HT82A821R
Basic USB Speaker OTP MCU

Features

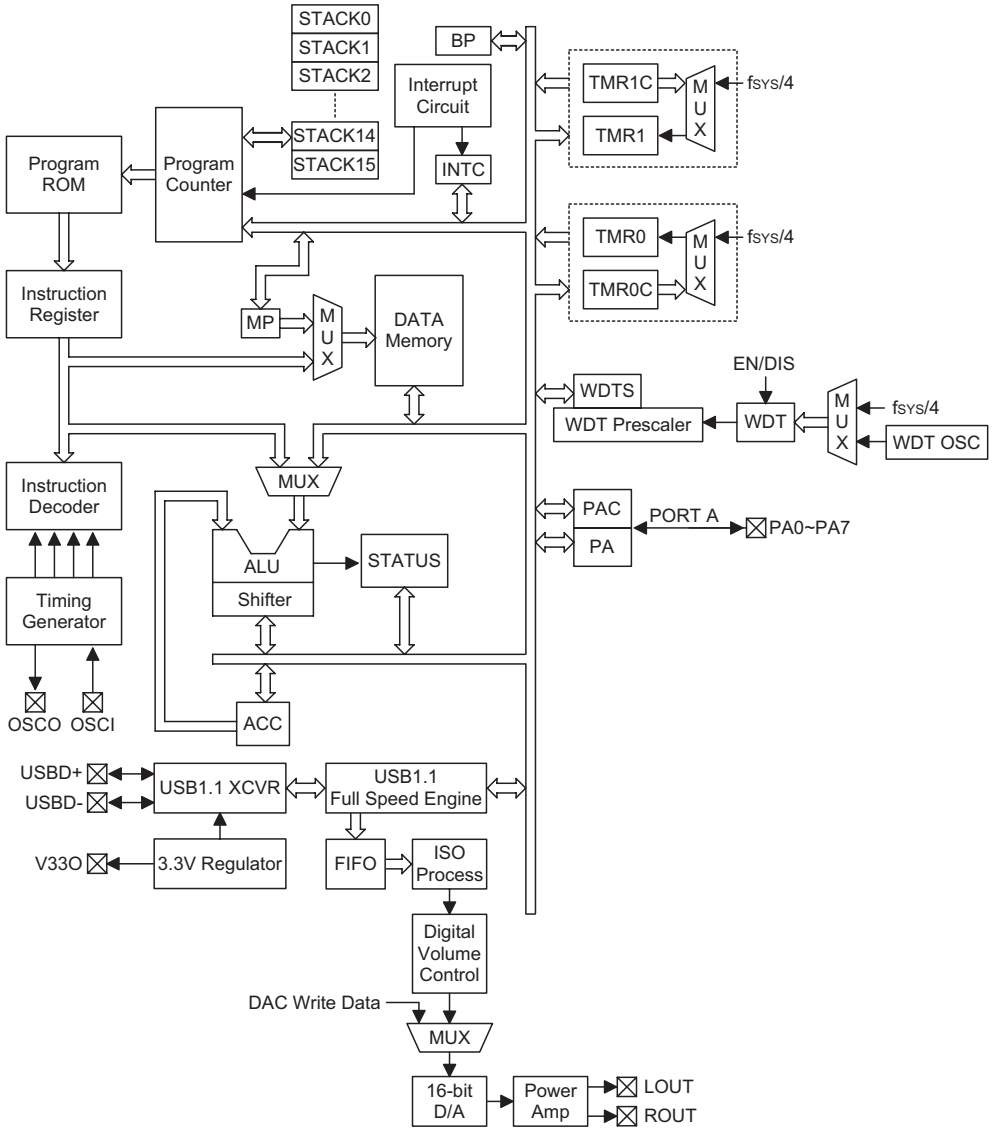
- USB 2.0 full speed compatible
- USB spec v1.1 full speed operation and USB audio device class spec v1.0
- Operating voltage: f_{SYS}= 6MHz/12MHz: 4.0V~5.5V
- Low voltage reset function (3.0V±0.3V)
- High-performance 48kHz sampling rate for audio playback
- Embedded class AB power amplifier for speaker driving
- Embedded High Performance 16 bit audio DAC
- Support digital volume control
- HID support which can remote control of playback volume/mute
- 3 endpoints supported (endpoint 0 included)
- Support 1 Control , 1 Interrupt , 1 Isochronous transfer
- Total FIFO size are 400 byte (8, 8, 384 for EP0~EP2)
- 2048×15 program memory ROM
- 192×8 MCU type data memory RAM (Bank0)
- HALT function and wake-up feature reduce power consumption
- 8 bidirectional I/O lines (max.)
- Two 16-bit programmable timer/event counter and overflow interrupts
- Watchdog Timer
- 16-level subroutine nesting
- Bit manipulation instruction
- 15-bit table read instruction
- 63 powerful instructions
- All instructions in one or two machine cycles
- 24-pin SSOP (150mil), 24-pin SOP (300mil) package

General Description

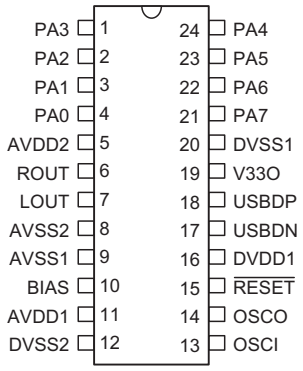
This HT82A821R is an 8-bit high performance RISC-like microcontroller designed for USB Speaker product applications. The HT82A821R combines a 16-bit DAC, USB transceiver, SIE (Serial Interface Engine), audio class processing unit, FIFO, 8-bit MCU into a single chip. The DAC in the HT82A821R is operating at the 48kHz sampling rate. The HT82A821R has a digital programmable gain amplifier. The gain range is from −32dB to +6dB.

The HT82A821R has a Human Interface Device function that allows a user to control the playback volume at the device side. The HT82A821R also can mute the analog output signal by the operation of HID buttons.

Block Diagram



Pin Assignment



HT82A821R
– 24 SSOP-A/SOP-A

Pin Description

Pin No.	Pin Name	I/O	Description
4~1, 24~21	PA0~PA7	I/O	Bidirectional 8-bit input/output port. Each bit can be configured as wake-up input by mask option. Software instructions determine the CMOS output or Schmitt trigger input with or without pull-high (by mask option).
5	AVDD2	—	Audio power amplifier positive power supply.
6	ROUT	O	Right driver analog output
7	LOUT	O	Left driver analog output
8	AVSS2	—	Audio power amplifier negative power supply, ground
9	AVSS1	—	Audio DAC negative power supply, ground
10	BIAS	O	Connect a capacitor to ground to increase half-supply stability
11	AVDD1	—	Audio DAC positive power supply
12	DVSS2	—	Negative digital & I/O power supply, ground
13 14	OSCI OSCO	I O	OSCI, OSCO are connected to an 6MHz or 12MHz crystal/resonator (determined by software instructions) for the internal system clock
15	$\overline{\text{RESET}}$	I	Schmitt trigger reset input, active low
16	DVDD1	—	Positive digital power supply
17	USBDN	I/O	USBDN is USBD- line USB function is controlled by software control register
18	USBDP	I/O	USBDP is USBD+ line USB function is controlled by software control register
19	V33O	O	3.3V regulator output
20	DVSS1	—	Negative digital power supply, ground

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	5V	—	4	5	5.5	V
I_{DD}	Operating Current	5V	No load, f_{SYS}=12MHz	—	9	—	mA
I_{STB1}	Standby Current	5V	No load, system HALT, USB suspend	—	3	—	μA
I_{STB2}	Standby Current	5V	No load, system HALT, USB transceiver and 3.3V regulator on	—	146	—	μA
V_{IL1}	Input Low Voltage for I/O Ports	5V	—	0	—	$0.3V_{DD}$	V
V_{IH1}	Input High Voltage for I/O Ports	5V	—	$0.7V_{DD}$	—	V_{DD}	V
V_{IL2}	Input Low Voltage ($\overline{RESET}$)	5V	—	0	—	$0.4V_{DD}$	V
V_{IH2}	Input High Voltage ($\overline{RESET}$)	5V	—	$0.9V_{DD}$	—	V_{DD}	V
I_{OL}	I/O Port Sink Current	5V	$V_{OL}=0.1V_{DD}$	—	5	—	mA
I_{OH}	I/O Port Source Current	5V	$V_{OH}=0.7V_{DD}$	—	−5	—	mA
R_{PH}	Pull-high Resistance	5V	—	30	50	90	kΩ
V_{LVR}	Low Voltage Reset	5V	—	2.7	3	3.3	V
V_{V33O}	3.3V Regulator Output	5V	I_{V33O}=−5mA	3	3.3	3.6	V
DAC+Power Amp: Test condition: Measurement bandwidth 20Hz to 20kHz, f_S= 48kHz. Line output series capacitor with 220μF.							
THD+N	THD+N[Note1]	5V	4Ω load	—	−30	—	dB
			8Ω load	—	−35	—	
SNR	Signal to Noise Ratio[Note1]	5V	4Ω load	—	81	—	dB
			8Ω load	—	82	—	
DR	Dynamic Range	5V	4Ω load	—	87	—	dB
			8Ω load	—	88	—	
P_{OUT}	Output Power	5V	4Ω load, THD=10%	—	400	—	mW/ch
			8Ω load, THD=10%	—	200	—	

Note: 1. Sine wave input at 1kHz, −6dB

A.C. Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
f_{SYS}	System Clock (Crystal OSC)	5V	—	0.4	—	12	MHz
t_{WDTOSC}	Watchdog Oscillator Period	5V	—	—	100	—	μs
t_{RES}	RESET Input Pulse Width	—	—	1	—	—	μs
t_{SST}	System Start-up Timer Period	—	—	—	1024	—	t_{SYS}
t_{INT}	Interrupt Pulse Width	—	—	1	—	—	μs

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

Functional Description

Execution Flow

The system clock for the micro-controller is from a crystal oscillator. The system clock 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 an instruction cycle while decoding and execution takes the next instruction cycle. However, the pipelining scheme causes each instruction to be effectively executed in a cycle. If an instruction changes 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.

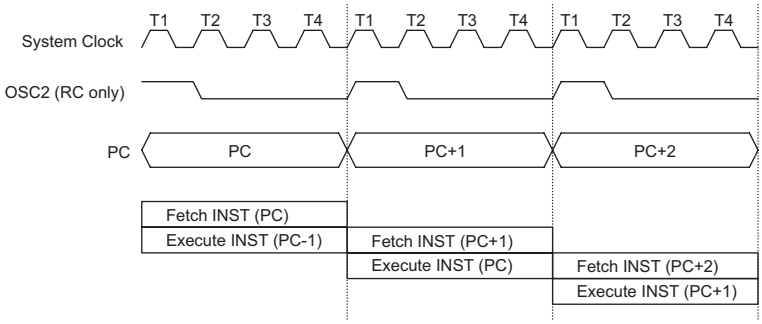
After accessing a program memory word to fetch an instruction code, the contents of the program counter are incremented by one. The program counter then points to the memory word containing the next instruction code.

When executing a jump instruction, conditional skip execution, loading to the PCL register, performing a 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 with the next instruction.

The lower byte of the program counter (PCL) is a readable and writeable 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										
	*10	*9	*8	*7	*6	*5	*4	*3	*2	*1	*0
Initial Reset	0	0	0	0	0	0	0	0	0	0	0
USB Interrupt	0	0	0	0	0	0	0	0	1	0	0
Timer/Event Counter 0 Overflow	0	0	0	0	0	0	0	1	0	0	0
Timer/Event Counter 1 Overflow	0	0	0	0	0	0	0	1	1	0	0
Skip	Program Counter+2										
Loading PCL	*10	*9	*8	@7	@6	@5	@4	@3	@2	@1	@0
Jump, Call Branch	#10	#9	#8	#7	#6	#5	#4	#3	#2	#1	#0
Return from Subroutine	S10	S9	S8	S7	S6	S5	S4	S3	S2	S1	S0

Program Counter

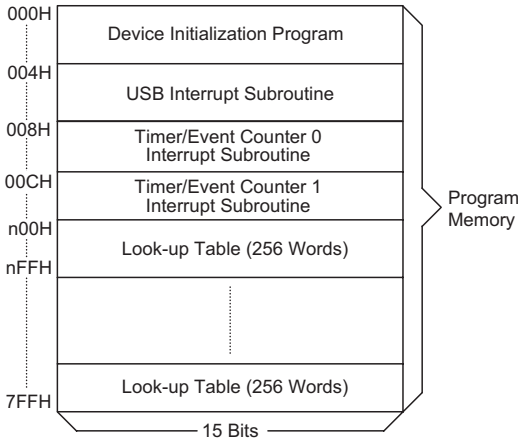
Note: *10~*0: Program counter bits
#10~#0: Instruction code bits
S10~S0: Stack register bits
@7~@0: PCL bits

Program Memory – PROM

The program memory is used to store the program instructions which are to be executed. It also contains data, table, and interrupt entries, and is organized into 2048×15 bits, addressed by the program counter and table pointer.

Certain locations in the program memory are reserved for special usage:

- Location 000H
  This area is reserved for program initialization. After a chip reset, the program always begins execution at location 000H.
- Location 004H
  This area 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 004H.
- Location 008H
  This area 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
  This location is reserved for the Timer/Event Counter 1 interrupt service program. If a timer interrupt results from a Timer/Event Counter 1 overflow, and the interrupt is enabled and the stack is not full, the program begins execution at location 00CH.



Note: n ranges from 0 to 7

Program Memory

- Table location
  Any location in the program memory can be used as look-up tables. There are three method to read the ROM data by 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).
  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, and the remaining 1-bit words are read as "0". The Table Higher-order byte register (TBLH) is read only. 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 (If the OTP option TBHP is disabled, the value in TBHP has no effect). The TBLH is read only and cannot be restored. 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 are likely to be changed by the table read instruction used in the ISR. Errors can occur. In other words, using the table read instruction in the main routine and 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.

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 16 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 interrupt acknowledge signal, the contents of the program counter are pushed onto the stack. At the end of a subroutine or an interrupt routine,

Instruction	Table Location										
	*10	*9	*8	*7	*6	*5	*4	*3	*2	*1	*0
TABRDC [m]	P10	P9	P8	@7	@6	@5	@4	@3	@2	@1	@0
TABRDL [m]	1	1	1	@7	@6	@5	@4	@3	@2	@1	@0

Table Location

Note: *10~*0: Table location bits
@7~@0: TBLP bits
P10~P8: Current program counter bits when TBHP is disabled
TBHP register bit2~bit0 when TBHP is enabled

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 acknowledge signal will be inhibited. When the stack pointer is decremented (by RET or RETI), the interrupt will be serviced. This feature prevents stack overflow allowing the programmer to use the structure more easily. In a similar case, 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 16 return addresses are stored).

Data Memory – RAM

The data memory (RAM) is designed with 192×8 bits. The data memory is divided into two functional groups: namely; special function registers 54×8 bits and general purpose data memory, Bank0: 192×8 bits. Most are read/write, but some are read only.

The special function registers include the indirect addressing registers (R0;00H, R1;02H), Bank register (BP, 04H), Timer/Event Counter 0 higher order byte register (TMR0H;0CH), Timer/Event Counter 0 lower order byte register (TMR0L;0DH), Timer/Event Counter 0 control register (TMR0C;0EH), Timer/Event Counter 1 higher order byte register (TMR1H;0FH), Timer/Event Counter 1 lower order byte register (TMR1L;10H), Timer/Event Counter 1 control register (TMR1C;11H), program counter lower-order byte register (PCL;06H), memory pointer registers (MP0;01H, MP1;03H), accumulator (ACC;05H), table pointer (TBLP;07H, TBHP;1FH), table higher-order byte register (TBLH;08H), status register (STATUS;0AH), interrupt control register0 (INTC0;0BH), Watchdog Timer option setting register (WDTS;09H), I/O registers (PA;12H), I/O control registers (PAC;13H). Digital Volume Control Register (USVC;1CH). USB status and control register (USC;20H), USB endpoint interrupt status register (USR;21H), system clock control register (UCC;22H). Address and remote wakeup register (AWR;23H), STALL register(24H), SIES register (25H), MISC register(26H), SETIO register(27H), FIFO0~FIFO2 register (28H~2AH). DAC_Limit_L register (2DH), DAC_Limit_H register (2EH), DAC_WR register (2FH).

The remaining space before the 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 or MP1).

	Bank 0 Special Register
00H	Indirect Addressing Register 0
01H	MP0
02H	Indirect Addressing Register 1
03H	MP1
04H	BP
05H	ACC
06H	PCL
07H	TBLP
08H	TBLH
09H	WDTS
0AH	STATUS
0BH	INTC0
0CH	TMR0H
0DH	TMR0L
0EH	TMR0C
0FH	TMR1H
10H	TMR1L
11H	TMR1C
12H	PA
13H	PAC
14H ⋮ 1BH	
1CH	USVC
1DH	
1EH	
1FH	TBHP
20H	USC
21H	USR
22H	UCC
23H	AWR
24H	STALL
25H	SIES
26H	MISC
27H	SETIO
28H	FIFO0
29H	FIFO1
2AH	FIFO2
2BH ⋮ 2CH	
2DH	DAC_Limit_L
2EH	DAC_Limit_H
2FH	DAC_WR
30H ⋮ 3FH	
40H ⋮ FFH	General Purpose Data RAM (192 Bytes)

RAM Mapping

Indirect Addressing Register

Locations 00H and 02H are indirect addressing registers that are not physically implemented. Any read/write operation on [00H] ([02H]) will access the data memory pointed to by MP0 (MP1). Reading location 00H (02H) indirectly will return the result 00H. Writing indirectly results in no operation.

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

Bank Pointer

The bank pointer is used to assign the accessed RAM bank. When the users want to access the RAM bank 0, a ″0″ should be loaded onto BP. RAM locations before 40H in any bank are overlapped.

Accumulator

The accumulator is closely related to ALU operations. It is also mapped to location 05H of the data memory and can carry out immediate data operations. 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 ....)

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

Status Register – STATUS

This 8-bit register (0AH) contains the zero flag (Z), carry flag (C), auxiliary carry flag (AC), overflow flag (OV), power down flag (PDF), and watchdog time-out flag (TO). It also records the status information and controls the operation sequence.

With the exception of the TO and PDF flags, bits in the status register can be altered by instructions like most other registers. Any data written into the status register will not change the TO or PDF flag. In addition, operations related to the status register may give different results from those intended.

The TO flag can be affected only by a system power-up, a WDT time-out or executing the ″CLR WDT″ or ″HALT″ instruction. The PDF flag can be affected only by executing the ″HALT″ or ″CLR WDT″ instruction or during a system power-up.

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

In addition, upon 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 are important and if the subroutine can corrupt the status register, precautions must be taken to save it properly.

Interrupt

The device provides USB interrupt and internal timer/event counter interrupts. The Interrupt Control Register0 (INTC0;0BH) contains 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

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

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 INTC 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 SP 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 a 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.

The USB interrupts are triggered by the following USB events and the related interrupt request flag (USBF; bit 4 of the INTC0) will be set.

- Access of the corresponding USB FIFO from PC
- The USB suspend signal from PC
- The USB resume signal from PC
- USB Reset signal

When the interrupt is enabled, the stack is not full and the external interrupt is active, a subroutine call to location 04H 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 HT82A821R, the corresponding request bit of USR is set, and a USB interrupt is triggered. So user can easy to decide which FIFO is accessed. When the interrupt has been served, the corresponding bit should be cleared by firmware. When HT82A821R receive a USB Suspend signal from Host PC, the suspend line (bit0 of USC) of the HT82A821R is set and a USB interrupt is also triggered.

When the HT82A821R receives a Resume signal from the Host PC, the resume line (bit3 of the USC) of the HT82A821R are set and a USB interrupt is triggered.

Also when HT82A821R receive a Resume signal from Host PC, the resume line (bit3 of USC) of HT82A821R is set and a USB interrupt is triggered.

The internal Timer/Event Counter 0 interrupt is initialized by setting the Timer/Event Counter 0 interrupt request flag (bit 5 of 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/Even Counter 1 interrupt is initialized by setting the Timer/Event Counter 1 interrupt request flag (bit 6 of 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.

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.

No.	Interrupt Source	Priority	Vector
a	USB interrupt	1	04H
b	Timer/Event Counter 0 overflow	2	08H
c	Timer/Event Counter 1 overflow	3	0CH

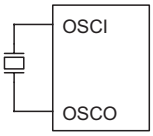
It is recommended that a program does not use the ″CALL subroutine″ within the interrupt subroutine. Interrupts often occur in an unpredictable manner or need to be serviced immediately in some applications. If only one stack is left and enabling the interrupt is not well controlled, the original control sequence will be damaged once the ″CALL″ operates in the interrupt subroutine.

Bit No.	Label	Function
0	EMI	Controls the master (global) interrupt (1=enable; 0=disable)
1	EUI	Controls the USB 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	USBF	USB 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

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 OSCI and OSCO 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 OSCI and OSCO to get a frequency reference, but two external capacitors in OSCI and OSCO are required.

The WDT oscillator is a free running on-chip RC oscillator, and no external components are required. Even if the system enters the power down mode, 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

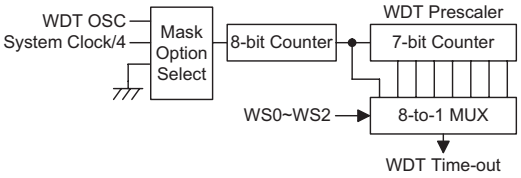
The WDT clock source is implemented by a dedicated RC oscillator (WDT oscillator) or a instruction clock (system clock/4). The timer is designed to prevent a software malfunction or sequence from jumping to an unknown location with unpredictable results. The WDT can be disabled by options. But if the WDT is disabled, all executions related to the WDT lead to no operation.

When the WDT clock source is selected, it will be first divided by 256 (8-stage) to get the nominal time-out period. By invoking the WDT prescaler, longer time-out periods can be realized. Writing data to WS2, WS1, WS0 can give different time-out periods.

The WDT OSC period is typical 65μs. This time-out period may vary with temperature, VDD and process variations. The WDT OSC always works for any operation mode.

If the instruction clock is selected as the WDT clock source, the WDT operates in the same manner except in the halt mode. In the mode, the WDT stops counting and lose its protecting purpose. In this situation the logic can only be re-started by external logic. The high nibble and bit3 of the WDTS are reserved for user defined flags, which can be used to indicate some specified status.

The WDT overflow under normal operation initializes a ″chip reset″ and sets the status bit ″TO″. In the HALT mode, the overflow initializes a ″warm reset″, and only the PC and SP are reset to zero. To clear the contents of the WDT, there are three methods to be adopted, i.e., external reset (a low level to $\overline{\text{RESET}}$), software instruction, and a ″HALT″ instruction. There are two types of software instructions; ″CLR WDT″ and the other set ″CLR WDT1″ and ″CLR WDT2″. Of these two types of instruction, only one type of instruction can be active at a time depending on the options ″CLR WDT″ times selection option. If the ″CLR WDT″ is selected (i.e., CLR WDT times equal one), any execution of the ″CLR WDT″ instruction clears the WDT. In the case that ″CLR WDT1″ and ″CLR WDT2″ are chosen (i.e., CLR WDT times equal two), these two instructions have to be executed to clear the WDT; otherwise, the WDT may reset the chip due to time-out.



Watchdog Timer

Bit No.	Label	Function
0 1 2	WS0 WS1 WS2	Watchdog Timer division ratio selection bits Bit 2,1,0 = 000, division ratio = 1:1 Bit 2,1,0 = 001, division ratio = 1:2 Bit 2,1,0 = 010, division ratio = 1:4 Bit 2,1,0 = 011, division ratio = 1:8 Bit 2,1,0 = 100, division ratio = 1:16 Bit 2,1,0 = 101, division ratio = 1:32 Bit 2,1,0 = 110, division ratio = 1:64 Bit 2,1,0 = 111, division ratio = 1:128
3	—	Unused bit, read as ″0″
7~4	T3~T0	Test mode setting bits (T3, T2, T1, T0)=(0, 1, 0, 1), enter DAC write mode. Otherwise normal operation.

WDTS (09H) Register

Power Down Operation – HALT

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

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

The system can leave the HALT mode by means of an external reset, an interrupt, an external falling edge signal on port A or a WDT overflow. An external reset causes a device initialization and the WDT overflow performs a "warm reset". After the TO and PDF flags are examined, the cause for chip reset can be determined. The PDF flag is cleared by a system power-up or executing the "CLR WDT" instruction and is set when executing the "HALT" instruction. The TO flag is set if the WDT time-out occurs, and causes a wake-up that only resets the program counter and SP; the others remain 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 mask option. Awakening from an I/O port stimulus, the program will resume execution of the next instruction. If it awakens from an interrupt, two sequence 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. If the interrupt is enabled and the stack is not full, the regular interrupt response takes place. If an interrupt request flag is set to "1" before entering the HALT mode, the wake-up function of the related interrupt will be disabled. Once a wake-up event occurs, it takes 1024 t_{SYS} (system clock period) to resume normal operation. In other words, a dummy period will be inserted after a wake-up. If the wake-up results from an interrupt acknowledge signal, the actual interrupt subroutine execution will be delayed by one or more cycles. 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 four ways in which a reset can occur:

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

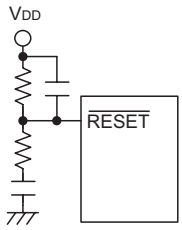
The WDT time-out during HALT is different from other chip reset conditions, since it can perform a "warm reset" that resets only the program counter and SP, leaving the other circuits in their original state. Some registers remain unchanged during other reset conditions. Most registers are reset to the "initial condition" when the reset conditions are met. By examining the PDF and TO flags, the program can distinguish between different "chip resets".

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

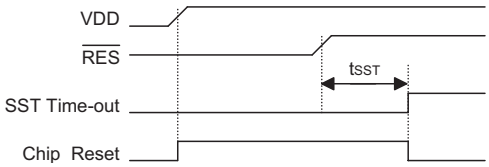
Note: "u" stands for "unchanged"

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 resets (power-up, WDT time-out or $\overline{\text{RES}}$ reset) or the system awakes from the HALT state.

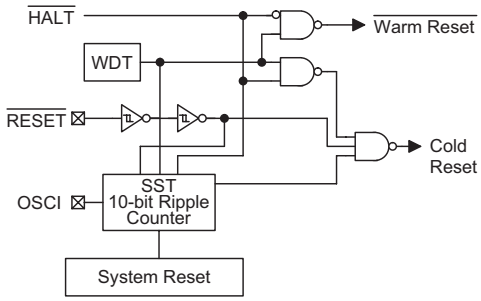
When a system reset occurs, the SST delay is added during the reset period. Any wake-up from HALT will enable the SST delay.



Reset Circuit



Reset Timing Chart



Reset Configuration

The functional unit chip reset status are shown below.

Program Counter	000H
Interrupt	Disable
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

The registers status are summarized in the following table.

Register	Reset (Power On)	WDT Time-out (Normal Operation)	$\overline{\text{RES}}$ Reset (Normal Operation)	$\overline{\text{RES}}$ Reset (HALT)	WDT Time-Out (HALT)*	USB-Reset (Normal)	USB-Reset (HALT)
MP0	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
MP1	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
Program Counter	000H	000H	000H	000H	000H	000H	000H
TBLP	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLH	-xxx xxxx	-uuu uuuu	-uuu uuuu	-uuu uuuu	-uuu uuuu	-uuu uuuu	-uuu uuuu
WDTS	0000 0111	0000 0111	0000 0111	0000 0111	uuuu uuuu	0000 0111	0000 0111
STATUS	--00 xxxx	--1u uuuu	--uu uuuu	--01 uuuu	--11 uuuu	--uu uuuu	--01 uuuu
INTC0	-000 0000	-000 0000	-000 0000	-000 0000	-uuu uuuu	-000 0000	-000 0000
TMR0H	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TMR0L	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
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	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TMR1L	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	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
USC	1000 0000	uuxx uuuu	10xx 0000	10xx 0000	10xx uuuu	1000 0u00	1000 0u00
USR	0000 0000	uuuu uuuu	0000 0000	0000 0000	uuuu uuuu	00uu 0000	00uu 0000
UCC	0000 0000	uuuu uuuu	0000 0000	0000 0000	uuuu uuuu	0u00 u000	0u00 u000
AWR	0000 0000	uuuu uuuu	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
STALL	0000 0000	uuuu uuuu	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
SIES	0000 0000	uuuu uuuu	0000 0000	0000 0000	uuuu uuuu	0u00 u000	0u00 u000
MISC	0000 0000	uuuu uuuu	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
SETIO	xxxx x010	xxxx x010	xxxx x010	xxxx x010	xxxx x010	xxxx x010	xxxx x010
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
DAC_LIMIT_L	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
DAC_LIMIT_H	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
DAC_WR	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000

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

Timer/Event Counter

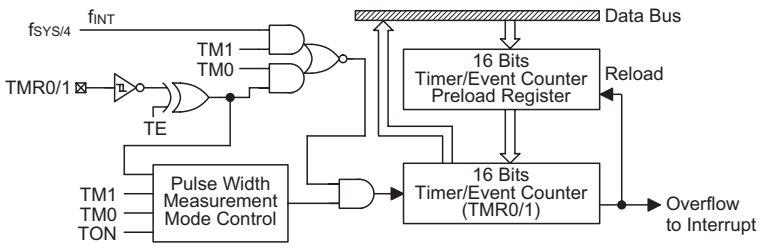
Two timer/event counters (TMR0, TMR1) are implemented in the microcontroller. The timer/event counter 0/1 contains a 16-bit programmable count-up counter and the clock may come from an external source or an internal clock source. An internal clock source comes from $f_{SYS}/4$. The external clock input allows the user to count external events, measure time intervals or pulse widths, or to generate an accurate time base. There are six registers related to the Timer/Event Counter 0; TMR0H (0CH), TMR0L (0DH), TMR0C (0EH) and the Timer/Event Counter 1; TMR1H (0FH), TMR1L (10H), TMR1C (11H). For 16-bit timer to write data to TMR0/1L will only put the written data to an internal lower-order byte buffer (8-bit) and writing TMR0/1H will transfer the specified data and the contents of the lower-order byte buffer to TMR0/1H and TMR0/1L registers. The Timer/Event Counter 0/1 preload register is changed by each writing TMR0/1H operations. Reading TMR0/1H will latch the contents of TMR0/1H and TMR0/1L counters to the destination and the lower-order byte buffer, respectively. Reading the TMR0/1L will read the contents of the lower-order byte buffer. The TMR0/1C is the Timer/Event Counter 0/1 control register, which defines the operating mode, counting enable or disable and an active edge.

The TM0 and TM1 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 clock source. Finally, the pulse width measurement mode can be used to count the high level or low level duration of the external signal (TMR0, TMR1), and the counting is based on the internal 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. 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 INTC0, T1F; bit 6 of INTC0). In the pulse width measurement mode with the values of the TON and TE bits equal to 1, after the TMR0 (TMR1) has received a transient from low to high (or high to low if the TE bit is ″0″), it will start counting until the TMR0 (TMR1) returns to the original level and resets the TON. 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 TON is set. The cycle measurement will re-function as long as it receives further transient pulse. In this operation mode, the timer/event counter begins counting not according to the logic level but to the transient edges. In the case of counter overflows, the counter is reloaded from the timer/event counter register and issues an interrupt request, as in the other two modes, i.e., event and timer modes.

Bit No.	Label	Function
0~2, 5	—	Unused bit, read as ″0″
3	TE	Defines the TMR active edge of the timer/event counter In Event counter mode (TM1, TM0)=(0, 1): 1=count on falling edge; 0=count on rising edge In Pulse width measurement mode (TM1, TM0)=(1, 1): 1=start counting on the rising edge, stop on the falling edge; 0=start counting on the falling edge, stop on the rising edge
4	TON	Enable/disable the timer counting (0=disable; 1=enable)
6 7	TM0 TM1	Defines the operating mode 01=Event count mode (external clock) 10=Timer mode (internal clock) 11=Pulse width measurement mode 00=Unused

TMRC (11H) Register



Timer/Event Counter 0/1

To enable the counting operation, the Timer ON bit (TON; bit 4 of TMR0C or TMR1C) should be set to 1. In the pulse width measurement mode, TON is automatically cleared after the measurement cycle is completed. But in the other two modes, the TON 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 turn 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.

Input/Output Ports

There are 8 bidirectional input/output lines (PA) in the microcontroller, which are mapped to the data memory of [12H] 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). For output operation, all the data is latched and remains unchanged until the output latch is rewritten.

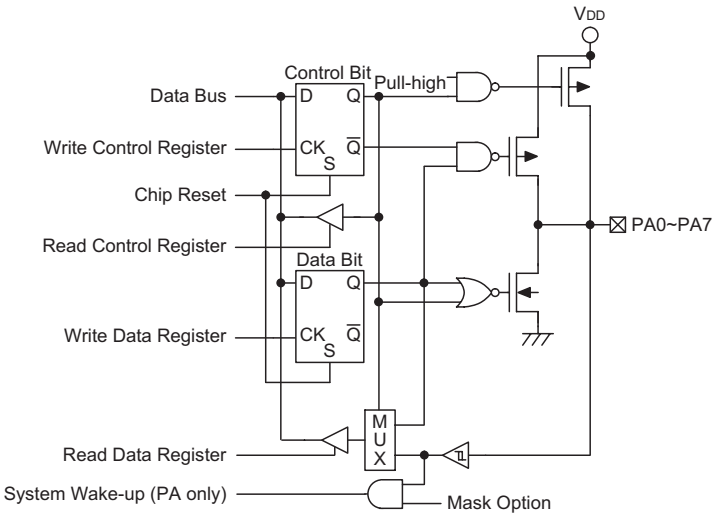
Each I/O line has its own control register (PAC) to control the input/output configuration. With this control register, CMOS output or Schmitt trigger input with or without pull-high resistor structures can be reconfigured dynamically (i.e., on-the-fly) under software control. To function as an input, the corresponding latch of the control register must write ″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 configurations can be selected. The control register is mapped to location 13H.

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

Each line of port A has 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.



Input/Output Ports

Low Voltage Reset – LVR (by ROM Code Option)

The LVR option is 3.0V.

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 (0.9V~V_{LVR}) has to remain in their original state to exceed 1ms. If the low voltage state does not exceed 1ms, the LVR will ignore it and do not perform a reset function.
- The LVR uses the ″OR″ function with the external $\overline{\text{RESET}}$ signal to perform chip reset.

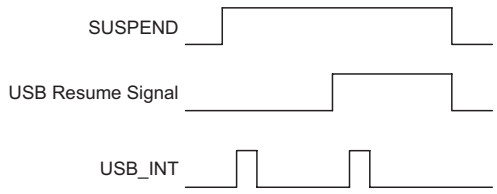
Suspend Wake-Up and Remote Wake-Up

If there is no signal on the USB bus for over 3ms, the HT82A821R will go into a 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 HT82A821R should jump to the suspend state to meet the USB suspend current spec.

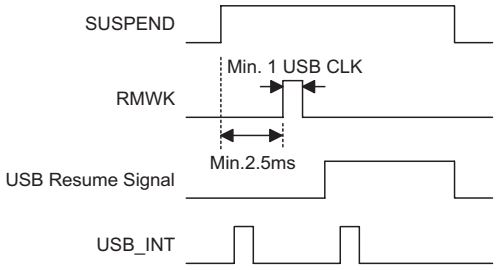
In order to meet the suspend current, the firmware should disable the USB clock by clearing the USBCKEN (bit3 of the UCC) to ″0″.

Also the user can further decrease the suspend current by set the SUSP2 (bit4 of the UCC).

When the resume signal is sent out by the host, the HT82A821R will wake up the MCU by USB interrupt and the Resume line (bit 3 of USC) is set. In order to make HT82A821R work properly, the firmware must set the USBCKEN (bit 3 of UCC) to 1 and clear the 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 USC) is going to ″0″. So when the MCU is detecting the Suspend line (bit0 of USC), the Resume line should be remembered and token into consideration. 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 USC). Once the USB Host receive the wake-up signal from HT82A821R, it will send a Resume signal to device. The timing as follow:



USB Interface

The HT82A821R have 3 Endpoints (EP0 ~EP2). EP0 supports Control transfer. EP1 supports Interrupt transfer. EP2 supports Isochronous transfer.

These registers, including USC (20H), USR (21H), UCC (22H), AWR (23H), STALL (24H), SIES (25H), MISC (26H), SETIO (27H), FIFO0 (28H), FIFO1 (29H), FIFO2 (2AH) used for the USB function.

The FIFO size of each FIFO is 8 byte (FIFO0), 8 byte (FIFO1), 384 byte (FIFO2), and total are 400 bytes.

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

Bit No.	Label	R/W	Reset	Functions
0	SUSP	R	0	Read only, USB suspend indication. When this bit is set to ″1″ (set by SIE), it indicates the USB bus enters suspend mode. The USB interrupt is also triggered on changing from low to high of this bit.
1	RMWK	R/W	0	USB remote wake-up command. It is set by MCU to force the USB host leaving the suspend mode.
2	URST	R/W	0	USB reset indication. This bit is set/cleared by USB SIE. This bit is used to detect USB reset event on USB bus. When this bit is set to ″1″, this indicates an USB reset is occurred and an USB interrupt will be initialized.
3	RESUME	R	0	USB resume indication. When the USB leaves suspend mode, this bit is set to ″1″ (set by SIE). When the RESUME is set by SIE, an interrupt will be generated to wake-up the MCU. In order to detecting the suspend state, MCU should set USBCKEN and clear SUSP2 (in UCC register) to enable the SIE detecting function. The RESUME will be cleared while the SUSP is going ″0″. When MCU is detecting the SUSP, the RESUME (causes MCU to wake-up) should be remembered and token into consideration.
4	V33O	R/W	0	0/1: Turn-off/on V33O output
5~6	—	—	—	Undefined bit, read as ″0″.
7	URD	R/W	1	USB reset signal control function definition 1: USB reset signal will reset MCU 0: USB reset signal cannot reset MCU

USC (20H) Register

The USR (USB endpoint interrupt status register) register is used to indicate which endpoint is accessed and to select serial bus (USB). The endpoint request flags (EP0F, EP1F, EP2F) 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 USB 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″ by software.

Bit No.	Label	R/W	Reset	Functions
0	EP0F	R/W	0	When this bit is set to ″1″ (set by SIE). It indicates the endpoint 0 is accessed and an USB interrupt will occur. When the interrupt has been served, this bit should be cleared by software.
1	EP1F	R/W	0	When this bit is set to ″1″ (set by SIE). It indicates the endpoint 1 is accessed and an USB interrupt will occur. When the interrupt has been served, this bit should be cleared by software.
2	EP2F	R/W	0	When this bit is set to ″1″ (set by SIE). It indicates the endpoint 2 is accessed and an USB interrupt will occur. When the interrupt has been served, this bit should be cleared by software.
3~7	—	—	—	Undefined bit, read as ″0″.

USR (21H) Register

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)

And to define which endpoint FIFO is select by EPS2, EPS1 and EPS0.

Bit No.	Label	R/W	Reset	Functions
0~2	EPS0~EPS2	R/W	0	Accessing endpoint FIFO selection, EPS2, EPS1, EPS0: 000: Select endpoint 0 FIFO 001: Select endpoint 1 FIFO 010: Select endpoint 2 FIFO 011: reserved for future expansion, cannot be used 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 are not existed, the related functions will be absent.
3	USBCKEN	R/W	0	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	0	This bit is used for reducing power consumption in suspend mode. In normal mode, clean this bit to ″0″ In HALT mode, set this bit to ″1″ for reducing power consumption.
5	f_{SYS}24MHz	R/W	0	This bit is used to define the MCU system clock comes form external OSC or system clock comes PLL output 24MHz clock. 0: system clock comes from OSC 1: system clock comes from PLL output 24MHz
6	SYSCLK	R/W	0	This bit is used to specify the system clock oscillator frequency used by 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″.

UCC (22H) Register

Note: Isochronous endpoint 2 is implemented by hardware, so FIFO2 can not read/write by firmware.

AWR register contains current address and a remote wake up function control bit. The initial value of AWR is ″00H″. The address value extracted from the USB command has not to be loaded into this register until the SETUP stage being finished.

Bit No.	Label	R/W	Power-on	Functions
0	WKEN	R/W	0	USB remote-wake-up enable/disable (1/0)
1~7	AD0~AD6	R/W	0	USB device address

AWR (23H) Register

STALL register shows where 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 USB reset signal.

Bit No.	Label	R/W	Power-on	Functions
0~2	STL0~STL2	R/W	0	Set by users when related USB endpoints were stalled. They are cleared by USB reset and Setup Token event.
3~7	STL3~STL7	—	0	Undefined bit, read as ″0″.

STALL (24H) Register

Bit No.	Label	R/W	Power-on	Functions
0	ASET	R/W	0	This bit is used to configure the SIE automatically change the device address by the value stored in the AWR register. When this bit is set to ″1″ by firmware, the SIE will update the device address by the value stored in the AWR register after PC host is successfully read the data from 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 next valid SETUP token is received.
1	ERR	R/W	0	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	0	This bit is used to indicate there are OUT token (except the OUT zero length token) has 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	0	This bit is used to indicate the current USB receiving signal from PC host is IN token.
4	NAK	R	0	This bit is used to indicate the SIE is transmitted NAK signal to host in response to PC host IN or OUT token.
5	CRCF	R/W	0	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	1	Token package active flag, low active.
7	NMI	R/W	0	NAK token interrupt mask flag. If this bit set, when device sent a NAK token to host, the interrupt will not happen. Otherwise when this bit is cleared, device sent a NAK token to host will enter the interrupt sub-routine.

SIES (25H) Register

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

Bit No.	Label	R/W	Power-on	Functions
0	REQUEST	R/W	0	After setting others status of desired one, FIFO can be requested by setting this bit high active. After work has been done, this bit must be set low.
1	TX	R/W	0	To represent the direction and transition end MCU accesses, When being set logic 1, MCU wants to write data to FIFO. After the work being done, this bit must be set logic 0 before terminating request to represent transition end. For reading action, this bit must be set logic 0 to represent MCU want to read and must be set logic 1 after the work done.
2	CLEAR	R/W	0	To represent MCU clear requested FIFO, even 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	—	R	0	Undefined bit, read as ″0″.
4	ISOEN-	R/W	0	To enable the isochronous pipe interrupt.
5	SETCMD	R/W	0	To show that the data in FIFO is setup command. This bit will last this state until next one entering the FIFO.
6	READY	R	0	To tell that the desired FIFO is ready to work.
7	LEN0	R	0	To tell that host sent a 0-sized packet to MCU. This bit must be cleared by read action to corresponding FIFO.

USB MISC (26H) Register

Bit No.	Label	R/W	Power-on	Functions
0	DATATG*	R/W	0	To toggle this bit, all the DATA token will send DATA0 first.
1	SETIO1**	R/W	1	Set endpoint1 input or output pipe (1/0), default input pipe(1)
2	SETIO2**	R/W	0	Set endpoint2 input or output pipe (1/0), default output pipe(0)
3~7	—	—	—	Reserved

SETIO Register, USB Endpoint 1~Endpoint 2 Set IN/OUT Pipe Register

Note: *USB definition: when host send a ″set Configuration″, the Data pipe should send the DATA0 (about the Data toggle) first. So, when Device received a ″set configuration″ setup command, user need to toggle this bit for next data will send a Data0 first.

**Only need to set the data pipe as a input pile or output pile. The purpose of this function is to avoid the host sent a abnormal IN or OUT token and make the endpoint disability.

Bit No.	Label	R/W	Power-on	Functions
0~6	USVC0~ USVC6	R/W	0	Volume control Bit0~Bit6
7	$\overline{\text{MUTE}}$	R/W	0	Mute control, low active.

USB Speaker Volume Control

Result (dB)	USVC	Result (dB)	USVC	Result (dB)	USVC	Result (dB)	USVC
6	000_1100	–2	111_1100	–10	110_1100	–24	101_1100
5.5	000_1011	–2.5	111_1011	–10.5	110_1011	–25	101_1011
5	000_1010	–3	111_1010	–11	110_1010	–26	101_1010
4.5	000_1001	–3.5	111_1001	–11.5	110_1001	–27	101_1001
4	000_1000	–4	111_1000	–12	110_1000	–28	101_1000
3.5	000_0111	–4.5	111_0111	–13	110_0111	–29	101_0111
3	000_0110	–5	111_0110	–14	110_0110	–30	101_0110
2.5	000_0101	–5.5	111_0101	–15	110_0101	–31	101_0101
2	000_0100	–6	111_0100	–16	110_0100	–32	101_0100
1.5	000_0011	–6.5	111_0011	–17	110_0011	—	—
1	000_0010	–7	111_0010	–18	110_0010	—	—
0.5	000_0001	–7.5	111_0001	–19	110_0001	—	—
0	000_0000	–8	111_0000	–20	110_0000	—	—
–0.5	111_1111	–8.5	110_1111	–21	101_1111	—	—
–1	111_1110	–9	110_1110	–22	101_1110	—	—
–1.5	111_1101	–9.5	110_1101	–23	101_1101	—	—

Speaker mute Control:

$\overline{\text{MUTE}}$= 0: Mute Speaker output.

$\overline{\text{MUTE}}$= 1: Normal.

Registers	R/W	Power-on	Functions
FIFO0~ FIFO2	R/W	xxH	EPi accessing register (i = 0~2). When an endpoint is disabled, the corresponding accessing register should be disabled.

USB Endpoint Accessing Registers Definitions

DAC_Limit_L and DAC_Limit_H are used to define the 16-bit DAC output limit. DAC_Limit_L and DAC_Limit_H are unsigned value. If the 16-bit data from Host over the range defined by DAC_Limit_L and DAC_Limit_H, the output digital code to DAC will be clamp.

DAC_Limit_L	DAC output limit low byte
DAC_Limit_H	DAC output limit high byte

Setting DAC output limit value example:

```
;-----------------------------------------------------------
; DAC Limit POR Value=8000H
; Set DAC Limit Value=FF00H
;-----------------------------------------------------------
clr       [02DH]                ; Set DAC Limit low byte=00H
set       [02EH]                ; Set DAC Limit high byte=FFH
;-----------------------------------------------------------
```

In order to prevent the pop noise of speaker output, power amplifier should be output at the value of VDD/2 (send 8000H to DAC) during the initial power on state. If software set high then clear the bit DAC_WR_TRIG (bit 3 of DAC_WR register), the value on the DAC_Limit_L and DAC_Limit_H registers will write to DAC.

Bit No.	Label	R/W	Power-on	Functions
0~2, 4~7	—	R	0	Undefined bit, read as ″0″.
3	DAC_WR_TRIG	R/W	0	DAC write trigger bit

DAC_WR (2FH) Register

Example to avoid popping noise:

```
System_Initial:
;-----------------------------------------------------------
; Avoid Pop Noise
;-----------------------------------------------------------
mov       a,WDTS
mov       FIFO_TEMP,a           ;Save WDTS value
mov       a,01010000b
andm      a,WDTS
mov       a,01010000b
orm       a,WDTS                ;Enter DAC Write Data mode, high nibble of WDTS=0101b
clr       [02DH]                ;Set DAC data low byte=00H
mov       a,80H
mov       [02EH],a              ;Set DAC data high byte=80H
nop
;Write 8000H to DAC
set       [02FH].3
nop
clr       [02FH].3
nop
;-----------------------------------------------------------
mov       a,FIFO_TEMP           ;Restore WDTS value
mov       WDTS,a                ;Quit DAC Write Data mode
;-----------------------------------------------------------
```

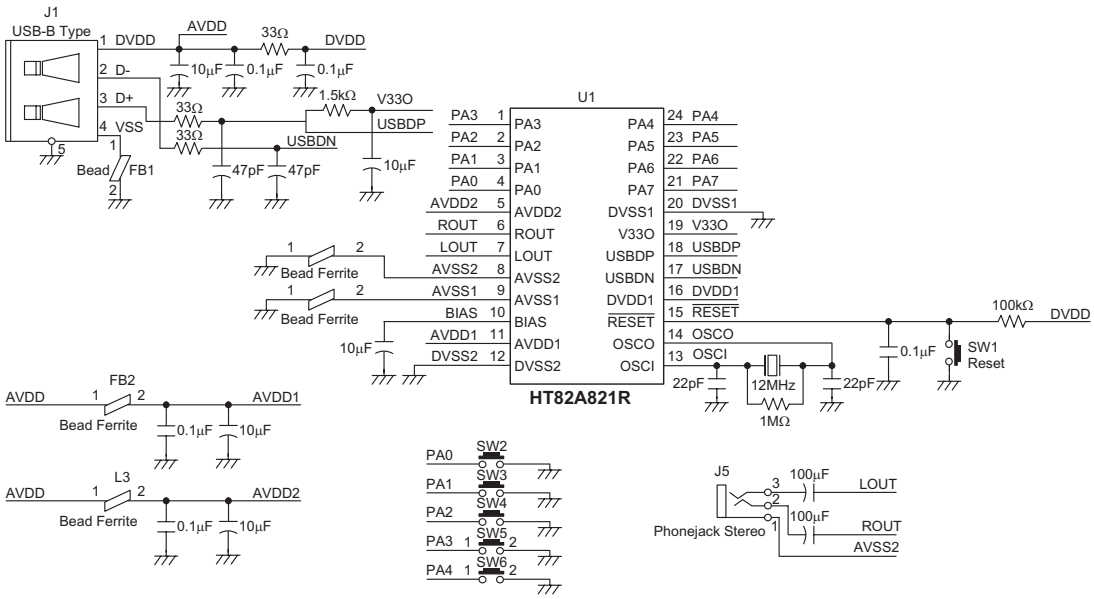
Note: At DAC write data mode (high nibble of WDTS register is 0101b), DAC_Limit_L and DAC_Limit_H registers will be the 16-bit DAC input data register at falling edge of DAC_WR_TRIG. Otherwise, these two registers are used to define the 16-bit DAC output limit.

Options

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

No.	Option
1	PA0~PA7 pull-high resistor enabled or disabled (by bit)
2	LVR enable or disable
3	WDT enable or disable
4	WDT clock source: f_{SYS}/4 or WDTOSC
5	CLRWDT instruction(s): 1 or 2
6	PA0~PA7 wake-up enabled or disabled (by bit)
7	TBHP enable or disable (default disable)

Application Circuits



Instruction Set Summary

Mnemonic	Description	Instruction Cycle	Flag Affected
Arithmetic			
ADD A,[m]	Add data memory to ACC	1	Z,C,AC,OV
ADDM A,[m]	Add ACC to data memory	1(1)	Z,C,AC,OV
ADD A,x	Add immediate data to ACC	1	Z,C,AC,OV
ADC A,[m]	Add data memory to ACC with carry	1	Z,C,AC,OV
ADCM A,[m]	Add ACC to data memory with carry	1(1)	Z,C,AC,OV
SUB A,x	Subtract immediate data from ACC	1	Z,C,AC,OV
SUB A,[m]	Subtract data memory from ACC	1	Z,C,AC,OV
SUBM A,[m]	Subtract data memory from ACC with result in data memory	1(1)	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 and result in data memory	1(1)	Z,C,AC,OV
DAA [m]	Decimal adjust ACC for addition with result in data memory	1(1)	C
Logic Operation			
AND A,[m]	AND data memory to ACC	1	Z
OR A,[m]	OR data memory to ACC	1	Z
XOR A,[m]	Exclusive-OR data memory to ACC	1	Z
ANDM A,[m]	AND ACC to data memory	1(1)	Z
ORM A,[m]	OR ACC to data memory	1(1)	Z
XORM A,[m]	Exclusive-OR ACC to data memory	1(1)	Z
AND A,x	AND immediate data to ACC	1	Z
OR A,x	OR immediate data to ACC	1	Z
XOR A,x	Exclusive-OR immediate data to ACC	1	Z
CPL [m]	Complement data memory	1(1)	Z
CPLA [m]	Complement data memory with result in ACC	1	Z
Increment & Decrement			
INCA [m]	Increment data memory with result in ACC	1	Z
INC [m]	Increment data memory	1(1)	Z
DECA [m]	Decrement data memory with result in ACC	1	Z
DEC [m]	Decrement data memory	1(1)	Z
Rotate			
RRA [m]	Rotate data memory right with result in ACC	1	None
RR [m]	Rotate data memory right	1(1)	None
RRCA [m]	Rotate data memory right through carry with result in ACC	1	C
RRC [m]	Rotate data memory right through carry	1(1)	C
RLA [m]	Rotate data memory left with result in ACC	1	None
RL [m]	Rotate data memory left	1(1)	None
RLCA [m]	Rotate data memory left through carry with result in ACC	1	C
RLC [m]	Rotate data memory left through carry	1(1)	C
Data Move			
MOV A,[m]	Move data memory to ACC	1	None
MOV [m],A	Move ACC to data memory	1(1)	None
MOV A,x	Move immediate data to ACC	1	None
Bit Operation			
CLR [m].i	Clear bit of data memory	1(1)	None
SET [m].i	Set bit of data memory	1(1)	None

Mnemonic	Description	Instruction Cycle	Flag Affected
Branch			
JMP addr	Jump unconditionally	2	None
SZ [m]	Skip if data memory is zero	1(2)	None
SZA [m]	Skip if data memory is zero with data movement to ACC	1(2)	None
SZ [m].i	Skip if bit i of data memory is zero	1(2)	None
SNZ [m].i	Skip if bit i of data memory is not zero	1(2)	None
SIZ [m]	Skip if increment data memory is zero	1(3)	None
SDZ [m]	Skip if decrement data memory is zero	1(3)	None
SIZA [m]	Skip if increment data memory is zero with result in ACC	1(2)	None
SDZA [m]	Skip if decrement data memory is zero with result in ACC	1(2)	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](5)	Read ROM code (locate by TBLPand TBHP) to data memory and TBLH	2(1)	None
TABRDC [m](6)	Read ROM code (current page) to data memory and TBLH	2(1)	None
TABRDL [m]	Read ROM code (last page) to data memory and TBLH	2(1)	None
Miscellaneous			
NOP	No operation	1	None
CLR [m]	Clear data memory	1(1)	None
SET [m]	Set data memory	1(1)	None
CLR WDT	Clear Watchdog Timer	1	TO,PDF
CLR WDT1	Pre-clear Watchdog Timer	1	TO(4),PDF(4)
CLR WDT2	Pre-clear Watchdog Timer	1	TO(4),PDF(4)
SWAP [m]	Swap nibbles of data memory	1(1)	None
SWAPA [m]	Swap nibbles of data memory with result in ACC	1	None
HALT	Enter power down mode	1	TO,PDF

Note: x: Immediate data

m: Data memory address

A: Accumulator

i: 0~7 number of bits

addr: Program memory address

√: Flag is affected

–: Flag is not affected

(1): If a loading to the PCL register occurs, the execution cycle of instructions will be delayed for one more cycle (four system clocks).

(2): If a skipping to the next instruction occurs, the execution cycle of instructions will be delayed for one more cycle (four system clocks). Otherwise the original instruction cycle is unchanged.

(3): (1) and (2)

(4): The flags may be affected by the execution status. If the Watchdog Timer is cleared by executing the ″CLR WDT1″ or ″CLR WDT2″ instruction, the TO and PDF are cleared.
Otherwise the TO and PDF flags remain unchanged.

(5): ″ROM code TBHP option″ is enabled

(6): ″ROM code TBHP option″ is disabled

Instruction Definition

ADC A,[m] Add data memory and carry to the accumulator

Description The contents of the specified data memory, accumulator and the carry flag are added simultaneously, leaving the result in the accumulator.

Operation ACC ← ACC+[m]+C

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	√	√	√	√

ADCM A,[m] Add the accumulator and carry to data memory

Description The contents of the specified data memory, accumulator and the carry flag are added simultaneously, leaving the result in the specified data memory.

Operation [m] ← ACC+[m]+C

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	√	√	√	√

ADD A,[m] Add data memory to the accumulator

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)

TO	PDF	OV	Z	AC	C
—	—	√	√	√	√

ADD A,x Add immediate data to the accumulator

Description The contents of the accumulator and the specified data are added, leaving the result in the accumulator.

Operation ACC ← ACC+x

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	√	√	√	√

ADDM A,[m] Add the accumulator to the data memory

Description The contents of the specified data memory and the accumulator are added. The result is stored in the data memory.

Operation [m] ← ACC+[m]

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	√	√	√	√

AND A,[m] Logical AND accumulator with data memory

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)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

AND A,x Logical AND immediate data to the accumulator

Description Data in the accumulator and the specified data perform a bitwise logical_AND operation. The result is stored in the accumulator.

Operation ACC ← ACC ″AND″ x

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

ANDM A,[m] Logical AND data memory with the accumulator

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)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

CALL addr Subroutine call

Description The instruction unconditionally calls a subroutine located at the indicated address. The program counter increments once to obtain the address of the next instruction, and pushes this onto the stack. The indicated address is then loaded. Program execution continues with the instruction at this address.

Operation Stack ← Program Counter+1
Program Counter ← addr

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

CLR [m] Clear data memory

Description The contents of the specified data memory are cleared to 0.

Operation [m] ← 00H

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

CLR [m].i Clear bit of data memory

Description The bit i of the specified data memory is cleared to 0.

Operation [m].i ← 0

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

CLR WDT Clear Watchdog Timer

Description The WDT is cleared (clears the WDT). The power down bit (PDF) and time-out bit (TO) are cleared.

Operation WDT ← 00H
PDF and TO ← 0

Affected flag(s)

TO	PDF	OV	Z	AC	C
0	0	—	—	—	—

CLR WDT1 Preclear Watchdog Timer

Description Together with CLR WDT2, clears the WDT. PDF and TO are also cleared. Only execution of this instruction without the other preclear instruction just sets the indicated flag which implies this instruction has been executed and the TO and PDF flags remain unchanged.

Operation WDT ← 00H*
PDF and TO ← 0*

Affected flag(s)

TO	PDF	OV	Z	AC	C
0*	0*	—	—	—	—

CLR WDT2 Preclear Watchdog Timer

Description Together with CLR WDT1, clears the WDT. PDF and TO are also cleared. Only execution of this instruction without the other preclear instruction, sets the indicated flag which implies this instruction has been executed and the TO and PDF flags remain unchanged.

Operation WDT ← 00H*
PDF and TO ← 0*

Affected flag(s)

TO	PDF	OV	Z	AC	C
0*	0*	—	—	—	—

CPL [m] Complement data memory

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

Operation [m] ← $[\overline{m}]$

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

CPLA [m] Complement data memory and place result in the accumulator

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

Operation ACC ← $[\overline{m}]$

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

DAA [m] Decimal-Adjust accumulator for addition

Description The accumulator value is adjusted to the BCD (Binary Coded Decimal) code. The accumulator is divided into two nibbles. Each nibble is adjusted to the BCD code and an internal carry (AC1) will be done if the low nibble of the accumulator is greater than 9. The BCD adjustment is done by adding 6 to the original value if the original value is greater than 9 or a carry (AC or C) is set; otherwise the original value remains unchanged. The result is stored in the data memory and only the carry flag (C) may be affected.

Operation If ACC.3~ACC.0 >9 or AC=1
then [m].3~[m].0 ← (ACC.3~ACC.0)+6, AC1=$\overline{AC}$
else [m].3~[m].0 ← (ACC.3~ACC.0), AC1=0
and
If ACC.7~ACC.4+AC1 >9 or C=1
then [m].7~[m].4 ← ACC.7~ACC.4+6+AC1,C=1
else [m].7~[m].4 ← ACC.7~ACC.4+AC1,C=C

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	√

DEC [m] Decrement data memory

Description Data in the specified data memory is decremented by 1.

Operation [m] ← [m]–1

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

DECA [m] Decrement data memory and place result in the accumulator

Description Data in the specified data memory is decremented by 1, leaving the result in the accumulator. The contents of the data memory remain unchanged.

Operation ACC ← [m]–1

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

HALT Enter power down mode

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

Operation Program Counter ← Program Counter+1
PDF ← 1
TO ← 0

Affected flag(s)

TO	PDF	OV	Z	AC	C
0	1	—	—	—	—

INC [m] Increment data memory

Description Data in the specified data memory is incremented by 1

Operation [m] ← [m]+1

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

INCA [m] Increment data memory and place result in the accumulator

Description Data in the specified data memory is incremented by 1, leaving the result in the accumulator. The contents of the data memory remain unchanged.

Operation ACC ← [m]+1

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

JMP addr Directly jump

Description The program counter are replaced with the directly-specified address unconditionally, and control is passed to this destination.

Operation Program Counter ←addr

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

MOV A,[m] Move data memory to the accumulator

Description The contents of the specified data memory are copied to the accumulator.

Operation ACC ← [m]

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

MOV A,x Move immediate data to the accumulator

Description The 8-bit data specified by the code is loaded into the accumulator.

Operation ACC ← x

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

MOV [m],A Move the accumulator to data memory

Description The contents of the accumulator are copied to the specified data memory (one of the data memories).

Operation [m] ←ACC

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

NOP No operation

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

Operation Program Counter ← Program Counter+1

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

OR A,[m] Logical OR accumulator with data memory

Description Data in the accumulator and the specified data memory (one of the data memories) perform a bitwise logical_OR operation. The result is stored in the accumulator.

Operation ACC ← ACC ″OR″ [m]

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

OR A,x Logical OR immediate data to the accumulator

Description Data in the accumulator and the specified data perform a bitwise logical_OR operation. The result is stored in the accumulator.

Operation ACC ← ACC ″OR″ x

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

ORM A,[m] Logical OR data memory with the accumulator

Description Data in the data memory (one of the data memories) 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)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

RET Return from subroutine

Description The program counter is restored from the stack. This is a 2-cycle instruction.

Operation Program Counter ← Stack

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

RET A,x Return and place immediate data in the accumulator

Description The program counter is restored from the stack and the accumulator loaded with the specified 8-bit immediate data.

Operation Program Counter ← Stack
ACC ← x

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

RETI Return from interrupt

Description The program counter is restored from the stack, and interrupts are enabled by setting the EMI bit. EMI is the enable master (global) interrupt bit.

Operation Program Counter ← Stack
EMI ← 1

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

RL [m] Rotate data memory left

Description The contents of the specified data memory are rotated 1 bit left with bit 7 rotated into bit 0.

Operation [m].(i+1) ← [m].i; [m].i:bit i of the data memory (i=0~6)
[m].0 ← [m].7

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

RLA [m] Rotate data memory left and place result in the accumulator

Description Data in the specified data memory is rotated 1 bit left with bit 7 rotated into bit 0, leaving the rotated result in the accumulator. The contents of the data memory remain unchanged.

Operation ACC.(i+1) ← [m].i; [m].i:bit i of the data memory (i=0~6)
ACC.0 ← [m].7

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

RLC [m] Rotate data memory left through carry

Description The contents of the specified data memory and the carry flag are rotated 1 bit left. Bit 7 replaces the carry bit; the original carry flag is rotated into the bit 0 position.

Operation [m].(i+1) ← [m].i; [m].i:bit i of the data memory (i=0~6)
[m].0 ← C
C ← [m].7

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	√

RLCA [m] Rotate left through carry and place result in the accumulator

Description Data in the specified data memory and the carry flag are rotated 1 bit left. Bit 7 replaces the carry bit and the original carry flag is rotated into bit 0 position. The rotated result is stored in the accumulator but the contents of the data memory remain unchanged.

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	√

RR [m] Rotate data memory right

Description The contents of the specified data memory are rotated 1 bit right with bit 0 rotated to bit 7.

Operation [m].i ← [m].(i+1); [m].i:bit i of the data memory (i=0~6)
[m].7 ← [m].0

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

RRA [m] Rotate right and place result in the accumulator

Description Data in the specified data memory is rotated 1 bit right with bit 0 rotated into bit 7, leaving the rotated result in the accumulator. The contents of the data memory remain unchanged.

Operation ACC.(i) ← [m].(i+1); [m].i:bit i of the data memory (i=0~6)
ACC.7 ← [m].0

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

RRC [m] Rotate data memory right through carry

Description The contents of the specified data memory and the carry flag are together rotated 1 bit right. Bit 0 replaces the carry bit; the original carry flag is rotated into the bit 7 position.

Operation [m].i ← [m].(i+1); [m].i:bit i of the data memory (i=0~6)
[m].7 ← C
C ← [m].0

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	√

RRCA [m] Rotate right through carry and place result in the accumulator

Description Data of the specified data memory and the carry flag are rotated 1 bit right. Bit 0 replaces the carry bit and the original carry flag is rotated into the bit 7 position. The rotated result is stored in the accumulator. The contents of the data memory remain unchanged.

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	√

SBC A,[m] Subtract data memory and carry from the accumulator

Description The contents of the specified data memory and the complement of the carry flag are subtracted from the accumulator, leaving the result in the accumulator.

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	√	√	√	√

SBCM A,[m] Subtract data memory and carry from the accumulator

Description The contents of the specified data memory and the complement of the carry flag are subtracted from the accumulator, leaving the result in the data memory.

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	√	√	√	√

SDZ [m] Skip if decrement data memory is 0

Description The contents of the specified data memory are decremented by 1. If the result is 0, the next instruction is skipped. If the result is 0, the following instruction, fetched during the current instruction execution, is discarded and a dummy cycle is replaced to get the proper instruction (2 cycles). Otherwise proceed with the next instruction (1 cycle).

Operation Skip if ([m]–1)=0, [m] ← ([m]–1)

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

SDZA [m] Decrement data memory and place result in ACC, skip if 0

Description The contents of the specified data memory are decremented by 1. If the result is 0, the next instruction is skipped. The result is stored in the accumulator but the data memory remains unchanged. If the result is 0, the following instruction, fetched during the current instruction execution, is discarded and a dummy cycle is replaced to get the proper instruction (2 cycles). Otherwise proceed with the next instruction (1 cycle).

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

SET [m] Set data memory

Description Each bit of the specified data memory is set to 1.

Operation [m] ← FFH

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

SET [m]. i Set bit of data memory

Description Bit i of the specified data memory is set to 1.

Operation [m].i ← 1

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

SIZ [m] Skip if increment data memory is 0

Description The contents of the specified data memory are incremented by 1. If the result is 0, the following instruction, fetched during the current instruction execution, is discarded and a dummy cycle is replaced to get the proper instruction (2 cycles). Otherwise proceed with the next instruction (1 cycle).

Operation Skip if ([m]+1)=0, [m] ← ([m]+1)

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

SIZA [m] Increment data memory and place result in ACC, skip if 0

Description The contents of the specified data memory are incremented by 1. If the result is 0, the next instruction is skipped and the result is stored in the accumulator. The data memory remains unchanged. If the result is 0, the following instruction, fetched during the current instruction execution, is discarded and a dummy cycle is replaced to get the proper instruction (2 cycles). Otherwise proceed with the next instruction (1 cycle).

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

SNZ [m].i Skip if bit i of the data memory is not 0

Description If bit i of the specified data memory is not 0, the next instruction is skipped. If bit i of the data memory is not 0, the following instruction, fetched during the current instruction execution, is discarded and a dummy cycle is replaced to get the proper instruction (2 cycles). Otherwise proceed with the next instruction (1 cycle).

Operation Skip if [m].i≠0

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

SUB A,[m] Subtract data memory from the accumulator

Description The specified data memory is subtracted from the contents of the accumulator, leaving the result in the accumulator.

Operation ACC ← ACC+[$\overline{m}$]+1

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	√	√	√	√

SUBM A,[m] Subtract data memory from the accumulator

Description The specified data memory is subtracted from the contents of the accumulator, leaving the result in the data memory.

Operation [m] ← ACC+[$\overline{m}$]+1

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	√	√	√	√

SUB A,x Subtract immediate data from the accumulator

Description The immediate data specified by the code is subtracted from the contents of the accumulator, leaving the result in the accumulator.

Operation ACC ← ACC+$\overline{x}$+1

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	√	√	√	√

SWAP [m] Swap nibbles within the data memory

Description The low-order and high-order nibbles of the specified data memory (1 of the data memories) are interchanged.

Operation [m].3~[m].0 ↔ [m].7~[m].4

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

SWAPA [m] Swap data memory and place result in the accumulator

Description The low-order and high-order nibbles of the specified data memory are interchanged, writing the result to 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)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

SZ [m] Skip if data memory is 0

Description If the contents of the specified data memory are 0, the following instruction, fetched during the current instruction execution, is discarded and a dummy cycle is replaced to get the proper instruction (2 cycles). Otherwise proceed with the next instruction (1 cycle).

Operation Skip if [m]=0

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

SZA [m] Move data memory to ACC, skip if 0

Description The contents of the specified data memory are copied to the accumulator. If the contents is 0, the following instruction, fetched during the current instruction execution, is discarded and a dummy cycle is replaced to get the proper instruction (2 cycles). Otherwise proceed with the next instruction (1 cycle).

Operation Skip if [m]=0

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

SZ [m].i Skip if bit i of the data memory is 0

Description If bit i of the specified data memory is 0, the following instruction, fetched during the current instruction execution, is discarded and a dummy cycle is replaced to get the proper instruction (2 cycles). Otherwise proceed with the next instruction (1 cycle).

Operation Skip if [m].i=0

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

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

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

TABRDC [m] Move the ROM code (current page) to TBLH and data memory (ROM code TBHP is disabled)

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

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

TABRDL [m] Move the ROM code (last page) to TBLH and data memory

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

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

XOR A,[m] Logical XOR accumulator with data memory

Description Data in the accumulator and the indicated data memory perform a bitwise logical Exclusive_OR operation and the result is stored in the accumulator.

Operation ACC ← ACC ″XOR″ [m]

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

XORM A,[m] Logical XOR data memory with the accumulator

Description Data in the indicated data memory and the accumulator perform a bitwise logical Exclusive_OR operation. The result is stored in the data memory. The 0 flag is affected.

Operation [m] ← ACC ″XOR″ [m]

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

XOR A,x Logical XOR immediate data to the accumulator

Description Data in the accumulator and the specified data perform a bitwise logical Exclusive_OR operation. The result is stored in the accumulator. The 0 flag is affected.

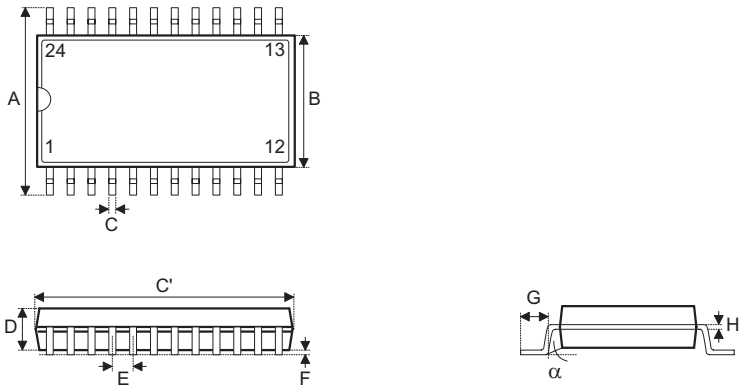
Operation ACC ← ACC ″XOR″ x

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

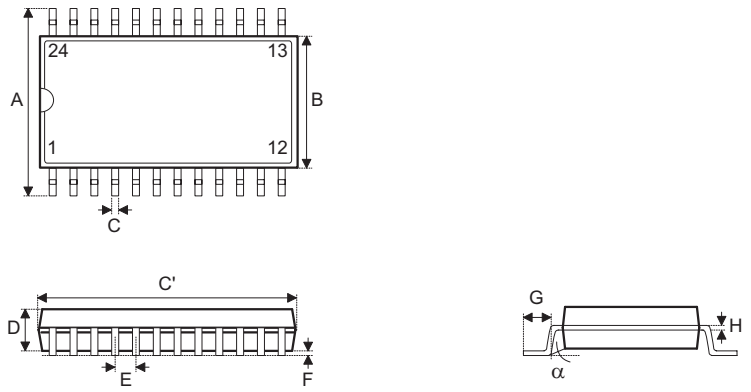
Package Information

24-pin SSOP (150mil) Outline Dimensions



Symbol	Dimensions in mil		
	Min.	Nom.	Max.
A	228	—	244
B	150	—	157
C	8	—	12
C′	335	—	346
D	54	—	60
E	—	25	—
F	4	—	10
G	22	—	28
H	7	—	10
α	0°	—	8°

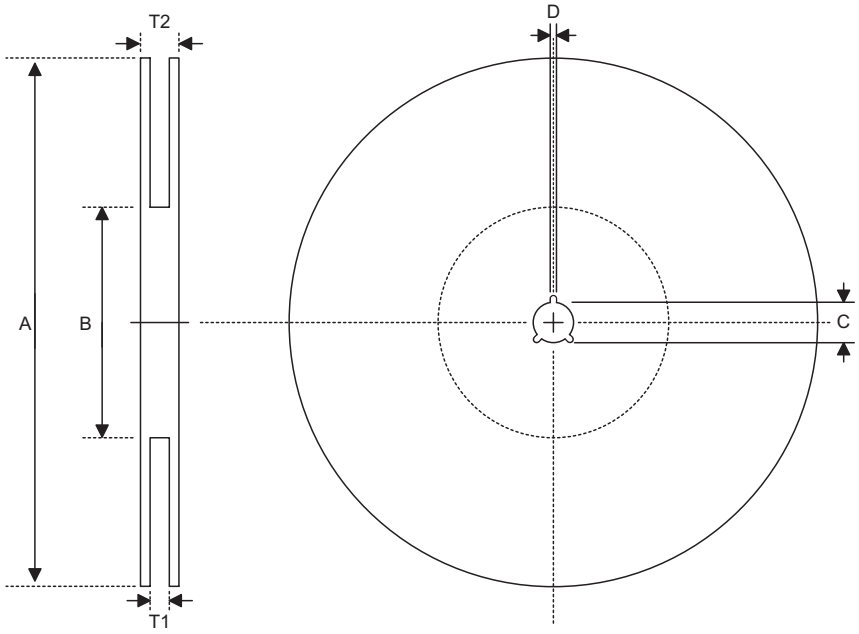
24-pin SOP (300mil) Outline Dimensions



Symbol	Dimensions in mil		
	Min.	Nom.	Max.
A	394	—	419
B	290	—	300
C	14	—	20
C′	590	—	614
D	92	—	104
E	—	50	—
F	4	—	—
G	32	—	38
H	4	—	12
α	0°	—	10°

Product Tape and Reel Specifications

Reel Dimensions



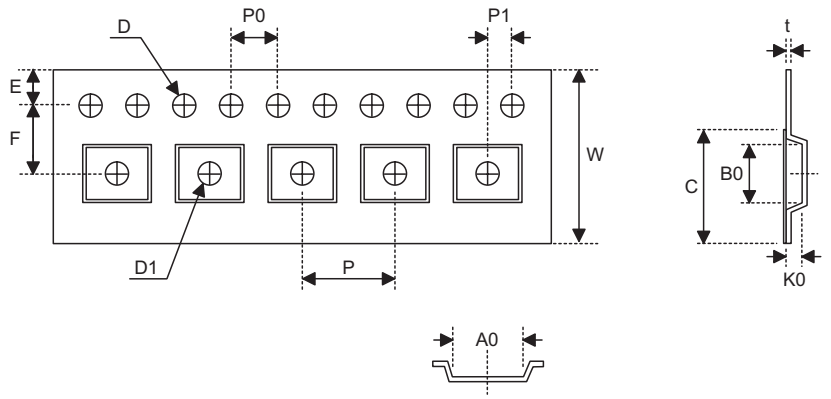
SSOP 24S (150mil)

Symbol	Description	Dimensions in mm
A	Reel Outer Diameter	330±1
B	Reel Inner Diameter	62±1.5
C	Spindle Hole Diameter	13+0.5 −0.2
D	Key Slit Width	2±0.5
T1	Space Between Flange	16.8+0.3 −0.2
T2	Reel Thickness	22.2±0.2

SOP 24W

Symbol	Description	Dimensions in mm
A	Reel Outer Diameter	330±1
B	Reel Inner Diameter	62±1.5
C	Spindle Hole Diameter	13+0.5 −0.2
D	Key Slit Width	2±0.5
T1	Space Between Flange	24.8+0.3 −0.2
T2	Reel Thickness	30.2±0.2

Carrier Tape Dimensions



SSOP 24S (150mil)

Symbol	Description	Dimensions in mm
W	Carrier Tape Width	16+0.3 −0.1
P	Cavity Pitch	8±0.1
E	Perforation Position	1.75±0.1
F	Cavity to Perforation (Width Direction)	7.5±0.1
D	Perforation Diameter	1.5+0.1
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	6.5±0.1
B0	Cavity Width	9.5±0.1
K0	Cavity Depth	2.1±0.1
t	Carrier Tape Thickness	0.3±0.05
C	Cover Tape Width	13.3

SOP 24W

Symbol	Description	Dimensions in mm
W	Carrier Tape Width	24±0.3
P	Cavity Pitch	12±0.1
E	Perforation Position	1.75±0.1
F	Cavity to Perforation (Width Direction)	11.5±0.1
D	Perforation Diameter	1.55+0.1
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	10.9±0.1
B0	Cavity Width	15.9±0.1
K0	Cavity Depth	3.1±0.1
t	Carrier Tape Thickness	0.35±0.05
C	Cover Tape Width	21.3

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: 021-6485-5560
Fax: 021-6485-0313
http://www.holtek.com.cn

Holtek Semiconductor Inc. (Shenzhen Sales Office)
5/F, Unit A, Productivity Building, Cross of Science M 3rd Road and Gaoxin M 2nd Road, Science Park, Nanshan District, Shenzhen, China 518057
Tel: 0755-8616-9908, 8616-9308
Fax: 0755-8616-9533

Holtek Semiconductor Inc. (Beijing Sales Office)
Suite 1721, Jinyu Tower, A129 West Xuan Wu Men Street, Xicheng District, Beijing, China 100031
Tel: 010-6641-0030, 6641-7751, 6641-7752
Fax: 010-6641-0125

Holtek Semiconductor Inc. (Chengdu Sales Office)
709, Building 3, Champagne Plaza, No.97 Dongda Street, Chengdu, Sichuan, China 610016
Tel: 028-6653-6590
Fax: 028-6653-6591

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

Copyright © 2006 by HOLTEK SEMICONDUCTOR INC.

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