HT45RM03

Brushless DC Motor Type 8-Bit OTP MCU

Technical Document

- Tools Information
- FAQs
- Application Note

Features

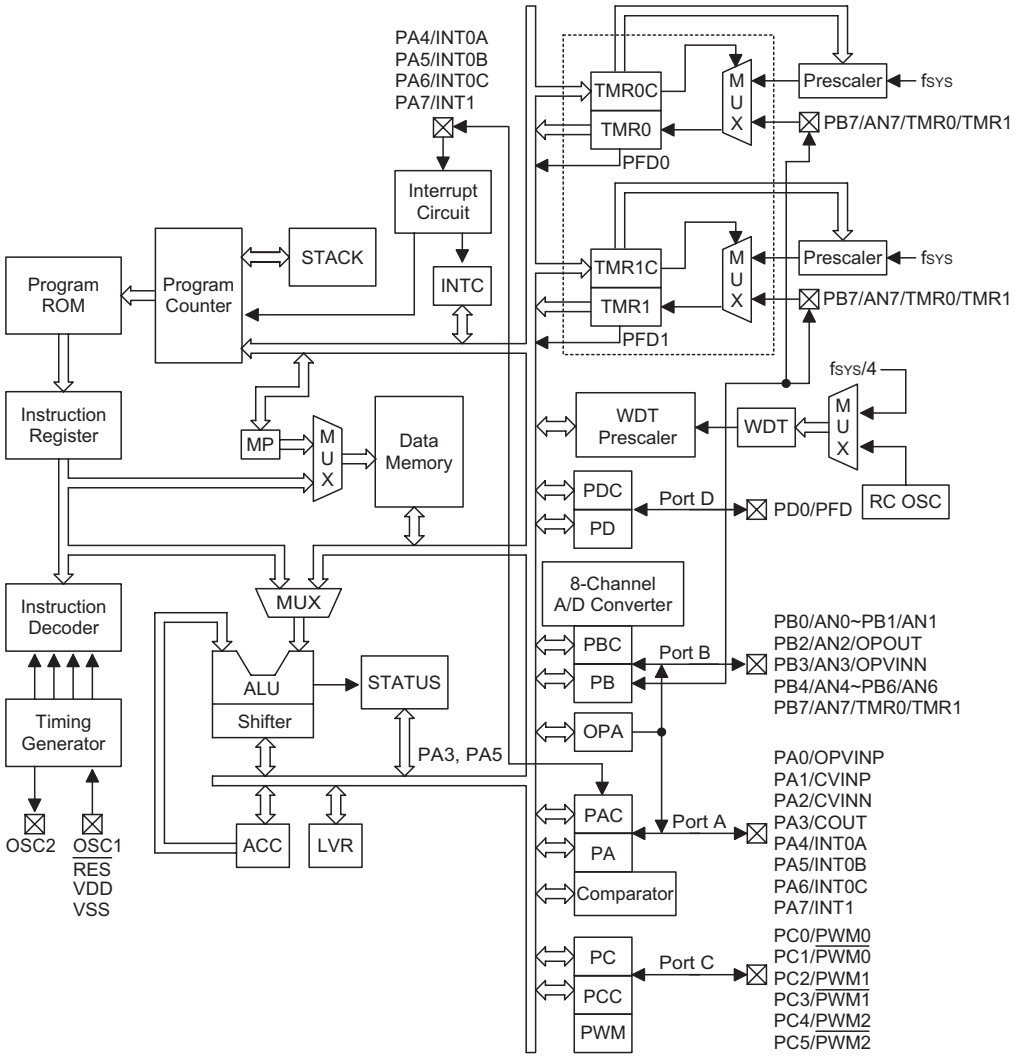
- Operating voltage:
  f_{SYS}=4MHz: 2.2V~5.5V
  f_{SYS}=8MHz: 3.3V~5.5V
  f_{SYS}=12MHz: 4.5V~5.5V
- 23 bidirectional I/O lines (max.)
- 4 interrupt input shared with 4 I/O line
- Two 8-bit programmable timer/event counter with overflow interrupt and 7-stage prescaler
- On-chip crystal and RC oscillator
- 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 0.33μs instruction cycle with 12MHz system clock at V_{DD}=5V
- 8-level subroutine nesting
- 8 channels 9-bit resolution A/D converter
- 3-channel 10-bit PWM with complementary output shared with six I/O lines
- Bit manipulation instruction
- 15-bit table read instruction
- 63 powerful instructions
- All instructions in one or two machine cycles
- Low voltage reset function
- One operational Amplifier
- One Comparator with interrupt function
- 28-pin SKDIP/SOP packages

General Description

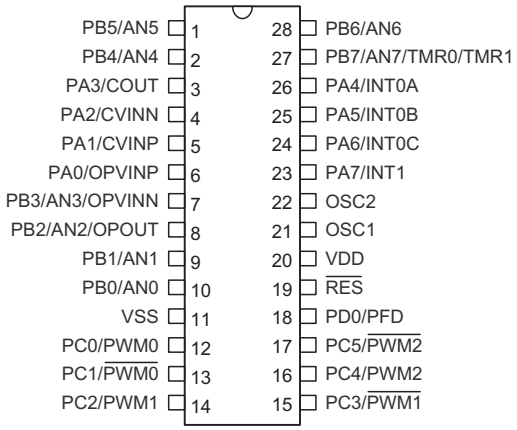
The HT45RM03 is 8-bit, high performance, RISC architecture microcontroller devices specifically designed for A/D applications that interface directly to analog signals, such as those from sensors.

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

Block Diagram



Pin Assignment



HT45RM03
— 28 SKDIP-A/SOP-A

Pad Description

Pad Name	I/O	Option	Description
PA0/OPVINP PA1/CVINP PA2/CVINN PA3/COUT PA4/INT0A PA5/INT0B PA6/INT0C PA7/INT1	I/O	Wake-up Pull-high	Bidirectional 8-bit input/output port. Each bit can be configured as wake-up input by ROM code option. Software instructions determine the CMOS output or Schmitt trigger input with or without pull-high resistor (determined by pull-high options: bit option). The INT0A, INT0B, INT0C and INT1 are pin-shared with PA4~PA7. The CVINP, CVINN and COUT are pin-shared with PA1, PA2 and PA3. Once the Comparator function is used, the internal registers related to PA1, PA2 cannot be used, PA3 can be used as input only, and the PA1, PA2 I/O function and pull-high resistor are disabled automatically. Software instructions determine the Comparator function to be used. The OPVINP is pin-shared with PA0.
PB0/AN0 PB1/AN1 PB2/AN2/OPOUT PB3/AN3/OPVINN PB4/AN4~PB6/AN6 PB7/AN7/TMR0/TMR1	I/O	Pull-high	Bidirectional 8-bit input/output port. Software instructions determine the 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. The OPVINP, OPVINN and OPOUT are pin-shared with PA0, PB3/AN3 and PB2/AN2 respectively. Once the OPA function is used, the internal registers related to PA0, PB3 and PB2 cannot be used, and the I/O function and pull-high resistor are disabled automatically. Software instructions determine the OPA function to be used.
PC0/$\overline{\text{PWM0}}$ PC1/$\overline{\text{PWM0}}$ PC2/$\overline{\text{PWM1}}$ PC3/$\overline{\text{PWM1}}$ PC4/$\overline{\text{PWM2}}$ PC5/$\overline{\text{PWM2}}$	I/O	Pull-high	Bidirectional 6-bit input/output port. Software instructions determine the CMOS output, Schmitt trigger input with or without a pull-high resistor (determined by pull-high option: byte option). The PWM0~PWM2 and $\overline{\text{PWM0}}$~$\overline{\text{PWM2}}$ output function are pin-shared with PC0, PC2, PC4 and PC1, PC3, PC5 respectively by software control.
PD0/PFD	I/O	Pull-high PFD	Bidirectional 1-bit input/output port. Software instructions determine the CMOS output, Schmitt trigger input with or without a pull-high resistor (determined by pull-high option: bit option). The PFD output function is pin-shared with PD0.
$\overline{\text{RES}}$	I	—	Schmitt trigger reset input. Active low.
VDD	—	—	Positive power supply
VSS	—	—	Negative power supply, ground.
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.

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

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

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

D.C. Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	Operating Voltage	—	f_{SYS}=4MHz	2.2	—	5.5	V
		—	f_{SYS}=8MHz	3.3	—	5.5	V
		—	f_{SYS}=12MHz	4.5	—	5.5	V
I_{DD1}	Operating Current (Crystal OSC, RC OSC)	3V	No load, f_{SYS}=4MHz ADC disable	—	1	2	mA
		5V		—	2.5	5	mA
I_{DD2}	Operating Current (Crystal OSC, RC OSC)	3V	No load, f_{SYS}=8MHz ADC disable	—	2	4	mA
		5V		—	4	8	mA
I_{DD3}	Operating Current (Crystal OSC, RC OSC)	5V	No load, f_{SYS}=12MHz ADC disable	—	5	10	mA
I_{STB1}	Standby Current (WDT Enabled)	3V	No load, system HALT	—	—	5	μA
		5V		—	—	10	μA
I_{STB2}	Standby Current (WDT Disabled)	3V	No load, system HALT	—	—	1	μA
		5V		—	—	2	μA
V_{IL1}	Input Low Voltage for I/O Ports, TMR0, TMR1, INT0A, INT0B, INT0C and INT1	—	—	0	—	$0.3V_{DD}$	V
V_{IH1}	Input High Voltage for I/O Ports, TMR0, TMR1, INT0A, INT0B, INT0C and INT1	—	—	$0.7V_{DD}$	—	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_{LVR1}	Low Voltage Reset 1	—	Configuration option: 4.2V	3.98	4.2	4.42	V
V_{LVR2}	Low Voltage Reset 2	—	Configuration option: 3.15V	2.98	3.15	3.32	V
V_{LVR3}	Low Voltage Reset 3	—	Configuration option: 2.1V	1.98	2.1	2.22	V
I_{OL}	I/O Port Sink Current	3V	V_{OL}=0.1V_{DD}	4	8	—	mA
		5V	V_{OL}=0.1V_{DD}	10	20	—	mA
I_{OH}	I/O Port Source Current	3V	V_{OH}=0.9V_{DD}	−2	−4	—	mA
		5V	V_{OH}=0.9V_{DD}	−5	−10	—	mA
R_{PH}	Pull-high Resistance	3V	—	20	60	100	kΩ
		5V	—	10	30	50	kΩ
V_{AD}	A/D Input Voltage	—	—	0	—	V_{DD}	V
E_{AD}	A/D Conversion Error	—	—	—	±0.5	±1	LSB
I_{ADC}	Additional Power Consumption if A/D Converter is Used	3V	—	—	0.5	1	mA
		5V		—	1.5	3	mA

A.C. Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
f_{SYS}	System Clock	—	2.2V~5.5V	400	—	4000	kHz
		—	3.3V~5.5V	400	—	8000	kHz
		—	4.5V~5.5V	400	—	12000	kHz
f_{TIMER}	Timer I/P Frequency (TMR0/TMR1)	—	—	0	—	4000	kHz
t_{WDTOSC}	Watchdog Oscillator Period	3V	—	45	90	180	μs
		5V	—	32	65	130	μs
t_{RES}	External Reset Low Pulse Width	—	—	1	—	—	μs
t_{SST}	System Start-up Timer Period	—	Wake-up from HALT	—	1024	—	*t_{SYS}
t_{INT}	Interrupt Pulse Width	—	—	1	—	—	μs
t_{AD}	A/D Clock Period	—	—	1	—	—	μs
t_{ADC}	A/D Conversion Time	—	—	—	72	—	t_{AD}
t_{ADCS}	A/D Sampling Time	—	—	—	32	—	t_{AD}
t_{LVR}	Low Voltage Width to Reset	—	—	0.25	1	2	ms

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

OP Amplifier Electrical Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
D.C. Electrical Characteristic							
V_{DD}	Operating Voltage	—	—	3	—	5.5	V
V_{OS}	Input Offset Voltage	5V	By calibration	–2	—	+2	mV
V_{CM}	Common Mode Voltage Range	—	—	V_{SS}	—	V_{DD}–1.4	V
PSRR	Power Supply Rejection Ratio	—	—	60	—	—	dB
CMRR	Common Mode Rejection Ratio	—	V_{DD}=5V V_{CM}=0~V_{DD}–1.4V	60	—	—	dB
A.C. Electrical Characteristic							
A_{OL}	Open Loop Gain	—	—	60	80	—	dB
SR	Slew Rate+, Rate–	—	No load	—	1	—	V/μs
GBW	Gain Band Width	—	RL=1MΩ, CL=100pF	—	—	100	kHz

Comparator Electrical Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	Operating Voltage	—	—	3	—	5.5	V
V_{OS}	Comparator Input Offset Voltage	5V	By calibration	–2	—	2	mV
V_{CM}	Comparator Common Mode Voltage Range	—	—	0	—	V_{DD}–1.4	V
t_{PD}	Comparator Response Time	—	—	—	—	2	μs

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.

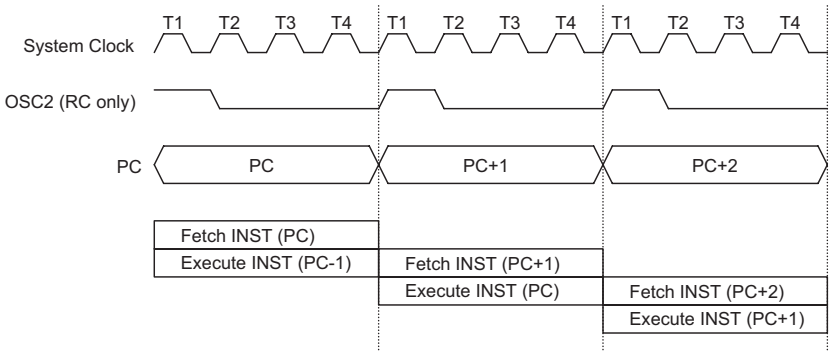
After accessing a program memory word to fetch an instruction code, the contents of the program counter are incremented by one. The program counter then points to the memory word containing the next instruction code.

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.

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.



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
Comparator Interrupt	0	0	0	0	0	0	0	0	0	1	0	0
External Interrupt 0	0	0	0	0	0	0	0	0	1	0	0	0
External Interrupt 1	0	0	0	0	0	0	0	0	1	1	0	0
PWM Period Interrupt	0	0	0	0	0	0	0	1	0	0	0	0
Timer/Event Counter 0 Overflow	0	0	0	0	0	0	0	1	0	1	0	0
Timer/Event Counter 1 Overflow	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

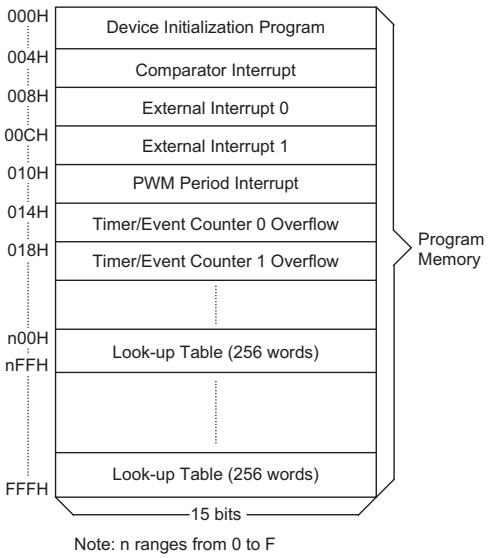
Note: *11~*0: Program counter bits
#11~#0: Instruction code bits
S11~S0: Stack register bits
@7~@0: PCL bits

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
  These area is reserved for program initialization. After chip reset, the program always begins execution at location 000H.
- Location 004H
  These area is reserved for the Comparator interrupt service program. If the Comparator output pin is activated, and if the interrupt is enable and the stack is not full, the program begins execution at location 004H.
- Location 008H
  These area is reserved for the external interrupt 0 service program. If the INT0A, INT0B or INT0C input pin is activated, the interrupt is enabled and the stack is not full, the program begins execution at location 008H.
- Location 00CH
  These area is reserved for the external interrupt 1 service program. If the INT1 input pin is activated, the interrupt is enabled and the stack is not full, the program begins execution at location 00CH.
- Location 010H
  These area is reserved for the PWM period interrupt service program. If a PWM period interrupt results from a PWM counter overflow, and if the interrupt is enabled and the stack is not full, the program begins execution at location 010H.
- Location 014H
  These area 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 014H.
- Location 018H
  These area 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 018H.



Program Memory

- 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.

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
P11~P8: Current program counter bits
@7~@0: Table pointer bits

Stack Register – STACK

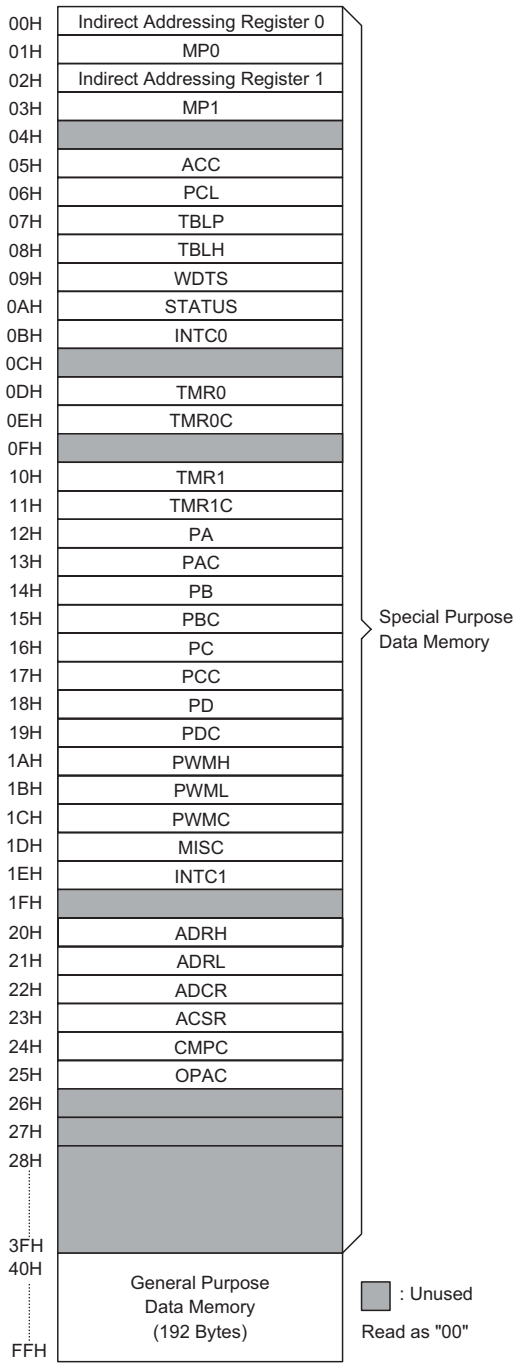
This is a special part of the memory which is used to save the contents of the Program Counter only. The stack is organized into 8 levels and is neither part of the data nor part of the program space, and is neither readable nor writeable. The activated level is indexed by the stack pointer (SP) and 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 226×8 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 include the indirect addressing registers (00H;02H), Timer/Event Counter 0 (TMR0;0DH), Timer/Event Counter 0 control register (TMR0C;0EH), Timer/Event Counter 1 (TMR1:10H), Timer/Event Counter 1 control register (TMR1C; 11H), program counter lower-order byte register (PCL;06H), memory pointer registers (MP0;01H, MP1;03H), accumulator (ACC;05H), table pointer (TBLP;07H), table higher-order byte register (TBLH;08H), status register (STATUS;0AH), interrupt control register 0 (INTC0; 0BH), PWM higher-order byte register (PWMH;1AH), PWM lower-order byte register (PWML;1BH), PWM Control register (PWMC;1CH), Miscellaneous register (MISC;1DH), the A/D result lower-order byte register (ADRL;20H), the A/D result higher-order byte register (ADRH;21H), the A/D control register (ADCR;22H), the A/D clock setting register (ACSR;23H), the Comparator Control register (CMPC;24H), the Operational Amplifier Control register (OPAC;25H), I/O registers (PA;12H, PB;14H, PC;16H, PD;18H) and I/O control registers (PAC;13H, PBC;15H, PCC;17H, PDC;19H). The remaining space before the 26H or 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.



RAM Mapping

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

The method of indirect addressing allows data manipulation using memory pointers instead of the usual direct memory addressing method where the actual memory address is defined. Any action on the indirect addressing registers will result in corresponding read/write operations to the memory location specified by the corresponding memory pointers. This device contains two indirect addressing registers known as IAR0 and IAR1 and two memory pointers MP0 and MP1. Note that these indirect addressing registers are not physically implemented and that reading the indirect addressing registers indirectly will return a result of "00H" and writing to the registers indirectly will result in no operation.

The two memory pointers, MP0 and MP1, are physically implemented in the data memory and can be manipulated in the same way as normal registers providing a convenient way with which to address and track data. When any operation to the relevant indirect addressing registers is carried out, the actual address that the microcontroller is directed to is the address specified by the related memory pointer.

Direct data transfer between two indirect addressing registers is not supported. The memory pointer registers, MP0 and MP1, are both 8-bit registers used to access the Program Memory 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 (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.

Bit No.	Label	Function
0	C	C is set if the 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 the 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 the 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 system power-up or executing the "CLR WDT" or "HALT" instruction. TO is set by a WDT time-out.
6~7	—	Unused bit, read as "0"

Status (0AH) Register

Interrupt

The devices provides four external interrupts, two internal Timer/Event Counter 0/1 interrupts, one comparator interrupt, and PWM period 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, all the other interrupts will be blocked (by clearing the EMI bit). This scheme may prevent any further interrupt nesting. Other interrupt requests may happen during this interval but only the interrupt request flag is recorded. If a certain interrupt requires servicing within the service routine, the EMI bit and the corresponding bit of INTC0 and INTC1 may be set to allow interrupt nesting. If the stack is full, the interrupt request will not be acknowledged, even if the related interrupt is enabled, until the SP is decremented. If immediate service is desired, the stack must be prevented from becoming full.

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

The Comparator output Interrupt is initialized by setting the Comparator 0 output Interrupt request flag (CF; bit 4 of the INTC0), which is caused by a falling edge transition of comparator output. After the interrupt is enabled, and the stack is not full, and the CF bit is set, a subroutine call to location 04H occurs. The related interrupt request flag (CF) is reset, and the EMI bit is cleared to disable further maskable interrupts.

External interrupts are triggered by a an edge transition of INT0A, INT0B, INT0C or INT1 (software control: high to low, low to high, low to high or high to low), and the related interrupt request flag (EIF0; bit 5 of INTC0, EIF1; bit 6 of 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 08H or 0CH occurs. The interrupt request flag (EIF0 or EIF1) and EMI bits are all cleared to disable other maskable interrupts.

The PWM period interrupt is initialized by setting the PWM period interrupt request flag (PWMF; bit 4 of INTC1), that is caused by a regular PWM period signal. After the interrupt is enabled, and the stack is not full, and the PWMF bit is set, a subroutine call to location 10H occurs. The related interrupt request flag (PWMF) 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 5 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 014H 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 6 of the INTC1) and its subroutine call location is 018H. The related interrupt request flag (T1F) will be reset and the EMI bit cleared to disable further interrupts.

Bit No.	Label	Function
0	EMI	Control the master (global) interrupt (1=enabled; 0=disabled)
1	ECI	Control the Comparator interrupt (1=enabled; 0=disabled)
2	EEI0	Control the external INT0A, INT0B, INT0C interrupt(1=enabled; 0=disabled)
3	EEI1	Control the external INT1 interrupt (1=enabled; 0=disabled)
4	CF	The Comparator request flag (1=active; 0=inactive)
5	EI0F	External interrupt INT0A, INT0B, INT0C request flag (1=active; 0=inactive)
6	EI1F	External interrupt INT1 request flag (1=active; 0=inactive)
7	—	Unused bit, read as ″0″.

INTC0 (0BH) Register

Bit No.	Label	Function
0	EPWMI	Control the PWM period interrupt (1=enabled; 0=disabled)
1	ET0I	Control the Timer/Event Counter 0 interrupt (1=enabled; 0=disabled)
2	ET1I	Control the Timer/Event Counter 1 interrupt (1=enabled; 0=disabled)
3, 7	—	Unused bit, read as ″0″.
4	PWMF	PWM period request flag (1=active; 0=inactive)
5	T0F	Internal Timer/Event Counter 0 request flag (1=active; 0=inactive)
6	T1F	Internal Timer/Event Counter 1 request flag (1=active; 0=inactive)

INTC1 (1EH) Register

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

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

Interrupt Source	Priority	Vector
Comparator output Interrupt	1	04H
External INT0A, INT0B, INT0C Interrupt	2	08H
External INT1 Interrupt	3	0CH
PWM Period Interrupt	4	10H
Timer/Event Counter 0 Overflow	5	14H
Timer/Event Counter 1 Overflow	6	18H

The Comparator interrupt request flag (CF), 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.

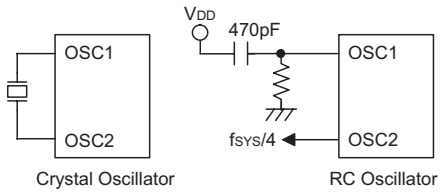
The PWM period interrupt request flag (PWMF), the Timer/Event Counter 1 interrupt request flag (T1F), the Timer/Event Counter 0 interrupt request flag (T0F), enable PWM period interrupt bit (EPWMI), enable Timer/Event Counter 1 interrupt bit (ET1I), and enable Timer/Event Counter 0 interrupt bit (ET0I), constitute the Interrupt Control register 1 (INTC1) which is located at 1EH in the RAM.

EMI, EEI0, EEI1, ECI, ET0I, ET1I, and EPWMI 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, CF, T0F, T1F, PWMF) 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 with pull-high resistor, 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 resona-

tor can also be connected between OSC1 and OSC2 to get a frequency reference, but two external capacitors in OSC1 and OSC2 are required (If the oscillating frequency is less than 1MHz).

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

Watchdog Timer – WDT

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

Once the internal WDT oscillator (RC oscillator with a period of 65μs at 5V normally) is selected, it is first divided by 256 (8-stage) to get the nominal time-out period of approximately 17ms at 5V. This time-out period may vary with temperatures, VDD and process variations. By invoking the WDT prescaler, longer time-out periods can be realized. Writing data to WS2, WS1, WS0 (bit 2,1,0 of the WDTS) can give different time-out periods. If WS2, WS1, and WS0 are all equal to 1, the division ratio is up to 1:128, and the maximum time-out period is 2.1s at 5V seconds. If the WDT oscillator is disabled, the WDT clock may still come from the instruction clock and operate in the same manner except that in the HALT state the WDT may stop counting and lose its protecting purpose. In this situation the logic can only be restarted by external logic. The high nibble and bit 3 of the WDTS are reserved for user's defined flags, which can be used to indicate some specified status.

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.

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

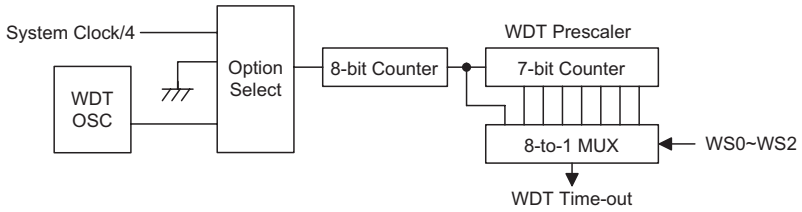
WDTS (09H) Register

The WDT overflow under normal operation will initialize "chip reset" and set the status bit "TO". But in the HALT mode, the overflow will initialize a "warm reset", and only the Program Counter and SP are reset to zero. To clear the contents of WDT (including the WDT prescaler), three methods are adopted; external reset (a low level to RES), software instruction and a "HALT" instruction. The software instruction 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 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 the case that "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 as a result of time-out.

Power Down Operation – HALT

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

- The system oscillator 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).



Watchdog Timer

- All of the I/O ports maintain their original status.
- The PDF 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 PDF flags are examined, the reason for chip reset can be determined. The PDF 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 SP; 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 options. Awakening from an I/O port stimulus, the program will resume execution of the next instruction. If it is awakening from an interrupt, two sequences may happen. 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

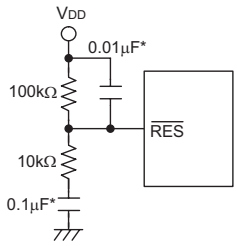
The WDT time-out during HALT is different from other chip reset conditions, since it can perform a "warm reset" that resets only the program counter and SP, leaving the other circuits in their original state. Some registers remain unchanged during other reset conditions. Most registers are reset to the "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"

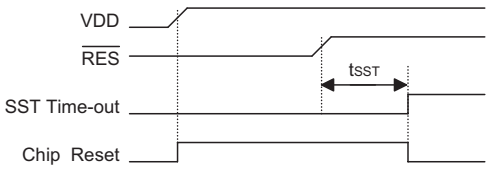
To guarantee that the system oscillator is started and stabilized, the SST (System Start-up Timer) provides an extra-delay of 1024 system clock pulses when the system reset (power-up, WDT time-out or $\overline{RES}$ reset) or the system awakes from the HALT state.

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.

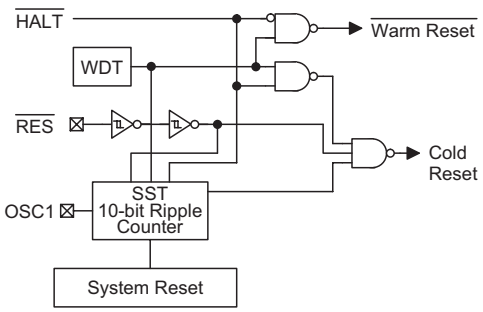


Reset Circuit

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



Reset Timing Chart



Reset Configuration

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
WDT	Clear. After master reset, WDT begins counting
Timer/Event Counter	Off
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	-xxx xxxx	-uuu uuuu	-uuu uuuu	-uuu uuuu	-uuu uuuu
STATUS	--00 xxxx	--1u uuuu	--uu uuuu	--01 uuuu	--11 uuuu
WDTS	0000 0111	0000 0111	0000 0111	0000 0111	uuuu uuuu
INTC0	-000 0000	-000 0000	-000 0000	-000 0000	-uuu uuuu
TMR0	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
TMR0C	00-0 1000	00-0 1000	00-0 1000	00-0 1000	uu-u uuuu
TMR1	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
TMR1C	00-0 1000	00-0 1000	00-0 1000	00-0 1000	uu-u uuuu
PA	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PB	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PBC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PC	0011 1111	0011 1111	0011 1111	0011 1111	uuuu uuuu
PCC	--11 1111	--11 1111	--11 1111	--11 1111	--uu uuuu
PD	---- ---1	---- ---1	---- ---1	---- ---1	---- ---u
PDC	---- ---1	---- ---1	---- ---1	---- ---1	---- ---u
PWMH	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PWML	---- --00	---- --00	---- --00	---- --00	---- --uu
PWMC	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
MISC	-000 0000	-000 0000	-000 0000	-000 0000	-uuu uuuu
INTC1	-000 -000	-000 -000	-000 -000	-000 -000	-uuu -uuu
ADRH	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
ADRL	x--- ----	x--- ----	x--- ----	x--- ----	u--- ----
ADCR	0100 0000	0100 0000	0100 0000	0100 0000	uuuu uuuu
ACSR	---- --00	---- --00	---- --00	---- --00	---- --uu
CMPC	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
OPAC	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu

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

Timer/Event Counter

Two timer/event counters (TMR0,TMR1) are implemented in the microcontroller. The Timer/Event Counter 0 and Timer/Event Counter 1 contain 8-bit programmable count-up counter and the clock may come from an external source or an internal clock source.

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.

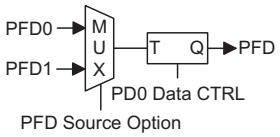
In the event count or timer mode, the Timer/Event Counter 0/1 starts counting at the current contents in the timer/event counter and ends at FFH. Once an overflow occurs, the counter is reloaded from the timer/event counter preload register, and generates an interrupt request flag (T0F; bit 5 of the INTC1, T1F; bit 6 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.

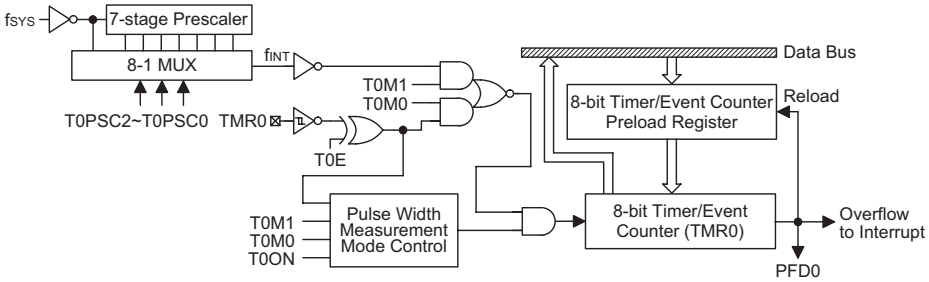
The measured result remains in the timer/event counter even if the activated transient occurs again. In other words, only 1-cycle measurement can be made until the T0ON/T1ON is set. The cycle measurement will re-function as long as it receives further transient pulse. In this operation mode, the timer/event 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.

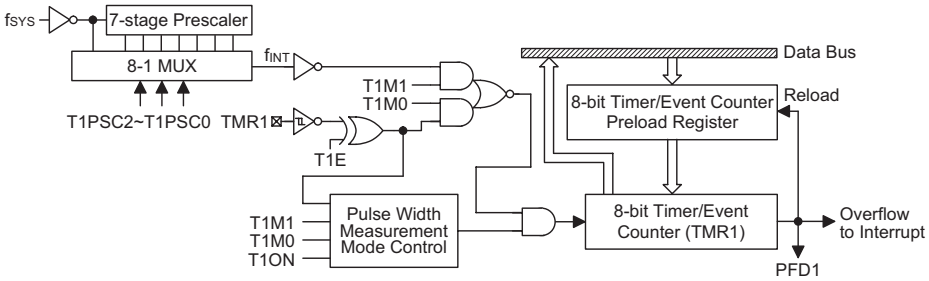
The overflow of the Timer/Event Counter 0/1 is one of the wake-up sources and the Timer/Event Counter 0/1 can also be applied to a PFD (Programmable Frequency Divider) output at PA3 by software control. Only one PFD (PFD0 or PFD1) can be applied to PA3 by software. No



PFD Source Option



Timer/Event Counter 0



Timer/Event Counter 1

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.

After this procedure, the timer/event function can be operated normally. The bit0~bit2 of the TMR0C/TMR1C 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.

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.

Bit No.	Label	Function
0~2	T0PSC0~ T0PSC2	To 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	To define the TMR0 active edge of Timer/Event Counter 0 (0=active on low to high; 1=active on high to low)
4	T0ON	To enable or disable Timer 0 counting (0=disabled; 1=enabled)
5	—	Unused bit, read as "0"
6 7	T0M0 T0M1	To define the operating mode, T0M1, T0M0= 01=Event count mode (external clock) 10=Timer mode (internal clock) 11=Pulse width measurement mode 00=Unused

TMR0C (0EH) Register

Bit No.	Label	Function
0~2	T1PSC0~ T1PSC2	To define the prescaler stages, T1PSC2, T1PSC1, T1PSC0= 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	T1E	To define the TMR1 active edge of Timer/Event Counter 1 (0=active on low to high; 1=active on high to low)
4	T1ON	To enable or disable Timer 1 counting (0=disabled; 1=enabled)
5	—	Unused bit, read as"0"
6 7	T1M0 T1M1	To define the operating mode, T1M1, T1M0= 01=Event count mode (external clock) 10=Timer mode (internal clock) 11=Pulse width measurement mode 00=Unused

TMR1C (11H) Register

Input/Output Ports

There are 23 bidirectional input/output lines in the microcontroller, labeled as PA, PB, PC and PD, which are mapped to the data memory of [12H], [14H], [16H] and [18H] respectively. All of these I/O ports can be used for input and output operations. For input operation, these ports are non-latching, that is, the inputs must be ready at the T2 rising edge of instruction "MOV A,[m]" (m=12H, 14H, 16H or 18H). For output operation, all the data is latched and remains unchanged until the output latch is rewritten.

Each I/O line has its own control register (PAC, PBC, PCC, PDC) 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 (i.e. on-the-fly) under software control. To function as an input, the corresponding latch of the control register must write "1". The input source also depends on the control register. If the control register bit is "1", the input will read the pad state. If the control register bit is "0", the contents of the latches will move to the internal bus. The latter is possible in the "read-modify-write" instruction.

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

After a chip reset, these input/output lines remain at high levels or floating state (dependent on pull-high options). Each bit of these input/output latches can be set or cleared by "SET [m].i" and "CLR [m].i" (m=12H, 14H, 16H or 18H) 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. The highest 7-bit of port D are not physically implemented; on reading them a "0" is returned whereas writing then results in a no-operation. See Application note.

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 PD0 is pin-shared with the PFD signal. If the PFD function is selected, the output signal in output mode of PD0 will be the PFD signal generated by the timer/event counter overflow signal. The input mode is always remaining its original functions. Once the PFD option is selected, the PFD output signal is controlled by PD0 data register only. Writing "1" to PD0 data register will enable the PFD output function and writing "0" will force the PD0 to remain at "0". The I/O functions of PD0 are shown below.

I/O Mode	I/P (Normal)	O/P (Normal)	I/P (PFD)	O/P (PFD)
PD0	Logical Input	Logical Output	Logical Input	PFD (Timer on)

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

The PA4, PA5, PA6, PA7 and PB7 are pin-shared with INT0A, INT0B, INT0C, INT1 and TMR0/TMR1 pins respectively. The CVINN and COUT are pin-shared with PA1, PA2 and PA3. Once the Comparator function is used, the internal registers related to PA1, PA2 cannot be used, PA3 can be used as input only (if the COUTEN is "1"), and the PA1, PA2 I/O function, PA3 output function and pull-high resistor are disabled automatically(if the COUTEN is "1"). Once the Comparator function is used, the PA3 is the GPIO when the COUTEN is "0". Software instructions determine the Comparator function to be used.

CMPEN	COUTEN	PA1, PA2, PA3
0	X	PA1, PA2, PA3 is GPIO
1	0	PA1, PA2 is Comparator input pin and PA3 is GPIO. PA3 falling edge can generate the Comparator interrupt.
1	1	PA1, PA2 is comparator input pin and PA3 is Comparator output pin. PA3 can read the Comparateor output status.

The OPVINP, OPVINN and OPOUT are pin-shared with PA0, PB3/AN3 and PB2/AN2 respectively. Once the OPA function is used, the internal registers related to PA0, PB3 and PB2 cannot be used, and the I/O function and pull-high resistor are disabled automatically. Software instructions determine the OPA function to be used.

OPAEN	PA0, PA3/AN3, PB2/AN2
0	PA0 is GPIO and PB2/AN2, PB3/AN3 is GPIO or analog ADC input by ADCR register.
1	PA0, PB3/AN3 is OPA input pin and PB2/AN2 is OPA output pin. The PB2/AN2 and PB3/AN3 can be analog ADC input if the related ADC function is enabled.

The PB can also be used as A/D converter inputs. The A/D function will be described later. 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.

The PWM0~PWM2 and $\overline{\text{PWM0}}$~$\overline{\text{PWM2}}$ output function are pin-shared with PC0, PC2, PC4 and PC1, PC3, PC5 respectively by software control.

There is a PWM function shared with PC0~PC5. If the PWM function is enabled, the PWM0~PWM2 and $\overline{\text{PWM0}}$~$\overline{\text{PWM2}}$ signal will appear on PC0, PC2, PC4 and PC1, PC3, PC5 respectively (if PC0/PC1/PC2/PC3/PC4/PC5 is operating in output mode).

Writing "1" to PC0/PC2/PC4 data register will enable the PWM0/PWM1/PWM2 output function and writing "0" will force the PC0/PC2/PC4 to remain at "0" (inactive state). Writing "1" to PC1/PC3/PC5 data register will enable the $\overline{\text{PWM0}}$/$\overline{\text{PWM1}}$/$\overline{\text{PWM2}}$ output function and writing "0" will force the PC1/PC3/PC5 to remain at "0" inactive state). The I/O functions of PC0~PC5 are as shown.

PWMEN	PC7	PC6	PC4	PC2	PC0
0	X	X	Logical	Logical	Logical
1	0	0	Logical	Logical	PWM
1	0	1	Logical	PWM	Logical
1	1	0	PWM	Logical	Logical
1	1	1	PWM	PWM	PWM

PC0, PC2, PC4 Output Function

PWMCEN	PC7	PC6	PC5	PC3	PC1
0	X	X	Logical	Logical	Logical
1	0	0	Logical	Logical	$\overline{\text{PWM}}$
1	0	1	Logical	$\overline{\text{PWM}}$	Logical
1	1	0	$\overline{\text{PWM}}$	Logical	Logical
1	1	1	$\overline{\text{PWM}}$	$\overline{\text{PWM}}$	$\overline{\text{PWM}}$

PC1, PC3, PC5 Output Function

There is only one channel PWM and $\overline{\text{PWM}}$ output at a time. The PC.6 and PC.7 is to determine which PWM0/$\overline{\text{PWM0}}$, PWM1/$\overline{\text{PWM1}}$ or PWM2/$\overline{\text{PWM2}}$ appeared to the PC0/PC1, PC2/PC3 or PC4/PC5.

PC7, PC6	PWM (PWMEN=1)	$\overline{\text{PWM}}$ (PWMCEN=1)
0, 0	PC0/PWM0 is PWM output, if the PC0 is output mode (PCC.0 = "0"). The PC2 and PC4 are the GPIO. Writing "1" to PC0 data register will enable the PWM0 output function and writing "0" will force the PC0 to remain at PWM0 output inactive state.	PC1/$\overline{\text{PWM0}}$ is $\overline{\text{PWM}}$ output, if the PC1 is output mode (PCC.1 = "0"). The PC3 and PC5 are the GPIO. Writing "1" to PC1 data register will enable the $\overline{\text{PWM0}}$ output function and writing "0" will force the PC1 to remain at $\overline{\text{PWM0}}$ output inactive state.
0, 1	PC2/PWM1 is PWM output, if the PC2 is output mode (PCC.2 = "0"). The PC0 and PC4 are the GPIO. Writing "1" to PC2 data register will enable the PWM1 output function and writing "0" will force the PC2 to remain at PWM1 output inactive state.	PC3/$\overline{\text{PWM1}}$ is $\overline{\text{PWM}}$ output, if the PC3 is output mode (PCC.3 = "0"). The PC1 and PC5 are the GPIO. Writing "1" to PC3 data register will enable the $\overline{\text{PWM1}}$ output function and writing "0" will force the PC3 to remain at $\overline{\text{PWM1}}$ output inactive state.
1, 0	PC4/PWM2 is PWM output, if the PC4 is output mode (PCC.4 = "0"). The PC0 and PC2 are the GPIO. Writing "1" to PC4 data register will enable the PWM2 output function and writing "0" will force the PC4 to remain at PWM2 output inactive state.	PC5/$\overline{\text{PWM2}}$ is $\overline{\text{PWM}}$ output, if the PC5 is output mode (PCC.5 = "0"). The PC1 and PC3 are the GPIO. Writing "1" to PC5 data register will enable the $\overline{\text{PWM2}}$ output function and writing "0" will force the PC5 to remain at $\overline{\text{PWM2}}$ output inactive state.
1, 1	PC0/PWM0, PC2/PWM1 and PC4/PWM2 are PWM output, if the PC0, PC2 and PC4 are output mode (PCC.0, 2, 4 = "0"). Writing "1" to PC0/PC2/PC4 data register will enable the PWM0/PWM1/PWM2 output function and writing "0" will force the PC0/PC2/PC4 to remain at PWM output inactive state.	PC1/$\overline{\text{PWM0}}$, PC3/$\overline{\text{PWM1}}$ and PC5/$\overline{\text{PWM2}}$ are $\overline{\text{PWM}}$ output, if the PC1, PC3 and PC5 are output mode (PCC.1, 3, 5 = "0"). Writing "1" to PC1/PC3/PC5 data register will enable the $\overline{\text{PWM0}}$/$\overline{\text{PWM1}}$/$\overline{\text{PWM2}}$ output function and writing "0" will force the PC1/PC3/PC5 to remain at PWM output inactive state.

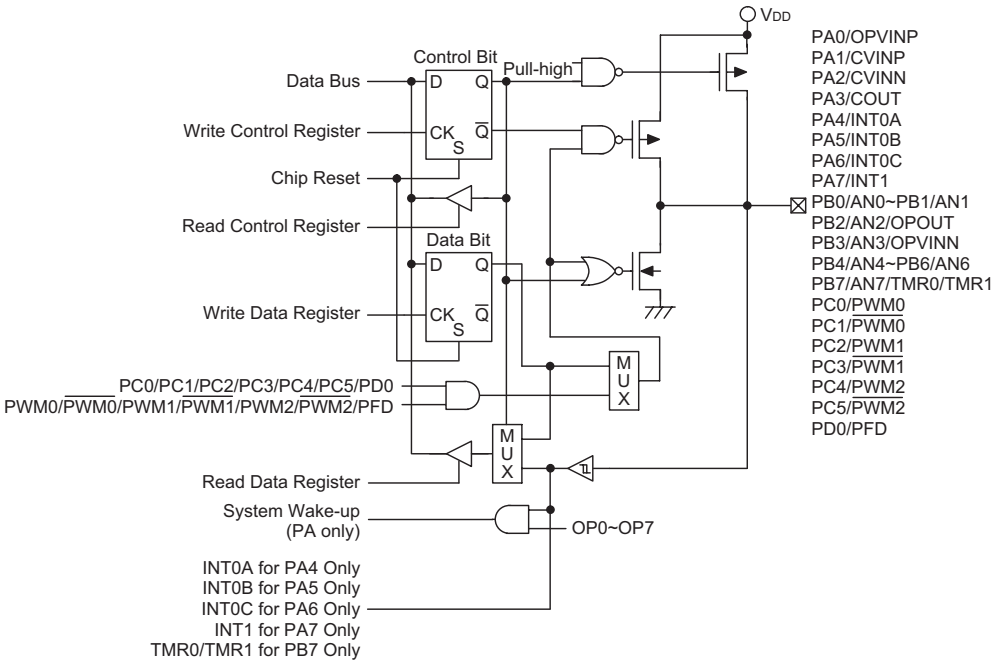
Note: If PWMEN=0, the PWM function output is disable and the PC0, PC2 and PC4 are the GPIO.

If PWMCEN=0, the $\overline{\text{PWM}}$ function output is disable and the PC1, PC3 and PC5 are the GPIO.

The PWMEN and PWMCEN are independent to enable or disable the PWM and PWM Complement function.

If the PCC.x is "1", the PC.x is input mode. "x" is from 0~5.

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.



Input/Output Ports

PWM

The microcontroller is provided with three channel PWM and Complementary PWM output shared with PC0~PC5, named PWM0~PWM2 and $\overline{PWM0}$~$\overline{PWM2}$, with standard 10 bits output, (9+1), (8+2), or (7+3) mode (configuration option determined). The PWM function provides output with a varied frequency and duty cycle by setting particular values into PWMC and PWML, PWMH registers. The frequency source of the PWM counter comes from f_{PWM}. The f_{PWM} clock source can be chosen from f_{SYS}~f_{SYS}/8 (software option determined).

The PWM channel has their data registers denoted as PWMH (1AH) and PWML (1BH). These two registers define the PWM output duty cycle. The PWMH is an 8-bit register, and it is a higher-order data register, the PWML is a lower-order register and two bits (bit1~bit0) are provided in this register. The PWM duty cycle is specified by writing to these two PWM data registers, writing PWML will only put the written data to an internal lower-order byte buffer (2-bit) and writing PWMH will transfer the specified data and the contents of the lower-order byte buffer to PWMH and PWML registers, respectively. Once the PC0~PC5 is selected as the PWM and Complementary PWM outputs and the output function of PC0~PC5 is enabled (PCC.0 ~ PCC.5 = ″0″), writing ″1″ to PC0, PC2 and PC4 register will enable the PWM output function and writing ″0″ will force the PC0, PC2 and PC4 stay at inactive state (″0″ if the PWMLEV option is select active high). Writing ″1″ to PC1, PC3 and PC5 register will enable the Complementary PWM output function and writing ″0″ will force the PC1, PC3 and PC5 stay at inactive state (″0″ if the PWMCLEV option is select active high).

The PWM clock source can be chosen form f_{SYS}~f_{SYS}/8 (dependent on software option). When the f_{PWM} clock source is selected from f_{SYS}~f_{SYS}/8, the PWM function provides with a fixed frequency output (f_{SYS}/1024~(f_{SYS}/8192). The PWM output provides standard 10-bit, (9+1), (8+2) and (7+3) output mode, it's duty cycle is decided by writing to the PWMH and PWML registers, the waveform of PWMH output is as shown.

When the PWM output function is enabled (writing ″1″ to PC0~PC5 register, when PWM and Complementary PWM output function is enable) and the f_{PWM} clock source is selected from PWM prescaler, the value of PWMPSC0~PWMPSC1, PWML and PWMH can be written to at any time even if the PWM and Complementary PWM output is running.

Register	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PWMH	D9	D8	D7	D6	D5	D4	D3	D2
PWML	—	—	—	—	—	—	D0	D1

PWMH (1AH), PWML (1BH) Register

A (9+1) bits mode PWM cycle is divided into two modulation cycles (modulation cycle 0~modulation cycle 1). Each modulation cycle has 512 PWM input clock period. In a (9+1) bit PWM function, the contents of the PWM register is divided into two groups. Group 1 of the PWM registers are denoted by DC which are the values of PWMH.7~PWMH.0 and PWML.1 (9 bit from PWMH~PWML.1). The group 2 is denoted by AC which is the value of PWML.0. In a (9+1) bits mode PWM cycle, the duty cycle of each modulation cycle is shown in the table.

Parameter	AC0~AC1	Duty Cycle
Modulation Cycle I (i=0~1)	i<AC	(DC+1)/512
	i≥AC	DC/512

A (8+2) bits mode PWM cycle is divided into four modulation cycles (modulation cycle 0~modulation cycle 3). Each modulation cycle has 256 PWM input clock period. In a (8+2) bit PWM function, the contents of the PWM register is divided into two groups. Group 1 of the PWM register is denoted by DC which is the value of PWMH.7~PWMH.0. The group 2 is denoted by AC which is the value of PWML.1~PWML.0 (PWML). In a (8+2) bits mode PWM cycle, the duty cycle of each modulation cycle is shown in the table.

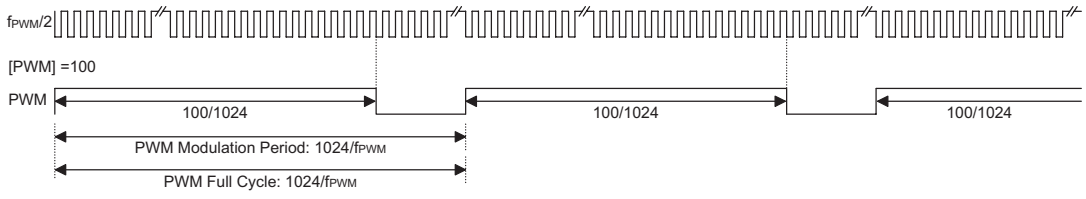
Parameter	AC0~AC3	Duty Cycle
Modulation Cycle I (i=0~3)	i<AC	(DC+1)/256
	i≥AC	DC/256

A (7+3) bits mode PWM cycle is divided into eight modulation cycles (modulation cycle 0~modulation cycle 7). Each modulation cycle has 128 PWM input clock period. In a (7+3) bit PWM function, the contents of the PWM register is divided into two groups. Group 1 of the PWM register is denoted by DC which is the value of PWMH.7~PWMH.1. The group 2 is denoted by AC which are the value of PWMH.0 and PWML.1~PWML.0 (PWMH.0~PWML.0). In a (7+3) bits mode PWM cycle, the duty cycle of each modulation cycle is shown in the table.

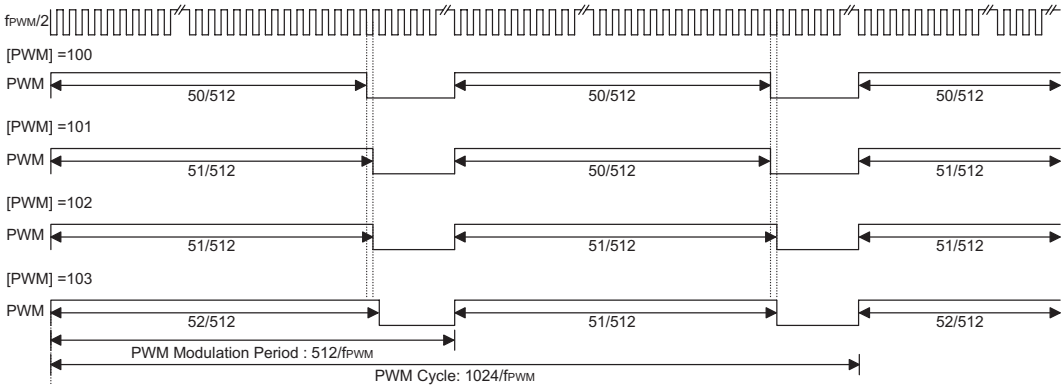
Parameter	AC0~AC7	Duty Cycle
Modulation Cycle I (i=0~7)	i<AC	(DC+1)/128
	i≥AC	DC/128

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

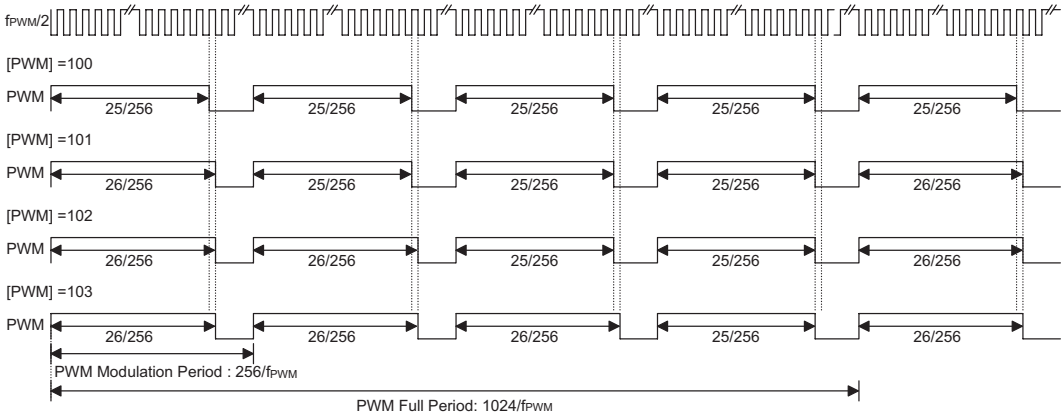
PWM Modulation Frequency	PWM Cycle Frequency	PWM Cycle Duty
f_{PWM}/1024 for standard 10-bits mode f_{PWM}/512 for (9+1) bits mode f_{PWM}/256 for (8+2) bits mode f_{PWM}/128 for (7+3) bits mode	f_{PWM}/1024	[PWM]/1024



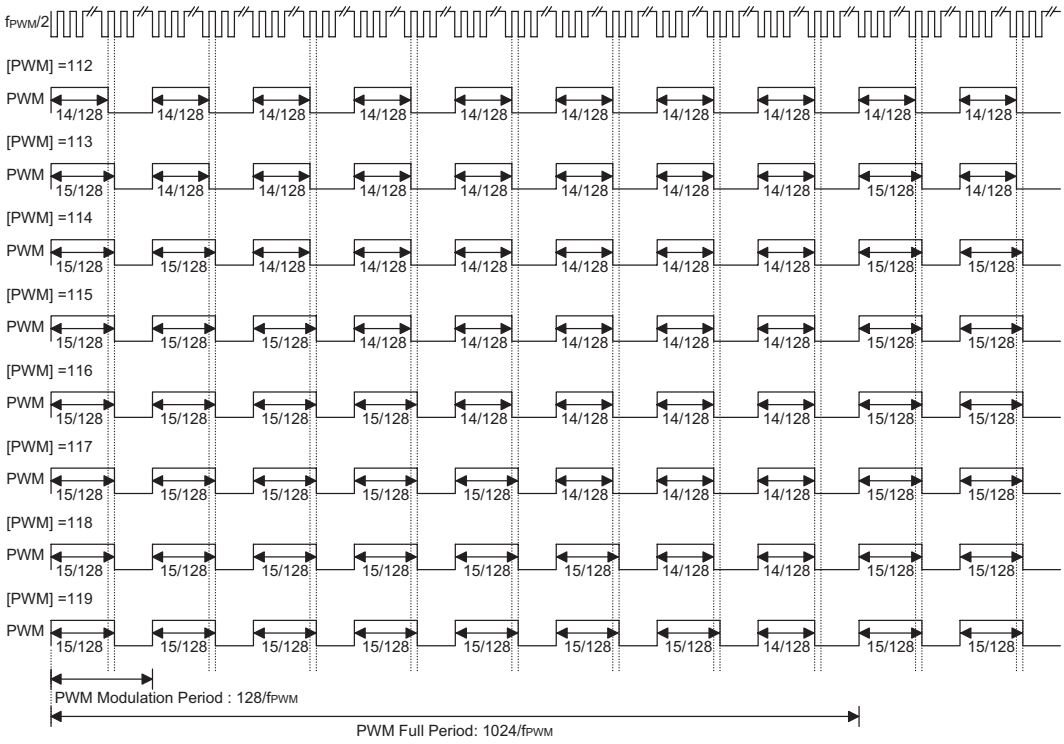
Standard 10 Bit PWM Mode



(9+1) PWM Mode



(8+2) PWM Mode



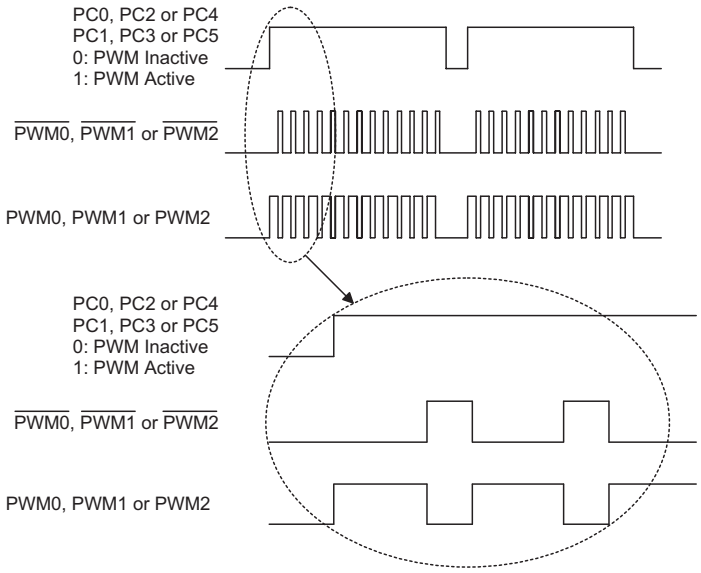
(7+3) PWM Mode

PWM and Complementary PWM output channels are shown below.

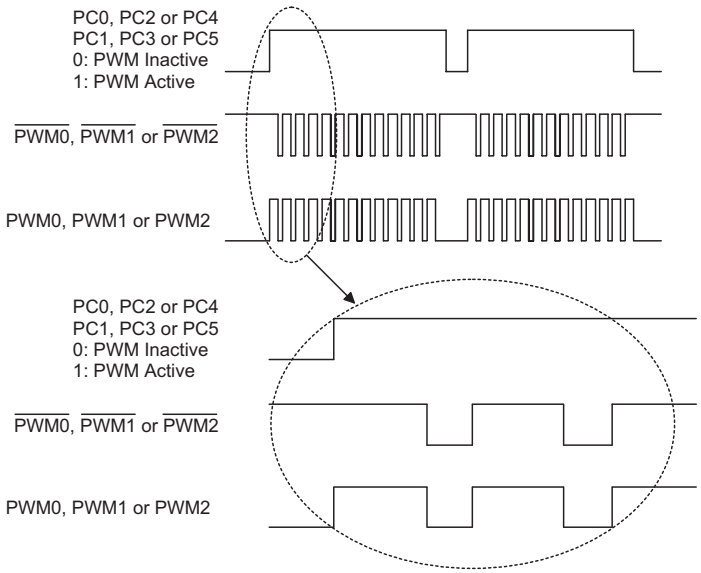
PWMEN	PC7	PC6	PWM
0	X	X	Disable PWM0, PWM1, PWM2
1	0	0	Enable PWM0
1	0	1	Enable PWM1
1	1	0	Enable PWM2
1	1	1	Enable PWM0, PWM1, PWM2

PWMCEN	PC7	PC6	PWM
0	X	X	Disable $\overline{PWM0}$, $\overline{PWM1}$, $\overline{PWM2}$
1	0	0	Enable $\overline{PWM0}$
1	0	1	Enable $\overline{PWM1}$
1	1	0	Enable $\overline{PWM2}$
1	1	1	Enable $\overline{PWM0}$, $\overline{PWM1}$, $\overline{PWM2}$

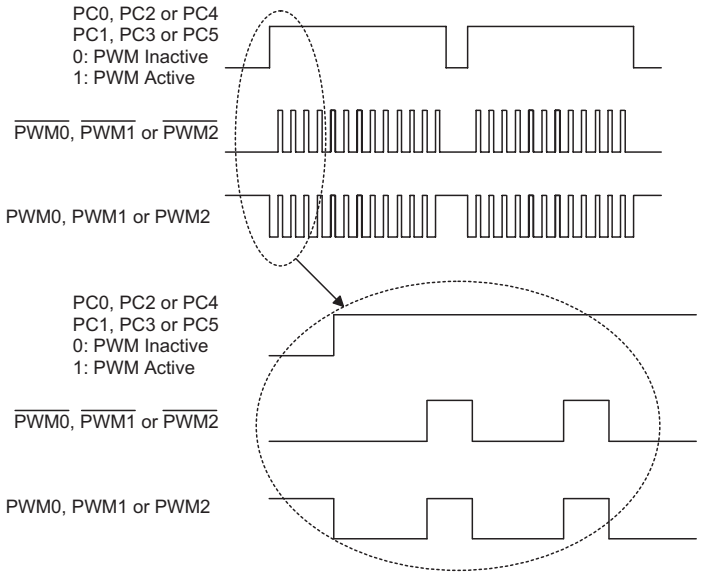
The PWM period interrupt will occur, when PWM counter is overflow. The PWM period interrupt is shown below.
PWM and Complementary PWM output are shown below.



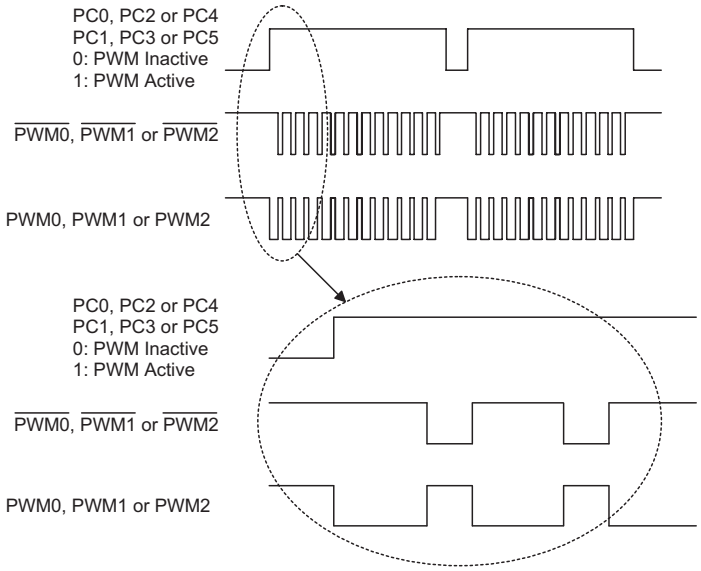
PWM and Complementary PWM Output Without Dead Time (PWMLEV & PWMCLEV = ″0″ & ″0″)



PWM and Complementary PWM Output Without Dead Time (PWMLEV & PWMCLEV = ″0″ & ″1″)

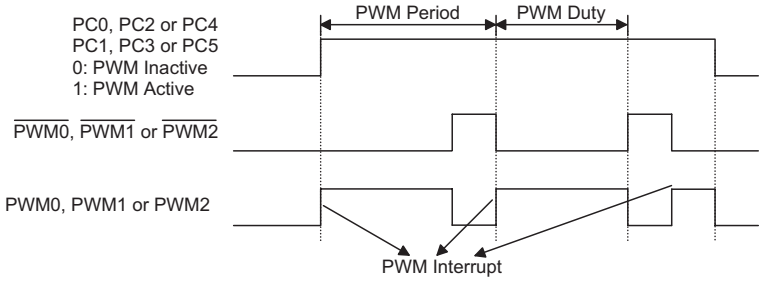


PWM and Complementary PWM Output Without Dead Time (PWMLEV & PWMCLEV = ″1″ & ″0″)



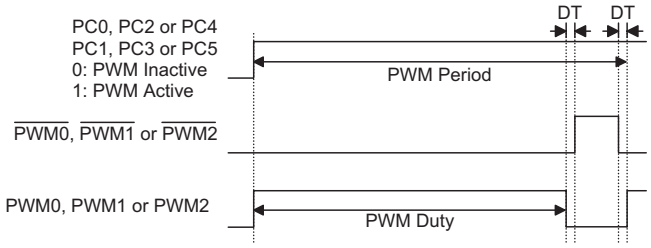
PWM and Complementary PWM Output Without Dead Time (PWMLEV & PWMCLEV = ″1″ & ″1″)

The Complementary PWM output provides with dead time function. The dead time is from $f_{SYS}/2$~$f_{SYS}/16$. The Comple-



PWM and Complementary PWM Output Without Dead Time (PWMLEV & PWMCLEV = ″0″ & ″0″)

mentary PWM output with dead time is shown below.



PWM and Complementary PWM Output Without Dead Time (PWMLEV & PWMCLEV = ″0″ & ″0″)

Bit No.	Label	Function
0	PWMEN	To enable/disable PWM output (0=disabled; 1=enabled)
1	PWMCEN	To enable/disable PWM Complementary output (0=disabled; 1=enabled)
2	DTEN	To enable/disable PWM Complementary output with dead time (0=without dead time; 1=with dead time)
3 4 5	PWMPS0 PWMPS1 PWMPS2	These three bits select the PWM clock prescaler rate.
6 7	PWMDT0 PWMDT1	These two bits select the PWM Complementary output dead time

PWMC (1CH) Register

The bits 3~5 of the PWM control register (PWMC) can be used to define the pre-scaling stages of the PWM clock.

PWMPS2	PWMPS1	PWMPS0	To Define the Prescaler Stages
0	0	0	$f_{PWM}=f_{SYS}$
0	0	1	$f_{PWM}=f_{SYS}/2$
0	1	0	$f_{PWM}=f_{SYS}/4$
0	1	1	$f_{PWM}=f_{SYS}/8$
1	0	0	$f_{PWM}=f_{SYS}/16$
1	0	1	$f_{PWM}=f_{SYS}/32$
1	1	0	$f_{PWM}=f_{SYS}/64$
1	1	1	$f_{PWM}=f_{SYS}/128$

PWM period according different PWM prescaler rate bits and system clock.

PWMPS2	PWMPS1	PWMPS0	f_{SYS}=4MHz	f_{SYS}=8MHz	f_{SYS}=12MHz	Unit
0	0	0	3.91	7.81	11.72	kHz
0	0	1	1.95	3.91	5.86	
0	1	0	0.98	1.95	2.93	
0	1	1	0.49	0.98	1.46	
1	0	0	0.24	0.49	0.73	
1	0	1	0.12	0.24	0.37	
1	1	0	0.06	0.12	0.18	
1	1	1	0.03	0.06	0.09	

10-bit PWM Full Frequency (f_{SYS} is 4, 8, 12MHz)

The bits 7~6 of the PWM control register (PWMC) can be used to define the PWM Complementary output dead time.

PWMDT1	PWMDT0	To Define the PWM Complementary Output Dead Time
0	0	Dead Time=f_{SYS}/2
0	1	Dead Time=f_{SYS}/4
1	0	Dead Time=f_{SYS}/8
1	1	Dead Time=f_{SYS}/16

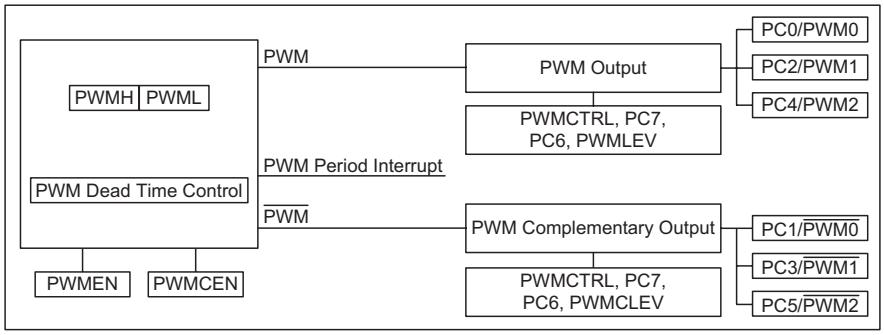
PWMDT1	PWMDT0	f_{SYS}=4MHz	f_{SYS}=8MHz	f_{SYS}=12MHz	Unit
0	0	0.50	0.25	0.17	μs
0	1	1.00	0.50	0.33	
1	0	2.00	1.00	0.67	
1	1	4.00	2.00	1.33	

Dead Time Setting (f_{SYS} is 4, 8, 12MHz)

PWM Options

There are two options to define the PWM and PWM Complementary output level. These two bits can be read by software.

Options	Description
PWM output level selection; PWMLEV.	This option is to determine the PWM output level. Active Low or Active High selection. Disable this bit to ″0″, the PWM output will be defined as an active high output, Enable this bit to ″1″, the PWM output will be defined as an active low output. If the PWM function disable, this bit is invalid.
PWM Complementary output level selection; PWMCLEV.	This option is to determine the PWM Complementary output level. Active low or active high selection. Disable this bit to ″0″, the PWM Complementary output will be defined as an active high output, Enable this bit to ″1″, the PWM Complementary output will be defined as an active low output. If the PWM Complementary function disable, this bit is invalid.



PWM

A/D Converter

The 8 channels and 9-bit resolution A/D converter are implemented in this microcontroller. The reference voltage is VDD. The A/D converter contains 4 special registers which are; ADRL (20H), ADRH (21H), ADCR (22H) and ACSR (23H). The ADRH and ADRL are A/D result register higher-order byte and lower-order byte and are read-only. After the A/D conversion is completed, the ADRH and ADRL should be read to get the conversion result data. The ADCR is an A/D converter control register, which defines the A/D channel number, analog channel select, start A/D conversion control bit and the end of A/D conversion flag. If the users want to start an A/D conversion, define PB configuration, select the converted analog channel, and give START bit a raising edge and falling edge (0→1→0). At the end of A/D conversion, the EOCB bit is cleared and an A/D converter interrupt occurs (if the A/D converter interrupt is enabled). The ACSR is A/D clock setting register, which is used to select the A/D clock source.

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

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
2~7	—	Unused bit, read as ″0″

ACSR (23H) Register

Bit No.	Label	Function
0 1 2	ACS0 ACS1 ACS2	Defines the analog channel select.
3 4 5	PCR0 PCR1 PCR2	Defines the port B configuration select. If PCR0, PCR1 and PCR2 are all 0, the ADC circuit is power 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 (22H) 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

ACS2	ACS1	ACS0	Analog 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

Analog Input Channel Selection

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.

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

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

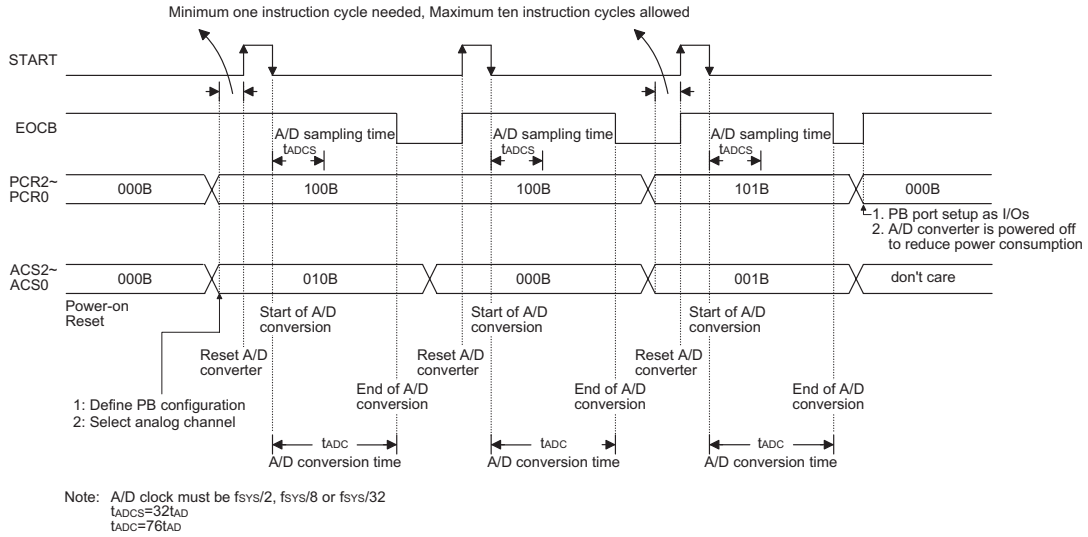
ADRL (20H), ADRH (21H) Register

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

```
                        ; 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
```

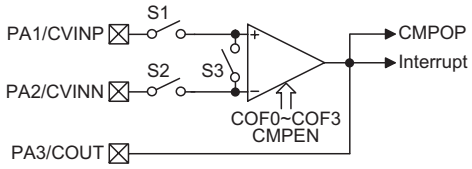


A/D Conversion Timing

Comparator

There is one Comparator in the device. The CMPEN bits is used as the enable or disable bits, if the CMPEN is cleared to ″0″, the Comparator is disabled, the PA1/CVINP, PA2/CVINN, PA3/COUT are all GPIO pins, if the CMPEN is set to ″1″, the Comparator is enabled, the PA1/CVINP, PA2/CVINN are Comparator input pins, PA3/COUT is a Comparator output pin.

The CVINP, CVINN and COUT are pin-shared with PA1, PA2 and PA3. Once the Comparator function is used, the internal registers related to PA1, PA2 cannot be used, PA3 can be used as input only(if the COUTEN is ″1″), and the PA1, PA2 I/O function, PA3 output function and pull-high resistor are disabled automatically(if the COUTEN is ″1″). Once the Comparator function is used, the PA3 is the GPIO when the COUTEN is ″0″. Software instructions determine the Comparator function to be used.



CRS	COFM	S1	S2	S3
0	0	ON	ON	OFF
0	1	OFF	ON	ON
1	0	ON	ON	OFF
1	1	ON	OFF	ON

Comparator Block Diagram

Bit No.	Label	Function
0~3	COF0~COF3	Comparator input offset voltage cancellation control bits
4	CRS	Comparator input offset voltage cancellation reference selection bit 1/0: select CVINP/CVINN as the reference input
5	COFM	Input offset voltage cancellation mode and comparator mode selection 1/0: input offset voltage cancellation mode/comparator mode
6	CMPOP	Comparator output; positive logic. This bit is read only.
7	CMPEN	Comparator enable/disable (1/0)

CMPC Register (Comparator Control Register)

CMPEN	COUTEN	PA1, PA2, PA3
0	X	PA1, PA2, PA3 is GPIO
1	0	PA1, PA2 is Comparator input pin and PA3 is GPIO.
1	1	PA1, PA2 is Comparator input pin and PA3 is Comparator output pin. PA3 can read the Comparator output status.

The Comparator enable/disable register, Comparator output pin enable/disable register and Comparator output flag are shown below.

COUTEN	PA3/COUT selection 0: PA3/COUT is as a GPIO pin 1: PA3/COUT is Comparator output, and the status of COUT can be read by reading the PA3 register.

The Comparator can be used for stopping the PWM by PWWSP0 and PWMSP1 register, which are shown below. The stopping PWM is used to clear the PWMCTRL bit to ″0″.

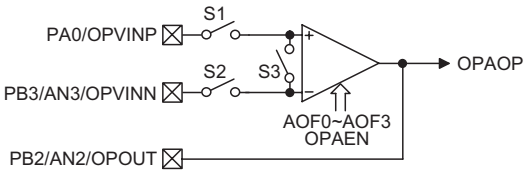
PWMSP0	Stopping the PWMand $\overline{\text{PWM}}$ using hardware selection. The stopping PWM method is to clear PWMCTRL bit to ″0″ by hardware. 00: PWM module output can be stop by software control only 01: PWM module output can be stop by COUT falling edge 10: PWM module output can be stop by INT1 interrupt 11: PWM module output can be stop by COUT falling edge or by INT1 interrupt
PWMSP1	

OPA (Operation Amplifier)

There is one OPA in the device. This OPA can be used for amplifier by user requirement. The OPAEN bits is used as the enable or disable bits, if the OPAEN is cleared to ″0″, the OPA is disabled and power off to save power consumption, the PB2/AN2/OPOUT, PB3/AN3/OPVINN, PA0/OPVINP are all GPIO pins, if the OPAEN is set to ″1″, the OPA is enabled, the PB3/AN3/OPVINN and PA0/OPVINP are OPA inverting and non-inverting input pins, PB2/AN2/OPOUT is a OPA output pin, PB2, PB3, PA0 output and pull-high resistor are disabled.

The OPA can be disabled or enabled by software control.

The OPVINP, OPVINN and OPOUT are pin-shared with PA0, PB3/AN3 and PB2/AN2 respectively. Once the OPA function is used, the internal registers related to PA0, PB3 and PB2 cannot be used, and the I/O function and pull-high resistor are disabled automatically. Software instructions determine the OPA function to be used.



ARS	AOFM	S1	S2	S3
0	0	ON	ON	OFF
0	1	OFF	ON	ON
1	0	ON	ON	OFF
1	1	ON	OFF	ON

Operational Amplifier Block Diagram

Bit No.	Label	Function
0~3	AOF0~AOF3	Operational amplifier input offset voltage cancellation control bits
4	ARS	Operational amplifier input offset voltage cancellation reference selection bit 1/0: select OPP/OPN as the reference input
5	AOFM	Input offset voltage cancellation mode and operational amplifier mode selection 1/0: input offset voltage cancellation mode/operational amplifier mode
6	OPAOP	Operational amplifier output; positive logic. This bit is read only.
7	OPAEN	Operational amplifier enable/disable (1/0)

OPAC Register (Operational Amplifier Control Register)

OPAEN	PA0, PA3/AN3, PB2/AN2
0	PA0 is GPIO and PB2/AN2, PB3/AN3 is GPIO or analog ADC input by ADCR register.
1	PA0, PB3/AN3 is OPA input pin and PB2/AN2 is OPA output pin. The PB2/AN2 and PB3/AN3 can be analog ADC input if the related ADC function is enabled.

Miscellaneous Register

There is one miscellaneous control registers for various functions. The Miscellaneous register is to enable/disable Comparator, enable/disable OPA, enable/disable Comparator COUT pin, enable/disable stopping PWM by hardware.

Bit No.	Label	Function
0	PWMLEV	This bit is read only. It's is the PWM output level option. 0: PWM output will be defined as an active high 1: PWM output will be defined as an active low If the PWM function disable, this bit is invalid.
1	PWMCLEV	This bit is read only. It's is the $\overline{\text{PWM}}$ output level option. 0: $\overline{\text{PWM}}$ output will be defined as an active high 1: $\overline{\text{PWM}}$ output will be defined as an active low If the $\overline{\text{PWM}}$ function disable, this bit is invalid.
2	PWMSP0	Stopping the PWM and $\overline{\text{PWM}}$ using hardware selection 00: PWM module output can be stop by software control only 01: PWM module output can be stop by COUT falling edge 10: PWM module output can be stop by INT1 interrupt 11: PWM module output can be stop by COUT falling edge or by INT1 interrupt
3	PWMSP1	
4	PWMCTRL	To active/inactive PWM and PWMC Complementary output (0=inactive; 1=active)
5	COUTEN	PA3/COUT selection 0: PA3/COUT is as a GPIO pin 1: PA3/COUT is Comparator output, and the status of COUT can be read by reading the PA3 register.
6	COUTR	Comparator output flag. This bit is read only 0: Comparator output is low state 1: Comparator output is high state
7	—	Unused bit, read as ″0″

MISC (1DH) Register

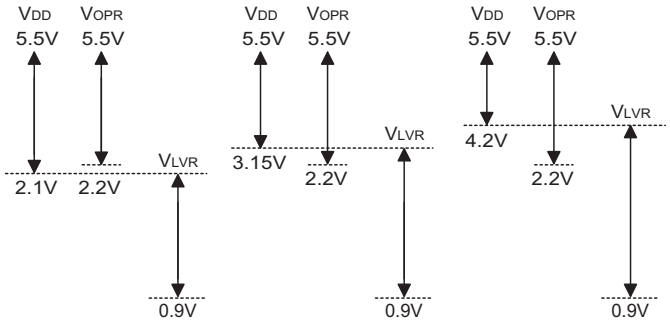
Low Voltage Reset – LVR

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

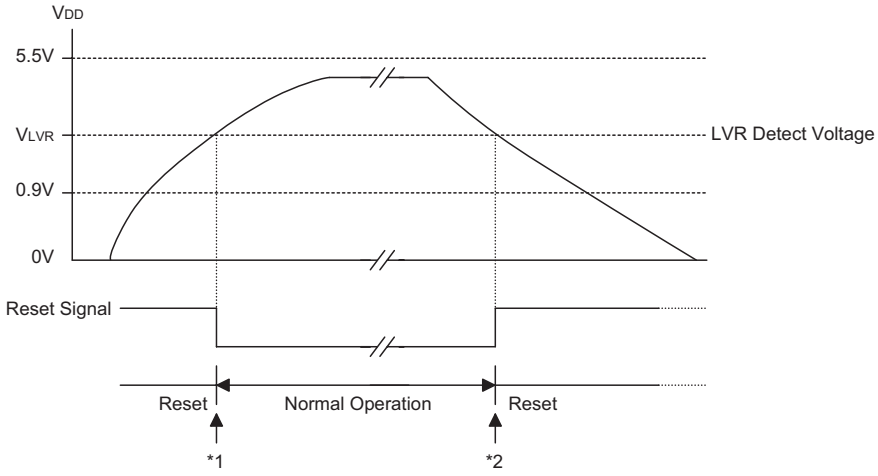
The LVR includes the following specifications:

- The low voltage (0.9V~V_{LVR}) has to remain in their original state to exceed t_{LVR}. If the low voltage state does not exceed t_{LVR}, 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.

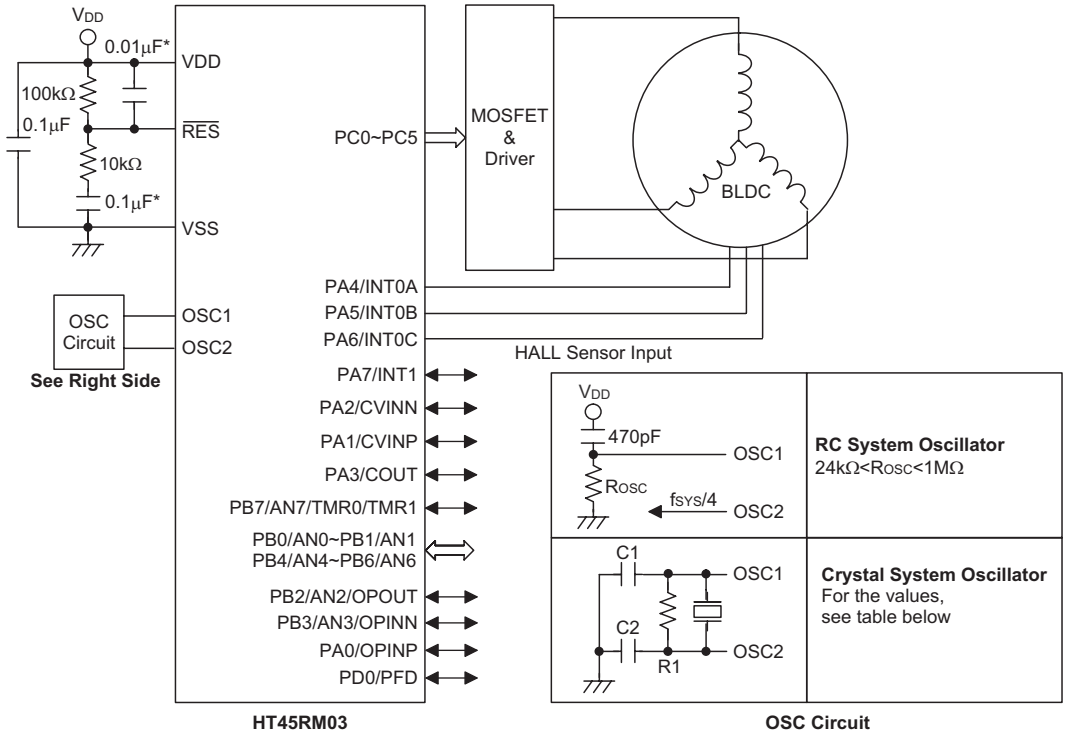
*2: Since low voltage state has to be maintained in its original state for over t_{LVR}, therefore after t_{LVR} delay, the device enters the reset mode.

Options

No.	Options
1	OSC type selection. This option is to decide if an RC or crystal oscillator is chosen as system clock.
2	WDT source selection. There are three types of selection: on-chip RC oscillator, instruction clock or disable the WDT.
3	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.
4	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.
5	Pull-high selection. This option is to decide whether a pull-high resistance is visible or not in the input mode of the I/O ports. PA0~PA7, PB0~PB7, can be independently selected.
6	PFD selection: PD0: level output or PFD output
7	Low voltage reset selection: Enable or disable LVR function, and LVR voltage selection.
8	PWM mode selection: 10 (9+1), (8+2) or (7+3) mode
9	INT0A, INT0B, INT0C and INT1 trigger edge: disable; high to low; low to high or low to high or high to low.
10	PWM output level selection; PWMLEV. This option is to determine the PWM output level. Active Low or Active High selection. Disable this bit to ″0″, the PWM output will be defined as an active high output, Enable this bit to ″1″, the PWM output will be defined as an active low output. If the PWM function disable, this bit is invalid.
11	PWM Complementary output level selection; PWMCLEV. This option is to determine the PWM Complementary output level. Active low or active high selection. Disable this bit to ″0″, the PWM Complementary output will be defined as an active high output, Enable this bit to ″1″, the PWM Complementary output will be defined as an active low output. If the PWM Complementary function disable, this bit is invalid.

Application Circuits

Application Circuits 1

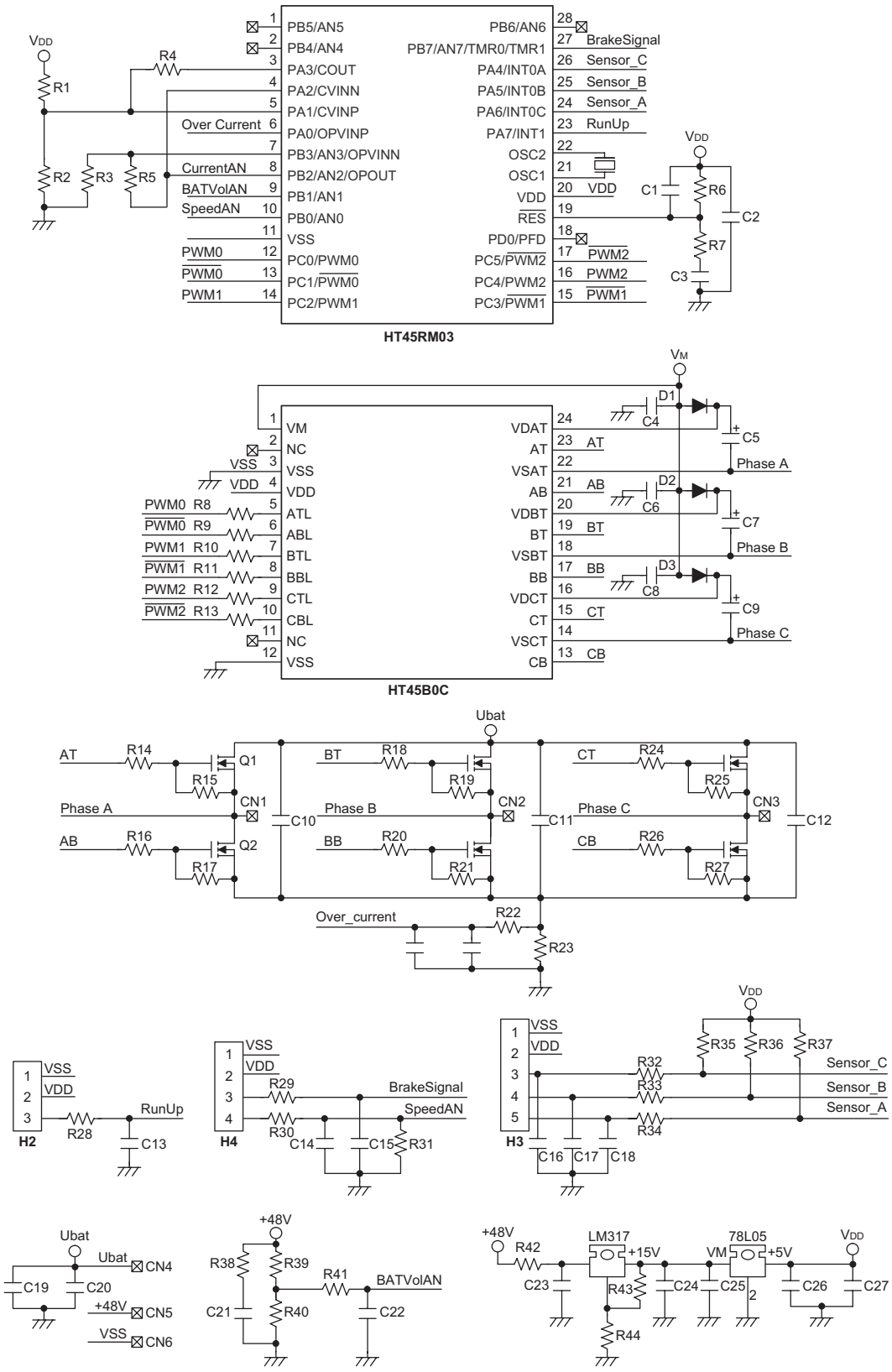


The following table shows the C1, C2 and R1 values corresponding to the different crystal values. (For reference only)

Crystal or Resonator	C1, C2	R1
8MHz Crystal	0pF	5.3kΩ
8MHz Resonator	10pF	6.3kΩ
4MHz Crystal	0pF	13kΩ
4MHz Resonator	10pF	12kΩ
3.58MHz Crystal	0pF	15kΩ
3.58MHz Resonator	25pF	10kΩ
2MHz Crystal	25pF	10kΩ
2MHz Resonator	25pF	12kΩ
1MHz Crystal	35pF	15kΩ
480kHz Resonator	300pF, 100pF	9.1kΩ
455kHz Resonator	300pF	10kΩ
429kHz Resonator	300pF	10kΩ
400kHz 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 $\overline{RES}$ high.
"*" Make the length of the wiring, which is connected to the $\overline{RES}$ pin as short as possible, to avoid noise interference.

Application Circuits 2



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.

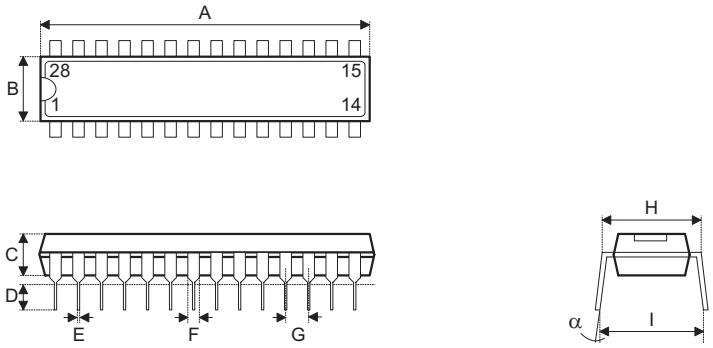
Operation ACC ← ACC ″XOR″ x

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

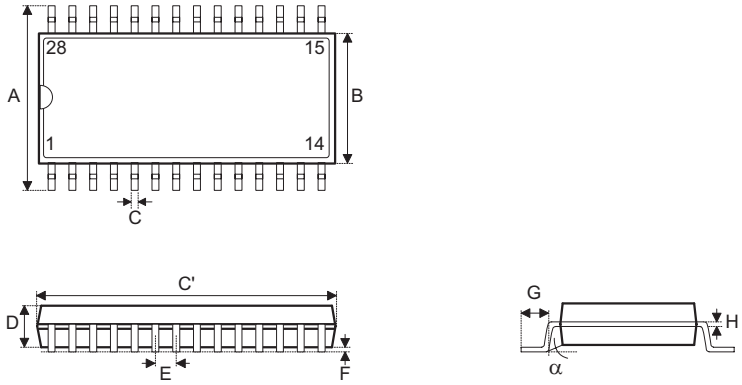
Package Information

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°

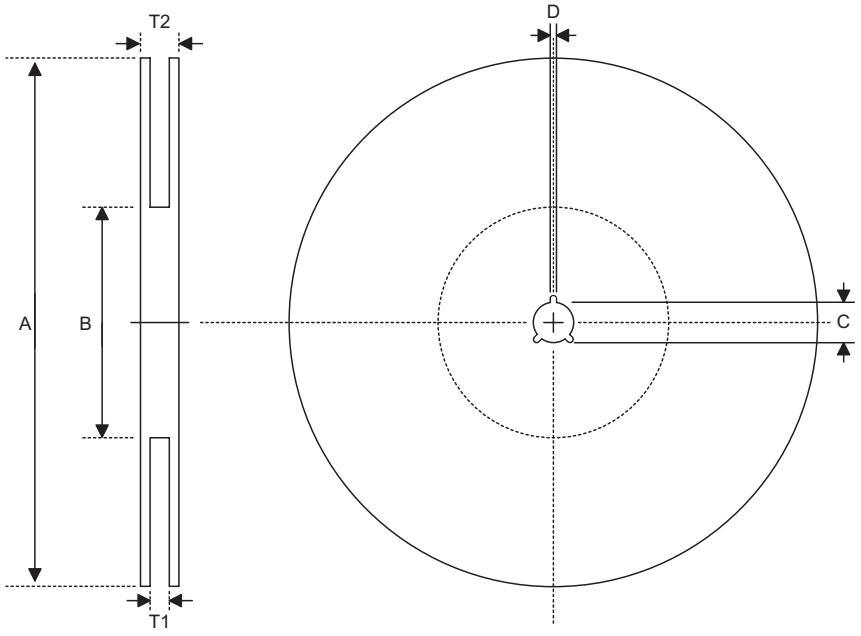
28-pin SOP (300mil) Outline Dimensions



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

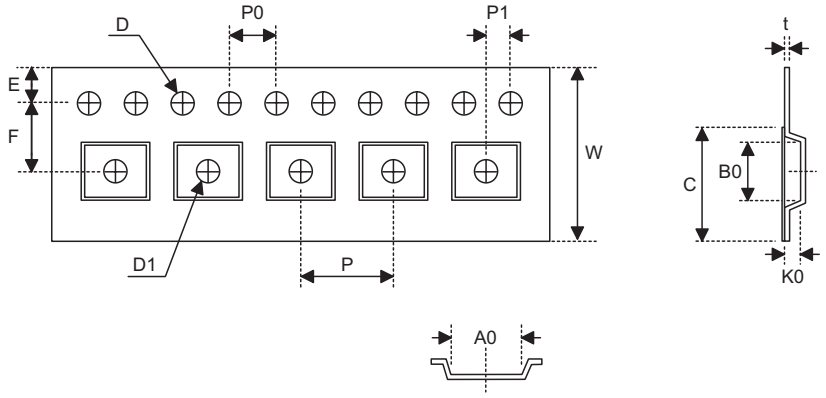
Reel Dimensions



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: 86-21-6485-5560
Fax: 86-21-6485-0313
http://www.holtek.com.cn

Holtek Semiconductor Inc. (Shenzhen Sales Office)
5/F, Unit A, Productivity Building, Cross of Science M 3rd Road and Gaoxin M 2nd Road, Science Park, Nanshan District, Shenzhen, China 518057
Tel: 86-755-8616-9908, 86-755-8616-9308
Fax: 86-755-8616-9533

Holtek Semiconductor Inc. (Beijing Sales Office)
Suite 1721, Jinyu Tower, A129 West Xuan Wu Men Street, Xicheng District, Beijing, China 100031
Tel: 86-10-6641-0030, 86-10-6641-7751, 86-10-6641-7752
Fax: 86-10-6641-0125

Holtek Semiconductor Inc. (Chengdu Sales Office)
709, Building 3, Champagne Plaza, No.97 Dongda Street, Chengdu, Sichuan, China 610016
Tel: 86-28-6653-6590
Fax: 86-28-6653-6591

Holmate Semiconductor, Inc. (North America Sales Office)
46729 Fremont Blvd., Fremont, CA 94538
Tel: 1-510-252-9880
Fax: 1-510-252-9885
http://www.holmate.com