HT46R14
A/D Type 8-Bit OTP MCU

Technical Document

- Tools Information
- FAQs
- Application Note
  - HA0004E HT48 & HT46 MCU UART Software Implementation Method
  - HA0005E Controlling the I2C bus with the HT48 & HT46 MCU Series
  - HA0011E HT48 & HT46 Keyboard Scan Program
  - HA0013E HT48 & HT46 LCM Interface Design
  - HA0102E Using the HT46R14 in a CCFL Lamp Inverter

Features

- Operating voltage:
  f_{SYS}=4MHz: 2.2V~5.5V
  f_{SYS}=8MHz: 3.3V~5.5V
- 20 bidirectional I/O lines
- Two interrupt input shared with an I/O line
- Two 8-bit programmable timer/event counter with overflow interrupt and 7-stage prescaler
- Two 8-bit programmable pulse generator (PPG) output channels, with prescaler and 8-bit programmable timer counter, and supports active low or active high output
- On-chip crystal and RC oscillator
- Watchdog Timer
- 4096×15 program memory
- 192×8 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
- 8-level subroutine nesting
- 8 channels 9-bit resolution A/D converter
- Two comparators with interrupt function
- Bit manipulation instruction
- 15-bit table read instruction
- 63 powerful instructions
- All instructions in one or two machine cycles
- Low voltage reset function
- 28-pin SKDIP/SOP package

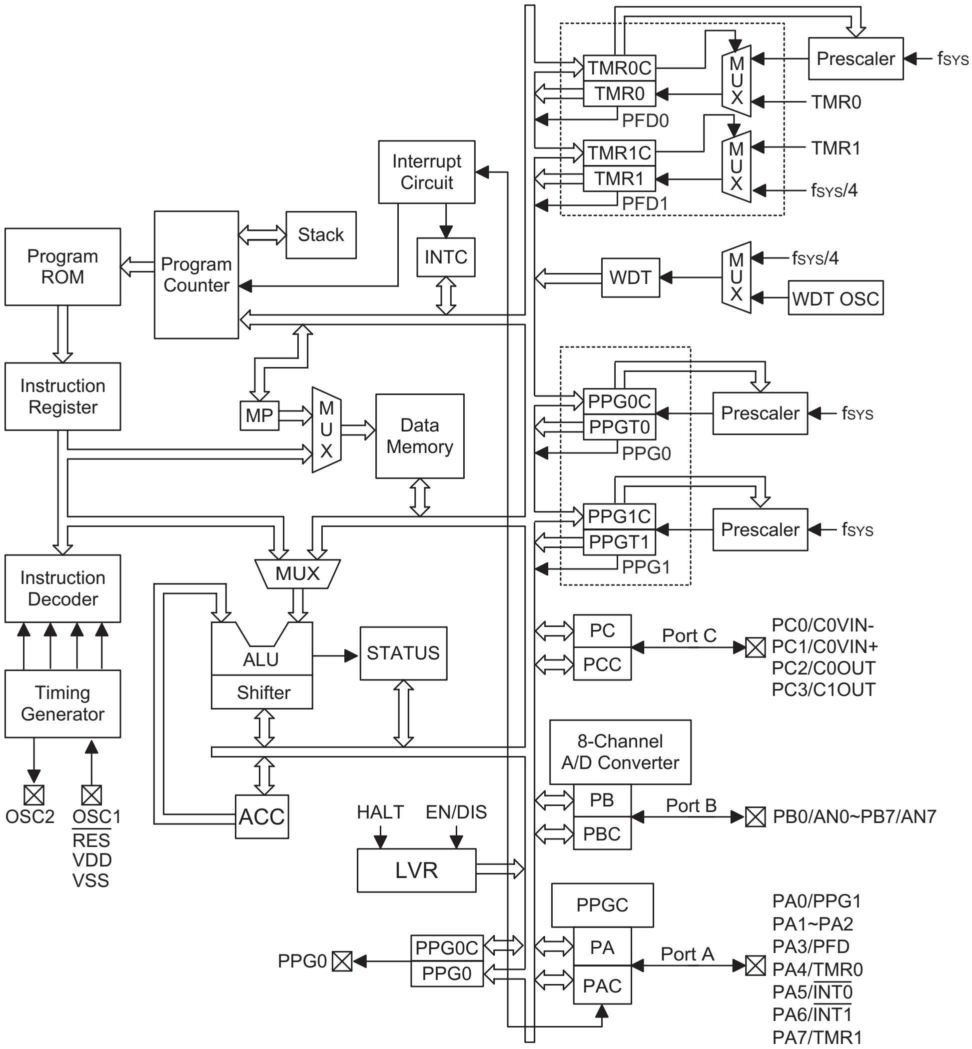
General Description

The device 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 advantages of low power consumption, I/O flexibility, programmable frequency divider, timer functions, oscillator options, multi-channel A/D Converter, 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, etc.

The device provides two comparators and programmable pulse generator (PPG). It is particularly suitable for use in products such as induction cooker and home appliances.

Block Diagram



Pin Assignment

HT46R14
— 28 SKDIP-A/SOP-A

Pin Description

Pin Name	I/O	Option	Description
PA0/PPG1 PA1~PA2 PA3/PFD PA4/TMR0 PA5/$\overline{\text{INT0}}$ PA6/$\overline{\text{INT1}}$ PA7/TMR1	I/O	Pull-high Wake-up PA3 or PFD	Bidirectional 8-bit input/output port. Each bit can be configured as wake-up input by options. Software instructions determine the CMOS output or Schmitt trigger input with or without pull-high resistor (determined by pull-high option: bit option). The PFD, $\overline{\text{INT0}}$ and $\overline{\text{INT1}}$ are pin-shared with PA3, PA5 and PA6. The TMR0 is pin-shared with PA4, TMR1 is pin shared with PA7, respectively. The PPG1 is a programmable pulse generator1 output pin, pin shared with PA0. The PPG1 or I/O function is selected via configuration option. The PPG1 output pin is floating during power-on reset, $\overline{\text{RES}}$ pin reset or LVR reset. The PPG1 output level (active low or active high) can be selected via configuration option.
PB0/AN0~ PB7/AN7	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 (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 disabled automatically.
PC0/C0VIN- PC1/C0VIN+ PC2/C0OUT PC3/C1OUT C1VIN+ C1VIN-	I/O	Pull-high I/O or Comparator	Bidirectional 4-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). C0VIN+, C0VIN- and C0OUT are pin-shared with PC1, PC0 and PC2. Once the Comparator 0 function is used, the internal registers related to PC1, PC0 cannot be used, PC2 can be used as input only, the PC1, PC0 I/O function, PC2 output function and pull-high resistor are disabled automatically. Software instructions determine the Comparator 0 function to be used. C1VIN+ and C1VIN- are Comparator 1 input, C1OUT is pin-shared with PC3. Once the Comparator 1 function is used, the internal register related to PC3 can be used as input only, the PC3 output function and pull-high resistor are disabled automatically. Software instructions determine the Comparator 1 function to be used.
PPG0	O	—	This is a programmable pulse generator 0 output pin, the pin is floating during power-on reset, $\overline{\text{RES}}$ pin reset or LVR reset. The PPG0 output level (active low or active high) can be selected via configuration option
OSC1 OSC2	I O	Crystal or RC	OSC1, OSC2 are connected to an RC network or a Crystal (determined by options) for the internal system clock. In the case of RC operation, OSC2 is the output terminal for 1/4 system clock.
$\overline{\text{RES}}$	I	—	Schmitt trigger reset input. Active low.
VDD	—	—	Positive power supply
VSS	—	—	Negative power supply, ground

Absolute Maximum Ratings

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

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

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

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
I_{DD3}	Operating Current (Crystal OSC, RC OSC)	5V	No load, f_{SYS}=8MHz ADC disable	—	4	8	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 and TMR1	—	—	0	—	$0.3V_{DD}$	V
V_{IH1}	Input High Voltage for I/O Ports, TMR0 and TMR1	—	—	$0.7V_{DD}$	—	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	—	—	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
V_{OS}	Comparator Input Offset Voltage	—	—	–10	—	10	mV
V_I	Comparator Input Voltage Range	—	—	0.2	—	V_{DD}–0.8	V

Ta=25°C

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	—	Power-up or 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_{COMP}	Comparator Response Time	—	—	—	—	3	μs

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

Functional Description

Execution Flow

The system clock for the microcontroller is derived from either a crystal or an RC oscillator. The system clock is internally divided into four non-overlapping clocks. One instruction cycle consists of four system clock cycles.

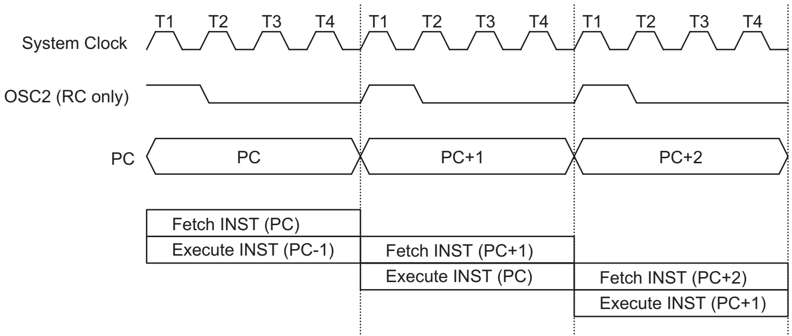
Instruction fetching and execution are pipelined in such a way that a fetch takes an instruction cycle while decoding and execution takes the next instruction cycle. However, the pipelining scheme causes each instruction to effectively execute in a cycle. If an instruction changes the program counter,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

Program Counter – PC

The program counter (PC) controls the sequence in which the instructions stored in program PROM are executed and its contents specify full range of program memory.

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

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



Execution Flow

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

Program Counter

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

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

The lower byte of the program counter (PCL) is a readable and writeable register (06H). Moving data into the PCL performs a short jump. The destination will be within 256 locations. When a control transfer takes place, an additional dummy cycle is required.

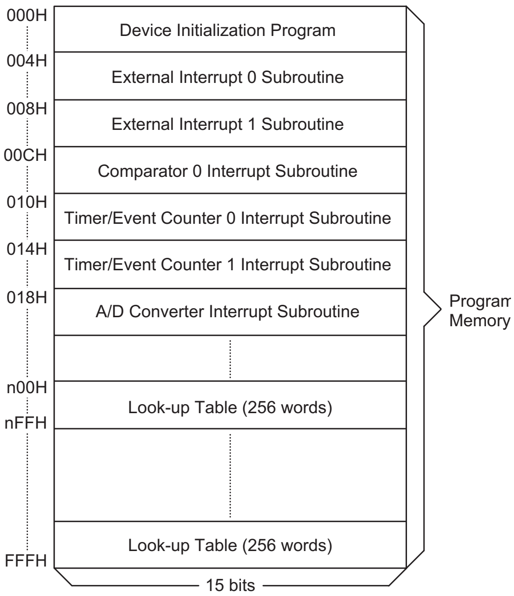
Program memory – ROM

The program memory is used to store the program instructions which are to be executed. It also contains data, table, and interrupt entries, and is organized into 4096×15 bits, addressed by the program counter and table pointer.

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

- Location 000H
  Location 000H is reserved for program initialization. After chip reset, the program always begins execution at location 000H.
- Location 004H
  Location 004H is reserved for the external interrupt 0 service program. If the $\overline{INT0}$ input pin is activated, 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 external Interrupt 1 service program. If the $\overline{INT1}$ input pin is activated, the interrupt is enabled and the stack is not full, the program begins execution at location 008H.
- Location 00CH
  Location 00CH is reserved for the Comparator 0 interrupt service program. If the Comparator 0 output pin is activated, and if the interrupt is enable and the stack is not full, the program begins execution at location 00CH.
- Location 010H
  Location 010H 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 010H.
- Location 014H
  Location 014H 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 014H.
- Location 018H
  Location 018H 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 018H.
- Table location
  Any location in the PROM space can be used as look-up tables. The instructions "TABRDC [m]" (the current page, 1 page=256 words) and "TABRDL [m]" (the last page) transfer the contents of the lower-order byte to the specified data memory, and the higher-order byte to TBLH (08H). Only the destination of the lower-order byte in the table is well-defined, the other bits of the table word are transferred to the lower portion of TBLH, and the remaining 1 bit is read as "0".

The Table Higher-order byte register (TBLH) is read only. The table pointer (TBLP) is a read/write register (07H), which indicates the table location. Before accessing the table, the location must be placed in TBLP. The TBLH is read only and cannot be restored. If the main routine and the ISR (Interrupt Service Routine) both employ the table read instruction, the contents of the TBLH in the main routine are likely to be changed by the table read instruction used in the ISR. Errors can occur. In other words, using the table read instruction in the main routine and the ISR simultaneously should be avoided. However, if the table read instruction has to be applied in both the main routine and the ISR, the interrupt is supposed to be disabled prior to the table read instruction. It will not be enabled until the TBLH has been backed up. All table related instructions require two cycles to complete the operation. These areas may function as normal program memory depending upon the requirements.



Note: n ranges from 0 to F

Program Memory

Stack Register – STACK

This is a special part of the memory which is used to save the contents of the program counter only. The stack is organized into 8 levels and is neither part of the data nor part of the program space, and is neither readable nor writeable. The activated level is indexed by the stack pointer (SP) and is neither readable nor writeable. At a subroutine call or interrupt acknowledgment, the contents of the program counter are pushed onto the stack. At the end of a subroutine or an interrupt routine, signaled by a return instruction (RET or RETI), the program counter is restored to its previous value from the stack. After a chip reset, the SP will point to the top of the stack.

If the stack is full and a non-masked interrupt takes place, the interrupt request flag will be recorded but the acknowledgment will be inhibited. When the stack pointer is decremented (by RET or RETI), the interrupt will be serviced. This feature prevents stack overflow allowing the programmer to use the structure more easily. In a similar case, if the stack is full and a "CALL" is subsequently executed, stack overflow occurs and the first entry will be lost (only the most recent 8 return addresses are stored).

Data Memory – RAM

The data memory is designed with 221×8 bits. The data memory is divided into two functional groups: special function registers and general purpose data memory (192×8). Most are read/write, but some are read only.

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), 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 (TMR0;0DH), a Timer/Event Counter 0 control register (TMR0C;0EH), a Timer/Event Counter 1 (TMR1:10H), a Timer/Event Counter 1 control register (TMR1C; 11H), Interrupt control register 1 (INTC1;1EH) , 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) and I/O control registers (PAC;13H, PBC;15H, PCC;

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

Table Location

Note: *11~*0: Table location bits
@7~@0: Table pointer bits
P11~P8: Current program counter bits

17H), the programmable pulse generator0 (PPG0) control register (PPG0C;20H), the programmable pulse generator timer register (PPGT0;21H), the programmable pulse generator1 (PPG1) control register (PPG1C;22H), the programmable pulse generator timer register (PPGT1;23H).

The remaining space before the 40H is reserved for future expanded usage and reading these locations will get ″00H″. The general purpose data memory, addressed from ″40H″ to ″FFH″, 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).

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

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

Arithmetic and Logic Unit – ALU

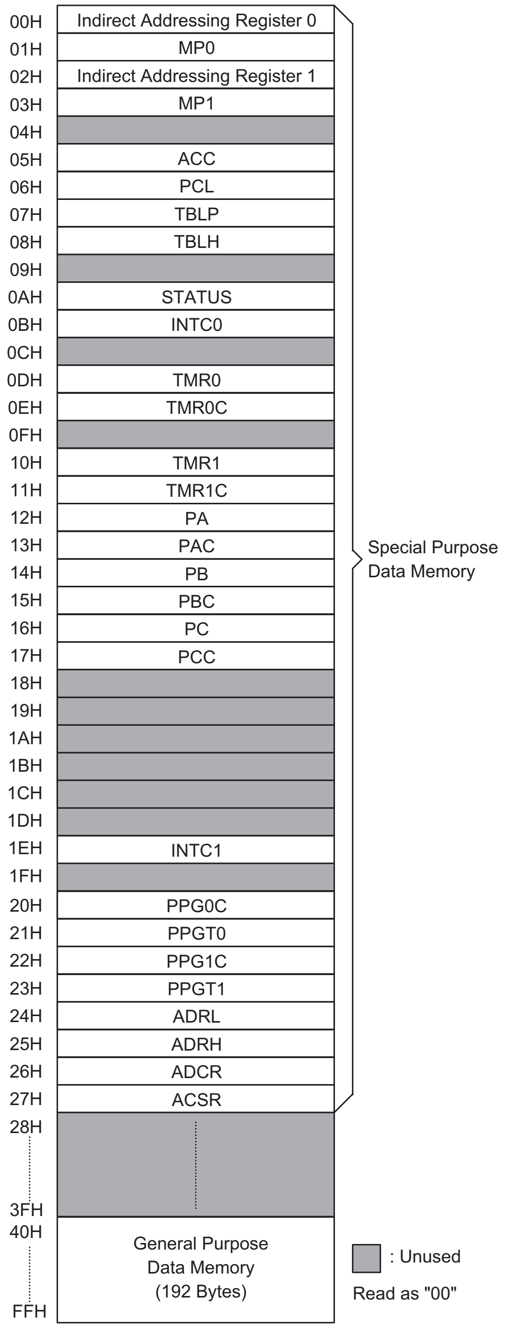
This circuit performs 8-bit arithmetic and logic operations. The ALU provides the following functions:

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

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

Status Register – STATUS

This 8-bit register (0AH) contains the zero flag (Z), carry flag (C), auxiliary carry flag (AC), overflow flag (OV), power down flag (PDF), and watchdog time-out flag



RAM Mapping

(TO). It also records the status information and controls the operation sequence.

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

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

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

Interrupt

The devices provides two external interrupts, two internal timer/event counter 0/1 interrupts, one comparator interrupt, and A/D converter interrupt. The interrupt control register 0 (INTC0;0BH) and interrupt control register 1 (INTC1;1EH) both contain the interrupt control bits that are used to set the enable/disable status and interrupt request flags.

Once an interrupt subroutine is serviced, other interrupts are all blocked (by clearing the EMI bit). This scheme may prevent any further interrupt nesting. Other interrupt requests may take place during this interval, but only the interrupt request flag will be recorded. If a certain interrupt requires servicing within the service routine, the EMI bit and the corresponding bit of the INTC0 or of INTC1 may be set in order to allow interrupt nesting. Once the stack is full, the interrupt request will not be acknowledged, even if the related interrupt is enabled, until the SP is decremented. If immediate service is desired, the stack should be prevented from becoming full.

All these interrupts can support a wake-up function. As an interrupt is serviced, a control transfer occurs by pushing the contents of the program counter onto the stack followed by a branch to a subroutine at the specified location in the ROM. Only the contents of the program counter is pushed onto the stack. If the contents of the register or of the status register (STATUS) is 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 $\overline{\mathrm{INT0}}$ or $\overline{\mathrm{INT1}}$, and the related interrupt request flag (EI0F;bit 4 of the INTC0, EI1F;bit 5 of the INTC0) is set as well. After the interrupt is enabled, the stack is not full, and the external interrupt is active, a subroutine call to location 04H or 08H occurs. The interrupt request flag (EI0F or EI1F) and EMI bits are all cleared to disable other interrupts.

The Comparator 0 output Interrupt is initialized by setting the Comparator 0 output Interrupt request flag (C0F; bit 6 of the INTC0), which is caused by a falling edge transition of comparator 0 output. After the interrupt is enabled, and the stack is not full, and the C0F bit is set, a subroutine call to location 0CH occurs. The related interrupt request flag (C0F) is reset, and the EMI bit is cleared to disable further maskable interrupts.

The internal Timer/Event Counter 0 interrupt is initialized by setting the Timer/Event Counter 0 interrupt request flag (T0F;bit 4 of the INTC1), 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 010H occurs. The related interrupt request flag (T0F) is reset, and the EMI bit is cleared to disable further interrupts. The Timer/Event Counter 1 is operated in the same manner, The Timer/Event Counter 1 related interrupt request flag is T1F (bit 5 of the INTC1) and its subroutine call location is 014H. The related interrupt request flag (T1F) will be reset and the EMI bit cleared to disable further interrupts.

Bit No.	Label	Function
0	C	C is set if an operation results in a carry during an addition operation or if a borrow does not take place during a subtraction operation, otherwise C is cleared. C is also affected by a rotate through carry instruction.
1	AC	AC is set if an operation results in a carry out of the low nibbles in addition or no borrow from the high nibble into the low nibble in subtraction, otherwise AC is cleared.
2	Z	Z is set if the result of an arithmetic or logic operation is zero, otherwise Z is cleared.
3	OV	OV is set if an operation results in a carry into the highest-order bit but not a carry out of the highest-order bit, or vice versa, otherwise OV is cleared.
4	PDF	PDF is cleared by system power-up or executing the ″CLR WDT″ instruction. PDF is set by executing the ″HALT″ instruction.
5	TO	TO is cleared by a system power-up or executing the ″CLR WDT″ or ″HALT″ instruction. TO is set by a WDT time-out.
6, 7	—	Unused bit, read as ″0″

Status (0AH) Register

The A/D converter interrupt is initialized by setting the A/D converter request flag (ADF; bit 6 of the INTC1), is 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 018H will occur. The related interrupt request flag (ADF) will be reset and the EMI bit cleared to disable further interrupts. After the interrupt is enabled, and a subroutine call to location 018H occurs.

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

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

Interrupt Source	Priority	Vector
External interrupt 0	1	04H
External interrupt 1	2	08H
Comparator 0 output interrupt	3	0CH
Timer/Event Counter 0 overflow	4	10H
Timer/Event Counter 1 overflow	5	14H
A/D converter interrupt	6	18H

The Comparator 0 interrupt request flag (C0F), external interrupt 1 request flag (EI1F), External Interrupt 0 request flag (EI0F), Enable Comparator 0 output interrupt bit (EC0I), Enable External interrupt 1 bit (EEI1), Enable External Interrupt 0 bit (EEI0), and enable master interrupt bit (EMI) make up of the Interrupt Control register 0 (INTC0) which is located at 0BH in the RAM.

Bit No.	Label	Function
0	EMI	Controls the master (global) interrupt (1=enable; 0=disable)
1	EEI0	Controls the external interrupt 0 (1=enable; 0=disable)
2	EEI1	Controls the external interrupt 1 (1=enable; 0=disable)
3	EC0I	Control the Comparator 0 interrupt (1= enable; 0= disable)
4	EI0F	External interrupt 0 request flag (1=active; 0=inactive)
5	EI1F	External interrupt 1 request flag (1=active; 0=inactive)
6	C0F	The Comparator 0 request flag (1=active; 0=inactive)
7	—	For test mode used only. Must be written as ″0″; otherwise may result in unpredictable operation.

INTC0 (0BH) Register

Bit No.	Label	Function
0	ET0I	Controls the Timer/Event Counter 0 interrupt (1=enable; 0=disable)
1	ET1I	Controls the Timer/Event Counter 1 interrupt (1=enable; 0=disable)
2	EADI	Controls the A/D converter interrupt (1=enable; 0=disable)
3	—	Unused bit, read as ″0″
4	T0F	Internal Timer/Event Counter 0 request flag (1=active; 0=inactive)
5	T1F	Internal Timer/Event Counter 1 request flag (1=active; 0=inactive)
6	ADF	The A/D converter request flag (1=active; 0=inactive)
7	—	Unused bit, read as ″0″

INTC1 (1EH) Register

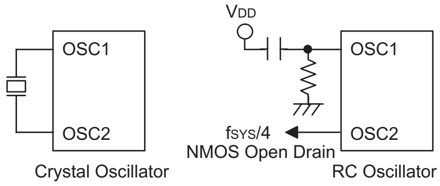
The A/D converter request flag (ADF), the Timer/Event Counter 1 interrupt request flag (T1F), the Timer/Event Counter 0 interrupt request flag (T0F), enable A/D converter interrupt bit (EADI), enable Timer/Event Counter 1 interrupt bit (ET1I), and enable Timer/Event Counter 0 interrupt bit (ET0I), constitute the Interrupt Control register 1 (INTC1) which is located at 1EH in the RAM.

EMI, EEI0, EEI1, EC0I, ET0I, ET1I, and EADI are all used to control the enable/disable status of interrupts. These bits prevent the requested interrupt from being serviced. Once the interrupt request flags (EI0F, EI1F, C0F, T0F, T1F, ADF) are all set, they remain in the INTC1 or INTC0 respectively 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 options. 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 24kΩ to 1MΩ. 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).

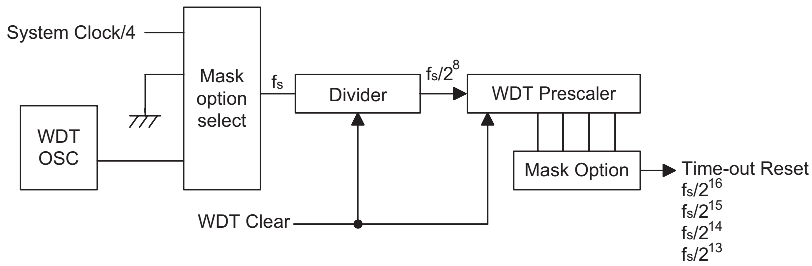
The WDT oscillator is a free running on-chip RC oscillator, and no external components are required. Even if the system enters the power down mode, the system clock is stopped, but the WDT oscillator still works within a period of approximately 65μs@5V. The WDT oscillator can be disabled by options to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

Watchdog Timer – WDT

The clock source of the WDT is implemented by a dedicated RC oscillator (WDT oscillator) or instruction clock (system clock divided by 4) determined 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 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 a period of 65μs/@5V normally) is selected, it is divided by 2^{13}, 2^{14}, 2^{15} or 2^{16} (by options) to get the WDT time-out period. The minimum WDT time-out period is about 600ms. This time-out period may vary with temperature, VDD and process variations. By selection the WDT options, longer time-out periods can be realized. If the WDT time-out is selected to $f_S/2^{16}$, the maximum time-out period is about 4.7s.

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.



Watchdog Timer

The WDT overflow under normal operation will initialize a ″chip reset″ and set the status bit TO. Whereas in the Halt mode, the overflow will initialize a ″warm reset″ in which only the program counter and stack pointer are reset to zero. To clear the contents of the 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 depending on the mask option – ″CLR WDT times selection option″. If the ″CLR WDT″ is selected (i.e. CLRWDT times equal one), any execution of the CLR WDT instruction will clear the WDT. In case ″CLR WDT1″ and ″CLR WDT2″ are chosen (i.e. CLRWDT times equal two), these two instructions must be executed to clear the WDT; otherwise, the WDT may reset the chip because of time-out.

If the WDT time-out period is selected $f_s/2^{12}$ (by options), 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 will be turned off but the WDT oscillator keeps running (if the WDT oscillator is selected).
- The contents of the on chip RAM and registers remain unchanged.
- WDT will be cleared and recounted again (if the WDT clock is from the WDT oscillator).
- All of the I/O ports maintain their original status.
- The PD flag is set and the TO flag is cleared.

The system can leave the HALT mode by means of an external reset, an interrupt, an external falling edge signal on port A or a WDT overflow. An external reset causes a device initialization and the WDT overflow performs a ″warm reset″. After the TO and PD flags are examined, the reason for chip reset can be determined. The PD flag is cleared by system power-up or executing the ″CLR WDT″ instruction and is set when executing the ″HALT″ instruction. The TO flag is set if the WDT time-out occurs, and causes a wake-up that only resets the program counter and stack pointer; the others keep 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 mask option. Awakening from an I/O port stimulus, the program will resume execution of the next instruction. If it awakens 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. If the interrupt is enabled and the stack is not full, the regular interrupt response takes place. If an interrupt request flag is set to ″1″ before entering the HALT mode, the wake-up function of the related interrupt will be disabled. Once a wake-up event occurs, it takes 1024 t_{SYS} (system clock period) to resume normal operation. In other words, a dummy period will be inserted after wake-up. If the wake-up results from an interrupt acknowledgment, the actual interrupt subroutine execution will be delayed by one or more cycles. If the wake-up results in the next instruction execution, this will be executed immediately after the dummy period is finished.

To 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 can 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 is different from other chip reset conditions, since it can perform a ″warm reset″ that resets only the program counter and SP, leaving the other circuits in their original state. Some registers remain unchanged during other reset conditions. Most registers are reset to the ″initial condition″ when the reset conditions are met. By examining the PDF and TO flags, the program can distinguish between different ″chip resets″.

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

Note: ″u″ means ″unchanged″

To guarantee that the system oscillator is started and stabilized, the SST (System Start-up Timer) provides an extra-delay of 1024 system clock pulses when the system reset (power-up, WDT time-out or $\overline{RES}$ reset) or the system awakes from the HALT state.

When a system reset occurs, the SST delay is added during the reset period. Any wake-up from HALT will enable the SST delay. The functional unit chip reset status are shown below.

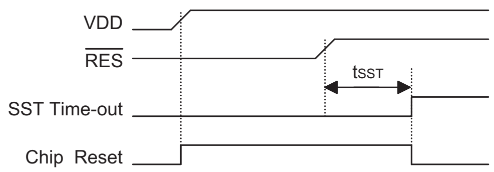
An extra option load time delay is added during system reset (power-up, WDT time-out at normal mode or $\overline{RES}$ reset).

Program Counter	000H
Interrupt	Disable
Prescaler, Divider	Cleared
WDT	Clear. After master reset, WDT begins counting
Timer/Event Counter	Off
PPG Timer	Off
PPG output	Floating
Input/Output Ports	Input mode
Stack Pointer	Points to the top of the stack

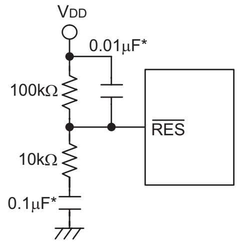
The registers states are summarized in the following table.

Register	Reset (Power On)	WDT Time-out (Normal Operation)	$\overline{RES}$ Reset (Normal Operation)	$\overline{RES}$ Reset (HALT)	WDT Time-out (HALT)*
MP0	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
MP1	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
Program Counter	000H	000H	000H	000H	000H
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
TMR0	xxxx xxxx	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
TMR0C	00-0 1000	00-0 1000	00-0 1000	00-0 1000	uu-u uuuu
TMR1	xxxx xxxx	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
TMR1C	00-0 1---	00-0 1---	00-0 1---	00-0 1---	uu-u u---
PA	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PB	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PBC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PC	---- 1111	---- 1111	---- 1111	---- 1111	---- uuuu
PCC	---- 1111	---- 1111	---- 1111	---- 1111	---- uuuu
INTC1	-000 -000	-000 -000	-000 -000	-000 -000	-uuu -uuu
PPG0C	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PPGT0	xxxx xxxx	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
PPG1C	0000 00-0	0000 00-0	0000 00-0	0000 00-0	uuuu uu-u
PPGT1	xxxx xxxx	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
ADRL	x--- ----	x--- ----	x--- ----	x--- ----	u--- ----
ADRH	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
ADCR	0100 0000	0100 0000	0100 0000	0100 0000	uuuu uuuu
ACSR	---- --00	---- --00	---- --00	---- --00	---- --uu

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

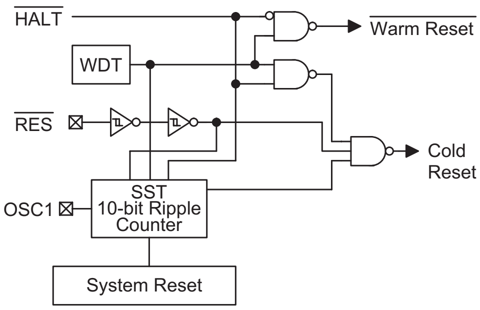


Reset Timing Chart



Reset Circuit

Note: "*" Make the length of the wiring, which is connected to the RES pin as short as possible, to avoid noise interference.



Reset Configuration

Timer/Event Counter

Two timer/event counters (TMR0, TMR1) are implemented in the microcontroller. The Timer/Event counter 0 contains an 8-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 8-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 system clock/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.

Using the internal system clock, the timer/event counter is only one reference time-base. The internal clock source comes from external events, measure time intervals or pulse widths, or generate an accurate time base. Using the internal clock allows the user to generate an accurate time base.

There are four registers related to the Timer/Event Counter 0; TMR0 (0DH), TMR0C (0EH), the Timer/Event Counter 1; TMR1(10H), TMR1C (11H). Writing TMR0/TMR1 makes the starting value be placed in the timer/event counter 0/1 preload register and reading TMR0/1 get the contents of the timer/event counter 0/1. The TMR0C and TMR1C are Timer/Event Counter control register 0/1, which defines the operating mode, counting enable or disable and an active edge.

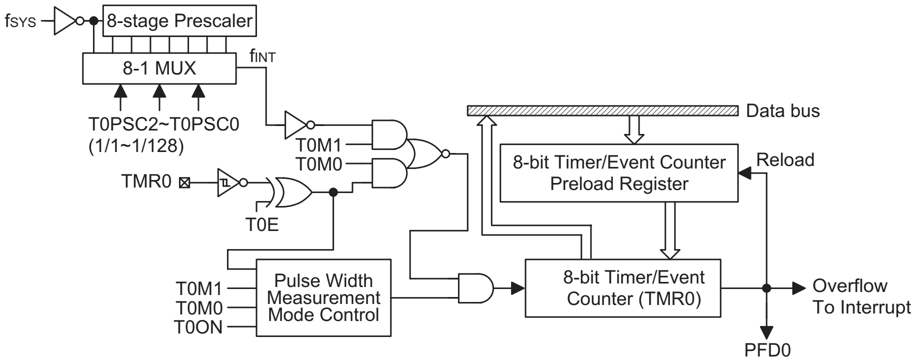
The T0M0/T1M0 and T0M1/T1M1 bits define the operation mode. The event count mode is used to count external events, which means that the clock source is from an external (TMR0, TMR1) pin. The timer mode functions as a normal timer with the clock source coming from the internal selected clock source. Finally, 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 0/1 starts counting at the current contents in the timer/event counter and ends at FFH. Once an overflow occurs, the counter is reloaded from the timer/event counter preload register, and generates an interrupt request flag (T0F; bit 4 of the INTC1, T1F; bit 5 of the INTC1).

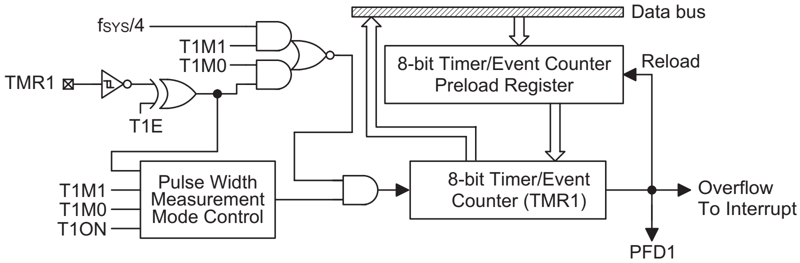
In the pulse width measurement mode with the values of the T0ON/T1ON and T0E/T1E bits equal to "1", after the TMR0/TMR1 has received a transient from low to high (or high to low if the T0E/T1E bit is "0"), it will start counting until the TMR0/TMR1 returns to the original level and resets the T0ON/T1ON.

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.

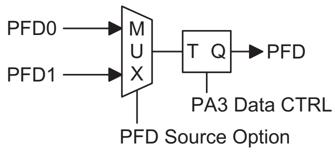
To enable the counting operation, the Timer ON bit (T0ON; bit 4 of the TMR0C or T1ON; bit 4 of the TMR1C) should be set to "1". In the pulse width measurement mode, the T0ON/T1ON is automatically cleared after the measurement cycle is completed. But in the other two modes, the T0ON/T1ON can only be reset by instructions. The overflow of the Timer/Event Counter 0/1 is one of the wake-up sources and the Timer/Event Counter 0/1 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. 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 the PFD output and executing "CLR [PA].3" instruction to disable the PFD output.



Timer/Event Counter 0



Timer/Event Counter 1



PFD Source Option

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.

Bit No.	Label	Function
0 1 2	T0PSC0 T0PSC1 T0PSC2	Define 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: $f_{INT}=f_{SYS}/128$
3	T0E	Defines the TMR0 active edge of the timer/event counter: In Event Counter Mode (T0M1,T0M0)=(0,1): 1:count on falling edge; 0:count on rising edge In Pulse Width measurement mode (T0M1,T0M0)=(1,1): 1: start counting on the rising edge, stop on the falling edge; 0: start counting on the falling edge, stop on the rising edge
4	T0ON	Enable/disable the timer counting (0=disable; 1=enable)
5	—	Unused bit, read as ″0″
6 7	T0M0 T0M1	Define the operating mode (T0M1, T0M0) 01=Event count mode (External clock) 10=Timer mode (Internal clock) 11=Pulse Width measurement mode (External clock) 00=Unused

TMR0C (0EH) Register

Bit No.	Label	Function
0~2	—	Unused bit, read as ″0″
3	T1E	Defines the TMR1 active edge of the timer/event counter: In Event Counter Mode (T1M1,T1M0)=(0,1): 1:count on falling edge; 0:count on rising edge In Pulse Width measurement mode (T1M1,T1M0)=(1,1): 1: start counting on the rising edge, stop on the falling edge; 0: start counting on the falling edge, stop on the rising edge
4	T1ON	Enable/disable timer counting (0= disable; 1= enable)
5	—	Unused bit, read as ″0″
6 7	T1M0 T1M1	Define the operating mode (T1M1, T1M0) 01= Event count mode (External clock) 10= Timer mode (Internal clock) 11= Pulse Width measurement mode (External clock) 00= Unused

TMR1C (11H) Register

Programmable Pulse Generator − PPG

This device provides two 8-bit PPG output channels. Each PPG has a programmable period of $256\times T$, where "T" can be $1/f_{SYS}$, $2/f_{SYS}$, $4/f_{SYS}$, $8/f_{SYS}$, $16/f_{SYS}$, $32/f_{SYS}$, $64/f_{SYS}$, $128/f_{SYS}$ for an output pulse width.

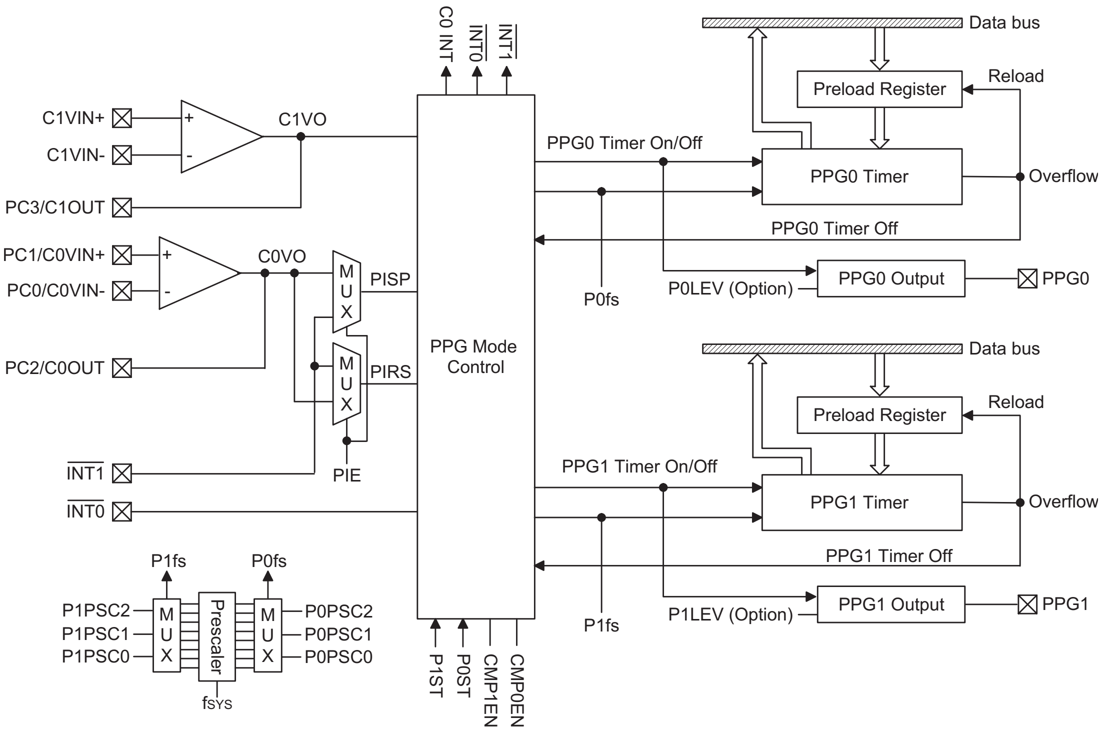
The PPG detects the falling edge of a trigger input, and outputs a single pulse, the falling edge trigger may come from comparator, $\overline{INT0}$, $\overline{INT1}$ or software trigger bit, which can be selected by software. The PPG is capable of generating signals from 0.25μs to 8.192ms pulse width when the system frequency is operating at 4MHz. The PPG can set the polarity control bit (PxLEV) as active low or active high output (by mask option). A "00H" data write to the PPGTx register yields a pulse width $256\times T$ output.

- PPG functional description

  The PPG module consists of PPG timers, a PPG Mode Control, two comparators. The PPG timer consists of a prescaler, one 8-bit up-counter timer, and an 8-bit preload data register. The programmable pulse generator (PPG) starts counting at the current contents in the preload register and ends at "FFH→00H". Once an overflow occurs, the counter is reloaded from the PPG timer counter preload register, and generates a signal to stop the PPG timer. The software trigger bit (PxST) will be cleared when the PPG timer overflow occurs.

There are four registers related to the PPG output function, two control registers PPG0C and PPG1C and two timer preload register PPGT0 and PPGT1. Two control registers PPG0C and PPG1C define the PPG0 and PPG1 input control mode (trigger source), enable or disable the comparators, define the PPG0 or PPG1 timer prescaler rate, range form $f_{SYS}/1$, $f_{SYS}/2$, $f_{SYS}/4$, $f_{SYS}/8$, $f_{SYS}/16$, $f_{SYS}/32$, $f_{SYS}/64$, $f_{SYS}/128$, enable or disable stopping the PPG0/PPG1 timer using PISP/$\overline{INT0}$ triggered input, enable or disable the restarting of the PPG0/PPG1 timer using C1VO/PIRS triggered input, and control the PPG0/PPG1 software trigger bit to trigger the PPG0/PPG1 timer On or Off. The PPGT0 is the PPG0 preload register and PPGT1 is PPG1 preload register, these two register contents decide the output pulse width.

The PPG1 output is pin-shared with the PA0, the function is selected via configuration option. If not selected, the pin can operate as a normal I/O pin, if the pin is selected as a PPG1 output pin, the I/O function is disabled automatically.



PPG Block Diagram

• PPG0C control register

Bit No.	7	6	5	4	3	2	1	0
PPG0C (20H)	P0ST	P0RSEN	P0SPEN	P0PSC2	P0PSC1	P0PSC0	CMP1EN	CMP0EN
POR value	0	0	0	0	0	0	0	0

CMP0EN: Enables or disables the Comparator 0 (0=disable, 1=enable)
CMP1EN: Enables or disables the Comparator 1 (0=disable, 1=enable)
P0PSC2, P0PSC1, P0PSC0: These three bits select the PPG0 timer prescaler rate.
P0SPEN: Enables or disables the stopping of the PPG0 timer using PISP trigger input (0=disable, 1=enable)
P0RSEN: Enables or disables the restarting of the PPG0 timer using C1VO trigger input. (0=disable, 1=enable)
P0ST: PPG0 software trigger bit. (0=Stop PPG0, 1=Restart PPG0)

The CMP0EN and CMP1EN bits are used as the enable or disable bits, if the CMP0EN is cleared to ″0″, the Comparator 0 is disabled, the PC0/C0VIN-, PC1/C0VIN+, PC2/C0OUT are all GPIO pins, if the CMP0EN is set to ″1″, the Comparator 0 is enabled, the PC0/C0VIN-, PC1/C0VIN+ are Comparator 0 input pins, PC2/C0OUT is a Comparator 0 output pin, PC2 output and Pull-high resistor are disabled, PC2 can still be used as input pin, the status of C0VO can be read by reading the PC2 register. If the CMP1EN is cleared to ″0″, the Comparator 1 is disabled, the PC3/C1OUT is a GPIO pin, If the CMP1EN is set to ″1″, the Comparator 1 is enabled, PC3/C1OUT is a Comparator 1 output pin, PC3 output and PC3 Pull-high resistor are disabled, PC3 can still be used as input pin, the status of C1VO can be read by reading the PC3 register.

PPG0C: CMP0EN, CMP1EN comparators enable/disable bits

CMP0EN	Description
0	Disable the Comparator 0. PC0/C0VIN-, PC1/C0VIN+, PC2/C0OUT are all GPIO pins.
1	Enable the Comparator 0, the PC0/C0VIN-, PC1/C0VIN+ are Comparator 0 input pins, PC2/C0OUT is a Comparator 0 output pin, PC2 output disable, PC2 Pull-high resister disable.

CMP1EN	Description
0	Disable the Comparator 1. PC3/C1OUT is a PGIO pin.
1	Enable the Comparator 1. PC3/C1OUT is a Comparator 1 output pin, PC3 output disable, PC3 Pull-high resistor disable.

The bits 4~2 of the PPG0 control register (PPG0C) can be used to define the pre-scaling stages of the PPG0 timer counter clock.

PPG0C: PPG0 timer prescaler rate bits

P0PSC2	P0PSC1	P0PSC0	Define the prescaler stages
0	0	0	$P0f_S=f_{SYS}$
0	0	1	$P0f_S=f_{SYS}/2$
0	1	0	$P0f_S=f_{SYS}/4$
0	1	1	$P0f_S=f_{SYS}/8$
1	0	0	$P0f_S=f_{SYS}/16$
1	0	1	$P0f_S=f_{SYS}/32$
1	1	0	$P0f_S=f_{SYS}/64$
1	1	1	$P0f_S=f_{SYS}/128$

The P0SPEN is the PPG0 stopping enable or disable bit using PISP trigger input, if this bit is enabled, the PPG0 stopping input can be triggered by PISP falling edge. The PISP signal may come from C0VO, PC2 or $\overline{\text{INT1}}$, determined by PIE (bit0 of PPG1C). The P0RSEN is the PPG0 restarting enable or disable bit using C1VO triggered input, if this bit is enabled, the PPG0 timer restarting input can be triggered by C1VO or PC3 falling edge. User can read the status of C0VO or C1VO by setting the PC2 or PC3 to be an input pin when Comparator 0 or Comparator 1 is enabled.

P0SPEN	Description
0	Disable stopping the PPG0 timer using C0VO trigger input. PPG0 module output can be stopped by software control (P0ST) only.
1	Enable stopping the PPG0 timer using PISP trigger input. PPG0 module output can be stopped by PISP (C0VO or $\overline{\text{INT1}}$) falling edge trigger or software control (P0ST bit is cleared to ″0″).

P0RSEN	Description
0	Disable restarting the PPG0 timer using C1VO trigger input. PPG0 module output can be restarted by software control (P0ST) only.
1	Enable restarting the PPG0 timer using C1VO trigger input. PPG0 module output can be restarted by C1VO falling edge trigger or software control (P0ST is set to ″1″)

The P0ST is a software trigger bit, if this bit is set to ″1″, the PPG0 timer will start counting and this bit will be cleared when the PPG0 timer overflow occurs or the PPG0 timer stop counting, if this bit is cleared to ″0″, the PPG0 timer will stop counting, when the PPG0 timer is counting and if a falling edge is generated from C1VO, PC3 or a software control bit (P0ST) is set, the PPG0 timer counter is not affected, the trigger from C1V0, PC3 or P0ST is not useful. The P0ST can also be used as a status bit of the PPG0 timer output.

- PPG1C control register

Bit No.	7	6	5	4	3	2	1	0
PPG1C (22H)	P1ST	P1RSEN	P1SPEN	P1PSC2	P1PSC1	P1PSC0	—	PIE
POR value	0	0	0	0	0	0	—	0

PIE: PPG input exchange bit (0=disable, 1=enable).

—: Undefined, cannot be used

P1PSC2, P1PSC1, P1PSC0: These three bits select the PPG1 timer prescaler rate.

P1SPEN: Enables or disables stopping the PPG1 timer using INT0 trigger input. (0=disable, 1=enable)

P1RSEN: Enables or disables restarting the PPG1 timer using PIRS trigger input. (0=disable, 1=enable)

P1ST: PPG1 software trigger bit. (0=Stop PPG1, 1=Restart PPG1)

The PIE bit is used as C0VO and $\overline{\text{INT1}}$ exchange bit. When PIE bit is reset to ″0″, the PISP signal comes from $\overline{\text{INT1}}$ and the PIRS signal comes from C0VO. When PIE bit is set to ″1″, the PISP signal comes from C0VO and the PIRS signal comes from $\overline{\text{INT1}}$.

The P1SPEN and P1RSEN should be disabled before setting the PIE bit.

PPG1C – PIE; C0VO and $\overline{\text{INT1}}$ exchange bit

PIE	Description
0	PISP signal comes from $\overline{\text{INT1}}$ and the PIRS signal comes form C0VO
1	PISP signal comes from C0VO and the PIRS signal comes from $\overline{\text{INT1}}$.

The bits 4~2 of the PPG1 control register (PPG1C) can be used to define the pre-scaling stages of the PPG1 timer counter clock.

PPG1C: PPG1 timer prescaler rate bits

P1PSC2	P1PSC1	P1PSC0	To define the prescaler stages
0	0	0	$P1f_S=f_{SYS}$
0	0	1	$P1f_S=f_{SYS}/2$
0	1	0	$P1f_S=f_{SYS}/4$
0	1	1	$P1f_S=f_{SYS}/8$
1	0	0	$P1f_S=f_{SYS}/16$
1	0	1	$P1f_S=f_{SYS}/32$
1	1	0	$P1f_S=f_{SYS}/64$
1	1	1	$P1f_S=f_{SYS}/128$

The P1SPEN is the PPG1 timer Off enable or disable bit using $\overline{INT0}$ trigger input, if this bit is enabled, the PPG1 stopping input can be triggered by $\overline{INT0}$ falling edge. The P1RSEN is the PPG1 restarting enable or disable bit using trigger input, if this bit is enabled, the PPG1 timer restarting input can be triggered by PIRS falling edge. The PIRS signal may come from C0VO, PC2 or $\overline{INT1}$, determined by PIE (bit0 of the PPG1C). User can read the status of C0VO or C1VO by setting the PC2 or PC3 as an input pin when Comparator 0 or Comparator 1 is enabled.

P1SPEN	Description
0	Disable stopping the PPG1 timer using $\overline{INT0}$ trigger input. PPG1 module output can be stopped by software control (P1ST) only.
1	Enable stopping the PPG0 timer using INT0 trigger input. PPG0 module output can be stopped by $\overline{INT0}$ falling edge trigger or software control (P1ST bit is cleared to ″0″).

P1RSEN	Description
0	Disable restarting the PPG1 timer using PIRS trigger input. PPG1 module output can be restarted by software control (P1ST) only
1	Enable restarting the PPG1 timer using PIRS trigger input. PPG1 module output can be restarted by PIRS (C0VO or INT1) falling edge trigger or software control (P1ST is set to ″1″)

The P1ST is a software trigger bit, if this bit is set to ″1″, the PPG1 timer will start counting and this bit will be cleared when the PPG1 timer overflow occurs or PPG1 timer stop counting, if this bit is cleared to ″0″, the PPG1 timer will stop counting, when the PPG timer is counting and if a falling edge is generated from PIRS or a software control bit (P1ST) is set, the PPG1 timer counter is not affected, the trigger from PIRS or P1ST is not useful. The P1ST can also be used as a status bit of PPG1 timer output.

Input/Output Ports

There are 20 bidirectional input/output lines in the microcontroller, labeled as PA, PB, PC, which are mapped to the data memory of [12H], [14H] and [16H] 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 or 16H). For output operation, all the data is latched and remains unchanged until the output latch is rewritten.

Each I/O line has its own control register (PAC, PBC, PCC) 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 and 17H.

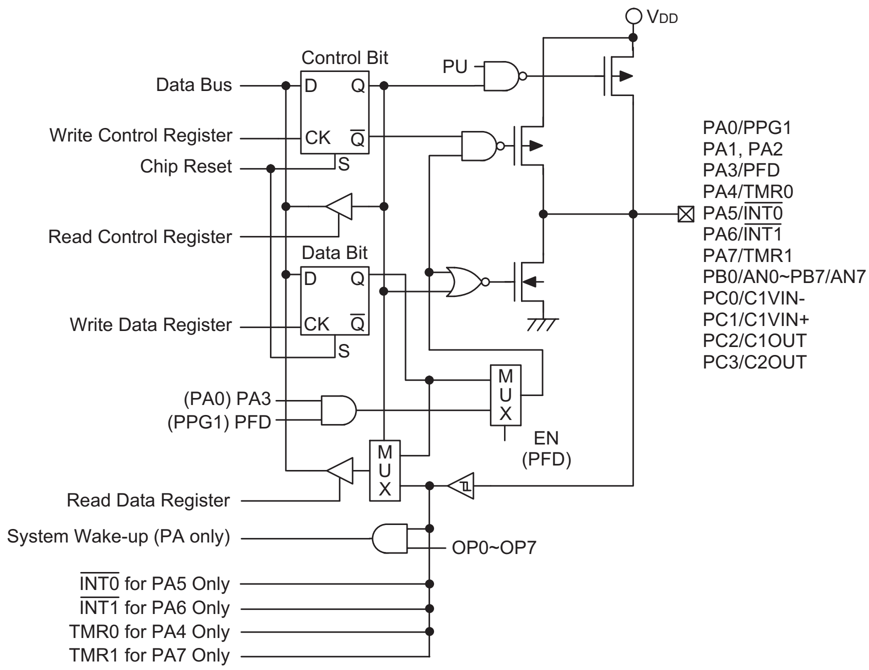
After a chip reset, these input/output lines remain at high levels or floating state (depending on pull-high options). Each bit of these input/output latches can be set or cleared by ″SET [m].i″ and ″CLR [m].i″ (m=12H, 14H or 16H) instructions.

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

Each line of port A has the capability of waking-up the device. 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 PA0, PA3, PA4, PA5, PA6 and PA7 are pin-shared with PPG1, PFD, TMR0, $\overline{\text{INT0}}$, $\overline{\text{INT1}}$ and TMR1 pins respectively. And the PC0, PC1, PC2 and PC3 are pin-shared with C0VIN1-, C0VIN+, C0OUT, C1OUT.

The PA3 is pin-shared with the PFD signal. If the PFD option is selected, the output signal in output mode of PA3 will be the PFD signal generated by the 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 a ″1″ to PA3 data register will enable the PFD output function and writing ″0″ will force the PA3 to remain at ″0″.



Input/Output Ports

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.

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.

The PFD (PFD0 or PFD1) output shares pin with PA3, as determined by options. When the PFD (PFD0 or PFD1) option is selected, setting PA3 ″1″ (″SET PA.3″) will enable the PFD output and setting PA3 ″0″ (″CLR PA.3″) will disable the PFD output and PA3 output at low level.

The definitions of PFD control signal and PFD output frequency are listed in the following table.

Timer	Timer Preload Value	PA3 Data Register	PA3 Pad State	PFD Frequency
OFF	X	0	0	X
OFF	X	1	U	X
ON	N	0	0	X
ON	N	1	PFD	$f_{TMR}/[2\times(M-N)]$

Note: ″X″ stands for unused
″U″ stands for unknown
″M″ is ″256″ for PFD
″N″ is preload value for timer/event counter
″f_{TMR}″ is input clock frequency for timer/event counter

A/D Converter

The 8 channels and 9-bit resolution A/D (8-bit accuracy) converter are implemented in this microcontroller. The reference voltage is VDD. The A/D converter contains four 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 ($0\rightarrow1\rightarrow0$). 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′s a total of eight channels to select. The bit5~bit3 of the ADCR are used to set the PB configurations. PB can be an analog input or as digital I/O line determined 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 powered 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).

Bit 7 of the ACSR register is used for test purposes only and must not be used for other purposes by the application program. Bit1 and bit0 of the ACSR register are used to select the A/D clock source.

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

Important Note for A/D initialization:
Special care must be taken to initialize the A/D converter each time the Port B A/D channel selection bits are modified, otherwise the EOCB flag may be in an undefined condition. An A/D initialization is implemented by setting the START bit high and then clearing it to zero within 10 instruction cycles of the Port B channel selection bits being modified. Note that if the Port B channel selection bits are all cleared to zero then an A/D initialization is not required.

Bit No.	Label	Function
0 1	ADCS0 ADCS1	Selects the A/D converter clock source 00=system clock/2 01=system clock/8 10=system clock/32 11=undefined
2~6	—	Unused bit, read as ″0″
7	TEST	For test mode used only

ACSR (27H) Register

Bit No.	Label	Function
0 1 2	ACS0 ACS1 ACS2	ACS2, ACS1, ACS0: Select A/D channel 0, 0, 0: AN0 0, 0, 1: AN1 0, 1, 0: AN2 0, 1, 1: AN3 1, 0, 0: AN4 1, 0, 1: AN5 1, 1, 0: AN6 1, 1, 1: AN7
3 4 5	PCR0 PCR1 PCR2	Defines the port B configuration select. If PCR0, PCR1 and PCR2 are all zero, the ADC circuit is powered off to reduce power consumption
6	EOCB	Indicates end of A/D conversion. (0 = end of A/D conversion) Each time bits 3~5 change state the A/D should be initialized by issuing a START signal, otherwise the EOCB flag may have an undefined condition. See ″Important note for A/D initialization″.
7	START	Starts the A/D conversion. (0→1→0= start; 0→1= Reset A/D converter and set EOCB to ″1″)

ADCR (26H) Register

Register	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADRL (24H)	D0	—	—	—	—	—	—	—
ADRH (25H)	D8	D7	D6	D5	D4	D3	D2	D1

Note: D0~D8 is A/D conversion result data bit LSB~MSB.

ADRL (24H), ADRH (25H) Register

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

Port B Configuration

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.

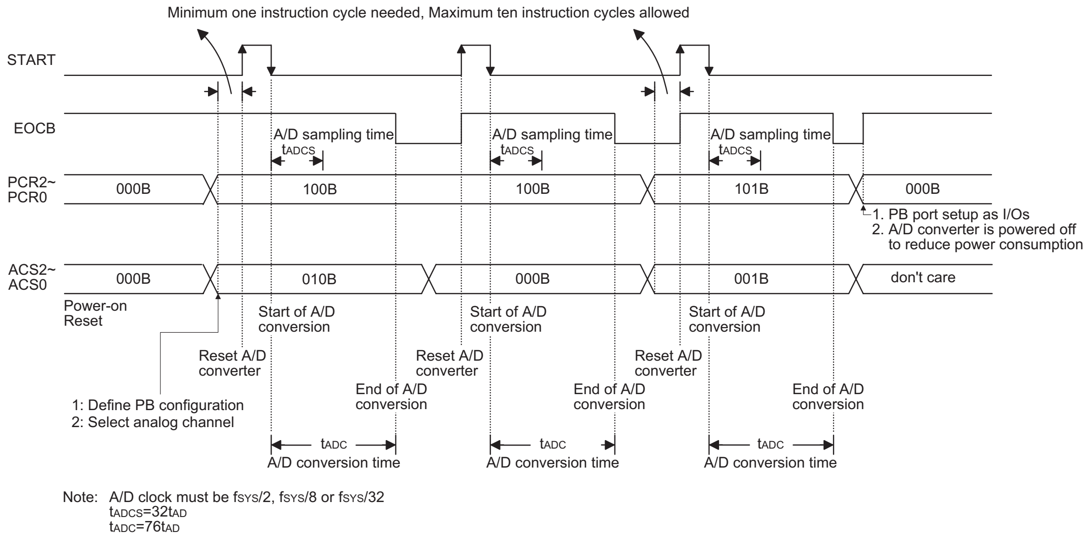
Example: using EOCB Polling Method to detect end of conversion

```
      clr     EADI                ; disable ADC interrupt
      mov     a,00000001B
      mov     ACSR,a              ; setup the ACSR register to select fSYS/8 as the A/D clock
      mov     a,00100000B         ; setup ADCR register to configure Port PB0~PB3 as A/D inputs
      mov     ADCR,a              ; and select AN0 to be connected to the A/D converter
              :
              :                   ; As the Port B channel bits have changed the following START
                                  ; signal (0-1-0) must be issued within 10 instruction cycles
              :
Start_conversion:
      clr     START
      set     START               ; reset A/D
      clr     START               ; start A/D
Polling_EOC:
      sz      EOCB                ; poll the ADCR register EOCB bit to detect end of A/D conversion
      jmp     polling_EOC         ; continue polling
      mov     a,ADRH              ; read conversion result high byte value from the ADRH register
      mov     adrh_buffer,a       ; save result to user defined memory
      mov     a,ADRL              ; read conversion result low byte value from the ADRL register
      mov     adrl_buffer,a       ; save result to user defined memory
              :
              :
      jmp     start_conversion    ; start next A/D conversion
```

Example: using interrupt method to detect end of conversion

```
      clr     EADI                ; disable ADC interrupt
      mov     a,00000001B
      mov     ACSR,a              ; setup the ACSR register to select fSYS/8 as the A/D clock

      mov     a,00100000B         ; setup ADCR register to configure Port PB0~PB3 as A/D inputs
      mov     ADCR,a              ; and select AN0 to be connected to the A/D converter
              :
                                  ; As the Port B channel bits have changed the following START
                                  ; signal (0-1-0) must be issued within 10 instruction cycles
              :
Start_conversion:
      clr     START
      set     START               ; reset A/D
      clr     START               ; start A/D
      clr     ADF                 ; clear ADC interrupt request flag
      set     EADI                ; enable ADC interrupt
      set     EMI                 ; enable global interrupt
              :
              :
              :
; ADC interrupt service routine
ADC_ISR:
      mov     acc_stack,a         ; save ACC to user defined memory
      mov     a,STATUS
      mov     status_stack,a      ; save STATUS to user defined memory
              :
              :
      mov     a,ADRH              ; read conversion result high byte value from the ADRH register
      mov     adrh_buffer,a       ; save result to user defined register
      mov     a,ADRL              ; read conversion result low byte value from the ADRL register
      mov     adrl_buffer,a       ; save result to user defined register
      clr     START
      set     START               ; reset A/D
      clr     START               ; start A/D
              :
              :
EXIT_INT_ISR:
      mov     a,status_stack
      mov     STATUS,a            ; restore STATUS from user defined memory
      mov     a,acc_stack         ; restore ACC from user defined memory
      reti
```



A/D Conversion Timing

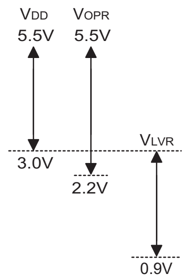
Low Voltage Reset – LVR

The microcontroller provides a 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.

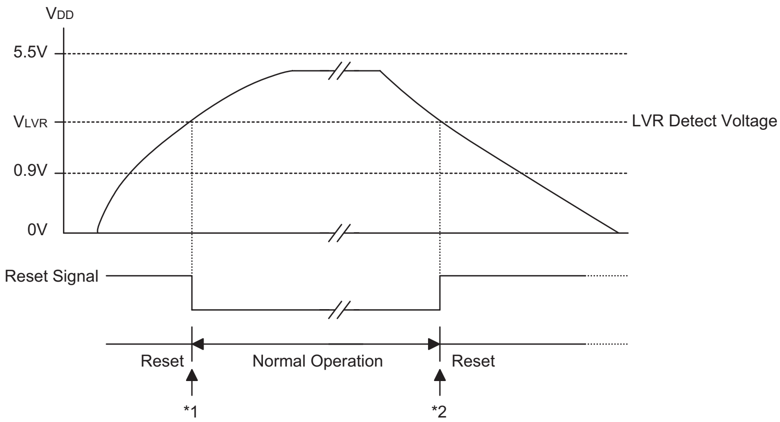
The LVR includes the following specifications:

- 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.



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.

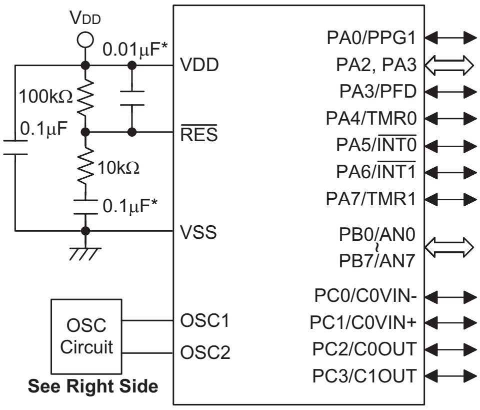
*2: Since low voltage state has to be maintained in its original state for over 1ms, therefore after 1ms delay, the device enters the reset mode.

Options

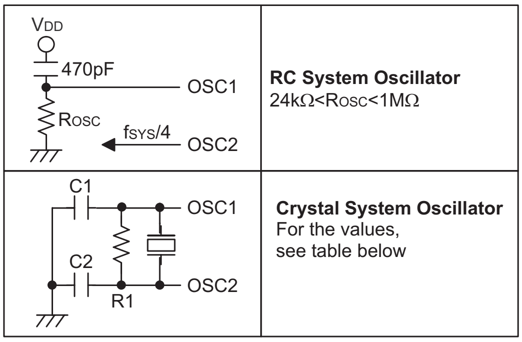
The following table shows all kinds of options in the microcontroller. All of the options must be defined to ensure having a proper functioning system.

Options
OSC type selection. This option is to determine whether an RC or crystal oscillator is chosen as system clock.
WDT clock source selection. WDT oscillator or fsys/4.
WDT enable/disable selection. WDT can be enabled or disabled by option.
WDT time-out period selection. There are four types of selection: $f_S/2^{13}$, $f_S/2^{14}$, $f_S/2^{15}$ and $f_S/2^{16}$
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.
Pull-high selection. This option is to determine whether a pull-high resistance is viable or not in the input mode of the I/O ports. PA0~PA7, can be independently selected.
Pull-high selection. This option is to determine whether a pull-high resistance is viable or not in the input mode of the I/O ports. PB0~PB7, can be independently selected.
This option is to determine whether a pull-high resistance is viable or not in the input mode of the I/O ports. PC0~PC3 byte option.
I/O pins share with other function selections. PA0/PPG1: PA0 can be set as I/O pins or PPG1 output.
I/O pins share with other function selections. PA3/PFD: PA3 can be set as I/O pins or PFD output.
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 generated by the timer overflow signals of the Timer/Event Counter 0, Timer/Event Counter 1 respectively.
Low voltage reset selection. Enable or disable the LVR function.
PPG0 output level selection; P0LEV. This option is to determine the PPG0 output level. Active Low or Active High selection. Disable this bit to ″0″, the PPG0 output will be defined as an active high output, Enable this bit to ″1″, the PPG0 output will be defined as an active low output.
PPG1 output level selection; P1LEV. This option is to determine the PPG1 output level. Active Low or Active High selection. Disable this bit to ″0″, the PPG1 output will be defined as an active high output, Enable this bit to ″1″, the PPG1 output will be defined as an active low output.
PPG0 timer start counting synchronized with clock; P0TSYN. This option is to determine whether the PPG0 timer start counting is synchronized with input clock or not.
PPG1 timer start counting synchronized with clock; P1TSYN. This option is to determine the PPG1 timer start counting is synchronized with input clock or not.

Application Circuits



HT46R14



OSC Circuit

The following table shows the C1, C2 and R1 values corresponding to the different crystal values. (For reference only)

Crystal or Resonator	C1, C2	R1
4MHz Crystal	0pF	10kΩ
4MHz Resonator	10pF	12kΩ
3.58MHz Crystal	0pF	10kΩ
3.58MHz Resonator	25pF	10kΩ
2MHz Crystal & Resonator	25pF	10kΩ
1MHz Crystal	35pF	27kΩ
480kHz Resonator	300pF	9.1kΩ
455kHz Resonator	300pF	10kΩ
429kHz Resonator	300pF	10kΩ
The function of the resistor R1 is to ensure that the oscillator will switch off should low voltage conditions occur. Such a low voltage, as mentioned here, is one which is less than the lowest value of the MCU operating voltage. Note however that if the LVR is enabled then R1 can be removed.		

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 $\overline{\text{RES}}$ to high.
"*" Make the length of the wiring, which is connected to the $\overline{\text{RES}}$ 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 ← Program Counter+1
Program Counter ← addr

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

CLR [m] Clear data memory

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

Operation: [m] ← 00H

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

CLR [m].i Clear bit of data memory

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

Operation [m].i ← 0

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

CLR WDT Clear Watchdog Timer

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

Operation WDT ← 00H
PDF and TO ← 0

Affected flag(s)

TO	PDF	OV	Z	AC	C
0	0	—	—	—	—

CLR WDT1 Preclear Watchdog Timer

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

Operation WDT ← 00H*
PDF and TO ← 0*

Affected flag(s)

TO	PDF	OV	Z	AC	C
0*	0*	—	—	—	—

CLR WDT2 Preclear Watchdog Timer

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

Operation WDT ← 00H*
PDF and TO ← 0*

Affected flag(s)

TO	PDF	OV	Z	AC	C
0*	0*	—	—	—	—

CPL [m] Complement data memory

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

Operation [m] ← $[\overline{m}]$

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

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

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

Operation: ACC ← $\overline{[m]}$

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

DAA [m] Decimal-Adjust accumulator for addition

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

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	√

DEC [m] Decrement data memory

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

Operation: [m] ← [m]−1

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

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

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

Operation: ACC ← [m]−1

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

HALT Enter power down mode

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

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
0	1	—	—	—	—

INC [m] Increment data memory

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

Operation: [m] ← [m]+1

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

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

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

Operation: ACC ← [m]+1

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

JMP addr Directly jump

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

Operation: Program Counter ← addr

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

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

Operation: ACC ← [m]

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

MOV A,x Move immediate data to the accumulator

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

Operation: ACC ← x

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

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

Operation: [m] ← ACC

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

NOP No operation

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

Operation: Program Counter ← Program Counter+1

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

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

Operation: ACC ← ACC ″OR″ [m]

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

OR A,x Logical OR immediate data to the accumulator

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

Operation: ACC ← ACC ″OR″ x

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

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

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

Operation: [m] ← ACC ″OR″ [m]

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

RET Return from subroutine

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

Operation: Program Counter ← Stack

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

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

Operation: Program Counter ← Stack
ACC ← x

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

RETI Return from interrupt

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

Operation: Program Counter ← Stack
EMI ← 1

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

RL [m] Rotate data memory left

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

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

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

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

RLC [m] Rotate data memory left through carry

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

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	√

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

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

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	√

RR [m] Rotate data memory right

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

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

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

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

RRC [m] Rotate data memory right through carry

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

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	√

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

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

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	√

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

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

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	√	√	√	√

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

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

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	√	√	√	√

SDZ [m] Skip if decrement data memory is 0

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

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

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

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

SET [m] Set data memory

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

Operation: [m] ← FFH

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

SET [m]. i Set bit of data memory

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

Operation: [m].i ← 1

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

SIZ [m] Skip if increment data memory is 0

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

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

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

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

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

Operation: Skip if [m].i≠0

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

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

Operation: ACC ← 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.

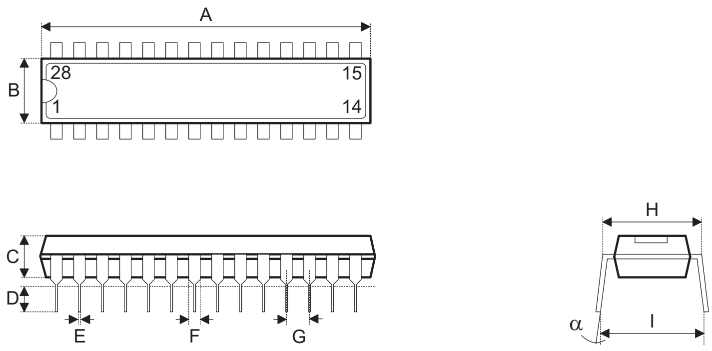
Operation: ACC ← ACC ″XOR″ x

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

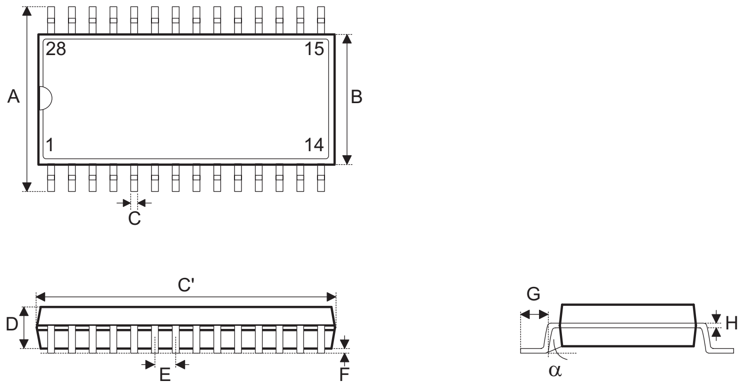
Package Information

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°

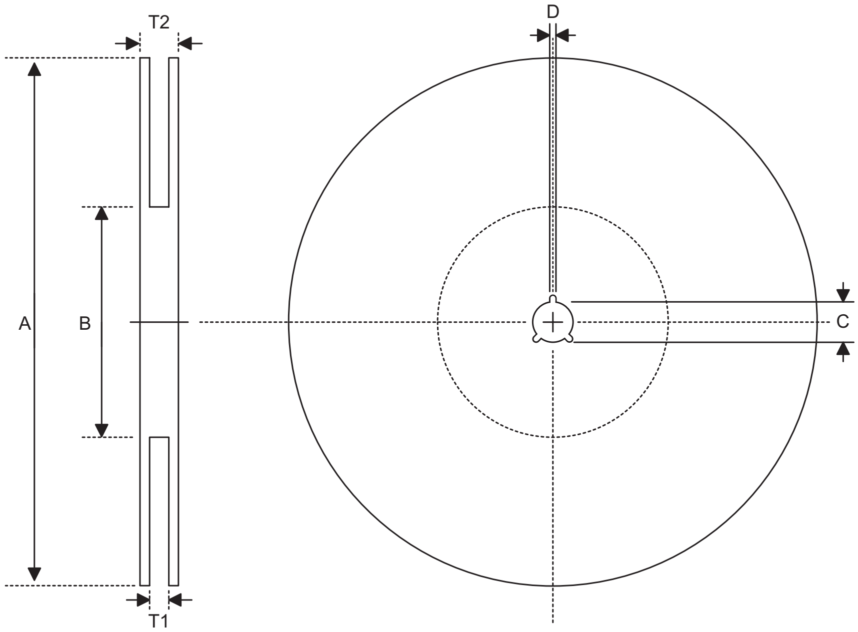
28-pin SOP (300mil) Outline Dimensions



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°

Product Tape and Reel Specifications

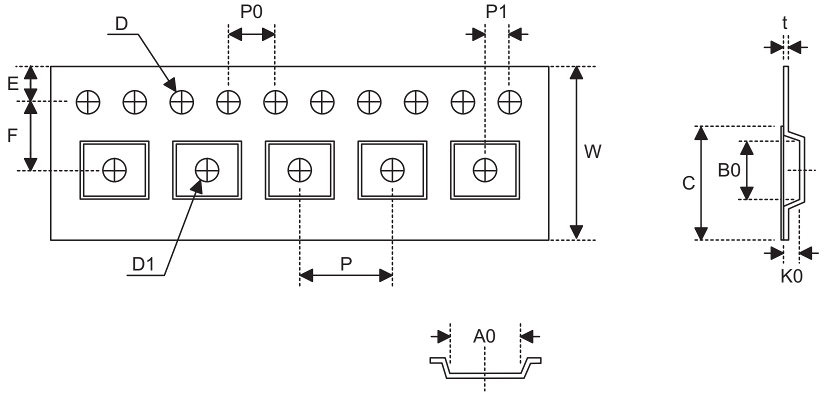
Reel Dimensions



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

Carrier Tape Dimensions



SOP 28W

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

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