HT46RU22
A/D Type 8-Bit MCU

Technical Document

- Tools Information
- FAQs
- Application Note
  - HA0004E HT48 & HT46 MCU UART Software Implementation Method
  - HA0005E Controlling the I^2C bus with the HT48 & HT46 MCU Series
  - HA0013E HT48 & HT46 LCM Interface Design
  - HA0047E An PWM application example using the HT46 series of MCUs
  - HA0075E MCU Reset and Oscillator Circuits Application Note

Features

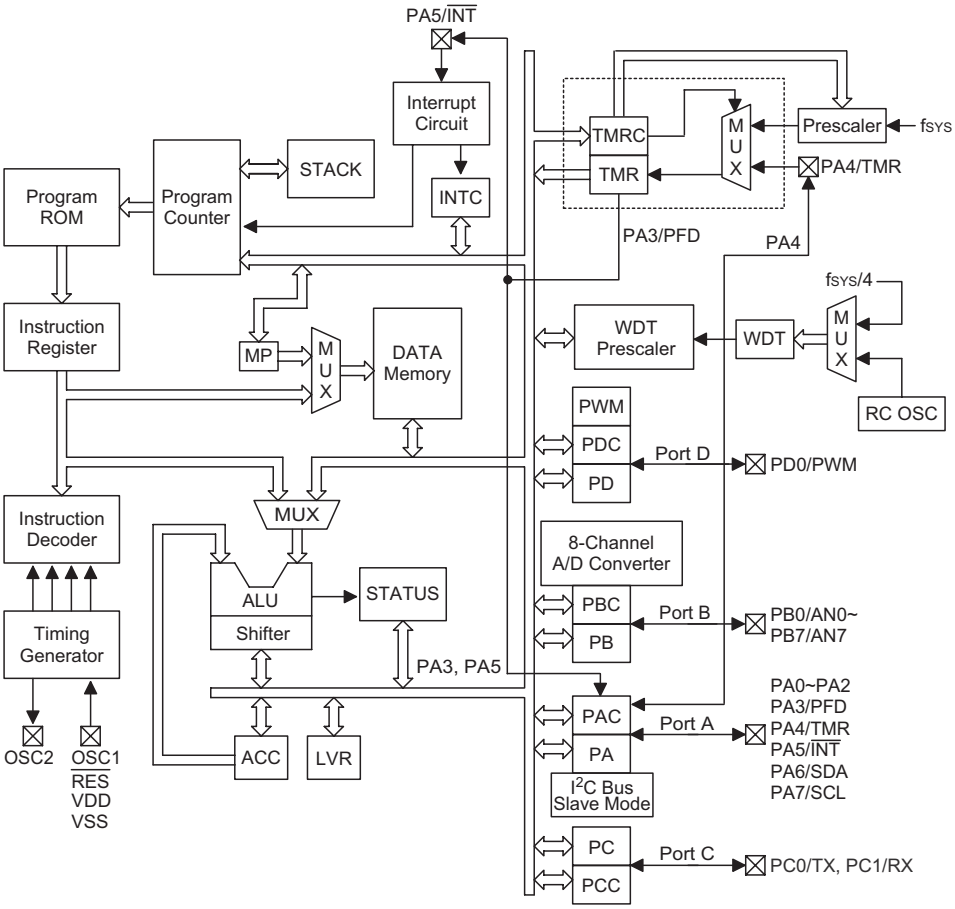
- Operating voltage:
  f_{SYS}=4MHz: 2.2V~5.5V
  f_{SYS}=8MHz: 3.3V~5.5V
- 19 bidirectional I/O lines (max.)
- 1 interrupt input shared with an I/O line
- 8-bit programmable timer/event counter with overflow interrupt and 7-stage prescaler
- On-chip crystal and RC oscillator
- Watchdog Timer
- 2048×14 program memory
- 64×8 data memory RAM
- Supports PFD for sound generation
- HALT function and wake-up feature reduce power consumption
- Up to 0.5μs instruction cycle with 8MHz system clock at V_{DD}=5V
- 6-level subroutine nesting
- 8 channels 9-bit resolution A/D converter
- 1-channel 8-bit PWM output shared with one I/O line
- Universal Asynchronous Receiver Transmitter (UART)
- Bit manipulation instruction
- 14-bit table read instruction
- 63 powerful instructions
- All instructions in one or two machine cycles
- Low voltage reset function
- I^2C Bus (slave mode)
- 24-pin SKDIP/SOP/SSOP package

General Description

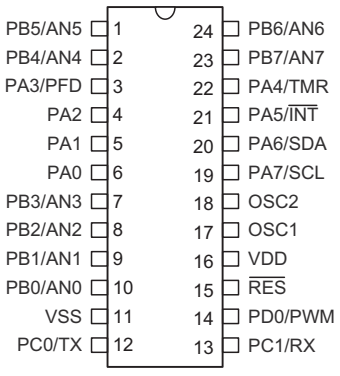
The HT46RU22 are 8-bit, high performance, RISC architecture microcontroller devices specifically designed for A/D applications that interface directly to analog signals, such as those from sensors.

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

Block Diagram



Pin Assignment



HT46RU22
– 24 SKDIP-A/SOP-A/SSOP-A

Pad Description

Pad Name	I/O	Options	Description
PA0~PA2 PA3/PFD PA4/TMR PA5/$\overline{INT}$ PA6/SDA PA7/SCL	I/O	Pull-high Wake-up PA3 or PFD I/O or Serial Bus	Bidirectional 8-bit input/output port. Each bit can be configured as wake-up input by options. Software instructions determine the CMOS output or Schmitt trigger input with or without pull-high resistor (determined by pull-high options: bit option). The PFD, TMR and $\overline{INT}$ are pin-shared with PA3, PA4 and PA5, respectively. Once the I^2C Bus function is used, the internal registers related to PA6 and PA7 can not be used.
PB0/AN0 PB1/AN1 PB2/AN2 PB3/AN3 PB4/AN4 PB5/AN5 PB6/AN6 PB7/AN7	I/O	Pull-high	Bidirectional 8-bit input/output port. Software instructions determine the CMOS output, Schmitt trigger input with or without pull-high resistor (determined by pull-high option: port option) or A/D input. Once a PB line is selected as an A/D input (by using software control), the I/O function and pull-high resistor are disabled automatically.
PC0/TX PC1/RX	I/O	Pull-high	Bidirectional 2-bit input/output port. Software instructions determine the CMOS output, Schmitt trigger input with or without pull-high resistor (determine by pull-high option: bit option). TX and RX are pin-shared with PC0 and PC1, once the UART Bus function is used, the internal registers related to PC0 and PC1 can not be used. Software instructions determine the UART function to be used.
PD0/PWM	I/O	Pull-high I/O or PWM	Bidirectional 1-bit input/output port. Software instructions determine the CMOS output, Schmitt trigger input with or without a pull-high resistor (determined by pull-high option: port option). The PWM output function is pin-shared with PD0 (dependent on PWM options).
OSC1 OSC2	I O	Crystal or RC	OSC1, OSC2 are connected to an RC network or a Crystal (determined by options) for the internal system clock. In the case of RC operation, OSC2 is the output terminal for 1/4 system clock.
$\overline{RES}$	I	—	Schmitt trigger reset input. Active low.
VSS	—	—	Negative power supply, ground.
VDD	—	—	Positive power supply
TEST1 TEST2 TEST3	I	—	TEST mode input pin It disconnects in normal operation

Absolute Maximum Ratings

Supply Voltage ..... V_{SS}–0.3V to V_{SS}+6.0V
Input Voltage ..... V_{SS}–0.3V to V_{DD}+0.3V
I_{OL} Total 150mA
Total Power Dissipation 500mW

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

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

D.C. Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	Operating Voltage	—	f_{SYS}=4MHz	2.2	—	5.5	V
		—	f_{SYS}=8MHz	3.3	—	5.5	V
I_{DD1}	Operating Current (Crystal OSC)	3V	No load, f_{SYS}=4MHz ADC Off, UART Off	—	0.6	1.5	mA
		5V		—	2	4	mA
I_{DD2}	Operating Current (RC OSC)	3V	No load, f_{SYS}=4MHz ADC Off, UART Off	—	0.8	1.5	mA
		5V		—	2.5	4	mA
I_{DD3}	Operating Current (Crystal OSC, RC OSC)	3V	No load, f_{SYS}=4MHz, ADC Off, UART On	—	1	2	mA
		5V		—	3	6	mA
I_{DD4}	Operating Current (Crystal OSC, RC OSC)	5V	No load, f_{SYS}=8MHz, ADC Off, UART Off	—	4	8	mA
I_{DD5}	Operating Current (Crystal OSC, RC OSC)	5V	No load, f_{SYS}=8MHz, ADC Off, UART On	—	5	10	mA
I_{STB1}	Standby Current (WDT Enabled)	3V	No load, system HALT	—	—	5	μA
		5V		—	—	10	μA
I_{STB2}	Standby Current (WDT Disabled)	3V	No load, system HALT	—	—	1	μA
		5V		—	—	2	μA
V_{IL1}	Input Low Voltage for I/O Ports, 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	—	—	2.7	3	3.3	V
I_{OL}	I/O Port Sink Current	3V	V_{OL}=0.1V_{DD}	4	8	—	mA
		5V	V_{OL}=0.1V_{DD}	10	20	—	mA
I_{OH}	I/O Port Source Current	3V	V_{OH}=0.9V_{DD}	–2	–4	—	mA
		5V	V_{OH}=0.9V_{DD}	–5	–10	—	mA
R_{PH}	Pull-high Resistance	3V	—	20	60	100	kΩ
		5V	—	10	30	50	kΩ
V_{AD}	A/D Input Voltage	—	—	0	—	V_{DD}	V
E_{AD}	A/D Conversion Error	—	—	—	±0.5	±1	LSB
I_{ADC}	Additional Power Consumption if A/D Converter is Used	3V	—	—	0.5	1	mA
		5V		—	1.5	3	mA

A.C. Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
f_{SYS}	System Clock (Crystal OSC, RC OSC)	—	2.2V~5.5V	400	—	4000	kHz
		—	3.3V~5.5V	400	—	8000	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_{RES}	External Reset Low Pulse Width	—	—	1	—	—	μs
t_{SST}	System Start-up Timer Period	—	Wake-up from HALT	—	1024	—	*t_{SYS}
t_{LVR}	Low Voltage Width to Reset	—	—	0.25	1	2	ms
t_{INT}	Interrupt Pulse Width	—	—	1	—	—	μs
t_{AD}	A/D Clock Period	—	—	1	—	—	μs
t_{ADC}	A/D Conversion Time	—	—	—	76	—	t_{AD}
t_{ADCS}	A/D Sampling Time	—	—	—	32	—	t_{AD}
t_{IIC}	I^2C Bus Clock Period	—	Connect to external pull-high resistor 2kΩ	64	—	—	*t_{SYS}

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

Functional Description

Execution Flow

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

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

Program Counter – PC

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

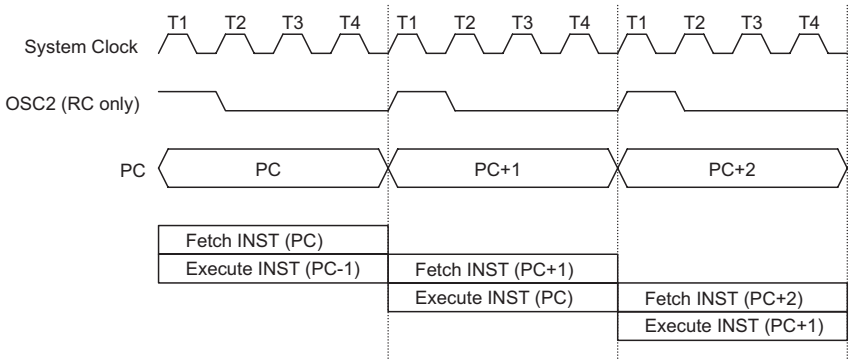
After accessing a program memory word to fetch an instruction code, the contents of the program counter are incremented by 1. The program counter then points to the memory word containing the next instruction code.

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

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

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

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



Execution Flow

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

Program Counter

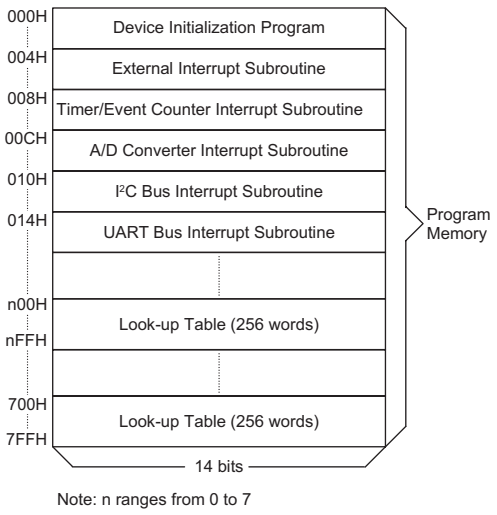
Note: *10~*0: Program counter bits
#10~#0: Instruction code bits
S10~S0: Stack register bits
@7~@0: PCL bits

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 2048×14 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 $\overline{INT}$ 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.
- Location 00CH
  This area is reserved for the A/D converter interrupt service program. If an A/D converter interrupt results from an end of A/D conversion, and if the interrupt is enabled and the stack is not full, the program begins execution at location 00CH.
- Location 010H
  This area is reserved for the I^2C Bus interrupt service program. If the I^2C Bus interrupt resulting from a slave address is match or completed 1 byte of data transfer, and if the interrupt is enable and the stack is not full, the program begins execution at location 010H.
- Location 014H
  This area is reserved for the UART interrupt service program. If the UART interrupt resulting from transmission/reception is completed, and if the interrupt is enable and the stack is not full, the program begins execution at location 014H.
- Table location
  Any location in the ROM space can be used as look-up tables. The instructions "TABRDC [m]" (the current page, 1 page=256 words) and "TABRDL [m]" (the last page) transfer the contents of the lower-order byte to the specified data memory, and the higher-order byte to TBLH (08H). Only the destination of the lower-order byte in the table is well-defined, the other bits of the table word are transferred to the lower portion of TBLH, and the remaining 2 bit is read as "0". The Table Higher-order byte register (TBLH) is read only. The table pointer (TBLP) is a read/write register (07H), which indicates the table location. Before accessing the table, the location must be placed in TBLP. The TBLH is read only and cannot be restored. If the main routine and the ISR (Interrupt Service Routine) both employ the table read instruction, the contents of the TBLH in the main routine are likely to be changed by the table read instruction used in the ISR. Errors can occur. In other words, using the table read instruction in the main routine and the ISR simultaneously should be avoided. However, if the table read instruction has to be applied in both the main routine and the ISR, the interrupt is supposed to be disabled prior to the table read instruction. It will not be enabled until the TBLH has been backed up. All table related instructions require two cycles to complete the operation. These areas may function as normal program memory depending upon the requirements.



Note: n ranges from 0 to 7

Program Memory

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

Table Location

Note: *10~*0: Table location bits
P10~P8: Current program counter bits
@7~@0: Table pointer bits

Stack Register – STACK

This is a special part of the memory which is used to save the contents of the program counter (PC) 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 acknowledgment, the contents of the program counter are pushed onto the stack. At the end of a subroutine or an interrupt routine, signaled by a return instruction (RET or RETI), the program counter is restored to its previous value from the stack. After a chip reset, the SP will point to the top of the stack.

If the stack is full and a non-masked interrupt takes place, the interrupt request flag will be recorded but the acknowledgment will be inhibited. When the stack pointer is decremented (by RET or RETI), the interrupt will be serviced. This feature prevents stack overflow allowing the programmer to use the structure more easily. In a similar case, if the stack is full and a ″CALL″ is subsequently executed, stack overflow occurs and the first entry will be lost (only the most recent 6 return addresses are stored).

Data Memory – RAM

The data memory is designed with 92×8 bits. The data memory is divided into two functional groups: special function registers and general purpose data memory (64×8). Most are read/write, but some are read only.

The remaining space before the 40H is reserved for future expanded usage and reading these locations will get ″00H″. The general purpose data memory, addressed from 40H to 7FH, is used for data and control information under instruction commands. All of the data memory areas can handle arithmetic, logic, increment, decrement and rotate operations directly. Except for some dedicated bits, each bit in the data memory can be set and reset by ″SET [m].i″ and ″CLR [m].i″. They are also indirectly accessible through memory pointer register (MP;01H).

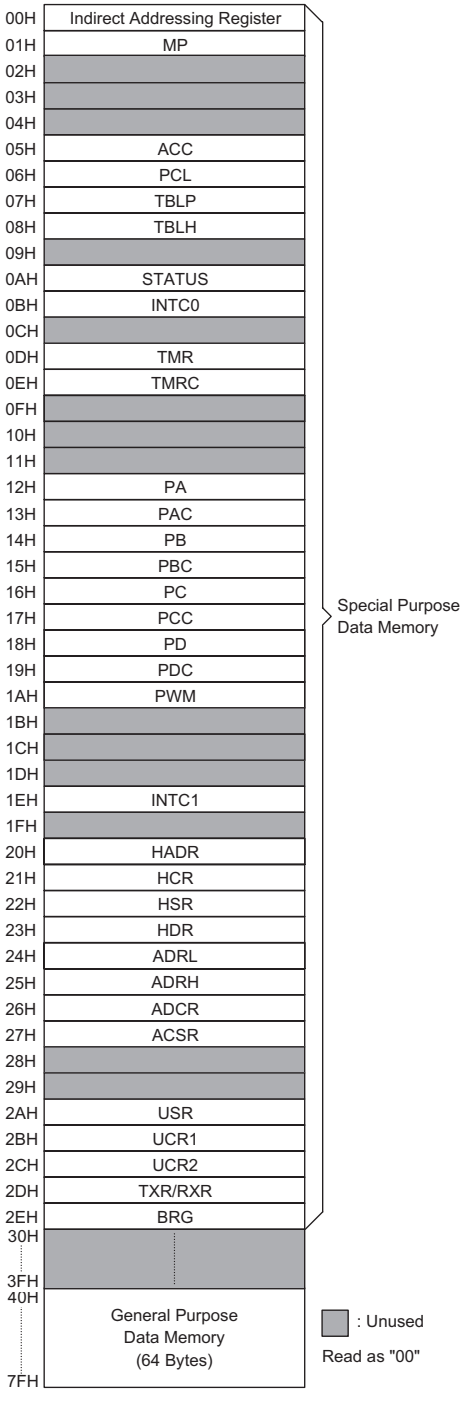
Indirect Addressing Register

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 (01H). Reading location 00H itself indirectly will return the result 00H. Writing indirectly results in no operation.

The memory pointer register MP (01H) is a 7-bit register. The bit 7 of MP is undefined and reading will return the result ″1″. Any writing operation to MP will only transfer the lower 7-bit data to MP.

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.



RAM Mapping

Arithmetic and Logic Unit – ALU

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

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

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

Status Register – STATUS

This 8-bit register (0AH) contains the 0 flag (Z), carry flag (C), auxiliary carry flag (AC), overflow flag (OV), power down flag (PDF), and watchdog time-out flag (TO). It also records the status information and controls the operation sequence.

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

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

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

Interrupt

The device provides an external interrupt, an internal timer/event counter interrupt, the A/D converter interrupt, the I^2C Bus interrupts and an UART interrupt. The interrupt control register 0 (INTC0;0BH) and interrupt control register 1 (INTC1;1EH) contains the interrupt control bits to set the enable or 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 flag is recorded. If a certain interrupt requires servicing within the service routine, the EMI bit and the corresponding bit of INTC0 and INTC1 may be set to allow interrupt nesting. If the stack is full, the interrupt request will not be acknowledged, even if the related interrupt is enabled, until the SP is decremented. If immediate service is desired, the stack must be prevented from becoming full.

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

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 0; 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

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

The internal timer/event counter interrupt is initialized by setting the timer/event counter interrupt request flag (TF; bit 5 of INTC0), caused by a timer overflow. When the interrupt is enabled, the stack is not full and the TF bit is set, a subroutine call to location 08H will occur. The related interrupt request flag (TF) will be reset and the EMI bit cleared to disable further interrupts.

The A/D converter interrupt is initialized by setting the A/D converter request flag (ADF; bit 6 of INTC0), caused by an end of A/D conversion. When the interrupt is enabled, the stack is not full and the ADF is set, a subroutine call to location 0CH will occur. The related interrupt request flag (ADF) will be reset and the EMI bit cleared to disable further interrupts.

The I^2C Bus interrupt is initialized by setting the I^2C Bus interrupt request flag (HIF; bit 4 of INTC1), caused by a slave address match (HAAS="1") or 1 byte of data transfer is completed. When the interrupt is enabled, the stack is not full and the HIF bit is set, a subroutine call to location 10H will occur. The related interrupt request flag (HIF) will be reset and the EMI bit cleared to disable further interrupts.

The UART Bus interrupt is initialized by setting the UART Bus interrupt request flag, URF; bit 7 of the INTC1 register, caused by transmit data register empty (TXIF), received data available(RXIF), transmission idle (TIDLE), Over run error (OERR) or Address detected. When the interrupt is enabled, the stack is not full and the TXIF, RXIF, TIDLE, OERR bit is set or an address is detected, a subroutine call to location 014H will occur. The related interrupt request flag, URF, will be reset and the EMI bit cleared to disable further interrupts.

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

Bit No.	Label	Function
0	EMI	Controls the master (global) interrupt (1= enabled; 0= disabled)
1	EEI	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	External interrupt request flag (1= active; 0= inactive)
5	TF	Internal timer/event counter request flag (1= active; 0= inactive)
6	ADF	A/D converter request flag (1= active; 0= inactive)
7	—	For test mode used only. Must be written as "0"; otherwise may result in unpredictable operation.

INTC0 (0BH) Register

Bit No.	Label	Function
0	EHI	Controls the I^2C Bus interrupt (1=enabled; 0=disabled)
1	EURI	Control the UART interrupt (1=enable ; 0=disable)
2~3, 6~7	—	Unused bit, read as "0"
4	HIF	I^2C Bus interrupt request flag (1=active; 0=inactive)
5	URF	UART request flag (1=active; 0:inactive)

INTC1 (1EH) Register

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

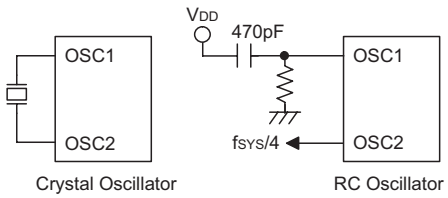
Interrupt Source	Priority	Vector
External Interrupt	1	04H
Timer/Event Counter Overflow	2	08H
A/D Converter Interrupt	3	0CH
I^2C Bus Interrupt	4	10H
UART Interrupt	5	14H

Once the interrupt request flags, composed of TF, EIF, ADF and HIF, are set, they will remain in the INTC0 and INTC1 register until the interrupts are serviced or cleared by a software instruction.

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

Oscillator Configuration

There are two oscillator circuits in the microcontroller.



System Oscillator

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

If an RC oscillator is used, an external resistor between OSC1 and VSS is required and the resistance must range from 24kΩ to 1MΩ. The system clock, divided by 4, is available on OSC2 with pull-high resistor, which can be used to synchronize external logic. The RC oscillator provides the most cost effective solution. However, the frequency of oscillation may vary with VDD, temperatures and the chip itself due to process variations. It is, therefore, not suitable for timing sensitive operations where an accurate oscillator frequency is desired.

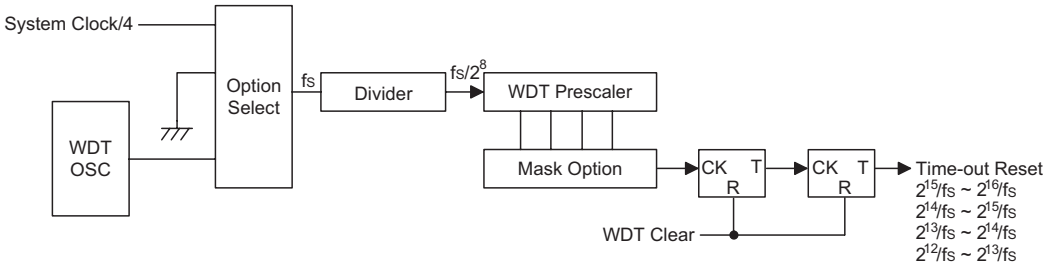
If the Crystal oscillator is used, a crystal across OSC1 and OSC2 is needed to provide the feedback and phase shift required for the oscillator, and no other external components are required. Instead of a crystal, a resonator can also be connected between OSC1 and OSC2 to get a frequency reference, but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

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

Watchdog Timer – WDT

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

Once an internal WDT oscillator (RC oscillator with period 65μs at 5V normally) is selected, it is divided by 2^{12}~2^{15} (by option to get the WDT time-out period). The minimum period of WDT time-out period is about 300ms~600ms. This time-out period may vary with tem-



Watchdog Timer

perature, VDD and process variations. By selection the WDT options, longer time-out periods can be realized. If the WDT time-out is selected 2^{15}, the maximum time-out period is divided by 2^{15}~2^{16}about 2.1s~4.3s.

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

The WDT overflow under normal operation will initialize "chip reset" and set the status bit TO. Whereas in the HALT mode, the overflow will initialize a "warm reset" only the program counter and stack pointer are reset to 0. To clear the contents of WDT, three methods are adopted; external reset (a low level to $\overline{RES}$), software instructions, or a HALT instruction. The software instructions include CLR WDT and the other set – CLR WDT1 and CLR WDT2. Of these two types of instruction, only one can be active depending on the options – "CLR WDT times selection option". If the "CLR WDT" is selected (i.e. CLRWDT times equal 1), any execution of the CLR WDT instruction will clear the WDT. In case "CLR WDT1" and "CLR WDT2" are chosen (i.e. CLRWDT times equal two), these two instructions must be executed to clear the WDT; otherwise, the WDT may reset the chip because of time-out.

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

Power Down Operation – HALT

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

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

The system can leave the HALT mode by means of an external reset, an interrupt, an external falling edge signal on port A or a WDT overflow. An external reset causes a device initialization and the WDT overflow performs a "warm reset". After the TO and PDF flags are examined, the reason for chip reset can be determined.

The PDF flag is cleared by system power-up or executing the "CLR WDT" instruction and is set when executing the "HALT" instruction. The TO flag is set if the WDT time-out occurs, and causes a wake-up that only resets the program counter and stack pointer; the others keep their original status.

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

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

Reset

There are three ways in which a reset can occur:

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

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

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

Note: "u" means unchanged

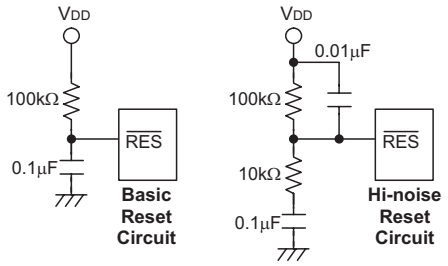
To guarantee that the system oscillator is started and stabilized, the SST (System Start-up Timer) provides an extra-delay of 1024 system clock pulses when the system reset (power-up, WDT time-out or $\overline{RES}$ reset) or the system awakes from the HALT state.

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

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

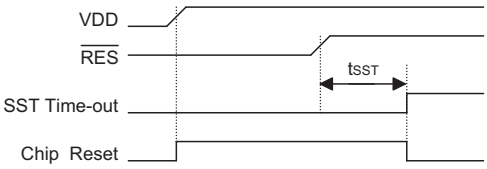
The functional unit chip reset status are shown below.

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

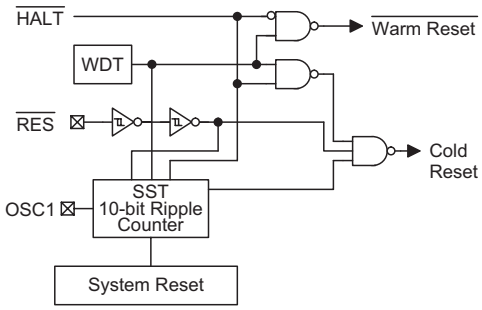


Reset Circuit

Note: Most applications can use the Basic Reset Circuit as shown, however for applications with extensive noise, it is recommended to use the Hi-noise Reset Circuit.



Reset Timing Chart



Reset Configuration

The registers states are summarized in the following table.

Register	Reset (Power On)	WDT Time-out (Normal Operation)	$\overline{RES}$ Reset (Normal Operation)	$\overline{RES}$ Reset (HALT)	WDT Time-out (HALT)*
TMR	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMRC	00-0 1000	00-0 1000	00-0 1000	00-0 1000	uu-u uuuu
Program Counter	000H	000H	000H	000H	000H
MP	-xxx xxxx	-uuu uuuu	-uuu uuuu	-uuu uuuu	-uuu uuuu
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLP	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLH	--xx xxxx	--uu uuuu	--uu uuuu	--uu uuuu	--uu uuuu
STATUS	--00 xxxx	--1u uuuu	--uu uuuu	--01 uuuu	--11 uuuu
INTC0	-000 0000	-000 0000	-000 0000	-000 0000	-uuu uuuu
INTC1	--xx --xx	--xx --xx	--xx --xx	--xx --xx	--uu --uu
PA	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PB	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PBC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PC	---- --11	---- --11	---- --11	---- --11	---- --uu
PCC	---- --11	---- --11	---- --11	---- --11	---- --uu
PD	---- ---1	---- ---1	---- ---1	---- ---1	---- ---u
PDC	---- ---1	---- ---1	---- ---1	---- ---1	---- ---u
PWM	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
HADR	xxxx xxx-	xxxx xxx-	xxxx xxx-	xxxx xxx-	uuuu uuu-
HCR	0--0 0---	0--0 0---	0--0 0---	0--0 0---	u--u u---
HSR	100- -0-1	100- -0-1	100- -0-1	100- -0-1	uuu- -u-u
HDR	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
ADRL	x--- ----	x--- ----	x--- ----	x--- ----	u--- ----
ADRH	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
ADCR	0100 0000	0100 0000	0100 0000	0100 0000	uuuu uuuu
ACSR	1--- --00	1--- --00	1--- --00	1--- --00	u--- --uu
USR	0000 1011	0000 1011	0000 1011	0000 1011	uuuu uuuu
UCR1	0000 00x0	0000 00x0	0000 00x0	0000 00x0	uuuu uuuu
UCR2	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
TXR/RXR	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
BGR	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 and the clock may come from an external source or the system clock.

Using the internal system clock, there is only one reference time-base. The internal clock source comes from f_{SYS}. Using 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.

There are two registers related to the timer/event counter; TMR ([0DH]), TMRC ([0EH]). Two physical registers are mapped to TMR location; writing TMR makes the starting value be placed in the timer/event counter preload register and reading TMR gets the contents of the timer/event counter. The TMRC is a timer/event counter control register, which defines some options.

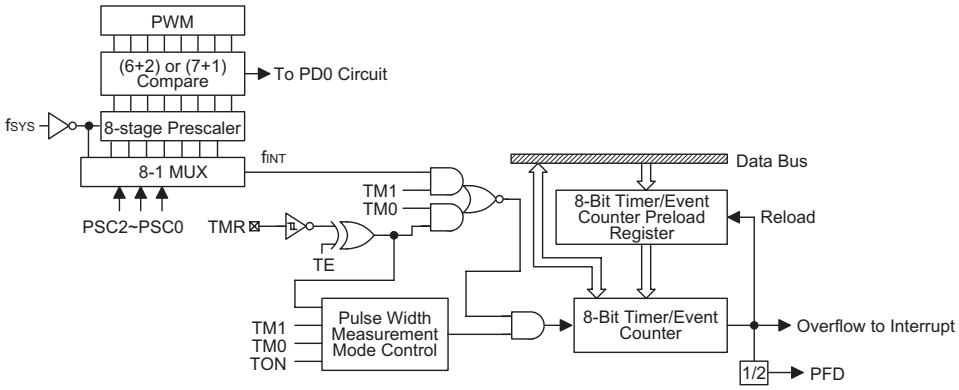
The TM0, TM1 bits define the operating mode. The event count mode is used to count external events, which means the clock source comes from an external (TMR) pin. The timer mode functions as a normal timer with the clock source coming from the f_{INT} clock. The pulse width measurement mode can be used to count the high or low level duration of the external signal (TMR). The counting is based on the 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 overflow occurs, the counter is reloaded from the timer/event counter preload register and generates the interrupt request flag (TF; bit 5 of INTC0) at the same time.

In the pulse width measurement mode with the TON and TE bits equal to 1, once the TMR has received a transient from low to high (or high to low if the TE bits is ″0″) it will start counting until the TMR returns to the original level and resets the TON. The measured result will remain in the timer/event counter even if the activated transient occurs again. In other words, only 1 cycle measurement can be done. Until setting the TON, the cycle measurement will function again as long as it receives further transient pulse. 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 counter overflows, the counter is reloaded from the timer/event counter preload register and issues the 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 will be cleared automatically after the measurement cycle is completed. But in the other two modes the TON 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.

Bit No.	Label	Function
0 1 2	PSC0 PSC1 PSC2	To define 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	To enable or disable timer counting (0=disabled; 1=enabled)
5	—	Unused bit, read as ″0″
6 7	TM0 TM1	To define the operating mode 01=Event count mode (external clock) 10=Timer mode (internal clock) 11=Pulse width measurement mode 00=Unused

TMRC (0EH) Register



Timer/Event Counter

In the case of 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 overflow occurs. When the timer/event counter (reading TMR) 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 definitions are as shown. The overflow signal of timer/event counter can be used to generate the PFD signal.

Input/Output Ports

There are 19 bidirectional input/output lines in the microcontroller, labeled as PA, PB, PC and PD, which are mapped to the data memory of [12H], [14H], [16H] and [18H] respectively. All of these I/O ports can be used for input and output operations. For input operation, these ports are non-latching, that is, the inputs must be ready at the T2 rising edge of instruction "MOV A,[m]" (m=12H, 14H, 16H or 18H). For output operation, all the data is latched and remains unchanged until the output latch is rewritten.

Each I/O line has its own control register (PAC, PBC, PCC, PDC) to control the input/output configuration. With this control register, CMOS output or Schmitt trigger input with or without pull-high resistor structures can be reconfigured dynamically (i.e. on-the-fly) under software control. To function as an input, the corresponding latch of the control register must write "1". The input source also depends on the control register. If the control register bit is "1", the input will read the pad state. If the control register bit is "0", the contents of the latches will move to the internal bus. The latter is possible in the "read-modify-write" instruction.

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

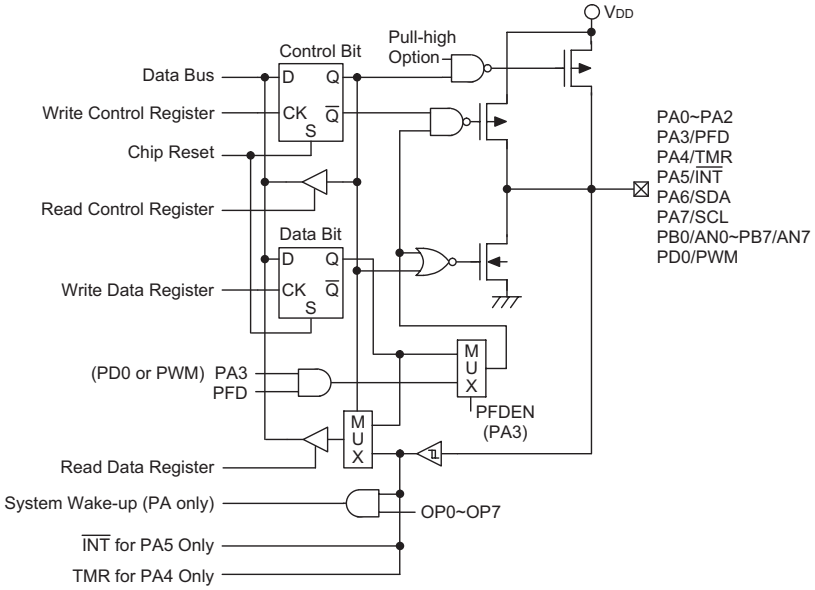
After a chip reset, these input/output lines remain at high levels or floating state (dependent on pull-high options). Each bit of these input/output latches can be set or cleared by "SET [m].i" and "CLR [m].i" (m=12H, 14H, 16H or 18H) instructions.

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

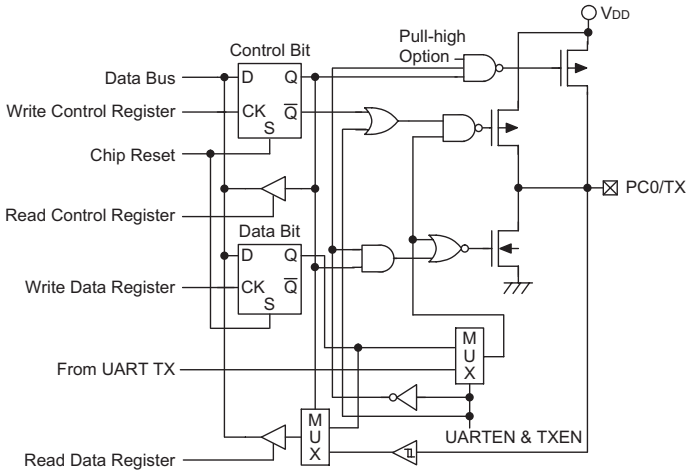
Each line of port A has the capability of waking-up the device. The highest 6-bit of port C and 7-bit of port D are not physically implemented; on reading them a "0" is returned whereas writing then results in a no-operation. See Application note.

Each I/O port has a pull-high option. Once the pull-high option is selected, the I/O port has a pull-high resistor, otherwise, there's none. Take note that a non-pull-high I/O port operating in input mode will cause a floating state.

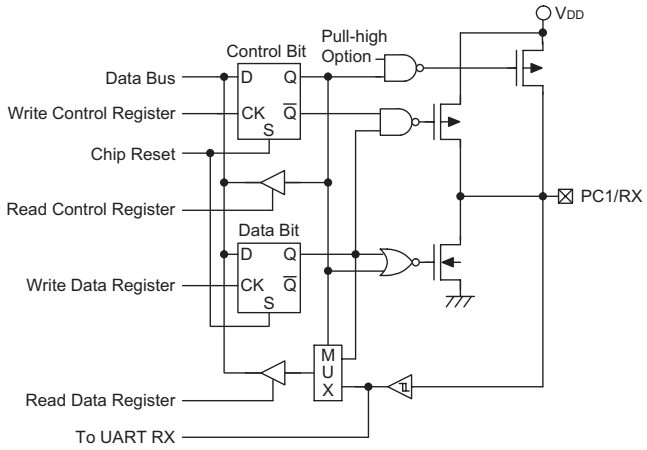
The PA3 is pin-shared with the PFD signal. If the PFD option is selected, the output signal in output mode of PA3 will be the PFD signal generated by timer/event counter overflow signal. The input mode always remaining its original functions. Once the PFD option is selected, the PFD output signal is controlled by PA3 data register only. Writing "1" to PA3 data register will enable the PFD output function and writing "0" will force the PA3 to remain at "0".



Input/Output Ports



PC0/TX Input/Output Ports



PC1/RX Input/Output Ports

The I/O functions of PA3 are shown below.

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

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

The PA5 and PA4 are pin-shared with $\overline{INT}$ and TMR pins respectively.

The PB can also be used as A/D converter inputs. The A/D function will be described later. There is a PWM function shared with PD0. If the PWM function is enabled, the PWM signal will appear on PD0 (if PD0 is operating in output mode). Writing "1" to PD0 data register will enable the PWM output function and writing "0" will force the PD0 to remain at "0". The I/O functions of PD0 is as shown.

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 lines should be set as output pins by software instruction to avoid consuming power under input floating state.

PWM

The microcontroller provides 1 channels (6+2)/(7+1) (dependent on options) bits PWM output shared with PD0. The PWM channel has its data registers denoted as PWM(1AH). The frequency source of the PWM counter comes from f_{SYS}. The PWM registers is a 8-bit register. The waveforms of PWM outputs are as shown. Once the PD0 is selected as the PWM outputs and the output function of PD0 is enabled (PDC.0="0"), writing 1 to PD0 data register will enable the PWM output function and writing "0" will force the PD0 to stay at "0".

A (6+2) bits mode PWM cycle is divided into four modulation cycles (modulation cycle 0~modulation cycle 3). Each modulation cycle has 64 PWM input clock period.

In a (6+2) bit PWM function, the contents of the PWM register is divided into two groups. Group 1 of the PWM register is denoted by DC which is the value of PWM.7~PWM.2. The group 2 is denoted by AC which is the value of PWM.1~PWM.0.

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

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

A (7+1) bits mode PWM cycle is divided into two modulation cycles (modulation cycle 0 ~ modulation cycle 1). Each modulation cycle has 128 PWM input clock period.

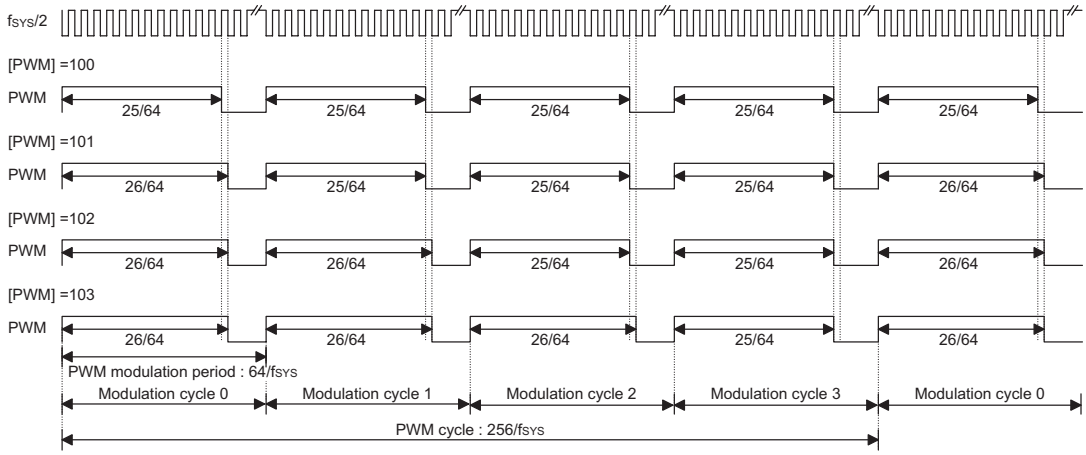
In a (7+1) bits PWM function, the contents of the PWM register is divided into two groups. Group 1 of the PWM register is denoted by DC which is the value of PWM.7~PWM.1. The group 2 is denoted by AC which is the value of PWM.0.

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

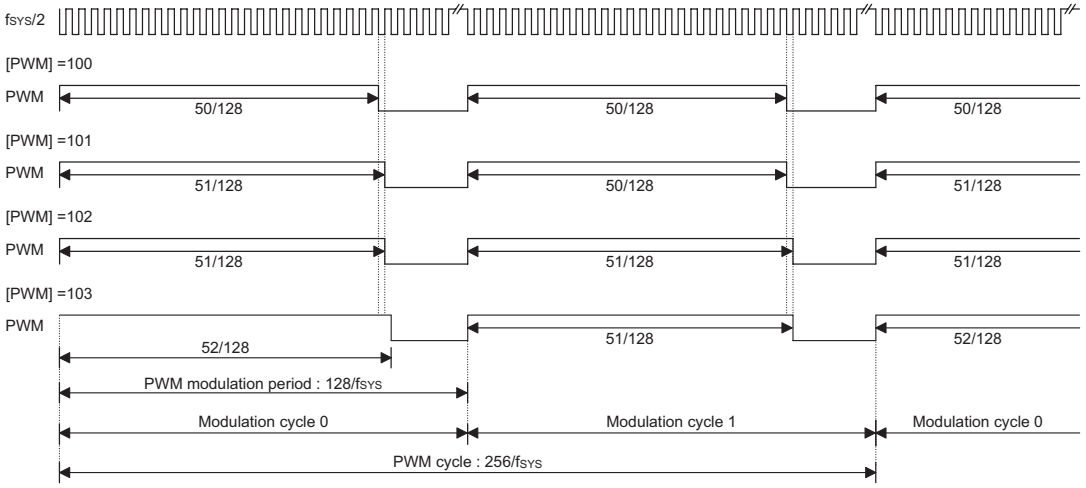
Parameter	AC (0~1)	Duty Cycle
Modulation cycle i (i=0~1)	i<AC	$\frac{DC+1}{128}$
	i≥AC	$\frac{DC}{128}$

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

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



(6+2) PWM Mode



(7+1) PWM Mode

A/D Converter

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

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

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

ACSR (27H) Register

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

ADCR (26H) Register

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

Port B Configuration

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

Analog Input Channel Selection

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

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

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

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

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

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

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

Example: using EOCB Polling Method to detect end of conversion

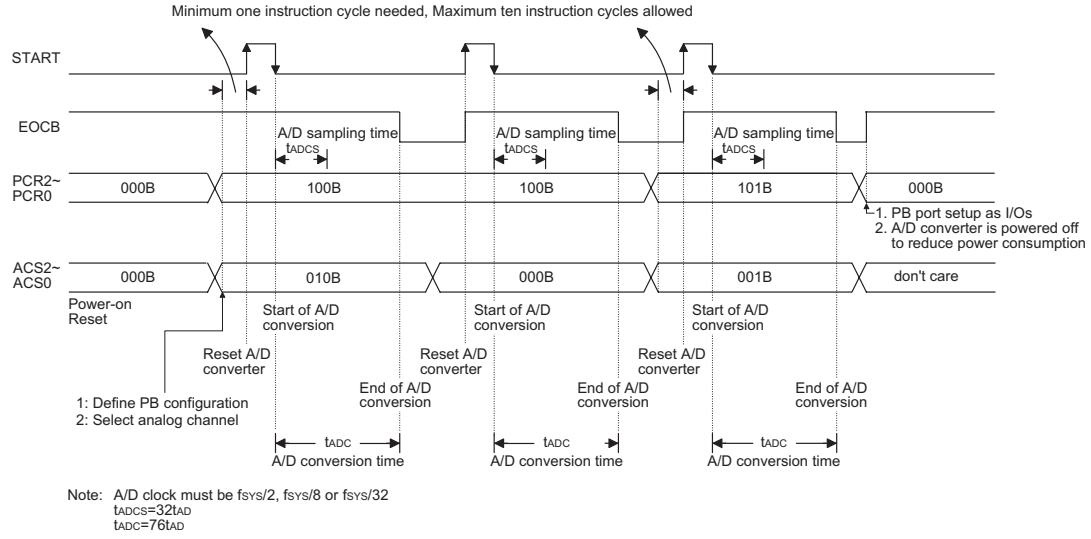
```
    clr     EADI                ; disable ADC interrupt
    mov     a,00000001B
    mov     ACSR,a              ; setup the ACSR register to select fSYS/8 as the A/D clock
    mov     a,00100000B         ; setup ADCR register to configure Port PB0~PB3 as A/D inputs
    mov     ADCR,a            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

Example: using interrupt method to detect end of conversion

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

    mov     a,00100000B         ; setup ADCR register to configure Port PB0~PB3 as A/D inputs
    mov     ADCR,a              ; and select AN0 to be connected to the A/D converter
            :
                                ; As the Port B channel bits have changed the following START
                                ; signal (0-1-0) must be issued within 10 instruction cycles
```

```
        :
Start_conversion:
    clr     START
    set     START           ; reset A/D
    clr     START           ; start A/D
    clr     ADF             ; clear ADC interrupt request flag
    set     EADI            ; enable ADC interrupt
    set     EMI             ; enable global interrupt
        :
        :
        :
; ADC interrupt service routine
ADC_ISR:
    mov     acc_stack,a     ; save ACC to user defined memory
    mov     a,STATUS
    mov     status_stack,a  ; save STATUS to user defined memory
        :
        :
    mov     a,ADRH          ; read conversion result high byte value from the ADRH register
    mov     adrh_buffer,a   ; save result to user defined register
    mov     a,ADRL          ; read conversion result low byte value from the ADRL register
    mov     adrl_buffer,a   ; save result to user defined register
    clr     START
    set     START           ; reset A/D
    clr     START           ; start A/D
        :
        :
EXIT_INT_ISR:
    mov     a,status_stack
    mov     STATUS,a        ; restore STATUS from user defined memory
    mov     a,acc_stack     ; restore ACC from user defined memory
    reti
```



Note: A/D clock must be $f_{SYS}/2$, $f_{SYS}/8$ or $f_{SYS}/32$
$t_{ADCS}=32t_{AD}$
$t_{ADC}=76t_{AD}$

A/D Conversion Timing

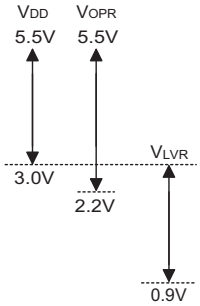
Low Voltage Reset – LVR

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~3.3V, such as changing a battery, the LVR will automatically reset the device internally.

The LVR includes the following specifications:

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

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



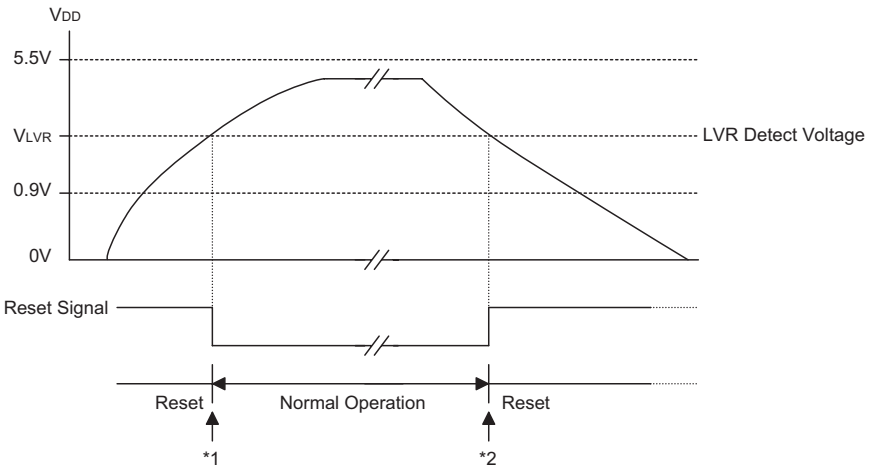
Note: V_{OPR} is the voltage range for proper chip operation at 4MHz system clock.

I²C Bus Serial Interface

I²C Bus is implemented in the device. The I²C Bus is a bidirectional 2-wire lines. The data line and clock line are implement in SDA pin and SCL pin. The SDA and SCL are NMOS open drain output pin. They must connect a pull-high resistor respectively.

Using the I²C Bus, the device has two ways to transfer data. One is in slave transmit mode, the other is in slave receive mode. There are four registers related to I²C Bus; HADR([20H]), HCR([21H]), HSR([22H]), HDR([23H]). The HADR register is the slave address setting of the device, if the master sends the calling address which match, it means that this device is selected. The HCR is I²C Bus control register which defines the device enable or disable the I²C Bus as a transmitter or as a receiver. The HSR is I²C Bus status register, it responds with the I²C Bus status. The HDR is input/output data register, data to transmit or receive must be via the HDR register.

The I²C Bus control register contains three bits. The HEN bit define the enable or disable the I²C Bus. If the data wants transfer via I²C Bus, this bit must be set. The HTX bit defines whether the I²C Bus is in transmit or receive mode. If the device is as a transmitter, this bit must be set to "1". The TXAK defines the transmit acknowledge signal, when the device received 8-bit data, the device sends this bit to I²C Bus at the 9th clock. If the receiver wants to continue to receive the next data, this bit must be reset to "0" before receiving data.



Low Voltage Reset

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

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

The I²C Bus status register contains 5 bits. The HCF bit is reset to "0" when one data byte is being transferred. If one data transfer is completed, this bit is set to "1". The HASS bit is set "1" when the address is match, and the I²C Bus interrupt request flag is set to "1". If the interrupt is enabled and the stack is not full, a subroutine call to location 10H will occur. Writing data to the I²C Bus control register clears HAAS bit. If the address is not match, this bit is reset to "0". The HBB bit is set to respond the I²C Bus is busy. It mean that a START signal is detected. This bit is reset to "0" when the I²C Bus is not busy. It means that a STOP signal is detected and the I²C Bus is free. The SRW bit defines the read/write command bit, if the calling address is match. When HAAS is set to "1", the device check SRW bit to determine whether the device is working in transmit or receive mode. When SRW bit is set "1", it means that the master wants to read data from I²C Bus, the slave device must write data to I²C Bus, so the slave device is working in transmit mode. When SRW is reset to "0", it means that the master wants to write data to I²C Bus, the slave device must read data from the bus, so the slave device is working in receive mode. The RXAK bit is reset "0" indicates an acknowledges signal has been received. In the transmit mode, the transmitter checks RXAK bit to know the receiver which wants to receive the next data byte, so the transmitter continue to write data to the I²C Bus until the RXAK bit is set to "1" and the transmitter releases the SDA line, so that the master can send the STOP signal to release the bus.

The HADR bit7~bit1 define the device slave address. At the beginning of transfer, the master must select a device by sending the address of the slave device. The bit 0 is unused and is not defined. If the I²C Bus receives a start signal, all slave device notice the continuity of the 8-bit data. The front of 7 bits is slave address and the first bit is MSB. If the address is match, the HAAS status bit is set and generate an I²C Bus interrupt. In the ISR, the slave device must check the HAAS bit to know the I²C Bus interrupt comes from the slave address that has match or completed one 8-bit data transfer. The last bit of the 8-bit data is read/write command bit, it responds in SRW bit. The slave will check the SRW bit to know if the master wants to transmit or receive data. The device check SRW bit to know it is as a transmitter or receiver.

Bit7~Bit1	Bit0
Slave Address	—

"—" means undefined

HADR (20H) Register

The HDR register is the I²C Bus input/output data register. Before transmitting data, the HDR must write the data which we want to transmit. Before receiving data, the device must dummy read data from HDR. Transmit or Receive data from I²C Bus must be via the HDR register. At the beginning of the transfer of the I²C Bus, the device must initial the bus, the following are the notes for initialing the I²C Bus.

Note:

1: Write the I²C Bus address register (HADR) to define its own slave address.

2: Set HEN bit of I²C Bus control register (HCR) bit 0 to enable the I²C Bus.

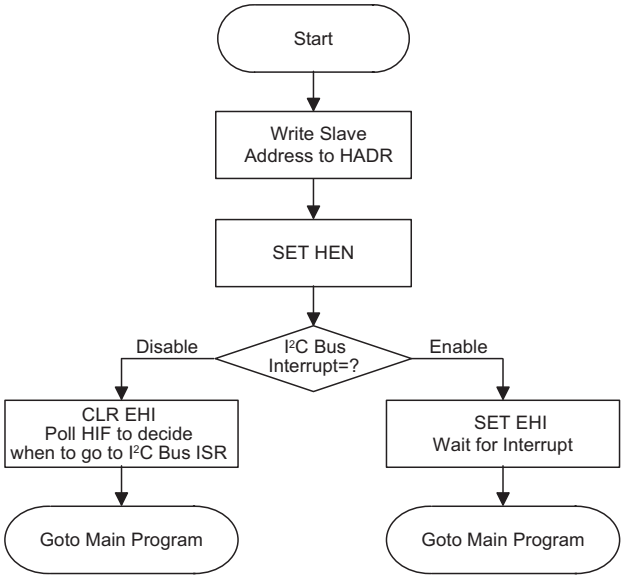
Bit No.	Label	Function
7	HEN	To enable or disable I²C Bus function (0= disable; 1= enable)
6~5	—	Unused bit, read as "0"
4	HTX	To define the transmit or receive mode (0= receive mode; 1= transmit)
3	TXAK	To enable or disable transmit acknowledge (0=acknowledge; 1=don't acknowledge)
2~0	—	Unused bit, read as "0"

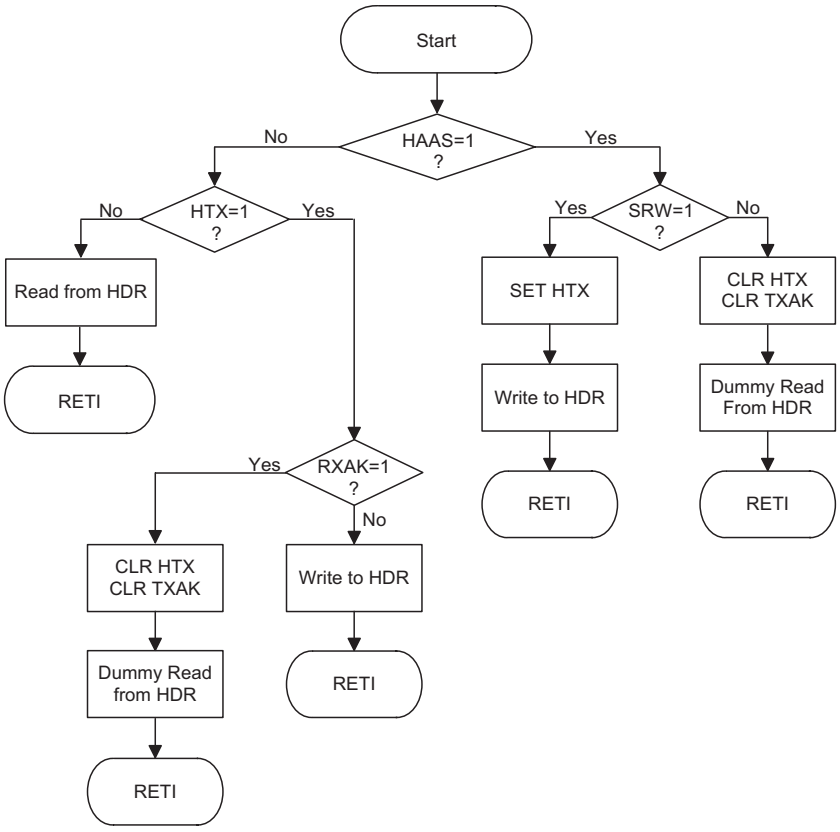
HCR (21H) Register

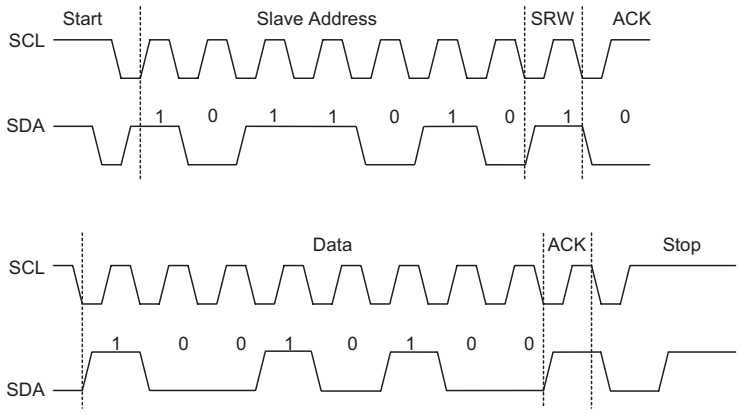
3: Set EHI bit of the interrupt control register 1 (INTC1) bit 0 to enable the I²C Bus interrupt.

Bit No.	Label	Function
7	HCF	HCF is clear to "0" when one data byte is being transferred, HCF is set to "1" indicating 8-bit data communication has been finished.
6	HAAS	HAAS is set to "1" when the calling address has matched, and I²C Bus interrupt will occur and HIF is set.
5	HBB	HBB is set to "1" when I²C Bus is busy and HBB is cleared to "0" means that the I²C Bus is not busy.
4~3	—	Unused bit, read as "0"
2	SRW	SRW is set to "1" when the master wants to read data from the I²C Bus, so the slave must transmit data to the master. SRW is cleared to "0" when the master wants to write data to the I²C Bus, so the slave must receive data from the master.
1	—	Unused bit, read as "0"
0	RXAK	RXAK is cleared to "0" when the master receives an 8-bit data and acknowledgment at the 9th clock, RXAK is set to "1" means not acknowledged.

HSR (22H) Register


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


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



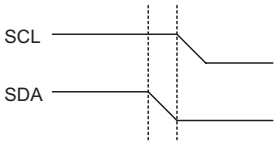
S=Start (1 bit)
SA=Slave Address (7 bits)
SR=SRW bit (1 bit)
M=Slave device send acknowledge bit (1 bit)
D=Data (8 bits)
A=ACK (RXAK bit for transmitter, TXAK bit for receiver 1 bit)
P=Stop (1 bit)

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

I²C Communication Timing Diagram

Start Signal

The START signal is generated only by the master device. The other device in the bus must detect the START signal to set the I²C Bus busy bit (HBB). The START signal is SDA line from high to low, when SCL is high.



Start Bit

Slave Address

The master must select a device for transferring the data by sending the slave device address after the START signal. All device in the I²C Bus will receive the I²C Bus slave address (7 bits) to compare with its own slave address (7 bits). If the slave address is matched, the slave device will generate an interrupt and save the following bit (8th bit) to SRW bit and sends an acknowledge bit (low level) to the 9th bit. The slave device also sets the status flag (HAAS), when the slave address is matched.

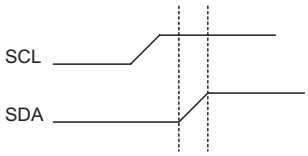
In interrupt subroutine, check HAAS bit to know whether the I²C Bus interrupt comes from a slave address that is matched or a data byte transfer is completed. When the slave address is matched, the device must be in transmit mode or receive mode and write data to HDR or dummy read from HDR to release the SCL line.

SRW Bit

The SRW bit means that the master device wants to read from or write to the I²C Bus. The slave device check this bit to understand itself if it is a transmitter or a receiver. The SRW bit is set to ″1″ means that the master wants to read data from the I²C Bus, so the slave device must write data to a bus as a transmitter. The SRW is cleared to ″0″ means that the master wants to write data to the I²C Bus, so the slave device must read data from the I²C Bus as a receiver.

Acknowledge Bit

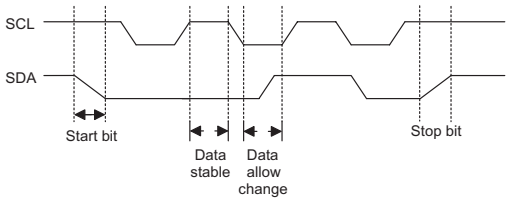
One of the slave device generates an acknowledge signal, when the slave address is matched. The master device can check this acknowledge bit to know if the slave device accepts the calling address. If no acknowledge bit, the master must send a STOP bit and end the communication. When the I^2C Bus status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



Stop Bit

Data Byte

The data is 8 bits and is sent after the slave device has acknowledges the slave address. The first bit is MSB and the 8th bit is LSB. The receiver sends the acknowledge signal (″0″) and continues to receive the next 1 byte data. If the transmitter checks and there′s no acknowledge signal, then it release the SDA line, and the master sends a STOP signal to release the I^2C Bus. The data is stored in the HDR register. The transmitter must write data to the HDR before transmit data and the receiver must read data from the HDR after receiving data.



Data Timing Diagram

Receive Acknowledge Bit

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

UART Bus Serial Interface

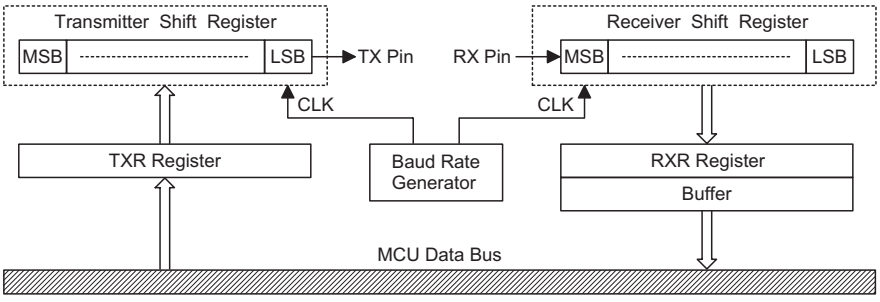
The HT46RU22 devices contain an integrated full-duplex asynchronous serial communications UART interface that enables communication with external devices that contain a serial interface. The UART function has many features and can transmit and receive data serially by transferring a frame of data with eight or nine data bits per transmission as well as being able to detect errors when the data is overwritten or incorrectly framed. The UART function possesses its own internal interrupt which can be used to indicate when a reception occurs or when a transmission terminates.

- UART features
  The integrated UART function contains the following features:
  - Full-duplex, asynchronous communication
  - 8 or 9 bits character length
  - Even, odd or no parity options
  - One or two stop bits
  - Baud rate generator with 8-bit prescaler
  - Parity, framing, noise and overrun error detection
  - Support for interrupt on address detect (last character bit=1)
  - Separately enabled transmitter and receiver
  - 2-byte Deep Fifo Receive Data Buffer
  - Transmit and receive interrupts
  - Interrupts can be initialized by the following conditions:
    - Transmitter Empty
    - Transmitter Idle
    - Receiver Full
    - Receiver Overrun
    - Address Mode Detect

- UART external pin interfacing
  To communicate with an external serial interface, the internal UART has two external pins known as TX and RX. The TX pin is the UART transmitter pin, which can be used as a general purpose I/O pin if the pin is not configured as a UART transmitter, which occurs when the TXEN bit in the UCR2 control register is equal to zero. Similarly, the RX pin is the UART receiver pin, which can also be used as a general purpose I/O pin, if the pin is not configured as a receiver, which occurs if the RXEN bit in the UCR2 register is equal to zero. Along with the UARTEN bit, the TXEN and RXEN bits, if set, will automatically setup these I/O pins to their respective TX output and RX input conditions and disable any pull-high resistor option which may exist on the RX pin.

- UART data transfer scheme
  The block diagram shows the overall data transfer structure arrangement for the UART. The actual data to be transmitted from the MCU is first transferred to the TXR register by the application program. The data will then be transferred to the Transmit Shift Register from where it will be shifted out, LSB first, onto the TX pin at a rate controlled by the Baud Rate Generator. Only the TXR register is mapped onto the MCU Data Memory, the Transmit Shift Register is not mapped and is therefore inaccessible to the application program.
  Data to be received by the UART is accepted on the external RX pin, from where it is shifted in, LSB first, to the Receiver Shift Register at a rate controlled by the Baud Rate Generator. When the shift register is full, the data will then be transferred from the shift register to the internal RXR register, where it is buffered and can be manipulated by the application program. Only the RXR register is mapped onto the MCU Data Memory, the Receiver Shift Register is not mapped and is therefore inaccessible to the application program.
  It should be noted that the actual register for data transmission and reception, although referred to in the text, and in application programs, as separate TXR and RXR registers, only exists as a single shared register in the Data Memory. This shared register known as the TXR/RXR register is used for both data transmission and data reception.

- UART status and control registers
  There are five control registers associated with the UART function. The USR, UCR1 and UCR2 registers control the overall function of the UART, while the BRG register controls the Baud rate. The actual data to be transmitted and received on the serial interface is managed through the TXR/RXR data registers.



UART Data Transfer Scheme

- USR register

  The USR register is the status register for the UART, which can be read by the program to determine the present status of the UART. All flags within the USR register are read only.

  Further explanation on each of the flags is given below:

  - TXIF

    The TXIF flag is the transmit data register empty flag. When this read only flag is ″0″ it indicates that the character is not transferred to the transmit shift registers. When the flag is ″1″ it indicates that the transmit shift register has received a character from the TXR data register. The TXIF flag is cleared by reading the UART status register (USR) with TXIF set and then writing to the TXR data register. Note that when the TXEN bit is set, the TXIF flag bit will also be set since the transmit buffer is not yet full.

  - TIDLE

    The TIDLE flag is known as the transmission complete flag. When this read only flag is ″0″ it indicates that a transmission is in progress. This flag will be set to ″1″ when the TXIF flag is ″1″ and when there is no transmit data, or break character being transmitted. When TIDLE is ″1″ the TX pin becomes idle. The TIDLE flag is cleared by reading the USR register with TIDLE set and then writing to the TXR register. The flag is not generated when a data character, or a break is queued and ready to be sent.

  - RXIF

    The RXIF flag is the receive register status flag. When this read only flag is ″0″ it indicates that the RXR read data register is empty. When the flag is ″1″ it indicates that the RXR read data register contains new data. When the contents of the shift register are transferred to the RXR register, an interrupt is generated if RIE=1 in the UCR2 register. If one or more errors are detected in the received word, the appropriate receive-related flags NF, FERR, and/or PERR are set within the same clock cycle. The RXIF flag is cleared when the USR register is read with RXIF set, followed by a read from the RXR register, and if the RXR register has no data available.

  - RIDLE

    The RIDLE flag is the receiver status flag. When this read only flag is ″0″ it indicates that the receiver is between the initial detection of the start bit and the completion of the stop bit. When the flag is ″1″ it indicates that the receiver is idle. Between the completion of the stop bit and the detection of the next start bit, the RIDLE bit is ″1″ indicating that the UART is idle.

  - OERR

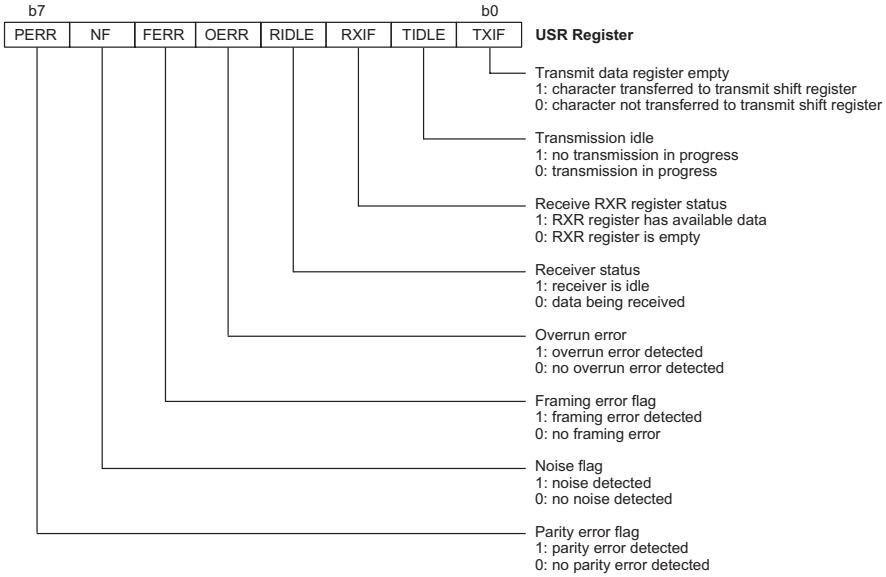
    The OERR flag is the overrun error flag, which indicates when the receiver buffer has overflowed. When this read only flag is ″0″ there is no overrun error. When the flag is ″1″ an overrun error occurs which will inhibit further transfers to the RXR receive data register. The flag is cleared by a software sequence, which is a read to the status register USR followed by an access to the RXR data register.

  - FERR

    The FERR flag is the framing error flag. When this read only flag is ″0″ it indicates no framing error. When the flag is ″1″ it indicates that a framing error has been detected for the current character. The flag can also be cleared by a software sequence which will involve a read to the USR status register followed by an access to the RXR data register.

  - NF

    The NF flag is the noise flag. When this read only flag is ″0″ it indicates a no noise condition. When the flag is ″1″ it indicates that the UART has detected noise on the receiver input. The NF flag is set during the same cycle as the RXIF flag but will not be set in the case of an overrun. The NF flag can be cleared by a software sequence which will involve a read to the USR status register, followed by an access to the RXR data register.



- PERR

  The PERR flag is the parity error flag. When this read only flag is ″0″ it indicates that a parity error has not been detected. When the flag is ″1″ it indicates that the parity of the received word is incorrect. This error flag is applicable only if Parity mode (odd or even) is selected. The flag can also be cleared by a software sequence which involves a read to the USR status register, followed by an access to the RXR data register.

- UCR1 register

  The UCR1 register together with the UCR2 register are the two UART control registers that are used to set the various options for the UART function, such as overall on/off control, parity control, data transfer bit length etc.

  Further explanation on each of the bits is given below:

  - TX8

    This bit is only used if 9-bit data transfers are used, in which case this bit location will store the 9th bit of the transmitted data, known as TX8. The BNO bit is used to determine whether data transfers are in 8-bit or 9-bit format.

  - RX8

    This bit is only used if 9-bit data transfers are used, in which case this bit location will store the 9th bit of the received data, known as RX8. The BNO bit is used to determine whether data transfers are in 8-bit or 9-bit format.

  - TXBRK

    The TXBRK bit is the Transmit Break Character bit. When this bit is ″0″ there are no break characters and the TX pin operates normally. When the bit is ″1″ there are transmit break characters and the transmitter will send logic zeros. When equal to ″1″ after the buffered data has been transmitted, the transmitter output is held low for a minimum of a 13-bit length and until the TXBRK bit is reset.

  - STOPS

    This bit determines if one or two stop bits are to be used. When this bit is equal to ″1″ two stop bits are used, if the bit is equal to ″0″ then only one stop bit is used.

  - PRT

    This is the parity type selection bit. When this bit is equal to ″1″ odd parity will be selected, if the bit is equal to ″0″ then even parity will be selected.

  - PREN

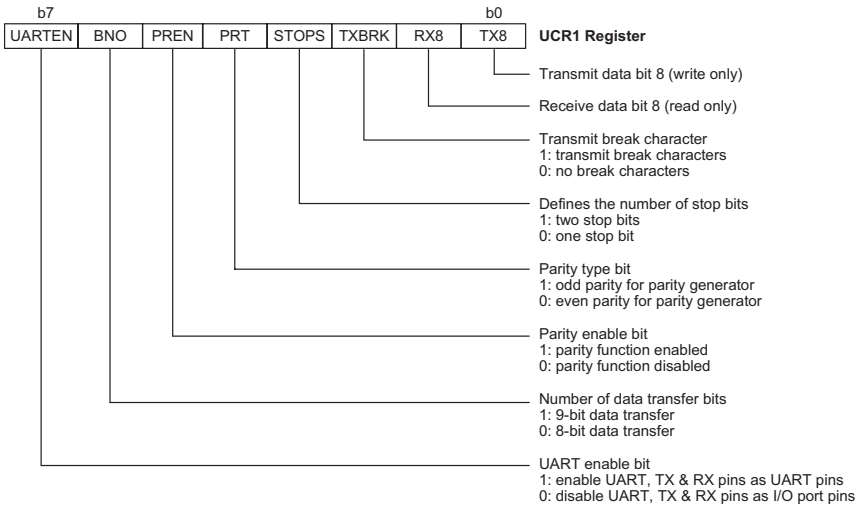
    This is parity enable bit. When this bit is equal to ″1″ the parity function will be enabled, if the bit is equal to ″0″ then the parity function will be disabled.

  - BNO

    This bit is used to select the data length format, which can have a choice of either 8-bits or 9-bits. If this bit is equal to ″1″ then a 9-bit data length will be selected, if the bit is equal to ″0″ then an 8-bit data length will be selected. If 9-bit data length is selected then bits RX8 and TX8 will be used to store the 9th bit of the received and transmitted data respectively.

  - UARTEN

    The UARTEN bit is the UART enable bit. When the bit is ″0″ the UART will be disabled and the RX and TX pins will function as General Purpose I/O pins. When the bit is ″1″ the UART will be enabled and the TX and RX pins will function as defined by the TXEN and RXEN control bits. When the UART is disabled it will empty the buffer so any character remaining in the buffer will be discarded. In addition, the baud rate counter value will be reset. When the UART is disabled, all error and status flags will be reset. The TXEN, RXEN, TXBRK, RXIF, OERR, FERR, PERR, and NF bits will be cleared, while the TIDLE, TXIF and RIDLE bits will be set. Other control bits in UCR1, UCR2, and BRG registers will remain unaffected. If the UART is active and the UARTEN bit is cleared, all pending transmissions and receptions will be terminated and the module will be reset as defined above. When the UART is re-enabled it will restart in the same configuration.



- UCR2 register

  The UCR2 register is the second of the two UART control registers and serves several purposes. One of its main functions is to control the basic enable/disable operation of the UART Transmitter and Receiver as well as enabling the various UART interrupt sources. The register also serves to control the baud rate speed, receiver wake-up enable and the address detect enable.

  Further explanation on each of the bits is given below:

  - TEIE

    This bit enables or disables the transmitter empty interrupt. If this bit is equal to ″1″ when the transmitter empty TXIF flag is set, due to a transmitter empty condition, the UART interrupt request flag will be set. If this bit is equal to ″0″ the UART interrupt request flag will not be influenced by the condition of the TXIF flag.

  - TIIE

    This bit enables or disables the transmitter idle interrupt. If this bit is equal to ″1″ when the transmitter idle TIDLE flag is set, the UART interrupt request flag will be set. If this bit is equal to ″0″ the UART interrupt request flag will not be influenced by the condition of the TIDLE flag.

  - RIE

    This bit enables or disables the receiver interrupt. If this bit is equal to ″1″ when the receiver overrun OERR flag or receive data available RXIF flag is set, the UART interrupt request flag will be set. If this bit is equal to ″0″ the UART interrupt will not be influenced by the condition of the OERR or RXIF flags.

  - WAKE

    This bit enables or disables the receiver wake-up function. If this bit is equal to ″1″ and if the MCU is in the Power Down Mode, a low going edge on the RX input pin will wake-up the device. If this bit is equal to ″0″ and if the MCU is in the Power Down Mode, any edge transitions on the RX pin will not wake-up the device.

  - ADDEN

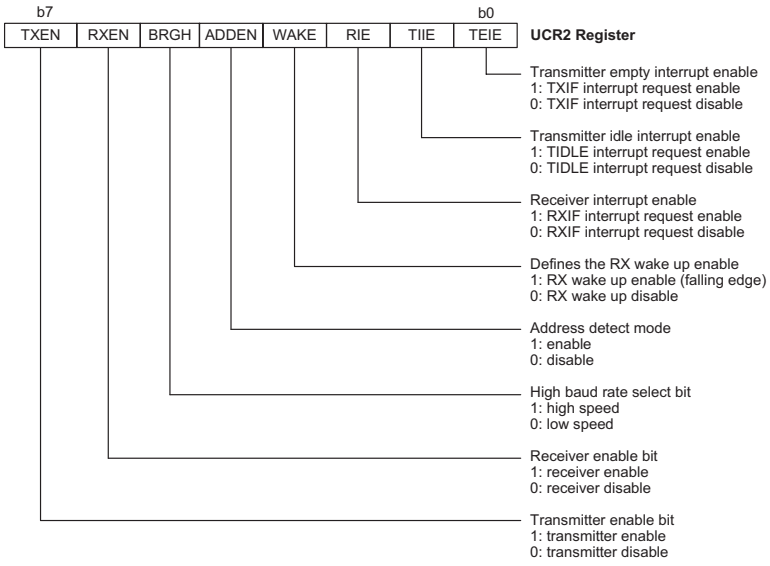
    The ADDEN bit is the address detect mode bit. When this bit is ″1″ the address detect mode is enabled. When this occurs, if the 8th bit, which corresponds to RX7 if BNO=0, or the 9th bit, which corresponds to RX8 if BNO=1, has a value of ″1″ then the received word will be identified as an address, rather than data. If the corresponding interrupt is enabled, an interrupt request will be generated each time the received word has the address bit set, which is the 8 or 9 bit depending on the value of BNO. If the address bit is ″0″ an interrupt will not be generated, and the received data will be discarded.

  - BRGH

    The BRGH bit selects the high or low speed mode of the Baud Rate Generator. This bit, together with the value placed in the BRG register, controls the Baud Rate of the UART. If this bit is equal to ″1″ the high speed mode is selected. If the bit is equal to ″0″ the low speed mode is selected.

  - RXEN

    The RXEN bit is the Receiver Enable Bit. When this bit is equal to ″0″ the receiver will be disabled with any pending data receptions being aborted. In addition the buffer will be reset. In this situation the RX pin can be used as a general purpose I/O pin. If the RXEN bit is equal to ″1″ the receiver will be enabled and if the UARTEN bit is equal to ″1″ the RX pin will be controlled by the UART. Clearing the RXEN bit during a transmission will cause the data reception to be aborted and will reset the receiver. If this occurs, the RX pin can be used as a general purpose I/O pin.



- TXEN
  The TXEN bit is the Transmitter Enable Bit. When this bit is equal to ″0″ the transmitter will be disabled with any pending transmissions being aborted. In addition the buffer will be reset. In this situation the TX pin can be used as a general purpose I/O pin. If the TXEN bit is equal to ″1″ the transmitter will be enabled and if the UARTEN bit is equal to ″1″ the TX pin will be controlled by the UART. Clearing the TXEN bit during a transmission will cause the transmission to be aborted and will reset the transmitter. If this occurs, the TX pin can be used as a general purpose I/O pin.

• Baud rate generator
To setup the speed of the serial data communication, the UART function contains its own dedicated baud rate generator. The baud rate is controlled by its own internal free running 8-bit timer, the period of which is determined by two factors. The first of these is the value placed in the BRG register and the second is the value of the BRGH bit within the UCR2 control register. The BRGH bit decides, if the baud rate generator is to be used in a high speed mode or low speed mode, which in turn determines the formula that is used to calculate the baud rate. The value in the BRG register determines the division factor, N, which is used in the following baud rate calculation formula. Note that N is the decimal value placed in the BRG register and has a range of between 0 and 255.

UCR2 BRGH Bit	0	1
Baud Rate	$\frac{f_{SYS}}{[64\ (N+1)]}$	$\frac{f_{SYS}}{[16\ (N+1)]}$

By programming the BRGH bit which allows selection of the related formula and programming the required value in the BRG register, the required baud rate can be setup. Note that because the actual baud rate is determined using a discrete value, N, placed in the BRG register, there will be an error associated between the actual and requested value. The following example shows how the BRG register value N and the error value can be calculated.

Calculating the register and error values

For a clock frequency of 8MHz, and with BRGH set to ″0″ determine the BRG register value N, the actual baud rate and the error value for a desired baud rate of 9600.

From the above table the desired baud rate BR $= \frac{f_{SYS}}{[64\ (N+1)]}$

Re-arranging this equation gives $N = \frac{f_{SYS}}{(BRx64)} - 1$

Giving a value for $N = \frac{8000000}{(9600x64)} - 1 = 12.0208$

To obtain the closest value, a decimal value of 12 should be placed into the BRG register. This gives an actual or calculated baud rate value of

$BR = \frac{8000000}{[64(12+1)]} = 9615$

Therefore the error is equal to = 0.16%

The following tables show actual values of baud rate and error values for the two values of BRGH.

Baud Rate K/BPS	Baud Rates for BRGH=0											
	f_{SYS}=8MHz			f_{SYS}=7.159MHz			f_{SYS}=4MHz			f_{SYS}=3.579545MHz		
	BRG	Kbaud	Error	BRG	Kbaud	Error	BRG	Kbaud	Error	BRG	Kbaud	Error
0.3	—	—	—	—	—	—	207	0.300	0.00	185	0.300	0.00
1.2	103	1.202	0.16	92	1.203	0.23	51	1.202	0.16	46	1.19	-0.83
2.4	51	2.404	0.16	46	2.38	-0.83	25	2.404	0.16	22	2.432	1.32
4.8	25	4.807	0.16	22	4.863	1.32	12	4.808	0.16	11	4.661	-2.9
9.6	12	9.615	0.16	11	9.322	-2.9	6	8.929	-6.99	5	9.321	-2.9
19.2	6	17.857	-6.99	5	18.64	-2.9	2	20.83	8.51	2	18.643	-2.9
38.4	2	41.667	8.51	2	37.29	-2.9	1	—	—	1	—	—
57.6	1	62.5	8.51	1	55.93	-2.9	0	62.5	8.51	0	55.93	-2.9
115.2	0	125	8.51	0	111.86	-2.9	—	—	—	—	—	—

Baud Rates and Error Values for BRGH = 0

Baud Rate K/BPS	Baud Rates for BRGH=1											
	f_{SYS}=8MHz			f_{SYS}=7.159MHz			f_{SYS}=4MHz			f_{SYS}=3.579545MHz		
	BRG	Kbaud	Error	BRG	Kbaud	Error	BRG	Kbaud	Error	BRG	Kbaud	Error
0.3	—	—	—	—	—	—	—	—	—	—	—	—
1.2	—	—	—	—	—	—	207	1.202	0.16	185	1.203	0.23
2.4	207	2.404	0.16	185	2.405	0.23	103	2.404	0.16	92	2.406	0.23
4.8	103	4.808	0.16	92	4.811	0.23	51	4.808	0.16	46	4.76	-0.83
9.6	51	9.615	0.16	46	9.520	-0.832	25	9.615	0.16	22	9.727	1.32
19.2	25	19.231	0.16	22	19.454	1.32	12	19.231	0.16	11	18.643	-2.9
38.4	12	38.462	0.16	11	37.287	-2.9	6	35.714	-6.99	5	37.286	-2.9
57.6	8	55.556	-3.55	7	55.93	-2.9	3	62.5	8.51	3	55.930	-2.9
115.2	3	125	8.51	3	111.86	-2.9	1	125	8.51	1	111.86	-2.9
250	1	250	0	—	—	—	0	250	0	—	—	—

Baud Rates and Error Values for BRGH = 1

- Setting up and controlling the UART
  - Introduction

    For data transfer, the UART function utilizes a non-return-to-zero, more commonly known as NRZ, format. This is composed of one start bit, eight or nine data bits, and one or two stop bits. Parity is supported by the UART hardware, and can be setup to be even, odd or no parity. For the most common data format, 8 data bits along with no parity and one stop bit, denoted as 8, N, 1, is used as the default setting, which is the setting at power-on. The number of data bits and stop bits, along with the parity, are setup by programming the corresponding BNO, PRT, PREN, and STOPS bits in the UCR1 register. The baud rate used to transmit and receive data is setup using the internal 8-bit baud rate generator, while the data is transmitted and received LSB first. Although the UART′s transmitter and receiver are functionally independent, they both use the same data format and baud rate. In all cases stop bits will be used for data transmission.
  - Enabling/disabling the UART

    The basic on/off function of the internal UART function is controlled using the UARTEN bit in the UCR1 register. As the UART transmit and receive pins, TX and RX respectively, are pin-shared with normal I/O pins, one of the basic functions of the UARTEN control bit is to control the UART function of these two pins. If the UARTEN, TXEN and RXEN bits are set, then these two I/O pins will be setup as a TX output pin and an RX input pin respectively, in effect disabling the normal I/O pin function. If no data is being transmitted on the TX pin then it will default to a logic high value.

    Clearing the UARTEN bit will disable the TX and RX pins and allow these two pins to be used as normal I/O pins. When the UART function is disabled the buffer will be reset to an empty condition, at the same time discarding any remaining residual data. Disabling the UART will also reset the error and status flags with bits TXEN, RXEN, TXBRK, RXIF, OERR, FERR, PERR and NF being cleared while bits TIDLE, TXIF and RIDLE will be set. The remaining control bits in the UCR1, UCR2 and BRG registers will remain unaffected. If the UARTEN bit in the UCR1 register is cleared while the UART is active, then all pending transmissions and receptions will be immediately suspended and the UART will be reset to a condition as defined above. If the UART is then subsequently re-enabled, it will restart again in the same configuration.
  - Data, parity and stop bit selection

    The format of the data to be transferred, is composed of various factors such as data bit length, parity on/off, parity type, address bits and the number of stop bits. These factors are determined by the setup of various bits within the UCR1 register. The BNO bit controls the number of data bits which can be set to either 8 or 9, the PRT bit controls the choice of odd or even parity, the PREN bit controls the parity on/off function and the STOPS bit decides whether one or two stop bits are to be used. The following table shows various formats for data transmission. The address bit identifies the frame as an address character. The number of stop bits, which can be either one or two, is independent of the data length.

Start Bit	Data Bits	Address Bits	Parity Bits	Stop Bit
Example of 8-bit Data Formats				
1	8	0	0	1
1	7	0	1	1
1	7	1[1]	0	1
Example of 9-bit Data Formats				
1	9	0	0	1
1	8	0	1	1
1	8	1[1]	0	1

Transmitter Receiver Data Format

The following diagram shows the transmit and receive waveforms for both 8-bit and 9-bit data formats.

- UART transmitter

  Data word lengths of either 8 or 9 bits, can be selected by programming the BNO bit in the UCR1 register. When BNO bit is set, the word length will be set to 9 bits. In this case the 9th bit, which is the MSB, needs to be stored in the TX8 bit in the UCR1 register. At the transmitter core lies the Transmitter Shift Register, more commonly known as the TSR, whose data is obtained from the transmit data register, which is known as the TXR register. The data to be transmitted is loaded into this TXR register by the application program. The TSR register is not written to with new data until the stop bit from the previous transmission has been sent out. As soon as this stop bit has been transmitted, the TSR can then be loaded with new data from the TXR register, if it is available. It should be noted that the TSR register, unlike many other registers, is not directly mapped into the Data Memory area and as such is not available to the application program for direct read/write operations. An actual transmission of data will normally be enabled when the TXEN bit is set, but the data will not be transmitted until the TXR register has been loaded with data and the baud rate generator has defined a shift clock source. However, the transmission can also be initiated by first loading data into the TXR register, after which the TXEN bit can be set. When a transmission of data begins, the TSR is normally empty, in which case a transfer to the TXR register will result in an immediate transfer to the TSR. If during a transmission the TXEN bit is cleared, the transmission will immediately cease and the transmitter will be reset. The TX output pin will then return to having a normal general purpose I/O pin function.

- Transmitting data

  When the UART is transmitting data, the data is shifted on the TX pin from the shift register, with the least significant bit first. In the transmit mode, the TXR register forms a buffer between the internal bus and the transmitter shift register. It should be noted that if 9-bit data format has been selected, then the MSB will be taken from the TX8 bit in the UCR1 register. The steps to initiate a data transfer can be summarized as follows:

  - Make the correct selection of the BNO, PRT, PREN and STOPS bits to define the required word length, parity type and number of stop bits.
  - Setup the BRG register to select the desired baud rate.
  - Set the TXEN bit to ensure that the TX pin is used as a UART transmitter pin and not as an I/O pin.
  - Access the USR register and write the data that is to be transmitted into the TXR register. Note that this step will clear the TXIF bit.
  - This sequence of events can now be repeated to send additional data.

  It should be noted that when TXIF=0, data will be inhibited from being written to the TXR register. Clearing the TXIF flag is always achieved using the following software sequence:

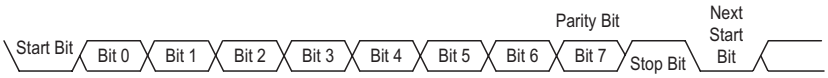
  1. A USR register access
  2. A TXR register write execution

  The read-only TXIF flag is set by the UART hardware and if set indicates that the TXR register is empty and that other data can now be written into the TXR register without overwriting the previous data. If the TEIE bit is set then the TXIF flag will generate an interrupt.

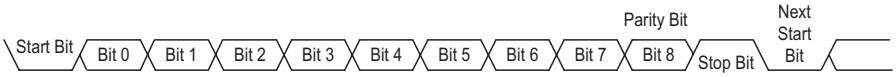
  During a data transmission, a write instruction to the TXR register will place the data into the TXR register, which will be copied to the shift register at the end of the present transmission. When there is no data transmission in progress, a write instruction to the TXR register will place the data directly into the shift register, resulting in the commencement of data transmission, and the TXIF bit being immediately set. When a frame transmission is complete, which happens after stop bits are sent or after the break frame, the TIDLE bit will be set. To clear the TIDLE bit the following software sequence is used:

  1. A USR register access
  2. A TXR register write execution

  Note that both the TXIF and TIDLE bits are cleared by the same software sequence.



8-Bit Data Format



9-Bit Data Format

- Transmit break

  If the TXBRK bit is set then break characters will be sent on the next transmission. Break character transmission consists of a start bit, followed by 13× N '0' bits and stop bits, where N=1, 2, etc. If a break character is to be transmitted then the TXBRK bit must be first set by the application program, then cleared to generate the stop bits. Transmitting a break character will not generate a transmit interrupt. Note that a break condition length is at least 13 bits long. If the TXBRK bit is continually kept at a logic high level then the transmitter circuitry will transmit continuous break characters. After the application program has cleared the TXBRK bit, the transmitter will finish transmitting the last break character and subsequently send out one or two stop bits. The automatic logic highs at the end of the last break character will ensure that the start bit of the next frame is recognized.

- UART receiver
  - Introduction

    The UART is capable of receiving word lengths of either 8 or 9 bits. If the BNO bit is set, the word length will be set to 9 bits with the MSB being stored in the RX8 bit of the UCR1 register. At the receiver core lies the Receive Serial Shift Register, commonly known as the RSR. The data which is received on the RX external input pin, is sent to the data recovery block. The data recovery block operating speed is 16 times that of the baud rate, while the main receive serial shifter operates at the baud rate. After the RX pin is sampled for the stop bit, the received data in RSR is transferred to the receive data register, if the register is empty. The data which is received on the external RX input pin is sampled three times by a majority detect circuit to determine the logic level that has been placed onto the RX pin. It should be noted that the RSR register, unlike many other registers, is not directly mapped into the Data Memory area and as such is not available to the application program for direct read/write operations.
  - Receiving data

    When the UART receiver is receiving data, the data is serially shifted in on the external RX input pin, LSB first. In the read mode, the RXR register forms a buffer between the internal bus and the receiver shift register. The RXR register is a two byte deep FIFO data buffer, where two bytes can be held in the FIFO while a third byte can continue to be received. Note that the application program must ensure that the data is read from RXR before the third byte has been completely shifted in, otherwise this third byte will be discarded and an overrun error OERR will be subsequently indicated. The steps to initiate a data transfer can be summarized as follows:
    - Make the correct selection of BNO, PRT, PREN and STOPS bits to define the word length, parity type and number of stop bits.
    - Setup the BRG register to select the desired baud rate.
    - Set the RXEN bit to ensure that the RX pin is used as a UART receiver pin and not as an I/O pin.

    At this point the receiver will be enabled which will begin to look for a start bit.

    When a character is received the following sequence of events will occur:
    - The RXIF bit in the USR register will be set when RXR register has data available, at least one more character can be read.
    - When the contents of the shift register have been transferred to the RXR register, then if the RIE bit is set, an interrupt will be generated.
    - If during reception, a frame error, noise error, parity error, or an overrun error has been detected, then the error flags can be set.

    The RXIF bit can be cleared using the following software sequence:
    1. A USR register access
    2. An RXR register read execution
  - Receive break

    Any break character received by the UART will be managed as a framing error. The receiver will count and expect a certain number of bit times as specified by the values programmed into the BNO and STOPS bits. If the break is much longer than 13 bit times, the reception will be considered as complete after the number of bit times specified by BNO and STOPS. The RXIF bit is set, FERR is set, zeros are loaded into the receive data register, interrupts are generated if appropriate and the RIDLE bit is set. If a long break signal has been detected and the receiver has received a start bit, the data bits and the invalid stop bit, which sets the FERR flag, the receiver must wait for a valid stop bit before looking for the next start bit. The receiver will not make the assumption that the break condition on the line is the next start bit. A break is regarded as a character that contains only zeros with the FERR flag set. The break character will be loaded into the buffer and no further data will be received until stop bits are received. It should be noted that the RIDLE read only flag will go high when the stop bits have not yet been received. The reception of a break character on the UART registers will result in the following:
    - The framing error flag, FERR, will be set.
    - The receive data register, RXR, will be cleared.
    - The OERR, NF, PERR, RIDLE or RXIF flags will possibly be set.
  - Idle status

    When the receiver is reading data, which means it will be in between the detection of a start bit and the reading of a stop bit, the receiver status flag in the USR register, otherwise known as the RIDLE flag, will have a zero value. In between the reception of a stop bit and the detection of the next start bit, the RIDLE flag will have a high value, which indicates the receiver is in an idle condition.

- Receiver interrupt
  The read only receive interrupt flag RXIF in the USR register is set by an edge generated by the receiver. An interrupt is generated if RIE=1, when a word is transferred from the Receive Shift Register, RSR, to the Receive Data Register, RXR. An overrun error can also generate an interrupt if RIE=1.

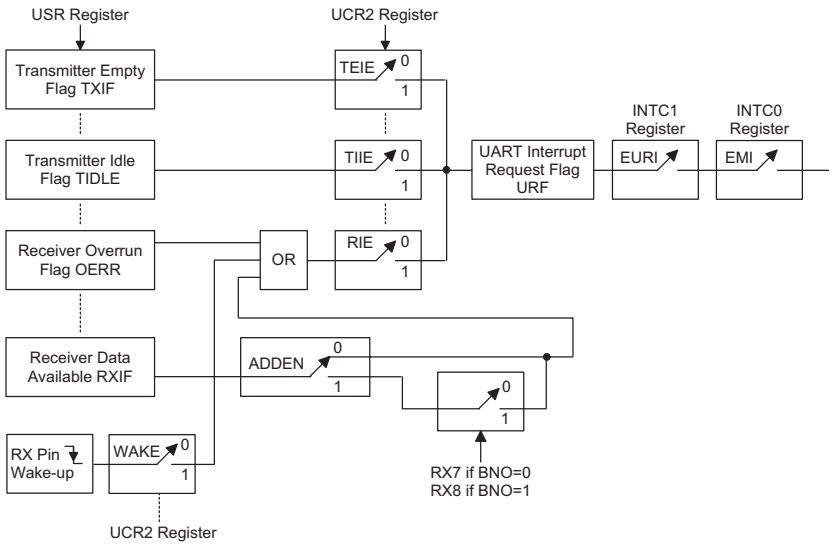
• Managing receiver errors
  Several types of reception errors can occur within the UART module, the following section describes the various types and how they are managed by the UART.

- Overrun Error - OERR flag
  The RXR register is composed of a two byte deep FIFO data buffer, where two bytes can be held in the FIFO register, while a third byte can continue to be received. Before this third byte has been entirely shifted in, the data should be read from the RXR register. If this is not done, the overrun error flag OERR will be consequently indicated.
  In the event of an overrun error occurring, the following will happen:
  - The OERR flag in the USR register will be set.
  - The RXR contents will not be lost.
  - The shift register will be overwritten.
  - An interrupt will be generated if the RIE bit is set.

  The OERR flag can be cleared by an access to the USR register followed by a read to the RXR register.
- Noise Error - NF Flag
  Over-sampling is used for data recovery to identify valid incoming data and noise. If noise is detected within a frame the following will occur:
  - The read only noise flag, NF, in the USR register will be set on the rising edge of the RXIF bit.
  - Data will be transferred from the Shift register to the RXR register.
  - No interrupt will be generated. However this bit rises at the same time as the RXIF bit which itself generates an interrupt.

  Note that the NF flag is reset by a USR register read operation followed by an RXR register read operation.
- Framing Error - FERR Flag
  The read only framing error flag, FERR, in the USR register, is set if a zero is detected instead of stop bits. If two stop bits are selected, both stop bits must be high, otherwise the FERR flag will be set. The FERR flag is buffered along with the received data and is cleared on any reset.
- Parity Error - PERR Flag
  The read only parity error flag, PERR, in the USR register, is set if the parity of the received word is incorrect. This error flag is only applicable if the parity is enabled, PREN = 1, and if the parity type, odd or even is selected. The read only PERR flag is buffered along with the received data bytes. It is cleared on any reset. It should be noted that the FERR and PERR flags are buffered along with the corresponding word and should be read before reading the data word.

• UART interrupt scheme
  The UART internal function possesses its own internal interrupt and independent interrupt vector. Several individual UART conditions can generate an internal UART interrupt. These conditions are, a transmitter data register empty, transmitter idle, receiver data available, receiver overrun, address detect and an RX pin wake-up. When any of these conditions are created, if the UART interrupt is enabled and the stack is not full, the program will jump to the UART interrupt vector where it can be serviced before returning to the main program. Four of these conditions, have a corresponding USR register flag, which will generate a UART interrupt if its associated interrupt enable flag in



UART Interrupt Scheme

the UCR2 register is set. The two transmitter interrupt conditions have their own corresponding enable bits, while the two receiver interrupt conditions have a shared enable bit. These enable bits can be used to mask out individual UART interrupt sources.

The address detect condition, which is also a UART interrupt source, does not have an associated flag, but will generate a UART interrupt when an address detect condition occurs if its function is enabled by setting the ADDEN bit in the UCR2 register. An RX pin wake-up, which is also a UART interrupt source, does not have an associated flag, but will generate a UART interrupt if the microcontroller is woken up by a low going edge on the RX pin, if the WAKE and RIE bits in the UCR2 register are set. Note that in the event of an RX wake-up interrupt occurring, there will be a delay of 1024 system clock cycles before the system resumes normal operation.

Note that the USR register flags are read only and cannot be cleared or set by the application program, neither will they be cleared when the program jumps to the corresponding interrupt servicing routine, as is the case for some of the other interrupts. The flags will be cleared automatically when certain actions are taken by the UART, the details of which are given in the UART register section. The overall UART interrupt can be disabled or enabled by the EURI bit in the INTC1 interrupt control register to prevent a UART interrupt from occurring.

- Address detect mode

  Setting the Address Detect Mode bit, ADDEN, in the UCR2 register, enables this special mode. If this bit is enabled then an additional qualifier will be placed on the generation of a Receiver Data Available interrupt, which is requested by the RXIF flag. If the ADDEN bit is enabled, then when data is available, an interrupt will only be generated, if the highest received bit has a high value. Note that the EURI and EMI interrupt enable bits must also be enabled for correct interrupt generation. This highest address bit is the 9th bit if BNO=1 or the 8th bit if BNO=0. If this bit is high, then the received word will be defined as an address rather than data. A Data Available interrupt will be generated every time the last bit of the received word is set. If the ADDEN bit is not enabled, then a Receiver Data Available interrupt will be generated each time the RXIF flag is set, irrespective of the data last bit status. The address detect mode and parity enable are mutually exclusive functions. Therefore if the address detect mode is enabled, then to ensure correct operation, the parity function should be disabled by resetting the parity enable bit to zero.

ADDEN	Bit 9 if BNO=1, Bit 8 if BNO=0	UART Interrupt Generated
0	0	√
	1	√
1	0	X
	1	√

ADDEN Bit Function

- UART operation in power down mode

  When the MCU is in the Power Down Mode the UART will cease to function. When the device enters the Power Down Mode, all clock sources to the module are shutdown. If the MCU enters the Power Down Mode while a transmission is still in progress, then the transmission will be terminated and the external TX transmit pin will be forced to a logic high level. In a similar way, if the MCU enters the Power Down Mode while receiving data, then the reception of data will likewise be terminated. When the MCU enters the Power Down Mode, note that the USR, UCR1, UCR2, transmit and receive registers, as well as the BRG register will not be affected.

  The UART function contains a receiver RX pin wake-up function, which is enabled or disabled by the WAKE bit in the UCR2 register. If this bit, along with the UART enable bit, UARTEN, the receiver enable bit, RXEN and the receiver interrupt bit, RIE, are all set before the MCU enters the Power Down Mode, then a falling edge on the RX pin will wake-up the MCU from the Power Down Mode. Note that as it takes 1024 system clock cycles after a wake-up, before normal microcontroller operation resumes, any data received during this time on the RX pin will be ignored.

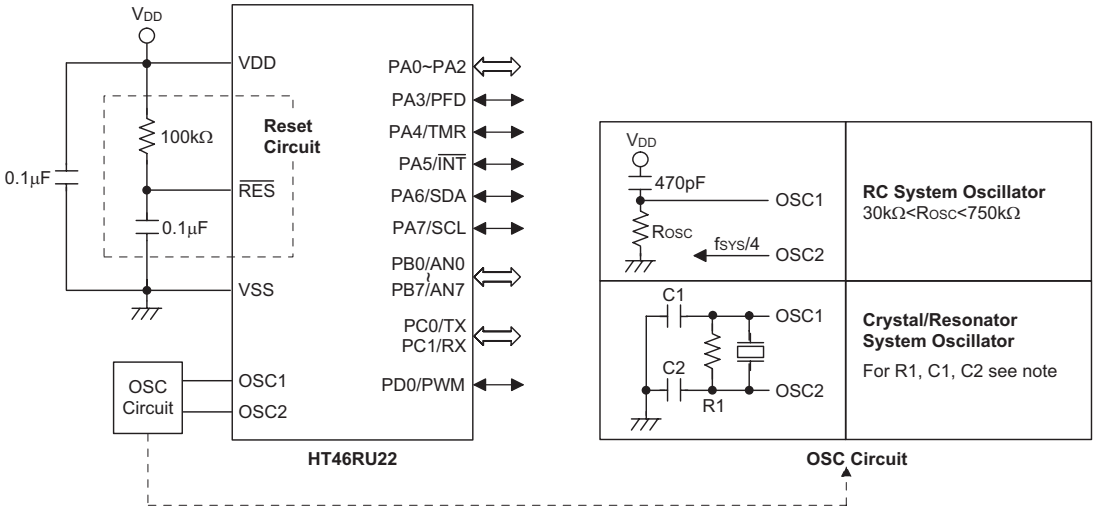
  For a UART wake-up interrupt to occur, in addition to the bits for the wake-up being set, the global interrupt enable bit, EMI, and the UART interrupt enable bit, EURI must also be set. If these two bits are not set then only a wake up event will occur and no interrupt will be generated. Note also that as it takes 1024 system clock cycles after a wake-up before normal microcontroller resumes, the UART interrupt will not be generated until after this time has elapsed.

Options

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

No.	Options
1	OSC type selection. This option is to decide if an RC or crystal oscillator is chosen as system clock.
2	WDT source selection. There are three types of selection: On-chip RC oscillator, instruction clock or disable the WDT.
3	CLRWDT times selection. This option defines how to clear the WDT by instruction. ″One time″ means that the CLR WDT instruction can clear the WDT. ″Two times″ means only if both of the CLR WDT1 and CLR WDT2 instructions have been executed, then WDT can be cleared.
4	Wake-up selection. This option defines the wake-up function activity. External I/O pins (PA only) all have the capability to wake-up the chip from a HALT.
5	Pull-high selection. This option is to decide whether a pull-high resistance is visible or not in the input mode of the I/O ports. PA0~PA7, can be independently selected.
6	PFD selection. PA3: Level output or PFD output
7	PWM selection: (7+1) or (6+2) mode PD0: Level output or PWM output
8	WDT time-out period selection. $2^{12}/f_S$~$2^{13}/f_S$, $2^{13}/f_S$~$2^{14}/f_S$, $2^{14}/f_S$~$2^{15}/f_S$, $2^{15}/f_S$~$2^{16}/f_S$.
9	Low voltage reset selection: Enable or disable LVR function.
10	I^2C Bus function: Enable or disable

Application Circuits



Note: 1. Crystal/resonator system oscillators
For crystal oscillators, C1 and C2 are only required for some crystal frequencies to ensure oscillation. For resonator applications C1 and C2 are normally required for oscillation to occur. For most applications it is not necessary to add R1. However if the LVR function is disabled, and if it is required to stop the oscillator when VDD falls below its operating range, it is recommended that R1 is added. The values of C1 and C2 should be selected in consultation with the crystal/resonator manufacturer specifications.

2. Reset circuit
The reset circuit resistance and capacitance values should be chosen to ensure that VDD is stable and remains within its operating voltage range before the $\overline{RES}$ pin reaches a high level. Ensure that the length of the wiring connected to the $\overline{RES}$ pin is kept as short as possible, to avoid noise interference.

3. For applications where noise may interfere with the reset circuit and for details on the oscillator external components, refer to Application Note HA0075E for more information.

Instruction Set

Introduction

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

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

Instruction Timing

Most instructions are implemented within one instruction cycle. The exceptions to this are branch, call, or table read instructions where two instruction cycles are required. One instruction cycle is equal to 4 system clock cycles, therefore in the case of an 8MHz system oscillator, most instructions would be implemented within 0.5μs and branch or call instructions would be implemented within 1μs. Although instructions which require one more cycle to implement are generally limited to the JMP, CALL, RET, RETI and table read instructions, it is important to realize that any other instructions which involve manipulation of the Program Counter Low register or PCL will also take one more cycle to implement. As instructions which change the contents of the PCL will imply a direct jump to that new address, one more cycle will be required. Examples of such instructions would be ″CLR PCL″ or ″MOV PCL, A″. For the case of skip instructions, it must be noted that if the result of the comparison involves a skip operation then this will also take one more cycle, if no skip is involved then only one cycle is required.

Moving and Transferring Data

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

Arithmetic Operations

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

Logical and Rotate Operations

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

Branches and Control Transfer

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

Bit Operations

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

Table Read Operations

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

Other Operations

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

Instruction Set Summary

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

Table conventions:

x: Bits immediate data

m: Data Memory address

A: Accumulator

i: 0~7 number of bits

addr: Program memory address

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

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

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

Instruction Definition

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

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

Operation ACC ← ACC + [m] + C

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

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

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

Operation [m] ← ACC + [m] + C

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

ADD A,[m] Add Data Memory to ACC

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

Operation ACC ← ACC + [m]

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

ADD A,x Add immediate data to ACC

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

Operation ACC ← ACC + x

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

ADDM A,[m] Add ACC to Data Memory

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

Operation [m] ← ACC + [m]

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

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

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

Operation ACC ← ACC "AND" [m]

Affected flag(s) Z

AND A,x Logical AND immediate data to ACC

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

Operation ACC ← ACC "AND" x

Affected flag(s) Z

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

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

Operation [m] ← ACC "AND" [m]

Affected flag(s) Z

CALL addr Subroutine call

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

Operation:
Stack ← Program Counter + 1
Program Counter ← addr

Affected flag(s): None

CLR [m] Clear Data Memory

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

Operation: [m] ← 00H

Affected flag(s): None

CLR [m].i Clear bit of Data Memory

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

Operation: [m].i ← 0

Affected flag(s): None

CLR WDT Clear Watchdog Timer

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

Operation:
WDT cleared
TO ← 0
PDF ← 0

Affected flag(s): TO, PDF

CLR WDT1 Pre-clear Watchdog Timer

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

Operation:
WDT cleared
TO ← 0
PDF ← 0

Affected flag(s): TO, PDF

CLR WDT2 Pre-clear Watchdog Timer

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

Operation:
WDT cleared
TO ← 0
PDF ← 0

Affected flag(s): TO, PDF

CPL [m] Complement Data Memory

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

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

Affected flag(s) Z

CPLA [m] Complement Data Memory with result in ACC

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

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

Affected flag(s) Z

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

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

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

Affected flag(s) C

DEC [m] Decrement Data Memory

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

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

Affected flag(s) Z

DECA [m] Decrement Data Memory with result in ACC

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

Operation $ACC \leftarrow [m] - 1$

Affected flag(s) Z

HALT Enter power down mode

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

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

Affected flag(s) TO, PDF

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

OR A,x Logical OR immediate data to ACC

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

Operation: ACC ← ACC ″OR″ x

Affected flag(s): Z

ORM A,[m] Logical OR ACC to Data Memory

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

Operation: [m] ← ACC ″OR″ [m]

Affected flag(s): Z

RET Return from subroutine

Description: The Program Counter is restored from the stack. Program execution continues at the restored address.

Operation: Program Counter ← Stack

Affected flag(s): None

RET A,x Return from subroutine and load immediate data to ACC

Description: The Program Counter is restored from the stack and the Accumulator loaded with the specified immediate data. Program execution continues at the restored address.

Operation: Program Counter ← Stack
ACC ← x

Affected flag(s): None

RETI Return from interrupt

Description: The Program Counter is restored from the stack and the interrupts are re-enabled by setting the EMI bit. EMI is the master interrupt global enable bit. If an interrupt was pending when the RETI instruction is executed, the pending Interrupt routine will be processed before returning to the main program.

Operation: Program Counter ← Stack
EMI ← 1

Affected flag(s): None

RL [m] Rotate Data Memory left

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

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

Affected flag(s): None

RLA [m] Rotate Data Memory left with result in ACC

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

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

Affected flag(s): None

RLC [m] Rotate Data Memory left through Carry

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

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

Affected flag(s) C

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

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

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

Affected flag(s) C

RR [m] Rotate Data Memory right

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

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

Affected flag(s) None

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

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

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

Affected flag(s) None

RRC [m] Rotate Data Memory right through Carry

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

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

Affected flag(s) C

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

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

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

Affected flag(s) C

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

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

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

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

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

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

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

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

SDZ [m] Skip if decrement Data Memory is 0

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

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

Affected flag(s) None

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

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

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

Affected flag(s) None

SET [m] Set Data Memory

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

Operation $[m] \leftarrow FFH$

Affected flag(s) None

SET [m].i Set bit of Data Memory

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

Operation $[m].i \leftarrow 1$

Affected flag(s) None

SIZ [m] Skip if increment Data Memory is 0

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

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

Affected flag(s): None

SIZA [m] Skip if increment Data Memory is zero with result in ACC

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

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

Affected flag(s): None

SNZ [m].i Skip if bit i of Data Memory is not 0

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

Operation: Skip if [m].i $\neq$ 0

Affected flag(s): None

SUB A,[m] Subtract Data Memory from ACC

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

Operation: ACC ← ACC – [m]

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

SUBM A,[m] Subtract Data Memory from ACC with result in Data Memory

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

Operation: [m] ← ACC – [m]

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

SUB A,x Subtract immediate data from ACC

Description: The immediate data specified by the code is subtracted from the contents of the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.

Operation: ACC ← ACC – x

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

SWAP [m] Swap nibbles of Data Memory

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

Operation [m].3~[m].0 ↔ [m].7 ~ [m].4

Affected flag(s) None

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

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

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

Affected flag(s) None

SZ [m] Skip if Data Memory is 0

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

Operation Skip if [m] = 0

Affected flag(s) None

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

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

Operation ACC ← [m]
Skip if [m] = 0

Affected flag(s) None

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

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

Operation Skip if [m].i = 0

Affected flag(s) None

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

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

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

Affected flag(s) None

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

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

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

Affected flag(s) None

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

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

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

Package Information

24-pin SKDIP (300mil) Outline Dimensions

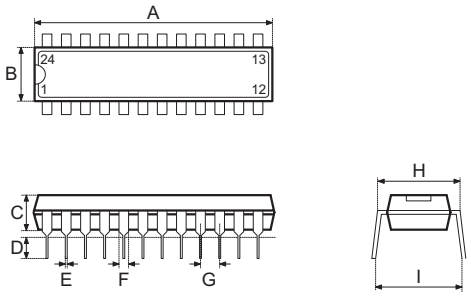


Fig1. Full Lead Packages

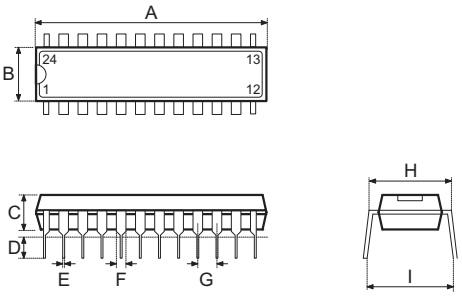


Fig2. 1/2 Lead Packages

• MS-001d (see fig1)

Symbol	Dimensions in mil		
	Min.	Nom.	Max.
A	1230	—	1280
B	240	—	280
C	115	—	195
D	115	—	150
E	14	—	22
F	45	—	70
G	—	100	—
H	300	—	325
I	—	—	430

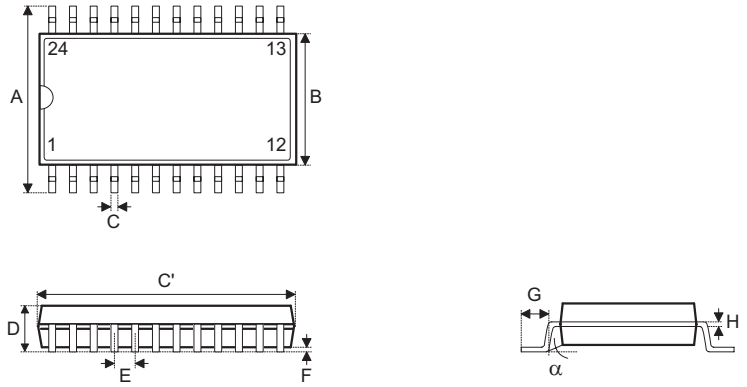
• MS-001d (see fig2)

Symbol	Dimensions in mil		
	Min.	Nom.	Max.
A	1160	—	1195
B	240	—	280
C	115	—	195
D	115	—	150
E	14	—	22
F	45	—	70
G	—	100	—
H	300	—	325
I	—	—	430

- MO-095a (see fig2)

Symbol	Dimensions in mil		
	Min.	Nom.	Max.
A	1145	—	1185
B	275	—	295
C	120	—	150
D	110	—	150
E	14	—	22
F	45	—	60
G	—	100	—
H	300	—	325
I	—	—	430

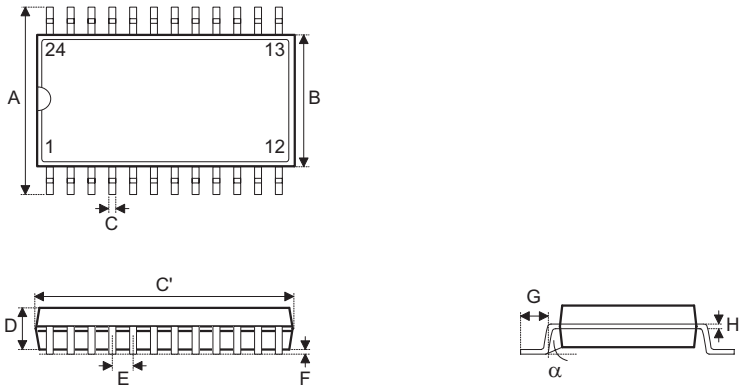
24-pin SOP (300mil) Outline Dimensions



• MS-013

Symbol	Dimensions in mil		
	Min.	Nom.	Max.
A	393	—	419
B	256	—	300
C	12	—	20
C′	598	—	613
D	—	—	104
E	—	50	—
F	4	—	12
G	16	—	50
H	8	—	13
α	0°	—	8°

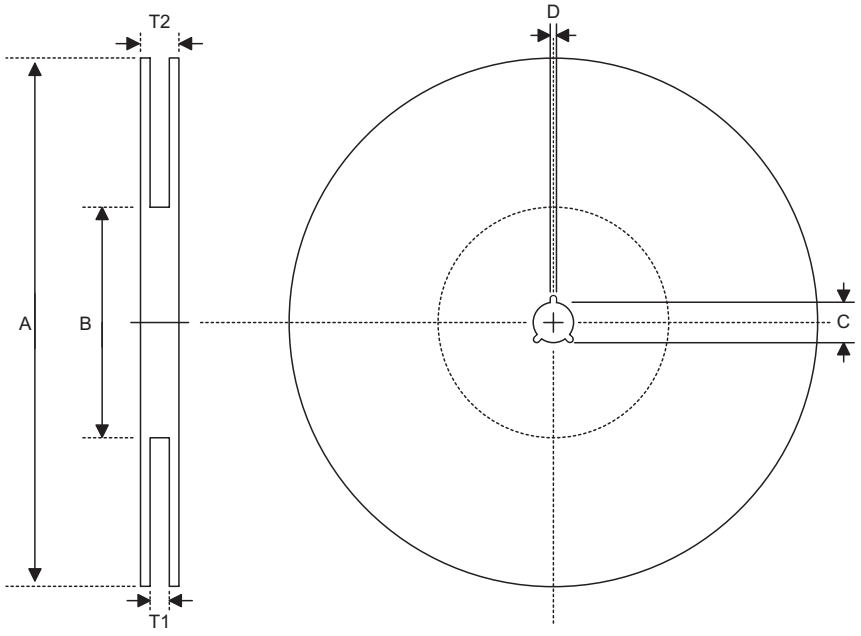
24-pin SSOP (150mil) Outline Dimensions



Symbol	Dimensions in mil		
	Min.	Nom.	Max.
A	228	—	244
B	150	—	157
C	8	—	12
C′	335	—	346
D	54	—	60
E	—	25	—
F	4	—	10
G	22	—	28
H	7	—	10
α	0°	—	8°

Product Tape and Reel Specifications

Reel Dimensions



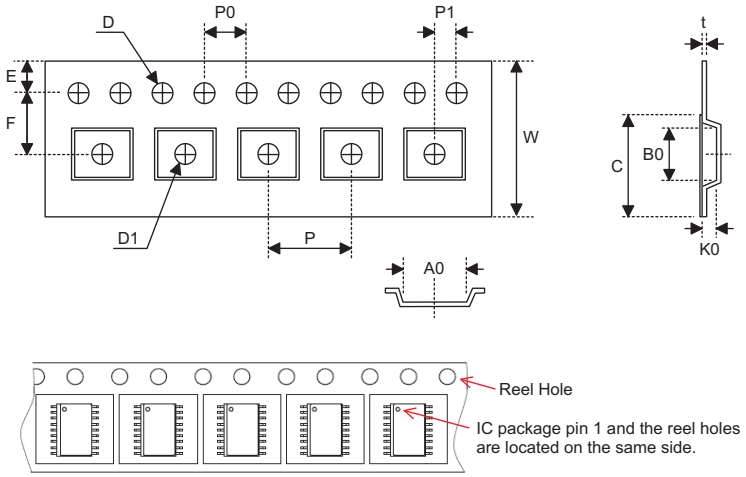
SOP 24W

Symbol	Description	Dimensions in mm
A	Reel Outer Diameter	330.0±1.0
B	Reel Inner Diameter	100.0±1.5
C	Spindle Hole Diameter	$13.0^{+0.5/-0.2}$
D	Key Slit Width	2.0±0.5
T1	Space Between Flange	$24.8^{+0.3/-0.2}$
T2	Reel Thickness	30.2±0.2

SSOP 24S (150mil)

Symbol	Description	Dimensions in mm
A	Reel Outer Diameter	330.0±1.0
B	Reel Inner Diameter	100.0±1.5
C	Spindle Hole Diameter	$13.0^{+0.5/-0.2}$
D	Key Slit Width	2.0±0.5
T1	Space Between Flange	$16.8^{+0.3/-0.2}$
T2	Reel Thickness	22.2±0.2

Carrier Tape Dimensions



SOP 24W

Symbol	Description	Dimensions in mm
W	Carrier Tape Width	24.0±0.3
P	Cavity Pitch	12.0±0.1
E	Perforation Position	1.75±0.1
F	Cavity to Perforation (Width Direction)	11.5±0.1
D	Perforation Diameter	$1.55^{+0.10/-0.00}$
D1	Cavity Hole Diameter	$1.50^{+0.25/-0.00}$
P0	Perforation Pitch	4.0±0.1
P1	Cavity to Perforation (Length Direction)	2.0±0.1
A0	Cavity Length	10.9±0.1
B0	Cavity Width	15.9±0.1
K0	Cavity Depth	3.1±0.1
t	Carrier Tape Thickness	0.35±0.05
C	Cover Tape Width	21.3±0.1

SSOP 24S (150mil)

Symbol	Description	Dimensions in mm
W	Carrier Tape Width	$16.0^{+0.3/-0.1}$
P	Cavity Pitch	8.0±0.1
E	Perforation Position	1.75±0.10
F	Cavity to Perforation (Width Direction)	7.5±0.1
D	Perforation Diameter	$1.5^{+0.1/-0.0}$
D1	Cavity Hole Diameter	$1.50^{+0.25/-0.00}$
P0	Perforation Pitch	4.0±0.1
P1	Cavity to Perforation (Length Direction)	2.0±0.1
A0	Cavity Length	6.5±0.1
B0	Cavity Width	9.5±0.1
K0	Cavity Depth	2.1±0.1
t	Carrier Tape Thickness	0.30±0.05
C	Cover Tape Width	13.3±0.1

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 (China) Inc. (Dongguan Sales Office)
Building No. 10, Xinzhu Court, (No. 1 Headquarters), 4 Cuizhu Road, Songshan Lake, Dongguan, China 523808
Tel: 86-769-2626-1300
Fax: 86-769-2626-1311

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