



HT46R321

A/D Type 8-Bit OTP MCU with OPA and 8×4 LED Driver

Technical Document

- [Tools Information](#)
- [FAQs](#)
- [Application Note](#)
 - [HA0003E Communicating between the HT48 & HT46 Series MCUs and the HT93LC46 EEPROM](#)
 - [HA0049E Read and Write Control of the HT1380](#)
 - [HA0051E Li Battery Charger Demo Board - Using the HT46R47](#)
 - [HA0052E Microcontroller Application - Battery Charger](#)
 - [HA0075E MCU Reset and Oscillator Circuits Application Note](#)
 - [HA0083E Li Battery Charger Demo Board - Using the HT46R46](#)

Features

- Operating voltage:
f_{sys}=4MHz: 2.2V~5.5V
f_{sys}=8MHz: 3.3V~5.5V
- 21 bidirectional I/O lines (max.)
- support 8×4 LED driver
- Single interrupt input shared with an I/O line
- 8-bit programmable timer/event counter with overflow interrupt
- Integrated crystal and RC oscillator
- Watchdog Timer
- 2048×15 Program Memory capacity
- 88×8 Data Memory capacity
- Integrated PFD function for sound generation
- Power-down and wake-up functions reduce power consumption
- Up to 0.5μs instruction cycle with 8MHz system clock at V_{DD}=5V
- 6-level subroutine nesting
- 6 channel 12-bit resolution A/D converter
- Integrated single operational amplifier or comparator selectable via configuration option
- Peripheral clock output – PCK
- Dual 8-bit PWM outputs shared with I/O lines
- Bit manipulation instruction
- Full table read instruction
- 63 powerful instructions
- All instructions executed in one or two machine cycles
- Low voltage reset function
- 28-pin SKDIP/SOP/SSOP packages

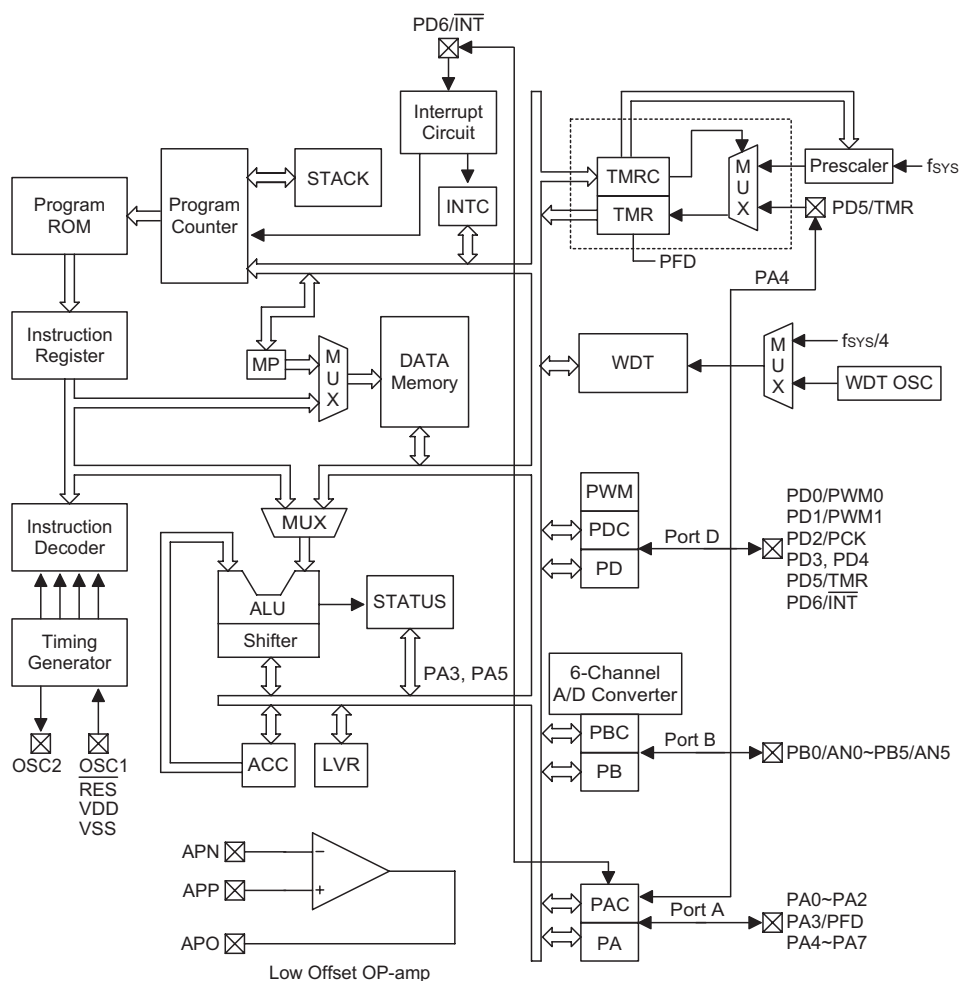
General Description

The HT46R321 is 8-bit, high performance, RISC architecture microcontroller devices. With their fully integrated A/D converter they are especially suitable for applications which interface to analog signals, such as those from sensors. The addition of an internal operational amplifier/comparator and PWM modulation functions further adds to the analog capability of these devices.

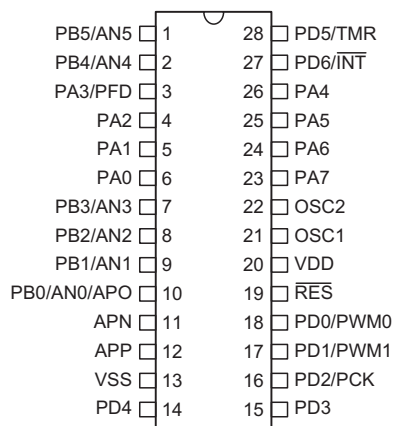
With the comprehensive features of low power consumption, I/O flexibility, programmable frequency divider, timer functions, oscillator options, multi-channel A/D Converter, OP/Comparator, Pulse Width Modulation function, LED driver, Power-down and wake-up functions etc, the application scope of these devices is broad and encompasses areas such as sensor signal processing, motor driving, industrial control, consumer products, subsystem controllers, etc.



Block Diagram



Pin Assignment



HT46R321
— 28 SKDIP-A/SOP-A/SSOP-A

Pin Description

| Pin Name | I/O | Options | Description |
|---|--------|---|---|
| PA0~PA2 PA3/PFD PA4~PA7 | I/O | Pull-high Wake-up PA3 or PFD | Bidirectional 8-bit input/output port. Each pin can be configured as wake-up input by configuration options. Software instructions determine if the pin is a CMOS output or Schmitt Trigger input. Configuration options determine which pins on the port have pull-high resistors. The PFD pin is pin-shared with PA3. PA0~PA7 can be used as LED driver (source end). |
| PB0/AN0/APO, PB1/AN1~ PB5/AN5 | I/O | Pull-high | 6 lines AD input pin-shared with PB. Bidirectional 6-bit input/output port. Software instructions determine if the pin is a CMOS output or Schmitt Trigger input. Configuration options determine which pins on the port have pull-high resistors. Pins PB are 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 are disabled automatically. About PB0/AN0/APO: If the pin is PB0 (setting by ADCS.PCR3~0=00), ADC and OP amp should be power off automatically. If the pin is AN0 (setting by ADCS.PCR3~0), APO is connected with AN0 pin together and OP amp on/off is controlled by OPAC.OPAEN. |
| PD0/PWM0 PD1/PWM1 PD2/PCK PD3~PD4, PD5/TMR PD6/INT | I/O | Pull-high PD0 or PWM0 PD1 or PWM1 PD2 or PCK | Bi-directional 4-bit input/output port. 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. PD0/PD1 are pin-shared with the PWM0/PWM1 outputs selected via configuration option. PD0, PD1, PD2, PD5, PD6 are pin-shared with the PWM0, PWM1, PCK, TMR, INT output selected via configuration option. PD1~PD4 can be used as LED driver (sink end). |
| APN APP | I I | — | APN and APP are the internal operational amplifier, negative input pin and positive input pin respectively . |
| RES | I | — | Schmitt trigger reset input. Active low. |
| VDD | — | — | Positive power supply |
| VSS | — | — | Negative power supply, ground. |
| OSC1 OSC2 | I O | Crystal or RC | OSC1, OSC2 are connected to an external RC network or external crystal, determined by configuration option, for the internal system clock. If the RC system clock option is selected, pin OSC2 can be used to measure the system clock at 1/4 frequency. |

Absolute Maximum Ratings

| | | | |
|-------------------------------|--------------------------------|-----------------------------|----------------------------------|
| Supply Voltage | $V_{SS}-0.3V$ to $V_{SS}+6.0V$ | Storage Temperature | $-50^{\circ}C$ to $125^{\circ}C$ |
| Input Voltage | $V_{SS}-0.3V$ to $V_{DD}+0.3V$ | Operating Temperature | $-40^{\circ}C$ to $85^{\circ}C$ |
| I_{OL} Total | 150mA | I_{OH} Total | $-100mA$ |
| Total Power Dissipation | 500mW | | |

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

Operating Temperature: $40^{\circ}C \sim +85^{\circ}C$, $T_a = 25^{\circ}C$

| 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 disable | — | 0.6 | 1.5 | mA |
| | | 5V | | — | 2.0 | 4.0 | mA |
| I_{DD2} | Operating Current (RC OSC) | 3V | No load, $f_{SYS}=4MHz$ ADC disable | — | 0.8 | 1.5 | mA |
| | | 5V | | — | 2.5 | 4.0 | mA |
| I_{DD3} | Operating Current (Crystal OSC, RC OSC) | 5V | No load, $f_{SYS}=8MHz$ ADC disable | — | 3 | 5 | mA |
| I_{STB1} | Standby Current (WDT Enabled) | 3V | No load, system HALT | — | — | 5 | μA |
| | | 5V | | — | — | 10 | μA |
| I_{STB2} | Standby Current (WDT Disabled) | 3V | No load, system HALT | — | — | 1 | μA |
| | | 5V | | — | — | 2 | μA |
| V_{IL1} | Input Low Voltage for I/O Ports, TMR and INT | — | — | 0 | — | $0.3V_{DD}$ | V |
| V_{IH1} | Input High Voltage for I/O Ports, TMR and 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 | — | Temperature= $25^{\circ}C$ | 2.7 | 3.0 | 3.3 | V |
| I_{OL1} | I/O Port Sink Current (PA, PB, PD0, PD5, PD6) | 3V | $V_{OL}=0.1V_{DD}$ | 4 | 8 | — | mA |
| | | 5V | $V_{OL}=0.1V_{DD}$ | 10 | 20 | — | mA |
| I_{OH1} | I/O Port Source Current (PB, PD) | 3V | $V_{OH}=0.9V_{DD}$ | -2 | -4 | — | mA |
| | | 5V | $V_{OH}=0.9V_{DD}$ | -5 | -10 | — | mA |
| I_{OL2} | PD1~PD4 Ports Sink Current for LED Driver | 3V | $V_{OL}=0.1V_{DD}$ | 8 | 16 | — | mA |
| | | 5V | $V_{OL}=0.1V_{DD}$ | 20 | 40 | — | mA |
| I_{OH2} | PA Ports Source Current for LED Driver | 3V | $V_{OH}=0.9V_{DD}$ | -4 | -8 | — | mA |
| | | 5V | $V_{OH}=0.9V_{DD}$ | -10 | -20 | — | mA |
| R_{PH} | Pull-high Resistance | 3V | — | 20 | 60 | 100 | $k\Omega$ |
| | | 5V | — | 10 | 30 | 50 | $k\Omega$ |
| V_{AD} | A/D Input Voltage | — | — | 0 | — | V_{DD} | V |

| Symbol | Parameter | Test Conditions | | Min. | Typ. | Max. | Unit |
|------------------|---|-----------------|----------------------|------|------|------|------|
| | | V _{DD} | Conditions | | | | |
| I _{ADC} | Additional Power Consumption if A/D Converter is Used | 3V | — | — | 0.5 | 1 | mA |
| | | 5V | | — | 1.5 | 3 | mA |
| DNL | ADC Differential Non-Linearity | 5V | t _{AD} =1μs | — | — | ±2 | mA |
| INL | ADC Integral Non-Linearity | 5V | t _{AD} =1μs | — | ±2.5 | 4 | mA |
| RESOLU | Resolution | — | — | — | — | 12 | Bits |

A.C. Characteristics

Ta=25°C

| Symbol | Parameter | Test Conditions | | Min. | Typ. | Max. | Unit |
|---------------------|---|-----------------|-------------------|-----------------|------|-----------------|---------------------|
| | | V _{DD} | Conditions | | | | |
| f _{SYS} | System Clock (Crystal OSC, RC OSC) | — | 2.2V~5.5V | 400 | — | 4000 | kHz |
| | | — | 3.3V~5.5V | 400 | — | 8000 | kHz |
| f _{TIMER} | Timer I/P Frequency (TMR) | — | 2.2V~5.5V | 0 | — | 4000 | kHz |
| | | — | 3.3V~5.5V | 0 | — | 8000 | kHz |
| t _{WDTOSC} | Watchdog Oscillator Period | 3V | — | 45 | 90 | 180 | μs |
| | | 5V | — | 32 | 65 | 130 | μs |
| t _{WDT1} | Watchdog Time-out Period (WDT OSC) | — | — | 2 ¹⁵ | — | 2 ¹⁶ | t _{WDTOSC} |
| t _{WDT2} | Watchdog Time-out Period (System Clock) | — | — | 2 ¹⁷ | — | 2 ¹⁸ | t _{SYS} |
| t _{RES} | External Reset Low Pulse Width | — | — | 1 | — | — | μs |
| t _{SST} | System Start-up Timer Period | — | Wake-up from HALT | — | 1024 | — | *t _{SYS} |
| t _{LVR} | Low Voltage Width to Reset | — | — | 0.25 | 1 | 2 | ms |
| t _{INT} | Interrupt Pulse Width | — | — | 1 | — | — | μs |
| t _{AD} | A/D Clock Period | — | — | 1 | — | — | μs |
| t _{ADC} | A/D Conversion Time | — | — | — | 80 | — | t _{AD} |
| t _{ADCS} | A/D Sampling Time | — | — | — | 32 | — | t _{AD} |

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

OP Amplifier Electrical Characteristics

Ta=25°C

| Symbol | Parameter | Test Conditions | | Min. | Typ. | Max. | Unit |
|--------------------------------|------------------------------|-----------------|--|-----------------|------|-----------------------|------|
| | | V _{DD} | Conditions | | | | |
| D.C. Electrical Characteristic | | | | | | | |
| V _{DD} | Operating Voltage | — | — | 3 | — | 5.5 | V |
| V _{OPOS1} | Input Offset Voltage | 5V | — | −10 | — | 10 | mV |
| V _{OPOS2} | Input Offset Voltage | 5V | By Calibration | −2 | — | 2 | mV |
| V _{CM} | Common Mode Voltage Range | — | — | V _{SS} | — | V _{DD} -1.4V | V |
| PSRR | Power Supply Rejection Ratio | — | — | 60 | 80 | — | dB |
| CMRR | Common Mode Rejection Ratio | 5V | V _{CM} =0~V _{DD} -1.4V | 60 | 80 | — | dB |
| t _{RES} | Response Time (Comparator) | — | Input overdrive=±10mV | — | — | 2 | μs |
| A.C. Electrical Characteristic | | | | | | | |
| A _{OL} | Open Loop Gain | — | — | 60 | 80 | — | dB |
| SR | Slew Rate +, Slew Rate - | — | No load | — | 0.1 | — | V/μs |
| GBW | Gain Band Width | — | R _L =1M, C _L =100p | — | — | 100 | kHz |

Functional Description

Execution Flow

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

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

Program Counter – PC

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

After accessing a program memory word to fetch an instruction code, the contents of the program counter are

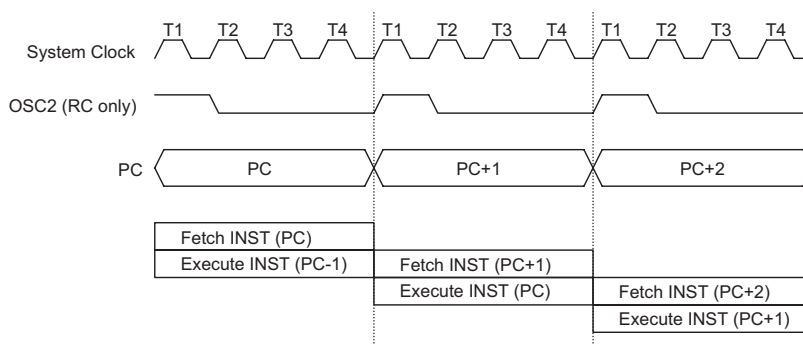
incremented by one. The program counter then points to the memory word containing the next instruction code.

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

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

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

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



Execution Flow

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

Program Counter

Note: PC10~PC8: Current Program Counter bits
#10~#0: Instruction Code bits

S10~S0: Stack register bits
@7~@0: PCL bits

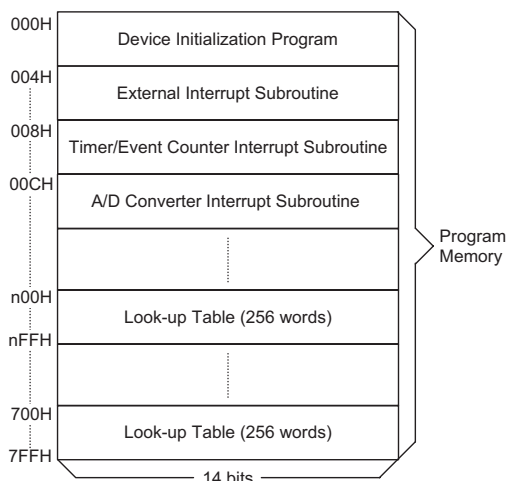
Program Memory – ROM

The program memory is used to store the program instructions which are to be executed as well as table data and interrupt entries. It is structured into 4K×15 bits device, which can be addressed by both the program counter and table pointer.

Certain locations in the program memory are reserved for use by the reset and by the interrupt vectors.

- **Location 000H**
This vector is reserved for program initialisation. After a device reset is initiated, the program will jump to this location and begin execution.
- **Location 004H**
This vector is used by the external interrupt $\overline{\text{INT}}$. If the external interrupt pin on the device receives a low going edge, the program will jump to this location and begin execution if the external interrupt is enabled and the stack is not full.
- **Location 008H**
This vector is used by the Timer/Event Counter. If a timer overflow occurs, the program will jump to this location and begin execution if the timer interrupt is enabled and the stack is not full.
- **Location 00CH**
This vector is used by the A/D converter. When an A/D cycle conversion is complete, the program will jump to this location and begin execution if the A/D interrupt is enabled and the stack is not full.
- **Table location**
Any location in the Program Memory space can be used as a look-up table. The instructions "TABRDC [m]" (the current page, 1 page=256 words) and "TABRDL [m]" (the last page) transfer the contents of the lower-order byte to the specified data memory, and the higher-order byte to TBLH. Only the destination of the lower-order byte in the table is well-defined, the other bits of the table word are transferred to the lower portion of TBLH, and the remaining bits are read as "0". The Table Higher-order byte register (TBLH) is read only. The table pointer (TBLP) is a read/write register, which indicates the table location. Before accessing the table, the location must be placed in TBLP. The TBLH register is read only and cannot be restored. If the main routine and the ISR, Interrupt Service Routine, both employ the table read instruction, the contents of the TBLH in the main routine are likely to be changed by the table read instruction used in the ISR.

In such a case errors can occur. Therefore, 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 used in both the main routine and the ISR, the interrupt is should be disabled prior to the table read instruction. It should not be re-enabled until the TBLH has been backed up. All table related instructions require two cycles to complete their operation. These areas may function as normal program memory depending upon requirements.



Program Memory

Stack Register – STACK

This is a special part of the memory which is used to save the contents of the program counter only. The stack is organized into 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, known as stack pointer, and is also neither readable nor writeable. At a subroutine call or interrupt acknowledgment, the contents of the program counter are pushed onto the stack. At the end of a subroutine or an interrupt routine, signaled by a return instruction, RET or RETI, the program counter is restored to its previous value from the stack. After a device reset, the stack pointer will point to the top of the stack.

If the stack is full and a non-masked interrupt takes place, the interrupt request flag will be recorded but the acknowledgment will be inhibited. When the stack

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

Table Location

Note: *10~*0: Table location bits
@7~@0: Table pointer bits

P10~P8: Current program counter bits

pointer is decremented, using a RET or RETI instruction, the interrupt will be serviced. This feature prevents a stack overflow allowing the programmer to use the structure more easily. In a similar case, if the stack is full and a "CALL" is subsequently executed, stack overflow occurs and the first entry will be lost. Only the most recent 6 return addresses are stored.

Data Memory – RAM

The data memory is divided into two functional groups: special function registers and general purpose data memory. The general purpose memory has a structure of 88x8 bits. Most locations are read/write, but some are read only.

The remaining space between the end of the Special Purpose Data Memory and the beginning of the General Purpose Data Memory is reserved for future expanded usage, reading these locations will obtain a result of "00H". The general purpose data memory, addressed from 28H to 7FH, is used for user 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 the "SET [m].i" and "CLR [m].i" instructions. They are also indirectly accessible through memory pointer register, MP.

Indirect Addressing Register

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

The memory pointer register, MP, is a 7-bit register. The bit 7 of MP is undefined and read as "1".

Accumulator

The accumulator is closely related to ALU operations and can carry out immediate data operations. Any data movement between two data memory locations must pass through the accumulator.

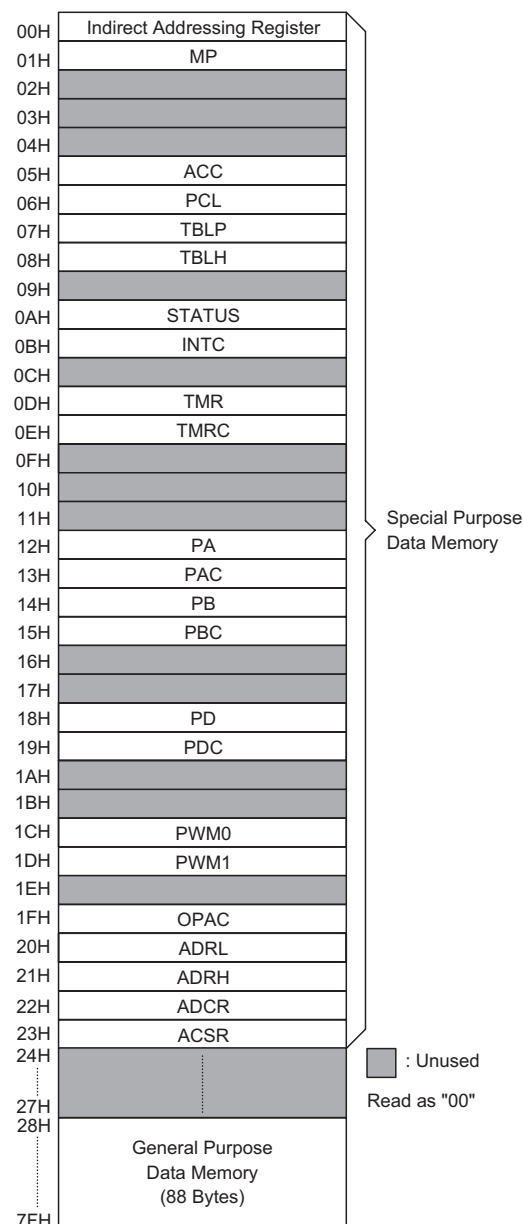
Arithmetic and Logic Unit – ALU

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

- Arithmetic operations - ADD, ADC, SUB, SBC, DAA
- Logic operations - AND, OR, XOR, CPL

- Rotation - RL, RR, RLC, RRC
- Increment and Decrement - INC, DEC
- Branch decision - SZ, SNZ, SIZ, SDZ

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



RAM Mapping

Status Register – STATUS

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

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

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

In addition, on entering the interrupt sequence or executing the subroutine call, the status register will not be

pushed onto the stack automatically. If the contents of the status are important and if the subroutine can corrupt the status register, precautions must be taken to save it properly.

Interrupt

The devices provide an external interrupt, an internal timer/event counter interrupt and an A/D converter interrupt. The Interrupt Control Register, INTC, contains the interrupt control bits to set the enable or disable and the interrupt request flags.

Once an interrupt subroutine is serviced, all the other interrupts will be blocked by clearing the EMI bit. This scheme may prevent any further interrupt nesting. Other interrupt requests may happen during this interval but only the interrupt request flag is recorded. If a certain interrupt requires servicing within the service routine, the EMI bit and the corresponding bit in 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 stack pointer is decremented. If immediate service is desired, the stack must be prevented from becoming full.

| 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 a system power-up or executing the "CLR WDT" instruction. PDF is set by executing the "HALT" instruction. |
| 5 | TO | TO is cleared by a system power-up or executing the "CLR WDT" or "HALT" instruction. TO is set by a WDT time-out. |
| 6, 7 | — | Unused bit, read as "0" |

Status (0AH) Register

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

INTC (0BH) Register

All 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 a specified location in the program memory. Only the program counter is pushed onto the stack. If the contents of the register or status register are altered by the interrupt service program which corrupts the desired control sequence, the contents should be saved in advance.

External interrupts are triggered by a high to low transition on the $\overline{\text{INT}}$ pin, which will set the related interrupt request flag, EIF, which is bit 4 of INTC. 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 initialised by setting the timer/event counter interrupt request flag, TF, which is bit 5 of INTC, caused by a timer overflow. When the interrupt is enabled, the stack is not full and the TF bit is set, a subroutine call to location 008H will occur. The related interrupt request flag, TF, will be reset and the EMI bit cleared to disable further interrupts.

The A/D converter interrupt is initialised by setting the A/D converter request flag, ADF, which is bit 6 of INTC1, caused by an end of A/D conversion. When the interrupt is enabled, the stack is not full and the ADF bit is set, a subroutine call to location 00CH 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. Of course, the stack must not be full. To return from the interrupt subroutine, a RET or RETI instruction may be executed. A RETI instruction will set the EMI bit to enable an interrupt service, but a RET instruction 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 | 004H |
| Timer/Event Counter Overflow | 2 | 008H |
| A/D Converter Interrupt | 3 | 010H |

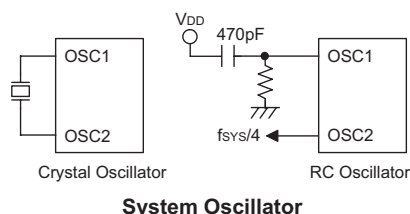
Once the interrupt request flags, TF, EIF, ADF, are set, they will remain in the INTC register until the interrupts are serviced or cleared by a software instruction. It is recommended that a program does not use the CALL subroutine within the interrupt subroutine. Interrupts often occur in an unpredictable manner or need to be serviced immediately in some applications. If only one stack is left and enabling the interrupt is not well controlled, the original control sequence will be damaged once the "CALL" operates in the interrupt subroutine.

Oscillator Configuration

There are two oscillator circuits in the microcontroller, namely an RC oscillator and a crystal oscillator, the choice of which is determined by a configuration option. When the system enters the Power-down mode the system oscillator stops and ignores external signals to conserve power.

If an RC oscillator is used, an external resistor between OSC1 and VSS is required whose resistance value must range from 24k Ω to 1M Ω . The system clock, divided by 4, can be monitored on pin OSC2 if a pull-high resistor is connected. This signal can be used to synchronise external logic. The RC oscillator provides the most cost effective solution, however the frequency of oscillation may vary with VDD, temperature and the process variations. It is, therefore, not suitable for timing sensitive operations where an accurate oscillator frequency is desired.

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



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

Watchdog Timer – WDT

The WDT clock source comes from either its own integrated RC oscillator, known as the WDT oscillator, or the instruction clock, which is the system clock divided by 4. The choice of which one is used is decided by a configuration option. This timer is designed to prevent a software malfunction or sequence from jumping to an unknown location with unpredictable results. The Watchdog Timer can be disabled by a configuration option. If the Watchdog Timer is disabled, all the executions related to the WDT result in no operation.

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

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

The WDT overflow under normal operation will initialise a "chip reset" and set the status bit "TO". But in the Power-down mode, the overflow will initialize a "warm reset", and only the program counter and SP are reset to zero. To clear the contents of the WDT, three methods are adopted; external reset (a low level on the $\overline{\text{RES}}$ pin), a software instruction and a HALT instruction. The software instruction include "CLR WDT" and the other set – "CLR WDT1" and "CLR WDT2". Of these two types of instruction, only one can be active depending on the

configuration option – "CLR WDT times selection option". If the "CLR WDT" is selected (i.e. CLR WDT times equal one), any execution of the "CLR WDT" instruction will clear the WDT. In the case that "CLR WDT1" and "CLR WDT2" are chosen (i.e. CLR WDT times equal two), these two instructions must be executed to clear the WDT; otherwise, the WDT may reset the chip as a result of time-out.

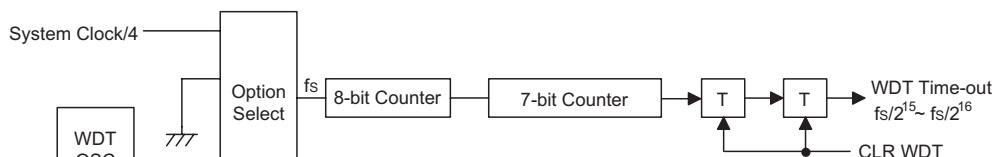
Power Down Operation – HALT

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

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

The system can leave the HALT mode by means of an external reset, an interrupt, an external falling edge signal on port A or a WDT overflow. An external reset causes a device initialisation and the WDT overflow performs a "warm reset". After the TO and PDF flags are examined, the reason for the chip reset can be determined. The PDF flag is cleared by a system power-up or executing the "CLR WDT" instruction and is set when executing the "HALT" instruction. The TO flag is set if the WDT time-out occurs, and causes a wake-up that only resets the program counter and Stack Pointer; the others keep their original status.

The port A wake-up and interrupt methods can be considered as a continuation of normal execution. Each bit in port A can be independently selected to wake up the device by the options. Awakening from an I/O port stimulus, the program will resume execution of the next instruction. If it is awakening from an interrupt, two sequences may happen. If the related interrupt is disabled or the interrupt is enabled but the stack is full, the program will resume execution at the next instruction. If the interrupt is enabled and the stack is not full, the regu-



Watchdog Timer

lar interrupt response takes place. If an interrupt request flag is set to "1" before entering the HALT mode, the wake-up function of the related interrupt will be disabled. Once a wake-up event occurs, it takes $1024 t_{SYS}$ (system clock period) to resume normal operation. In other words, a dummy period will be inserted after wake-up. If the wake-up results from an interrupt acknowledgment, the actual interrupt subroutine execution will be delayed by one or more cycles. If the wake-up results in the next instruction execution, this will be executed immediately after the dummy period is finished.

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

Reset

There are three ways in which a reset can occur:

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

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

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

Note: "u" means "unchanged"

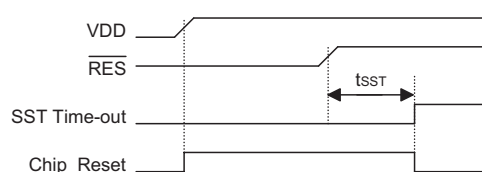
To guarantee that the system oscillator is started and stabilized, the SST (System Start-up Timer) provides an extra-delay of 1024 system clock pulses when the system reset (power-up, WDT time-out or \overline{RES} reset) or the system awakes from the HALT state.

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

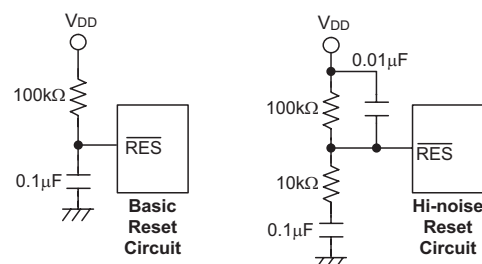
An extra option load time delay is added during system reset (power-up, WDT time-out at normal mode or \overline{RES} reset).

The functional unit chip reset status are shown below.

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

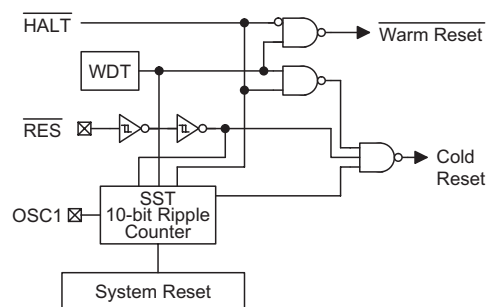


Reset Timing Chart



Reset Circuit

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



Reset Configuration

The registers' states are summarised in the following table.

| Register | Reset (Power On) | WDT Time-out (Normal Operation) | RES Reset (Normal Operation) | RES Reset (HALT) | WDT Times-out (HALT)* |
|----------|---------------------|------------------------------------|---------------------------------|---------------------|--------------------------|
| MP | 1xxx xxxx | 1uuu uuuu | 1uuu uuuu | 1uuu uuuu | 1uuu uuuu |
| ACC | xxxx xxxx | uuuu uuuu | uuuu uuuu | uuuu uuuu | uuuu uuuu |
| PCL | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 |
| TBLP | xxxx xxxx | uuuu uuuu | uuuu uuuu | uuuu uuuu | uuuu uuuu |
| TBLH | --xx xxxx | --uu uuuu | --uu uuuu | --uu uuuu | --uu uuuu |
| STATUS | --00 xxxx | --1u uuuu | --uu uuuu | --01 uuuu | --11 uuuu |
| INTC | -000 0000 | -000 0000 | -000 0000 | -000 0000 | -uuu uuuu |
| 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 |
| PA | 1111 1111 | 1111 1111 | 1111 1111 | 1111 1111 | uuuu uuuu |
| PAC | 1111 1111 | 1111 1111 | 1111 1111 | 1111 1111 | uuuu uuuu |
| PB | --11 1111 | --11 1111 | --11 1111 | --11 1111 | --uu uuuu |
| PBC | --11 1111 | --11 1111 | --11 1111 | --11 1111 | --uu uuuu |
| PD | -111 -111 | -111 -111 | -111 -111 | -111 -111 | -uuu -uuu |
| PDC | -111 -111 | -111 -111 | -111 -111 | -111 -111 | -uuu -uuu |
| PWM0 | xxxx xxxx | xxxx xxxx | xxxx xxxx | xxxx xxxx | uuuu uuuu |
| PWM1 | xxxx xxxx | xxxx xxxx | xxxx xxxx | xxxx xxxx | uuuu uuuu |
| OPAC | 0000 1000 | 0000 1000 | 0000 1000 | 0000 1000 | 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 | 1--- --00 | 1--- --00 | 1--- --00 | 1--- --00 | u--- --uu |

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

Timer/Event Counter

Two 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 external source or an internal clock source. An internal clock source comes from $f_{SYS}/4$. Using an external clock input allows the user to count external events, measure time intervals or pulse widths, or generate an accurate time base. While using the internal clock allows the user to generate an accurate time base.

There are two registers related to the Timer/Event Counter; TMR, TMRC. 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 & 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. In the pulse width measurement mode with the values of the TON & TE 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-function as long as it receives further transient pulse. In this operation mode, the timer/event counter begins counting not according to the logic level but to the transient edges. In the case of counter overflow, 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 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 by options. When the PFD function is selected, executing "SET [PA].3" instruction to enable PFD output and executing "CLR [PA].3" instruction to disable PFD output.

| Bit No. | Label | Function |
|-------------|-------------------------|--|
| 0 1 2 | TPSC0 TPSC1 TPSC2 | Defines the prescaler stages, PSC2, PSC1, PSC0= 000: $f_{INT}=f_{SYS}$ 001: $f_{INT}=f_{SYS}/2$ 010: $f_{INT}=f_{SYS}/4$ 011: $f_{INT}=f_{SYS}/8$ 100: $f_{INT}=f_{SYS}/16$ 101: $f_{INT}=f_{SYS}/32$ 110: $f_{INT}=f_{SYS}/64$ 111: $f_{INT}=f_{SYS}/128$ |
| 3 | TE | Defines the TMR active edge of the timer/event counter: In Event Counter Mode (TM1,TM0)=(0,1): 1:count on falling edge; 0:count on rising edge In Pulse Width measurement mode (TM1,TM0)=(1,1): 1: start counting on the rising edge, stop on the falling edge; 0: start counting on the falling edge, stop on the rising edge |
| 4 | TON | Enable or disable the timer counting (0=disable; 1=enable) |
| 5 | — | Unused bits, read as "0" |
| 6 7 | TM0 TM1 | Defines the operating mode (TM1, TM0)= 01=Event count mode (external clock) 10=Timer mode (internal clock) 11=Pulse width measurement mode 00=Unused |

TMRC (0EH) Register



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

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

After a device reset, the input/output lines will default to inputs and remain at a high level or floating state, dependent upon the pull-high configuration options. Each bit of these input/output latches can be set or cleared by the "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 line has a pull-high option. Once the pull-high configuration option is selected, the I/O line has a pull-high resistor, otherwise, there's none. Take note that a non-pull-high I/O line operating in input mode will cause a floating state.

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

"0" will force the PA3 to remain at "0". The I/O functions for 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.

Pins PD6 and PD5 are pin-shared with $\overline{\text{INT}}$, TMR pins respectively.

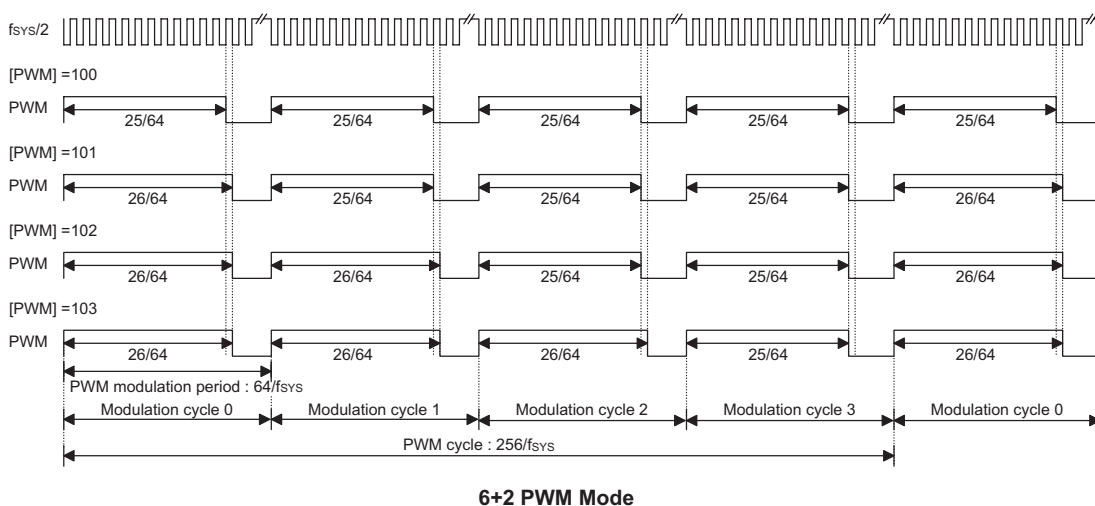
The PB can also be used as A/D converter inputs. The A/D function will be described later. There are two PWM functions shared with pins PD0 and PD1. If the PWM functions are enabled, the PWM signals will appear on PD0 and PD1, the pins are setup as outputs. Writing a "1" to the PD0 or PD1 data register will enable the PWM outputs to function while writing a "0" will force the PD0 and PD1 outputs to remain at "0". The I/O functions of PD0 and PD1 are as shown.

| I/O Mode | I/P (Normal) | O/P (Normal) | I/P (PWM) | O/P (PWM) |
|------------|---------------|----------------|---------------|--------------|
| PD0 PD1 | Logical Input | Logical Output | Logical Input | PWM0 PWM1 |

It is recommended that unused I/O lines should be setup as output pins by software instructions to avoid consuming power under input floating states.

PWM

The microcontroller provides a 2 channel (6+2) bits PWM0/PWM1 output shared with PD0/PD1. The PWM channel has its data register denoted as PWM0 and PWM1. The frequency source of the PWM counter comes from f_{SYS} . The PWM register is an eight bit register. Once PD0/PD1 are selected as PWM outputs and the output function of PD0/PD1 is enabled (PDC.0="0" or



PDC.1="0"), writing a 1 to the PD0/PD1 data register will enable the PWM output function while writing a "0" will force the PD0/PD1 outputs to stay at "0".

A PWM cycle is divided into four modulation cycles (modulation cycle 0~modulation cycle 3). Each modulation cycle has 64 PWM input clock 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.

Group 2 is denoted by AC which is the value of PWM.1~PWM.0.

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

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

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

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

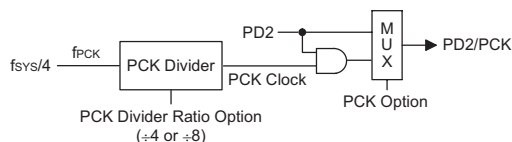
Peripheral Clock Output – PCK

The device also provides a Peripheral Clock Output (PCK) which is pin-shared with PD2. Once the PD2 is selected as the PCK outputs and the output function of PD2 is enabled (PDC.2="0"), writing 1 to PD2 data register will enable the PCK output function and writing "0" will force the PD2 to stay at "0".

The PCK clock frequency can be optional : $f_{SYS}/16$ or $f_{SYS}/32$ (by configuration option).

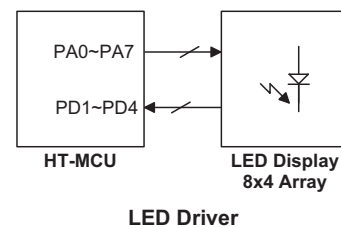
PCK output = 500kHz or 250kHz if $f_{SYS}=8\text{MHz}$.

Recommend to clear PD2 before entering HALT mode.



LED Driver

The device provides a maximum of 8x4 LED drivers which uses I/O Ports PA0~PA7 and PD1~PD4 with double the usual sink/source current drive capabilities. To use the LED driver function, PA0~PA7 and PD1~PD4 must be setup as outputs.



A/D Converter

The 6 channel 12-bit resolution A/D converter is implemented in the microcontrollers. The reference voltage for the A/D is VDD. The A/D converter contains 4 special registers, which are; ADRL, ADRH, ADCR and ACSR. The ADRH and ADRL registers are the A/D conversion result register higher-order byte and lower-order byte and are read-only. After the A/D conversion has completed, the ADRL and ADRH registers should be read to get the conversion result data. The ADCR register 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. It is used to start an A/D conversion, define the PB configuration, select the converted analog channel, and give the START bit a raising edge and a falling edge (0→1→0). At the end of an 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 register is an 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. Bit2~bit0 of the ADCR register is used to select an analog input channel. There are a total of 6 channels to select. Bit5~bit3 of the ADCR register is used to set the PB configurations. PB can be configured as an analog input or as a digital I/O line decided by these 3 bits. Once a PB line is selected as an analog input, the I/O functions and pull-high resistor of this I/O line are disabled, and the A/D converter circuit is powered on. The EOCB bit, bit6 of ADCR, is the end of A/D conversion flag. This bit is monitored to check when the A/D conversion has completed. The START bit of the ADCR register is used to initiate the A/D conversion process. When the START bit is provided with a raising edge and then a falling edge, the A/D conversion process will begin. In order to ensure that the A/D conversion is completed, the START should remain at "0" until the EOCB flag is cleared to "0" which indicates the end of the A/D conversion.

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

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

Important Note for A/D 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.

| Register | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|----------|------|------|------|------|------|------|------|------|
| ADRL | D3 | D2 | D1 | D0 | — | — | — | — |
| ADRH | D11 | D10 | D9 | D8 | D7 | D6 | D5 | D4 |

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

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

| Bit No. | Label | Function |
|---------|-------|--|
| 0 | ACS0 | ACS3[ACSR], ACS2, ACS1, ACS0: selected A/D channel |
| 1 | ACS1 | |
| 2 | ACS2 | |
| 3 | PCR0 | Port B, C configuration selection. If PCR0, PCR1, PCR2 and PCR3[ACSR] are all zero, the ADC circuit is power off to reduce power consumption |
| 4 | PCR1 | |
| 5 | PCR2 | |
| 6 | EOCB | Indicates end of A/D conversion. (0 = end of A/D conversion) Each time bits 3~5 change state the A/D should be 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. (0→1→0= start; 0→1= Reset A/D converter and set EOCB to "1") |

ADCR (22H) Register

| ACS2, ACS1, ACS0 | Channel |
|------------------|--------------------------|
| 000 | AN0 |
| 001 | AN1 |
| 010 | AN2 |
| 011 | AN3 |
| 100 | AN4 |
| 101 | AN5 |
| 11x | Undefined, can't be used |

A/D Channel

| Bit No. | Label | Function |
|---------|-------|---|
| 0 | ADCS0 | Select the A/D converter clock source. 0,0=f _{SYS} /2; 0,1=f _{SYS} /8 1,0=f _{SYS} /32; 1,1=Undefined |
| 1 | ADCS1 | |
| 2~6 | — | Unused bit, read as "0". |
| 7 | TEST | For internal test only. |

ACSR (23H) Register

| PCR2 | PCR1 | PCR0 | 5 | 4 | 3 | 2 | 1 | 0 |
|------|------|------|-----|-----|-----|-----|-----|-----|
| 0 | 0 | 0 | PB5 | PB4 | PB3 | PB2 | PB1 | PB0 |
| 0 | 0 | 1 | PB5 | PB4 | PB3 | PB2 | PB1 | AN0 |
| 0 | 1 | 0 | PB5 | PB4 | PB3 | PB2 | AN1 | AN0 |
| 0 | 1 | 1 | PB5 | PB4 | PB3 | AN2 | AN1 | AN0 |
| 1 | 0 | 0 | PB5 | PB4 | AN3 | AN2 | AN1 | AN0 |
| 1 | 0 | 1 | PB5 | AN4 | AN3 | AN2 | AN1 | AN0 |
| 1 | 1 | x | AN5 | AN4 | AN3 | AN2 | AN1 | AN0 |

Port B, Port C Configuration

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

Example: using EOCB Polling Method to detect end of conversion

```

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

```

Example: using interrupt method to detect end of conversion

```

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

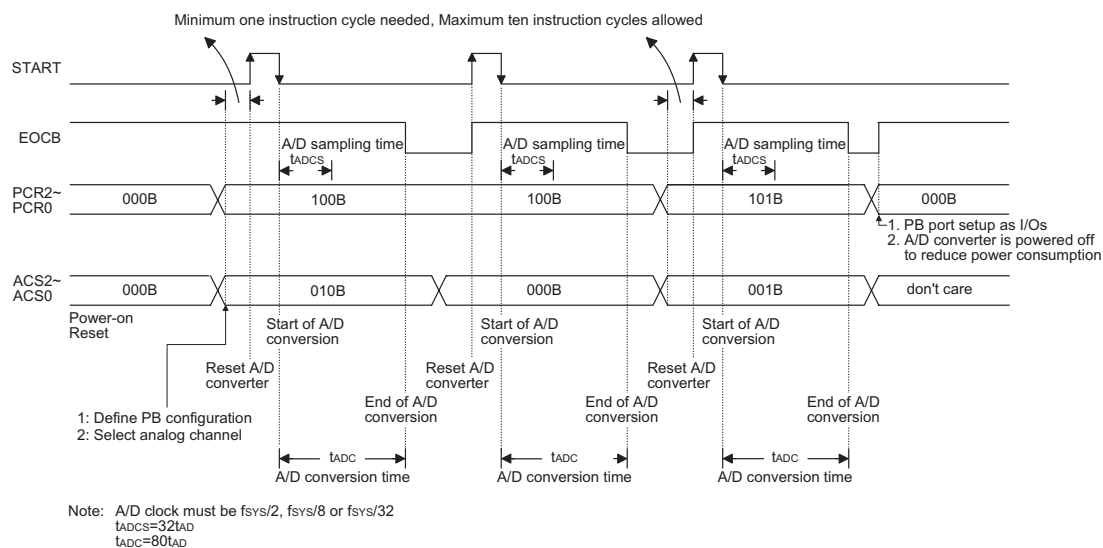
mov    a,00100000B        ; setup ADCR register to configure Port PB0~PB3 as A/D inputs
mov    ADCR,a              ; and select AN0 to be connected to the A/D converter
:
:                          ; As the Port B channel bits have changed the following START
:                          ; signal (0-1-0) must be issued within 10 instruction cycles
:
Start_conversion:
clr    START
set    START                ; reset A/D
clr    START                ; start A/D
clr    ADF                  ; clear ADC interrupt request flag

```

```

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

```


A/D Conversion Timing

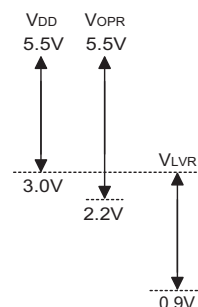
Low Voltage Reset – LVR

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

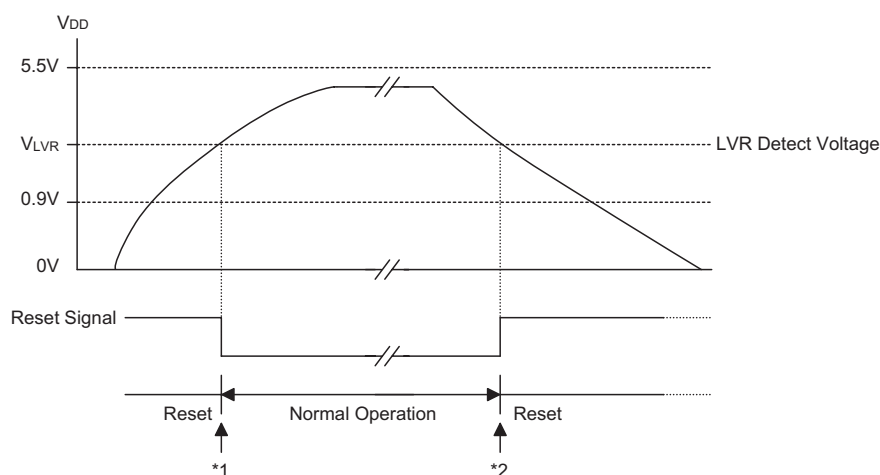
The LVR includes the following specifications:

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

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



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



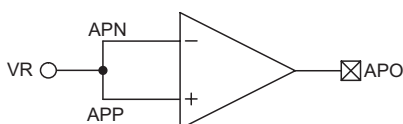
Low Voltage Reset

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

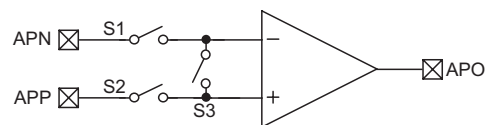
*2: Since the low voltage has to maintain in its original state and exceed t_{LVR} , therefore t_{LVR} delay enter the reset mode.

OP Amplifier/Comparator

The devices include an integrated operational amplifier or comparator, selectable via configuration option. The default is function is comparator. The input voltage offset is adjustable by using a common mode input to calibrate the offset value.



The calibration process is as follows:



- Set bit AOFM=1 to select the offset cancellation mode - this closes switch S3
- Set the ARS bit to select which input pin is the reference voltage - closes either switch S1 or S2
- Adjust bits AOF0~AOF3 until the output status OPAOP has changed.
- Set AOFM=0 to select the normal operating mode

| Bit No. | Label | Function |
|------------------|------------------------------|--|
| 0 1 2 3 | AOF0 AOF1 AOF2 AOF3 | OP amp/comparator input offset voltage cancellation control bits |
| 4 | ARS | OP amp/comparator input offset voltage cancellation reference selection bit 1/0 : select OPP/OPN (CP/CN) as the reference input |
| 5 | AOFM | Input offset voltage cancellation mode and OP amp/comparator mode selection 1: input offset voltage cancellation mode 0: OP amp/comparator |
| 6 | OPAOP | OP amp/comparator output; positive logic |
| 7 | OPAEN | OP amp/comparator enable/disable (1/0) If OP/comparator is disabled, output is floating. |

OPAC (1FH) Register

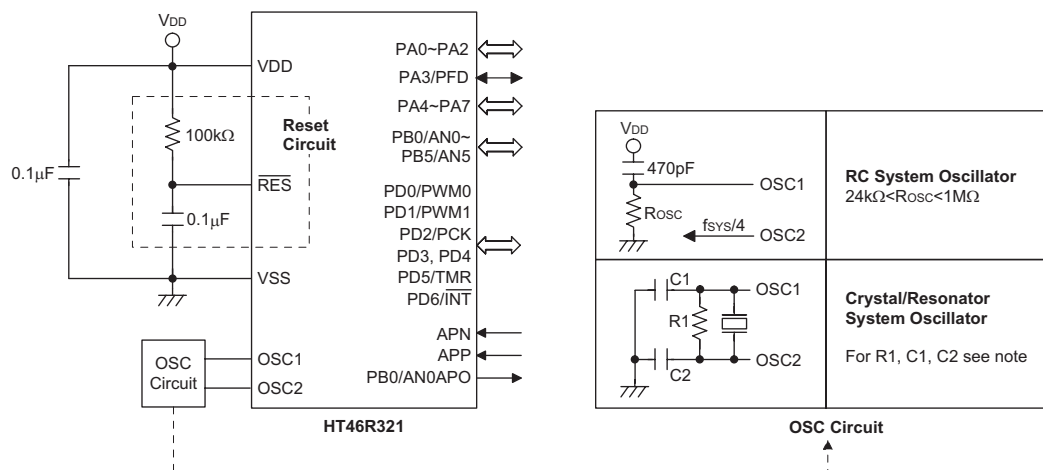
If the OP amp/comparator is disabled, the power consumption will be very small. To ensure that power consumption is minimised when the device is in the Power-down mode, the OP amp/comparator should be switched off by clearing bit OPAEN to 0 before entering the Power-down mode.

Configuration Options

The following table shows the various microcontroller configuration options. All of the configuration options must be properly defined to ensure correct system functioning.

| No. | Options |
|-----|--|
| 1 | WDT clock source: WDTOSC or T1 ($f_{SYS}/4$) |
| 2 | WDT function: enable or disable |
| 3 | CLRWDT instruction(s): one or two clear WDT instruction(s) |
| 4 | System oscillator: RC or crystal |
| 5 | Pull-high resistors (PA, PB, PD): none or pull-high |
| 6 | PWM enable or disable |
| 7 | PA0~PA7 wake-up: enable or disable |
| 8 | PFD enable or disable |
| 9 | Low voltage reset selection: enable or disable LVR function. |
| 10 | Comparator or OP selection |
| 11 | PCK or I/O |

Application Circuits



Note: 1. Crystal/resonator system oscillators

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

2. Reset circuit

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

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

Instruction Set 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

$ACC \leftarrow ACC + [m] + C$

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

$[m] \leftarrow ACC + [m] + C$

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

$ACC \leftarrow ACC + [m]$

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

$ACC \leftarrow ACC + x$

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

$[m] \leftarrow ACC + [m]$

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

 $ACC \leftarrow ACC \text{ "AND" } [m]$

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

 $ACC \leftarrow ACC \text{ "AND" } x$

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

 $[m] \leftarrow ACC \text{ "AND" } [m]$

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

 $Stack \leftarrow Program\ Counter + 1$
 $Program\ Counter \leftarrow addr$

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

 $[m] \leftarrow 00H$

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

 $[m].i \leftarrow 0$

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

 $WDT \leftarrow 00H$
 $PDF \text{ and } TO \leftarrow 0$

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

 $WDT \leftarrow 00H^*$
 $PDF \text{ and } TO \leftarrow 0^*$

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

 $WDT \leftarrow 00H^*$
 $PDF \text{ and } TO \leftarrow 0^*$

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

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

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 | $ACC \leftarrow \overline{[m]}$ | | | | | | | | | | | | |
| Affected flag(s) | <table><tr><td>TO</td><td>PDF</td><td>OV</td><td>Z</td><td>AC</td><td>C</td></tr><tr><td>—</td><td>—</td><td>—</td><td>√</td><td>—</td><td>—</td></tr></table> | TO | PDF | OV | Z | AC | C | — | — | — | √ | — | — |
| 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 $ACC.3 \sim ACC.0 > 9$ or $AC=1$ then $[m].3 \sim [m].0 \leftarrow (ACC.3 \sim ACC.0) + 6$, $AC1 = \overline{AC}$ else $[m].3 \sim [m].0 \leftarrow (ACC.3 \sim ACC.0)$, $AC1 = 0$ and If $ACC.7 \sim ACC.4 + AC1 > 9$ or $C = 1$ then $[m].7 \sim [m].4 \leftarrow ACC.7 \sim ACC.4 + 6 + AC1$, $C = 1$ else $[m].7 \sim [m].4 \leftarrow ACC.7 \sim ACC.4$, $C = C$ | | | | | | | | | | | | |
| Affected flag(s) | <table><tr><td>TO</td><td>PDF</td><td>OV</td><td>Z</td><td>AC</td><td>C</td></tr><tr><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td><td>√</td></tr></table> | TO | PDF | OV | Z | AC | C | — | — | — | — | — | √ |
| TO | PDF | OV | Z | AC | C | | | | | | | | |
| — | — | — | — | — | √ | | | | | | | | |
| DEC [m] | Decrement data memory | | | | | | | | | | | | |
| Description | Data in the specified data memory is decremented by 1. | | | | | | | | | | | | |
| Operation | $[m] \leftarrow [m] - 1$ | | | | | | | | | | | | |
| Affected flag(s) | <table><tr><td>TO</td><td>PDF</td><td>OV</td><td>Z</td><td>AC</td><td>C</td></tr><tr><td>—</td><td>—</td><td>—</td><td>√</td><td>—</td><td>—</td></tr></table> | TO | PDF | OV | Z | AC | C | — | — | — | √ | — | — |
| 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 | $ACC \leftarrow [m] - 1$ | | | | | | | | | | | | |
| Affected flag(s) | <table><tr><td>TO</td><td>PDF</td><td>OV</td><td>Z</td><td>AC</td><td>C</td></tr><tr><td>—</td><td>—</td><td>—</td><td>√</td><td>—</td><td>—</td></tr></table> | TO | PDF | OV | Z | AC | C | — | — | — | √ | — | — |
| 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

 $ACC \leftarrow x$

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

 $[m] \leftarrow ACC$

Affected flag(s)

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

NOP

No operation

Description

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

Operation

 $Program\ Counter \leftarrow Program\ Counter + 1$

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

 $ACC \leftarrow ACC \text{ "OR" } [m]$

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

 $ACC \leftarrow ACC \text{ "OR" } x$

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

 $[m] \leftarrow ACC \text{ "OR" } [m]$

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 \leftarrow 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 \leftarrow Stack

ACC \leftarrow 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 \leftarrow Stack

EMI \leftarrow 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) \leftarrow [m].i; [m].i:bit i of the data memory (i=0~6)

[m].0 \leftarrow [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) \leftarrow [m].i; [m].i:bit i of the data memory (i=0~6)

ACC.0 \leftarrow [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 | $[m].(i+1) \leftarrow [m].i$; $[m].i$:bit i of the data memory (i=0~6) $[m].0 \leftarrow C$ $C \leftarrow [m].7$ | | | | | | | | | | | | |
| Affected flag(s) | <table><tr><td>TO</td><td>PDF</td><td>OV</td><td>Z</td><td>AC</td><td>C</td></tr><tr><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td><td>√</td></tr></table> | TO | PDF | OV | Z | AC | C | — | — | — | — | — | √ |
| 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 | $ACC.(i+1) \leftarrow [m].i$; $[m].i$:bit i of the data memory (i=0~6) $ACC.0 \leftarrow C$ $C \leftarrow [m].7$ | | | | | | | | | | | | |
| Affected flag(s) | <table><tr><td>TO</td><td>PDF</td><td>OV</td><td>Z</td><td>AC</td><td>C</td></tr><tr><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td><td>√</td></tr></table> | TO | PDF | OV | Z | AC | C | — | — | — | — | — | √ |
| 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 | $[m].i \leftarrow [m].(i+1)$; $[m].i$:bit i of the data memory (i=0~6) $[m].7 \leftarrow [m].0$ | | | | | | | | | | | | |
| Affected flag(s) | <table><tr><td>TO</td><td>PDF</td><td>OV</td><td>Z</td><td>AC</td><td>C</td></tr><tr><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td></tr></table> | TO | PDF | OV | Z | AC | C | — | — | — | — | — | — |
| 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 | $ACC.(i) \leftarrow [m].(i+1)$; $[m].i$:bit i of the data memory (i=0~6) $ACC.7 \leftarrow [m].0$ | | | | | | | | | | | | |
| Affected flag(s) | <table><tr><td>TO</td><td>PDF</td><td>OV</td><td>Z</td><td>AC</td><td>C</td></tr><tr><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td></tr></table> | TO | PDF | OV | Z | AC | C | — | — | — | — | — | — |
| 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 | $[m].i \leftarrow [m].(i+1)$; $[m].i$:bit i of the data memory (i=0~6) $[m].7 \leftarrow C$ $C \leftarrow [m].0$ | | | | | | | | | | | | |
| Affected flag(s) | <table><tr><td>TO</td><td>PDF</td><td>OV</td><td>Z</td><td>AC</td><td>C</td></tr><tr><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td><td>√</td></tr></table> | TO | PDF | OV | Z | AC | C | — | — | — | — | — | √ |
| 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 | $ACC.i \leftarrow [m].(i+1)$; $[m].i$: bit i of the data memory (i=0~6) $ACC.7 \leftarrow C$ $C \leftarrow [m].0$ |

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 | $ACC \leftarrow ACC + \overline{[m]} + C$ |

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 | $[m] \leftarrow ACC + \overline{[m]} + C$ |

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 $([m]-1)=0$, $[m] \leftarrow ([m]-1)$

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 $([m]-1)=0$, $ACC \leftarrow ([m]-1)$

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

 $[m] \leftarrow FFH$

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

 $[m].i \leftarrow 1$

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 $([m]+1)=0$, $[m] \leftarrow ([m]+1)$

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 $([m]+1)=0$, $ACC \leftarrow ([m]+1)$

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 $[m].i \neq 0$

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

$$ACC \leftarrow ACC + \overline{[m]} + 1$$

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

$$[m] \leftarrow ACC + \overline{[m]} + 1$$

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

$$ACC \leftarrow ACC + \overline{x} + 1$$

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

$$[m].3 \sim [m].0 \leftrightarrow [m].7 \sim [m].4$$

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

$$ACC.3 \sim ACC.0 \leftarrow [m].7 \sim [m].4$$

$$ACC.7 \sim ACC.4 \leftarrow [m].3 \sim [m].0$$

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) | <table><tr><td>TO</td><td>PDF</td><td>OV</td><td>Z</td><td>AC</td><td>C</td></tr><tr><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td></tr></table> | TO | PDF | OV | Z | AC | C | — | — | — | — | — | — |
| 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) | <table><tr><td>TO</td><td>PDF</td><td>OV</td><td>Z</td><td>AC</td><td>C</td></tr><tr><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td></tr></table> | TO | PDF | OV | Z | AC | C | — | — | — | — | — | — |
| 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) | <table><tr><td>TO</td><td>PDF</td><td>OV</td><td>Z</td><td>AC</td><td>C</td></tr><tr><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td></tr></table> | TO | PDF | OV | Z | AC | C | — | — | — | — | — | — |
| 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) | <table><tr><td>TO</td><td>PDF</td><td>OV</td><td>Z</td><td>AC</td><td>C</td></tr><tr><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td></tr></table> | TO | PDF | OV | Z | AC | C | — | — | — | — | — | — |
| 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) | <table><tr><td>TO</td><td>PDF</td><td>OV</td><td>Z</td><td>AC</td><td>C</td></tr><tr><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td></tr></table> | TO | PDF | OV | Z | AC | C | — | — | — | — | — | — |
| 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

 $ACC \leftarrow ACC \text{ "XOR" } [m]$

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

 $[m] \leftarrow ACC \text{ "XOR" } [m]$

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

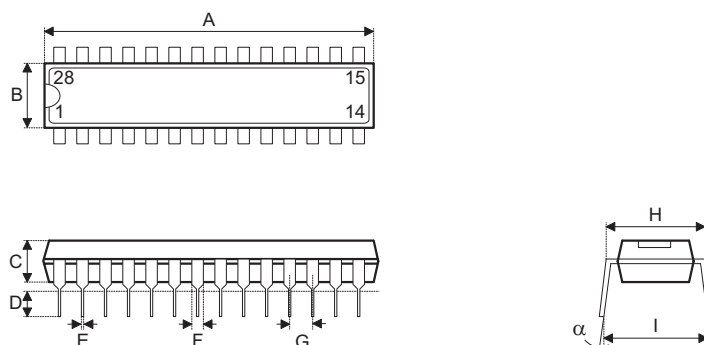
 $ACC \leftarrow ACC \text{ "XOR" } x$

Affected flag(s)

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

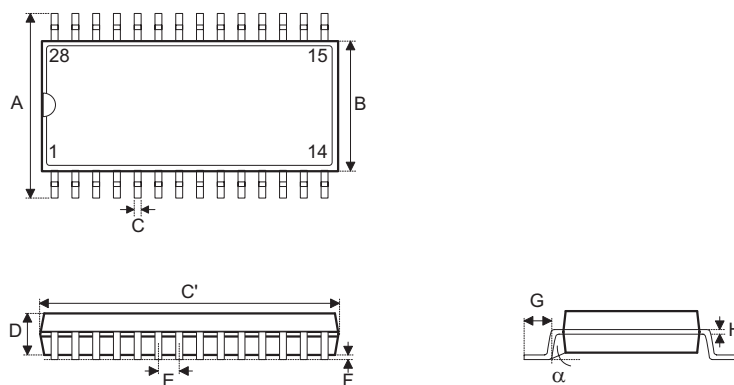
Package Information

28-pin SKDIP (300mil) Outline Dimensions

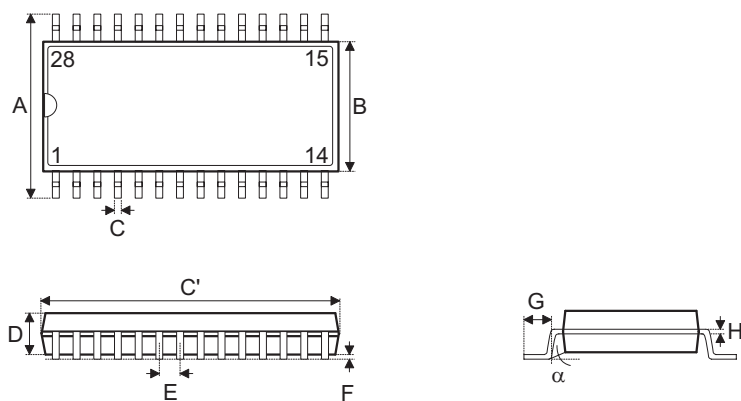


| Symbol | Dimensions in mil | | |
|----------|-------------------|------|------|
| | Min. | Nom. | Max. |
| A | 1375 | — | 1395 |
| B | 278 | — | 298 |
| C | 125 | — | 135 |
| D | 125 | — | 145 |
| E | 16 | — | 20 |
| F | 50 | — | 70 |
| G | — | 100 | — |
| H | 295 | — | 315 |
| I | 330 | — | 375 |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

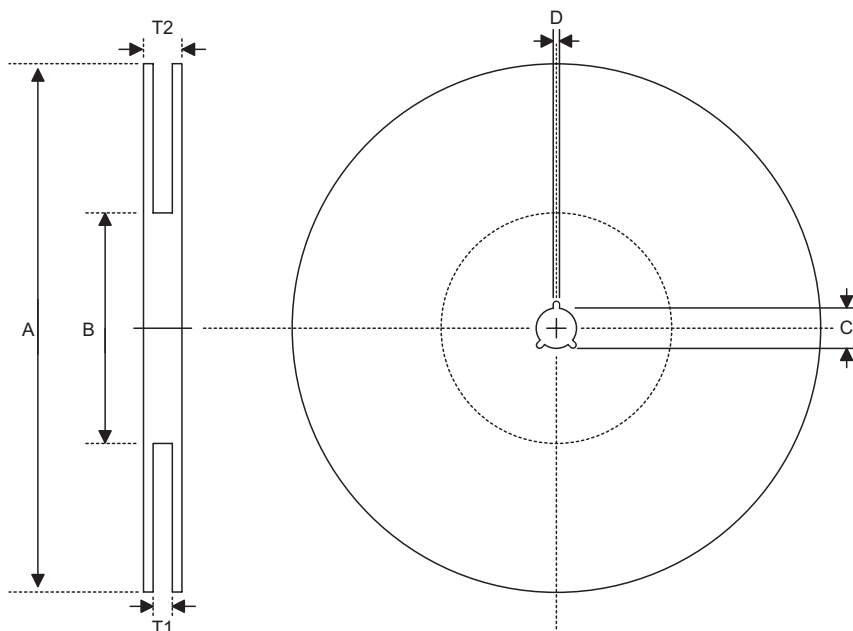
28-pin SOP (300mil) Outline Dimensions



| Symbol | Dimensions in mil | | |
|----------|-------------------|------|------|
| | Min. | Nom. | Max. |
| A | 394 | — | 419 |
| B | 290 | — | 300 |
| C | 14 | — | 20 |
| C' | 697 | — | 713 |
| D | 92 | — | 104 |
| E | — | 50 | — |
| F | 4 | — | — |
| G | 32 | — | 38 |
| H | 4 | — | 12 |
| α | 0° | — | 10° |

28-pin SSOP (150mil) Outline Dimensions


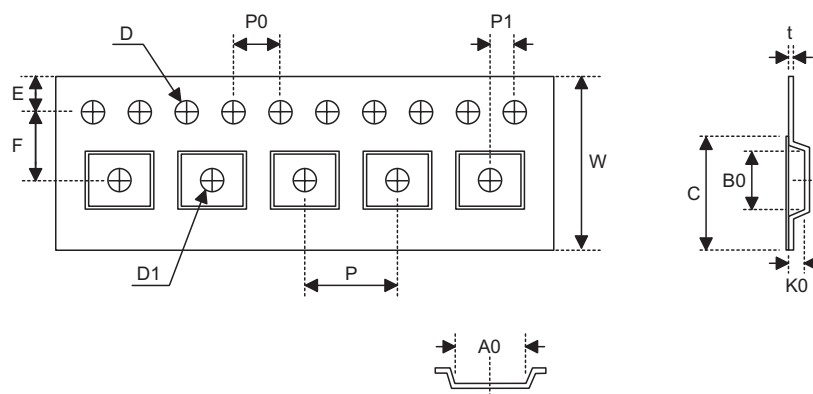
| Symbol | Dimensions in mil | | |
|----------|-------------------|------|------|
| | Min. | Nom. | Max. |
| A | 228 | — | 244 |
| B | 150 | — | 157 |
| C | 8 | — | 12 |
| C' | 386 | — | 394 |
| D | 54 | — | 60 |
| E | — | 25 | — |
| F | 4 | — | 10 |
| G | 22 | — | 28 |
| H | 7 | — | 10 |
| α | 0° | — | 8° |

Product Tape and Reel Specifications
Reel Dimensions

SOP 28W (300mil)

| Symbol | Description | Dimensions in mm |
|--------|-----------------------|------------------|
| A | Reel Outer Diameter | 330±1.0 |
| B | Reel Inner Diameter | 62±1.5 |
| C | Spindle Hole Diameter | 13.0+0.5 -0.2 |
| D | Key Slit Width | 2.0±0.5 |
| T1 | Space Between Flange | 24.8+0.3 -0.2 |
| T2 | Reel Thickness | 30.2±0.2 |

SSOP 28S (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 -0.2 |
| D | Key Slit Width | 2±0.5 |
| T1 | Space Between Flange | 16.8+0.3 -0.2 |
| T2 | Reel Thickness | 22.2±0.2 |

Carrier Tape Dimensions

SOP 28W

| Symbol | Description | Dimensions in mm |
|--------|--|------------------|
| W | Carrier Tape Width | 24.0±0.3 |
| P | Cavity Pitch | 12.0±0.1 |
| E | Perforation Position | 1.75±0.1 |
| F | Cavity to Perforation (Width Direction) | 11.5±0.1 |
| D | Perforation Diameter | 1.5±0.1 |
| D1 | Cavity Hole Diameter | 1.5±0.25 |
| P0 | Perforation Pitch | 4.0±0.1 |
| P1 | Cavity to Perforation (Length Direction) | 2.0±0.1 |
| A0 | Cavity Length | 10.85±0.1 |
| B0 | Cavity Width | 18.34±0.1 |
| K0 | Cavity Depth | 2.97±0.1 |
| t | Carrier Tape Thickness | 0.35±0.01 |
| C | Cover Tape Width | 21.3 |

SSOP 28S (150mil)

| Symbol | Description | Dimensions in mm |
|--------|--|------------------|
| W | Carrier Tape Width | 16±0.3 |
| 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 | 1.55±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 | 10.3±0.1 |
| K0 | Cavity Depth | 2.1±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: 86-21-6485-5560
Fax: 86-21-6485-0313
<http://www.holtek.com.cn>

Holtek Semiconductor Inc. (Shenzhen Sales Office)

5/F, Unit A, Productivity Building, Cross of Science M 3rd Road and Gaoxin M 2nd Road, Science Park, Nanshan District, Shenzhen, China 518057
Tel: 86-755-8616-9908, 86-755-8616-9308
Fax: 86-755-8616-9722

Holtek Semiconductor Inc. (Beijing Sales Office)

Suite 1721, Jinyu Tower, A129 West Xuan Wu Men Street, Xicheng District, Beijing, China 100031
Tel: 86-10-6641-0030, 86-10-6641-7751, 86-10-6641-7752
Fax: 86-10-6641-0125

Holtek Semiconductor Inc. (Chengdu Sales Office)

709, Building 3, Champagne Plaza, No.97 Dongda Street, Chengdu, Sichuan, China 610016
Tel: 86-28-6653-6590
Fax: 86-28-6653-6591

Holtek Semiconductor (USA), Inc. (North America Sales Office)

46729 Fremont Blvd., Fremont, CA 94538
Tel: 1-510-252-9880
Fax: 1-510-252-9885
<http://www.holtek.com>

Copyright © 2007 by HOLTEK SEMICONDUCTOR INC.

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