HOLTEK

HT82A850R Audio MCU

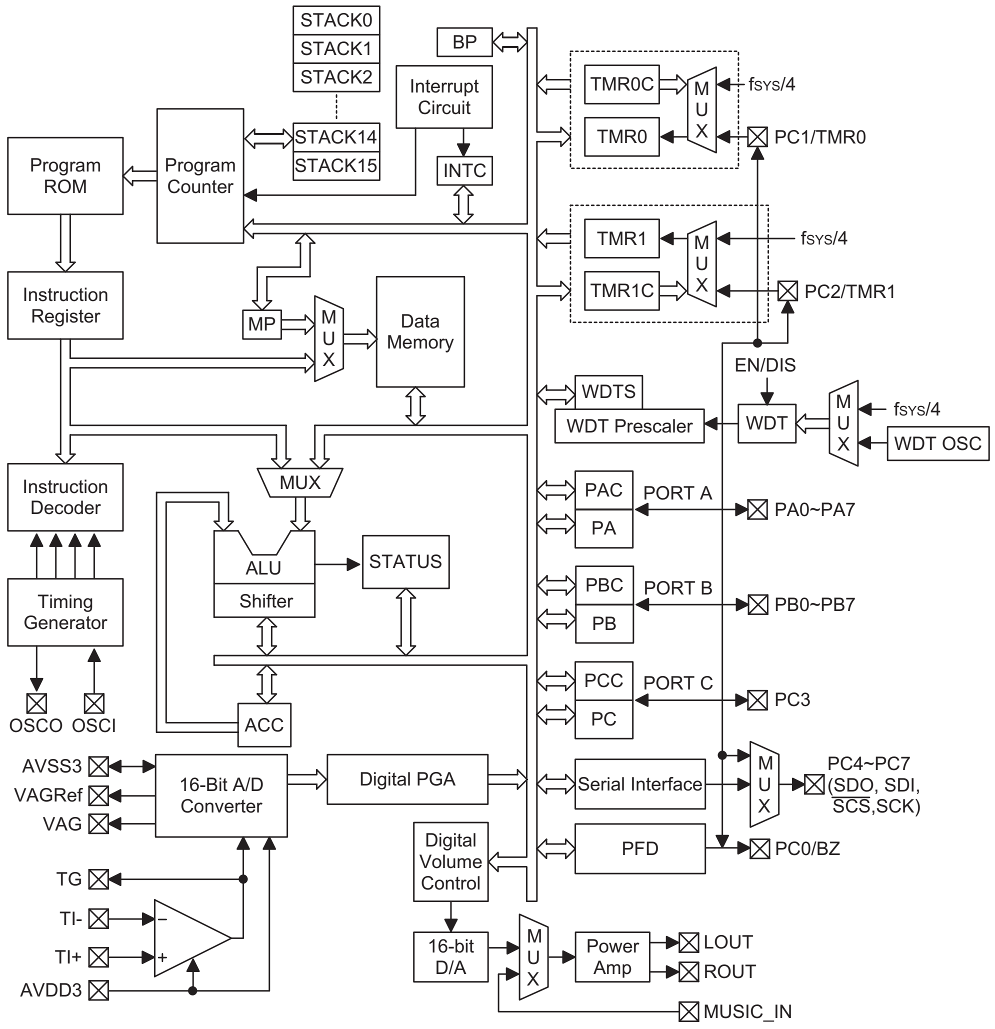
Features

- Operating voltage: f_{SYS}= 6MHz/12MHz: 3.3V~5.5V
- Low voltage reset function (3.0V±0.3V)
- Embedded high performance 16 bit PCM DAC
- Built-in digital Programmable Gain Amplifier - PGA
- High-performance 48kHz/8kHz sampling rate for audio software playback
- 8kHz audio recording sampling rate
- Embedded class AB power amplifier for speaker driving
- Audio playback digital volume control
- 4096×15 program memory
- 384×8 data memory RAM (Bank0,1)
- Programmable frequency divider - PFD
- Integrated SPI hardware circuit
- Play/record interrupt
- HALT function and wake-up feature reduce power consumption
- 24 bidirectional I/O lines (max.)
- Two 16-bit programmable timer/event counter and overflow interrupts
- Watchdog Timer
- 16-level subroutine nesting
- Bit manipulation instruction
- 15-bit table read instruction
- 63 powerful instructions
- All instructions executed within one or two machine cycles
- 48-pin LQFP package

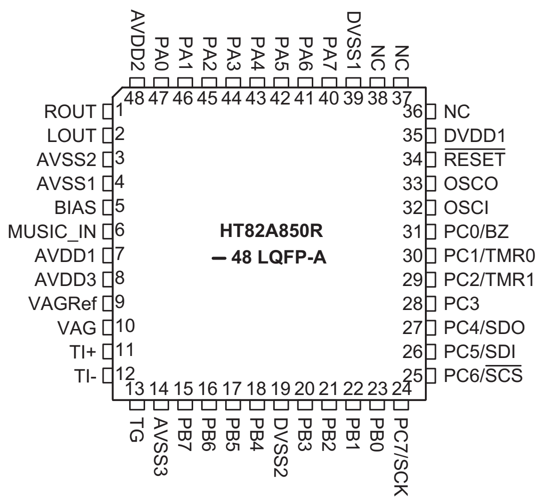
General Description

The HT82A850R is an 8-bit high performance RISC-like microcontroller designed for USB Phone product applications. The HT82A850R integrates a 16-bit PCM ADC and an 8-bit MCU into a single device. The DAC in the HT82A850R operates at a sampling rate of 48/8kHz and the 16-bit PCM ADC operates at a frequency of 8kHz for the Microphone input. The DAC in the HT82A850R also has a digital programmable gain amplifier, whose gain ranges from −32dB to +6dB. For the ADC input, the digital gain range is from 0dB to 19.5dB.

Block Diagram



Pin Assignment



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 a Schmitt trigger input. Pull-high resistor can be connected to the pins via configuration options - nibble 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 between this pin and ground for half-supply stability
MUSIC_IN	I	Power amplifier signal source if register bit SELW ="1". The analog signal input will be amplified by the power amp then output to pins 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 be left open or connected via a bypass capacitor (Ex:100pF) to ground
VAG	O	ADC analog ground voltage - should be connected via a bypass capacitor (Ex:1μF) to ground
TI+	I	OP AMP non-inverting input
TI-	I	OP AMP inverting input
TG	O	OP AMP gain setting output
AVSS3	—	ADC negative power supply, ground
PB0~PB7	I/O	Bidirectional 8-bit input/output port. Software instructions determine if the pin is a CMOS output or a Schmitt trigger input. Pull-high resistor can be connected to the pins via configuration options - nibble 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.

Pin Name	I/O	Description
PC6/SCS	I/O	Can be software optioned as a bidirectional input/output or serial interface slave select signal.
PC5/SDI	I/O or O	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 a Schmitt trigger input. Pull-high resistor can be connected to the pins via configuration options.
PC2/TMR1, PC1/TMR0	I/O	Software instructions determine if the pin is a CMOS output or a Schmitt trigger input. Pull-high resistor can be connected to the pins via configuration options.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
$\overline{\text{RESET}}$	I	Schmitt trigger reset input, active low
NC	—	No connection
DVDD1	—	Positive digital power supply
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
I_{OL} Total	150mA
Total Power Dissipation	500mW
Storage Temperature	–50°C to 125°C
Operating Temperature	–40°C to 85°C
I_{OH} Total	–100mA

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	3.3V	—	3.3	3.6	5.5	V
I_{DD1}	Operating Current	3.3V	No load, f_{SYS}=12MHz ADC on, DAC on	—	9	—	mA
I_{DD2}	Operating Current	3.3V	No load, f_{SYS}=12MHz ADC off, DAC off	—	5	—	mA
I_{STB}	Standby Current	3.3V	No load, system HALT	—	155	—	μA
V_{IL1}	Input Low Voltage for I/O Ports	3.3V	—	0	—	$0.3V_{DD}$	V
V_{IH1}	Input High Voltage for I/O Ports	3.3V	—	$0.7V_{DD}$	—	V_{DD}	V
V_{IL2}	Input Low Voltage ($\overline{\text{RESET}}$)	3.3V	—	0	—	$0.4V_{DD}$	V
V_{IH2}	Input High Voltage ($\overline{\text{RESET}}$)	3.3V	—	$0.9V_{DD}$	—	V_{DD}	V
I_{OL}	I/O Port Sink Current	3.3V	$V_{OL}=0.1V_{DD}$	—	3	—	mA
I_{OH}	I/O Port Source Current	3.3V	$V_{OH}=0.7V_{DD}$	—	–2	—	mA
R_{PH}	Pull-high Resistance	3.3V	—	110	80	40	kΩ
V_{LVR}	Low Voltage Reset	3.3V	—	2.7	3	3.3	V

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
DAC+Power Amp Test condition: Measurement bandwidth 20Hz to 20kHz, f_S= 48kHz. Line output series capacitor with 220μF.							
THD+N	THD+N[Note]	5V	4Ω load	—	−30	—	dB
			8Ω load	—	−35	—	
SNR_{DA}	Signal to Noise Ratio[Note]	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	3.3V	—	—	77	—	dB
VAG	Reference Voltage	3.3V	—	—	1.12	—	V
V_{PEAK}	Peak Single Frequency Tone Amplitude without Clipping	3.3V	—	—	1.575	—	V_{PK}

Note: 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)	3.3V	—	0.4	—	12	MHz
t_{WDTOSC}	Watchdog Oscillator Period	3.3V	—	—	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 microcontroller system clock 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 memory are executed. Its contents specify the full program memory range.

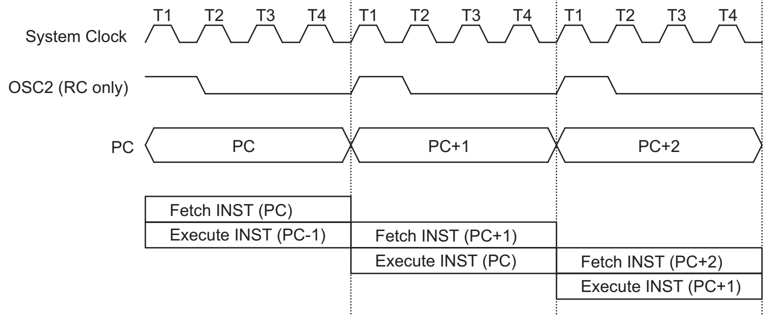
After accessing a program memory word to fetch an instruction code, the contents of the program counter are incremented by one. The program counter then points to the memory word containing the next instruction code.

When executing a jump instruction, a conditional skip execution, loading to the PCL register, performing a subroutine call or returning from a subroutine, an initial reset, an 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 the next instruction is executed.

The lower byte of the program counter, PCL, is a readable and writeable register. Moving data into the PCL performs a short jump. The destination will be within the current program memory page.

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



Execution Flow

Mode	Program Counter											
	*11	*10	*9	*8	*7	*6	*5	*4	*3	*2	*1	*0
Initial Reset	0	0	0	0	0	0	0	0	0	0	0	0
Reserved	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

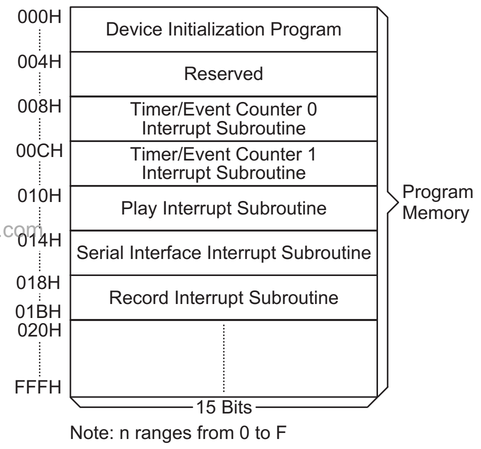
Note: *11~*0: Program counter bits
#11~#0: Instruction code bits
S11~S0: Stack register bits
@7~@0: PCL bits

Program Memory

The program memory is used to store the executable program instructions. 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 location is reserved for program initialisation. After a device reset, the program always begins execution at location 000H.
- Location 008H
  This location 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 will jump to this location and begin execution.
- Location 00CH
  This location is reserved for the Timer/Event Counter 1 interrupt service program. If a timer interrupt results from a Timer/Event Counter 1 overflow, and the interrupt is enabled and the stack is not full, the program will jump to this location and begin execution.
- Location 010H
  This location is reserved for the play interrupt service program. If the play data is valid, and the interrupt is enabled and the stack is not full, the program will jump to this location and begin execution.


000H
004H
008H
00CH
010H
014H
018H
01BH
020H
FFFH
Device Initialization Program
Reserved
Timer/Event Counter 0 Interrupt Subroutine
Timer/Event Counter 1 Interrupt Subroutine
Play Interrupt Subroutine
Serial Interface Interrupt Subroutine
Record Interrupt Subroutine
Program Memory
15 Bits
Note: n ranges from 0 to F

Program Memory

- Location 014H
  This location is reserved for when 8 bits of data have been received or transmitted successful from the serial interface. If the related interrupts are enabled, and the stack is not full, the program will jump to this location and begin execution.
- Location 018H
  This location 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 will jump to this location and begin execution.
- Table location
  Any location in the program memory can be used as a look-up table. There are three method to read the program memory data. The first method uses the TABRDC instruction to transfer the contents of the current page lower-order byte to the specified data memory, and the current page higher-order byte to the TBLH register. The second method uses the TABRDL instruction to transfer the contents of the last page lower-order byte to the specified data memory, and the last page higher-order byte to the TBLH register. The third method uses the TABRDC instruction together with the TBLP and TBHP pointers to transfer the contents of the lower order byte at the specified address to the specified data memory, and the higher order byte at the specified address to the TBLH register. Before accessing the table data, the address to be read must be placed in the table pointer registers, TBLP and TBHP. Note that if the configuration option TBHP is disabled, then the value in TBHP has no effect. 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 word is read as ″0″.The Table Higher-order byte register, TBLH, is read only. The TBLH register 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 TBLH in the main routine are likely to be changed by the table read instruction used in the ISR. In such cases errors can occur. Therefore, using the table read instruction in the main routine and the ISR simultaneously should be avoided. However, if the table read instruction has to be used in both the main routine and the ISR, the interrupt should be disabled

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
@7~@0: Table pointer bits

P11~P8: Current program counter bits when TBHP is disabled
TBHP register bit3~bit0 when TBHP is enabled

prior to the table read instruction. It should not be re-enabled until 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 upon requirements.

Stack Register – STACK

This is a special part of the memory which is used to save the contents of the program counter only. The stack is organised 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, which 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 SP will point to the top of the stack.

If the stack is full and a non-masked interrupt takes place, the interrupt request flag will be recorded but the acknowledge signal will be inhibited. When the stack pointer is decremented, using RET or RETI, the interrupt will be serviced. This feature prevents a 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, a stack overflow will occur and the first entry will be lost. Only the most recent 16 return addresses are stored.

Data Memory

The data memory is divided into two functional groups. These are the special function registers and the general purpose data memory in Bank0 and Bank1: 384×8 bits. Most are read/write, but some are read only. The special function registers are overlapped in all banks.

Any unused space before 40H is reserved for future expanded usage and if read will return a value of ″00H″. The general purpose data memory, addressed from 40H to FFH, is used for data and control information under instruction commands.

All data memory areas can handle arithmetic, logical, 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 the memory pointer registers, MP0 or MP1.

00H	Indirect Addressing Register 0	Special Purpose Data Memory
01H	MP0	
02H	Indirect Addressing Register 1	
03H	MP1	
04H	BP	
05H	ACC	
06H	PCL	
07H	TBLP	
08H	TBLH	
09H	WDTS	
0AH	STATUS	
0BH	INTC0	
0CH	TMR0H	
0DH	TMR0L	
0EH	TMR0C	
0FH	TMR1H	
10H	TMR1L	
11H	TMR1C	
12H	PA	
13H	PAC	
14H	PB	
15H	PBC	
16H	PC	
17H	PCC	
18H	▨	
19H	▨	
1AH	▨	
1BH	▨	
1CH	USVC	
1DH	▨	
1EH	INTC1	
1FH	TBHP	
20H	▨	
21H	▨	
22H	UCC	
23H ⋮ 27H	▨	
28H ⋮ 2CH	▨	
2DH	DAC_LIMIT_L	
2EH	DAC_LIMIT_H	
2FH	DAC_WR	
30H	PGA_CTRL	
31H	PFDC	
32H	PFDD	
33H	▨	
34H	MODE_CTRL	
35H	SBCR	
36H	SBDR	
37H ⋮ 3DH	▨	
3EH	RECORD_DATA_L	
3FH	RECORD_DATA_H	
40H ⋮ FFH	General Purpose Data Memory (192 Bytes) Bank0/Bank1	

▨ : Unused

Read as "00"

RAM Mapping

Indirect Addressing Register

Locations 00H and 02H are the indirect addressing registers, however they are not physically implemented. Any read/write operation to [00H] or [02H] will access the data memory pointed to by MP0 and MP1. Reading location 00H or 02H indirectly will return a result of 00H. Writing indirectly results in no operation.

Data transfer between two indirect addressing registers is not supported. The memory pointer registers, MP0 and MP1, are 8-bit registers which are used to access the Data Memory in combination with indirect addressing registers.

Bank Pointer

The bank pointer is used to select the required Data Memory bank. If Data Memory bank 0 is to be selected, then a ″0″ should be loaded into the BP register. Data Memory 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 contains the zero flag (Z), carry flag (C), auxiliary carry flag (AC), overflow flag (OV), power down flag (PDF), and watchdog time-out flag (TO). It also records the status information and controls the operation sequence.

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

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

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

In addition, upon entering the interrupt sequence or executing a subroutine call, the status register will not be automatically pushed onto the stack. If the contents of the status are important and if the subroutine can corrupt the status register, precautions must be taken to save it properly.

Interrupt

The device provides an internal timer/event counter interrupt, play/record data valid interrupt and a serial interface interrupt. The Interrupt Control Register0, INTC0, and the 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, other interrupts are all blocked, as the EMI bit is cleared automatically, preventing further interrupt nesting. Other interrupt

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

requests may take place during this interval, but only the interrupt request flag will be recorded. If a certain interrupt requires servicing within the service routine, the EMI bit and the corresponding bit of INTC0 or of INTC1 may be set in order to allow interrupt nesting. Once 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 should be prevented from becoming full.

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

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

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

The play interrupt is initialised by setting the play interrupt request flag, bit 4 of INTC1, caused by a valid play interrupt. When the interrupt is enabled, the stack is not full and 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 the MODE_CTRL register, is set to ″1″, the play interrupt frequency will change to 8kHz, otherwise the interrupt frequency is 48 kHz. The firmware will write 16-bit unsigned data to the DAC when a play interrupt occurs.

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 HT82A850R 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 bit is set, a subroutine call to location 14H occurs.

The record interrupt is initialised 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 the ADC is powered down, AD_ENB =1, the record interrupt will be disabled.

Bit No.	Label	Function
0	EMI	Controls the master (global) interrupt (1=enabled; 0=disabled)
1, 4, 7	—	Unused bit, read as ″0″
2	ET0I	Controls the Timer/Event Counter 0 interrupt (1=enabled; 0=disabled)
3	ET1I	Controls the Timer/Event Counter 1 interrupt (1=enabled; 0=disabled)
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)

INTC 0 (0BH) Register

Bit No.	Label	Function
0	EPLAYI	Play interrupt (1= enabled; 0= disabled)
1	ESII	Control Serial interface interrupt (1= enabled; 0= disabled)
2	ERECI	Record interrupt (1= enabled; 0= disabled)
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)

INTC 1 (1EH) Register

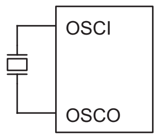
During the execution of an interrupt subroutine, other interrupt acknowledge signals are held until the ″RETI″ instruction is executed or the EMI bit and the related interrupt control bit are set to 1 (if the stack is not full). To return from the interrupt subroutine, a ″RET″ or ″RETI″ instruction should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



Crystal Oscillator

System Oscillator

An internal oscillator circuit is integrated within the microcontroller to implement the system clock. When the device enters the Power-down mode the system oscillator will stop running and external signals will be ignored to conserve power.

A crystal across OSCI and OSCO is required to provide the feedback and phase shift required for the oscillator. No other external components are required. Instead of a crystal, a resonator can also be connected between OSCI and OSCO to obtain the correct frequency reference, however two external capacitors between the OSCI, OSCO pins and ground are required.

A WDT oscillator is also contained within the device. This is a free running fully integrated RC oscillator requiring no external components. Even if the system enters the power down mode, where the system clock is stopped, the WDT oscillator 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 its own dedicated internal RC oscillator (WDT oscillator) or by using the instruction clock, which is the system clock/4. The WDT timer is designed to prevent a software malfunction or sequence from jumping to an unknown location with unpredictable results. The WDT can be disabled using a configuration option. Note that if the WDT is disabled, all executions of instructions related to the WDT will result in no operation.

When the WDT clock source is selected, it will be first divided by 256 (8-stage) to obtain a nominal time-out period. By using this WDT prescaler, longer time-out periods can be implemented. This is achieved by writing data to the the WS2, WS1, WS0 bits.

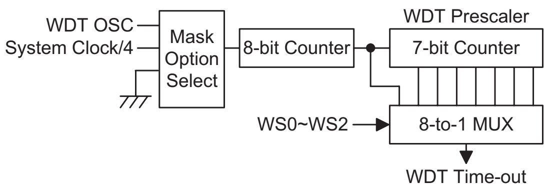
The WDT OSC period has a typical value of 65μs. This time-out period may vary with temperature, VDD and process variations. The WDT OSC keeps running in any operational mode.

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

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

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

Power Down Operation

The Power-down mode is entered by the execution of a ″HALT″ instruction and results in the following:

- The system oscillator will be turned off but the WDT oscillator keeps running if the internal WDT oscillator is selected.
- The contents of the on-chip data memory and registers remain unchanged.
- The WDT and WDT prescaler will be cleared and will start counting again if the WDT clock is sourced from the internal WDT oscillator.
- All of the I/O ports remain in their original condition.
- The PDF flag is set and the TO flag is cleared.

The system can leave the Power-down 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 initialisation and the WDT overflow performs a ″warm reset″. After the TO and PDF flags are examined, the cause for the device reset can be determined. The PDF flag is cleared by a system power-up or by executing the ″CLR WDT″ instruction and is set when executing the ″HALT″ instruction. The TO flag is set if the WDT time-out occurs, and causes a wake-up that only resets the program counter and SP; the others remain in their original status.

A port A wake-up and interrupt methods can be considered as a continuation of normal execution. Each pin in port A can be independently selected to wake-up the device using configuration options. After awakening from an I/O port stimulus, the program will resume execution at the next instruction. If the device is awakened 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 Power-down 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 periods) to resume normal operation, i.e., a dummy period is inserted. 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 minimise power consumption, all the I/O pins should be carefully managed before entering the Power-down mode.

The ADC, DAC and PA will all be powered down when in the HALT mode.

Reset

There are four ways in which a reset can occur:

- $\overline{\text{RES}}$ reset during normal operation
- $\overline{\text{RES}}$ reset when in the Power-down mode
- WDT time-out reset during normal operation
- USB reset

The WDT time-out when in the Power-down mode is different from other device 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 their ″initial condition″ when the reset conditions are met. By examining the PDF and TO flags, the program can distinguish between different ″device 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

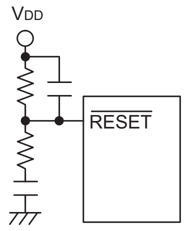
Note: ″u″ stands for ″unchanged″

To guarantee that the system oscillator is started and stabilised, 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 Power-down mode.

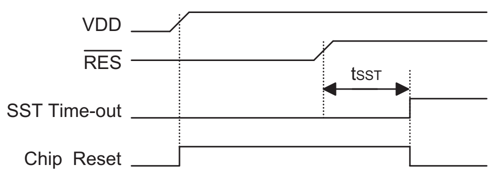
When a system reset occurs, the SST delay is added during the reset period. Any wake-up from the Power-down mode will enable the SST delay.

The status of the device after a reset is shown below.

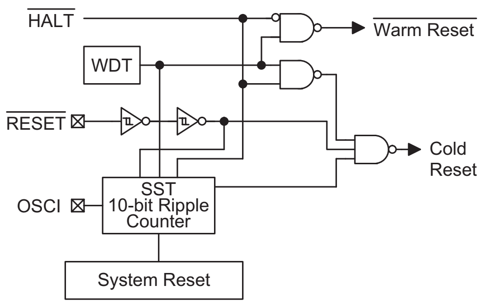
Program Counter	000H
Interrupt	Disabled
WDT	Cleared. After a master reset, WDT begins counting
Timer/Event Counter	Off
Input/output Ports	Input mode
Stack Pointer	Points to the top of the stack



Reset Circuit



Reset Timing Chart



Reset Configuration

The registers status are summarised in the following table.

Register	Reset (Power On)	WDT Time-out (Normal Operation)	$\overline{\text{RES}}$ Reset (Normal Operation)	$\overline{\text{RES}}$ Reset (HALT)	WDT Time-Out (HALT)*
MP0	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
MP1	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
BP	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
Program Counter	000H	000H	000H	000H	000H
TBLP	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLH	-xxx xxxx	-uuu uuuu	-uuu uuuu	-uuu uuuu	-uuu uuuu
WDTS	0000 0111	0000 0111	0000 0111	0000 0111	uuuu uuuu
STATUS	--00 xxxx	--1u uuuu	--uu uuuu	--01 uuuu	--11 uuuu
INTC0	-000 0000	-000 0000	-000 0000	-000 0000	-uuu uuuu
TMR0H	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TMR0L	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TMR0C	00-0 1000	00-0 1000	00-0 1000	00-0 1000	uu-u uuuu
TMR1H	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TMR1L	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TMR1C	00-0 1---	00-0 1---	00-0 1---	00-0 1---	uu-u u---
PA	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PB	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PBC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PCC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
USVC	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
INTC1	-000 0000	-000 0000	-000 0000	-000 0000	-uuu uuuu
TBHP	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
UCC	0000 0000	uuuu uuuu	0000 0000	0000 0000	uuuu uuuu
DAC_LIMIT_L	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
DAC_LIMIT_H	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
DAC_WR	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PGA_CTRL	0000 0000	0000 0000	0000 0000	0000 0000	0000 0000
PFDC	0000 0000	0000 0000	0000 0000	0000 0000	0000 0000
PFDD	0000 0000	0000 0000	0000 0000	0000 0000	0000 0000
MODE_CTRL	0000 0000	0000 0000	0000 0000	0000 0000	0000 0uuu
SBCR	0110 0000	0110 0000	0110 0000	0110 0000	uuuu uuuu
SBDR	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
RECOED_ DATA_L	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
RECOED_ DATA_H	0000 0000	0000 0000	0000 0000	0000 0000	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 are implemented in the microcontroller. Each timer contains a 16-bit programmable count-up counter whose clock may be sourced from an external or internal clock source. The internal clock source comes from $f_{SYS}/4$. The external clock input allows external events to be counted, time intervals or pulse widths to be measured, or to generate an accurate time base. There are three registers related to Timer/Event Counter 0, TMR0H, TMR0L and TMR0C, and another three related to Timer/Event Counter 1, TMR1H, TMR1L and TMR1C. When writing data to the TMR0L and TMR1L registers, note that the data will only be written into a lower-order byte buffer. The data will not be actually written into the TMR0L and TMR1L registers until a write operation to the TMR0H and TMR1H registers is implemented. Reading the TMR0L and TMR1L registers will read the contents of the lower-order byte buffer. The TMR0C and TMR1C registers are the Timer/Event Counter control registers, which define the operating mode, the count enable or disable and the 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 sourced from the external TMR0 or TMR1 pin. The timer mode functions as a normal timer with the clock source coming from the internal clock. Finally, the pulse width measurement mode can be used to count the high level or low level duration of an external signal on pins TMR0 or TMR1, whose counting is based on the internal clock source.

In the event count or timer mode, the timer/event counter starts counting from 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, or 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 or TMR1 pin 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 or TMR1 pin returns to its original level and resets the TON bit. The measured result remains in the timer/event counter even if the activated transient occurs again. Therefore, only 1-cycle measurement is made. Not until the TON bit is again set can the cycle measurement re-function. In this operational 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.

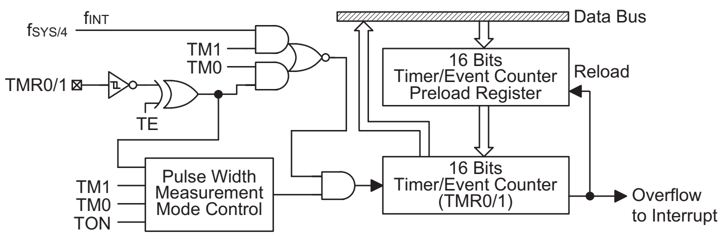
To enable a count operation, the Timer ON bit (TON; bit 4 of TMR0C or TMR1C) should be set to 1. In the pulse width measurement mode, TON is automatically cleared after the measurement cycle is completed. But in the other two modes, the TON bit can only be reset by instructions. A Timer/Event Counter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

If the timer/event counter is turned OFF, writing data to the timer/event counter preload register will also reload the data into the timer/event counter. But if the timer/event counter is turned on, data written to the timer/event counter is kept only in the timer/event counter preload register. The timer/event counter keeps operating until an overflow occurs.

When the timer/event counter is read, the clock is blocked to avoid errors, which may result in a counting error. Blocking of the clock should be taken into account by the programmer.

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



f_INT
f_SYS/4
TM1
TM0
TMR0/1
TE
Data Bus
16 Bits Timer/Event Counter Preload Register
Reload
Pulse Width Measurement Mode Control
TM1
TM0
TON
16 Bits Timer/Event Counter (TMR0/1)
Overflow to Interrupt

Timer/Event Counter 0/1

Input/Output Ports

There are 24 bidirectional input/output lines in the microcontroller, labeled from PA to PC. 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 or 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 or PCC) to control the input/output configuration. With this control register, either a CMOS output or Schmitt trigger input with or without pull-high resistor structures can be reconfigured dynamically under software control. To function as an input, the corresponding latch of the control register must write ″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.

After a device reset, the input/output lines will default to input high levels or a floating state, depending on the pull-high configuration options. Each bit of these input/output latches can be set or cleared using the ″SET [m].i″ and ″CLR [m].i″ (m=12H, 14H or 16H) instructions.

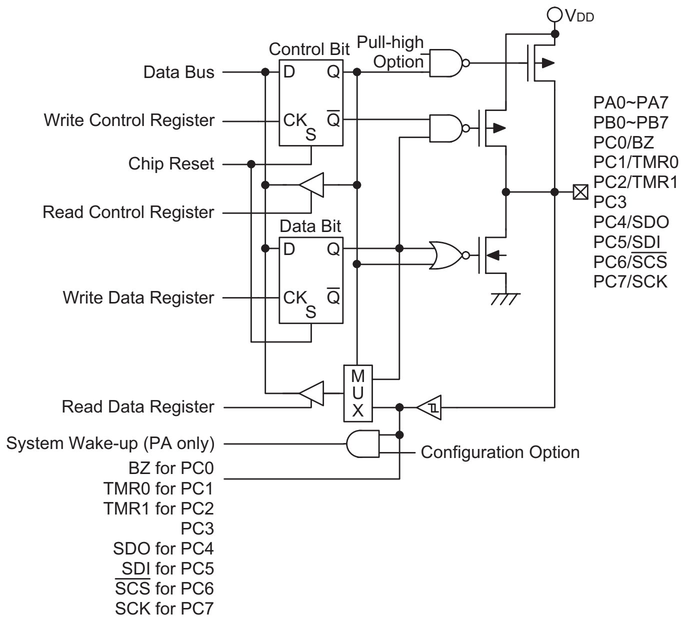
Some instructions first input data and then follow the output operations. For example, ″SET [m].i″, ″CLR [m].i″, ″CPL [m]″, ″CPLA [m]″ read the entire port states into the CPU, execute the defined operations (bit-operation), and then write the results back to the latches or the accumulator.

Each line of port A has the capability of waking-up the device.

Low Voltage Reset – LVR

The LVR function is enabled or disabled using a configuration option. The LVR voltage is 3.0V.

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



Input/Output Ports

The LVR includes the following specifications:

- The low voltage (0.9V~V_{LVR}) has to remain in this condition for a time greater than 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 an ″OR″ function with the external $\overline{\text{RESET}}$ signal to perform a device reset.

System Clock Selection

This register consists of system clock selection (SYSCLK)

Bit No.	Label	R/W	Reset	Functions
0~2	—	—	0	Reserved
5	f_{SYS}16MHz	R/W	0	Defines the MCU system clock - sourced from the external OSC or from the PLL output - 16MHz clock. 0: system clock sourced from OSC 1: system clock sourced from the PLL output - 16MHz
6	SYSCLK	R/W	0	Used to specify the system clock oscillator frequency used by MCU. If a 6MHz crystal oscillator or resonator is used, this bit should be set to ″1″. If a 12MHz crystal oscillator or resonator is used. this bit should be cleared to ″0″.

UCC (22H) Register

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

- Speaker mute control:
  $\overline{\text{MUTE}}$= 0: Mute speaker output.
  $\overline{\text{MUTE}}$= 1: Normal.

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

Digital 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

The DAC_Limit_L and DAC_Limit_H registers are used to define the 16-bit DAC output limit. DAC_Limit_L and DAC_Limit_H have unsigned values. If the 16-bit data from the Host exceeds the range defined by the DAC_Limit_L and DAC_Limit_H, the output digital code to the 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 speaker popping sounds, the power amplifier should be setup to output a value of VDD/2, implemented by sending 8000H to the DAC, during the initial power on state. A falling edge on the DAC_WR_TRIG bit (bit 3 of DAC_WR register), will write the values in the DAC_Limit_L and DAC_Limit_H registers into 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 speaker popping noise:

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

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

Digital PGA

Bit No.	Label	Functions
0~5	PGA0~PGA5	Digital PGA control bits with range 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 following table.
6	ADC_RESET	ADC_RESET="1": PCM ADC at reset condition ADC_RESET="0": PCM ADC during normal operation - default=0 The following conditions will reset the ADC: - MCU Reset - Set ADC_RESET to "1" using the program

7	MUTE_MKB	Microphone mute Control:
		MUTE_MKB =0: Mute microphone input.
		MUTE_MKB =1: Normal.

PGA_CTRL 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

PFDC (31H), PFDD (32H) Register

A Programmable Frequency Divider, PFD, is implemented within the HT82A850R. It is composed of two sections, 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 known as PFDD.

The PFDD is write inhibited 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:

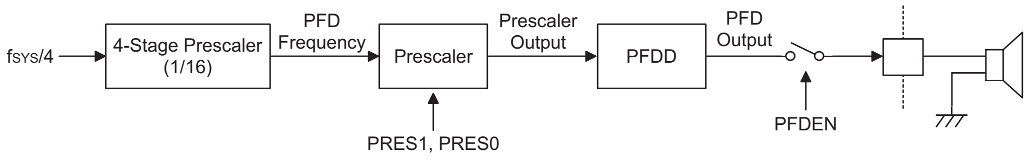
PRES1	PRES0	Prescaler Output
0	0	PFD frequency source ÷ 1
0	1	PFD frequency source ÷ 2
1	0	PFD frequency source ÷ 4
1	1	PFD frequency source ÷ 8

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

PFD_IO="1"	PC0 is PFD output
PFD_IO="0"	PC0 is a general IO pin 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 is sourced from MUSIC_IN, otherwise the speaker output is the USB Audio data.

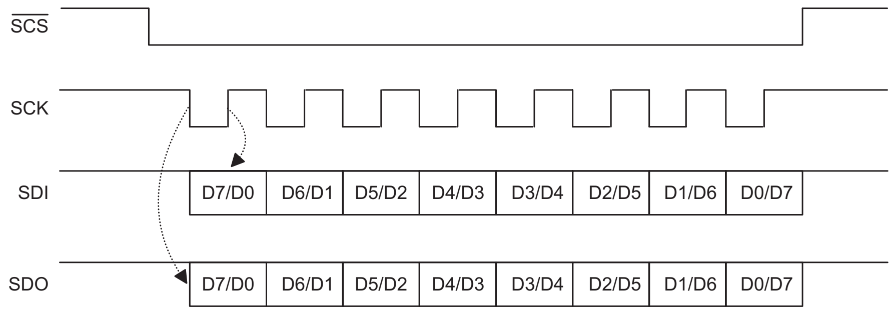
SELW="1"	The power amplifier signal is sourced from the MUSIC_IN pin
SELW="0"	The power amplifier signal is sourced from the USB Audio data (Default=0)



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 is similar to the Motorola SPI, where four basic signals are included. These are the 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 and SBDR, are provided for serial interface 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 $\rightarrow$ 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) $\rightarrow$ start transmission/reception automatically

      Master mode: when data has been transferred $\rightarrow$ set TRF

      Slave mode: when a SCK (and $\overline{SCS}$ dependent on CSEN) is received, data in the 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 external pull-high resistors should be added 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 the data is transferring

    $\rightarrow$ writing will be ignored if data is written to SBDR (TXRX buffer) when the data is transferring
    WCOL will be set by hardware and cleared by software.
  - Bit 0 (TRF): data transferred or data received $\rightarrow$ used to generate an interrupt

    Note: data reception is still operational when the MCU enters the Power-down mode

- SBDR: Serial bus data register

  Data written to SBDR $\rightarrow$ write data to the TXRX buffer only

  Data read from SBDR $\rightarrow$ read from SBDR only
  - Operating Mode description:

    Master transmitter: clock sending and data I/O started by writing to SBDR

    Master clock sending started by writing to SBDR

    Slave transmitter: data I/O started by clock reception

    Slave receiver: data I/O started by clock reception

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

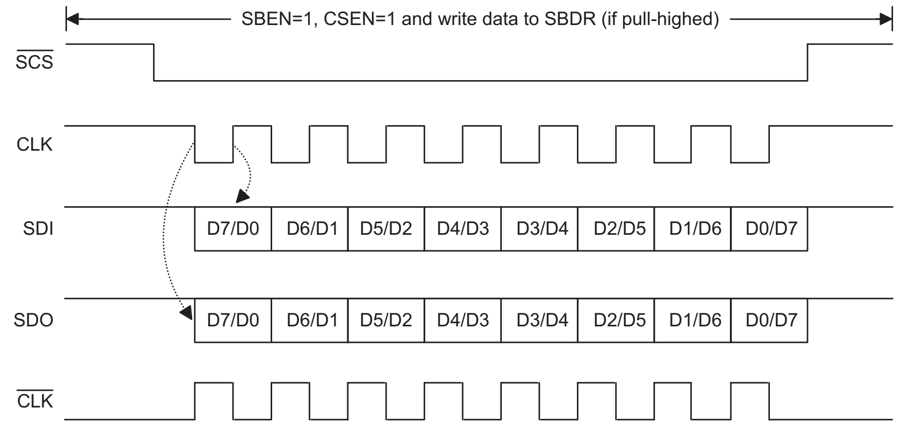
  Serial Interface Operation:

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 the TXRX buffer → output CLK (and $\overline{SCS}$) signals → go to step 5 → (SIO internal operation → data stored in the TXRX buffer, and the SDI data is shifted into the TXRX buffer → data transferred, data in the 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
Slavehans	• 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 the 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 the TXRX buffer → data transferred, data in the 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. The WCOL function can be enabled/disabled by a software option (SIO_WCOL bit of MODE_CTRL register). WCOL is set by SIO and cleared by the user.

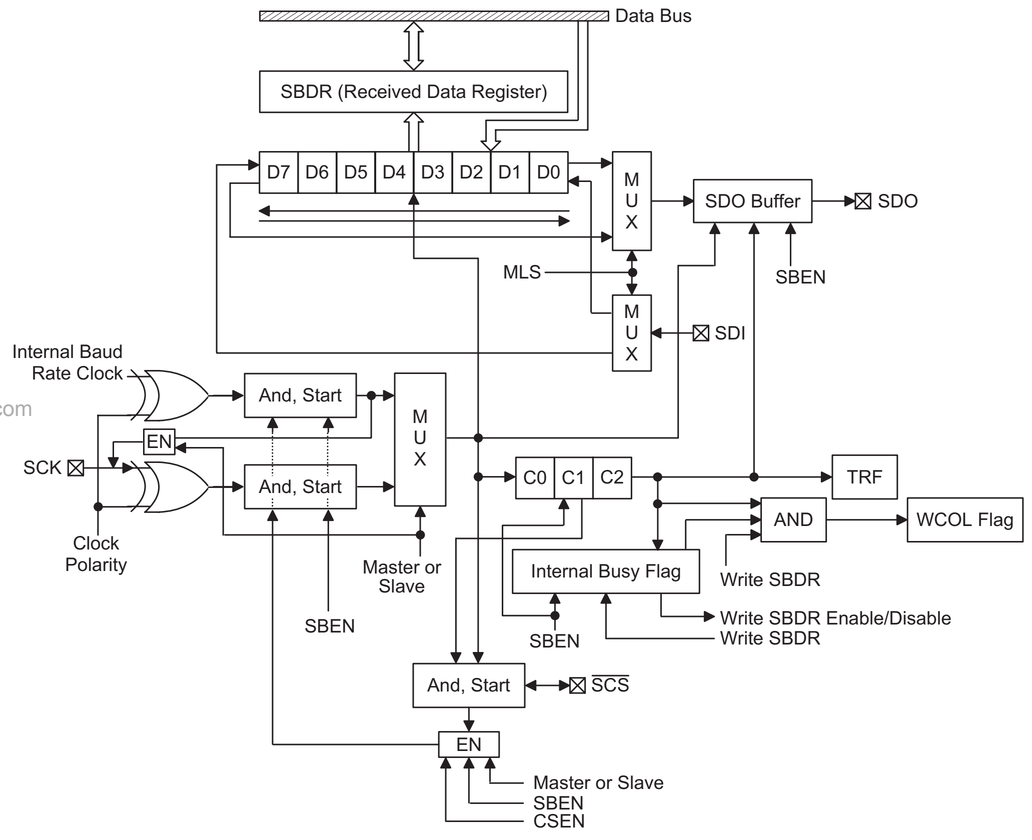
  Data transmission and reception will continue to operated when the MCU enters the power-down mode.

  CPOL is used to select the clock polarity of CLK and 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. The master should output a $\overline{SCS}$ signal before the CLK signal 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 the user. When the data is 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 device selection function pin Master mode: 1/0=with/without $\overline{\text{SCS}}$ output control Slave mode: 1/0= with/without $\overline{\text{SCS}}$ 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 in a floating state
TRF	1= data transmitted or received 0= data is transmitting or still not received

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

Mode Control

The MODE_CTRL register is used to control the DAC and ADC operational mode and the 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
```

Record Data

The record interrupt will be activated when the record data is valid in the RECORD_DATA registers. The RECORD_DATA registers will latch data until next interrupt occurs. The RECORD_DATA is 2′s complement value (8000H~7FFFH).

The update rate of the RECORD_DATA is 8kHz. All these registers (3EH~3FH) are read only.

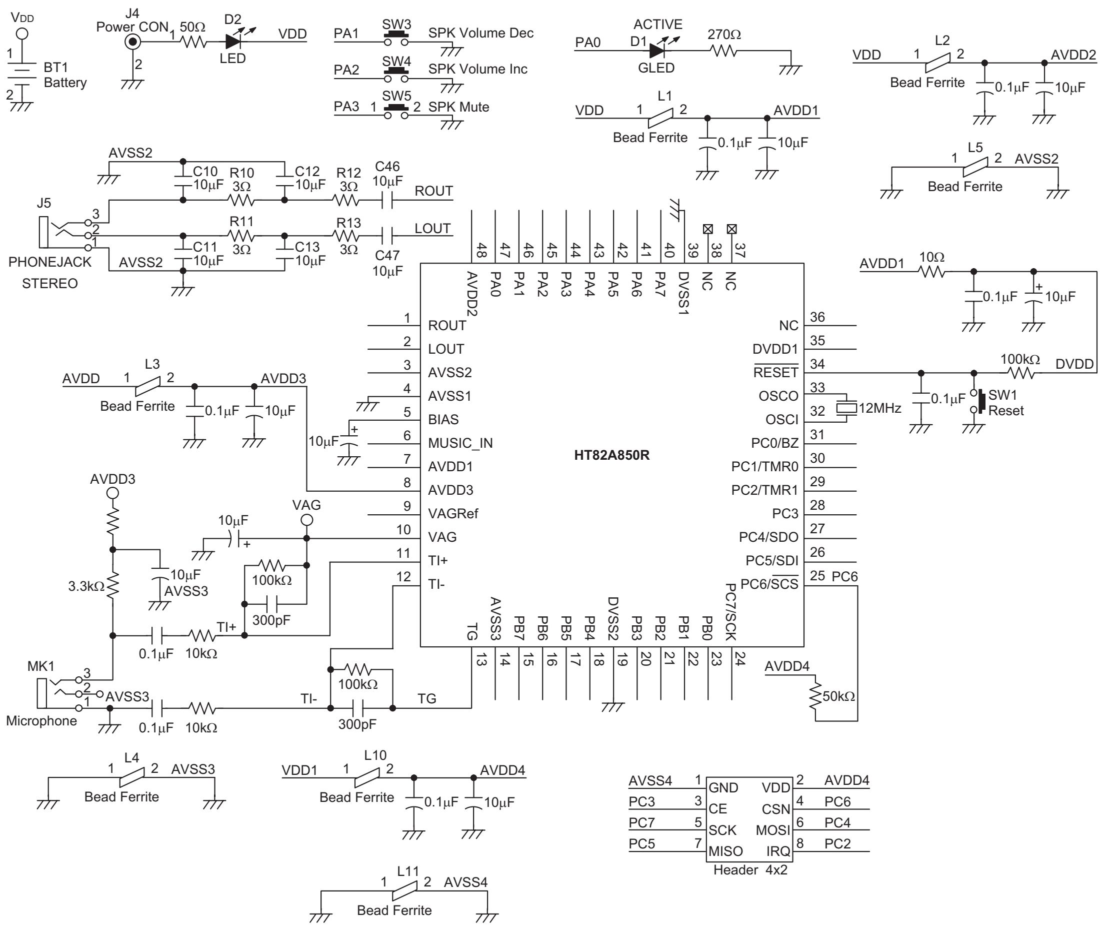
Address	Label	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
3EH	RECORD_DATA_L	R_D7	R_D6	R_D5	R_D4	R_D3	R_D2	R_D1	R_D0
3FH	RECORD_DATA_H	R_D15	R_D14	R_D13	R_D12	R_D11	R_D10	R_D9	R_D8

Configuration Options

The following table shows the microcontroller configuration options . All of the OTP options must be defined to ensure proper system functioning.

No.	Option
1	PA0~PA7 pull-high resistor enabled or disabled - bit option
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 - bit option
7	PB0~PB7 pull-high resistor enabled or disable - bit option
8	PC0~PC7 pull-high resistor enabled or disabled - nibble option
9	TBHP enable or disable - default disable

Application Circuits


VDD
BT1 Battery
J4 Power CON
50Ω
D2 LED
VDD
PA1 SW3 SPK Volume Dec
PA2 SW4 SPK Volume Inc
PA3 SW5 SPK Mute
PA0 ACTIVE D1 GLED 270Ω
VDD L1 Bead Ferrite AVDD1 0.1μF 10μF
VDD L2 Bead Ferrite AVDD2 0.1μF 10μF
L5 Bead Ferrite AVSS2
AVSS2 C10 10μF R10 3Ω C12 10μF R12 3Ω C46 10μF ROUT
J5 PHONEJACK STEREO
C11 10μF R11 3Ω C13 10μF R13 3Ω C47 10μF LOUT
AVSS2
HT82A850R
ROUT LOUT AVSS2 AVSS1 BIAS MUSIC_IN AVDD1 AVDD3 VAGRef VAG TI+ TI-
AVDD2 PA0 PA1 PA2 PA3 PA4 PA5 PA6 PA7 DVSS1 NC NC
NC DVDD1 RESET OSCO OSCI PC0/BZ PC1/TMR0 PC2/TMR1 PC3 PC4/SDO PC5/SDI PC6/SCS
TG AVSS3 PB7 PB6 PB5 PB4 DVSS2 PB3 PB2 PB1 PB0 PC7/SCK
AVDD1 10Ω 0.1μF 10μF
100kΩ DVDD
0.1μF SW1 Reset
12MHz
AVDD L3 Bead Ferrite AVDD3 0.1μF 10μF
10μF
AVDD3
10μF VAG
100kΩ 300pF
3.3kΩ 10μF AVSS3
0.1μF 10kΩ TI+
MK1 Microphone
AVSS3 0.1μF 10kΩ TI-
100kΩ 300pF TG
PC6
AVDD4 50kΩ
L4 Bead Ferrite AVSS3
VDD1 L10 Bead Ferrite AVDD4 0.1μF 10μF
L11 Bead Ferrite AVSS4
AVSS4 GND VDD AVDD4
PC3 CE CSN PC6
PC7 SCK MOSI PC4
PC5 MISO IRQ PC2
Header 4x2

Instruction Set

Introduction

Central to the successful operation of any microcontroller is its instruction set, which is a set of program instruction codes that directs the microcontroller to perform certain operations. In the case of Holtek microcontrollers, a comprehensive and flexible set of over 60 instructions is provided to enable programmers to implement their application with the minimum of programming overheads.

For easier understanding of the various instruction codes, they have been subdivided into several functional groupings.

Instruction Timing

Most instructions are implemented within one instruction cycle. The exceptions to this are branch, call, or table read instructions where two instruction cycles are required. One instruction cycle is equal to 4 system clock cycles, therefore in the case of an 8MHz system oscillator, most instructions would be implemented within 0.5μs and branch or call instructions would be implemented within 1μs. Although instructions which require one more cycle to implement are generally limited to the JMP, CALL, RET, RETI and table read instructions, it is important to realize that any other instructions which involve manipulation of the Program Counter Low register or PCL will also take one more cycle to implement. As instructions which change the contents of the PCL will imply a direct jump to that new address, one more cycle will be required. Examples of such instructions would be ″CLR PCL″ or ″MOV PCL, A″. For the case of skip instructions, it must be noted that if the result of the comparison involves a skip operation then this will also take one more cycle, if no skip is involved then only one cycle is required.

Moving and Transferring Data

The transfer of data within the microcontroller program is one of the most frequently used operations. Making use of three kinds of MOV instructions, data can be transferred from registers to the Accumulator and vice-versa as well as being able to move specific immediate data directly into the Accumulator. One of the most important data transfer applications is to receive data from the input ports and transfer data to the output ports.

Arithmetic Operations

The ability to perform certain arithmetic operations and data manipulation is a necessary feature of most microcontroller applications. Within the Holtek microcontroller instruction set are a range of add and subtract instruction mnemonics to enable the necessary arithmetic to be carried out. Care must be taken to ensure correct handling of carry and borrow data when results exceed 255 for addition and less than 0 for subtraction. The increment and decrement instructions INC, INCA, DEC and DECA provide a simple means of increasing or decreasing by a value of one of the values in the destination specified.

Logical and Rotate Operations

The standard logical operations such as AND, OR, XOR and CPL all have their own instruction within the Holtek microcontroller instruction set. As with the case of most instructions involving data manipulation, data must pass through the Accumulator which may involve additional programming steps. In all logical data operations, the zero flag may be set if the result of the operation is zero. Another form of logical data manipulation comes from the rotate instructions such as RR, RL, RRC and RLC which provide a simple means of rotating one bit right or left. Different rotate instructions exist depending on program requirements. Rotate instructions are useful for serial port programming applications where data can be rotated from an internal register into the Carry bit from where it can be examined and the necessary serial bit set high or low. Another application where rotate data operations are used is to implement multiplication and division calculations.

Branches and Control Transfer

Program branching takes the form of either jumps to specified locations using the JMP instruction or to a subroutine using the CALL instruction. They differ in the sense that in the case of a subroutine call, the program must return to the instruction immediately when the subroutine has been carried out. This is done by placing a return instruction RET in the subroutine which will cause the program to jump back to the address right after the CALL instruction. In the case of a JMP instruction, the program simply jumps to the desired location. There is no requirement to jump back to the original jumping off point as in the case of the CALL instruction. One special and extremely useful set of branch instructions are the conditional branches. Here a decision is first made regarding the condition of a certain data memory or individual bits. Depending upon the conditions, the program will continue with the next instruction or skip over it and jump to the following instruction. These instructions are the key to decision making and branching within the program perhaps determined by the condition of certain input switches or by the condition of internal data bits.

Bit Operations

The ability to provide single bit operations on Data Memory is an extremely flexible feature of all Holtek microcontrollers. This feature is especially useful for output port bit programming where individual bits or port pins can be directly set high or low using either the ″SET [m].i″ or ″CLR [m].i″ instructions respectively. The feature removes the need for programmers to first read the 8-bit output port, manipulate the input data to ensure that other bits are not changed and then output the port with the correct new data. This read-modify-write process is taken care of automatically when these bit operation instructions are used.

Table Read Operations

Data storage is normally implemented by using registers. However, when working with large amounts of fixed data, the volume involved often makes it inconvenient to store the fixed data in the Data Memory. To overcome this problem, Holtek microcontrollers allow an area of Program Memory to be setup as a table where data can be directly stored. A set of easy to use instructions provides the means by which this fixed data can be referenced and retrieved from the Program Memory.

Other Operations

In addition to the above functional instructions, a range of other instructions also exist such as the ″HALT″ instruction for Power-down operations and instructions to control the operation of the Watchdog Timer for reliable program operations under extreme electric or electromagnetic environments. For their relevant operations, refer to the functional related sections.

Instruction Set Summary

The following table depicts a summary of the instruction set categorised according to function and can be consulted as a basic instruction reference using the following listed conventions.

Table conventions:

x: Bits immediate data

m: Data Memory address

A: Accumulator

i: 0~7 number of bits

addr: Program memory address

Mnemonic	Description	Cycles	Flag Affected
Arithmetic			
ADD A,[m]	Add Data Memory to ACC	1	Z, C, AC, OV
ADDM A,[m]	Add ACC to Data Memory	1 Note	Z, C, AC, OV
ADD A,x	Add immediate data to ACC	1	Z, C, AC, OV
ADC A,[m]	Add Data Memory to ACC with Carry	1	Z, C, AC, OV
ADCM A,[m]	Add ACC to Data memory with Carry	1 Note	Z, C, AC, OV
SUB A,x	Subtract immediate data from the ACC	1	Z, C, AC, OV
SUB A,[m]	Subtract Data Memory from ACC	1	Z, C, AC, OV
SUBM A,[m]	Subtract Data Memory from ACC with result in Data Memory	1 Note	Z, C, AC, OV
SBC A,[m]	Subtract Data Memory from ACC with Carry	1	Z, C, AC, OV
SBCM A,[m]	Subtract Data Memory from ACC with Carry, result in Data Memory	1 Note	Z, C, AC, OV
DAA [m]	Decimal adjust ACC for Addition with result in Data Memory	1 Note	C
Logic Operation			
AND A,[m]	Logical AND Data Memory to ACC	1	Z
OR A,[m]	Logical OR Data Memory to ACC	1	Z
XOR A,[m]	Logical XOR Data Memory to ACC	1	Z
ANDM A,[m]	Logical AND ACC to Data Memory	1 Note	Z
ORM A,[m]	Logical OR ACC to Data Memory	1 Note	Z
XORM A,[m]	Logical XOR ACC to Data Memory	1 Note	Z
AND A,x	Logical AND immediate Data to ACC	1	Z
OR A,x	Logical OR immediate Data to ACC	1	Z
XOR A,x	Logical XOR immediate Data to ACC	1	Z
CPL [m]	Complement Data Memory	1 Note	Z
CPLA [m]	Complement Data Memory with result in ACC	1	Z
Increment & Decrement			
INCA [m]	Increment Data Memory with result in ACC	1	Z
INC [m]	Increment Data Memory	1 Note	Z
DECA [m]	Decrement Data Memory with result in ACC	1	Z
DEC [m]	Decrement Data Memory	1 Note	Z

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

Note: 1. For skip instructions, if the result of the comparison involves a skip then two cycles are required, if no skip takes place only one cycle is required.

2. Any instruction which changes the contents of the PCL will also require 2 cycles for execution.

3. For the ″CLR WDT1″ and ″CLR WDT2″ instructions the TO and PDF flags may be affected by the execution status. The TO and PDF flags are cleared after both ″CLR WDT1″ and ″CLR WDT2″ instructions are consecutively executed. Otherwise the TO and PDF flags remain unchanged.

Instruction Definition

ADC A,[m]	Add Data Memory to ACC with Carry
Description	The contents of the specified Data Memory, Accumulator and the carry flag are added. The result is stored in the Accumulator.
Operation	ACC ← ACC + [m] + C
Affected flag(s)	OV, Z, AC, C

ADCM A,[m]	Add ACC to Data Memory with Carry
Description	The contents of the specified Data Memory, Accumulator and the carry flag are added. The result is stored in the specified Data Memory.
Operation	[m] ← ACC + [m] + C
Affected flag(s)	OV, Z, AC, C

ADD A,[m]	Add Data Memory to ACC
Description	The contents of the specified Data Memory and the Accumulator are added. The result is stored in the Accumulator.
Operation	ACC ← ACC + [m]
Affected flag(s)	OV, Z, AC, C

ADD A,x	Add immediate data to ACC
Description	The contents of the Accumulator and the specified immediate data are added. The result is stored in the Accumulator.
Operation	ACC ← ACC + x
Affected flag(s)	OV, Z, AC, C

ADDM A,[m]	Add ACC to Data Memory
Description	The contents of the specified Data Memory and the Accumulator are added. The result is stored in the specified Data Memory.
Operation	[m] ← ACC + [m]
Affected flag(s)	OV, Z, AC, C

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

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

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

CALL addr	Subroutine call
Description	Unconditionally calls a subroutine at the specified address. The Program Counter then increments by 1 to obtain the address of the next instruction which is then pushed onto the stack. The specified address is then loaded and the program continues execution from this new address. As this instruction requires an additional operation, it is a two cycle instruction.
Operation	Stack ← Program Counter + 1 Program Counter ← addr
Affected flag(s)	None
CLR [m]	Clear Data Memory
Description	Each bit of the specified Data Memory is cleared to 0.
Operation	[m] ← 00H
Affected flag(s)	None
CLR [m].i	Clear bit of Data Memory
Description	Bit i of the specified Data Memory is cleared to 0.
Operation	[m].i ← 0
Affected flag(s)	None
CLR WDT	Clear Watchdog Timer
Description	The TO, PDF flags and the WDT are all cleared.

Operation	WDT cleared
	TO ← 0
	PDF ← 0

Affected flag(s)	TO, PDF
CLR WDT1	Pre-clear Watchdog Timer
Description	The TO, PDF flags and the WDT are all cleared. Note that this instruction works in conjunction with CLR WDT2 and must be executed alternately with CLR WDT2 to have effect. Repetitively executing this instruction without alternately executing CLR WDT2 will have no effect.

Operation	WDT cleared
	TO ← 0
	PDF ← 0

Affected flag(s)	TO, PDF
CLR WDT2	Pre-clear Watchdog Timer
Description	The TO, PDF flags and the WDT are all cleared. Note that this instruction works in conjunction with CLR WDT1 and must be executed alternately with CLR WDT1 to have effect. Repetitively executing this instruction without alternately executing CLR WDT1 will have no effect.

Operation	WDT cleared
	TO ← 0
	PDF ← 0

Affected flag(s)	TO, PDF

CPL [m]	Complement Data Memory
Description	Each bit of the specified Data Memory is logically complemented (1′s complement). Bits which previously contained a 1 are changed to 0 and vice versa.
Operation	[m] ← $\overline{[m]}$
Affected flag(s)	Z

CPLA [m]	Complement Data Memory with result in ACC
Description	Each bit of the specified Data Memory is logically complemented (1′s complement). Bits which previously contained a 1 are changed to 0 and vice versa. The complemented result is stored in the Accumulator and the contents of the Data Memory remain unchanged.
Operation	ACC ← $\overline{[m]}$
Affected flag(s)	Z

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

DEC [m]	Decrement Data Memory
Description	Data in the specified Data Memory is decremented by 1.
Operation	[m] ← [m] − 1
Affected flag(s)	Z

DECA [m]	Decrement Data Memory with result in ACC
Description	Data in the specified Data Memory is decremented by 1. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.
Operation	ACC ← [m] − 1
Affected flag(s)	Z

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

Operation	TO ← 0
	PDF ← 1

Affected flag(s)	TO, PDF

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

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

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

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

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

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

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

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

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

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

Affected flag(s)	None

RLC [m]	Rotate Data Memory left through Carry
Description	The contents of the specified Data Memory and the carry flag are rotated left by 1 bit. Bit 7 replaces the Carry bit and the original carry flag is rotated into bit 0.
Operation	[m].(i+1) ← [m].i; (i = 0~6) [m].0 ← C C ← [m].7
Affected flag(s)	C

RLCA [m]	Rotate Data Memory left through Carry with result in ACC
Description	Data in the specified Data Memory and the carry flag are rotated left by 1 bit. Bit 7 replaces the Carry bit and the original carry flag is rotated into the bit 0. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

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

Affected flag(s)	C

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

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

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

Affected flag(s)	None

RRC [m]	Rotate Data Memory right through Carry
Description	The contents of the specified Data Memory and the carry flag are rotated right by 1 bit. Bit 0 replaces the Carry bit and the original carry flag is rotated into bit 7.
Operation	[m].i ← [m].(i+1); (i = 0~6) [m].7 ← C C ← [m].0
Affected flag(s)	C

RRCA [m]	Rotate Data Memory right through Carry with result in ACC
Description	Data in the specified Data Memory and the carry flag are rotated right by 1 bit. Bit 0 replaces the Carry bit and the original carry flag is rotated into bit 7. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

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

Affected flag(s)	C

SBC A,[m]	Subtract Data Memory from ACC with Carry
Description	The contents of the specified Data Memory and the complement of the carry flag are subtracted from the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.
Operation	ACC ← ACC − [m] − $\overline{C}$
Affected flag(s)	OV, Z, AC, C
SBCM A,[m]	Subtract Data Memory from ACC with Carry and result in Data Memory
Description	The contents of the specified Data Memory and the complement of the carry flag are subtracted from the Accumulator. The result is stored in the Data Memory. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.
Operation	[m] ← ACC − [m] − $\overline{C}$
Affected flag(s)	OV, Z, AC, C
SDZ [m]	Skip if decrement Data Memory is 0
Description	The contents of the specified Data Memory are first decremented by 1. If the result is 0 the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.
Operation	[m] ← [m] − 1 Skip if [m] = 0
Affected flag(s)	None
SDZA [m]	Skip if decrement Data Memory is zero with result in ACC
Description	The contents of the specified Data Memory are first decremented by 1. If the result is 0, the following instruction is skipped. The result is stored in the Accumulator but the specified Data Memory contents remain unchanged. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0, the program proceeds with the following instruction.
Operation	ACC ← [m] − 1 Skip if ACC = 0
Affected flag(s)	None
SET [m]	Set Data Memory
Description	Each bit of the specified Data Memory is set to 1.
Operation	[m] ← FFH
Affected flag(s)	None
SET [m].i	Set bit of Data Memory
Description	Bit i of the specified Data Memory is set to 1.
Operation	[m].i ← 1
Affected flag(s)	None

SIZ [m]	Skip if increment Data Memory is 0
Description	The contents of the specified Data Memory are first incremented by 1. If the result is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.
Operation	[m] ← [m] + 1 Skip if [m] = 0
Affected flag(s)	None
SIZA [m]	Skip if increment Data Memory is zero with result in ACC
Description	The contents of the specified Data Memory are first incremented by 1. If the result is 0, the following instruction is skipped. The result is stored in the Accumulator but the specified Data Memory contents remain unchanged. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.
Operation	ACC ← [m] + 1 Skip if ACC = 0
Affected flag(s)	None
SNZ [m].i	Skip if bit i of Data Memory is not 0
Description	If bit i of the specified Data Memory is not 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is 0 the program proceeds with the following instruction.
Operation	Skip if [m].i ≠ 0
Affected flag(s)	None
SUB A,[m]	Subtract Data Memory from ACC
Description	The specified Data Memory is subtracted from the contents of the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.
Operation	ACC ← ACC − [m]
Affected flag(s)	OV, Z, AC, C
SUBM A,[m]	Subtract Data Memory from ACC with result in Data Memory
Description	The specified Data Memory is subtracted from the contents of the Accumulator. The result is stored in the Data Memory. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.
Operation	[m] ← ACC − [m]
Affected flag(s)	OV, Z, AC, C
SUB A,x	Subtract immediate data from ACC
Description	The immediate data specified by the code is subtracted from the contents of the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.
Operation	ACC ← ACC − x
Affected flag(s)	OV, Z, AC, C

SWAP [m]	Swap nibbles of Data Memory
Description	The low-order and high-order nibbles of the specified Data Memory are interchanged.
Operation	[m].3~[m].0 ↔ [m].7 ~ [m].4
Affected flag(s)	None

SWAPA [m]	Swap nibbles of Data Memory with result in ACC
Description	The low-order and high-order nibbles of the specified Data Memory are interchanged. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.
Operation	ACC.3 ~ ACC.0 ← [m].7 ~ [m].4 ACC.7 ~ ACC.4 ← [m].3 ~ [m].0
Affected flag(s)	None

SZ [m]	Skip if Data Memory is 0
Description	If the contents of the specified Data Memory is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.
Operation	Skip if [m] = 0
Affected flag(s)	None

SZA [m]	Skip if Data Memory is 0 with data movement to ACC
Description	The contents of the specified Data Memory are copied to the Accumulator. If the value is zero, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.
Operation	ACC ← [m] Skip if [m] = 0
Affected flag(s)	None

SZ [m].i	Skip if bit i of Data Memory is 0
Description	If bit i of the specified Data Memory is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0, the program proceeds with the following instruction.
Operation	Skip if [m].i = 0
Affected flag(s)	None

TABRDC [m]	Read table (current page) to TBLH and Data Memory
Description	The low byte of the program code (current page) addressed by the table pointer (TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.
Operation	[m] ← program code (low byte) TBLH ← program code (high byte)
Affected flag(s)	None

TABRDL [m]	Read table (last page) to TBLH and Data Memory
Description	The low byte of the program code (last page) addressed by the table pointer (TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.
Operation	[m] ← program code (low byte) TBLH ← program code (high byte)
Affected flag(s)	None

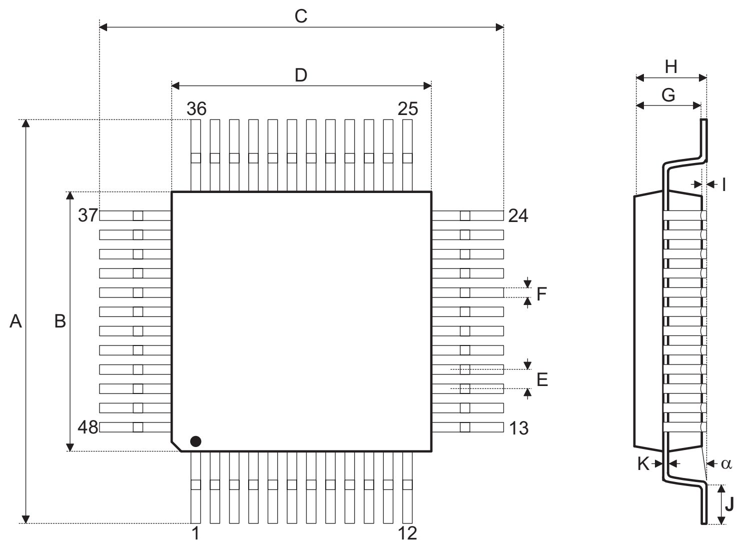
XOR A,[m]	Logical XOR Data Memory to ACC
Description	Data in the Accumulator and the specified Data Memory perform a bitwise logical XOR operation. The result is stored in the Accumulator.
Operation	ACC ← ACC ″XOR″ [m]
Affected flag(s)	Z

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

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

Package Information

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°

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

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

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

Holtek Semiconductor Inc. (Shenzhen Sales Office)
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: 86-755-8616-9908, 86-755-8616-9308
Fax: 86-755-8616-9722

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

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

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