HT82A523R

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

Technical Document

- Tools Information
- FAQs
- Application Note

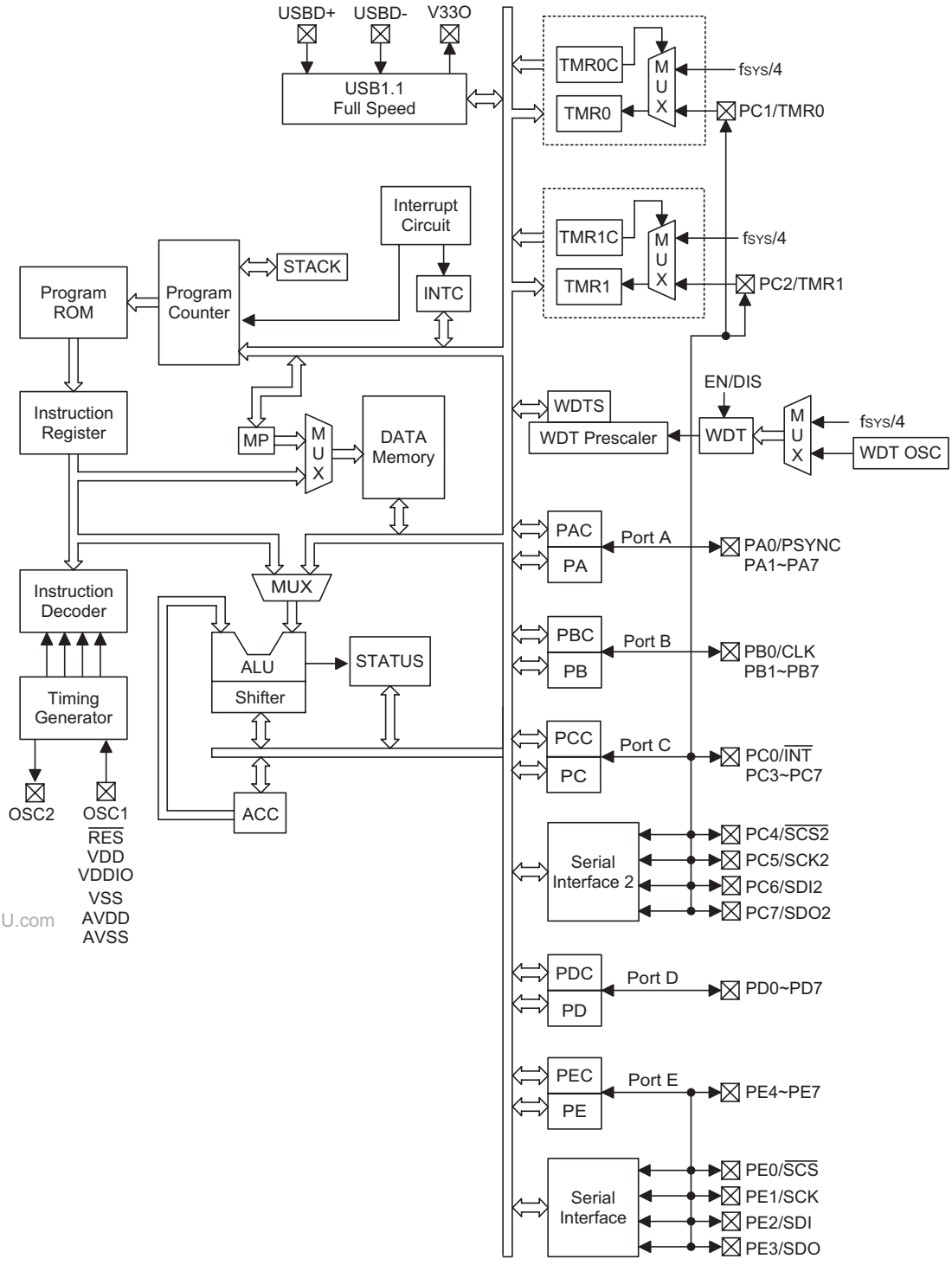
Features

- Operating voltage:
  f_{SYS}=6MHz & 12MHz: 3.3V~5.5V
- 40 bidirectional I/O lines (max.)
- One 16-bit programmable timer/event counter with overflow interrupt
- One 8-bit programmable timer/event counter with overflow interrupt
- Only crystal oscillator (6MHz or 12MHz)
- Watchdog Timer
- 4096×15 program memory ROM
- 192×8 data memory RAM
- HALT function and wake-up feature reduce power consumption
- 6-level subroutine nesting
- 2 sets of SIO (synchronous serial I/O) function
- Supports Interrupt, Control, Bulk transfer
- USB 2.0 full speed function compatible
- 4 endpoints supported (endpoint 0 included)
- Total FIFO size is 152 bytes
  (8, 8, 8, 64×2 for EP0~EP3)
- Bit manipulation instruction
- 15-bit table read instruction
- 63 powerful instructions
- All instructions in one or two machine cycles
- Low voltage reset function
- 32-pin LQFP package
  48-pin SSOP package
  52-pin QFP package

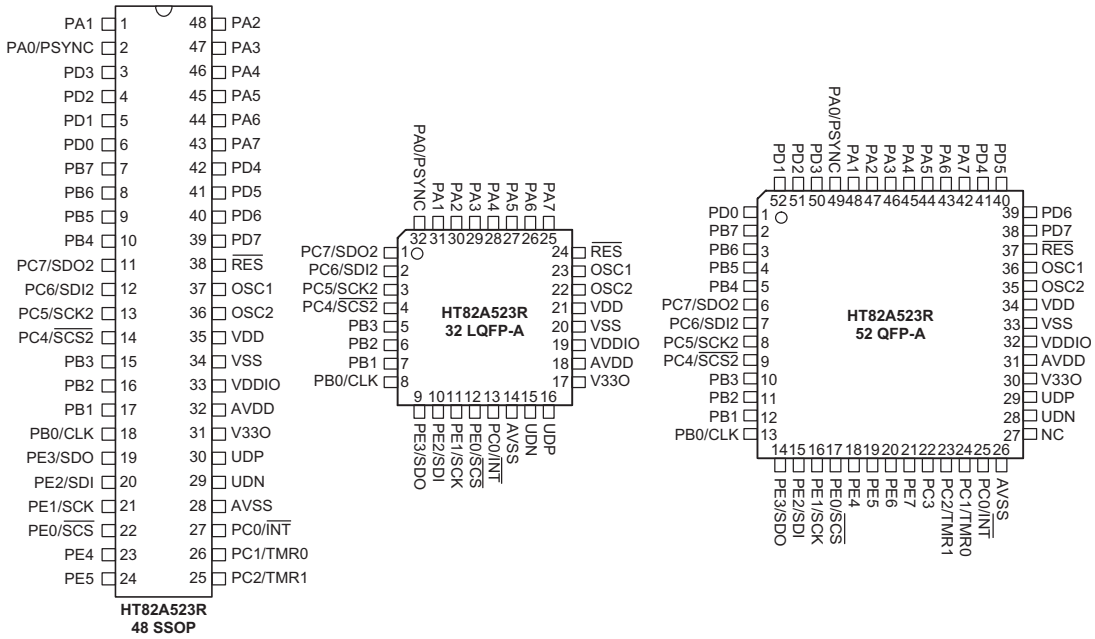
General Description

This device is an 8-bit high performance RISC-like microcontroller designed for USB product applications. It is particularly suitable for use in products such as USB mouse or keyboard and supports interface for finger-print controller. A HALT feature is included to reduce power consumption.

Block Diagram



Pin Assignment



Pin Description

Pin Name	I/O	Options	Description
PA0/PSYNC, PA1~PA7	I/O	Pull-high (bit option) Wake-up (bit option) I/O or PSYNC	Bi-directional 8-bit input/output port. Each bit can be configured as a wake-up input by ROM code option. The input or output mode is controlled by PAC (PA control register, bit option). Pull-high resistor options: PA0~PA7, bit option, Wake up options: PA0~PA7 (PA0~PA3 support falling edge wake-up only, PA4~PA7 support both rising and falling edge wake-up) The PSYNC is pin-shared with PA0 (dependent on PSYNC option)
PB0/CLK, PB1~PB7	I/O	Pull-high (bit option) CMOS/NMOS wake-up (nibble option) I/O or CLK	Bidirectional 8-bit input/output port. Software instructions determine the CMOS output or Schmitt trigger input with pull-high resistor (determined by pull-high options: bit option). CMOS/NMOS options: PB0~PB7. Wake up options: PB0~PB7 (PB0~PB7 support falling edge wake-up). The CLK is pin-shared with PB0 (dependent on CLK option) and the frequency of the CLK output is the same as system clock.
PC0/INT PC1/TMR0 PC2/TMR1 PC3 PC4/SCS2 PC5/SCK2 PC6/SDI2 PC7/SDO2	I/O	Pull-high (nibble option) wake-up (nibble option)	Bidirectional I/O lines. Software instructions determine the CMOS output or Schmitt trigger input with pull-high resistor (determined by pull high options, nibble option). The INT, TMR0, TMR1 are pin-shared with PC0, PC1, PC2 respectively. The SCS2 is pin-shared with PC4. SCS2 is serial interface chip select pin, master mode is output, slave mode is input. The SCK2 is pin-shared with PC5. SCK2 is serial interface serial clock input/output (slave/master, initial is input). The SDI2 is pin-shared with PC6. SDI2 is serial interface serial input. The SDO2 is pin-shared with PC7. SDO2 is serial interface serial output. Wake up options: PC0~PC7(PC0~PC7 support falling edge wake-up)
PD0~PD7	I/O	Pull-high (nibble option) wake-up (nibble option)	Bidirectional I/O lines. Software instructions determine the CMOS output or Schmitt trigger input with pull-high resistor (determined by pull-high options: nibble option). Wake-up options: PD0~PD7(PD0~PD7 support falling edge wake-up)

Pin Name	I/O	Options	Description
PE0/$\overline{SCS}$ PE1/SCK PE2/SDI PE3/SDO PE4~PE7	I/O	Pull-high (nibble option) wake-up (nibble option)	Bidirectional I/O lines. Software instructions determine the CMOS output or Schmitt trigger input with pull-high resistor (determined by pull-high options: nibble option). The PE0 is pin-shared with $\overline{SCS}$. $\overline{SCS}$ is a chip select pin of the Serial interface, Master mode is output, Slave mode is input. The PE1 is pin-shared with SCK. SCK is a Serial interface serial clock input/output (Initial is input). The PE2 is pin-shared with SDI. SDI is Serial interface serial input. The PE3 is pin-shared with SDO. SDO is a Serial interface serial output. Wake up options: PE0~PE7(PE0~PE7 support falling edge wake-up)
$\overline{RES}$	I	—	Schmitt trigger reset input, active low
VSS	—	—	Negative power supply, ground
AVSS	—	—	ADC negative power supply, ground
VDD	—	—	Positive power supply
VDDIO	—	—	Positive power supply, VDDIO is used for PA, PB, PC, PD and PE.
AVDD	—	—	ADC positive power supply, AVDD should be externally connected to VDD.
OSC1 OSC2	I O	—	OSC1 and OSC2 are connected to a 6MHz or 12MHz Crystal/resonator (determined by software instructions) for the internal system clock.
V33O	O	—	3.3V regulator output, can be disabled by firmware.
UDP	I/O	—	UDP is USBD+ line USB function is controlled by software control register.
UDN	I/O	—	UDN is USBD- line USB function is controlled by software control register.

Absolute Maximum Ratings

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

Note: These are stress ratings only. Stresses exceeding the range specified under "Absolute Maximum Ratings" may cause substantial damage to 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 or 12MHz	3.3	—	5.5	V
V_{DDIO}	Operating Voltage for I/O Ports	—	—	3.0	—	5.5	V
I_{DD1}	Operating Current (6MHz Crystal)	5V	No load, f_{SYS}=6MHz	—	6.5	12	mA
I_{DD2}	Operating Current (12MHz Crystal)	5V	No load, f_{SYS}=12MHz	—	7.5	16	mA
I_{STB1}	Standby Current	5V	No load, system HALT, USB suspend	—	300	500	μA
I_{STB2}	Standby Current	5V	No load, system HALT, input/output mode, set SUSP2 [22H].4	—	—	20	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_{OL}	I/O Port Sink Current	—	V_{OL}=0.33V	2	5	—	mA
I_{OH}	I/O Port Source Current	—	$V_{OH}=V_{DDIO}-0.33V$	–2	–5	—	mA
R_{PH}	Pull-high Resistance	—	V_{DDIO}=0.33V	20	60	100	kΩ
V_{LVR}	Low Voltage Reset Voltage	—	—	2.0	2.2	2.4	V
V_{33O}	3.3V Regulator Output	5V	I_{V33O}=5mA	3.0	3.3	3.6	V

A.C. Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
f_{SYS}	System Clock (Crystal OSC)	5V	—	400	—	12000	kHz
f_{TIMER}	Timer I/P Frequency (TMR0/TMR1)	5V	—	0	—	12000	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

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

Functional Description

Execution Flow

The system clock is derived from either a crystal or an RC oscillator. It is internally divided into four non-overlapping clocks. One instruction cycle consists of four system clock cycles. Instruction fetching and execution are pipelined in such a way that a fetch takes one instruction cycle while decoding and execution takes the next instruction cycle. The pipelining scheme makes it possible for each instruction to be effectively executed in a cycle. If an instruction changes the value of the program counter, two cycles are required to complete the instruction.

Program Counter – PC

The program counter (PC) controls the sequence in which the instructions stored in the program ROM are executed and its contents specify a full range of program memory.

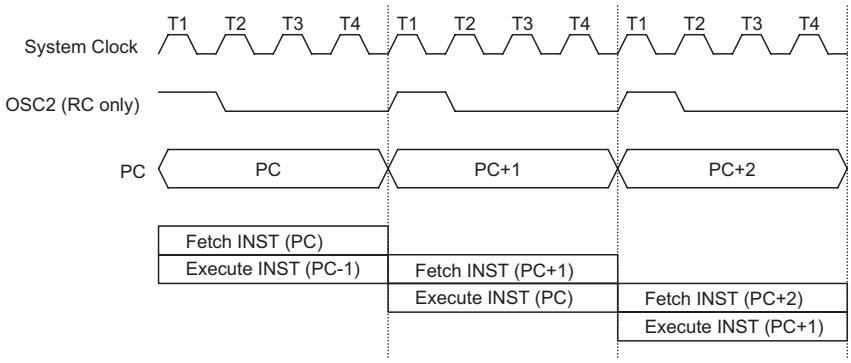
After accessing a program memory word to fetch an instruction code, the contents of the program counter are incremented by one. The program counter then points to the memory word containing the next instruction code.

When executing a jump instruction, conditional skip execution, loading PCL register, subroutine call or return from subroutine, initial reset, internal interrupt, external interrupt or return from interrupts, the PC manipulates the program transfer by loading the address corresponding to each instruction.

The conditional skip is activated by instructions. Once the condition is met, the next instruction, fetched during the current instruction execution, is discarded and a dummy cycle replaces it to get the proper instruction. Otherwise proceed to the next instruction.

The lower byte of the program counter (PCL) is a readable and write able register (06H). Moving data into the PCL performs a short jump. The destination will be within the current program ROM page.

When a control transfer takes place, an additional dummy cycle is required.



Execution Flow

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 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

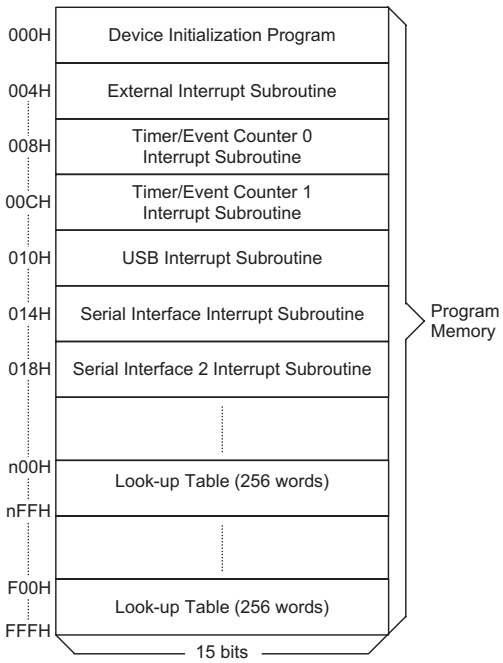
Note: *11~*0: Program counter bits
#11~#0: Instruction code bits
S11~S0: Stack register bits
@7~@0: PCL bits

Program Memory – EPROM

The program memory (EPROM) is used to store the program instructions which are to be executed. It also contains data, table, and interrupt entries, and is organized into 4096×15 bits which are addressed by the Program Counter and table pointer.

Certain locations in the ROM are reserved for special usage:

- 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 8 bits data have been received or transmitted successful from serial interface , and the related interrupts are enabled, and the stack is not full, the program begins execution at location 014H.
- Location 018H
  Location 018H is reserved for 8 bits data have been received or transmitted successful from serial interface 2, and the related interrupts are enabled, and the stack is not full, the program begins execution at location 018H.



Note: n ranges from 0 to F

Program Memory

- 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.

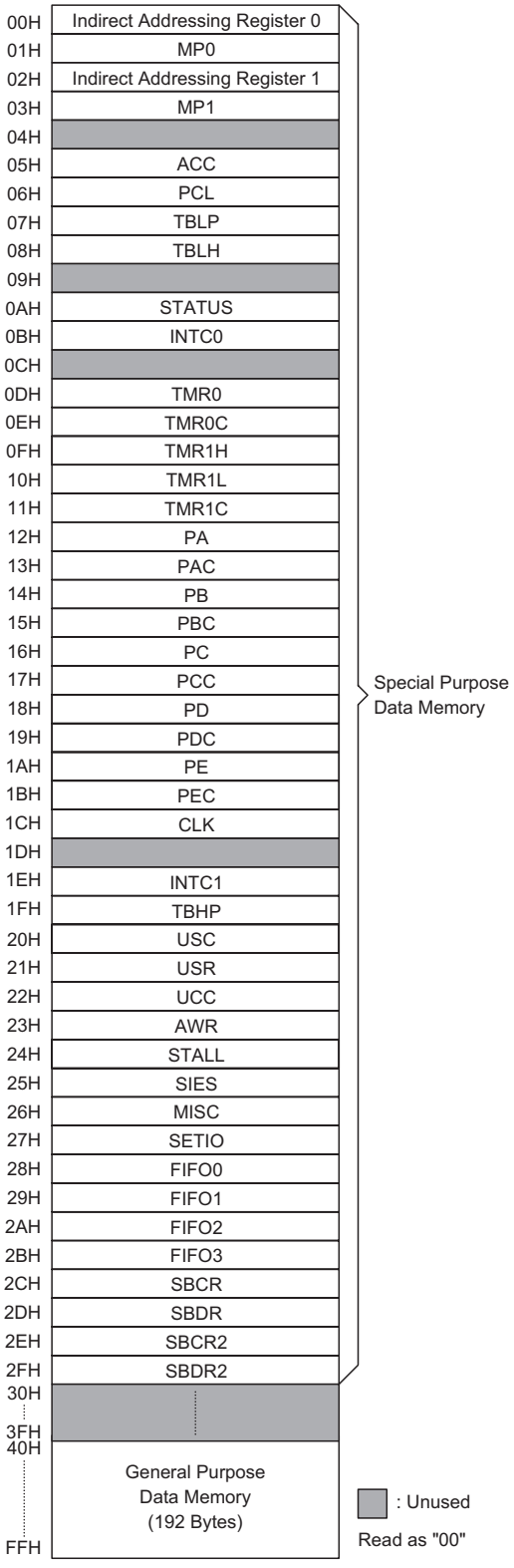
Stack Register – STACK

This is a special part of the memory which is used to save the contents of the Program Counter only. The stack is organized into 6 levels and is neither part of the data nor part of the program space, and is neither readable nor writeable. The activated level is indexed by the stack pointer (SP) and is neither readable nor writeable. At a subroutine call or an interrupt acknowledgment, the contents of the program counter are pushed onto the stack. At the end of the 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 chip reset, the SP will point to the top of the stack.

If the stack is full and a non-masked interrupt takes place, the interrupt request flag will be recorded but the acknowledgment will be inhibited. When the stack pointer is decremented (by RET or RETI), the interrupt is serviced. This feature prevents stack overflow, allowing the programmer to use the structure more easily. If the stack is full and a "CALL" is subsequently executed, stack overflow occurs and the first entry will be lost (only the most recent 6 return addresses are stored).

Data Memory – RAM

The data memory (RAM) is designed with 235×8 bits,



RAM Mapping

and is divided into two functional groups, namely; special function registers (43×8 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.

Arithmetic and Logic Unit – ALU

This circuit performs 8-bit arithmetic and logic operations. The ALU provides the following functions:

- Arithmetic operations (ADD, ADC, SUB, SBC, DAA)
- Logic operations (AND, OR, XOR, CPL)
- Rotation (RL, RR, RLC, RRC)
- Increment and Decrement (INC, DEC)
- Branch decision (SZ, SNZ, SIZ, SDZ, etc.)

The ALU not only saves the results of a data operation but also changes the status register.

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.

Bit No.	Label	Function
0	C	C is set if an operation results in a carry during an addition operation or if a borrow does not take place during a subtraction operation, otherwise C is cleared. C is also affected by a rotate through carry instruction.
1	AC	AC is set if an operation results in a carry out of the low nibbles in addition or no borrow from the high nibble into the low nibble in subtraction, otherwise AC is cleared.
2	Z	Z is set if the result of an arithmetic or logic operation is zero, otherwise Z is cleared.
3	OV	OV is set if an operation results in a carry into the highest-order bit but not a carry out of the highest-order bit, or vice versa, otherwise OV is cleared.
4	PDF	PDF is cleared by a system power-up or executing the ″CLR WDT″ instruction. PDF is set by executing the ″HALT″ instruction.
5	TO	TO is cleared by a system power-up or executing the ″CLR WDT″ or ″HALT″ instruction. TO is set by a WDT time-out.
6, 7	—	Unused bit, read as ″0″

Status (0AH) Register

Interrupts

This device provides external interrupts ($\overline{INT}$ pin interrupt, A/D Converter 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	Controls the master (global) interrupt (1= enable; 0= disable)
1	EEI	Controls the external interrupt (1= enable; 0= disable)
2	ET0I	Controls the Timer/Event Counter 0 interrupt (1= enable; 0= disable)
3	ET1I	Controls the Timer/Event Counter 1 interrupt (1= enable; 0= disable)
4	EIF	External interrupt request flag (1= active; 0= inactive)
5	T0F	Internal Timer/Event Counter 0 request flag (1= active; 0= inactive)
6	T1F	Internal Timer/Event Counter 1 request flag (1= active; 0= inactive)
7	—	Unused bit, read as ″0″

INTC0 (0BH) Register

Bit No.	Label	Function
0	EUI	Control the USB interrupt (1= enable; 0= disable)
1	ESII	Control Serial interface interrupt (1= enabled; 0= disabled)
2	ES2II	Control Serial interface 2 interrupt (1= enabled; 0= disabled)
3, 7	—	Unused bit, read as ″0″
4	USBF	USB interrupt request flag (1= active; 0= inactive)
5	SIF	Serial interface interrupt request flag (1= active; 0= inactive)
6	SI2F	Serial interface 2 interrupt request flag (1= active; 0= inactive)

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 HT82A523R, 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 HT82A523R receives a USB Suspend signal from the Host PC, the suspend line (bit0 of the USC) of the HT82A523R is set and a USB interrupt is also triggered.

Also when the HT82A523R receives a Resume signal from the Host PC, the resume line (bit3 of the USC) of the HT82A523R 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 (SIF: bit 5 of INTC1 or SI2F: bit 6 of INTC1), that is caused by received or transferred a complete 8-bit data between HT82A523R and external device. The serial interface interrupt is controlled by setting the Serial interface interrupt control bit (ESII: bit 1 of INTC1 or ESI2I: bit2 of INTC1). After the interrupt is enabled (by setting SBEN; bit 4 of SBCR 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.

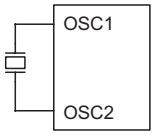
Interrupts, occurring in the interval between the rising edges of two consecutive T2 pulses, will be serviced on the latter of the two T2 pulses, if the corresponding interrupts are enabled. In the case of simultaneous requests the following table shows the priority that is applied. These can be masked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

There is an oscillator circuit in the microcontroller.



Crystal Oscillator

System Oscillator

This oscillator is designed for system clocks. The HALT mode stops the system oscillator and ignores an external signal to conserve power.

A crystal across OSC1 and OSC2 is needed to provide the feedback and phase shift required for the oscillator. No other external components are required. Instead of a crystal, a resonator can also be connected between OSC1 and OSC2 to get a frequency reference, but two external capacitors in OSC1 and OSC2 are required.

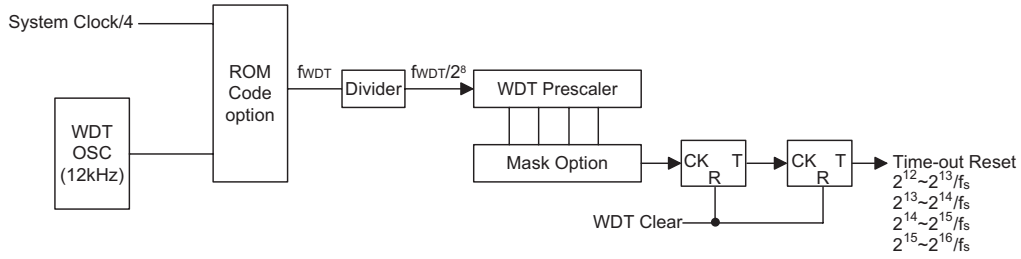
The WDT oscillator is a free running on-chip RC oscillator, and no external components are required. Even if the system enters a power down mode and the system clock is stopped, but the WDT oscillator still works. The WDT oscillator can be disabled by ROM code option to conserve power.

Watchdog Timer – WDT

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

Once an internal WDT oscillator (RC oscillator with a period of 65μs, normally at 5V) is selected, it is divided by 2^{12}~2^{16} (by option to get the WDT time-out period). The WDT time-out minimum period is about 300ms. 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}~2^{16}, the maximum time-out period is divided by 2^{15} which about 2.3s.

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. In this situation the logic can only be restarted by external logic. If the device operates in a noisy environment, using the on-chip RC oscillator (WDT OSC) is strongly recommended, since the HALT will stop the system clock.



Watchdog Timer

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. CLRWDT 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.

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 or the real time clock 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 or the real time clock).
- 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 or rising 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:

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

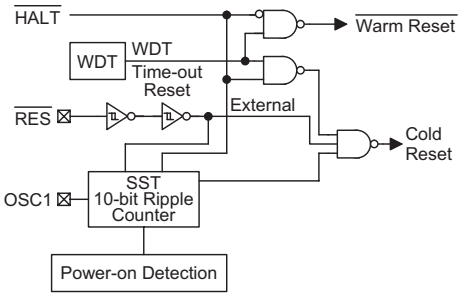
The WDT time-out during HALT differs from other chip reset conditions, for it can perform a "warm reset" that resets only the program counter and stack pointer, leaving the other circuits in their original state. Some registers remain unaffected during any other reset conditions. Most registers are reset 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	$\overline{RES}$ reset during power-up
u	u	$\overline{RES}$ reset during normal operation
0	1	$\overline{RES}$ 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.

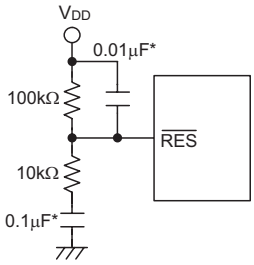
Awaking from the HALT state or system power up, an SST delay is added. An extra SST delay is added during power up period, and any wake-up from HALT may enable only the SST delay.



Reset Configuration

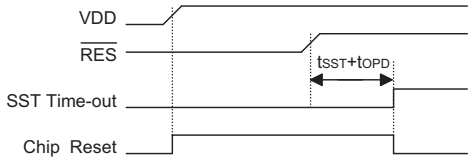
The functional unit chip reset status are shown below.

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 Circuit

Note: "*" Make the length of the wiring, which is connected to the $\overline{RES}$ 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)	$\overline{RES}$ Reset (Normal Operation)	$\overline{RES}$ 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	-xxxx xxxx	-uuu uuuu	-uuu uuuu	-uuu uuuu	-uuu uuuu	-uuu uuuu	-uuu uuuu
STATUS	--00 xxxx	--1u uuuu	--uu 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	0000 0000	0000 0000	0000 0000	uuuu uuuu	uuuu uuuu	uuuu uuuu

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)
TMR0C	00-0 1000	00-0 1000	00-0 1000	00-0 1000	uu-u uuuu	00-0 1000	00-0 1000
TMR1H	xxxx xxxx	0000 0000	0000 0000	0000 0000	uuuu uuuu	uuuu uuuu	uuuu uuuu
TMR1L	xxxx xxxx	0000 0000	0000 0000	0000 0000	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
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
CLK	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
INTC1	-000 0000	-000 0000	-000 0000	-000 0000	-uuu uuuu	-000 0000	-000 0000
TBHP	xxxx xxxx	0000 uuuu	0000 uuuu	0000 uuuu	0000 uuuu	0000 uuuu	0000 uuuu
USC	1-00 0000	u-uu uuuu	1-00 0000	1-00 0000	u-uu uuuu	u-00 0100	u-00 0100
USR	0000 0000	uuuu uuuu	0000 0000	0000 0000	uuuu 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	0100 0000	uuuu uuuu	0100 0000	0100 0000	uuuu uuuu	0100 0000	0100 0000
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
SBCR	0110 0000	0110 0000	0110 0000	0110 0000	uuuu uuuu	uuuu uuuu	uuuu uuuu
SBDR	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

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

Timer/Event Counter

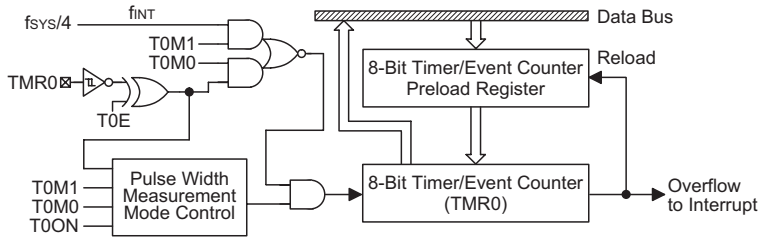
Two Timer/Event Counters (TMR0, TMR1) are implemented in the microcontroller. The Timer/Event Counter 0 contains a 8-bit programmable count-up counter and the clock may come from an external source or an internal clock source. An internal clock source comes from f_{SYS}. The Timer/Event Counter 1 contains 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.

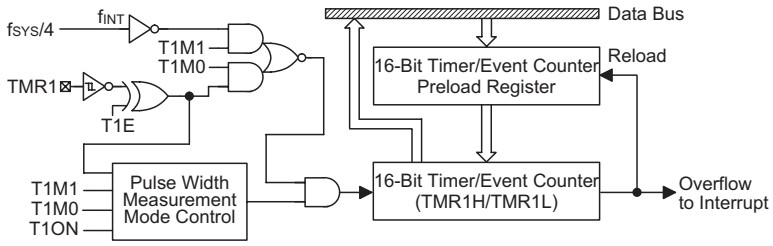
The T0M0, T0M1 (TMR0C) and T1M0, T1M1 (TMR1C) bits define the operation mode. The event count mode is used to count external events, which means that the clock source is from an external (TMR0, TMR1) pin. The timer mode functions as a normal timer with the clock source coming from the internal selected 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,



Timer/Event Counter 0



Timer/Event Counter 1

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	Defines the TMR active edge of the timer/ event counter (0=active on low to high; 1=active on high to low)
4	T0ON	Enable/disable timer counting (0=disable; 1=enable)
5	—	Unused bit, read as ″0″
6 7	T0M0 T0M1	Defines the operating mode, T0M1, T0M0: 01=Event count mode (external clock) 10=Timer mode (internal clock) 11=Pulse width measurement mode 00=Unused

TMR0C (0EH) Register

Bit No.	Label	Function
0~2, 5	—	Unused bit, read as ″0″
3	T1E	Defines the TMR active edge of the timer/ event counter (0=active on low to high; 1=active on high to low)
4	T1ON	Enable/disable timer counting (0=disable; 1=enable)
6 7	T1M0 T1M1	Defines the operating mode, T1M1, T1M0: 01=Event count mode (external clock) 10=Timer mode (internal clock) 11=Pulse width measurement mode 00=Unused

TMR1C (11H) Register

Input/Output Ports

There are 40 bidirectional input/output lines in the microcontroller, labeled from PA to PE, which are mapped to the data memory of [12H], [14H], [16H], [18H] and [1A] 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 or 1A). 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) 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 and 1BH.

PA0 is pin-shared with PSYNC signal (dependent on PSYNC option, the rising edge of PSYNC is used to synchronize the SBDR data of the serial interface which is pin-shared with port E).

PB0 is pin-shared with CLK signal (dependent on CLK 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 or 1AH) instructions.

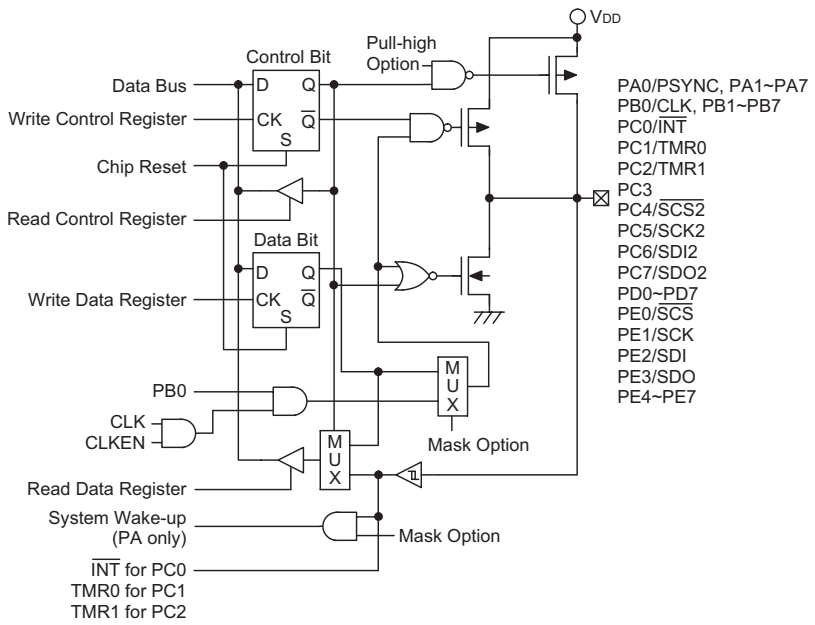
Some instructions first input data and then follow the output operations. For example, "SET [m].i", "CLR [m].i", "CPL [m]", "CPLA [m]" read the entire port states into the CPU, execute the defined operations (bit-operation), and then write the results back to the latches or the accumulator.

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.

Bit No.	Label	Function
0	CLKEN	CLKEN=1, enable CLK output. CLKEN=0, disable CLK output.
1	CLKAUTOB	CLKAUTOB=0, CLK output is disabled when FIFO is full or CLKEN is 0. CLKAUTOB=1, CLK output is controlled by CLKEN only.
2~7	—	Unused bit, read as "0"

CLK (1CH) Register



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:

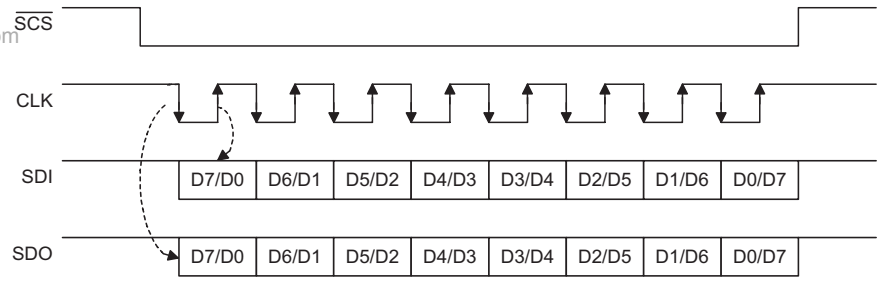
- The low voltage range (0.9V~V_{LVR}) has to be maintained for over 1ms, otherwise, the LVR will ignore it and do not perform a reset function.
- The LVR uses the "OR" function with the external $\overline{RES}$ signal to perform a chip reset.

Serial Interface

Serial interface function similar to SPI (Motorola), has four basic signals included. They are SDI (serial data input), SDO (serial data output), SCK (serial clock) and $\overline{SCS}$ (slave select pin). There are two serial interfaces. One serial interface is pin-shared with port E, and the other serial interface is pin-shared with port C.

Two registers (SBCR and SBDR) unique to serial interface provide control, status, and data storage.

- SBCR: Serial bus control register
  Bit7 (CKS) clock source selection:
  0=f_{SIO}=f_{SYS}/4
  1=f_{SIO}=f_{SYS}
  Bit6 (M1), Bit5 (M0) master/slave mode and baud rate selection
  M1, M0= 00 → Master Mode, Baud Rate= f_{SIO}
  01 → master mode, baud rate= f_{SIO}/4
  10 → master mode, baud rate= f_{SIO}/16
  11 → slave mode
- Bit4 (SBEN) → serial bus enable/disable (1/0)
  - Enable: ($\overline{SCS}$ dependent on CSEN bit)
    Disable → enable: SCK, SDI, SDO, $\overline{SCS}$= 0 (SCKB= "0") and waiting for writing data to SBDR (TXRX buffer)
    Master mode: write data to SBDR (TXRX buffer) start transmission/reception automatically
    Master mode: when data has been transferred, set TRF
    Slave mode: when an SCK (and $\overline{SCS}$ dependent on CSEN) is received, data in TXRX buffer is shifted-out and data on SDI is shifted-in.
  - Disable: SCK ($\overline{SCK}$), SDI, SDO, $\overline{SCS}$ floating
    Bit3 (MLS) → MSB or LSB (1/0) shift first control bit
    Bit2 (CSEN) → serial bus selection signal enable/disable ($\overline{SCS}$), when CSEN=0, $\overline{SCS}$ is floating.
    The $\overline{SCS}$ should be pulled high externally to avoid malfunction.
    Bit1 (WCOL) → this bit is set to 1 if data is written to SBDR (TXRX buffer) when data is transferred,
    writing will be ignored if data is written to SBDR (TXRX buffer) when data is transferred.
    Bit0 (TRF) → data transferred or data received used to generate an interrupt.
- SBDR: Serial bus data register
  Data written to SBDR → write data to TXRX buffer only
  Data read from SBDR → read from SBDR only
  Operating Mode description:
  Master transmitter: clock sending and data I/O started by writing SBDR
  Master clock sending started by writing SBDR
  Slave transmitter: data I/O started by clock received
  Slave receiver: data I/O started by clock received



	D7	D6	D5	D4	D3	D2	D1	D0	
SBCR	CKS	M1	M0	SBEN	MLS	CSEN	WCOL	TRF	SBCR : SERIAL BUS
DEFAULT	0	1	1	0	0	0	0	0	CONTROL REGISTER
SBDR	D7	D6	D5	D4	D3	D2	D1	D0	SBDR : SERIAL BUS
DEFAULT	U	U	U	U	U	U	U	U	DATA REGISTER

Note: "U" means unchanged.

Clock polarity= rising ($\overline{CLK}$) or falling (CLK): 1 or 0 (mask option)

Modes	Operations
Master	1. Select CKS and select M1, M0 = 00,01,10 2. Select CSEN, MLS (the same as the slave) 3. Set SBEN 4. Writing data to SBDR → data is stored in TXRX buffer → output CLK (and $\overline{SCS}$) signals → go to step 5 → (SIO internal operation → data stored in TXRX buffer, and SDI data is shifted into TXRX buffer → data transferred, data in TXRX buffer is latched into SBDR) 5. Check WCOL; WCOL= 1 → clear WCOL and go to step 4; WCOL= 0 → go to step 6 6. Check TRF or waiting for SBI (serial bus interrupt) 7. Read data from SBDR 8. Clear TRF 9. Go to step 4
Slave	1. CKS don't care and select M1, M0= 11 2. Select CSEN, MLS (the same as the master) 3. Set SBEN 4. Writing data to SBDR → data is stored in TXRX buffer → waiting for master clock signal (and $\overline{SCS}$): CLK → go to step 5 → (SIO internal operations → CLK ($\overline{SCS}$) received → output data in TXRX buffer and SDI data is shifted into TXRX buffer → data transferred, data in TXRX buffer is latched into SBDR) 5. Check WCOL; WCOL= 1 → clear WCOL, go to step 4; WCOL= 0 → go to step 6 6. Check TRF or wait for SBI (serial bus interrupt) 7. Read data from SBDR 8. Clear TRF 9. Go to step 4

Operation of Serial Interface

WCOL: master/slave mode, set while writing to SBDR when data is transferring (transmitting or receiving) and this writing will then be ignored. WCOL function can be enabled/disabled by mask option. WCOL is set by SIO and cleared by users.

Data transmission and reception are still working when the MCU enters the HALT mode.

CPOL is used to select the clock polarity of CLK. It is a mask option.

MLS: MSB or LSB first selection

CSEN: chip select function enable/disable, CSEN=1 → $\overline{SCS}$ signal function is active. Master should output $\overline{SCS}$ signal before CLK signal is set and slave data transferring should be disabled (or enabled) before (after) $\overline{SCS}$ signal is received. CSEN= 0, $\overline{SCS}$ signal is not needed, $\overline{SCS}$ pin (master and slave) should be floating. CSEN has 2 options: CSEN mask option is used to enable/disable software CSEN function. If CSEN mask option is disabled, the software CSEN is always disabled. If CSEN mask option is enabled, software CSEN function can be used.

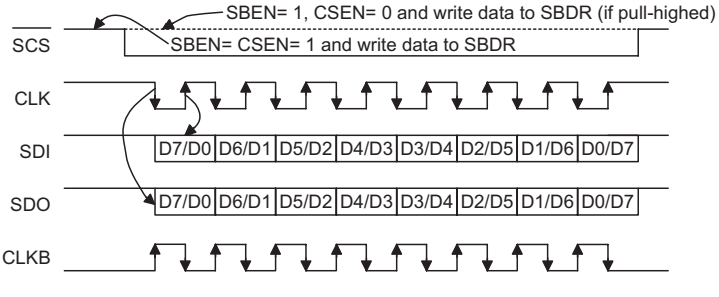
SBEN= 1 → serial bus standby; $\overline{SCS}$ (CSEN= 1) = 1; $\overline{SCS}$= floating (CSEN= 0); SDI= floating; SDO= 1; master CLK= output 1/0 (dependent on CPOL mask option), slave CLK= floating

SBEN= 0 → serial bus disabled; $\overline{SCS}$= SDI= SDO= CLK= floating

TRF is set by SIO and cleared by users. When data transfer (transmission and reception) is completed, TRF is set to generate SBI (serial bus interrupt).

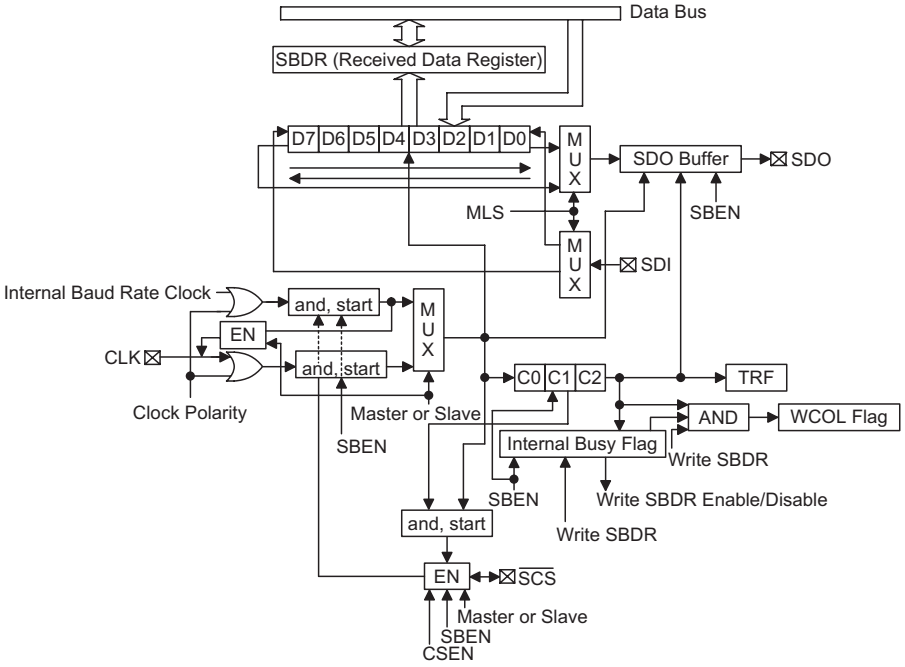
If the PSYNC option is enabled, use the SYNC signal to synchronize the SDBR (the next data which the SDBR receives will be bit7). Please refer to the following timing (only for the SPI interface which is pin-shared with port E.):

The SYNC signal is used to synchronize the SDBR when both the HT82A523R is working under slave mode and PSYNC option is enabled. Otherwise, if the HT82A523R is working under master mode, the PSYNC option is ignored and the SYNC signal is not used to synchronize the SDBR.

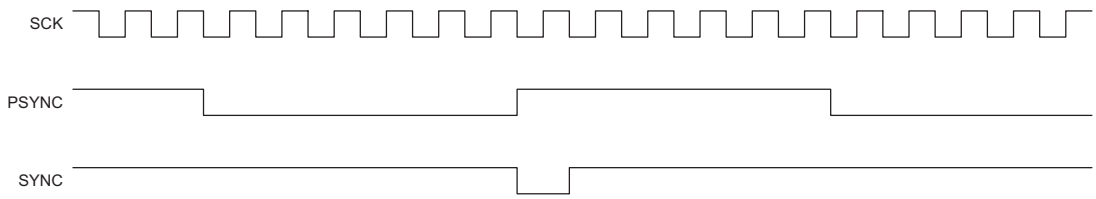


	D7	D6	D5	D4	D3	D2	D1	D0
SBCR	CKS	M1	M0	SBEN	MLS	CSEN	WCOL	TRF
Default	0	1	1	0	0	0	0	0
SBDR	D7	D6	D5	D4	D3	D2	D1	D0
Default	u	u	u	u	u	u	u	u

Note: "u" means unchanged.



WCOL: set by SIO cleared by users
CSEN: enable/disable chip selection function pin
1. master mode 1/0: with/without SCSB output function
2. slave mode 1/0: with/without SCSB input control function
SBEN : enable/disable serial bus (0: initialize all status flags)
1. When SBEN= 0, all status flags should be initialized
2. When SBEN= 0, all SIO related function pins should stay at floating state
TRF 1: data transmitted or received, 0: data is transmitting or still not received
CPOL 1/0 : clock polarity rising/falling edge : mask option

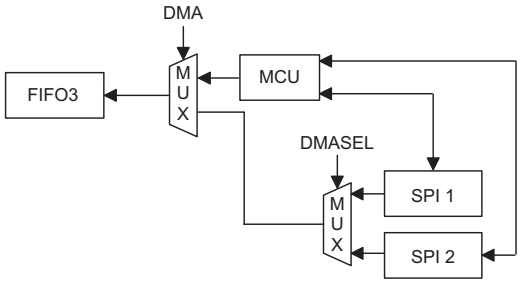


DMA Function

DMA function is enabled when DMA bit of the USB MISC register is set to 1. If DMA function is enabled, either the SBDR of the serial interface 1 (which is pin-shared with port E) or the SBDR of the serial interface 2 (which is pin-shared with port C) will be written to the FIFO3 directly.

If SBEN bit and DMA bit are set, the SPI interface which is select with DMASEL bit will send out the SCK clock during master mode. The frequency of the SCK is controlled by CKS & M1 & M0 also. The SCK clock output will be stopped automatically when FIFO3 is full and will restart automatically when FIFO3 is not full.

The SCK clock will stay at low level if CPOL is 1 and stay at high level if CPOL is 0 when it is stopped.



Suspend Wake-Up or Remote Wake-Up

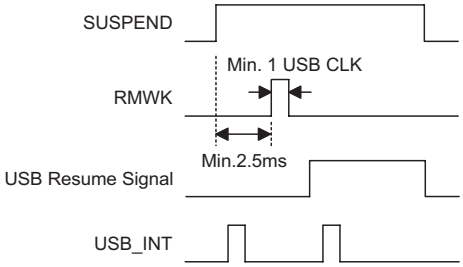
If there is no signal on the signal bus for over 3ms, the HT82A523R will go into suspend mode. The Suspend line (bit 0 of the USC) will be set to 1 and a USB interrupt is triggered to indicate that the HT82A523R should jump to suspend state to meet the 500μA USB suspend current spec.

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 400μA.

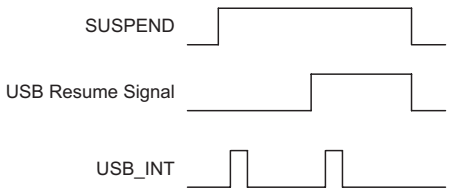
The user can also further decrease the suspend current by setting the SUSP2 (bit4 of the UCC). But if the SUSP2 is set, user should make sure not to enable the LVR OPT option, otherwise, the HT82A523R will be reset. If user set the SUSP2 (bit4 of the UCC) In the USB mode, user must set Rctrl (bit7 of the UCC) before set SUSP2 (bit4 of the UCC), otherwise, USB will disconnected.

When the resume signal is sent out by the host, the HT82A523R will wake-up the by USB interrupt and the Resume line (bit 3 of the USC) is set. In order to make the HT82A523R 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 HT82A523R, it will send a Resume signal to the device. The timing is as follow:



USB Interface

The HT82A523R has 4 Endpoints (EP0~EP3). EP0~EP2 are support Interrupt transfer, 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), 8 byte (FIFO2) and 128 byte (FIFO3), and total of 152 bytes.

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

Bit No.	Label	R/W	Function
0	SUSP	R	Read only, USB suspend indication. 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. 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. This bit is set/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. 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	0/1: Turn-off/on V33O output
5	PLL	R/W	0: Turn-on the PLL (default mode); 1: turn-of the PLL
6	—	—	Undefined bit, read as ″0″
7	URD	R/W	USB reset signal control function definition 1: USB reset signal will reset the MCU 0: USB reset signal 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 the USB interrupt will occur (if the USB interrupt is enabled, the corresponding interrupt is enabled and the stack is not full). 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	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	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	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	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	Control the endpoint 0 interrupt (1=enabled; 0=disable).
5	EEP1I	R/W	Control the endpoint 1 interrupt (1=enabled; 0=disable).
6	EEP2I	R/W	Control the endpoint 2 interrupt (1=enabled; 0=disable).
7	EEP3I	R/W	Control the endpoint 3 interrupt (1=enabled; 0=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)

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. When this bit is set to ″1″, it indicates that the USB clock is enabled. Otherwise, the USB clock is turned-off.
4	SUSP2	R/W	This bit is used to reduce power consumption in suspend mode. In normal mode, clear this bit to 0 (default) In HALT mode, set this bit to 1 to reduce power consumption.
5	f_{SYS} (24MHz)	R/W	This bit is used to define the MCU system clock to come from either the external OSC or from PLL output 24MHz clock. 0: system clock comes from OSC 1: system clock comes from PLL output 24MHz
6	SYSCLK	R/W	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″ (default).
7	RCtrl	R/W	This bit is used to control whether there is 7.5K ohm resistor between D+ and Vbus 0: no 7.5kΩ between D+ and Vbus (default) 1: has 7.5kΩ between D+ and Vbus

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 ″00H″. 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 (1/0)
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	Set by users when the related USB endpoints are stalled. They are cleared by USB reset and Setup Token event
4~7	—	—	Undefined bit, read as ″0″

STALL (24H) Definitions

Bit No.	Label	R/W	Function
0	ASET	R/W	This bit is used to configure the SIE to automatically change the device address with the value stored in the AWR register. When this bit is set to "1" by firmware, the SIE will update the device address with the value stored in the AWR register after the PC host has successfully read the data from the device by IN operation. Otherwise, when this bit is cleared to "0", the SIE will update the device address immediately after an address is written to the AWR register. So, in order to work properly, firmware has to clear this bit after the next valid SETUP token is received.
1	ERR	R/W	This bit is used to indicate there are some errors occurred during the FIFO0 is accessed. This bit is set by SIE and should be cleared by firmware.
2	OUT	R/W	This bit is used to indicate there are OUT token (except for the OUT zero length token) that have been received. The firmware clears this bit after the OUT data has been read. Also, this bit will be cleared by SIE after the next valid SETUP token is received.
3	IN	R	This bit is used to indicate that the current USB receiving signal from the PC host is IN token. (1=IN token; 0=Non IN token)
4	NAK	R	This bit is used to indicate that the SIE has transmitted a NAK signal to the host in response to the PC host IN or OUT token. (1=NAK signal; 0=Non NAK signal)
5	CRCF	R/W	Error condition failure flag include CRC, PID, no integrate token error, CRCF will be set by hardware and the CRCF need to be cleared by firmware.
6	EOT	R	Token Package active flag, low active.
7	NMI	R/W	NAK token interrupt mask flag. If this bit is set, when the device sent a NAK token to the host, interrupt will not occur. Otherwise, when this bit is cleared, and the device sent a NAK token to the host, it will enter the interrupt subroutine. The NMI is designed for all endpoints.

SIES (25H) Definitions

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	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	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	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 want to read data.
3	DMA	R/W	1: Enable SBDR of the serial interfaces (which is pin-shared with port E or port C) being written to FIFO3 directly. 0: Disable 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	DMASEL	R/W	0: Serial interface (pin-shared with port E) uses the DMA function. 1: Serial interface 2 (pin-shared with port C) uses the DMA function.
5	SETCMD	R/W	To show that the data in FIFO is setup command. This bit will last this state until next one entering the FIFO. (1=SETCMD token; 0=Non SETCMD token)
6	READY	R	To tell that the desired FIFO is ready to work. (1=Ready to work; 0=Non ready to work)
7	LEN0	R/W	To tell that host sent a 0-sized packet to MCU. This bit must be cleared by read action to corresponding FIFO. (1=Host sent a 0-sized packet)

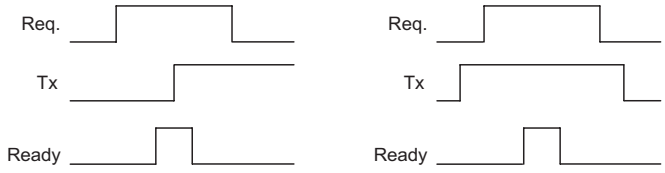
MISC (26H) Definitions

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
Read 0-sized packet sequence form FIFO0	00H→01H→delay 2μs, check 81H→read once (01H)→03H→02H
Write 0-sized packet sequence to FIFO0	02H→03H→delay 2μs, check 03H→07H→06H→00H

Read or Write FIFO Table

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



Read FIFO Timing **Write FIFO Timing**

Bit No.	Label	R/W	Function
0	DATATG*	R/W	To toggle this bit, all the DATA token will send a DATA0 first.
1	SETIO1**	R/W	Set endpoint 1 input or output pile (1/0), default input pipe (1)
2	SETIO2**	R/W	Set endpoint 2 input or output pile (1/0), default input pipe (1)
3	SETIO3**	R/W	Set endpoint 3 input or output pile (1/0), default input pipe (1)
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~FIFO03 (28H~2BH) Register, USB Endpoint Accessing Registers Definitions

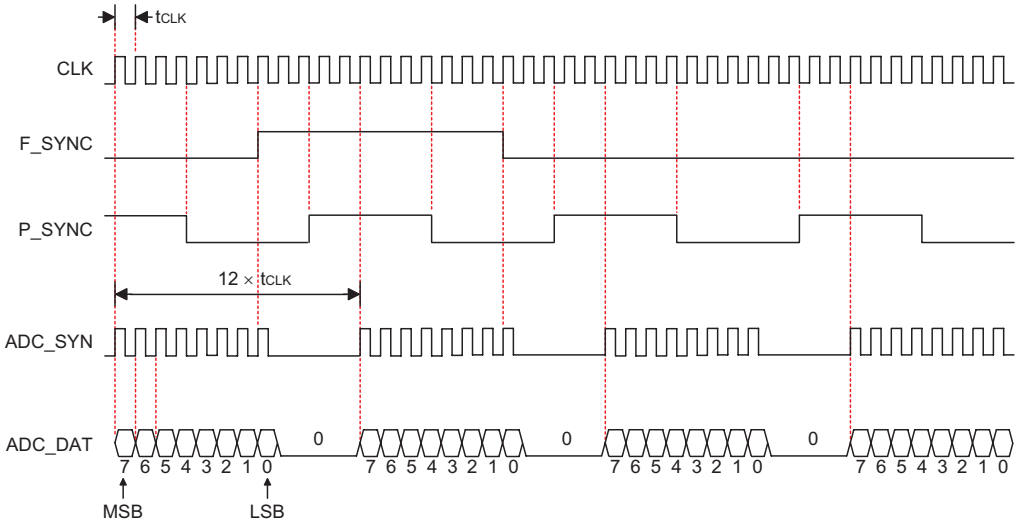
Options

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

No.	Option
1	PA0~PA7 pull-high resistor enable or disable (by bit) (default non-pull-high)
2	PB0~PB7 pull down resistor enable or disable (by bit) (default non-pull-high)
3	PC0~PC7 pull-high resistor enable or disable (by nibble) (default non-pull-high)
4	PD0~PD7 pull-high resistor enable or disable (by nibble) (default non-pull-high)
5	PE0~PE7 pull-high resistor enable or disable (by nibble) (default non-pull-high)
6	LVR enable or disable (default disable)
7	CLK enable or disable (if CLK enable then PB0 I/O port will be disable) (default disable)
8	SIO (Serial Interface) enable or disable (if SIO is enabled then PE0~PE3 I/O port will be disabled) (default disable)
9	SIO2 (Serial Interface 2) enable or disable (if SIO2 enable then PC4~PC7 I/O port will be disable) (default disable), 0=disable; 1=enable
10	SIO_ CPOL: Clock polarity 1/0: clock polarity rising or falling edge (default falling edge)
11	SIO_ CPOL2: Clock polarity 1/0 : clock polarity rising/falling edge (default falling edge ″0″) 0=falling edge 1=rising edge
12	SIO_WCOL: enable or disable (default disable)
13	SIO_WCOL2: enable or disable (default disable ″0″) 0=disable; 1=enable
14	SIO_CSEN: enable or disable, CSEN mask option is used to enable/disable (1/0) software CSEN function (default disable)
15	SIO_CSEN2: enable or disable, CSEN mask option is used to enable/disable software CSEN function (default disable ″0″) 0=disable; 1=enable
16	WDT enable or disable (default disable)
17	WDT clock source: $f_{SYS}/4$ or WDTOSC (default WDTOSC)
18	WDT timeout period: 2^{12}~$2^{13}/f_S$ (00), 2^{13}~$2^{14}/f_S$ (01), 2^{14}~$2^{15}/f_S$ (10), 2^{15}~$2^{16}/f_S$ (11) (default 2^{12}~$2^{13}/f_S$)
19	″CLRWDT″ instruction (s): 1 or 2 (default 1 instruction)
20	PA0~PA7 wake-up enable or disable (by bit) (default disable)
21	PB0~PB7 wake-up enabled or disabled (by nibble) (default disable) 0= non-wakeup; 1=wake-up
22	PB0~PB7 output structures: CMOS/NMOS (by bit) (default NMOS ″0″) 0=NMOS; 1=CMOS
23	PC0~PC7 wake-up enabled/disabled (by nibble) (default disable ″0″) 0=non-wakeup; 1=wakeup
24	PD0~PD7 wake-up enabled/disabled (by nibble) (default disable ″0″) 0=non-wakeup; 1=wakeup
25	PE0~PE7 wake-up enabled/disabled (by nibble) (default disable ″0″) 0=non-wakeup; 1=wakeup
26*	EP1~EP3 Data pipe enable: EP1, EP2, EP3 enable or disable. (default is enable)
27	PSYNC Enable/Disable (if PSYNC Enable then PA0 I/O port will be disable and the rising edge of PSYNC is used to synchronize the SBDR data) (default disable)
	Table High Byte Pointer for Current Table Read (TBHP) 0=disable; 1=enable

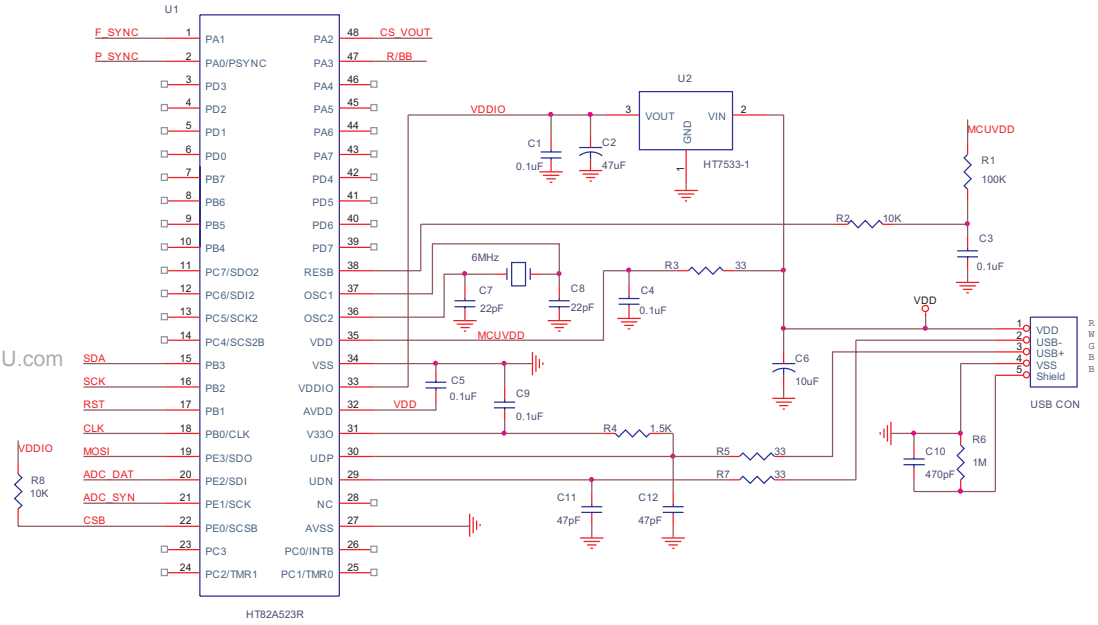
Note: *: The purpose of this option is to enable the endpoint that will be used, and disable the endpoint that will not be used (USB chapter 8 will test this function).

Timing Diagram



Application Circuits

Crystal or Ceramic Resonator for Multiple I/O Applications.



Note: The resistor and capacitor reset circuit should be designed in such a way as to ensure that the VDD is stable and remains within a valid operating voltage range before the $\overline{\text{RES}}$ line is brought high.

X1 can be a 6MHz or 12MHz crystal located as close to the OSC1 and OSC2 pins as possible

The 22pF capacitors are only applied if ceramic resonators are used

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

Mnemonic	Description	Cycles	Flag Affected
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
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]	Read table (current page) to TBLH and Data Memory	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.

Instruction Definition

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

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

Operation: ACC ← ACC + [m] + C

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

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

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

Operation: [m] ← ACC + [m] + C

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

ADD A,[m] Add Data Memory to ACC

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

Operation: ACC ← ACC + [m]

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

ADD A,x Add immediate data to ACC

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

Operation: ACC ← ACC + x

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

ADDM A,[m] Add ACC to Data Memory

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

Operation: [m] ← ACC + [m]

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

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

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

Operation: ACC ← ACC ″AND″ [m]

Affected flag(s): Z

AND A,x Logical AND immediate data to ACC

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

Operation: ACC ← ACC ″AND″ x

Affected flag(s): Z

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

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

Operation: [m] ← ACC ″AND″ [m]

Affected flag(s): Z

CALL addr	Subroutine call
Description	Unconditionally calls a subroutine at the specified address. The Program Counter then increments by 1 to obtain the address of the next instruction which is then pushed onto the stack. The specified address is then loaded and the program continues execution from this new address. As this instruction requires an additional operation, it is a two cycle instruction.
Operation	Stack ← Program Counter + 1 Program Counter ← addr
Affected flag(s)	None
CLR [m]	Clear Data Memory
Description	Each bit of the specified Data Memory is cleared to 0.
Operation	[m] ← 00H
Affected flag(s)	None
CLR [m].i	Clear bit of Data Memory
Description	Bit i of the specified Data Memory is cleared to 0.
Operation	[m].i ← 0
Affected flag(s)	None
CLR WDT	Clear Watchdog Timer
Description	The TO, PDF flags and the WDT are all cleared.
Operation	WDT cleared TO ← 0 PDF ← 0
Affected flag(s)	TO, PDF
CLR WDT1	Pre-clear Watchdog Timer
Description	The TO, PDF flags and the WDT are all cleared. Note that this instruction works in conjunction with CLR WDT2 and must be executed alternately with CLR WDT2 to have effect. Repetitively executing this instruction without alternately executing CLR WDT2 will have no effect.
Operation	WDT cleared TO ← 0 PDF ← 0
Affected flag(s)	TO, PDF
CLR WDT2	Pre-clear Watchdog Timer
Description	The TO, PDF flags and the WDT are all cleared. Note that this instruction works in conjunction with CLR WDT1 and must be executed alternately with CLR WDT1 to have effect. Repetitively executing this instruction without alternately executing CLR WDT1 will have no effect.
Operation	WDT cleared TO ← 0 PDF ← 0
Affected flag(s)	TO, PDF

CPL [m] Complement Data Memory

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

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

Affected flag(s) Z

CPLA [m] Complement Data Memory with result in ACC

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

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

Affected flag(s) Z

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

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

Operation
$[m] \leftarrow ACC + 00H$ or
$[m] \leftarrow ACC + 06H$ or
$[m] \leftarrow ACC + 60H$ or
$[m] \leftarrow ACC + 66H$

Affected flag(s) C

DEC [m] Decrement Data Memory

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

Operation $[m] \leftarrow [m] - 1$

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 $ACC \leftarrow [m] - 1$

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
$TO \leftarrow 0$
$PDF \leftarrow 1$

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 $ACC \leftarrow ACC - [m] - \overline{C}$

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 $[m] \leftarrow ACC - [m] - \overline{C}$

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 $[m] \leftarrow [m] - 1$
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 $ACC \leftarrow [m] - 1$
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 $[m] \leftarrow FFH$

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 $[m].i \leftarrow 1$

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

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

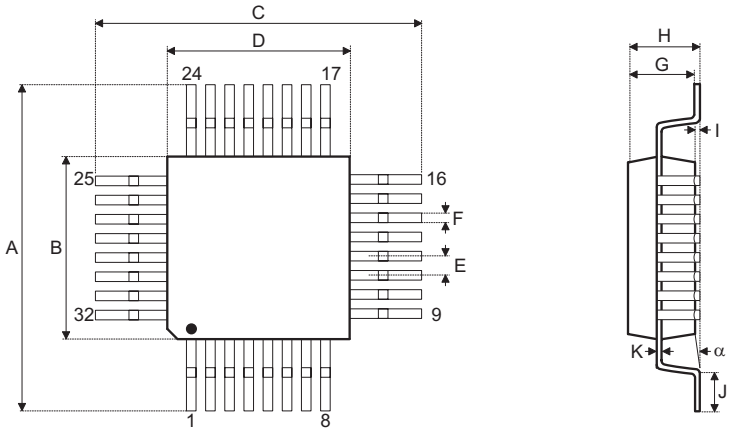
XOR A,[m]	Logical XOR Data Memory to ACC
Description	Data in the Accumulator and the specified Data Memory perform a bitwise logical XOR operation. The result is stored in the Accumulator.
Operation	ACC ← ACC ″XOR″ [m]
Affected flag(s)	Z

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

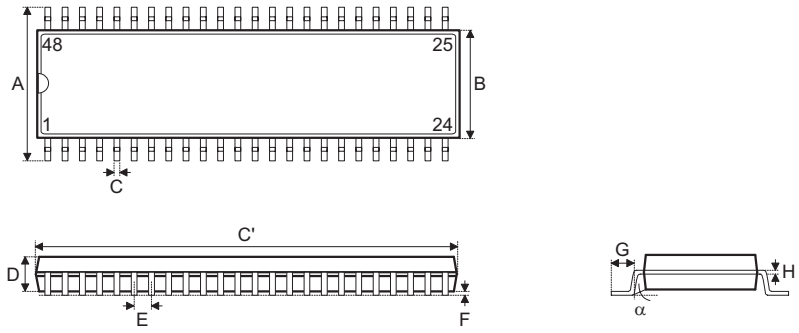
Package Information

32-pin LQFP (7×7) Outline Dimensions



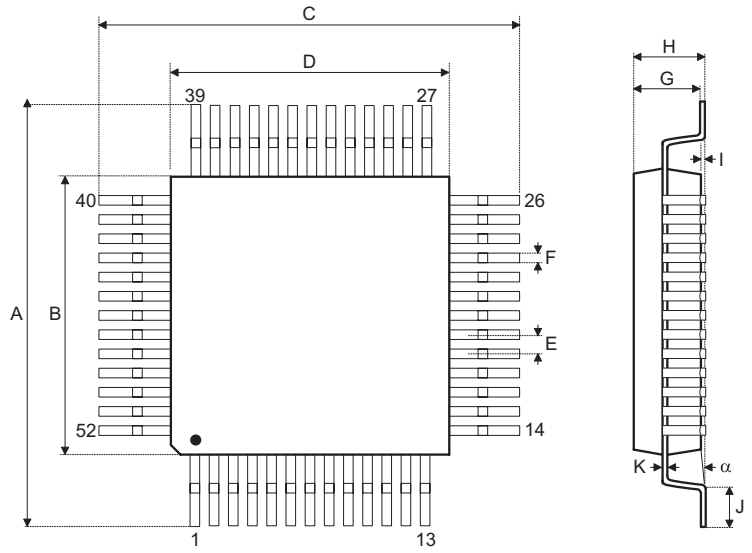
Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	8.9	—	9.1
B	6.9	—	7.1
C	8.9	—	9.1
D	6.9	—	7.1
E	—	0.8	—
F	—	0.35	—
G	1.35	—	1.45
H	—	—	1.6
I	—	0.1	—
J	0.45	—	0.75
K	0.1	—	0.2
α	0°	—	7°

48-pin SSOP (300mil) Outline Dimensions



Symbol	Dimensions in mil		
	Min.	Nom.	Max.
A	395	—	420
B	291	—	299
C	8	—	12
C′	613	—	637
D	85	—	99
E	—	25	—
F	4	—	10
G	25	—	35
H	4	—	12
α	0°	—	8°

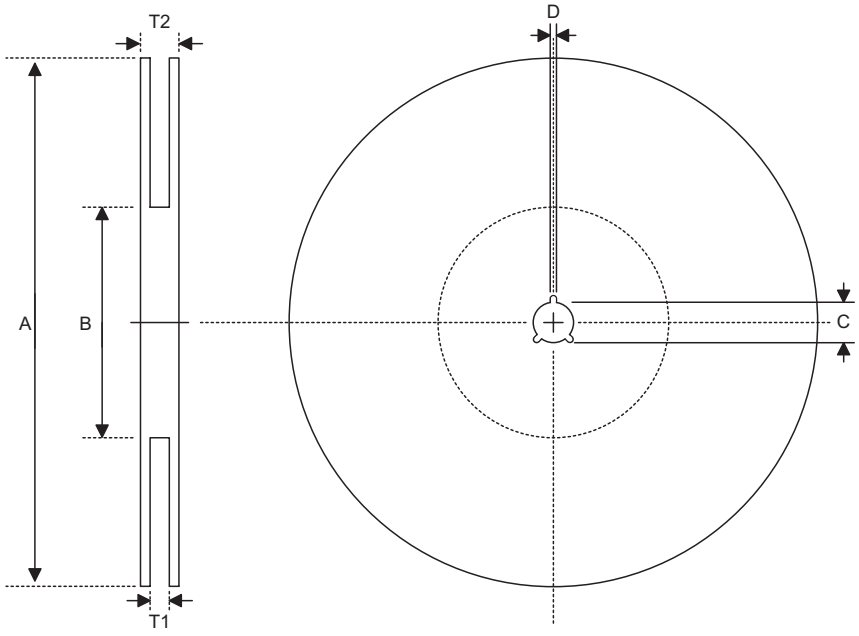
52-pin QFP (14×14) Outline Dimensions



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

Product Tape and Reel Specifications

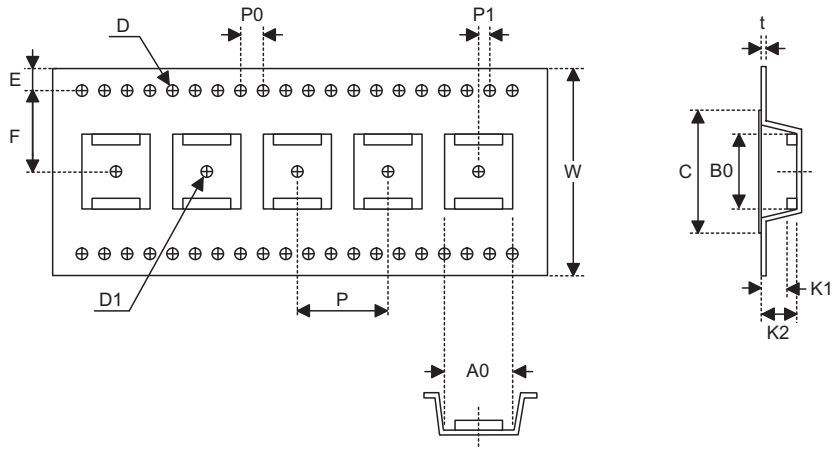
Reel Dimensions



SSOP 48W

Symbol	Description	Dimensions in mm
A	Reel Outer Diameter	330±1
B	Reel Inner Diameter	100±0.1
C	Spindle Hole Diameter	13+0.5 −0.2
D	Key Slit Width	2±0.5
T1	Space Between Flange	32.2+0.3 −0.2
T2	Reel Thickness	38.2±0.2

Carrier Tape Dimensions



SSOP 48W

Symbol	Description	Dimensions in mm
W	Carrier Tape Width	32±0.3
P	Cavity Pitch	16±0.1
E	Perforation Position	1.75±0.1
F	Cavity to Perforation (Width Direction)	14.2±0.1
D	Perforation Diameter	2 Min.
D1	Cavity Hole Diameter	1.5+0.25
P0	Perforation Pitch	4±0.1
P1	Cavity to Perforation (Length Direction)	2±0.1
A0	Cavity Length	12±0.1
B0	Cavity Width	16.2±0.1
K1	Cavity Depth	2.4±0.1
K2	Cavity Depth	3.2±0.1
t	Carrier Tape Thickness	0.35±0.05
C	Cover Tape Width	25.5

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

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

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

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

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

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

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