Features

8-bit microcontroller

- Operating voltage: 2.4V~5.2V
- 2K × 14 or 4K × 15 program ROM
- 80 × 8 or 112 × 8 data RAM
- 24 bidirectional I/O lines
- An interrupt input
- An 8-bit programmable timer/event counter with overflow interrupts
- A watch dog timer
- On-chip crystal and RC types of oscillator
- Halt function to reduce power consumption and a wake-up feature
- 63 powerful instructions
- Up to a 1μs instruction cycle with a 4MHz system clock at V_{DD}=5V
- All instructions in 1 or 2 machine cycles
- 14 bit table read instruction
- Four level subroutine nesting
- Bit manipulation instruction

Voice and melody synthesizer

- 32K × 8~128K × 8 voice ROM
- 3/4 bit ADPCM coding algorithm
- 26 kinds of voice sampling rates
- A tone level of 4 octaves
- 14 kinds of melody beats
- Current type of D/A switch output
- A tone generator counter
- Controllable volume

Applications

- Intelligent educational toys
- High end toy controllers
- Talking alarm clocks
- Alert & warning system
- Public address systems
- Sound effect generators

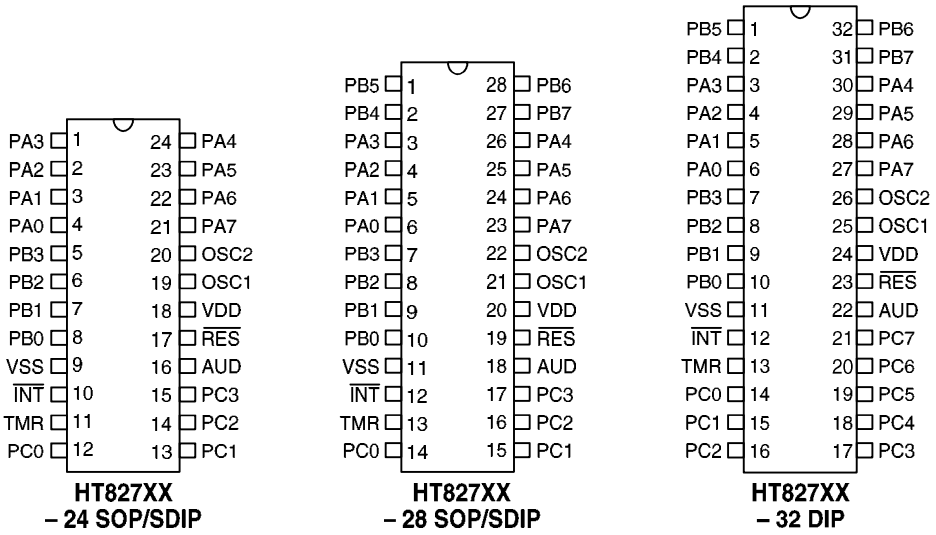
General Description

The HT827XX series are an 8 bit high performance microcontroller with a voice synthesizer and tone generator. They are designed for applications on multiple I/Os with sound effects. The LSIs provide 26 kinds of voice sampling rates, 4 octaves of tone level as well as a high quality of current type D/A output. With such a flexible structure, the HT827XX series are excellent for applications on versatile voice and sound effect products. They also include a halt function to reduce power consumption.

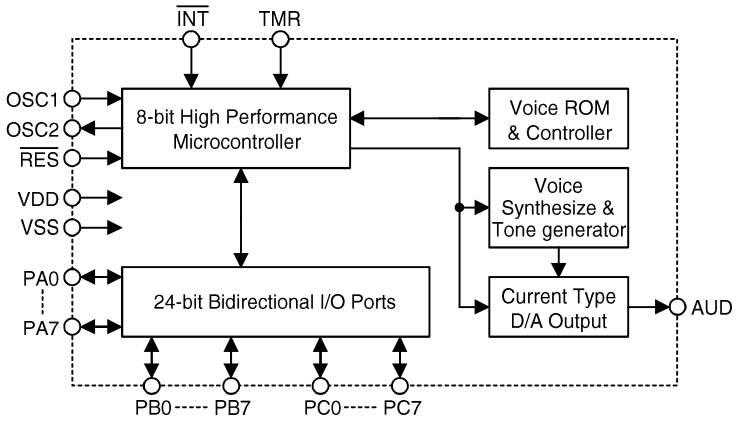
Selection Table

Item	Program ROM	Data RAM	Voice ROM
HT82700	4K × 15	112 × 8	128K × 8
HT82720	4K × 15	112 × 8	96K × 8
HT82740	4K × 15	112 × 8	64K × 8
HT82770	2K × 14	80 × 8	64K × 8
HT82780	2K × 14	80 × 8	32K × 8

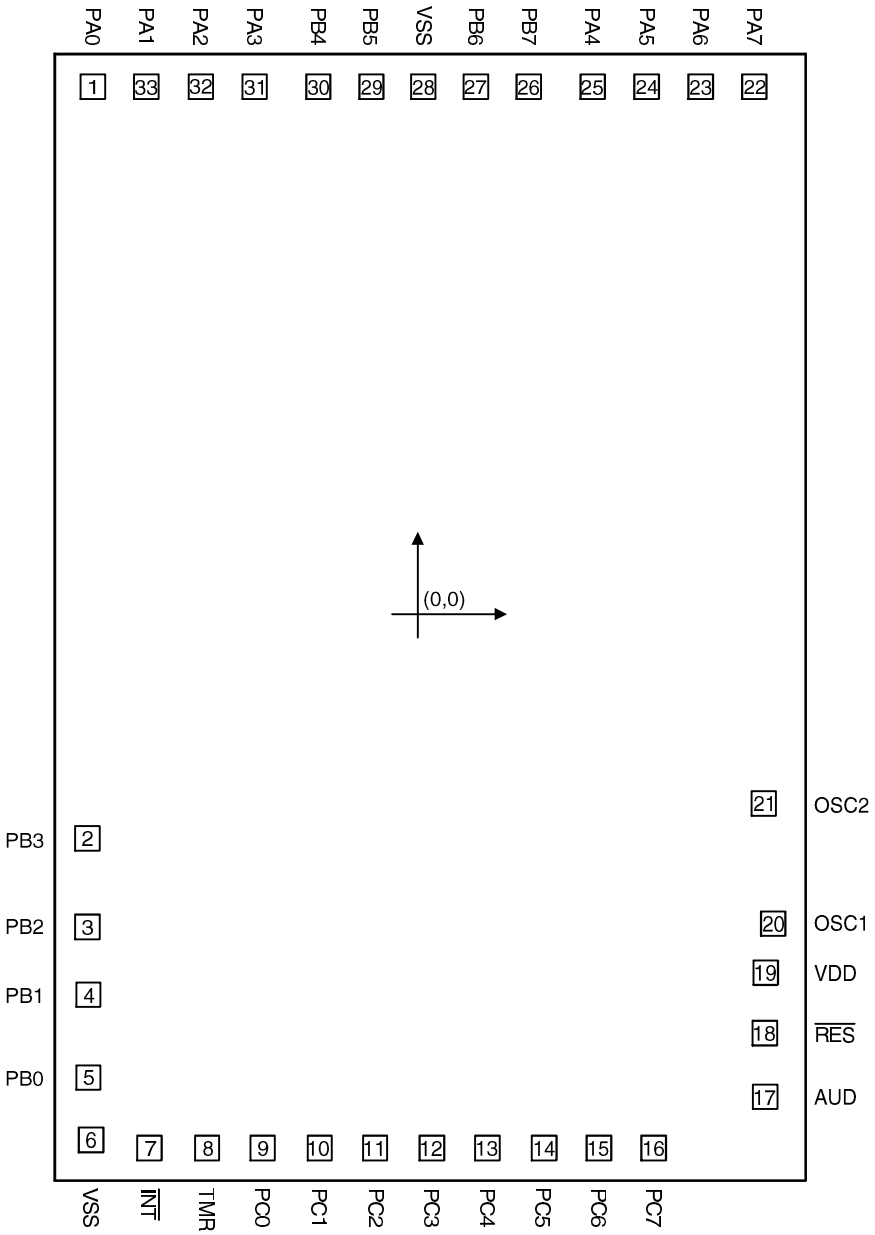
Pin Assignment



Block Diagram



Pad Assignment (HT82700)



Chip size: 3760×5680 $(\mu m)^2$

* The IC substrate should be connected to VSS in the PCB layout artwork.

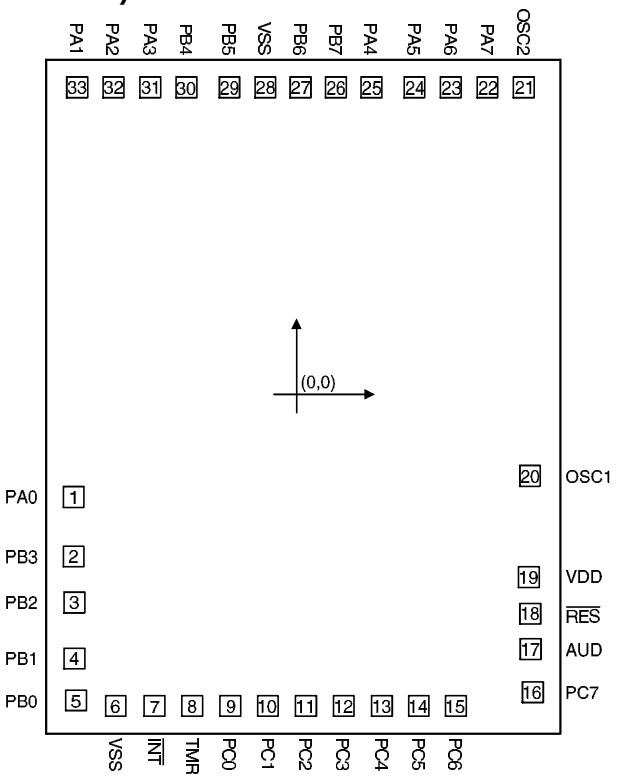
* The TMR pad must be bound to VDD or VSS if it is not used.

Pad Coordinates (HT82700)

Unit: μm

Pad No.	X	Y	Pad No.	X	Y
1	–1597.90	2589.80	18	1703.20	–2057.70
2	–1624.50	–1101.80	19	1708.20	–1761.80
3	–1624.90	–1536.00	20	1744.20	–1521.20
4	–1618.80	–1869.00	21	1698.80	–930.60
5	–1618.80	–2276.80	22	1652.90	2589.80
6	–1605.70	–2582.30	23	1389.20	2589.80
7	–1318.60	–2622.80	24	1122.10	2589.80
8	–1035.80	–2622.80	25	858.40	2589.80
9	–762.50	–2622.80	26	544.80	2589.90
10	–483.80	–2622.80	27	285.10	2589.90
11	–210.50	–2622.80	28	27.80	2589.70
12	68.20	–2622.80	29	–229.50	2589.90
13	341.50	–2622.80	30	–489.20	2589.90
14	620.20	–2622.80	31	–802.80	2589.80
15	889.35	–2622.80	32	–1066.50	2589.80
16	1158.10	–2622.80	33	–1334.20	2589.80
17	1706.20	–2371.60			

Pad Assignment (HT82720)



Chip size: 3750×4970 $(\mu m)^2$

* The IC substrate should be connected to VSS in the PCB layout artwork.

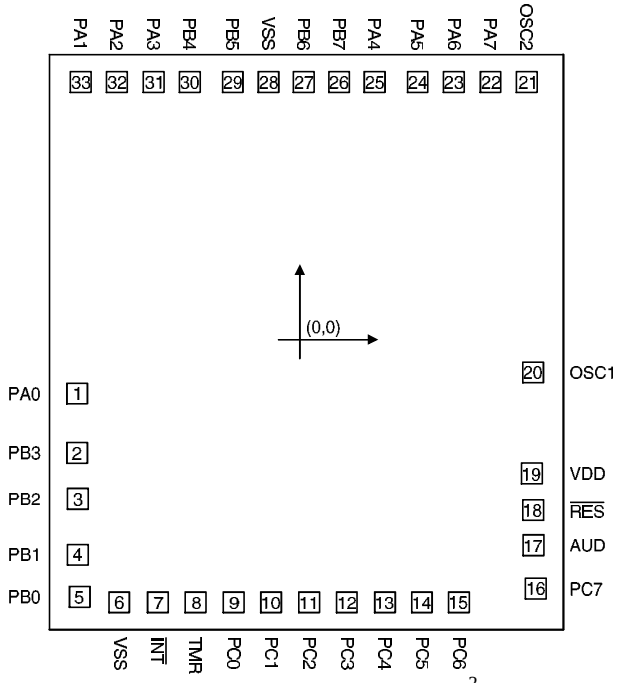
* The TMR pad must be bound to VDD or VSS if it is not used.

Pad Coordinates (HT82720)

Unit: μm

Pad No.	X	Y	Pad No.	X	Y
1	–1624.40	–746.80	18	1701.10	–1596.30
2	–1624.80	–1181.00	19	1689.40	–1329.10
3	–1618.70	–1514.00	20	1698.90	–592.70
4	–1618.70	–1921.80	21	1653.00	2234.80
5	–1605.60	–2227.30	22	1389.30	2234.80
6	–1318.50	–2267.80	23	1122.20	2234.80
7	–1035.70	–2267.80	24	858.50	2234.80
8	–762.40	–2267.80	25	544.90	2234.90
9	–483.70	–2267.80	26	285.20	2234.90
10	–210.40	–2267.80	27	27.90	2234.70
11	68.30	–2267.80	28	–229.40	2234.90
12	341.60	–2267.80	29	–489.10	2234.90
13	620.30	–2267.80	30	–802.70	2234.80
14	889.50	–2267.80	31	–1066.40	2234.80
15	1158.20	–2267.80	32	–1334.10	2234.80
16	1718.00	–2167.10	33	–1597.80	2234.80
17	1703.30	–1853.10			

Pad Assignment (HT82740)



Chip size: 3750 × 4260 (μm)^2

* The IC substrate should be connected to VSS in the PCB layout artwork.

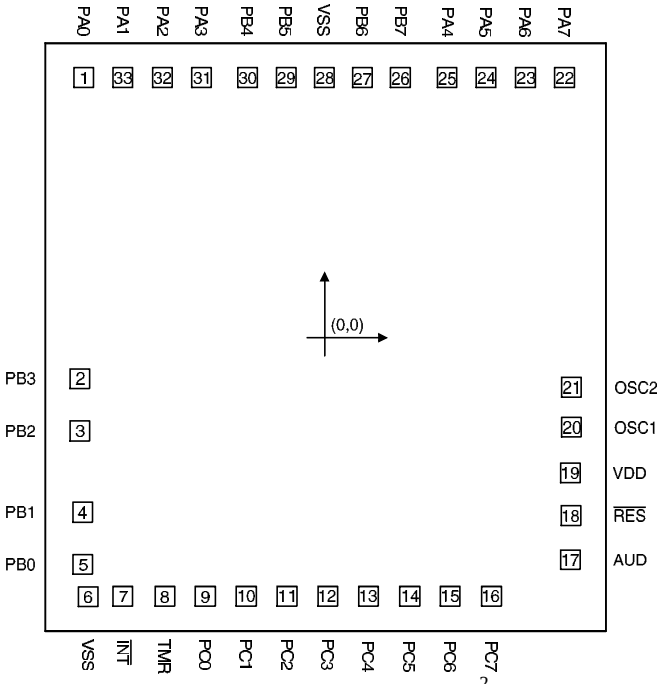
* The TMR pad must be bound to VDD or VSS if it is not used.

Pad Coordinates (HT82740)

Unit: μm

Pad No.	X	Y	Pad No.	X	Y
1	–1624.40	–391.60	18	1701.10	–1241.10
2	–1624.80	–825.80	19	1689.40	–973.90
3	–1618.70	–1158.80	20	1698.90	–237.50
4	–1618.70	–1566.60	21	1653.00	1879.60
5	–1605.60	–1872.10	22	1389.30	1879.60
6	–1318.50	–1912.60	23	1122.20	1879.60
7	–1035.70	–1912.60	24	858.50	1879.60
8	–762.40	–1912.60	25	544.90	1879.70
9	–483.70	–1912.60	26	285.20	1879.70
10	–210.40	–1912.60	27	27.90	1879.50
11	68.30	–1912.60	28	–229.40	1879.70
12	341.60	–1912.60	29	–489.10	1879.70
13	620.30	–1912.60	30	–802.70	1879.60
14	889.50	–1912.60	31	–1066.40	1879.60
15	1158.20	–1912.60	32	–1334.10	1879.60
16	1718.00	–1811.90	33	–1597.80	1879.60
17	1703.30	–1497.90			

Pad Assignment (HT82770)



Chip size: 3700 × 3940 (μm)^2

* The IC substrate should be connected to VSS in the PCB layout artwork.

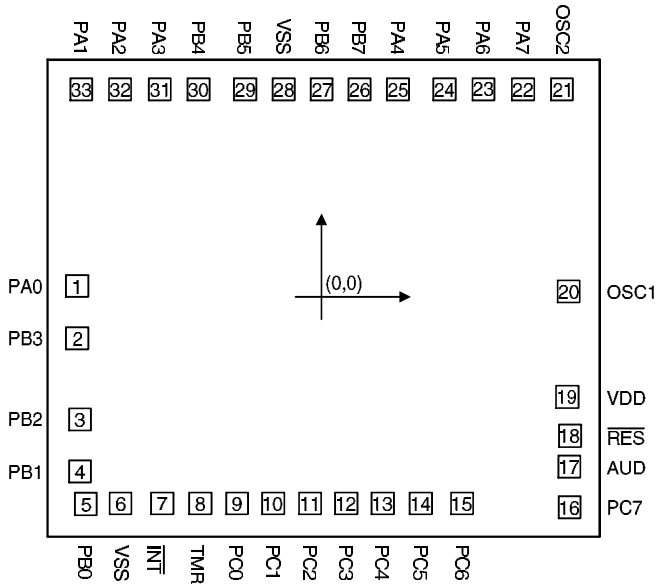
* The TMR pad must be bound to VDD or VSS if it is not used.

Pad Coordinates (HT82770)

Unit: μm

Pad No.	X	Y	Pad No.	X	Y
1	–1624.40	1757.90	18	1676.50	–1343.85
2	–1651.60	–277.10	19	1681.50	–1097.55
3	–1651.60	–631.10	20	1717.50	–916.65
4	–1631.50	–1179.30	21	1672.10	–326.05
5	–1631.50	–1530.80	22	1626.40	1757.90
6	–1592.50	–1750.10	23	1362.70	1757.90
7	–1358.70	–1746.70	24	1095.60	1757.90
8	–1075.90	–1746.70	25	831.90	1757.90
9	–802.60	–1746.70	26	518.30	1758.00
10	–523.90	–1746.70	27	258.60	1758.00
11	–250.60	–1746.70	28	1.30	1757.80
12	28.10	–1746.70	29	–256.00	1758.00
13	301.40	–1746.70	30	–515.00	1758.00
14	580.10	–1746.70	31	–829.30	1757.90
15	849.25	–1746.70	32	–1093.00	1857.90
16	1118.00	–1746.70	33	–1360.70	1857.90
17	1679.50	–1636.65			

Pad Assignment (HT82780)



Chip size: 3700 × 3230 (μm)^2

* The IC substrate should be connected to VSS in the PCB layout artwork.

* The TMR pad must be bound to VDD or VSS if it is not used.

Pad Coordinates (HT82780)

Unit: μm

Pad No.	X	Y	Pad No.	X	Y
1	–1651.60	75.05	18	1681.50	–936.55
2	–1651.60	–278.95	19	1662.60	–674.95
3	–1631.50	–827.15	20	1672.10	39.15
4	–1631.50	–1178.65	21	1626.40	1405.75
5	–1592.50	–1397.95	22	1362.70	1405.75
6	–1358.70	–1394.55	23	1095.60	1405.75
7	–1075.90	–1394.55	24	831.90	1405.75
8	–817.60	–1394.55	25	518.30	1405.85
9	–568.90	–1394.55	26	258.60	1405.85
10	–325.60	–1394.55	27	1.30	1405.65
11	–76.90	–1394.55	28	–256.00	1405.85
12	166.40	–1394.55	29	–515.70	1405.85
13	415.10	–1394.55	30	–829.30	1405.75
14	673.40	–1394.55	31	–1093.00	1405.75
15	952.10	–1394.55	32	–1360.70	1405.75
16	1679.50	–1420.85	33	–1624.40	1405.75
17	1676.50	–1147.05			

Pad Description

Pad No.	Pad Name	I/O	Mask Option	Function
1 31~33 22~25	PA0~PA7	I/O	Wake-up Pull-High or none	Bidirectional 8 bit input/output ports Each bit can be configured as a wake-up input by mask option. Software instructions determine the CMOS output or schmitt trigger input with or without a pull high resistor (mask option).
2~5 26,27 29,30	PB0~PB7	I/O	Pull-High or none	Bidirectional 8 bit input/output ports Software instructions determine the CMOS output or schmitt trigger input with or without a pull high resistor (mask option).
6,28	VSS	—	—	Negative power supply (GND)
7	$\overline{\text{INT}}$	I	—	External interrupt schmitt trigger input with a pull-high resistor Edge triggered is activated on a high to low transition.
8	TMR	I	—	Schmitt trigger input for a timer/event counter
9~16	PC0~PC7	I/O	Pull-High or none	Bidirectional 8 bit input/output ports Software instructions determine the CMOS output or schmitt trigger input with or without a pull-high resistor (mask option).

Pad No.	Pad Name	I/O	Mask Option	Function
17	AUD	O	—	Audio output for driving an external transistor PMOS open drain output
18	$\overline{RES}$	I	—	Schmitt trigger reset input, active low
19	VDD	—	—	Positive power supply
20 21	OSC1 OSC2	I O	Crystal or RC	OSC1 and OSC2 connect to an RC network or crystal (determined by mask option) for an internal system clock. In the case of RC operation, an oscillation resistor connects to OSC1. OSC2 is the output terminal of a 1/4 system clock.

Absolute Maximum Ratings

Supply Voltage .....–0.3V to 5.5V
Input Voltage..... V_{SS}–0.3V to V_{DD}+0.3V
Storage Temperature.....–50°C to 125°C
Operating Temperature.....–25°C to 70°C

D.C. Characteristics

(Ta=25°C)

Symbol	Parameter	Test Condition		Min.	Typ.	Max.	Unit
		V_{DD}	Condition				
V_{DD}	Operating Voltage	—	—	2.4	—	5.2	V
I_{DD1}	Operating Current (Crystal OSC)	3V	No load, f_{SYS}=4MHz	—	0.7	1.5	mA
		5V		—	2	3	mA
I_{DD2}	Operating Current (RC OSC)	3V	No load, f_{SYS}=4MHz	—	1.5	3	mA
		5V		—	2.5	5	mA
I_{STB1}	Stand-by Current (WDT Enabled)	3V	No load, system HALT	—	—	10	μA
		5V		—	—	20	μA
I_{STB2}	Stand-by Current (WDT Disabled)	3V	No load, system HALT	—	—	3	μA
		5V		—	—	5	μA
V_{IL}	Input Low Voltage for I/O Ports	3V	—	0	—	0.6	V
		5V	—	0	—	1.0	V
V_{IH}	Input High Voltage for I/O Ports	3V	—	2.4	—	3	V
		5V	—	4.0	—	5	V
V_{IL1}	Input Low Voltage ($\overline{RES}$,TMR, $\overline{INT}$)	3V	—	0	—	0.6	V
		5V	—	0	—	1.0	V

Symbol	Parameter	Test Condition		Min.	Typ.	Max.	Unit
		V_{DD}	Condition				
V_{IH1}	Input High Voltage ($\overline{RES}$,TMR, $\overline{INT}$)	3V	—	2.4	—	3	V
		5V	—	4.0	—	5	V
I_{OL1}	I/O Port Sink Current (PortA, PortC)	3V	V_{DD}=3V, V_{OL}=0.3V	2	4	—	mA
		5V	V_{DD}=5V, V_{OL}=0.5V	6	10	—	mA
I_{OH1}	I/O Port Source Current (PortA, PortC)	3V	V_{DD}=3V, V_{OH}=2.7V	–1	–1.5	—	mA
		5V	V_{DD}=5V, V_{OH}=4.5V	–2	–4	—	mA
I_{OL2}	PortB Sink Current	3V	V_{DD}=3V, V_{OL}=0.3V	6	10	—	mA
		5V	V_{DD}=5V, V_{OL}=0.5V	20	25	—	mA
I_{OH2}	PortB Source Current	3V	V_{DD}=3V, V_{OH}=2.7V	–1	–1.5	—	mA
		5V	V_{DD}=5V, V_{OH}=4.5V	–2	–4	—	mA
R_{PH}	Pull-High Resistance of I/O Ports & $\overline{INT}$	3V	—	25	50	100	KΩ
		5V	—	10	30	60	KΩ
I_O	Max. AUD Output Current	3V	V_{OH}=0.6V	–1.5	–2	—	mA

A.C. Characteristics

(Ta=25°C)

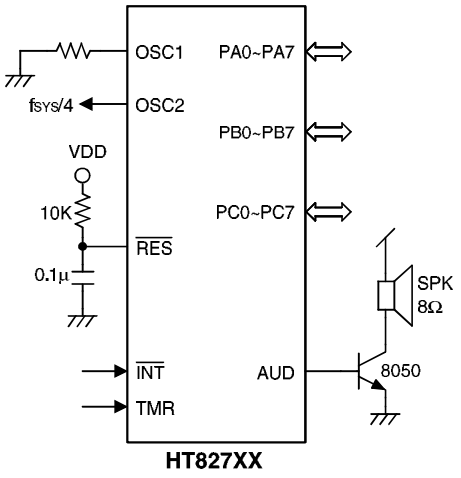
Symbol	Parameter	Test Condition		Min.	Typ.	Max.	Unit
		V_{DD}	Condition				
f_{SYS1}	System Clock (Crystal OSC)	3V	—	400	—	4000	KHz
		5V	—	400	—	4000	KHz
f_{SYS2}	System Clock (RC OSC)	3V	—	400	—	4000	KHz
		5V	—	400	—	4000	KHz
f_{TIMER}	Timer I/P Frequency (TMR)	3V	—	0	—	4000	KHz
		5V	—	0	—	4000	KHz
t_{WDTOSC}	Watch Dog Oscillator	5V	—	31	78	140	μs

Symbol	Parameter	Test Condition		Min.	Typ.	Max.	Unit
		V_{DD}	Condition				
t_{WDT1}	Watch Dog Timeout Period (RC)	5V	Without WDT prescaler	8	20	36	ms
t_{WDT2}	Watch Dog Timeout Period (System Clock)	5V	Without WDT prescaler	—	1024	—	tSYS
t_{RES}	External Reset Low Pulse Width	5V	—	1	—	—	μs
t_{INT}	Interrupt Pulse Width	5V	—	1	—	—	μs

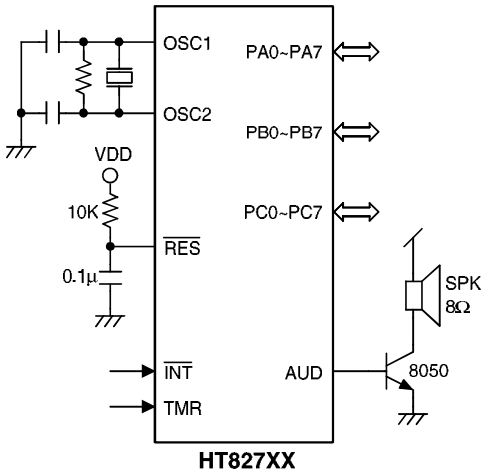
Note: $t_{SYS}=1/(f_{SYS})$

Application Circuits

RC oscillator for multiple I/O applications



Crystal oscillator for multiple I/O applications



System Architecture

Executive flow

The HT827XX series provide a system clock which is derived from a crystal or an RC type of oscillator. The clock is internally divided into four non-overlapping clocks denoted by P1, P2, P3 and P4. An instruction cycle consists of T1~T4.

Instruction fetching and execution are pipelined in such a way that a fetch takes an instruction cycle while decoding and execution take the next instruction cycle. The pipelining scheme causes each instruction to execute effectively in a cycle. If an instruction changes the program counter, two cycles are required to complete that instruction.

Program counter - PC

The program counter (PC) controls the sequence in which the instructions stored in the program ROM are executed.

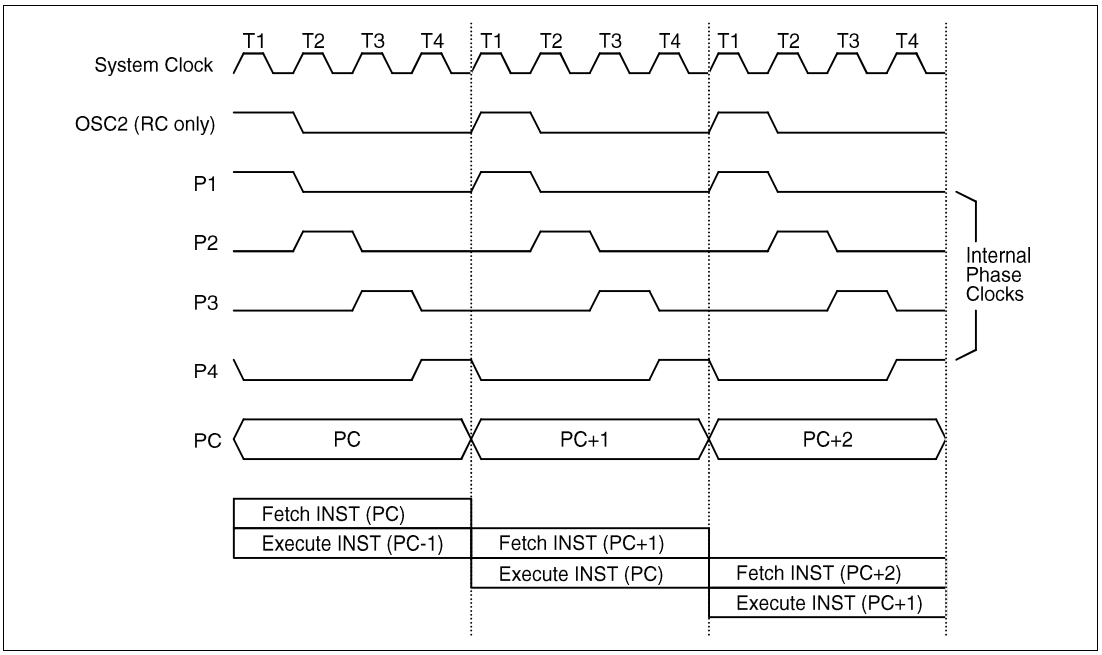
The contents of the program counter are incremented by one after a program memory word is accessed to fetch an instruction code. The program counter then points to a memory word containing the next instruction code.

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

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

The lower byte of the program counter (PCL) is a readable and writable register (06H). Moving data into PCL performs a short jump. The destination is within 256 locations.

Once a control transfer takes place, the execution suffers from an additional dummy cycle.



Execution Flow

Program memory - ROM

The program memory stores the to-be-executed program instructions. It also includes data, table and interrupt entries, addressed by the program counter along with the table pointer.

The program memory size for HT827XX series are shown as following.

HT82700, HT82720, HT82740	4K × 15
HT82770, HT82780	2K × 14

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

- Location 000H

  This area is reserved for program initialization. The program always begins execution at location 000H each time the system is reset.

- Location 004H

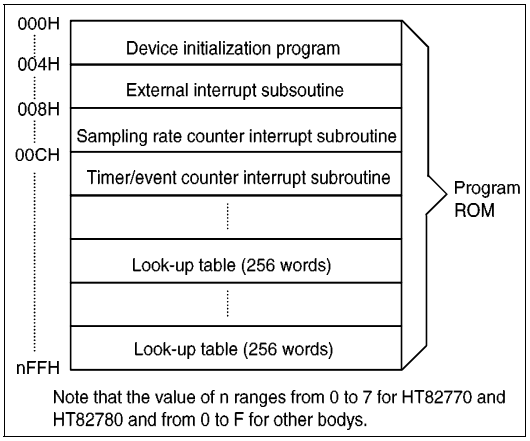
  This area is reserved for an external interrupt service program. The program begins execution at location 004H if the $\overline{INT}$ input pin is activated, the interrupt is enabled and the stack is not full.

- Location 008H

  This area is reserved for a voice sampling rate counter interrupt service program. The program begins execution at location 008H if a timer interrupt results from a sampling rate counter overflow, the interrupt is enabled and the stack is not full.

- Location 00CH

  This area is reserved for a timer/event counter interrupt service program. The program begins execution at location 00CH if an interrupt results from a timer/event counter overflow, the interrupt is enabled and the stack is not full.



Program Memory

Mode	Program Counter											
	***11**	***10**	***9**	***8**	***7**	***6**	***5**	***4**	***3**	***2**	***1**	***0**
Initial reset	0	0	0	0	0	0	0	0	0	0	0	0
External interrupt	0	0	0	0	0	0	0	0	0	1	0	0
Sampling rate counter overflow	0	0	0	0	0	0	0	0	1	0	0	0
Timer/event Counter overflow	0	0	0	0	0	0	0	0	1	1	0	0
Skip	PC+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

Notes: *11(*10)~*0: Bits of program counter S11(*S10)~S0: Bits of stack register
#11(#10)~#0: Bits of instruction code @7~@0: Bits of PCL
The bit 11 of PC counter is omitted for HT82770 and HT82780

- Table location

  Any location in the program ROM can be used as a look-up table. The instructions "TABRDC [m]" (the current page, 1 page=256 words) and "TABRDL [m]" (the last page) transfer the contents of the lower-order byte to the specified data memory, and the higher-order byte to TBLH (08H). Only the destination of the lower-order byte in the table is well-defined. The other bits of the table word are transferred to the lower portion of TBLH. The higher-order byte register (TBLH) of the table is read only. The table pointer (TBLP), on the other hand, is a read/write register (07H) indicating the table location. This location must be placed in TBLP before accessing the table. All the table related instructions require 2 cycles to complete an operation. These areas may function as a normal program memory depending upon the user's requirements.

Stack register - Stack

The stack register is a special part of the memory used to save the contents of the program counter (PC). This stack is organized into 4 levels. It is neither part of the data nor program space, and is neither readable nor writable. Its activated level is indexed by a stack pointer (SP) and is neither readable nor writable. At a subroutine call or interrupt acknowledgment, the contents of the program counter are pushed onto the stack. The program counter is restored to its previous value from the stack at the end of a subroutine or interrupt routine, which is signaled by a return instruction (RET or RETI). After a chip reset, SP will point to the top of the stack.

The interrupt request flag will be recorded but the acknowledgment be inhibited when the stack is full and a non-masked interrupt takes place. After the stack pointer is decremented (by RET or RETI), the interrupt will be serviced. This feature prevents stack overflow and allows programmers to use the structure more easily. In a similar case, if the stack is full and a "CALL" is subsequently executed, stack overflow occurs and the first entry is lost (only the most recent two return addresses are stored).

Data memory - RAM

The data memory is further divided into two functional groups, namely special function registers and general purpose data memories. Although most of them are readable/writable, some are read only.

The data memory size for HT827XX series are shown as following.

Item	RAM Size	Special RAM	General RAM Address
HT82700 HT82720 HT82740	137 × 8	00H~2FH	30H~9FH
HT82770 HT82780	105 × 8	00H~2FH	30H~7FH

The special function registers include an indirect addressing register (00H), timer/event counter (TMR; 10H), timer/event counter control register (TMRC; 11H), program counter lower-order byte register (PCL; 06H), memory

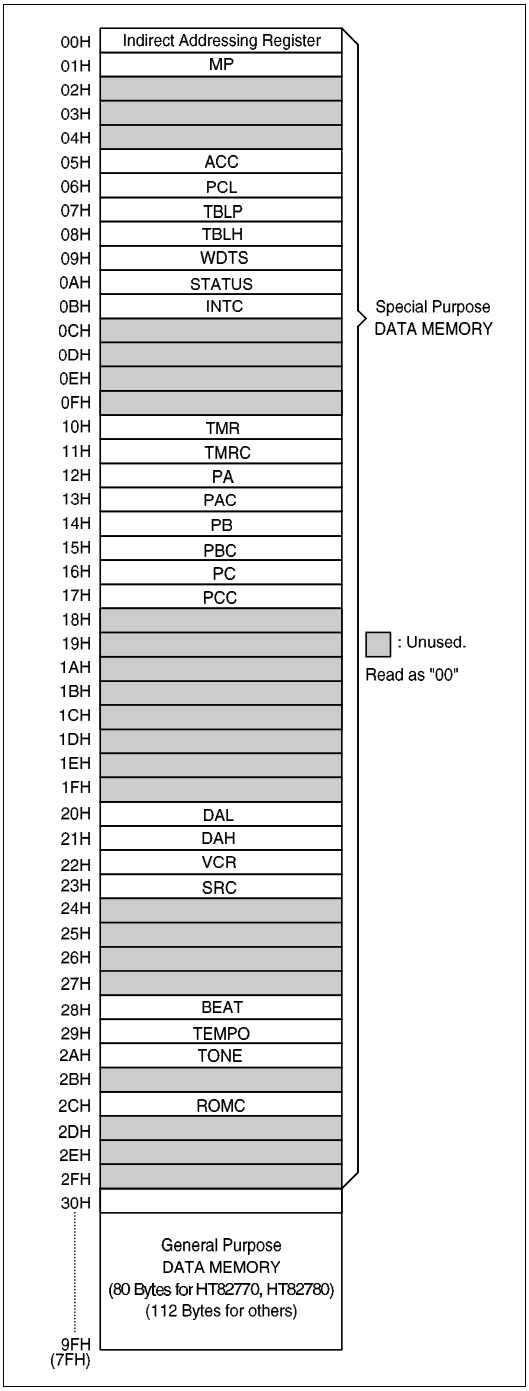
Instruction(s)	Table Location											
	*11	*10	*9	*8	*7	*6	*5	*4	*3	*2	*1	*0
TABRDC [m]	P11	P10	P9	P8	@7	@6	@5	@4	@3	@2	@1	@0
TABRDL [m]	1	1	1	1	@7	@6	@5	@4	@3	@2	@1	@0

Table Location

Notes: *11(*10)~*0: Bits of table location P11(P10)~P8: Bits of current program counter

@7~@0: Bits of table pointer

The bit 11 is omitted for HT82770 and HT82780



RAM Mapping

pointer register (MP; 01H), accumulator (ACC; 05H), table pointer (TBLP; 07H), table higher-order byte register (TBLH; 08H), status register (STATUS; 0AH), interrupt control register (INTC; 0BH), watch dog timer option setting register (WDTS; 09H), I/O registers (PA; 12H, PB; 14H, PC; 16H) and I/O control registers (PAC; 13H, PBC; 15H, PCC; 17H). The 20H to 2FH are used for sound and tone (melody) synthesis. The function registers include a lower-order byte register (DAL; 20H) of D/A data, higher-order byte register (DAH;21H) of D/A data , volume control register (VCR; 22H), sampling rate control register (SRC; 23H), beat control register (BEAT; 28H), tempo control register (TEMPO; 29H), tone control register (TONE; 2AH) and voice ROM control register (ROMC; 2CH). The remaining space before 30H is reserved for future expansion. Reading these remaining locations will get "00H". The general purpose data memory, addressed from 30H to 9FH (7FH), is used for data and control information under instruction commands.

All of the areas of data memory can handle arithmetic, logic, increment, decrement and rotate operations directly. Except some dedicated bits, each bit in the data memory can be set and reset by "SET [m].i" and "CLR [m].i", and can also be indirectly accessed through a memory pointer register (MP; 01H).

Indirect addressing register

Location 00H is an indirect addressing register that is not physically implemented. Any read/write operation of [00H] accesses the data memory pointed to by MP (01H). Indirectly reading location 00H will return the result 00H whereas indirectly writing it will lead to no operations. The memory pointer register MP (01H) is a 7 bit register. Bit 7 of MP is undefined and reading it will return the result "1". Any writing operation to MP will transfer only the lower 7 bit data to MP.

Arithmetic and logic unit - ALU

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

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

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

Status register - STATUS

This 8 bit register (0AH) consists of a zero flag (Z), carry flag (C), auxiliary carry flag (AC), overflow flag (OV), power down flag (PD) and watch dog time-out flag (TO). It also records the status information and controls the operation sequence.

Except the TO and PD flags, bits in the status register can be altered by instructions similar to other registers. Any data written into the status register will not change the TO or PD flag. Operations related to the status register may yield different results from those intended. The TO and PD flags can be altered only by a watch dog timer overflow, chip power-up, clearing the watch dog time or executing the "HALT" instruction.

The Z, OV, AC and C flags generally reflect the statuses of the latest operations.
The status register will not be pushed onto the stack automatically on entering the interrupt sequence or executing the subroutine call. If the status contents are important and the subroutine may corrupt the status register, the programmer must take precautions and save it properly.

Interrupt

The HT827XX series provide an external interrupt in addition to two internal timer/event counter interrupts. The interrupt control register (INTC; 0BH) includes interrupt control bits to set the enable/disable and the interrupt request flags.

Once an interrupt subroutine is serviced, all the other interrupts will be blocked (by clearing the EMI bit). This scheme may prevent any further interrupt nesting. Other interrupt requests may happen during this interval but only the interrupt request flag is recorded. If a

Labels	Bits	Function
C	0	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. It is also affected by a rotate through carry instruction.
AC	1	AC is set if the operation results in a carry out of the low nibbles in addition or if no borrow from the high nibble into the low nibble in subtraction takes place; otherwise AC is cleared.
Z	2	Z is set if the result of an arithmetic or logic operation is zero; otherwise Z is cleared.
OV	3	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.
PD	4	PD is cleared by a system power-up or executing the "CLR WDT" instruction. PD is set by executing the "HALT" instruction.
TO	5	TO is cleared by a system power-up or executing the "CLR WDT" or "HALT" instructions. TO is set by a WDT time-out.
—	6	Undefined, read as "0".
—	7	Undefined, read as "0".

STATUS Register

interrupt needs servicing within the service routine, the programmer may set the EMI bit and the corresponding bit of INTC, allowing interrupt nesting. If the stack is full, the interrupt request will not be acknowledged till the SP is decremented, whether or not the related interrupt is enabled. If immediate service is desired, the stack has to be prevented from being full.

All these interrupts have a wakeup capability. As an interrupt is serviced, a control transfer occurs by pushing the program counter onto the stack and then branching to subroutines at the specified location(s) in the program memory. Only the program counter is pushed onto the stack. The programmer must save the contents of the register or status register (STATUS) in advance if they are altered by an interrupt service program which corrupts the desired control sequence.

External interrupts are triggered by a high to low transition of $\overline{\text{INT}}$. The related interrupt request flag (EIF; bit 4 of INTC) are also set. When an interrupt is enabled, the stack is not full and the external interrupt is active, a subroutine call to location 04H will occur. The interrupt request flag (EIF) and EMI bits will be cleared to disable other interrupts.

The sampling rate counter interrupt is initialized by setting a sampling rate counter interrupt request flag (SRF; bit 5 of INTC), which is caused by a timer overflow. When an interrupt is enabled, the stack is not full and the SRF bit is set, a subroutine call to location 08H will occur. The related interrupt request flag (SRF) will be reset and the EMI bit be cleared to disable further interrupts.

The internal timer/event counter interrupt is initialized by setting a timer/event counter interrupt request flag (TF; bit 6 of INTC), which is caused by a timer overflow. When an interrupt is enabled, the stack is not full and the TF bit is set, a subroutine call to location 0CH will occur. The related interrupt request flag (TF) will be reset and the EMI bit be cleared to disable further interrupts.

During the execution of an interrupt subroutine, other interrupt acknowledgments are all 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 an interrupt subroutine, "RET" or "RETI"

Register	Bit No.	Label	Function
INTC (0BH)	0	EMI	Controls a master (global) interrupt (1=enabled; 0=disabled)
	1	EEI	Controls an external interrupt (1=enabled; 0=disabled)
	2	ESI	Controls a sampling rate counter interrupt (1=enabled; 0=disabled)
	3	ETI	Controls a timer/event counter interrupt (1=enabled; 0=disabled)
	4	EIF	External interrupt request flag (1=active; 0=inactive)
	5	SRF	Sampling rate counter request flag (1=active; 0=inactive)
	6	TF	Internal timer/event counter request flag (1=active; 0=inactive)
	7	–	Unused bit, read as "0".

INTC Register

may be invoked. RETI will set the EMI bit to enable an interrupt service, but RET will not.

Interrupts occurring in an interval between the rising edges of two consecutive T2 pulses will be serviced at the latter of the two T2 pulses if the corresponding interrupts are enabled. In the case of simultaneous requests, they can be masked by resetting the EMI bit. The following table illustrates the priority of applying the simultaneous requests:

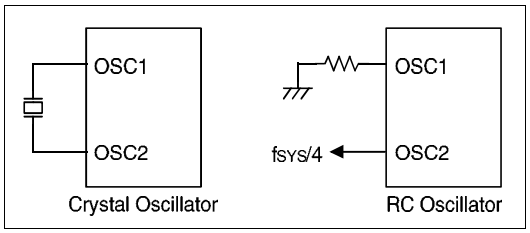
No.	Interrupt Source	Priority	Vector
a	External Interrupt	1	04H
b	Sampling Rate Counter Overflow	2	08H
c	Timer/Event Counter Overflow	3	0CH

The timer/event counter interrupt request flag (TF), external interrupt request flag (EIF), sampling rate counter interrupt request flag (SRF), enable timer/event counter bit (ETI), enable external interrupt bit (EEI), enable sampling rate counter bit (ESI) and enable master interrupt bit (EMI) make up of an interrupt control register (INTC) which is located at 0BH in the data memory. EMI, EEI, ESI and ETI are used to control the enable/disable status of interrupts. These bits prevent the requested interrupt from being serviced. Once the interrupt request flags (TF, SRF, EIF) are all set, they will remain in the INTC register till the interrupts are serviced or cleared by a software instruction.

The "CALL subroutine" had better not to be used within the interrupt subroutine. This is because interrupts often occur in an unpredictable manner or required to be serviced immediately in certain applications. If only one stack is left and enabling the interrupt is not well controlled, operation of the "call" in the interrupt subroutine will damage the original control sequence.

Oscillator configuration

The HT827XX series provide two kinds of oscillator circuits, namely RC and crystal oscillators, for system clocks. Selection of the oscillator circuit is determined by mask option. No matter what kind of oscillator type is chosen, the system clock is provided by the signal. The HALT mode stops the system oscillator and resists in an external signal for conserving power.



System Oscillator

If an RC type of oscillator is used, an external resistor between OSC1 and GND is required and the range of the resistance has to be from 51KΩ to 1MΩ. The system, divided by 4, is available on OSC2, which synchronizes external logic. The RC type of oscillator provides the most cost-effective solution. Nonetheless, the frequency of the oscillation may vary with VDD, temperature and the chip itself due to process variations. It is, therefore, not suitable for timing sensitive operations where an accurate oscillator frequency is demanded.

On the other hand, if a crystal type of oscillator is used instead, a crystal across OSC1 and OSC2 is required, providing feedback and phase shift for the oscillator. No other external components are needed. The resonator can replace the crystal and connects between OSC1 and OSC2 so that a frequency reference can be derived. But two external capacitors in OSC1 and OSC2 are required.

The WDT oscillator is a free running on chip RC oscillator, requiring no external components. The WDT oscillator still works a period of approximately 78 μs even when the system enters the power down mode and the system clock is terminated. It nonetheless can be disabled by mask option for conserving power.

Watch dog timer - WDT

The clock source of WDT is implemented by a dedicated RC oscillator (WDT oscillator) or instruction clock (system clock divided by 4), de-

cided by mask option. The watch dog timer is designed to prevent a software malfunction or sequence jumping to an unknown location with unpredictable results. It can be disabled by mask option. After it is disabled, all executions related to WDT result in no operations.

WDT is first divided by 256 (8 stages) to get a nominal time-out period of approximately 20 ms once an internal WDT oscillator (RC type of oscillator normally with a period of 78μs) is selected. This time-out period may vary with temperature, VDD and process variations. By invoking the WDT prescaler, a longer time-out period can be realized. Writing data to WS2, WS1 and WS0 (bits 2, 1 and 0 of WDTS) can derive different time-out periods. If WS2, WS1 and WS0 all equal to 1, the division ratio is up to 1:128, and the maximum time-out period is 2.6 seconds.

WS2	WS1	WS0	Division Ratio
0	0	0	1:1
0	0	1	1:2
0	1	0	1:4
0	1	1	1:8
1	0	0	1:16
1	0	1	1:32
1	1	0	1:64
1	1	1	1:128

WDTS Register

If the WDT oscillator is disabled, the WDT clock may still come from an instruction clock. It operates in the same manner except that WDT may stops counting and loses its protecting purpose in the HALT state. In this situation the logic can only be re-initialized by external logic. The high nibble and bit 3 of WDTS are reserved for user's defined flags. The programmer may use these flags to indicate some specified statuses.

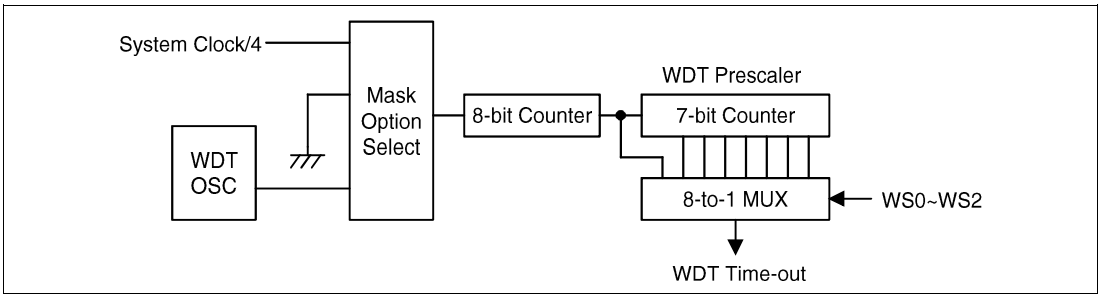
The on-chip RC oscillator (WDT OSC) is strongly recommended if the device operates in a noisy environment, since the HALT function will stop the system clock.

Overflow of the WDT under a normal operation initializes a "chip reset" and sets the status bit "TO". It will initialize a "warm reset", and only PC and SP are reset to zero in the HALT mode. To clear the contents of WDT (including the WDT prescaler), three methods are adopted, namely external reset (a low level to $\overline{RES}$), software instruction(s), and "HALT" instruction. The software instruction(s) include "CLR WDT" and the other sets - "CLR WDT1" and "CLR WDT2". Of these two types of instructions, only one can be active at a time by mask option - "CLR WDT times selection option". If "CLR WDT" is chosen (i.e., CLRWDT times equal one), any execution of the "CLR WDT" instruction will clear WDT. In the case that "CLR WDT1" and "CLR WDT2" are selected (i.e., CLRWDT times equal two), these two instructions must be executed to clear WDT; otherwise WDT may reset the chip as a result of time-out.

Power down operation - HALT

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

- The system oscillator is turned off but the WDT oscillator still keeps running (if the



Watch Dog Timer

WDT oscillator is selected).

- The contents of the on-chip RAM and registers remain unchanged.
- The WDT and WDT prescaler are cleared and re-counted (if the clock of WDT is from the WDT oscillator).
- All the I/O ports maintain their original statuses.
- The PD flag is set and the TO flag cleared.

The system can quit the HALT mode by an external reset, interrupt, external falling edge signal on port A or WDT overflow. An external reset leads to device initialization and a WDT overflow performs a "warm reset". The reason for chip reset can then be determined after examining the TO and PD flags. The PD flag is cleared when the system powers up or executes the "CLR WDT" instruction and set when the "HALT" instruction is executed. The TO flag is set if the WDT time-out occurs, and causes a wake-up that resets only PC and SP. The others maintain their original statuses.

The port A wake-up and interrupt methods can be considered as a continuation of normal execution. Each bit in port A can be independently selected to wake up the device by mask option. Awakening from an I/O port stimulus, the program resumes execution of the next instruction. However, if it is awakening from an interrupt, two sequences may happen. The program will resume execution at the next instruction if the related interrupt(s) is disabled or the interrupt(s) is enabled but the stack is full. Nonetheless, if the interrupt is enabled and the stack is not full, a regular interrupt response takes place.

Once the wake-up event(s) occurs, and the system clock comes from crystal, it takes 1024 t_{SYS} (system clock period) to resume a normal operation. That is to say, the HT827XX series will insert a dummy period after the wake-up. If the system clock, on the other hand, is from an RC type of oscillator, it will continue operation. The actual interrupt subroutine execution will be delayed by one or more cycles if the wake-up results from an interrupt acknowledge. It, on the other hand, will be executed immediately after the dummy period is finished if the wake-up results in the next instruction execution.

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

Reset

There are three ways in which a reset can occur:

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

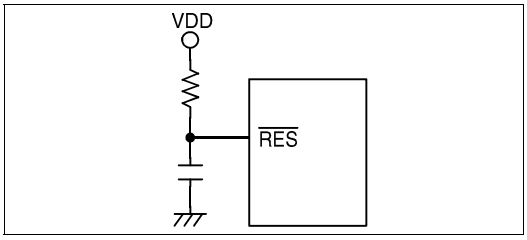
The WDT time-out during HALT is different from other chip reset conditions, since it can perform a "warm reset" that resets only PC and SP, leaving the other circuits remain in their original states. Some registers will remain unchanged during reset conditions. Most registers are reset to the "initial condition" once the reset conditions are met. The program can distinguish between different "chip resets" by examining the PD flag and TO flag.

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

Note: "u" means "unchanged".

To guarantee that the crystal oscillator is started and stabilized, XST (Crystal Start-up Timer) provides an extra-delay by an OSC mask option. The extra-delay delays 1024 system clock pulses when the system awakes from the HALT state or from system power-up and the $\overline{\text{RES}}$ transforms low to high. XST is automatically selected if the crystal oscillator is invoked. It, on the other hand, is disabled when the RC oscillator is chosen. The XST delay is added after XST is chosen and awakening from the HALT state or the system powers up.

The reset duration comes only from $\overline{\text{RES}}$ if an RC oscillator is selected. An extra delay, on the other hand, is added during the power-up period and any wakeup from HALT only if a crystal oscillator is chosen instead.



Reset Circuit

The HT827XX series provide another useful feature for purposes of testing and synchronization. Releasing $\overline{RES}$ high will start execution if $\overline{RES}$ keeps low long enough.

PC	000H
Interrupt	Disabled
Prescaler	Cleared
WDT	Cleared. After a master reset, WDT begins counting.
Timer/Event Counter	Off
Input/Output Ports	Input mode
SP	Point to the top of the stack.

The states of the registers are summarized in the following table:

Register	Reset (Power On)	WDT Time-out (Normal Operation)	$\overline{RES}$ Reset (Normal Operation)	$\overline{RES}$ Reset (HALT)	WDT Time-out (HALT)
TMR	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TMRC	00-0 1—-	00-0 1—-	00-0 1—-	00-0 1—-	uu-u u—-
PC	000H	000H	000H	000H	000H*
MP	-xxx xxxx	-uuu uuuu	-uuu uuuu	-uuu uuuu	-uuu uuuu
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLP	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLH	—xx xxxx	—uu uuuu	—uu uuuu	—uu uuuu	—uu uuuu
STATUS	—00 xxxx	—1u uuuu	—uu uuuu	—01 uuuu	—11 uuuu
INTC	—00 -000	—00 -000	—00 -000	—00 -000	—uu -uuu
WDTS	0000 0111	0000 0111	0000 0111	0000 0111	uuuu uuuu
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	—11 1111	—11 1111	—11 1111	—11 1111	—uu uuuu
PCC	—11 1111	—11 1111	—11 1111	—11 1111	—uu uuuu

Notes: “*” means “warm reset”.

“u” means “unchanged”.

“x” means “unknown”.

Sampling rate counter

The HT827XX series offer a sampling rate counter. This counter contains a 5 bit programmable count-up counter. The clock may come from 128KHz or 2KHz by code option.

When the 128KHz clock is selected, 26 kinds of sampling rate are provided for a voice synthesizer. Following is a table of the 26 kinds of sampling rates:

Code	Freq.	Code	Freq.
xx00 0000	3.50KHz	xx00 1101	5.89KHz
xx00 0001	3.61KHz	xx00 1110	6.21KHz
xx00 0010	3.72KHz	xx00 1111	6.58KHz
xx00 0011	3.86KHz	xx01 0000	6.99KHz
xx00 0100	3.99KHz	xx01 0001	7.46KHz
xx00 0101	4.14KHz	xx01 0010	7.99KHz
xx00 0110	4.30KHz	xx01 0011	8.61KHz
xx00 0111	4.48KHz	xx01 0100	9.32KHz
xx00 1000	4.66KHz	xx01 0101	10.17KHz
xx00 1001	4.86KHz	xx01 0110	11.19KHz
xx00 1010	5.08KHz	xx01 0111	12.43KHz
xx00 1011	5.33KHz	xx01 1000	13.98KHz
xx00 1100	5.59KHz	xx01 1001	15.98KHz

Sampling Rate Table

Note: "XX" means don't care.

On the other hand, when the 2KHz clock is chosen, 32 kinds of time periods of the envelope decay is offered. Following is a table of the envelope decay:

Code	Freq.	Code	Freq.
xx10 0000	54.6Hz	xx11 0000	109.3Hz
xx10 0001	56.4Hz	xx11 0001	116.5Hz
xx10 0010	58.3Hz	xx11 0010	124.9Hz
xx10 0011	60.3Hz	xx11 0011	134.5Hz
xx10 0100	62.4Hz	xx11 0100	145.7Hz
xx10 0101	64.7Hz	xx11 0101	158.9Hz
xx10 0110	67.2Hz	xx11 0110	174.8Hz
xx10 0111	69.9Hz	xx11 0111	194.2Hz
xx10 1000	72.8Hz	xx11 1000	218.5Hz
xx10 1001	76.0Hz	xx11 1001	249.7Hz
xx10 1010	79.5Hz	xx11 1010	291.3Hz
xx10 1011	83.2Hz	xx11 1011	349.6Hz
xx10 1100	87.4Hz	xx11 1100	437.0Hz
xx10 1101	92.0Hz	xx11 1101	582.7Hz
xx10 1110	97.1Hz	xx11 1110	874.0Hz
xx10 1111	102.8Hz	xx11 1111	1.75KHz

Envelope Decay Table

Note: "xx" means don't care.

One of the relative counter values is preloaded to the sampling rate counter after a code is written to the counter (SRC;23H). Once the sampling rate counter starts counting, it will count from its current contents to 1FH. The counter is reloaded from the sampling rate counter preload register, and generates an interrupt request flag (SRF; bit 5 of INTC) if overflow of the divide-by-1 counter occurs.

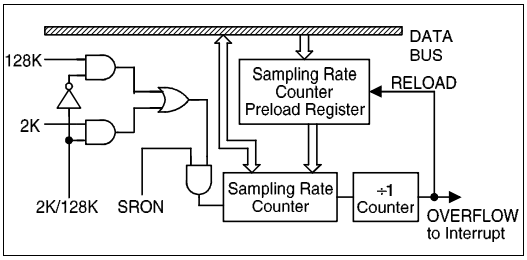
To enable a counting operation, the ON bit (SRON; bit 7 of TEMPO) of the counter should be set to 1. Overflow of the sampling rate counter is one of the wake-up sources. Writing a "0" to ESI will disable the interrupt service.

Writing data to the sampling rate preload register will also reload the data to the sampling rate counter in the case of 1F condition of the sampling rate counter. Data written to the sampling rate counter will, on the other hand, be kept only in the counter preload register if the counter is turned on. The sampling rate counter still goes on working till an overflow of the divide-by-1 counter occurs.

The clock is blocked to avoid errors once the sampling rate counter is read. The programmer should take the counting error into account since blocking of the clock may result in a counting error.

Labels	Bits	Function
SR0~SR4	0~4	To define an voice sampling rate or envelope decaying time
2K/128K	5	To define an input clock source (0=128K; 1=2K)
—	6~7	Unused bits, read as "0".

SRC Register



Sampling Rate Counter

Timer/Event counter

The HT827XX series provide a timer/event counter. This counter contains an 8 bit programmable count-up counter. The clock may come from an external source or the system clock divided by 4. Only one reference time-base is available when an internal instruction clock is selected. The external clock input allows the user to count external events, measure time intervals or pulse widths, or generate an accurate time base.

There are two registers related to the timer/event counter, namely TMR ([10H]) and TMRC ([11H]). Two physical registers are mapped to the TMR location. Writing TMR makes the starting value put in the timer/event counter preload register and reading it gets the contents of the timer/event counter. TMRC is a timer/event counter control register which defines some options.

The TM0 and TM1 bits define the operation mode. The event counting mode counts external events, indicating that the source of the clock is from an external (TMR) pin. The timer mode functions as a normal timer with the clock source coming from an instruction clock. The pulse width measurement mode can be used to count a high to low level duration of an external signal (TMR). This counting is based on the instruction clock.

In the event counting or timer mode, after the timer/event counter starts counting, it will count from its current contents to FFH. Once an overflow occurs, the counter is reloaded from the timer/event counter preload register and generates an interrupt request flag (TF; bit 6 of INTC).

In the pulse width measurement mode with the TON and TE bits equal to one, after TMR receives a transient from low to high (or high to low when the TE bit is "0"), it will start counting till it returns to the original level and resets the TON. The measured result still remains in the timer/event counter even when the activated transient re-occurs. In other words, only one cycle can be measured till TON is set. The cycle measurement will go on functioning as long as further transient pulses are received. In this operation mode, the timer/event counter starts counting not according to the logic level but to the transient edges. In the case of a counter overflow, the counter is reloaded from the timer/event counter preload register and issues an interrupt request, like the other two modes.

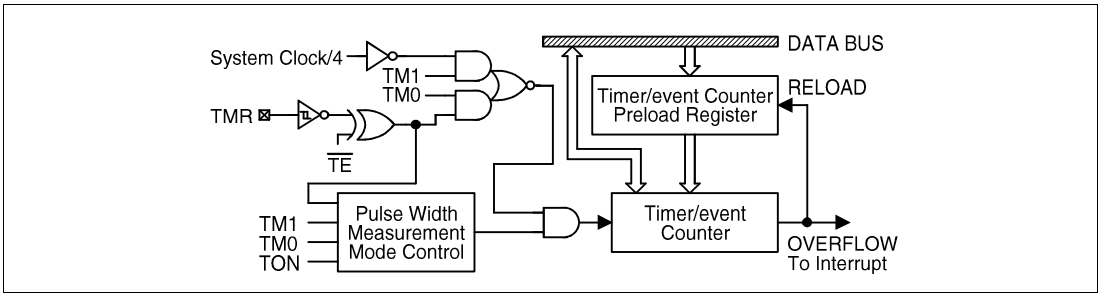
The timer ON bit (TON; bit 4 of TMRC) should be set to 1 to enable a counting operation. In the pulse width measurement mode, TON will be cleared automatically after the measurement cycle is completed. In the other two modes, it TON can be reset only by instructions. The overflow of the timer/event counter is one of the wake-up sources. Writing a "0" to ETI will disable the interrupt service no matter what kind of operation mode is chosen.

In the case of the timer/event counter OFF condition, writing data to the timer/event counter preload register will also reload it to the timer/event counter. Data written to the timer/event counter will however be kept in the timer/event counter preload register if the timer/event counter is turned on. The timer/event counter will be still operating till an overflow occurs.

The clock will be blocked to avoid errors after the timer/event counter (reading TMR) is read. The programmer should take the counting error into account since clock blocking may result in a counting error.

Labels (TMRC)	Bits	Function
—	0-2	Unused bits, read as “0”.
TE	3	To define the TMR active edge of a timer/event counter (0=active on low to high; 1=active on high to low)
TON	4	To enable/disable timer counting (0=disabled; 1=enabled)
—	5	Unused bits, read as “0”.
TM0 TM1	6 7	To define the operation mode 01=Event count mode (External clock) 10=Timer mode (Internal clock) 11=Pulse width measurement mode 00=Unused

TMRC Register



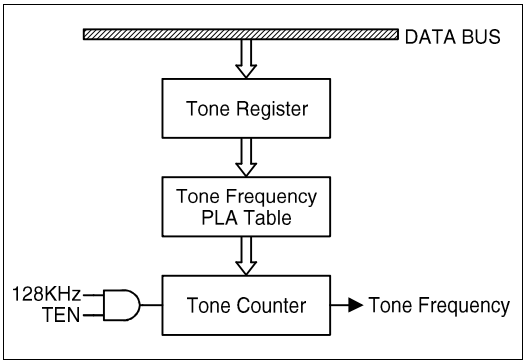
Timer/Event Counter

Tone and melody generator

The HT827XX series provide a tone frequency register (TONE; 2AH), beat frequency register (BEAT; 28H) as well as tempo frequency register (TEMPO; 29H) for generating melody and sound effects.

The LSI can generate four octaves, labeled from C2# to C6. Desired frequencies can be obtained by first writing the related data into a tone frequency register (TONE; 2AH) and then enabling the tone counter. A Tone frequence is generated and remained if the tone counter overflows.

Labels	Bits	Function
TN0~ TN3	0~3	To define the tone frequency (refer to the tone frequency table)
OCT0 OCT1	4 5	To define the 4 octave tone frequencies (refer to the tone frequency table)
—	6	Unused bit, read as “0”.
TEN	7	To enable/disable the tone counter (0=disabled; 1=enabled)



TONE Counter

The BEAT register counts melody beats. Bit 7 (BTO) of the BEAT register is set when the beat counter overflows. No interrupt is generated if the beat counter overflows. So bit 7 (BTO) of the BEAT register must be polled to generate correct beat frequencies. After reading the BTO status, the bit 7 should be cleared by the programmer to avoid the malfunction of the next polling.

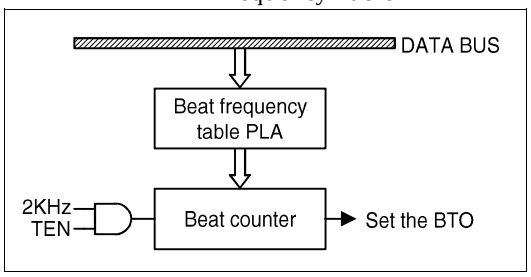
Labels	Bits	Function
B0~B6	0~6	To define the beat frequency (refer to the beat frequency table)
BTO	7	BTO is set when the beat counter is time-out.

BEAT Register

Code	Beat
1xxx xxxx	Beat time-out
0000 0000	1/24 BEAT
0000 0010	1/8 BEAT
0000 0011	1/6 BEAT
0000 0101	1/4 BEAT
0000 0111	1/3 BEAT
0000 1011	1/2 BEAT
0000 1111	2/3 BEAT
0001 0001	3/4 BEAT
0001 0111	1 BEAT
0010 0011	3/2 BEAT
0010 1111	2 BEATS
0100 0100	3 BEATS
0101 1111	4 BEATS
0111 0111	5 BEATS
Other codes	—

Note: "—" means unknown beats.

BEAT Frequency Table



BEAT Counter

Code	Frequency	Tone	Code	Frequency	Tone
1x00 0000	—	—	1x10 0000	—	—
1x00 0001	138.5Hz	C2#	1x10 0001	553.8Hz	C4#
1x00 0010	146.4Hz	D2	1x10 0010	585.7Hz	D4
1x00 0011	155.4Hz	D2#	1x10 0011	621.5Hz	D4#
1x00 0100	164.5Hz	E2	1x10 0100	658.1Hz	E4
1x00 0101	174.8Hz	F2	1x10 0101	699.2Hz	F4
1x00 0110	185.2Hz	F2#	1x10 0110	740.9Hz	F4#
1x00 0111	195.6Hz	G2	1x10 0111	782.3Hz	G4
1x00 1000	207.2Hz	G2#	1x10 1000	828.7Hz	G4#
1x00 1001	220.2Hz	A2	1x10 1001	880.9Hz	A4
1x00 1010	233.1Hz	A2#	1x10 1010	932.3Hz	A4#
1x00 1011	247.5Hz	B2	1x10 1011	990.0Hz	B4
1x00 1100	261.4Hz	C3	1x10 1100	1045.6Hz	C5
1x00 1101	—	—	1x10 1101	—	—
1x00 1110	—	—	1x10 1110	—	—
1x00 1111	—	—	1x10 1111	—	—
1x01 0000	—	—	1x11 0000	—	—
1x01 0001	279.6Hz	C3#	1x11 0001	1107.7Hz	C5#
1x01 0010	292.9Hz	D3	1x11 0010	1171.5Hz	D5
1x01 0011	310.8Hz	D3#	1x11 0011	1243.1Hz	D5#
1x01 0100	329.0Hz	E3	1x11 0100	1316.2Hz	E5
1x01 0101	349.6Hz	F3	1x11 0101	1398.4Hz	F5
1x01 0110	370.4Hz	F3#	1x11 0110	1481.8Hz	F5#
1x01 0111	391.2Hz	G3	1x11 0111	1564.7Hz	G5
1x01 1000	414.4Hz	G3#	1x11 1000	1657.4Hz	G5#
1x01 1001	440.5Hz	A3	1x11 1001	1761.8Hz	A5
1x01 1010	466.1Hz	A3#	1x11 1010	1864.6Hz	A5#
1x01 1011	495.0Hz	B3	1x11 1011	1980.1Hz	B5
1x01 1100	522.8Hz	C4	1x11 1100	2091.1Hz	C6
1x01 1101	—	—	1x11 1101	—	—
1x01 1110	—	—	1x11 1110	—	—
1x01 1111	—	—	1x11 1111	—	—

TONE Frequency Table

Note: “x” means don’t care. “—” means invalid.

The TEMPO register counts melodies. A tempo frequency is generated after tempo data are loaded and the TEMPO counter is enabled also. The tempo decides the beat time period.

Labels	Bits	Function
TN0~ TN3	0~3	To define the tempo frequency (refer to the tempo frequency table)
—	4,5	Unused bits, read as "0".
TMPEN	6	To enable/disable the tempo counter (0=disabled; 1=enabled)
SRON	7	To enable/disable the D/A output, sampling rate counter and counter ROM (0=disabled; 1=enabled)

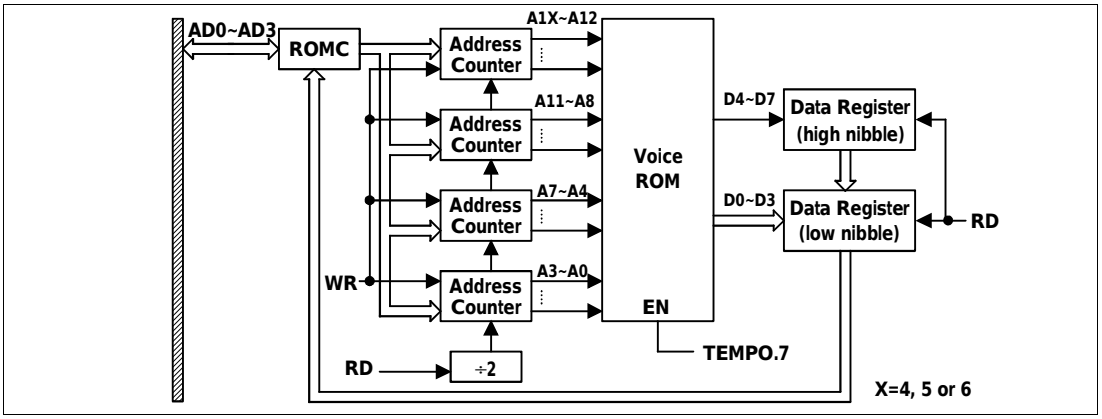
Voice ROM

The HT827XX series include a ROM for storing sound and tone (melody) data. Coded data can be saved in an internal mask ROM by changing one layer of the mask after the sound and tone (melody) sources are coded by Holtek's tools. The voice ROM size for HT827XX series are shown as following.

Item	Voice ROM Size
HT82700	128K × 8
HT82720	96K × 8
HT82740	64K × 8
HT82770	64K × 8
HT82780	32K × 8

The handshaking between the microcontroller and voice ROM is through a ROM control register (ROMC; 2CH). To enable the voice ROM, the bit 7 of the TEMPO register should be set as "1". The related ROM address has to be saved in the ROM control register first if the microcontroller attempts to read the sound or tone (melody) data in the mask ROM. The ROM is comprised by a set of address counters internally. After the microcontroller finishes reading a byte of data, its internal address counter will automatically be increased by one. In this case, reading continuous data only requires loading the starting address to the ROM control register.

Code	TEMPO CLK	TEMPO BPM
0000	30.5Hz	68.3
0001	32.55Hz	72.8
0010	34.88Hz	78.0
0011	37.56Hz	84.0
0100	40.69Hz	91.0
0101	44.39Hz	99.3
0110	48.83Hz	109.3



Voice ROM

Code	TEMPO CLK	TEMPO BPM
0111	54.25Hz	121.4
1000	61Hz	136.6
1001	65.1Hz	145.7
1010	69.8Hz	156.1
1011	75.12Hz	168.1
1100	81.38Hz	182.1
1101	88.78Hz	198.6
1110	97.66Hz	218.5
1111	108.5Hz	242.8

TEMPO Frequency Table

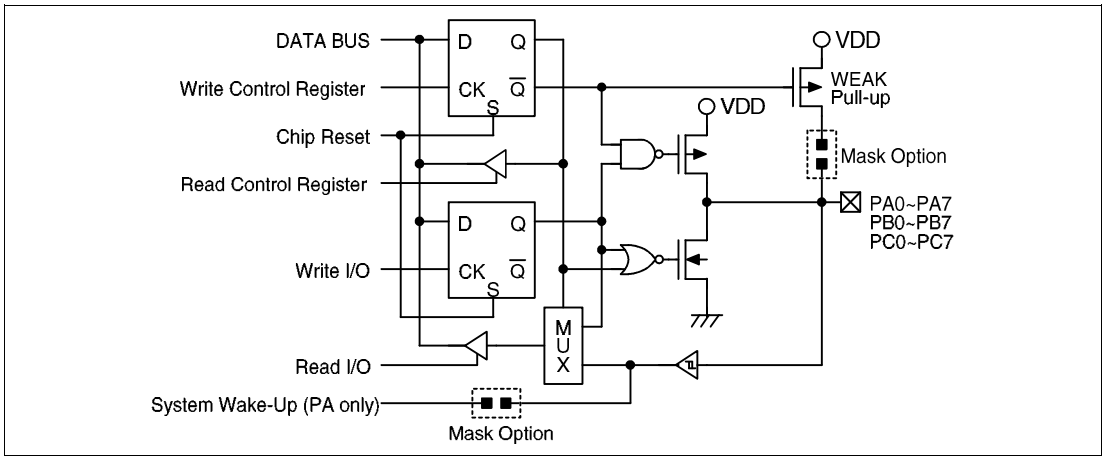
The lower-order nibble is valid whereas the higher-order nibble invalid of the ROM control register. Based on this difference, the start address has to be divided into four nibbles, and written into the ROM control register four times with respect to the divided four nibbles. The lower-order nibble and higher-order nibble data can then be read back by two time's reading after sound or tone (melody) starting address is written. For example, if the starting address of the sound data to be read is 0CF0H, the program of reading one byte of sound or tone (melody) data is as follows:

Read-New-Data:

```
SET       TEMPO.7
MOV A, 00H
MOV [ROMC],A;     Write the first nibble address
MOV A,0FH
MOV[ROMC],A;      Write the second nibble address
MOV A,0CH
MOV [ROMC],A;     Write the third nibble address
MOV A,00H
MOV [ROMC],A;     Write the fourth nibble address
MOV A,[ROMC];     Read the lower-order nibble data
MOV [DATA],A;
MOV A,[ROMC];     Read the high-order nibble data
SWAPA [ACC];
ORM A,[DATA];     Combine the lower-order data and higher-order data
```

Input/Output ports

The HT827XX series include 24 bidirectional input/output lines, labeled from PA to PC, which are mapped to the data memories of



Input/Output Ports

[12H], [14H] and [16H], respectively. All of these I/O ports can be used as input and output operations. For input operation, these ports are non-latched, i.e., the inputs must be ready at the T2 rising edge of the instruction "MOV A,[m]" (m=12, 14 or 16H). For output operation, all the data are latched and remain unchanged till the output latch is re-written.

Each I/O line has its own control register (PAC, PBC and PCC) to control the input/output configuration. With a control register, a CMOS output or schmitt trigger input can be re-configured dynamically (i.e., on-the-fly)with or without pull-high resistor structures under a software control. To function as an input, the corresponding latch of a control register must write "1". The pull-high resistance will be automatically exhibited if the pull-high option is chosen. The input source(s) also depends on the control register. If the bit of the control register bit "1", the input will read the pad state. If it is "0", the contents of the latches will move to the internal bus. The latter is possible only in the "read-modify-write" instruction. For the output function, CMOS is the only configuration. These control registers are all mapped to locations 13H, 15H and 17H.

These input/output lines stay at a high level or floating (decided by mask option) after a chip reset. Each bit of the input/output latches can be set or cleared by the "SET[m].i" and "CLR[m].i" (m=12H, 14H or 16H) instructions.

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

Each line of port A is capable of waking up the device. The highest two bits of port C are not physically implemented. A "0" will return on reading the highest two bits, but writing them will result in no operations.

Mask option

The following table illustrates 5 kinds of mask option in the HT827XX series. All of them have to be defined to ensure a proper system function.

No.	Mask Option
1	OSC type selection This option decides the selection of a system clock to be an RC or crystal type of oscillator.
2	WDT source selection Three selections are provided, namely on-chip RC oscillator, instruction clock and WDT disable.
3	CLRWDT times selection This option defines the way of clearing WDT by instructions. "Once" means "CLR WDT" can clear WDT. "Twice" means WDT can be cleared only if both "CLR WDT1" and "CLR WDT2" are executed.
4	Wake-up selection This option defines the activity of the wake-up function. All of the external I/O pins (PA only) are capable of waking up the chip from a HALT mode.
5	Pull-high selection This option decides whether or not the pull-high resistance exists in the input mode of the I/O ports. Each bit of the I/O port can be independently selected.

Instruction Set

Instruction set summary

Mnemonic	Description	Flag Affected
Arithmetic		
ADD A,[m]	Add data memory to ACC	Z,C,AC,OV
ADDM A,[m]	Add ACC to data memory	Z,C,AC,OV
ADD A,x	Add immediate data to ACC	Z,C,AC,OV
ADC A,[m]	Add data memory to ACC with carry	Z,C,AC,OV
ADCM A,[m]	Add ACC to register with carry	Z,C,AC,OV
SUB A,x	Subtract immediate data from ACC	Z,C,AC,OV
SUB A,[m]	Subtract data memory from ACC	Z,C,AC,OV
SUBM A,[m]	Subtract data memory from ACC with result in data memory	Z,C,AC,OV
SBC A,[m]	Subtract data memory from ACC with carry	Z,C,AC,OV
SBCM A,[m]	Subtract data memory from ACC with carry leaving result in data memory	Z,C,AC,OV
DAA [m]	Decimal adjust ACC for addition with result in data memory	C
Logic Operation		
AND A,[m]	AND data memory to ACC	Z
OR A,[m]	OR data memory to ACC	Z
XOR A,[m]	Exclusive-OR data memory to ACC	Z
ANDM A,[m]	AND ACC to data memory	Z
ORM A,[m]	OR ACC to data memory	Z
XORM A,[m]	Exclusive-OR ACC to data memory	Z
AND A,x	AND immediate data to ACC	Z
OR A,x	OR immediate data to ACC	Z
XOR A,x	Exclusive-OR immediate data to ACC	Z
CPL [m]	Complement data memory	Z
CPLA [m]	Complement data memory with result in ACC	Z
Increment & Decrement		
INCA [m]	Increment data memory with result in ACC	Z
INC [m]	Increment data memory	Z
DECA [m]	Decrement data memory with result in ACC	Z
DEC [m]	Decrement data memory	Z

Mnemonic	Description	Flag Affected
Rotate		
RRA [m]	Rotate data memory right with result in ACC	None
RR [m]	Rotate data memory right	None
RRCA [m]	Rotate data memory right through carry with result in ACC	C
RRC [m]	Rotate data memory right through carry	C
RLA [m]	Rotate data memory left with result in ACC	None
RL [m]	Rotate data memory left	None
RLCA [m]	Rotate data memory left through carry with result in ACC	C
RLC [m]	Rotate data memory left through carry	C
Data Move		
MOV A,[m]	Move data memory to ACC	None**
MOV [m],A	Move ACC to data memory	None
MOV A,x	Move immediate data to ACC	None
Bit Operation		
CLR [m].i	Clear bit of data memory	None
SET [m].i	Set bit of data memory	None
Branch		
JMP addr	Jump unconditionally	None
SZ [m]	Skip if data memory is zero	None
SZA [m]	Skip if data memory is zero with data movement to ACC	None
SZ [m].i	Skip if bit i of data memory is zero	None
SNZ [m].i	Skip if bit i of data memory is not zero	None
SIZ [m]	Skip if increment data memory is zero	None
SDZ [m]	Skip if decrement data memory is zero	None
SIZA [m]	Skip if increment data memory is zero with result in ACC	None
SDZA [m]	Skip if decrement data memory is zero with result in ACC	None
CALL addr	Subroutine call	None
RET	Return from subroutine	None
RET A,x	Return from subroutine and load immediate data to ACC	None
RETI	Return from interrupt	None
Table Read		
TABRDC [m]	Read ROM code (current page) to data memory and TBLH	None
TABRDL [m]	Read ROM code (last page) to data memory and TBLH	None

Mnemonic	Description	Flag Affected
Miscellaneous		
NOP	No operation	None
CLR [m]	Clear data memory	None
SET [m]	Set data memory	None
CLR WDT	Clear Watchdog timer	TO,PD
CLR WDT1	Pre-clear Watchdog timer	TO*,PD*
CLR WDT2	Pre-clear Watchdog timer	TO*,PD*
SWAP [m]	Swap nibbles of data memory	None
SWAPA [m]	Swap nibbles of data memory with result in ACC	None
HALT	Enter power down mode	TO,PD

Notes:

x = 8 bit immediate data

m = 7 bit data memory address

A = accumulator

i = 0...7 number of bits

addr = 11 bit program memory address

√ = Flag(s) is affected

– = Flag(s) is unaffected

* = Flag(s) may be affected by the execution status.

** = For the old version of the E.V. chip, the zero flag (Z) can be affected by executing the MOV A,[M] instruction.
For the new version of the E.V. chip, the zero flag will not be altered after executing the MOV A,[M] instruction.

Instruction Definition

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

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

Operation ACC ← ACC+[m]+C

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	√	√	√	√

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

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

Operation [m] ← ACC+[m]+C

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	√	√	√	√

ADD A,[m] Add data memory to accumulator

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

Operation ACC ← ACC+[m]

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	√	√	√	√

ADD A,x Add immediate data to accumulator

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

Operation ACC ← ACC+x

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	√	√	√	√

ADDM A,[m] Add accumulator to data memory

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

Operation [m] ← ACC+[m]

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	√	√	√	√

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

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

Operation ACC ← ACC "AND" [m]

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	√	–	–

AND A,x Logical AND immediate data to accumulator

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

Operation ACC ← ACC "AND" x

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	√	–	–

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

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

Operation [m] ← ACC "AND" [m]

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	√	–	–

CALL addr Subroutine call

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

Operation Stack ← PC+1
PC ← addr

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

CLR [m] Clear data memory

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

Operation [m] ← 00H

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

CLR [m].i Clear bit of data memory

Description Bit i of the specified data memory is cleared to zero.

Operation [m].i ← 0

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

CLR WDT Clear watch dog timer

Description The WDT and WDT Prescaler are cleared (re-count from zero). The power down bit (PD) and time-out bit (TO) are both cleared.

Operation WDT & WDT Prescaler ← 00H
PD & TO ← 0

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	0	0	–	–	–	–

CLR WDT1 Preclear watch dog timer

Description The PD, TO flags, WDT and the WDT Prescaler are all cleared (re-count from zero) if the other preclear WDT instruction has been executed. Execution only of this instruction without the other preclear instruction sets the indicated flag, which implies that this instruction is executed and the PD and TO flags remain unchanged.

Operation WDT & WDT Prescaler ← 00H*
PD & TO ← 0*

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	0*	0*	–	–	–	–

CLR WDT2 Preclear watch dog timer

Description The PD, TO flags, WDT and the WDT Prescaler are all cleared (re-count from zero) if the other preclear WDT instruction has been executed. Execution only of this instruction without the other preclear instruction sets the indicated flag, which implies that this instruction is executed and the PD and TO flags remain unchanged.

Operation WDT & WDT Prescaler ← 00H*
PD & TO ← 0*

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	0*	0*	–	–	–	–

CPL [m] Complement data memory

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

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

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	√	–	–

CPLA [m] Complement data memory-place result in accumulator

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

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

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	√	–	–

DAA [m] Decimal-Adjust accumulator for addition

Description The value of the accumulator is adjusted to a BCD (Binary Code Decimal) code. If bits 0~3 of the accumulator are greater than 9 or AC is one, six is added to the low-order nibble of the accumulator, deriving a BCD digit in the low-order nibble. Similarly, if bits 4~7 of the accumulator are greater than 9 or C is one, six is added to the high-order nibble of the accumulator, generating a BCD digit in the high-order nibble. The result is stored in the data memory.

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

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	√

DEC [m] Decrement data memory

Description Data in the specified data memory are decremented by one.

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

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	√	–	–

DECA [m] Decrement data memory-place result in accumulator

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

Operation ACC ← [m]–1

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	√	–	–

HALT Enter power down mode

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

Operation PC ← PC+1
PD ← 1
TO ← 0

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	0	1	–	–	–	–

INC [m] Increment data memory

Description Data in the specified data memory are incremented by one.

Operation [m] ← [m]+1

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	√	–	–

INCA [m] Increment data memory-place result in accumulator

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

Operation ACC ← [m]+1

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	√	–	–

JMP addr Direct Jump

Description Bits 0~10 of the program counter are replaced with the directly–specified addresses unconditionally, and the control is passed to this destination.

Operation PC ← addr

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

MOV A,[m] Move data memory to accumulator

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

Operation ACC ← [m]

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	**	–	–

MOV A,x Move immediate data to accumulator

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

Operation ACC ← x

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

MOV [m],A Move accumulator to data memory

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

Operation [m] ← ACC

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

NOP No operation

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

Operation PC ← PC+1

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

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

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

Operation ACC ← ACC "OR" [m]

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	√	–	–

OR A,x Logical OR immediate data to accumulator

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

Operation ACC ← ACC "OR" x

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	√	–	–

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

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

Operation [m] ← ACC "OR" [m]

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	√	–	–

RET Return from subroutine

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

Operation PC ← Stack

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

RET A,x Return and place immediate data in accumulator

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

Operation PC ← Stack
ACC ← x

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

RETI Return from interrupt

Description The program counter is restored from the stack, and interrupts enabled by setting the EMI bit. EMI is an enable master(global) interrupt bit (bit 0; register INTC).

Operation PC ← Stack
EMI ← 1

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

RL [m] Rotate data memory left

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

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

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

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

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

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

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

RLC [m] Rotate data memory left through carry

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

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

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	√

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

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

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

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	√

RR [m] Rotate data memory right

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

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

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

RRA [m] Rotate right-place result in accumulator

Description Data in the specified data memory are rotated one bit right with bit 0 rotated to bit 7, leaving the rotation result in the accumulator. The contents of the data memory remain unchanged.

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

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

RRC [m] Rotate data memory right through carry

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

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

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	√

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

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

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

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	√

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

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

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

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	√	√	√	√

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

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

Operation [m] ←ACC+[$\overline{m}$]+C

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	√	√	√	√

SDZ [m] Skip if decrement data memory is zero

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

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

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

SDZA [m] Decrement data memory-place result in ACC, skip if zero

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

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

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

SET [m] Set data memory

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

Operation [m] ← FFH

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

SET [m].i Set bit of data memory

Description Bit “i” of the specified data memory is set to one.

Operation [m].i ← 1

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

SIZ [m] Skip if increment data memory is zero

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

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

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

SIZA [m] Increment data memory-place result in ACC, skip if zero

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

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

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

SNZ [m].i Skip if bit “i” of the data memory is not zero

Description If bit “i” of the specified data memory is not zero, the following instruction, fetched during the current instruction execution, is discarded and a dummy cycle is replaced to get a proper instruction. This is a 2 cycle instruction. Otherwise proceed with the next instruction.

Operation Skip if [m].i≠0

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

SUB A,[m] Subtract data memory from accumulator

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

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

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	√	√	√	√

SUBM A,[m] Subtract data memory from accumulator

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

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

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	√	√	√	√

SUB A,x Subtract immediate data from accumulator

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

Operation ACC ← ACC+$\overline{x}$+1

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	√	√	√	√

SWAP [m] Swap nibbles within the data memory

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

Operation [m].3~[m].0 ↔ [m].7~[m].4

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

SWAPA [m] Swap data memory-place result in accumulator

description The low-order and high-order nibbles of the specified data memory are interchanged, writing the result to the accumulator. The contents of the data memory remain unchanged.

Operation ACC.3~ACC.0 ← [m].7~[m].4
ACC.7~ACC.4 ← [m].3~[m].0

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

SZ [m] Skip if data memory is zero

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

Operation Skip if [m]=0

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

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

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

Operation Skip if [m]=0

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

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

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

Operation Skip if [m].i=0

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

TABRDC [m] Move ROM code (current page) to TBLH & data memory

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

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

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

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

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

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

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

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

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

Operation ACC ← ACC "XOR" [m]

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	√	–	–

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

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

Operation [m] ← ACC "XOR" [m]

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	√	–	–

XOR A,x Logical XOR immediate data to accumulator

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

Operation ACC ← ACC "XOR" x

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	√	–	–