HOLTEK

HT46R24/HT46C24
A/D Type 8-Bit MCU

Features

- Operating voltage:
  f_{SYS}=4MHz: 2.2V~5.5V
  f_{SYS}=8MHz: 3.3V~5.5V
- 40 bidirectional I/O lines (max.)
- 1 interrupt input shared with an I/O line
- Two 16-bit programmable timer/event counter with overflow interrupt
- On-chip crystal and RC oscillator
- Watchdog Timer
- 8192×16 program memory
- 384×8 data memory RAM
- Supports PFD for sound generation
- HALT function and wake-up feature reduce power consumption
- Up to 0.5μs instruction cycle with 8MHz system clock at V_{DD}=5V
- 16-level subroutine nesting
- 8 channels 10-bit resolution A/D converter
- 4-channel 8-bit PWM output shared with four I/O lines
- Bit manipulation instruction
- 16-bit table read instruction
- 63 powerful instructions
- All instructions in one or two machine cycles
- Low voltage reset function
- I^2C Bus (slave mode)
- 28-pin SKDIP/SOP, 48-pin SSOP package

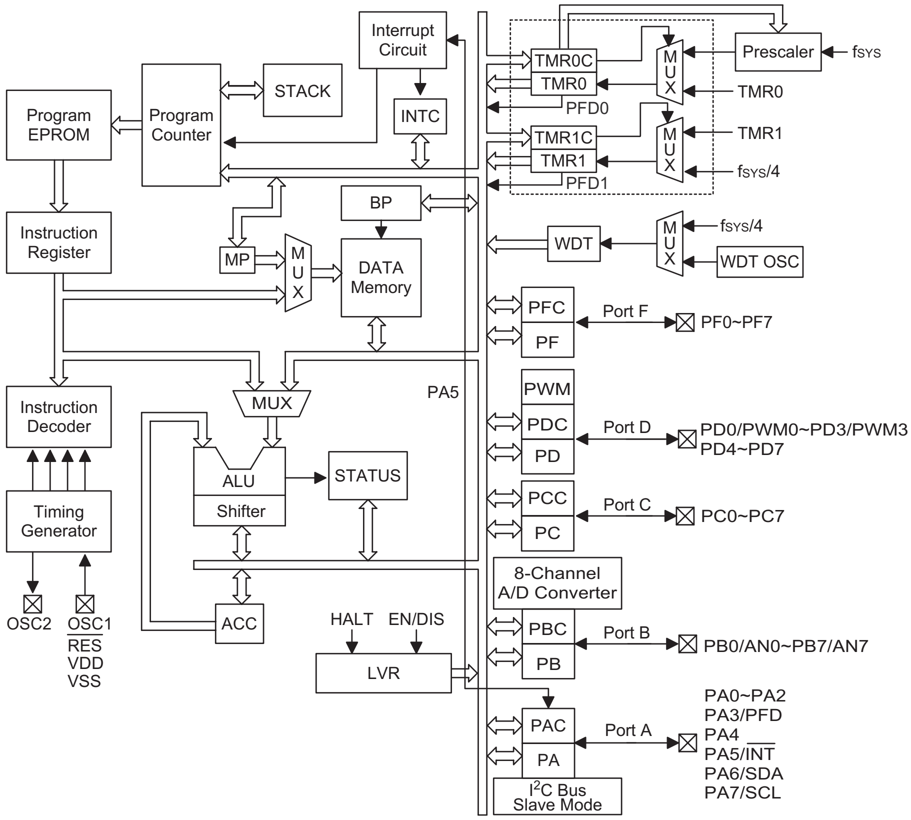
General Description

The HT46R24/HT46C24 are 8-bit, high performance, RISC architecture microcontroller devices specifically designed for A/D applications that interface directly to analog signals, such as those from sensors. The mask version HT46C24 is fully pin and functionally compatible with the OTP version HT46R24 device.

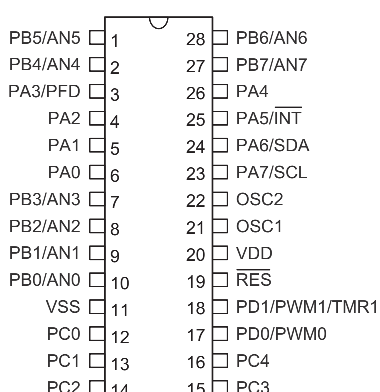
The advantages of low power consumption, I/O flexibility, programmable frequency divider, timer functions, oscillator options, multi-channel A/D Converter, Pulse Width Modulation function, I^2C interface, HALT and wake-up functions, enhance the versatility of these devices to suit a wide range of A/D application possibilities such as sensor signal processing, motor driving, industrial control, consumer products, subsystem controllers, etc.

I^2C is a trademark of Philips Semiconductors.

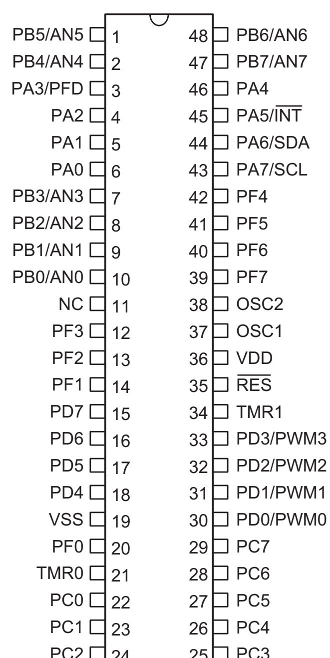
Block Diagram



Pin Assignment



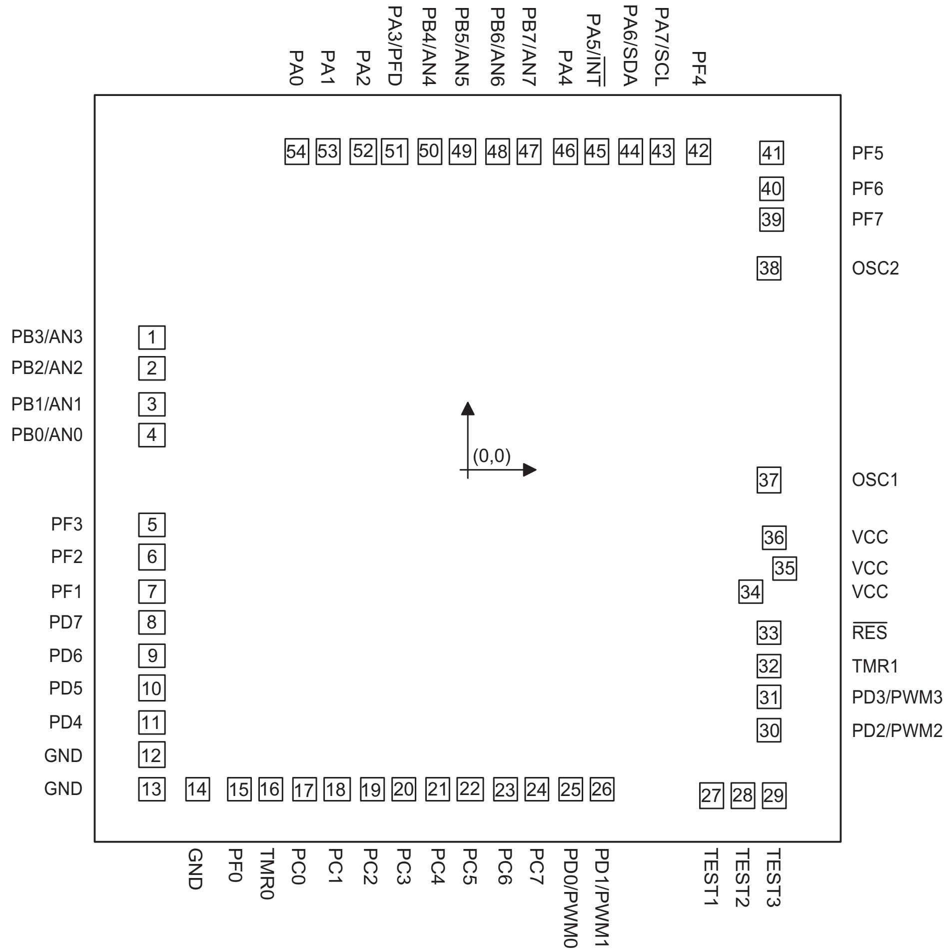
HT46R24/HT46C24
— 28 SKDIP-A/SOP-A



HT46R24/HT46C24
— 48 SSOP-A

Pad Assignment

HT46C24



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

Pin Description

Pin Name	I/O	Options	Description
PA0~PA2 PA3/PFD PA4 PA5/INT PA6/SDA PA7/SCL	I/O	Pull-high Wake-up PA3 or PFD I/O or Serial Bus	Bidirectional 8-bit input/output port. Each bit can be configured as wake-up input by option (bit option). Software instructions determine the CMOS output or Schmitt trigger input with or without pull-high resistor (determined by pull-high options: bit option). The PFD and INT are pin-shared with PA3 and PA5, respectively. Once the I^2C Bus function is used, the internal registers related to PA6 and PA7 cannot be used.
PB0/AN0 PB1/AN1 PB2/AN2 PB3/AN3 PB4/AN4 PB5/AN5 PB6/AN6 PB7/AN7	I/O	Pull-high	Bidirectional 8-bits input/output port. Software instructions determine the CMOS output, Schmitt trigger input with or without pull-high resistor (determined by pull-high option: bit option) or A/D input. Once a PB line is selected as an A/D input (by using software control), the I/O function and pull-high resistor are automatically disabled.

Pin Name	I/O	Options	Description
PC0~PC4 (28-pin package only) PC0~PC7 (48-pin package only)	I/O	Pull-high	Bidirectional 8-bit input/output port. Software instructions determine the CMOS output, Schmitt trigger input with or without pull-high resistor (determine by pull-high option: byte option).
PD0/PWM0 PD1/PWM1/TMR1 (28-pin package only)	I/O	Pull-high PWM	Bidirectional 8-bit input/output port. Software instructions determine the CMOS output, Schmitt trigger input with or without a pull-high resistor. The PWM0 output function is pin-shared with PD0. The PWM1 output function is pin-shared with PD1 and TMR1. (determined by pull-high option: byte option)
PD0/PWM0 PD1/PWM1 PD2/PWM2 PD3/PWM3 PD4~PD7 (48-pin package only)	I/O	Pull-high PWM	Bidirectional 8-bit input/output port. Software instructions determine the CMOS output, Schmitt trigger input with or without a pull-high resistor (determined by pull-high option: byte option). The PWM0/PWM1/PWM2/PWM3 output function are pin-shared with PD0/PD1/PD2/PD3 (depending on the PWM options).
PF0~PF7 (48-pin package only)	I/O	Pull-high	Bidirectional 8-bit input/output port. Software instructions determine the CMOS output, Schmitt trigger input with or without pull-high resistor (determine by pull-high option: byte option).
TMR0	I	—	Timer/Event Counter 0 Schmitt trigger input (without pull-high resistor)
TMR1 (48-pin package only)	I	—	Timer/Event Counter 1 Schmitt trigger input (without pull-high resistor).
$\overline{RES}$	I	—	Schmitt trigger reset input, active low
VSS	—	—	Negative power supply, ground
VDD	—	—	Positive power supply
OSC1 OSC2	I O	Crystal or RC	OSC1 and OSC2 are connected to an RC network or a crystal (by options) for the internal system clock. In the case of RC operation, OSC2 is the output terminal for 1/4 system clock.
TEST1~3	I	—	Test mode input pin it disconnects in normal operation.
NC	—	—	No connection

Absolute Maximum Ratings

Supply Voltage ....V_{SS}–0.3V to V_{SS}+6.0V

Input Voltage.............................V_{SS}–0.3V to V_{DD}+0.3V

Storage Temperature ....–50°C to 125°C

Operating Temperature..........................–40°C to 85°C

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

D.C. Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	Operating Voltage	—	f_{SYS}=4MHz	2.2	—	5.5	V
			f_{SYS}=8MHz	3.3	—	5.5	V
I_{DD1}	Operating Current (Crystal OSC)	3V	No load, f_{SYS}=4MHz ADC disable	—	0.6	1.5	mA
		5V		—	2	4	mA
I_{DD2}	Operating Current (RC OSC)	3V	No load, f_{SYS}=4MHz ADC disable	—	0.8	1.5	mA
		5V		—	2.5	4	mA

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{DD3}	Operating Current	5V	No load, f_{SYS}=8MHz ADC disable	—	3	5	mA
I_{STB1}	Standby Current (WDT Enabled)	3V	No load, system HALT	—	—	5	μA
		5V		—	—	10	μA
I_{STB2}	Standby Current (WDT Disabled)	3V	No load, system HALT	—	—	1	μA
		5V		—	—	2	μA
V_{IL1}	Input Low Voltage for I/O Ports, TMR0, TMR1 and $\overline{INT}$	—	—	0	—	$0.3V_{DD}$	V
V_{IH1}	Input High Voltage for I/O Ports, TMR0, TMR1 and $\overline{INT}$	—	—	$0.7V_{DD}$	—	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
V_{LVR}	Low Voltage Reset Voltage	—	—	2.7	3	3.3	V
I_{OL}	I/O Port Sink Current	3V	$V_{OL}=0.1V_{DD}$	4	8	—	mA
		5V		10	20	—	mA
I_{OH}	I/O Port Source Current	3V	$V_{OH}=0.9V_{DD}$	–2	–4	—	mA
		5V		–5	–10	—	mA
R_{PH}	Pull-high Resistance	3V	—	20	60	100	kΩ
		5V		10	30	50	kΩ
V_{AD}	A/D Input Voltage	—	—	0	—	V_{DD}	V
E_{AD}	A/D Conversion Error	—	—	—	±0.5	±1	LSB
I_{ADC}	Additional Power Consumption if A/D Converter is Used	3V	—	—	0.5	1	mA
		5V		—	1.5	3	mA

A.C. Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
f_{SYS}	System Clock	—	2.2V~5.5V	400	—	4000	kHz
		—	3.3V~5.5V	400	—	8000	kHz
f_{TIMER}	Timer I/P Frequency (TMR0/TMR1)	—	2.2V~5.5V	0	—	4000	kHz
		—	3.3V~5.5V	0	—	8000	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	—	Wake-up from HALT	—	1024	—	$*t_{SYS}$
t_{INT}	Interrupt Pulse Width	—	—	1	—	—	μs
t_{AD}	A/D Clock Period	—	—	1	—	—	μs
t_{ADC}	A/D Conversion Time	—	—	—	76	—	t_{AD}
t_{ADCS}	A/D Sampling Time	—	—	—	32	—	t_{AD}
t_{IIC}	I^2C Bus Clock Period	—	Connect to external pull-high resistor 2kΩ	64	—	—	$*t_{SYS}$

Note: $*t_{SYS}=1/f_{SYS}$

Functional Description

Execution Flow

The system clock is derived from either a crystal or an RC oscillator. It is internally divided into four non-overlapping clocks. One instruction cycle consists of four system clock cycles. Instruction fetching and execution are pipelined in such a way that a fetch takes one instruction cycle while decoding and execution takes the next instruction cycle. The pipelining scheme makes it possible for each instruction to be effectively executed in a cycle. If an instruction changes the value of the program counter, two cycles are required to complete the instruction.

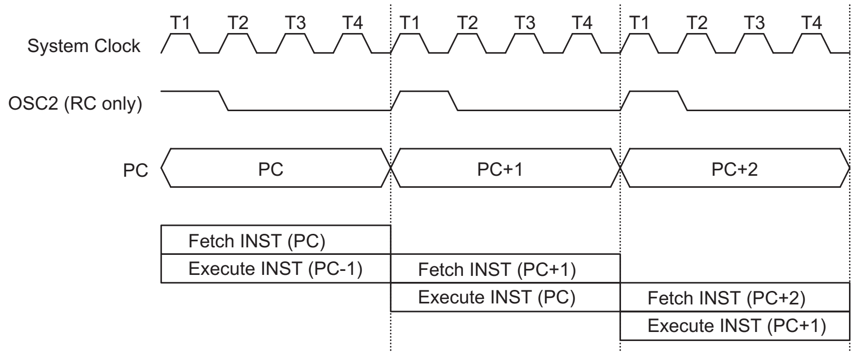
Program Counter – PC

The program counter (PC) is 13 bits wide and it controls the sequence in which the instructions stored in the program ROM are executed. The contents of the PC can specify a maximum of 8192 addresses. After accessing a program memory word to fetch an instruction code, the value of the PC is incremented by 1. The PC then points to the memory word containing the next instruction code. When executing a jump instruction, conditional skip execution, loading a PCL register, a subroutine call, an initial reset, an internal interrupt, an external interrupt, or returning from a subroutine, the PC manipulates the program transfer by loading the address corresponding to each instruction.

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

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

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



Execution Flow

Mode	Program Counter												
	*12	*11	*10	*9	*8	*7	*6	*5	*4	*3	*2	*1	*0
Initial Reset	0	0	0	0	0	0	0	0	0	0	0	0	0
External Interrupt	0	0	0	0	0	0	0	0	0	0	1	0	0
Timer/Event Counter 0 Overflow	0	0	0	0	0	0	0	0	0	1	0	0	0
Timer/Event Counter 1 Overflow	0	0	0	0	0	0	0	0	0	1	1	0	0
A/D Converter Interrupt	0	0	0	0	0	0	0	0	1	0	0	0	0
I^2C Bus Interrupt	0	0	0	0	0	0	0	0	1	0	1	0	0
Skip	PC+2												
Loading PCL	*12	*11	*10	*9	*8	@7	@6	@5	@4	@3	@2	@1	@0
Jump, Call Branch	#12	#11	#10	#9	#8	#7	#6	#5	#4	#3	#2	#1	#0
Return from Subroutine	S12	S11	S10	S9	S8	S7	S6	S5	S4	S3	S2	S1	S0

Program Counter

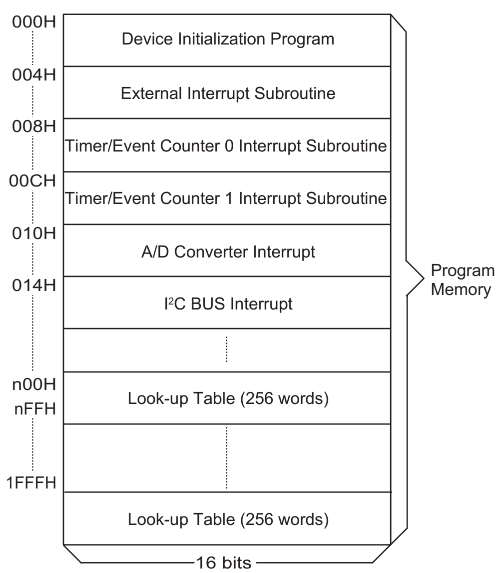
Note: *12~*0: Program counter bits
#12~#0: Instruction code bits
S12~S0: Stack register bits
@7~@0: PCL bits

Program Memory – EPROM

The program memory (EPROM) is used to store the program instructions which are to be executed. It also contains data, table, and interrupt entries, and is organized into 8192×16 bits which are addressed by the PC and table pointer.

Certain locations in the ROM are reserved for special usage:

- Location 000H
  Location 000H is reserved for program initialization. After chip reset, the program always begins execution at this location.
- Location 004H
  Location 004H is reserved for the external interrupt service program. If the $\overline{INT}$ input pin is activated, and the interrupt is enabled, and the stack is not full, the program begins execution at location 004H.
- Location 008H
  Location 008H 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.



Note: n ranges from 0 to 1F

Program Memory

- Location 00CH
  Location 00CH 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
  Location 010H is reserved for the A/D converter interrupt service program. If an A/D converter interrupt results from an end of A/D conversion, and if the interrupt is enabled and the stack is not full, the program begins execution at location 010H.
- Location 014H
  This area is reserved for the I^2C Bus interrupt service program. If the I^2C Bus interrupt resulting from a slave address is match or completed one byte of data transfer, and if the interrupt is enable and the stack is not full, the program begins execution at location 014H.
- Table location
  Any location in the ROM can be used as a look-up table. The instructions ″TABRDC [m]″ (the current page, page=256 words) and ″TABRDL [m]″ (the last page) transfer the contents of the lower-order byte to the specified data memory, and the contents of the higher-order byte to TBLH (Table Higher-order byte register) (08H). Only the destination of the lower-order byte in the table is well-defined; the other bits of the table word are all transferred to the lower portion of TBLH. The TBLH is read only, and the table pointer (TBLP) is a read/write register (07H), indicating the table location. Before accessing the table, the location should be placed in TBLP. All the table related instructions require 2 cycles to complete the operation. These areas may function as a normal ROM depending upon the users requirements

Stack Register – STACK

This is a special part of the memory which is used to save the contents of the program counter (PC) only. The stack is organized into 16 levels and is neither part of the data nor part of the program space, and is neither readable nor writeable. The activated level is indexed by the stack pointer (SP) and is neither readable nor writeable. At the state of a subroutine call or an interrupt acknowledgment, the contents of the program counter are pushed onto the stack. At the end of the subroutine or an interrupt routine, signaled by a return instruction (RET or RETI), the program counter is restored to its previous

Instruction	Table Location												
	*12	*11	*10	*9	*8	*7	*6	*5	*4	*3	*2	*1	*0
TABRDC [m]	P12	P11	P10	P9	P8	@7	@6	@5	@4	@3	@2	@1	@0
TABRDL [m]	1	1	1	1	1	@7	@6	@5	@4	@3	@2	@1	@0

Table Location

Note: *12~*0: Table location bits
@7~@0: Table pointer bits
P12~P8: Current program counter bits

value from the stack. After a chip reset, the SP will point to the top of the stack.

If the stack is full and a non-masked interrupt takes place, the interrupt request flag will be recorded but the acknowledgment will be inhibited. When the stack pointer is decremented (by RET or RETI), the interrupt is serviced. This feature prevents stack overflow, allowing the programmer to use the structure more easily. If the stack is full and a ″CALL″ is subsequently executed, stack overflow occurs and the first entry will be lost (only the most recent 16 return addresses are stored).

Data Memory – RAM

The data memory (RAM) is designed with 424×8 bits, and is divided into two functional groups, namely; special function registers (40×8 bits) and general purpose data memory (Bank 0:192×8 bits and Bank 1:192×8 bits) most of which are readable/writeable, although some are read only.

The special function registers are overlapped in any banks. Of the two types of functional groups, the special function registers consist of an Indirect addressing register 0 (00H), a Memory pointer register 0 (MP0;01H), an Indirect addressing register 1 (02H), a Memory pointer register 1 (MP1;03H), a Bank pointer (BP;04H), an Accumulator (ACC;05H), a Program counter lower-order byte register (PCL;06H), a Table pointer (TBLP;07H), a Table higher-order byte register (TBLH;08H), a Status register (STATUS;0AH), an Interrupt control register 0 (INTC0;0BH), a Timer/Event Counter 0 (TMR0H:0CH; TMR0L:0DH), a Timer/Event Counter 0 control register (TMR0C;0EH), a Timer/Event Counter 1 (TMR1H:0FH; TMR1L:10H), a Timer/Event Counter 1 control register (TMR1C; 11H), Interrupt control register 1 (INTC1;1EH), PWM data register (PWM0;1AH, PWM1;1BH, PWM2;1CH, PWM3;1DH), the I^{2}C Bus slave address register (HADR;20H), the I^{2}C Bus control register (HCR;21H), the I^{2}C Bus status register (HSR;22H), the I^{2}C Bus data register (HDR;23H),the A/D result lower-order byte register (ADRL;24H), the A/D result higher-order byte register (ADRH;25H), the A/D control register (ADCR;26H), the A/D clock setting register (ACSR;27H), I/O registers (PA;12H, PB;14H, PC;16H, PD;18H, PF; 28H) and I/O control registers (PAC;13H, PBC;15H, PCC;17H, PDC;19H, PFC;29H). The remaining space before the 40H is reserved for future expanded usage and reading these locations will get ″00H″. The space before 40H is overlapping in each bank. The general purpose data memory, addressed from 40H to FFH (Bank0; BP=0 or Bank1; BP=1), is used for data and control information under instruction commands.

All of the data memory areas can handle arithmetic, logic, increment, decrement and rotate operations directly. Except for some dedicated bits, each bit in the data memory can be set and reset by ″SET [m].i″ and ″CLR [m].i″. They are also indirectly accessible through memory pointer registers (MP0;01H/MP1;03H). The space before 40H is overlapping in each bank.

After first setting up BP to the value of ″01H″ or ″02H″ to access either bank 1 or bank 2 respectively, these banks must then be accessed indirectly using the Memory Pointer MP1. With BP set to a value of either ″01H″ or ″02H″, using MP1 to indirectly read or write to the data memory areas with addresses from 40H~FFH will result

Address	Register	
00H	Indirect Addressing Register 0	Special Purpose DATA MEMORY
01H	MP0	
02H	Indirect Addressing Register 1	
03H	MP1	
04H	BP	
05H	ACC	
06H	PCL	
07H	TBLP	
08H	TBLH	
09H	(Unused)	
0AH	STATUS	
0BH	INTC0	
0CH	TMR0H	
0DH	TMR0L	
0EH	TMR0C	
0FH	TMR1H	
10H	TMR1L	
11H	TMR1C	
12H	PA	
13H	PAC	
14H	PB	
15H	PBC	
16H	PC	
17H	PCC	
18H	PD	
19H	PDC	
1AH	PWM0	
1BH	PWM1	
1CH	PWM2	
1DH	PWM3	
1EH	INTC1	
1FH	(Unused)	
20H	HADR	
21H	HCR	
22H	HSR	
23H	HDR	
24H	ADRL	
25H	ADRH	
26H	ADCR	
27H	ACSR	
28H	PF	
29H	PFC	
2AH~3FH	(Unused)	
40H~FFH	General Purpose DATA MEMORY (192 Bytes×2 Bank: Bank0,Bank1)	

(Unused): Read as "00"

RAM Mapping

in operations to either bank 1 or bank 2. Directly addressing the Data Memory will always result in Bank 0 being accessed irrespective of the value of BP.

Indirect Addressing Register

Location 00H and 02H are indirect addressing registers that are not physically implemented. Any read/write operation of [00H] and [02H] accesses the RAM pointed to by MP0 (01H) and MP1(03H) respectively. Reading location 00H or 02H indirectly returns the result 00H. While, writing it indirectly leads to no operation. The function of data movement between two indirect addressing registers is not supported. The memory pointer registers, MP0 and MP1, are both 8-bit registers used to access the RAM by combining corresponding indirect addressing registers.

Accumulator – ACC

The accumulator is closely related to ALU operations. It is also mapped to location 05H of the RAM and capable of operating with immediate data. The data movement between two data memory locations must pass through the accumulator.

Arithmetic and Logic Unit – ALU

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

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

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

Status Register – STATUS

The status register (0AH) is 8 bits wide and contains, a carry flag (C), an auxiliary carry flag (AC), a zero flag (Z), an overflow flag (OV), a power down flag (PDF), and a Watchdog time-out flag (TO). It also records the status information and controls the operation sequence. Except for the TO and PDF flags, bits in the status register can be altered by instructions similar to other registers. Data written into the status register does not alter the TO or PDF flags. Operations related to the status register, however, may yield different results from those intended. The TO and PDF flags can only be changed by a Watchdog Timer overflow, chip power-up, or clearing the Watchdog Timer and executing the ″HALT″ instruction.

The Z, OV, AC, and C flags reflect the status of the latest operations. On entering the interrupt sequence or executing the subroutine call, the status register will not be automatically pushed onto the stack. If the contents of the status is important, and if the subroutine is likely to corrupt the status register, the programmer should take precautions and save it properly.

Interrupts

The device provides an external interrupt, two internal timer/event counter interrupt, the A/D converter interrupt and the I^2C Bus interrupts. The interrupt control register 0 (INTC0;0BH) and interrupt control register 1 (INTC1;1EH) contains the interrupt control bits to set the enable/disable and the interrupt request flags.

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

Labels	Bits	Function
C	0	C is set if an operation results in a carry during an addition operation or if a borrow does not take place during a subtraction operation; otherwise C is cleared. C is also affected by a rotate through carry instruction.
AC	1	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	2	Z is set if the result of an arithmetic or logic operation is zero; otherwise Z is cleared.
OV	3	OV is set if an operation results in a carry into the highest-order bit but not a carry out of the highest-order bit, or vice versa; otherwise OV is cleared.
PDF	4	PDF is cleared by system power-up or executing the ″CLR WDT″ instruction. PDF is set by executing the ″HALT″ instruction.
TO	5	TO is cleared by system power-up or executing the ″CLR WDT″ or ″HALT″ instruction. TO is set by a WDT time-out.
—	6, 7	Unused bit, read as ″0″

Status Register

EMI bit and the corresponding bit of INTC0 and INTC1 may be set to allow interrupt nesting. If the stack is full, the interrupt request will not be acknowledged, even if the related interrupt is enabled, until the SP is decremented. If immediate service is desired, the stack must be prevented from becoming full.

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

External interrupts are triggered by a high to low transition of $\overline{INT}$ and the related interrupt request flag (EIF; bit 4 of INTC0) will be set. When the interrupt is enabled, the stack is not full and the external interrupt is active, a subroutine call to location 04H will occur. The interrupt request flag (EIF) and EMI bits will be cleared to disable other interrupts.

The internal Timer/Event Counter 0 interrupt is initialized by setting the Timer/Event Counter 0 interrupt request flag (T0F; bit 5 of INTC0), which is normally caused by a timer overflow. After the interrupt is enabled, and the stack is not full, and the T0F bit is set, a subroutine call to location 08H occurs. The related interrupt request flag (T0F) is reset, and the EMI bit is cleared to disable further maskable interrupts. The Timer/Event Counter 1 is operated in the same manner but its related interrupt request flag is T1F (bit 6 of INTC0) and its subroutine call location is 0CH.

The A/D converter interrupt is initialized by setting the A/D converter request flag (ADF; bit 4 of INTC1), caused by an end of A/D conversion. When the interrupt is enabled, the stack is not full and the ADF is set, a subroutine call to location 10H will occur. The related interrupt request flag (ADF) will be reset and the EMI bit cleared to disable further interrupts.

The I^2C Bus interrupt is initialized by setting the I^2C Bus interrupt request flag (HIF; bit 5 of INTC1), caused by a slave address match (HAAS=″1″) or one byte of data transfer is completed. When the interrupt is enabled, the stack is not full and the HIF bit is set, a subroutine call to location 14H will occur. The related interrupt request flag (HIF) will be reset and the EMI bit cleared to disable further interrupts.

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

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

Register	Bit No.	Label	Function
INTC0 (0BH)	0	EMI	Controls the master (global) interrupt (1= enabled; 0= disabled)
	1	EEI	Controls the external interrupt (1= enabled; 0= disabled)
	2	ET0I	Controls the Timer/Event Counter 0 interrupt (1= enabled; 0= disabled)
	3	ET1I	Controls the Timer/Event Counter 1 interrupt (1= enabled; 0= disabled)
	4	EIF	External interrupt request flag (1= active; 0= inactive)
	5	T0F	Internal Timer/Event Counter 0 request flag (1= active; 0= inactive)
	6	T1F	Internal Timer/Event Counter 1 request flag (1= active; 0= inactive)
	7	—	Unused bit, read as ″0″
INTC1 (1EH)	0	EADI	Control the A/D converter interrupt (1= enabled; 0=disabled)
	1	EHI	Control the I^2C Bus interrupt (1= enabled; 0= disabled)
	2, 3	—	Unused bit, read as ″0″
	4	ADF	A/D converter request flag (1= active; 0= inactive)
	5	HIF	I^2C Bus interrupt request flag (1= active; 0= inactive)
	6, 7	—	Unused bit, read as ″0″

INTC Register

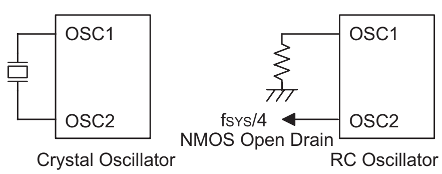
Interrupt Source	Priority	Vector
External Interrupt	1	04H
Timer/Event Counter 0 Overflow	2	08H
Timer/Event Counter 1 Overflow	3	0CH
A/D Converter Interrupt	4	10H
I^2C Bus Interrupt	5	14H

The Timer/Event Counter 0/1 interrupt request flag (T0F, T1F), external interrupt request flag (EIF), A/D converter request flag (ADF), the I^2C Bus interrupt request flag (HIF), enable timer/event counter bit (ET0I, ET1I), enable external interrupt bit (EEI), enable A/D converter interrupt bit (EADI), enable I^2C Bus interrupt bit (EHI) and enable master interrupt bit (EMI) constitute an interrupt control register 0 (INTC0) and an interrupt control register 1 (INTC1) which are located at 0BH and 1EH in the data memory. EMI, EEI, ET0I, ET1I, EADI, EHI are used to control the enabling/disabling of interrupts. These bits prevent the requested interrupt from being serviced. Once the interrupt request flags (T0F, T1F, EIF, ADF, HIF) are set, they will remain in the INTC0 and INTC1 register until the interrupts are serviced or cleared by a software instruction.

It is recommended that a program does not use the ″CALL subroutine″ within the interrupt subroutine. Interrupts often occur in an unpredictable manner or need to be serviced immediately in some applications. If only one stack is left and enabling the interrupt is not well controlled, the original control sequence will be damaged once the ″CALL″ operates in the interrupt subroutine.

Oscillator Configuration

There are two oscillator circuits in the microcontroller.



System Oscillator

Both are designed for system clocks, namely the RC oscillator and the Crystal oscillator, which are determined by the option. No matter what oscillator type is selected, the signal provides the system clock. The HALT mode stops the system oscillator and ignores an external signal to conserve power.

If an RC oscillator is used, an external resistor between OSC1 and VSS is required and the resistance must range from 30kΩ to 750kΩ. The system clock, divided by 4, is available on OSC2, which can be used to synchronize external logic. The RC oscillator provides the most cost effective solution. However, the frequency of oscillation may vary with VDD, temperatures and the chip itself due to process variations. It is, therefore, not suitable for timing sensitive operations where an accurate oscillator frequency is desired.

If the Crystal oscillator is used, a crystal across OSC1 and OSC2 is needed to provide the feedback and phase shift required for the oscillator, and no other external components are required. Instead of a crystal, a resonator can also be connected between OSC1 and OSC2 to get a frequency reference, but two external capacitors in OSC1 and OSC2 are required (If the oscillating frequency is less than 1MHz).

The WDT oscillator is a free running on-chip RC oscillator, and no external components are required. Even if the system enters the power down mode, the system clock is stopped, but the WDT oscillator still works with a period of approximately 65μs at 5V. The WDT oscillator can be disabled by option to conserve power.

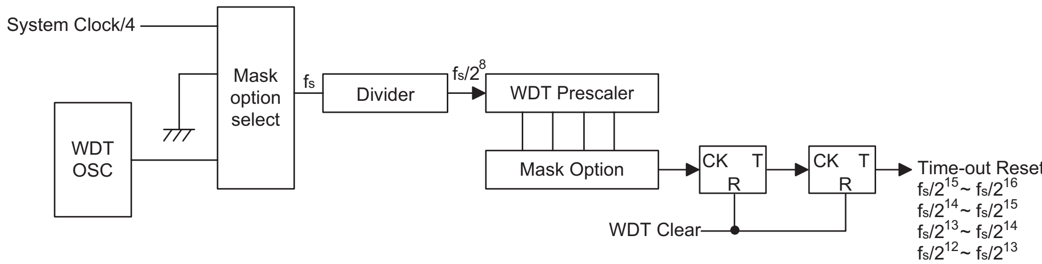
Watchdog Timer – WDT

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

Once an internal WDT oscillator (RC oscillator with period 65μs at 5V normally) is selected, it is divided by 2^{12}~2^{15} (by option to get the WDT time-out period). The WDT time-out minimum period is 300ms~600ms. This time-out period may vary with temperature, VDD and process variations. By selection from the WDT option, longer time-out periods can be realized. If the WDT time-out is selected 2^{15}, the maximum time-out period is divided by 2^{15}~2^{16}about 2.1s~4.3s.

If the WDT oscillator is disabled, the WDT clock may still come from the instruction clock and operate in the same manner except that in the HALT state the WDT may stop counting and lose its protecting purpose. In this situation the logic can only be restarted by external logic. If the device operates in a noisy environment, using the on-chip RC oscillator (WDT OSC) is strongly recommended, since the HALT will stop the system clock.

The WDT overflow under normal operation will initialize ″chip reset″ and set the status bit TO. Whereas in the HALT mode, the overflow will initialize a ″warm reset″ only the PC and SP are reset to zero. To clear the contents of WDT, three methods are adopted; external reset (a low level to $\overline{RES}$), software instructions, or a HALT instruction. The software instructions include CLR WDT and the other set CLR WDT1 and CLR WDT2. Of these two types of instruction, only one can be active depend-



Watchdog Timer

ing on the option – ″CLR WDT times selection option″. If the ″CLR WDT″ is selected (i.e. CLRWDT times equal one), any execution of the CLR WDT instruction will clear the WDT. In case ″CLR WDT1″ and ″CLR WDT2″ are chosen (i.e. CLRWDT times equal two), these two instructions must be executed to clear the WDT; otherwise, the WDT may reset the chip because of time-out.

If the WDT time-out period is selected $f_s/2^{12}$ (option), the WDT time-out period ranges from $f_s/2^{12}$~$f_s/2^{13}$, since the ″CLR WDT″ or ″CLR WDT1″ and ″CLR WDT2″ instructions only clear the last two stages of the WDT.

Power Down Operation – HALT

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

- The system oscillator turned off but the WDT oscillator keeps running (if the WDT oscillator or the real time clock is selected).
- The contents of the on-chip RAM and registers remain unchanged
- The WDT will be cleared and start recounting (if the WDT clock source is from the WDT oscillator or the real time clock)
- All of the I/O ports maintain their original status
- The PDF flag is set and the TO flag is cleared

The system quits the HALT mode by an external reset, an interrupt, an external falling edge signal on port A or a WDT overflow. An external reset causes a device initialization and the WDT overflow performs a ″warm reset″. After examining the TO and PDF flags, the reason for chip reset can be determined. The PDF flag is cleared by system power-up or by executing the ″CLR WDT″ instruction and is set when executing the ″HALT″ instruction. On the other hand, the TO flag is set if the WDT time-out occurs, and causes a wake-up that only resets the PC program counter and SP; and leaves the others in their original status.

The port A wake-up and interrupt methods can be considered as a continuation of normal execution. Each bit in port A can be independently selected to wake up the device by the option. Awakening from an I/O port stimulus, the program will resume execution of the next instruction. If it is awakening from an interrupt, two sequences may occur. If the related interrupt is disabled or the interrupt is enabled but the stack is full, the program will resume execution at the next instruction. But if the interrupt is enabled and the stack is not full, the regular interrupt response takes place. When an interrupt request flag is set to ″1″ before entering the HALT mode, the wake-up function of the related interrupt will be disabled. If wake-up event occurs, it takes 1024 f_{SYS} (system clock period) to resume normal operation. In other words, a dummy period is inserted after wake-up. If the wake-up results from an interrupt acknowledgment, the actual interrupt subroutine execution is delayed by more than one cycle. However, if the wake-up results in the next instruction execution, this will be executed performed immediately after the dummy period is finished.

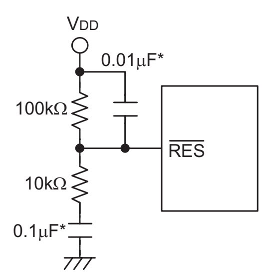
To minimize power consumption, all the I/O pins should be carefully managed before entering the HALT status.

Reset

There are three ways in which a reset may occur:

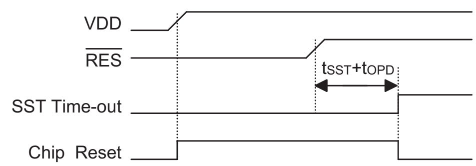
- $\overline{RES}$ reset during normal operation
- $\overline{RES}$ reset during HALT
- WDT time-out reset during normal operation

The WDT time-out during HALT differs from other chip reset conditions, for it can perform a ″warm reset″ that resets only the PC and SP, leaves the other circuits at their original state. Some registers remain unaffected during any other reset conditions. Most registers are reset to the ″initial condition″ when the reset conditions are met. Examining the PDF and TO flags, the program can distinguish between different ″chip resets″.

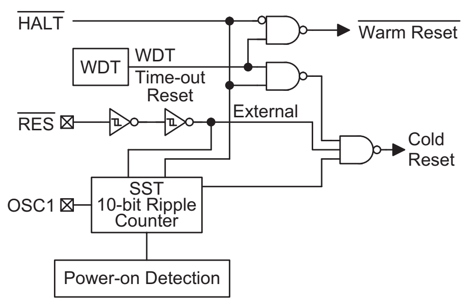


Reset Circuit

Note: ″*″ Make the length of the wiring, which is connected to the $\overline{RES}$ pin as short as possible, to avoid noise interference.



Reset Timing Chart



Reset Configuration

TO	PDF	RESET Conditions
0	0	RES reset during power-up
u	u	RES reset during normal operation
0	1	RES wake-up HALT
1	u	WDT time-out during normal operation
1	1	WDT wake-up HALT

Note: ″u″ stands for ″unchanged″

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

Awaking from the HALT state or system power up an SST delay is added. An extra SST delay is added during power up period, and any wake-up from HALT may enable only the SST delay. The functional unit chip reset status are shown below.

PC	000H
Interrupt	Disable
Prescaler, Divider	Cleared
WDT	Clear. After master reset, WDT begins counting
Timer/event Counter	Off
Input/output Ports	Input mode
SP	Points to the top of the stack

Timer/Event Counter

Two Timer/Event Counters (TMR0,TMR1) are implemented in the microcontroller. The timer/event counter 0 contains an 16-bit programmable count-up counter and the clock may come from an external source or an internal clock source. An internal clock source comes from f_{SYS}. The timer/event counter 1 contains an 16-bit programmable count-up counter and the clock may come from an external source or an internal clock source. An internal clock source comes from $f_{SYS}/4$. The external clock input allows the user to count external events, measure time intervals or pulse widths, or to generate an accurate time base.

There are six registers related to the Timer/Event Counter 0; TMR0H (0CH), TMR0L (0DH), TMR0C (0EH) and the Timer/Event Counter 1; TMR1H (0FH), TMR1L (10H), TMR1C (11H). Writing TMR0L (TMR1L) will only put the written data to an internal lower-order byte buffer (8-bit) and writing TMR0H (TMR1H) will transfer the specified data and the contents of the lower-order byte buffer to TMR0H (TMR1H) and TMR0L (TMR1L) registers, respectively. The Timer/Event Counter 1/0 preload register is changed by each writing TMR0H (TMR1H) operations. Reading TMR0H (TMR1H) will latch the contents of TMR0H (TMR1H) and TMR0L (TMR1L) counters to the destination and the lower-order byte buffer, respectively. Reading the TMR0L (TMR1L) will read the contents of the lower-order byte buffer. The TMR0C (TMR1C) is the Timer/Event Counter 0 (1) control register, which defines the operating mode, counting enable or disable and an active edge.

The T0M0, T0M1 (TMR0C) and T1M0, T1M1 (TMR1C) bits define the operation mode. The event count mode is used to count external events, which means that the clock source is from an external (TMR0, TMR1) pin. The timer mode functions as a normal timer with the clock source coming from the internal selected clock source. Finally, the pulse width measurement mode can be used to count the high or low level duration of the external signal (TMR0, TMR1), and the counting is based on the internal selected clock source.

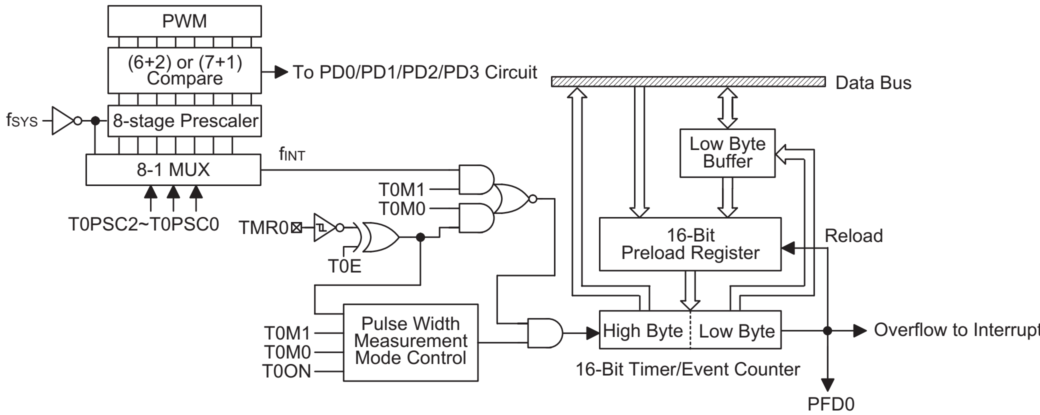
In the event count or timer mode, the timer/event counter starts counting at the current contents in the timer/event counter and ends at FFFFH. Once an overflow occurs, the counter is reloaded from the timer/event counter preload register, and generates an interrupt request flag (T0F; bit 5 of INTC0, T1F; bit 6 of INTC0).

In the pulse width measurement mode with the values of the T0ON/T1ON and T0E/T1E bits equal to 1, after the TMR0 (TMR1) has received a transient from low to high (or high to low if the T0E/T1E bit is ″0″), it will start counting until the TMR0 (TMR1) returns to the original level and resets the T0ON/T1ON. The measured result remains in the timer/event counter even if the activated transient occurs again. In other words, only 1-cycle measurement can be made until the T0ON/T1ON is set. The cycle measurement will re-function as long as it receives further transient pulse. In this operation mode, the timer/event counter begins counting not according to the logic level but to the transient edges. In the case of counter overflows, the counter is reloaded from the timer/event counter register and issues an interrupt request, as in the other two modes, i.e., event and timer modes.

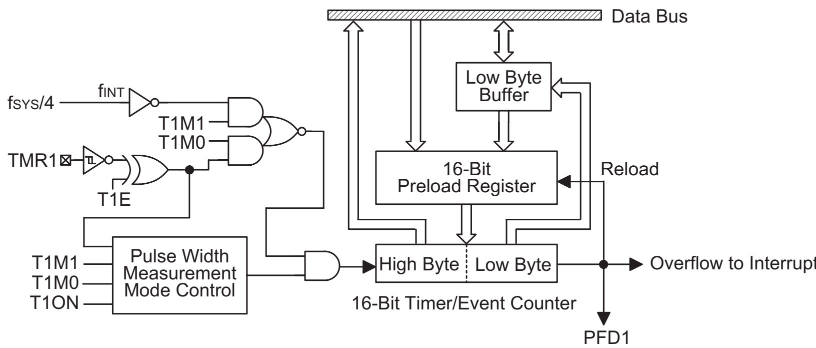
The registers states are summarized in the following table.

Register	Reset(Power On)	WDT Time-out (Normal Operation)	$\overline{RES}$ Reset (Normal Operation)	$\overline{RES}$ Reset (HALT)	WDT Time-out (HALT)*
TMR0H	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMR0L	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMR0C	00-0 1000	00-0 1000	00-0 1000	00-0 1000	uu-u uuuu
TMR1H	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMR1L	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMR1C	00-0 1---	00-0 1---	00-0 1---	00-0 1---	uu-u u---
Program Counter	000H	000H	000H	000H	000H
MP0	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
MP1	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
BP	---- ---0	---- ---0	---- ---0	---- ---0	---- ---u
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLP	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLH	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
STATUS	--00 xxxx	--1u uuuu	--uu uuuu	--01 uuuu	--11 uuuu
INTC0	-000 0000	-000 0000	-000 0000	-000 0000	-uuu uuuu
INTC1	--00 --00	--00 --00	--00 --00	--00 --00	--uu --uu
PA	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PB	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PBC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PCC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PD	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PDC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PF	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PFC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PWM0	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
PWM1	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
PWM2	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
PWM3	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
HADR	xxxx xxx-	xxxx xxx-	xxxx xxx-	xxxx xxx-	uuuu uuu-
HCR	0--0 0---	0--0 0---	0--0 0---	0--0 0---	u--u u---
HSR	100- -0-1	100- -0-1	100- -0-1	100- -0-1	uuuu uuuu
HDR	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
ADRL	xx-- ----	xx-- ----	xx-- ----	xx-- ----	uu-- ----
ADRH	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu	uuuu uuuu
ADCR	0100 0000	0100 0000	0100 0000	0100 0000	uuuu uuuu
ACSR	1--- --00	1--- --00	1--- --00	1--- --00	u--- --uu

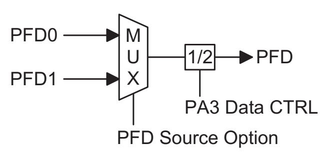
Note: ″*″ stands for warm reset
″u″ stands for unchanged
″x″ stands for unknown



Timer/Event Counter 0



Timer/Event Counter 1



PFD Source Option

To enable the counting operation, the Timer ON bit (T0ON: bit 4 of TMR0C; T1ON: bit 4 of TMR1C) should be set to 1. In the pulse width measurement mode, the T0ON/T1ON is automatically cleared after the measurement cycle is completed. But in the other two modes, the T0ON/T1ON can only be reset by instructions. The overflow of the Timer/Event Counter 0/1 is one of the wake-up sources and can also be applied to a PFD (Programmable Frequency Divider) output at PA3 by options. Only one PFD (PFD0 or PFD1) can be applied to PA3 by options. If PA3 is set as PFD output, there are two types of selections; One is PFD0 as the PFD output, the other is PFD1 as the PFD output. PFD0, PFD1 are the timer overflow signals of the Timer/Event Counter 0, Timer/Event Counter 1 respectively. No matter what the operation mode is, writing a 0 to ET0I or ET1I disables the related interrupt service. When the PFD function is selected, executing ″SET [PA].3″ instruction to enable PFD output and executing ″CLR [PA].3″ instruction to disable PFD output.

In the case of timer/event counter OFF condition, writing data to the timer/event counter preload register also reloads that data to the timer/event counter. But if the timer/event counter is turn on, data written to the timer/event counter is kept only in the timer/event counter preload register. The timer/event counter still continues its operation until an overflow occurs.

When the timer/event counter (reading TMR0/TMR1) is read, the clock is blocked to avoid errors, as this may results in a counting error. Blocking of the clock should be taken into account by the programmer. It is strongly recommended to load a desired value into the TMR0/TMR1 register first, before turning on the related timer/event counter, for proper operation since the initial value of TMR0/TMR1 is unknown. Due to the timer/event scheme, the programmer should pay special attention on the instruction to enable then disable the timer for the first time, whenever there is a need to use the timer/event function, to avoid unpredictable result. After this procedure, the timer/event function can be operated normally.

The bit0~bit2 of the TMR0C can be used to define the pre-scaling stages of the internal clock sources of timer/event counter. The definitions are as shown. The overflow signal of timer/event counter can be used to generate the PFD signal. The timer prescaler is also used as the PWM counter.

Label (TMR0C)	Bits	Function
T0PSC0 T0PSC1 T0PSC2	0 1 2	Defines the prescaler stages, T0PSC2, T0PSC1, T0PSC0= 000: $f_{INT}=f_{SYS}$ 001: $f_{INT}=f_{SYS}/2$ 010: $f_{INT}=f_{SYS}/4$ 011: $f_{INT}=f_{SYS}/8$ 100: $f_{INT}=f_{SYS}/16$ 101: $f_{INT}=f_{SYS}/32$ 110: $f_{INT}=f_{SYS}/64$ 111: $f_{INT}=f_{SYS}/128$
T0E	3	Defines the TMR active edge of timer/ event counter (0=active on low to high; 1=active on high to low)
T0ON	4	Enable/disable timer counting (0=disabled; 1=enabled)
—	5	Unused bit, read as ″0″
T0M0 T0M1	6 7	Defines the operating mode, T0M1, T0M0: 01=Event count mode (external clock) 10=Timer mode (internal clock) 11=Pulse width measurement mode 00=Unused

TMR0C Register

Label (TMR1C)	Bits	Function
—	0~2	Unused bit, read as ″0″
T1E	3	Defines the TMR active edge of timer/ event counter (0=active on low to high; 1=active on high to low)
T1ON	4	Enable/disable timer counting (0=disabled; 1=enabled)
—	5	Unused bit, read as ″0″
T1M0 T1M1	6 7	Defines the operating mode, T1M1, T1M0: 01=Event count mode (external clock) 10=Timer mode (internal clock) 11=Pulse width measurement mode 00=Unused

TMR1C Register

Input/Output Ports

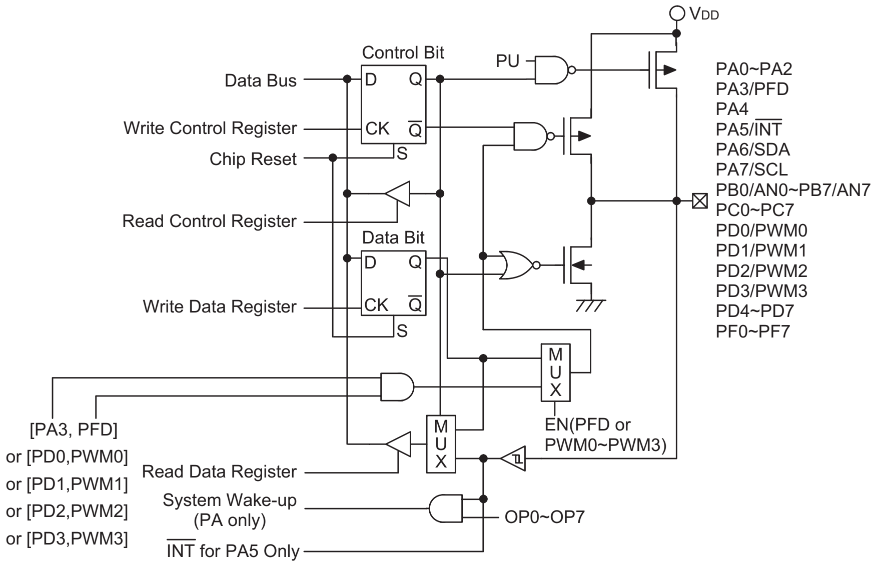
There are 40 bidirectional input/output lines in the microcontroller, labeled as PA, PB, PC, PD and PF, which are mapped to the data memory of [12H], [14H], [16H], [18H] and [28H] respectively. All of these I/O ports can be used for input and output operations. For input operation, these ports are non-latching, that is, the inputs must be ready at the T2 rising edge of instruction ″MOV A,[m]″ (m=12H, 14H, 16H, [18H] or 28H). For output operation, all the data is latched and remains unchanged until the output latch is rewritten.

Each I/O line has its own control register (PAC, PBC, PCC, PDC, PFC) to control the input/output configuration. With this control register, CMOS output or Schmitt trigger input with or without pull-high resistor structures can be reconfigured dynamically under software control.

To function as an input, the corresponding latch of the control register must write ″1″. The input source also depends on the control register. If the control register bit is ″1″, the input will read the pad state. If the control register bit is ″0″, the contents of the latches will move to the internal bus. The latter is possible in the ″read-modify-write″ instruction.

For output function, CMOS is the only configuration. These control registers are mapped to locations 13H, 15H, 17H, 19H and 29H.

After a chip reset, these input/output lines remain at high levels or floating state (depends on pull-high options). Each bit of these input/output latches can be set or cleared by ″SET [m].i″ and ″CLR [m].i″ (m=12H, 14H, 16H 18H or 28H) instructions.



Input/Output Ports

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

Each line of port A has the capability of waking-up the device. Each I/O port has a pull-high option. Once the pull-high option is selected, the I/O port has a pull-high resistor, otherwise, there′s none. Take note that a non-pull-high I/O port operating in input mode will cause a floating state.

The PA3 and PA5 are pin-shared with the PFD and $\overline{INT}$ pins respectively. If the PFD option is selected, the output signal in output mode of PA3 will be the PFD signal generated by timer/event counter overflow signal. The input mode always remain in its original functions. Once the PFD option is selected, the PFD output signal is controlled by PA3 data register only. Writing ″1″ to PA3 data register will enable the PFD output function and writing 0 will force the PA3 to remain at ″0″. The I/O functions of PA3 are shown below.

I/O Mode	I/P (Normal)	O/P (Normal)	I/P (PFD)	O/P (PFD)
PA3	Logical Input	Logical Output	Logical Input	PFD (Timer on)

Note: The PFD frequency is the timer/event counter overflow frequency divided by 2.

The PB can also be used as A/D converter inputs. The A/D function will be described later. There is a PWM function shared with PD0/PD1/PD2/PD3. If the PWM function is enabled, the PWM0/PWM1/PWM2/PWM3 signal will appear on PD0/PD1/PD2/PD3 (if PD0/PD1/PD2/PD3 is operating in output mode). The I/O functions of PD0/PD1/PD2/PD3 are as shown.

I/O Mode	I/P (Normal)	O/P (Normal)	I/P (PWM)	O/P (PWM)
PD0 PD1 PD2 PD3	Logical Input	Logical Output	Logical Input	PWM0 PWM1 PWM2 PWM3

It is recommended that unused or not bonded out I/O lines should be set as output pins by software instruction to avoid consuming power under input floating state.

PWM

The microcontroller provides 4 channels (6+2)/(7+1) (depends on options) bits PWM output shared with PD0/PD1/PD2/PD3. The PWM channels have their data registers denoted as PWM0 (1AH), PWM1 (1BH), PWM2 (1CH) and PWM3 (1DH). The frequency source of the PWM counter comes from f_{SYS}. The PWM registers are four 8-bit registers. The waveforms of PWM outputs are as shown. Once the PD0/PD1/PD2/PD3 are selected as the PWM outputs and the output function of PD0/PD1/PD2/PD3 are enabled (PDC.0/PDC.1/PDC.2/PDC.3 =″0″), writing ″1″ to PD0/PD1/PD2/PD3 data register will enable the PWM output function and writing ″0″ will force the PD0/PD1/PD2/PD3 to stay at ″0″.

A (6+2) bits mode PWM cycle is divided into four modulation cycles (modulation cycle 0~modulation cycle 3). Each modulation cycle has 64 PWM input clock period. In a (6+2) bit PWM function, the contents of the PWM register is divided into two groups. Group 1 of the PWM register is denoted by DC which is the value of PWM.7~PWM.2.

The group 2 is denoted by AC which is the value of PWM.1~PWM.0.

In a (6+2) bits mode PWM cycle, the duty cycle of each modulation cycle is shown in the table.

Parameter	AC (0~3)	Duty Cycle
Modulation cycle i (i=0~3)	i<AC	$\frac{DC+1}{64}$
	i≥AC	$\frac{DC}{64}$

A (7+1) bits mode PWM cycle is divided into two modulation cycles (modulation cycle0~modulation cycle 1). Each modulation cycle has 128 PWM input clock period.

In a (7+1) bits PWM function, the contents of the PWM register is divided into two groups. Group 1 of the PWM register is denoted by DC which is the value of PWM.7~PWM.1.

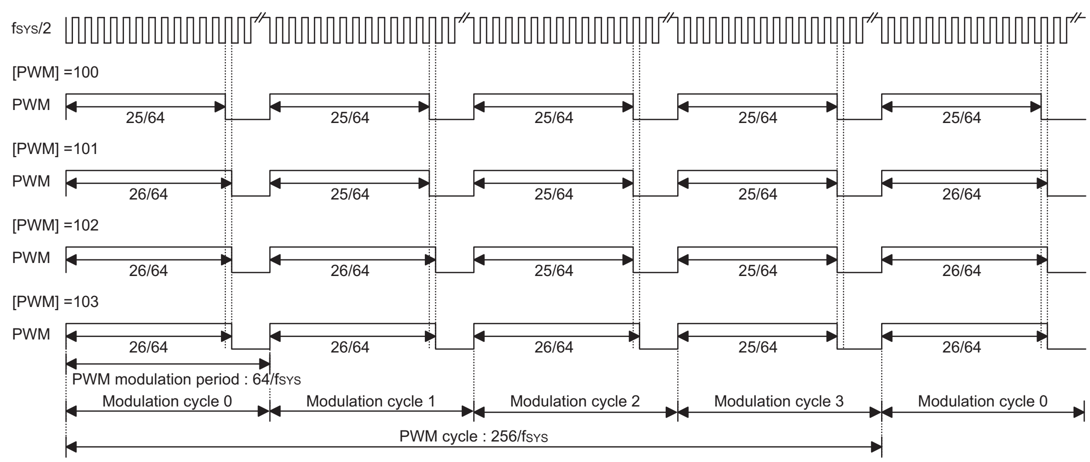
The group 2 is denoted by AC which is the value of PWM.0.

In a (7+1) bits mode PWM cycle, the duty cycle of each modulation cycle is shown in the table.

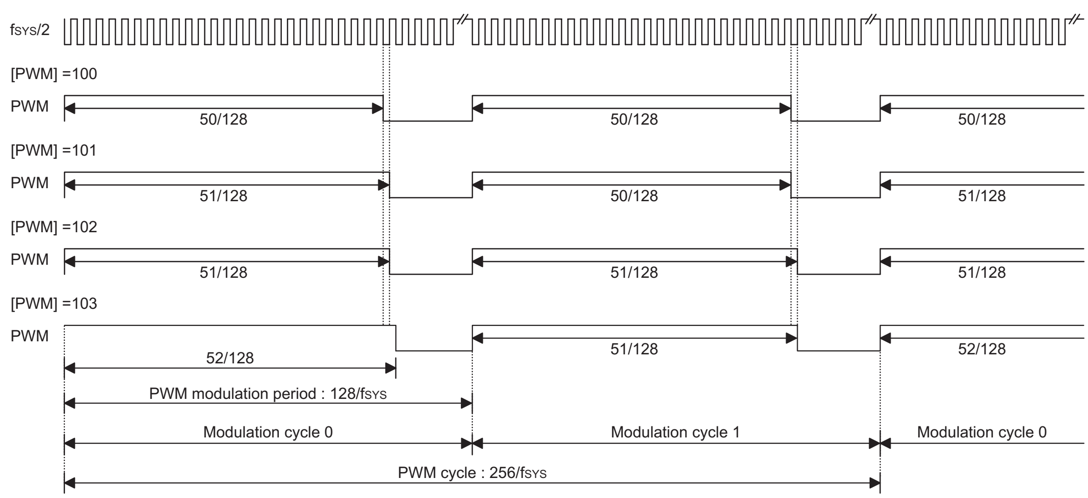
Parameter	AC (0~1)	Duty Cycle
Modulation cycle i (i=0~1)	i<AC	$\frac{DC+1}{128}$
	i≥AC	$\frac{DC}{128}$

The modulation frequency, cycle frequency and cycle duty of the PWM output signal are summarized in the following table.

PWM Modulation Frequency	PWM Cycle Frequency	PWM Cycle Duty
f_{SYS}/64 for (6+2) bits mode f_{SYS}/128 for (7+1) bits mode	f_{SYS}/256	[PWM]/256



(6+2) PWM Mode



(7+1) PWM Mode

A/D Converter

The 8 channels and 10-bit resolution A/D (9-bit accuracy) converter are implemented in this microcontroller. The reference voltage is VDD. The A/D converter contains 4 special registers which are; ADRL (24H), ADRH (25H), ADCR (26H) and ACSR (27H). The ADRH and ADRL are A/D result register higher-order byte and lower-order byte and are read-only. After the A/D conversion is completed, the ADRH and ADRL should be read to get the conversion result data. The ADCR is an A/D converter control register, which defines the A/D channel number, analog channel select, start A/D conversion control bit and the end of A/D conversion flag. If the users want to start an A/D conversion. Define PB configuration, select the converted analog channel, and give START bit a raising edge and falling edge (0→1→0). At the end of A/D conversion, the EOCB bit is cleared and an A/D converter interrupt occurs (if the A/D converter interrupt is enabled). The ACSR is A/D clock setting register, which is used to select the A/D clock source.

The A/D converter control register is used to control the A/D converter. The bit2~bit0 of the ADCR are used to select an analog input channel. There are a total of eight channels to select. The bit5~bit3 of the ADCR are used to set PB configurations. PB can be an analog input or as digital I/O line decided by these 3 bits. Once a PB line is selected as an analog input, the I/O functions and pull-high resistor of this I/O line are disabled and the A/D converter circuit is power on. The EOCB bit (bit6 of the ADCR) is end of A/D conversion flag. Check this bit to know when A/D conversion is completed. The START bit of the ADCR is used to begin the conversion of the A/D converter. Giving START bit a rising edge and falling edge means that the A/D conversion has started. In order to ensure the A/D conversion is completed, the START should remain at ″0″ until the EOCB is cleared to ″0″ (end of A/D conversion).

The bit 7 of the ACSR is used for testing purposes only. It cannot be used by the users. The bit1 and bit0 of the ACSR are used to select A/D clock sources.

Label (ACSR)	Bits	Function
ADCS0 ADCS1	0 1	Selects the A/D converter clock source 00= system clock/2 01= system clock/8 10= system clock/32 11= undefined #See other note3*
—	2~6	Unused bit, read as ″0″
TEST	7	For test mode used only

ACSR Register

Label (ADCR)	Bits	Function
ACS0 ACS1 ACS2	0 1 2	Defines the analog channel select
PCR0 PCR1 PCR2	3 4 5	Defines the port B configuration select. If PCR0, PCR1 and PCR2 are all zero, the ADC circuit is power off to reduce power consumption
EOCB	6	Provides response at the end of the A/D conversion. (0= end of A/D conversion)
START	7	Starts the A/D conversion. (0→1→0= start; 0→1= reset A/D converter)

ADCR Register

ACS2	ACS1	ACS0	Analog Channel
0	0	0	A0
0	0	1	A1
0	1	0	A2
0	1	1	A3
1	0	0	A4
1	0	1	A5
1	1	0	A6
1	1	1	A7

Analog Input Channel Selection

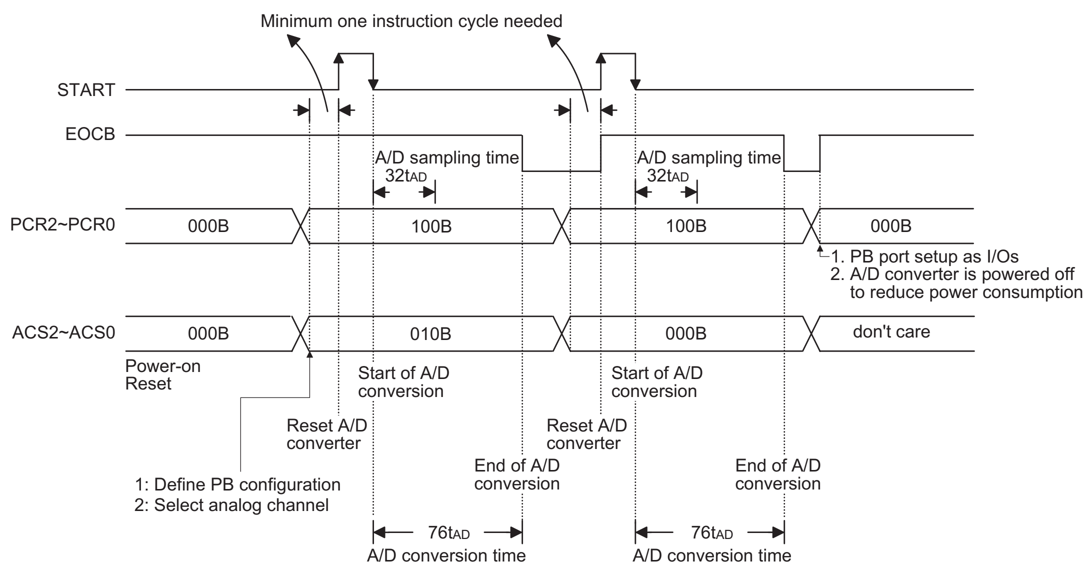
When the A/D conversion is completed, the A/D interrupt request flag is set. The EOCB bit is set to ″1″ when the START bit is set from ″0″ to ″1″.

Register	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADRL	D1	D0	—	—	—	—	—	—
ADRH	D9	D8	D7	D6	D5	D4	D3	D2

Note: D0~D9 is A/D conversion result data bit LSB~MSB.

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	A0
0	1	0	PB7	PB6	PB5	PB4	PB3	PB2	A1	A0
0	1	1	PB7	PB6	PB5	PB4	PB3	A2	A1	A0
1	0	0	PB7	PB6	PB5	PB4	A3	A2	A1	A0
1	0	1	PB7	PB6	PB5	A4	A3	A2	A1	A0
1	1	0	PB7	PB6	A5	A4	A3	A2	A1	A0
1	1	1	A7	A6	A5	A4	A3	A2	A1	A0

Port B Configuration



Note: A/D clock must be $f_{SYS}/2$, $f_{SYS}/8$ or $f_{SYS}/32$

A/D Conversion Timing

The following two programming examples illustrate how to setup and implement an A/D conversion. In the first example, the method of polling the EOCB bit in the ADCR register is used to detect when the conversion cycle is complete, whereas in the second example, the A/D interrupt is used to determine when the conversion is complete.

Example: using EOCB Polling Method to detect end of conversion

```
    clr INTC1.0              ; disable A/D interrupt in interrupt control register
    mov a,00100000B
    mov ADCR,a               ; setup ADCR register to configure Port PB0~PB3 as A/D inputs and select
                             ; AN0 to be connected to the A/D converter
    mov a,00000001B
    mov ACSR,a               ; setup the ACSR register to select fSYS/8 as the A/D clock

Start_conversion:
    clr ADCR.7
    set ADCR.7               ; reset A/D
    clr ADCR.7               ; start A/D

Polling_EOC:
    sz ADCR.6                ; poll the ADCR register EOCB bit to detect end of A/D conversion
    jmp polling_EOC          ; continue polling
    mov a,ADRH               ; read conversion result from the high byte ADRH register
    mov adrh_buffer,a        ; save result to user defined register
    mov a,ADRL               ; read conversion result from the low byte ADRL register
    mov adrl_buffer,a        ; save result to user defined register
            :
            :
    jmp start_conversion     ; start next A/D conversion
```

Example: using Interrupt method to detect end of conversion

```
    set INTC0.0              ; interrupt global enable
    set INTC1.0              ; enable A/D interrupt in interrupt control register
    mov a,00100000B
    mov ADCR,a               ; setup ADCR register to configure Port PB0~PB3 as A/D inputs and select
                             ; AN0 to be connected to the A/D converter
    mov a,00000001B
    mov ACSR,a               ; setup the ACSR register to select fSYS/8 as the A/D clock

start_conversion:
    clr ADCR.7
    set ADCR.7               ; reset A/D
    clr ADCR.7               ; start A/D
            :
            :

; interrupt service routine
EOC_service routine:
    mov a_buffer,a           ; save ACC to user defined register
    mov a,ADRH               ; read conversion result from the high byte ADRH register
    mov adrh_buffer,a        ; save result to user defined register
    mov a,ADRL               ; read conversion result from the low byte ADRL register
    mov adrl_buffer,a        ; save result to user defined register

    clr ADCR.7
    set ADCR.7               ; reset A/D
    clr ADCR.7               ; start A/D

    mov a,a_buffer           ; restore ACC from temporary storage
    reti
```

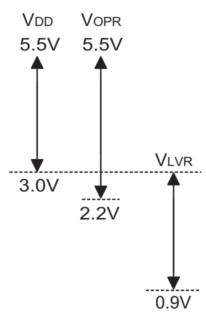
Low Voltage Reset – LVR

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

The LVR includes the following specifications:

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

The relationship between V_{DD} and V_{LVR} is shown below.



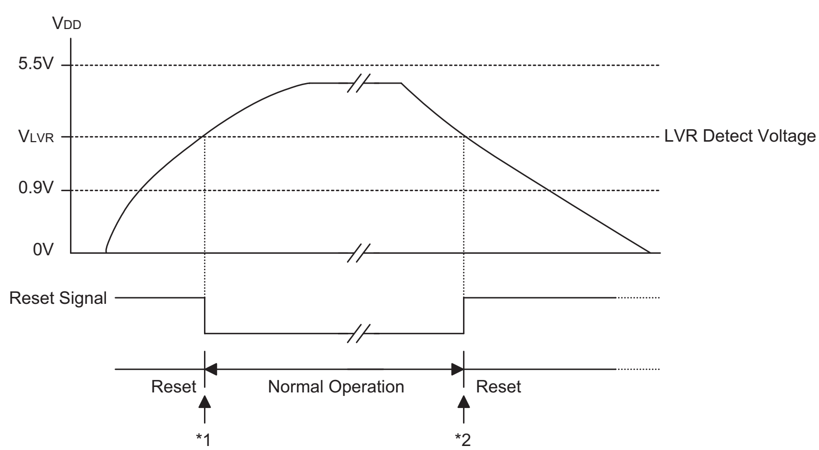
Note: V_{OPR} is the voltage range for proper chip operation at 4MHz system clock.

I^2C Bus Serial Interface

I^2C Bus is implemented in the device. The I^2C Bus is a bidirectional two-wire lines. The data line and clock line are implement in SDA pin and SCL pin. The SDA and SCL are NMOS open drain output pin. They must connect a pull-high resistor respectively.

Using the I^2C Bus, the device has two ways to transfer data. One is in slave transmit mode, the other is in slave receive mode. There are four registers related to I^2C Bus; HADR([20H]), HCR([21H]), HSR([22H]), HDR([23H]). The HADR register is the slave address setting of the device, if the master sends the calling address which match, it means that this device is selected. The HCR is I^2C Bus control register which defines the device enable or disable the I^2C Bus as a transmitter or as a receiver. The HSR is I^2C Bus status register, it responds with the I^2C Bus status. The HDR is input/output data register, data to transmit or receive must be via the HDR register.

The I^2C Bus control register contains three bits. The HEN bit defines whether to enable or disable the I^2C Bus. If the data wants to transfer via I^2C Bus, this bit must be set. The HTX bit defines whether the I^2C Bus is in transmit or receive mode. If the device is as a transmitter, this bit must be set to ″1″. The TXAK defines the transmit acknowledge signal, when the device received 8-bit data, the device sends this bit to I^2C Bus at the 9th clock. If the receiver wants to continue to receive the next data, this bit must be reset to ″0″ before receiving data.



Low Voltage Reset

Note: *1: To make sure that the system oscillator has stabilized, the SST provides an extra delay of 1024 system clock pulses before entering the normal operation.

*2: Since low voltage state has to be maintained in its original state for over 1ms, therefore after 1ms delay, the device enters the reset mode.

The I^2C Bus status register contains 5 bits. The HCF bit is reset to ″0″ when one data byte is being transferred. If one data transfer is completed, this bit is set to ″1″. The HAAS bit is set ″1″ when the address is match, and the I^2C Bus interrupt request flag is set to ″1″. If the interrupt is enabled and the stack is not full, a subroutine call to location 10H will occur. Writing data to the I^2C Bus control register clears HAAS bit. If the address is not match, this bit is reset to ″0″. The HBB bit is set to respond the I^2C Bus is busy. It mean that a START signal is detected. This bit is reset to ″0″ when the I^2C Bus is not busy. It means that a STOP signal is detected and the I^2C Bus is free. The SRW bit defines the read/write command bit, if the calling address is match. When HAAS is set to ″1″, the device check SRW bit to determine whether the device is working in transmit or receive mode. When SRW bit is set ″1″, it means that the master wants to read data from I^2C Bus, the slave device must write data to I^2C Bus, so the slave device is working in transmit mode. When SRW is reset to ″0″, it means that the master wants to write data to I^2C Bus, the slave device must read data from the bus, so the slave device is working in receive mode. The RXAK bit is reset ″0″ indicates an acknowledges signal has been received. In the transmit mode, the transmitter checks RXAK bit to know the receiver which wants to receive the next data byte, so the transmitter continue to write data to the I^2C Bus until the RXAK bit is set to ″1″ and the transmitter releases the SDA line, so that the master can send the STOP signal to release the bus.

The HADR bit7-bit1 define the device slave address. At the beginning of transfer, the master must select a device by sending the address of the slave device. The bit 0 is unused and is not defined. If the I^2C Bus receives a start signal, all slave device notice the continuity of the 8-bit data. The front of 7 bits is slave address and the first bit is MSB. If the address is match, the HAAS status bit is set and generate an I^2C Bus interrupt. In the ISR, the slave device must check the HAAS bit to know the I^2C Bus interrupt comes from the slave address that has match or completed one 8-bit data transfer. The last bit of the 8-bit data is read/write command bit, it responds in SRW bit. The slave will check the SRW bit to know if the master wants to transmit or receive data. The device check SRW bit to know it is as a transmitter or receiver.

Bit7~Bit1	Bit0
Slave Address	—

Note: ″—″ means undefined

HADR Register

The HDR register is the I^2C Bus input/output data register. Before transmitting data, the HDR must write the data which needs to be transmitted. Before receiving data, the device must dummy read data from HDR. Transmit or Receive data from I^2C Bus must be via the HDR register.

At the beginning of the transfer of the I^2C Bus, the device must initial the bus, the following are the notes for initialing the I^2C Bus:

Note:

1: Write the I^2C Bus address register (HADR) to define its own slave address.

2: Set HEN bit of I^2C Bus control register (HCR) bit 0 to enable the I^2C Bus.

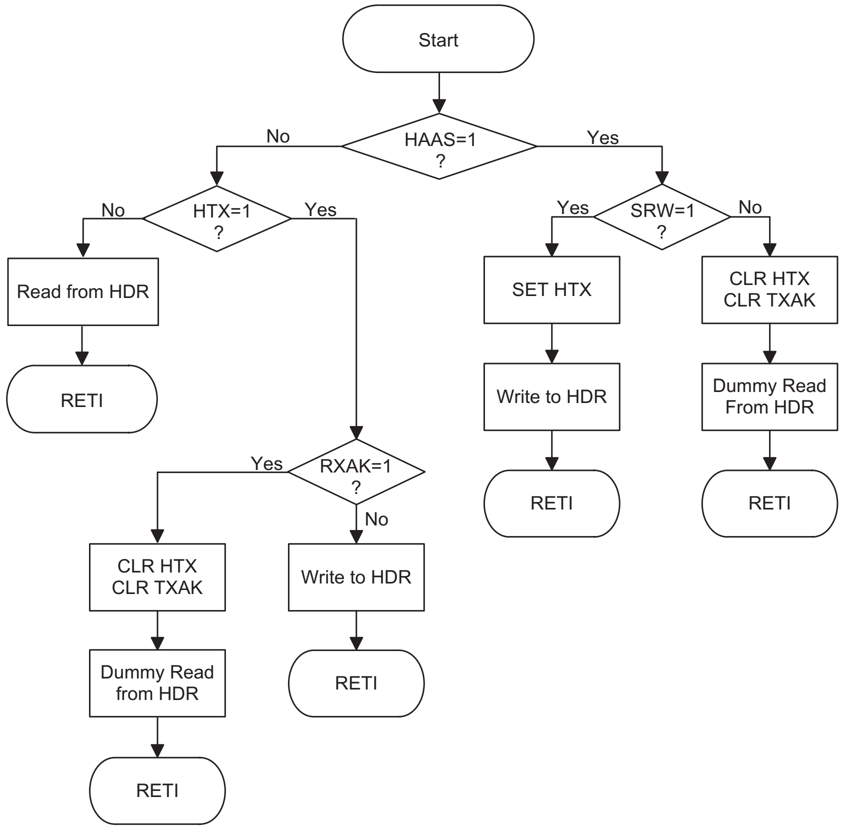
Label (HCR)	Bits	Function
HEN	7	Enable/disable I^2C Bus function (0= disable; 1= enable)
—	6~5	Unused bit, read as ″0″
HTX	4	Defines the transmit/receive mode (0= receive mode; 1= transmit)
TXAK	3	Enable/disable transmit acknowledge (0= acknowledge; 1= don′t acknowledge)
—	0~2	Unused bit, read as ″0″

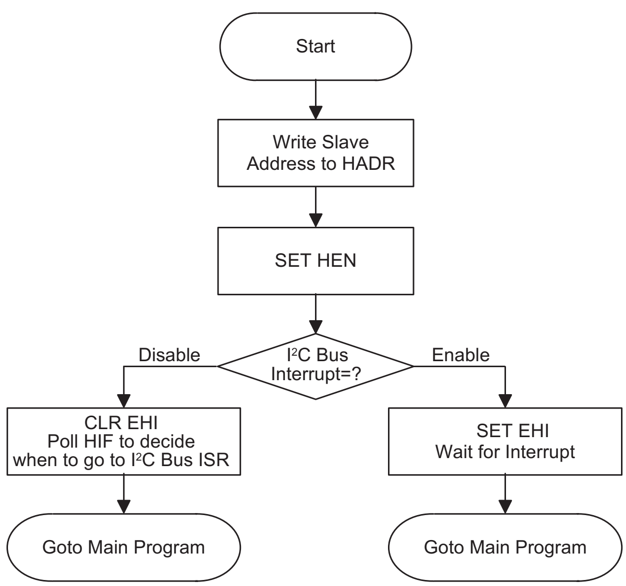
HCR Register

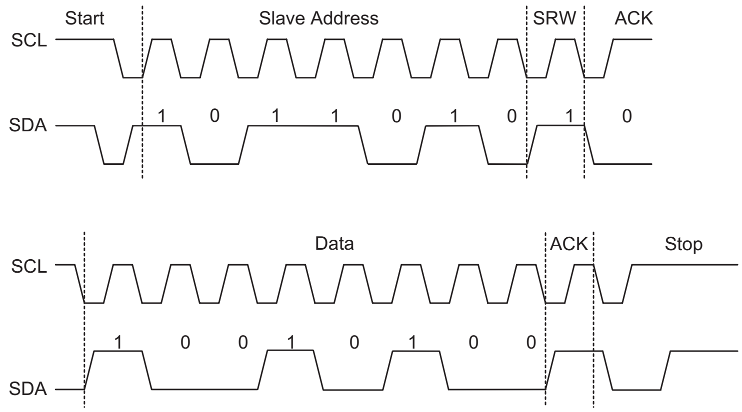
3: Set EHI bit of the interrupt control register 1 (INTC1) bit 0 to enable the I^2C Bus interrupt.

Label (HSR)	Bits	Function
HCF	7	HCF is cleared to ″0″ when one data byte is being transferred, HCF is set to ″1″ indicating 8-bit data communication has been finished.
HAAS	6	HAAS is set to ″1″ when the calling address has matched, and I^2C Bus interrupt will occur and HCF is set.
HBB	5	HBB is set to ″1″ when I^2C Bus is busy and HBB is cleared to ″0″ means that the I^2C Bus is not busy.
—	4~3	Unused bit, read as ″0″
SRW	2	SRW is set to ″1″ when the master wants to read data from the I^2C Bus, so the slave must transmit data to the master. SRW is cleared to ″0″ when the master wants to write data to the I^2C Bus, so the slave must receive data from the master.
—	1	Unused bit, read as ″0″
RXAK	0	RXAK is cleared to ″0″ when the master receives an 8-bit data and acknowledgment at the 9th clock, RXAK is set to ″1″ means not acknowledged.

HSR Register


Start
HAAS=1
?
No
Yes
HTX=1
?
No
Yes
Read from HDR
RETI
SRW=1
?
Yes
No
SET HTX
Write to HDR
RETI
CLR HTX
CLR TXAK
Dummy Read
From HDR
RETI
RXAK=1
?
Yes
No
CLR HTX
CLR TXAK
Dummy Read
from HDR
RETI
Write to HDR
RETI


Start
Write Slave
Address to HADR
SET HEN
I2C Bus
Interrupt=?
Disable
Enable
CLR EHI
Poll HIF to decide
when to go to I2C Bus ISR
Goto Main Program
SET EHI
Wait for Interrupt
Goto Main Program



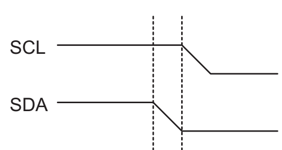
S=Start (1 bit)
SA=Slave Address (7 bits)
SR=SRW bit (1 bit)
M=Slave device send acknowledge bit (1 bit)
D=Data (8 bits)
A=ACK (RXAK bit for transmitter, TXAK bit for receiver 1 bit)
P=Stop (1 bit)

S	SA	SR	M	D	A	D	A	--------	S	SA	SR	M	D	A	D	A	--------	P

I^2C Communication Timing Diagram

Start Signal

The START signal is generated only by the master device. The other device in the bus must detect the START signal to set the I^2C Bus busy bit (HBB). The START signal is SDA line from high to low, when SCL is high.



Start Bit

Slave Address

The master must select a device for transferring the data by sending the slave device address after the START signal. All device in the I^2C Bus will receive the I^2C Bus slave address (7 bits) to compare with its own slave address (7 bits). If the slave address is matched, the slave device will generate an interrupt and save the following bit (8th bit) to SRW bit and sends an acknowledge bit (low level) to the 9th bit. The slave device also sets the status flag (HAAS), when the slave address is matched.

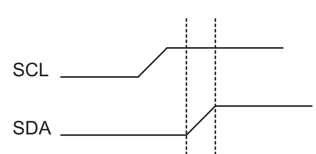
In interrupt subroutine, check HAAS bit to know whether the I^2C Bus interrupt comes from a slave address that is matched or a data byte transfer is completed. When the slave address is matched, the device must be in transmit mode or receive mode and write data to HDR or dummy read from HDR to release the SCL line.

SRW Bit

The SRW bit means that the master device wants to read from or write to the I^2C Bus. The slave device check this bit to understand itself if it is a transmitter or a receiver. The SRW bit is set to ″1″ means that the master wants to read data from the I^2C Bus, so the slave device must write data to a bus as a transmitter. The SRW is cleared to ″0″ means that the master wants to write data to the I^2C Bus, so the slave device must read data from the I^2C Bus as a receiver.

Acknowledge Bit

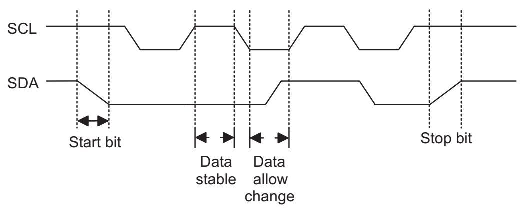
One of the slave device generates an acknowledge signal, when the slave address is matched. The master device can check this acknowledge bit to know if the slave device accepts the calling address. If no acknowledge bit, the master must send a STOP bit and end the communication. When the I^2C Bus status register bit 6 HAAS is high, it means the address is matched, so the slave must check SRW as a transmitter (set HTX) to ″1″ or as a receiver (clear HTX) to ″0″.



Stop Bit

Data Byte

The data is 8 bits and is sent after the slave device has acknowledged the slave address. The first bit is MSB and the 8th bit is LSB. The receiver sends the acknowledge signal (″0″) and continues to receive the next one byte data. If the transmitter checks and there′s no acknowledge signal, then it release the SDA line, and the master sends a STOP signal to release the I^2C Bus. The data is stored in the HDR register. The transmitter must write data to the HDR before transmitting data and the receiver must read data from the HDR after receiving data.



Data Timing Diagram

Receive Acknowledge Bit

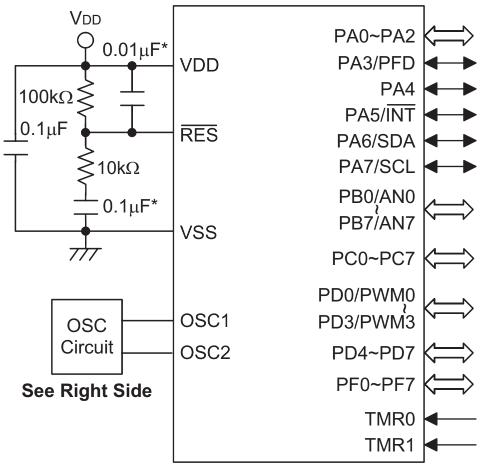
When the receiver wants to continue to receive the next data byte, it generates an acknowledge bit (TXAK) at the 9th clock. The transmitter checks the acknowledge bit (RXAK) to continue to write data to the I^2C Bus or change to receive mode and dummy read the HDR register to release the SDA line and the master sends the STOP signal.

Options

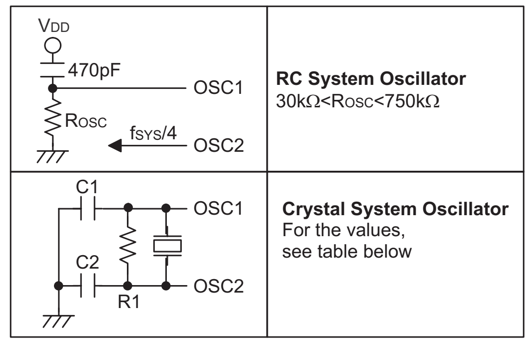
The following shows kinds of options in the device. ALL the options must be defined to ensure proper system function.

Options
OSC type selection. This option is to decide if an RC or crystal oscillator is chosen as system clock.
WDT source selection. There are three types of selection: on-chip RC oscillator, instruction clock or disable the WDT.
CLRWDT times selection. This option defines how to clear the WDT by instruction. ″One time″ means that the CLR WDT instruction can clear the WDT. ″Two times″ means only if both of the CLR WDT1 and CLR WDT2 instructions have been executed, then WDT can be cleared.
Wake-up selection. This option defines the wake-up function activity. External I/O pins (PA only) all have the capability to wake-up the chip from a HALT by a falling edge. (Bit option)
Pull-high selection. This option is to decide whether a pull-high resistance is visible or not in the input mode of the I/O ports. PA and PB are bit option; PC, PD and PF are port option.
PFD selection: If PA3 is set as PFD output, there are two types of selections; One is PFD0 as the PFD output, the other is PFD1 as the PFD output. PFD0, PFD1 are the timer overflow signals of the Timer/Event Counter 0, Timer/Event Counter 1 respectively.
PWM selection: (7+1) or (6+2) mode PD0: level output or PWM0 output PD1: level output or PWM1 output PD2: level output or PWM2 output PD3: level output or PWM3 output
WDT time-out period selection. There are four types of selection: WDT clock source divided by 2^{12}, 2^{13}, 2^{14} and 2^{15}
I^2C Bus function: enable or disable
LVR selection. LVR has enable or disable options

Application Circuits



HT46R24/HT46C24



OSC Circuit

The following table shows the C1, C2 and R1 value according different crystal values.

Crystal or Resonator	C1, C2	R1
4MHz Crystal	0pF	10kΩ
4MHz Resonator (3 pin)	0pF	12kΩ
4MHz Resonator (2 pin)	10pF	12kΩ
3.58MHz Crystal	0pF	10kΩ
3.58MHz Resonator (2 pin)	25pF	10kΩ
2MHz Crystal & Resonator (2 pin)	25pF	10kΩ
1MHz Crystal	35pF	27kΩ
480kHz Resonator	300pF	9.1kΩ
455kHz Resonator	300pF	10kΩ
429kHz Resonator	300pF	10kΩ

Note: The resistance and capacitance for reset circuit should be designed in such a way as to ensure that the VDD is stable and remains within a valid operating voltage range before bringing $\overline{RES}$ to high.

"*" Make the length of the wiring, which is connected to the $\overline{RES}$ pin as short as possible, to avoid noise interference.

Instruction Set Summary

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

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

Note: x: Immediate data

m: Data memory address

A: Accumulator

i: 0~7 number of bits

addr: Program memory address

√: Flag is affected

–: Flag is not affected

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

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

$^{(3)}$: $^{(1)}$ and $^{(2)}$

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

Instruction Definition

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

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

Operation: ACC ← ACC+[m]+C

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	√	√	√	√

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

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

Operation: [m] ← ACC+[m]+C

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	√	√	√	√

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

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

Operation: ACC ← ACC+[m]

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	√	√	√	√

ADD A,x Add immediate data to the accumulator

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

Operation: ACC ← ACC+x

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	√	√	√	√

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

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

Operation: [m] ← ACC+[m]

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	√	√	√	√

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

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

Operation: ACC ← ACC ″AND″ [m]

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

AND A,x Logical AND immediate data to the accumulator

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

Operation: ACC ← ACC ″AND″ x

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

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

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

Operation: [m] ← ACC ″AND″ [m]

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

CALL addr Subroutine call

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

Operation: Stack ← PC+1
PC ← addr

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

CLR [m] Clear data memory

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

Operation: [m] ← 00H

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

CLR [m].i Clear bit of data memory

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

Operation: [m].i ← 0

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

CLR WDT Clear Watchdog Timer

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

Operation: WDT ← 00H
PDF and TO ← 0

Affected flag(s)

TO	PDF	OV	Z	AC	C
0	0	—	—	—	—

CLR WDT1 Preclear Watchdog Timer

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

Operation: WDT ← 00H*
PDF and TO ← 0*

Affected flag(s)

TO	PDF	OV	Z	AC	C
0*	0*	—	—	—	—

CLR WDT2 Preclear Watchdog Timer

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

Operation: WDT ← 00H*
PDF and TO ← 0*

Affected flag(s)

TO	PDF	OV	Z	AC	C
0*	0*	—	—	—	—

CPL [m] Complement data memory

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

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

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

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

Operation: ACC ← $\overline{[m]}$

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

DAA [m] Decimal-Adjust accumulator for addition

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

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	√

DEC [m] Decrement data memory

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

Operation: [m] ← [m]–1

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

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

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

Operation: ACC ← [m]–1

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

HALT Enter power down mode

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

Operation: PC ← PC+1
PDF ← 1
TO ← 0

Affected flag(s)

TO	PDF	OV	Z	AC	C
0	1	—	—	—	—

INC [m] Increment data memory

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

Operation: [m] ← [m]+1

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

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

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

Operation: ACC ← [m]+1

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

JMP addr Directly jump

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

Operation: PC ←addr

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

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

Operation: ACC ← [m]

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

MOV A,x Move immediate data to the accumulator

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

Operation ACC ← x

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

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

Operation [m] ←ACC

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

NOP No operation

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

Operation PC ← PC+1

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

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

Operation ACC ← ACC ″OR″ [m]

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

OR A,x Logical OR immediate data to the accumulator

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

Operation ACC ← ACC ″OR″ x

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

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

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

Operation [m] ←ACC ″OR″ [m]

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

RET Return from subroutine

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

Operation: PC ← Stack

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

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

Operation: PC ← Stack
ACC ← x

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

RETI Return from interrupt

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

Operation: PC ← Stack
EMI ← 1

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

RL [m] Rotate data memory left

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

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

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

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

RLC [m] Rotate data memory left through carry

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

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	√

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

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

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	√

RR [m] Rotate data memory right

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

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

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

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

RRC [m] Rotate data memory right through carry

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

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	√

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

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

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	√

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

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

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	√	√	√	√

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

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

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	√	√	√	√

SDZ [m] Skip if decrement data memory is 0

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

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

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

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

SET [m] Set data memory

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

Operation: [m] ← FFH

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

SET [m]. i Set bit of data memory

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

Operation: [m].i ← 1

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

SIZ [m] Skip if increment data memory is 0

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

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

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

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

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

Operation: Skip if [m].i≠0

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

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

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	√	√	√	√

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

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

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	√	√	√	√

SUB A,x Subtract immediate data from the accumulator

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

Operation: ACC ← ACC+$\overline{x}$+1

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	√	√	√	√

SWAP [m] Swap nibbles within the data memory

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

Operation: [m].3~[m].0 ↔ [m].7~[m].4

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

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

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

SZ [m] Skip if data memory is 0

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

Operation: Skip if [m]=0

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

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

Operation: Skip if [m]=0

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

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

Operation: Skip if [m].i=0

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

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

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

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

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

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

Operation: ACC ← ACC ″XOR″ [m]

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

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

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

Operation: [m] ← ACC ″XOR″ [m]

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

XOR A,x Logical XOR immediate data to the accumulator

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

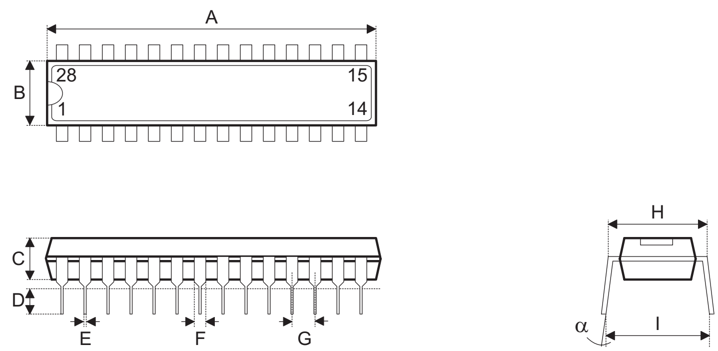
Operation: ACC ← ACC ″XOR″ x

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

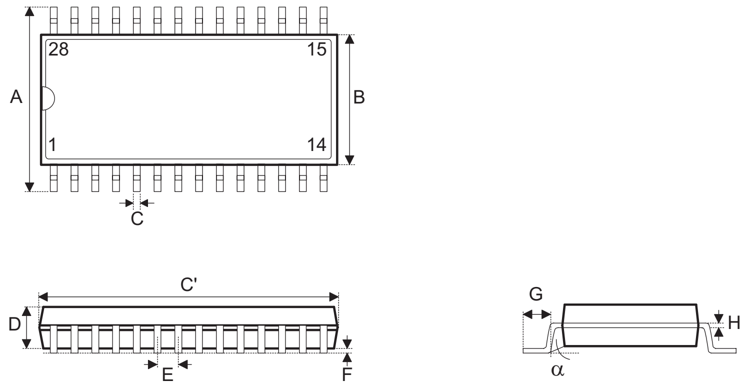
Package Information

28-pin SKDIP (300mil) Outline Dimensions



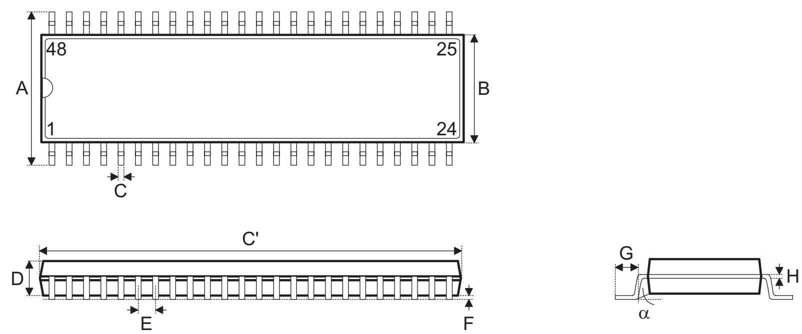
Symbol	Dimensions in mil		
	Min.	Nom.	Max.
A	1375	—	1395
B	278	—	298
C	125	—	135
D	125	—	145
E	16	—	20
F	50	—	70
G	—	100	—
H	295	—	315
I	330	—	375
α	0°	—	15°

28-pin SOP (300mil) Outline Dimensions



Symbol	Dimensions in mil		
	Min.	Nom.	Max.
A	394	—	419
B	290	—	300
C	14	—	20
C′	697	—	713
D	92	—	104
E	—	50	—
F	4	—	—
G	32	—	38
H	4	—	12
α	0°	—	10°

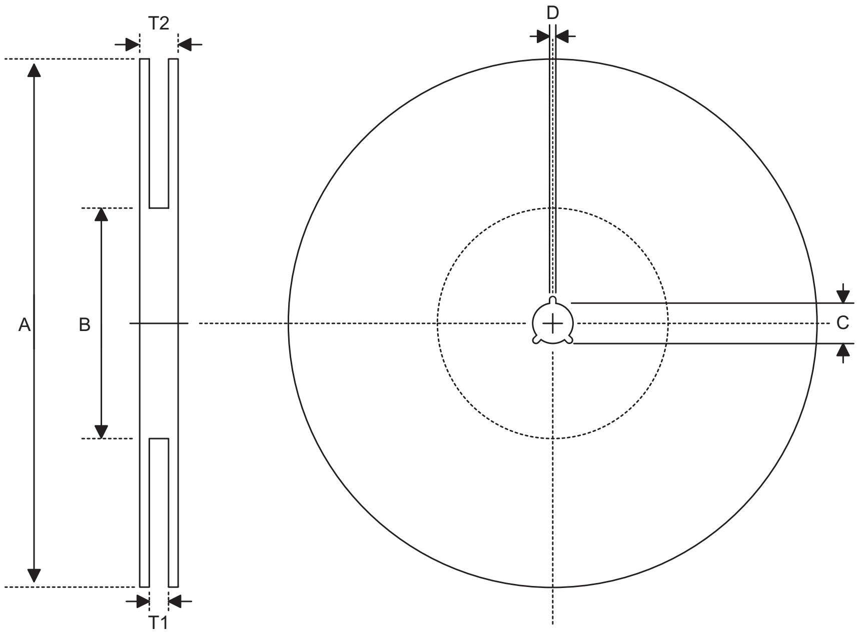
48-pin SSOP (300mil) Outline Dimensions



Symbol	Dimensions in mil		
	Min.	Nom.	Max.
A	395	—	420
B	291	—	299
C	8	—	12
C′	613	—	637
D	85	—	99
E	—	25	—
F	4	—	10
G	25	—	35
H	4	—	12
α	0°	—	8°

Product Tape and Reel Specifications

Reel Dimensions



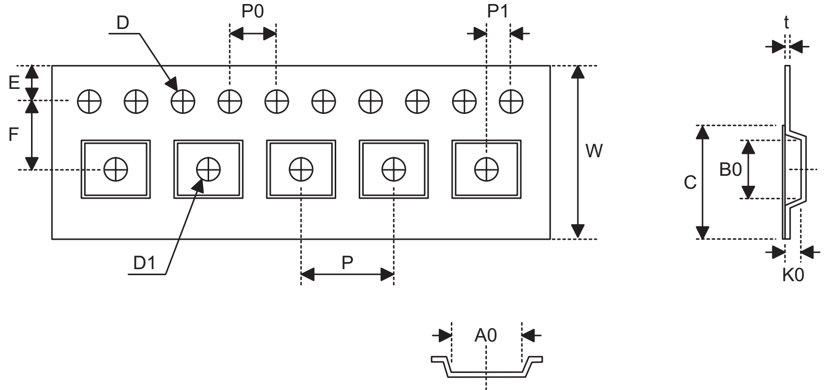
SOP 28W (300mil)

Symbol	Description	Dimensions in mm
A	Reel Outer Diameter	330±1.0
B	Reel Inner Diameter	62±1.5
C	Spindle Hole Diameter	13.0+0.5 –0.2
D	Key Slit Width	2.0±0.5
T1	Space Between Flange	24.8+0.3 –0.2
T2	Reel Thickness	30.2±0.2

SSOP 48W

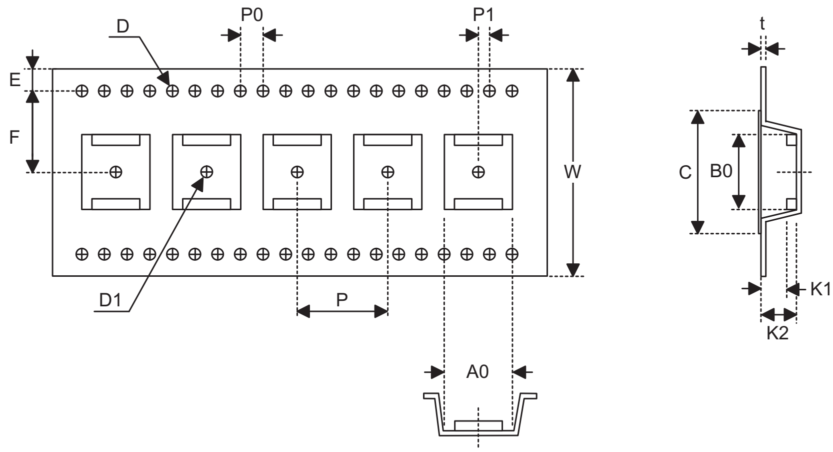
Symbol	Description	Dimensions in mm
A	Reel Outer Diameter	330±1.0
B	Reel Inner Diameter	100±0.1
C	Spindle Hole Diameter	13.0+0.5 –0.2
D	Key Slit Width	2.0±0.5
T1	Space Between Flange	32.2+0.3 –0.2
T2	Reel Thickness	38.2±0.2

Carrier Tape Dimensions



SOP 28W (300mil)

Symbol	Description	Dimensions in mm
W	Carrier Tape Width	24.0±0.3
P	Cavity Pitch	12.0±0.1
E	Perforation Position	1.75±0.1
F	Cavity to Perforation (Width Direction)	11.5±0.1
D	Perforation Diameter	1.5+0.1
D1	Cavity Hole Diameter	1.5+0.25
P0	Perforation Pitch	4.0±0.1
P1	Cavity to Perforation (Length Direction)	2.0±0.1
A0	Cavity Length	10.85±0.1
B0	Cavity Width	18.34±0.1
K0	Cavity Depth	2.97±0.1
t	Carrier Tape Thickness	0.35±0.01
C	Cover Tape Width	21.3



SSOP 48W

Symbol	Description	Dimensions in mm
W	Carrier Tape Width	32.0±0.3
P	Cavity Pitch	16.0±0.1
E	Perforation Position	1.75±0.1
F	Cavity to Perforation (Width Direction)	14.2±0.1
D	Perforation Diameter	2.0 Min.
D1	Cavity Hole Diameter	1.5+0.25
P0	Perforation Pitch	4.0±0.1
P1	Cavity to Perforation (Length Direction)	2.0±0.1
A0	Cavity Length	12.0±0.1
B0	Cavity Width	16.20±0.1
K1	Cavity Depth	2.4±0.1
K2	Cavity Depth	3.2±0.1
t	Carrier Tape Thickness	0.35±0.05
C	Cover Tape Width	25.5

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

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

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

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

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

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