# HT46R51A/HT46R52A
# A/D Type 8-Bit OTP MCU

## Technical Document

- Tools Information
- FAQs
- Application Note
  - HA0003E Communicating between the HT48 & HT46 Series MCUs and the HT93LC46 EEPROM
  - HA0004E HT48 & HT46 MCU UART Software Implementation Method
  - HA0084E NiMH Battery Charger Demo Board - Using the HT46R52

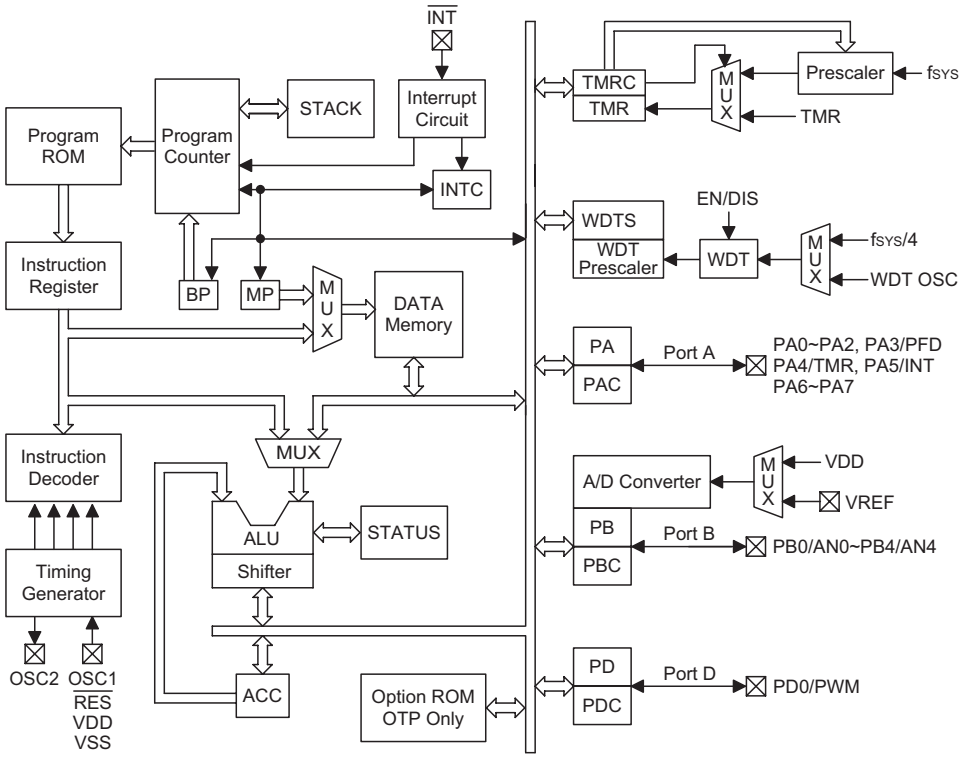
## Features

- Low-power fully static CMOS design
- Operating voltage:
  $f_{SYS}$=4MHz: 2.2V~5.5V
  $f_{SYS}$=8MHz: 3.3V~5.5V
- Program Memory:
  1K×15 OTP (HT46R51A)
  2K×15 OTP (HT46R52A)
- Data memory:
  96×8 RAM (HT46R51A)
  128×8 RAM (HT46R52A)
- A/D converter: 12bits×5Ch
  External A/D converter reference voltage input pin
- 14 bidirectional I/O lines
- 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
- 6-level subroutine nesting
- Watchdog Timer
- Low voltage reset function
- HALT function
- Up to 0.5μs instruction cycle with 8MHz system clock at $V_{DD}$=5V
- 1-channel 8-bit PWM output shared with an I/O line
- PFD function
- Bit manipulation instruction
- Table read instruction
- 63 powerful instructions
- All instructions in one or two machine cycles
- 18-pin DIP, 20-pin SOP/SSOP package

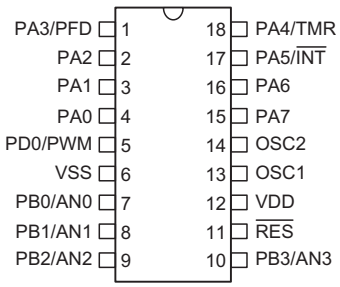
## General Description

The HT46R51A/HT46R52A 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, timer functions, oscillator options, multi-channel A/D converter, Pulse Width Modulation function, HALT and wake-up functions, watchdog timer, as well as low cost, enhance the versatility of these devices to suit a wide range of A/D application possibilities such as sensor signal processing, chargers, motor driving, industrial control, consumer products, subsystem controllers, etc.

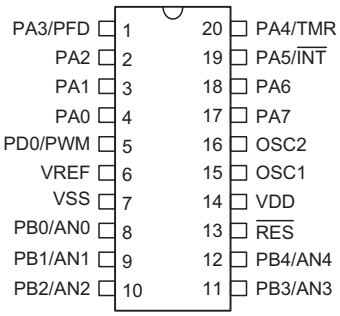
## Block Diagram




## Pin Assignment




**HT46R51A/HT4R52A**
**—18 DIP-A**




**HT46R51A/HT4R52A**
**—20 SOP-A/SSOP-A**

## Pin Description

| Pin Name | I/O | Options | Description |
|---|---|---|---|
| PA0~PA2<br>PA3/PFD<br>PA4/TMR<br>PA5/$\overline{\text{INT}}$<br>PA6~PA7 | I/O | Pull-high<br>Wake-up<br>PA3 or PFD | Bidirectional 8-bit input/output port. Each individual bit on this port can be configured as a wake-up input by configuration option. Software instructions determine if the pin is a CMOS output or Schmitt trigger input. Configuration options determine which pins on this port have pull-high resistors. The PFD, TMR and external interrupt input are pin-shared with PA3, PA4, and PA5 respectively. |
| PB0/AN0<br>PB1/AN1<br>PB2/AN2<br>PB3/AN3<br>PB4/AN4 | I/O | Pull-high | Bidirectional 5-bit input/output port. Software instructions determine the CMOS output or Schmitt trigger input with or without pull-high resistor. Configuration options determine which pins on this port have pull-high resistors. PB is pin-shared with the A/D input pins. The A/D inputs are selected via software instructions Once selected as an A/D input, the I/O function and pull-high resistor functions are disabled automatically. |
| PD0/PWM | I/O | Pull-high<br>PD0 or PWM | Bidirectional 1-bit input/output port. Software instructions determine the CMOS output or Schmitt trigger input with or without pull-high resistor. One configuration option determines which pin on this port has pull-high resistor. PD0 is pin-shared with the PWM output selected via configuration option. |
| OSC1<br>OSC2 | I<br>O | Crystal or RC | OSC1, OSC2 are connected to an external RC network or external crystal (determined by configuration option) for the internal system clock. For external RC system clock operation, OSC2 is an output pin for 1/4 system clock. |
| $\overline{\text{RES}}$ | I | — | Schmitt trigger reset input, active low |
| VDD | — | — | Positive power supply |
| VSS | — | — | Negative power supply, ground |
| VREF | I | — | A/D Converter Reference Input voltage pins. Connect this pin to the desired A/D reference voltage. The VREF pin is connected to $V_{DD}$ for the 18-pin DIP package |

## 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 disabled | — | 0.6 | 1.5 | mA |
| | | 5V | | — | 2 | 4 | mA |
| $I_{DD2}$ | Operating Current (RC OSC) | 3V | No load, $f_{SYS}$=4MHz ADC disabled | — | 0.8 | 1.5 | mA |
| | | 5V | | — | 2.5 | 4 | mA |
| $I_{DD3}$ | Operating Current | 5V | No load, $f_{SYS}$=8MHz ADC disabled | — | 4 | 8 | mA |

| Symbol | Parameter | Test Conditions | | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|---|---|
| | | $V_{DD}$ | Conditions | | | | |
| $I_{STB1}$ | Standby Current (WDT Enabled) | 3V | No load, system HALT | — | — | 5 | μA |
| | | 5V | | — | — | 10 | μA |
| $I_{STB2}$ | Standby Current (WDT & AD 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 Voltage | — | Configuration option: 3V | 2.7 | 3 | 3.3 | V |
| $I_{OL}$ | I/O Port Sink Current | 3V | $V_{OL}=0.1V_{DD}$ | 4 | 8 | — | mA |
| | | 5V | | 10 | 20 | — | mA |
| $I_{OH}$ | I/O Port Source Current | 3V | $V_{OH}=0.9V_{DD}$ | –2 | –4 | — | mA |
| | | 5V | | –5 | –10 | — | mA |
| $R_{PH}$ | Pull-high Resistance of I/O Ports | 3V | — | 20 | 60 | 100 | kΩ |
| | | 5V | | 10 | 30 | 50 | kΩ |
| $V_{AD}$ | A/D Input Voltage | — | — | 0 | — | $V_{REF}$ | V |
| $V_{REF}$ | ADC Input Reference Voltage Range | — | — | 1.2 | — | VDD | V |
| DNL | ADC Differential Non-Linear | — | — | — | — | ±2 | LSB |
| INL | ADC Integral Non-Linear | — | — | — | ±2.5 | ±4 | LSB |
| RESOLU | Resolution | — | — | — | — | 12 | Bits |
| $I_{ADC}$ | 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) | — | 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 | — | $t_{SYS}$ |
| $t_{INT}$ | Interrupt Pulse Width | — | — | 1 | — | — | μs |
| $t_{AD}$ | A/D Clock Period | — | — | 1 | — | — | μs |
| $t_{ADC}$ | A/D Conversion Time | — | — | — | 80 | — | $t_{AD}$ |
| $t_{ADCS}$ | A/D Sampling Time | — | — | — | 32 | — | $t_{AD}$ |

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 4 system clock cycles.

Instruction fetching and execution are pipelined in such a way that a fetch and decoding takes an instruction cycle while execution take the next instruction cycle. 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

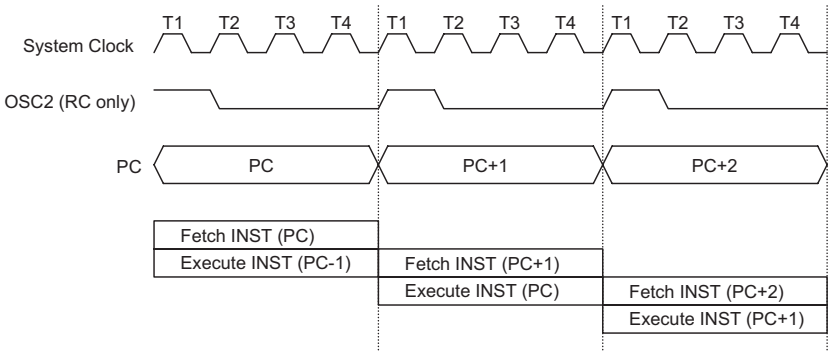
For HT46R51A, the program counter (PC) is 10 bits wide and controls the sequence in which the instructions stored in the program ROM are executed. The contents of the PC can specify a maximum of 1024 addresses.

For HT46R52A, the program counter (PC) is 11 bits wide and controls the sequence in which the instructions stored in the program ROM are executed. The contents of the PC can specify a maximum of 2048 addresses.

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

When executing a jump instruction, conditional skip execution, loading register, subroutine call or return from subroutine, initial reset, internal interrupt, external interrupt or return from interrupts, 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 to the next instruction.




**Execution Flow**

| Mode | Program Counter | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | *b10 | *b9 | *b8 | *b7 | *b6 | *b5 | *b4 | *b3 | *b2 | *b1 | *b0 |
| 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 |
| Skip | Program Counter+2 | | | | | | | | | | |
| Loading PCL | PC10 | PC9 | PC8 | @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: *b10~*b0: Program counter bits
S10~S0: Stack register bits
#10~#0: Instruction code bits
@7~@0: PCL bits, PC10~PC8: Original PC counter, remain unchanged
For the HT46R51A, since the program counter is 10 bits wide (b0~b9), the b10 columns in the table are not applicable.
For the HT46R52A, since the program counter is 11 bits wide (b0~b10)

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

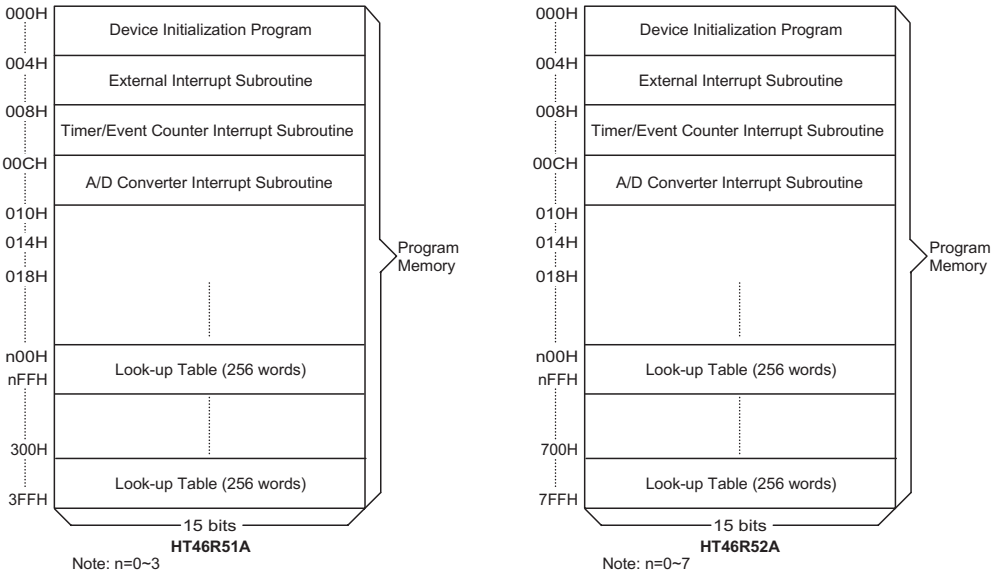
When a control transfer takes place, an additional dummy cycle is required.

**Program Memory – EPROM**

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 1024×15 (HT46R51A) or 2048×15 (HT46R52A) bits, addressed by the Program Counter and table pointer.

Certain locations in the ROM are reserved for special usage:

- Location 000H
  This location is reserved for program initialization. After a chip reset, the program always begins execution at location 000H.
- Location 004H
  This location is reserved for the external interrupt service program. If the $\overline{\text{INT}}$ input pin is activated, the interrupt is enabled and the stack is not full, the program begins execution at this location.
- Location 008H
  This location is reserved for the timer/event counter interrupt service program. If a timer interrupt results from a timer/event counter overflow, and the interrupt is enabled and the stack is not full, the program begins execution at location 008H.
- Location 00CH
  Location 00CH is reserved for the 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.
- Table location
  Any location in the program memory can be used as look-up tables. The instructions "TABRDC [m]" (the current page) 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). The lower-order byte table pointer TBLP (07H) are read/write registers, which indicate the table locations. Before accessing the table, the location has to 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. Given this, 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 main routine and the ISR, the interrupt should be disabled prior to the table read instruction. It will not be enabled until the TBLH in the main routine has been backed-up. All table related instructions require 2 cycles to complete the operation.




**Program Memory**

| Instruction | Table Location | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | b10 | b9 | b8 | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
| 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: b10~b0: Table location bits
P10~P8: Current program counter bits
@7~@0: Table pointer bits
For the Ht46R51A, since the program counter is 10 bits wide (b0~b9), the b10 column in the table are not applicable
For the HT46R52A, since the program counter is 11 bits wide (b0~b10)

**Stack Register – STACK**

This is a special part of the memory which is used to save the contents of the program counter only. The stack is organized into 6 levels and is neither part of the data nor part of the program space, and is neither readable nor writeable. The activated level is indexed by the stack pointer (SP) and is neither readable nor writeable. At the state of a subroutine call or an interrupt acknowledgment, the contents of the program counter are pushed onto the stack. At the end of the subroutine or an interrupt routine, signaled by a return instruction (RET or RETI), the program counter is restored to its previous value from the stack. After a chip reset, the SP will point to the top of the stack.

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

**Data Memory – RAM**

The data memory (RAM) is designed with 119×8 bits (HT46R51A), 151×8 bits (HT46R52A) and is divided into two functional groups, namely; special function registers (23×8 bits) and general purpose data memory (96×8bit for HT46R51A, 128×8bit for HT46R52A) most of which are readable/writable, although some are read only. The unused space before 28H is reserved for future expanded usage and reading these locations will return the result ″00H″. The general purpose data memory, addressed from 28H to 87H and 28H to AFH, is used for data and control information under instruction commands. All of the data memory areas can handle arithmetic, logic, increment, decrement and rotate operations directly. Except for some dedicated bits, each bit in the data memory can be set and reset by ″SET [m].i″ and ″CLR [m].i″. They are also indirectly accessible through memory pointer registers (MP0;01H or MP1;03H).

**Indirect Addressing Register**

Location 00H and 02H are indirect addressing registers that are not physically implemented. Any read/write operation of [00H] ([02H]) will access the data memory pointed to by MP0 (MP1). Reading location 00H (02H) itself indirectly will return the result ″00H″. Writing indirectly results in no operation. The memory pointer registers (MP0 and MP1) are 8-bit registers.

**Accumulator – ACC**

The accumulator closely relates to ALU operations. It is also mapped to location ″05H″ of the data memory which can operate with immediate data. The data movement between two data memories has to pass through the accumulator.

**Arithmetic and Logic Unit – ALU**

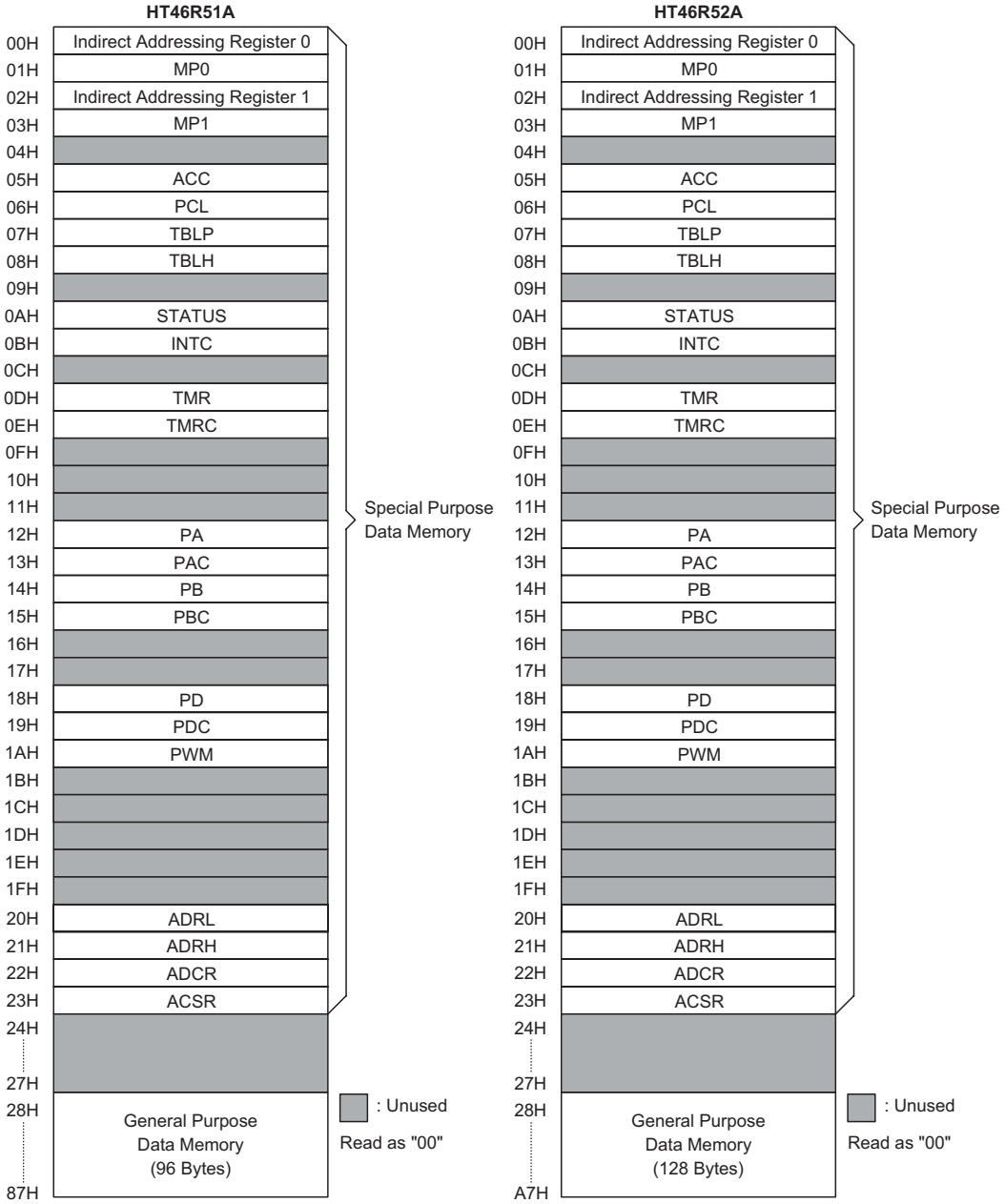
This circuit performs 8-bit arithmetic and logic operations. The ALU provides the following functions:

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

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

**Status Register – STATUS**

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




**RAM Mapping**

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

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

**Interrupts**

The device provides an external interrupt, an internal timer/event counter interrupt, and an A/D converter interrupt. The interrupt control register (INTC;0BH) contains the interrupt control bits to set the enable/disable and the interrupt request flags.

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

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

External interrupts are triggered by a high to low transition of $\overline{\text{INT}}$ and the related interrupt request flag (EIF; bit 4 of the INTC) 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 the INTC), which is normally caused by a timer overflow. After the interrupt is enabled, and the stack is not full, and the TF bit is set, a subroutine call to location ″08H″ occurs. The related interrupt request flag (TF) is reset, and the EMI bit is cleared to disable further maskable interrupts.

The A/D converter interrupt is initialized by setting the A/D converter request flag (ADF; bit 6 of the INTC), 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.

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 (if the stack is not full). To return from the interrupt subroutine, ″RET″ or ″RETI″ may be invoked. RETI will set the EMI bit to enable an interrupt service, but RET will not.

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

| Interrupt Source | Priority | Vector |
|---|---|---|
| External Interrupt | 1 | 04H |
| Timer/Event Counter Overflow | 2 | 08H |
| A/D Converter Interrupt | 3 | 0CH |

| Bit No. | Label | Function |
|---|---|---|
| 0 | EMI | Controls the master (global) interrupt (1= enable; 0= disable) |
| 1 | EEI | Controls the external interrupt (1= enable; 0= disable) |
| 2 | ETI | Controls the Timer/Event Counter interrupt (1= enable; 0= disable) |
| 3 | EADI | Control the A/D converter interrupt (1= enable; 0= disable) |
| 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.<br>Must be written as ″0″; otherwise may result in unpredictable operation. |

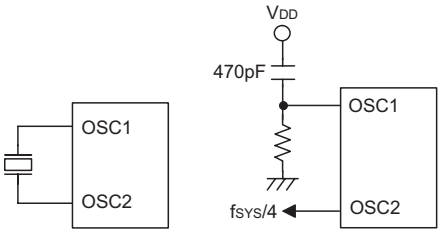
**INTC (0BH) Register**

EMI, EEI, ETI, and EADI are used to control the enabling/disabling of interrupts. These bits prevent the requested interrupt from being serviced. Once the interrupt request flags (TF, EIF, and ADF) are set, they will remain in the INTC register until the interrupts are serviced or cleared by a software instruction.

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

### Oscillator Configuration

There are two oscillator circuits in the microcontroller.




**System Oscillator**

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

If an RC oscillator is used, an external resistor between OSC1 and VSS is required and the resistance must range from 30kΩ to 750kΩ. 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 oscillator can be disabled by options to conserve power).

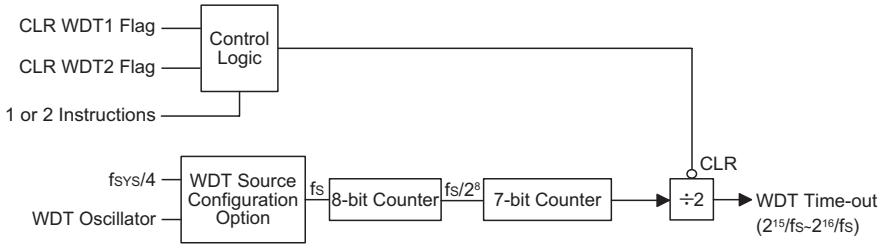
The WDT oscillator is a free running on-chip RC oscillator, and no external components are required. Even if the system enters the power down mode, the system clock is stopped, but the WDT oscillator still works with a period of approximately 65μs at 5V. The WDT oscillator can be disabled by option to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

### Watchdog Timer – WDT

The clock source of the WDT is implemented by a dedicated RC oscillator (WDT oscillator) or instruction clock (system clock divided by 4) decided by options. This timer is designed to prevent a software mal-function 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.

The WDT clock ($f_S$) is further divided by an internal counter to give longer watchdog time-outs. The division ratio is fixed by an internal counter which gives a $2^{15}$ fixed division ratio.

Once an internal WDT oscillator (RC oscillator with period of 65μs normally) is selected, it is divided by $2^{16}$ to get the time-out period of approximately 4.3s. This time-out period may vary with temperature, VDD and process variations.




**Watchdog Timer**

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

The WDT time-out period is fixed to $f_s/2^{16}$, because the "CLR WDT" or "CLR WDT1" and "CLR WDT2" instructions will clear the whole counter of the WDT.

### Power Down Operation – HALT

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

- The system oscillator is turned off but the WDT oscillator keeps running (if the WDT oscillator or the real time clock is selected).
- The contents of the on-chip RAM and registers remain unchanged
- The WDT and WDT prescaler will be cleared to zero. If the WDT clock source is from the RTC/WDT oscillator, the WDT will remain active, and if the WDT clock source is $f_{SYS}/4$, the WDT will stop running.
- All of the I/O ports maintain their original status
- The PDF flag is set and the TO flag is cleared

The system quits the HALT mode by way 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 examining the TO and PDF flags, the cause for a chip reset can be determined. The PDF flag is cleared by system power-up or by executing the "CLR WDT" instruction and is set when executing the "HALT" instruction. On the other hand, the TO flag is set if the WDT time-out occurs, and causes a wake-up that only resets the Program Counter and SP, and leaves the others in their original status.

The port A wake-up and interrupt methods can be considered as a continuation of normal execution. Each bit in port A can be independently selected to wake-up the device by options. Awakening from an I/O port stimulus, the program resumes execution of the next instruction. On the other hand, awakening from an interrupt, two sequence may occur. If the related interrupt is disabled or the interrupt is enabled but the stack is full, the program resumes execution at the next instruction. But if the interrupt is enabled, and the stack is not full, the regular interrupt response takes place. When an interrupt request flag is set before entering the "HALT" status, the system cannot be awakened using that interrupt. If wake-up events occur, it takes 1024 $t_{SYS}$ (system clock period) to resume normal operation. In other words, a dummy period is inserted after the wake-up. If the wake-up results from an interrupt acknowledgment, the actual interrupt subroutine execution is delayed by more than one cycle. However, if the Wake-up results in the next instruction execution, the execution will be performed immediately after the dummy period is finished.

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

### Reset

There are three ways in which a reset may occur:

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

The WDT time-out during HALT differs from other chip reset conditions, for it can perform a "warm reset" that resets only the Program Counter and SP, leaving the other circuits at their original state. Some registers remain unaffected during any other reset conditions. Most registers are reset to the "initial condition" when the re-

set conditions are met. Examining the PDF and TO flags, the program can distinguish between different "chip resets".

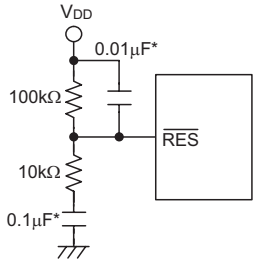
| TO | PDF | RESET Conditions |
|---|---|---|
| 0 | 0 | $\overline{RES}$ reset during power-up |
| u | u | $\overline{RES}$ reset during normal operation |
| 0 | 1 | $\overline{RES}$ wake-up HALT |
| 1 | u | WDT time-out during normal operation |
| 1 | 1 | WDT wake-up HALT |

Note: "u" stands for "unchanged"

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

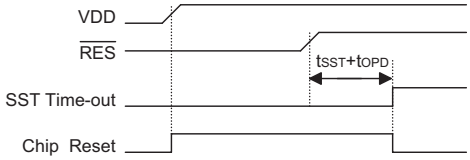
The functional unit chip reset status are shown below.

| | |
|---|---|
| Program Counter | 000H |
| Interrupt | Disable |
| Prescaler, Divider | Cleared |
| 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 |

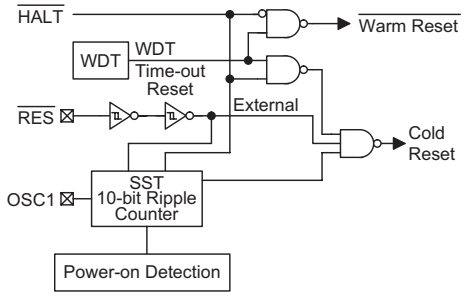



**Reset Circuit**

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




**Reset Timing Chart**




**Reset Configuration**

The register states are summarized below:

| 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 | 0000H | 0000H | 0000H | 0000H | 0000H |
| MP0 | xxxx xxxx | xxxx xxxx | xxxx xxxx | xxxx xxxx | uuuu uuuu |
| MP1 | xxxx xxxx | xxxx xxxx | xxxx xxxx | xxxx xxxx | uuuu uuuu |
| ACC | xxxx xxxx | uuuu uuuu | uuuu uuuu | uuuu uuuu | uuuu uuuu |
| TBLP | xxxx xxxx | uuuu uuuu | uuuu uuuu | uuuu uuuu | uuuu uuuu |
| TBLH | xxxx xxxx | uuuu uuuu | uuuu uuuu | uuuu uuuu | uuuu uuuu |
| STATUS | --00 xxxx | --1u uuuu | --uu uuuu | --01 uuuu | --11 uuuu |
| INTC | -000 0000 | -000 0000 | -000 0000 | -000 0000 | -uuu uuuu |

| Register | Reset(Power On) | WDT Time-out (Normal Operation) | $\overline{RES}$ Reset (Normal Operation) | $\overline{RES}$ Reset (HALT) | WDT Time-out (HALT)* |
|---|---|---|---|---|---|
| PA | 1111 1111 | 1111 1111 | 1111 1111 | 1111 1111 | uuuu uuuu |
| PAC | 1111 1111 | 1111 1111 | 1111 1111 | 1111 1111 | uuuu uuuu |
| PB | ---1 1111 | ---1 1111 | ---1 1111 | ---1 1111 | ---u uuuu |
| PBC | ---1 1111 | ---1 1111 | ---1 1111 | ---1 1111 | ---u uuuu |
| PD | ---- ---1 | ---- ---1 | ---- ---1 | ---- ---1 | ---- ---u |
| PDC | ---- ---1 | ---- ---1 | ---- ---1 | ---- ---1 | ---- ---u |
| PWM | xxxx xxxx | xxxx xxxx | xxxx xxxx | xxxx xxxx | uuuu uuuu |
| ADRL | xxxx ---- | xxxx ---- | xxxx ---- | xxxx ---- | uuuu ---- |
| ADRH | xxxx xxxx | xxxx xxxx | xxxx xxxx | xxxx xxxx | uuuu uuuu |
| ADCR | 0100 0000 | 0100 0000 | 0100 0000 | 0100 0000 | uuuu uuuu |
| ACSR | ---- --00 | ---- --00 | ---- --00 | ---- --00 | ---- --uu |

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

**Timer/Event Counter**

Only one timer/event counter (TMR) are 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 an internal clock source. An internal clock source comes from $f_{SYS}$. The external clock input allows the user to count external events, measure time intervals or pulse widths, or to generate an accurate time base.

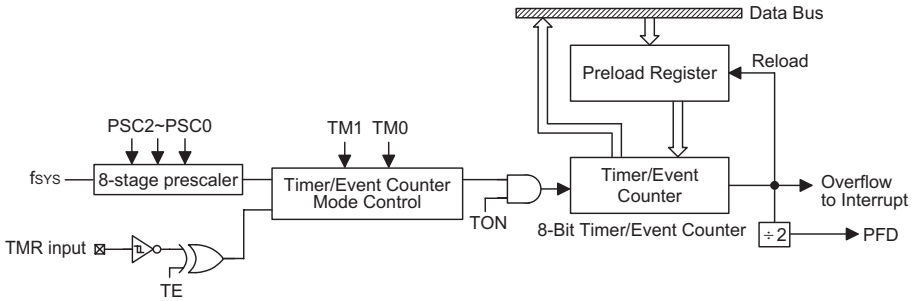
There are two registers related to the Timer/event counter; TMR (0DH), TMRC (0EH). Writing TMR will transfer the specified data to timer/event counter registers. Reading the TMR will read the contents of the timer/event counter. The TMRC is a control register, which defines the operating mode, counting enable or disable and an active edge.

The TM0 and TM1 bits define the operation mode. The event count mode is used to count external events, which means that the clock source is from an external (TMR) pin. The timer mode functions as a normal timer with the clock source coming from the internal selected clock source. Finally, the pulse width measurement mode can be used to count the high or low level duration of the external signal (TMR), and the counting is based on the internal selected clock source.

In the event count or timer mode, the timer/event counter starts counting at the current contents in the timer/event counter and ends at FFH. Once an overflow occurs, the counter is reloaded from the timer/event counter preload register, and generates an interrupt request flag (TF; bit 5 of the INTC ). In the pulse width measurement mode with the values of the TON and TE bits equal to 1, after the TMR has received a transient from low to high (or high to low if the TE bit is "0"), it will start counting until the TMR returns to the original level and resets the TON. The measured result remains in the timer/event counter even if the activated transient occurs again. In other words, only 1-cycle measurement can be made until the TON is set. The cycle measurement will re-operate as long as it receives further transient pulse. In this operation mode, the timer/event counter begins counting not according to the logic level but to the transient edges. In the case of counter overflows, the counter is reloaded from the timer/event counter register and issues an interrupt request, as in the other two modes, i.e., event and timer modes.

To enable the counting operation, the Timer ON bit (TON; bit 4 of the TMRC) should be set to "1". In the pulse width measurement mode, the TON is automatically cleared 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 and can also be applied to a PFD (Programmable Frequency Divider) output at PA3 by options. No matter what the operation mode is, writing a "0" to ETI (bit2 of the INTC) disables the related interrupt service. When the PFD function is selected, executing "SET [PA].3" instruction to enable the PFD output and executing "CLR [PA].3" instruction to disable the PFD output.

In the case of timer/event counter OFF condition, writing data to the timer/event counter preload register also reloads that data to the timer/event counter. But if the timer/event counter is turn on, data written to the




**8-Bit Timer/Event Counter Structure**

timer/event counter is kept only in the timer/event counter preload register. The timer/event counter still continues its operation until an overflow occurs.

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

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

| Bit No. | Label | Function |
|---|---|---|
| 0<br>1<br>2 | PSC0<br>PSC1<br>PSC2 | Defines the prescaler stages, PSC2, PSC1, PSC0=<br>000: $f_{INT}=f_{SYS}$<br>001: $f_{INT}=f_{SYS}/2$<br>010: $f_{INT}=f_{SYS}/4$<br>011: $f_{INT}=f_{SYS}/8$<br>100: $f_{INT}=f_{SYS}/16$<br>101: $f_{INT}=f_{SYS}/32$<br>110: $f_{INT}=f_{SYS}/64$<br>111: $f_{INT}=f_{SYS}/128$ |
| 3 | TE | Defines the TMR active edge of the timer/event counter:<br>In Event Counter Mode (TM1,TM0)=(0,1):<br>1:count on falling edge;<br>0:count on rising edge<br>In Pulse Width measurement mode (TM1,TM0)=(1,1):<br>1: start counting on the rising edge, stop on the falling edge;<br>0: start counting on the falling edge, stop on the rising edge |
| 4 | TON | Enable/disable timer counting (0=disable; 1=enable) |
| 5 | — | Unused bit, read as ″0″ |
| 6<br>7 | TM0<br>TM1 | Defines the operating mode, TM1, TM0:<br>01=Event count mode (external clock)<br>10=Timer mode (internal clock)<br>11=Pulse width measurement mode<br>00=Unused |

**TMRC (0EH) Register**

**Input/Output Ports**

There are 14 bidirectional input/output lines in the microcontroller, labeled as PA, PB and PD, which are mapped to the data memory of [12H], [14H] 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 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, 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 re-configured dynamically under software control. To function as an input, the corresponding latch of the control register must write ″1″. The input source also depends on the control register. If the control register bit is ″1″, the input will read the pad state. If the control register bit is ″0″, the contents of the latches will move to the internal bus. The latter is possible in the ″read-modify-write″ instruction.

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

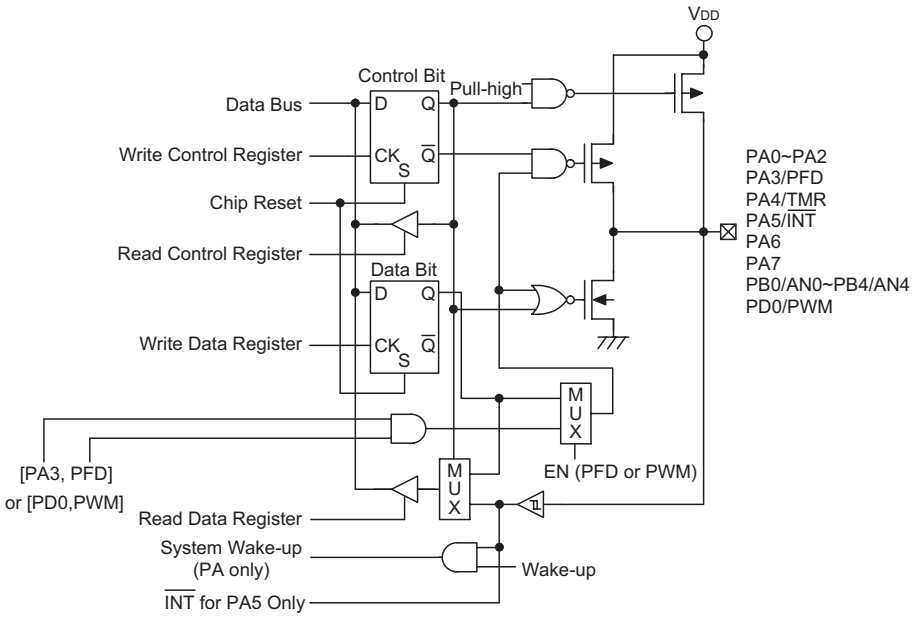
Each line of port A has the capability of waking-up the device. Each I/O port has a pull-high option. Once the pull-high option is selected, the I/O port has a pull-high resistor, otherwise, there′s none. Take note that a non-pull-high I/O port operating in input mode will cause a floating state.

The PA3, PA4 and PA5 are pin-shared with PFD, TMR and $\overline{\text{INT}}$ pins respectively.

If the PFD option is selected, the output signal in output mode of PA3 will be the PFD signal generated by the timer/event counter overflow signal. The input mode always remain in its original functions. Once the PFD option is selected, the PFD output signal is controlled by the PA3 data register only. 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.




**Input/Output Ports**

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

| Timer | Timer Preload Value | PA3 Data Register | PA3 Pad State | Frequency |
|---|---|---|---|---|
| OFF | X | 0 | 0 | X |
| OFF | X | 1 | U | X |
| ON | N | 0 | 0 | X |
| ON | N | 1 | PFD | $f_{INT}/(2\times(256-N))$ |

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

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). The I/O functions of PD0 are 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**

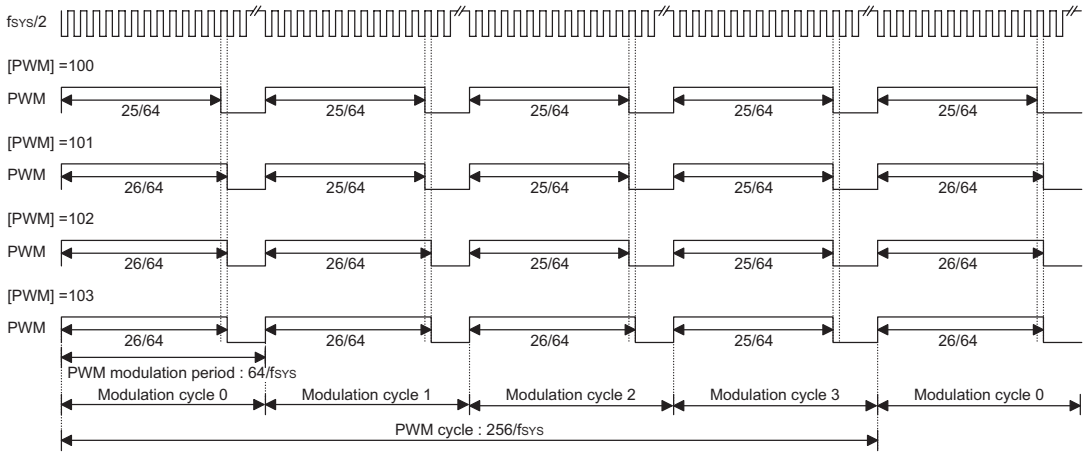
The microcontroller provides one channel PWM output shared with PD0. The PWM supports 6+2 mode. The PWM channel has their data register denoted as PWM(1AH). The frequency source of the PWM counter comes from $f_{SYS}$. The PWM register is an 8-bit register. The waveforms of the PWM outputs are as shown. Once the PD0 are selected as the PWM outputs and the output function of the PD0 are 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 | $\frac{DC}{64}$ |

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 | $f_{SYS}/256$ | [PWM]/256 |




**(6+2) PWM Mode**

**A/D Converter**

The 5 channels 12-bit resolution A/D converter are implemented in this microcontroller.

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

The A/D converter control register is used to control the A/D converter. The bit2~bit0 of the are used to select an analog input channel. There are a total of five 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 determined by these 3 bits. Once a PB line is selected as an analog input, the I/O functions and pull-high resistor of this I/O line are disabled and the A/D converter circuit is powered on. The EOCB bit (bit6 of the ADCR) is end of A/D conversion flag. Check this bit to know when the 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 that the A/D conversion is completed, the START should remain at ″0″ until the EOCB is cleared to ″0″ (end of A/D conversion). The bit 7 of the ACSR is used for testing purposes only. 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 initialisation:
Special care must be taken to initialise 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 initialisation 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 initialisation is not required.

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

**ACSR (23H) Register**

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

**ADCR (22H) Register**

| 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 | * |
| 1 | 1 | 0 | * |
| 1 | 1 | 1 | * |

**Analog Input Channel Selection**

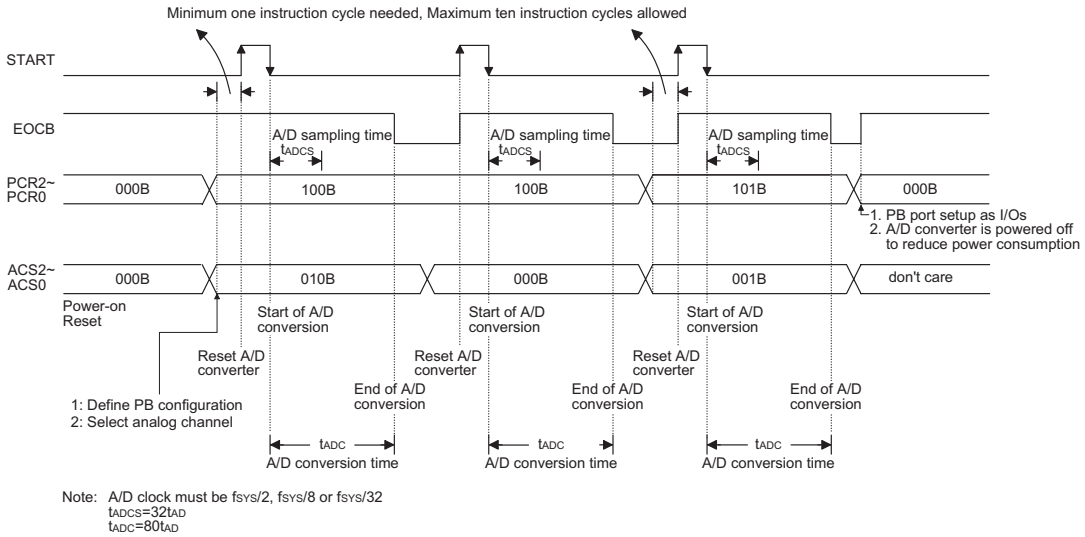
Note: * undefined, cannot be used

| Register | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|
| ADRL (20H) | D3 | D2 | D1 | D0 | 0 | 0 | 0 | 0 |
| ADRH (21H) | D11 | D10 | D9 | D8 | D7 | D6 | D5 | D4 |

Note: D0~D11 is A/D conversion result data bit LSB~MSB.

| PCR2 | PCR1 | PCR0 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | PB4 | PB3 | PB2 | PB1 | PB0 |
| 0 | 0 | 1 | PB4 | PB3 | PB2 | PB1 | AN0 |
| 0 | 1 | 0 | PB4 | PB3 | PB2 | AN1 | AN0 |
| 0 | 1 | 1 | PB4 | PB3 | AN2 | AN1 | AN0 |
| 1 | 0 | 0 | PB4 | AN3 | AN2 | AN1 | AN0 |
| 1 | 0 | 1 | AN4 | AN3 | AN2 | AN1 | AN0 |
| 1 | 1 | 0 | Undefined, cannot be used | | | | |
| 1 | 1 | 1 | | | | | |

**Port B Configuration**




**A/D Conversion Timing**

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

Example: using EOCB Polling Method to detect end of conversion

```
    clr     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
```

**Low Voltage Reset – LVR**

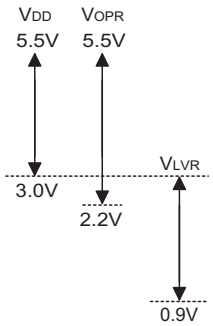
There is a low voltage reset circuit (LVR) implemented in the microcontrollers. The function can be enabled/disabled by options.

If the supply voltage of the device is within the range 0.9V~$V_{LVR}$ such as changing a battery, the LVR will automatically reset the device internally.

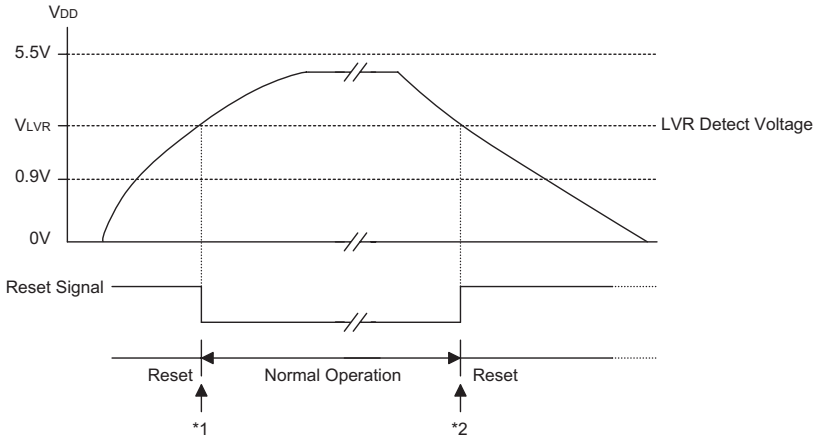
The LVR includes the following specifications:

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

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




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




**Low Voltage Reset**

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

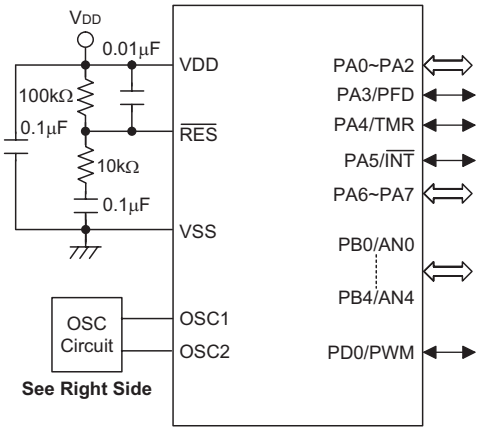
*2: Since low voltage state has to be maintained in its original state for over 1ms, therefore after 1ms delay, the device enters the reset mode.

**Options**

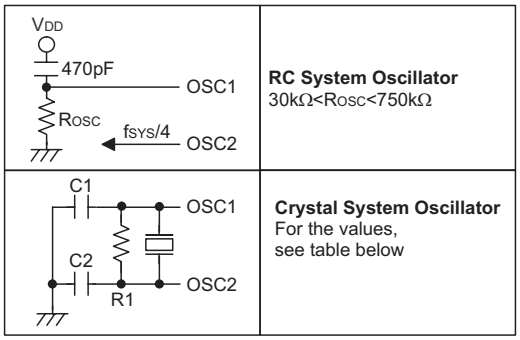
The following shows kinds of options in the device. ALL the options must be defined to ensure having a proper functioning system.

| Options |
|---|
| OSC type selection.<br>This option is to decide if an RC or crystal oscillator is chosen as system clock. |
| WDT source selection.<br>There are three types of selection: on-chip RC oscillator, instruction clock or disable the WDT. |
| CLRWDT times selection.<br>This option defines how to clear the WDT by instruction. ″One time″ means that the ″CLR WDT″ instruction can clear the WDT. ″Two times″ means only if both of the ″CLR WDT1″ and ″CLR WDT2″ instructions have been executed, then WDT can be cleared. |
| Wake-up selection.<br>This option defines the wake-up function activity. External I/O pins (PA only) all have the capability to wake-up the chip from a HALT by a falling edge. (Bit option) |
| Pull-high selection.<br>This option is to decide whether a pull-high resistance is visible or not in the input mode of the I/O ports. PA, PB and PD are bit option. |
| PFD selection.<br>PA3: Level output or PFD output. |
| PWM selection.<br>PD0: level output or PWM output |
| LVR selection.<br>Enable or disable LVR function. |

## Application Circuits




HT46R51A/HT46R52A




OSC Circuit

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

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

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

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

## Instruction Set Summary

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

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

Note: x: Immediate data

m: Data memory address

A: Accumulator

i: 0~7 number of bits

addr: Program memory address

√: Flag is affected

–: Flag is not affected

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

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

(3): (1) and (2)

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

## Instruction Definition

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

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

Operation ACC ← ACC+[m]+C

Affected flag(s)

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | √ | √ | √ | √ |

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

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

Operation [m] ← ACC+[m]+C

Affected flag(s)

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | √ | √ | √ | √ |

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

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

Operation ACC ← ACC+[m]

Affected flag(s)

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | √ | √ | √ | √ |

**ADD A,x** Add immediate data to the accumulator

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

Operation ACC ← ACC+x

Affected flag(s)

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | √ | √ | √ | √ |

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

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

Operation [m] ← ACC+[m]

Affected flag(s)

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | √ | √ | √ | √ |

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

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

Operation ACC ← ACC ″AND″ [m]

Affected flag(s)

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | — | √ | — | — |

**AND A,x** Logical AND immediate data to the accumulator

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

Operation ACC ← ACC ″AND″ x

Affected flag(s)

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | — | √ | — | — |

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

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

Operation [m] ← ACC ″AND″ [m]

Affected flag(s)

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | — | √ | — | — |

**CALL addr** Subroutine call

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

Operation Stack ← Program Counter+1
Program Counter ← addr

Affected flag(s)

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | — | — | — | — |

**CLR [m]** Clear data memory

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

Operation [m] ← 00H

Affected flag(s)

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | — | — | — | — |

**CLR [m].i** Clear bit of data memory

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

Operation [m].i ← 0

Affected flag(s)

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | — | — | — | — |

**CLR WDT** Clear Watchdog Timer

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

Operation WDT ← 00H
PDF and TO ← 0

Affected flag(s)

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| 0 | 0 | — | — | — | — |

**CLR WDT1** Preclear Watchdog Timer

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

Operation WDT ← 00H*
PDF and TO ← 0*

Affected flag(s)

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| 0* | 0* | — | — | — | — |

**CLR WDT2** Preclear Watchdog Timer

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

Operation WDT ← 00H*
PDF and TO ← 0*

Affected flag(s)

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| 0* | 0* | — | — | — | — |

**CPL [m]** Complement data memory

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

Operation [m] ← $[\overline{m}]$

Affected flag(s)

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | — | √ | — | — |

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

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

Operation ACC ← $[\overline{m}]$

Affected flag(s)

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | — | √ | — | — |

**DAA [m]** Decimal-Adjust accumulator for addition

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

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

Affected flag(s)

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | — | — | — | √ |

**DEC [m]** Decrement data memory

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

Operation [m] ← [m]–1

Affected flag(s)

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | — | √ | — | — |

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

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

Operation ACC ← [m]–1

Affected flag(s)

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | — | √ | — | — |

**HALT** Enter power down mode

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

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

Affected flag(s)

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| 0 | 1 | — | — | — | — |

**INC [m]** Increment data memory

Description Data in the specified data memory is incremented by 1

Operation [m] ← [m]+1

Affected flag(s)

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | — | √ | — | — |

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

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

Operation ACC ← [m]+1

Affected flag(s)

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | — | √ | — | — |

**JMP addr** Directly jump

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

Operation Program Counter ←addr

Affected flag(s)

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | — | — | — | — |

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

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

Operation ACC ← [m]

Affected flag(s)

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | — | — | — | — |

**MOV A,x** Move immediate data to the accumulator

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

Operation ACC ← x

Affected flag(s)

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | — | — | — | — |

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

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

Operation [m] ←ACC

Affected flag(s)

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | — | — | — | — |

**NOP** No operation

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

Operation Program Counter ← Program Counter+1

Affected flag(s)

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | — | — | — | — |

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

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

Operation ACC ← ACC ″OR″ [m]

Affected flag(s)

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | — | √ | — | — |

**OR A,x** Logical OR immediate data to the accumulator

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

Operation ACC ← ACC ″OR″ x

Affected flag(s)

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | — | √ | — | — |

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

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

Operation [m] ←ACC ″OR″ [m]

Affected flag(s)

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | — | √ | — | — |

**RET** Return from subroutine

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

Operation Program Counter ← Stack

Affected flag(s)

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | — | — | — | — |

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

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

Operation Program Counter ← Stack
ACC ← x

Affected flag(s)

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | — | — | — | — |

**RETI** Return from interrupt

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

Operation Program Counter ← Stack
EMI ← 1

Affected flag(s)

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | — | — | — | — |

**RL [m]** Rotate data memory left

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

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

Affected flag(s)

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | — | — | — | — |

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

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

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

Affected flag(s)

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | — | — | — | — |

**RLC [m]** Rotate data memory left through carry

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

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

Affected flag(s)

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | — | — | — | √ |

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

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

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

Affected flag(s)

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | — | — | — | √ |

**RR [m]** Rotate data memory right

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

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

Affected flag(s)

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | — | — | — | — |

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

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

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

Affected flag(s)

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | — | — | — | — |

**RRC [m]** Rotate data memory right through carry

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

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

Affected flag(s)

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | — | — | — | √ |

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

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

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

Affected flag(s)

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | — | — | — | √ |

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

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

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

Affected flag(s)

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | √ | √ | √ | √ |

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

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

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

Affected flag(s)

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | √ | √ | √ | √ |

**SDZ [m]** Skip if decrement data memory is 0

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

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

Affected flag(s)

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | — | — | — | — |

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

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

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

Affected flag(s)

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | — | — | — | — |

**SET [m]** Set data memory

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

Operation [m] ← FFH

Affected flag(s)

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | — | — | — | — |

**SET [m]. i** Set bit of data memory

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

Operation [m].i ← 1

Affected flag(s)

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | — | — | — | — |

**SIZ [m]** Skip if increment data memory is 0

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

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

Affected flag(s)

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | — | — | — | — |

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

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

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

Affected flag(s)

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | — | — | — | — |

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

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

Operation Skip if [m].i≠0

Affected flag(s)

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | — | — | — | — |

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

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

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

Affected flag(s)

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | √ | √ | √ | √ |

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

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

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

Affected flag(s)

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | √ | √ | √ | √ |

**SUB A,x** Subtract immediate data from the accumulator

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

Operation ACC ← ACC+$\overline{x}$+1

Affected flag(s)

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | √ | √ | √ | √ |

**SWAP [m]** Swap nibbles within the data memory

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

Operation [m].3~[m].0 ↔ [m].7~[m].4

Affected flag(s)

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | — | — | — | — |

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

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

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

Affected flag(s)

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | — | — | — | — |

**SZ [m]** Skip if data memory is 0

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

Operation Skip if [m]=0

Affected flag(s)

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | — | — | — | — |

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

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

Operation Skip if [m]=0

Affected flag(s)

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | — | — | — | — |

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

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

Operation Skip if [m].i=0

Affected flag(s)

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | — | — | — | — |

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

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

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

Affected flag(s)

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | — | — | — | — |

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

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

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

Affected flag(s)

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | — | — | — | — |

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

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

Operation ACC ← ACC ″XOR″ [m]

Affected flag(s)

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | — | √ | — | — |

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

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

Operation [m] ← ACC ″XOR″ [m]

Affected flag(s)

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | — | √ | — | — |

**XOR A,x** Logical XOR immediate data to the accumulator

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

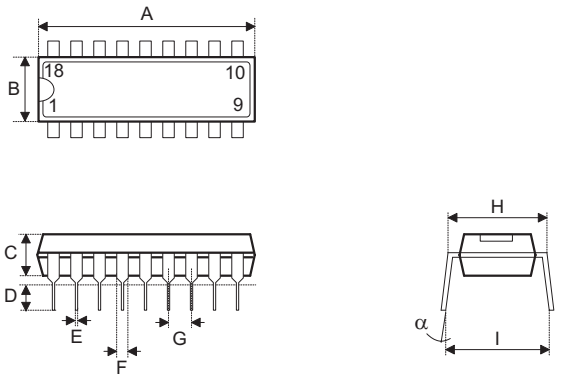
Operation ACC ← ACC ″XOR″ x

Affected flag(s)

| TO | PDF | OV | Z | AC | C |
|---|---|---|---|---|---|
| — | — | — | √ | — | — |

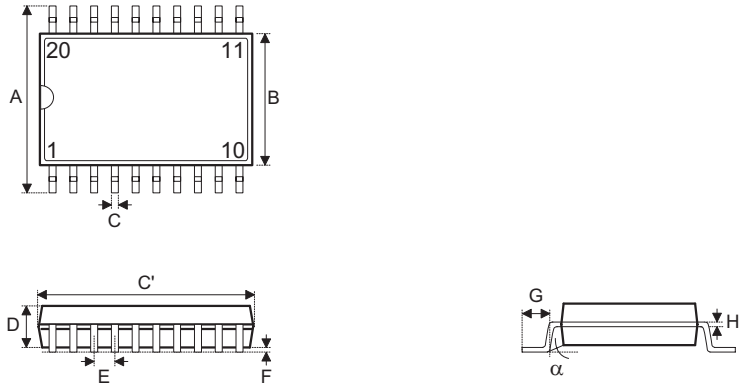
## Package Information

**18-pin DIP (300mil) Outline Dimensions**



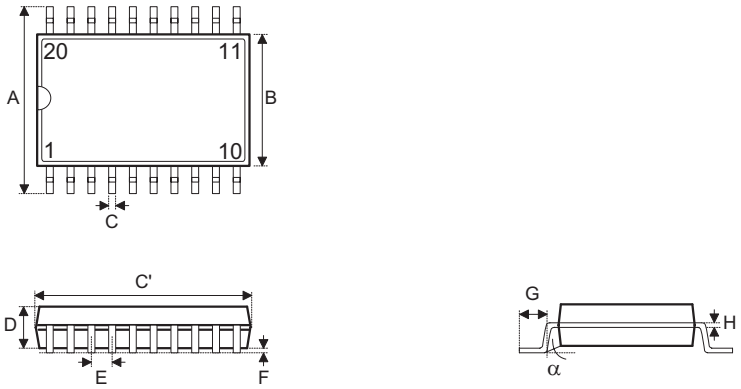

| Symbol | Dimensions in mil | | |
|---|---|---|---|
| | Min. | Nom. | Max. |
| A | 895 | — | 915 |
| B | 240 | — | 260 |
| C | 125 | — | 135 |
| D | 125 | — | 145 |
| E | 16 | — | 20 |
| F | 50 | — | 70 |
| G | — | 100 | — |
| H | 295 | — | 315 |
| I | 335 | — | 375 |
| α | 0° | — | 15° |

**20-pin SOP (300mil) Outline Dimensions**




| Symbol | Dimensions in mil | | |
|---|---|---|---|
| | Min. | Nom. | Max. |
| A | 394 | — | 419 |
| B | 290 | — | 300 |
| C | 14 | — | 20 |
| C′ | 490 | — | 510 |
| D | 92 | — | 104 |
| E | — | 50 | — |
| F | 4 | — | — |
| G | 32 | — | 38 |
| H | 4 | — | 12 |
| α | 0° | — | 10° |

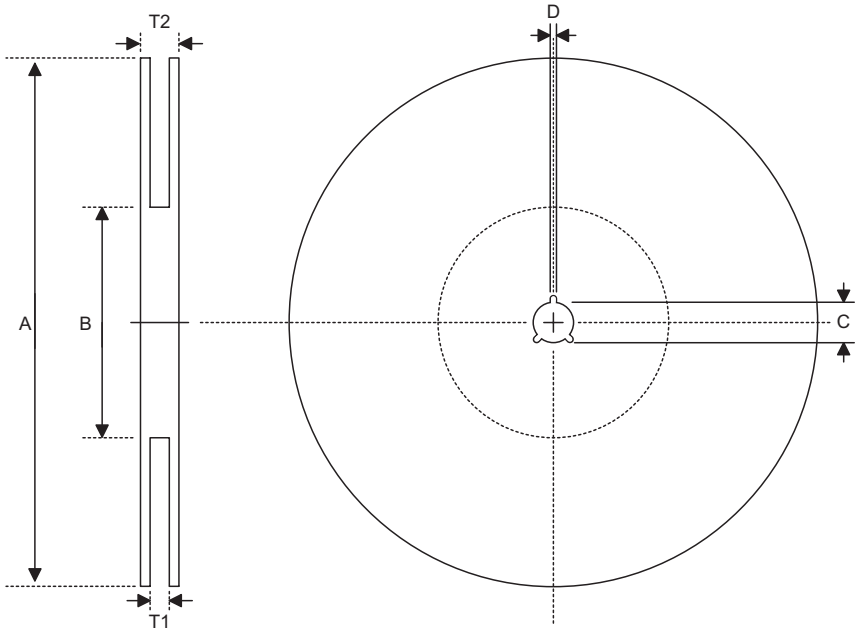
**20-pin SSOP (150mil) Outline Dimensions**




| Symbol | Dimensions in mil | | |
|---|---|---|---|
| | Min. | Nom. | Max. |
| A | 228 | — | 244 |
| B | 150 | — | 158 |
| C | 8 | — | 12 |
| C′ | 335 | — | 347 |
| D | 49 | — | 65 |
| E | — | 25 | — |
| F | 4 | — | 10 |
| G | 15 | — | 50 |
| H | 7 | — | 10 |
| α | 0° | — | 8° |

## Product Tape and Reel Specifications

**Reel Dimensions**



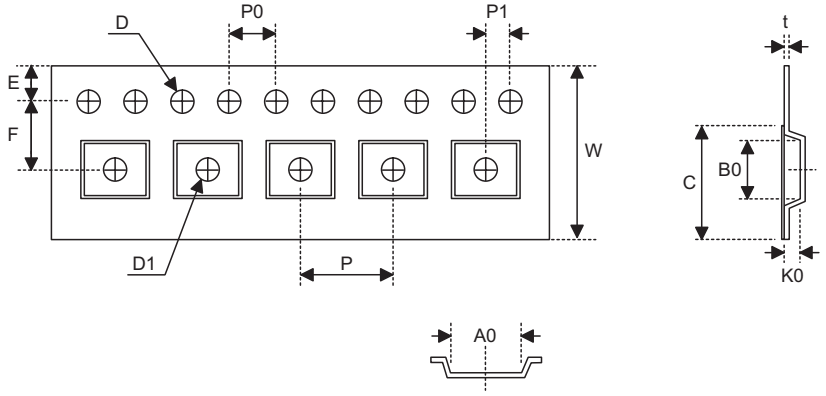

SOP 20W

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

SSOP 20S (150mil)

| Symbol | Description | Dimensions in mm |
|---|---|---|
| A | Reel Outer Diameter | 330±1 |
| B | Reel Inner Diameter | 62±1.5 |
| C | Spindle Hole Diameter | 13+0.5<br>−0.2 |
| D | Key Slit Width | 2±0.5 |
| T1 | Space Between Flange | 16.8+0.3<br>−0.2 |
| T2 | Reel Thickness | 22.2±0.2 |

**Carrier Tape Dimensions**




SOP 20W

| Symbol | Description | Dimensions in mm |
|---|---|---|
| W | Carrier Tape Width | 24+0.3<br>−0.1 |
| P | Cavity Pitch | 12±0.1 |
| E | Perforation Position | 1.75±0.1 |
| F | Cavity to Perforation (Width Direction) | 11.5±0.1 |
| D | Perforation Diameter | 1.5+0.1 |
| D1 | Cavity Hole Diameter | 1.5+0.25 |
| P0 | Perforation Pitch | 4±0.1 |
| P1 | Cavity to Perforation (Length Direction) | 2±0.1 |
| A0 | Cavity Length | 10.8±0.1 |
| B0 | Cavity Width | 13.3±0.1 |
| K0 | Cavity Depth | 3.2±0.1 |
| t | Carrier Tape Thickness | 0.3±0.05 |
| C | Cover Tape Width | 21.3 |

SSOP 20S (150mil)

| Symbol | Description | Dimensions in mm |
|---|---|---|
| W | Carrier Tape Width | 16+0.3<br>−0.1 |
| P | Cavity Pitch | 8±0.1 |
| E | Perforation Position | 1.75±0.1 |
| F | Cavity to Perforation (Width Direction) | 7.5±0.1 |
| D | Perforation Diameter | 1.5+0.1 |
| D1 | Cavity Hole Diameter | 1.5+0.25 |
| P0 | Perforation Pitch | 4±0.1 |
| P1 | Cavity to Perforation (Length Direction) | 2±0.1 |
| A0 | Cavity Length | 6.5±0.1 |
| B0 | Cavity Width | 9±0.1 |
| K0 | Cavity Depth | 2.3±0.1 |
| t | Carrier Tape Thickness | 0.3±0.05 |
| C | Cover Tape Width | 13.3 |

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

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

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

**Holtek Semiconductor Inc. (Shenzhen Sales Office)**
5/F, Unit A, Productivity Building, Cross of Science M 3rd Road and Gaoxin M 2nd Road, Science Park, Nanshan District, Shenzhen, China 518057
Tel: 0755-8616-9908, 8616-9308
Fax: 0755-8616-9533

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

**Holtek Semiconductor Inc. (Chengdu Sales Office)**
709, Building 3, Champagne Plaza, No.97 Dongda Street, Chengdu, Sichuan, China 610016
Tel: 028-6653-6590
Fax: 028-6653-6591

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

Copyright © 2006 by HOLTEK SEMICONDUCTOR INC.

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