HT82A832R
Basic USB Phone OTP MCU

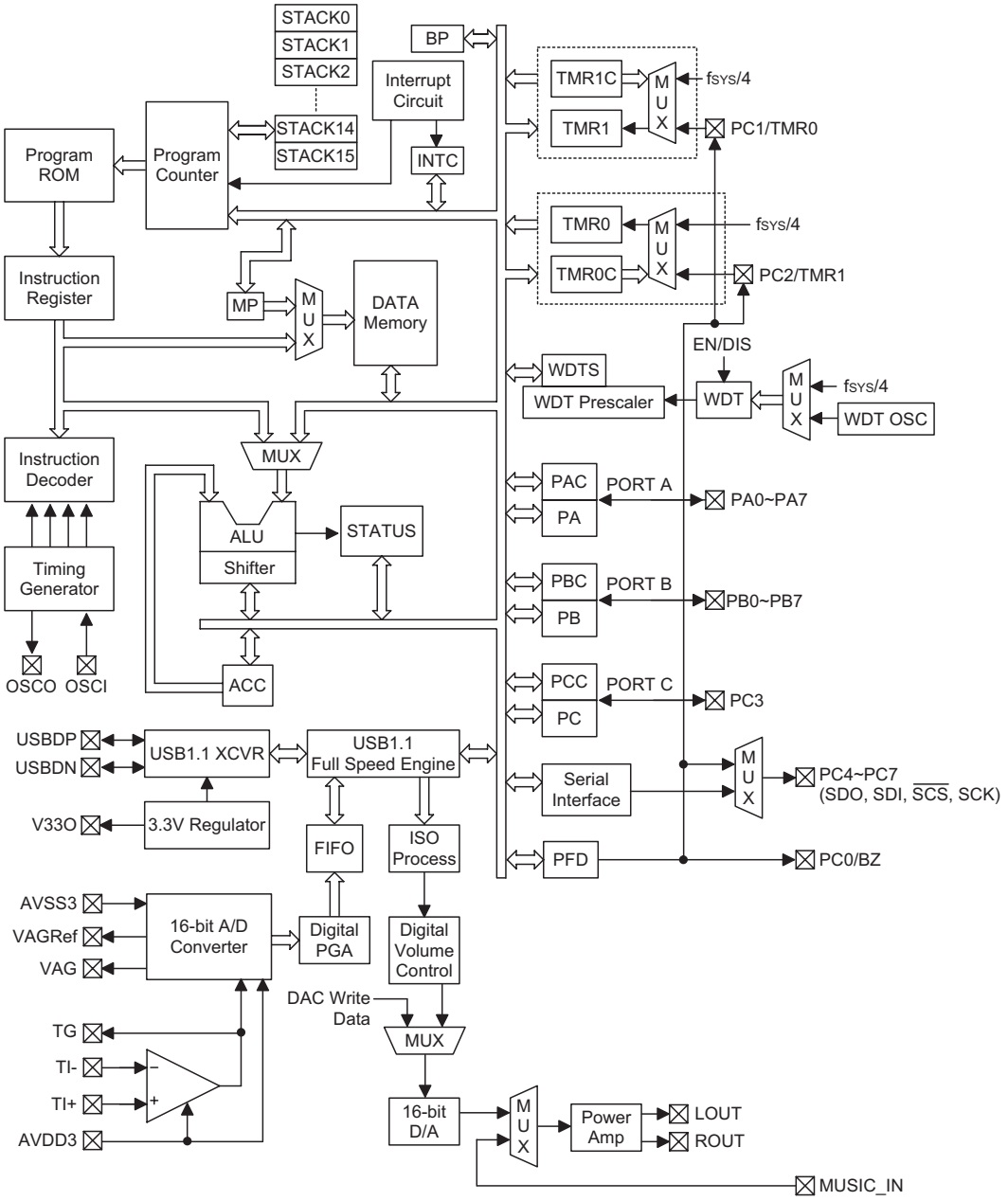
Features

- Operating voltage: f_{SYS} = 6M/12MHz: 4.0V~5.5V
- 24 bidirectional I/O lines (max.)
- Two 16-bit programmable timer/event counters and overflow interrupts
- 4096×15 program memory ROM
- 192×8 data memory RAM (Bank 0)
- USB 2.0 full speed compatible
- USB spec V1.1 full speed operation and USB audio device class spec V1.0
- Embedded high-performance 16 bit PCM ADC
- Built-in Digital PGA (Programmable Gain Amplifier)
- 48kHz/8kHz sampling rate for audio playback controlled by software option
- 8kHz audio recording sampling rate
- Embedded class AB speaker driver power amplifier
- Embedded High Performance 16 bit audio DAC
- Supports audio playback digital volume control
- 5 endpoints supported (endpoint 0 included)
- Supports 1 Control, 2 Interrupt, 2 Isochronous transfer
- Two hardware implemented Isochronous transfers
- Total FIFO size: 464 bytes
  (8, 8, 384, 32, 32 for EP0~EP4)
- Programmable frequency divider (PFD)
- Integrated SPI hardware circuit
- Play/Record Interrupt
- HALT and wake-up features reduce power consumption
- Watchdog Timer
- 16-level subroutine nesting
- Bit manipulation instruction
- 15-bit table read instruction
- 63 powerful instructions
- All instructions executed within one or two machine cycles
- Low voltage reset function (3.0V±0.3V)
- 48-pin SSOP/LQFP package

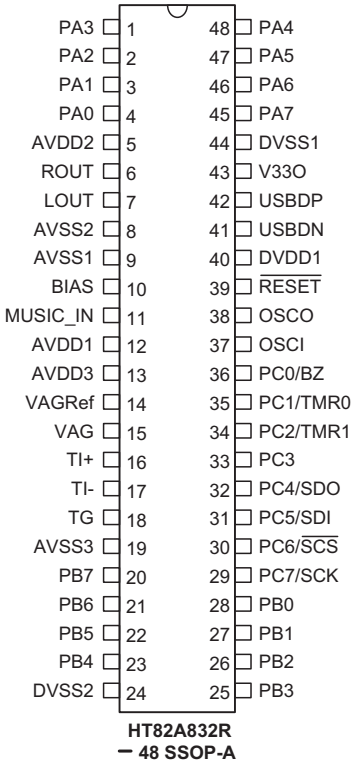
General Description

The HT82A832R is an 8-bit high performance RISC-like microcontroller designed for USB Phone product applications. The HT82A832R combines a 16-bit PCM ADC, USB transceiver, SIE (Serial Interface Engine), audio class processing unit, FIFO and an 8-bit MCU into a single chip. The DAC in the HT82A832R operates at a sampling rate of 48kHz/8kHz and the 16-bit PCM ADC operates at 8kHz for the Microphone input. For the DAC, the HT82A832R has a digital programmable gain amplifier. The gain range is from -32dB to +6dB. For the ADC input, the digital gain range is from 0dB to 19.5dB.

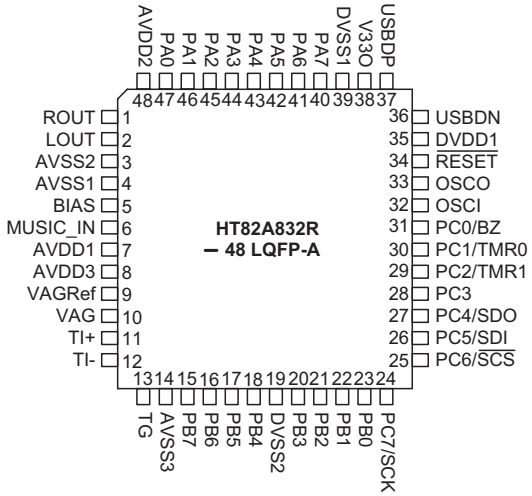
Block Diagram



Pin Assignment



HT82A832R
— 48 SSOP-A



HT82A832R
— 48 LQFP-A

Pin Description

Pin Name	I/O	Description
PA0~PA7	I/O	Bidirectional 8-bit input/output port. Each bit can be configured as a wake-up input by a configuration option. Software instructions determine if the pin is a CMOS output or Schmitt trigger input with or without a pull-high resistor (by configuration option).
AVDD2	—	Audio power amplifier positive power supply
ROUT	O	Right driver analog output
LOUT	O	Left driver analog output
AVSS2	—	Audio power amplifier negative power supply, ground
AVSS1	—	Audio DAC negative power supply, ground
BIAS	—	A capacitor should be connected to ground to increase half-supply stability
MUSIC_IN	I	Power amplifier signal source if register bit SELW =″1″. The analog signal input will amplify by the power amp then output to ROUT and LOUT at the same time.
AVDD1	—	Audio DAC positive power supply
AVDD3	—	ADC positive power supply
VAGRef	O	ADC analog ground reference voltage (should left open or connect a bypass capacitor (Ex:100 pF) to ground)
VAG	O	ADC analog ground voltage (should connect a bypass capacitor (Ex:1 uF) to ground)
TI+	I	OP AMP non-inverting input
TI-	I	OP AMP inverting input
TG	O	OP AMP gain setting output

Pin Name	I/O	Description
AVSS3	—	ADC negative power supply, ground
PB7~PB0	I/O	Bidirectional 8-bit input/output port. Software instructions determine if the pin is a CMOS output or Schmitt trigger input with pull-high resistor (determined by pull-high options, bit option).
DVSS2	—	Negative digital & I/O power supply, ground
PC7/SCK	I/O	Can be software optioned as a bidirectional input/output or serial interface clock signal.
PC6/$\overline{SCS}$	I/O	Can be software optioned as a bidirectional input/output or serial interface slave select signal.
PC5/SDI	I/O or I	Can be software optioned as a bidirectional input/output or serial data input.
PC4/SDO	I/O or O	Can be software optioned as a bidirectional input/output or serial data output.
PC3	I/O	Bidirectional I/O lines. Software instructions determine if the pin is a CMOS output or Schmitt trigger input with pull-high resistor (determined by configuration option).
PC2/TMR1, PC1/TMR0	I/O	Software instructions determine if the pin is a CMOS output or Schmitt trigger input with pull-high resistor. TMR0, TMR1 are pin shared with PC1, PC2 respectively
PC0/BZ	I/O or O	Can be software optioned as a bidirectional input/output or as a PFD output.
OSCI OSCO	I O	OSCI, OSCO are connected to an 6MHz or 12MHz crystal/resonator (determined by software instructions) for the internal system clock
$\overline{RESET}$	I	Schmitt trigger reset input, active low
DVDD1	—	Positive digital power supply
USBDN	I/O	USBD- line. The USB function is controlled by a software control register
USBDP	I/O	USBD+ line. The USB function is controlled by a software control register
V33O	O	3.3V regulator output
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	—	4.0	5.0	5.5	V
I_{DD}	Operating Current	5V	No load, f_{SYS}=12MHz, ADC On, DAC On	—	12	—	mA
		5V	No load, f_{SYS}=12MHz, ADC Off, DAC Off	—	8	—	mA
I_{SUS}	Suspend Current	5V	No load, system HALT, USB transceiver and 3.3V regulator on	—	330	—	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	40	80	kΩ
V_{LVR}	Low Voltage Reset	5V	—	2.7	3.0	3.3	V
V_{V33O}	3.3V Regulator Output	5V	I_{V33O}=–5mA	3.0	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_{DA}	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	—	
PCM ADC							
SNR_{AD}	Signal to Noise Ratio	5V	—	—	77	—	dB
VAG	Reference Voltage	5V	—	—	2	—	V
V_{PEAK}	Peak Single Frequency Tone Amplitude without Clipping	5V	—	—	1.575	—	V_{PK}

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 microcontroller is sourced 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

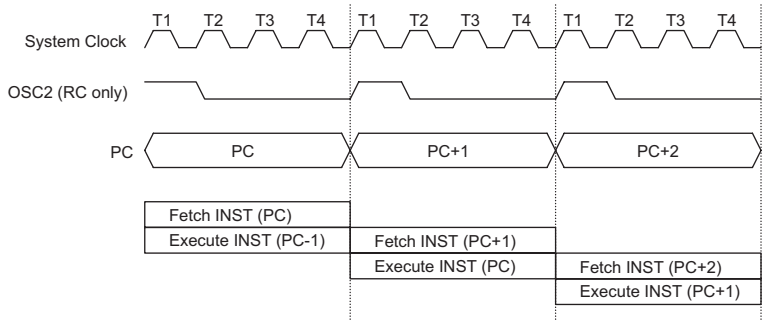
The program counter (PC) controls the sequence in which the instructions stored in the program ROM are executed and its contents specify a full range of program memory.

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



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
USB 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
Play 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
Record 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

the PCL performs a short jump. The destination will be within the current program ROM page.

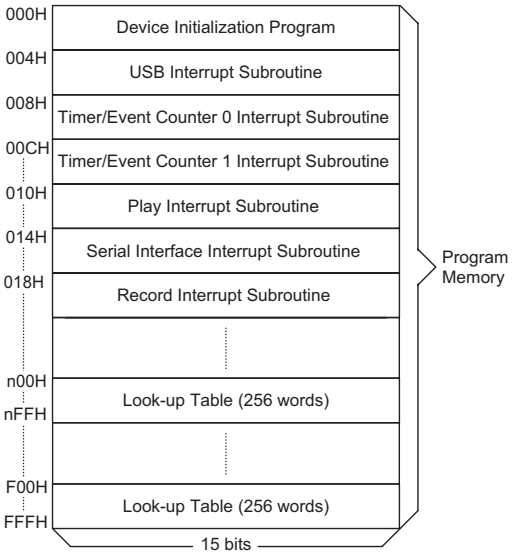
When a control transfer takes place, an additional dummy cycle is required.

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 4096×15 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 area 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.
- Location 010H
  This area is reserved for the play interrupt service program. If play data is valid, and the interrupt is enabled and the stack is not full, the program begins execution at location 010H.
- Location 014H
  This area is reserved for when 8 bits of data have been received or transmitted successfully from the serial interface. If the related interrupts are enabled, and the stack is not full, the program begins execution at location 014H.
- Location 018H
  This area is reserved for the record interrupt service program. If the record data valid, the interrupt is enabled and the stack is not full, the program begins execution at location 018H.
- Table location
  Any location in the program memory can be used as a look-up table. 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 instruction "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 register TBLP (07H) in the last page (0F00H~0FFFH).



Note: n ranges from 1 to F

Program Memory

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
TBHP register bit3~bit0 when TBHP is enabled
P11~P8: Current program counter bits when TBHP is disabled
@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, 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 registers. (If the configuration option TBHP is disabled, the value in TBHP has no effect). 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 is likely to be changed by the table read instruction used in the ISR. As a result errors may occur. In other words, using the table read instruction in the main routine and in the ISR simultaneously should be avoided. However, if the table read instruction has to be applied in both the main routine and the ISR, the interrupt should be disabled prior to the table read instruction.

It will not be enabled until the TBLH has been backed up. All table related instructions require two cycles to complete the operation. These areas may function as normal program memory depending on the requirements.

Once the TBHP is enabled, the instruction ″TABRDC [m]″ reads the ROM data as defined by the 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 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, 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 stack pointer 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

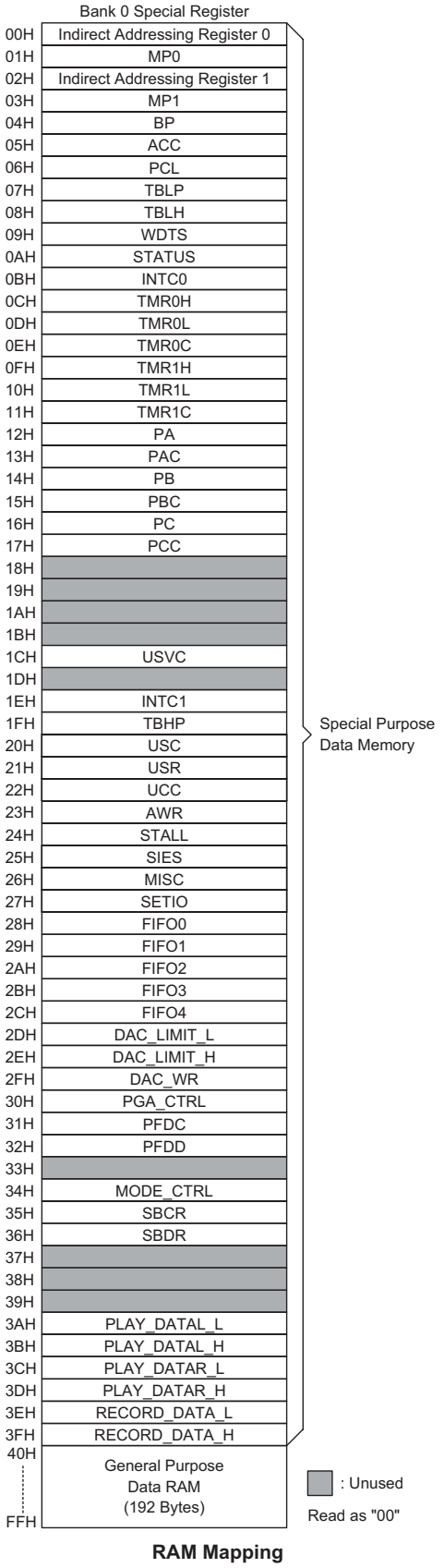
The data memory is divided into two functional groups: namely; special function registers and general purpose data memory, Bank 0: 192×8 bits. Most are read/write, but some are read only.

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, PB;14H, PC;16H), I/O control registers (PAC;13H, PBC;15H, PCC;17H). 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~FIFO4 register (28H~2CH). DAC_Limit_L register (2DH), DAC_Limit_H register (2EH), DAC_WR register (2FH). PGA_CTRL register (30H). PFD control register (PFDC;31H). PFD data register (PFDD;32H). MODE_CTRL register (34H). Serial bus control register (SBCR;35H), serial bus data register (SBDR;36H). Play data left channel (PLAY_DATAL_L;3AH, PLAY_DATAL_H;3BH), play data right channel (PLAY_DATAR_L;3CH, PLAY_DATAR_H;3DH). Record data (RECORD_DATA_L; 3EH, RECORD_DATA_H; 3FH).

The remaining space before the 40H is reserved for future expanded usage, reading these locations will return a result of ″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).



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

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

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 a USB interrupt, internal timer/event counter interrupts, play/record data valid interrupt and a serial interface interrupt. 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.

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.

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

When the interrupt is enabled, the stack is not full and the USB 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 the PC Host accesses the FIFO of the HT82A832R, the corresponding request bit of the USR is set, and a USB interrupt is triggered. So the user can easily determine which FIFO has been accessed. When the interrupt has been served, the corresponding bit should be cleared by firmware. When the HT82A832R receives a USB Suspend signal from the Host PC, the suspend line (bit0 of USC) of the HT82A832R is set and a USB interrupt is also triggered.

Also when the HT82A832R receives a Resume signal from the Host PC, the resume line (bit3 of USC) of the HT82A832R 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/Event 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 T1F is set, a subroutine call to location 0CH will occur. The

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

Bit No.	Label	Function
0	EPLAYI	Play interrupt (1=enable; 0=disable)
1	ESII	Control Serial interface interrupt (1=enable; 0=disable)
2	RECI	Record interrupt (1=enable; 0=disable)
3, 7	—	Unused bit, read as ″0″
4	PLAYF	Play interrupt request flag (1=active; 0=inactive)
5	SIF	Serial interface interrupt request flag (1=active; 0=inactive)
6	RECF	Record interrupt request flag (1=active; 0=inactive)

INTC1 (1EH) Register

related interrupt request flag (T1F) will be reset and the EMI bit cleared to disable further interrupts.

The play interrupt is initialized by setting the play interrupt request flag (bit 4 of INTC1), caused by a play data valid. When the interrupt is enabled, the stack is not full and the PLAYF is set, a subroutine call to location 10H will occur. The related interrupt request flag (PLAYF) will be reset and the EMI bit cleared to disable further interrupts. If PLAY_MODE (bit 3 of MODE_CTRL register) is set to ″1″, the play interrupt frequency will change to 8KHz, otherwise the interrupt frequency is 48kHz.

The serial interface interrupt is indicated by the interrupt flag (SIF; bit 5 of INTC1), that is generated by the reception or transfer of a complete 8-bits of data between the HT82A832R and the external device. The serial interface interrupt is controlled by setting the Serial interface interrupt control bit (ESII; bit 1 of INTC1). After the interrupt is enabled (by setting SBEN; bit 4 of SBCR), and the stack is not full and the SIF is set, a subroutine call to location 14H occurs.

The record interrupt is initialized by setting the record interrupt request flag (bit 6 of INTC1), caused by a record data valid. When the interrupt is enabled, the stack is not full and RECF is set, a subroutine call to location 18H will occur. The related interrupt request flag (RECF) will be reset and the EMI bit cleared to disable further interrupts. If ADC powered down (AD_ENB =1) or USB clock disabled (USBCKEN=0), the record interrupt will be disabled.

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.

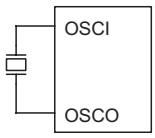
Interrupts, occurring in the interval between the rising edges of two consecutive T2 pulses, will be serviced on the latter of the two T2 pulses, if the corresponding interrupts are enabled. In the case of simultaneous requests the following table shows the priority that is applied. These can be masked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

Interrupt Source	Priority	Vector
USB interrupt	1	04H
Timer/Event Counter 0 overflow	2	08H
Timer/Event Counter 1 overflow	3	0CH
Play Interrupt	4	10H
Serial Interface Interrupt	5	14H
Record 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

The microcontroller contains an integrated oscillator circuit.



Crystal Oscillator

System Oscillator

This oscillator is designed for the system clock. The HALT mode stops the system oscillator and ignores any external signals 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. If preferred, a resonator can also be connected between OSCI and OSCO for oscillation to occur, but two external capacitors connected between OSCI, OSCO and ground 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 stops running, but the WDT oscillator still continues to run. The WDT oscillator can be disabled by a configuration option to conserve power.

Watchdog Timer – WDT

The WDT clock source is implemented by a dedicated RC oscillator (WDT oscillator) or the 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 a configuration option. However, 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.

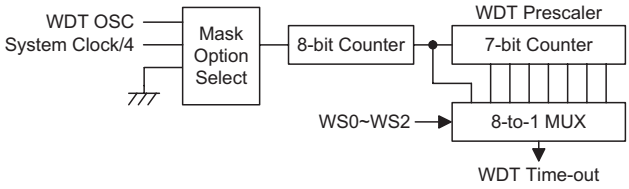
The WDT OSC period is typically 65μs. This time-out period may vary with temperature, VDD and process variations. The WDT OSC always keeps running in any operation mode.

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 HALT 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 of the WDTS is reserved for the DAC write mode.

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 program counter and stack pointer are reset to zero. To clear the contents of the WDT, there are three methods to be adopted, i.e., an external reset (a low level to $\overline{\text{RESET}}$), a software instruction, and a ″HALT″ instruction. There are two types of software instructions;

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



Watchdog Timer

″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 configuration option ″CLR WDT″ times selection option. If the ″CLR WDT″ is selected (i.e., CLR WDT times equal one), any execution of the ″CLR WDT″ instruction 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 a time-out.

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 a WDT time-out occurs, and causes a wake-up that only resets the program counter and stack pointer; 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 configuration 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 stack pointer, 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

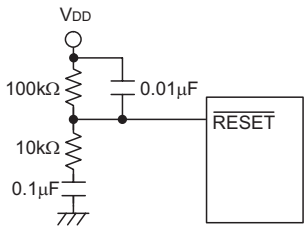
Note: ″u″ stands for ″unchanged″

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.

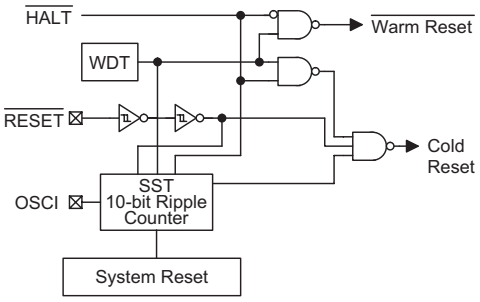
When a system reset occurs, the SST delay is added during the reset period. Any wake-up from HALT will enable the SST delay.

The functional unit chip reset status are shown below.

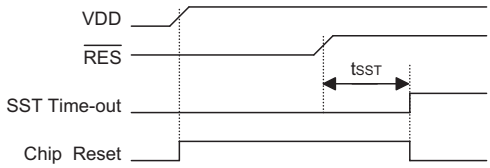
Program Counter	000H
Interrupt	Disable
WDT	Clear. After master reset, WDT begins counting
Timer/event Counter	Off
Input/output Ports	Input mode
Stack Pointer	Points to the top of the stack



Reset Circuit



Reset Configuration



Reset Timing Chart

The registers status are summarized in the following table.

Register	Reset (Power On)	WDT Time-out (Normal Operation)	RES Reset (Normal Operation)	RES Reset (HALT)	WDT Time-Out (HALT)*	USB Reset (Normal)	USB Reset (HALT)
MP0	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
MP1	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
Program Counter	000H	000H	000H	000H	000H	000H	000H
TBLP	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLH	-xxx xxxx	-uuu uuuu	-uuu uuuu	-uuu uuuu	-uuu uuuu	-uuu uuuu	-uuu uuuu
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
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
USVC	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	uuuu uuuu	uuuu uuuu	uuuu uuuu	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)
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
FIFO3	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	0000 0000	0000 0000
FIFO4	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
PGA_CTRL	0000 0000	0000 0000	0000 0000	0000 0000	0000 0000	00uu uuuu	00uu uuuu
PFDC	0000 0000	0000 0000	0000 0000	0000 0000	0000 0000	0uuu 0000	0uuu 0000
PFDD	0000 0000	0000 0000	0000 0000	0000 0000	0000 0000	0uuu 0000	0uuu 0000
MODE_CTRL	0000 0000	0000 0000	0000 0000	0000 0000	0000 0uuu	0000 0uuu	0000 0uuu
SBCR	0110 0000	0110 0000	0110 0000	0110 0000	uuuu uuuu	uuuu uuuu	uuuu uuuu
SBDR	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
PLAY_DATAL_L	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	uuuu uuuu	uuuu uuuu
PLAY_DATAL_H	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	uuuu uuuu	uuuu uuuu
PLAY_DATAR_L	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	uuuu uuuu	uuuu uuuu
PLAY_DATAR_H	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	uuuu uuuu	uuuu uuuu
RECORD_DATA_L	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	uuuu uuuu	uuuu uuuu
RECORD_DATA_H	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	uuuu uuuu	uuuu uuuu

Note: "*" stands for "warm reset"
"u" stands for "unchanged"
"x" stands for "unknown"
"-" 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 TMR0C (TMR1C) is the Timer/Event Counter 0 (1) control register, which defines the operating mode, counting enable or disable and an active edge.

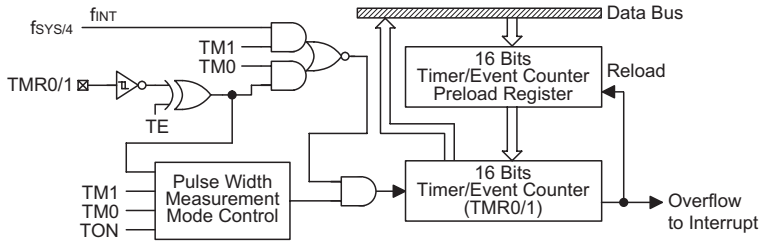
The TM0 and TM1 bits define the operation mode. The event count mode is used to count external events, which means that the clock source is from an external (TMR0, TMR1) pin. The timer mode functions as a normal timer with the clock source coming from the internal clock source. Finally, the pulse width 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 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 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.

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, 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.



Timer/Event Counter 0/1

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

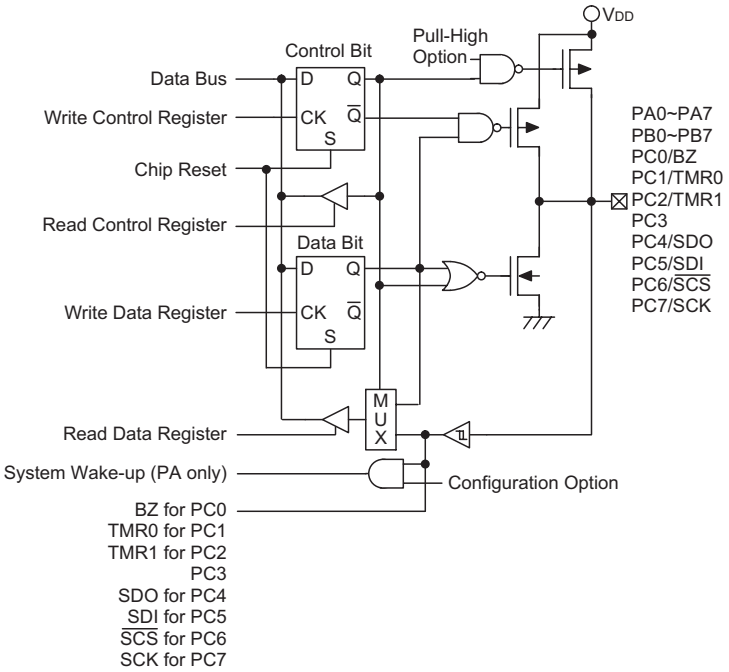
TMR0C (0EH), TMR1C (11H) Register

Input/Output Ports

There are 24 bidirectional input/output lines in the micro-controller, labeled from PA to PC, which are mapped to the data memory of [12H], [14H], [16H] 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). 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) 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. These control registers are mapped to locations 13H, 15H, 17H.

After a chip reset, these input/output lines remain at high levels or floating state (depending on the pull-high



Input/Output Ports

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) 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.

Low Voltage Reset – LVR (by Configuration Option)

The LVR option is 3.0V.

The microcontroller provides a 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}, the LVR will automatically reset the device internally.

The LVR includes the following specifications:

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

Suspend Wake-Up and Remote Wake-Up

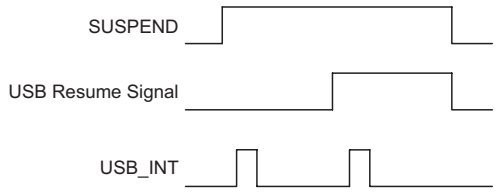
If there is no signal on the USB bus for over 3ms, the HT82A832R will go into a suspend mode. The Suspend line (bit 0 of the USC) will be set to ″1″ and a USB interrupt is triggered to indicate that the HT82A832R should jump to the suspend state to meet the requirements of the USB suspend current spec.

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

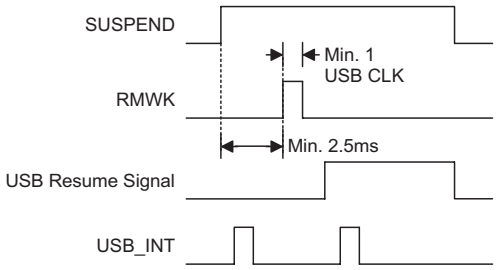
Also the user can further decrease the suspend current by setting SUSP2 (bit4 of the UCC).

When the resume signal is sent out by the host, the HT82A832R will be woken up by the USB interrupt and the Resume line (bit 3 of USC) will be set. In order to make the HT82A832R work properly, the firmware must set USBCKEN (bit 3 of UCC) to ″1″ and clear 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) will go to ″0″. So when the MCU is detecting the Suspend line (bit0 of USC), the condition of the Resume line should be noted and taken 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 receives the wake-up signal from the HT82A832R, it will send a Resume signal to the device. The timing is as follows:



USB Interface

The HT82A832R device has 5 Endpoints (EP0~EP4). EP0 supports Control transfer. EP1 and EP4 support Interrupt transfer. EP2 supports Isochronous out transfer. EP3 supports Isochronous in transfer.

These registers, including USC (20H), USR (21H), UCC (22H), AWR (23H), STALL (24H), SIES (25H), MISC (26H), FIFO0 (28H), FIFO1 (29H), FIFO2 (2AH), FIFO3 (2BH), FIFO4 (2CH) are used for the USB function.

The FIFO size of each FIFO is 8 bytes (FIFO0), 8 bytes (FIFO1), 384 bytes (FIFO2), 32 bytes (FIFO3), 32 bytes (FIFO4). The total is 464 bytes.

URD (bit7 of USC) is the 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 that the USB bus has entered the suspend mode. The USB interrupt is also triggered when this bit changes from low to high.
1	RMWK	R/W	0	USB remote wake-up command. It is set by MCU to force the USB host to leave the suspend mode.
2	URST	R/W	0	USB reset indication. This bit is set/cleared by the USB SIE. This bit is used to detect a USB reset event on the USB bus. When this bit is set to ″1″, this indicates that a USB reset has occurred and that a USB interrupt will be initialized.
3	RESUME	R	0	USB resume indication. When the USB leaves the 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 detect the suspend state, the MCU should set USBCKEN and clear SUSP2 (in the UCC register) to enable the SIE detect function. RESUME will be cleared when the SUSP goes to ″0″. When the MCU is detecting the SUSP, the condition of RESUME (causes the MCU to wake-up) should be noted and taken into consideration.
4	V33C	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 the serial bus (USB). The endpoint request flags (EP0F, EP1F, EP2F, EP3F, EP4F) 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 and the stack is not full). When the active endpoint request flag is serviced, 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 that endpoint 0 has been accessed and a USB interrupt will occur. When the interrupt has been serviced, this bit should be cleared by software.
1	EP1F	R/W	0	When this bit is set to ″1″ (set by SIE), it indicates that endpoint 1 has been accessed and a USB interrupt will occur. When the interrupt has been serviced, this bit should be cleared by software.
2	EP2F	R/W	0	When this bit is set to ″1″ (set by SIE), it indicates that endpoint 2 has been accessed and a USB interrupt will occur. When the interrupt has been serviced, this bit should be cleared by software.
3	EP3F	R/W	0	When this bit is set to ″1″ (set by SIE), it indicates that endpoint 3 has been accessed and a USB interrupt will occur. When the interrupt has been serviced, this bit should be cleared by software.
4	EP4F	R/W	0	When this bit is set to ″1″ (set by SIE), it indicates that endpoint 4 has been accessed and a USB interrupt will occur. When the interrupt has been serviced, this bit should be cleared by software.
5~7	—	—	—	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 a USB clock control bit (USBCKEN), a second suspend mode control bit (SUSP2) and a system clock selection bit (SYSCLK).

The endpoint selection is determined 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: Select endpoint 3 FIFO 100: Select endpoint 4 FIFO 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 function 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 the suspend mode. In normal mode, clear this bit to ″0″ In the HALT mode, set this bit to ″1″ to reducing power consumption.
5	f_{SYS}24MHz	R/W	0	This bit is used to define if the MCU system clock comes form an external OSC or comes from the PLL output 24MHz clock. 0: system clock sourced from OSC 1: system clock sourced from the PLL output 24MHz
6	SYSCLK	R/W	0	This bit is used to specify the MCU system clock oscillator frequency. For a 6MHz crystal oscillator or resonator, set this bit to ″1″. For a 12MHz crystal oscillator or resonator, clear this bit to ″0″.

Note: Isochronous endpoint 2 and endpoint 3 are implemented by hardware, so FIFO2 and FIFO3 cannot read/write via firmware.

UCC (22H) Register

The AWR register contains the current address and a remote wake up function control bit. The initial value of AWR is ″00H″. The address value extracted from the the USB command has not to be loaded into this register until the SETUP stage has 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	0000000	USB device address

AWR (23H) Register

The STALL register shows if 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 register will be cleared by a USB reset signal.

Bit No.	Label	R/W	Power-on	Functions
0~4	STL0~STL4	R/W	00000	Set by the user when related USB endpoints were stalled. Cleared by a USB reset and a Setup Token event.
5~7	STL5~STL7	—	000	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 to 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 the PC host has successfully read the data from the device by an 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, the firmware has to clear this bit after a next valid SETUP token is received.
1	ERR	R/W	0	This bit is used to indicate that some errors have occurred when 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 the 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 an IN token.
4	NAK	R	0	This bit is used to indicate the SIE is a transmitted NAK signal to the host in response to the 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 pakcage active flag, low active.
7	NMI	R/W	0	NAK token interrupt mask flag. If this bit set, when the device sent a NAK token to the host, an interrupt will be disabled. Otherwise if this bit is cleared, when the device sends a NAK token to the host, it will enter the interrupt sub-routine.

SIES (25H) Register

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

Bit No.	Label	R/W	Power-on	Functions
0	REQUEST	R/W	0	After setting the status of the desired one, FIFO can be requested by setting this bit high . After finishing, this bit must be set low.
1	TX	R/W	0	To represent the direction and transition end MCU access. When set to logic 1, the MCU desires to write data to the FIFO. After finishing, this bit must be set to logic 0 before terminating request to represent transition end. For an MCU read operation, this bit must be set to logic 0 and set to logic 1 after finishing.
2	CLEAR	R/W	0	MCU requests to clear the FIFO, even if the FIFO is not ready. After clearing the FIFO, the USB interface will send force_tx_err to tell the Host that data under-run if the Host wants to read data.
3	ISO_IN_EN	R/W	0	Enables the isochronous in pipe interrupt.
4	ISO_OUT_EN	R/W	0	Enables the isochronous out pipe interrupt.
5	SETCMD	R/W	0	To show that the data in the FIFO is a setup command. This bit will remain in this state until the next one enters the FIFO.
6	READY	R	0	To show that the desired FIFO is ready
7	LEN0	R	0	To show that the host sent a 0-sized packet to the MCU. This bit must be cleared by a read action to the corresponding FIFO.

MISC (26H) Register

Bit No.	Label	R/W	Power-on	Functions
0	DATATG*	R/W	0	DATA token toggle bit
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	SETIO3**	R/W	1	Set endpoint3 input or output pipe (1/0), default input pipe(1)
4	SETIO4**	R/W	1	Set endpoint4 input or output pipe (1/0), default input pipe(1)
5~7	—	—	—	Undefined bit, read as ″0″

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

**It is only required to set the data pipe as an input pile or output pile. The purpose of this function is to avoid the host sending a abnormal IN or OUT token and disabling the endpoint.

SETIO (27H) Register, USB Endpoint 1 ~ Endpoint 4 Set IN/OUT Pipe Register

The speaker output volume and speaker mute/un-mute are controlled by the USB Speaker Volume Control register. The range of the volume is set from 6 dB to -32 dB by software.

Speaker mute control:

MUTEB=0: Mute Speaker output

MUTEB=1: Normal

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 (1CH) Register

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 Volume Control Table

Label	R/W	Power-on	Functions
FIFO0~ FIFO4	R/W	xxH	EPi accessing register (i = 0~4). When an endpoint is disabled, the corresponding accessing register should be disabled.

FIFO0~4 (28H~2CH) USB Endpoint Accessing Register 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 the Host over the range defined by DAC_Limit_L and DAC_Limit_H, the output digital code to DAC will be clamped.

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

Example to set the DAC output limit value:

```
;------------------------------------------------------------
; 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 a popping noise from the speaker output, the power amplifier should output a value of VDD/2 (send 8000H to DAC) during the initial power on state. If the software is 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 the 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,00001111b
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.

Digital PGA

Bit No.	Label	Functions
0~5	PGA0~PGA5	There are six bits to control the digital PGA (0~19.5 dB). The PGA is a digital amplifier used to amplify the 16-bit data that comes from the PCM ADC. The PGA value versus gain relationship is shown in the follow table.
6	—	Undefined bit, read as ″0″.
7	MUTE_MKB	Microphone mute Control: MUTE_MKB =0: Mute microphone input. MUTE_MKB =1: Normal.

PGA_CTRL (30H) Register

PGA_CRTL Value (PGA5~PGA0)	Gain (dB)
000000	≈ 0
000001	≈ 0.5
: :	: :
100111	≈ 19.5
101000	≈ 19.5
: :	: :
111111	≈ 19.5

PFD Control

Label	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PFDC	0	PRES1	PRES0	PFDEN	0	0	PFD_IO	SELW
PFDD	PFDD7	PFDD6	PFDD5	PFDD4	PFDD3	PFDD2	PFDD1	PFDD0

The PFD (programmable frequency divider) is implemented in the HT82A832R. It is composed of two portions: a prescaler and a general counter.

The prescaler is controlled by the register bits, PRES0 and PRES1. The 4-stage prescaler is divided by 16. The general counter is programmed by an 8-bit register PFDD.

The PFDD is inhibited to write while the PFD is disabled. To modify the PFDD contents, the PFD must be enabled. When the generator is disabled, the PFDD is cleared by hardware.

PFD prescaler selection:

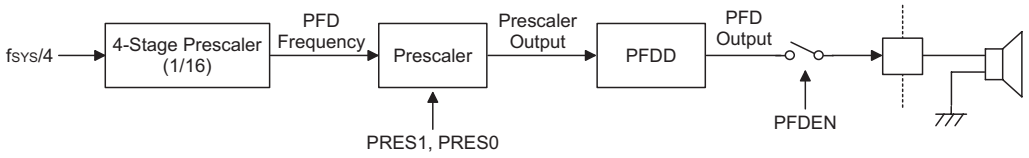
PRES1	PRES0	Prescaler Output
0	0	PFD frequency source ÷ 1
0	1	PFD frequency source ÷ 2
1	0	PFD frequency source ÷ 4
1	1	PFD frequency source ÷ 8

The bit PFD_IO is used to determine whether PC0 is a general purpose I/O port or a PFD output.

Label	Functions
PFD_IO=1	″PC0″ is PFD output
PFD_IO=0	″PC0″ is a general purpose IO Port (Default =0)

The SELW bit is used to control the power amplifier input source. The software should set SELW= ″1″ when the power amplifier signal come from MUSIC_IN, otherwise the speaker output will come from USB Audio data.

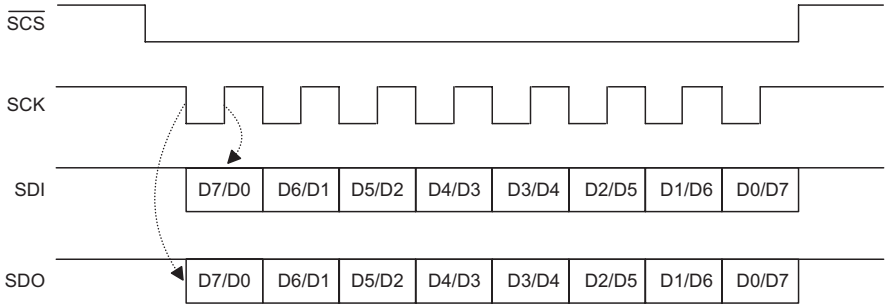
Label	Functions
SELW=1	Power amplifier signal is sourced from MUSIC_IN pin
SELW=0	Power amplifier signal is sourced from USB Audio data (default =0)



Note: PFD Output Frequency = $\frac{\text{Prescaler Output}}{2 \times (N+1)}$, where N = the value of the PFD data

SPI

The serial interface function similar to SPI (Motorola), where four basic signals are included. They are SDI (Serial Data Input), SDO (Serial Data Output), SCK (serial clock) and $\overline{SCS}$ (slave select pin).



SPI Timing

Label	Functions	D7	D6	D5	D4	D3	D2	D1	D0
SBCR	Serial Bus Control Register	CKS	M1	M0	SBEN	MLS	CSEN	WCOL	TRF
Default		0	1	1	0	0	0	0	0
SBDR	Serial Bus Data Register	D7	D6	D5	D4	D3	D2	D1	D0
Default		U	U	U	U	U	U	U	U

Note: ″U″ unchanged

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

- SBCR: Serial bus control register
  - Bit7 (CKS): clock source selection: $f_{SIO}=f_{SYS}/2$, select as 0; $f_{SIO}=f_{SYS}$, select as 1
  - 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 ($\overline{SCK}$=″0″) and wait to write 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 a 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 and related pins are IO ports.

Label	Functions
SBEN=1	PC4~PC7 are SPI function pins (pin $\overline{SCS}$ will go low if CSEN=1).
SBEN=0	PC4~PC7 are general purpose I/O Port pins (Default)

Note: 1. If SBEN="1", the pull-high resistors on PC4~PC7 will be disabled. When this happens the user should add external pull-high resistors to the SPI related pins if necessary (EX: pin $\overline{SCS}$).
2. If CSEN="0", the $\overline{SCS}$ pin will enter a floating state.

- 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
- Bit1 (WCOL): this bit is set to 1 if data is written to SBDR (TXRX buffer) when a data is transferring
  → writing will be ignored if data is written to SBDR (TXRX buffer) when a data is transferring
  WCOL will be set by hardware and clear by software.
- Bit 0 (TRF): data transferred or data received → used to generate interrupt
  Note: data receiving is still working when MCU enters halt mode

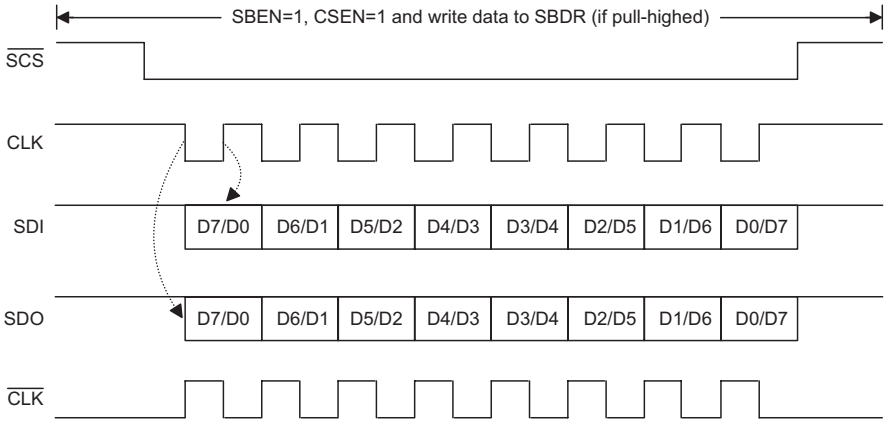
- SBDR: Serial bus data register
  Data written to SBDR → write data to TXRX buffer only
  Data read from SBDR → read from SBDR only
  - Operating Mode description:
    Master transmitter: clock sending and data I/O started by writing SBDR
    Master clock sending started by writing SBDR
    Slave transmitter: data I/O started by clock received
    Slave receiver: data I/O started by clock received

- Clock polarity = rising ($\overline{CLK}$) or falling (CLK): 1 or 0 (software option)
  Operation of Serial Interface:

Label	Functions
Master	• Select CKS and select M1,M0 = 00, 01, 10 • Select CSEN, MLS (same as slave) • Set SBEN • 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) • Check WCOL; WCOL = 1 → clear WCOL and go to step 4; WCOL = 0 → go to step 6 • Check TRF or waiting for SBI (serial bus interrupt) • Read data from SBDR • Clear TRF • Go to step 4
Slave	• CKS don't care and select M1, M0 = 11 • Select CSEN, MLS (same as master) • Set SBEN • Writing data to SBDR → data is store 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) • Check WCOL; WCOL = 1 → clear WCOL, go to step 4; WCOL = 0 → go to step 6 • Check TRF or waiting for SBI (serial bus interrupt) • Read data from SBDR • Clear TRF • Go to step 4

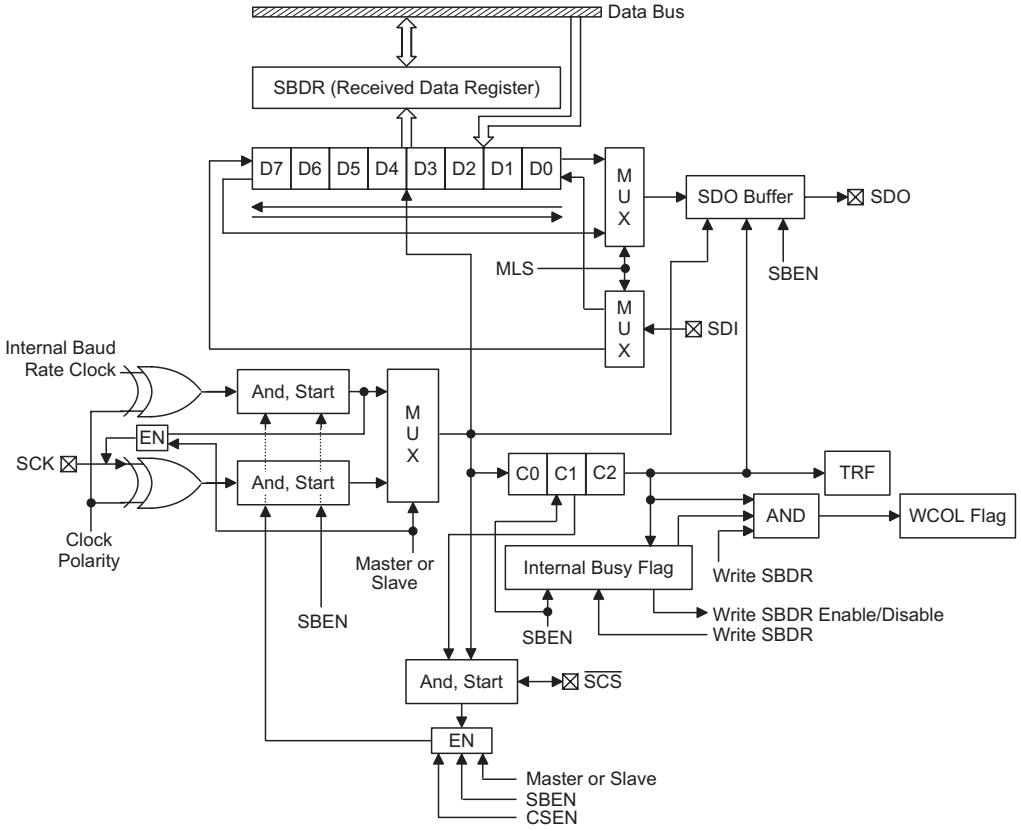
- WCOL: master/slave mode, set if writing to SBDR when data is transferring (transmitting or receiving) and this writing will be ignored. WCOL function can be enabled/disabled by software option (SIO_WCOL bit of MODE_CTRL register). WCOL is set by SIO and cleared by users.
  Data transmission and reception are still workable when MCU enters halt mode.
  CPOL is used to select the clock polarity of CLK. It is a software option (SIO_CPOL bit of MODE_CTRL register).

- 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 setting and slave data transferring should be disabled(enabled) before(after) $\overline{SCS}$ signal received. CSEN = 0, $\overline{SCS}$ signal is not needed, $\overline{SCS}$ pin (master and slave) should be floating.
- CSEN: CSEN software option (SIO_CSEN bit of MODE_CTRL register) is used to enable/disable software CSEN function. If CSEN software option is disable, software CSEN always disabled. If CSEN software 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 software option), slave CLK = floating
- SBEN = 0 → serial bus disable; $\overline{SCS}$ = SDI = SDO = CLK = floating
- TRF is set by SIO and cleared by users. When data transferring (transmission and reception) is complete, TRF is set to generate SBI (serial bus interrupt).



SIO Timing

Label	Functions	D7	D6	D5	D4	D3	D2	D1	D0
SBCR	Serial Bus Control Register	CKS	M1	M0	SBEN	MLS	CSEN	WCOL	TRF
Default		0	1	1	0	0	0	0	0
SBDR	Serial Bus Data Register	D7	D6	D5	D4	D3	D2	D1	D0
Default		U	U	U	U	U	U	U	U



Block Diagram of SIO

Label	Functions
WCOL	set by SIO cleared by users
CESN	Enable or disable chip selection function pin Master mode: 1/0=with/without $\overline{SCS}$ output control Slave mode: 1/0= with/without $\overline{SCS}$ input control
SBEN	Enable or disable serial bus (0= initialize all status flags) When SBEN=0, all status flags should be initialized 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

If clock polarity set to rising edge (SIO_CPOL=1), serial clock timing follow $\overline{CLK}$, otherwise (SIO_CPOL=0) CLK is the serial clock timing.

Mode Control

The MODE_CTRL register is used to control DAC and ADC operation mode and SPI function.

Bit No.	Label	Functions
0	DA_L_ENB	DAC enable/disable control (left channel) 1= DAC Left Channel disable 0= DAC Left Channel enable (default)
1	DA_R_ENB	DAC enable/disable control (right channel) 1= DAC Right Channel disable 0= DAC Right Channel enable (default)
2	AD_ENB	ADC enable/disable control 1= ADC power down 0= ADC power on (default)
3	PLAY_MODE	DAC play mode control 1= 8kHz/16-bit 0= 48kHz/16-bit (default)
4	SIO_CPOL	There are three bits used to control the mode of SPI operation. 1= clock polarity rising edge 0= clock polarity falling edge (default)
5	SIO_WCOL	1= WCOL bit of SBCR register enable 0= WCOL bit of SBCR register disable (default)
6	SIO_CSEN	1= CSEN bit of SBCR register enable 0= CSEN bit of SBCR register disable (Default)
7	—	Undefined bit, read as ″0″

MODE_CTRL (34H) Register

SPI Usage Example

```
SPI_Test:
          clr   UCC.@UCC_SYSCLK    ;12MHz SYSCLK
          set   SIO_CSEN           ;SPI Chip Select Function Enable
          clr   SIO_CPOL           ;falling edge change data
          ;Master Mode, SCLK=fSIO
          clr   M1
          clr   M0
          ;--------------
          clr   CKS                ;fSIO=fsys/2
          clr   TRF                ;clear TRF flag
          clr   TRF_INT            ;clear Interrupt SPI flag
          set   MLS                ;MSB shift first
          set   CSEN               ;Chip Select Enable
          set   SBEN               ;SPI Enable, SCS will go low
if  POLLING_MODE
          clr   ESII               ;SPI Interrupt Disable
          ;WRITE INTO "WRITE ENABLE" INSTRUCTION
          MOV   A,OP_WREN
          MOV   SBDR,A
$0:
          snz   TRF
          jmp   $0
          clr   TRF
else
          set   ESII               ;SPI Interrupt Enable
;WRITE INTO "WRITE ENABLE" INSTRUCTION
          MOV   A,OP_WREN
          MOV   SBDR,A
$0:
          snz   TRF_INT             ;set at SPI Interrupt
          jmp   $0
          clr   TRF_INT
endif
```

Play/Record Data

The play/record interrupt will be activated when play/record data is valid on PLAY_DATA/ RECORD_DATA registers. The PLAY_DATA/RECORD_DATA registers will latch data until next interrupt happen. The PLAY_DATA is unsigned value (0~FFFFH). RECORD_DATA is 2′s complement value (8000H~7FFFH).

The update rate of RECORD_DATA is 8KHz. The update rate of PLAY_DATA is 48KHz (PLAY_MODE=0) or 8KHz (PLAY_MODE=1). All these registers (3AH~3FH) are read only.

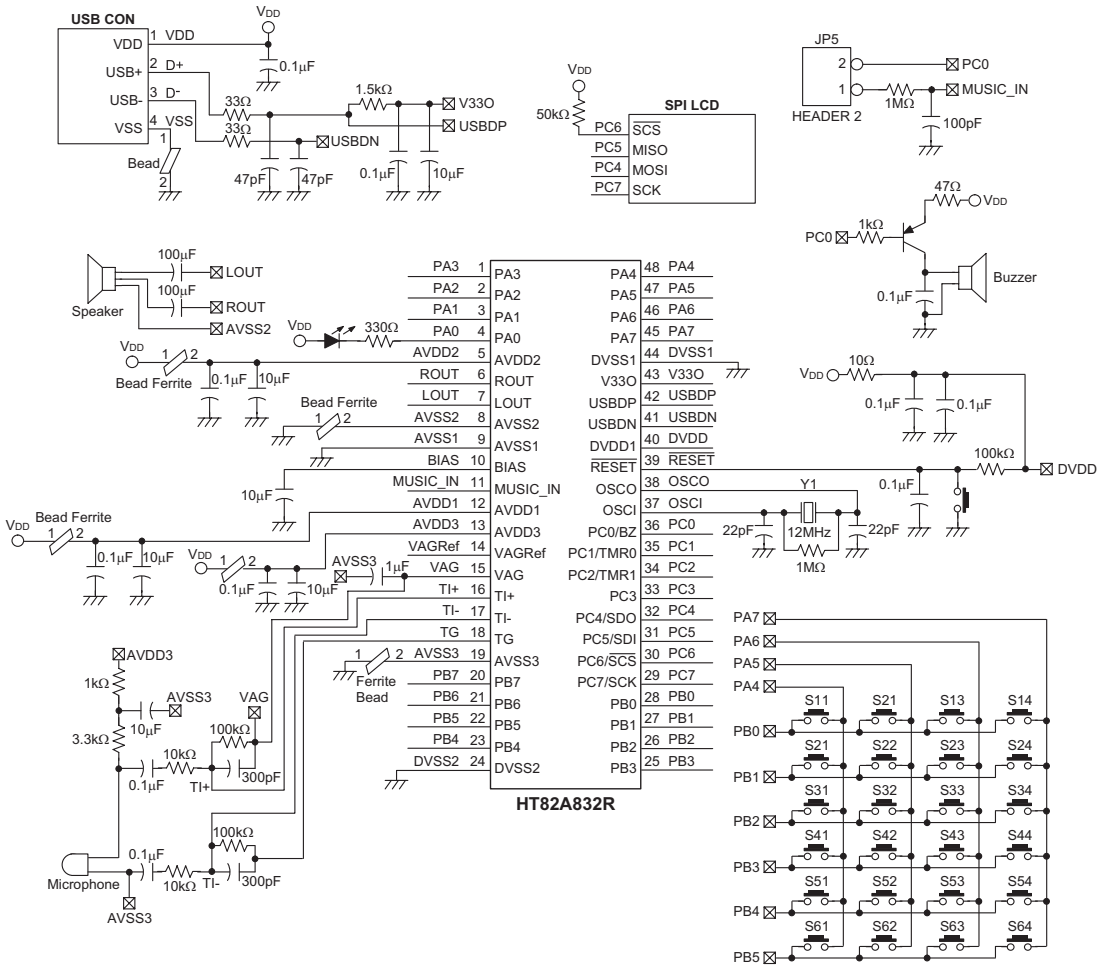
Address	Label	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
3AH	PLAY_DATAL_L	PL_D7	PL_D6	PL_D5	PL_D4	PL_D3	PL_D2	PL_D1	PL_D0
3BH	PLAY_DATAL_H	PL_D15	PL_D14	PL_D13	PL_D12	PL_D11	PL_D10	PL_D9	PL_D8
3CH	PLAY_DATAR_L	PR_D7	PR_D6	PR_D5	PR_D4	PR_D3	PR_D2	PR_D1	PR_D0
3DH	PLAY_DATAR_H	PR_D15	PR_D14	PR_D13	PR_D12	PR_D11	PR_D10	PR_D9	PR_D8
3EH	RECORD_DATA_L	R_D7	R_D6	R_D5	R_D4	R_D3	R_D2	R_D1	R_D0
3FH	RECORD_DATA_H	R_D15	R_D14	R_D13	R_D12	R_D11	R_D10	R_D9	R_D8

Configuration Options

The following table shows all of the configuration options 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	PB0~PB7 pull-high resistor enabled or disabled (by bit)
8	PC0~PC7 pull-high resistor enabled or disabled (by nibble)
9	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]	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.

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

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 (current page) to TBLH and data memory

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.

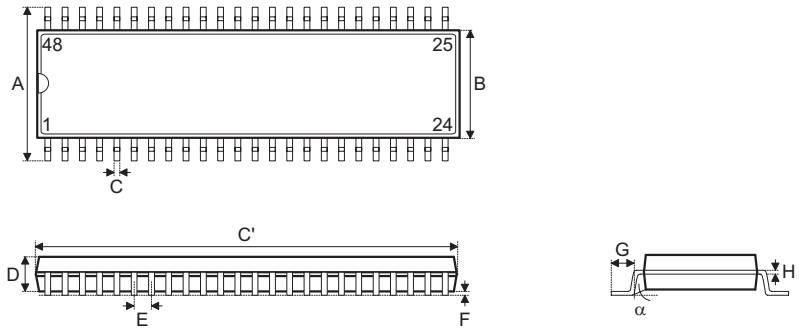
Operation ACC ← ACC ″XOR″ x

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

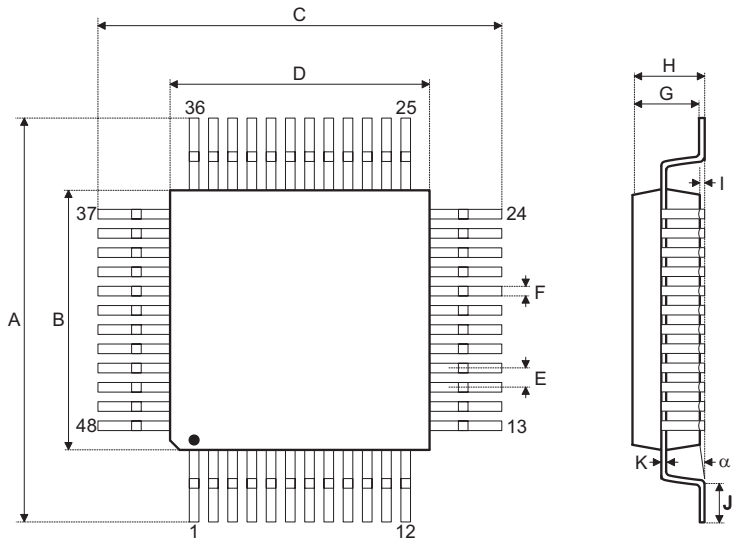
Package Information

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°

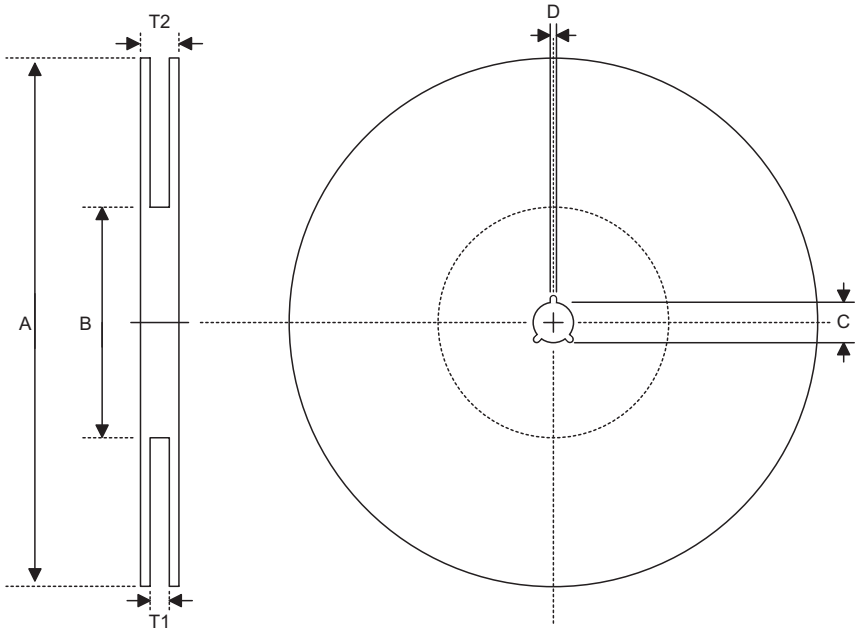
48-pin LQFP (7×7) Outline Dimensions



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	—	0.5	—
F	—	0.2	—
G	1.35	—	1.45
H	—	—	1.6
I	—	0.1	—
J	0.45	—	0.75
K	0.1	—	0.2
α	0°	—	7°

Product Tape and Reel Specifications

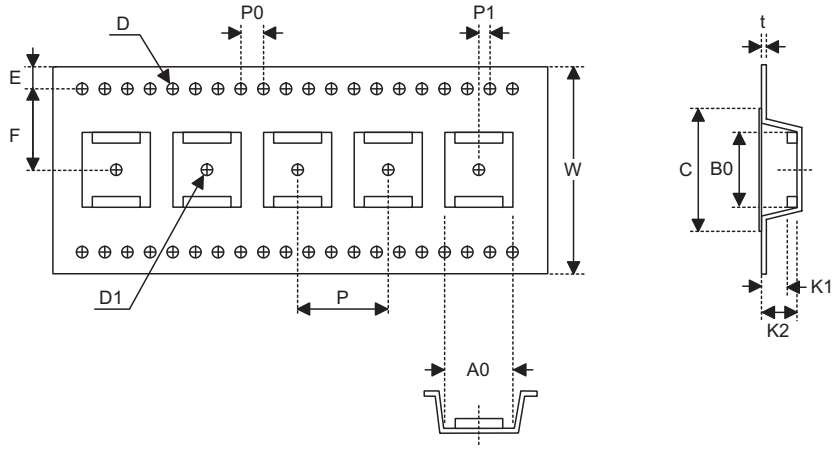
Reel Dimensions



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: 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