Remote Type 8-bit MCU

HT48RA0-6

Revision: V1.20 Date: March 24, 2016

www.holtek.com

Table of Contents

Features

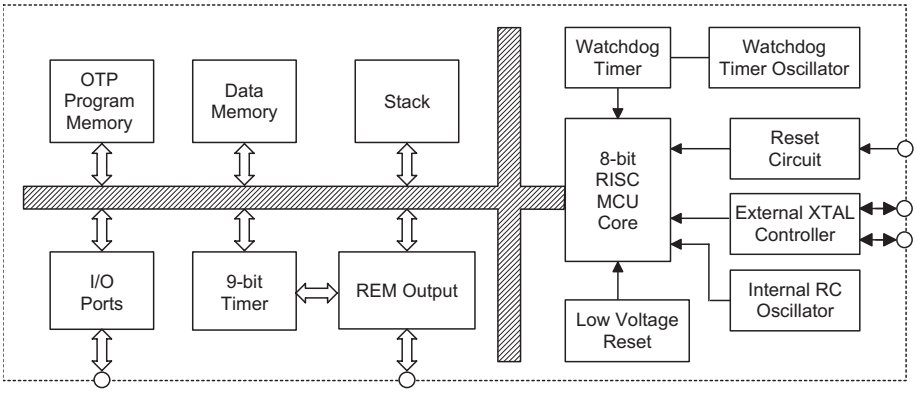
- Operating voltage:
  – f_{SYS}=4MHz at V_{DD}=2.0V~3.6V (LVR enabled)
  – f_{SYS}=4MHz at V_{DD}=1.8V~3.6V (LVR disabled)
- Oscillator types:
  – External high frequency Crystal – HXT
  – Internal high frequency RC – HIRC
- 1K×14 program memory
- 32×8 data memory
- One-level subroutine nesting
- 16 bidirectional I/O lines
- Fully integrated internal 4095kHz oscillator requires no external components
- One programmable carrier output - using 9-bit timer
- Carrier output pin (REM/REMDRV)
- Build-in IR Driver (330mA@3.0V)
- Watchdog Timer
- Low voltage reset function
- Power-down and wake-up features reduce power consumption
- 14-bit table read instructions
- Up to 1μs instruction cycle with 4MHz system clock
- 63 powerful instructions
- All instructions executed in 1 or 2 machine cycles
- Bit manipulation instructions
- 16-pin NSOP and 20-pin SSOP packages
- HT48RA0-6B use in internal IR driver

General Description

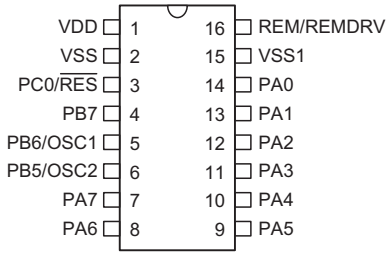
The HT48RA0-6 is 8-bit high performance, RISC architecture microcontroller device specifically designed for multiple I/O control product applications.

The advantages of low power consumption, I/O flexibility, timer functions, watchdog timer, HALT and wake-up functions, as well as low cost, enhance the versatility of this device to suit a wide range of application possibilities such as industrial control, consumer products, and particularly suitable for use in products such as infrared remote controllers and various subsystem controllers.

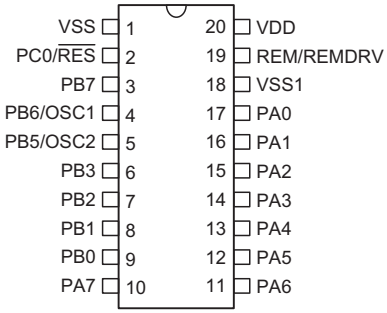
Block Diagram



Pin Assignment



HT48RA0-6B
16 NSOP-B



HT48RA0-6B
20 SSOP-B

Pin Description

Pin Name	I/O	Configuration Option	Description
PA0~PA7	I/O	—	Bidirectional 8-bit input/output port with pull-high resistors. Software instructions determine if the pin is an NMOS output or Schmitt Trigger input. Each pin can have a wake-up function if configured as an input pin.
PB0~PB3 PB5/OSC2 PB6/OSC1 PB7	I/O	OSC	Bidirectional 7-bit input/output port with pull-high resistors. Software instructions determine if the pin is an NMOS output or Schmitt Trigger input. Each pin can have a wake-up function if configured as an input pin. PB5 and PB6 are pin-shared with the external crystal pins named OSC2 and OSC1 respectively determined by a configuration option.
PC0/$\overline{RES}$	ST	$\overline{RES}$	Bidirectional 1-bit input/output port without a pull-high resistor. Software instructions determine if the pin is an NMOS output or Schmitt Trigger input. This pin has the capability of wake-up when it is configured as an input pin. PC0 is pin-shared with the external reset pin named $\overline{RES}$ determined by a configuration option.
REM/REMDRV	O	REM/REMDRV	Carrier output dual function pin. REM is a CMOS carrier output pin with an initial low level after a reset. REMDRV pin is a high sink current NMOS open drain carrier output pin which will be in a floating condition after a reset. The selection of REM or REMDRV is determined by a configuration option.
VDD	—	—	Positive power supply
VSS	—	—	Negative power supply, ground
VSS1	—	—	Negative power supply, ground of REM/REMDRV pin.

Absolute Maximum Ratings

Supply Voltage ..... V_{SS}−0.3V to V_{SS}+4.0V
Input Voltage ..... V_{SS}−0.3V to V_{DD}+0.3V
Storage Temperature..... -50˚C to 125˚C
Operating Temperature..... -20˚C to 70˚C
I_{OH} Total ..... -100mA
I_{OL} Total 150mA
Total Power Dissipation 500mW

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

D.C. Characteristics

Ta= 25˚C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	Operating Voltage	—	LVR disable	1.8	—	3.6	V
			LVR enable	2.0	—	3.6	V
I_{DD}	Operating Current	3V	No load, f_{SYS}=4MHz	—	0.7	1.5	mA
I_{STB1}	Standby Current	3V	No load, system HALT, WDT disable	—	0.1	1.0	μA
I_{STB2}	Standby Current	3V	No load, system HALT, WDT enable	—	—	5.0	μA
V_{IL1}	Input Low Voltage for I/O Ports	—	—	0	—	$0.2V_{DD}$	V
V_{IH1}	Input Low Voltage for I/O Ports	—	—	$0.8V_{DD}$	—	V_{DD}	V
V_{IL2}	Input Low Voltage($\overline{RES}$)	—	—	0	—	$0.4V_{DD}$	V
V_{IH2}	Input Low Voltage($\overline{RES}$)	—	—	$0.9V_{DD}$	—	V_{DD}	V
I_{OH}	REM Output Source Current	3V	$V_{OH}=0.9V_{DD}$	-5	-7	—	mA
I_{OL1}	PA, PB, REM Output Sink Current	3V	$V_{OL}=0.1V_{DD}$	6	12	—	mA
I_{OL2}	PC0 Sink Current	3V	$V_{OL}=0.1V_{DD}$	0.8	1.2	—	mA
I_{OL3}	REMDRV Output Sink Current	3V	$V_{OL}=0.6V_{DD}$	300	330	—	mA
R_{PH}	Pull-high Resistance of Port A,Port B	3V	—	100	150	200	kΩ
V_{LVR}	Low Voltage Reset Voltage	—	—	1.8	1.9	2.0	V

A.C. Characteristics

Ta= 25˚C

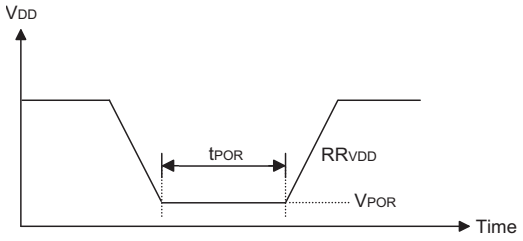
Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Condition				
f_{SYS}	System Clock	1.8V~3.6V	Ta= -20°C~70°C	400	—	4000	kHz
f_{HIRC}	System Clock (HIRC)	2.2V~3.6V	Ta= -10°C~50°C	-1%	4095	+1%	kHz
t_{SST}	System Start-up Timer Period	—	Power-up or wake-up from HALT	—	1024	—	t_{SYS}
t_{WDTOSC}	Watchdog Oscillator	3V	—	45	90	180	μs
t_{LVR}	Low Voltage Width to Reset	—	—	0.25	1.00	2.00	ms

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

Power-on Reset Characteristics

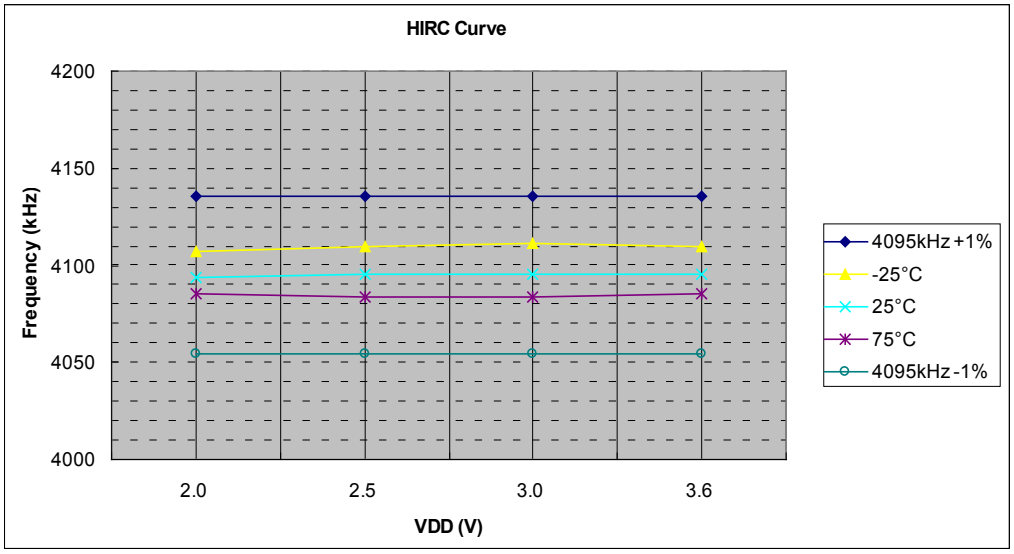
Ta= 25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{POR}	V_{DD} Start Voltage to EnsurePower-on Reset	—	—	—	—	100	mV
RR_{VDD}	V_{DD} Raising Rate to Ensure Power-on Reset	—	—	0.035	—	—	V/ms
t_{POR}	Minimum Time for V_{DD} Stays at V_{POR} to ensure Power-on Reset	—	—	1	—	—	ms

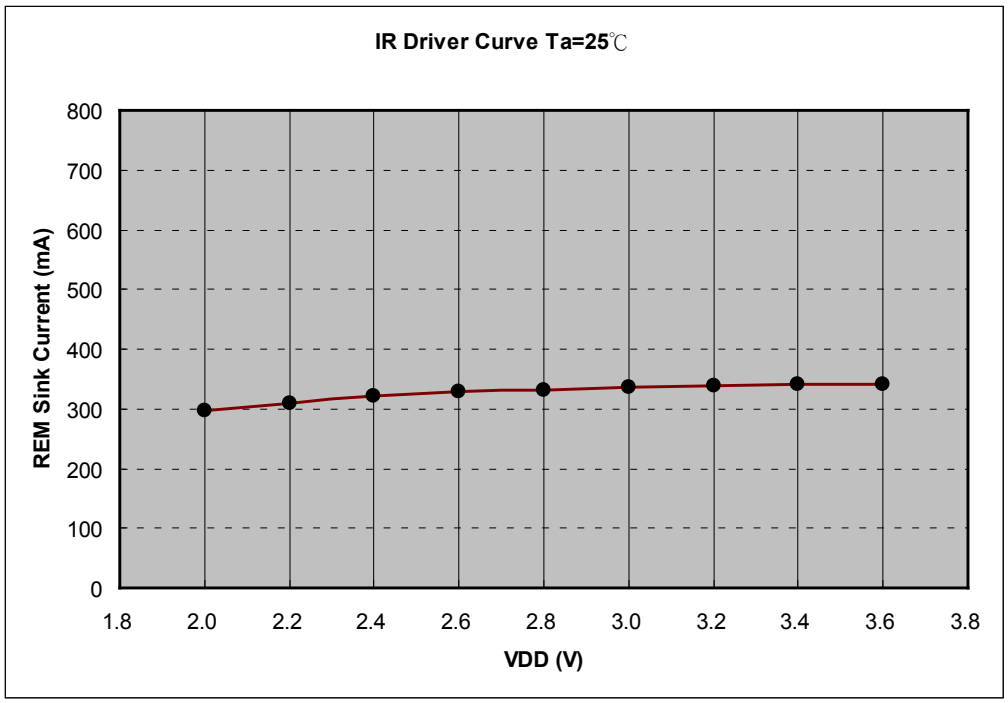


Characteristics Curves

HIRC Oscillator Voltage/Temperature vs. Frequency



IR Sink Current vs. VDD



Functional Description

Execution Flow

The main system clock is derived from either an external crystal oscillator which requires the connection of the external crystal or resonator or an internal RC oscillator which requires no external component for its operation. 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. However, the pipelining scheme causes each instruction to effectively execute within one cycle. If an instruction changes the program counter, two cycles are required to complete the instruction.

Program Counter – PC

The 10-bit program counter (PC) controls the sequence in which the instructions stored in program memory are executed and its contents specify a maximum of 1024 addresses.

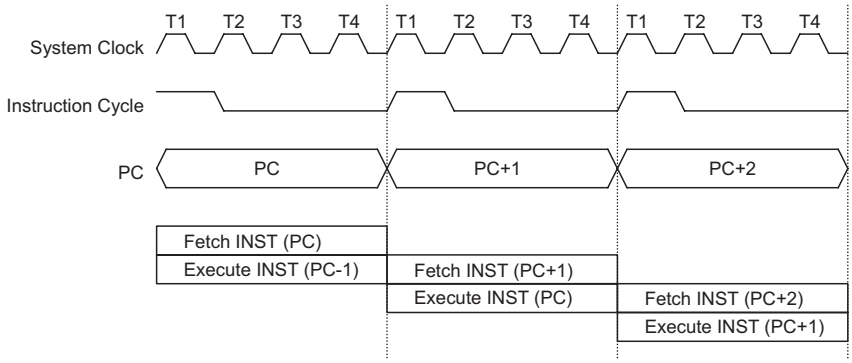
After accessing a program memory word to fetch an instruction code, the contents of the program counter are incremented by one. The program counter then points to the memory word containing the next instruction code.

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

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

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

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



Execution Flow

Mode	Program Counter									
	*9	*8	*7	*6	*5	*4	*3	*2	*1	*0
Initial reset	0	0	0	0	0	0	0	0	0	0
Skip	Program Counter + 2									
Loading PCL	*9	*8	@7	@6	@5	@4	@3	@2	@1	@0
Jump, call branch	#9	#8	#7	#6	#5	#4	#3	#2	#1	#0
Return from subroutine	S9	S8	S7	S6	S5	S4	S3	S2	S1	S0

Program Counter

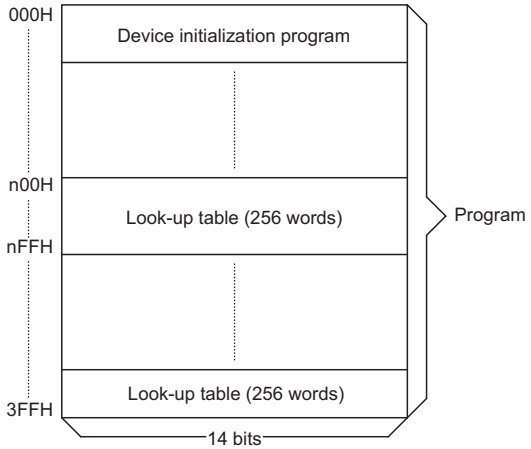
Note: *9~*0: Program counter bits
S9~S0: Stack register bits
#9~#0: Instruction code bits
@7~@0: PCL bits

Program Memory - ROM

The program memory is used to store the program instructions which are to be executed. It also contains data and table and is organized into 1024×14 bits, addressed by the program counter and table pointer.

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

- Location 000H
  This area is reserved for the initialization program. After a device reset, the program always begins execution at location 000H.
- Table location
  Any location in the Program Memory space can be used as a look-up table. The instructions TABRDC [m] (the current page, one page=256 words) and TABRDL [m] (the last page) transfer the contents of the lower-order byte to the specified data memory register, and the higher-order byte to TBLH (08H). Only the destination of the lower-order byte in the table is well-defined, the other bits of the table word are transferred to the lower portion of TBLH, the remaining 2 bits are read as "0". The Table Higher-order byte register (TBLH) is read only. The table pointer (TBLP) is a read/write register (07H), where P indicates the table location. Before accessing the table, the location must be placed in TBLP. The TBLH is read only and cannot be restored. All table related instructions need 2 cycles to complete the operation. These areas may function as normal program memory depending upon the requirements.



Note: n ranges from 0 to 3

Program Memory

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

Table Location

Note: *9~*0: Table location bits
PC9~PC8: Current Program Counter bits
@7~@0: Table Pointer bits

Stack

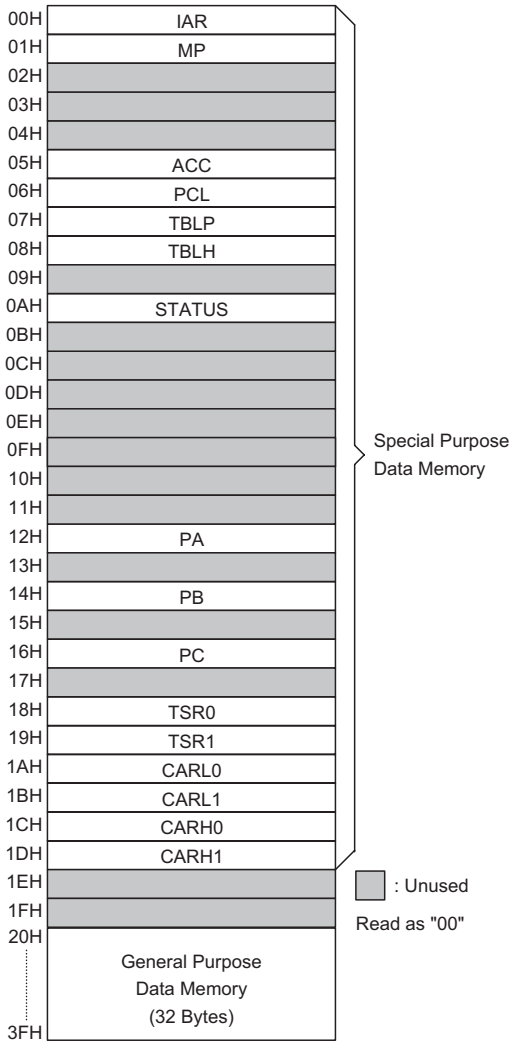
This is a special part of the memory which is used to save the contents of the Program Counter only. The stack is organized into one level and neither part of the data nor part of the program space, and is neither readable nor writeable. The activated level is indexed by the Stack Pointer, and is neither readable nor writeable. At a subroutine call or interrupt acknowledge signal, the contents of the Program Counter are pushed onto the stack. At the end of a subroutine or an interrupt routine, signaled by a return instruction, RET or RETI, the Program Counter is restored to its previous value from the stack. After a device reset, the Stack Pointer will point to the top of the stack.

If the stack is full and a "CALL" is subsequently executed, stack overflow occurs and the first entry will be lost and only the most recent return address is stored.

Data Memory – RAM

The data memory is divided into two functional groups: special function registers and general purpose data memory (32×8). Most are read/write, but some are read only.

The remaining space before the 20H is reserved for future expanded usage and reading these locations will return the result 00H. The general purpose data memory, addressed from 20H to 3FH, is used for data and control information under instruction command. All data memory areas can handle arithmetic, logic, increment, decrement and rotate operations directly. Except for some dedicated bits, each bit in the data memory can be set and reset by the SET [m].i and CLR [m].i instructions, respectively. They are also indirectly accessible through memory pointer register (MP;01H).



RAM Mapping

Indirect Addressing Register

Location 00H is an indirect addressing register that is not physically implemented. Any read/write operation to [00H] accesses the data memory pointed to by MP (01H). Reading location 00H itself indirectly will return the result 00H. Writing indirectly results in no operation.

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

Accumulator

The accumulator closely relates to ALU operations. It is also mapped to location 05H of the data memory and is capable of carrying out immediate data operations. Data movement between two data memory locations has to pass through the accumulator.

Arithmetic and Logic Unit – ALU

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

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

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

Status Register – STATUS

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

With the exception of the TO and PDF flags, the other status register bits can be altered by instructions like most other register. Any data written into the status register will not change the TO or PDF flags. In addition it should be noted that operations related to the status register may give different results from those intended. The TO and PDF flags can only be changed by the Watchdog Timer overflow, device power-up, clearing the Watchdog Timer and executing the HALT instruction.

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

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

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

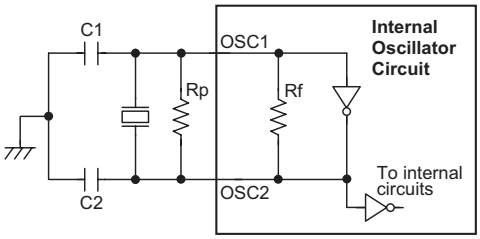
Oscillator Configuration

In this device there are two methods of generating the system clock, one external crystal oscillator and one internal RC oscillator.

External Crystal/Ceramic Oscillator – HXT

The External Crystal/Ceramic System Oscillator is one of the system oscillator choices, which is selected via a configuration option. For the crystal oscillator configuration, the connection of a crystal across OSC1 and OSC2 will create the necessary phase shift and feedback for oscillation. Two external capacitors may be required to be connected as shown. However, the feedback resistor named R_f shown in the following diagram for the crystal oscillator to oscillate properly can be selected as either an internally or externally connected type via a configuration option. When the external connection type of the feedback resistor is selected, the recommended value of the external connected feedback resistor ranges from 300kΩ to 500kΩ. Using a ceramic resonator will usually require two small value capacitors, C1 and C2, to be connected as shown for oscillation to occur. The values of C1 and C2 should be selected in consultation with the crystal or resonator manufacturer's specification.

For oscillator stability and to minimise the effects of noise and crosstalk, it is important to ensure thatthe crystal and any associated resistors andcapacitors along with interconnectinglines are all located as close to the MCUas possible.



Note: 1. Rp is normally not required. C1 and C2 are required.
2. Although not shown OSC1/OSC2 pins have a parasitic capacitance of around 7pF.

Crystal/Resonator Oscillator – HXT

Crystal Oscillator C1 and C2 Values		
Crystal Frequency	**C1**	**C2**
4MHz	8pF	10pF
Note: C1 and C2 values are for guidance only.		

Crystal Recommended Capacitor Values

Internal RC Oscillator – HIRC

The internal RC oscillator is a fully integrated system oscillator requiring no external components. The internal RC oscillator has a fixed frequency of 4095kHz. Device trimming during the manufacturing process and the inclusion of internal frequency compensation circuits are used to ensure that the influence of the power supply voltage, temperature and process variations on the oscillation frequency are minimised. As a result, at a power supply ranging from 2.2V to 3.6V and in a temperature range from -10°C to 50°C degrees, the fixed oscillation frequency of 4095kHz will have a tolerance within 1%. Note that if this internal system clock option is selected, as it requires no external pins for its operation, I/O pins PB5 and PB6 are free for use as normal I/O pins.

Watchdog Timer – WDT

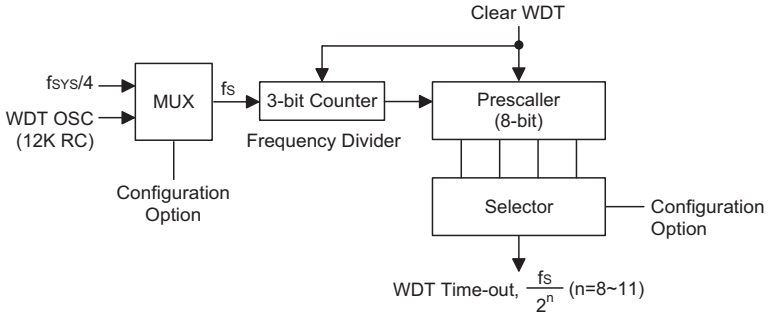
The WDT clock source is implemented by the instruction clock which is the system clock divided by 4 or the internal RC oscillator with the frequency of 12kHz. The clock source is processed by a frequency divider and a prescaler to provide various time out periods.

$$\text{WDT time out period} = \frac{\text{Clock Source}}{2^n}$$

Where n= 8~11 selected by a configuration option.

The WDT 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 configuration option. If the Watchdog Timer is disabled, all the executions related to the WDT result in no operation and the WDT will lose its protection purpose. In this situation the logic can only be restarted by external logic.

A WDT overflow under normal operation will initialise a "chip reset" and set the status bit "TO". To clear the contents of the WDT prescaler, two methods are adopted, software instructions or a HALT instruction. There are two types of software instructions. One type is the single instruction "CLR WDT", the other type comprises two instructions, "CLR WDT1" and "CLR WDT2". Of these two types of instructions, only one can be active depending on the configuration option "CLR WDT times selection option". If the "CLR WDT" is selected (i.e.. CLR WDT times equal one), any execution of the CLR WDT instruction will clear the WDT. In case "CLR WDT1" and "CLR WDT2" are chosen (i.e. CLR WDT times equal two), these two instructions must be executed to clear the WDT; otherwise, the WDT may reset the chip due to a time-out.



Power Down Operation – HALT

The Power-down mode is initialised by the HALT instruction and results in the following:

- The system oscillator will be turned off but the WDT oscillator remains running (if the WDT oscillator is selected).
- The contents of the on-chip Data Memory and registers remain unchanged.
- WDT prescaler is cleared.
- All 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 by means of an external falling edge signal on all I/O ports. By examining the TO and PDF flags, the reason for chip reset can be determined. The PDF flag is cleared when the system powers up or when a CLR WDT instruction is executed and is set when the HALT instruction is executed. The TO flag is set if the WDT time-out occurs during normal operation.

The I/O ports wake-up can be considered as a continuation of normal execution. Each bit in I/O port can be independently selected to wake up the device by the code option. Awakening from an I/O port stimulus, the program will resume execution of the next instruction.

Once a wake-up event(s) occurs, it takes 1024 t_{SYS} (system clock periods) to resume normal operation. In other words, a dummy cycle period will be inserted after the wake-up.

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

Reset

There are several ways in which a reset can occur:

- Power On reset
- $\overline{RES}$ pin reset
- Low Voltage reset
- WDT time-out reset during normal operation

Some registers remain unchanged during reset conditions. Most registers are reset to the "initial condition" when the reset conditions are met. By examining the PDF and TO flags, the program can distinguish between different chip resets.

TO	PDF	RESET Conditions
0	0	Power-on reset
u	u	$\overline{RES}$ or LVR reset during NORMAL operation
1	u	WDT time-out reset during NORMAL operation
1	1	WDT time-out reset during power-down operation

Note: "u" stands for unchanged

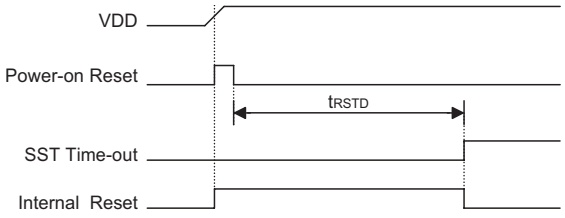
To guarantee that the system oscillator has started and stabilized, the SST (System Start-up Timer) provides an extra-delay of 1024 system clock pulses when the system powers up or when the system awakes from a HALT state.

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

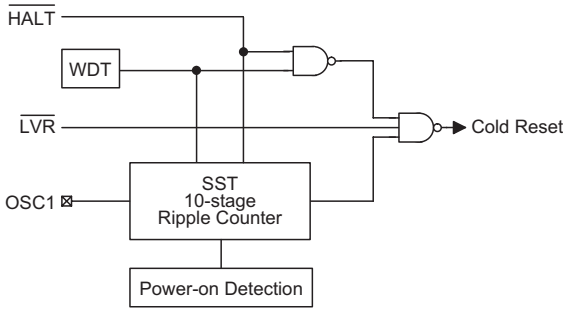
The functional unit chip reset status is shown below.

Program Counter	000H
WDT Prescaler	Clear
Timer/Event Counter	Timer Counter will be turned off
Input/Output Ports	Input mode
Stack Pointer	Points to the top of the stack
Carrier output	Low level state or floating state*

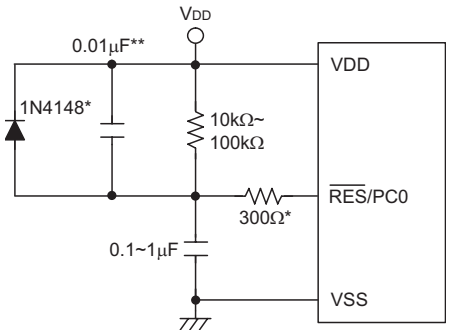
"*" Determined by configuration option



Reset Timing Chart



Reset Configuration



Note: "*" It is recommended that this component is added for added ESD protection

"**" It is recommended that this component is added in environments where power line noise is significant.

External $\overline{\text{RES}}$ Circuit

The chip reset status of the registers is summarised in the following table:

Register	Power-on Reset	$\overline{\text{RES}}$ or LVR Reset	WDT Time-out (Operation)	WDT Time-out (HALT)*
PC	0 0 0 H	0 0 0 H	0 0 0 H	0 0 0 H
MP	-xxx xxxx	-uuu uuuu	-uuu uuuu	-uuu uuuu
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLP	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLH	--xx xxxx	--uu uuuu	--uu uuuu	--uu uuuu
STATUS	--00 xxxx	--uu uuuu	--1u uuuu	--11 uuuu
PA	1111 1111	1111 1111	1111 1111	uuuu uuuu
PB	111x 1111	111x 1111	111x 1111	uuuu uuuu
PC	---- ---1	---- ---1	---- ---1	---- ---u
TSR0	0000 0000	0000 0000	0000 0000	uuuu uuuu
TSR1	1000 0000	1000 0000	1000 0000	uuuu uuuu
CARL0	0000 0000	0000 0000	0000 0000	uuuu uuuu
CARL1	0000 0000	0000 0000	0000 0000	uuuu uuuu
CARH0	0000 0000	0000 0000	0000 0000	uuuu uuuu
CARH1	0000 0010	0000 0010	0000 0010	uuuu uuuu

Note: "-" stands for unimplemented

"u" means unchanged

"x" means unknown

Input/Output Ports

There are up to 17 bidirectional input/output lines in the device, labeled PA, PB and PC which are mapped to [12H], [14H] and [16H] of the Data Memory, respectively. Each line of PA and PB can be selected as NMOS output or Schmitt trigger input with pull-high resistor by a software instruction. PC0 can be used as an input line with Schmitt trigger but without pull-high resistor or as the external $\overline{RES}$ pin determined by the configuration option.

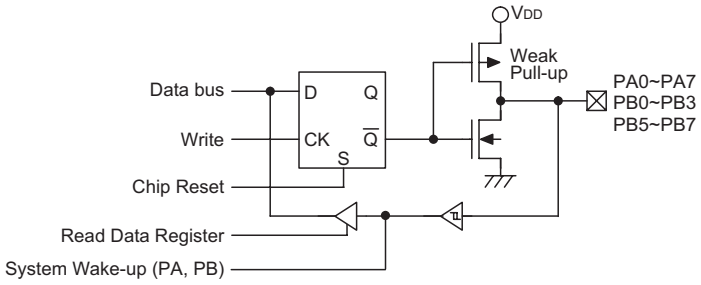
When the I/O ports are used for input operation, these ports are non-latched, that is, the inputs should be ready at the T2 rising edge of the instruction "MOV A, [m]" (m=12H, 14H or 16H). For I/O ports output operations, all data is latched and remains unchanged until the output latch is rewritten.

When the I/O Ports are used for input operations, it should be noted that before reading data from the pads, a "1" should be written to the related bits to disable the NMOS device. That is, the instruction "SET [m].i" (i=0~7 for PA, i=0~3, 5~7 for PB, i=0 for PC) is executed first to disable related NMOS device, and then "MOV A, [m]" to get stable data.

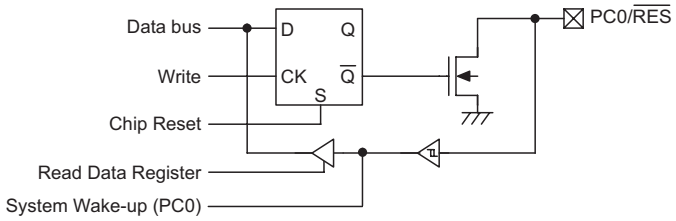
After chip reset, the I/O Ports remain at a high level input line. Each bit of the I/O ports output latches can be set or cleared by the "SET [m].i" and "CLR [m].i" (m=12H, 14H or 16H) instructions respectively.

Some instructions first input data and then follow the output operations. For example, "SET [m].i", "CLR [m]", "CPL [m]" and "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 to the accumulator.

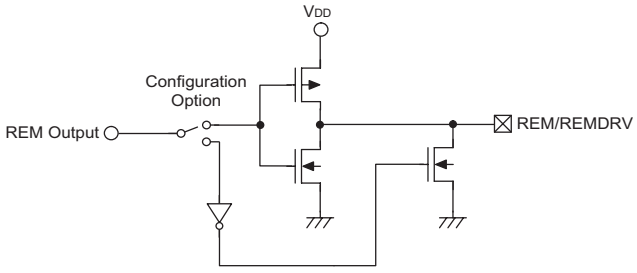
Each line of the I/O ports has a wake-up capability when the relevant pin is configured as an input line.



PA and PB Input/Output Ports



PC0 Input/Output Port



REM/REMDRV Output Pin Structure

Timer

The timer is an internal unit for creating a remote control transmission pattern. As shown, it consists of a 9-bit down counter (t8 to t0), a flag (t9) permitting the 1-bit timer output, and a zero detector.

No.	Label	Function
0~7	t0~t7	Down counter

TSR0 (18H) Register

No.	Label	Function
0	t8	Down counter
1	t9	Timer enable, initial value is “0”.
2~6	—	Unused bit, read as “0”.
7	TOEF	Timer operation end flag, initial value is “1”.

TSR1 (19H) Register

Timer Operation

The timer starts counting down when a value other than “0” is set for the down counter with a timer manipulation instruction. The timer manipulation instructions for making the timer start operation are shown below:

```
MOV A,XXH     ; XX = 00H ~ FFH
MOV TSR0,A
MOV A,XXH     ; XX   01H, t8
MOV TSR1,A
SET TSR1.1    ; The timer is started by set t9=1
```

Addition notes for the 9-bit timer:

- Writing to TSR0 will only put the written data to the TSR0 register (t7~t0) and writing to TSR1 (t8) will transfer the specified data and contents of TSR0 to the Down Counter. TOEF will be cleared after the data transferred from TSR1 and TSR0 to the Down Counter is completed and then wait until TSR1.1 is set by user.
- Setting TSR1.1=1, the timer will start counting. The timer will stop when its count is equal to “0” and then TOEF is set equal to “1”.
- If the TSR1.1 is cleared during the timer counting, the timer will be stopped. Once the TSR1.1 is set (1→0→1), the down counter will reload data from t8~t0, and then the down counter begins counting down with the new load data.
- If TSR1.1 and TOEF are equal to 1 both, the timer can re-start, after new data is written to TSR0, TSR1 (t0~t8) in sequence.

Note: If the contents of the Down counter is 000H, set the t9 to start the timer counting, the timer will only count 1 step. The timer output time=$64/f_{SYS}$. →[(0+1) ×$64/f_{SYS}$=$64/f_{SYS}$]

The down counter is decremented (-1) in the cycle of $64/f_{SYS}$. If the value of the down counter becomes "0", the zero detector generates the timer operation end signal to stop the timer operation. At this time, TOEF will be set to "1". The output of the timer operation end signal is continued while the down counter is "0" and the timer is stopped. The following relational expression applies between the timer's output time and the down counter's set value.

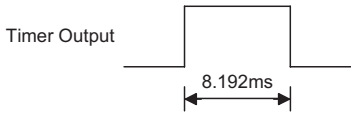
Timer output time = (Set value+1) $\times 64/f_{SYS}$

An example is shown below for f_{SYS} = 4MHz

```
MOV A,0FFH
MOV TSR0,A
MOV A,01H
MOV TSR1,A
SET TSR1.1
```

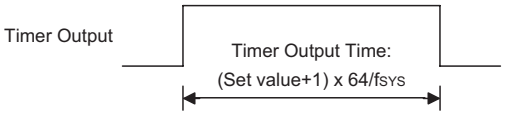
In the case above, the timer output time is as follows.

(Set value+1) $\times$ $64/f_{SYS}$= (511+1) $\times$16μs= 8.192ms

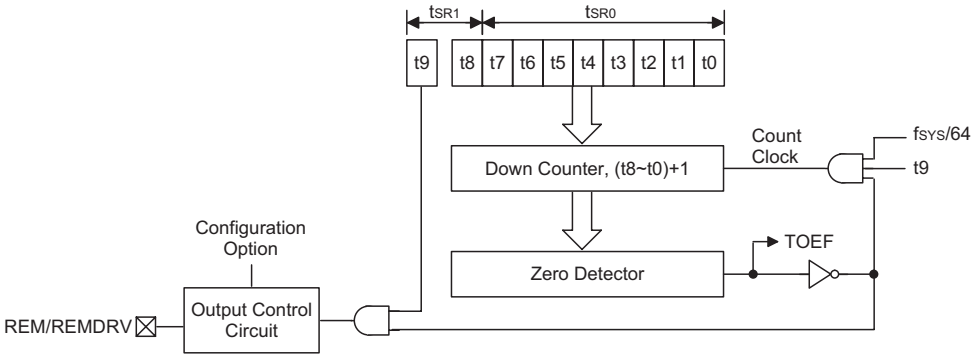


Setting the t9 bit channels the timer to the REM pin. The REM pin will be a combination of the timer and carrier signals.

Note: The carrier output results if bit 9 of the high-level period setting modulo register (CARH) is cleared ("0").



Timer Output when Carrier is not Output



Timer Configuration

Carrier Output

Carrier output generator

The carrier generator consists of a 9-bit counter and two modulo registers for setting the high-level and low-level periods - CARH and CARL respectively.

Register	7	6	5	4	3	2	1	0
CARL0	CL.7	CL.6	CL.5	CL.4	CL.3	CL.2	CL.1	CL.0
CARL1	—	—	—	—	—	—	Fix“0”	CL.8
CARH0	CH.7	CH.6	CH.5	CH.4	CH.3	CH.2	CH.1	CH.0
CARH1	—	—	—	—	—	—	CH.9 (CARY)	CH.8

CARL0 (1AH) Register, CARL1 (1BH), CARH0 (1CH) Register, CARH1 (1DH), Register

Note: 1. CARH1.1 (CARY) initial value is “1”.

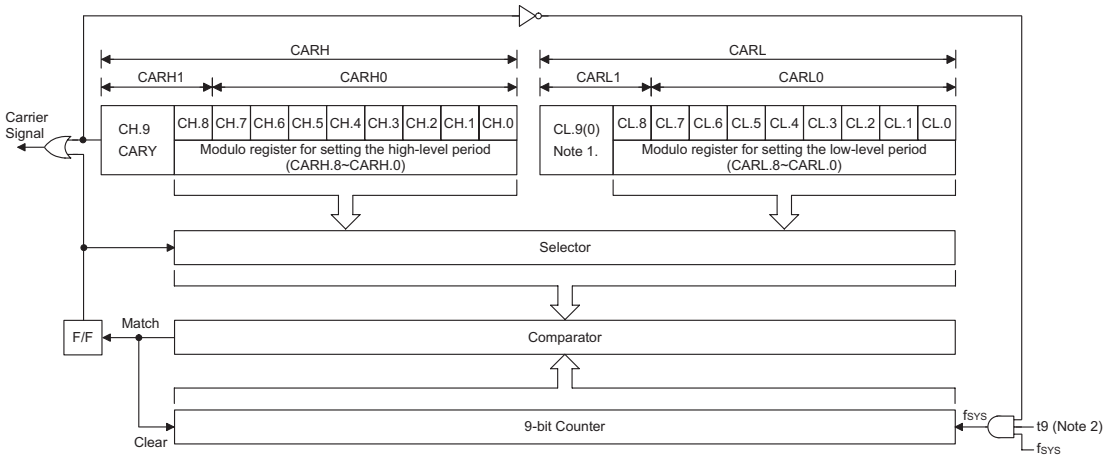
2. CARL1.2 (CARH1.2) ~ CARL1.7 (CARH1.7) are unused bits, read as “0”.

The carrier duty ratio and carrier frequency can be determined by setting the high-level and low-level widths using the respective modulo registers. Each of these widths can be set in a range of 500ns to 64μs at f_{SYS}=4MHz.

CARH (CARH1.0, CARH0.7~CARH0.0) and CARL (CARL1.0, CARL0.7~CARL0.0) are read and written using instructions.

- Example:

```
MOV     A,XXH       ; XXH = 00H~FFH
MOV     CARL0,A
MOV     A,XXH       ; XXH 01H, CL.8 (CARL1.0)
MOV     CARL1,A
MOV     A,XXH       ; XXH = 00H~FFH
MOV     CARH0,A
MOV     A,XXH       ; XXH 02H, CH.8 (CARH1.0)
MOV     CARH1,A
CLR     CARH1.1     ; The carrier is started by clearing CARY(CARH1.1)= "0"
```



Configuration of Remote Controller Carrier Generator

Note: 1. Bit 9 of the modulo register for setting the low-level period (CARL) is fixed to “0”.

2. t9: Flag that enables timer output (timer block, see Timer Configuration)

The values of CARH and CARL can be calculated from the following expressions.

CARL (CARL1.0, CARL0.7~CARL0.0) = (f_{SYS} × (1 D) × T) – 1 …… (1)

CARH (CARH1.0, CARH0.7~CARH0.0) = (f_{SYS} × D × T) – 1 ……. (2)

(1) + (2) => CARL + CARH = (f_{SYS} × T) – 2 => Actual Carrier Frequency = f_{SYS}/(CARL + CARH +2)

D: Carrier duty ratio (0 < D < 1)

f_{SYS}: Input clock (MHz)

T: Carrier cycle (μs)

Ensure to input values in the range of 001H to 1FFH to CARL and CARH.

• Example:

```
If fSYS= 4095kHz, Target fc= 38kHz, T= 1/fc= 26.3157μs= tL+tH, duty= 1/3
CARL= (4.095M×(1-1/3)×26.3157μs) - 1= 70.842
       select 71= 47H , actual tL= (71+1)/4.095M= 17.58μs
CARH= (4.095M×1/3×26.3157μs) - 1= 34.921
       select 35= 23H , actual tH= (35+1)/4.095M= 8.79μs

For actual Carrier Frequency= fSYS/(CARL+CARH+2)
So, actual fc= fSYS/(CARL+CARH+2)= 4095kHz/(71+35+2)= 37.917kHz
MOV    A,045H
MOV    CARL0,A
MOV    A,022H
MOV    CARH0,A
CLR    CARH1.1     ; The carrier is started by clearing CARY(CARH1.1)= "0"
```

Carrier output control

The remote controller carrier can be output from the REM pin by clearing to zero bit 9 (CARY) of the modulo register for setting the high-level period (CARH).

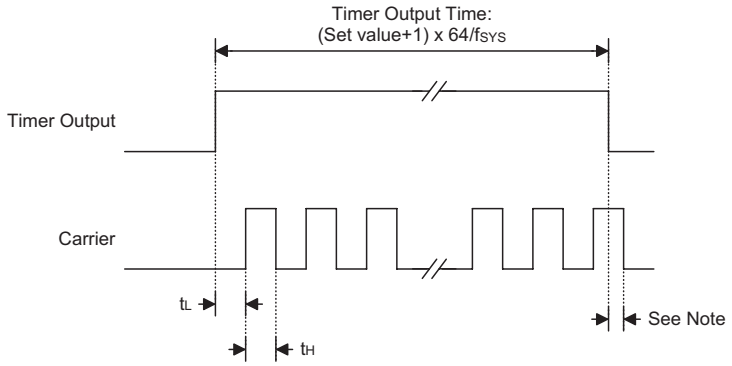
When performing a carrier output, be sure to set the timer operation after setting the CARH (CARH1.0, CARH0.7~CARH0.0) and CARL (CARL1.0, CARL0.7~CARL0.0) values.

Note that a malfunction may occur if the values of CARH (CARH1.0, CARH0.7~CARH0.0) and CARL (CARL1.0, CARL0.7~CARL0.0) are changed while the carrier is being output on the REM pin.

Executing the timer manipulation instruction starts the carrier output from the low level.

There is a dual function remote controller carrier output pin named REM. The selection of REM or REMDRV is determined by a configuration option. After a reset, the REM carrier output pin will have a low level while the REMDRV carrier output pin will be in a floating condition. The generic structures of the REM or REMDRV function are illustrated in the accompanying diagram. As the exact construction of the carrier output pin will differ from these drawings, they are supplied as a guide only to assist with the functional understanding of the remote carrier output pins.

The output from the REM pin is in accordance with the value of bit 9 (CARY) of CARH and the timer output enable flag (t9), and the value of the timer 9-bit down counter (t0 to t8).



Timer Output when Carrier is Output

Note: When the carrier signal is active and during the time when the signal is high, if the timer output should go low, the carrier signal will first complete its high level period before going low.

CARH1.1	Timer Output Enable Flag (t9: TSR1.1)	9-bit Down Counter	REM Function (CMOS Output)	REMDRV Function (NMOS Output)
0	0	0	Low-level output	Floating output
0	0	Other than 0		
0	1	0	$64/f_{SYS}$ (with carrier output)	$64/f_{SYS}$ (with carrier output)
0	1	Other than 0	Carrier output (Note)	Carrier output
1	0	—	Low-level output	Floating output
1	1	—	High-level output	Low-level output

REM Pin Output Control

Note: Input values in the range of 001H to 1FFH to CARH (CARH1.0, CARH0.7~CARH0.0) and CARL (CARL1.0, CARL0.7~CARL0.0).

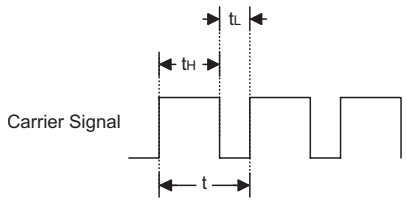
Caution: CARH (CARH1.0, CARH0.7~CARH0.0) and CARL (CARL1.0, CARL0.7~CARL0.0) must be set while the REM pin is at a low level (t9 = 0 or t0 to t8 = 0).

Target		Setting		Actual			
f_C(kHz)	Duty	CARH(CARH1.0, CARH0.7~CARH0.0)	CARL(CARL1.0, CARL0.7~CARL0.0)	t_H(μs)	t_L(μs)	T(μs)	f_C(kHz)
36	1/3	25H	4BH	9.28	18.56	27.84	35.92
38	1/3	23H	47H	8.79	17.58	26.37	37.92
56	1/3	18H	2FH	6.11	11.72	17.83	56.10
56	1/2	23H	24H	8.79	9.04	17.83	56.10

Carrier Frequency Setting (f_{SYS}= 4095kHz)

Target		Setting		Actual			
f_C(kHz)	Duty	CARH(CARH1.0, CARH0.7~CARH0.0)	CARL(CARL1.0, CARL0.7~CARL0.0)	t_H(μs)	t_L(μs)	T(μs)	f_C(kHz)
36	1/3	24H	49H	9.25	18.50	27.75	36.04
38	1/3	22H	45H	8.75	17.50	26.25	38.10
56	1/3	17H	2EH	6.00	11.75	17.75	56.34
56	1/2	23H	22H	9.00	8.75	17.75	56.34

Carrier Frequency Setting (f_{SYS}= 4MHz)

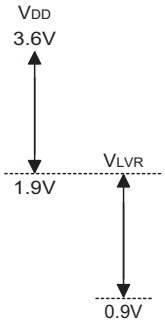


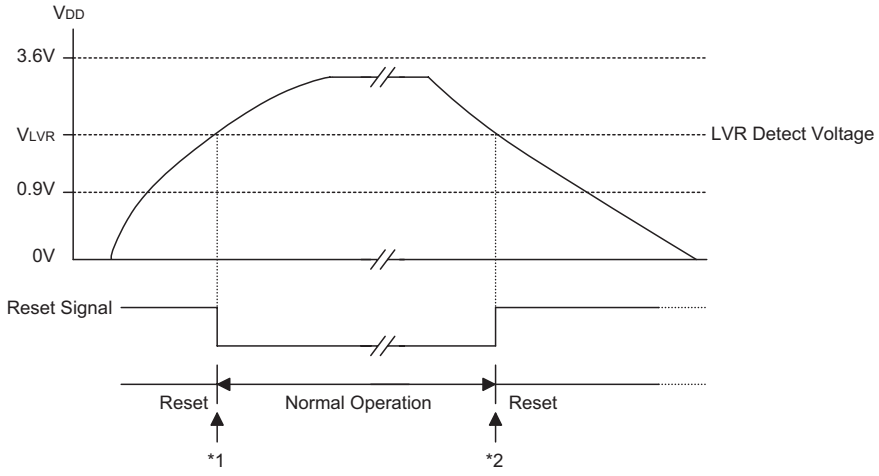
Low Voltage Reset – LVR

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

The LVR includes the following specifications:

- The low voltage (0.9V~V_{LVR}) has to remain in this state for a time in excess of 1ms. If the low voltage state does not exceed 1ms, the LVR will ignore it and will not perform a reset function. The relationship between V_{DD} and V_{LVR} is shown below.





Low Voltage Reset

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

"*2" Since low voltage has to be maintained in its original state and exceed 1ms, a 1ms delay enters the reset mode.

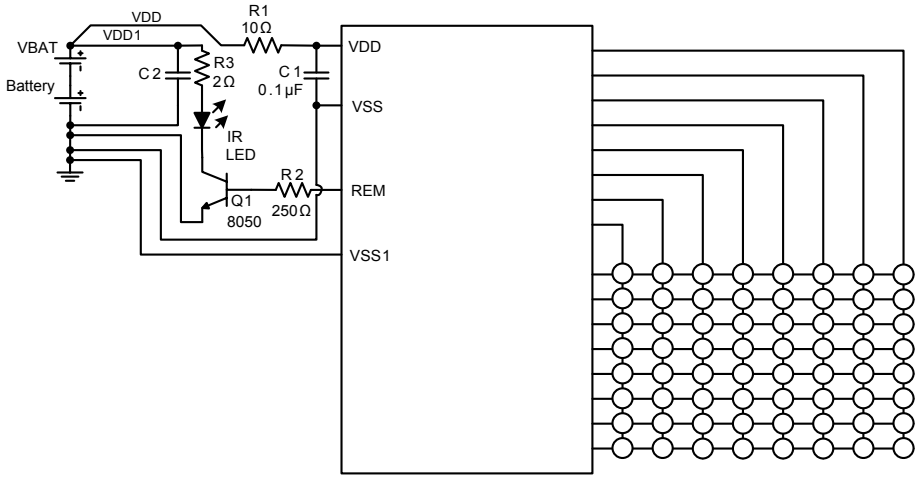
Configuration Options

The following table shows the range of configuration options for the device. All the configuration options must be defined to ensure proper system functioning.

No.	Code Options
Oscillator Options	
1	System oscillator selection - f_{SYS}: XTAL oscillator without internal feedback resistor XTAL oscillator with internal feedback resistor Internal 4095kHz RC oscillator
REM/REMDRV Pin Options	
2	REM or REMDRV output function selection
Watchdog Options	
3	WDT clock selection - f_S: Internal RC oscillator or $f_{SYS}/4$
4	WDT function: enable or disable
5	CLRWDT instruction selections: 1 or 2 instructions
6	WDT time-out period selections: $2^8/f_S$, $2^9/f_S$, $2^{10}/f_S$, $2^{11}/f_S$
LVR Options	
7	LVR function: enable or disable
Reset Pin Options	
8	I/O or $\overline{RES}$ pin selection

Application Circuits

- The following application circuit shows the situation when a transistor is added to IR driver circuit. Here the MCU REM function must be enabled.



Note: 1. The values of R1 and C2 should be selected in consultation with the actual application, R1=10Ω, C1=0.1μF, C2=200~330μF are recommended values.

2. To obtain a better frequency stability and longer transmission distances, C2=330μF is a recommended value. The frequency stability may be different and the transmission distance may be shorter if a value other than 330μF is used.

3. To maintain the accuracy of the internal HIRC oscillator frequency, a 0.1μF decoupling capacitor should be connected between VDD and VSS on the PCB.

4. The C1 (0.1μF) decoupling capacitor should be located as close to the VDD and VSS pins as possible.

5. R1 and C1 should be located as close to the VDD pin as possible.

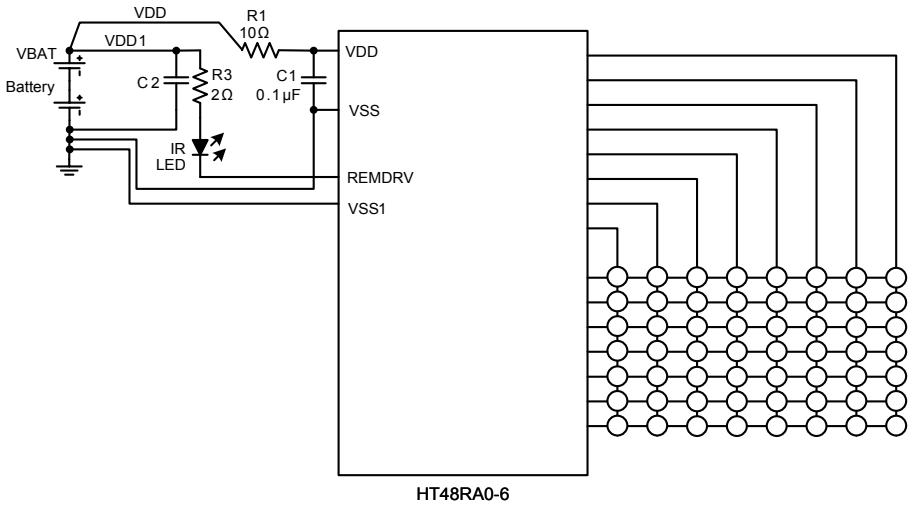
6. VSS, VSS1, C2 and Q1 must be connected to the power GND terminal.

7. VDD and VDD1 must be connected to the power VBAT terminal.

8. The values of R1 and C2 should be selected in consultation with the actual application.

9. It should to be noted that when programming the device, the HT48RA0-6 writer type is the e-Writer PRO, which when used together with the e-Socket, can ensure that the HIRC oscillator frequency will have a tolerance within 1% in the actual application circuit.

- The following application circuit shows the situation when the internal IR driver circuit is used. Here the MCU REMDRV function must be enabled.



Note: 1. The values of R1 and C2 should be selected in consultation with the actual application, R1=10Ω, C1=0.1μF, C2=47~100μF are recommended values.

2. To obtain a better frequency stability and longer transmission distances, C2=100μF is a recommended value. The frequency stability may be different and the transmission distance may be shorter if a value other than 100μF is used.

3. To maintain the accuracy of the internal HIRC oscillator frequency, a 0.1μF decoupling capacitor should be connected between VDD and VSS on the PCB.

4. The C1 (0.1μF) decoupling capacitor should be located as close to the VDD and VSS pins as possible.

5. R1 and C1 should be located as close to the VDD pin as possible.

6. VSS, VSS1, C2 and Q1 must be connected to the power GND terminal.

7. VDD and VDD1 must be connected to the power VBAT terminal.

8. The values of R1 and C2 should be selected in consultation with the actual application.

9. It should to be noted that when programming the device, the HT48RA0-6 writer type is the e-Writer PRO, which when used together with the e-Socket, can ensure that the HIRC oscillator frequency will have a tolerance within 1% in the actual application circuit.

Instruction Set

Introduction

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

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

Instruction Timing

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

Moving and Transferring Data

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

Arithmetic Operations

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

Logical and Rotate Operation

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

Branches and Control Transfer

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

Bit Operations

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

Table Read Operations

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

Other Operations

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

Instruction Set Summary

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

Table Conventions

x: Bits immediate data
m: Data Memory address
A: Accumulator
i: 0~7 number of bits
addr: Program memory address

Mnemonic	Description	Cycles	Flag Affected
Arithmetic			
ADD A,[m]	Add Data Memory to ACC	1	Z, C, AC, OV
ADDM A,[m]	Add ACC to Data Memory	1Note	Z, C, AC, OV
ADD A,x	Add immediate data to ACC	1	Z, C, AC, OV
ADC A,[m]	Add Data Memory to ACC with Carry	1	Z, C, AC, OV
ADCM A,[m]	Add ACC to Data memory with Carry	1Note	Z, C, AC, OV
SUB A,x	Subtract immediate data from the ACC	1	Z, C, AC, OV
SUB A,[m]	Subtract Data Memory from ACC	1	Z, C, AC, OV
SUBM A,[m]	Subtract Data Memory from ACC with result in Data Memory	1Note	Z, C, AC, OV
SBC A,[m]	Subtract Data Memory from ACC with Carry	1	Z, C, AC, OV
SBCM A,[m]	Subtract Data Memory from ACC with Carry, result in Data Memory	1Note	Z, C, AC, OV
DAA [m]	Decimal adjust ACC for Addition with result in Data Memory	1Note	C
Logic Operation			
AND A,[m]	Logical AND Data Memory to ACC	1	Z
OR A,[m]	Logical OR Data Memory to ACC	1	Z
XOR A,[m]	Logical XOR Data Memory to ACC	1	Z
ANDM A,[m]	Logical AND ACC to Data Memory	1Note	Z
ORM A,[m]	Logical OR ACC to Data Memory	1Note	Z
XORM A,[m]	Logical XOR ACC to Data Memory	1Note	Z
AND A,x	Logical AND immediate Data to ACC	1	Z
OR A,x	Logical OR immediate Data to ACC	1	Z
XOR A,x	Logical XOR immediate Data to ACC	1	Z
CPL [m]	Complement Data Memory	1Note	Z
CPLA [m]	Complement Data Memory with result in ACC	1	Z
Increment & Decrement			
INCA [m]	Increment Data Memory with result in ACC	1	Z
INC [m]	Increment Data Memory	1Note	Z
DECA [m]	Decrement Data Memory with result in ACC	1	Z
DEC [m]	Decrement Data Memory	1Note	Z
Rotate			
RRA [m]	Rotate Data Memory right with result in ACC	1	None
RR [m]	Rotate Data Memory right	1Note	None
RRCA [m]	Rotate Data Memory right through Carry with result in ACC	1	C
RRC [m]	Rotate Data Memory right through Carry	1Note	C
RLA [m]	Rotate Data Memory left with result in ACC	1	None
RL [m]	Rotate Data Memory left	1Note	None
RLCA [m]	Rotate Data Memory left through Carry with result in ACC	1	C
RLC [m]	Rotate Data Memory left through Carry	1Note	C

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

Note: 1. For skip instructions, if the result of the comparison involves a skip then two cycles are required, if no skip takes place only one cycle is required.

2. Any instruction which changes the contents of the PCL will also require 2 cycles for execution.

3. For the “CLR WDT1” and “CLR WDT2” instructions the TO and PDF flags may be affected by the execution status. The TO and PDF flags are cleared after both “CLR WDT1” and “CLR WDT2” instructions are consecutively executed. Otherwise the TO and PDF flags remain unchanged.

Instruction Definition

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

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

Operation ACC ← ACC + [m] + C

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

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

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

Operation [m] ← ACC + [m] + C

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

ADD A,[m] Add Data Memory to ACC

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

Operation ACC ← ACC + [m]

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

ADD A,x Add immediate data to ACC

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

Operation ACC ← ACC + x

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

ADDM A,[m] Add ACC to Data Memory

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

Operation [m] ← ACC + [m]

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

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

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

Operation ACC ← ACC "AND" [m]

Affected flag(s) Z

AND A,x Logical AND immediate data to ACC

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

Operation ACC ← ACC "AND" x

Affected flag(s) Z

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

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

Operation [m] ← ACC "AND" [m]

Affected flag(s) Z

CALL addr Subroutine call

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

Operation Stack ← Program Counter + 1
Program Counter ← addr

Affected flag(s) None

CLR [m] Clear Data Memory

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

Operation [m] ← 00H

Affected flag(s) None

CLR [m].i Clear bit of Data Memory

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

Operation [m].i ← 0

Affected flag(s) None

CLR WDT Clear Watchdog Timer

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

Operation WDT cleared
TO ← 0
PDF ← 0

Affected flag(s) TO, PDF

CLR WDT1 Pre-clear Watchdog Timer

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

Operation WDT cleared
TO ← 0
PDF ← 0

Affected flag(s) TO, PDF

CLR WDT2 Pre-clear Watchdog Timer

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

Operation WDT cleared
TO ← 0
PDF ← 0

Affected flag(s) TO, PDF

CPL [m] Complement Data Memory

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

Operation [m] ← $\overline{[m]}$

Affected flag(s) Z

CPLA [m] Complement Data Memory with result in ACC

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

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

Affected flag(s) Z

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

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

Operation [m] ← ACC + 00H or
[m] ← ACC + 06H or
[m] ← ACC + 60H or
[m] ← ACC + 66H

Affected flag(s) C

DEC [m] Decrement Data Memory

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

Operation [m] ← [m] − 1

Affected flag(s) Z

DECA [m] Decrement Data Memory with result in ACC

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

Operation ACC ← [m] − 1

Affected flag(s) Z

HALT Enter power down mode

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

Operation TO ← 0
PDF ← 1

Affected flag(s) TO, PDF

INC [m] Increment Data Memory

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

Operation [m] ← [m] + 1

Affected flag(s) Z

INCA [m] Increment Data Memory with result in ACC

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

Operation ACC ← [m] + 1

Affected flag(s) Z

JMP addr Jump unconditionally

Description The contents of the Program Counter are replaced with the specified address. Program execution then continues from this new address. As this requires the insertion of a dummy instruction while the new address is loaded, it is a two cycle instruction.

Operation Program Counter ← addr

Affected flag(s) None

MOV A,[m] Move Data Memory to ACC

Description The contents of the specified Data Memory are copied to the Accumulator.

Operation ACC ← [m]

Affected flag(s) None

MOV A,x Move immediate data to ACC

Description The immediate data specified is loaded into the Accumulator.

Operation ACC ← x

Affected flag(s) None

MOV [m],A Move ACC to Data Memory

Description The contents of the Accumulator are copied to the specified Data Memory.

Operation [m] ← ACC

Affected flag(s) None

NOP No operation

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

Operation No operation

Affected flag(s) None

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

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

Operation ACC ← ACC ″OR″ [m]

Affected flag(s) Z

OR A,x Logical OR immediate data to ACC

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

Operation ACC ← ACC ″OR″ x

Affected flag(s) Z

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

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

Operation [m] ← ACC ″OR″ [m]

Affected flag(s) Z

RET Return from subroutine

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

Operation Program Counter ← Stack

Affected flag(s) None

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

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

Operation Program Counter ← Stack
ACC ← x

Affected flag(s) None

RETI Return from interrupt

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

Operation Program Counter ← Stack
EMI ← 1

Affected flag(s) None

RL [m] Rotate Data Memory left

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

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

Affected flag(s) None

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

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

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

Affected flag(s) None

RLC [m] Rotate Data Memory left through Carry

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

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

Affected flag(s) C

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

Description Data in the specified Data Memory and the carry flag are rotated left by 1 bit. Bit 7 replaces the

Carry bit and the original carry flag is rotated into the bit 0. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

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

Affected flag(s) C

RR [m] Rotate Data Memory right

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

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

Affected flag(s) None

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

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

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

Affected flag(s): None

RRC [m] Rotate Data Memory right through Carry

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

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

Affected flag(s): C

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

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

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

Affected flag(s): C

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

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

Operation: ACC ← ACC − [m] − C

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

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

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

Operation: [m] ← ACC − [m] − C

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

SDZ [m] Skip if decrement Data Memory is 0

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

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

Affected flag(s): None

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

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

Operation ACC ← [m] − 1
Skip if ACC=0

Affected flag(s) None

SET [m] Set Data Memory

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

Operation [m] ← FFH

Affected flag(s) None

SET [m].i Set bit of Data Memory

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

Operation [m].i ← 1

Affected flag(s) None

SIZ [m] Skip if increment Data Memory is 0

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

Operation [m] ← [m] + 1
Skip if [m]=0

Affected flag(s) None

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

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

Operation ACC ← [m] + 1
Skip if ACC=0

Affected flag(s) None

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

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

Operation Skip if [m].i ≠ 0

Affected flag(s) None

SUB A,[m] Subtract Data Memory from ACC

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

Operation ACC ← ACC − [m]

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

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

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

Operation: [m] ← ACC − [m]

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

SUB A,x Subtract immediate data from ACC

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

Operation: ACC ← ACC − x

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

SWAP [m] Swap nibbles of Data Memory

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

Operation: [m].3~[m].0 ↔ [m].7~[m].4

Affected flag(s): None

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

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

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

Affected flag(s): None

SZ [m] Skip if Data Memory is 0

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

Operation: Skip if [m]=0

Affected flag(s): None

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

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

Operation: ACC ← [m]
Skip if [m]=0

Affected flag(s): None

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

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

Operation: Skip if [m].i=0

Affected flag(s): None

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

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

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

Affected flag(s) None

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

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

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

Affected flag(s) None

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

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

Operation ACC ← ACC ″XOR″ [m]

Affected flag(s) Z

XORM A,[m] Logical XOR ACC to Data Memory

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

Operation [m] ← ACC ″XOR″ [m]

Affected flag(s) Z

XOR A,x Logical XOR immediate data to ACC

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

Operation ACC ← ACC ″XOR″ x

Affected flag(s) Z

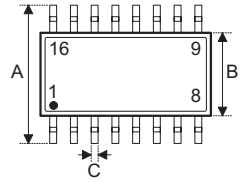
Package Information

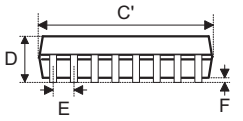
Note that the package information provided here is for consultation purposes only. As this information may be updated at regular intervals users are reminded to consult the Holtek website for the latest version of the Package/Carton Information.

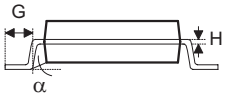
Additional supplementary information with regard to packaging is listed below. Click on the relevant section to be transferred to the relevant website page.

- Package Information (include Outline Dimensions, Product Tape and Reel Specifications)
- The Operation Instruction of Packing Materials
- Carton information

16-pin NSOP (150mil) Outline Dimensions



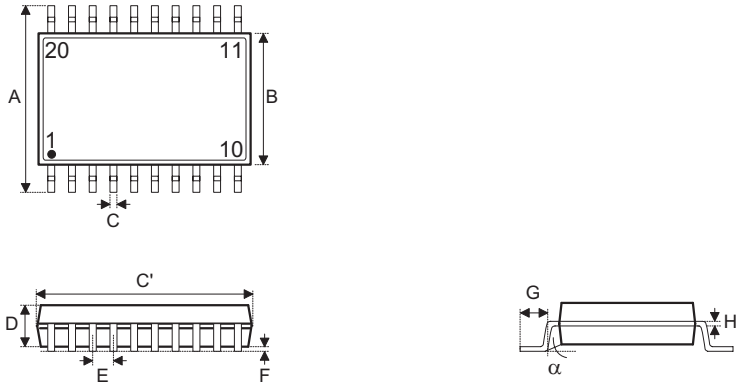




Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	—	0.236 BSC	—
B	—	0.154 BSC	—
C	0.012	—	0.020
C'	—	0.390 BSC	—
D	—	—	0.069
E	—	0.050 BSC	—
F	0.004	—	0.010
G	0.016	—	0.050
H	0.004	—	0.010
α	0°	—	8°

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	—	6.000 BSC	—
B	—	3.900 BSC	—
C	0.31	—	0.51
C'	—	9.900 BSC	—
D	—	—	1.75
E	—	1.270 BSC	—
F	0.10	—	0.25
G	0.40	—	1.27
H	0.10	—	0.25
α	0°	—	8°

20-pin SSOP (150mil) Outline Dimensions



Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	—	0.236 BSC	—
B	—	0.154 BSC	—
C	0.008	—	0.012
C’	—	0.341 BSC	—
D	—	—	0.069
E	—	0.025 BSC	—
F	0.004	—	0.0098
G	0.016	—	0.05
H	0.004	—	0.01
α	0°	—	8°

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	—	6.000 BSC	—
B	—	3.900 BSC	—
C	0.20	—	0.30
C’	—	8.660 BSC	—
D	—	—	1.75
E	—	0.635 BSC	—
F	0.10	—	0.25
G	0.41	—	1.27
H	0.10	—	0.25
α	0°	—	8°