www.DataSheet4U.com

HT46RB50
A/D Type USB 8-Bit MCU

Technical Document

- Tools Information
- FAQs
- Application Note

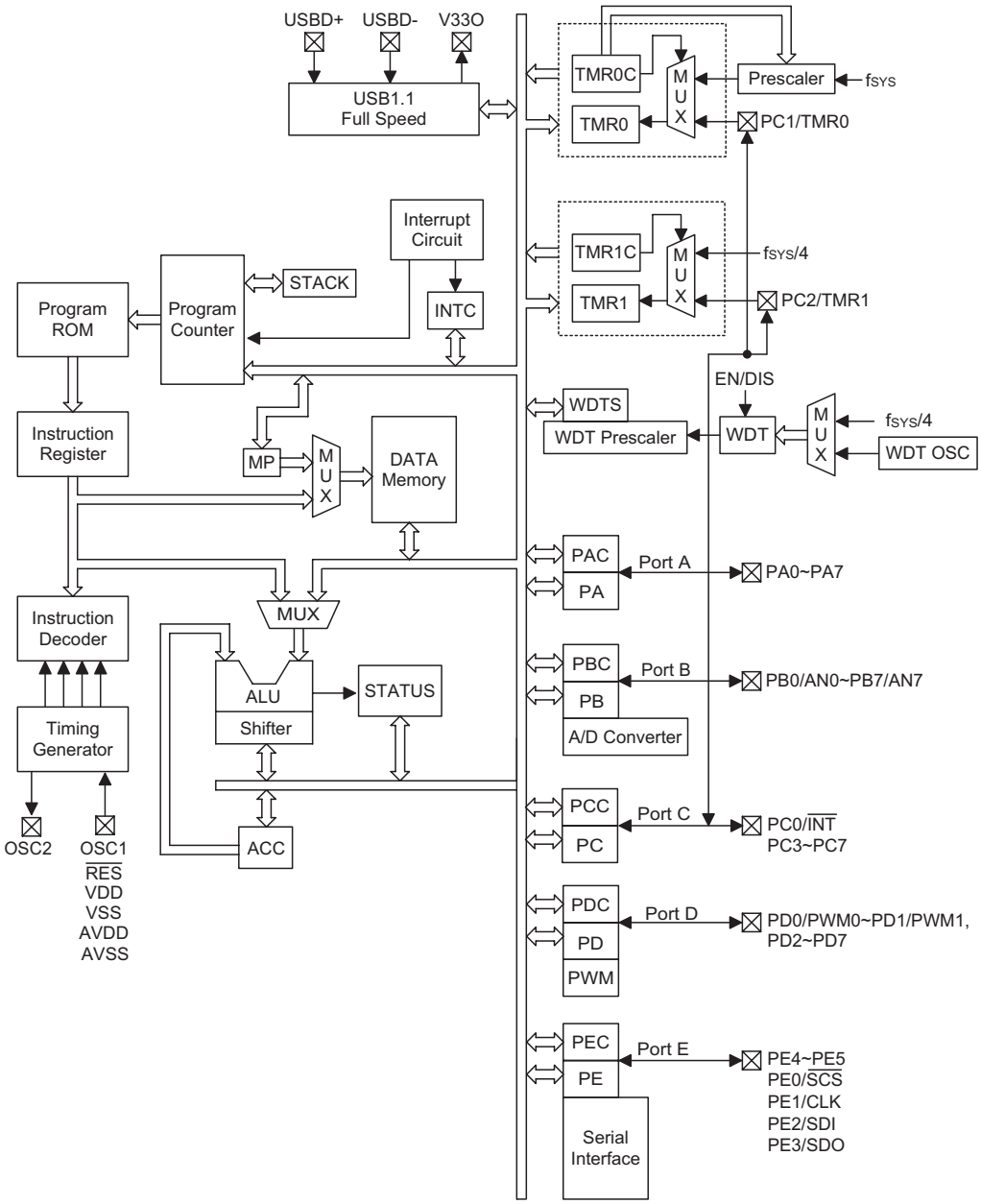
Features

- Operating voltage:
  f_{SYS}=6MHz: 2.2V~5.5V
  f_{SYS}=12MHz: 2.7V~5.5V
- 38 bidirectional I/O lines (max.)
- 1 interrupt input shared with an I/O line
- One 16-bit programmable timer/event counter with overflow interrupt
- One 8-bit programmable timer/event counter with overflow interrupt and 7 stage prescaler
- Only crystal oscillator (6MHz or 12MHz)
- Watchdog Timer
- 4096×15 program memory
- 192×8 data memory RAM
- HALT function and wake-up feature reduce power consumption
- Up to 0.33μs instruction cycle with 12MHz system clock at V_{DD}=5V
- 6-level subroutine nesting
- 8 channels 10-bit resolution A/D converter
- 2-channel 8-bit PWM output shared with two I/O lines
- SIO (synchronous serial I/O) function
- Supports Interrupt, Control, Bulk transfer
- USB 1.1 full speed function compatible
- 4 endpoints supported (endpoint 0 included)
- Total FIFO size is 88 byte (8, 8, 8, 64 for EP0~EP3)
- Bit manipulation instruction
- 15-bit table read instruction
- 63 powerful instructions
- All instructions in one or two machine cycles
- Low voltage reset function
- 28-pin SOP/SKDIP, 48-pin SSOP package

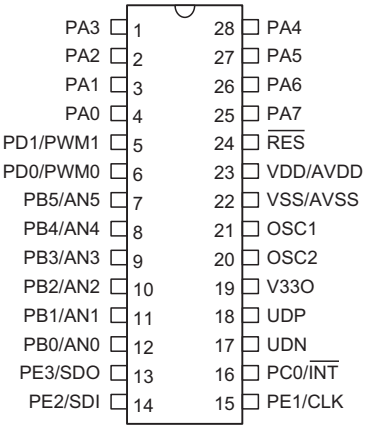
General Description

This device is an 8-bit high performance RISC architecture microcontroller designed for USB product applications. It is particularly suitable for use in products such as USB and/or SPI touch-panels, USB and/or SPI touch-pads, PS II joysticks, XBOX joysticks, USB Mice keyboards and joystick. A HALT feature is included to reduce power consumption.

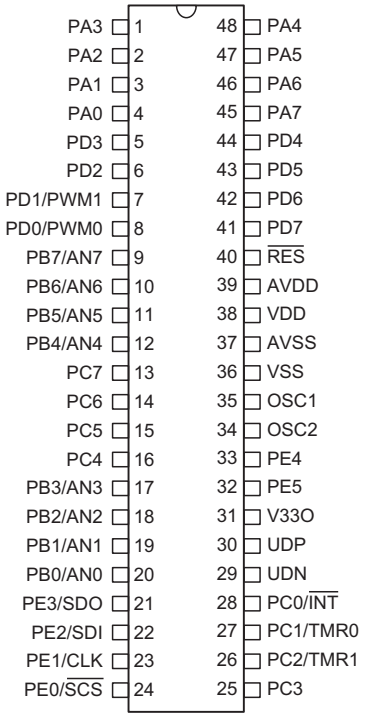
Block Diagram



Pin Assignment



HT46RB50
−28 SOP-A/SKDIP-A



HT46RB50
− 48 SSOP-A

Pin Description

Pin Name	I/O	Options	Description
PA0~PA7	I/O	Pull-high (bit option) Wake-up (bit option)	Bidirectional 8-bit input/output port. Each bit can be configured as a wake-up input by ROM code option. The input or output mode is controlled by PAC (PA control register, bit option). Pull-high resistor options: PA0~PA7, bit option, wake-up options: PA0~PA7.
PB0/AN0~ PB7/AN7	I/O	Pull-high (bit option)	Bidirectional 8-bit input/output port. Software instructions determine the CMOS output or Schmitt trigger input with pull-high resistor (determined by pull-high options: bit option). The PB can be used as analog input of the analog to digital converter.
PC0/$\overline{\text{INT}}$ PC1/TMR0 PC2/TMR1 PC3~PC7	I/O	Pull-high (nibble option)	Bidirectional I/O lines. Software instructions determine the CMOS output or Schmitt trigger input with pull-high resistor (determined by pull-high options: nibble option). The PC0, PC1 PC2 are pin-shared with $\overline{\text{INT}}$, TMR0 or TMR1, respectively.
PD0/PWM0~ PD1/PWM1 PD2~PD7	I/O	Pull-high (nibble option) I/O or PWM	Bidirectional I/O lines. Software instructions determine the CMOS output or Schmitt trigger input with pull-high resistor (determined by pull-high options: nibble option). The PD0/PD1 are pin-shared with PWM0/PWM1 (dependent on PWM options).
PE0/$\overline{\text{SCS}}$	I/O	Pull-high (nibble option)	Bidirectional I/O lines. Software instructions determine the CMOS output or Schmitt trigger input with pull-high resistor (determined by pull-high options: nibble option). The PE0 is pin-shared with $\overline{\text{SCS}}$. $\overline{\text{SCS}}$ is a chip select pin of the Serial interface, Master mode is output, Slave mode is input.
PE1/CLK	I/O	Pull-high (nibble option)	Bidirectional I/O lines. Software instructions determine the CMOS output or Schmitt trigger input with pull-high resistor (determined by pull-high options: nibble option). The PE1 is pin-shared with CLK. CLK is a Serial interface serial clock input/output (Initial is input).

Pin Name	I/O	Options	Description
PE2/SDI	I/O	Pull-high (nibble option)	Bidirectional I/O lines. Software instructions determine the CMOS output or Schmitt trigger input with pull-high resistor (determined by pull-high options: nibble option). The PE2 is pin-shared with SDI. SDI is Serial interface serial input.
PE3/SDO	I/O	Pull-high (nibble option)	Bidirectional I/O lines. Software instructions determine the CMOS output or Schmitt trigger input with pull-high resistor (determined by pull-high options: nibble option). The PE3 is pin-shared with SDO. SDO is a Serial interface serial output.
PE4~PE5	I/O	Pull-high (nibble option)	Bidirectional I/O lines. Software instructions determine the CMOS output or Schmitt trigger input with pull-high resistor (determined by pull-high options: nibble option).
$\overline{RES}$	I	—	Schmitt trigger reset input, active low
VSS	—	—	Negative power supply, ground
AVSS	—	—	ADC negative power supply, ground
VDD	—	—	Positive power supply
AVDD	—	—	ADC positive power supply, AVDD should be externally connected to VDD.
OSC1 OSC2	I O	—	OSC1 and OSC2 are connected to a 6MHz or 12MHz Crystal/resonator (determined by software instructions) for the internal system clock.
V33O	O	—	3.3V regulator output.
UDP	I/O	—	UDP is USBD+ line USB function is controlled by software control register.
UDN	I/O	—	UDN is USBD- line USB function is controlled by software control register.

Absolute Maximum Ratings

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

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

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

D.C. Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	Operating Voltage	—	f_{SYS}=6MHz	2.2	—	5.5	V
			f_{SYS}=12MHz	2.7	—	5.5	V
I_{DD1}	Operating Current (6MHz Crystal)	5V	No load, f_{SYS}=6MHz	—	6.5	12	mA
I_{DD2}	Operating Current (12MHz Crystal)	3V	No load, f_{SYS}=12MHz	—	3.6	10	mA
		5V		—	7.5	16	mA
I_{STB1}	Standby Current (WDT Enabled)	3V	No load, system HALT, USB suspended	—	—	5	μA
		5V		—	—	10	μA
I_{STB2}	Standby Current (WDT Disabled)	3V	No load, system HALT, USB suspended	—	—	1	μA
		5V		—	—	2	μA

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{STB3}	Standby Current (WDT Disabled)	5V	No load, system HALT, USB transceiver and 3.3V regulator On	—	150	200	μA
V_{IL1}	Input Low Voltage for I/O Ports	—	—	0	—	$0.3V_{DD}$	V
V_{IH1}	Input High Voltage for I/O Ports	—	—	$0.7V_{DD}$	—	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
I_{OL}	I/O Port Sink Current	3V	$V_{OL}=0.1V_{DD}$	4	8	—	mA
		5V		10	20	—	mA
I_{OH}	I/O Port Source Current	3V	$V_{OH}=0.9V_{DD}$	–2	–4	—	mA
		5V		–5	–10	—	mA
R_{PH}	Pull-high Resistance	3V	—	20	60	100	kΩ
		5V		10	30	50	kΩ
V_{LVR}	Low Voltage Reset Voltage	—	Option 3.0V	2.7	3	3.3	V
V_{V33O}	3.3V Regulator Output	5V	I_{V33O}=–5mA	3	3.3	3.6	V
E_{AD}	A/D Conversion Error	—	—	—	±0.5	±1	LSB

A.C. Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
f_{SYS}	System Clock	—	2.2V~5.5V	400	—	6000	kHz
		—	3.3V~5.5V	400	—	12000	kHz
f_{TIMER}	Timer I/P Frequency (TMR0/TMR1)	—	2.2V~5.5V	0	—	6000	kHz
		—	3.3V~5.5V	0	—	12000	kHz
t_{WDTOSC}	Watchdog Oscillator Period	3V	—	45	90	180	μs
		5V	—	32	65	130	μs
t_{RES}	External Reset Low Pulse Width	—	—	1	—	—	μs
t_{SST}	System Start-up Timer Period	—	Wake-up from HALT	—	1024	—	$*t_{SYS}$
t_{INT}	Interrupt Pulse Width	—	—	1	—	—	μs
t_{AD}	A/D Clock Period	—	—	1	—	—	μs
t_{ADC}	A/D Conversion Time	—	—	—	76	—	t_{AD}
t_{ADCS}	A/D Sampling Time	—	—	—	32	—	t_{AD}

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

Functional Description

Execution Flow

The system clock is derived from a crystal. It 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 one instruction cycle while decoding and execution takes the next instruction cycle. The pipelining scheme makes it possible for each instruction to be effectively executed in a cycle. If an instruction changes the value of the program counter, two cycles are required to complete the instruction.

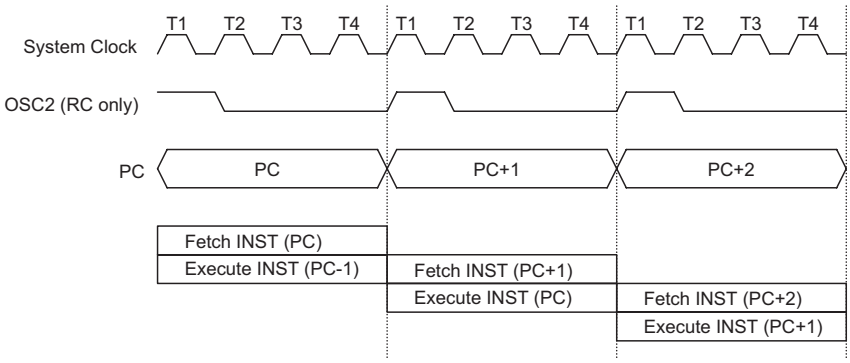
Program Counter – PC

The program counter (PC) is 12 bits wide and it controls the sequence in which the instructions stored in the program ROM are executed. The contents of the PC can specify a maximum of 4096 addresses. After accessing a program memory word to fetch an instruction code, the value of the PC is incremented by 1. The PC then points to the memory word containing the next instruction code. When executing a jump instruction, conditional skip execution, loading a PCL register, a subroutine call, an initial reset, an internal interrupt, an external interrupt, or returning from a subroutine, the PC manages 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 a proper instruction, otherwise proceed to the next instruction.

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

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



Execution Flow

Mode	Program Counter											
	*11	*10	*9	*8	*7	*6	*5	*4	*3	*2	*1	*0
Initial Reset	0	0	0	0	0	0	0	0	0	0	0	0
External Interrupt	0	0	0	0	0	0	0	0	0	1	0	0
Timer/Event Counter 0 Overflow	0	0	0	0	0	0	0	0	1	0	0	0
Timer/Event Counter 1 Overflow	0	0	0	0	0	0	0	0	1	1	0	0
USB Interrupt	0	0	0	0	0	0	0	1	0	0	0	0
A/D Converter Interrupt	0	0	0	0	0	0	0	1	0	1	0	0
Serial Interface Interrupt	0	0	0	0	0	0	0	1	1	0	0	0
Skip	Program Counter+2											
Loading PCL	*11	*10	*9	*8	@7	@6	@5	@4	@3	@2	@1	@0
Jump, Call Branch	#11	#10	#9	#8	#7	#6	#5	#4	#3	#2	#1	#0
Return from Subroutine	S11	S10	S9	S8	S7	S6	S5	S4	S3	S2	S1	S0

Program Counter

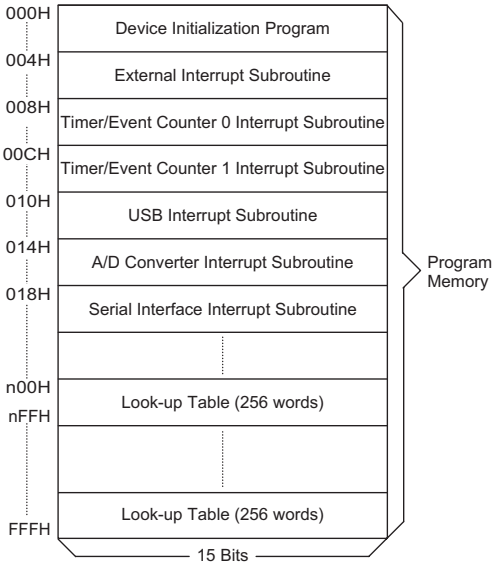
Note: *11~*0: Program counter bits
#11~#0: Instruction code bits
S11~S0: Stack register bits
@7~@0: PCL bits

Program Memory – EPROM

The program memory (EPROM) is used to store the program instructions which are to be executed. It also contains data, table, and interrupt entries, and is organized into 4096×15 bits which are addressed by the Program Counter and table pointer.

Certain locations in the ROM are reserved for special usage:

- Location 000H
  Location 000H is reserved for program initialization. After a chip reset, the program always begins execution at this location.
- Location 004H
  Location 004H is reserved for the external interrupt service program. If the $\overline{INT}$ input pin is activated, and the interrupt is enabled, and the stack is not full, the program begins execution at location 004H.
- Location 008H
  Location 008H is reserved for the Timer/Event Counter 0 interrupt service program. If a timer interrupt results from a Timer/Event Counter 0 overflow, and if the interrupt is enabled and the stack is not full, the program begins execution at location 008H.



Note: n ranges from 0 to F

Program Memory

- Location 00CH
  Location 00CH is reserved for the Timer/Event Counter 1 interrupt service program. If a timer interrupt results from a Timer/Event Counter 1 overflow, and if the interrupt is enabled and the stack is not full, the program begins execution at location 00CH.
- Location 010H
  Location 010H is reserved for the USB interrupt service program. If the USB interrupt is activated, the interrupt is enabled and the stack is not full, the program begins execution at location 010H.
- Location 014H
  Location 014H is reserved for the A/D converter interrupt service program. If an A/D converter interrupt results from an end of A/D conversion, and the stack is not full, the program begins execution at location 014H.
- Location 018H
  Location 018H is reserved when 8 bits data have been received or transmitted successfully from serial interface, and the related interrupts are enabled, and the stack is not full, the program begins execution at location 018H.
- Table location
  Any location in the ROM can be used as a look-up table. The instructions "TABRDC [m]" (the current page, page=256 words) and "TABRDL [m]" (the last page) transfer the contents of the lower-order byte to the specified data memory, and the contents of the higher-order byte to TBLH (Table Higher-order byte register) (08H). Only the destination of the lower-order byte in the table is well-defined; the other bits of the table word are all transferred to the lower portion of TBLH. The TBLH is read only, and the table pointer (TBLP) is a read/write register (07H), indicating the table location. Before accessing the table, the location should be placed in TBLP. All the table related instructions require 2 cycles to complete the operation. These areas may function as a normal ROM depending upon users requirements.

Stack Register – STACK

This is a special part of the memory which is used to save the contents of the Program Counter only. The stack is organized into 6 levels and is neither part of the data nor part of the program space, and is neither readable nor writeable. The activated level is indexed by the stack pointer (SP) and is neither readable nor writeable.

Instruction	Table Location											
	*11	*10	*9	*8	*7	*6	*5	*4	*3	*2	*1	*0
TABRDC [m]	P11	P10	P9	P8	@7	@6	@5	@4	@3	@2	@1	@0
TABRDL [m]	1	1	1	1	@7	@6	@5	@4	@3	@2	@1	@0

Table Location

Note: *11~*0: Table location bits
@7~@0: Table pointer bits
P11~P8: Current program counter bits

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

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

Data Memory – RAM

The data memory (RAM) is designed with 238×8 bits, and is divided into two functional groups, namely; special function registers (46×8 bits) and general purpose data memory (192×8 bits) most of which are readable/writeable, although some are read only.

The remaining space before 40H is reserved for future expanded usage and reading these locations will get ″00H″. The general purpose data memory, addressed from 40H to FFH, is used for data and control information under instruction commands.

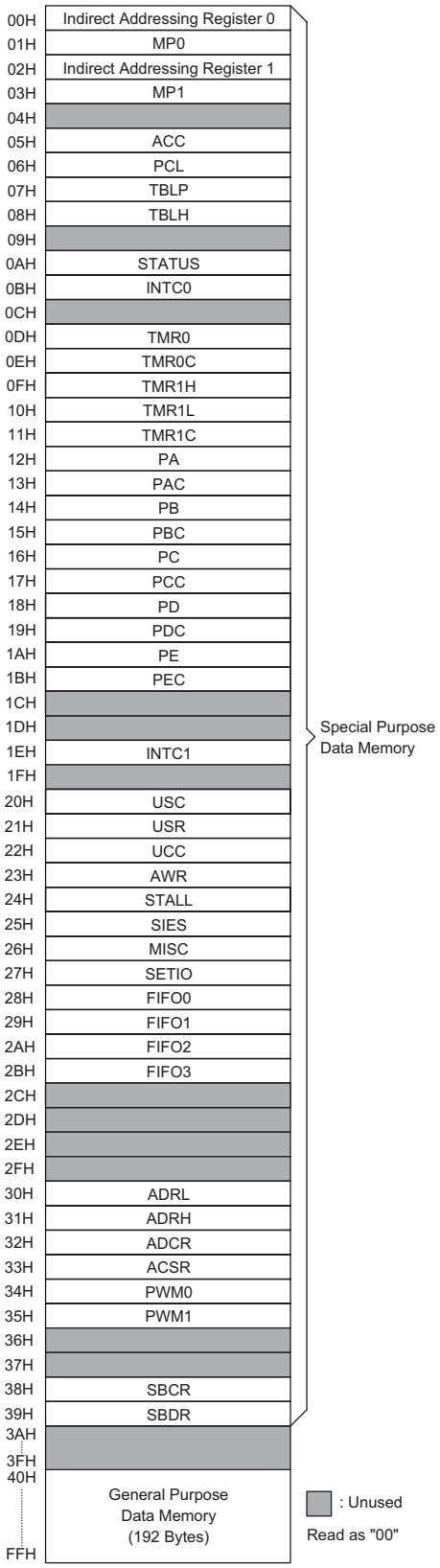
All of the data memory areas can handle arithmetic, logic, increment, decrement and rotate operations directly. Except for some dedicated bits, each bit in the data memory can be set and reset by ″SET [m].i″ and ″CLR [m].i″. They are also indirectly accessible through memory pointer registers (MP0;01H/MP1;03H).

Indirect Addressing Register

Location 00H and 02H are indirect addressing registers that are not physically implemented. Any read/write operation of [00H] and [02H] accesses the RAM pointed to by MP0 (01H) and MP1 (03H) respectively. Reading location 00H or 02H indirectly returns the result 00H. While, writing into it, indirectly leads to no operation. The function of data movement between two indirect addressing registers is not supported. The memory pointer registers, MP0 and MP1, are both 8-bit registers used to access the RAM by combining corresponding indirect addressing registers.

Accumulator – ACC

The accumulator is closely related to ALU operations. It is also mapped to location 05H of the RAM and capable of operating with immediate data. The data movement between two data memory locations must pass through the accumulator.



RAM Mapping

Arithmetic and Logic Unit – ALU

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

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

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

Status Register – STATUS

The status register (0AH) is 8 bits wide and contains, a carry flag (C), an auxiliary carry flag (AC), a zero flag (Z), an overflow flag (OV), a power down flag (PDF), and a Watchdog time-out flag (TO). It also records the status information and controls the operation sequence. Except for the TO and PDF flags, bits in the status register can be altered by instructions similar to other registers. Data written into the status register does not alter the TO or PDF flags. Operations related to the status register, however, may yield different results from those intended. The TO and PDF flags can only be changed by a Watchdog Timer overflow, chip power-up, or clearing the Watchdog Timer and executing the ″HALT″ instruction.

The Z, OV, AC, and C flags reflect the status of the latest operations. On entering the interrupt sequence or executing a subroutine call, the status register will not be automatically pushed onto the stack. If the contents of the status is important, and if the subroutine is likely to corrupt the status register, the programmer should take precautions and save it properly.

Interrupts

This device provides external interrupts ($\overline{\text{INT}}$ pin interrupt, A/D Converter interrupt, Serial Interface interrupt) and internal timer/event counter interrupts. The Interrupt Control Register0 (INTC0;0BH) and interrupt control register1 (INTC1:1EH) both contain the interrupt control bits that are used to set the enable/disable status and interrupt request flags.

Once an interrupt subroutine is serviced, all the other interrupts will be blocked (by clearing the EMI bit). This scheme may prevent any further interrupt nesting. Other interrupt requests may occur during this interval but only the interrupt request flag is recorded. If a certain interrupt requires servicing within the service routine, the EMI bit and the corresponding bit of the INTC0 or INTC1 may be set to allow interrupt nesting. If the stack is full, the interrupt request will not be acknowledged, even if the related interrupt is enabled, until the SP is decremented. If immediate service is desired, the stack must be prevented from becoming full.

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

External interrupts can are triggered by a falling edge transition of $\overline{\text{INT}}$), and the related interrupt request flag (EIF; bit4 of the INTC0) is set as well. After the interrupt is enabled, the stack is not full, and the external interrupt is active ($\overline{\text{INT}}$ pin), a subroutine call at location 04H occurs. The interrupt flag (EIF) and EMI bits are all cleared to disable other maskable interrupts.

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

The internal Timer/Event Counter 0 interrupt is initialized by setting the Timer/Event Counter 0 interrupt request flag (bit 5 of the INTC0), caused by a Timer 0 overflow. When the interrupt is enabled, the stack is not full and the T0F bit is set, a subroutine call to location 08H will occur. The related interrupt request flag (T0F) will be reset and the EMI bit cleared to disable further interrupts.

The internal Timer/Event Counter 1 interrupt is initialized by setting the Timer/Event Counter 1 interrupt request flag (bit 6 of the INTC0), caused by a Timer 1 overflow. When the interrupt is enabled, the stack is not full and the T1F is set, a subroutine call to location 0CH will occur. The related interrupt request flag (T1F) will be reset and the EMI bit cleared to disable further interrupts.

USB interrupts are triggered by the following USB events and the related interrupt request flag (USBF; bit 4 of the INTC1) will be set.

- The access of the corresponding USB FIFO from PC
- The USB suspend signal from the PC
- The USB resume signal from the PC
- USB Reset signal

When the interrupt is enabled, the stack is not full and the external interrupt is active, a subroutine call to location 10H will occur. The interrupt request flag (USBF) and EMI bits will be cleared to disable other interrupts.

When PC Host access the FIFO of the HT46RB50, the corresponding request bit of USR is set, and a USB interrupt is triggered. So user can easily determine which FIFO is accessed. When the interrupt has been served, the corresponding bit should be cleared by firmware. When the HT46RB50 receives a USB Suspend signal from the Host PC, the suspend line (bit0 of the USC) of the HT46RB50 is set and a USB interrupt is also triggered.

Also when the HT46RB50 receives a Resume signal from the Host PC, the resume line (bit3 of the) of the HT46RB50 is set and a USB interrupt is triggered.

Whenever a USB reset signal is detected, a USB interrupt is triggered.

The A/D converter interrupt is controlled by setting the A/D interrupt control bit (EADI; bit 1 of the INTC1). When the interrupt is enabled, the stack is not full and the A/D conversion is finished, a subroutine call to location 14H will occur. The related interrupt request flag ADF (bit5 of the INTC1) will be reset and the EMI bit cleared to disable further interrupts.

The serial interface interrupt is indicated by the interrupt flag (SIF; bit 6 of the INTC1), that is caused by a reception or a complete transmission of an 8-bit data between the HT46RB50 and an external device. The serial interface interrupt is controlled by setting the Serial interface interrupt control bit (ESII; bit 2 of the INTC1). After the

Bit No.	Label	Function
0	EMI	Controls the master (global) interrupt (1= enable; 0= disable)
1	EEI	Controls the external interrupt (1= enable; 0= disable)
2	ET0I	Controls the Timer/Event Counter 0 interrupt (1= enable; 0= disable)
3	ET1I	Controls the Timer/Event Counter 1 interrupt (1= enable; 0= disable)
4	EIF	External interrupt request flag (1= active; 0= inactive)
5	T0F	Internal Timer/Event Counter 0 request flag (1= active; 0= inactive)
6	T1F	Internal Timer/Event Counter 1 request flag (1= active; 0= inactive)
7	—	Unused bit, read as ″0″

INTC0 (0BH) Register

Bit No.	Label	Function
0	EUI	Control the USB interrupt (1= enable; 0= disable)
1	EADI	Control the A/D converter interrupt (1= enable; 0=disable)
2	ESII	Control Serial interface interrupt (1= enable; 0= disable)
3, 7	—	Unused bit, read as ″0″
4	USBF	USB interrupt request flag (1= active; 0= inactive)
5	ADF	A/D converter request flag (1= active; 0= inactive)
6	SIF	Serial interface interrupt request flag (1= active; 0= inactive)

INTC1 (1EH) Register

interrupt is enabled (by setting SBEN; bit 4 of the SBCR), and the stack is not full and the SIF is set, a subroutine call to location 18H occurs.

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

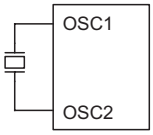
Interrupts, occurring in the interval between the rising edges of two consecutive T2 pulses, will be serviced on the latter of the two T2 pulses, if the corresponding interrupts are enabled. In the case of simultaneous requests the following table shows the priority that is applied. These can be masked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

Interrupt Source	Priority	Vector
External Interrupt	1	04H
Timer/Event Counter 0 Overflow	2	08H
Timer/Event Counter 1 Overflow	3	0CH
USB Interrupt	4	10H
A/D Converter Interrupt	5	14H
Serial Interface Interrupt	6	18H

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 is an oscillator circuit in the microcontroller.



Crystal Oscillator

System Oscillator

This oscillator is designed for system clocks. The HALT mode stops the system oscillator and ignores an external signal to conserve power.

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.

The WDT oscillator is a free running on-chip RC oscillator, and no external components are required. Even if the system enters a power down mode and the system clock is stopped, but the WDT oscillator still works. The WDT oscillator can be disabled by ROM code option to conserve power.

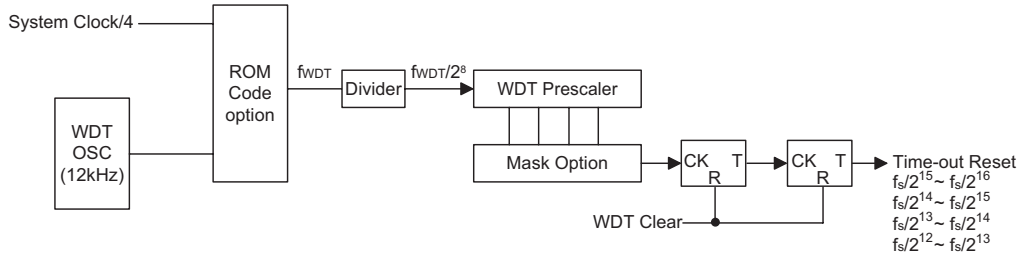
Watchdog Timer – WDT

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

Once an internal WDT oscillator (RC oscillator with a period of 65μs, normally at 5V) is selected, it is divided by 2^{12}~2^{15} (by option to get the WDT time-out period). The WDT time-out minimum period is 300ms~600ms. This time-out period may vary with temperature, VDD and process variations. By selection from the WDT option, longer time-out periods can be realized. If the WDT time-out is selected as 2^{15}, the maximum time-out period is divided by 2^{15}~2^{16} which about 2.3s~4.7s.

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

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



Watchdog Timer

Power Down Operation – HALT

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

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

The system can quit the HALT mode in many ways, by an external reset, an interrupt, an external falling edge signal on Port A or a WDT overflow. An external reset causes a device initialization and the WDT overflow performs a "warm reset". After examining the TO and PDF flags, the cause for a chip reset can be determined. The PDF flag is cleared by a system power-up or by executing the "CLR WDT" instruction and is set when executing the "HALT" instruction. On the other hand, the TO flag is set if the WDT time-out occurs, and causes a wake-up that only resets the Program Counter and SP; and leaves the others in their original status.

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

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

Reset

There are three ways in which a reset may occur:

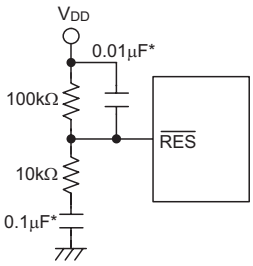
- $\overline{RES}$ reset during normal operation
- $\overline{RES}$ reset during HALT
- WDT time-out reset during normal operation

The WDT time-out during HALT differs from other chip reset conditions, for it can perform a "warm reset" that resets only the Program Counter and SP, leaving the other circuits in their original state. Some registers remain unaffected during any other reset conditions. Most registers are reset to the "initial condition" when the reset conditions are met. Examining the PDF and TO flags, the program can distinguish between different "chip resets".

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

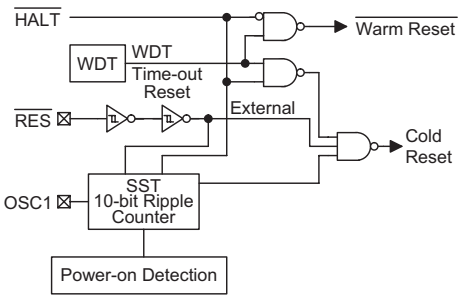
Note: "u" stands for "unchanged"

To guarantee that the system oscillator is started and stabilized, the SST (System Start-up Timer) provides an extra-delay of 1024 system clock pulses when the system awakes from the HALT state or during power up.

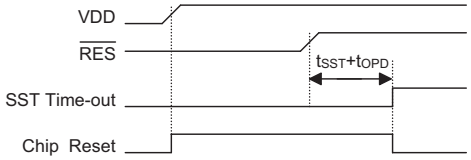


Reset Circuit

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



Reset Configuration



Reset Timing Chart

Awaking from the HALT state or system power up, an SST delay is added. An extra SST delay is added during power up period, and any wake-up from HALT may enable only the SST delay. The functional unit chip reset status are shown below.

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

The registers states are summarized in the following table.

Register	Reset (Power On)	WDT Time-out (Normal Operation)	RES Reset (Normal Operation)	RES Reset (HALT)	WDT Time-out (HALT)*	USB Reset (Normal)	USB Reset (HALT)
TMR0	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu	xxxx xxxx	xxxx xxxx
TMR0C	00-0 1000	00-0 1000	00-0 1000	00-0 1000	uu-u uuuu	00-0 1000	00-0 1000
TMR1H	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu	xxxx xxxx	xxxx xxxx
TMR1L	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu	xxxx xxxx	xxxx xxxx
TMR1C	00-0 1---	00-0 1---	00-0 1---	00-0 1---	uu-u u---	00-0 1---	00-0 1---
Program Counter	000H	000H	000H	000H	000H	000H	000H
MP0	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
MP1	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLP	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLH	-xxxx xxxx	-uuu uuuu	-uuu uuuu	-uuu uuuu	-uuu uuuu	-uuu uuuu	-uuu uuuu
STATUS	--00 xxxx	--1u uuuu	--uu uuuu	--01 uuuu	--11 uuuu	--uu uuuu	--01 uuuu
INTC0	-000 0000	-000 0000	-000 0000	-000 0000	-uuu uuuu	-000 0000	-000 0000
INTC1	-000 -000	-000 -000	-000 -000	-000 -000	-uuu -uuu	-000 -000	-000 -000
PA	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu	1111 1111	1111 1111

Register	Reset (Power On)	WDT Time-out (Normal Operation)	$\overline{\text{RES}}$ Reset (Normal Operation)	$\overline{\text{RES}}$ Reset (HALT)	WDT Time-out (HALT)*	USB Reset (Normal)	USB Reset (HALT)
PAC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu	1111 1111	1111 1111
PB	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu	1111 1111	1111 1111
PBC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu	1111 1111	1111 1111
PC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu	1111 1111	1111 1111
PCC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu	1111 1111	1111 1111
PD	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu	1111 1111	1111 1111
PDC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu	1111 1111	1111 1111
PE	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu	1111 1111	1111 1111
PEC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu	1111 1111	1111 1111
AWR	0000 0000	uuuu uuuu	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
STALL	---- 1110	---- uuuu	---- 1110	---- 1110	---- uuuu	---- 1110	---- 1110
MISC	0xx- -000	uxx- -uuu	0xx- -000	0xx- -000	uxx- -uuu	000- -000	000- -000
SETIO	---- 1110	---- uuuu	---- 1110	---- 1110	---- uuuu	---- 1110	---- 1110
FIFO0	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	0000 0000	0000 0000
FIFO1	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	0000 0000	0000 0000
FIFO2	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	0000 0000	0000 0000
FIFO3	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	0000 0000	0000 0000
USC	1-00 0000	u-uu uuuu	1-00 0000	1-00 0000	u-uu uuuu	u-00 0100	u-00 0100
USR	--00 0000	--uu uuuu	--00 0000	--00 0000	--uu uuuu	--00 0000	--00 0000
UCC	-000 0000	-uuu uuuu	-000 0000	-000 0000	-uuu uuuu	-uu0 u000	-uu0 u000
SIES	0100 0000	uuuu uuuu	0100 0000	0100 0000	uuuu uuuu	0100 0000	0100 0000
ADRL	xx-- ----	xx-- ----	xx-- ----	xx-- ----	uu-- ----	xx-- ----	xx-- ----
ADRH	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu	xxxx xxxx	xxxx xxxx
ADCR	0100 0000	0100 0000	0100 0000	0100 0000	uuuu uuuu	0100 0000	0100 0000
ACSR	1--- --00	1--- --00	1--- --00	1--- --00	u--- --uu	1--- --00	1--- --00
PWM0	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
PWM1	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
SBCR	0110 0000	0110 0000	0110 0000	0110 0000	uuuu uuuu	uuuu uuuu	uuuu uuuu
SBDR	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu

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

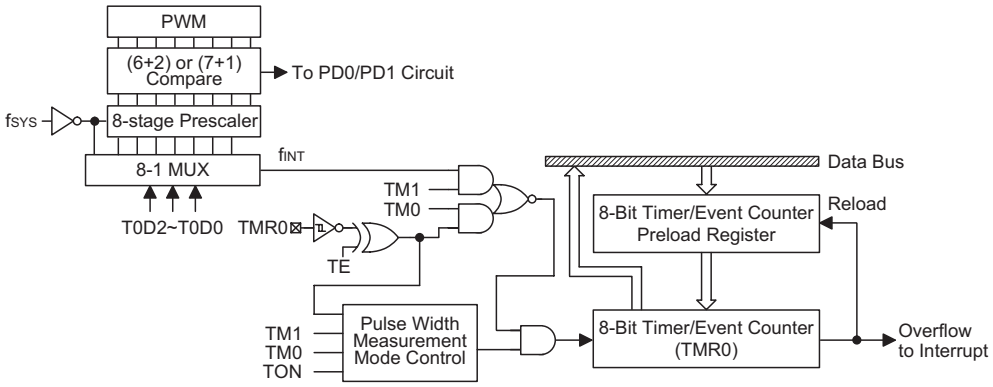
Timer/Event Counter

Two Timer/Event Counters (TMR0, TMR1) are implemented in the microcontroller. The Timer/Event Counter 0 contains a 8-bit programmable count-up counter and the clock may come from an external source or an internal clock source. An internal clock source comes from f_{SYS}. The Timer/Event Counter 1 contains a 16-bit programmable count-up counter and the clock may come from an external source or an internal clock source. An internal clock source comes from $f_{SYS}/4$. The external clock input allows the user to count external events, measure time intervals or pulse widths, or generate an accurate time base.

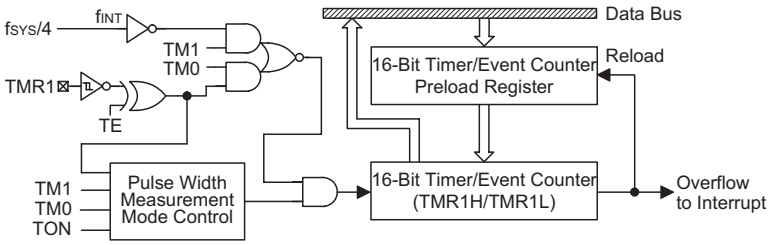
There are five registers related to the Timer/Event Counter 0; TMR0 (0DH), TMR0C (0EH) and the Timer/Event Counter 1; TMR1H (0FH), TMR1L (10H), TMR1C (11H). For 16bits timer to Write data to TMR1L will only put the written data to an internal lower-order byte buffer (8-bit) and writing TMR1H will transfer the specified data and the contents of the lower-order byte buffer to TMR1H and TMR1L registers. The Timer/Event Counter 1 preload register is changed by each writing TMR1H operations. Reading TMR1H will latch the contents of TMR1H and TMR1L counters to the destination and the lower-order byte buffer, respectively. Reading the TMR1L will read the contents of the lower-order byte buffer. The TMR0C (TMR1C) is the Timer/Event Counter 0 (1) control register, which defines the operating mode, counting enable or disable and an active edge.

The T0M0, T0M1 (TMR0C) and T1M0, T1M1 (TMR1C) bits define the operation mode. The event count mode is used to count external events, which means that the clock source is from an external (TMR0, TMR1) pin. The timer mode functions as a normal timer with the clock source coming from the internal selected clock source. Finally, the pulse width measurement mode can be used to count the high or low level duration of the external signal (TMR0, TMR1), and the counting is based on the internal selected clock source.

In the event count or timer mode, the timer/event counter starts counting at the current contents in the timer/event counter and ends at FFFFH(for 16 bits timer is FFFFH, bit 8 bits timer will be FFH). Once an overflow occurs, the counter is reloaded from the timer/event counter preload register, and generates an interrupt request flag (T0F; bit 5 of the INTC0, T1F; bit 6 of the INTC0).



Timer/Event Counter 0



Timer/Event Counter 1

In the pulse width measurement mode with the values of the T0ON/T1ON and T0E/T1E bits equal to 1, after the TMR0 (TMR1) has received a transient from low to high (or high to low if the T0E/T1E bit is ″0″), it will start counting until the TMR0 (TMR1) returns to the original level and resets the T0ON/T1ON. The measured result remains in the timer/event counter even if the activated transient occurs again. In other words, only 1-cycle measurement can be made until the T0ON/T1ON is set. The cycle measurement will re-function as long as it receives further transient pulse. In this operation mode, the timer/event counter begins counting not according to the logic level but to the transient edges. In the case of counter overflows, the counter is reloaded from the timer/event counter register and issues an interrupt request, as in the other two modes, i.e., event and timer modes.

To enable the counting operation, the Timer ON bit (T0ON: bit 4 of the TMR0C; T10N: bit 4 of the TMR1C) should be set to 1. In the pulse width measurement mode, the T0ON/T1ON is automatically cleared after the measurement cycle is completed. But in the other two modes, the T0ON/T1ON can only be reset by instructions. The overflow of the Timer/Event Counter 0/1 is one of the wake-up sources. No matter what the operation mode is, writing a 0 to ET0I or ET1I disables the related interrupt service.

In the case of timer/event counter OFF condition, writing data to the timer/event counter preload register also reloads that data to the timer/event counter. But if the timer/event counter is turned on, data written to the timer/event counter is kept only in the timer/event counter preload register. The timer/event counter still continues its operation until an overflow occurs.

When the timer/event counter (reading TMR0/TMR1) is read, the clock is blocked to avoid errors, as this may results in a counting error. Blocking of the clock should be taken into account by the programmer. It is strongly recommended to load a desired value into the TMR0/TMR1 register first, before turning on the related timer/event counter, for proper operation since the initial value of TMR0/TMR1 is unknown. Due to the timer/event scheme, the programmer should pay special attention on the instruction to enable then disable the timer for the first time, whenever there is a need to use the timer/event function, to avoid unpredictable result. After this procedure, the timer/event function can be operated normally.

The bit0~bit2 of the TMR0C can be used to define the pre-scaling stages of the internal clock sources of timer/event counter. The definitions are as shown. The timer prescaler is also used as the PWM counter.

Bit No.	Label	Function
0 1 2	T0PSC0 T0PSC1 T0PSC2	Defines the prescaler stages, T0PSC2, T0PSC1, T0PSC0= 000: $f_{INT}=f_{SYS}$ 001: $f_{INT}=f_{SYS}/2$ 010: $f_{INT}=f_{SYS}/4$ 011: $f_{INT}=f_{SYS}/8$ 100: $f_{INT}=f_{SYS}/16$ 101: $f_{INT}=f_{SYS}/32$ 110: $f_{INT}=f_{SYS}/64$ 111: $f_{INT}=f_{SYS}/128$
3	T0E	Defines the TMR active edge of the timer/ event counter (0=active on low to high; 1=active on high to low)
4	T0ON	Enable/disable timer counting (0=disable; 1=enable)
5	—	Unused bit, read as ″0″
6 7	T0M0 T0M1	Defines the operating mode, T0M1, T0M0: 01=Event count mode (external clock) 10=Timer mode (internal clock) 11=Pulse width measurement mode 00=Unused

TMR0C (0EH) Register

Bit No.	Label	Function
0~2, 5	—	Unused bit, read as ″0″
3	T1E	Defines the TMR active edge of the timer/ event counter (0=active on low to high; 1=active on high to low)
4	T1ON	Enable/disable timer counting (0=disable; 1=enable)
6 7	T1M0 T1M1	Defines the operating mode, T1M1, T1M0: 01=Event count mode (external clock) 10=Timer mode (internal clock) 11=Pulse width measurement mode 00=Unused

TMR1C (11H) Register

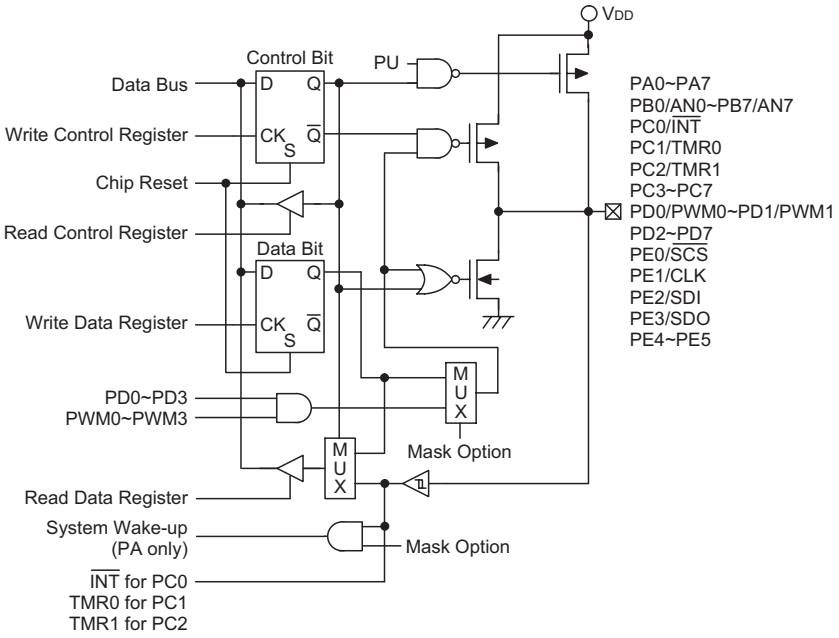
Input/Output Ports

There are 38 bidirectional input/output lines in the microcontroller, labeled from PA to PE, which are mapped to the data memory of [12H], [14H], [16H], [18H] and [1A] respectively. All of these I/O ports can be used for input and output operations. For input operation, these ports are non-latching, that is, the inputs must be ready at the T2 rising edge of instruction ″MOV A,[m]″ (m=12H, 14H, 16H, 18H or 1A). For output operation, all the data is latched and remains unchanged until the output latch is rewritten.

Each I/O line has its own control register (PAC, PBC, PCC, PDC, PEC) to control the input/output configuration. With this control register, CMOS output or Schmitt trigger input with or without pull-high resistor structures can be reconfigured dynamically under software control. To function as an input, the corresponding latch of the control register must write a ″1″. The input source also depends on the control register. If the control register bit is ″1″ the input will read the pad state. If the control register bit is ″0″ the contents of the latches will move to the internal bus. The latter is possible in the ″Read-modify-write″ instruction. For output function, CMOS is the only configuration. These control registers are mapped to locations 13H, 15H, 17H, 19H and 1BH.

After a chip reset, these input/output lines remain at high levels or floating state (depending on the pull-high options). Each bit of these input/output latches can be set or cleared by ″SET [m].i″ and ″CLR [m].i″ (m=12H, 14H, 16H, 18H or 1AH) 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.



Input/Output Ports

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

It is recommended that unused or not bonded out I/O lines should be set as output pins by software instruction to avoid consuming power under input floating state.

Pulse Width Modulator – PWM

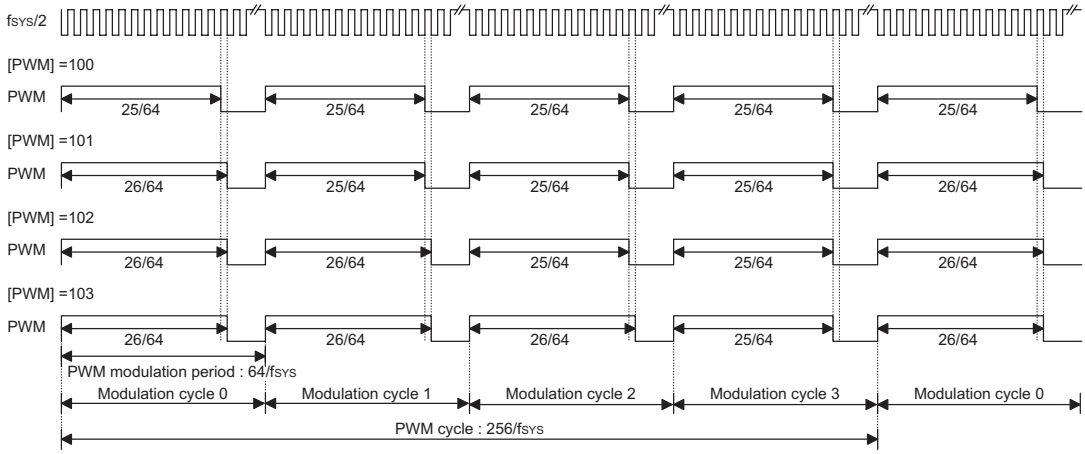
The microcontroller provides 2 channels (6+2)/(7+1) (depending on options) bits PWM output shared with PD0/PD1. The data register of the PWM channels are denoted as PWM0 (34H) and PWM1 (35H). The frequency source of the PWM counter comes from f_{SYS}. There are four 8-bit PWM registers. The waveforms of the PWM outputs are as shown. Once the PD0/PD1 are selected as the PWM outputs and the output function of PD0/PD1 are enabled (PDC.0/PDC.1=″0″), writing a ″1″ to PD0/PD1 data register will enable the PWM output function and writing a ″0″ will force the PD0/PD1 to remain at ″0″.

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

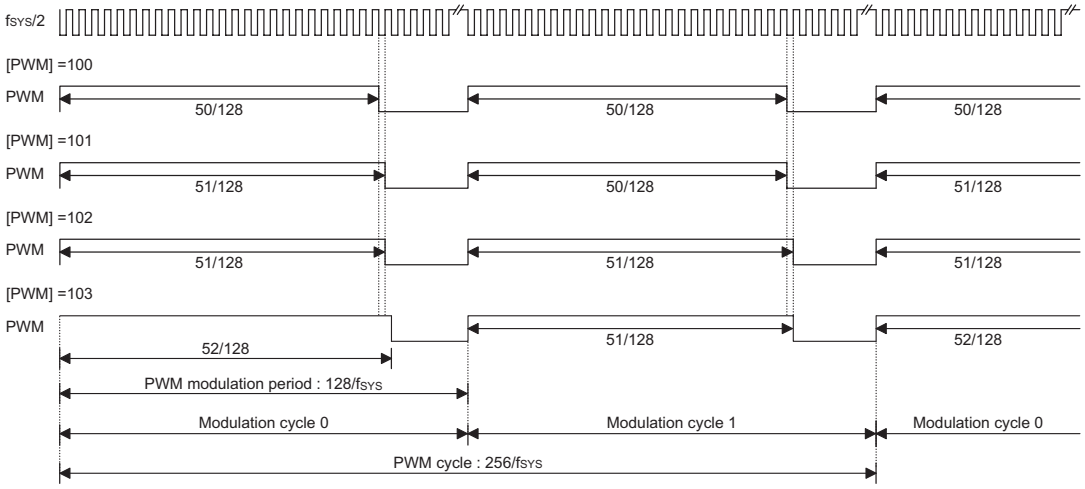
Group 2 is denoted by AC which is the value of PWM.1~PWM.0.

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

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



(6+2) PWM Mode



(7+1) PWM Mode

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

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

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

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

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

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

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

A/D Converter

This microcontroller has 8 channels and 10-bit resolution A/D (9-bit accuracy) converter. The reference voltage is VDD. The A/D converter contains 4 special registers which are; ADRL (30H), ADRH (31H), ADCR (32H) and ACSR (33H). The ADRH and ADRL are A/D result register higher-order byte and lower-order byte and are read-only. After the A/D conversion is completed, the ADRH and ADRL should be read to get the conversion result data. The ADCR is an A/D converter control register, which defines the A/D channel number, analog channel select, start A/D conversion control bit and end of A/D conversion flag. If users want to start an A/D conversion, first, define PB configuration, select the converted analog channel, and give START bit a raising edge and falling edge (0→1→0). At the end of A/D conversion, the EOCB bit is cleared and an A/D converter interrupt occurs (if the A/D converter interrupt is enabled). The ACSR is A/D clock setting register, which is used to select the A/D clock source.

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

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

ACSR (33H) Register

Bit No.	Label	Function
0 1 2	ACS0 ACS1 ACS2	Defines the analog channel select
3 4 5	PCR0 PCR1 PCR2	Defines the Port B configuration select. If PCR0, PCR1 and PCR2 are all zero, the ADC circuit is powered off to reduce power consumption
6	EOCB	Indicates end of A/D conversion. (0= end of A/D conversion) Each time bits 3~5 change state the A/D should be 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 (32H) Register

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

Analog Input Channel Selection

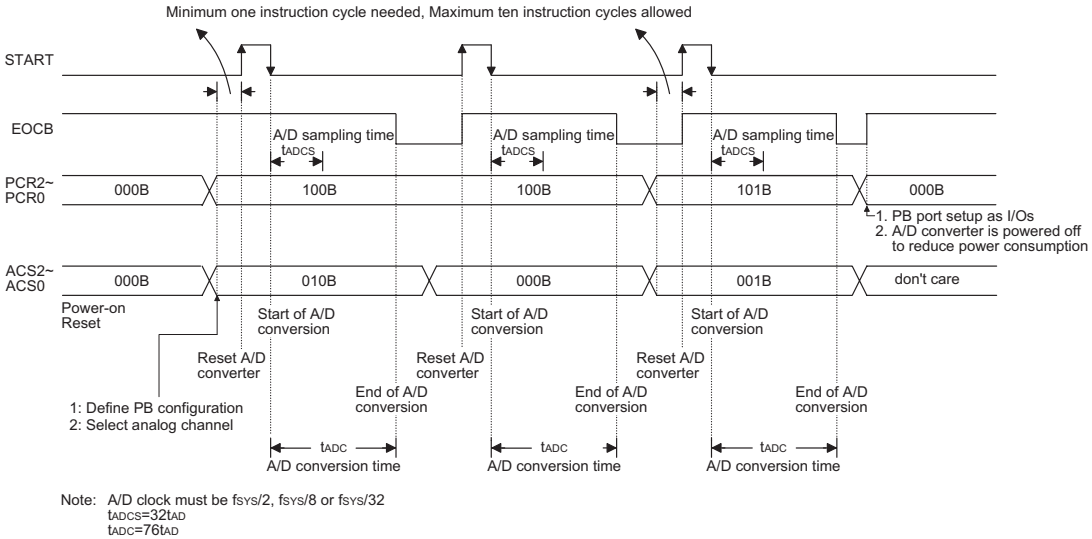
When the A/D conversion is completed, the A/D interrupt request flag is set. The EOCB bit is set to ″1″ when the START bit is set from ″0″ to ″1″.

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

Note: D0~D9 is A/D conversion result data bit LSB~MSB.

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

Port B Configuration



A/D Conversion Timing

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

Example: using EOCB Polling Method to detect end of conversion

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

Example: using Interrupt Method to detect end of conversion

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

    mov     a,00100000B         ; setup ADCR register to configure Port PB0~PB3 as A/D inputs
    mov     ADCR,a              ; and select AN0 to be connected to the A/D converter
            :
                                ; As the Port B channel bits have changed the following START
                                ; signal (0-1-0) must be issued within 10 instruction cycles
            :
Start_conversion:
    clr     START
    set     START               ; reset A/D
    clr     START               ; start A/D
    clr     ADF                 ; clear ADC interrupt request flag
    set     EADI                ; enable ADC interrupt
    set     EMI                 ; enable global interrupt
            :
            :
            :
; ADC interrupt service routine
ADC_ISR:
    mov     acc_stack,a         ; save ACC to user defined memory
    mov     a,STATUS
    mov     status_stack,a      ; save STATUS to user defined memory
            :
            :
    mov     a,ADRH              ; read conversion result high byte value from the ADRH register
    mov     adrh_buffer,a       ; save result to user defined register
    mov     a,ADRL              ; read conversion result low byte value from the ADRL register
    mov     adrl_buffer,a       ; save result to user defined register
    clr     START
    set     START               ; reset A/D
    clr     START               ; start A/D
            :
            :
```

```
EXIT_INT_ISR:
    mov     a,status_stack
    mov     STATUS,a            ; restore STATUS from user defined memory
    mov     a,acc_stack         ; restore ACC from user defined memory
    reti
```

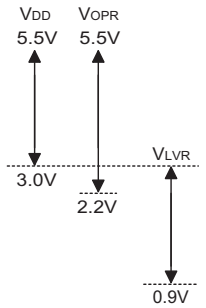
Low Voltage Reset – LVR

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

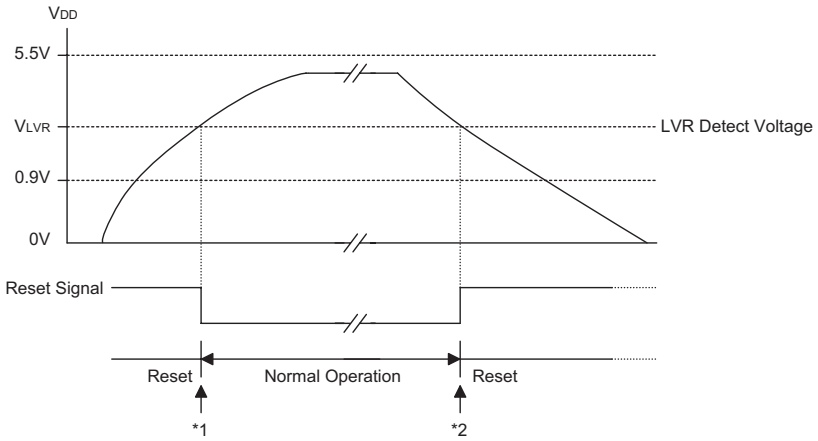
The LVR includes the following specifications:

- The low voltage range (0.9V~V_{LVR}) has to be maintained for over 1ms, otherwise, the LVR will ignore it and do not perform a reset function.
- The LVR uses the "OR" function with the external $\overline{RES}$ signal to perform a chip reset.

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



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



Low Voltage Reset

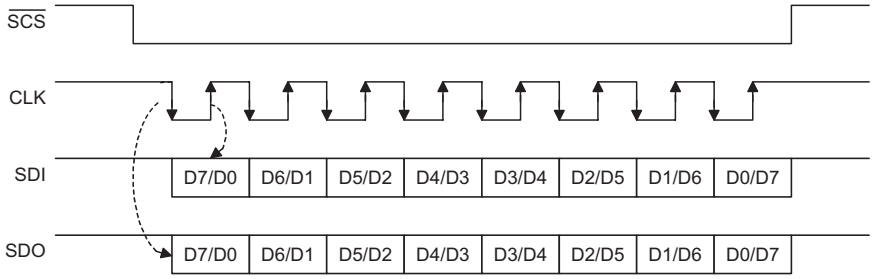
Note: *1: To make sure that the system oscillator has stabilized, the SST provides an extra delay of 1024 system clock pulses before resuming normal operation.

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

Serial Interface

Serial interface function has four basic signals included. They are SDI (serial data input), SDO (serial data output), SCK (serial clock) and $\overline{SCS}$ (slave select pin).

Note: $\overline{SCS}$ can be named $\overline{SCS}$ in the design note.



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

Note: "U" means unchanged.

Two registers (SBCR and SBDR) unique to serial interface provide control, status, and data storage.

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

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

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

Operation of Serial Interface

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

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

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

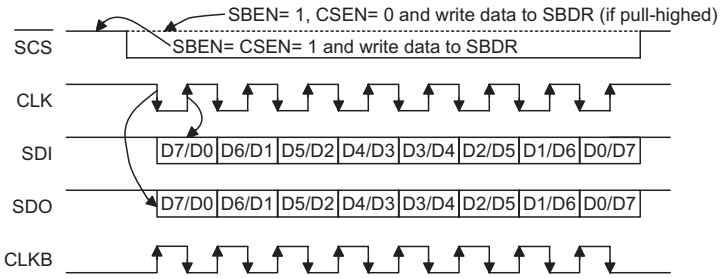
MLS: MSB or LSB first selection

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

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

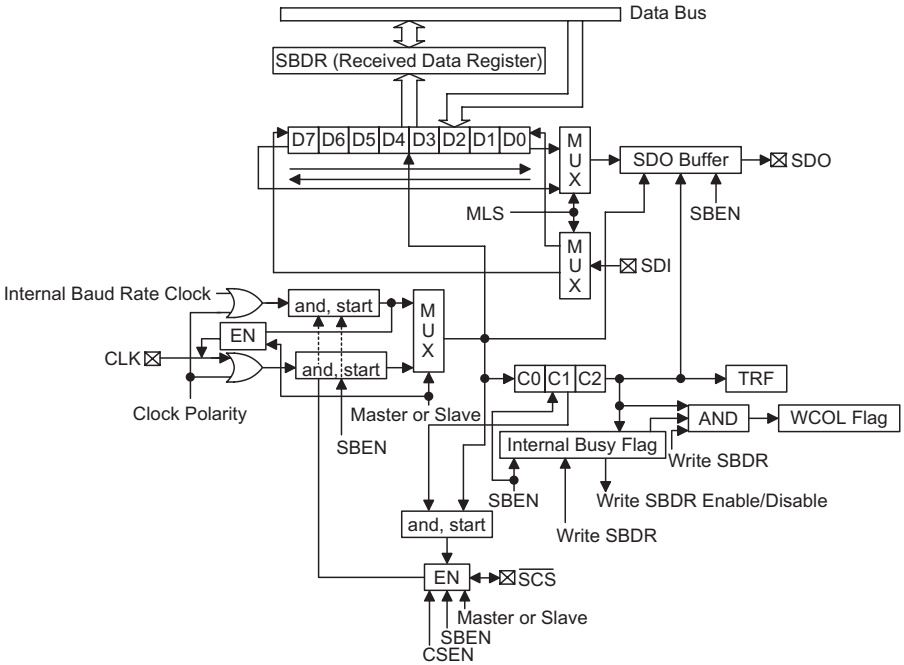
SBEN= 0 → serial bus disabled; $\overline{SCS}$= SDI= SDO= CLK= floating

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



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

Note: "u" means unchanged.



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

Suspend Wake-Up or Remote Wake-Up

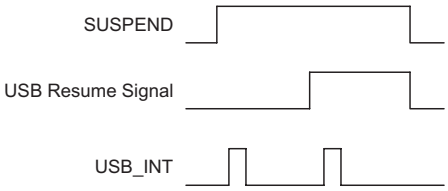
If there is no signal on the signal bus for over 3ms, the HT46RB50 will go into suspend mode. The Suspend line (bit 0 of the USC) will be set to 1 and a USB interrupt is triggered to indicate that the HT46RB50 should jump to suspend state to meet the 500μA USB suspend current spec.

In order to meet the 500μA suspend current, the firmware should disable the USB clock by clearing the USBCKEN (bit3 of the UCC) to ″0″. The suspend current is about 400μA.

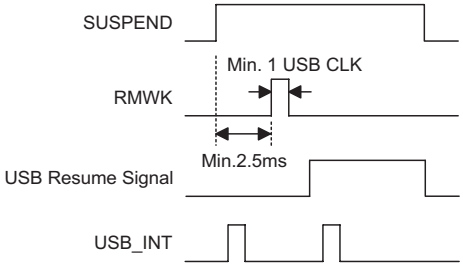
The user can also further decrease the suspend current to 250μA by setting the SUSP2 (bit4 of the UCC). But if the SUSP2 is set, user should make sure not to enable the LVR OPT option, otherwise, the HT46RB50 will be reset.

When the resume signal is sent out by the host, the HT46RB50 will wake-up the by USB interrupt and the Resume line (bit 3 of the USC) is set. In order to make the HT46RB50 work properly, the firmware must set the USBCKEN (bit 3 of the UCC) to 1 and clear the SUSP2 (bit4 of the UCC). Since the Resume signal will be cleared before the Idle signal is sent out by the host and the Suspend line (bit 0 of the USC) is going to ″0″. So when the MCU is detecting the Suspend line (bit0 of USC), the Resume line should be remembered and taken into consideration.

After finishing the resume signal, the suspend line will go inactive and a USB interrupt is triggered. The following is the timing diagram:



The device with remote wake-up function can wake-up the USB Host by sending a wake-up pulse through RMWK (bit 1 of the USC). Once the USB Host receive the wake-up signal from the HT46RB50, it will send a



Resume signal to the device. The timing is as follow:

USB Interface

The HT46RB50 has 4 Endpoints (EP0~EP3). EP0~EP2 are support Interrupt transfer, EP3 is support Bulk transfer.

There are 12 registers, including USC (20H), USR (21H), UCC (22H), AWR (address+remote wake-up 23H), STALL (24H), SIES (25H), MISC (26H), SETIO (27H), FIFO0 (28H), FIFO1 (29H), FIFO2 (2AH) and FIFO3 (2BH) used for the USB function.

The FIFO size of each FIFO is 8 byte (FIFO0), 8 byte (FIFO1), 8 byte (FIFO2) and 64 byte (FIFO3), and total of 88 bytes.

URD (bit7 of the USC) is USB reset signal control function definition bit.

Bit No.	Label	R/W	Function
0	SUSP	R	Read only, USB suspend indication. When this bit is set to ″1″ (set by SIE), it indicates that the USB bus enters the suspend mode. The USB interrupt is also triggered on any changes of this bit.
1	RMWK	R/W	USB remote wake-up command. It is set by the MCU to force the USB host leaving the suspend mode. Set RMWK bit to ″1″ to enable remote wake-up. When this bit is set to ″1″, a 2μs delay for clearing this bit to ″0″ is needed to insure that the RMWK command is accepted by the SIE.
2	URST	R/W	USB reset indication. This bit is set/cleared by USB SIE. When the URST is set to ″1″, this indicates that a USB reset has occurred and a USB interrupt will be initialized.
3	RESUME	R	USB resume indication. When the USB leaves the suspend mode, this bit is set to ″1″ (set by SIE). This bit will appear for 20ms, waiting for the MCU to detect it. When the RESUME is set by SIE, an interrupt will be generated to wake-up the MCU. In order to detect the suspend state, MCU should set the USBCKEN and SUSP2 (in the SCC register) to enable the SIE detect function. The RESUME will be cleared while the SUSP is set to ″0″. When MCU detects the SUSP, the RESUME (which causes MCU to wake-up) should be remembered and token into consideration.
4	V33C	R/W	0/1: Turn-off/on V33O output
5	PLL	R/W	0:Turn-on the PLL (default mode); 1: turn-of the PLL
6	—	—	Undefined bit, read as ″0″
7	URD	R/W	USB reset signal control function definition 1: USB reset signal will reset the MCU 0: USB reset signal cannot reset the MCU

USC (20H) Definitions

The USR (USB endpoint interrupt status register) register is used to indicate which endpoint is accessed and then selects A/D converter operation modes. The endpoint request flags (EP0IF, EP1IF, EP2IF and EP3IF) are used to indicate which endpoints are accessed. If an endpoint is accessed, the related endpoint request flag will be set to ″1″ and the USB interrupt will occur (if the USB interrupt is enabled and the stack is not full). When the active endpoint request flag is served, the endpoint request flag has to be cleared to ″0″.

Bit No.	Label	R/W	Function
0	EP0IF	R/W	When this bit is set to ″1″ (set by SIE), it indicates that the endpoint 0 is accessed and a USB interrupt will occur. When the interrupt has been served, this bit should be cleared by firmware.
1	EP1IF	R/W	When this bit is set to ″1″ (set by SIE), it indicates that the endpoint 1 is accessed and a USB interrupt will occur. When the interrupt has been served, this bit should be cleared by firmware.
2	EP2IF	R/W	When this bit is set to ″1″ (set by SIE), it indicates that the endpoint 2 is accessed and a USB interrupt will occur. When the interrupt has been served, this bit should be cleared by firmware.
3	EP3IF	R/W	When this bit is set to ″1″ (set by SIE), it indicates that the endpoint 3 is accessed and a USB interrupt will occur. When the interrupt has been served, this bit should be cleared by firmware.
4~7	—	—	Undefined bit, read as ″0″

USR (21H) Definitions

There is a system clock control register implemented to select the clock used in the MCU. This register consists of USB clock control bit (USBCKEN), second suspend mode control bit (SUSP2) and system clock selection (SYSCLK)

The following table defines which endpoint FIFO is selected, EPS2, EPS1 and EPS0.

Bit No.	Label	R/W	Function
0~2	EPS0~EPS2	R/W	Accessing endpoint FIFO selection. EPS2, EPS1, EPS0: 000: Select endpoint 0 FIFO 001: Select endpoint 1 FIFO 010: Select endpoint 2 FIFO 011: Select endpoint 3 FIFO 100: Reserved for future expansion, cannot be used 101: Reserved for future expansion, cannot be used 110: Reserved for future expansion, cannot be used 111: Reserved for future expansion, cannot be used If the selected endpoints do not exist, the related functions are not available.
3	USBCKEN	R/W	USB clock control bit. When this bit is set to ″1″, it indicates that the USB clock is enabled. Otherwise, the USB clock is turned-off.
4	SUSP2	R/W	This bit is used to reduce power consumption in suspend mode. In normal mode, clear this bit to 0 (default) In HALT mode, set this bit to 1 to reduce power consumption.
5	f_{SYS} (24MHz)	R/W	This bit is used to define the MCU system clock to come from either the external OSC or from PLL output 24MHz clock. 0: system clock comes from OSC 1: system clock comes from PLL output 24MHz
6	SYSCLK	R/W	This bit is used to specify the system clock oscillator frequency used by the MCU. If a 6MHz crystal oscillator or resonator is used, this bit should be set to ″1″. If a 12MHz crystal oscillator or resonator is used. this bit should be cleared to ″0″ (default).
7	—	—	Undefined, read as ″0″

UCC (22H) Definitions

The AWR register contains the current address and the remote wake-up function control bit. The initial value of the AWR is ″00H″. The address value extracted from the USB command is not to be loaded into this register until the SETUP stage is finished.

Bit No.	Label	R/W	Function
0	WKEN	R/W	Remote wake-up enable/disable
7~1	AD6~AD0	R/W	USB device address

AWR (23H) Definitions

The STALL register shows whether the corresponding endpoint works properly or not. As soon as the endpoint works improperly, the related bit in the STALL has to be set to ″1″. The STALL will be cleared by the USB reset signal.

Bit No.	Label	R/W	Function
3~0	STL3~STL0	R/W	Set by users when the related USB endpoints are stalled. They are cleared by USB reset and Setup Token event
7~4	—	—	Undefined bit, read as ″0″

STALL (24H) Definitions

Bit No.	Label	R/W	Function
0	ASET	R/W	This bit is used to configure the SIE to automatically change the device address with the value stored in the AWR register. When this bit is set to ″1″ by firmware, the SIE will update the device address with the value stored in the AWR register after the PC host has successfully read the data from the device by IN operation. Otherwise, when this bit is cleared to ″0″, the SIE will update the device address immediately after an address is written to the AWR register. So, in order to work properly, firmware has to clear this bit after the next valid SETUP token is received.
1	ERR	R/W	This bit is used to indicate there are some errors occurred during the FIFO0 is accessed. This bit is set by SIE and should be cleared by firmware.
2	OUT	R/W	This bit is used to indicate there are OUT token (except for the OUT zero length token) that have been received. The firmware clears this bit after the OUT data has been read. Also, this bit will be cleared by SIE after the next valid SETUP token is received.
3	IN	R	This bit is used to indicate that the current USB receiving signal from the PC host is IN token. (1=IN token; 0=Non IN token)
4	NAK	R	This bit is used to indicate that the SIE has transmitted a NAK signal to the host in response to the PC host IN or OUT token. (1=NAK signal; 0=Non NAK signal)
5	CRCF	R/W	Error condition failure flag include CRC, PID, no integrate token error, CRCF will be set by hardware and the CRCF need to be cleared by firmware.
6	EOT	R	Token Package active flag, low active.
7	NMI	R/W	NAK token interrupt mask flag. If this bit is set, when the device sent a NAK token to the host, interrupt will not occur. Otherwise, when this bit is cleared, and the device sent a NAK token to the host, it will enter the interrupt sub-routine.

SIES (25H) Definitions

MISC register combines a command and status to control the desired endpoint FIFO action and to show the status of the desired endpoint FIFO. The MISC will be cleared by USB reset signal.

Bit No.	Label	R/W	Function
0	REQUEST	R/W	After selecting the desired endpoint, FIFO can be requested by setting this bit as high active. Afterwards, this bit must be set low.
1	TX	R/W	This indicates the direction and transition end which the MCU accesses. When set as logic 1, the MCU writes data to FIFO. Afterwards, this bit must be set to logic 0 before terminating request to indicate transition end. For reading action, this bit must be set to logic 0 to indicate that the MCU wants to read and must be set to logic 1 afterwards.
2	CLEAR	R/W	This indicates an MCU clear requested FIFO, even if the FIFO is not ready. After clearing the FIFO, USB interface will send force_tx_err to tell Host that data under-run if Host want to read data.
3~4	—	R/W	Reserved bit
5	SETCMD	R/W	To show that the data in FIFO is setup command. This bit will last this state until next one entering the FIFO. (1=SETCMD token; 0=Non SETCMD token)
6	READY	R	To tell that the desired FIFO is ready to work. (1=Ready to work; 0=Non ready to work)
7	LEN0	R/W	To tell that host sent a 0-sized packet to MCU. This bit must be cleared by read action to corresponding FIFO. (1=Host sent a 0-sized packet)

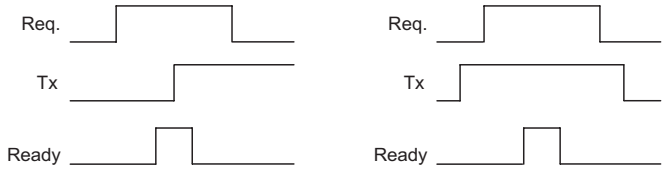
MISC (26H) Definitions

There are some timing constrains and usages illustrated here. By setting the MISC register, MCU can perform reading, writing and clearing actions. There are some examples shown in the following table for endpoint FIFO reading, writing and clearing.

Actions	MISC Setting Flow and Status
Read FIFO0 sequence	00H→01H→delay 2μs, check 41H→read* from FIFO0 register and check not ready (01H)→03H→02H
Write FIFO0 sequence	02H→03H→delay 2μs, check 43H→write* to FIFO0 register and check not ready (03H)→01H→00H
Check whether FIFO0 can be read or not	00H→01H→delay 2μs, check 41H (ready) or 01H (not ready)→00H
Check whether FIFO0 can be written or not	02H→03H→delay 2μs, check 43H (ready) or 03H (not ready)→02H
Read 0-sized packet sequence form FIFO0	00H→01H→delay 2μs, check 81H→read once (01H)→03H→02H
Write 0-sized packet sequence to FIFO0	02H→03H→delay 2μs, check 03H→07H→06H→00H

Read or Write FIFO Table

Note: *: There are 2μs existing between 2 reading action or between 2 writing action



Read FIFO Timing **Write FIFO Timing**

Bit No.	Label	R/W	Function
0	DATATG*	R/W	To toggle this bit, all the DATA token will send a DATA0 first.
1	SETIO1**	R/W	Set endpoint 1 input or output pile (1/0), default input pipe (1)
2	SETIO2**	R/W	Set endpoint 2 input or output pile (1/0), default input pipe (1)
3	SETIO3**	R/W	Set endpoint 3 input or output pile (1/0), default input pipe (1)
4~7	—	—	Undefined bit, read as ″0″

SETIO Register (27H), USB Endpoint 1~Endpoint5 Set IN/OUT Pipe Register

Note: *USB definition: when the host sends a ″set Configuration″, the Data pipe should send the DATA0 (Data toggle) first. So, when the device receives a ″set configuration″ setup command, user needs to toggle this bit so the next data will send a Data0 first.

**Needs to set the data pipe as an input pile or output pile. The purpose of this function is to avoid the host from abnormally sending only an IN or OUT token and disables the endpoint.

Options

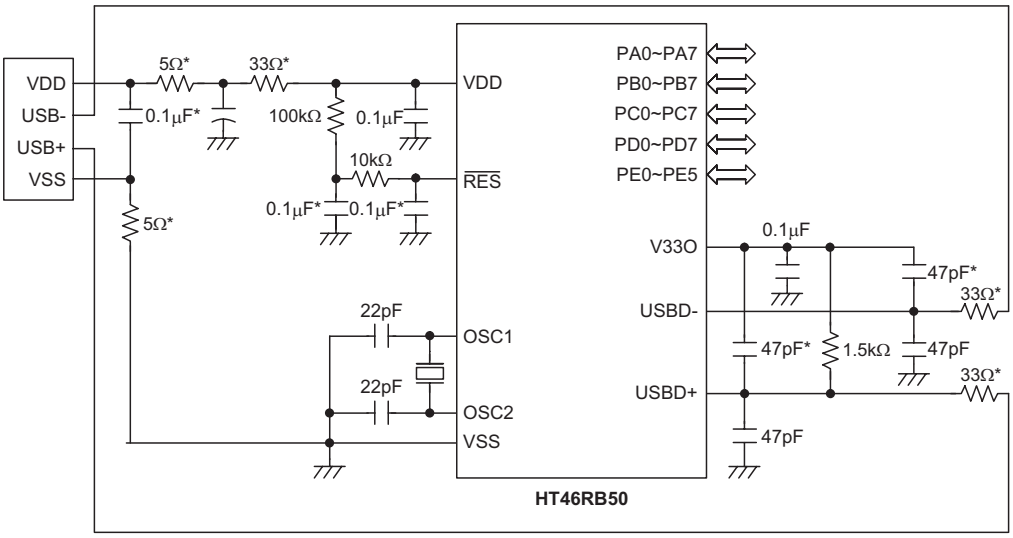
The following table shows all kinds of options in the microcontroller. All of the OTP options must be defined to ensure a proper functioning system.

No.	Option
1	PA0~PA7 pull-high resistor enable or disable (by bit)
2	PB0~PB7 pull down resistor enable or disable (by bit)
3	PC0~PC7 pull-high resistor enable or disable (by nibble)
4	PD0~PD7 pull-high resistor enable or disable (by nibble)
5	PE0~PE5 pull-high resistor enable or disable (by nibble)
6	LVR enable or disable

No.	Option
7	PWM selection: (7+1) or (6+2) mode PD0: level output or PWM0 output PD1: level output or PWM1 output
8	SIO (Serial Interface) enable/disable (if SIO is enabled then PE0~PE3 I/O port will be disabled)
9	SIO_ CPOL: Clock polarity 1/0: clock polarity rising or falling edge
10	SIO_WCOL: Enable/disable
11	SIO_CSEN: Enable/disable, CSEN mask option is used to enable/disable (1/0) software CSEN function
12	WDT enable/disable
13	WDT clock source: $f_{SYS}/4$ or WDTOSC
14	WDT timeout period: $2^{12}/f_S$~$2^{13}/f_S$, $2^{13}/f_S$~$2^{14}/f_S$, $2^{14}/f_S$~$2^{15}/f_S$, $2^{15}/f_S$~$2^{16}/f_S$
15	″CLRWDT″ instruction (s): 1 or 2
16	PA0~PA7 wake-up enable/disable (by bit)
17*	EP1~EP3 Data pipe enable: EP1, EP2, EP3 enable/disable. (Default is enable)

Note: *: The purpose of this option is to enable the endpoint that will be used, and disable the endpoint that will not be used.

Application Circuits



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

X1 can use 6MHz or 12MHz, X1 as close to OSC1 and OSC2 as possible

Components with ″*″ are used for EMC issue

22pF capacitance are used for resonator only

Instruction Set Summary

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

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

Note: x: Immediate data

m: Data memory address

A: Accumulator

i: 0~7 number of bits

addr: Program memory address

√: Flag is affected

–: Flag is not affected

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

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

(3): (1) and (2)

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

Instruction Definition

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

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

Operation ACC ← ACC+[m]+C

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	√	√	√	√

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

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

Operation [m] ← ACC+[m]+C

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	√	√	√	√

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

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

Operation ACC ← ACC+[m]

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	√	√	√	√

ADD A,x Add immediate data to the accumulator

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

Operation ACC ← ACC+x

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	√	√	√	√

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

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

Operation [m] ← ACC+[m]

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	√	√	√	√

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

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

Operation ACC ← ACC ″AND″ [m]

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

AND A,x Logical AND immediate data to the accumulator

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

Operation ACC ← ACC ″AND″ x

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

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

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

Operation [m] ← ACC ″AND″ [m]

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

CALL addr Subroutine call

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

Operation Stack ← Program Counter+1
Program Counter ← addr

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

CLR [m] Clear data memory

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

Operation [m] ← 00H

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

CLR [m].i Clear bit of data memory

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

Operation [m].i ← 0

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

CLR WDT Clear Watchdog Timer

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

Operation WDT ← 00H
PDF and TO ← 0

Affected flag(s)

TO	PDF	OV	Z	AC	C
0	0	—	—	—	—

CLR WDT1 Preclear Watchdog Timer

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

Operation WDT ← 00H*
PDF and TO ← 0*

Affected flag(s)

TO	PDF	OV	Z	AC	C
0*	0*	—	—	—	—

CLR WDT2 Preclear Watchdog Timer

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

Operation WDT ← 00H*
PDF and TO ← 0*

Affected flag(s)

TO	PDF	OV	Z	AC	C
0*	0*	—	—	—	—

CPL [m] Complement data memory

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

Operation [m] ← $[\overline{m}]$

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

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

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

Operation ACC ← $[\overline{m}]$

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

DAA [m] Decimal-Adjust accumulator for addition

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

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	√

DEC [m] Decrement data memory

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

Operation [m] ← [m]–1

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

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

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

Operation ACC ← [m]–1

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

HALT Enter power down mode

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

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
0	1	—	—	—	—

INC [m] Increment data memory

Description Data in the specified data memory is incremented by 1

Operation [m] ← [m]+1

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

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

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

Operation ACC ← [m]+1

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

JMP addr Directly jump

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

Operation Program Counter ←addr

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

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

Operation ACC ← [m]

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

MOV A,x Move immediate data to the accumulator

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

Operation ACC ← x

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

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

Operation [m] ←ACC

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

NOP No operation

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

Operation Program Counter ← Program Counter+1

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

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

Operation ACC ← ACC ″OR″ [m]

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

OR A,x Logical OR immediate data to the accumulator

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

Operation ACC ← ACC ″OR″ x

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

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

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

Operation [m] ←ACC ″OR″ [m]

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

RET Return from subroutine

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

Operation Program Counter ← Stack

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

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

Operation Program Counter ← Stack
ACC ← x

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

RETI Return from interrupt

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

Operation Program Counter ← Stack
EMI ← 1

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

RL [m] Rotate data memory left

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

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

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

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

RLC [m] Rotate data memory left through carry

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

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	√

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

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

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	√

RR [m] Rotate data memory right

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

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

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

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

RRC [m] Rotate data memory right through carry

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

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	√

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

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

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	√

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

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

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	√	√	√	√

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

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

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	√	√	√	√

SDZ [m] Skip if decrement data memory is 0

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

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

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

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

SET [m] Set data memory

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

Operation [m] ← FFH

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

SET [m]. i Set bit of data memory

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

Operation [m].i ← 1

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

SIZ [m] Skip if increment data memory is 0

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

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

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

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

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

Operation Skip if [m].i≠0

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

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

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	√	√	√	√

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

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

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	√	√	√	√

SUB A,x Subtract immediate data from the accumulator

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

Operation ACC ← ACC+$\overline{x}$+1

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	√	√	√	√

SWAP [m] Swap nibbles within the data memory

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

Operation [m].3~[m].0 ↔ [m].7~[m].4

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

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

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

SZ [m] Skip if data memory is 0

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

Operation Skip if [m]=0

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

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

Operation Skip if [m]=0

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

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

Operation Skip if [m].i=0

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

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

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

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

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

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

Operation ACC ← ACC ″XOR″ [m]

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

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

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

Operation [m] ← ACC ″XOR″ [m]

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

XOR A,x Logical XOR immediate data to the accumulator

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

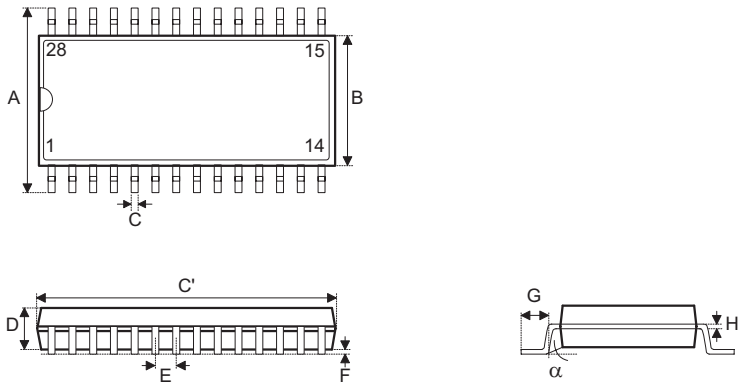
Operation ACC ← ACC ″XOR″ x

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

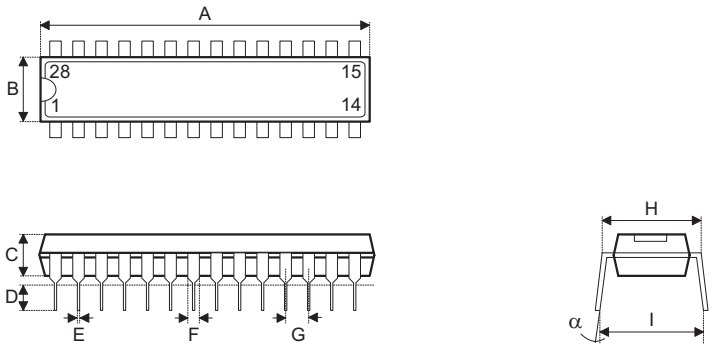
Package Information

28-pin SOP (300mil) Outline Dimensions



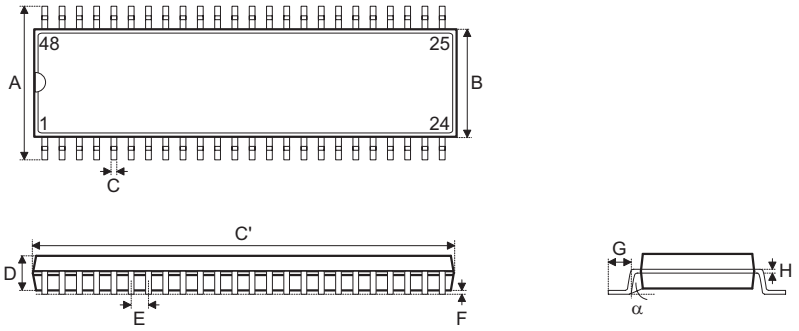
Symbol	Dimensions in mil		
	Min.	Nom.	Max.
A	394	—	419
B	290	—	300
C	14	—	20
C′	697	—	713
D	92	—	104
E	—	50	—
F	4	—	—
G	32	—	38
H	4	—	12
α	0°	—	10°

28-pin 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°

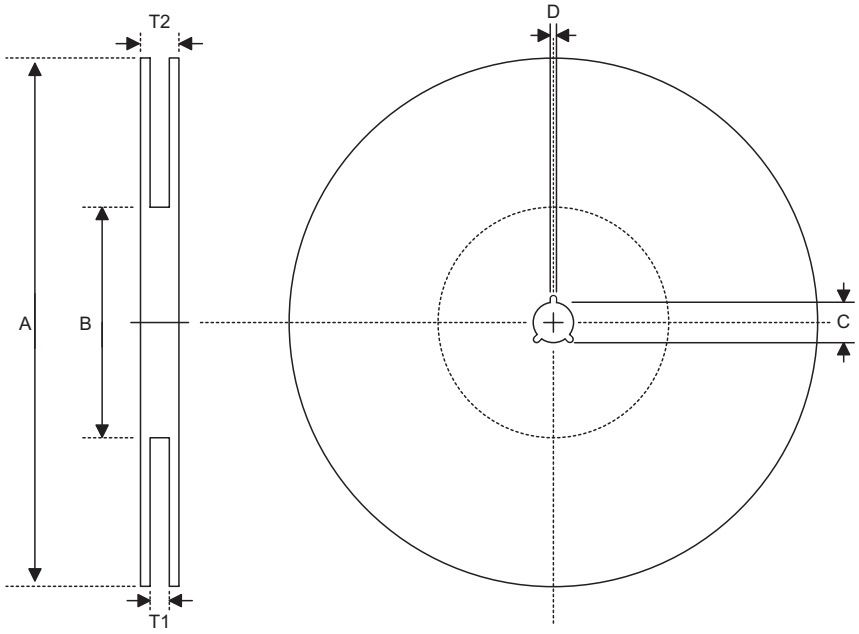
48-pin SSOP (300mil) Outline Dimensions



Symbol	Dimensions in mil		
	Min.	Nom.	Max.
A	395	—	420
B	291	—	299
C	8	—	12
C′	613	—	637
D	85	—	99
E	—	25	—
F	4	—	10
G	25	—	35
H	4	—	12
α	0°	—	8°

Product Tape and Reel Specifications

Reel Dimensions



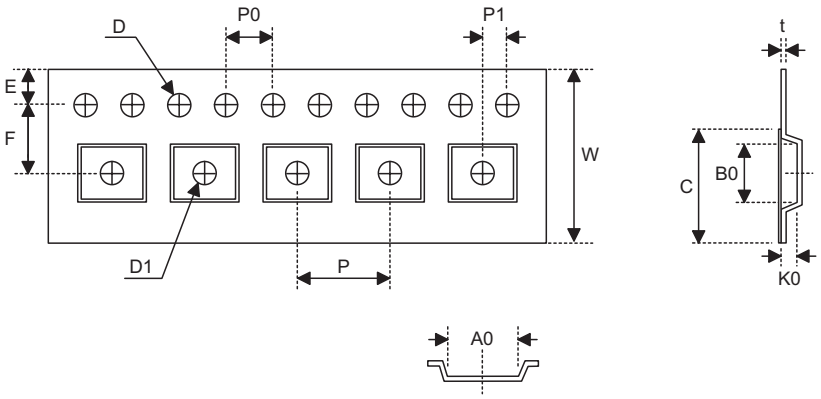
SOP 28W (300mil)

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

SSOP 48W

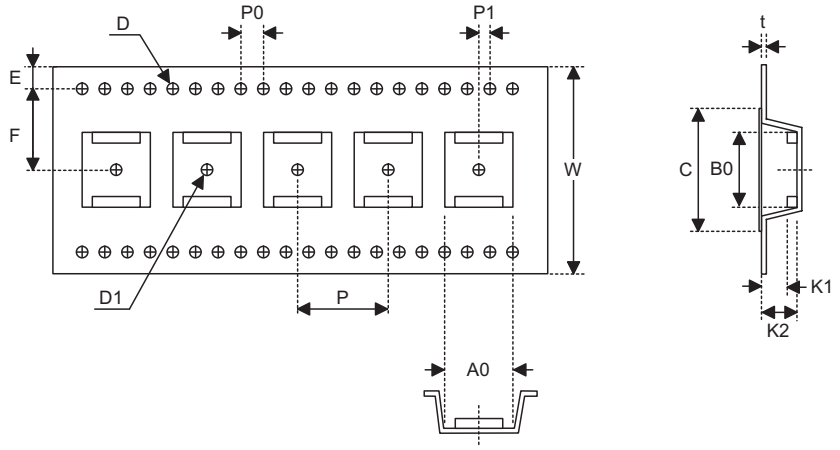
Symbol	Description	Dimensions in mm
A	Reel Outer Diameter	330±1
B	Reel Inner Diameter	100±0.1
C	Spindle Hole Diameter	13+0.5 −0.2
D	Key Slit Width	2±0.5
T1	Space Between Flange	32.2+0.3 −0.2
T2	Reel Thickness	38.2±0.2

Carrier Tape Dimensions



SOP 28W (300mil)

Symbol	Description	Dimensions in mm
W	Carrier Tape Width	24±0.3
P	Cavity Pitch	12±0.1
E	Perforation Position	1.75±0.1
F	Cavity to Perforation (Width Direction)	11.5±0.1
D	Perforation Diameter	1.5+0.1
D1	Cavity Hole Diameter	1.5+0.25
P0	Perforation Pitch	4±0.1
P1	Cavity to Perforation (Length Direction)	2±0.1
A0	Cavity Length	10.85±0.1
B0	Cavity Width	18.34±0.1
K0	Cavity Depth	2.97±0.1
t	Carrier Tape Thickness	0.35±0.01
C	Cover Tape Width	21.3



SSOP 48W

Symbol	Description	Dimensions in mm
W	Carrier Tape Width	32±0.3
P	Cavity Pitch	16±0.1
E	Perforation Position	1.75±0.1
F	Cavity to Perforation (Width Direction)	14.2±0.1
D	Perforation Diameter	2 Min.
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	12±0.1
B0	Cavity Width	16.2±0.1
K1	Cavity Depth	2.4±0.1
K2	Cavity Depth	3.2±0.1
t	Carrier Tape Thickness	0.35±0.05
C	Cover Tape Width	25.5

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

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

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

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

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

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

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

Copyright © 2006 by HOLTEK SEMICONDUCTOR INC.

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