Technical Document

- Tools Information
- FAQs
- Application Note

Features

- Operating voltage:
  f_{SYS}=4MHz: 2.2V~5.5V crystal clock mode
- f_{SYS}=12MHz: 2.7V~3.7V RC clock mode
- 35 bidirectional I/O lines (max.)
- 2 interrupt inputs shared with I/O lines
- 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 ROM
- 216×8 data memory RAM
- PFD function for sound generation
- HALT function and wake-up feature reduces power consumption
- Up to 0.5μs instruction cycle with 8MHz system clock at V_{DD}=5V
- 6-level subroutine nesting
- 16 channel 8-bit resolution A/D converter
- 1 channel (6+2)-bit PWM output shared with an I/O line
- Bit manipulation instruction
- 15-bit table read instruction
- 63 powerful instructions
- LVR reset voltage of 3V±0.3V
- All instructions executed in one or two machine cycles
- PB2, PB3, PD4, PD7 can be optioned as either CMOS or NMOS outputs
- Integrated dual SPI interfaces
- 28-pin SKDIP/SOP and 44-pin QFP packages

General Description

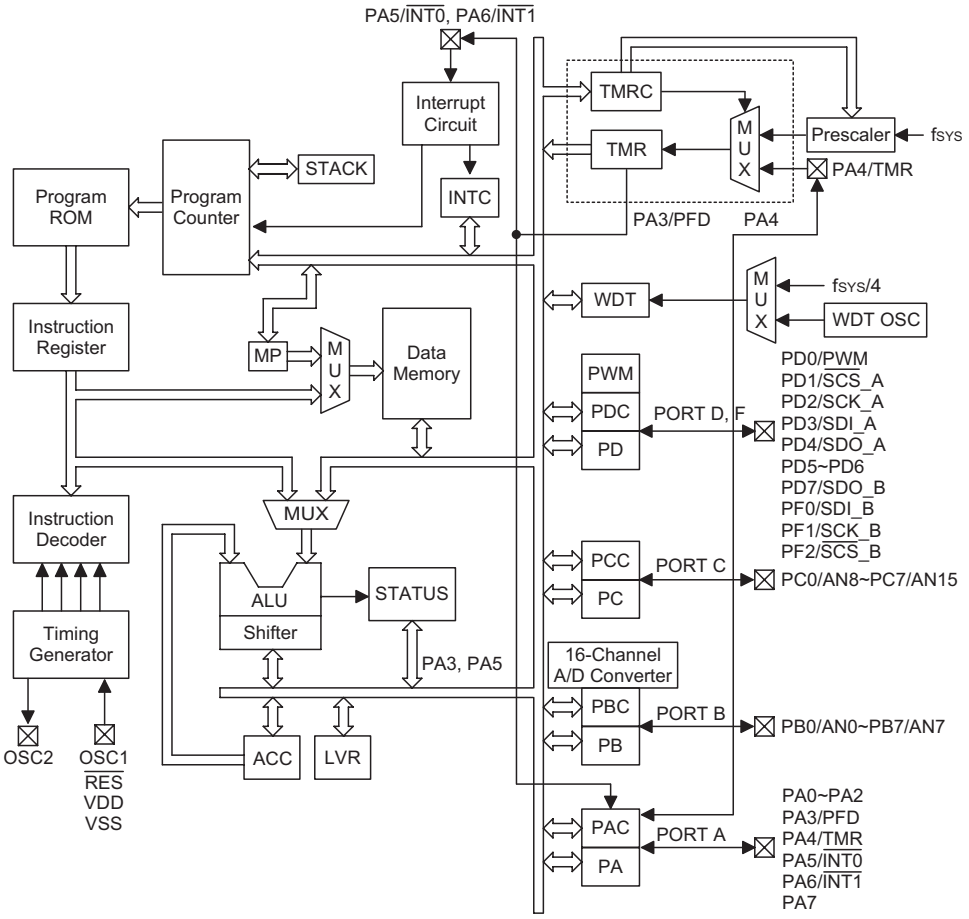
The HT82J30R/HT82J30A are 8-bit high performance, RISC architecture microcontroller devices specifically designed for A/D applications that interface directly to analog signals, such as those from sensors.

The mask version HT82J30A is fully pin and functionally compatible with the OTP version HT82J30R device.

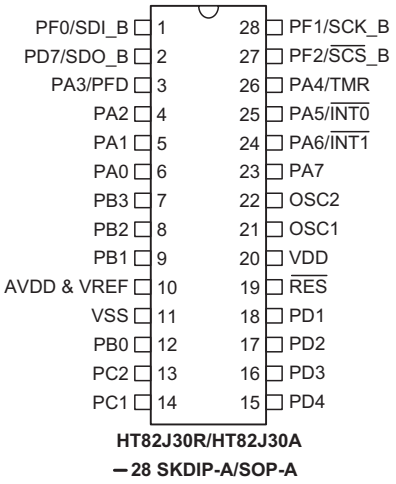
The advantages of low power consumption, I/O flexibility, programmable frequency divider, timer functions, oscillator options, multi-channel A/D Converter, Pulse Width Modulation function, Watchdog timer, SPI interfaces, Power Down and wake-up functions, enhance the versatility of these devices to suit a wide range of A/D application possibilities such as sensor signal processing, motor driving, industrial control, consumer products, subsystem controllers, etc. With the provision of dual SPI interfaces the devices are especially suitable for Joystick Encoder applications.

The HT82J30A is under development and will be available soon.

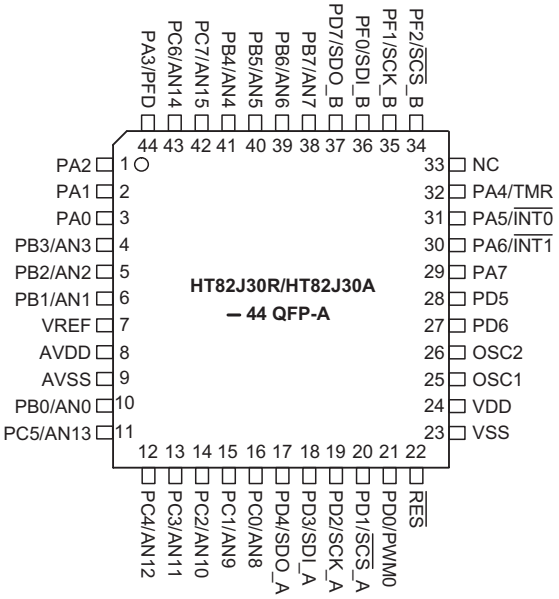
Block Diagram



Pin Assignment



HT82J30R/HT82J30A
— 28 SKDIP-A/SOP-A



HT82J30R/HT82J30A
— 44 QFP-A

Pin Description

Pin Name	I/O	Options	Description
PA0~PA2 PA3/PFD PA4/TMR PA5/$\overline{\text{INT0}}$ PA6/$\overline{\text{INT1}}$ PA7	I/O	Pull-high* Wake-up PA3 or PFD	Bidirectional 8-bit input/output port. Each individual pin on this port can be configured as a wake-up input by a configuration option. Software instructions determine if the pin is a CMOS output or Schmitt trigger input. Configuration options determine which pin on this port have pull-high resistors. The PFD, TMR and external interrupt input are pin-shared with PA3, PA4, and PA5 , PA6, respectively.
PB0/AN0~ PB7/AN7	I/O	Pull-high*	Bidirectional 8-bit input/output port. Software instructions determine if the pin is a CMOS output or Schmitt trigger input. Configuration options determine which pin on this port have pull-high resistors. PB is pin-shared with the A/D input pins. The A/D inputs are selected via software instructions Once selected as an A/D input, the I/O function and pull-high resistor functions are disable automatically. PB2, PB3 has CMOS or NMOS output option.
PC0/AN8~ PC7/AN15	I/O	Pull-high*	Bidirectional 8-bit input/output port. Software instructions determine if the pin is a CMOS output or Schmitt trigger input. Configuration options determine which pin on this port have pull-high resistors. PB is pin-shared with the A/D input pins. The A/D inputs are selected via software instructions Once selected as an A/D input, the I/O function and pull-high resistor functions are disabled automatically.
PD0/PWM0 PD1/$\overline{\text{SCS}}$_A PD2/SCK_A PD3/SDI_A PD4/SDO_A PD5~PD6 PD7/SDO_B	I/O	Pull-high* PD0 or PWM	Bi-directional 8-bit input/output port. Software instructions determine if the pin is a CMOS output or Schmitt trigger input. Configuration options determine which pin on this port have pull-high resistors. PD0 is pin-shared with the PWM output selected via configuration option. PD1~PD4 are pin-shared with SPI interface A. PD4, PD7 have CMOS or NMOS output options. PD7 is pin-shared with the SPI interface B.
PF0/SDI_B PF1/SCK_B PF2/$\overline{\text{SCS}}$_B	I/O	Pull-high*	Bidirectional 3-bit input/output port. Software instructions determine if the pin is a CMOS output or Schmitt trigger input. Configuration options determine which pin on this port have pull-high resistors. PF0~PF2 is pin-shared with the SPI interface B.
OSC1 OSC2	I O	Crystal or RC	OSC1 and OSC2 are connected to an external RC network or external crystal, determined by configuration option, for the internal system clock. If the RC system clock is selected, pin OSC2 can be used to measure the system clock at 1/4 system frequency.
$\overline{\text{RES}}$	I	—	Schmitt trigger reset input. Active low
VDD	—	—	Positive power supply
VSS	—	—	Negative power supply, ground
AVDD	—	—	Analog positive power supply
AVSS	—	—	Analog negative power supply
VREF	—	—	8-bit A/D reference voltage input pin

Note: * The pull-high resistors of each I/O port are controlled by options.

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.....0°C to 70°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}=12MHz at 56kΩ RC mode	2.7	—	3.7	V
I_{DD}	Operating Current (Crystal OSC, RC OSC)	3V	No load, f_{SYS}=12MHz ADC disable	—	1	2	mA
		5V		—	4	6	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, TMR and $\overline{INT}$	—	—	0	—	$0.3V_{DD}$	V
V_{IH1}	Input High Voltage for I/O Ports, TMR and $\overline{INT}$	—	—	$0.7V_{DD}$	—	V_{DD}	V
V_{IL2}	Input Low Voltage ($\overline{RES}$)	—	—	0	—	$0.4V_{DD}$	V
V_{IH2}	Input High Voltage ($\overline{RES}$)	—	—	$0.9V_{DD}$	—	V_{DD}	V
V_{LVR}	Low Voltage Reset	—	Configuration option: 3V	—	—	2.1	V
I_{OL}	I/O Port Sink Current	3V	V_{OL}=0.1V_{DD}	4	8	—	mA
		5V		10	20	—	mA
I_{OH}	I/O Port Source Current	3V	V_{OH}=0.9V_{DD}	–2	–4	—	mA
		5V		–5	–10	—	mA
R_{PH}	Pull-high Resistance	3V	—	20	60	100	kΩ
		5V	—	10	30	50	kΩ
V_{AD}	A/D Input Voltage	—	—	0	—	V_{DD}	V
E_{AD}	A/D Conversion Error	3V	—	—	±0.5	±1	LSB
		5V	—	—	±0.5	±1	LSB
I_{ADC}	Only ADC Enable, Others Disable	3V	No load	—	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_{SYS1}	System Clock (Crystal OSC)	—	2.2V~5.5V	400	—	4000	kHz
		—	3.3V~5.5V	400	—	8000	kHz
f_{SYS2}	System Clock (RC OSC)	—	2.2V~2.6V	1000	—	4000	kHz
		—	2.7V~5.5V	1000	12000	14000	kHz
f_{TIMER}	Timer I/P Frequency (TMR)	—	2.2V~5.5V	0	—	4000	kHz
		—	3.3V~5.5V	0	—	8000	kHz
t_{WDTOSC}	Watchdog Oscillator Period	3V	—	45	90	180	μs
		5V	—	32	65	130	μs
t_{WDT1}	Watchdog Time-out Period (WDT OSC)	—	—	2^{15}	—	2^{16}	t_{WDTOSC}
t_{WDT2}	Watchdog Time-out Period (System Clock)	—	—	2^{17}	—	2^{18}	$*t_{SYS}$
t_{RES}	External Reset Low Pulse Width	—	—	1	—	—	μs
t_{SST}	System Start-up Timer Period	—	Wake-up from HALT	—	1024	—	t_{SYS}
t_{INT}	Interrupt Pulse Width	—	—	1	—	—	μs
t_{AD}	A/D Clock Period	—	—	1	—	—	μs
t_{ADC}	A/D Conversion Time	—	—	—	76	—	t_{AD}
t_{ADCS}	A/D Sample Time	—	—	—	32	—	t_{AD}
t_{CS_SK}	SPI SCS to SCK Time	—	—	50	—	—	ns
t_{SPICK}	SPI Clock Time	—	—	400	—	—	ns

Note: $*t_{SYS}= 1/f_{SYS1}$ or $1/f_{SYS2}$

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 within one cycle. If an instruction changes the program counter, two cycles are required to complete the instruction.

Program Counter – PC

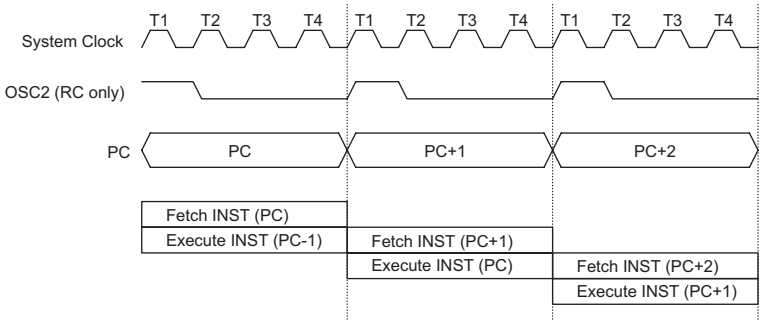
The program counter controls the sequence in which the instructions stored in the program memory are executed and its contents specify a full range of program memory.

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

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

The lower byte of the program counter is a readable and writeable register. 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
$\overline{\text{INT0}}$ External Interrupt	0	0	0	0	0	0	0	0	0	1	0	0
$\overline{\text{INT1}}$ External Interrupt	0	0	0	0	0	0	0	1	1	0	0	0
Timer/Event Counter Overflow	0	0	0	0	0	0	0	0	1	0	0	0
Reserved	0	0	0	0	0	0	0	0	1	1	0	0
SPI_A Interrupt	0	0	0	0	0	0	0	1	0	0	0	0
SPI_B Interrupt	0	0	0	0	0	0	0	1	0	1	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

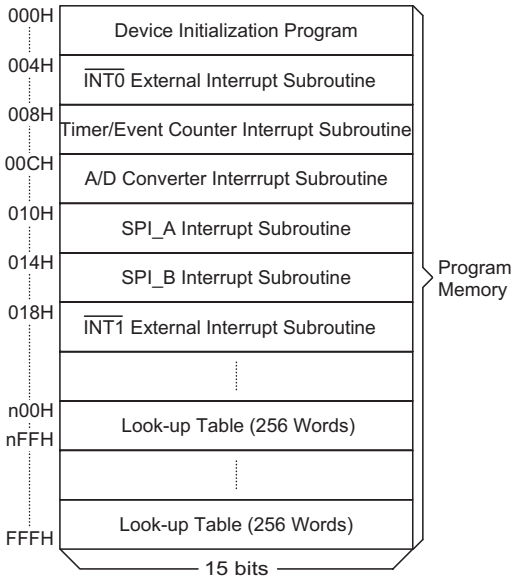
Note: *11~*0: Program counter bits
#11~#0: Instruction code bits
S11~S0: Stack register bits
@7~@0: PCL bits

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 4K×15 bits, addressed by the program counter and table pointer.

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

- Location 000H
  This area is reserved for program initialization. After chip reset, the program always begins execution at location 000H.
- Location 004H
  This area is reserved for the external interrupt service program. If the $\overline{\text{INT0}}$ input pin is activated, the interrupt is enabled and the stack is not full, the program begins execution at location 004H.
- Location 008H
  This area is reserved for the timer/event counter interrupt service program. If a timer interrupt results from a timer/event counter overflow, and if the interrupt is enabled and the stack is not full, the program begins execution at location 008H.



Note: n ranges from 0 to F

Program Memory

- Location 00CH
  This location is reserved for the A/D converter interrupt service program. If the interrupt is activated, when the A/D conversion is completed, if the interrupt is enabled and the stack is not full, the program begins execution at this location.
- Location 010H
  Location 010H is reserved for when 8 bits of data have been received or transmitted successfully from serial interface A. When the related interrupts are enabled, and the stack is not full, the program begins execution at location 010H.
- Location 014H
  Location 014H is reserved for when 8 bits of data have been received or transmitted successfully from serial interface B. When the related interrupts are enabled, and the stack is not full, the program begins execution at location 014H.
- Location 018H
  This location is reserved for the external interrupt service program. If the $\overline{\text{INT1}}$ input pin is activated, the interrupt is enabled and the stack is not full, the program begins execution at this location.
- Table location
  Any location in the Program Memory space can be used as a look-up table. The instructions "TABRDC [m]" (the current page, one page=256 words) and "TABRDL [m]" (the last page) transfers the contents of the lower-order byte to the specified data memory, and the higher-order byte to the TBLH register. 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. Any unused bits are read as "0". The Table Higher-order byte register, TBLH, is read only. The table pointer, TBLP, is a read/write register, which indicates the table location. Before accessing the table, the location must be placed in TBLP. The TBLH register 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 TBLH in the main routine are likely to be changed by the table read instruction used in the ISR and 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

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

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 should not be enabled until 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.

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 6 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 acknowledge signal, the contents of the program counter are pushed onto the stack. At the end of a subroutine or an interrupt routine, signaled by a return instruction, RET or RETI, the program counter is restored to its previous value from the stack. After a 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 acknowledge signal will be inhibited. When the stack pointer is decremented, by RET or RETI, the interrupt will be serviced. This feature prevents stack overflow allowing the programmer to use the structure more easily. 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 as only the most recent 6 return addresses are stored.

Data Memory – RAM

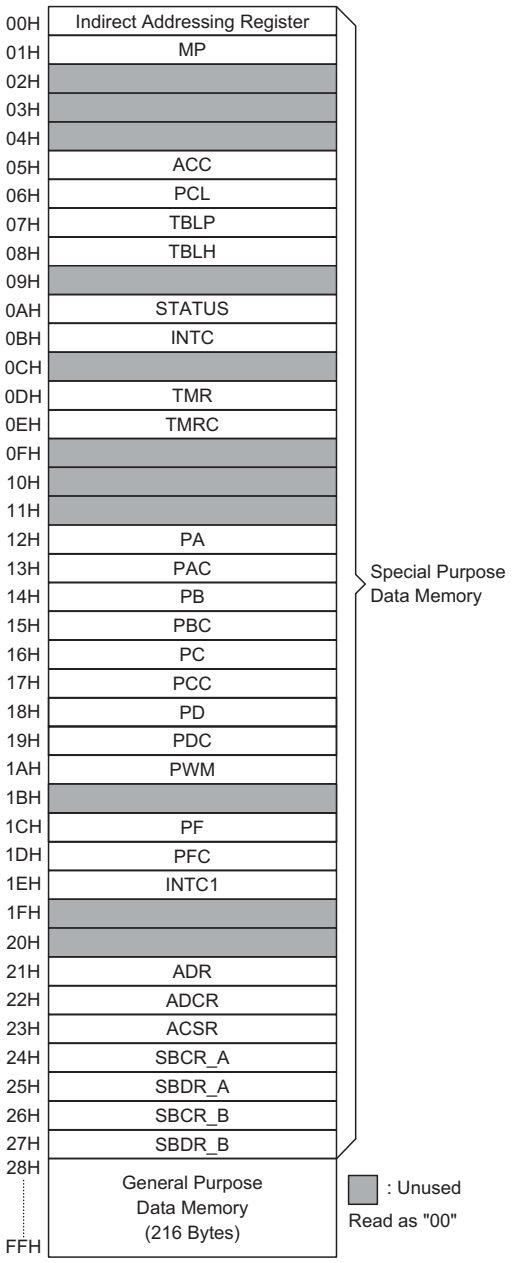
The data memory is designed with 226×8 bits. The data memory is divided into 2 functional groups: special function registers and general purpose data memory. Most of them are read/write, but some are read only.

Reading any unused locations will return the result ″00H″. The general purpose data memory, addressed from 28H to FFH, is used for data and control information under instruction commands.

All of the data memory areas can handle arithmetic, logic, increment, decrement and rotate operations directly. Except for some dedicated bits, each bit in the data memory can be set and reset by ″SET [m].i″ and ″CLR [m].i″. They are also indirectly accessible through the memory pointer registers MP.

Indirect Addressing Register

Location 00H is an indirect addressing register that is not physically implemented. Any read/write operation of [00H] accesses data memory pointed to by MP. Reading location 00H itself indirectly will return the result 00H. Writing indirectly results in no operation.



RAM Mapping

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 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 flags. 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 during a system power-up.

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

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

Interrupt

The microcontroller provides two external interrupts, an internal timer/event counter overflow interrupt, an A/D converter end-of-conversion interrupt and two SPI interrupts. The interrupt control registers INTC and INTC1 both contains the interrupt control bits to set the enable/disable and the interrupt request flags.

Once an interrupt subroutine is serviced, all the other interrupts will be blocked by clearing the EMI bit. This scheme may prevent any further interrupt nesting. Other interrupt requests may happen during this interval but only the interrupt request flags are recorded. If a certain interrupt requires servicing within the service routine, the programmer may set the EMI bit and the corresponding bit of INTC or INTC1 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 decreased. If immediate service is desired, the stack has to be prevented from becoming full.

All interrupts have a wake-up capability. As an interrupt is serviced, a control transfer occurs by pushing the program counter onto the stack and then branching to subroutines at a specified location in the program memory. Only the program counter is pushed onto the stack. If the contents of the register or status register are altered by the interrupt service program, which corrupts the desired control sequence, the programmer should save these contents first.

External interrupts are triggered by a high to low transition on pins $\overline{\text{INT0}}$ or $\overline{\text{INT1}}$ which will in turn set the related interrupt request flag, which is bit 4 of INTC or bit 6 of INTC1. When the respective interrupt is enabled, the stack is not full and the external interrupt is active, a subroutine call to location 004H or 018H will occur. The external interrupt request flag and EMI bits will cleared to disable other interrupts.

The internal timer/event counter interrupt is initialized by setting the timer/event counter interrupt request flag (bit 5 of INTC), caused by a timer overflow. When the interrupt is enabled, the stack is not full and the timer/event

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

Bit No.	Label	Function
0	EMI	Controls the master (global) interrupt (1= enabled; 0= disabled)
1	EEI_A	Controls the external interrupt (1= enabled; 0= disabled)
2	ETI	Controls the Timer/Event Counter interrupt (1= enabled; 0= disabled)
3	EADI	Controls the A/D converter interrupt (1= enabled; 0= disabled)
4	EIF_A	External interrupt request flag (1= active; 0= inactive)
5	TF	Internal Timer/Event Counter request flag (1= active; 0= inactive)
6	ADF	End of A/D conversion interrupt request flag (1= active; 0= inactive)
7	—	Unused bit, read as ″0″

INTC (0BH) Register

Bit No.	Label	Function
0	ESII_A	Controls the serial interface interrupt (1= enabled; 0= disabled)
1	ESII_B	Controls the serial interface interrupt (1= enabled; 0= disabled)
2	EEI_B	Controls the $\overline{\text{INT1}}$ external interrupt (1= enabled; 0= disabled)
3, 7	—	Unused bits, read as ″0″
4	SIF_A	Serial interface interrupt request flag (1= active; 0= inactive)
5	SIF_B	Serial interface interrupt request flag (1= active; 0= inactive)
6	EIF_B	$\overline{\text{INT1}}$ External interrupt request flag (1= active; 0= inactive)

INTC1 (1EH) Register

counter interrupt request flag is set, a subroutine call to location 00CH will occur. The related interrupt request flag will be reset and the EMI bit cleared to disable further interrupts.

The A/D converter end-of-conversion interrupt is initialized by setting the A/D end-of-conversion interrupt request flag (bit 6 of INTC), caused by an end of A/D conversion. When the interrupt is enabled, the stack is not full and the end of A/D conversion interrupt request flag is set, a subroutine call to location 0CH will occur. The related interrupt request flag will be reset and the EMI bit cleared to disable further interrupts.

There are two serial interface interrupts, which will be generated when the interface receives or transmits 8-bits of data. These interrupts are indicated by the interrupt flags, SIF_A; bit 4 of INTC1, and SIF_B; bit 5 of INTC1. The serial interface interrupts are enabled by setting the serial interface interrupt control bits, ESII_A; bit 0 of INTC1 and ESII_B; bit 1 of INTC1. After the respective interface is enabled by setting the corresponding SBEN bit, which is bit 4 of either SBCR_A or SBCR_B, if the stack is not full and the corresponding SIF_A or SIF_B bit is set, a subroutine call to location 10H or 14H occurs.

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 bits are set to ″1″ (if the stack is not full). To return from the interrupt subroutine, a ″RET″ or ″RETI″ instruction may be executed. RETI will set the EMI bit to enable further interrupts, but RET will not.

Interrupts, occurring in the interval between rising edge of two consecutive T2 pulses, will be serviced on the later of the two T2 pulses, if the corresponding interrupts are enabled. In the case of simultaneous requests the priorities in the follow table apply.

These can be masked by clearing the EMI bit.

Interrupt Source	Priority	Vector
$\overline{\text{INT0}}$ External Interrupt	1	04H
Timer/Event Counter Overflow	2	08H
End of A/D Conversion Interrupt	3	0CH
SPI_A Interrupt	4	10H
SPI_B Interrupt	5	14H
$\overline{\text{INT1}}$ External Interrupt	6	18H

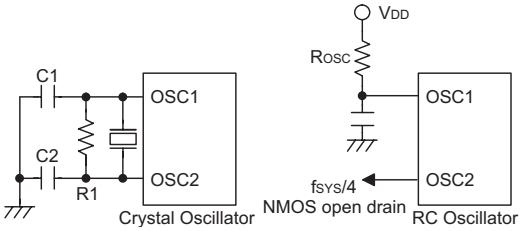
The timer/event counter interrupt request flag (TF), external interrupt request flag (EIF), A/D converter request flag (ADF), enable timer/event counter bit (ETI), enable external interrupt bit (EEI), enable A/D converter interrupt bit (EADI) and enable master interrupt bit (EMI) constitute an interrupt control register (INTC) which is located at 0BH in the data memory. EMI, EEI, ETI, EADI are used to control the enabling/disabling of interrupts. These bits

prevent the requested interrupt from being serviced. Once the interrupt request flags (TF, EIF, ADF) are set, they will remain in the INTC register until the interrupts are serviced or cleared by a software instruction.

It is recommended that a program does not use the CALL subroutine within the interrupt subroutine. 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 if a ″CALL″ instruction is executed in the interrupt subroutine.

Oscillator configuration

There are 2 oscillator circuits in the microcontroller.



System Oscillator

Both of them are designed for system clocks, namely the external RC oscillator, the external Crystal oscillator and the internal RC oscillator, the choice of which is determined by configuration options. The Power Down mode stops the system oscillator to conserve power.

If an RC oscillator is used, an external resistor between OSC1 and VDD is required and the resistance must range from 47kΩ to 750kΩ. The system clock, divided by 4, is available on OSC2, which can be used to synchronize external logic. The RC oscillator provides the most cost effective solution. However, the oscillation frequency may vary with VDD, temperature and 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. No other external components are required. Instead of a crystal, a resonator can also be connected between OSC1 and OSC2 to obtain a frequency reference, but two external capacitors connected between OSC1 and OSC2 and ground are required.

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

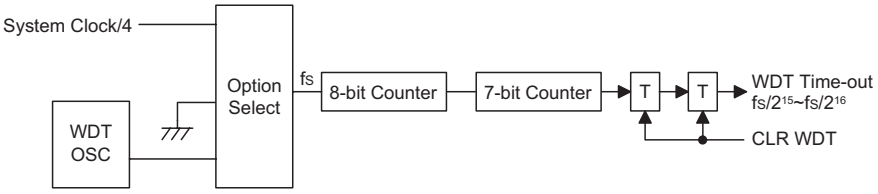
Watchdog Timer – WDT

The clock source of WDT is implemented by a dedicated RC oscillator (WDT oscillator) or instruction clock (system clock divided by 4), chosen via a configuration option. This timer is designed to prevent software malfunctions or the program jumping to unknown locations. The Watchdog Timer can be disabled by a configuration option. If the Watchdog Timer is disabled, all the executions related to the WDT result in no operation.

If the internal oscillator, which is an RC oscillator with a nominal period of 65μs at 5V, is selected, it is first divided by 32768~65536 to get a time-out period of approximately 2.1s~4.3s. This time-out period may vary with temperature, VDD and process variations. If the WDT oscillator is disabled, the WDT clock may still come from the instruction clock and operate in the same manner except that in the HALT state the WDT may stop counting and lose its protecting purpose. In this situation the logic can only be restarted by external logic.

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

The WDT overflow under normal operation will initialize a ″chip reset″ and set the status bit ″TO″. But in the Power-down mode, the overflow will initialize a ″warm reset″, and only the program counter and the SP are reset to zero. To clear the contents of the WDT, three methods are adopted; an external reset (a low level on the $\overline{RES}$ pin), a software instruction or a HALT instruction. The software instructions include ″CLR WDT″ and the other set – ″CLR WDT1″ and ″CLR WDT2″. Of these two types of instruction, only one can be active depending on the configuration option – ″CLR WDT times selection option″. If the ″CLR WDT″ is selected (i.e. CLR WDT 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. CLR WDT times equal two), these two instructions must be executed to clear the WDT; otherwise, the WDT may reset the chip as a result of a time-out.



Watchdog Timer

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 remains running (if the WDT oscillator is selected).
- The contents of the on chip RAM and registers remain unchanged.
- The WDT and WDT prescaler will be cleared and resume counting again (if the WDT clock comes from the WDT oscillator).
- 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 Power-down 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 a 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 remain in their original status.

The port A wake-up and interrupt methods of wake-up can be considered as a continuation of normal execution. Each bit in port A can be independently selected to wake-up the device using configuration options. Awakening from an I/O port stimulus, the program will resume execution of the next instruction. If it awakens from an interrupt, two sequences may occur. If the related interrupt is disabled or the interrupt is enabled but the stack is full, the program will resume execution at the next instruction. If the interrupt is enabled and the stack is not full, the regular interrupt response takes place. If an interrupt request flag is set to ″1″ before entering the Power-down 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 a wake-up. If the wake-up results from an interrupt acknowledge signal, 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 has finished.

To minimize power consumption, all the I/O pins should be carefully managed before entering the Power-down mode.

Reset

There are three ways in which a reset can occur:

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

The WDT time-out during Power-down 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 their ″initial condition″ when the reset conditions are met. By examining the PDF and TO flags, the program can distinguish between different ″chip resets″.

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

Note: ″u″ stands for ″unchanged″

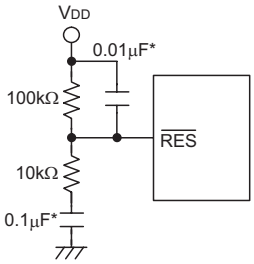
To guarantee that the system oscillator is started and stabilized, the SST (System Start-up Timer) provides an extra-delay of 1024 system clock pulses when the system reset (power-up, WDT time-out or $\overline{\text{RES}}$ reset) or the system awakes from the Power-down mode.

When a system reset occurs, the SST delay is added during the reset period. Any wake-up from Power-down will enable the SST delay.

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

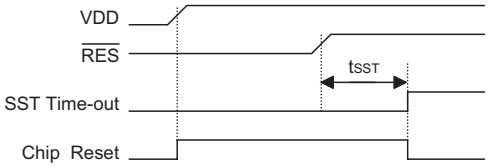
The functional unit chip reset status is shown below.

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

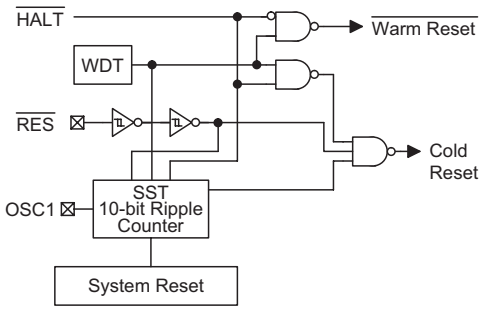


Reset Circuit

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

The register states is summarized in the table.

Register	Reset (Power-on)	WDT Time-out (Normal Operation)	$\overline{RES}$ Reset (Normal Operation)	$\overline{RES}$ Reset (HALT)	WDT Time-out (HALT)*
MP	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
INTC	-000 0000	-000 0000	-000 0000	-000 0000	-uuu uuuu
INTC1	-000 -000	-000 -000	-000 -000	-000 -000	-uuu -uuu
WDTS	0000 0111	0000 0111	0000 0111	0000 0111	uuuu 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	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PCC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PD	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PDC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PF	---- -111	---- -111	---- -111	---- -111	--- -uuu

Register	Reset (Power-on)	WDT Time-out (Normal Operation)	$\overline{RES}$ Reset (Normal Operation)	$\overline{RES}$ Reset (HALT)	WDT Time-out (HALT)*
PFC	---- -111	---- -111	---- -111	---- -111	--- -uuu
INTC1	-000 0000	-000 0000	-000 0000	-000 0000	-uuu uuuu
PWM	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
ADR	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
ADCR	0100 0000	0100 0000	0100 0000	0100 0000	uuuu uuuu
ACSR	1--- --00	1--- --00	1--- --00	1--- --00	u--- --uu
SBCR_A	0110 0000	0110 0000	0110 0000	0110 0000	uuuu uuuu
SBDR_A	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
SBCR_B	0110 0000	0110 0000	0110 0000	0110 0000	uuuu uuuu
SBDR_B	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu

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

Timer/Event Counter

A timer/event counter (TMR) is implemented in the microcontroller. The timer/event counter contains an 8-bit programmable count-up counter whose clock may come from an external source or the system clock.

Using an external clock input allows the user to count external events, measure time internals or pulse widths, or generate an accurate time base. While using the internal clock allows the user to generate an accurate time base.

The timer/event counter can generate PFD signal by using an external or internal clock. The PFD frequency is determined by the equation $f_{INT}/[2\times(256-N)]$.

There are 2 registers related to the timer/event counter; TMR and TMRC. Two physical registers are mapped to the TMR location; writing to TMR places the start value into the timer/event counter preload register. Reading TMR retrieves the contents of the timer/event counter. The TMRC register is a timer/event counter control register, which defines some options.

The TM0, TM1 bits define the operating mode. The event count mode is used to count external events, which means the clock source comes from the external TMR pin. The timer mode functions as a normal timer with the clock source coming from the f_{INT} clock. The pulse width measurement mode can be used to measure a high or low level duration of the external TMR pin. The counting is based on f_{INT}.

In the event count or timer mode, once the timer/event counter starts counting, it will count from the current contents in the timer/event counter to FFH. Once an overflow occurs, the counter is reloaded from the timer/event counter preload register and generates an interrupt request flag (TF; bit 5 of INTC) at the same time.

In the pulse width measurement mode with the TON and TE bits equal to one, once the TMR pin has received a transient from low to high (or high to low if the TE bit is "0") it will start counting until the TMR pin returns to its original level and resets the TON bit. The measured result will remain in the timer/event counter even if the activated transient occurs again. In other words, only a one cycle measurement can be implemented. Until the TON bit is again set, the cycle measurement will not function even if it receives further transient pulses. Note that, in this operating mode, the timer/event counter starts counting not according to the logic level but according to the transient edges. In the case of a counter overflow, the counter is reloaded from the timer/event counter preload register and issues an interrupt request just like the other two modes. To enable the counting operation, the timer ON bit (TON; bit 4 of TMRC) should be set to 1. In the pulse width measurement mode, the TON bit will be cleared automatically after the measurement cycle is completed. But in the other two modes the TON bit can only be reset by instructions. The overflow of the timer/event counter is one of the wake-up sources. No matter what the operation mode is, writing a 0 to ETI can disable the interrupt service.

In the case of a timer/event counter OFF condition, writing data to the timer/event counter preload register will also reload that data to the timer/event counter. But if the timer/event counter is turned on, data written to it will only be kept in the timer/event counter preload register. The timer/event counter will still operate until an overflow occurs. When the timer/event counter is read, the clock will be blocked to avoid errors. As clock blocking may results in a counting error, this must be taken into consideration by the programmer.

The bit0~bit2 of the TMRC can be used to define the pre-scaling stages of the internal clock sources of timer/event counter. The overflow signal of the timer/event counter can be used to generate the PFD signal.

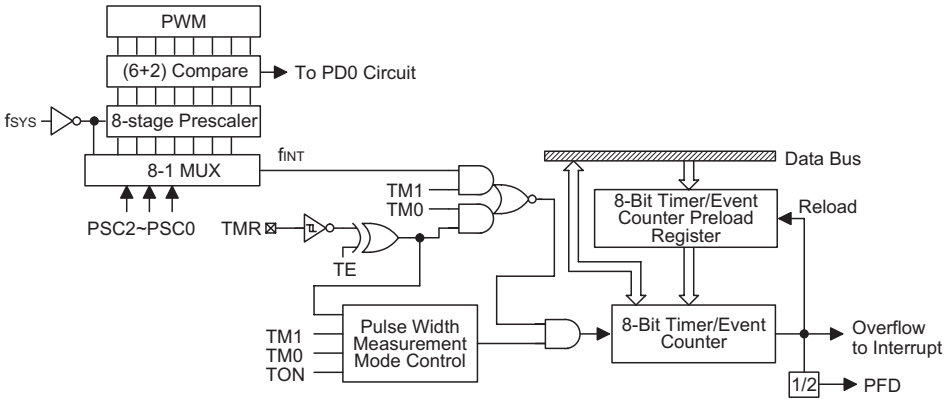
Input/Output Ports

There are 35 bidirectional input/output lines in the microcontroller, labeled as PA, PB, PC, PD and PF, which are mapped to the data memory of [12H], [14H], [16H], [18H] and [22H] respectively. All of these I/O ports can be used for input and output operations. For input operation, these ports are non-latching, that is, the inputs must be ready at the T2 rising edge of instruction "MOV A,[m]" (m=12H, 14H, 16H, 18H or 22H). For output operation, all the data is latched and remains unchanged until the output latch is rewritten.

Each I/O line has its own control register (PAC, PBC, PCC, PDC, PFC) to control the input/output configuration. With this control register, a 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

Bit No.	Label	Function
0 1 2	PSC0 PSC1 PSC2	Defines the prescaler stages, PSC2, PSC1, PSC0= 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	TE	Defines the TMR active edge of the timer/event counter: In Event Counter Mode (TM1,TM0)=(0,1): 1:count on falling edge; 0:count on rising edge In Pulse Width measurement mode (TM1,TM0)=(1,1): 1: start counting on the rising edge, stop on the falling edge; 0: start counting on the falling edge, stop on the rising edge
4	TON	Enable or disable the timer counting (0=disable; 1=enable)
5	—	Unused bits, read as "0"
6 7	TM0 TM1	Defines the operating mode (TM1, TM0)= 01=Event count mode (external clock) 10=Timer mode (internal clock) 11=Pulse width measurement mode 00=Unused

TMRC (0EH) Register



Timer/Event Counter

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, all I/Os except PB2, PB3, PD2, PD7 are CMOS types. There are options to define PB2, PB3, PD2, PD7 as either CMOS or NMOS types. These control registers are mapped to locations 13H, 15H, 17H,19H and 23H.

After a device reset, these input/output lines remain at high levels or a floating state, depending upon the pull-high configuration options. Each bit of these input/output latches can be set or cleared using the ″SET [m].i″ and ″CLR [m].i″ instructions.

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

Each line of port A has the capability of waking-up the device.

Each I/O line has a pull-high option. Once a pull-high option is selected, the I/O line has a pull-high resistor connected, otherwise, there are none. Take note that a non-pull-high I/O line operating as an input will be in a floating state.

Pin PA3 is pin-shared with the PFD signal. If the PFD configuration option is selected, the output signal on PA3, if setup as an output, will be the PFD signal generated by the timer/event counter overflow signal. If setup as an input then the pin will retain its input function. Once the PFD configuration option is selected, the PFD output signal is controlled by the PA3 data register. Writing a ″1″ to the PA3 data register will enable the PFD output function and writing ″0″ will force pin PA3 to remain at ″0″. The I/O functions of PA3 are shown below.

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

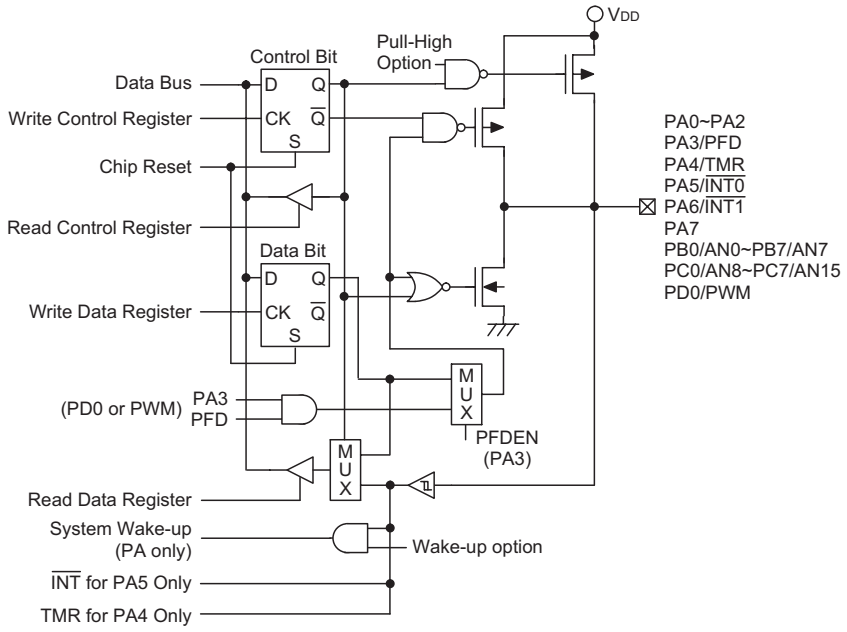
Note: The PFD frequency is the timer/event counter overflow frequency divided by 2.

PA6, PA5 and PA4 are pin-shared with the $\overline{\text{INT0}}$, $\overline{\text{INT1}}$ and TMR pins respectively.

PB and PC can also be used as A/D converter inputs. The A/D function will be described later. There are two SPI interfaces which are shared with pins PD1~PD4, PD7 and PF0~PF2. The SPI function will be described later. There is a PWM function shared with pin PD0. If the PWM function is enabled, the PWM signal will appear on pin PD0 (if PD0 is setup as an output). Writing a ″1″ to the PD0 data register will enable the PWM output function and writing a ″0″ will force the PD0 to remain at ″0″. The I/O functions of PD0 are shown in the table.

I/O Mode	I/P (Normal)	O/P (Normal)	I/P (PWM)	O/P (PWM)
PD0	Logical Input	Logical Output	Logical Input	PWM

It is recommended that unused or not bonded out I/O



Input/Output Ports

lines should be set as output pins using software instructions to minimise power consumption should they be inadvertently setup as floating inputs.

Pulse Width Modulator – PWM

The microcontroller provides a single channel (6+2) bit Pulse Width Modulator output shared with pin PD0. Its data register is known as PWM. The frequency source for the PWM counter comes from f_{SYS}. The PWM register is an eight bit register. If the configuration option selects pin PD0 to be a PWM output, and if the pin is setup as an output by setting bit PDC.0 to ″0″, then writing 1 to the PD0 data register will enable the PWM output function. Writing a ″0″ will force the PD0 to stay at ″0″.

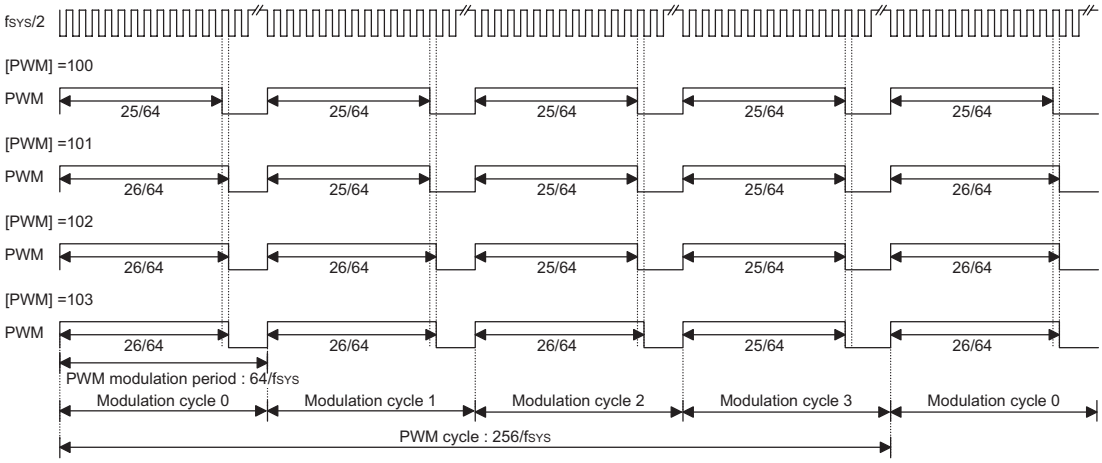
A PWM cycle is divided into four modulation cycles (modulation cycle 0~modulation cycle 3). Each modulation cycle has 64 PWM input clock periods. In a (6+2) bit PWM function, the contents of the PWM register is divided into two groups. Group 1 of the PWM register is denoted by DC which has the value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

In a PWM cycle, the duty cycle of each modulation cycle is shown in the table.

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

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

PWM Modulation Frequency	PWM Cycle Frequency	PWM Cycle Duty
$f_{SYS}/64$	$f_{SYS}/256$	[PWM]/256



PWM

A/D Converter

A 16 channel 8-bit resolution A/D converter is integrated within the microcontroller. The A/D reference voltage is VDD. The A/D converter contains several special registers, which are ADR, ADCR and ACSR. The ADR register is the A/D result register and is read-only. After an A/D conversion has completed, the ADR register is read to obtain the conversion result data. The EOCB flag will also be automatically cleared to indicate the end of conversion. The ADCR register is the A/D converter control register, which selects the analog channel, contains the start A/D conversion control bit and the end of A/D conversion flag. To initiate an A/D conversion, the analog channel is first selected and then the START bit is given a falling edge. When the conversion is complete, the EOCB bit will be cleared and an A/D converter interrupt is generated. The ACSR register selects the A/D clock source as well as selecting which pins are to be used as A/D inputs. Bits 0~3 of ADCR are used to select an analog input channel. There are a total of 16 channels to select. Bits 3~6 of ADSR are used to select which pins on Port B and Port C are setup as normal I/Os or A/D inputs. The EOCB bit in the ADCR registers, is end of A/D conversion flag. This bit can be monitored to check when the A/D conversion has completed. The START bit in the ADCR register is used to initiate A/D conversion process. Providing the START bit with a rising edge will reset and start the A/D conversion. When checking for the end of an A/D conversion, the START bit should remain at ″0″ and the EOCB bit monitored until it is cleared to ″0″ which indicates the end of conversion.

Bit 7 of the ACSR register is used for testing purposes only and should not be used. Bits 1 and bit 0 are used to select the A/D converter clock source.

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

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
D7	D6	D5	D4	D3	D2	D1	D0

ADR (21H) Register

ACS3	ACS2	ACS1	ACS0	Analog Channel
0	0	0	0	AN0
0	0	0	1	AN1
0	0	1	0	AN2
0	0	1	1	AN3
0	1	0	0	AN4
0	1	0	1	AN5
0	1	1	0	AN6
0	1	1	1	AN7
1	0	0	0	AN8
1	0	0	1	AN9
1	0	1	0	AN10
1	0	1	1	AN11
1	1	0	0	AN12
1	1	0	1	AN13
1	1	1	0	AN14
1	1	1	1	AN15

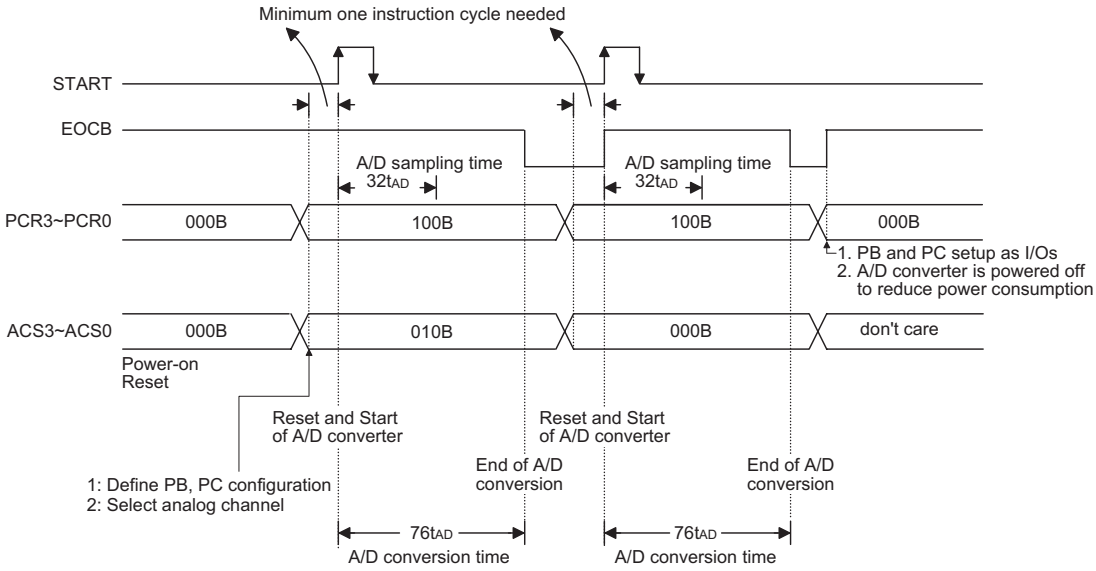
Bit No.	Label	Function
0~3	ACS0~ ACS3	Analog channel selection
4~5	—	Reserved bit
6	EOCB	Indicates end of A/D conversion (read only). (0 = end of A/D conversion)
7	START	Starts the A/D conversion. (0→1→0= start; 0→1= Reset A/D converter and set EOCB to ″1″)

ADCR (22H) Register

Bit No.	Label	Function
0 1	ADCS0 ADCS1	ADCS1,ADCS0 : Selects the A/D converter clock source 0, 0: $f_{SYS}/2$ 0, 1: $f_{SYS}/8$ 1, 0: Undefined 1, 1: Undefined (f_{WDT} for test only)
2	—	Unused bit, read as ″0″.
3~6	PCR0~ PCR3	Port B & Port C configuration selection. If PCR0, PCR1 and PCR2, PCR3 are all zero, the ADC circuit is power off to reduce power consumption
7	TEST	For internal test only, read as ″1″.

ACSR (23H) Register

PCR3	PCR2	PCR1	PCR0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0	PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0
0	0	0	1	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0	PB7	PB6	PB5	PB4	PB3	PB2	PB1	AN0
0	0	1	0	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0	PB7	PB6	PB5	PB4	PB3	PB2	AN1	AN0
0	0	1	1	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0	PB7	PB6	PB5	PB4	PB3	AN2	AN1	AN0
0	1	0	0	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0	PB7	PB6	PB5	PB4	AN3	AN2	AN1	AN0
0	1	0	1	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0	PB7	PB6	PB5	AN4	AN3	AN2	AN1	AN0
0	1	1	0	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0	PB7	PB6	AN5	AN4	AN3	AN2	AN1	AN0
0	1	1	1	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0	PB7	AN6	AN5	AN4	AN3	AN2	AN1	AN0
1	0	0	0	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0	AN7	AN6	AN5	AN4	AN3	AN2	AN1	AN0
1	0	0	1	PC7	PC6	PC5	PC4	PC3	PC2	PC1	AN8	AN7	AN6	AN5	AN4	AN3	AN2	AN1	AN0
1	0	1	0	PC7	PC6	PC5	PC4	PC3	PC2	AN9	AN8	AN7	AN6	AN5	AN4	AN3	AN2	AN1	AN0
1	0	1	1	PC7	PC6	PC5	PC4	PC3	AN10	AN9	AN8	AN7	AN6	AN5	AN4	AN3	AN2	AN1	AN0
1	1	0	0	PC7	PC6	PC5	PC4	AN11	AN10	AN9	AN8	AN7	AN6	AN5	AN4	AN3	AN2	AN1	AN0
1	1	0	1	PC7	PC6	PC5	AN12	AN11	AN10	AN9	AN8	AN7	AN6	AN5	AN4	AN3	AN2	AN1	AN0
1	1	1	0	PC7	PC6	AN13	AN12	AN11	AN10	AN9	AN8	AN7	AN6	AN5	AN4	AN3	AN2	AN1	AN0
1	1	1	1	AN15	AN14	AN13	AN12	AN11	AN10	AN9	AN8	AN7	AN6	AN5	AN4	AN3	AN2	AN1	AN0



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

A/D Conversion Timing

Example: using EOC Polling Method to detect end of conversion.

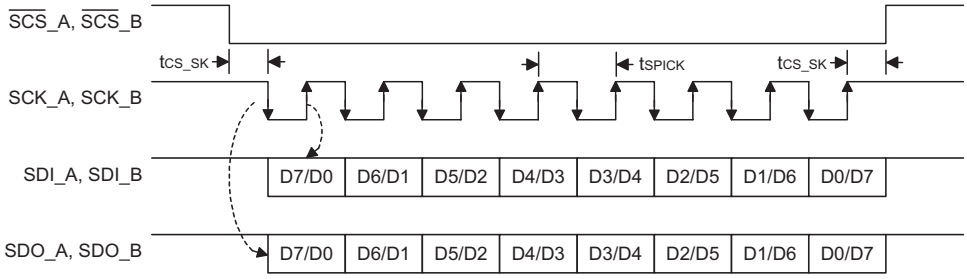
```
    clr     INTC                ; disable A/D interrupt in interrupt control register
    mov a,  0010B
    mov     ADCR,a              ; setup ADCR register to configure Port PB0~PB3 as A/D inputs and select
                                ; AN0 to be connected to the A/D converter
    mov a,  00000001B
    mov     ACSR,a              ; setup the ACSR register to select fSYS/8 as the A/D clock

Start_conversion:
    clr .7
    set ADCR.7                  ; reset A/D
    clr ADCR.7                  ; start A/D
Polling_EOC:
    sz ADCR.6                   ; poll the ADCR register EOC bit to detect end of A/D conversion
    jmp     polling_EOC         ; continue polling
    mov a,  ADR                 ; read conversion result from the high byte ADRH register
    mov     adr_buffer,a        ; save result to user defined register
            :
            :
    jmp     start_conversion    ; start next A/D conversion
```

Example: using Interrupt method to detect end of conversion.

```
    set     INTC                ; enable A/D interrupt in interrupt control register
    mov a,  0010B
    mov     ADCR,a              ; setup ADCR register to configure Port PB0~PB3 as A/D inputs and select
                                ; AN0 to be connected to the A/D converter
    mov a,  00000001B
    mov     ACSR,a              ; setup the ACSR register to select fSYS/8 as the A/D clock
            :
Start_conversion:
    clr     ADCR.7
    set     ADCR.7              ; reset A/D
    clr     ADCR.7              ; start A/D
            :
            :
                                ; interrupt service routine
EOC_service routine:
    mov a_buffer,a              ; save ACC to user defined register
    mov a,ADR                   ; read conversion result from the high byte ADRH register
    mov adr_buffer,a            ; save result to user defined register
    clr .7
    set ADCR.7                  ; reset A/D
    clr ADCR.7                  ; start A/D
    mov a,a_buffer              ; restore ACC from temporary storage
    reti
```

SPI Serial Interface

There are two SPI interfaces, with each interface containing four basic signals and pins. These are SDI (serial data input), SDO (serial data output), SCK (serial clock) and $\overline{SCS}$ (slave select pin). Note that each of these pin names will be suffixed with either an A or B to denote which SCI interface is being used, however to minimise repetition, this will not be



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

Note: "U" means unchanged.

mentioned in the description.

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

- SBCR_A, SBCR_B: Serial bus control register
  Bit7 (CKS) clock source selection: $f_{SIO}=f_{SYS}/4$, select as 0
  Bit6 (M1), Bit5 (M0) master/slave mode and baud rate selection
  M1, M0: 00 → MASTER MODE, BAUD RATE= f_{SIO}
  01 → MASTER MODE, BAUD RATE= $f_{SIO}/4$
  10 → MASTER MODE, BAUD RATE= $f_{SIO}/16$
  11 → SLAVE MODE
- Bit4 (SBEN) → serial bus enable/disable (1/0)
  - Enable: ($\overline{SCS}$ dependent on CSEN bit)
    Disable → enable: SCK, SDI, SDO, $\overline{SCS}$= 0 (SCKB= ″0″) and waiting for writing data to SBDR (TXRX buffer)
    Master mode: write data to SBDR (TXRX buffer) start transmission/reception automatically
    Master mode: when the data has been transferred, set TRF
    Slave mode: when an SCK (and $\overline{SCS}$ dependent on CSEN) is received, data in the TXRX buffer is shifted-out and data on SDI is shifted-in.
  - Disable: SCK ($\overline{SCK}$), SDI, SDO, $\overline{SCS}$ floating
    Bit3 (MLS) → MSB or LSB (1/0) shift first control bit
    Bit2 (CSEN) → serial bus selection signal enable/disable ($\overline{SCS}$), when CSEN=0, SCSB is floating.
    Bit1 (WCOL) → this bit is set to 1 if data is written to the SBDR register (TXRX buffer) when data is transferred, writing will be ignored if data is written to SBDR (TXRX buffer) when data is transferred.
    Bit0 (TRF) → data transferred or data received used to generate an interrupt.
    Note: data reception is still in operation when the MCU enters the Power-down mode.
- SBDR_A, SBDR_B: Serial bus data register
  Data written to SBDR → write data to the TXRX buffer only
  Data read from SBDR → read from SBDR only
  Operating Mode description:
  Master transmitter: clock transmission and data I/O started by writing to SBDR
  Master clock transmission initiated by writing to SBDR
  Slave transmitter: data I/O started by clock reception
  Slave receiver: data I/O started by clock reception

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

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

Operation of Serial Interface

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

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

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

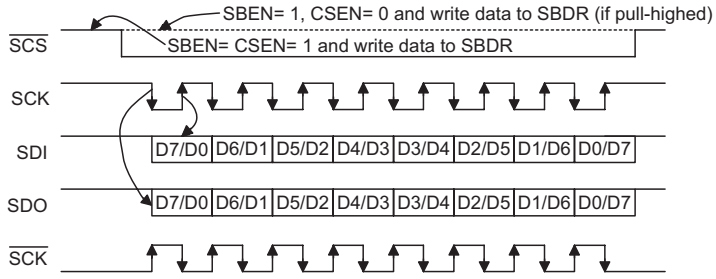
MLS: MSB or LSB first selection.

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

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

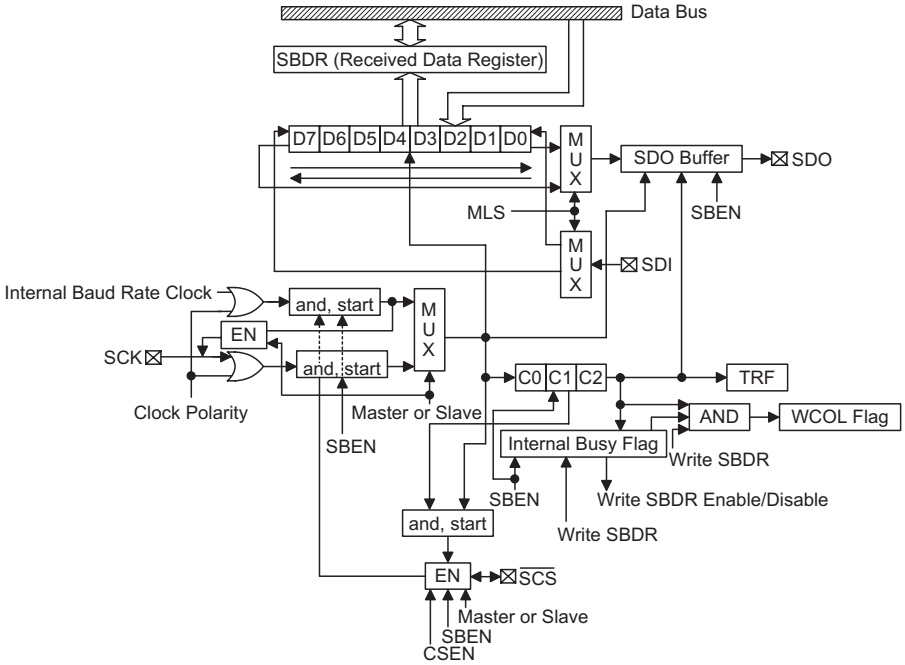
SBEN= 0 → serial bus disabled; $\overline{SCS}$=SDO=1, SDI=SCK= floating in master mode, SDI=SDO=SCK= floating, $\overline{SCS}$=1 in slave mode.

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



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

Note: "u" means unchanged.



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

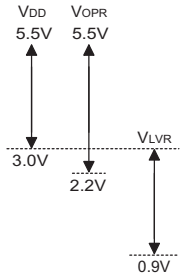
Low Voltage Reset – LVR

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 when changing a battery, the LVR will automatically reset the device internally.

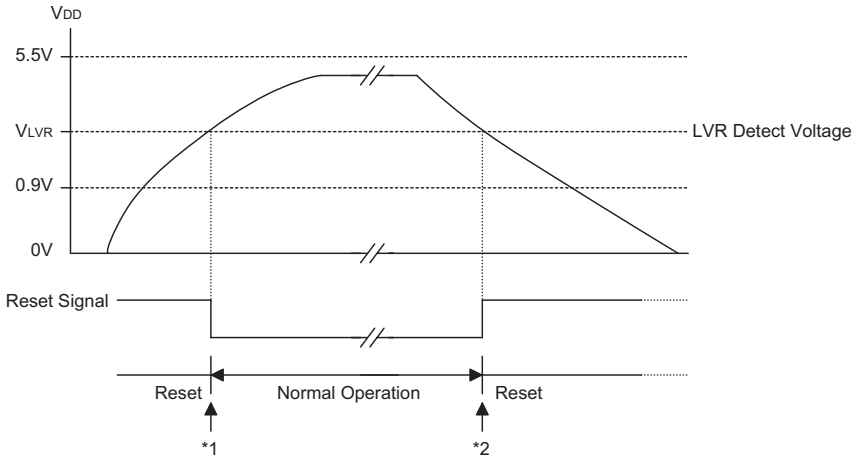
The LVR includes the following specifications:

- The low voltage (0.9V~V_{LVR}) has to remain in its original state for longer than 1ms. If the low voltage state does not exceed 1ms, the LVR will ignore it and will not perform a reset function.
- The LVR uses an "OR" function with the external $\overline{RES}$ signal to perform a 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 starting normal operation.

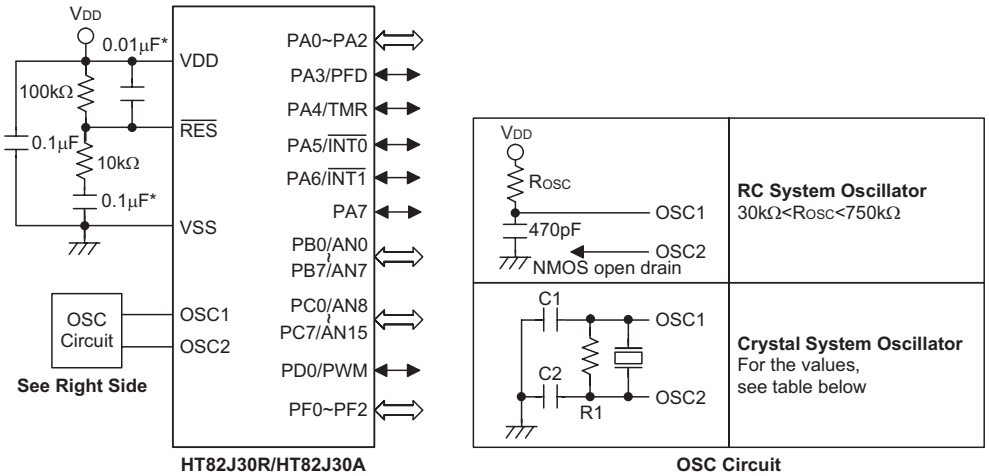
*2: Since a low voltage has to be maintained in its original state for longer than 1ms, therefore a 1ms delay enters the reset mode.

Options

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

Items	Options
1	Lock or unlock (1/0)
2	PA0~PA7 wake-up enable or disable (1/0) options
3	WDT clock source (fS) : WDTOSC/ fTID (0/1)
4	CLR WDT instruction(s): one or two clear WDT instruction(s) (1/0)
5	WDT enable or disable (0/1)
6	PA pull-high enable or disable (1/0)
7	PB pull-high enable or disable (1/0) by nibble
8	PC pull-high enable or disable (1/0) by nibble
9	PD pull-high enable or disable (1/0) by nibble
10	PF pull-high enable or disable (1/0) by nibble
11	PWM enable or disable
12	PFD enable or disable
13	System oscillators 0/1: RC/crystal
14	Low voltage reset: Enable or disable
15	LVR voltage: 3.0V/3.8V (0/1)
16	SIO_A (Serial Interface) enable or disable (default disable)
17	SIO_A_ CPOL: Clock polarity 1/0 : clock polarity rising or falling edge (default falling edge)
18	SIO_A_WCOL: Enable or disable (default disable)
19	SIO_A_CSEN: Enable or disable, CSEN mask option is used to enable or disable software CSEN function (default disable)
20	PD4, PB2, PB3, PD7 CMOS or NMOS output (default CMOS)
21	SIO_B (Serial Interface) enable or disable (default disable)
22	SIO_ B_CPOL: Clock polarity 1/0: clock polarity rising/falling edge (default falling edge)
23	SIO_B_WCOL: Enable or disable (default disable)
24	SIO_B_CSEN: Enable or disable, CSEN mask option is used to enable or disable software CSEN function (default disable)

Application Circuits



Note: The resistance and capacitance for the reset circuit should be designed to ensure that VDD is stable and remains within a valid range of the operating voltage before bringing RES high.

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

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

Crystal or Resonator	C1, C2	R1
4MHz Crystal	0pF	10kΩ
4MHz Resonator (3 pins)	0pF	12kΩ
4MHz Resonator (2 pins)	10pF	12kΩ
3.58MHz Crystal	0pF	10kΩ
3.58MHz Resonator (2 pins)	25pF	10kΩ
2MHz Crystal & Resonator (2 pins)	25pF	10kΩ
1MHz Crystal	35pF	27kΩ
480kHz Resonator	300pF	9.1kΩ
455kHz Resonator	300pF	10kΩ
429kHz Resonator	300pF	10kΩ
The function of the resistor R1 is to ensure that the oscillator will switch off should low voltage conditions occur. Such a low voltage, as mentioned here, is one which is less than the lowest value of the MCU operating voltage. Note however that if the LVR is enabled then R1 can be removed.		

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.

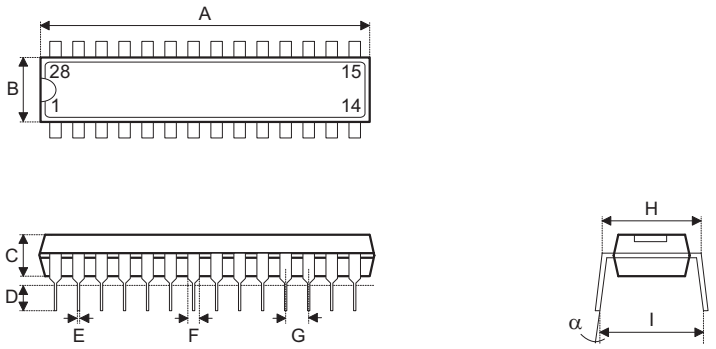
Operation ACC ← ACC ″XOR″ x

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

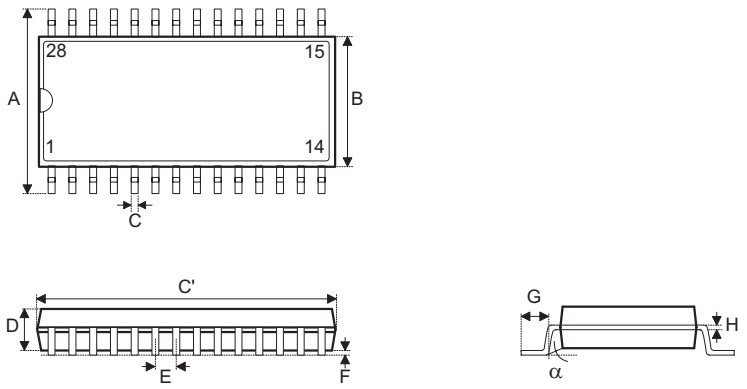
Package Information

28-pin SKDIP (300mil) Outline Dimensions



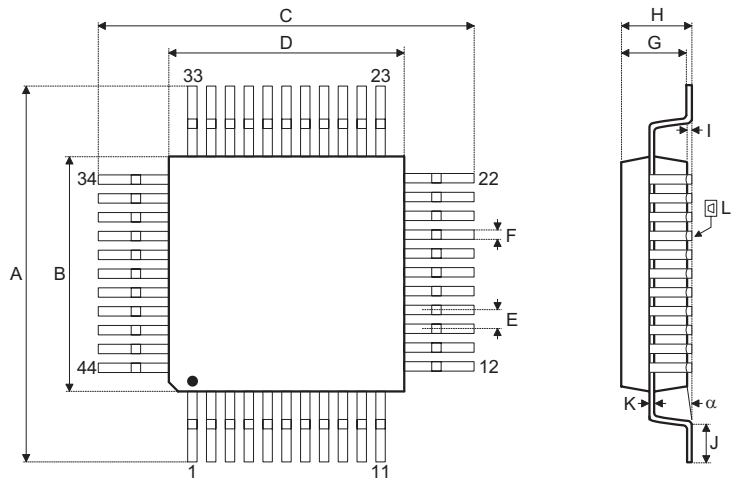
Symbol	Dimensions in mil		
	Min.	Nom.	Max.
A	1375	—	1395
B	278	—	298
C	125	—	135
D	125	—	145
E	16	—	20
F	50	—	70
G	—	100	—
H	295	—	315
I	330	—	375
α	0°	—	15°

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°

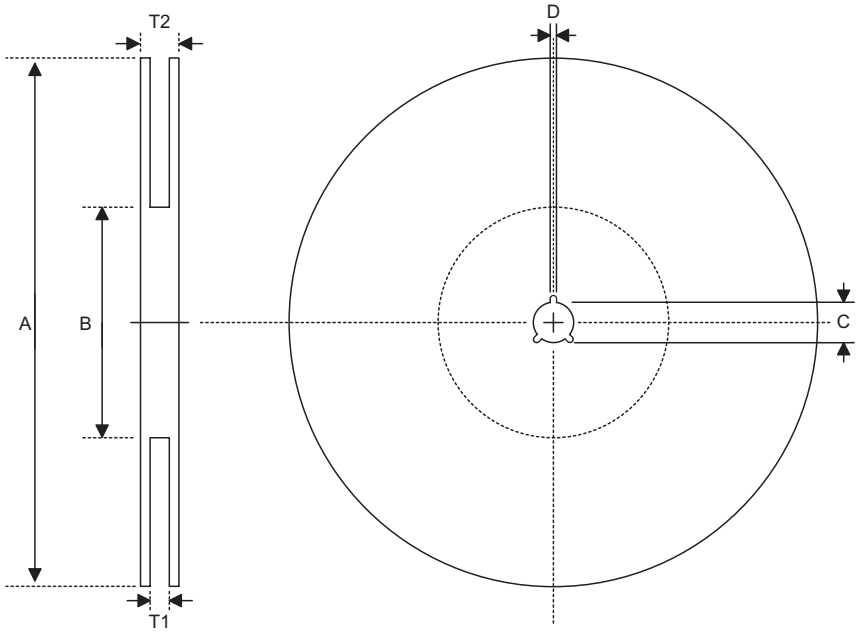
44-pin QFP (10×10) Outline Dimensions



Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	13	—	13.4
B	9.9	—	10.1
C	13	—	13.4
D	9.9	—	10.1
E	—	0.8	—
F	—	0.3	—
G	1.9	—	2.2
H	—	—	2.7
I	0.25	—	0.5
J	0.73	—	0.93
K	0.1	—	0.2
L	—	0.1	—
α	0°	—	7°

Product Tape and Reel Specifications

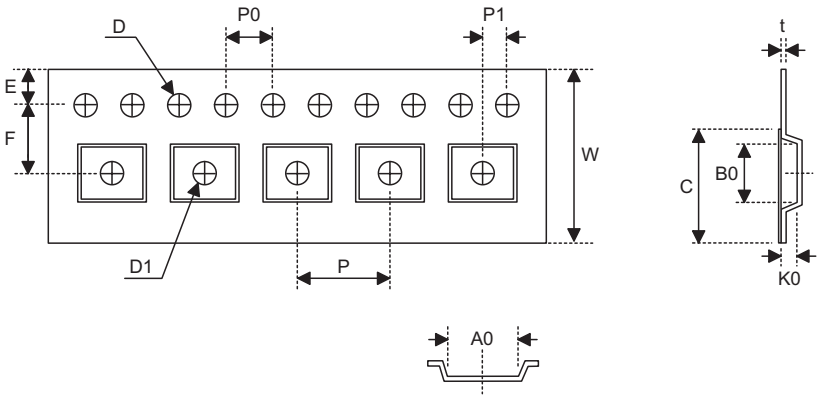
Reel Dimensions



SOP 28W (300mil)

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

Carrier Tape Dimensions



SOP 28W (300mil)

Symbol	Description	Dimensions in mm
W	Carrier Tape Width	24±0.3
P	Cavity Pitch	12±0.1
E	Perforation Position	1.75±0.1
F	Cavity to Perforation (Width Direction)	11.5±0.1
D	Perforation Diameter	1.5+0.1
D1	Cavity Hole Diameter	1.5+0.25
P0	Perforation Pitch	4±0.1
P1	Cavity to Perforation (Length Direction)	2±0.1
A0	Cavity Length	10.85±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

Copyright © 2006 by HOLTEK SEMICONDUCTOR INC.

The information appearing in this Data Sheet is believed to be accurate at the time of publication. However, Holtek assumes no responsibility arising from the use of the specifications described. The applications mentioned herein are used solely for the purpose of illustration and Holtek makes no warranty or representation that such applications will be suitable without further modification, nor recommends the use of its products for application that may present a risk to human life due to malfunction or otherwise. Holtek's products are not authorized for use as critical components in life support devices or systems. Holtek reserves the right to alter its products without prior notification. For the most up-to-date information, please visit our web site at http://www.holtek.com.tw.