3	D2	D1	D0	
SBCRn	CKSn	M1n	M0n	SBENn	MLSn	CSENn	WCOLn	TRFn	SBCRn: Serial Bus Control Register
Default	0	1	1	0	0	0	0	0	
SBDRn	D7	D6	D5	D4	D3	D2	D1	D0	SBDRn: Serial Bus Data Register
Default	X	X	X	X	X	X	X	X	

Note: "X" means unknow.

SPIn Bus Timing

If the PSYNC function is enabled, this device will generate an internal SYNC signal to synchronize the received data in SDBR register, to indicate that the next receiving bit of data will be the MSB. The following timing diagram illustrates the relationship between the SCK and PSYNC control signals in slave mode. Note that if the SPI is working under master mode, the PSYNC function and the SYNC will be ignored by hardware.





SPI Transfer Control Flowchart

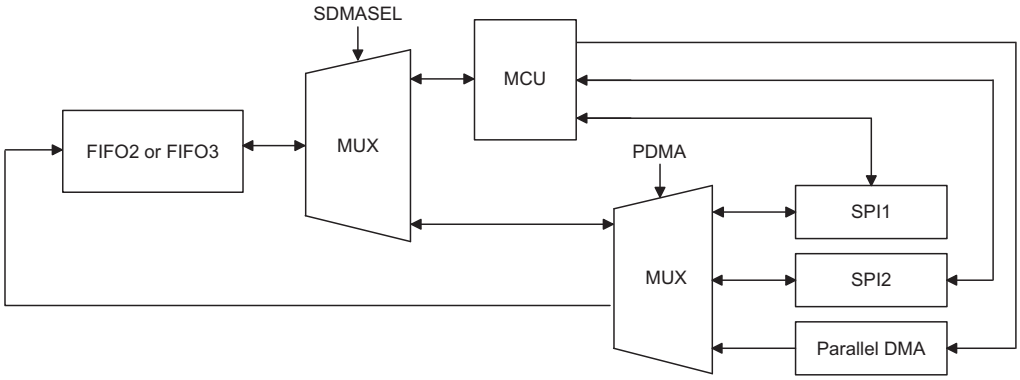
DMA Function

The HT82A525R provides both serial and parallel interface DMA functions. The DMA serial interface is implemented using one of the SPI interfaces, either SPI1 or SPI2. The parallel DMA is implemented using Port D.

DMA Overview

For the serial DMA interface, the SPI1 related pins are pin shared with PE0~PE3, and the SPI 2 related pins are pin shared with PC4~PC7. The Parallel DMA data lines are pin shared with Port D, HSYNC is pin shared with PA4 and the PCLK is pin shared with PA5. When the PDMA bit in the CLK register is set high, the parallel DMA function will be selected and the data will be written to FIFO2 directly. The FIFO2 size is 64×3 bytes for parallel DMA Mode. FIFO3 can′t be use for parallel DMA Mode.

PSYNC/VSYNC is pin shared with the PA0 pin. VSYNC is used to synchronise the Parallel DMA data received data and PSYNC is used to synchronise the serial DMA received data. The SCLK is pin shared with the PB0 pin and used as the clock output for serial or parallel DMA functions. The clock output is controlled by the SCLKEN and CLKAUTOB bits in the CLK register. The clock output frequency, 12, 16, 24, 6, 8, 4, 3, 2 MHz is selected by the FSCLK configuration option.



DMA Block Diagram

Bit No.	Label	R/W	Function
0	SCLKEN	R/W	SCLK pin output control bit 0: disable; 1: enable
1	CLKAUTOB	R/W	SCLK pin output halt control bit 0: when FIFO3 is full or SCLKEN is ″0″ ; 1: when SCLKEN is ″0″
2	PDMA	R/W	Parallel DMA function control bit 0: disable; 1: enable
3	PDATA_SEL	R/W	Parallel DMA data format select bit 0: Receive odd Parallel DMA data only; 1: Receive all Parallel DMA data
4	PDMA_MOD	R/W	Parallel DMA function select bit 0: Fingerprint mode - for fingerprint application only 1: Normal mode, for general purpose application
5	FIG_PIX	R/W	Fingerprint frame pixel select bit 0: QVGA mode (320×240); 1: CIF mode (352×288) This bit is used to select the Fingerprint frame pixel if the PDMA_MOD is cleared to ″0″.
6~7	—	—	Undefined bit, read as ″0″

This control bit is used to control the SCLK output no mater in serial or parallel DMA mode.

CLK Register

If the PDMA bit is cleared to zero and the SDMAEN bit in the MISC register is set high, then the serial DMA function will be enabled. The SBDR1 data of the SPI1 or the SBDR2 data of the SPI2, selected by the SDMSEL bit in the MISC register, will be written to FIFO3.

Serial DMA

In the Serial DMA mode, the SPI1 or SPI2 data, selected by SDMASEL bit, can be written to FIFO3 directly.

SCLK is pin shared with the PB0 pin and used as the clock output for serial or parallel DMA function. The clock output function is controlled by the SCLKEN and CLKAUTOB bits in the CLK register. The clock output frequency, (12, 16, 24, 6, 8, 4, 3, 2 MHz), is selected by the FSCLK by configuration option.

If the SBEN, SDMAEN bits are set high, and PDMA is cleared to zero, the SPI interface DMA will enter the master mode and start to send out the SCK clock. If the SPI is in the master mode and is used as a receiver, the SCK clock will be stopped automatically when FIFO3 is full and will restart again when FIFO3 is not full. If the SPI is in the slave mode and is used as a receiver, if FIFO3 is full, the SPI will stop receiving data. It will restart again if FIFO3 is not full. The frequency of the SCK is selected using the CKS, M1 and M0 bits in the SBCR register. Note that the SCK clock output will stop at a low level if the CPOL bit is set high and stop at a high level if the CPOL bit is cleared to zero.

Each SPI interface can support both master or slave mode DMA. The direction of the SPI DMA is determined by the SETIO3 bit. The corresponding buffer size of 8, 16, 32 or 64 bytes is determined by bits DLEN 0~1 by configuration option.

Parallel DMA

In the Parallel DMA mode, HSYNC is pin shared with PA4, PCLK is pin shared with PA5 pin, the VSYNC is pin shared with PA0 pin and the Parallel DMA data D0~D7 pins are pin shared with PD0~PD7. HSYNC and the PCLK are used to synchronize the data of the parallel interface pins which are pin shared with port D.

SCLK is pin shared with the PB0 pin and is used as the clock output for serial or parallel DMA functions. The clock output function is controlled by the SCLKEN and CLKAUTOB bits in the CLK register. The clock output frequency of 12, 16, 24, 6, 8, 4, 3 or 2 MHz is selected by the FSCLK configuration option.

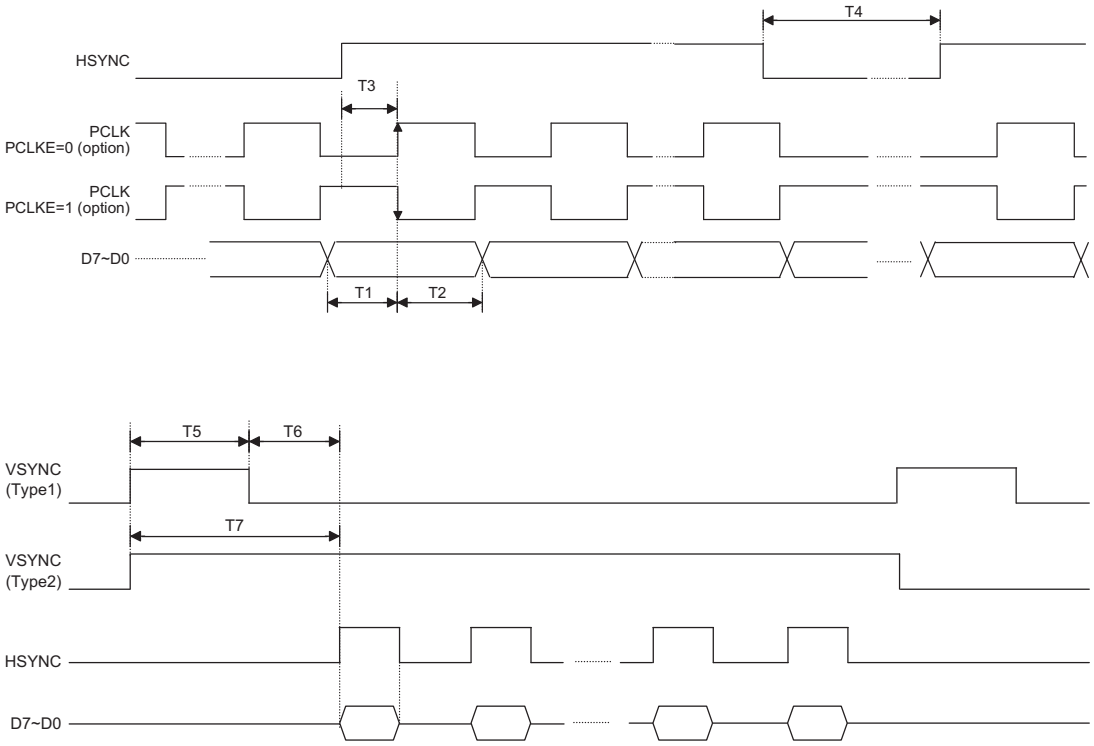
The parallel DMA function can be used in fingerprint mode or normal mode, which is selected using the PDMA_MOD bit in the CLK register.

In the normal mode, the VSYNC signal of the parallel DMA received data will be ignored. In the fingerprint mode, there are two frame pixel modes, the QVGA mode (320x240 pixels) or the CIF mode (352x288 pixels). They are selected by the FIG_PIX bit in the CLK register.

If the PDMA bit is set high, the parallel DMA data will be written to FIFO2, the size of which is 64x3 bytes. There is only a slave mode for the parallel DMA and the data length can be selected as 5 bits or 8 bits by configuration option.

The FIFO2 buffers will start receiving data when HSYNC is set high and stop receiving data when HSYNC is cleared to zero. If FIFO2 is not full and HSYNC is cleared to zero to stop receiving data, then FIFO2 should not receive any incoming data. Only the received data in FIFO2 can be delivered to the Endpoint. Not until the next HSYNC high signal can FIFO2 start receiving data from the next buffer. If all buffers of the FIFO2 are full before HSYNC has a falling edge, this received data should be ignored. There are two Parallel DMA data selections, to receive odd Parallel DMA data only or to receive all Parallel DMA data, selected by the PDATA_SEL bit in the CLK register.

If VSYNC is enabled, it can be used to synchronize the Parallel DMA data D0~D7. There are two types of VSYNC control signal. The following diagram illustrates the Parallel DMA interface control signals.



Symbol	Parameter	Min.	Max.	Unit
	PCLK frequency	—	4	MHz
	PCLK duty cycle	45	55	%
T1	Data setup time to PCLK	5	—	ns
T2	Data hold time to PCLK	7	—	ns
T3	HSYNC ↑ to PCLK delay	5	—	ns
T4	Horizontal blank time (T_{hblank})	24	—	T_{PCLK}
T5	VSYNC(Tpye1) high pulse time	200	—	ns
T6	VSYNC(Tpye1) ↓ to HSYNC delay time	500	—	ns
T7	VSYNC(Tpye2) ↑ to HSYNC delay time	700	—	ns

Parallel DMA Data Timing Description

Suspend Wake-Up or Remote Wake-Up

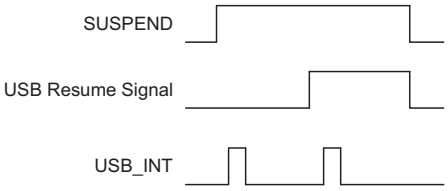
If there is no signal on the signal bus for over 3ms, the HT82A525R will go into suspend mode. The Suspend line (bit 0 of the USC) will be set to 1 and a USB interrupt is triggered to indicate that the HT82A525R should jump to suspend state to meet the 500μA USB suspend current spec.

In order to meet the 500μA suspend current, the firmware should disable the USB clock by clearing the USBCKEN (bit3 of the UCC) to ″0″. The suspend current is about 300μA.

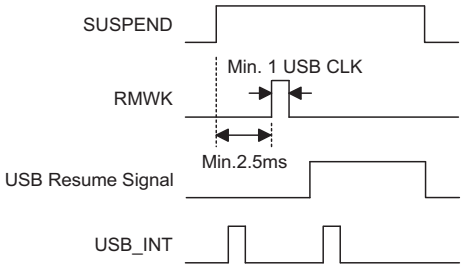
The user can also further decrease the suspend current by setting the SUSP2 (bit4 of the UCC). But if the SUSP2 is set, user should make sure not to enable the LVR option, otherwise, the HT82A525R will be reset. If user set the SUSP2 (bit4 of the UCC) In the USB mode, user must set Rctrl (bit7 of the UCC) before set SUSP2 (bit4 of the UCC), otherwise, USB will disconnected.

When the resume signal is sent out by the host, the HT82A525R will wake-up the by USB interrupt and the Resume line (bit 3 of the USC) is set. In order to make the HT82A525R work properly, the firmware must set the USBCKEN (bit 3 of the UCC) to 1 and clear the SUSP2 (bit4 of the UCC). If user set the Rctrl (bit7 of the UCC) and SUSP2 (bit4 of the UCC) In the USB suspend, when it will wake-up user must clr Rctrl (bit7 of the UCC) before clr SUSP2 (bit4 of the UCC). Since the Resume signal will be cleared before the Idle signal is sent out by the host and the Suspend line (bit 0 of the USC) is going to ″0″. So when the MCU is detecting the Suspend line (bit0 of USC), the Resume line should be remembered and taken into consideration.

After finishing the resume signal, the suspend line will go inactive and a USB interrupt is triggered. The following is the timing diagram:



The device with remote wake-up function can wake-up the USB Host by sending a wake-up pulse through RMWK (bit 1 of the USC). Once the USB Host receive the wake-up signal from the HT82A525R, it will send a Resume signal to the device. The timing is as follow:



USB Interface

The HT82A525R has 4 Endpoints (EP0~EP3). EP0~EP1 are support Interrupt transfer, EP2~EP3 is support Bulk transfer.

There are 12 registers, including USC (20H), USR (21H), UCC (22H), AWR (address+remote wake-up 23H), STALL (24H), SIES (25H), MISC (26H), SETIO (27H), FIFO0 (28H), FIFO1 (29H), FIFO2 (2AH) and FIFO3 (2BH) used for the USB function.

The FIFO size of each FIFO is 8 byte (FIFO0), 8 byte (FIFO1), 64 byte (FIFO2) and 128 byte (FIFO3), and total of 208 bytes.

URD (bit7 of the USC) is USB reset signal control function definition bit.

Bit No.	Label	R/W	Function
0	SUSP	R	USB suspend indication flag 0: not in suspend mode 1: in suspend mode When this bit is set to ″1″ (set by SIE), it indicates that the USB bus enters the suspend mode. The USB interrupt is also triggered on any changes of this bit.
1	RMWK	R/W	USB remote wake-up command 0: disable 1: enable It is set by the MCU to force the USB host leaving the suspend mode. Set RMWK bit to ″1″ to enable remote wake-up. When this bit is set to ″1″, a 2μs delay for clearing this bit to ″0″ is needed to insure that the RMWK command is accepted by the SIE.
2	URST	R/W	USB reset indication bit 0: no USB reset 1: USB reset This bit is set or cleared by USB SIE. When the URST is set to ″1″, this indicates that a USB reset has occurred and a USB interrupt will be initialized.
3	RESUME	R	USB resume indication bit 0: in suspend mode 1: resume When the USB leaves the suspend mode, this bit is set to ″1″ (set by SIE). This bit will appear for 20ms, waiting for the MCU to detect it. When the RESUME is set by SIE, an interrupt will be generated to wake-up the MCU. In order to detect the suspend state, MCU should set the USBCKEN and SUSP2 (in the SCC register) to enable the SIE detect function. The RESUME will be cleared while the SUSP is set to ″0″. When MCU detects the SUSP, the RESUME (which causes MCU to wake-up) should be remembered and token into consideration.
4	V33O	R/W	V33O enable control bit 0: turn off 1: turn on
5	PLL	R/W	PLL enable control bit 0: turn on 1: turn off
6	—	—	Unused bit, read as ″0″
7	URD	R/W	USB reset signal control function definition 1: Will reset the MCU 0: Cannot reset the MCU

USC (20H) Definitions

The USR (USB endpoint interrupt status register) register is used to indicate which endpoint is accessed. The endpoint request flags (EP0IF, EP1IF, EP2IF and EP3IF) are used to indicate which endpoints are accessed. If an endpoint is accessed, the related endpoint request flag will be set to "1", and then the USB interrupt takes place or not will depend on the "Endpoint 0~3 interrupt function" via the configuration option. If the "Endpoint 0~3 interrupt function" is configured as "Disable", then the USB interrupt will occur (if the USB interrupt is enabled, the corresponding interrupt is enabled and the stack is not full). If the "Endpoint 0~3 interrupt function" is configured as "Enable", the USB interrupt will need to be managed by the Endpoint Interrupt control bits, EEP0I~EEP3I, in the USR register as well. The Endpoint Interrupt control bits are used to control which endpoint is accessed and generate interrupt. Only when the related Endpoint Interrupt control bits are selected as "Enable", then the related USB interrupt will take place, otherwise, the USB interrupt will not occur. When the active endpoint request flag is served, the endpoint request flag has to be cleared to "0".

Bit No.	Label	R/W	Function
0	EP0IF	R/W	The endpoint 0 interrupt request flag 0: inactive 1: active When this bit is set to "1" (set by SIE), it indicates that the endpoint 0 is accessed and a USB interrupt will occur. When the interrupt has been served, this bit should be cleared by firmware.
1	EP1IF	R/W	The endpoint 1 interrupt request flag 0: inactive 1: active When this bit is set to "1" (set by SIE), it indicates that the endpoint 1 is accessed and a USB interrupt will occur. When the interrupt has been served, this bit should be cleared by firmware.
2	EP2IF	R/W	The endpoint 2 interrupt request flag 0: inactive 1: active When this bit is set to "1" (set by SIE), it indicates that the endpoint 2 is accessed and a USB interrupt will occur. When the interrupt has been served, this bit should be cleared by firmware.
3	EP3IF	R/W	The endpoint 3 interrupt request flag 0: inactive 1: active When this bit is set to "1" (set by SIE), it indicates that the endpoint 3 is accessed and a USB interrupt will occur. When the interrupt has been served, this bit should be cleared by firmware.
4	EEP0I	R/W	The endpoint 0 interrupt enable 0: enable 1: disable
5	EEP1I	R/W	The endpoint 1 interrupt enable 0: enable 1: disable
6	EEP2I	R/W	The endpoint 2 interrupt enable 0: enable 1: disable
7	EEP3I	R/W	The endpoint 3 interrupt enable 0: enable 1: disable

USR (21H) Definitions

There is a system clock control register implemented to select the clock used in the MCU. This register consists of USB clock control bit (USBCKEN), second suspend mode control bit (SUSP2) and system clock selection (SYSCLK).

If HIRC oscillator is used and for USB mode should enable CLK_adj bit for PSD register to adjust system clock for temperature.

The following table defines which endpoint FIFO is selected, EPS2, EPS1 and EPS0.

Bit No.	Label	R/W	Function
0 1 2	EPS0 EPS1 EPS2	R/W	Accessing endpoint FIFO selection. EPS2, EPS1, EPS0: 000: Select endpoint 0 FIFO 001: Select endpoint 1 FIFO 010: Select endpoint 2 FIFO 011: Select endpoint 3 FIFO 100: Reserved for future expansion, cannot be used 101: Reserved for future expansion, cannot be used 110: Reserved for future expansion, cannot be used 111: Reserved for future expansion, cannot be used If the selected endpoints do not exist, the related functions are not available.
3	USBCKEN	R/W	USB clock Control bit 0: disable 1: enable
4	SUSP2	R/W	Suspend mode control bit 0: Normal Mode 1: HALT Mode This bit is used to reduce power consumption in suspend mode. In normal mode, clear this bit to ″0″. In HALT mode, set this bit to ″1″ to reduce power consumption.
5	—	—	Unused bit, read as ″0″
6	SYSCLK	R/W	System clock oscillator frequency control bit 0: 12MHz 1: 6MHz This bit is used to specify the system clock oscillator frequency used by the MCU. If a 6MHz crystal oscillator or resonator is used, this bit should be set to ″1″. If a 12MHz crystal oscillator or resonator is used, this bit should be cleared to ″0″ .
7	RCtrl	R/W	7.5kΩ resistor between USBDP and Vbus select bit 0: without resistor 1: with resistor

UCC (22H) Definitions

The AWR register contains the current address and the remote wake-up function control bit. The initial value of the AWR is ″000H″0. The address value extracted from the USB command is not to be loaded into this register until the SETUP stage is finished.

Bit No.	Label	R/W	Function
0	WKEN	R/W	Remote wake-up enable/disable 0: disable 1: enable
1~7	AD0~AD6	R/W	USB device address

AWR (23H) Definitions

The STALL register shows whether the corresponding endpoint works properly or not. As soon as the endpoint works improperly, the related bit in the STALL has to be set to "1". The STALL will be cleared by the USB reset signal.

Bit No.	Label	R/W	Function
0~3	STL0~ STL3	R/W	stalled USB endpoints control
4~7	—	—	Undefined bit, read as "0"

STALL (24H) Definitions

This bit is used to configure the USB SIE to automatically change the device address with the value of the Address+Remote_WakeUp Register (42H). When this bit is set to 1 by F/W, the USB SIE will update the device address with the value of the Address+Remote_WakeUp Register (42H) after the PC Host has successfully read the data from the device by the IN operation. The USB SIE will clear the bit after updating the device address.

Otherwise, when this bit is cleared to 0, the USB SIE will update the device address immediately after an address is written to the Address+Remote_WakeUp Register (42H).

The SIES Register is used to indicate the present signal state which the USB SIE received and also determines whether the USB SIE has to change the device address automatically.

Bit No.	Label	R/W	Function
0	Adr_set	R/W	Device address configuration bit 0: update device address immediately 1: update by the IN operation
1	F0_ERR	R/W	accessed FIFO0 errors indication bit 0: no errors 1: some errors This bit is set by the USB SIE and cleared by F/W.
2~6	—	—	Unused bit, read as "0"
7	NMI	R/W	NAK interrupt mask bit 0: not mask 1: mask

SIES Function Table

MISC register combines a command and status to control the desired endpoint FIFO action and to show the status of the desired endpoint FIFO. The MISC will be cleared by USB reset signal.

Bit No.	Label	R/W	Function
0	REQUEST	R/W	FIFO request control bit 0: not requested; 1: requested After selecting the desired endpoint, FIFO can be requested by setting this bit as high active. Afterwards, this bit must be set low.
1	TX	R/W	The direction and transition end indication bit 0: read data from FIFO; 1: write data to FIFO This indicates the direction and transition end which the MCU accesses. When set as logic ″1″, the MCU writes data to FIFO. Afterwards, this bit must be set to logic ″0″ before terminating request to indicate transition end. For reading action, this bit must be set to logic 0 to indicate that the MCU wants to read and must be set to logic ″1″ afterwards.
2	CLEAR	R/W	Clear requested FIFO control bit 0: not clear; 1: clear This indicates an MCU clear requested FIFO, even if the FIFO is not ready. After clearing the FIFO, USB interface will send force_tx_err to tell Host that data under-run if Host wants to read data.
3	SDMAEN	R/W	Serial DMA control bit 0: disable; 1: enable This bit is used to control the enable or disable the SBDR of the serial interfaces (which is pin-shared with port E or port C) being written to FIFO3 directly. SPI interfaces can be controlled by MCU and MCU can transmit or receive data by writing or reading SBDR. It is allowed changing from 1 to 0 when the FIFO is not full.
4	SDMASEL	R/W	Serial DMA interface selection bit 0: Serial interface 1 (pin-shared with port E) 1: Serial interface 2 (pin-shared with port C)
5	SETCMD	R/W	FIFO command data indication bit 0: not SETCMD token; 1: SETCMD token
6	READY	R	FIFO ready indication bit 0: not ready to work; 1: ready to work
7	LEN0	R/W	Host sent 0-sized packet indication bit 0: not 0-sized packet; 1: 0-sized packet

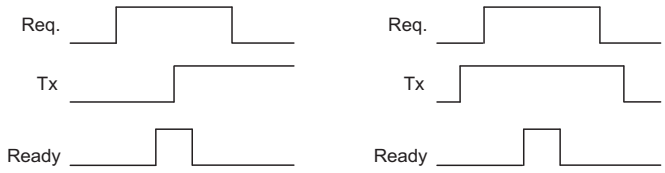
MISC (26H) Definitions

There are some timing constrains and usages illustrated here. By setting the MISC register, MCU can perform reading, writing and clearing actions. There are some examples shown in the following table for endpoint FIFO reading, writing and clearing.

Actions	MISC Setting Flow and Status
Read FIFO0 sequence	00H→01H→delay 2μs, check 41H→read* from FIFO0 register and check not ready (01H)→03H→02H
Write FIFO0 sequence	02H→03H→delay 2μs, check 43H→write* to FIFO0 register and check not ready (03H)→01H→00H
Check whether FIFO0 can be read or not	00H→01H→delay 2μs, check 41H (ready) or 01H (not ready)→00H
Check whether FIFO0 can be written or not	02H→03H→delay 2μs, check 43H (ready) or 03H (not ready)→02H
Write 0-sized packet sequence to FIFO0	02H→03H→delay 2μs, check 43H→01H→00H
Clear FIFO0 sequence	01H→delay 2μs→05H→delay 2μs→00H

Note: *: There are 2μs existing between 2 reading action or between 2 writing → action.

Read or Write FIFO Table



Read FIFO Timing　　Write FIFO Timing

Bit No.	Label	R/W	Function
0	—	—	Undefined bit, read as ″0″
1~3	SETIO1~3**	R/W	endpoints input or output pile selection bit 0: output pipe; 1: input pipe
4~7	—	—	Undefined bit, read as ″0″

SETIO (27H) Register, USB Endpoint 1~Endpoint3 Set IN/OUT Pipe Register

Note: *USB definition: when the host sends a ″set Configuration″, the Data pipe should send the DATA0 (Data toggle) first. So, when the device receives a ″set configuration″ setup command, user needs to toggle this bit so the next data will send a Data0 first.
**Needs to set the data pipe as an input pile or output pile. The purpose of this function is to avoid the host from abnormally sending only an IN or OUT token and disables the endpoint.

Label	R/W	Function
FIFOi	R/W	EPi accessing register (i = 0~3). When an endpoint is disabled, the corresponding accessing register should be disabled.

FIFO0~FIFO3 (28H~2BH) Register, USB Endpoint Accessing Registers Definitions

Bit No.	Label	R/W	Function
0	PS2_DAO	R/W	USBDM/DATA pin output control bit 0: output ″low″; 1: output ″high″ This control bit is used to generate the USBDM/DATA pin output signal in the 3D PS2 mouse mode. The default value is ″1″.
1	PS2_CKO	R/W	USBDP/CLK pin output control bit 0: output ″low″; 1: output ″high″ This control bit is used to generate the USBDP/CLK pin output signal in the 3D PS2 mouse mode. The default value is ″1″.
2	—	—	Undefined bit, read as unknown
3	PU	R/W	USBDP and USBDM internal 310kΩ pull-high resistor select bit 0: without pull up resistor; 1: with pull up resistor
4	CLK_adj	R/W	Automatic Clock adjustment control bit 0: disable (default); 1:enable This bit is used to adjust the HIRC mode system clock, to reduce the frequency deviation due to temperature issue. In the Power-down mode, this bit should be clear to reduce power consumption.
5	CLR_RCP	R/W	This bit is must enable and then disable by F/W to clear HIRC initial parameters after power on. 0: disable (default); 1:enable
6~7	—	—	Undefined bit, read as ″0″

PSD (3AH) Register

Pulse Width Modulator

The device contains three Pulse Width Modulation PWM outputs. Useful for such applications such as motor speed control, the PWM function provides an output with a variable frequency, and with a duty cycle that can be varied by setting particular values into the corresponding register pair.

Channel	PWM Mode	Output Pin	Register Names
0	8+4	PA1	PWM0DRL~ PWM0DRH
1	8+4	PA2	PWM1DRL~ PWM1DRH
2	8+4	PA3	PWM2DRL~ PWM2DRH

PWM Registers

Three register, located in the Data Memory are assigned to each Pulse Width Modulator output and are known as the PWM registers. It is in each register pair that the 12-bit value, which represents the overall duty cycle of one modulation cycle of the output waveform, should be placed. The PWM registers also contain the enable/disable control bit for the PWM outputs. To increase the PWM modulation frequency, each modulation cycle is modulated into sixteen individual modulation sub-sections, known as the 8+4 mode. Note that it is only necessary to write the required modulation value into the corresponding PWM register as the subdivision of the waveform into its sub-modulation cycles is implemented automatically within the microcontroller hardware.

This method of dividing the original modulation cycle into a further 16 sub-cycles enables the generation of higher PWM frequencies, which allow a wider range of applications to be served. As long as the periods of the generated PWM pulses are less than the time constants of the load, the PWM output will be suitable as such long time constant loads will average out the pulses of the PWM output. The difference between what is known as the PWM cycle frequency and the PWM modulation frequency should be understood.

As the PWM clock is defined by register PWMBR and the system clock f_{SYS} or $f_{SYS}/4$ (chosen via the PWMn_S bit), and as the PWM value is 12-bits wide, the overall PWM cycle frequency (f_{PWM}) as the following equation and the corresponding PWM modulation frequency for 8+4 mode is $f_{PWM}/256$.

$$f_{PWM}=\frac{1}{(1/f_{SYS})x(PWMBRn+1)}\text{ , when PWMn\_S=0}$$

$$f_{PWM}=\frac{1}{(4/f_{SYS})x(PWMBRn+1)}\text{ , when PWMn\_S=1}$$

where PWMBRn=0~255 and f_{SYS} may be 6MHz, 12MHz , n=0~2 , according register whether it is PWM0, PWM1 or PWM2

PWM Modulation Frequency	PWM Cycle Frequency	PWM Cycle Duty
$f_{PWM}/256$	$f_{PWM}/4096$	(PWM register value)/4096

8+4 PWM Mode Modulation

Each full PWM cycle, as it is 12-bits wide, has 4096 clock periods. However, in the 8+4 PWM mode, each PWM cycle is subdivided into sixteen individual sub-cycles known as modulation cycle 0 ~ modulation cycle 15, denoted as "i" in the table. Each one of these sixteen sub-cycles contains 256 clock cycles. In this mode, a modulation frequency increase of sixteen is achieved. The 12-bit PWM register value, which represents the overall duty cycle of the PWM waveform, is divided into two groups. The first group which consists of bit4~bit11 is denoted here as the DC value. The second group which consists of bit0~bit3 is known as the AC value. In the 8+4 PWM mode, the duty cycle value of each of the two modulation sub-cycles is shown in the following table.

Parameter	AC (0~15)	DC (Duty Cycle)
Modulation cycle i (i=0~15)	i < AC	$\frac{DC+1}{256}$
	i ≥ AC	$\frac{DC}{256}$

8+4 Mode Modulation Cycle Values

The accompanying diagram illustrates the waveforms associated with the 8+4 mode of PWM operation. It is important to note how the single PWM cycle is subdivided into 16 individual modulation cycles, numbered 0~15 and how the AC value is related to the PWM value.

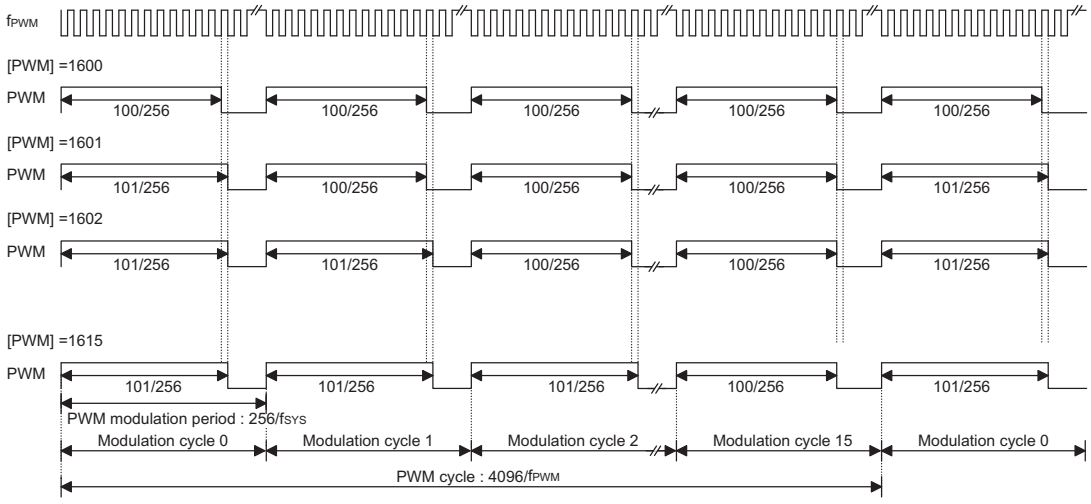
PWM Output Control

The three outputs, PWM0, PWM1, and PWM2 are shared with pins PA1, PA2 and PA3. To operate as a PWM output and not as an I/O pin, bit 0 of the relevant PWM low byte register bit must be set high. A zero must also be written to the corresponding bit in the PAC port control register, to ensure that the PWM0~2 output pin is setup as an output. After these two initial steps have been carried out, and of course after the required PWM 12-bit value has been written into the PWM register pair register, setting the corresponding bit in the PA data register high will enable the PWM data to appear on the pin. Writing a zero to the bit will disable the PWM output function and force the output low. In this way, the Port A data output register bits, can also be used as an on/off control for the PWM function. Note that if the enable bit in the PWM register is set high to enable the PWM function, but if the corresponding bit in the PAC control register is high to configure the pin as an input, then the pin can still function as a normal input line, with pull-high resistor selections.

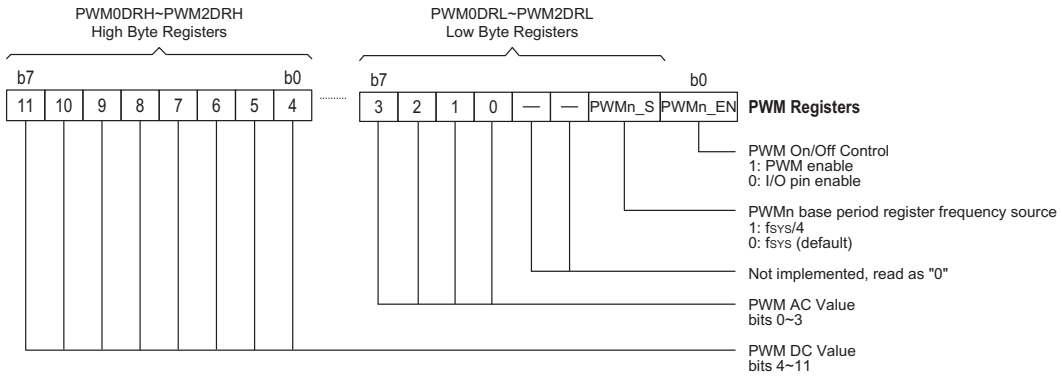
PWM Programming Example

The following sample program shows how the output is setup and controlled.

```
mov a,64h     ; setup PWM0 value to 1600 decimal which is 640H
mov pwm0h,a   ; setup PWM0H register value
clr pwm0l     ; setup PWM0L register value
clr pac.1     ; setup pin PA1 as an output
set pwm0en    ; set the PWM0 enable bit
set pa.1      ; Enable the PWM0 output
 :   :
 :   :
clr pa.1      ; PWM0 output disabled – PA1 will remain low
```



8+4 PWM Mode



PWM Register Pairs

Bit	R/W	Description
7~0	R/W	Used to define the base period of the PWM Range =1 ~ 256 × (1/f_{SYS} or 4/f_{SYS} chosen via the PWMn_S bit) where PWMBRn=0~255 (n= 0~2)

PWM Base Period Register PWMBR0 ~ PWMBR2

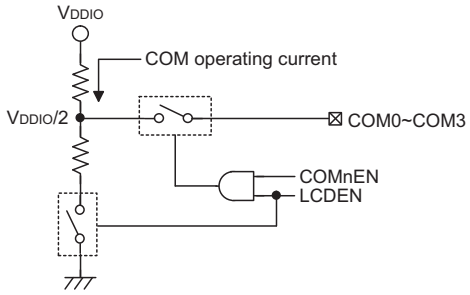
COM Function for LCD

The devices have the capability of driving external LCD panels. The common pins for LCD driving, COM0~COM3, are pin shared with certain pin on the PD0~ PD3 port. The LCD signals (COM and SEG) are generated using the application program.

LCD Operation

An external LCD panel can be driven using this device by configuring the PD0~PD3 pins as common pins and using other output ports lines as segment pins. The LCD driver function is controlled using the LCDC register which in addition to controlling the overall on/off function also controls the bias voltage setup function. This enables the LCD COM driver to generate the necessary

VDDIO/2 voltage levels for LCD 1/2 bias operation. The LCDEN bit in the LCDC register is the overall master control for the LCD Driver, however this bit is used in conjunction with the COMnEN bits to select which Port D pins are used for LCD driving. Note that the Port Control register does not need to first setup the pins as outputs to enable the LCD driver operation. If the parallel DMA function is disabled, PDMA control bit is set to ″0″, and the COMnEN bits is set to ″1″ will enable the corresponding LCD COM driving waveform to Port D pins.



LCD COM Circuit

LCDEN	COMnEN	Pin Function	O/P Level
0	X	I/O	0 or 1
1	0	I/O	0 or 1
1	1	COMn	$V_{DDIO}/2$

Output Control

LCD Bias Control

The LCD COM driver enables two kinds of selection to be provided to suit the requirement of the LCD panel which is being used. The bias resistor choice is implemented using the RSEL0 and RSEL1 bits in the LCDC register.

Bit No.	Label	R/W	Function
0	COM0EN	R/W	PD0 or COM0 selection 0: GPIO 1: COM3
1	COM1EN	R/W	PD1 or COM1 selection 0: GPIO 1: COM3
2	COM2EN	R/W	PD2 or COM2 selection 0: GPIO 1: COM3
3	COM3EN	R/W	PD3 or COM3 selection 0: GPIO 1: COM3
4	LCDEN	R/W	COM module control 0: disable 1: enable
5 6	RSEL0 RSEL1	R/W	Select SCOM typical bias current (V_{DD}=5V) 00: 25μA 01: 50μA 10: 100μA 11: 200μA
7	—	R/W	Reserved bit 0: correct level, bit must be reset to zero for correct operation 1: unpredictable operation, bit must not be set high

LCDC Register (3BH)

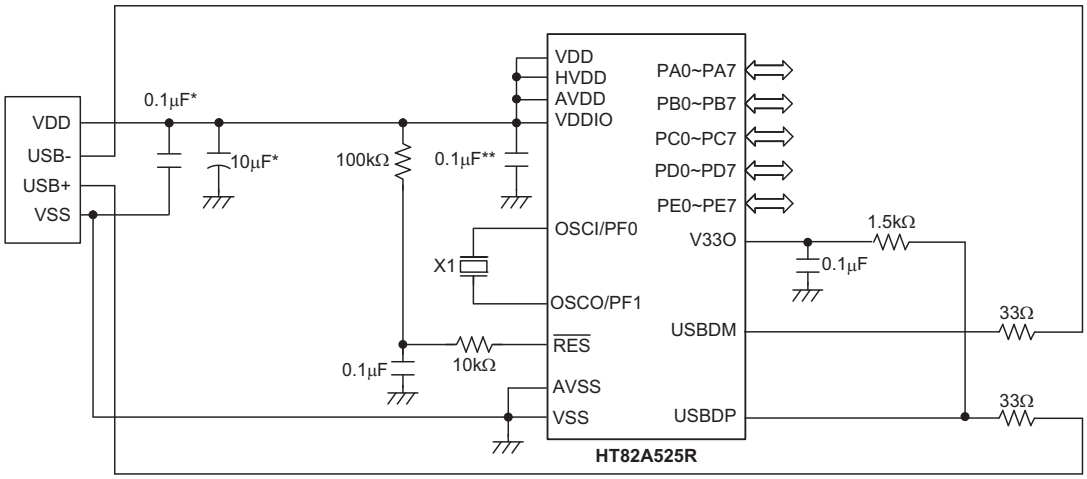
Options

The following table shows all kinds of OTP option in the microcontroller. All of the OTP options must be defined to ensure proper system functioning. The default values of the options are "0".

No.	Option
I/O Options	
1	PA0~PA7 wake-up function: enable or disable -- bit option
2	PA0~PA7 pull-high resistor function: enable or disable -- bit option
3	PB0~PB7 pull-high resistor function: enable or disable -- bit option
4	PB0~PB7 wake-up function: enable or disable -- nibble option
5	PB0~PB7 output structure option: CMOS or NMOS -- bit option
6	PC0~PC7 pull-high resistor option: enable or disable -- nibble option
7	PC0~PC7 wake-up function: enable or disable -- nibble option
8	PD0~PD7 pull-high resistor function: enable or disable -- nibble option
9	PD0~PD7 wake-up function: enable or disable -- nibble option
10	PE0~PE7 pull-high resistor function: enable or disable -- nibble option
11	PE0~PE7 wake-up function: enable or disable -- nibble option
12	PF0~PF1 pull-high resistor function: enable or disable -- nibble option
13	PF0~PF1 wake-up function: enable or disable -- nibble option
Oscillator Option	
14	OSC, OSC mode selection: external crystal or internal RC OSC
SPI Options	
15	SCLK function: enable or disable
16	FSCLK; SCLK frequency selection: 12MHz, 16MHz, 24MHz, 6MHz, 8MHz, 4MHz, 3MHz, 2MHz
17	SPI1 function: enable or disable
18	SPI1 WCOL function: enable or disable
19	SPI1 CSEN function: enable or disable
20	SPI1 Clock polarity selection: Rising or Falling edge
21	SPI1 data output mode option
22	SPI2 data output mode option
23	PCLK function: enable or disable
24	SPI2 function: enable or disable
25	SPI2 WCOL function: enable or disable
26	SPI2 CSEN function: enable or disable
27	SPI2 clock polarity selection: rising or falling edge

No.	Option
DMA Options	
28	PSYNC/VSYNC function: enable or disable
29	Parallel DMA data length option: 8bit or 5bit
30	DLEN;SPI DMA FIFO output data length option: 64 bytes, 32 bytes, 16 bytes, 8 bytes
31	HSYNC function: enable or disable
32	PCLKE clock polarity option: rising or falling edge
WDT Options	
33	WDT function: enable or disable
34	WDT clock source option: $f_{SYS}/4$ or WDTOSC
35	WDT timeout period option: $2^{13}/f_S$, $2^{14}/f_S$, $2^{15}/f_S$, $2^{16}/f_S$
36	Timer out WDT clear mode option: half WDT clear or full WDT clear
USB Options	
37	EP1, EP2, EP3 data pipe function: enable or disable
38	Endpoint 0~3 interrupt function: enable or disable
LVR Option	
39	LVR enable or disable
Other Options	
40	TBHP enable or disable
41	VSEL; LCD COM voltage option (ICE only): 3.3V or 5.0V

Application Circuits



Note: The resistance and capacitance for reset circuit should be designed in such a way as to ensure that VDD is stable and remains within a valid operating voltage range before bringing $\overline{RES}$ high.
X1 can use 6MHz or 12MHz, X1 as close OSCI & OSCO as possible.
* These capacitors should be placed close to the USB connector.
** This capacitor should be placed close to the MCU.

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	1Note	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	1Note	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	1Note	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	1Note	Z, C, AC, OV
DAA [m]	Decimal adjust ACC for Addition with result in Data Memory	1Note	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	1Note	Z
ORM A,[m]	Logical OR ACC to Data Memory	1Note	Z
XORM A,[m]	Logical XOR ACC to Data Memory	1Note	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	1Note	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	1Note	Z
DECA [m]	Decrement Data Memory with result in ACC	1	Z
DEC [m]	Decrement Data Memory	1Note	Z
Rotate			
RRA [m]	Rotate Data Memory right with result in ACC	1	None
RR [m]	Rotate Data Memory right	1Note	None
RRCA [m]	Rotate Data Memory right through Carry with result in ACC	1	C
RRC [m]	Rotate Data Memory right through Carry	1Note	C
RLA [m]	Rotate Data Memory left with result in ACC	1	None
RL [m]	Rotate Data Memory left	1Note	None
RLCA [m]	Rotate Data Memory left through Carry with result in ACC	1	C
RLC [m]	Rotate Data Memory left through Carry	1Note	C
Data Move			
MOV A,[m]	Move Data Memory to ACC	1	None
MOV [m],A	Move ACC to Data Memory	1Note	None
MOV A,x	Move immediate data to ACC	1	None

Mnemonic	Description	Cycles	Flag Affected
Bit Operation			
CLR [m].i	Clear bit of Data Memory	1Note	None
SET [m].i	Set bit of Data Memory	1Note	None
Branch			
JMP addr	Jump unconditionally	2	None
SZ [m]	Skip if Data Memory is zero	1Note	None
SZA [m]	Skip if Data Memory is zero with data movement to ACC	1note	None
SZ [m].i	Skip if bit i of Data Memory is zero	1Note	None
SNZ [m].i	Skip if bit i of Data Memory is not zero	1Note	None
SIZ [m]	Skip if increment Data Memory is zero	1Note	None
SDZ [m]	Skip if decrement Data Memory is zero	1Note	None
SIZA [m]	Skip if increment Data Memory is zero with result in ACC	1Note	None
SDZA [m]	Skip if decrement Data Memory is zero with result in ACC	1Note	None
CALL addr	Subroutine call	2	None
RET	Return from subroutine	2	None
RET A,x	Return from subroutine and load immediate data to ACC	2	None
RETI	Return from interrupt	2	None
Table Read			
TABRDC [m](4)	Read ROM code (locate by TBLP and TBHP) to data memory and TBLH	2Note	None
TABRDC [m](5)	Read ROM code (current page) to data memory and TBLH	2Note	None
TABRDL [m]	Read table (last page) to TBLH and Data Memory	2Note	None
Miscellaneous			
NOP	No operation	1	None
CLR [m]	Clear Data Memory	1Note	None
SET [m]	Set Data Memory	1Note	None
CLR WDT	Clear Watchdog Timer	1	TO, PDF
CLR WDT1	Pre-clear Watchdog Timer	1	TO, PDF
CLR WDT2	Pre-clear Watchdog Timer	1	TO, PDF
SWAP [m]	Swap nibbles of Data Memory	1Note	None
SWAPA [m]	Swap nibbles of Data Memory with result in ACC	1	None
HALT	Enter power down mode	1	TO, PDF

Note: 1. For skip instructions, if the result of the comparison involves a skip then 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. Configuration option ″TBHP option″ is enabled
5. Configuration option ″TBHP option″ is disabled

Instruction Definition

ADC A,[m]	Add Data Memory to ACC with Carry
Description	The contents of the specified Data Memory, Accumulator and the carry flag are added. The result is stored in the Accumulator.
Operation	ACC ← ACC + [m] + C
Affected flag(s)	OV, Z, AC, C
ADCM A,[m]	Add ACC to Data Memory with Carry
Description	The contents of the specified Data Memory, Accumulator and the carry flag are added. The result is stored in the specified Data Memory.
Operation	[m] ← ACC + [m] + C
Affected flag(s)	OV, Z, AC, C
ADD A,[m]	Add Data Memory to ACC
Description	The contents of the specified Data Memory and the Accumulator are added. The result is stored in the Accumulator.
Operation	ACC ← ACC + [m]
Affected flag(s)	OV, Z, AC, C
ADD A,x	Add immediate data to ACC
Description	The contents of the Accumulator and the specified immediate data are added. The result is stored in the Accumulator.
Operation	ACC ← ACC + x
Affected flag(s)	OV, Z, AC, C
ADDM A,[m]	Add ACC to Data Memory
Description	The contents of the specified Data Memory and the Accumulator are added. The result is stored in the specified Data Memory.
Operation	[m] ← ACC + [m]
Affected flag(s)	OV, Z, AC, C
AND A,[m]	Logical AND Data Memory to ACC
Description	Data in the Accumulator and the specified Data Memory perform a bitwise logical AND operation. The result is stored in the Accumulator.
Operation	ACC ← ACC ″AND″ [m]
Affected flag(s)	Z
AND A,x	Logical AND immediate data to ACC
Description	Data in the Accumulator and the specified immediate data perform a bitwise logical AND operation. The result is stored in the Accumulator.
Operation	ACC ← ACC ″AND″ x
Affected flag(s)	Z
ANDM A,[m]	Logical AND ACC to Data Memory
Description	Data in the specified Data Memory and the Accumulator perform a bitwise logical AND operation. The result is stored in the Data Memory.
Operation	[m] ← ACC ″AND″ [m]
Affected flag(s)	Z

CALL addr Subroutine call

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

Operation Stack ← Program Counter + 1
Program Counter ← addr

Affected flag(s) None

CLR [m] Clear Data Memory

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

Operation [m] ← 00H

Affected flag(s) None

CLR [m].i Clear bit of Data Memory

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

Operation [m].i ← 0

Affected flag(s) None

CLR WDT Clear Watchdog Timer

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

Operation WDT cleared
TO ← 0
PDF ← 0

Affected flag(s) TO, PDF

CLR WDT1 Pre-clear Watchdog Timer

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

Operation WDT cleared
TO ← 0
PDF ← 0

Affected flag(s) TO, PDF

CLR WDT2 Pre-clear Watchdog Timer

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

Operation WDT cleared
TO ← 0
PDF ← 0

Affected flag(s) TO, PDF

CPL [m] Complement Data Memory

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

Operation: [m] ← $\overline{[m]}$

Affected flag(s): Z

CPLA [m] Complement Data Memory with result in ACC

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

Operation: ACC ← $\overline{[m]}$

Affected flag(s): Z

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

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

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

Affected flag(s): C

DEC [m] Decrement Data Memory

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

Operation: [m] ← [m] − 1

Affected flag(s): Z

DECA [m] Decrement Data Memory with result in ACC

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

Operation: ACC ← [m] − 1

Affected flag(s): Z

HALT Enter power down mode

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

Operation:
TO ← 0
PDF ← 1

Affected flag(s): TO, PDF

INC [m]	Increment Data Memory
Description	Data in the specified Data Memory is incremented by 1.
Operation	[m] ← [m] + 1
Affected flag(s)	Z
INCA [m]	Increment Data Memory with result in ACC
Description	Data in the specified Data Memory is incremented by 1. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.
Operation	ACC ← [m] + 1
Affected flag(s)	Z
JMP addr	Jump unconditionally
Description	The contents of the Program Counter are replaced with the specified address. Program execution then continues from this new address. As this requires the insertion of a dummy instruction while the new address is loaded, it is a two cycle instruction.
Operation	Program Counter ← addr
Affected flag(s)	None
MOV A,[m]	Move Data Memory to ACC
Description	The contents of the specified Data Memory are copied to the Accumulator.
Operation	ACC ← [m]
Affected flag(s)	None
MOV A,x	Move immediate data to ACC
Description	The immediate data specified is loaded into the Accumulator.
Operation	ACC ← x
Affected flag(s)	None
MOV [m],A	Move ACC to Data Memory
Description	The contents of the Accumulator are copied to the specified Data Memory.
Operation	[m] ← ACC
Affected flag(s)	None
NOP	No operation
Description	No operation is performed. Execution continues with the next instruction.
Operation	No operation
Affected flag(s)	None
OR A,[m]	Logical OR Data Memory to ACC
Description	Data in the Accumulator and the specified Data Memory perform a bitwise logical OR operation. The result is stored in the Accumulator.
Operation	ACC ← ACC ″OR″ [m]
Affected flag(s)	Z

OR A,x	Logical OR immediate data to ACC
Description	Data in the Accumulator and the specified immediate data perform a bitwise logical OR operation. The result is stored in the Accumulator.
Operation	ACC ← ACC ″OR″ x
Affected flag(s)	Z
ORM A,[m]	Logical OR ACC to Data Memory
Description	Data in the specified Data Memory and the Accumulator perform a bitwise logical OR operation. The result is stored in the Data Memory.
Operation	[m] ← ACC ″OR″ [m]
Affected flag(s)	Z
RET	Return from subroutine
Description	The Program Counter is restored from the stack. Program execution continues at the restored address.
Operation	Program Counter ← Stack
Affected flag(s)	None
RET A,x	Return from subroutine and load immediate data to ACC
Description	The Program Counter is restored from the stack and the Accumulator loaded with the specified immediate data. Program execution continues at the restored address.
Operation	Program Counter ← Stack ACC ← x
Affected flag(s)	None
RETI	Return from interrupt
Description	The Program Counter is restored from the stack and the interrupts are re-enabled by setting the EMI bit. EMI is the master interrupt global enable bit. If an interrupt was pending when the RETI instruction is executed, the pending Interrupt routine will be processed before returning to the main program.
Operation	Program Counter ← Stack EMI ← 1
Affected flag(s)	None
RL [m]	Rotate Data Memory left
Description	The contents of the specified Data Memory are rotated left by 1 bit with bit 7 rotated into bit 0.
Operation	[m].(i+1) ← [m].i; (i = 0~6) [m].0 ← [m].7
Affected flag(s)	None
RLA [m]	Rotate Data Memory left with result in ACC
Description	The contents of the specified Data Memory are rotated left by 1 bit with bit 7 rotated into bit 0. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.
Operation	ACC.(i+1) ← [m].i; (i = 0~6) ACC.0 ← [m].7
Affected flag(s)	None

RLC [m] Rotate Data Memory left through Carry

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

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

Affected flag(s): C

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

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

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

Affected flag(s): C

RR [m] Rotate Data Memory right

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

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

Affected flag(s): None

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

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

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

Affected flag(s): None

RRC [m] Rotate Data Memory right through Carry

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

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

Affected flag(s): C

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

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

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

Affected flag(s): C

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

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

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

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

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

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

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

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

SDZ [m] Skip if decrement Data Memory is 0

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

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

Affected flag(s) None

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

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

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

Affected flag(s) None

SET [m] Set Data Memory

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

Operation [m] ← FFH

Affected flag(s) None

SET [m].i Set bit of Data Memory

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

Operation [m].i ← 1

Affected flag(s) None

SIZ [m] Skip if increment Data Memory is 0

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

Operation [m] ← [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

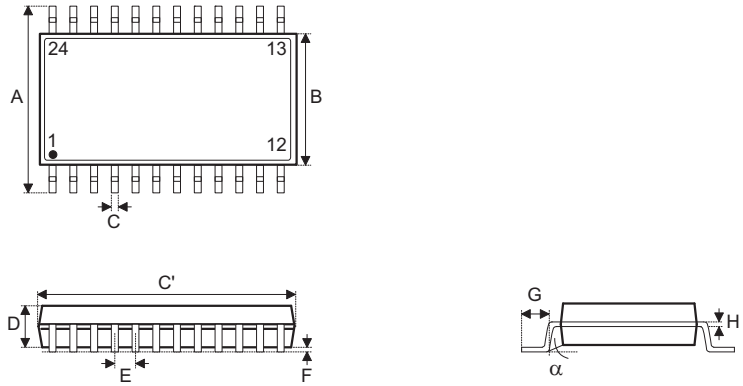
Package Information

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 information.

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

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

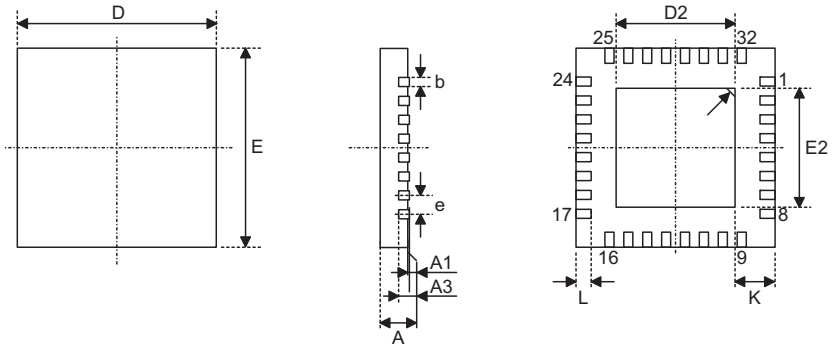
24-pin SSOP (150mil) Outline Dimensions



Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	—	0.236 BSC	—
B	—	0.154 BSC	—
C	0.008	—	0.012
C′	—	0.341 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	—	6.0 BSC	—
B	—	3.9 BSC	—
C	0.20	—	0.30
C′	—	8.66 BSC	—
D	—	—	1.75
E	—	0.635 BSC	—
F	0.10	—	0.25
G	0.41	—	1.27
H	0.10	—	0.25
α	0°	—	8°

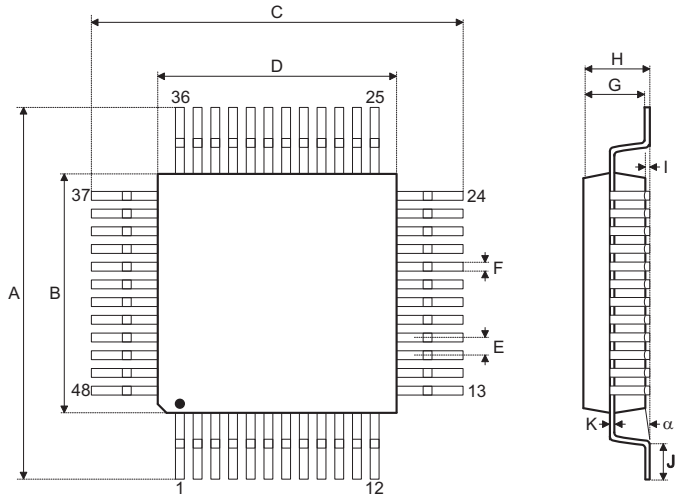
SAW Type 32-pin (5mm×5mm) QFN Outline Dimensions



Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	0.028	0.030	0.031
A1	0.000	0.001	0.002
A3	—	0.008 BSC	—
b	0.007	0.010	0.012
D	0.193	0.197	0.201
E	0.193	0.197	0.201
e	—	0.020 BSC	—
D2	0.122	0.126	0.130
E2	0.122	0.126	0.130
L	0.014	0.016	0.018
K	0.008	—	—

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	0.700	0.750	0.800
A1	0.000	0.020	0.050
A3	—	0.203 BSC	—
b	0.180	0.250	0.300
D	4.900	5.000	5.100
E	4.900	5.000	5.100
e	—	0.500 BSC	—
D2	3.100	3.200	3.30
E2	3.00	3.200	3.30
L	0.35	0.40	0.45
K	0.20	—	—

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	—	9.00 BSC	—
B	—	7.00 BSC	—
C	—	9.00 BSC	—
D	—	7.00 BSC	—
E	—	0.50 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°