Technical Document

- Tools Information
- FAQs
- Application Note

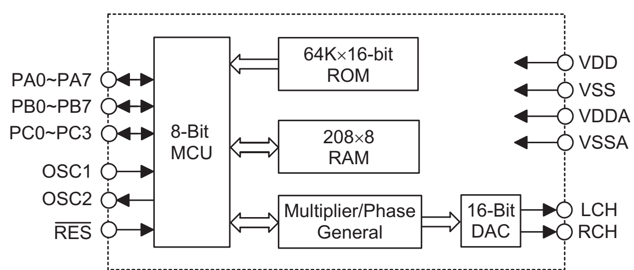
Features

- Operating voltage: 2.4V~5.0V
- Operating frequency:
  X′tal: 6MHz~8MHz
  R_{OSC}: typ. 6MHz
- Built-in 64K×16-bit (1M-bit) ROM for program/data shared
- Built-in 8 bit MCU with 208×8 bits RAM
- Two 8 bit programmable timer with 8 stage prescaler
- 20 bidirectional I/O lines
- Four polyphonic synthesizer
- Stereo 16-bit DAC
- Oscillation modes: XTAL/RCOSC
- Low voltage reset
- Eight-level subroutine nesting
- Watchdog timer
- Supports 8-bit table read instruction (TBLP)
- HALT function and wake-up feature reduce power consumption
- Bit manipulation instructions
- 63 powerful instructions
- All instructions in 1 or 2 machine cycles
- 20/32-pin SOP package

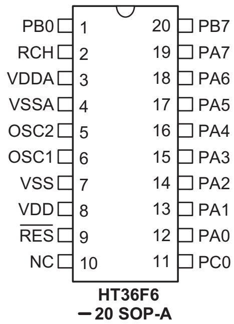
General Description

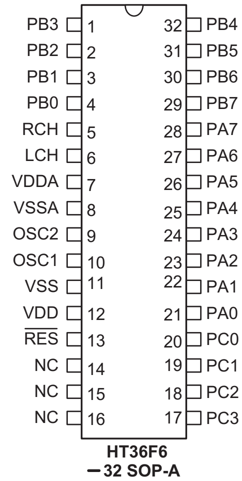
The HT36F6 is an 8-bit high performance RISC architecture microcontroller specifically designed for various music applications. It provides an 8-bit MCU and a 4-channel Wavetable synthesizer. It has a built-in 8-bit microprocessor which controls the synthesizer to generate the melody by setting the special register. A HALT feature is provided to reduce power consumption.

Block Diagram

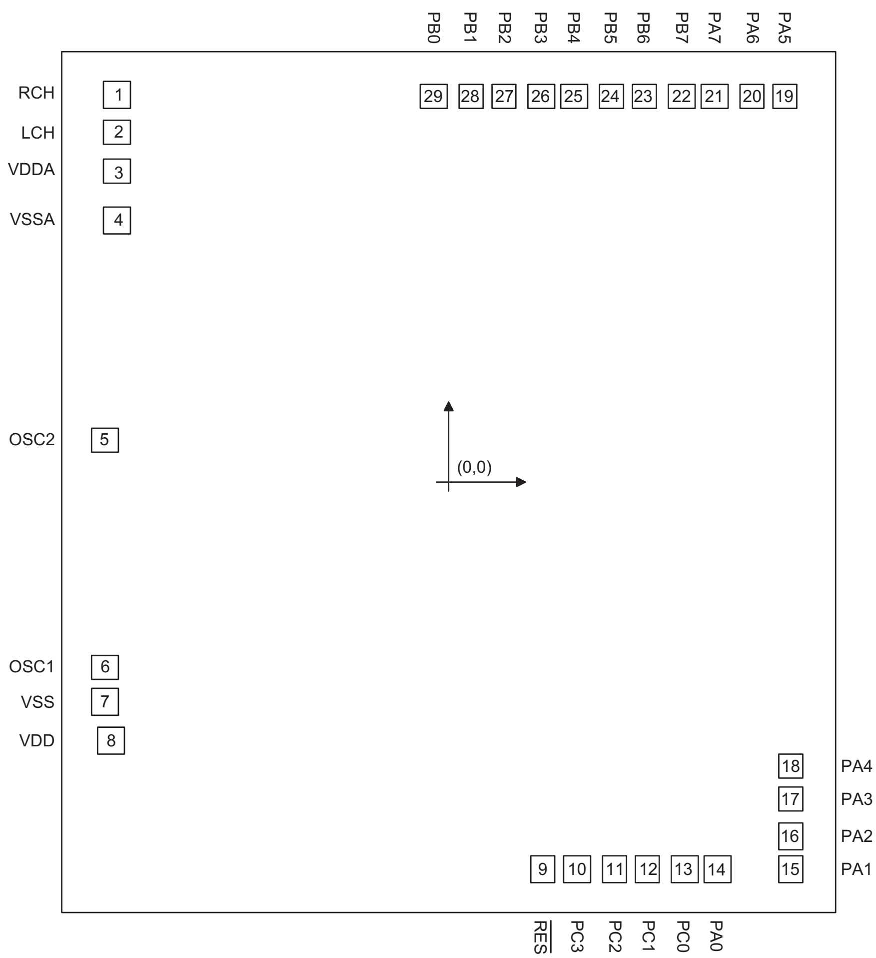


Pin Assignment





Pad Assignment



Chip size: 2355 × 2615 (μm)^2

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

Pad Coordinates

Unit: μm

Pad No.	X	Y	Pad No.	X	Y
1	−988.650	1160.250	16	1027.050	−1056.650
2	−988.650	1047.250	17	1027.050	−945.550
3	−988.650	934.800	18	1027.050	−845.550
4	−988.650	786.150	19	1010.150	1156.650
5	−1027.075	127.300	20	910.150	1156.650
6	−1027.050	−550.500	21	799.550	1156.650
7	−1027.050	−655.900	22	699.550	1156.650
8	−1006.750	−768.100	23	588.950	1156.650
9	285.550	−1156.650	24	488.950	1156.650
10	387.350	−1156.650	25	378.350	1156.650
11	497.950	−1156.650	26	278.350	1156.650
12	597.950	−1156.650	27	167.750	1156.650
13	708.550	−1156.650	28	67.750	1156.650
14	808.550	−1156.650	29	−42.850	1156.650
15	1027.050	−1156.650			

Pad Description

Pad No.	Pad Name	I/O	Internal Connection	Function
8, 7	VDD, VSS	—	—	Digital power supply, ground
3,4	VDDA, VSSA	—	—	DAC power supply
14~21	PA0~PA7	I/O	Wake-up, Pull-high or None	Bidirectional 8-bit I/O port, wake-up by mask option
29~22	PB0~PB7	I/O	Pull-high or None	Bidirectional 8-bit I/O port
13~10	PC0~PC3	I/O	Pull-high or None	Bidirectional 4-bit I/O port
9	$\overline{\text{RESET}}$	I	—	Reset input, active low
6	OSC1	I	X′tal/Resistor	XIN for X′tal or ROSCIN for resistor by mask option
5	OSC2	O	—	XOUT or T1
1	RCH	O	—	DAC output R channel
2	LCH	O	—	DAC output L channel

Note: In the form pin number value is for the 32 SOP type MCU.

Absolute Maximum Ratings

Supply Voltage ....V_{SS}−0.3V to V_{SS}+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

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.

Electrical Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	Operating Voltage	—	—	2.4	4.5	5	V
I_{DD}	Operating Current	3V	No load (OSC= 6MHz)	—	2	8	mA
		4.5V		—	8	10	
I_{STB}	Standby Current	3V	—	—	1	—	μA
		4.5V		—	1	3	
I_{OH}	Flag Source Current	3V	—	5	—	—	mA
		4.5V					
I_{OL}	Flag Sink Current	3V	—	5	—	—	mA
		4.5V					
V_{IH}	Input High Voltage for I/O Ports	—	—	$0.8V_{DD}$	—	V_{DD}	V
V_{IL}	Input Low Voltage for I/O Ports	—	—	0	—	$0.2V_{DD}$	V

Function Description

Execution Flow

The system clock for the HT36F6 is derived from either a crystal or an RC oscillator. The oscillator frequency divided by 2 is the system clock for the MCU and it is internally divided into four non-overlapping clocks. One instruction cycle consists of four system clock cycles.

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

Program Counter – PC

The 13-bit program counter (PC) controls the sequence in which the instructions stored in program ROM are executed and its contents specify a maximum of 8192 addresses for each bank.

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

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

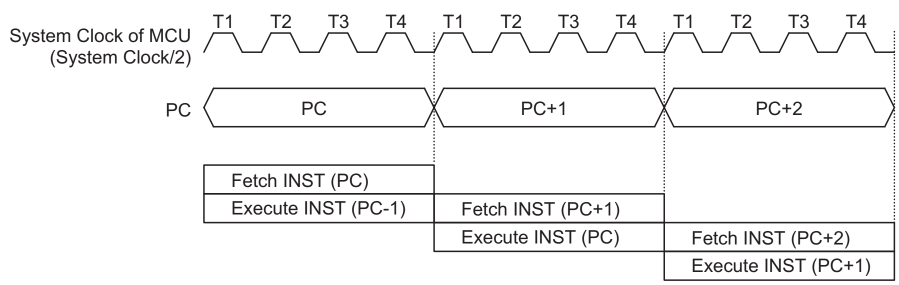
The conditional skip is activated by instruction. Once the condition is met, the next instruction, fetched during the current instruction execution, is discarded and a dummy cycle replaces it to retrieve the proper instruction. Otherwise proceed with the next instruction.

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

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

Program ROM

HT36F6 provides 17 address lines WA16~WA0 to read the Program ROM which is up to 1M bits, and is commonly used for the wavetable voice codes and the program memory. It provides two address types, one type is for program ROM, which is addressed by a bank pointer PF2~PF0 and a 13-bit program counter PC12~PC0;



Execution Flow

Mode	Program Counter															
	*15	*14	*13	*12	*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	0	0	0	0
Timer/Event Counter 0 Overflow	PF2	PF1	PF0	0	0	0	0	0	0	0	0	0	1	0	0	0
Timer/Event Counter 1 Overflow	PF2	PF1	PF0	0	0	0	0	0	0	0	0	0	1	1	0	0
Skip	Program Counter+2															
Loading PCL	PF2	PF1	PF0	*12	*11	*10	*9	*8	@7	@6	@5	@4	@3	@2	@1	@0
Jump, Call Branch	PF2	PF1	PF0	#12	#11	#10	#9	#8	#7	#6	#5	#4	#3	#2	#1	#0
Return From Subroutine	PF2	PF1	PF0	S12	S11	S10	S9	S8	S7	S6	S5	S4	S3	S2	S1	S0

Program Counter

Note: *12~*0: Bits of Program Counter
@7~@0: Bits of PCL
#12~#0: Bits of Instruction Code
S12~S0: Bits of Stack Register
@7~@0: Bits of PCL
PF2~PF1: Bits of Bank Register

and the other type is for wavetable code, which is addressed by the start address ST15~ST0. On the program type, WA16~WA0=PF2~PF0$\times 2^{13}$+PC12~PC0. On the wave table ROM type, WA16~WA0= ST15~ST0$\times 2^5$.

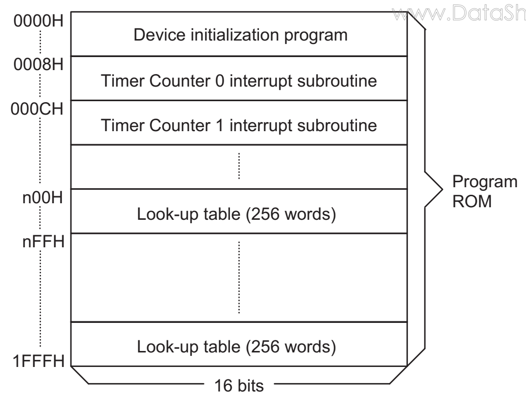
Note: Program ROM address use word as address unit, but wavetable ROM address use BYTE as address unit.

Program Memory – ROM

The program memory is used to store the program instructions which are to be executed. It also contains data, table, and interrupt entries, and is organized into 8192$\times$16 bits, addressed by the bank pointer, program counter and table pointer.

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

- Location 000H on bank0
  This area is reserved for the initialization program. After chip reset, the program always begins execution at location 000H on bank0.
- Location 008H
  This area is reserved for the Timer Counter 0 interrupt service program on each bank. If timer interrupt results from a timer counter 0 overflow, and if the interrupt is enabled and the stack is not full, the program begins execution at location 008H corresponding to its bank.
- Location 00CH
  This area is reserved for the Timer Counter 1 interrupt service program on each bank. If a timer interrupt results from a Timer Counter 1 overflow, and if the interrupt is enabled and the stack is not full, the program begins execution at location 00CH corresponding to its bank.
- Table location
  Any location in the ROM space can be used as look-up tables. 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 higher-order byte of the table word are transferred to the TBLH. The Table Higher-order byte register (TBLH) is read only. The Table Pointer (TBLP) is a read/write register (07H), which indicates the table location. Before accessing the table, the location must be placed in TBLP. The TBLH is read only and cannot be restored. If the main routine and the ISR (Interrupt Service Routine) both employ the table read instruction, the contents of the TBLH in the main routine are likely to be changed by the table read instruction used in the ISR. Errors can occur. In this case, using the table read instruction in the main routine and the ISR simultaneously should be avoided. However, if the table read instruction has to be applied in both the main routine and the ISR, the interrupt should be disabled prior to the table read instruction. It will not be enabled until the TBLH has been backed up. All table related instructions need 2 cycles to complete the operation. These areas may function as normal program memory depending upon user requirements.



Note: n ranges from 00 to 1F.

Program Memory for Each Bank

- Bank pointer
  The program memory is organized into 8 banks and each bank into 8192$\times$16 bits of program ROM. PF2~PF0 is used as the bank pointer. After an instruction has been executed to write data to the PF register to select a different bank, note that the new bank will not be selected immediately. It is not until the following instruction has completed execution that the bank will be actually selected. It should be note that the PF register has to be cleared before setting to output mode.

Instruction (s)	Table Location															
	*15	*14	*13	*12	*11	*10	*9	*8	*7	*6	*5	*4	*3	*2	*1	*0
TABRDC [m]	P15	P14	P13	P12	P11	P10	P9	P8	@7	@6	@5	@4	@3	@2	@1	@0
TABRDL [m]	P15	P14	P13	1	1	1	1	1	@7	@6	@5	@4	@3	@2	@1	@0

Table Location

Note: *12~*0: Bits of table location
@7~@0: Bits of table pointer
P12~P8: Bits of current Program Counter
P15~P13: Bits of bank PF2~PF0

Wavetable ROM

The ST15~ST0 is used to defined the start address of each sample on the wavetable and read the waveform data from the location. HT36F6 provides 17 output address lines from WA16~WA0, the ST15~ST0 is used to locate the major 12 bits i.e. WA16~WA5 and the undefined data from WA4~WA0 is always set to 00000b. So the start address of each sample have to be located at a multiple of 32. Otherwise, the sample will not be read out correctly because it has a wrong starting code.

Stack Register – Stack

This is a special part of the memory which is used to save the contents of the program counter only. The stack is organized into 8 levels and is neither part of the data nor part of the program space, and is neither readable nor writeable. The activated level is indexed by the stack pointer (SP) and is neither readable nor writeable. At a subroutine call or interrupt acknowledgment, 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 acknowledgment will be inhibited. When the stack pointer is decremented (by RET or RETI), the interrupt will be serviced. This feature prevents stack overflow allowing the programmer to use the structure more easily. In a similar case, if the stack is full and a CALL is subsequently executed, a stack overflow occurs and the first entry will be lost (only the most recent eight return address are stored).

Data Memory – RAM

The data memory is designed with 256 × 8 bits. The data memory is divided into three functional groups: special function registers, wavetable function register, and general purpose data memory (208×8). Most of them are read/write, but some are read only.

The special function registers include the Indirect Addressing register 0 (00H), the Memory Pointer register 0 (MP0;01H), the Indirect Addressing register 1 (02H), the Memory Pointer register 1 (MP1;03H), the Accumulator (ACC;05H), the Program Counter Lower-byte register (PCL;06H), the Table Pointer (TBLP;07H), the Table Higher-order byte register (TBLH;08H), the Watchdog Timer option Setting register (WDTS;09H), the Status register (STATUS;0AH), the Interrupt Control register (INTC;0BH), the Timer Counter 0 Lower-order byte register (TMR0L;0DH), the Timer Counter 0 Control register (TMR0C;0EH), the Timer Counter 1 Lower-order byte register (TMR1L;10H), the Timer Counter 1 Control register (TMR1C;11H), the I/O registers (PA;12H, PB;14H, PC;16H) and the I/O control registers (PAC;13H, PBC;15H, PCC;17H). The program ROM bank select (PF;1CH). The DAC High byte (DAH;1DH). The DAC low byte (DAL;1EH). The DAC control (DAC;1FH). The wavetable function registers is defined between 20H~2AH. The remaining space before the

00H	Indirect Addressing Register 0	Special Purpose DATA MEMORY
01H	MP0	
02H	Indirect Addressing Register 1	
03H	MP1	
04H		
05H	ACC	
06H	PCL	
07H	TBLP	
08H	TBLH	
09H	WDTS	
0AH	STATUS	
0BH	INTC	
0CH		
0DH	TMR0L	
0EH	TMR0C	
0FH		
10H	TMR1L	
11H	TMR1C	
12H	PA	
13H	PAC	
14H	PB	
15H	PBC	
16H	PC	
17H	PCC	
18H		
19H		
1AH		
1BH		
1CH	PF	
1DH	DAC High Byte (DAH)	Wavetable Function Register
1EH	DAC Low Byte (DAL)	
1FH	DAC Control (DAC)	
20H	Channel Number Select (CHAN)	
21H	Frequency Number High Byte (FreqNH)	
22H	Frequency Number Low Byte (FreqNL)	
23H	Start Address High Byte (AddrH)	
24H	Start Address Low Byte (AddrL)	
25H	Repeat Number High Byte (ReH)	
26H	Repeat Number Low Byte (ReL)	
27H	Volume Control High (ENV)	
28H		
29H	Left Volume Control (LVC)	
2AH	Right Volum Control (RVC)	
2BH ~ 2FH		
30H ~ FFH	General Purpose DATA MEMORY (208 Bytes)	

: Unused. Read as "00"

RAM Mapping

30H is reserved for future expanded usage and reading these locations will return the result 00H. The general purpose data memory, addressed from 30H to FFH, is used for data and control information under instruction command.

All data memory areas can handle arithmetic, logic, increment, decrement and rotate operations directly. Except for some dedicated bits, each bit in the data memory can be set and reset by the ″SET [m].i″ and ″CLR [m].i″ instructions, respectively. They are also indirectly accessible through Memory pointer registers (MP0:01H, MP1:03H).

Indirect Addressing Register

Location 00H and 02H are indirect addressing registers that are not physically implemented. Any read/write operation of [00H] and [02H] access data memory pointed to by MP0 (01H) and MP1 (03H) respectively. Reading location 00H or 02H directly will return the result 00H. And writing directly results in no operation.

The function of data movement between two indirect addressing registers, is not supported. The memory pointer registers, MP0 and MP1, are 8-bit register which can be used to access the data memory by combining corresponding indirect addressing registers.

Accumulator

The accumulator closely relates to ALU operations. It is mapped to location 05H of the data memory and it can operate with immediate data. The data movement between two data memory locations must pass through the accumulator.

Arithmetic and Logic Unit – ALU

This circuit performs 8-bit arithmetic and logic operation. 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 & Decrement (INC, DEC)
- Branch decision (SZ, SNZ, SIZ, SDZ ....)

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

Status Register – STATUS

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

With the exception of the TO and PDF flags, bits in the status register can be altered by instructions like any other register. Any data written into the status register will not change the TO or PDF flags. In addition it should be noted that operations related to the status register may give different results from those intended. The TO and PDF flags can only be changed by system power up, Watchdog Timer overflow, executing the ″HALT″ instruction and clearing the Watchdog Timer.

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

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

Interrupt

The HT36F6 provides two internal timer counter interrupts on each bank. The Interrupt Control register (INTC;0BH) contains the interrupt control bits that sets the enable/disable and the interrupt request flags.

Once an interrupt subroutine is serviced, all other interrupts will be blocked (by clearing the EMI bit). This scheme may prevent any further interrupt nesting. Other

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. Also it is 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 logical 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 either 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

interrupt requests may occur during this interval but only the interrupt request flag is recorded. If a certain interrupt needs servicing within the service routine, the programmer may set the EMI bit and the corresponding bit of the INTC to allow interrupt nesting. If the stack is full, the interrupt request will not be acknowledged, even if the related interrupt is enabled, until the SP is decremented. If immediate service is desired, the stack must be prevented from becoming full.

All these kinds of interrupt have a wake-up capability. As an interrupt is serviced, a control transfer occurs by pushing the program counter onto the stack and then branching to subroutines at specified locations in the program memory. Only the program counter is pushed onto the stack. If the contents of the register and Status register (STATUS) are altered by the interrupt service program which may corrupt the desired control sequence, then the programmer must save the contents first.

The internal Timer Counter 0 interrupt is initialized by setting the Timer Counter 0 interrupt request flag (T0F; bit 5 of INTC), caused by a Timer Counter 0 overflow. When the interrupt is enabled, and 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 Timer Counter 1 interrupt is operated in the same manner as Timer Counter 0. The related interrupt control bits ET1I and T1F of the Timer Counter 1 are bit 3 and bit 6 of the INTC respectively.

During the execution of an interrupt subroutine, other interrupt acknowledgments 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, the ″RET″ or ″RETI″ instruction may be invoked. RETI will set the EMI bit to enable an interrupt service, but RET will not.

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

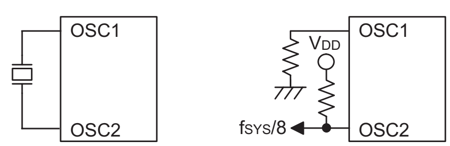
Interrupt Source	Priority	Vector
Timer Counter 0 overflow	1	08H
Timer Counter 1 overflow	2	0CH

The Timer Counter 0/1 interrupt request flag (T0F/T1F), Enable Timer Counter 0/1 bit (ET0I/ET1I) and Enable Master Interrupt bit (EMI) constitute an interrupt control register (INTC) which is located at 0BH in the data memory. EMI, ET0I, ET1I are used to control the enabling/disabling of interrupts. These bits prevent the requested interrupt from being serviced. Once the interrupt request flags (T0F, T1F) are set, they will remain in the INTC register until the interrupts are serviced or cleared by a software instruction.

It is recommended that a program does not use the ″CALL subroutine″ within the interrupt subroutine. Because 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, once the ″CALL subroutine″ operates in the interrupt subroutine, it may damage the original control sequence.

Oscillator Configuration

The HT36F6 provides two types of oscillator circuit for the system clock, .e., RC oscillator and crystal oscillator. No matter what type of oscillator, the signal divided by 2 is used for the system clock. The HALT mode stops the system oscillator and ignores external signal to conserve power. If the RC oscillator is used, an external resistor between OSC1 and VSS is required. The system clock, divided by 4, is available on OSC2 with pull-high resistor, which can be used to synchronize external



System Oscillator

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 Counter 0 interrupt (1=enabled; 0=disabled)
3	ET1I	Controls the Timer Counter 1 interrupt (1=enabled; 0=disabled)
5	T0F	Internal Timer Counter 0 request flag (1=active; 0=inactive)
6	T1F	Internal Timer Counter 1 request flag (1=active; 0=inactive)

INTC (0BH) Register

logic. The RC oscillator provides the most cost effective solution. However, 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 accurate oscillator frequency is desired.

On the other hand, if the crystal oscillator is selected, a crystal across OSC1 and OSC2 is needed to provide the feedback and phase shift required for the oscillator, and no other external components are required. A resonator may be connected between OSC1 and OSC2 to replace the crystal and to get a frequency reference, but two external capacitors in OSC1 and OSC2 are required.

The WDT oscillator is a free running on-chip RC oscillator, and no external components are required. Even if the system enters the Power Down Mode, the system clock is stopped, but the WDT oscillator still works with a period of approximately 78μs. The WDT oscillator can be disabled by mask option to conserve power.

Watchdog Timer – WDT

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

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

If the WDT oscillator is disabled, the WDT clock may still come from the instruction clock and operate in the same manner except that in the HALT state the WDT may stop counting and lose its protecting purpose. In this situation the logic can only be restarted by external logic. The high nibble and bit 3 of the WDTS are reserved for user defined flags, and the programmer may use these flags to indicate some specified status.

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

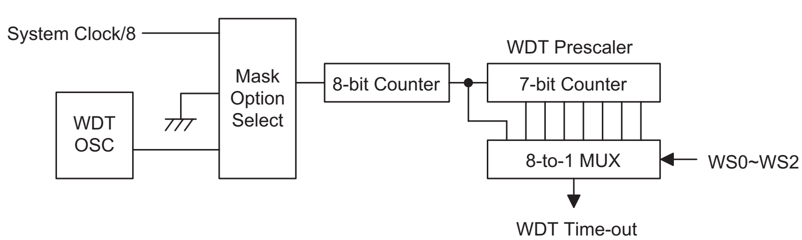
If the device operates in a noisy environment, using the on-chip RC oscillator (WDT OSC) is strongly recommended, since the HALT will stop the system clock.

The WDT overflow under normal operation will initialize a ″chip reset″ and set the status bit TO. Whereas in the HALT mode, the overflow will initialize a ″warm reset″ only the Program Counter and SP are reset to zero. To clear the WDT contents (including the WDT prescaler), 3 methods are implemented; external reset (a low level to $\overline{RES}$), software instructions, or a ″HALT″ instruction. The software instructions include ″CLR WDT″ and the other set – ″CLR WDT1″ and ″CLR WDT2″. Of these two types of instructions, only one can be active depending on the mask option – ″CLR WDT times selection option″. If the ″CLR WDT″ is selected (i.e. CLRWDT times equal one), any execution of the ″CLR WDT″ instruction will clear the WDT. In case ″CLR WDT1″ and ″CLR WDT2″ are chosen (i.e. CLRWDT times equal two), these two instructions must be executed to clear the WDT; otherwise, the WDT may reset the chip because of time-out.

Power Down Operation – HALT

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

- The system oscillator will turn off but the WDT oscillator keeps running (If the WDT oscillator is selected). Watchdog Timer – WDT
- The contents of the on-chip RAM and registers remain unchanged
- The WDT and WDT prescaler will be cleared and starts to count again (if the clock comes from the WDT oscillator).



Watchdog Timer

- All I/O ports maintain their original status.
- The PDF flag is set and the TO flag is cleared.
- The HALT pin will output a high level signal to disable the external ROM.

The system can leave the HALT mode by means of an external reset, an interrupt, an external falling edge signal on port A or a WDT overflow. An external reset causes a device initialization and the WDT overflow performs a ″warm reset″. By examining the TO and PDF flags, the cause for a chip reset can be determined. The PDF flag is cleared when there is a system power-up or by executing the ″CLR WDT″ instruction and it is set when a ″HALT″ instruction is executed. The TO flag is set if the WDT time-out occurs, and causes a wake-up that only resets the Program Counter and SP, the others remain in their original status.

The port A wake-up and interrupt methods can be considered as a continuation of normal execution. Each bit in port A can be independently selected to wake-up the device by mask option. Awakening from an I/O port stimulus, the program will resume execution of the next instruction. If awakening from an interrupt, two sequences may occur. If the related interrupts is disabled or the interrupts 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, a regular interrupt response takes place.

Once a wake-up event occurs, it takes 1024 t_{SYS} (system clock period) to resume to normal operation. In other words, a dummy cycle period will be inserted after the wake-up. If the wake-up results from an interrupt acknowledge, the actual interrupt subroutine will be delayed by one more cycle. If the wake-up results in next instruction execution, this will execute immediately after a dummy period has finished. If an interrupt request flag is set to ″1″ before entering the HALT mode, the wake-up function of the related interrupt will be disabled.

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

Reset

There are 3 ways in which a reset can occur:

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

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

TO	PDF	RESET Conditions
0	0	$\overline{\text{RES}}$ 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

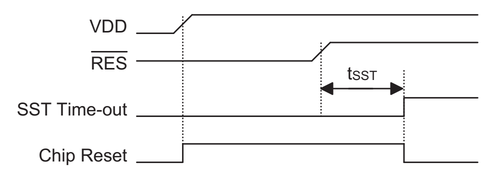
Note: ″u″ stands for ″unchanged″

To guarantee that the system oscillator has started and stabilized, the SST (System Start-up Timer) provides an extra-delay of 1024 system clock pulses during system power up or when the system awakes from a HALT state.

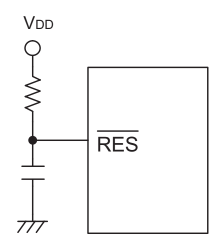
When a system power-up occurs, the SST delay is added during the reset period. But when the reset comes from the $\overline{\text{RES}}$ pin, the SST delay is disabled. Any wake-up from HALT will enable the SST delay.

The functional units chip reset status are shown below.

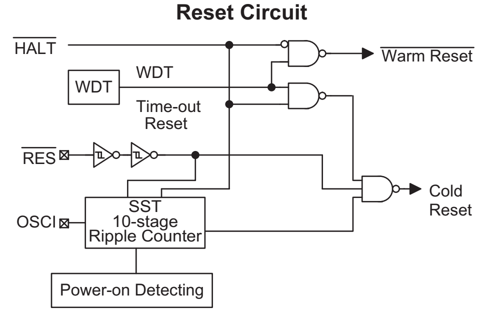
Program Counter	000H
Interrupt	Disable
Prescaler	Clear
WDT	Clear. After master reset, WDT begins counting
Timer Counter (0/1)	Off
Input/output ports	Input mode
Stack Pointer	Points to the top of stack



Reset Timing Chart



Reset Circuit



Reset Configuration

The registers status is summarized in the following table:

Register	Reset (Power On)	WDT Time-out (Normal Operation)	$\overline{\text{RES}}$ Reset (Normal Operation)	$\overline{\text{RES}}$ Reset (HALT)	WDT Time-out (HALT)*
MP0	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
MP1	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
Program Counter	0000H	0000H	0000H	0000H	0000H
TBLP	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLH	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
WDTS	0000 0111	0000 0111	0000 0111	0000 0111	uuuu uuuu
STATUS	--00 xxxx	--1u uuuu	--uu uuuu	--01 uuuu	--11 uuuu
INTC	-00- 00-0	-00- 00-0	-00- 00-0	-00- 00-0	-uu- uu-u
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 1uuu
TMR1L	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TMR1C	00-0 1000	00-0 1000	00-0 1000	00-0 1000	uu-u 1uuu
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	---- uuuu
PCC	---- 1111	---- 1111	---- 1111	---- 1111	---- uuuu
PF	---- 0000	---- 0000	---- 0000	---- 0000	---- uuuu
DAH	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
DAL	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
DAC	---- -000	---- -000	---- -000	---- -000	---- -uuu
CHAN	00-- --00	uu-- --uu	uu-- --uu	uu-- --uu	uu-- --uu
FreqNH	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
FreqNL	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
AddrH	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
AddrL	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
ReH	x--x xxxx	u--u uuuu	u--u uuuu	u--u uuuu	u--u uuuu
ReL	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
ENV	x-xx --xx	u-uu --uu	u-uu --uu	u-uu --uu	u-uu --uu
LVC	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
RVC	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu

Note: ″*″ stands for warm reset
″u″ stands for unchanged
″x″ stands for unknown
″–″ stands for unused

Timer 0/1

Timer 0 is an 8-bit counter, and its clock source comes from the system clock divided by an 8-stage prescaler. There are two registers related to Timer 0; TMR0L(0DH) and TMR0C(0EH). One physical registers are mapped to TMR0L location; writing TMR0L makes the starting value be placed in the Timer 0 preload register and reading the TMR0 gets the contents of the Timer 0 counter. The TMR0C is a control register, which defines the division ration of the prescaler and counting enable or disable.

Writing data to B2, B1 and B0 (bits 2, 1, 0 of TMR0C) can yield various clock sources.

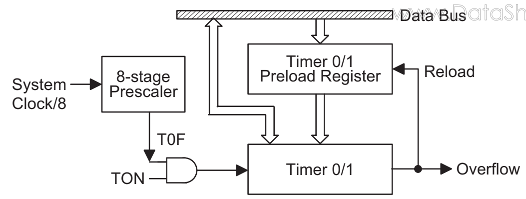
One the Timer 0 starts counting, it will count from the current contents in the counter to FFH. Once an overflow occurs, the counter is reloaded from a preload register, and generates an interrupt request flag (T0F; bit 2 of INTCH). To enable the counting operation, the timer On bit (TON; bit 4 of TMR0C) should be set to ″1″. For proper operation, bit 7 of TMR0C should be set to ″1″ and bit 3, bit 6 should be set to ″0″.

There are two registers related to the Timer Counter1; TMR1L(10H), TMR1C(11H). The Timer Counter 1 operates in the same manner as Timer Counter 0.

TMR0C/TMR1C			T0F
B2	B1	B0	
0	0	0	SYS CLK/16
0	0	1	SYS CLK/32
0	1	0	SYS CLK/64
0	1	1	SYS CLK/128
1	0	0	SYS CLK/256
1	0	1	SYS CLK/512
1	1	0	SYS CLK/1024
1	1	1	SYS CLK/2048

TMR0C Bit 4 to enable/disable timer counting (1=enable; 0=disable)

TMR0C Bit 3, always write ″0″.
TMR0C Bit 5, always write ″0″.
TMR0C Bit 6, always write ″0″.
TMR0C Bit 7, always write ″1″.



Timer 0/1

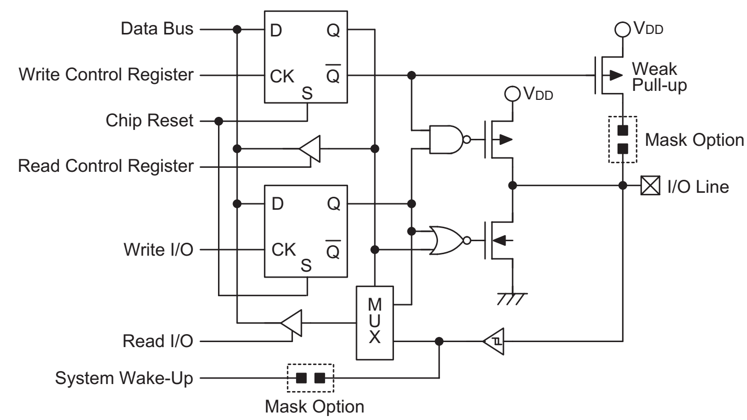
Input/Output Ports

There are 20 bidirectional input/output lines labeled from PA, PB, PC0~PC3, which are mapped to the data memory of [12H], [14H], [16H] respectively. All 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 data is latched and remains unchanged until the output latch is rewritten.

Each I/O line has its own control register (PAC, PBC, PCC0~PCC3) to control the input/output configuration. With this control register, CMOS output or Schmitt trigger input with or without pull-high resistor (mask option) structures can be reconfigured dynamically under software control. To function as an input, the corresponding latch of the control register must write a ″1″. The pull-high resistance will exhibit automatically if the pull-high option is selected. The input source also depends on the control register. If the control register bit is ″1″, 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 ″read-modify-write″ instruction. For output function, CMOS is the only configuration. These control registers are mapped to locations 13H, 15H and 17H.

After a chip reset, these input/output lines remain at high levels or floating (mask option). Each bit of these input/output latches can be set or cleared by the ″SET [m].i″ or ″CLR [m].i″ (m=12H, 14H or 16H) instruction.

Some instructions first input data and then follow the output operations. For example, the ″SET [m].i″, ″CLR [m].i″, ″CPL [m]″ and ″CPLA [m]″ instructions 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 to wake-up the device.



Input/Output Ports

Channel Wavetable Synthesizer

Name	Function	D7	D6	D5	D4	D3	D2	D1	D0
1DH	DAC high byte (no default value)	DA15	DA14	DA13	DA12	DA11	DA10	DA9	DA8
1EH	DAC low byte (no default value)	DA7	DA6	DA5	DA4	DA3	DA2	DA1	DA0
1FH	DAON=1: DAC ON DAON=0: DAC OFF (default) SELW=1: DAC data from wavetable SELW=0: DAC data from MCU	—	—	—	—	—	Left	DAON	SELW Right
20H	Channel number selection	VM	FR					CH1	CH0
21H	High byte frequency number	BL3	BL2	BL1	BL0	FR11	FR10	FR9	FR8
22H	Low byte frequency number	FR7	FR6	FR5	FR4	FR3	FR2	FR1	FR0
23H	High byte start address	ST15	ST14	ST13	ST12	ST11	ST10	ST9	ST8
24H	Low byte start address	ST7	ST6	ST5	ST4	ST3	ST2	ST1	ST0
25H	Wave bit select, High byte repeat number	WBS			RE12	RE11	RE10	RE9	RE8
26H	Low byte repeat number	RE7	RE6	RE5	RE4	RE3	RE2	RE1	RE0
27H	Envelope control, Volume control	A_R		VL9	VL8			VR9	VR8
28H	Unused								
29H	Left Volume control	VL7	VL6	VL5	VL4	VL3	VL2	VL1	VL0
2AH	Right Volume control	VR7	VR6	VR5	VR4	VR3	VR2	VR1	VR0
2BH~2FH	Unused								
30H~FFH	Data memory (RAM)	General purpose data memory (same as 8-Bit MCU)							

Memory Map (1DH~FFH) Register

Note: ″—″ No function, read only, read as ″0″.

Unused: No function, read only, read as ″0″.

- CH1~CH0 channel number selection
  The HT36F6 has a built-in 8 output channels and CH1~CH0 is used to define which channel is selected. When this register is written to, the wavetable synthesizer will automatically output the dedicated PCM code. So this register is also used as a start playing key and it has to be written to after all the other wavetable function registers are already defined.
- Change parameter selection
  These two bits, VM and FR, are used to define which register will be updated on this selected channel. There are two modes that can be selected to reduce the process of setting the register. Please refer to the statements of the following table:

VM	FR	Function
0	0	Update all the parameter
0	1	Only update the frequency number
1	0	Only update the volume

• Output frequency definition

The data on BL3~BL0 and FR11~FR0 are used to define the output speed of the PCM file, i.e. it can be used to generate the tone scale. When the FR11~FR0 is 800H and BL3~BL0 is 6H, each sample data of the PCM code will be sent out sequentially.

When the f_{OSC} is 6.4MHz, the formula of a tone frequency is:

$$f_{OUT}= f_{RECORD} \times \frac{25kHz}{SR} \times \frac{FR11\sim FR0}{2^{(17-BL3\sim BL0)}}$$

where f_{OUT} is the output signal frequency, f_{RECORD} and SR is the frequency and sampling rate on the sample code, respectively.

So if a voice code of C3 has been recorded which has the f_{RECORD} of 261Hz and the SR of 11025Hz, the tone frequency (f_{OUT}) of G3: f_{OUT}=98Hz.

Can be obtained by using the fomula:

$$98Hz= 261Hz \times \frac{25kHz}{11025Hz} \times \frac{FR11\sim FR0}{2^{(17-BL3\sim BL0)}}$$

A pair of the values FR11~FR0 and BL3~BL0 can be determined when the f_{OSC} is 6.4MHz.

• Start address definition

The HT36F6 provides two address types for extended use, one is the program ROM address which is program counter corresponding with PF value, the other is the start address of the PCM code.

The ST15~ST0 is used to define the start address of each PCM code and reads the waveform data from this location. The HT36F6 provides 16 input data lines from WA16~WA0, the ST15~ST0 is used to locate the major 12 bits .e. WA16~WA5 and the undefined data from WA4~WA0 is always set as 00000b. In other words, the WA16~WA0=ST15~ST0$\times 2^5$. So each PCM code has to be located at a multiple of 32. Otherwise, the PCM code will not be read out correctly because it has a wrong start code.

• Waveform format definition

The HT36F6 accepts two waveform formats to ensure a more economical data space. WBS is used to define the sample format of each PCM code.

- WBS=0 means the sample format is 8-bit
- WBS=1 means the sample format is 12-bit

The 12-bit sample format allocates location to each sample data. Please refer to the waveform format statement as shown below.

8-Bit	1B	2B	3B	4B	5B	6B	7B	8B	....

A sampling data code; B means one data byte.

12-Bit	1H	1M	1L	2L	2H	2M	3H	3M	3L	....

A sampling data code

Note: "1H" High Nibble
"1M" Middle Nibble
"1L" Low Nibble

Waveform Format

• Repeat number definition

The repeat number is used to define the address which is the repeat point of the sample. When the repeat number is defined, it will be output from the start code to the end code once and always output the range between the repeat address to the end code (80H) until the volume become close.

The RE12~RE0 is used to calculate the repeat address of the PCM code. The process for setting the RE12~RE0 is to write the 2's complement of the repeat length to RE12~RE0, with the highest carry ignored. The HT36F6 will get the repeat address by adding the RE12~RE0 to the address of the end code, then jump to the address to repeat this range.

• Volume control

The HT36F6 provides the volume control independently. The volume are controlled by VR9~VR0 respectively. The chip provides 1024 levels of controllable volume, the 000H is the maximum and 3FFH is the minimum output volume.

• The PCM code definition

The HT36F6 can only solve the voice format of the signed 8-bit or 12-bit raw PCM. And the MCU will take the voice code 80H as the end code.

So each PCM code section must be ended with the end code 80H.

• Digital to Analog Converter – DAC

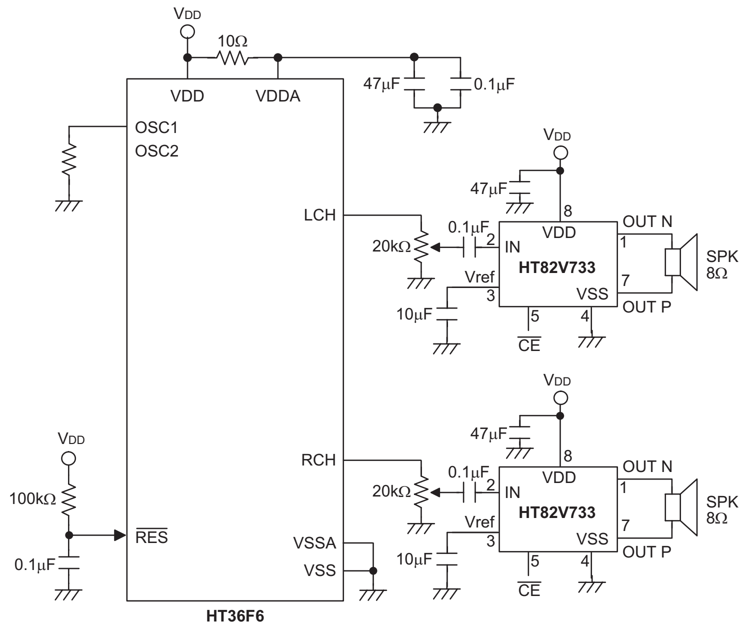
The HT36F6 provides one 16-bit voltage type DAC device controlled by the MCU or Wavetable Synthesizer for driving the external speaker through an external NPN transistor. It is in fact an optional object used for Wavetable Synthesizer DAC or general DAC, this is chosen by DAC control register. If the general DAC is chosen for application, then the Wavetable Synthesizer is disabled since the DAC is taken up and controlled by the MCU. If general DAC is selected, the programmer must write the voice data to register DAL and DAH to get the corresponding analog data. If Mask Option enables the DAC register and the SELW, then the following table comes useful.

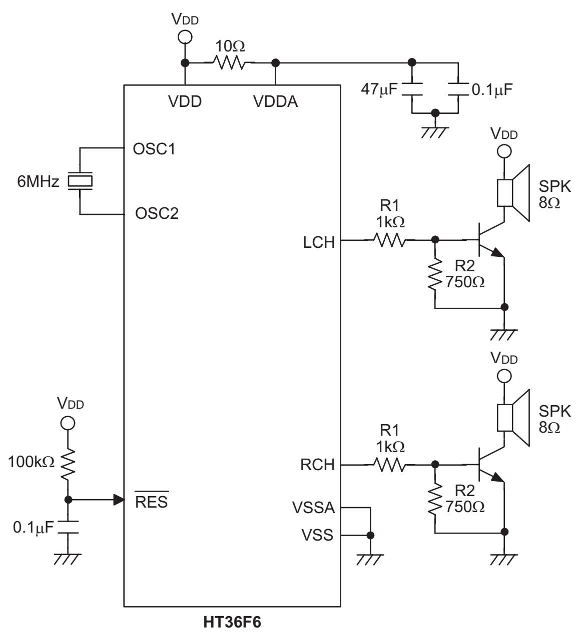
Bit No.	Label	Function
Bit7~Bit3	—	No used
Bit2	SELWL	SELWL=1, left channel DAC data from wavetable SELWL=0, left channel DAC data from MCU (Default)
Bit1	DAON	DAON=1: DAC ON DAON=0: DAC OFF (Default)
Bit0	SELWR	SELWR=1, Right Channel DAC data from Wavetable SELWR=0, Right Channel DAC data from MCU (Default)

Mask Option

No.	Mask Option	Function
1	WDT source	On-chip RC/Instruction clock/ disable WDT
2	CLRWDT times	One time, two times (CLR WDT1/WDT2)
3	Wake-up	PA only
4	Pull-High	PA, PB, PC input
5	OSC mode	Crystal or Resistor type
6	LVR	Enable/disable
7	LVD	2.2V/3.3V

Application Circuit





Note: R1 > R2

Instruction Set Summary

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

Mnemonic	Description	Instruction Cycle	Flag Affected
Branch			
JMP addr	Jump unconditionally	2	None
SZ [m]	Skip if data memory is zero	$1^{(2)}$	None
SZA [m]	Skip if data memory is zero with data movement to ACC	$1^{(2)}$	None
SZ [m].i	Skip if bit i of data memory is zero	$1^{(2)}$	None
SNZ [m].i	Skip if bit i of data memory is not zero	$1^{(2)}$	None
SIZ [m]	Skip if increment data memory is zero	$1^{(3)}$	None
SDZ [m]	Skip if decrement data memory is zero	$1^{(3)}$	None
SIZA [m]	Skip if increment data memory is zero with result in ACC	$1^{(2)}$	None
SDZA [m]	Skip if decrement data memory is zero with result in ACC	$1^{(2)}$	None
CALL addr	Subroutine call	2	None
RET	Return from subroutine	2	None
RET A,x	Return from subroutine and load immediate data to ACC	2	None
RETI	Return from interrupt	2	None
Table Read			
TABRDC [m]	Read ROM code (current page) to data memory and TBLH	$2^{(1)}$	None
TABRDL [m]	Read ROM code (last page) to data memory and TBLH	$2^{(1)}$	None
Miscellaneous			
NOP	No operation	1	None
CLR [m]	Clear data memory	$1^{(1)}$	None
SET [m]	Set data memory	$1^{(1)}$	None
CLR WDT	Clear Watchdog Timer	1	TO,PDF
CLR WDT1	Pre-clear Watchdog Timer	1	$TO^{(4)}$,$PDF^{(4)}$
CLR WDT2	Pre-clear Watchdog Timer	1	$TO^{(4)}$,$PDF^{(4)}$
SWAP [m]	Swap nibbles of data memory	$1^{(1)}$	None
SWAPA [m]	Swap nibbles of data memory with result in	1	None
HALT	Enter Power Down Mode	1	TO,PDF

Note: x: Immediate data

m: Data memory address

A: Accumulator

i: 0~7 number of bits

addr: Program memory address

√: Flag is affected

–: Flag is not affected

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

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

(3): (1) and (2)

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

Instruction Definition

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

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

Operation: ACC ← ACC+[m]+C

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	√	√	√	√

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

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

Operation: [m] ← ACC+[m]+C

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	√	√	√	√

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

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

Operation: ACC ← ACC+[m]

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	√	√	√	√

ADD A,x Add immediate data to the accumulator

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

Operation: ACC ← ACC+x

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	√	√	√	√

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

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

Operation: [m] ← ACC+[m]

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	√	√	√	√

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

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

Operation: ACC ← ACC ″AND″ [m]

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

AND A,x Logical AND immediate data to the accumulator

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

Operation: ACC ← ACC ″AND″ x

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

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

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

Operation: [m] ← ACC ″AND″ [m]

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

CALL addr Subroutine call

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

Operation: Stack ← Program Counter+1
Program Counter ← addr

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

CLR [m] Clear data memory

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

Operation: [m] ← 00H

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

CLR [m].i Clear bit of data memory

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

Operation: [m].i ← 0

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

CLR WDT Clear Watchdog Timer

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

Operation: WDT ← 00H
PDF and TO ← 0

Affected flag(s)

TO	PDF	OV	Z	AC	C
0	0	—	—	—	—

CLR WDT1 Preclear Watchdog Timer

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

Operation: WDT ← 00H*
PDF and TO ← 0*

Affected flag(s)

TO	PDF	OV	Z	AC	C
0*	0*	—	—	—	—

CLR WDT2 Preclear Watchdog Timer

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

Operation: WDT ← 00H*
PDF and TO ← 0*

Affected flag(s)

TO	PDF	OV	Z	AC	C
0*	0*	—	—	—	—

CPL [m] Complement data memory

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

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

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

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

Operation: ACC ← $[\overline{m}]$

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

DAA [m] Decimal-Adjust accumulator for addition

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

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	√

DEC [m] Decrement data memory

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

Operation: [m] ← [m]–1

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

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

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

Operation: ACC ← [m]–1

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

HALT Enter Power Down Mode

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

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
0	1	—	—	—	—

INC [m] Increment data memory

Description: Data in the specified data memory is incremented by 1

Operation: [m] ← [m]+1

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

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

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

Operation: ACC ← [m]+1

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

JMP addr Directly jump

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

Operation: Program Counter ←addr

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

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

Operation: ACC ← [m]

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

MOV A,x Move immediate data to the accumulator

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

Operation: ACC ← x

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

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

Operation: [m] ←ACC

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

NOP No operation

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

Operation: Program Counter ← Program Counter+1

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

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

Operation: ACC ← ACC ″OR″ [m]

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

OR A,x Logical OR immediate data to the accumulator

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

Operation: ACC ← ACC ″OR″ x

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

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

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

Operation: [m] ←ACC ″OR″ [m]

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

RET Return from subroutine

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

Operation: Program Counter ← Stack

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

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

Operation: Program Counter ← Stack
ACC ← x

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

RETI Return from interrupt

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

Operation: Program Counter ← Stack
EMI ← 1

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

RL [m] Rotate data memory left

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

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

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

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

RLC [m] Rotate data memory left through carry

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

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	√

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

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

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	√

RR [m] Rotate data memory right

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

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

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

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

RRC [m] Rotate data memory right through carry

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

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	√

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

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

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	√

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

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

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	√	√	√	√

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

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

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	√	√	√	√

SDZ [m] Skip if decrement data memory is 0

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

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

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

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

SET [m] Set data memory

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

Operation: [m] ← FFH

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

SET [m]. i Set bit of data memory

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

Operation: [m].i ← 1

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

SIZ [m] Skip if increment data memory is 0

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

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

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

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

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

Operation: Skip if [m].i≠0

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

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

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	√	√	√	√

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

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

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	√	√	√	√

SUB A,x Subtract immediate data from the accumulator

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

Operation $ACC \leftarrow ACC+\overline{x}+1$

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	√	√	√	√

SWAP [m] Swap nibbles within the data memory

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

Operation $[m].3\sim[m].0 \leftrightarrow [m].7\sim[m].4$

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

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

Operation $ACC.3\sim ACC.0 \leftarrow [m].7\sim[m].4$
$ACC.7\sim ACC.4 \leftarrow [m].3\sim[m].0$

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

SZ [m] Skip if data memory is 0

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

Operation: Skip if [m]=0

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

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

Operation: Skip if [m]=0

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

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

Operation: Skip if [m].i=0

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

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

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

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

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

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

Operation: ACC ← ACC ″XOR″ [m]

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

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

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

Operation: [m] ← ACC ″XOR″ [m]

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

XOR A,x Logical XOR immediate data to the accumulator

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

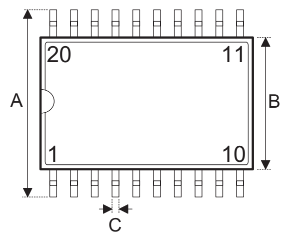
Operation: ACC ← ACC ″XOR″ x

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

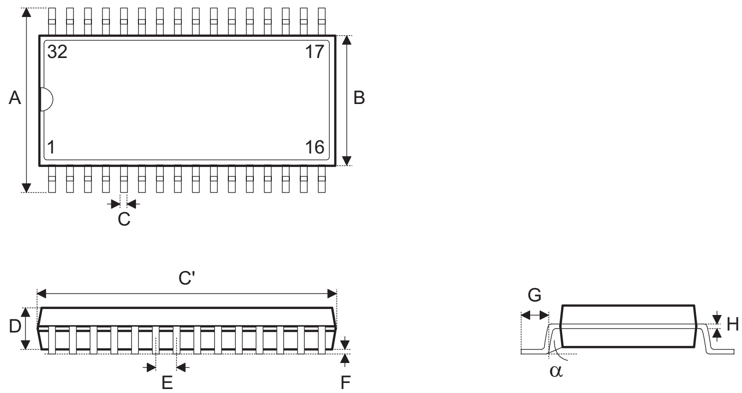
Package Information

20-pin SOP (300mil) Outline Dimensions



Symbol	Dimensions in mil		
	Min.	Nom.	Max.
A	394	—	419
B	290	—	300
C	14	—	20
C′	490	—	510
D	92	—	104
E	—	50	—
F	4	—	—
G	32	—	38
H	4	—	12
α	0°	—	10°

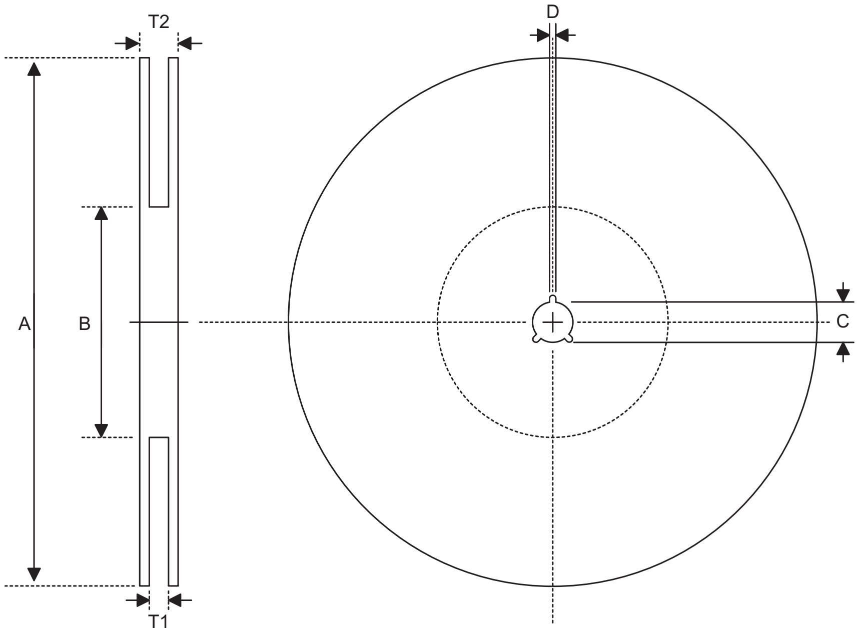
32-pin SOP (450mil) Outline Dimensions



Symbol	Dimensions in mil		
	Min.	Nom.	Max.
A	543	—	557
B	440	—	450
C	14	—	20
C′	—	—	817
D	100	—	112
E	—	50	—
F	4	—	—
G	32	—	38
H	4	—	12
α	0°	—	10°

Product Tape and Reel Specifications

Reel Dimensions



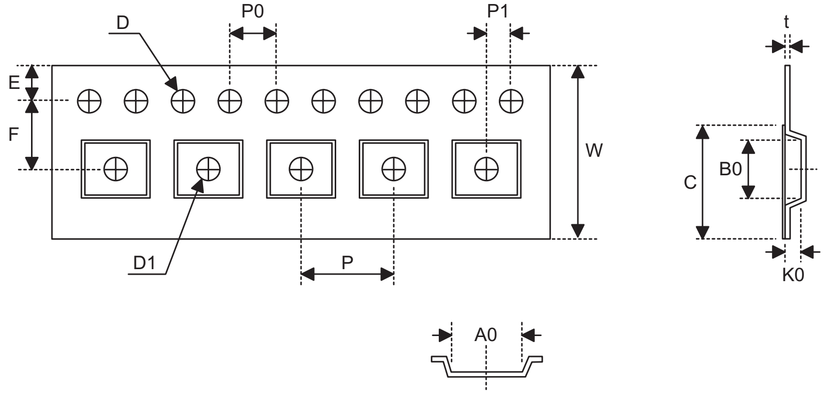
SOP 20W

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

SOP 32W

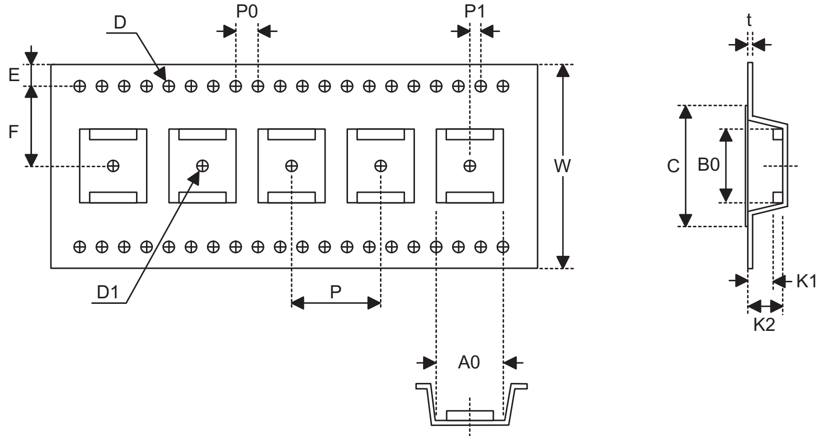
Symbol	Description	Dimensions in mm
A	Reel Outer Diameter	330±1
B	Reel Inner Diameter	100±0.1
C	Spindle Hole Diameter	13+0.5 –0.2
D	Key Slit Width	2±0.5
T1	Space Between Flange	32.8+0.3 –0.2
T2	Reel Thickness	38.2+0.2

Carrier Tape Dimensions



SOP 20W

Symbol	Description	Dimensions in mm
W	Carrier Tape Width	24+0.3 −0.1
P	Cavity Pitch	12±0.1
E	Perforation Position	1.75±0.1
F	Cavity to Perforation (Width Direction)	11.5±0.1
D	Perforation Diameter	1.5+0.1
D1	Cavity Hole Diameter	1.5+0.25
P0	Perforation Pitch	4±0.1
P1	Cavity to Perforation (Length Direction)	2±0.1
A0	Cavity Length	10.8±0.1
B0	Cavity Width	13.3±0.1
K0	Cavity Depth	3.2±0.1
t	Carrier Tape Thickness	0.3±0.05
C	Cover Tape Width	21.3



SOP 32W

Symbol	Description	Dimensions in mm
W	Carrier Tape Width	32+0.3 −0.1
P	Cavity Pitch	16±0.1
E	Perforation Position	1.75±0.1
F	Cavity to Perforation (Width Direction)	14.2±0.1
D	Perforation Diameter	1.55+0.1
D1	Cavity Hole Diameter	2+0.25
P0	Perforation Pitch	4±0.1
P1	Cavity to Perforation (Length Direction)	2±0.1
A0	Cavity Length	14.7±0.1
B0	Cavity Width	20.9±0.1
K1	Cavity Depth	3±0.1
K2	Cavity Depth	3.4±0.1
t	Carrier Tape Thickness	0.35±0.05
C	Cover Tape Width	25.5

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

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

Holtek Semiconductor Inc. (Shanghai Sales Office)
7th Floor, Building 2, No.889, Yi Shan Rd., Shanghai, China 200233
Tel: 021-6485-5560
Fax: 021-6485-0313
http://www.holtek.com.cn

Holtek Semiconductor Inc. (Shenzhen Sales Office)
43F, SEG Plaza, Shen Nan Zhong Road, Shenzhen, China 518031
Tel: 0755-8346-5589
Fax: 0755-8346-5590
ISDN: 0755-8346-5591

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

Holmate Semiconductor, Inc. (North America Sales Office)
46712 Fremont Blvd., Fremont, CA 94538
Tel: 510-252-9880
Fax: 510-252-9885
http://www.holmate.com