8-Channel Music Synthesizer 8-Bit MCU MCU

Note that the HT37B70/HT37B50/HT37B30 devices, although mentioned in this datasheet, have already been phased out and are presently no longer available.

Features

- Operating voltage:
  2.4V~5.5V for HT37B50/30
  3.0V~5.5V for HT37B90/70
- Operating frequency: 12.8MHz
- Oscillation modes for the oscillator clock
  f_{OSC}: crystal (12.8MHz)
  1-pin RC oscillation typ. 12.8MHz
- Built-in 8-bit MCU (HT-8) with 640×8 or 1280×8 (HT37B90) bits RAM
- Built-in 64K×16-bits to 512K×16-bits ROM for program/data shared
- Two 8 bit timer and one 16 bit timer
- Watchdog timer
- Power-down and Wake-up features for power saving operation
- 16-bit table read instructions for any bank/page read
- Support 32 to 40 bidirectional I/O lines
- Integrated 2-ch stereo 16-bit DAC converter
- 12-level subroutine nesting
- Integrated power amplifier
- Integrated 8-channel 12-bit A/D converter
- 16 channel polyphonic synthesizer
- Low voltage reset (Tolerance ±5%)
- External interrupt $\overline{INT}$
- External 2 Timer clock input
- 8 touch switch input
- ADPCM decoder
- IIS function for HT37B90/70
- UART function for HT37B90/70
- SPI function for HT37B90/70
- Bit manipulation instructions
- 63 powerful instructions
- All instructions in 1 or 2 machine cycles

General Description

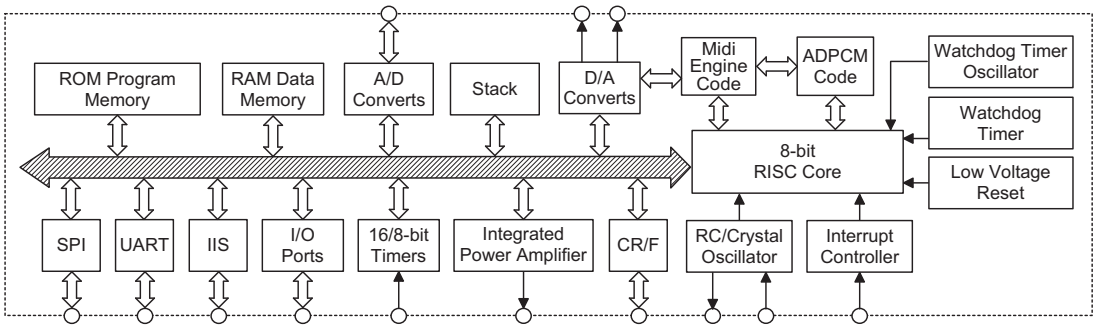
The device is an 8-bit high performance RISC architecture microcontroller specifically designed for various Music and ADPCM applications. It provides an 8-bit MCU and 16-channel wavetable synthesizer. It has a integrated 8-bit micro controller which controls the synthesizer to generate the melody by setting the special register. A Power-down function is included to reduce power consumption.

Selection Table

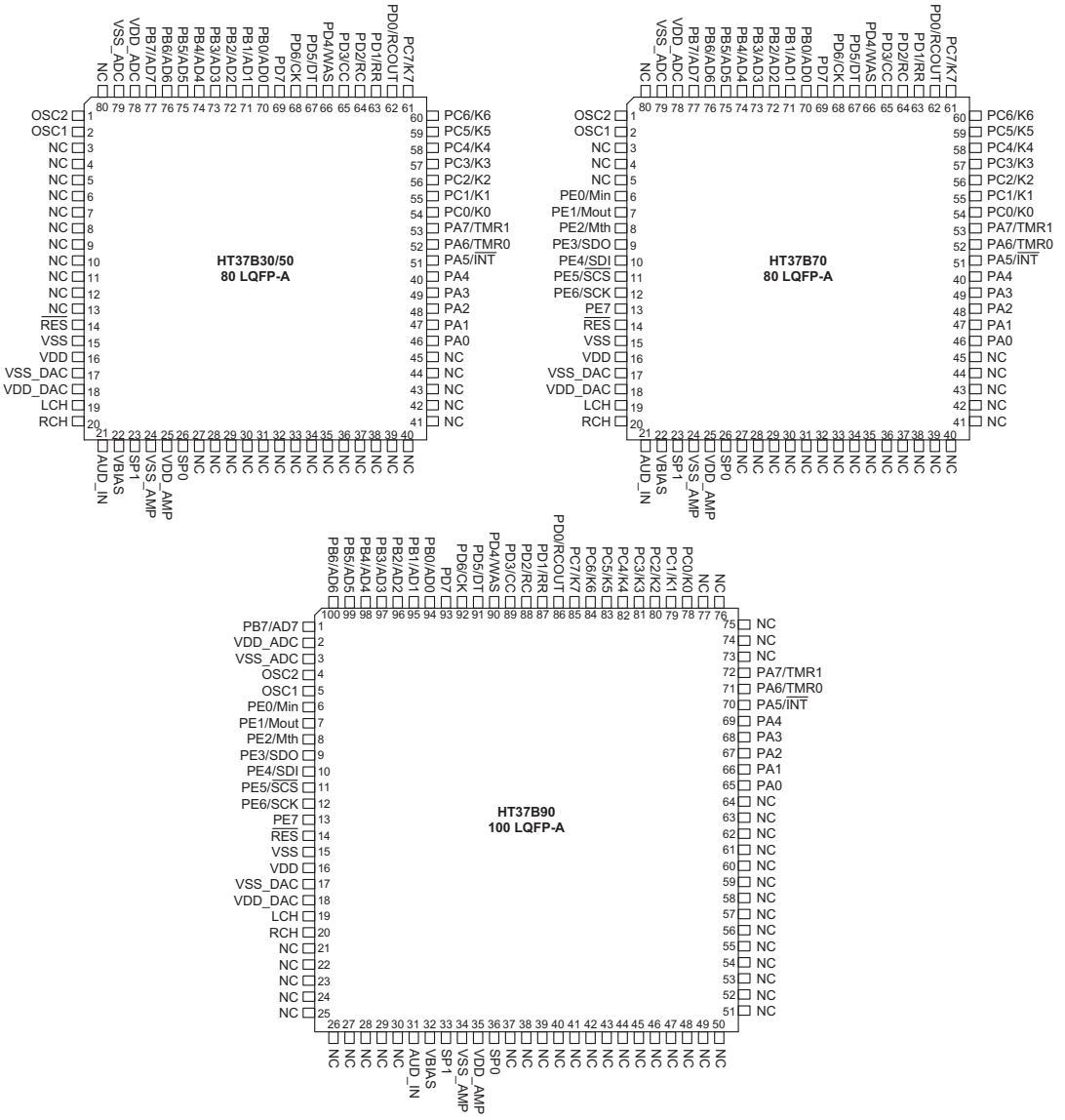
Most features are common to all devices, the main feature distinguishing them are Program Memory capacity, I/O count, A/D resolution, DAC output, R2F input and package types. The following table summarizes the main features of each device.

Part No.	VDD	Channel	OSC	Program ROM	RAM	I/O	D/A	Power AMP	CR/F	ADC	IIS	UART	SPI	Package Types
HT37B30	2.4V~5.5V	14+2	12.8MHz	64K×16	640×8	32	2ch-Stereo	√	8	12-bit×8	—	—	—	80LQFP
HT37B50				128K×16		32	2ch-Stereo	√	8		—	—	—	80LQFP
HT37B70	3.0V~5.5V			256K×16		40	2ch-Stereo	√	8		√	√	√	80LQFP
HT37B90				512K×16	1280×8	40	2ch-Stereo	√	8		√	√	√	100LQFP

Block Diagram



Pin Assignment



Pad Description

HT37B90, HT37B70

Pad Name	I/O	Configuration Option	Function
VDD	—	—	Positive digital power supply
VDD_DAC	—	—	Positive DAC circuit power supply
VDD_AMP	—	—	Positive power Amp. power supply
VDD_ADC	—	—	Positive ADC circuit power supply
VSS	—	—	Negative digital power supply, ground
VSS_DAC	—	—	Negative DAC power supply, ground
VSS_AMP	—	—	Negative AMP power supply, ground
VSS_ADC	—	—	Negative ADC power supply, ground
PA0~PA4 PA5/$\overline{INT}$ PA6/TMR0 PA7/TMR1	I/O	Wake-up, Pull-high or None	Bidirectional 8-bit input/output port. Each pin can be configured as a wake-up input by configuration option. Software instructions determine if the pin is a CMOS output or Schmitt Trigger input. Configuration options determine if all pins on this port have pull-high resistors. Pins PA5, PA6 and PA7 are pin-shared with $\overline{INT}$, TMR0 and TMR1, respectively.
PB0/AD0~ PB7/AD7	I/O	Pull-high or None	Bidirectional 8-bit input/output port. Software instructions determine if the pin is a CMOS output or Schmitt Trigger input. A configuration option determines if all pins on this port have pull-high resistors. Pins PB0 ~ PB7 are pin-shared with AD0 and AD7, respectively.
PC0/K0~ PC7/K7	I/O	Pull-high or None	Bidirectional 8-bit input/output port. Software instructions determine if the pin is a CMOS output or Schmitt Trigger input. A configuration option determines if all pins on this port have pull-high resistors. Pins PC0 ~ PC7 are pin-shared with K0 and K7, respectively (K0~K7 are CR/F function).
PD0/RCOUT PD1/RR PD2/RC PD3/CC PD4/WAS PD5/DT PD6/CK PD7	I/O	Pull-high or None	Bidirectional 8-bit input/output port. Software instructions determine if the pin is a CMOS output or Schmitt Trigger input. A configuration option determines if all pins on this port have pull-high resistors. Pins PD0~PD3 are pin-shared with R/F OSC input pins RCOUT, RR, RC and CC. PD4~PD6 are pin-shared with IIS interface pins WAS, DT and CK. PD7 is normal I/O port. RCOUT: Capacitor or resistor connection pin to RC OSC RR: Oscillation input pin RC: Reference resistor connection pin CC: Reference capacitor connection pin WAS, DT and CK control pin for IIS function WAS: IIS word select output DT: IIS data transmit output CK: IIS serial clock output
PE0/Min PE1/Mout PE2/Mth PE3/SDO PE4/$\overline{SDI}$ PE5/$\overline{SCS}$ PE6/SCK PE7	I/O	Pull-high or None	Bidirectional 8-bit input/output port. Software instructions determine if the pin is a CMOS output or Schmitt Trigger input. A configuration option determines if all pins on this port have pull-high resistors. Pins PE0~PE2 are pin-shared with UART interface pins Min, Mout and Mth. Pins PE3~PE6 are pin-shared with SPI interface pins SDO, SDI, $\overline{SCS}$ and SCK. PE7 is normal I/O port. Min, Mout and Mth control pin for UART Function Min: MIDI input Mout: MIDI output Mth: MIDI through SDO, SDI, $\overline{SCS}$ and SCK control pin for SPI Function SDO: SPI data output pin $\overline{SDI}$: SPI data input pin $\overline{SCS}$: SPI slave select signal. SCK: SPI clock
RCH	O	—	Audio right channel output

Pad Name	I/O	Configuration Option	Function
LCH	O	—	Audio left channel output
SP1, SP0	O	—	Power Amp. output pins
AUD_IN	I	—	Power Amp. input pin
VBIAS	O	—	Power Amp. voltage bias reference pin.
$\overline{RES}$	I	—	Schmitt Trigger reset input, active low
OSC1 OSC2	I O	Crystal or RC	OSC1, OSC2 are connected to an external RC network or external crystal, determined by configuration option, for the internal system clock. If the RC system clock option is selected, pin OSC2 can be used to measure the system clock at 1/8 frequency.

Note: 1. Each pin on PA can be programmed through a configuration option to have a wake-up function.

2. Individual pins can be selected to have pull-high resistors.

3. Because the two timers are used by MIDI the external timer pin functions are disabled.

HT37B50, HT37B30

Pad Name	I/O	Configuration Option	Function
VDD	—	—	Positive digital power supply
VDD_DAC	—	—	Positive DAC circuit power supply
VDD_AMP	—	—	Positive power Amp. power supply
VDD_ADC	—	—	Positive ADC circuit power supply
VSS	—	—	Negative digital power supply, ground
VSS_DAC	—	—	Negative DAC power supply, ground
VSS_AMP	—	—	Negative AMP power supply, ground
VSS_ADC	—	—	Negative ADC power supply, ground
PA0~PA4 PA5/$\overline{INT}$ PA6/TMR0 PA7/TMR1	I/O	Pull-high Wake-up	Bidirectional 8-bit input/output port. Each pin can be configured as a wake-up input by configuration option. Software instructions determine if the pin is a CMOS output or Schmitt Trigger input. Configuration options determine if all pins on this port have pull-high resistors. Pins PA5, PA6 and PA7 are pin-shared with $\overline{INT}$, TMR0 and TMR1, respectively.
PB0/AD0~ PB7/AD7	I/O	Pull-high	Bidirectional 8-bit input/output port. Software instructions determine if the pin is a CMOS output or Schmitt Trigger input. A configuration option determines if all pins on this port have pull-high resistors. Pins PB0 ~ PB7 are pin-shared with AD0 and AD7, respectively.
PC0/K0~ PC7/K7	I/O	Pull-high	Bidirectional 8-bit input/output port. Software instructions determine if the pin is a CMOS output or Schmitt Trigger input. A configuration option determines if all pins on this port have pull-high resistors. Pins PC0 ~ PC7 are pin-shared with K0 and K7, respectively (K0~K7 are CR/F function).
PD0/RCOUT PD1/RR PD2/RC PD3/CC PD4~PD7	I/O	Pull-high	Bi-directional 4-bit I/O port. Software instructions determined the CMOS output or Schmitt trigger with a pull-high resistor (determined by pull-high option: by option). Pins PD0~PD3 are pin-shared with CR/F OSC input pins RCOUT, RR, RC and CC. RCOUT, RR, RC and CC control pin for CR/F Function. Pins PD4, PD5, PD6 and PD7 are normal IO.
RCH	O	—	Audio right channel output
LCH	O	—	Audio left channel output
SP1, SP0	O	—	Power Amp. output pins

Pad Name	I/O	Configuration Option	Function
AUD_IN	I	—	Power Amp. input pin
VBIAS	O	—	Power Amp. voltage bias reference pin.
$\overline{RES}$	I	—	Schmitt Trigger reset input, active low
OSC1 OSC2	I O	Crystal or RC	OSC1, OSC2 are connected to an external RC network or external crystal, determined by configuration option, for the internal system clock. If the RC system clock option is selected, pin OSC2 can be used to measure the system clock at 1/8 frequency.

Note: 1. Each pin on PA can be programmed through a configuration option to have a wake-up function.

2. Individual pins can be selected to have pull-high resistors.

3. Because the two timers are used by MIDI the external timer pin functions are disabled.

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
I_{OL} Total 150mA
Total Power Dissipation 500mW

Storage Temperature .....–50°C to 125°C
Operating Temperature .....–40°C to 85°C
I_{OH} Total .....–100mA

Note: These are stress ratings only. Stresses exceeding the range specified under "Absolute Maximum Ratings" may cause substantial damage to the device. Functional operation of this device at other conditions beyond those listed in the specification is not implied and prolonged exposure to extreme conditions may affect device reliability.

D.C. Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	Operating Voltage	—	f_{OSC}=12.8MHz (37B50/30)	2.4	3.0	5.5	V
			f_{OSC}=12.8MHz (37B90/70)	3.0	—	5.5	V
I_{DD}	Operating Current (Crystal OSC or RC OSC)	3V	No load, f_{OSC}=8MHz~12.8MHz, DAC disable	—	8	12	mA
		5V		—	20	30	mA
I_{STB1}	Standby Current (WDT Disable)	3V	No load, system HALT, WDT disable	—	—	1	μA
		5V		—	—	2	μA
I_{STB2}	Standby Current (WDT Enable)	3V	No load, system HALT, WDT enable	—	—	5	μA
		5V		—	—	10	μA
V_{IL1}	Input Low Voltage for I/O Ports	—	—	0	—	$0.3V_{DD}$	V
V_{IH1}	Input High Voltage for I/O Ports	—	—	$0.7V_{DD}$	—	V_{DD}	V
V_{IL2}	Input Low Voltage ($\overline{RES}$)	—	—	0	—	$0.4V_{DD}$	V
V_{IH2}	Input High Voltage ($\overline{RES}$)	—	—	$0.9V_{DD}$	—	V_{DD}	V
I_{OL}	I/O Port Segment Logic Output Sink Current	3V	V_{OL}=0.1V_{DD}	6	12	—	mA
		5V		10	25	—	mA
I_{OH}	I/O Port Segment Logic Output Source Current	3V	V_{OH}=0.9V_{DD}	–2	–4	—	mA
		5V		–5	–8	—	mA

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
R_{PH}	Pull-high Resistance of I/O Ports	3V	—	20	60	100	kΩ
		5V	—	10	30	50	kΩ
V_{LVR}	Low Voltage Reset Voltage	3V	LVR 2.4V option	2.28	2.40	2.52	V
		5V	LVR 3.0V option	2.85	3.00	3.15	V
V_{LVD}	Low Voltage Detector Voltage	3V	LVD 2.5V option	2.375	2.500	2.625	V
		5V	LVD 3.1V option	2.945	3.100	3.255	V
I_O	AUD Current Source	—	V_{OH}=0.9V_{DD}	—	–3	—	mA
I_{ADC}	Additional Power Consumption if A/D Converter is Used	3V	—	—	0.5	1	mA
		5V		—	1.5	3	mA
P_O	Internal AMP Output Power	3V	(THD+N)/S≤1%, R_L=8Ω V_{IN}=1kHz Sinewave	—	90	—	mW
			(THD+N)/S≤10%, R_L=8Ω V_{IN}=1kHz Sinewave	—	125	—	mW
		5V	(THD+N)/S≤1%, R_L=8Ω V_{IN}=1kHz Sinewave	—	385	—	mW
			(THD+N)/S≤10%, R_L=8Ω V_{IN}=1kHz Sinewave	—	490	—	mW

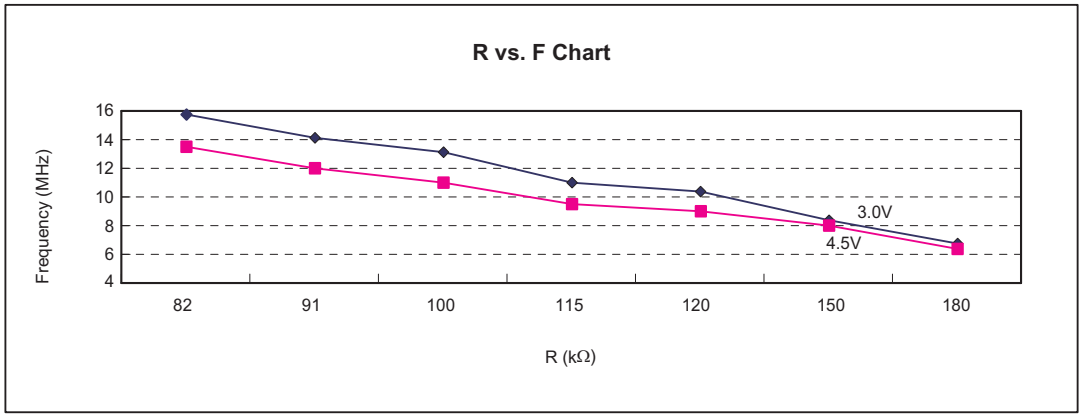
A.C. Characteristics

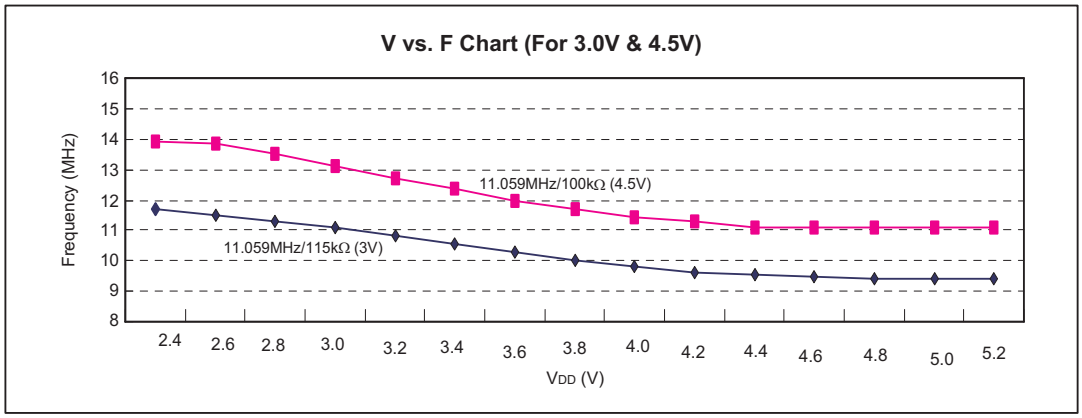
Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
f_{OSC}	Oscillator Clock (Crystal OSC/RC OSC)	—	2.4V~5.5V	8000	11059	12800	kHz
t_{WDTOSC}	Watchdog Oscillator Period	3V	—	45	90	180	μs
		5V	—	32	65	130	μs
t_{RES}	External Reset Low Pulse Width	—	—	1	—	—	μs
t_{SST}	System Start-up Timer Period	—	Power-up or wake-up from HALT	—	1024	—	t_{SYS}
t_{LVR}	Low Voltage Width to Reset	—	—	0.25	1.00	2.00	ms

Note: t_{SYS}= 1/f_{SYS}
f_{SYS}=f_{OSC}/2

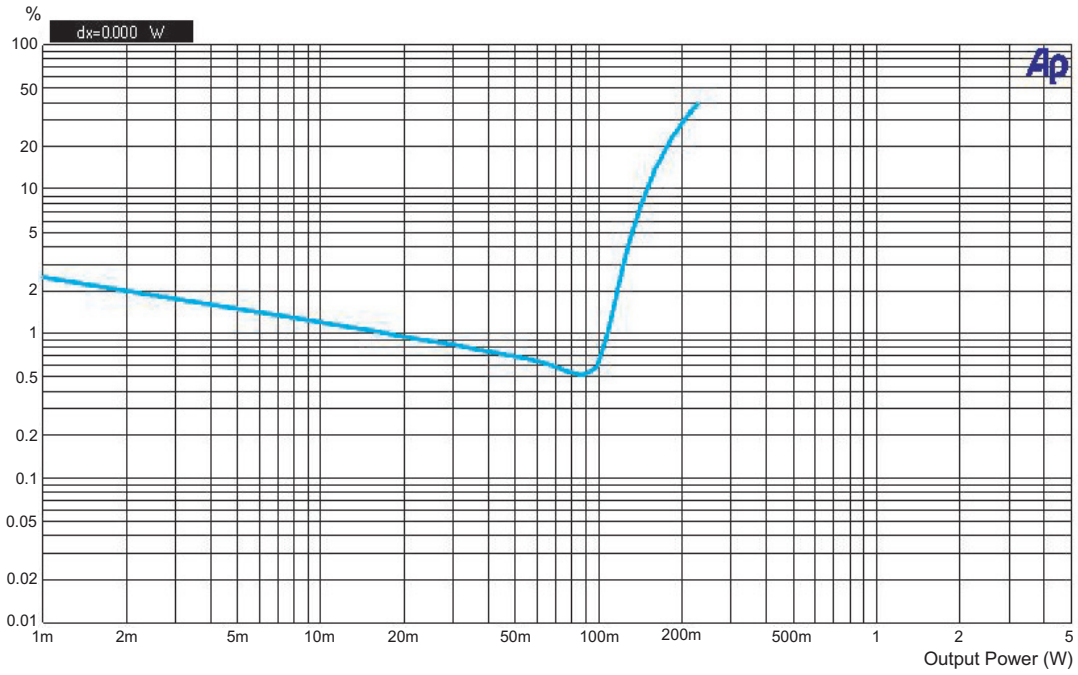
Characteristics Curves



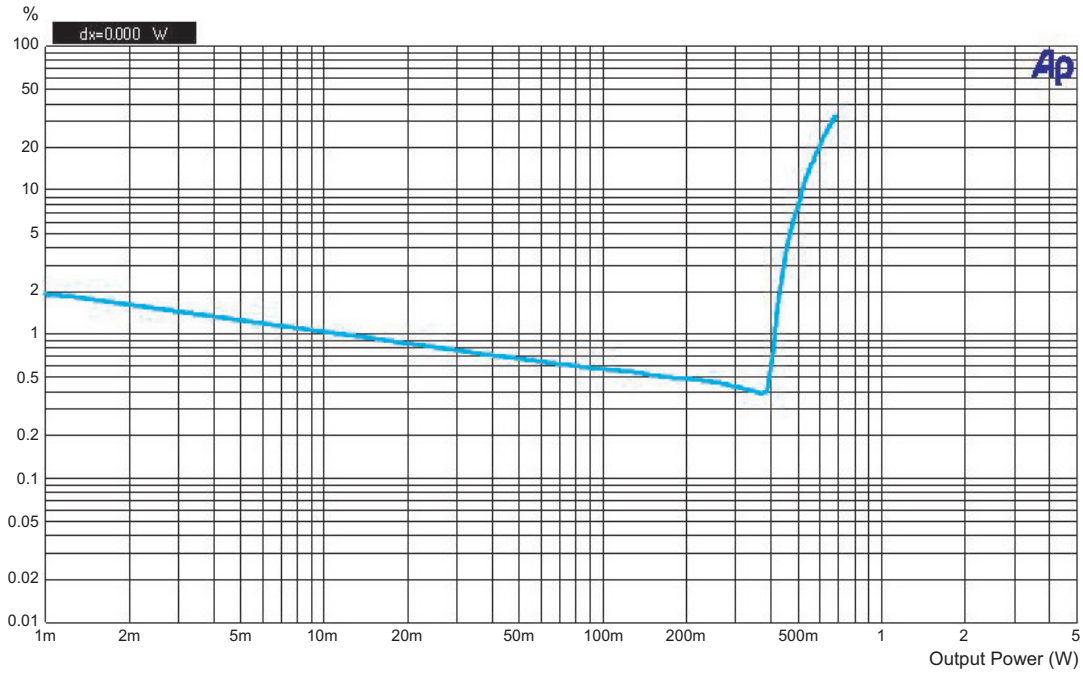


(THD+N) vs. Output Power

R_{LOAD}=8Ω, V_{IN}=1kHz Sinewave for 3.0V



R_{LOAD}=8Ω, V_{IN}=1kHz Sinewave for 5.0V



System Architecture

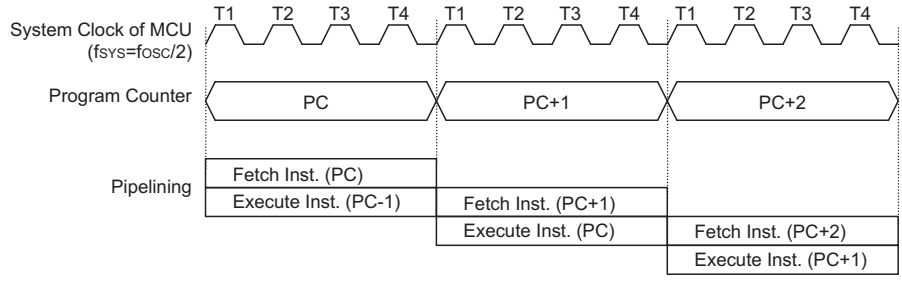
A key factor in the high-performance features of the Holtek range of Music Type microcontrollers is attributed to the internal system architecture. The range of devices take advantage of the usual features found within RISC microcontrollers providing increased speed of operation and enhanced performance. The pipelining scheme is implemented in such a way that instruction fetching and instruction execution are overlapped, hence instructions are effectively executed in one cycle, with the exception of branch or call instructions. An 8-bit wide ALU is used in practically all operations of the instruction set. It carries out arithmetic operations, logic operations, rotation, increment, decrement, branch decisions, etc. The internal data path is simplified by moving data through the Accumulator and the ALU. Certain internal registers are implemented in the Data Memory and can be directly or indirectly addressed. The simple addressing methods of these registers along with additional architectural features ensure that a minimum of external components is required to provide a functional I/O and A/D control system with maximum reliability and flexibility.

Clocking and Pipelining

The main system clock, derived from either a Crystal/Resonator or RC oscillator. The oscillator frequency divided by 2 is subdivided into four internally generated non-overlapping clocks, T1~T4. The Program Counter is incremented at the beginning of the T1 clock during which time a new instruction is fetched. The remaining T2~T4 clocks carry out the decoding and execution functions. In this way, one T1~T4 clock cycle forms one instruction cycle. Although the fetching and execution of instructions takes place in consecutive instruction cycles, the pipelining structure of the microcontroller ensures that instructions are effectively executed in one instruction cycle. The exception to this are instructions where the contents of the Program Counter are changed, such as subroutine calls or jumps, in which case the instruction will take one more instruction cycle to execute. When the RC oscillator is used, OSC2 is freed for use as a T1 phase clock synchronizing pin. This T1 phase clock has a frequency of $f_{OSC}/8$ with a 1:3 high/low duty cycle. For instructions involving branches, such as jump or call instructions, two machine cycles are required to complete instruction execution. An extra cycle is required as the program takes one cycle to first obtain the actual jump or call address and then another cycle to actually execute the branch. The requirement for this extra cycle should be taken into account by programmers in timing sensitive applications.

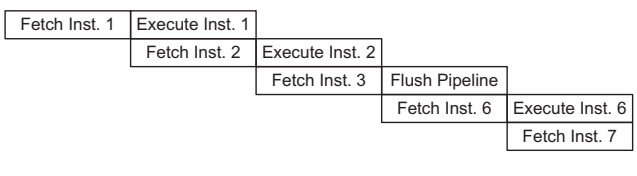
Program Counter

During program execution, the Program Counter is used to keep track of the address of the next instruction to be executed. It is automatically incremented by one each time an instruction is executed except for instructions, such as "JMP" or "CALL", that demand a jump to a non-consecutive Program Memory address. Note that the Program Counter width varies with the Program Memory capacity depending upon which device is selected. However, it must be noted that only the lower 8 bits, known as the Program Counter Low Register, are directly addressable by user.



System Clocking and Pipelining

```
1          MOV A,[12H]
2          CALL DELAY
3          CPL [12H]
4          :
5          :
6  DELAY:  NOP
```



Instruction Fetching

When executing instructions requiring jumps to non-consecutive addresses such as a jump instruction, a subroutine call, interrupt or reset, etc., the microcontroller manages program control by loading the required address into the Program Counter. For conditional skip instructions, once the condition has been met, the next instruction, which has already been fetched during the present instruction execution, is discarded and a dummy cycle takes its place while the correct instruction is obtained.

The lower byte of the Program Counter, known as the Program Counter Low register or PCL, is available for program control and is a readable and writable register. By transferring data directly into this register, a short program jump can be executed directly, however, as only this low byte is available for manipulation, the jumps are limited to the present page of memory, that is 256 locations. When such program jumps are executed it should also be noted that a dummy cycle will be inserted.

The lower byte of the Program Counter is fully accessible under program control. Manipulating the PCL might cause program branching, so an extra cycle is needed to pre-fetch. Further information on the PCL register can be found in the Special Function Register section.

Stack

This is a special part of the memory which is used to save the contents of the Program Counter only. The stack can have 12 levels depending upon which option is selected and is neither part of the data nor part of the program space, and is neither readable nor writable. The activated level is indexed by the Stack Pointer, SP, and is neither readable nor writable. At a subroutine call or interrupt acknowledge signal, the contents of the Program Counter are pushed onto the stack. At the end of a subroutine or an interrupt routine, signaled by a return instruction, RET or RETI, the Program Counter is restored to its previous value from the stack. After a device reset, the Stack Pointer will point to the top of the stack.

Mode	Program Counter													
	b18~b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Initial Reset	000000	0	0	0	0	0	0	0	0	0	0	0	0	0
Timer/Event Counter 0 Overflow	000000	0	0	0	0	0	0	0	0	0	1	0	0	0
Timer/Event Counter 1 Overflow	000000	0	0	0	0	0	0	0	0	0	1	1	0	0
Timer Counter 2 Overflow (UART)	000000	0	0	0	0	0	0	0	0	1	0	0	0	0
ERCOCI Interrupt	000000	0	0	0	0	0	0	0	0	1	0	1	0	0
ADPCM Interrupt	000000	0	0	0	0	0	0	0	0	1	1	0	0	0
Skip	Program Counter + 2 (Within Current Bank)													
Loading PCL	P18~P13	P12	P11	P10	P9	P8	@7	@6	@5	@4	@3	@2	@1	@0
Jump, Call Branch	BP1.5~BP1.0	#12	#11	#10	#9	#8	#7	#6	#5	#4	#3	#2	#1	#0
Return from Subroutine	S18~S13	S12	S11	S10	S9	S8	S7	S6	S5	S4	S3	S2	S1	S0

Program Counter

Note: P18~P8: Program Counter bits

@7~@0: PCL bits

#12~#0: Instruction code address bits

BP1.5~BP1.0: ROM bank pointer

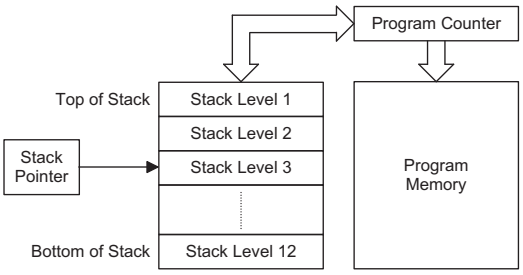
S18~S0: Stack register bits

For the HT37B90, the Program Counter is 19 bits wide, i.e. from b18~b0.

For the HT37B70, the Program Counter is 18 bits wide, i.e. from b17~b0, therefore the b18 column in the table is not applicable.

For the HT37B50, the Program Counter is 17 bits wide, i.e. from b16~b0, therefore the b17 and b18 column in the table is not applicable.

For the HT37B30, the Program Counter is 16 bits wide, i.e. from b15~b0, therefore the b18, b17 and b16 the columns in the table are not applicable.



If the stack is full and an enabled interrupt takes place, the interrupt request flag will be recorded but the acknowledge signal will be inhibited. When the Stack Pointer is decremented, by RET or RETI, the interrupt will be serviced. This feature prevents stack overflow allowing the programmer to use the structure more easily. However, when the stack is full, a CALL subroutine instruction can still be executed which will result in a stack overflow. Precautions should be taken to avoid such cases which might cause unpredictable program branching.

Arithmetic and Logic Unit – ALU

The arithmetic-logic unit or ALU is a critical area of the microcontroller that carries out arithmetic and logic operations of the instruction set. Connected to the main microcontroller data bus, the ALU receives related instruction codes and performs the required arithmetic or logical operations after which the result will be placed in the specified register. As these ALU calculation or operations may result in carry, borrow or other status changes, the status register will be correspondingly updated to reflect these changes. The ALU supports the following functions:

- Arithmetic operations: ADD, ADDM, ADC, ADCM, SUB, SUBM, SBC, SBCM, DAA
- Logic operations: AND, OR, XOR, ANDM, ORM, XORM, CPL, CPLA
- Rotation RRA, RR, RRCA, RRC, RLA, RL, RLCA, RLC
- Increment and Decrement INCA, INC, DECA, DEC
- Branch decision, JMP, SZ, SZA, SNZ, SIZ, SDZ, SIZA, SDZA, CALL, RET, RETI

Program Memory

The Program Memory is the location where the user code or program is stored. The type of memory is the mask ROM memory. It offer the most cost effective solutions for high volume products.

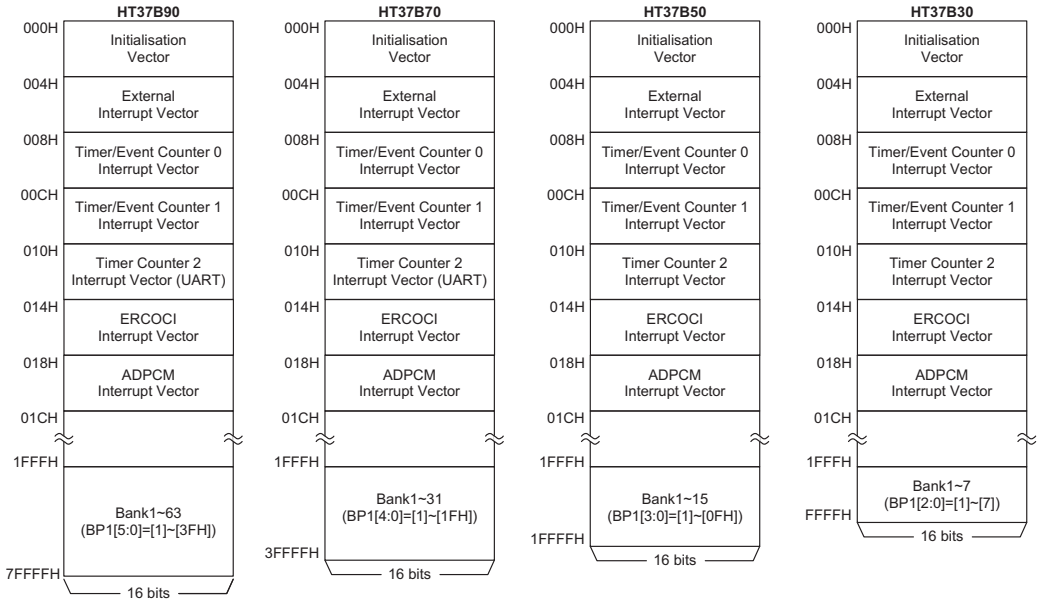
Structure

The Program Memory has a capacity of 512K by 16, 256K by 16, 128K by 16 or 64K by 16 bits depending upon which device is selected.

Special Vectors

Within the Program Memory, certain locations are reserved for special usage such as reset and interrupts.

- Location 000H
  This vector is reserved for use by the device reset for program initialization. After a device reset is initiated, the program will jump to this location and begin execution.
- Location 004H
  This vector is used by the external interrupt. If the external interrupt pin on the device goes low, the program will jump to this location and begin execution if the external interrupt is enabled and the stack is not full.
- Location 008H
  This vector is reserved for the Timer/Event Counter 0 interrupt service program. If a timer interrupt results from a Timer/Event Counter 0 overflow, and if the interrupt is enabled and the stack is not full, the program begins execution at location 008H.
- Location 00CH
  This vector is reserved for the Timer/Event Counter 1 interrupt service program. If a timer interrupt results from a Timer/Event Counter 1 overflow, and if the interrupt is enabled and the stack is not full, the program begins execution at location 00CH.
- Location 010H
  This vector is reserved for the Timer Counter 2 interrupt service program. If a timer interrupt results from a Timer Counter 2 overflow, and if the interrupt is enabled and the stack is not full, the program begins execution at location 0010H. Timer 2 Counter overflow interrupt share with UART interrupt require. Using the UART interrupt require is defined by enabled UART function enable configuration option.
- Location 014H
  This vector is reserved for the ERCOCI interrupt service program. If an external RC oscillation converter interrupt results from an external RC oscillation converter interrupt is activated, and the stack is not full, the program begins execution at location 0014H.
- Location 018H
  This vector is reserved for the Adpcm interrupt service program. If a Adpcm interrupt results, and if the interrupt is enabled and the stack is not full, the program begins execution at location 0018H.



Program Memory Structure

Look-up Table

Any location within the Program Memory can be defined as a look-up table where programmers can store fixed data. To use the look-up table, the table pointer must first be setup by placing the address of the look up data to be retrieved in the three table pointer registers, TBLP, TBMP and TBHP. This three registers define the address of the look-up table. After setting up the table pointer, the table data can be retrieved from the current Program Memory or last Program Memory page in the specific bank which defined by bank point register as BP1 using the "TABRDC[m]" or "TABRDL [m]" instructions, respectively. When these instructions are executed, the lower order table byte from the Program Memory will be transferred to the user defined Data Memory register [m] as specified in the instruction. The higher order table data byte from the Program Memory will be transferred to the TBLH special register. Any unused bits in this transferred higher order byte will have uncertain values.

The following diagram illustrates the addressing/data flow of the look-up table:

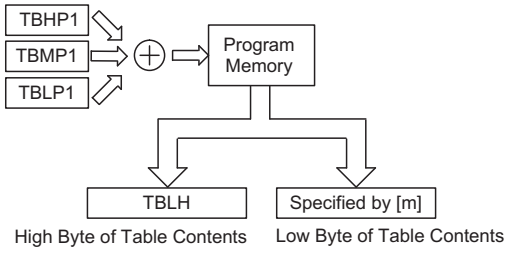


Table Program Example

The following example shows how the table pointer and table data is defined and retrieved from the HT37B90/70/50/30 microcontroller. This example uses raw table data located in the last page of ROM Bank 1 which is stored there using the ORG and ROMbank statement. The location at program ROM "3F00H" which refers to the start address of the last page within the Program Memory of the HT37B90/70/50/30 microcontroller. The table pointer is setup here to have an initial value of "06H". This will ensure that the first data read from the data table will be at the Program Memory address "3F06H" or 6 locations after the start of the last page in selected ROMbank. The high byte of the table data which in this case is equal to zero will be transferred to the TBLH register automatically when the "TABRDL [m]" instruction is executed. Because the TBLH register is a read-only register and cannot be restored, care should be taken to ensure its protection if both the main routine and Interrupt Service Routine use table read instructions. If using the table read instructions, the Interrupt Service Routines may change the value of the TBLH and subsequently cause errors if used again by the main routine. As a rule it is recommended that simultaneous use of the table read instructions should be avoided. However, in situations where simultaneous use cannot be avoided, the interrupts should be disabled prior to the execution of any main routine table-read instructions. Note that all table related instructions require two instruction cycles to complete their operation.

Instruction	Table Location Bits									
	b18~b13	b12~b8	b7	b6	b5	b4	b3	b2	b1	b0
TABRDC [m]	TBHP1_2~TBMP1_5	TBMP1_4~TBMP1_0	@7	@6	@5	@4	@3	@2	@1	@0
TABRDL [m]	BP1_5~BP1_0	11111	@7	@6	@5	@4	@3	@2	@1	@0

Table Location

Note: @7~@0: Table pointer lower-order bits are TBLP1 [7:0]
b17~b0: Current program ROM table address A [18:0]
TBMP1_4~TBMP1_0: TBMP1 bit 4 ~0
TBHP1_2~TBMP1_5: TBHP1 (bit 2 ~0) to TBMP1 (bit7 ~5)
BP1_5 ~BP1_0: Bits of bank BP1 bit0~5
For the HT37B90, the Table address location is 19 bits wide, i.e. from b18~b0.
For the HT37B70, the Table address location is 18 bits wide, i.e. from b17~b0.
For the HT37B50, the Table address location is 17 bits wide, i.e. from b16~b0.
For the HT37B30, the Table address location is 16 bits wide, i.e. from b15~b0.

```
tempreg1 db    ?        ; temporary register #1
tempreg2 db    ?        ; temporary register #2
tempreg3 db    ?        ; temporary register #3
tempreg4 db    ?        ; temporary register #4

    :
    :
mov a,01h               ; set ROM bank 1 point
mov bp1,a
mov a,06h               ; initialise table pointer
mov tblp1,a             ; to the last page
clr tbmpl
clr tbhpl

    :
    :

tabrdl   tempreg1       ; transfers value in table referenced by table pointer
                        ; to tempreg1
                        ; data at prog. memory address "3F06H" transferred to
                        ; tempreg1 and TBLH

dec tblp1               ; reduce value of table pointer by one

tabrdl   tempreg2       ; transfers value in table referenced by table pointer
                        ; to tempreg2
                        ; data at prog.memory address "3F05H" transferred to
                        ; tempreg2 and TBLH
                        ; in this example the data "1AH" is transferred to
                        ; tempreg1 and data "0FH" to register tempreg2
                        ; the value "00H" will be transferred to the high byte
                        ; register TBLH
    :
    :

mov a,04h               ; initialise table pointer low byte
mov tblp1,a             ;
mov a,3Fh               ; initialise table pointer middle byte

mov tbmp1,a
mov a,00h               ; initialise table pointer high byte

mov tbhp1,a
tabrdc   tempreg3       ;

:
:

rombank 1  romsumvalue1; sets rombank 1 initial address of last page
                        ;(for HT37B90/70/50/30)

romsumvalue1 .section at 1F00h 'code'


dc   00Ah, 00Bh, 00Ch, 00Dh, 00Eh, 00Fh, 01Ah, 01Bh
    :
    :
```

ROM Bank Pointer (2DH)

The program memory is organized into 64/32/16/8 banks for HT37B90/70/50/30 and each bank into 8K×16 bits of program ROM. BP1.7~BP1.0 is used as the bank pointer. After an instruction has been executed to write data to the BP1 register to select a different bank, note that the new bank will not be selected immediately. It is until the instruction "JMP" or "CALL" or interrupt has completed execution that the bank will be actually selected.

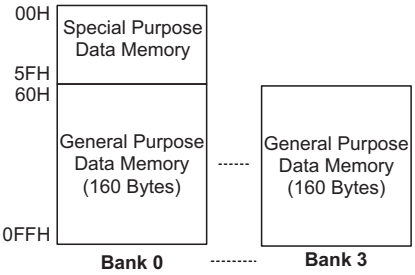
Register	Bit No.	Function
BP1 (2DH)	0~7	00000000b= Select ROM Bank0 (0000h~1FFFh) 00000001b= Select ROM Bank1 (2000h~3FFFh) 00000010b= Select ROM Bank2 (4000h~5FFFh) 00000011b= Select ROM Bank3 (6000h~7FFFh) : 00011110b= Select ROM Bank30 (3C000h~3DFFFh) 00011111b= Select ROM Bank31 (3E000h~3FFFFh) 00111110b= Select ROM Bank62 (7C000h~7DFFFh) 00111111b= Select ROM Bank63 (7E000h~7FFFFh)

Note: For the HT37B90, the ROM bank point register is 6 bits wide effectively, i.e. from b5~b0.
For the HT37B70, the ROM bank point register is 5 bits wide effectively, i.e. from b4~b0.
For the HT37B50, the ROM bank point register is 4 bits wide effectively, i.e. from b3~b0.
For the HT37B30, the ROM bank point register is 3 bits wide effectively, i.e. from b2~b0.

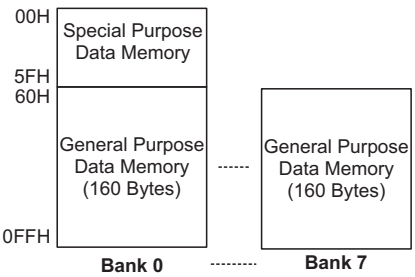
RAM Data Memory

The RAM Data Memory is a volatile area of 8-bit wide RAM internal memory and is the location where temporary information is stored. Divided into two sections, the first of these is an area of RAM where special function registers are located. These registers have fixed locations and are necessary for correct operation of the device.

Many of these registers can be read from and written to directly under program control, however, some remain protected from user manipulation. The second area of RAM Data Memory is reserved for general purpose use. All locations within this area are read and write accessible under program control.



Bank 0~Bank 3 RAM Data Memory Structure - HT37B70/50/30



Bank 0~Bank 7 RAM Data Memory Structure - HT37B90

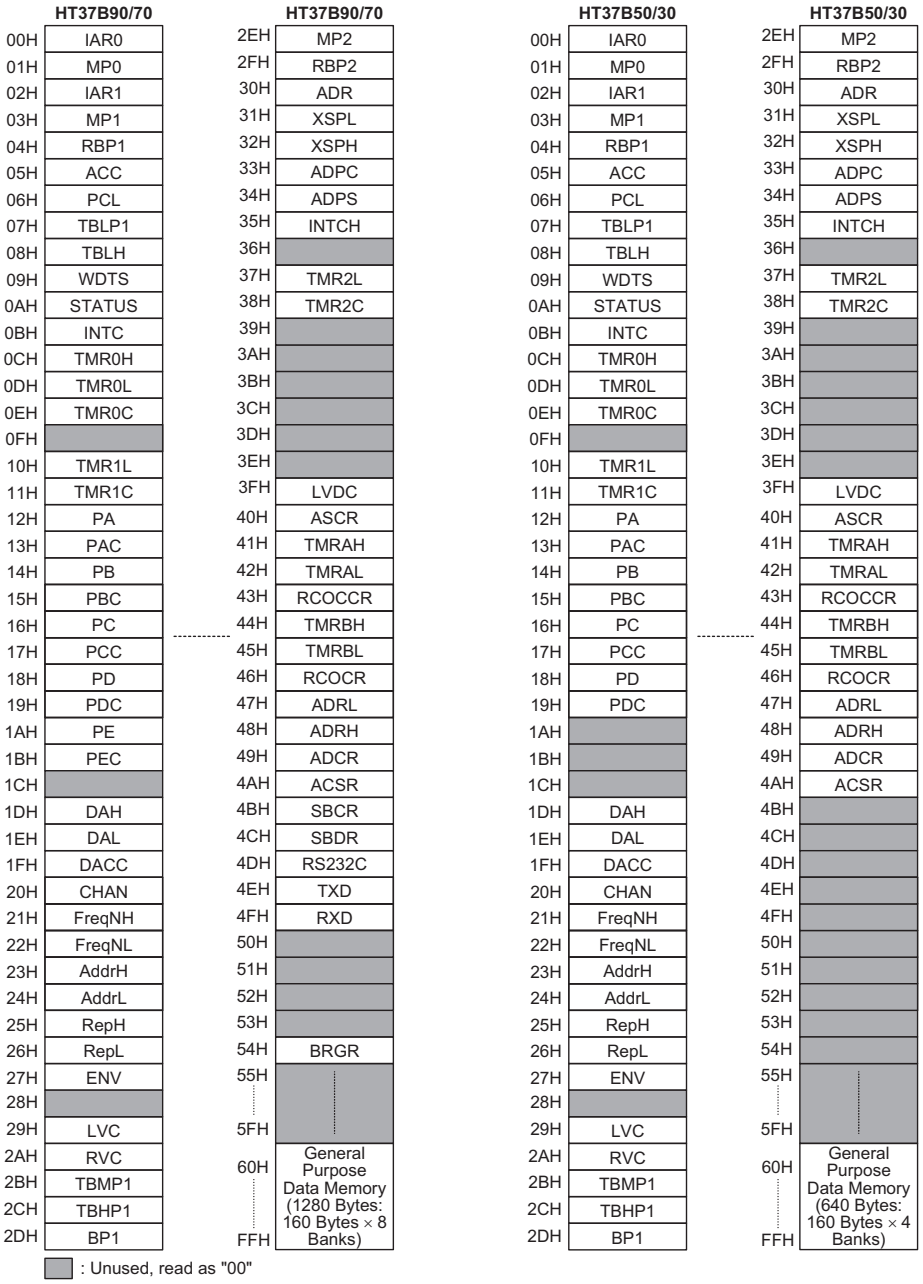
Structure

The RAM Data Memory is subdivided into 4 or 8 banks, known as Bank 0 to Bank 3 or Bank 7, all of which are implemented in 8-bit wide RAM. Most of the RAM Data Memory is located in Bank 0 which is also subdivided into two sections, the Special Purpose Data Memory and the General Purpose Data Memory. The start address of the Data Memory is the address "00H". Registers which are common to all microcontrollers, such as ACC, PCL, etc., have the same Data Memory address. Bank 1 of the RAM Data Memory is located at address "60H".

The RAM data memory is designed with 640×8 bits or 1280×8 bits with 4 or 8 RAM banks. There are two RAM BANK pointers (RBP1 and RBP2) control Bank 0~7.

The data memory is designed with 256 bytes and divided into five functional groups: special function registers (00H~1FH), music synthesis controller registers (20H~2FH), ADPCM decoder controller register (30H~35H), the other function (35H~5FH) and general purpose data memory (60H~FFH).

They are also indirectly accessible through Memory



Data Memory Structure

pointer registers MP0, MP1and MP2, where MP1/MP2 can deal with all banks of data memory but MP0 deal with Bank0 data memory only.

General Purpose Data Memory

All microcontroller programs require an area of read/write memory where temporary data can be stored and retrieved for use later. It is this area of RAM memory that is known as General Purpose Data Memory. This area of Data Memory is fully accessible by the user program for both read and write operations. The bank 0 data memory areas can handle arithmetic, logic, increment, decrement and rotate operations directly. By using the ″SET [m].i″ and ″CLR [m].i″ instructions individual bits can be set or reset under program control giving the user a large range of flexibility for bit manipulation in the Data Memory.

Special Purpose Data Memory

This area of Data Memory is where registers, necessary for the correct operation of the microcontroller, are stored. such as timers, interrupts, etc., as well as external functions such as I/O data control and A/D converter operation. Most of the registers are both readable and writable but some are protected and are readable only, the details of which are located under the relevant Special Function Register section. Note that for locations that are unused, any read instruction to these addresses will return the value ″00H″. Although the Special Purpose Data Memory registers are located in Bank 0, they will still be accessible even if the Bank Pointer has selected Bank 1.

Special Function Registers

To ensure successful operation of the microcontroller, certain internal registers are implemented in the RAM Data Memory area. These registers ensure correct operation of internal functions such as timers, interrupts, watchdog, etc., as well as external functions such as I/O data control. The location of these registers within the RAM Data Memory begins at the address ″00H″.

Indirect Addressing Register – IAR0, IAR1

The Indirect Addressing Registers, IAR0 and IAR1, although having their locations in normal RAM register space, do not actually physically exist as normal registers. The method of indirect addressing for RAM data manipulation uses these Indirect Addressing Registers and Memory Pointers, in contrast to direct memory addressing, where the actual memory address is specified. Actions on the IAR0 and IAR1 registers will result in no actual read or write operation to these registers but rather to the memory location specified by their corresponding Memory Pointer, MP0 or MP1/MP2. Acting as a pair, IAR0 and MP0 can together only access data from Bank 0, while the IAR1 and MP1/MP2 register can access data form Bank 0 to Bank 7. Using MP1 or MP2 are selected by DACC.7. As the Indirect Addressing Registers are not physically implemented, reading the Indirect Addressing Registers indirectly will return a result of ″00H″ and writing to the registers indirectly will result in no operation.

Memory Pointer – MP0, MP1, MP2

Three Memory Pointers, known as MP0, MP1 and MP2 are provided. These Memory Pointers are physically implemented in the Data Memory and can be manipulated in the same way as normal registers providing a convenient way with which to address and track data. When any operation to the relevant Indirect Addressing Registers is carried out, the actual address that the microcontroller is directed to, is the address specified by the related Memory Pointer. MP0, together with Indirect Addressing Register, IAR0, are used to access data from Bank 0 only, while MP1/MP2 and IAR1 are used to access data form Bank 0 to Bank 7. Using MP1 or MP2 are selected by DACC.7.

Example

The following example shows how to clear General Purpose Data Memory of bank0 by using MP0 and bank0~bank1 by using MP1 and MP2

```
code .section at 0 code
org 00h
RAM0TEST:
    MOV  A,60H
    MOV  MP0,A          ; loaded with first RAM address
LOOP0:
    CLR  IAR0           ; clear the data at address defined by MP0
    CLR  WDT
    SIZ  MP0            ; increase MP0, and skip out if MP0 is "0"
    JMP  LOOP0
    :
RAM1TEST:
    CLR  DACC.7         ; access data to iar1 by MP1
    CLR  rBP1           ; clear RAM bank pointer 1
RAM1_MP1:
    MOV  A,rBP1         ; load rBP1 data, and check if rBP1 is "25"
    XOR  A,25
    SZ   ZERO           ; jump to exit loop if rBP1 is "2"
    JMP  RAM1TEST_Exit
    MOV  A,60H          ; loaded with first RAM address to MP1
    MOV  MP1,A
LOOP1:
    CLR  WDT
    CLR  IAR1           ; clear the data at address defined by MP1
    SIZ  MP1            ; increase MP1, and skip out if MP1 is "0"
    JMP  LOOP1
    INC  rBP1           ; increase rBP1
    JMP  RAM1_MP1
RAM1TEST_Exit:
    :
RAM2TEST:
    Set  dacc.7         ; access data to iar1 by MP2
    CLR  rBP2           ; clear RAM bank pointer 2

RAM1MP2:
    MOV  A,RBP2
    XOR  A,25
    SZ   ZERO           ; jump to exit loop if rBP2 is "2"
    JMP  RAM2TEST_Exit
    MOV  A,60H          ; loaded with first RAM address to MP2
    MOV  MP2,A
LOOP2:
    CLR  WDT
    CLR  IAR1           ; clear the data at address defined by MP2
    SIZ  MP2            ; increase MP2, and skip out if MP2 is "0"
    JMP  LOOP2
    INC  rBP2           ; increase rBP2
    JMP  RAM1MP2
RAM2TEST_Exit:
    :
```

Bank Pointer – RBP1, RBP2

The RAM Data Memory is divided into 8 Banks, known as Bank 0 to Bank 7. Selecting the required Data Memory area is achieved using the RAM Bank Pointers which are RBP1 and RBP2. The RBP1 and RBP2 match up with MP1 and MP2 respectively. If data in Bank 0 is to be accessed, then the RBP registers must be loaded with the value ″00″, while if data in Bank 1 is to be accessed, then the RBP registers must be loaded with the value ″01″.

Register IAR0 will always access data from Bank 0, irrespective of the value of the Bank Pointer. The RBP1 and RBP2 register is located at memory location 60H in Bank 0 to Bank 7 and can only be accessed indirectly using two memory pointers MP1 and MP2 and the indirect addressing register IAR1 will always access data from Bank 0 to Bank 7.

The Data Memory is initialized to Bank 0 after a reset, except for the WDT time-out reset in the Power Down Mode, in which case, the Data Memory bank remains unaffected. It should be noted that Special Function Data Memory is not affected by the bank selection, which means that the Special Function Registers can be accessed from within Bank 0 to Bank 7. Directly addressing the Data Memory will always result in Bank 0 being accessed irrespective of the value of the Bank Pointer.

Register	Bit No.	Function
RBP1	0~2	RAM Bank Point 1 Select 000= Select RAM Bank0 001= Select RAM Bank1 010= Select RAM Bank2 011= Select RAM Bank3 100= Select RAM Bank4 101= Select RAM Bank5 110= Select RAM Bank6 111= Select RAM Bank7
	3~4	General bits. Can write and read.
	5~7	Unused bit

RBP1 (04H)

Register	Bit No.	Function
RBP2	0~2	RAM Bank Point 2 Select 000= Select RAM Bank0 001= Select RAM Bank1 010= Select RAM Bank2 011= Select RAM Bank3 100= Select RAM Bank4 101= Select RAM Bank5 110= Select RAM Bank6 111= Select RAM Bank7
	3~4	General bits. Can write and read.
	5~7	Unused bit

RBP2 (2FH)

Note: Using MP1 or MP2 are selected by DACC.7.

Accumulator – ACC

The Accumulator is central to the operation of any microcontroller and is closely related with operations carried out by the ALU. The Accumulator is the place where all intermediate results from the ALU are stored. Without the Accumulator it would be necessary to write the result of each calculation or logical operation such as addition, subtraction, shift, etc., to the Data Memory resulting in higher programming and timing overheads. Data transfer operations usually involve the temporary storage function of the Accumulator; for example, when transferring data between one user defined register and another, it is necessary to do this by passing the data through the Accumulator as no direct transfer between two registers is permitted.

Program Counter Low Register – PCL

To provide additional program control functions, the low byte of the Program Counter is made accessible to programmers by locating it within the Special Purpose area of the Data Memory. By manipulating this register, direct jumps to other program locations are easily implemented. Loading a value directly into this PCL register will cause a jump to the specified Program Memory location, however, as the register is only 8-bit wide, only jumps within the current Program Memory page are permitted. When such operations are used, note that a dummy cycle will be inserted.

Look-up Table Registers – TBLP1, TBMP1, TBHP1, TBLH

These seven special function registers are used to control operation of the look-up table which is stored in the Program Memory. TBLP1, TBMP1 and TBHP1 are the table pointer and indicate the location where the table data is located. Their value must be setup before any table read commands are executed. Their value can be changed, for example using the ″INC″ or ″DEC″ instructions, allowing for easy table data pointing and reading. TBLH is the location where the high order byte of the table data is stored after a table read data instruction has been executed. Note that the lower order table data byte is transferred to a user defined location.

Watchdog Timer Register – WDTS

The Watchdog feature of the microcontroller provides an automatic reset function giving the microcontroller a means of protection against spurious jumps to incorrect Program Memory addresses. To implement this, a timer is provided within the microcontroller which will issue a reset command when its value overflows. To provide variable Watchdog Timer reset times, the Watchdog Timer clock source can be divided by various division ratios, the value of which is set using the WDTS register. By writing directly to this register, the appropriate division ratio for the Watchdog Timer clock source can be

setup. Note that only the lower 3 bits are used to set division ratios between 1 and 128.

Status Register – STATUS

This 8-bit register contains the zero flag (Z), carry flag (C), auxiliary carry flag (AC), overflow flag (OV), power down flag (PDF), and watchdog time-out flag (TO). These arithmetic/logical operation and system management flags are used to record the status and operation of the microcontroller.

With the exception of the TO and PDF flags, bits in the status register can be altered by instructions like most other registers. Any data written into the status register will not change the TO or PDF flag. In addition, operations related to the status register may give different results due to the different instruction operations. The TO flag can be affected only by a system power-up, a WDT time-out or by executing the ″CLR WDT″ or ″HALT″ instruction. The PDF flag is affected only by executing the ″HALT″ or ″CLR WDT″ instruction or during a system power-up.

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

- **C** is set if an operation results in a carry during an addition operation or if a borrow does not take place during a subtraction operation; otherwise C is cleared. C is also affected by a rotate through carry instruction.
- **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.
- **Z** is set if the result of an arithmetic or logical operation is zero; otherwise Z is cleared.
- **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.
- **PDF** is cleared by a system power-up or executing the ″CLR WDT″ instruction. PDF is set by executing the ″HALT″ instruction.
- **TO** is cleared by a system power-up or executing the ″CLR WDT″ or ″HALT″ instruction. TO is set by a WDT time-out.

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

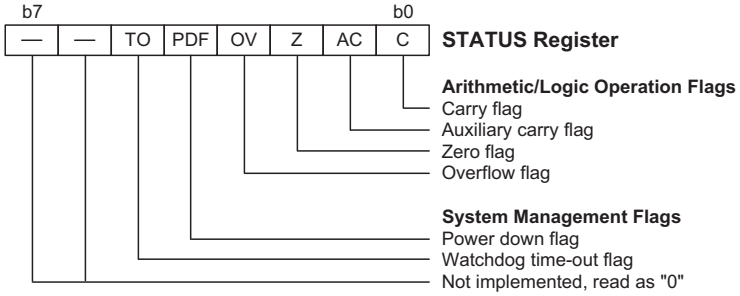
Interrupt Control Registers – INTC, INTCH

The two 8-bit registers, known as the INTC and INTCH register which control the operation of both external and internal timer, UART, CR/F and ADPCM interrupts, and By setting various bits within this register using standard bit manipulation instructions, the enable/disable function of the external and timer, UART, CR/F and ADPCM interrupts can be independently controlled. A master interrupt bit within this register, the EMI bit, acts like a global enable/disable and is used to set all of the interrupt enable bits on or off. This bit is cleared when an interrupt routine is entered to disable further interrupt and is set by executing the ″RETI″ instruction.

Note: In situations where other interrupts may require servicing within present interrupt service routines, the EMI bit can be manually set by the program after the present interrupt service routine has been entered.

Timer/Event Counter Registers – TMR0H, TMR0L, TMR1L, TMR2L, TMR0C, TMR1C, TMR2C

HT37B90/70/50/30 contains two 8-bit and a 16-bit Timer/Event Counters which has an associated register known as TMR0H and TMR0L. are the location where the timer's 16-bit value is located.TMR1L and TMR2L are the location where the timer's 8-bit value is located. An associated control register, known as TMR0C, TMR1C and TMR2C contains the setup information for the timer.



Status Register

Input/Output Ports and Control Registers – PA, PB, PC, PD, PE, PAC, PBC, PCC, PDC, PEC

Within the area of Special Function Registers, the I/O registers and their associated control registers play a prominent role. All I/O ports have a designated register correspondingly labeled as PA, PB, PC, PD and PE. These labeled I/O registers are mapped to specific addresses within the Data Memory as shown in the Data Memory table, which are used to transfer the appropriate output or input data on that port. with each I/O port there is an associated control register labeled PAC, PBC, PCC, PDC and PEC, also mapped to specific addresses with the Data Memory. The control register specifies which pins of that port are set as inputs and which are set as outputs. To setup a pin as an input, the corresponding bit of the control register must be set high, for an output it must be set low. During program initialization, it is important to first setup the control registers to specify which pins are outputs and which are inputs before reading data from or writing data to the I/O ports. One flexible feature of these registers is the ability to directly program single bits using the "SET [m].i" and "CLR [m].i" instructions. The ability to change I/O pins from output to input and vice versa by manipulating specific bits of the I/O control registers during normal program operation is a useful feature of these devices.

D/A Converter Registers – DAH, DAL, DACC

HT37B90/70/50/30 provide two 16-bit D/A converters, which can select stereo or mono output. The correct operation of the D/A requires the use of two data registers, and a control register. It contain a 16-bit D/A converter, there are two data registers, a high byte data register known as DAH, and a low byte data register known as DAL. These are the register locations where the digital value is placed before the completion of a digital to analog conversion cycle. The configuration of the D/A converter is setup via the control register DACC.

Wavetable Function Registers – CHAN, FreqNH, FreqNL, RepH, RepL, ENV, LVC, RVC

HT37B90/70/50/30 contains Wavetable synthesizer Function. The HT37B90/70/50/30 has a built-in 16 output channels. CHAN is channel number selection. FreqNH and FreqNL are used to define the output speed of the PCM file.

AddrH and AddrL is setup for the start address of the PCM code before Wavetable function implement. The repeat number register as known RepH and RepL are 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. It provides the left and right volume control independently. The 10-bit left and right volume are controlled by ENL, LVC and RVC respectively. The ENV contain both left and right volume some bit of high byte.

ADPCM Function Registers – ADR, XSPL, XSPH, ADPC, ADPS

HT37B90/70/50/30 contains ADPCM Decoder Function. The must set initial value of register known as XSPL and XSPH before implementing ADPCM Decoder procedure. There are two 4-bit ADPCM encode data of ADR. The data of ADR implement via ADPCM Decoder, and output 8-bit PCM data which is synthesized by MIDI synthesizer.

The ADPC is the control register for the ADPCM Decoder. The ADPS is the status register for the ADPCM Decoder.

CR/F Converter Registers – ASCR, TMRAH, TMRAL, RCOCCR, TMRBH, TMRBL, RCOCR

There are 8 analog switch lines in the microcontroller for K0~K7 for HT37B90/70/50/30 Analog Switch control registers known as ASCR. The RC oscillation converter contains two 16-bit programmable count-up counters and the Timer A clock source may come from the system clock ($f_{SYS}=f_{OSC}/2$) or system clock/4 ($f_{OSC}/8$). There are two data registers, a high byte data register known as TMRAH, and a low byte data register known as TMRAL. The timer B clock source may come from the external RC oscillator. There are two data registers, a high byte data register known as TMRBH, and a low byte data register known as TMRBL. There are two control and status registers known as RCOCCR and RCOCR.

A/D Converter Registers – ADRL, ADRH, ADCR, ACSR

HT37B90/70/50/30 contains a 8-channel 12-bit A/D converter. The correct operation of the A/D requires the use of two data registers, a control register and a clock source register. It contain a 12-bit A/D converter, there are two data registers, a high byte data register known as ADRH, and a low byte data register known as ADRL. These are the register locations where the digital value is placed after the completion of an analog to digital conversion cycle. The channel selection and configuration of the A/D converter is setup via the control register ADCR while the A/D clock frequency is defined by the clock source register, ACSR.

SPI Registers – SBCR, SBDR

The device contain an internal SPI function which is controlled via these two registers. The SBCR is the status and control register for the SPI function. The actual data that is to be transmitted or that is received on the serial interface is stored in the SBDR register.

UART Registers – RS232C, TXD, RXD, BRGR

The device contain an internal UART function which is controlled via these four registers. The RS232C is the status and control register for the UART .The actual data that is to be transmitted or that is received on the serial interface is stored in the TXD/RXD register. The BRGR register set to generates UART baud rate clock 31.25kHz according to fOSC.

Input/Output Ports

Holtek microcontrollers offer considerable flexibility on their I/O ports. With the input or output designation of every pin fully under user program control, pull-high options for all ports and wake-up options on certain pins, the user is provided with an I/O structure to meet the needs of a wide range of application possibilities. Depending upon which device or package is chosen, the microcontroller range provides from 32 to 40 bidirectional input/output lines labeled with port names PA, PB, PC, PD and PE. These I/O ports are mapped to the RAM Data Memory with specific addresses as shown in the Special Purpose Data Memory table. All of these I/O ports can be used for input and output operations. For input operation, these ports are non-latching, which means the inputs must be ready at the T2 rising edge of instruction "MOV A,[m]", where m denotes the port address.

For output operation, all the data is latched and remains unchanged until the output latch is rewritten.

Pull-high Resistors

Many product applications require pull-high resistors for their switch inputs usually requiring the use of an external resistor. To eliminate the need for these external resistors, I/O pins PA~PE, when configured as an input have the capability of being connected to an internal pull-high resistor. These pull-high resistors are selectable via PA~PE option respectively, located in the configuration. The pull-high resistors are implemented using weak PMOS transistors.

Port A Wake-up

If the HALT instruction is executed, the device will enter the Power Down Mode, where the system clock will stop resulting in power being conserved, a feature that is important for battery and other low-power applications.

Various methods exist to wake-up the microcontroller, one of which is to change the logic condition on one of the PA0~PA7 pins from high to low. After a HALT instruction forces the microcontroller into entering the Power Down Mode, the processor will remain idle or in a low-power state until the logic condition of the selected wake-up pin on Port A changes from high to low. This function is especially suitable for applications that can be woken up via external switches. Note that pins PA0 to PA7 can be selected individually to have this wake-up feature using an PA wake up option, located in the configuration.

I/O Port Control Registers

Each I/O port have their own control register, known as PAC, PBC, PCC, PDC and PEC, which control the input/output configuration. With this control register, each PA~PE I/O pin with or without pull-high resistors can be reconfigured by pull-hi option control. Pins PA~PE ports are directly mapped to a bit in its associated port control register. For the I/O pin to function as an input, the corresponding bit of the control register must be written as a "1". This will then allow the logic state of the input pin to be directly read by instructions. When the corresponding bit of the control register is written as a "0", the I/O pin will be setup as a CMOS output. If the pin is currently setup as an output, instructions can still be used to read the output register.

However, it should be noted that the program will in fact only read the status of the output data latch and not the actual logic status of the output pin.

- Pin-shared Functions
  The flexibility of the microcontroller range is greatly enhanced by the use of pins that have more than one function. Limited numbers of pins can force serious design constraints on designers but by supplying pins with multi-functions, many of these difficulties can be overcome. For some pins, the chosen function of the multi-function I/O pins is set by configuration options while for others the function is set by application program control.
- External Interrupt Input
  The external interrupt pin, $\overline{INT}$, is pin-shared with the I/O pin PA5. To use the pin as an external interrupt input the correct bits in the PA share pin option must be selected. The pin must also be setup as an input by setting the appropriate bit in the Port Control Register. A pull-high resistor can also be selected via the appropriate port pull-high option.

- A/D Inputs
  The HT37B90/70/50/30 have 8 A/D converter channel inputs. All of these analog inputs are pin-shared with PB0 to PB7. If these pins are to be used as A/D inputs and not as normal I/O pins then the corresponding bits in the A/D Converter Control Register, ADCR.3~5 and ADSR.4, must be properly set. There are no configuration options associated with the A/D function. If used as I/O pins, then full pull-high resistor selections remain, however if used as A/D inputs then any pull-high resistor selections associated with these pins will be automatically disconnected.

- CR/F analog switch Inputs
  The HT37B90/70/50/30 have 8 CR/F converter inputs. All of these analog inputs are pin-shared with PC0 to PC7. If these pins are to be used as CR/F analog switch Inputs and not as normal I/O pins then the corresponding bits in the Option, ″PC0~7 share pin configuration″. All of these analog inputs are pin-shared with PC0 to PC7. If these pins are to be used as CR/F analog switch Inputs and not as normal I/O pins then the corresponding bits in the configuration, ″PC0~7 share pin configuration″.

- CR/F oscillator pin
  The HT37B90/70/50/30 have 4 CR/F oscillator pins. All of these CR/F oscillator pin are pin-shared with PD0 to PD3. If these pins are to be used as CR/F oscillator pins and not as normal I/O pins then the corresponding bits in the Option, ″PD0~3 share pin Option″.

- UART pin
  The device have 3 UART pins. All of these UART pins are pin-shared with PE0 to PE2. If these pins are to be used as UART pins and not as normal I/O pins then the corresponding bits in the PE share pin configuration option.

- SPI pin
  The device have 4 SPI pins. All of these SPI pins are pin-shared with PE3 to PE6. If these pins are to be used as SPI pins and not as normal I/O pins then the corresponding bits in the PE share pin configuration option.

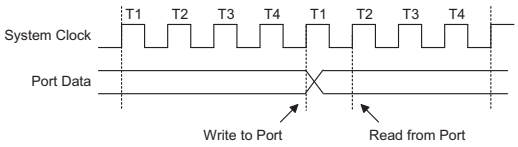
I/O Pin Structures

The diagrams illustrate the I/O pin internal structures. As the exact logical construction of the I/O pin may differ from these drawings, they are supplied as a guide only to assist with the functional understanding of the I/O pins.

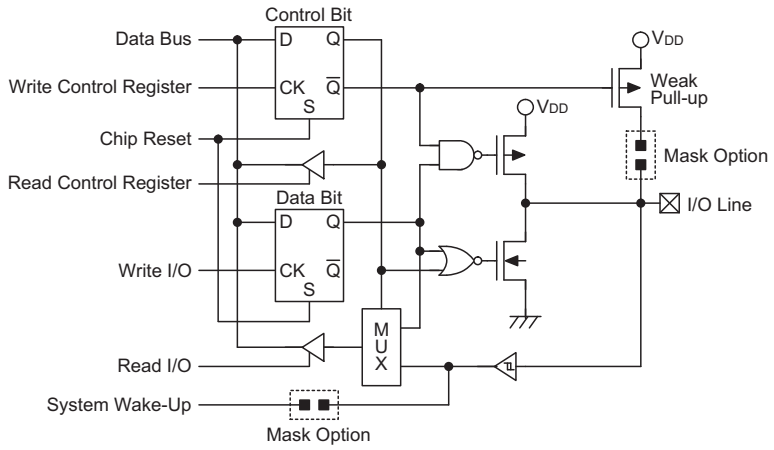
Programming Considerations

Within the user program, one of the first things to consider is port initialization. After a reset, the PA~PE data register and PAC~PEC port control register will be set high. This means that all I/O pins will default to an input state, the level of which depends on the other connected circuitry and whether pull-high options have been selected. If the PAC port control register, is then programmed to setup some pins as outputs, these output pins will have an initial high output value unless the associated PA port data register is first programmed. Selecting which pins are inputs and which are outputs can be achieved byte-wide by loading the correct value into the port control register or by programming individual bits in the port control register using the ″SET [m].i″ and ″CLR [m].i″ instructions.

Note that when using these bit control instructions, a read-modify-write operation takes place. The microcontroller must first read in the data on the entire port, modify it to the required new bit values and then rewrite this data back to the output ports.



Read/Write Timing



Input/Output Port

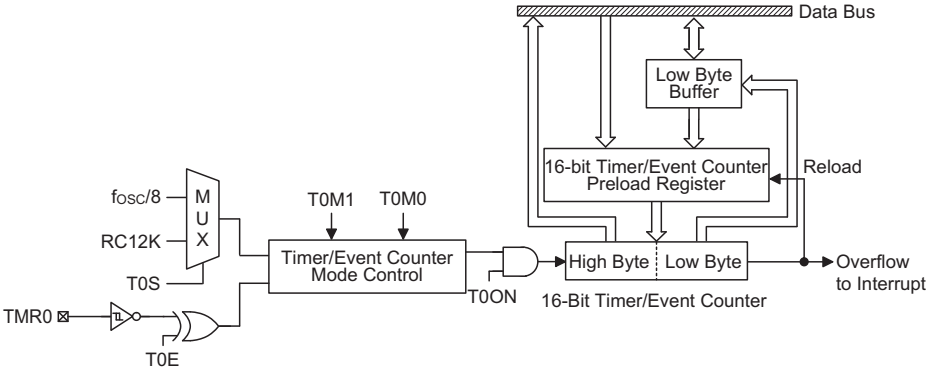
Timer/Event Counters

The provision of timers form an important part of any microcontroller, giving the designer a means of carrying out time related functions. The devices contain two count-up timers 8-bit capacity and one count-up timers 16-bit capacity. As the timer 0/1 has three different operating modes, they can be configured to operate as a general timer, an external event counter or as a pulse width measurement device. But the timer 2 only be configured to operate as a general timer. The provision of an internal prescaler to the clock circuitry of some of the timer/event counters gives added range to the timer 1/2.

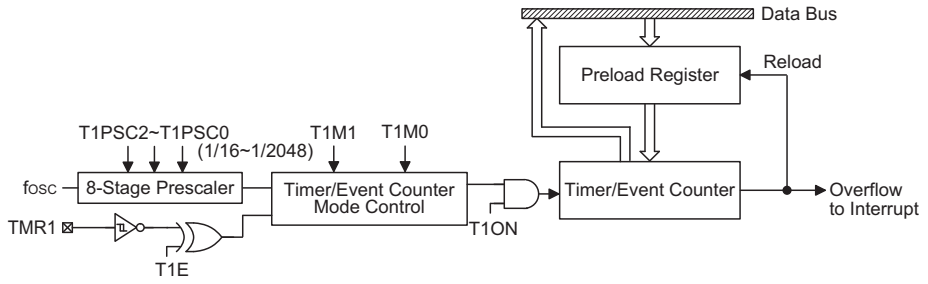
There are three types of registers related to the Timer/Event Counters 0. The first two register contain the actual high and low byte value of the timer and into which an initial value can be preloaded.

There are two types of registers related to the Timer/Event Counters 1/2. The first is the register that contains the actual value of the timer and into which an initial value can be preloaded. Reading from this register retrieves the contents of the Timer/Event Counter. The second type of associated register is the Timer Control Register which defines the timer options and determines how the timer is to be used. The Timer/Event Counter 0 can have the timer clock configured to come from the internal clock source. The clock source is $f_{OSC}/8$.In addition, the timer clock source of Timer/Event Counter 0 can also be configured to come from an internal RC 12kHz.

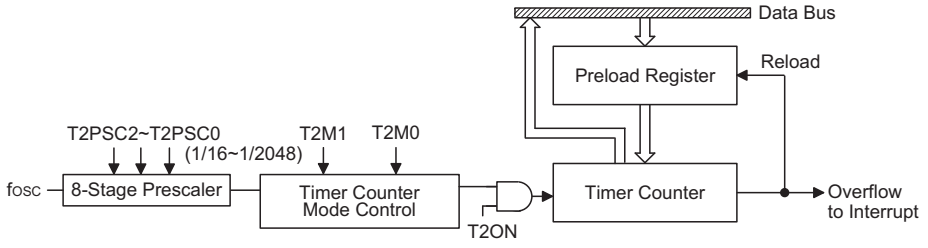
An external clock source is used when the timer is in the event counting mode, the clock source being provided on the external timer pin, known as TMR0 or TMR1. These external timer pins are pin-shared with other I/O pins. Depending upon the condition of PA share pin option, each high to low, or low to high transition on the external timer input pin will increment the counter by one.



16-bit Timer/Event Counter 0 Structure



8-bit Timer/Event Counter 1 Structure



8-bit Timer Counter 2 Structure

Configuring the Timer/Event Counter Input Clock Source

The internal timer′s clock can originate from various sources, depending upon timer is chosen. The system clock input timer source is used when the timer is in the timer mode or in the pulse width measurement mode.

For Timer/Event Counter 0, these system clock timer source is selected by TMR0C.5.

For Timer/Event Counter 1, 2 this system clock timer source is first divided by a prescaler, the division ratio of which is conditioned by the Timer Control Register bits T1PSC0~T1PSC2.

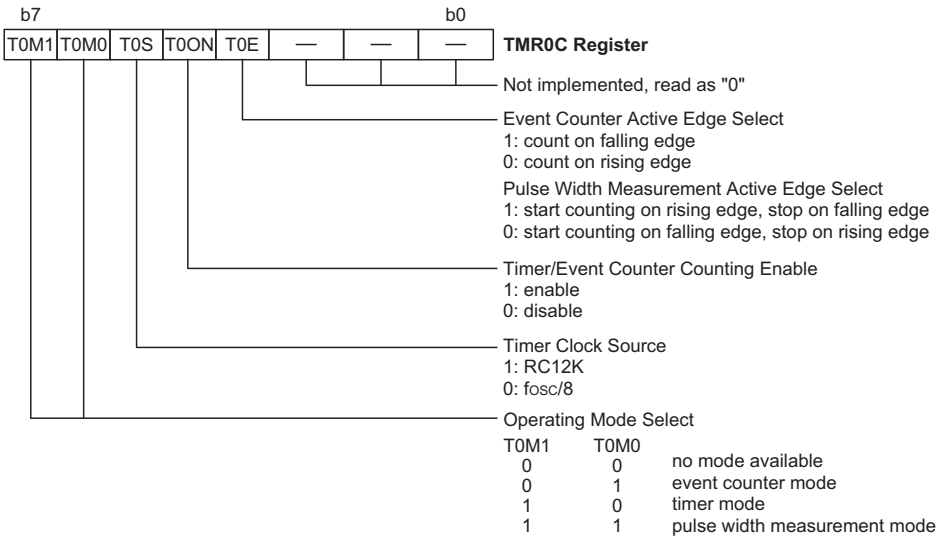
An external clock source is used when the timer is in the event counting mode, the clock source being provided on the external timer pin, known as TMR0 or TMR1. These external timer pins are pin-shared with other I/O pins. Depending upon the condition of PA share pin option, each high to low, or low to high transition on the external timer input pin will increment the counter by one.

Timer Registers – TMR0H/TMR0L, TMR1, TMR2

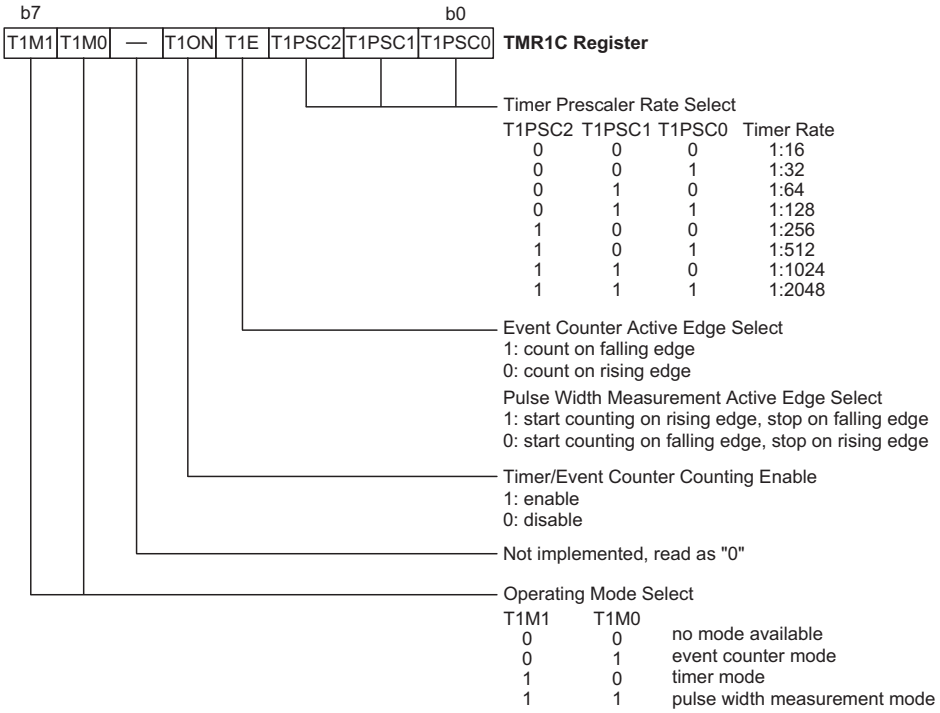
The timer registers are special function registers located in the special purpose Data Memory and is the place where the actual timer value is stored. For the 8-bit timer, this register is known as Timer/Event Counter 1/2. In the case of the 16-bit timer, a pair of 8-bit registers are required to store the 16-bit timer values. These are known as TMR1L/TMR1H. The value in the timer registers increases by one each time an internal clock pulse is received or an external transition occurs on the external timer pin. The timer will count from the initial value loaded by the preload register to the full count of FFH for the 8-bit timer or FFFFH for the 16-bit timers, at which point the timer overflows and a timer internal interrupt signal is generated. The timer value will then be reset with the initial preload register value and continue counting.

Note that to achieve a maximum full range count of FFH for the 8-bit timer or FFFFH for the 16-bit timers, the preload registers must first be cleared to all zeros. It should be noted that after power-on, the preload registers will be in an unknown condition. Note that if the Timer/Event Counters are in an OFF condition and data is written to their preload registers, this data will be immediately written into the actual counter. However, if the counter is enabled and counting, any new data written into the preload data register during this period will remain in the preload register and will only be written into the actual counter the next time an overflow occurs. Note also that when the timer registers are read, the timer clock will be blocked to avoid errors, however, as this may result in certain timing errors, programmers must take this into account.

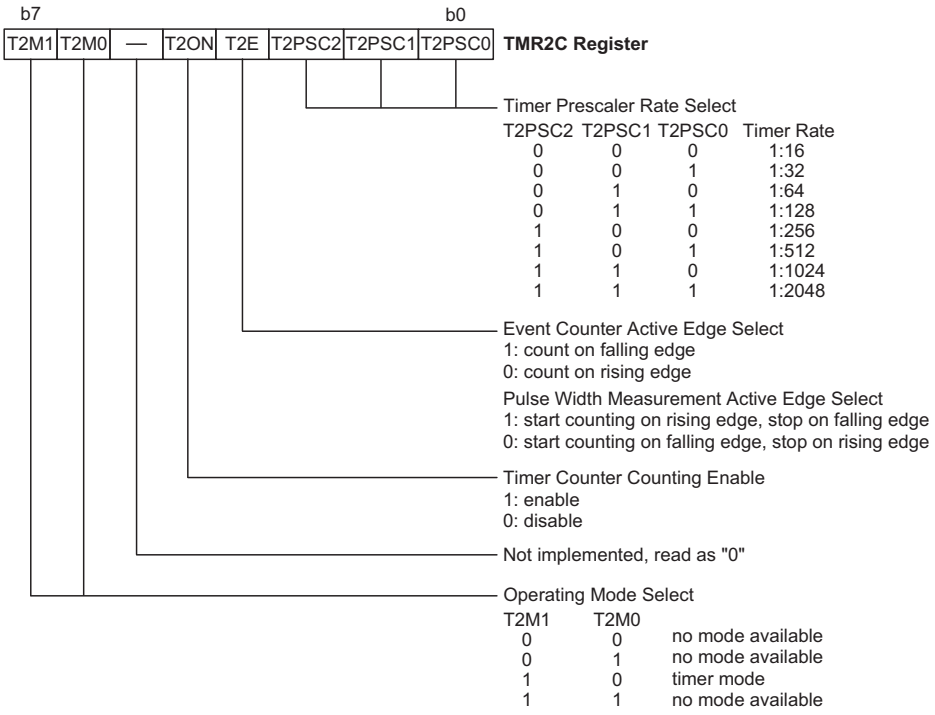
The 16-bit Timer/Event Counter have contained both low byte and high byte timer registers, accessing these registers is carried out in a specific way. It must be noted that when using instructions to preload data into the low byte register, namely TMR0L, the data will only be placed in a low byte buffer and not directly into the low byte register. The actual transfer of the data into the low byte register is only carried out when a write to its associated high byte register, namely TMR0H, is executed. On the other hand, using instructions to preload data into the high byte timer register will result in the data being directly written to the high byte register. At the same time the data in the low byte buffer will be transferred into its associated low byte register. For this reason, when preloading data into the 16-bit timer registers, the low byte should be written first. It must also be noted that to read the contents of the low byte register, a read to the



Timer/Event Counter 0 Control Register



Timer/Event Counter 1 Control Register



Timer Counter 2 Control Register

high byte register must first be executed to latch the contents of the low byte buffer into its associated low byte register. After this has been done, the low byte register can be read in the normal way. Note that reading the low byte timer register will only result in reading the previously latched contents of the low byte buffer and not the actual contents of the low byte timer register.

Timer Control Registers – TMR0C, TMR1C, TMR2C

The Timer/Event Counters0/1 enable them to operate in three different modes. the options of which are determined by the contents of their respective control register. There are four timer control registers, known as TMR0C, TMR1C and TMR2C. It is the timer control register together with its corresponding timer registers that control the full operation of the Timer/Event Counters. Before the timers can be used, it is essential that the appropriate timer control register is fully programmed with the right data to ensure its correct operation, a process that is normally carried out during program initialization. To choose which of the three modes the timer is to operate in, either in the timer mode, the event counting mode or the pulse width measurement mode, bits 7 and 6 of the Timer Control Register, which are known as the bit pair T0M1/T0M0, T1M1/T1M0 respectively, depending upon which timer is used, must be set to the required logic levels. The timer-on bit, which is bit 4 of the Timer Control Register and known as T0ON, T1ON or T2ON, depending upon which timer is used, provides the basic on/off control of the respective timer. Setting the bit high allows the counter to run, clearing the bit stops the counter. If the timer is in the event count or pulse width measurement mode, the active transition edge level type is selected by the logic level of bit 3 of the Timer Control Register which is known as T0E, T1E or T2E, depending upon which timer is used.

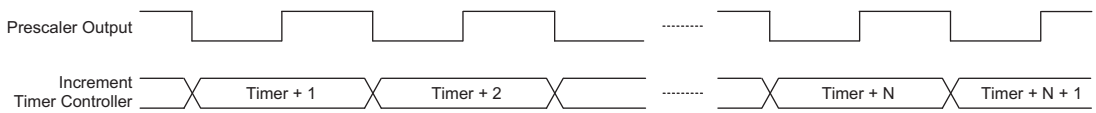
Configuring the Timer Mode

In this mode, the timer can be utilized to measure fixed time intervals, providing an internal interrupt signal each time the counter overflows. To operate in this mode, bits TM1 and TM0 of the TMR0C~TMR2C register must be set to 1 and 0 respectively. In this mode, the internal clock is used as the timer clock. The input clock frequency of 16 bit timer to the timer is $f_{OSC}/8$ and RC12K, selected by TMR0C.5. The input clock frequency of 8 bit timer to the timer is Fosc divided by the value programmed into the timer prescaler, the value of which is determined by bits PSC0~PSC2 of the TMR1C~TMR2C register. The timer-on bit, TON must be set high to enable the timer to run. Each time an internal clock high to low transition occurs, the timer increments by one. When the timer is full and overflows, the timer will be reset to the value already loaded into the preload register and continue counting. If the timer interrupt is enabled, an interrupt signal will also be generated. The timer interrupt can be disabled by ensuring that the ET0I~ET2I bit in the INTC and INTCH registers is cleared to zero. It should be noted that a timer overflow is one of the wake-up sources.

Configuring the Event Counter Mode

In this mode, two number of externally changing logic events, occurring on external pin PA6/TMR0 or PA7/TMR1, can be recorded by the internal timer. For the timer to operate in the event counting mode, bits TM1 and TM0 of the TMR0C or TMR1C registers must be set to 0 and 1 respectively. The timer-on bit, TON must be set high to enable the timer to count. With TE low, the counter will increment each time the PA6/TMR0 or PA7/TMR1 pin receives a low to high transition. If the TE bit is high, the counter will increment each time PA6/TMR0 or PA7/TMR1 pin receives a high to low transition. As in the case of the other two modes, when the counter is full and overflows, the timer will be reset to the value already loaded into the preload register and continue counting. If the timer interrupt is enabled, an interrupt signal will also be generated.

The timer interrupt can be disabled by ensuring that the ETI bit in the INTC and INTCH registers is cleared to zero. To ensure that the external pin PA6/TMR0 or PA7/TMR1 is configured to operate as an event counter input pin, two things have to happen. The first is to ensure that the TM0 and TM1 bits place the timer/event counter in the event counting mode, the second is to ensure that the share pin TMR0 or TMR1 are selected by option. It should be noted that a timer overflow is one of the wake-up sources. Also in the Event Counting mode, the Timer/Event Counter will continue to record externally changing logic events on the timer input pin, even if the microcontroller is in the Power Down Mode. As a result when the timer overflows it will generate a wake-up and if the interrupts are enabled also generate a timer interrupt signal.



Timer Mode Timing Diagram

Event Counter Mode Timing Diagram

Configuring the Pulse Width Measurement Mode

In this mode, the width of external pulses applied to the pin-shared external pin PA6/TMR0 or PA7/TMR1 can be measured. In the Pulse Width Measurement Mode, the timer clock source is supplied by the internal clock. For the timer to operate in this mode, bits TM0 and TM1 must both be set high. If the TE bit is low, once a high to low transition has been received on the PA6/TMR0 or PA7/TMR1 pin, the timer will start counting until the PA6/TMR0 or PA7/TMR1 pin returns to its original high level. At this point the TON bit will be automatically reset to zero and the timer will stop counting. If the TE bit is high, the timer will begin counting once a low to high transition has been received on the PA6/TMR0 or PA7/TMR1 pin and stop counting when the PA6/TMR0 or PA7/TMR1 pin returns to its original low level. As before, the TON bit will be automatically reset to zero and the timer will stop counting. It is important to note that in the Pulse Width Measurement Mode, the TON bit is automatically reset to zero when the external control signal on the external timer pin returns to its original level, whereas in the other two modes the TON bit can only be reset to zero under program control. The residual value in the timer, which can now be read by the program, therefore represents the length of the pulse received on pin PA6/TMR0 or PA7/TMR1. As the TON bit has now been reset any further transitions on the PA6/TMR0 or PA7/TMR1 pin will be ignored. Not until the TON bit is again set high by the program can the timer begin further pulse width measurements. In this way single shot pulse measurements can be easily made. It should be noted that in this mode the counter is controlled by logical transitions on the PA6/TMR0 or PA7/TMR1 pin and not by the logic level.

Prescaler

Bits PSC0~PSC2 of the TMRC1~ TMRC2 registers can be used to define the pre-scaling stages of the internal clock sources of the Timer/Event Counter.

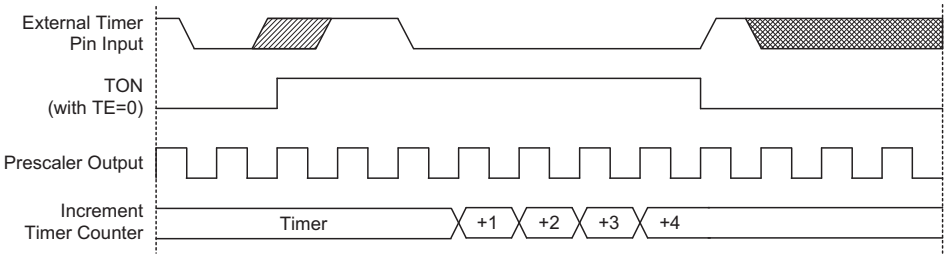
Note: Because the two timers are used by MIDI the external timer pin functions are disabled.

I/O Interfacing

The Timer/Event Counter, when configured to run in the event counter or pulse width measurement mode, require the use of the external PA6/TMR0 or PA7/TMR1 pin for correct operation. As this pin is a shared pin it must be configured correctly to ensure it is setup for use as a Timer/Event Counter input and not as a normal I/O pin. This is implemented by ensuring that the mode select bits in the Timer/Event Counter control register, select either the event counter or pulse width measurement mode. Additionally the PA share pin option must be selected to ensure that the pin is setup as an TMR0 and TMR1 input.

Programming Considerations

When configured to run in the timer mode, the internal system clock $f_{OSC}/8$ is used as the timer clock source and is therefore synchronized with the overall operation of the microcontroller. In this mode when the appropriate timer register is full, the microcontroller will generate an internal interrupt signal directing the program flow to the respective internal interrupt vector. For the pulse width measurement mode, the internal system clock is also used as the timer clock source but the timer will only run when the correct logic condition appears on the external timer input pin. As this is an external event and not synchronized with the internal timer clock, the microcontroller will only see this external event when the next timer clock pulse arrives. As a result, there may be



Prescaler Output is sampled at every falling edge of fosc/8 or RC12K.

Pulse Width Measure Mode Timing Diagram

small differences in measured values requiring programmers to take this into account during programming. The same applies if the timer is configured to be in the event counting mode, which again is an external event and not synchronized with the internal system or timer clock. When the Timer/Event Counter is read, or if data is written to the preload register, the clock is inhibited to avoid errors, however as this may result in a counting error, this should be taken into account by the programmer. Care must be taken to ensure that the timers are properly initialized before using them for the first time. The associated timer enable bits in the interrupt control register must be properly set otherwise the internal interrupt associated with the timer will remain inactive. The edge select, timer mode and clock source control bits in timer control register must also be correctly set to ensure the timer is properly configured for the required application. It is also important to ensure that an initial value is first loaded into the timer registers before the timer is switched on; this is because after power-on the initial values of the timer registers are unknown. After the timer has been initialized the timer can be turned on and off by controlling the enable bit in the timer control register. Note that setting the timer enable bit high to turn the timer on, should only be executed after the timer mode bits have been properly setup. Setting the timer enable bit high together with a mode bit modification, may lead to improper timer operation if executed as a single timer control register byte write instruction. When the Timer/Event counter overflows, its corresponding interrupt request flag in the interrupt control register will be set. If the timer interrupt is enabled this will in turn generate an interrupt signal. However irrespective of whether the interrupts are enabled or not, a Timer/Event counter overflow will also generate a wake-up signal if the device is in a Power-down condition. This situation may occur if the Timer/Event Counter is in the Event Counting Mode and if the external signal continues to change state. In such a case, the

Timer/Event Counter will continue to count these external events and if an overflow occurs the device will be woken up from its Power-down condition. To prevent such a wake-up from occurring, the timer interrupt request flag should first be set high before issuing the HALT instruction to enter the Power Down Mode.

Timer Program Example

This program example shows how the Timer/Event Counter registers are setup, along with how the interrupts are enabled and managed. Note how the Timer/Event Counter is turned on, by setting bit 4 of the Timer Control Register. The Timer/Event Counter can be turned off in a similar way by clearing the same bit.

This example program sets the Timer/Event Counter to be in the timer mode, which uses the internal system clock as the clock source. Show how to counter TMR0=1kHz, TMR1=2kHz, TMR2=4kHz, if f_{OSC} is 11.059MHz.

```
org 00h         ; Reset
jmp begin
org 04h         ; external interrupt vector
reti
org 08h         ; Timer/Event Counter 0 interrupt vector
jmp tmr0int     ; jump here when Timer0 overflows
org 0ch         ; Timer/Event Counter 1 interrupt vector
jmp tmr1int     ; jump here when Timer1 overflows
org 10h         ; Timer Counter 2 interrupt vector
jmp tmr2int     ; jump here when Timer2 overflows

org 20h         ; main program
;internal Timer0,1,2 Counter interrupt routine
tmr0int:
; Timer/Event Counter 0 main program placed here
:
reti

tmr1int:
; Timer/Event Counter 1 main program placed here
:
reti

tmr2int:
; Timer Counter 2 main program placed here
:
reti
:
```

```
begin:
; setup interrupt register
mov a, 0bh                  ; enable master interrupt, timer0 and timer1 interrupt
mov intc,a
mov a, 01h                  ; enable timer2 interrupt
mov intch,a

;setup Timer 0 registers
mov a, low (65536-1382)     ; setup Timer preload low byte value, interrupt in 1kHz
mov TMR0L,a;
mov a, high (65536-1382)    ; setup Timer preload high byte value
mov TMR0H,a;
mov a,080h                  ; setup Timer 0 control register
mov tmr0c,a                 ; timer mode and clock source is fosc/8 → 0.7234μs
set tmr0c.4                 ; start Timer - note mode bits must be previously setup

mov a, low (256-173)        ; setup Timer preload value, interrupt in 2kHz
mov TMR1,a;
mov a,080h                  ; setup Timer 1control register
mov tmr1c,a                 ; timer mode and Prescaler output is fosc/32 → 2.89μs
set tmr1c.4                 ; start Timer - note mode bits must be previously setup

mov a, low (256-173)         ; setup Timer preload value, interrupt in 4kHz
mov TMR2,a;
mov a,080h                  ; setup Timer2 control register
mov tmr2c,a                 ; timer mode and Prescaler output is fosc/16 → 1.447μs
set tmr2c.4                 ; start Timer - note mode bits must be previously setup
```

Interrupts

Interrupts are an important part of any microcontroller system. When an external event or an internal function such as a Timer/Event Counter 0/1/2 or ERCOCI require or an ADPCM empty requires microcontroller attention, their corresponding interrupt will enforce a temporary suspension of the main program allowing the microcontroller to direct attention to their respective needs. Each device in this series contains a single external interrupt and two internal interrupts functions. The external interrupt is controlled by the action of the external INT pin, while the internal interrupts are controlled by the Timer/Event 0/1Counter overflow or ERCOCI require or the ADPCM empty interrupt. Timer 2 counter overflow interrupt share with UART. Using the UART interrupt require is defined by enable UART function enable configuration option.

Interrupt Register

Overall interrupt control, which means interrupt enabling and request flag setting, is controlled by INTC and INTCH registers, which are located in Data Memory. By controlling the appropriate enable bits in this register each individual interrupt can be enabled or disabled. Also when an interrupt occurs, the corresponding request flag will be set by the microcontroller. The global enable flag if cleared to zero will disable all interrupts.

Interrupt Operation

Timer/Event 0/1/2 Counter overflow, UART interrupt, ERCOCI interrupt, ADPCM empty request or the external interrupt line being pulled low will all generate an interrupt request by setting their corresponding request flag, if their appropriate interrupt enable bit is set. When this happens, the Program Counter, which stores the address of the next instruction to be executed, will be transferred onto the stack. The Program Counter will then be loaded with a new address which will be the value of the corresponding interrupt vector. The microcontroller will then fetch its next instruction from this interrupt vector. The instruction at this vector will usually be a JMP statement which will jump to another section of program which is known as the interrupt service routine. Here is located the code to control the appropriate interrupt. The interrupt service routine must be terminated with a RETI statement, which retrieves the original Program Counter address from the stack and allows the microcontroller to continue with normal execution at the point where the interrupt occurred.

Once an interrupt subroutine is serviced, all the other interrupts will be blocked, as the EMI bit will be cleared automatically.

This will prevent any further interrupt nesting from occurring. However, if other interrupt requests occur during this interval, although the interrupt will not be immediately serviced, the request flag will still be recorded. If an interrupt requires immediate servicing while the program is already in another interrupt service routine, the EMI bit should be set after entering the routine, to allow interrupt nesting. If the stack is full, the interrupt request will not be acknowledged, even if the related interrupt is enabled, until the Stack Pointer is decremented. If immediate service is desired, the stack must be prevented from becoming full.

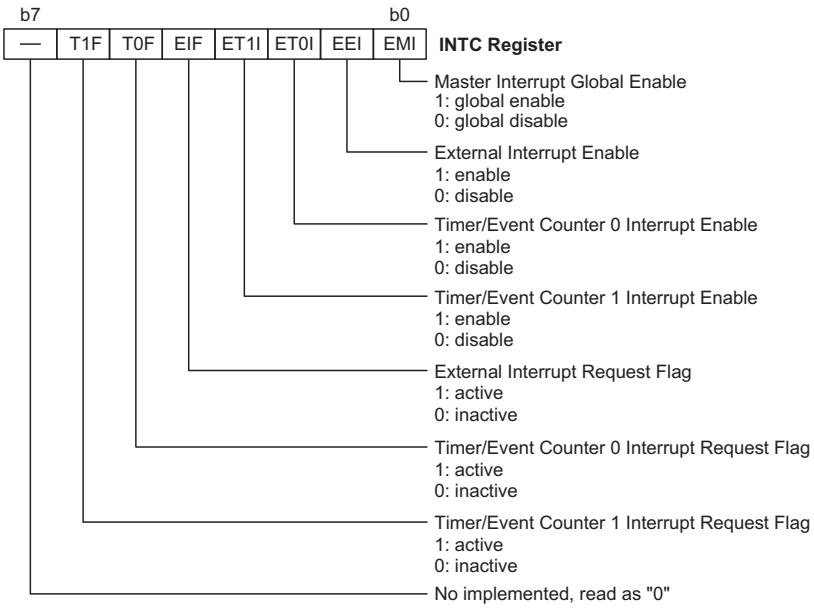
Interrupt Priority

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 case of simultaneous requests, the following table shows the priority that is applied. These can be masked by resetting the EMI bit. In cases where both external and internal interrupts are enabled and where an external and internal interrupt occurs simultaneously, the external interrupt will always have priority and will therefore be serviced first. Suitable masking of the individual interrupts using the INTC register can prevent simultaneous occurrences.

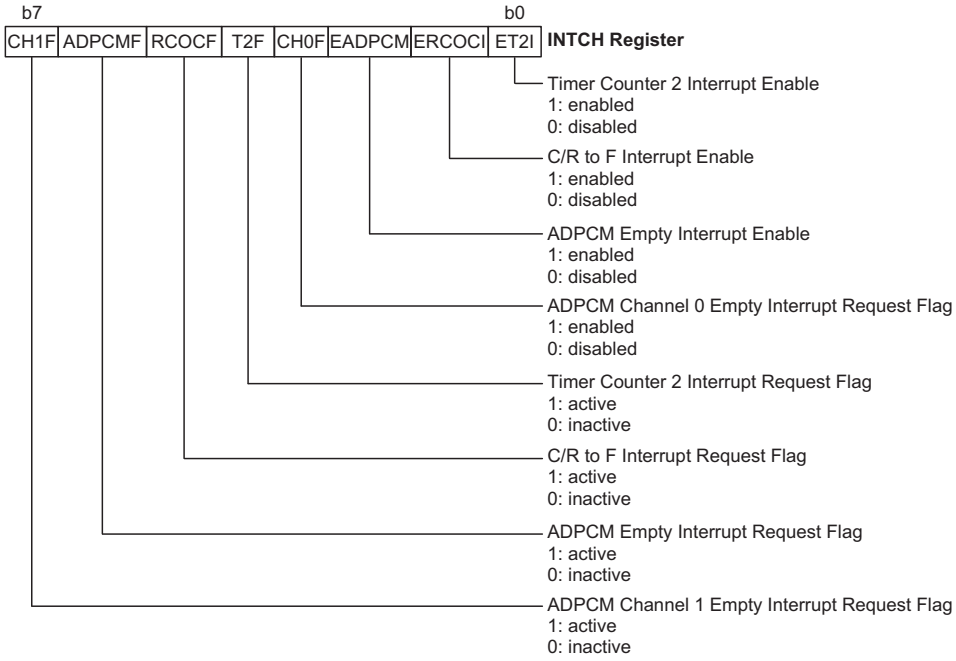
Interrupt Source	Priority	Vector
Reset	1	00H
External Interrupt	2	04H
Timer/Event Counter 0 Overflow	3	08H
Timer/Event Counter 1 Overflow	4	0CH
Timer Counter 2 overflow or UART Interrupt	5	10H
ERCOCI Interrupt	6	14H
ADPCM Empty Interrupt	7	18H

External Interrupt

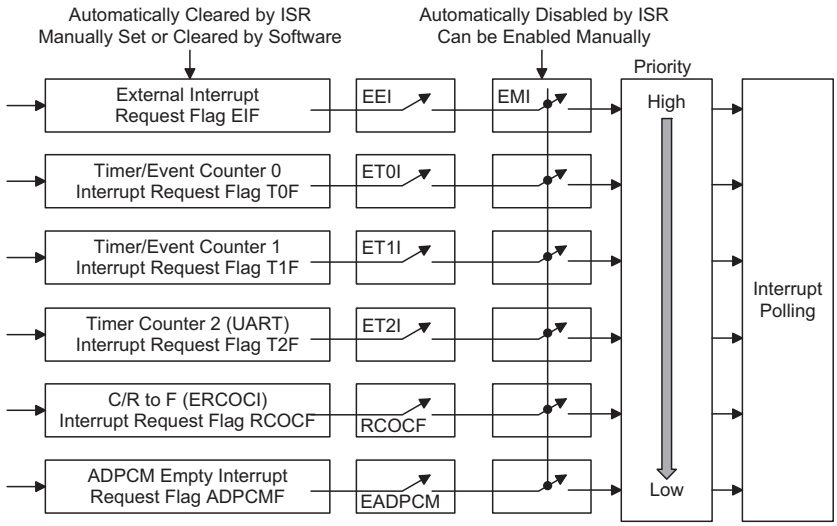
For an external interrupt to occur, the global interrupt enable bit, EMI, and external interrupt enable bit, EEI, must first be set. An actual external interrupt will take place when the external interrupt request flag, EIF, is set, a situation that will occur when a high to low transition appears on the INT line. The external interrupt pin is pin-shared with the I/O pin PA5 and can only be configured as an external interrupt pin if the corresponding external interrupt enable bit in the INTC register has been set. The pin must also be selected as by setting the corresponding PAC.5 bit in the port control register. When the interrupt is enabled, the stack is not full and a high to low transition appears on the external interrupt pin, a subroutine call to the external interrupt vector at location 04H, will take place. When the interrupt is serviced, the external interrupt request flag, EIF, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts.



Interrupt Low Byte Control Register



Interrupt High Byte Control Register



Interrupt Structure

Timer/Event Counter Interrupts

For a timer generated internal interrupt to occur, the corresponding internal interrupt enable bit must be first set. Each device have two internal Timer Counters, the Timer/Event Counter 0 interrupt enable is bit 2 of the INTC register and known as ET0I, the Timer/Event Counter 1 interrupt enable is bit 3 of the INTC register and known as ET1I and the Timer Counter 2 interrupt enable is bit 0 of the INTCH register and is known as ET2I. An actual Timer/Event Counter interrupt will be initialized when the Timer/Event Counter interrupt request flag is set, caused by a timer overflow. Each device has two timers, the Timer/Event Counter 0 request flag is bit 5 of the INTC register and known as T0F, the Timer/Event Counter 1 request flag is bit 6 of the INTC register and known as T1F, and the Timer Counter 2 request flag is bit 4 of the INTCH register and is known as T2F.

When the master interrupt global enable bit is set, the stack is not full and the corresponding timer interrupt enable bit is set, an internal interrupt will be generated when the corresponding timer overflows. Each device have two internal Timer/Event Counters, a subroutine call to location 08H will occur for Timer/Event Counter 0, a subroutine call to location 0CH for Timer/Event Counter 1, a subroutine call to location 10H for Timer Counter 2. After entering the timer interrupt execution routine, the corresponding timer interrupt request flag, either, T0F, T1F or T2F will be reset and the EMI bit will be cleared to disable other interrupts.

UART Interrupt

The device contain an internal UART function share with Timer Counter 2. It_s corresponding UART interrupt work by enabled UART function enable configuration option, which is bit 7 of the UART function enable configuration option. An actual UART interrupt will be initialized when the UART interrupt request flag T2F is set, which is bit 0 of the INTCH register. When the master interrupt global bit is set, the stack is not full and the corresponding ET2I interrupt enable bit is set, a UART internal interrupt will be generated when a UART interrupt request occurs. This will create a subroutine call to its corresponding vector location 010H. When a UART internal interrupt occurs, the interrupt request flag T2F

will be reset and the EMI bit cleared to disable other interrupts. There are various UART conditions, which can generate a UART interrupt, such as certain data transmission and reception conditions, overrun errors as well as an address detect condition. These conditions are reflected by various flags within the UART_s status register, known as the RS232C register. Various bits in the UART_s setup register, BRGR, determine if these flags can generate a UART interrupt signal. More details on these two registers and how they influence the operation of the UART interrupt can be found in the UART section of the datasheet.

RC/F Interrupt

The external RC Oscillation Converter interrupt is initialized by setting the external RC Oscillation Converter interrupt request flag, RCOCF; bit 5 of INTCH. This is caused by a Timer A or Timer B overflow. When the interrupt is enabled, and the stack is not full and the RCOCF bit is set, a subroutine call to location ″14H″ will occur.

The related interrupt request flag, RCOCF, will be reset and the EMI bit cleared to disable further interrupts.

ADPCM Interrupt

The internal ADPCM interrupt is initialized by setting the ADPCM interrupt request flag (ADPCMF: bit 6, CH0F: bit 3 and CH1F: bit 7 of INTCH).The CH0F and CH1F set by ADR0 or ADR1 empty respectively. The ADPCMF is set by ADR0 or ADR1 empty immediately. When the interrupt is enabled, and the stack is not full and the T0F bit is set, a subroutine call to location 18H will occur. The related interrupt request ADPCMF and CH0F/CH1F flag will be reset and the EMI bit cleared to disable further interrupts.

Programming Considerations

The interrupt request flags T0F, T1F, T2F, ADPCMF, CH0F, CH1F, together with the interrupt enable bits ET0I, ET1I, ET2I, EADPCM, form the interrupt control registers INTC, INTCH which are located in the Data Memory. By disabling the interrupt enable bits, a requested interrupt can be prevented from being serviced, however, once an interrupt request flag is set, it will remain in this condition in the INTC or INTCH register until the corresponding interrupt is serviced or until the request flag is cleared by a software instruction. It is recommended that programs do not use the ″CALL subroutine″ instruction within the interrupt subroutine. Interrupts often occur in an unpredictable manner or need to be serviced immediately in some applications. If only one stack is left and the interrupt is not well controlled, the original control sequence will be damaged once a ″CALL subroutine″ is executed in the interrupt subroutine.

All of these interrupts have the capability of waking up the processor when in the Power Down Mode. Only the Program Counter is pushed onto the stack. If the contents of the register or status register are altered by the interrupt service program, which may corrupt the desired control sequence, then the contents should be saved in advance.

Reset and Initialisation

A reset function is a fundamental part of any microcontroller ensuring that the device can be set to some predetermined condition irrespective of outside parameters. The most important reset condition is after power is first applied to the microcontroller. In this case, internal circuitry will ensure that the microcontroller, after a short delay, will be in a well defined state and ready to execute the first program instruction. After this power-on reset, certain important internal registers will be set to defined states before the program commences. One of these registers is the Program Counter, which will be reset to zero forcing the microcontroller to begin program execution from the lowest Program Memory address.

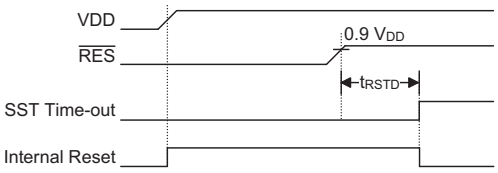
In addition to the power-on reset, situations may arise where it is necessary to forcefully apply a reset condition when the microcontroller is running. One example of this is where after power has been applied and the microcontroller is already running, the $\overline{\text{RES}}$ line is forcefully pulled low. In such a case, known as a normal operation reset, some of the microcontroller registers remain unchanged allowing the microcontroller to proceed with normal operation after the reset line is allowed to return high. Another type of reset is when the Watchdog Timer overflows and resets the microcontroller. All types of reset operations result in different register conditions being setup.

Another reset exists in the form of a Low Voltage Reset, LVR, where a full reset, similar to the $\overline{\text{RES}}$ reset is implemented in situations where the power supply voltage falls below a certain threshold.

Reset Functions

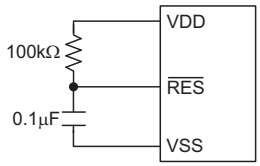
There are five ways in which a microcontroller reset can occur, through events occurring both internally and externally:

- Power-on Reset
  The most fundamental and unavoidable reset is the one that occurs after power is first applied to the microcontroller. As well as ensuring that the Program Memory begins execution from the first memory address, a power-on reset also ensures that certain other registers are preset to known conditions. All the I/O port and port control registers will power up in a high condition ensuring that all pins will be first set to inputs.
  Although the microcontroller has an internal RC reset function, if the VDD power supply rise time is not fast enough or does not stabilise quickly at power-on, the internal reset function may be incapable of providing proper reset operation. For this reason it is recommended that an external RC network is connected to the $\overline{\text{RES}}$ pin, whose additional time delay will ensure that the $\overline{\text{RES}}$ pin remains low for an extended period to allow the power supply to stabilise. During this time delay, normal operation of the microcontroller will be inhibited. After the $\overline{\text{RES}}$ line reaches a certain voltage value, the reset delay time t_{RSTD} is invoked to provide an extra delay time after which the microcontroller will begin normal operation. The abbreviation SST in the figures stands for System Start-up Timer.



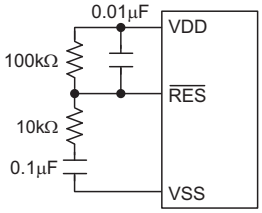
Power-On Reset Timing Chart

For most applications a resistor connected between VDD and the $\overline{\text{RES}}$ pin and a capacitor connected between VSS and the $\overline{\text{RES}}$ pin will provide a suitable external reset circuit. Any wiring connected to the $\overline{\text{RES}}$ pin should be kept as short as possible to minimise any stray noise interference.



Basic Reset Circuit

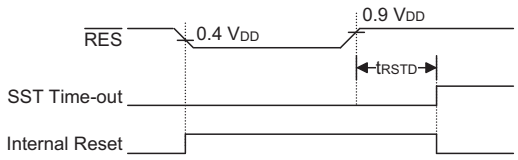
For applications that operate within an environment where more noise is present the Enhanced Reset Circuit shown is recommended.



Enhanced Reset Circuit

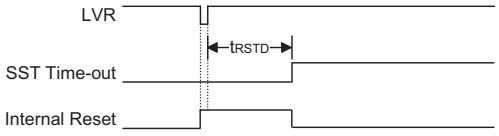
More information regarding external reset circuits is located in Application Note HA0075E on the Holtek website.

- $\overline{\text{RES}}$ Pin Reset
  This type of reset occurs when the microcontroller is already running and the $\overline{\text{RES}}$ pin is forcefully pulled low by external hardware such as an external switch. In this case as in the case of other reset, the Program Counter will reset to zero and program execution initiated from this point.



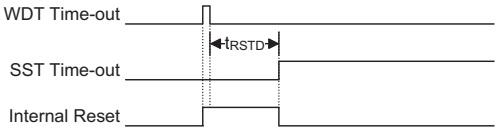
RES Reset Timing Chart

- Low Voltage Reset – LVR
  The microcontroller contains a low voltage reset circuit in order to monitor the supply voltage of the device, which is selected via a configuration option and The VLVR can select as 3.0V or 2.4V. If the supply voltage of the device drops to within a range of 0.9V~V_{LVR} such as might occur when changing the battery, the LVR will automatically reset the device internally. The LVR includes the following specifications: For a valid LVR signal, a low voltage, i.e., a voltage in the range between 0.9V~V_{LVR} must exist for greater than the value t_{LVR} specified in the A.C. characteristics. If the low voltage state does not exceed 1ms, the LVR will ignore it and will not perform a reset function.



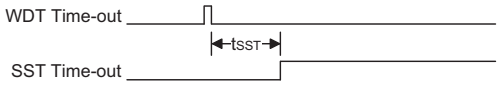
Low Voltage Reset Timing Chart

- Watchdog Time-out Reset during Normal Operation
  The Watchdog time-out Reset during normal operation is the same as a hardware RES pin reset except that the Watchdog time-out flag TO will be set to ″1″.



WDT Time-out Reset during Normal Operation Timing Chart

- Watchdog Time-out Reset during Power Down
  The Watchdog time-out Reset during Power Down is a little different from other kinds of reset. Most of the conditions remain unchanged except that the Program Counter and the Stack Pointer will be cleared to ″0″ and the TO flag will be set to ″1″. Refer to the A.C. Characteristics for t_{SST} details.



WDT Time-out Reset during Power Down Timing Chart

Reset Initial Conditions

The different types of reset described affect the reset flags in different ways. These flags, known as PDF and TO are located in the status register and are controlled by various microcontroller operations, such as the Power Down function or Watchdog Timer. The reset flags are shown in the table:

TO	PDF	RESET Conditions
0	0	RES reset during power-on
u	u	RES or LVR reset during normal operation
0	1	RES Wake-up HALT
1	u	WDT time-out reset during normal operation
1	1	WDT time-out reset during Power Down

Note: ″u″ stands for unchanged

The following table indicates the way in which the various components of the microcontroller are affected after a power-on reset occurs.

Item	Condition After RESET
Program Counter	Reset to zero
Interrupts	All interrupts will be disabled
WDT	Clear after reset, WDT begins counting
Timer/Event Counter	Timer Counter will be turned off
Prescaler	The Timer Counter Prescaler will be cleared
Input/Output Ports	I/O ports will be setup as inputs
Stack Pointer	Stack Pointer will point to the top of the stack

The different kinds of resets all affect the internal registers of the microcontroller in different ways. To ensure reliable continuation of normal program execution after a reset occurs, it is important to know what condition the microcontroller is in after a particular reset occurs. The following table describes how each type of reset affects each of the microcontroller internal registers.

HT37B90

Register	Reset (Power-on)	$\overline{RES}$ or LVR Reset	WDT Time-out (Normal Operation)	WDT Time-out (HALT)
TMR0H	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMR0L	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMR0C	0000 1---	0000 1---	0000 1---	uuuu u---
TMR1L	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMR1C	00-0 1000	00-0 1000	00-0 1000	uu-u uuuu
TMR2L	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMR2C	00-0 1000	00-0 1000	00-0 1000	uu-u uuuu
PCL	0000 0000	0000 0000	0000 0000	0000 0000
MP0	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
MP1	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
MP2	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
RBP1	---- -000	---- -000	---- -000	---- -uuu
RBP2	---- -000	---- -000	---- -000	---- -uuu
BP1	--00 0000	--00 0000	--00 0000	--uu uuuu
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLP1	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLH	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBMP1	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBHP1	---- -xxx	---- -uuu	---- -uuu	---- -uuu
STATUS	--00 xxxx	--uu uuuu	--1u uuuu	--11 uuuu
INTC	-000 0000	-000 0000	-000 0000	-uuu uuuu
INTCH	1000 1000	1000 1000	1000 1000	uuuu uuuu
PA	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAC	1111 1111	1111 1111	1111 1111	uuuu uuuu
PB	1111 1111	1111 1111	1111 1111	uuuu uuuu
PBC	1111 1111	1111 1111	1111 1111	uuuu uuuu
PC	1111 1111	1111 1111	1111 1111	uuuu uuuu
PCC	1111 1111	1111 1111	1111 1111	uuuu uuuu
PD	1111 1111	1111 1111	1111 1111	uuuu uuuu
PDC	1111 1111	1111 1111	1111 1111	uuuu uuuu
PE	1111 1111	1111 1111	1111 1111	uuuu uuuu
PEC	1111 1111	1111 1111	1111 1111	uuuu uuuu
DAC	0000 0000	0000 0000	0000 0000	uuuu uuuu
DAH	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
DAL	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
CHAN	00-- 0000	00-- 0000	00-- 0000	uu-- uuuu
FreqNH	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
FreqNL	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
AddrH	-xxx xxxx	-uuu uuuu	-uuu uuuu	-uuu uuuu
AddrL	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
RepH	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu

Register	Reset (Power-on)	$\overline{\text{RES}}$ or LVR Reset	WDT Time-out (Normal Operation)	WDT Time-out (HALT)
RepL	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
ENV	x–xx xxxx	u–uu uuuu	u–uu uuuu	u–uu uuuu
LVC	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
RVC	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
WDTS	0000 0111	0000 0111	0000 0111	0000 0uuu
ADR	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
XSPL	0000 0000	0000 0000	0000 0000	uuuu uuuu
XSPH	0000 0000	0000 0000	0000 0000	uuuu uuuu
ADPC	00–0 --00	00–0 --00	00–0 --00	uu–u --uu
ADPS	---- 1111	---- 1111	---- 1111	---- uuuu
ACSR	1--- --00	1--- --00	1--- --00	1--- --uu
ADRL	xxxx ----	xxxx ----	xxxx ----	uuuu ----
ADRH	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
ADCR	0100 0000	0100 0000	0100 0000	u1uu uuuu
ASCR	---- 0000	---- 0000	---- 0000	---- uuuu
TMRAH	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMRAL	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
RCOCCR	0000 1---	0000 1---	0000 1---	uuuu u---
TMRBH	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMRBL	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
RCOCR	---- --00	---- --00	---- --00	---- --uu
LVDC	---- --00	---- --00	---- --00	---- --uu
SBCR	0110 0000	0110 0000	0110 0000	uuuu uuuu
SBDR	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
RS232C	--10 --10	--10 --10	--10 --10	--uu --uu
RXD	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TXD	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
BRGR	0000 0000	0000 0000	0000 0000	uuuu uuuu

Note: "u" stands for unchanged
"x" stands for unknown
"–" stands for unimplemented

HT37B70

Register	Reset (Power-on)	$\overline{RES}$ or LVR Reset	WDT Time-out (Normal Operation)	WDT Time-out (HALT)
TMR0H	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMR0L	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMR0C	0000 1---	0000 1---	0000 1---	uuuu u---
TMR1L	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMR1C	00-0 1000	00-0 1000	00-0 1000	uu-u uuuu
TMR2L	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMR2C	00-0 1000	00-0 1000	00-0 1000	uu-u uuuu
PCL	0000 0000	0000 0000	0000 0000	0000 0000
MP0	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
MP1	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
MP2	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
RBP1	---- --00	---- --00	---- --00	---- --uu
RBP2	---- --00	---- --00	---- --00	---- --uu
BP1	---0 0000	---0 0000	---0 0000	---u uuuu
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLP1	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLH	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBMP1	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBHP1	---- --xx	---- --uu	---- --uu	---- --uu
STATUS	--00 xxxx	--uu uuuu	--1u uuuu	--11 uuuu
INTC	-000 0000	-000 0000	-000 0000	-uuu uuuu
INTCH	1000 1000	1000 1000	1000 1000	uuuu uuuu
PA	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAC	1111 1111	1111 1111	1111 1111	uuuu uuuu
PB	1111 1111	1111 1111	1111 1111	uuuu uuuu
PBC	1111 1111	1111 1111	1111 1111	uuuu uuuu
PC	1111 1111	1111 1111	1111 1111	uuuu uuuu
PCC	1111 1111	1111 1111	1111 1111	uuuu uuuu
PD	1111 1111	1111 1111	1111 1111	uuuu uuuu
PDC	1111 1111	1111 1111	1111 1111	uuuu uuuu
PE	1111 1111	1111 1111	1111 1111	uuuu uuuu
PEC	1111 1111	1111 1111	1111 1111	uuuu uuuu
DAC	0000 0000	0000 0000	0000 0000	uuuu uuuu
DAH	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
DAL	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
CHAN	00-- 0000	00-- 0000	00-- 0000	uu-- uuuu
FreqNH	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
FreqNL	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
AddrH	--xx xxxx	--uu uuuu	--uu uuuu	--uu uuuu
AddrL	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
RepH	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu

Register	Reset (Power-on)	$\overline{\text{RES}}$ or LVR Reset	WDT Time-out (Normal Operation)	WDT Time-out (HALT)
RepL	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
ENV	x–xx xxxx	u–uu uuuu	u–uu uuuu	u–uu uuuu
LVC	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
RVC	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
WDTS	0000 0111	0000 0111	0000 0111	0000 0uuu
ADR	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
XSPL	0000 0000	0000 0000	0000 0000	uuuu uuuu
XSPH	0000 0000	0000 0000	0000 0000	uuuu uuuu
ADPC	00–0 --00	00–0 --00	00–0 --00	uu–u --uu
ADPS	---- 1111	---- 1111	---- 1111	---- uuuu
ACSR	1--- --00	1--- --00	1--- --00	1--- --uu
ADRL	xxxx ----	xxxx ----	xxxx ----	uuuu ----
ADRH	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
ADCR	0100 0000	0100 0000	0100 0000	u1uu uuuu
ASCR	---- 0000	---- 0000	---- 0000	---- uuuu
TMRAH	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMRAL	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
RCOCCR	0000 1---	0000 1---	0000 1---	uuuu u---
TMRBH	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMRBL	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
RCOCR	---- --00	---- --00	---- --00	---- --uu
LVDC	---- --00	---- --00	---- --00	---- --uu
SBCR	0110 0000	0110 0000	0110 0000	uuuu uuuu
SBDR	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
RS232C	--10 --10	--10 --10	--10 --10	--uu --uu
RXD	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TXD	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
BRGR	0000 0000	0000 0000	0000 0000	uuuu uuuu

Note: "u" stands for unchanged
"x" stands for unknown
"–" stands for unimplemented

HT37B50

Register	Reset (Power-on)	$\overline{RES}$ or LVR Reset	WDT Time-out (Normal Operation)	WDT Time-out (HALT)
TMR0H	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMR0L	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMR0C	0000 1---	0000 1---	0000 1---	uuuu u---
TMR1L	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMR1C	00-0 1000	00-0 1000	00-0 1000	uu-u uuuu
TMR2L	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMR2C	00-0 1000	00-0 1000	00-0 1000	uu-u uuuu
PCL	0000 0000	0000 0000	0000 0000	0000 0000
MP0	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
MP1	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
MP2	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
RBP1	---- --00	---- --00	---- --00	---- --uu
RBP2	---- --00	---- --00	---- --00	---- --uu
BP1	---- 0000	---- 0000	---- 0000	---- uuuu
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLP1	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLH	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBMP1	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBHP1	---- ---x	---- ---u	---- ---u	---- ---u
STATUS	--00 xxxx	--uu uuuu	--1u uuuu	--11 uuuu
INTC	-000 0000	-000 0000	-000 0000	-uuu uuuu
INTCH	1000 1000	1000 1000	1000 1000	uuuu uuuu
PA	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAC	1111 1111	1111 1111	1111 1111	uuuu uuuu
PB	1111 1111	1111 1111	1111 1111	uuuu uuuu
PBC	1111 1111	1111 1111	1111 1111	uuuu uuuu
PC	1111 1111	1111 1111	1111 1111	uuuu uuuu
PCC	1111 1111	1111 1111	1111 1111	uuuu uuuu
PD	1111 1111	1111 1111	1111 1111	uuuu uuuu
PDC	1111 1111	1111 1111	1111 1111	uuuu uuuu
DAC	0000 0000	0000 0000	0000 0000	uuuu uuuu
DAH	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
DAL	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
CHAN	00-- 0000	00-- 0000	00-- 0000	uu-- uuuu
FreqNH	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
FreqNL	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
AddrH	---x xxxx	---u uuuu	---u uuuu	---u uuuu
AddrL	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
RepH	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
RepL	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
ENV	x-xx xxxx	u-uu uuuu	u-uu uuuu	u-uu uuuu

Register	Reset (Power-on)	$\overline{\text{RES}}$ or LVR Reset	WDT Time-out (Normal Operation)	WDT Time-out (HALT)
LVC	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
RVC	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
WDTS	0000 0111	0000 0111	0000 0111	0000 0uuu
ADR	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
XSPL	0000 0000	0000 0000	0000 0000	uuuu uuuu
XSPH	0000 0000	0000 0000	0000 0000	uuuu uuuu
ADPC	00-0 --00	00-0 --00	00-0 --00	uu-u --uu
ADPS	---- 1111	---- 1111	---- 1111	---- uuuu
ACSR	1--- --00	1--- --00	1--- --00	1--- --uu
ADRL	xxxx ----	xxxx ----	xxxx ----	uuuu ----
ADRH	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
ADCR	0100 0000	0100 0000	0100 0000	u1uu uuuu
ASCR	---- 0000	---- 0000	---- 0000	---- uuuu
TMRAH	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMRAL	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
RCOCCR	0000 1---	0000 1---	0000 1---	uuuu u---
TMRBH	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMRBL	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
RCOCR	---- --00	---- --00	---- --00	---- --uu
LVDC	---- --00	---- --00	---- --00	---- --uu

Note: "u" stands for unchanged
"x" stands for unknown
"-" stands for unimplemented

HT37B30

Register	Reset (Power-on)	$\overline{RES}$ or LVR Reset	WDT Time-out (Normal Operation)	WDT Time-out (HALT)
TMR0H	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMR0L	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMR0C	0000 1---	0000 1---	0000 1---	uuuu u---
TMR1L	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMR1C	00-0 1000	00-0 1000	00-0 1000	uu-u uuuu
TMR2L	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMR2C	00-0 1000	00-0 1000	00-0 1000	uu-u uuuu
PCL	0000 0000	0000 0000	0000 0000	0000 0000
MP0	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
MP1	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
MP2	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
RBP1	---- --00	---- --00	---- --00	---- --uu
RBP2	---- --00	---- --00	---- --00	---- --uu
BP1	---- -000	---- -000	---- -000	---- -uuu
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLP1	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLH	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBMP1	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
STATUS	--00 xxxx	--uu uuuu	--1u uuuu	--11 uuuu
INTC	-000 0000	-000 0000	-000 0000	-uuu uuuu
INTCH	1000 1000	1000 1000	1000 1000	uuuu uuuu
PA	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAC	1111 1111	1111 1111	1111 1111	uuuu uuuu
PB	1111 1111	1111 1111	1111 1111	uuuu uuuu
PBC	1111 1111	1111 1111	1111 1111	uuuu uuuu
PC	1111 1111	1111 1111	1111 1111	uuuu uuuu
PCC	1111 1111	1111 1111	1111 1111	uuuu uuuu
PD	1111 1111	1111 1111	1111 1111	uuuu uuuu
PDC	1111 1111	1111 1111	1111 1111	uuuu uuuu
DAC	0000 0000	0000 0000	0000 0000	uuuu uuuu
DAH	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
DAL	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
CHAN	00-- 0000	00-- 0000	00-- 0000	uu-- uuuu
FreqNH	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
FreqNL	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
AddrH	---- xxxx	---- uuuu	---- uuuu	---- uuuu
AddrL	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
RepH	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
RepL	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
ENV	x-xx xxxx	u-uu uuuu	u-uu uuuu	u-uu uuuu
LVC	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu

Register	Reset (Power-on)	$\overline{RES}$ or LVR Reset	WDT Time-out (Normal Operation)	WDT Time-out (HALT)
RVC	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
WDTS	0000 0111	0000 0111	0000 0111	0000 0uuu
ADR	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
XSPL	0000 0000	0000 0000	0000 0000	uuuu uuuu
XSPH	0000 0000	0000 0000	0000 0000	uuuu uuuu
ADPC	00-0 --00	00-0 --00	00-0 --00	uu-u --uu
ADPS	---- 1111	---- 1111	---- 1111	---- uuuu
ACSR	1--- --00	1--- --00	1--- --00	1--- --uu
ADRL	xxxx ----	xxxx ----	xxxx ----	uuuu ----
ADRH	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
ADCR	0100 0000	0100 0000	0100 0000	u1uu uuuu
ASCR	---- 0000	---- 0000	---- 0000	---- uuuu
TMRAH	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMRAL	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
RCOCCR	0000 1---	0000 1---	0000 1---	uuuu u---
TMRBH	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMRBL	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
RCOCR	---- --00	---- --00	---- --00	---- --uu
LVDC	---- --00	---- --00	---- --00	---- --uu

Note: "u" stands for unchanged
"x" stands for unknown
"-" stands for unimplemented

Oscillator

Various oscillator options offer the user a wide range of functions according to their various application requirements. Two types of system clocks can be selected while various clock source options for the Watchdog Timer are provided for maximum flexibility. All oscillator options are selected through the configuration options.

The two methods of generating the system clock are:

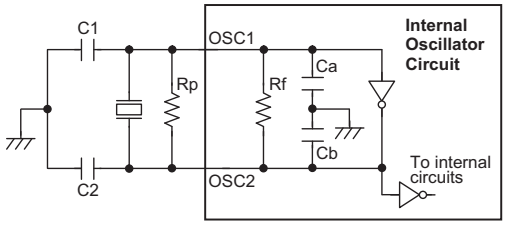
- External crystal/resonator oscillator
- External RC oscillator

One of these two methods must be selected using the configuration options.

More information regarding the oscillator is located in Application Note HA0075E on the Holtek website.

External Crystal/Resonator Oscillator

The simple connection of a crystal across OSC1 and OSC2 will create the necessary phase shift and feedback for oscillation, and will normally not require external capacitors. However, for some crystals and most resonator types, to ensure oscillation and accurate frequency generation, it may be necessary to add two small value external capacitors, C1 and C2. The exact values of C1 and C2 should be selected in consultation with the crystal or resonator manufacturer's specification. The external parallel feedback resistor, Rp, is normally not required but in some cases may be needed to assist with oscillation start up.



Note: 1. Rp is normally not required.
2. Although not shown OSC1/OSC2 pins have a parasitic capacitance of around 7pF.

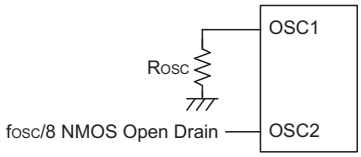
Crystal/Resonator Oscillator

Internal Ca, Cb, Rf Typical Values @ 5V, 25°C		
Ca	**Cb**	**Rf**
7pF~9pF	9pF~11pF	300kΩ

Oscillator Internal Component Values

External RC Oscillator

Using the external system RC oscillator requires that a resistor, with a value between 82kΩ and 180kΩ, is connected between OSC1 and VSS. The generated system clock divided by 8 will be provided on OSC2 as an output which can be used for external synchronization purposes. Note that as the OSC2 output is an NMOS open-drain type, a pull high resistor should be connected if it to be used to monitor the internal frequency. Although this is a cost effective oscillator configuration, the oscillation frequency can vary with VDD, temperature and process variations and is therefore not suitable for applications where timing is critical or where accurate oscillator frequencies are required. For the value of the external resistor. Note that it is the only microcontroller internal circuitry together with the external resistor, that determine the frequency of the oscillator. The external capacitor shown on the diagram does not influence the frequency of oscillation.



External RC Oscillator

Watchdog Timer Oscillator

The WDT oscillator is a fully self-contained free running on-chip RC oscillator with a typical period of 65μs at 5V requiring no external components. When the device enters the Power Down Mode, the system clock will stop running but the WDT oscillator continues to free-run and to keep the watchdog active. However, to preserve power in certain applications the WDT oscillator can be disabled via a configuration option.

Power Down Mode and Wake-up

Power Down Mode

All of the Holtek microcontrollers have the ability to enter a Power Down Mode, also known as the HALT Mode or Sleep Mode. When the device enters this mode, the normal operating current, will be reduced to an extremely low standby current level. This occurs because when the device enters the Power Down Mode, the system oscillator is stopped which reduces the power consumption to extremely low levels, however, as the device maintains its present internal condition, it can be woken up at a later stage and continue running, without requiring a full reset. This feature is extremely important in application areas where the MCU must have its power supply constantly maintained to keep the device in a known condition but where the power supply capacity is limited such as in battery applications.

Entering the Power Down Mode

There is only one way for the device to enter the Power Down Mode and that is to execute the "HALT" instruction in the application program. When this instruction is executed, the following will occur:

- The system oscillator will stop running and the application program will stop at the "HALT" instruction.
- The Data Memory contents and registers will maintain their present condition.
- The WDT will be cleared and resume counting if the WDT clock source is selected to come from the WDT oscillator. The WDT will stop if its clock source originates from the system clock.
- The I/O ports will maintain their present condition.
- In the status register, the Power Down flag, PDF, will be set and the Watchdog time-out flag, TO, will be cleared.

Standby Current Considerations

As the main reason for entering the Power Down Mode is to keep the current consumption of the MCU to as low a value as possible, perhaps only in the order of several micro-amps, there are other considerations which must also be taken into account by the circuit designer if the power consumption is to be minimized. Special attention must be made to the I/O pins on the device. All high-impedance input pins must be connected to either a fixed high or low level as any floating input pins could create internal oscillations and result in increased current consumption. This also applies to devices which have different package types, as there may be undonbed pins, which must either be setup as outputs or if setup as inputs must have pull-high resistors connected. Care must also be taken with the loads, which are connected to I/O pins, which are setup as outputs. These should be placed in a condition in which minimum current is drawn or connected only to external circuits that do not draw current, such as other CMOS inputs. Also note that additional standby current will also be required if the configuration options have enabled the Watchdog Timer internal oscillator.

Wake-up

After the system enters the Power Down Mode, it can be woken up from one of various sources listed as follows:

- An external reset
- An external falling edge on Port A
- A system interrupt
- A WDT overflow

If the system is woken up by an external reset, the device will experience a full system reset, however, if the device is woken up by a WDT overflow, a Watchdog Timer reset will be initiated. Although both of these wake-up methods will initiate a reset operation, the actual source of the wake-up can be determined by examining the TO and PDF flags. The PDF flag is cleared by a system power-up or executing the clear Watchdog Timer instructions and is set when executing the "HALT" instruction. The TO flag is set if a WDT time-out occurs, and causes a wake-up that only resets the Program Counter and Stack Pointer, the other flags remain in their original status.

Each pin on Port A can be setup via an individual configuration option to permit a negative transition on the pin to wake-up the system. When a Port A pin wake-up occurs, the program will resume execution at the instruction following the "HALT" instruction.

If the system is woken up by an interrupt, then two possible situations may occur. The first is where the related interrupt is disabled or the interrupt is enabled but the stack is full, in which case the program will resume execution at the instruction following the "HALT" instruction. In this situation, the interrupt which woke-up the device will not be immediately serviced, but will rather be serviced later when the related interrupt is finally enabled or when a stack level becomes free. The other situation is where the related interrupt is enabled and the stack is not full, in which case the regular interrupt response takes place. If an interrupt request flag is set to "1" before entering the Power Down Mode, the wake-up function of the related interrupt will be disabled.

No matter what the source of the wake-up event is, once a wake-up situation occurs, a time period equal to 1024 system clock periods will be required before normal system operation resumes. However, if the wake-up has originated due to an interrupt, the actual interrupt subroutine execution will be delayed by an additional one or more cycles. If the wake-up results in the execution of the next instruction following the "HALT" instruction, this will be executed immediately after the 1024 system clock period delay has ended.

Watchdog Timer

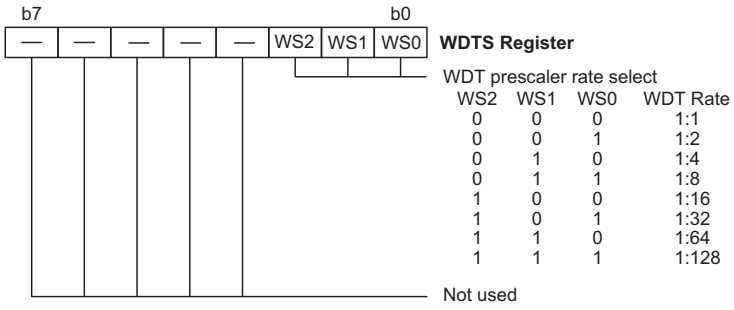
The Watchdog Timer is provided to prevent program malfunctions or sequences from jumping to unknown locations, due to certain uncontrollable external events such as electrical noise. It operates by providing a device reset when the WDT counter overflows. The WDT clock is supplied by one of two sources selected by configuration option: its own self contained dedicated internal WDT oscillator or $f_{OSC}/8$. Note that if the WDT configuration option has been disabled, then any instruction relating to its operation will result in no operation.

The internal WDT oscillator has an approximate period of 65μs at a supply voltage of 5V. If selected, it is first divided by 256 via an 8-stage counter to give a nominal period of 17ms. Note that this period can vary with VDD, temperature and process variations. For longer WDT time-out periods the WDT prescaler can be utilized. By writing the required value to bits 0, 1 and 2 of the WDTS register, known as WS0, WS1 and WS2, longer time-out periods can be achieved. With WS0, WS1 and WS2 all equal to 1, the division ratio is 1:128 which gives a maximum time-out period of about 2.1s.

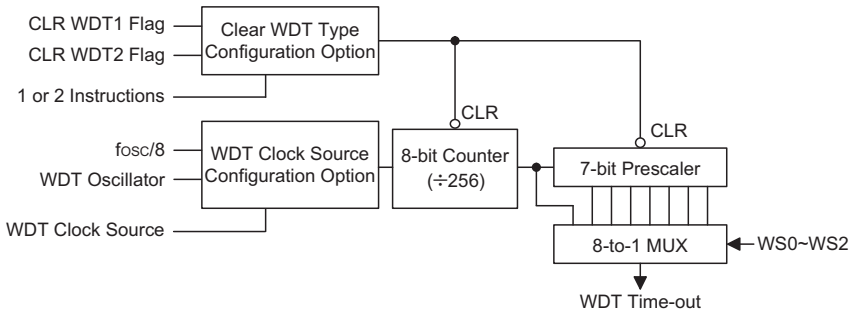
A configuration option can select the instruction clock, which is the system clock divided by 8, as the WDTclock source instead of the internal WDT oscillator. If the instruction clock is used as the clock source, it must be noted that when the system enters the Power Down Mode, as the system clock is stopped, then the WDT clock source will also be stopped. Therefore the WDT will lose its protecting purposes. In such cases the system cannot be restarted by the WDT and can only be restarted using external signals. For systems that operate in noisy environments, using the internal WDT oscillator is therefore the recommended choice.

Under normal program operation, a WDT time-out will initialise a device reset and set the status bit TO. However, if the system is in the Power Down Mode, when a WDT time-out occurs, only the Program Counter and Stack Pointer will be reset. Three methods can be adopted to clear the contents of the WDT and the WDT prescaler. The first is an external hardware reset, which means a low level on the $\overline{RES}$ pin, the second is using the watchdog software instructions and the third is via a "HALT" instruction.

There are two methods of using software instructions to clear the Watchdog Timer, one of which must be chosen by configuration option. The first option is to use the single "CLR WDT" instruction while the second is to use the two commands "CLR WDT1" and "CLR WDT2". For the first option, a simple execution of "CLR WDT" will clear the WDT while for the second option, both "CLR WDT1" and "CLR WDT2" must both be executed to successfully clear the WDT. Note that for this second option, if "CLR WDT1" is used to clear the WDT, successive executions of this instruction will have no effect, only the execution of a "CLR WDT2" instruction will clear the WDT. Similarly, after the "CLR WDT2" instruction has been executed, only a successive "CLR WDT1" instruction can clear the Watchdog Timer.



Watchdog Timer Register



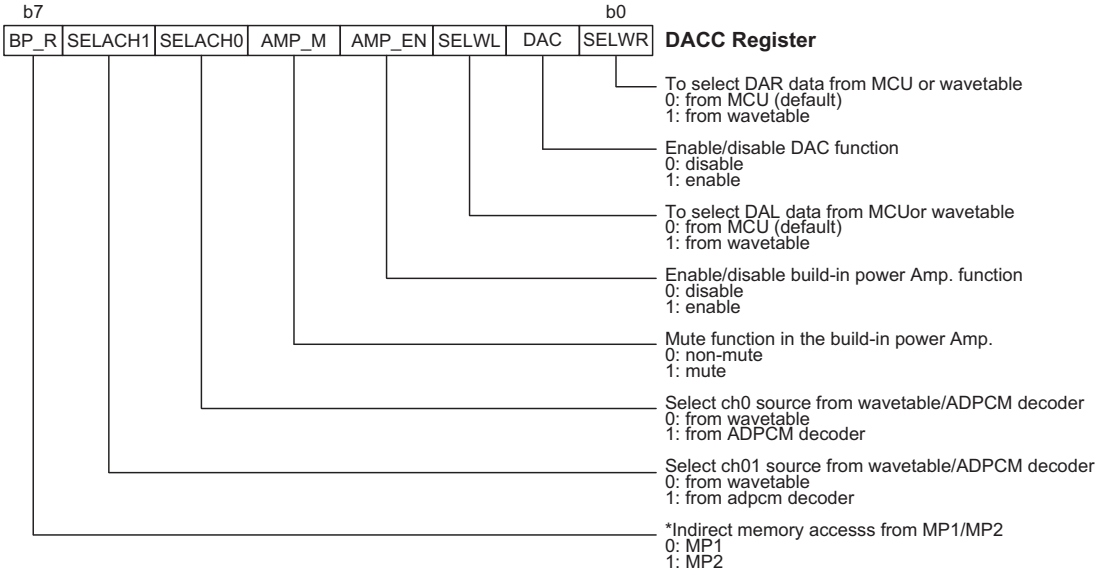
Watchdog Timer

Digital to Analog Converter (DACC)

The two D/A converters of HT37B90/70/50/30 are 16-bit high-resolution with excellent frequency response characteristics and good power consumption for stereo audio output.

		D7	D6	D5	D4	D3	D2	D1	D0
1Dh	DAC High Byte	B15	B14	B13	B12	B11	B10	B9	B8
1Eh	DAC Low Byte	B7	B6	B5	B4	B3	B2	B1	B0
1Fh	DAC Control (DACC)	BP_R	SELACH1	SELACH0	AMP_M	AMP_EN	SELWL	DAC	SELWR

Note: B15~B0 is D/A conversion result data bit MSB~LSB.

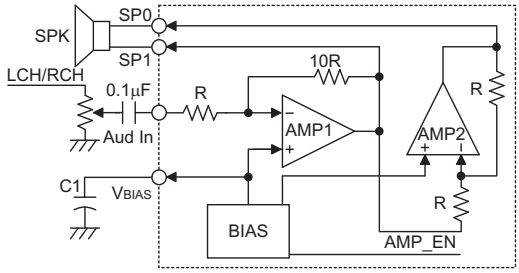


DACC (1FH) Register

Note: *Switch MP1and MP2 memory pointer by BP_R

The Integrated Power Amp.

The Power Amp. is an integrated class AB mono speaker driver contained in HT37B90/70/50/30. It provides property of high S/N ratio, high slew rate, low distortion, large output voltage swing, excellent power supply ripple rejection, low power consumption, low standby current and power off control etc.



Aud In: Audio input

V_{BIAS}: Speaker non-inverting input voltage reference

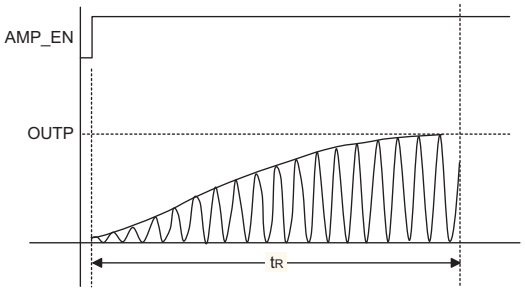
SP0: Audio Negative output

SP1: Audio Positive output

OUTP Rising Time (t_R)

When AMP_EN enable, the Power Amp. need rising time to output fully on OUTP pin. However, the rising time depends on.

C1. (*The C1 connects with V_{BIAS} and Vss)

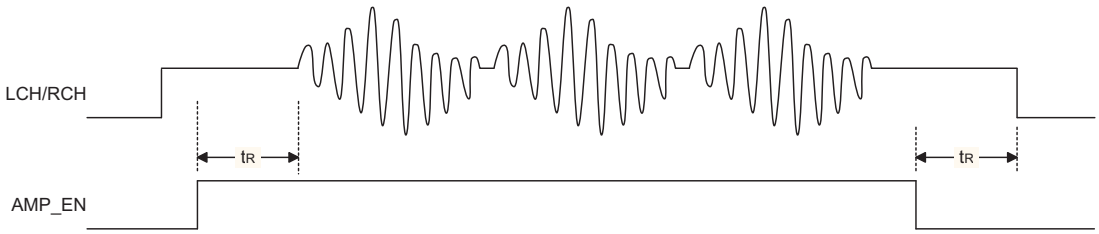


Capacitor / t_R / Voltage	0.1μF	1μF	4.7μF	10μF
2.2V	15ms	30ms	90ms	185ms
3V	15ms	30ms	90ms	185ms
4	15ms	30ms	90ms	185ms

For battery based applications, power consumption is a key issue, therefore the amplifier should be turned off when in the standby state. In order to eliminate any speaker sound bursts while turning the amplifier on, the application circuit, which will incorporate a capacitance value of C1, should be adjusted in accordance with the speaker s audio frequency response. A greater value of C1 will improve the noise burst while turning on the amplifier. The recommended operation sequence is:

Turn On: audio signal standby (1/2VDD) → enable amplifier → wait t_R for amplifier ready → audio output

Turn Off: audio signal finished → disable amplifier → wait t_R for amplifier off → audio signal off



If the application is not powered by batteries and there is no problem with amplifier On/Off issue, a capacitor value of 0.1μF for C1 is recommended.

How to use integrated power Amp?

- Connect the ″Internal Power Amp Circuit″, please refer to Application Circuits.
- Set DACC.3 to enable integrated power amp. Clear DACC.3 to disable integrated power amp.
- User can control it at ″PowerAmpDisable″ and ″PowerAmpEnable″ of HT-MDS.

Music Synthesis Controller – MSC

CH0~CH15 Channel Number Selection

Each devices with integrated 16 channels output is selected by 4bits option and CHAN[3:0] 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 change the frequency parameter
1	0	Only change the volume parameter
1	1	Unused

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 12.8MHz, the formula of a tone frequency is:

$$f_{OUT} = f_{RECORD} \times \frac{f_{osc} / (16 \times 16)}{SR} \times \frac{FR11 \sim FR0}{2^{(17 - BL3 \sim BL0)}}$$

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 formula: If FR=55h and BL=7, could get 98Hz.

$$98Hz = 261Hz \times \frac{50kHz}{11.025kHz} \times \frac{FR11 \sim FR0}{2^{(17 - BL3 \sim BL0)}}$$

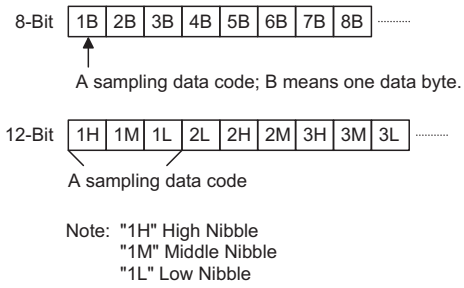
BL3~BL0: range from 00h~0Bh
FR11~FR0: range from 000h~3FFh

Waveform Format Definition

Each device 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 (PCM8)
WBS=1 means the sample format is 12-bit (PCM12)

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



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 RE14~RE0 is used to calculate the repeat address of the PCM code. The process for setting the RE14~RE0 is to write the 2′s complement of the repeat length to RE14~RE0, with the highest carry ignored. The HT37 will get the repeat address by adding the RE14~RE0 to the address of the end code, then jump to the address to repeat this range.

Volume Control

Each device 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 Each device 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.

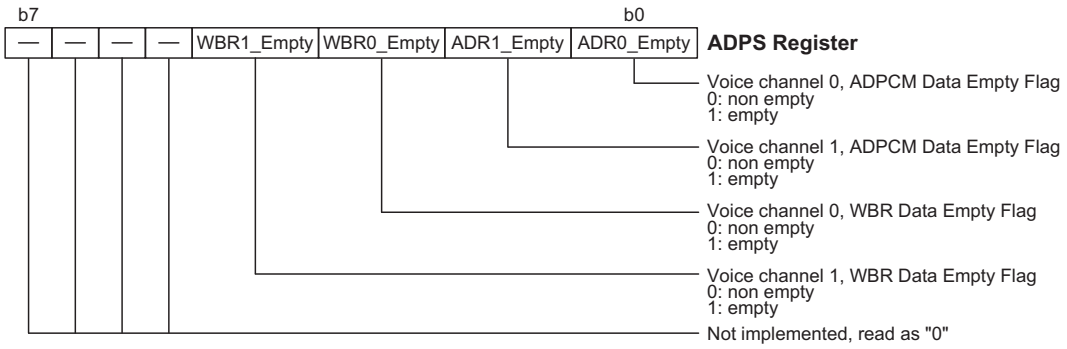
Name	Function	D7	D6	D5	D4	D3	D2	D1	D0
20H	Channel number selection (CHAN)	VM	FR	—	—	CH3	CH2	CH1	CH0
21H	Frequency number high byte (FreqNH)	BL3	BL2	BL1	BL0	FR11	FR10	FR9	FR8
22H	Frequency number low byte (FreqNL)	FR7	FR6	FR5	FR4	FR3	FR2	FR1	FR0
23H	Start address high byte (AddrH)	—	ST14	ST13	ST12	ST11	ST10	ST9	ST8
24H	Start address low byte (AddrL)	ST7	ST6	ST5	ST4	ST3	ST2	ST1	ST0
25H	Repeat number high byte (RepH)	WBS	RE14	RE13	RE12	RE11	RE10	RE9	RE8
26H	Repeat number low byte (RepL)	RE7	RE6	RE5	RE4	RE3	RE2	RE1	RE0
27H	Control register (ENV)	A_R	—	VL9	VL8	ENV1	ENV0	VR9	VR8
28H	—								
29H	Left volume control (LVC)	VL7	VL6	VL5	VL4	VL3	VL2	VL1	VL0
2AH	Right volume control (RVC)	VR7	VR6	VR5	VR4	VR3	VR2	VR1	VR0

Wavetable Register Memory Map (20H~2AH)

ADPCM

Address Offset	Register Name	R/W	Default Value	Description
30H	ADR	W	xxxx xxxx	ADPCM Data Register
31H	XSPL	W	0000 0000	Xn + SP Initial Register Low Byte
32H	XSPH	W	0000 0000	Xn + SP Initial Register High Byte
33H	ADPC	R/W	00x0 xx00	ADPCM Decoder control register
34H	ADPS	R	0000 1111	ADPCM Decoder Status Register

HT-ADPCM Decoder Registers



ADPS (34H) – ADPCM Status Register

External RC Oscillation Converter

An external RC oscillation converter is implemented in certain devices and is a function which allows touch switch functions to be implemented. When used in conjunction with the Analog Switch function up to eight touch switches can be implemented.

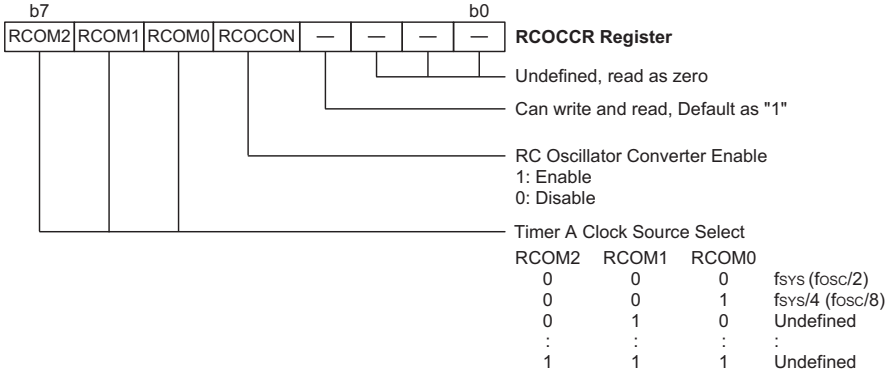
External RC Oscillation Converter Operation

The RC oscillation converter is composed of two 16-bit count-up programmable timers. One is Timer A and the other is counter known as Timer B. The RC oscillation converter is enabled when the RCO bit, which is bit 1 of the RCOCR register, is set high. The RC oscillation converter will then be composed of four registers, TMRAL, TMRAH, TMRBL and TMRBH. The Timer A clock source comes from the f_{SYS} or $f_{SYS}/4$, the choice of which is determined by bits in the RCOCCR register. The RC oscillation converter Timer B clock source comes from an external RC oscillator. As the oscillation frequency is dependent upon external capacitance and resistance values, it can therefore be used to detect the increased capacitance of a touch switch pad.

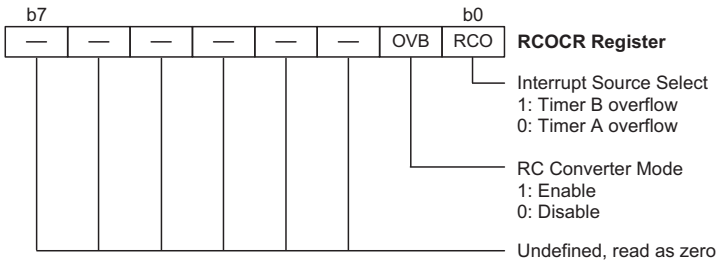
There are six registers related to the RC oscillation converter. These are, TMR2H, TMR2L, RCOCCR, TMR4H, TMR4L and RCOCR. The internal timer clock is the input clock source for TMRAH and TMRAL, while the external RC oscillator is the clock source input to TMRBH and TMRBL. The OVB bit, which is bit 0 of the RCOCR register, decides whether the timer interrupt is sourced from either the Timer A overflows or Timer B overflow. When a timer overflow occurs, the RCOCF bit is set and an external RC oscillation converter interrupt occurs. When the RC oscillation converter Timer A or Timer B overflows, the RCOCON bit is automatically reset to zero and stops counting.

The resistor and capacitor form an oscillation circuit and input to TMRBH and TMRBL. The RCOM0, RCOM1 and RCOM2 bits of RCOCCR define the clock source of Timer A.

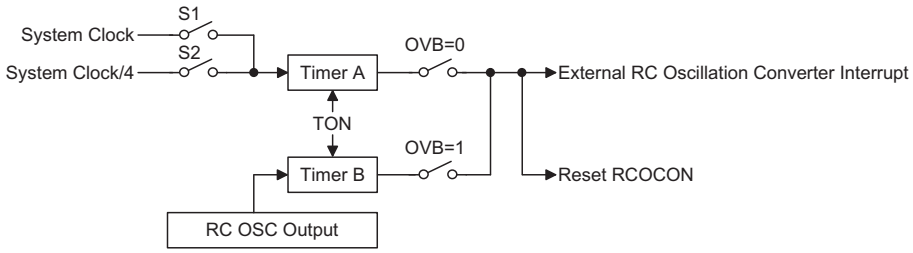
When the RCOCON bit, which is bit 4 of the RCOCCR register, is set high, Timer A and Timer B will start counting until Timer A or Timer B overflows. Now the timer counter will generate an interrupt request flag which is bit RCOCF, bit 5 of the INTCH register. Both Timer A and Timer B will then stop counting and the RCOCON bit will automatically be reset to ″0″ at the same time. Note that if the RCOCON bit is high, the TMRAH, TMRAL, TMRBH and TMRBL registers cannot be read or written to.



RCOCCR Register



RCOCR Register



Programming Considerations

As the 16-bit Timers have both low byte and high byte timer registers, accessing these registers is carried out in a specific way. It must be noted that when using instructions to preload data into the low byte registers, namely TMRAL or TMRAL, the data will only be placed into a low byte buffer and not directly into the low byte register. The actual transfer of the data into the low byte register is only carried out when a write to its associated high byte register, namely TMRAH or TMRBH, is executed. However, using instructions to preload data into the high byte timer register will result in the data being directly written to the high byte register. At the same time the data in the low byte buffer will be transferred into its associated low byte register. For this reason, when preloading data into the 16-bit timer registers, the low byte should be written first. It must also be noted that to read the contents of the low byte register, a read to the high byte register must first be executed to latch the contents of the low byte buffer into its associated low byte register. After this has been done, the low byte register can be read in the normal way. Note that reading the low byte timer register will only result in reading the previously latched contents of the low byte buffer and not the actual contents of the low byte timer register.

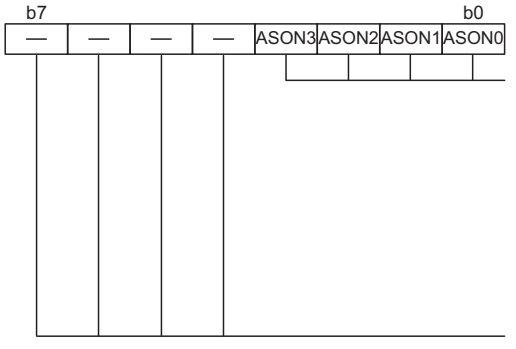
Program Example

External RC oscillation converter mode example program – Timer A overflow:

```
clr RCOCCR
mov a, 00000010b       ; Enable External RC oscillation mode and set Timer A overflow
mov RCOCR,a
clr intch.5            ; Clear External RC Oscillation Converter interrupt request flag
mov a, low (65536-1000); Give timer A initial value
mov tmral, a           ; Timer A count 1000 time and then overflow
mov a, high (65536-1000)
mov tmrah, a
mov a, 00h             ; Give timer B initial value
mov tmrbl, a
mov a, 00h
mov tmrbh, a
mov a, 00110000b       ; Timer A clock source=fSYS/4 and timer on
mov RCOCCR, a
p10:
clr wdt
Snz intch.5            ; Polling External RC Oscillation Converter interrupt request flag
jmp p10
clr intch.5            ; Clear External RC Oscillation Converter interrupt request flag
; Program continue
```

Analog Switch

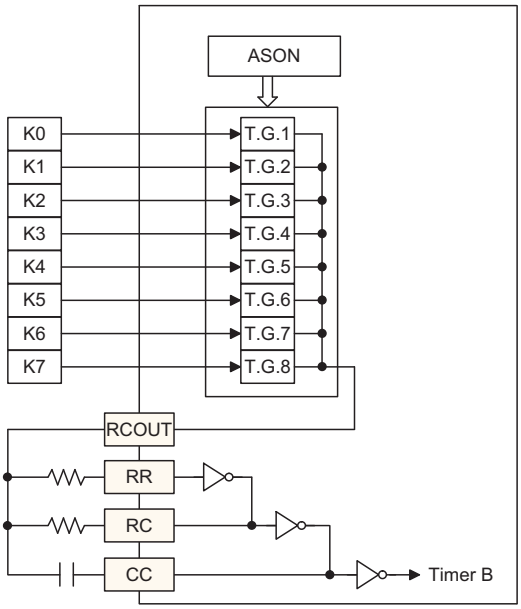
There are 8 analog switch lines in the microcontroller for K0~K7 for HT37B90/70/50/30 and the Analog Switch control register, which is mapped to the data memory. All of these Analog Switch lines can be used for touch key input keys.



Analog Switch Select

ASON3	ASON2	ASON1	ASON0	
0	0	0	0	K0 on, others off
0	0	0	1	K1 on, others off
0	0	1	0	K2 on, others off
0	0	1	1	K3 on, others off
0	1	0	0	K4 on, others off
0	1	0	1	K5 on, others off
0	1	1	0	K6 on, others off
0	1	1	1	K7 on, others off
1	X	X	X	All off, OSC off

Analog Switch Control Register – ASCR



Analog Switch

Analog to Digital Converter

The need to interface to real world analog signals is a common requirement for many electronic systems. However, to properly process these signals by a microcontroller, they must first be converted into digital signals by A/D converters. By integrating the A/D conversion electronic circuitry into the microcontroller, the need for external components is reduced significantly with the corresponding follow-on benefits of lower costs and reduced component space requirements.

A/D Overview

HT37B90/70/50/30 contains a 8-channel analog to digital converter which can directly interface to external analog signals, such as that from sensors or other control signals and convert these signals directly into either a 12-bit digital value.

Device	Input Channels	Conversion Bits	Input Pins
HT37B90/70, HT37B50/30	8	12	PB0~PB7

The following diagram shows the overall internal structure of the A/D converter, together with its associated registers.

A/D Converter Data Registers – ADR, ADRL, ADRH

HT37B90/70/50/30 have a 12-bit A/D converter, two registers are required, a high byte register, known as ADRH, and a low byte register, known as ADRL. After the conversion process takes place, these registers can be directly read by the microcontroller to obtain the digitized conversion value. HT37B90/70/50/30 use two A/D Converter Data Registers, note that only the high byte register ADRH utilizes its full 8-bit contents. The low byte register utilizes only 4 bit of its 8-bit contents as it contains only the lower 4 bit of the 12-bit converted value.

In the following tables, D0~D11 are the A/D conversion data result bits.

Register	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ADRL	D3	D2	D1	D0	—	—	—	—
ADRH	D11	D10	D9	D8	D7	D6	D5	D4

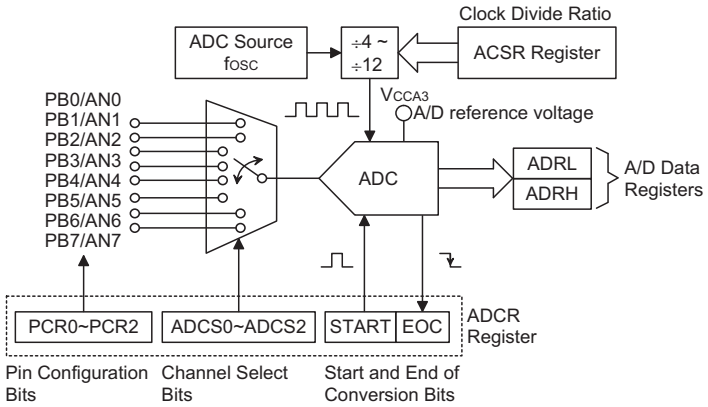
A/D Data Register

A/D Converter Control Register – ADCR

To control the function and operation of the A/D converter, a control register known as ADCR is provided. This 8-bit register defines functions such as the selection of which analog channel is connected to the internal A/D converter, which pins are used as analog inputs and which are used as normal I/Os as well as controlling the start function and monitoring the A/D converter end of conversion status.

One section of this register contains the bits ACS2~ACS0 which define the channel number. As each of the devices contains only one actual analog to digital converter circuit, each of the individual 8 analog inputs must be routed to the converter. It is the function of the ACS2~ACS0 bits in the ADCR register to determine which analog channel is actually connected to the internal A/D converter.

The ADCR control register also contains the PCR2~PCR0 bits which determine which pins on Port A are used as analog inputs for the A/D converter and which pins are to be used as normal I/O pins. Note that if the PCR2~PCR0 bits are all set to zero, then all the Port B pins will be setup as normal I/Os and the internal A/D converter circuitry will be powered off to reduce the power consumption.



A/D Converter Structure

The START bit in the ADCR register is used to start and reset the A/D converter. When the microcontroller sets this bit from low to high and then low again, an analog to digital conversion cycle will be initiated. When the START bit is brought from low to high but not low again, the EOCB bit in the ADCR register will be set high and the analog to digital converter will be reset. It is the START bit that is used to control the overall on/off operation of the internal analog to digital converter.

The EOCB bit in the ADCR register is used to indicate when the analog to digital conversion process is complete. This bit will be automatically cleared to zero by the microcontroller after a conversion cycle has ended. In addition, the corresponding A/D interrupt request flag will be set in the interrupt control register, and if the interrupts are enabled, an appropriate internal interrupt signal will be generated. This A/D internal interrupt signal will direct the program flow to the associated A/D internal interrupt address for processing. If the A/D internal interrupt is disabled, the microcontroller can be used to poll the EOCB bit in the ADCR register to check whether it has been cleared as an alternative method of detecting the end of an A/D conversion cycle.

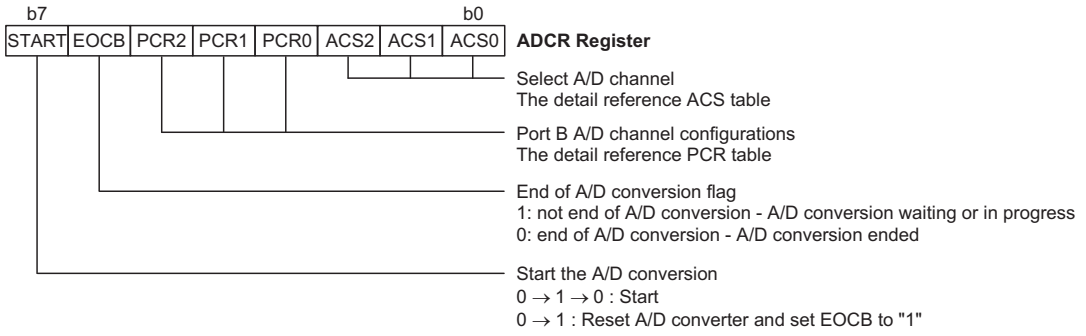
A/D Converter Clock Source Register – ACSR

The clock source for the A/D converter, which originates from the system clock f_{OSC}, is first divided by a division ratio, the value of which is determined by the ADCS1 and ADCS0 bits in the ACSR register.

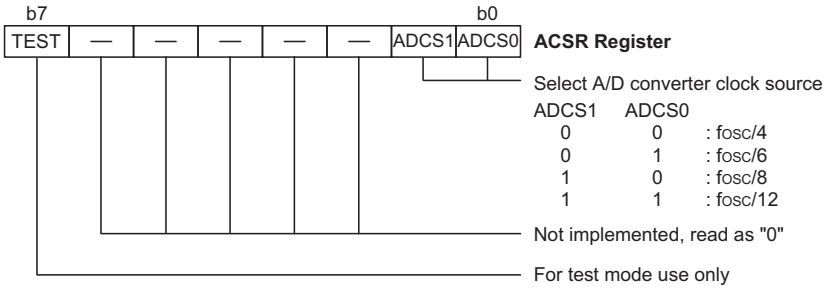
Although the A/D clock source is determined by the system clock f_{OSC}, and by bits ADCS1 and ADCS0, there are some limitations on the maximum A/D clock source speed that can be selected. Refer to the following table.

ACS3	ACS2	ACS1	ACS0	Analog Channel
0	0	0	0	AN0
0	0	0	1	AN1
0	0	1	0	AN2
0	0	1	1	AN3
0	1	0	0	AN4
0	1	0	1	AN5
0	1	1	0	AN6
0	1	1	1	AN7

ACS Table: A/D Channel Select Table



ADCR Register



ACSR Register

PCR2	PCR1	PCR0	7	6	5	4	3	2	1	0
0	0	0	PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0
0	0	1	PB7	PB6	PB5	PB4	PB3	PB2	PB1	AN0
0	1	0	PB7	PB6	PB5	PB4	PB3	PB2	AN1	AN0
0	1	1	PB7	PB6	PB5	PB4	PB3	AN2	AN1	AN0
1	0	0	PB7	PB6	PB5	PB4	AN3	AN2	AN1	PB0
1	0	1	PB7	PB6	AN5	AN4	AN3	AN2	AN1	AN0
1	1	0	PB7	PB6	AN5	AN4	AN3	AN2	AN1	AN0
1	1	1	AN7	AN6	AN5	AN4	AN3	AN2	AN1	AN0

PCR Table: Port A/D Channel Configuration Table

f_{osc}	A/D Clock Period (t_{AD})			
	ADCS1, ADCS0=00 (f_{osc}/4)	ADCS1, ADCS0=01 (f_{osc}/6)	ADCS1, ADCS0=10 (f_{osc}/8)	ADCS1, ADCS0=11 (f_{osc}/12)
8MHz	—	—	1μs	1.5μs
11.059MHz	—	—	—	1.08μs
12MHz	—	—	—	1μs

A/D Clock Period Examples

A/D Input Pins

All of the A/D analog input pins are pin-shared with the I/O pins on Port B. Bits PCR2~PCR0 in the ACSR registers, not configuration options, determine whether the input pins are setup as normal Port B input/output pins or whether they are setup as analog inputs. In this way, pins can be changed under program control to change their function from normal I/O operation to analog inputs and vice versa. Pull-high resistors, which are setup through configuration options, apply to the input pins only when they are used as normal I/O pins, if setup as A/D inputs the pull-high resistors will be automatically disconnected. Note that it is not necessary to first setup the A/D pin as an input in the PBC port control register to enable the A/D input, when the PCR2~PCR0 bits enable an A/D input, the status of the port control register will be overridden.

The VDD power supply pin is used as the A/D converter reference voltage, and as such analog inputs must not be allowed to exceed this value. Appropriate measures should also be taken to ensure that the VDD pin remains as stable and noise free as possible.

Initialising the A/D Converter

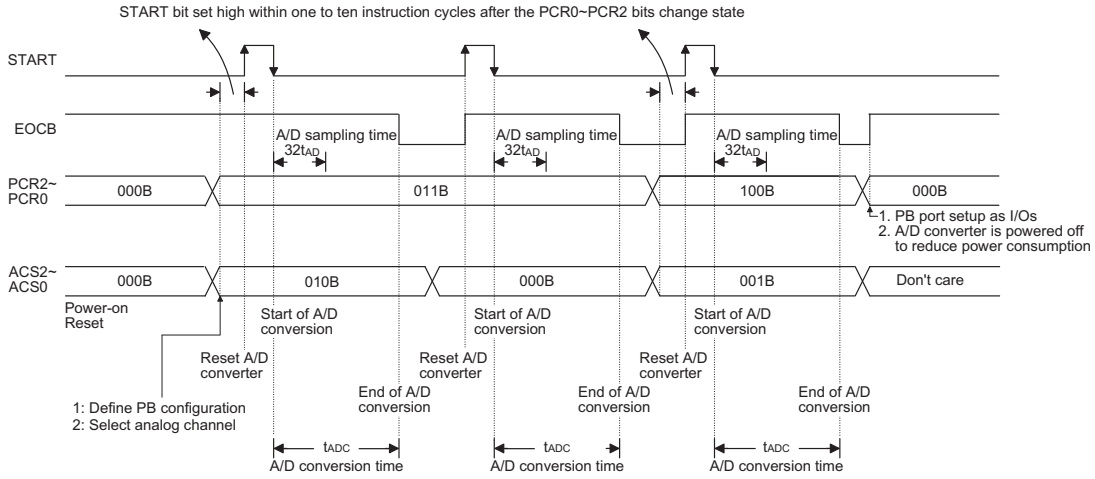
The internal A/D converter must be initialized in a special way. Each time the Port B A/D channel selection bits are modified by the program, the A/D converter must be re-initialised. If the A/D converter is not initialized after the channel selection bits are changed, the EOCB flag may have an undefined value, which may produce a false end of conversion signal. To initialize the A/D converter after the channel selection bits have changed, then, within a time frame of one to ten instruction cycles, the START bit in the ADCR register must first be set high and then immediately cleared to zero. This will ensure that the EOCB flag is correctly set to a high condition.

Summary of A/D Conversion Steps

The following summarizes the individual steps that should be executed in order to implement an A/D conversion process.

- Step 1
  Select the required A/D conversion clock by correctly programming bits ADCS1 and ADCS0 in the ACSR register.
- Step 2
  Select which channel is to be connected to the internal A/D converter by correctly programming the ACS2~ACS0 bits which are also contained in the ADCR register.
- Step 3
  Select which pins on Port B are to be used as A/D inputs and configure them as A/D input pins by correctly programming the PCR2~PCR0 bits in the ADCR register. Note that this step can be combined with Step 2 into ADCR registers programming operation.
- Step 4
  The analog to digital conversion process can now be initialised by setting the START bit in the ADCR register from "0" to "1" and then to "0" again. Note that this bit should have been originally set to "0".
- Step 5
  To check when the analog to digital conversion process is complete, the EOCB bit in the ADCR register can be polled. The conversion process is complete when this bit goes low. When this occurs the A/D data registers ADRL and ADRH can be read to obtain the conversion value.

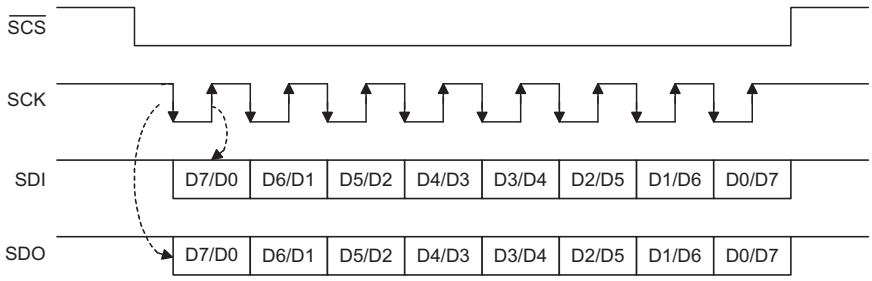
The following timing diagram shows graphically the various stages involved in an analog to digital conversion process and its associated timing.



A/D Conversion Timing

SPI Serial Interface

There are two SPI interfaces, with each interface containing four basic signals and pins. These are SDI (serial data input), SDO (serial data output), SCK (serial clock) and $\overline{SCS}$ (slave select pin).



	D7	D6	D5	D4	D3	D2	D1	D0	
SBCR	CKS	M1	M0	SBEN	MLS	CSEN	WCOL	TRF	SBCR : SERIAL BUS
DEFAULT	0	1	1	0	0	0	0	0	CONTROL REGISTER
SBDR	D7	D6	D5	D4	D3	D2	D1	D0	SBDR : SERIAL BUS
DEFAULT	U	U	U	U	U	U	U	U	DATA REGISTER

Note: "U" means unchanged.

SPI Timing

Two corresponding registers, SBCR and SBDR are unique to the serial interface and provide control, status, and data storage.

- SBCR: Serial bus control register
  - Bit7 (CKS) clock source selection: $f_{SIO}=f_{OSC}/2$, select as "0". $f_{SIO}=f_{OSC}$, select as "1".
  - Bit6 (M1), Bit5 (M0) master/slave mode and baud rate selection
    M1, M0:
    00 → MASTER MODE, BAUD RATE= f_{SIO}
    01 → MASTER MODE, BAUD RATE= $f_{SIO}/4$
    10 → MASTER MODE, BAUD RATE= $f_{SIO}/16$
    11 → SLAVE MODE
  - Bit4 (SBEN) → serial bus enable/disable (1/0)
    - Enable: ($\overline{SCS}$ dependent on CSEN bit)
      Disable → enable: SCK, SDI, SDO, $\overline{SCS}$= 0 ($\overline{SCK}$= "0") and waiting for writing data to SBDR (TXRX buffer)
      Master mode: write data to SBDR (TXRX buffer) start transmission/reception automatically
      Master mode: when the data has been transferred, set TRF
      Slave mode: when an SCK (and $\overline{SCS}$ dependent on CSEN) is received, data in the TXRX buffer is shifted-out and data on SDI is shifted-in.
    - Disable: SCK ($\overline{SCK}$), SDI, SDO, $\overline{SCS}$ floating
  - Bit3 (MLS) → MSB or LSB (1/0) shift first control bit
  - Bit2 (CSEN) → serial bus selection signal enable/disable ($\overline{SCS}$), when CSEN=0, SCSB is floating.
  - Bit1 (WCOL) → this bit is set to 1 if data is written to the SBDR register (TXRX buffer) when data is transferred, writing will be ignored if data is written to SBDR (TXRX buffer) when data is transferred.
  - Bit0 (TRF) → data transferred or data received used to generate an interrupt.

  Note: data reception is still in operation when the MCU enters the Power-down mode.
- SBDR: Serial bus data register
  Data written to SBDR → write data to the TXRX buffer only
  Data read from SBDR → read from SBDR only
  Operating Mode description:
  Master transmitter: clock transmission and data I/O started by writing to SBDR
  Master clock transmission initiated by writing to SBDR
  Slave transmitter: data I/O started by clock reception
  Slave receiver: data I/O started by clock reception

Clock polarity= rising ($\overline{SCK}$) or falling (SCK): 1 or 0 (mask option).

Modes	Operations
Master	1. Select CKS and select M1, M0 = 00,01,10 2. Select CSEN, MLS (the same as the slave) 3. Set SBEN 4. Writing data to SBDR → data is stored in TXRX buffer → output SCK (and $\overline{SCS}$) signals → go to step 5 → (SIO internal operation → data stored in TXRX buffer, and SDI data is shifted into TXRX buffer → data transferred, data in TXRX buffer is latched into SBDR) 5. Check WCOL; WCOL= 1 → clear WCOL and go to step 4; WCOL= 0 → go to step 6 6. Check TRF or waiting for SBI (serial bus interrupt) 7. Read data from SBDR 8. Clear TRF 9. Go to step 4
Slave	1. CKS don't care and select M1, M0= 11 2. Select CSEN, MLS (the same as the master) 3. Set SBEN 4. Writing data to SBDR → data is stored in TXRX buffer → waiting for master clock signal (and $\overline{SCS}$): SCK → go to step 5 → (SIO internal operations → SCK ($\overline{SCS}$) received → output data in TXRX buffer and SDI data is shifted into TXRX buffer → data transferred, data in TXRX buffer is latched into SBDR) 5. Check WCOL; WCOL= 1 → clear WCOL, go to step 4; WCOL= 0 → go to step 6 6. Check TRF or wait for SBI (serial bus interrupt) 7. Read data from SBDR 8. Clear TRF 9. Go to step 4

Operation of Serial Interface

WCOL: master/slave mode, set while writing to SBDR when data is transferring (transmitting or receiving) and this writing will then be ignored. WCOL function can be enabled/disabled by mask option. WCOL is set by SIO and cleared by users.

Data transmission and reception are still working when the MCU enters the HALT mode.

CPOL is used to select the clock polarity of SCK. It is a mask option.

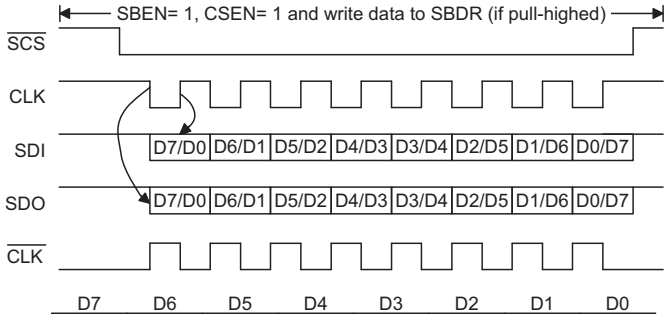
MLS: MSB or LSB first selection.

CSEN: chip select function enable/disable, CSEN=1 → $\overline{SCS}$ signal function is active. Master should output $\overline{SCS}$ signal before SCL signal is set and slave data transferring should be disabled (or enabled) before (after) $\overline{SCS}$ signal is received. CSEN= 0, $\overline{SCS}$ signal is not needed, $\overline{SCS}$ pin (master and slave) should be floating. CSEN has 2 options: CSEN mask option is used to enable/disable software CSEN function. If CSEN mask option is disabled, the software CSEN is always disabled. If CSEN mask option is enabled, software CSEN function can be used.

SBEN= 1 → serial bus standby; $\overline{SCS}$ (CSEN= 1) = 1; $\overline{SCS}$= floating (CSEN= 0); SDI= floating; SDO= 1; master SCK= output 1/0 (dependent on CPOL mask option), slave SCK= floating.

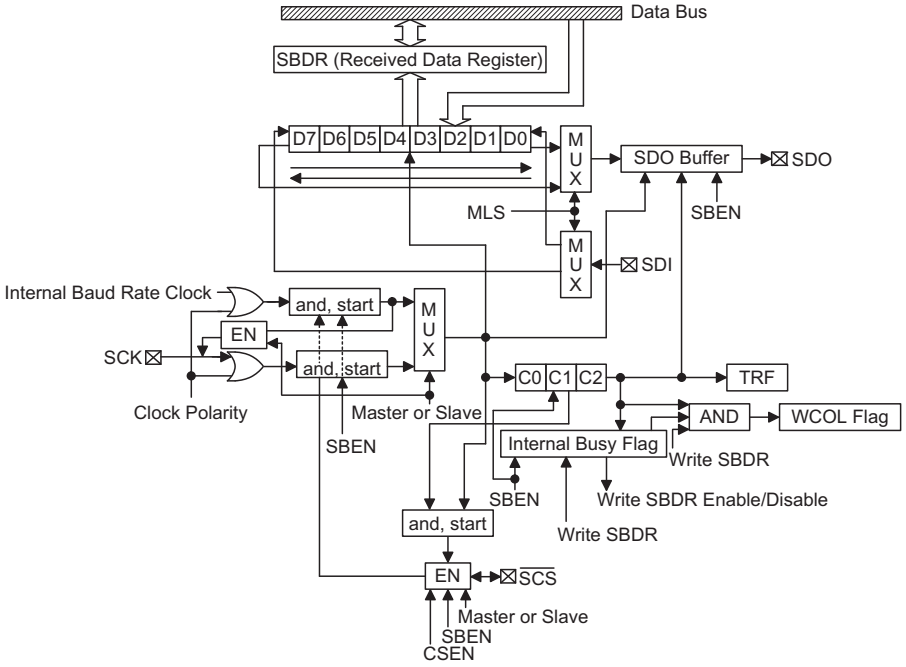
SBEN= 0 → serial bus disabled; $\overline{SCS}$=SDO=1, SDI=SCK= floating in master mode, SDI=SDO=SCK= floating, $\overline{SCS}$=1 in slave mode.

TRF is set by SIO and cleared by users. When data transfer (transmission and reception) is completed, TRF is set to generate SBI (serial bus interrupt).



	D7	D6	D5	D4	D3	D2	D1	D0
SBCR	CKS	M1	M0	SBEN	MLS	CSEN	WCOL	TRF
Default	0	1	1	0	0	0	0	0
SBDR	D7	D6	D5	D4	D3	D2	D1	D0
Default	u	u	u	u	u	u	u	u

Note: "u" means unchanged.



WCOL: set by SIO cleared by users
CSEN: enable/disable chip selection function pin
1. master mode 1/0: with/without $\overline{SCS}$ output function
2. slave mode 1/0: with/without $\overline{SCS}$ input control function
SBEN : enable/disable serial bus (0: initialize all status flags)
1. When SBEN= 0, all status flags should be initialized
2. When SBEN= 0, all SIO related function pins should stay at floating state
TRF 1: data transmitted or received, 0: data is transmitting or still not received
CPOL 1/0 : clock polarity rising/falling edge : mask option

IIS Function

D/A Converter Interface

The device provides the IIS serial data format to support the multiple D/A converters, one bit clock output and a word clock signal for left/right stereo serial data transmission.

Clock Signal

The bit clock output signals DCK are used to synchronize the IIS serial data. The word clock signal LOAD divides the serial data into left channel and right channel data for two-way audio output.

WAS Description

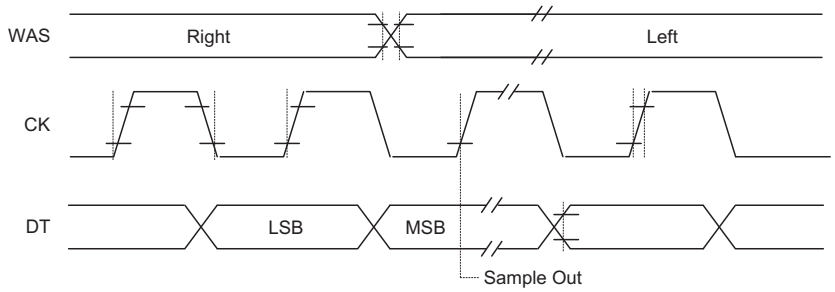
The word clock signal WAS is used for IIS serial data. The stereo serial data consists of 16-channel sound generator. On IIS format, a ″H″ state on WAS is used for the right channel, and a ″L″ state is used for the left channel.

CK Description

CK bit clock is the clock source for the signal. which can accept the IIS serial data format.

DT:Stereo Serial Data Format Transmit

The audio output data is in serial mode with 16 bit digital signal and LSB first output. There is a high sampling rate of 50kHz when the f_{OSC} is 12.8MHz and with two channel outputs for Right/Left channel. The device provides only one serial data format as IIS mode. The user could directly connect a D/A converter which can accept the IIS serial data format.



D/A Converter Timing

UART Bus Serial Interface

The device contains an integrated full-duplex asynchronous serial communications. The UART function can transmit and receive data serially by transferring a frame of data with eight per transmission. The UART function possesses its own internal interrupt which can be used to indicate when a reception occurs or when a transmission terminates. If the UART Clock always generates 31.25kHz, the value of BRGR should correspond to f_{OSC}.

Register Name	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
RS232C	—	—	PH	EN	—	—	BE	BF
TXD	D7	D6	D5	D4	D3	D2	D1	D0
RXD	D7	D6	D5	D4	D3	D2	D1	D0

Note: —: No function, read only, read as 0.
BF: RX buffer full; 0: Buffer not full; 1: Buffer full (Default=0)
BE: TX buffer empty; 0: Buffer not empty; 1: Buffer empty (Default=1)
EN: Enable/disable RS232 function; 0: disable; 1: enable
PH: RS232 input Pull High control (Default=1)
0= disable; 1= enable

BRGR								Prescaler
Decimal		Integer						
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	
1	1	0	0	0	0	0	0	0
1	1	1	0	0	0	0	1	2

BRGR								Prescaler
Decimal		Integer						
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	
1	1	0	1	0	0	1	0	60
1	1	1	1	0	0	0	0	3
1	1	0	0	1	0	0	0	61
1	1	1	0	1	0	0	0	41
1	1	0	1	1	0	0	0	55
1	1	1	1	1	0	0	0	4
1	1	0	0	0	1	0	0	62
1	1	1	0	0	1	0	0	33
1	1	0	1	0	1	0	0	50
1	1	1	1	0	1	0	0	42
1	1	0	0	1	1	0	0	56
1	1	1	0	1	1	0	0	20
1	1	0	1	1	1	0	0	36
1	1	1	1	1	1	0	0	5
1	1	0	0	0	0	1	0	63
1	1	1	0	0	0	1	0	53
1	1	0	1	0	0	1	0	31
1	1	1	1	0	0	1	0	34
1	1	0	0	1	0	1	0	51
1	1	1	0	1	0	1	0	13
1	1	0	1	1	0	1	0	15
1	1	1	1	1	0	1	0	43
1	1	0	0	0	1	1	0	57
1	1	1	0	0	1	1	0	17
1	1	0	1	0	1	1	0	28
1	1	1	1	0	1	1	0	21
1	1	0	0	1	1	1	0	37
1	1	1	0	1	1	1	0	24
1	1	0	1	1	1	1	0	45
1	1	1	1	1	1	1	0	6
1	1	0	0	0	0	0	1	0
1	1	1	0	0	0	0	1	59
1	1	0	1	0	0	0	1	40
1	1	1	1	0	0	0	1	54
1	1	0	0	1	0	0	1	32
1	1	1	0	1	0	0	1	49
1	1	0	1	1	0	0	1	19
1	1	1	1	1	0	0	1	35
1	1	0	0	0	1	0	1	52
1	1	1	0	0	1	0	1	30
1	1	0	1	0	1	0	1	12
1	1	1	1	0	1	0	1	14
1	1	0	0	1	1	0	1	16
1	1	1	0	1	1	0	1	27

BRGR								Prescaler
Decimal		Integer						
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	
1	1	0	1	1	1	0	1	23
1	1	1	1	1	1	0	1	44
1	1	0	0	0	0	1	1	58
1	1	1	0	0	0	1	1	39
1	1	0	1	0	0	1	1	48
1	1	1	1	0	0	1	1	18
1	1	0	0	1	0	1	1	29
1	1	1	0	1	0	1	1	11
1	1	0	1	1	0	1	1	26
1	1	1	1	1	0	1	1	22
1	1	0	0	0	1	1	1	38
1	1	1	0	0	1	1	1	47
1	1	0	1	0	1	1	1	10
1	1	1	1	0	1	1	1	25
1	1	0	0	1	1	1	1	46
1	1	1	0	1	1	1	1	9
1	1	0	1	1	1	1	1	8
1	1	1	1	1	1	1	1	7
0	0	1	1	1	1	1	1	7.75
0	1	1	1	1	1	1	1	7.25
1	0	1	1	1	1	1	1	7.5

BRGR Baud Rate Prescaler Table

Note: Baud Rate: 31.25kHz= f_{OSC}/(prescaler×32)

For HT37P00

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	f_{osc}	Baud Rate
1	1	0	1	1	1	1	1	8MHz	31.25kHz
1	1	1	0	1	0	1	1	11.059MHz	31.25kHz
0	0	0	1	0	1	0	1	12.8MHz	31.25kHz
1	1	0	0	1	1	0	1	16MHz	31.25kHz

UART Control Description

UART interface 3 pins: Mout , Min , Mth.

- Mout: transmit data output pin
- Min: receiver data input pin
- Mth: go through output pin

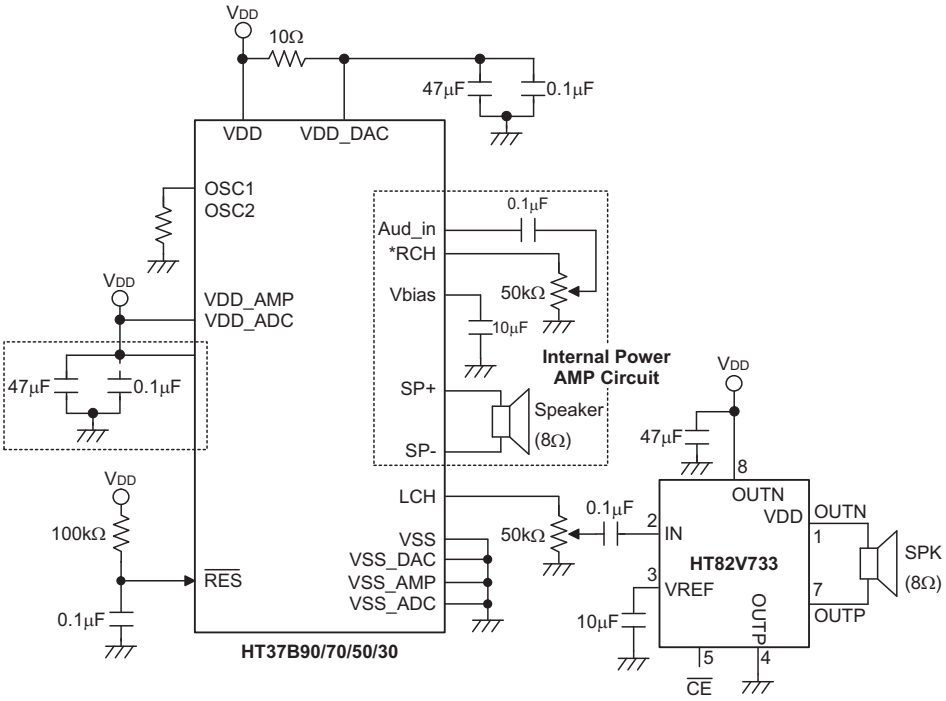
Note: Mout is hi-state when UART function initialized or finished to transmit.

Configuration Options

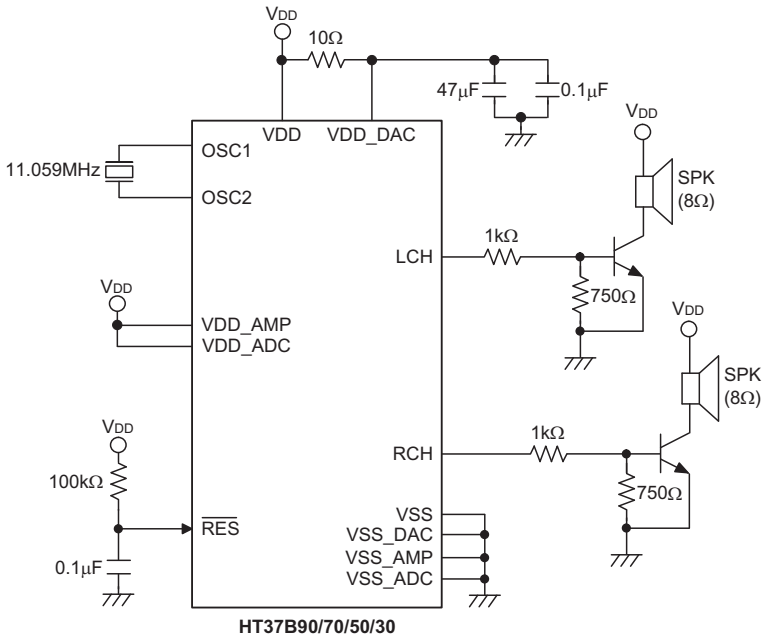
Configuration options refer to certain options within the MCU that are programmed into the device during the programming process. During the development process, these options are selected using the HT-MDS software development tools. As these options are programmed into the device using the hardware programming tools, once they are selected they cannot be changed later as the application software has no control over the configuration options. All options must be defined for proper system function, the details of which are shown in the table.

No.	Function
1	Watchdog Timer: enable or disable
2	Watchdog Timer clock source: T1 (f_{OSC}/8) or RC OSC
3	CLRWDT instructions: 1 or 2 instructions
4	PA0~PA7: wake-up enable or disable (bit option)
5	PA, PB, PC, PD and PE: pull-high enable or disable (port numbers are device dependent)
6	System oscillator: Xtal Mode or RC Mode
7	LVR function: enable or disable
8	LVR / LVD voltage selection 1. 3.0V/3.1V 2. 2.4V/2.5V
9	Share PIN - PA5/$\overline{INT}$: enable ($\overline{INT}$) or disable (PA5) Share PIN - PA6/TMR0: enable (TMR0) or disable (PA6) Share PIN - PA7/TMR1: enable (TMR1) or disable (PA7))
10	R to F Function : enable or normal I/O (PD0~PD3)
11	When R to F Enable, the Share PIN that using and depend on the Option. K0/PC0 K1/PC1 K2/PC2 K3/PC3 K4/PC4 K5/PC5 K6/PC6 K7/PC7
12	IIS/PD4~6 pin option: 1. IIS pin 2. PD4~PD6 are IO port
13	UART/PE0~PE2 pin option: 1. UART pin 2. PE0~PE2 are IO port
14	SPI/PE3~PE6 pin option: 1. SPI pin 2. PE3~PE6 are IO port

Application Circuits



Note: If user has used internal power AMP circuit, need to add two capacitances (47μF, 0.1μF) that be connected between VDD_AMP and VSS.



Instruction Set

Introduction

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

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

Instruction Timing

Most instructions are implemented within one instruction cycle. The exceptions to this are branch, call, or table read instructions where two instruction cycles are required. One instruction cycle is equal to 4 system clock cycles, therefore in the case of an 11.059MHz system oscillator, most instructions would be implemented within 0.723μs and branch or call instructions would be implemented within 1μs. Although instructions which require one more cycle to implement are generally limited to the JMP, CALL, RET, RETI and table read instructions, it is important to realize that any other instructions which involve manipulation of the Program Counter Low register or PCL will also take one more cycle to implement. As instructions which change the contents of the PCL will imply a direct jump to that new address, one more cycle will be required. Examples of such instructions would be ″CLR PCL″ or ″MOV PCL, A″. For the case of skip instructions, it must be noted that if the result of the comparison involves a skip operation then this will also take one more cycle, if no skip is involved then only one cycle is required.

Moving and Transferring Data

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

Arithmetic Operations

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

Logical and Rotate Operations

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

Branches and Control Transfer

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

Bit Operations

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

Table Read Operations

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

Other Operations

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

Instruction Set Summary

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

Table conventions:

x: Bits immediate data

m: Data Memory address

A: Accumulator

i: 0~7 number of bits

addr: Program memory address

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

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

Note: 1. For skip instructions, if the result of the comparison involves a skip then two cycles are required, if no skip takes place only one cycle is required.
2. Any instruction which changes the contents of the PCL will also require 2 cycles for execution.
3. For the ″CLR WDT1″ and ″CLR WDT2″ instructions the TO and PDF flags may be affected by the execution status. The TO and PDF flags are cleared after both ″CLR WDT1″ and ″CLR WDT2″ instructions are consecutively executed. Otherwise the TO and PDF flags remain unchanged.

Instruction Definition

ADC A,[m] Add Data Memory to ACC with Carry

Description The contents of the specified Data Memory, Accumulator and the carry flag are added. The result is stored in the Accumulator.

Operation ACC ← ACC + [m] + C

Affected flag(s) OV, Z, AC, C

ADCM A,[m] Add ACC to Data Memory with Carry

Description The contents of the specified Data Memory, Accumulator and the carry flag are added. The result is stored in the specified Data Memory.

Operation [m] ← ACC + [m] + C

Affected flag(s) OV, Z, AC, C

ADD A,[m] Add Data Memory to ACC

Description The contents of the specified Data Memory and the Accumulator are added. The result is stored in the Accumulator.

Operation ACC ← ACC + [m]

Affected flag(s) OV, Z, AC, C

ADD A,x Add immediate data to ACC

Description The contents of the Accumulator and the specified immediate data are added. The result is stored in the Accumulator.

Operation ACC ← ACC + x

Affected flag(s) OV, Z, AC, C

ADDM A,[m] Add ACC to Data Memory

Description The contents of the specified Data Memory and the Accumulator are added. The result is stored in the specified Data Memory.

Operation [m] ← ACC + [m]

Affected flag(s) OV, Z, AC, C

AND A,[m] Logical AND Data Memory to ACC

Description Data in the Accumulator and the specified Data Memory perform a bitwise logical AND operation. The result is stored in the Accumulator.

Operation ACC ← ACC ″AND″ [m]

Affected flag(s) Z

AND A,x Logical AND immediate data to ACC

Description Data in the Accumulator and the specified immediate data perform a bitwise logical AND operation. The result is stored in the Accumulator.

Operation ACC ← ACC ″AND″ x

Affected flag(s) Z

ANDM A,[m] Logical AND ACC to Data Memory

Description Data in the specified Data Memory and the Accumulator perform a bitwise logical AND operation. The result is stored in the Data Memory.

Operation [m] ← ACC ″AND″ [m]

Affected flag(s) Z

CALL addr Subroutine call

Description Unconditionally calls a subroutine at the specified address. The Program Counter then increments by 1 to obtain the address of the next instruction which is then pushed onto the stack. The specified address is then loaded and the program continues execution from this new address. As this instruction requires an additional operation, it is a two cycle instruction.

Operation Stack ← Program Counter + 1
Program Counter ← addr

Affected flag(s) None

CLR [m] Clear Data Memory

Description Each bit of the specified Data Memory is cleared to 0.

Operation [m] ← 00H

Affected flag(s) None

CLR [m].i Clear bit of Data Memory

Description Bit i of the specified Data Memory is cleared to 0.

Operation [m].i ← 0

Affected flag(s) None

CLR WDT Clear Watchdog Timer

Description The TO, PDF flags and the WDT are all cleared.

Operation WDT cleared
TO ← 0
PDF ← 0

Affected flag(s) TO, PDF

CLR WDT1 Pre-clear Watchdog Timer

Description The TO, PDF flags and the WDT are all cleared. Note that this instruction works in conjunction with CLR WDT2 and must be executed alternately with CLR WDT2 to have effect. Repetitively executing this instruction without alternately executing CLR WDT2 will have no effect.

Operation WDT cleared
TO ← 0
PDF ← 0

Affected flag(s) TO, PDF

CLR WDT2 Pre-clear Watchdog Timer

Description The TO, PDF flags and the WDT are all cleared. Note that this instruction works in conjunction with CLR WDT1 and must be executed alternately with CLR WDT1 to have effect. Repetitively executing this instruction without alternately executing CLR WDT1 will have no effect.

Operation WDT cleared
TO ← 0
PDF ← 0

Affected flag(s) TO, PDF

CPL [m] Complement Data Memory

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

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

Affected flag(s) Z

CPLA [m] Complement Data Memory with result in ACC

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

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

Affected flag(s) Z

DAA [m] Decimal-Adjust ACC for addition with result in Data Memory

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

Operation $[m] \leftarrow ACC + 00H$ or
$[m] \leftarrow ACC + 06H$ or
$[m] \leftarrow ACC + 60H$ or
$[m] \leftarrow ACC + 66H$

Affected flag(s) C

DEC [m] Decrement Data Memory

Description Data in the specified Data Memory is decremented by 1.

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

Affected flag(s) Z

DECA [m] Decrement Data Memory with result in ACC

Description Data in the specified Data Memory is decremented by 1. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.

Operation $ACC \leftarrow [m] - 1$

Affected flag(s) Z

HALT Enter power down mode

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

Operation $TO \leftarrow 0$
$PDF \leftarrow 1$

Affected flag(s) TO, PDF

INC [m]	Increment Data Memory
Description	Data in the specified Data Memory is incremented by 1.
Operation	[m] ← [m] + 1
Affected flag(s)	Z
INCA [m]	Increment Data Memory with result in ACC
Description	Data in the specified Data Memory is incremented by 1. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.
Operation	ACC ← [m] + 1
Affected flag(s)	Z
JMP addr	Jump unconditionally
Description	The contents of the Program Counter are replaced with the specified address. Program execution then continues from this new address. As this requires the insertion of a dummy instruction while the new address is loaded, it is a two cycle instruction.
Operation	Program Counter ← addr
Affected flag(s)	None
MOV A,[m]	Move Data Memory to ACC
Description	The contents of the specified Data Memory are copied to the Accumulator.
Operation	ACC ← [m]
Affected flag(s)	None
MOV A,x	Move immediate data to ACC
Description	The immediate data specified is loaded into the Accumulator.
Operation	ACC ← x
Affected flag(s)	None
MOV [m],A	Move ACC to Data Memory
Description	The contents of the Accumulator are copied to the specified Data Memory.
Operation	[m] ← ACC
Affected flag(s)	None
NOP	No operation
Description	No operation is performed. Execution continues with the next instruction.
Operation	No operation
Affected flag(s)	None
OR A,[m]	Logical OR Data Memory to ACC
Description	Data in the Accumulator and the specified Data Memory perform a bitwise logical OR operation. The result is stored in the Accumulator.
Operation	ACC ← ACC ″OR″ [m]
Affected flag(s)	Z

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

RLC [m] Rotate Data Memory left through Carry

Description The contents of the specified Data Memory and the carry flag are rotated left by 1 bit. Bit 7 replaces the Carry bit and the original carry flag is rotated into bit 0.

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

Affected flag(s) C

RLCA [m] Rotate Data Memory left through Carry with result in ACC

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

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

Affected flag(s) C

RR [m] Rotate Data Memory right

Description The contents of the specified Data Memory are rotated right by 1 bit with bit 0 rotated into bit 7.

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

Affected flag(s) None

RRA [m] Rotate Data Memory right with result in ACC

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

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

Affected flag(s) None

RRC [m] Rotate Data Memory right through Carry

Description The contents of the specified Data Memory and the carry flag are rotated right by 1 bit. Bit 0 replaces the Carry bit and the original carry flag is rotated into bit 7.

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

Affected flag(s) C

RRCA [m] Rotate Data Memory right through Carry with result in ACC

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

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

Affected flag(s) C

SBC A,[m] Subtract Data Memory from ACC with Carry

Description The contents of the specified Data Memory and the complement of the carry flag are subtracted from the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.

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

Affected flag(s) OV, Z, AC, C

SBCM A,[m] Subtract Data Memory from ACC with Carry and result in Data Memory

Description The contents of the specified Data Memory and the complement of the carry flag are subtracted from the Accumulator. The result is stored in the Data Memory. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.

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

Affected flag(s) OV, Z, AC, C

SDZ [m] Skip if decrement Data Memory is 0

Description The contents of the specified Data Memory are first decremented by 1. If the result is 0 the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation $[m] \leftarrow [m] - 1$
Skip if [m] = 0

Affected flag(s) None

SDZA [m] Skip if decrement Data Memory is zero with result in ACC

Description The contents of the specified Data Memory are first decremented by 1. If the result is 0, the following instruction is skipped. The result is stored in the Accumulator but the specified Data Memory contents remain unchanged. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0, the program proceeds with the following instruction.

Operation $ACC \leftarrow [m] - 1$
Skip if ACC = 0

Affected flag(s) None

SET [m] Set Data Memory

Description Each bit of the specified Data Memory is set to 1.

Operation $[m] \leftarrow FFH$

Affected flag(s) None

SET [m].i Set bit of Data Memory

Description Bit i of the specified Data Memory is set to 1.

Operation $[m].i \leftarrow 1$

Affected flag(s) None

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

SWAP [m] Swap nibbles of Data Memory

Description The low-order and high-order nibbles of the specified Data Memory are interchanged.

Operation [m].3~[m].0 ↔ [m].7 ~ [m].4

Affected flag(s) None

SWAPA [m] Swap nibbles of Data Memory with result in ACC

Description The low-order and high-order nibbles of the specified Data Memory are interchanged. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.

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

Affected flag(s) None

SZ [m] Skip if Data Memory is 0

Description If the contents of the specified Data Memory is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation Skip if [m] = 0

Affected flag(s) None

SZA [m] Skip if Data Memory is 0 with data movement to ACC

Description The contents of the specified Data Memory are copied to the Accumulator. If the value is zero, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation ACC ← [m]
Skip if [m] = 0

Affected flag(s) None

SZ [m].i Skip if bit i of Data Memory is 0

Description If bit i of the specified Data Memory is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0, the program proceeds with the following instruction.

Operation Skip if [m].i = 0

Affected flag(s) None

TABRDC [m] Read table (current page) to TBLH and Data Memory

Description The low byte of the program code (current page) addressed by the table pointer (TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.

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

Affected flag(s) None

TABRDL [m] Read table (last page) to TBLH and Data Memory

Description The low byte of the program code (last page) addressed by the table pointer (TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.

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

Affected flag(s) None

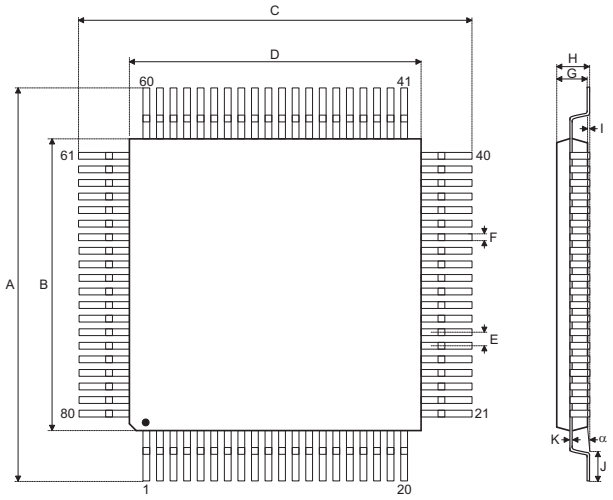
XOR A,[m]	Logical XOR Data Memory to ACC
Description	Data in the Accumulator and the specified Data Memory perform a bitwise logical XOR operation. The result is stored in the Accumulator.
Operation	ACC ← ACC ″XOR″ [m]
Affected flag(s)	Z

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

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

Package Information

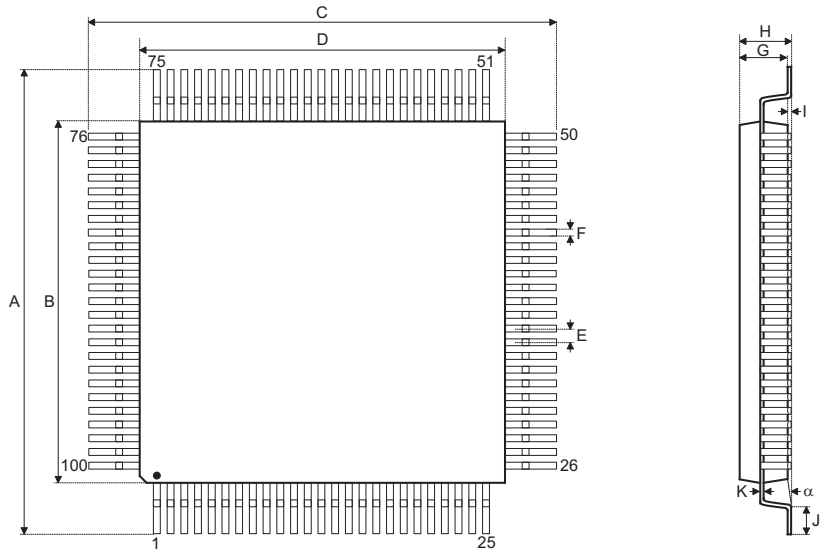
80-pin LQFP (10mm×10mm) Outline Dimensions



Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	—	0.472 BSC	—
B	—	0.394 BSC	—
C	—	0.472 BSC	—
D	—	0.394 BSC	—
E	—	0.016 BSC	—
F	0.007	0.009	0.0111
G	0.053	0.055	0.057
H	—	—	0.063
I	0.002	—	0.006
J	0.018	0.024	0.030
K	0.004	—	0.008
α	0°	—	7°

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	—	12 BSC	—
B	—	10 BSC	—
C	—	12 BSC	—
D	—	10 BSC	—
E	—	0.4 BSC	—
F	0.13	0.18	0.23
G	1.35	1.40	1.45
H	—	—	1.60
I	0.05	—	0.15
J	0.45		0.75
K	0.09	—	0.20
α	0°	—	7°

100-pin LQFP (14mm×14mm) Outline Dimensions



Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	—	0.630 BSC	—
B	—	0.551 BSC	—
C	—	0.630 BSC	—
D	—	0.551 BSC	—
E	—	0.020 BSC	—
F	0.007	0.009	0.011
G	0.053	0.055	0.057
H	—	—	0.063
I	0.002	—	0.006
J	0.018	0.024	0.030
K	0.004	—	0.008
α	0°	—	7°

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	—	16 BSC	—
B	—	14 BSC	—
C	—	16 BSC	—
D	—	14 BSC	—
E	—	0.50 BSC	—
F	0.17	0.22	0.27
G	1.35	1.4	1.45
H	—	—	1.60
I	0.05	—	0.15
J	0.45	0.06	0.75
K	0.09	—	0.20
α	0°	—	7°