HT47R10A-1/HT47C10-1
R-F Type 8-Bit MCU

Technical Document

- Tools Information
- FAQs
- Application Note
  - HA0029E Using the Time Base Function in the HT47R20A-1
  - HA0030E Using the RTC in the HT47R20A-1
  - HA0034E Using the Buzzer Function in the HT47R20A-1
  - HA0036E Using the PFD Function in the HT47R20A-1

Features

- Operating voltage:
  f_{SYS}=4MHz: 2.2V~5.5V
  f_{SYS}=8MHz: 3.3V~5.5V
- Eight bidirectional I/O lines
- Single external interrupt input
- Single 16-bit programmable timer/event counter
- On-chip crystal and RC oscillator for system clock
- 32.768kHz crystal oscillator for real time clock or system clock
- Watchdog Timer
- 1K×16 program memory
- 32×8 data memory RAM
- Real Time Clock (RTC)
- 8-bit prescaler for RTC
- Low voltage detector
- Low voltage reset circuit
- Buzzer output
- Power down and and wake-up functions reduce power consumption
- C type or R type LCD bias
- LCD driver circuits with 10×2, 10×3 or 9×4 segments
- Single channel RC type A/D converter
- Two-level subroutine nesting
- Bit manipulation instructions
- 16-bit table read instruction
- Up to 0.5μs instruction cycle with 8MHz system clock
- All instructions executed within one or two machine cycles
- 63 powerful instructions
- 44-pin QFP package

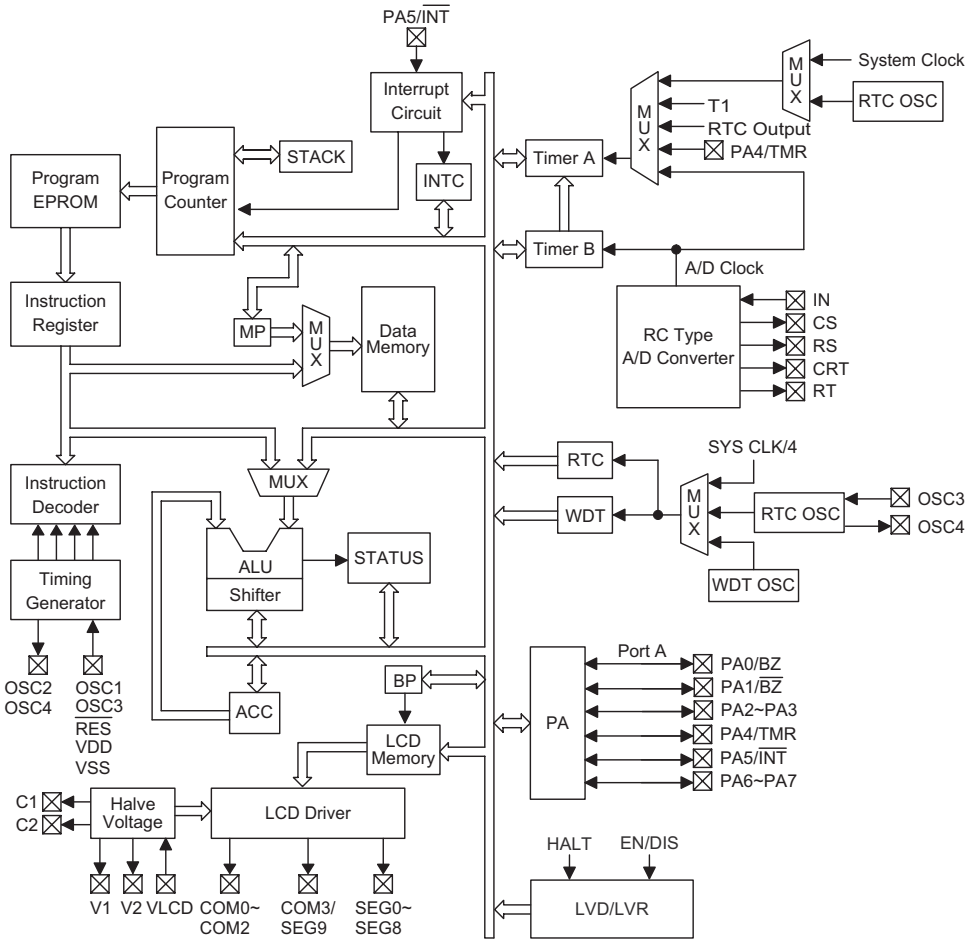
General Description

The HT47R10A-1/HT47C10-1 are 8-bit, high performance, RISC architecture microcontroller devices specifically designed for applications that interface directly to analog signals, such as those from sensors. The mask version HT47C10-1 device is fully pin and functionally compatible with the HT47R10A-1, OTP version device.

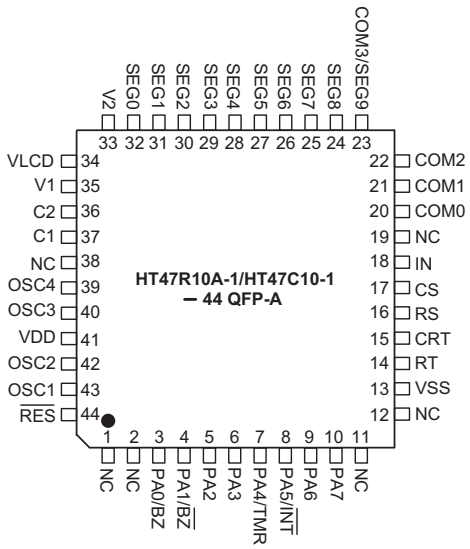
The advantages of low power consumption, I/O flexibility, programmable frequency divider, timer functions, oscillator options, RC type A/D Converter, LCD driver, Power Down and wake-up functions, enhance the versatility of these devices to suit a wide range of Resistor to Frequency application possibilities such as sensor signal processing, remote metering, industrial control, consumer products, subsystem controllers, etc.

The HT47C10-1 is under development and will be available soon.

Block Diagram



Pin Assignment



Pin Description

Pin Name	I/O	Option	Function
PA0/BZ PA1/$\overline{BZ}$ PA2~PA3 PA4/TMR PA5/$\overline{INT}$ PA6~PA7	I/O	Pull-high Wake-up Buzzer	Bidirectional 8-bit input/output port. Each individual pin on this port can be configured as a wake-up input by a configuration option. Software instructions determine if the pin is a CMOS output or Schmitt trigger input. Configuration options determine which pins on this port have pull-high resistors. The buzzer, TMR and external interrupt input are pin-shared with PA0, PA1, PA4 and PA5 respectively.
IN CS RS CRT RT	I O O O O	—	Oscillation input pin Reference capacitor connection pin Reference resistor connection pin Resistor/capacitor sensor connection pin Resistor sensor measurement connection pin
COM0~COM2 COM3/SEG9	O	1/2, 1/3 or 1/4 Duty	COM3/SEG9 can be set as an LCD common or segment output driver by a configuration option. COM0~COM2 are LCD panel plate outputs.
SEG0~SEG8	O	—	LCD panel segments driver outputs
V1, V2, C1, C2	—	—	LCD voltage pump
VLCD	I	—	LCD power supply
OSC2 OSC1	O I	Crystal or RC	OSC1 and OSC2 are connected to an RC network or a external crystal, determined by configuration by option, for the internal system clock. If the RC system clock option is selected, pin OSC2 can be used to measure the system clock at □ frequency. If the system clock originates from the RTC oscillator, which is connected to OSC3 and OSC4, these two pins can be left floating.
OSC4 OSC3	O I	RTC or System Clock	Real time clock oscillator. OSC3 and OSC4 are connected to a 32768Hz crystal oscillator or to a system clock source, determined by configuration option.
$\overline{RES}$	I	—	Schmitt trigger reset input, active low.
VSS	—	—	Negative power supply, ground
VDD	—	—	Positive power supply

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
Storage Temperature ..... –50°C to 125°C
Operating Temperature ..... –40°C to 85°C

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

D.C. Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	Operating Voltage	—	f_{SYS}=4MHz	2.2	—	5.5	V
			f_{SYS}=8MHz	3.3	—	5.5	V
V_{LCD}	LCD Power Supply (Note*)	—	VA≤5.5V	2.2	—	5.5	V
I_{DD1}	Operating Current (Crystal OSC, RC OSC)	3V	No load, f_{SYS}=4MHz	—	1	2	mA
		5V		—	2.5	5	mA

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{DD2}	Operating Current (Crystal OSC, RC OSC)	5V	No load, f_{SYS}=8MHz	—	4	8	mA
I_{DD3}	Operating Current (f_{SYS}=32768Hz)	3V	No load, LCD on, C type LVR and LVD disable	—	40	80	μA
		5V		—	80	160	μA
I_{STB1}	Standby Current (*f_S=f_{SYS}/4)	3V	No load, system HALT, LCD off at HALT	—	—	1	μA
		5V		—	—	2	μA
I_{STB2}	Standby Current (*f_S=RTC OSC)	3V	No load, system HALT, LCD on at HALT, C type	—	2.5	5	μA
		5V		—	10	20	μA
I_{STB3}	Standby Current (*f_S=WDT RC OSC)	3V	No load, system HALT LCD on at HALT, C type	—	2	5	μA
		5V		—	6	10	μA
I_{STB4}	Standby Current (*f_S=RTC OSC)	3V	No load, system HALT, LCD on at HALT, R type, 1/2 bias	—	17	30	μA
		5V		—	34	60	μA
I_{STB5}	Standby Current (*f_S=RTC OSC)	3V	No load, system HALT, LCD on at HALT, R type, 1/3 bias	—	13	25	μA
		5V		—	26	50	μA
I_{STB6}	Standby Current (*f_S=WDT RC OSC)	3V	No load, system HALT, LCD on at HALT, R type, 1/2 bias	—	14	25	μA
		5V		—	28	50	μA
I_{STB7}	Standby Current (*f_S=WDT RC OSC)	3V	No load, system HALT, LCD on at HALT, R type, 1/3 bias	—	10	20	μA
		5V		—	26	40	μA
V_{IL1}	Input Low Voltage for I/O Ports, TMR and $\overline{INT}$	—	—	0	—	$0.3V_{DD}$	V
V_{IH1}	Input High Voltage for I/O Ports, TMR and $\overline{INT}$	—	—	$0.7V_{DD}$	—	V_{DD}	V
V_{IL2}	Input Low Voltage ($\overline{RES}$)	—	—	0	—	$0.4V_{DD}$	V
V_{IH2}	Input High Voltage ($\overline{RES}$)	—	—	$0.9V_{DD}$	—	V_{DD}	V
I_{OL1}	I/O Port Sink Current	3V	V_{OL}=$0.1V_{DD}$	6	12	—	mA
		5V		10	25	—	mA
I_{OH1}	I/O Port Source Current	3V	V_{OH}=$0.9V_{DD}$	–2	-4	—	mA
		5V		–5	–8	—	mA
I_{OL2}	LCD Common and Segment Current	3V	V_{OL}=0.1VA	210	420	—	μA
		5V		350	700	—	μA
I_{OH2}	LCD Common and Segment Current	3V	V_{OH}=0.9VA	–80	–160	—	μA
		5V		–180	–360	—	μA
R_{PH}	Pull-high Resistance of I/O Ports and $\overline{INT}$	3V	—	20	60	100	kΩ
		5V		10	30	50	kΩ
V_{LVR}	Low Voltage Reset	—	—	2.7	3.0	3.3	V
V_{LVD}	Low Voltage Detector Voltage			3.0	3.3	3.6	V

Note: "*" for the value of VA refer to the LCD driver section.

"*f_S" please refer to WDT clock option

A.C. Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
f_{SYS1}	System Clock (Crystal OSC, RC OSC)	—	2.2V~5.5V	400	—	4000	kHz
		—	3.3V~5.5V	400	—	8000	kHz
f_{SYS2}	System Clock (32768Hz Crystal OSC)	—	—	—	32768	—	Hz
f_{RTCOSC}	RTC Frequency	—	—	—	32768	—	Hz
f_{TIMER}	Timer I/P Frequency	—	2.2V~5.5V	0	—	4000	kHz
		—	3.3V~5.5V	0	—	8000	kHz
t_{WDTOSC}	Watchdog Oscillator Period	3V	—	45	90	180	μs
		5V	—	32	65	130	μs
t_{RES}	External Reset Low Pulse Width	—	—	1	—	—	μs
t_{SST}	System Start-up Timer Period	—	Wake-up from HALT	—	1024	—	t_{SYS}
t_{LVR}	Low Voltage Width to Reset	—	—	0.25	1	2	ms
t_{INT}	Interrupt Pulse Width	—	—	1	—	—	μs

Note: *t_{SYS}=1/f_{SYS1}, 1/f_{SYS2}

Functional Description

Execution Flow

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

Instruction fetching and execution are pipelined in such a way that a fetch takes one instruction cycle while decoding and execution takes the next instruction cycle. However, the pipelining scheme causes each instruction to effectively execute in 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 the program memory are executed and its contents specify a maximum of 1024 addresses.

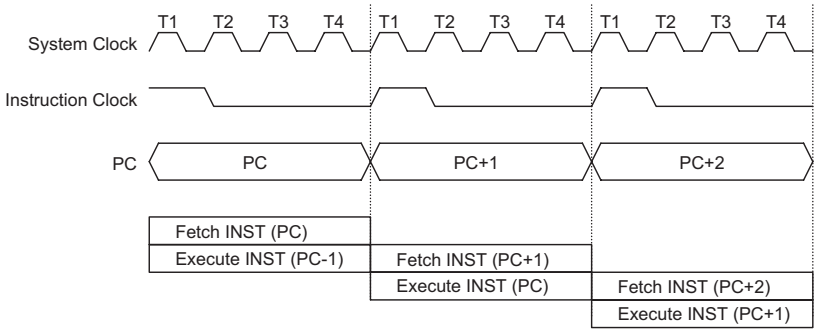
After accessing a program memory word to fetch an instruction code, the contents of the program counter are incremented by 1. The program counter then points to the memory word containing the next instruction code.

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

A 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 it will 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 must 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
External interrupt	0	0	0	0	0	0	0	1	0	0
Timer/event counter interrupt	0	0	0	0	0	0	1	0	0	0
RTC interrupt	0	0	0	0	0	0	1	1	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

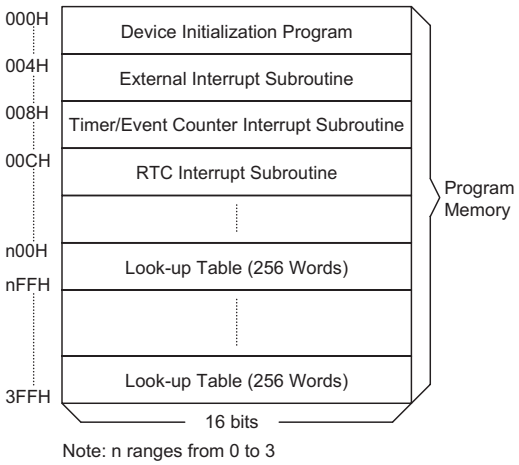
Note: *9~*0: Program counter bits
S9~S0: Stack register bits
#9~#0: Instruction code bits
@7~@0: PCL bits

Program Memory

The program memory is used to store the program instructions which are to be executed. It also contains data, table, and interrupt entries, and is organized into 1024×16 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 initialisation program. After a chip reset, the program always begins execution at location 000H.
- Location 004H
  This area is reserved for the external interrupt service program. If the $\overline{INT}$ interrupt is enabled and the stack is not full, the program begins execution at location 004H.
- Location 008H
  This area is reserved for the timer/event counter interrupt service program. If a timer interrupt results from a Timer/Event Counter A or B overflow, and if the interrupt is enabled and the stack is not full, the program begins execution at location 008H.
- Location 00CH
  This area is reserved for the real time clock interrupt service program. If a real time clock interrupt results from a real time clock overflow, and if the interrupt is enabled and the stack is not full, the program begins execution at location 00CH.
- Table location
  Any location in the Program Memory 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, and the higher-order byte to TBLH (08H). Only the destination of the lower-order byte in the table is well-defined, the higher-order byte of the table word is transferred to the TBLH register. The table higher-order byte register, TBLH, is read only. The TBLP table pointer register is a read/write register (07H), which indicates the table location. Before accessing the table, the location must be placed in TBLP. The TBLH register is read only and cannot be restored. If the main routine and the ISR (Interrupt Service Routine) both employ the table read instruction, the contents of the TBLH in the main routine are likely to be changed by the table read instruction used in the ISR and errors can occur. In other words, using the table read instruction in the main routine and the ISR simultaneously should be avoided. However, if the table read instruction has to be applied in both the main routine and the ISR, the interrupt should be disabled prior to executing the table read instruction. It will not be enabled until the TBLH has been backed up. All table related instructions need two cycles to complete the operation. These areas may function as normal program memory depending upon requirements.



Note: n ranges from 0 to 3

Program Memory

Stack Register – STACK

This is a special part of the memory which is used to save the contents of the Program Counter only. The stack is organised into two 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. During a subroutine call or interrupt acknowledgment, the contents of the program counter are pushed onto the stack. At the end of a subroutine or an interrupt routine, signaled by a return instruction, "RET" or "RETI", the program counter is restored to its previous value from the stack. After a chip reset, the SP will point to the top of the stack.

If the stack is full and a non-masked interrupt takes place, the interrupt request flag will be recorded but the acknowledgment will be inhibited. When the stack pointer is decremented using the "RET" or "RETI" instructions, the interrupt will be serviced. This feature prevents stack overflow allowing the programmer to use

Instruction(s)	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
P9~P8: Current program counter bits
@7~@0: Table pointer bits

the structure more easily. In a similar case, if the stack is full and a "CALL" is subsequently executed, a stack overflow will occur and the first entry will be lost. Only the most recent two return addresses are stored.

Data Memory – RAM

The data memory has a 52×8 bit structure. 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.

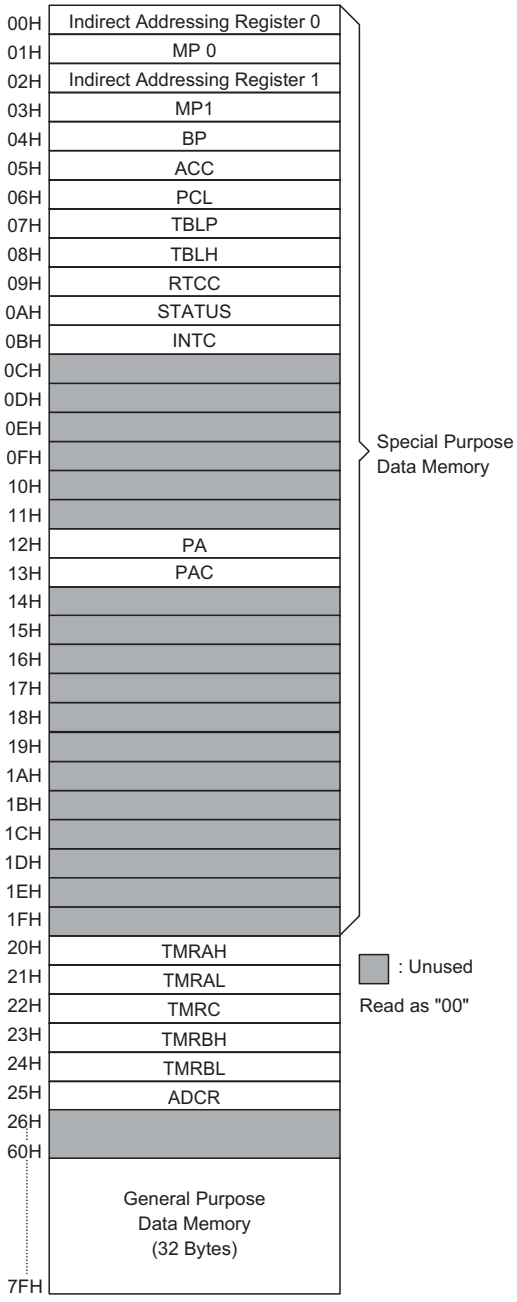
The special function registers include the indirect addressing register 0 (00H), the memory pointer register 0 (mp0; 01H), the indirect addressing register 1 (02H), the memory pointer register 1 (MP1;03H), the bank pointer (BP;04H), the accumulator (ACC;05H), the program counter lower-order byte register (PCL;06H), the table pointer (TBLP;07H), the table higher-order byte register (TBLH;08H), the real time clock control register (RTCC;09H), the status register (STATUS;0AH), the interrupt control register (INTC;0BH), the I/O registers (PA;12H), the I/O control registers (PAC;13H), the Timer/Event Counter A higher order byte register (TMRAH;20H), the Timer/Event Counter A lower order byte register (TMRAL;21H), the Timer/Event Counter control register (TMRC;22H), the Timer/Event Counter B higher order byte register (TMRBH;23H), the Timer/Event Counter B lower order byte register (TMRBL;24H), and the RC oscillator type A/D converter control register (ADCR; 25H). The remaining space before the 60H are reserved for future expanded usage and reading these location will return the result 00H. The general purpose data memory, addressed from 60H to 7FH, is used for data and control information under instruction command.

All data memory areas can handle arithmetic, logic, increment, decrement and rotate operations. 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" instruction, respectively. They are also indirectly accessible through memory pointer registers (MP0;01H, MP1;03H).

Indirect Addressing Register

Locations 00H and 02H are indirect addressing registers that are not physically implemented. Any read/write operation of [00H] and [02H] access data memory pointed to by MP0 (01H) and MP1 (03H) respectively. Reading location 00H or 02H indirectly will return the result 00H. Writing indirectly results in no operation.

The function of data movement between two indirect addressing registers are not supported. The memory pointer registers, MP0 and MP1, are both 8-bit registers which can be used to access the data memory in combination with their corresponding indirect addressing registers.



RAM Mapping (Bank 0)

MP0 can be applied only to data memory, while MP1 can be applied to data memory and the LCD display memory.

Accumulator

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

Arithmetic and Logic Unit – ALU

The ALU 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, etc.)

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

Status Register – STATUS

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

With the exception of the TO and PDF flags, bits in the status register can be altered by instructions like most other registers. Any data written into the status register will not change the TO or PDF 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, system 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 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 are important and if the subroutine can corrupt the status register, precautions must be taken to save it properly.

Interrupts

The HT47R10A-1/HT47C10-1 provides an external interrupt, an internal timer/event counter interrupt and an internal real time clock interrupt. The interrupt control register (INTC;0BH) contains the interrupt control bits to set the enable or disable and the interrupt request flags.

When an interrupt subroutine is serviced, all 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 needs servicing within the service routine, the EMI bit and the corresponding bit of the INTC may be set to allow interrupt nesting. If the stack is full, the interrupt request will not be acknowledged, even if the related interrupt is enabled, until the SP is decremented. If immediate service is desired, the stack must be prevented from becoming full.

All interrupts have a wake-up capability. As an interrupt is serviced, a control transfer occurs by pushing the program counter onto the stack and then by branching to subroutines at specified locations in the program memory. Only the program counter is pushed onto the stack. If the contents of the accumulator and status register are altered by the interrupt service program, this may corrupt the desired control sequence, therefore their contents must be saved first.

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

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 0; 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 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

Bit No.	Label	Function
0	EMI	Master (global) interrupt enable (1= enable; 0= disable)
1	EEI	External interrupt enable (1= enable; 0= disable)
2	ETI	Timer/event counter interrupt enable (1= enable; 0=disable)
3	ERTI	Real time clock interrupt enable (1= enable; 0= disable)
4	EIF	External interrupt request flag (1= active; 0= inactive)
5	TF	Timer/event counter request flag (1= active; 0= inactive)
6	RTF	Real time clock request flag (1= active; 0= inactive)
7	—	Unused bit, read as "0"

INTC (0BH) Register

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

The real time clock interrupt is initialised by setting the real time clock interrupt request flag (RTF; bit 6 of the INTC), caused by a regular real time clock signal. When the interrupt is enabled, and the stack is not full and the RTF bit is set, a subroutine call to location 0CH will occur. The related interrupt request flag, RTF, will be reset and the EMI bit cleared to disable further interrupts.

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

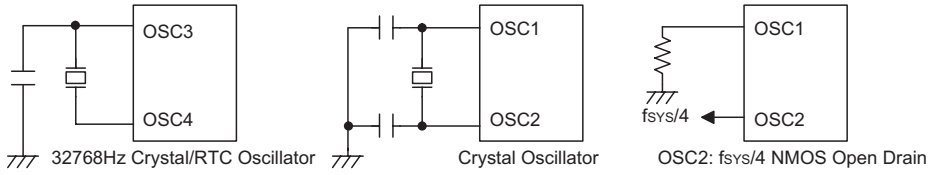
Interrupt Source	Priority	Vector
External Interrupt	1	04H
Timer/Event Counter Interrupt	2	08H
Real Time Clock Interrupt	3	0CH

The external interrupt request flag (EIF), real time clock interrupt request flag (RTF), timer/event counter request flag (TF), enable external interrupt bit (EEI), enable real time clock interrupt bit (ERTI), enable timer/event counter interrupt bit (ETI), and enable master interrupt bit (EMI) constitute an interrupt control register (INTC) which is located at 0BH in the data memory. EMI, EEI, ETI and ERTI are used to control the enabling/disabling of interrupts. These bits prevent the requested interrupt being serviced. Once the interrupt request flags (RTF, TF, EIF) are set, they remain in the INTC respectively until the interrupts are serviced or cleared by a software instruction.

It is recommended that a program does not use the "CALL subroutine" within the interrupt subroutine. Because 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, a CALL subroutine, if executed in the interrupt subroutine, will damage the original control sequence.

Oscillator Configuration

The HT47R10A-1/HT47C10-1 provides three oscillator circuits for system clocks, i.e., an RC oscillator, a crystal oscillator and a 32768Hz crystal oscillator, determined by configuration options. No matter what type of oscillator is selected, the signal is used for the system clock. The HALT mode stops the system oscillator (RC and crystal oscillator only) and ignores external signals to conserve power. The 32768Hz crystal system oscillator still runs during the Power Down mode. If the 32768Hz crystal oscillator is selected as the system oscillator, the system oscillator will not be stopped; however instruction execution will cease. Since the 32768Hz crystal



System Oscillator

oscillator (used as system oscillator or RTC oscillator) is also designed for timing purposes, the internal timing (RTC, time base, WDT) operation still runs even if the system enters the Power Down mode.

If the RC oscillator is used, an external resistor between OSC1 and ground is required, whose range should be between 24kΩ and 1MΩ. A frequency equal to the system clock divided by 4, is available on OSC2, which can be used for synchronisation purposes. As this is an open drain output, a pull-high resistor is required. The RC oscillator provides the most cost effective solution, however as its frequency of oscillation may vary with VDD, temperature and process variations, it is therefore not suitable for timing sensitive operations where accurate oscillator frequencies are desired.

On the other hand, if the crystal oscillator is selected, a crystal across OSC1 and OSC2 is needed to provide the feedback and phase shift required for oscillation, and no other external components are required. A resonator may be connected between OSC1 and OSC2 instead of the crystal to get a frequency reference, but two external capacitors connected between OSC1, OSC2 and ground are required.

Another oscillator circuit is designed for the real time clock, which has a fixed frequency of 32.768kHz. A 32.768kHz crystal should be connected between OSC3 and OSC4 for this function.

The RTC oscillator circuit can be controlled to start up quickly by clearing the QOSC bit, which is bit 4 in the RTCC register. At power on this bit will be low, allowing for fast start up, but it is recommended to set it high after around 2 seconds to conserve power.

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

Watchdog Timer – WDT

The WDT (f_S) clock source is implemented by a dedicated RC oscillator (WDT oscillator) or instruction clock (system clock divided by 4) or a real time clock oscillator (RTC oscillator), determined by configuration options. The timer is designed to prevent software malfunctions or a sequence jumping to an unknown location with unpredictable results. The Watchdog Timer can be disabled by a configuration option. If the Watchdog Timer is disabled, any instructions related to the WDT will result in no operation.

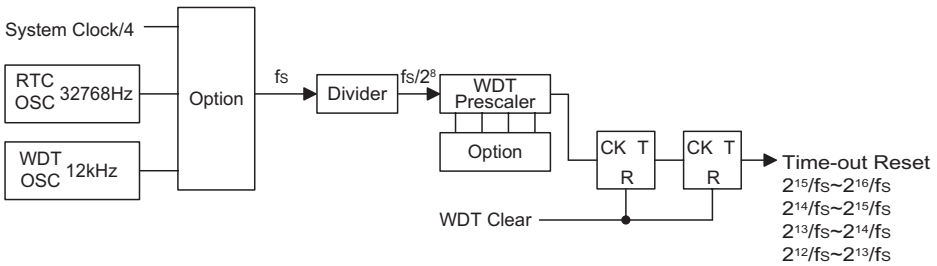
If the WDT clock source chooses the internal WDT oscillator, the time-out period may vary with temperature, VDD, and process variations. On the other hand, if the clock source selects the instruction clock and the "HALT" instruction is executed, the WDT will stop counting and lose its protecting purpose.

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

The WDT overflow under normal operation will initialise a "chip reset" and set the status bit TO. Whereas in the HALT mode, the overflow will initialise a "warm reset" in which only the Program Counter and Stack Pointer are reset to 0. To clear the WDT contents, three methods are adopted, an external reset (a low level to the $\overline{RES}$ pin), software instruction, or a "HALT" instruction.

There are two methods of using software instructions to clear the Watchdog Timer, one of which must be chosen by configuration option. The first option is to use the single "CLR WDT" instruction while the second is to use the two commands "CLR WDT1" and "CLR WDT2". For the first option, a simple execution of "CLR WDT" will clear the WDT while for the second option, both "CLR WDT1" and "CLR WDT2" must both be executed to successfully clear the WDT. Note that for this second option, if "CLR WDT1" is used to clear the WDT, successive executions of this instruction will have no effect, only the execution of a "CLR WDT2" instruction will clear the WDT. Similarly after the "CLR WDT2" instruction has been executed, only a successive "CLR WDT1" instruction can clear the Watchdog Timer.

The WDT time-out period ranges from $2^{15}/f_S$~$2^{16}/f_S$ since the clear Watchdog Timer instructions only clears the last two-stages of the WDT.



Watchdog Timer

Multi-function Timer

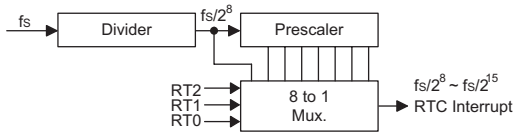
The HT47R10A-1/HT47C10-1 provides a multi-function timer for the WDT, time base and real time clock but with different time-out periods. The multi-function timer consists of an 8-stage divider and a 7-bit prescaler, with the clock source being sourced from the WDT OSC, RTC OSC or the instruction clock (i.e., system clock divided by 4). The multi-function timer also provides a selectable frequency signal (ranging from $f_S/2^2$ to $f_S/2^8$) for the LCD driver circuits, and a selectable frequency signal, ranging from $f_S/2^2$ to $f_S/2^9$, for the buzzer output chosen via configuration options. For proper LCD operation it is recommended to select a frequency as near as possible to 4kHz for the LCD driver circuits.

Real Time Clock – RTC

The real time clock or RTC operates in the same manner as the time base in that it is used to supply a regular internal interrupt. Its time-out period has a range between $f_S/2^8$ to $f_S/2^{15}$ whose actual value is chosen by software programming. Writing data to the RT2, RT1 and RT0 bits in the RTCC register, will provide various time-out periods. If an RTC time-out occurs, the related interrupt request flag, RTF- bit 6 of the INTC register, will be set. However if the interrupt is enabled, and the stack is not full, a subroutine call to location 0CH occurs. The real time clock time-out signal can also be utilised as a timer/event counter clock source, in order to get longer time-out periods.

RT2	RT1	RT0	Clock Divided Factor
0	0	0	2^{8}*
0	0	1	2^{9}*
0	1	0	2^{10}*
0	1	1	2^{11}*
1	0	0	2^{12}
1	0	1	2^{13}
1	1	0	2^{14}
1	1	1	2^{15}

Note: "*" not recommended for use



Real Time Clock

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 or RTC 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 will resume counting, if the WDT clock source is the WDT oscillator or the real time clock oscillator.
- All I/O ports maintain their original status.
- The PDF flag is set and the TO flag is cleared.
- The LCD driver keeps running if the appropriate Configuration option is chosen and if the WDT OSC or RTC OSC is selected.

The system can leave the Power Down mode by means of an external reset, an interrupt, an external falling edge signal on port A or a WDT overflow. An external reset causes a device initialisation and the WDT overflow executes a "warm reset". By examining the TO and PDF flags, the reason behind the chip reset can be determined. The PDF flag is cleared during a system power-up or executing a Clear Watchdog Timer instruction and is set when the "HALT" instruction is executed. The TO flag is set if a WDT time-out occurs, it causes a wake-up that only resets the Program Counter and SP, the others maintain 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 a configuration option. Awakening from an I/O port stimulus, the program will resume execution of the next instruction. If awakening 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. If the interrupt is enabled and the stack is not full, the regular interrupt response takes place.

If an interrupt request flag is set to "1" before entering the HALT mode the wake-up function of the related interrupt will be disabled.

Once a wake-up event occurs, it takes 1024 system clock periods before normal operation is resumed. In other words, a dummy period will be inserted after the wake-up. If the wake-up results from an interrupt acknowledgment, the actual interrupt subroutine execution is delayed by one more cycle. If the wake-up results in the next instruction execution, it will be executed immediately after the dummy period has finished.

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

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 is different from other chip reset conditions, since it performs a warm reset that only resets the Program Counter and Stack Pointer leaving the other circuits in their original state. Some registers remain unchanged during other reset conditions. Most registers are reset to their initial condition when the reset conditions are met. By examining the PDF and TO flags, the program can distinguish between different kinds of chip resets.

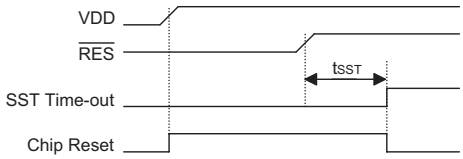
TO	PDF	RESET Conditions
0	0	$\overline{RES}$ reset during power-up
u	u	$\overline{RES}$ reset during normal operation
0	1	$\overline{RES}$ wake-up HALT
1	u	WDT time-out during normal operation
1	1	WDT wake-up HALT

Note: ″u″ means ″unchanged″.

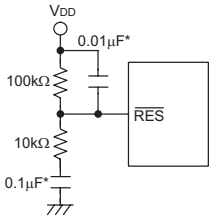
To guarantee that the system oscillator has started and stabilised, the SST (System Start-up Timer) provides an extra delay. There is an extra delay of 1024 system clock pulses when the system awakes from the Power Down mode or when the system powers up.

The functional unit chip reset status is shown below.

Program Counter	000H
Interrupt	Disabled
Prescaler, Divider	Cleared
WDT, RTC	Clear. After master reset, begin counting
Timer/Event Counter	Off
Input/output ports	Input mode
Stack Pointer	Points to the top of the stack

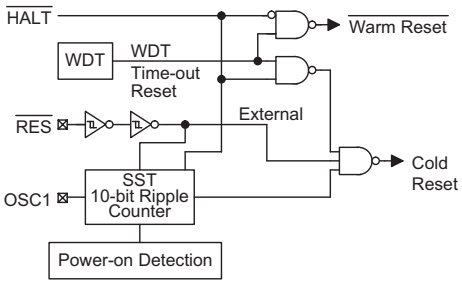


Reset Timing Chart



Reset Circuit

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



Reset Configuration

The registers states are summarised in the following table:

Register	Reset (Power-on)	WDT Time-out (Normal Operation)	$\overline{RES}$ Reset (Normal Operation)	$\overline{RES}$ Reset (HALT)	WDT Time-out (HALT)
MP0	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
MP1	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
BP	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
Program Counter	000H	000H	000H	000H	000H*
TBLP	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLH	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
RTCC	--00 0111	--00 0111	--00 0111	--00 0111	--uu uuuu
STATUS	--00 xxxx	--1u uuuu	--uu uuuu	--01 uuuu	--11 uuuu
INTC	-000 0000	-000 0000	-000 0000	-000 0000	-uuu uuuu
PA	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
TMRAH	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMRAL	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMRC	0000 1---	0000 1---	0000 1---	0000 1---	uuuu u---
TMRBH	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMRBL	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
ADCR	1xxx --00	1xxx --00	1xxx --00	1xxx --00	uuuu --uu

Note: "*" refers to warm reset
"u" means unchanged
"x" means unknown

Timer/Event Counter

One 16-bit timer/event counter or one single channel RC type A/D converter is implemented in the HT47R10A-1/HT47C10-1. The ADC/TM bit, which is bit 1 of the ADCR register, determines whether Timer A and Timer B is composed of one 16-bit timer/event counter or Timer A and Timer B is composed of a single channel RC type A/D converter.

The TMRAL, TMRAH, TMRBL, TMRBH registers constitute one 16-bit timer/event counter, when the ADC/$\overline{TM}$ bit is "0". The TMRBL and TMRBH registers are timer/event counter preload registers for the lower-order byte and higher-order byte respectively.

Using the internal clock, there are three reference time bases. The timer/event counter internal clock source may come from the system clock/RTC OSC, the system clock/4 or the RTC time-out signal to generate an accurate time base.

Using an external clock input allows external events to be counted, to count external RC type A/D clocks, measure time intervals or pulse widths or to generate an accurate time base.

There are six registers related to the timer/event counter operating mode. TMRAH ([20H]), TMRAL ([21H]), TMRC ([22H]), TMRBH ([23H]), TMRBL ([24H]) and ADCR ([25H]). Reading and writing to the timer/event counter must be conducted in a specific way. It is important to note that writing to the TMRBL register only writes the data into a low byte buffer and not into the timer preload register. However writing to the TMRBH register will write the high byte data, as well as the contents of the low byte buffer, into the time/event counter preload register simultaneously. The timer/event counter preload register is therefore only modified by TMRBH write operations, while TMRBL write operations keep the timer/event counter preload register unchanged.

Reading the TMRAH register will also latch the TMRAL data into the low byte buffer to avoid false timing problems. Reading the TMRAL only returns the contents of the low byte buffer. In other words, the low byte of the timer/event counter cannot be read directly. It must be read by first reading the TMRAH register first to transfer the low byte contents of the timer/event counter into the buffer.

If the timer/event counter is running, the TMRAH, TMRAL, TMRBH and TMRBL registers cannot be read or written to. To avoid an overlap between Timer A and Timer B, the TMRAH, TMRAL, TMRBH and TMRBL registers should be accessed with the "MOV" instruction when the timer is not running.

The TMRC register is the timer/event counter control register, which defines the timer/event counter options.

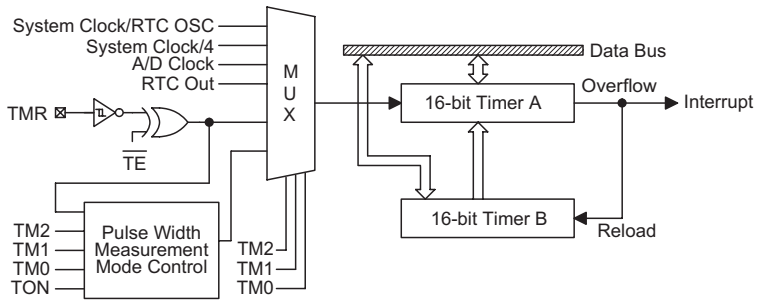
The timer/event counter control register defines the operating mode, counting enable or disable and the active edge.

Writing to Timer B places the timer start value into the timer/event counter preload register, while reading Timer A provides the contents of the timer/event counter. Timer B is a timer/event counter preload register.

The TM0, TM1 and TM2 bits define the operation mode. The event count mode is used to count external events, which are sourced on the external pin, TMR. The A/D clock mode is used to count external A/D clocks, the RC oscillation mode is determined by the ADCR register. 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 measure the high or low level duration of an external signal on pin TMR. The counting is based on the instruction clock.

In the event counting mode, the A/D clock or internal timer mode, once the timer/event counter starts counting, it will count from its current contents in the timer/event counter (TMRAH and TMRAL) to FFFFH. Once an overflow occurs, the counter is reloaded from the timer/event counter preload registers, TMRBH and TMRBL, and at the same time generates a corresponding interrupt request flag, which is TF, bit 5 in the INTC register.

In the pulse width measurement mode, with the TON and TE bits equal to 1, once the TMR pin has received a transient from low to high (or high to low if the TE bit is 0) it will start counting until the TMR pin returns to its original level and resets the TON bit. The measured result will remain in the timer/event counter even if the activated transient occurs again. In other words, only one cycle measurement can be made. The TON bit has to be set again by the program if further measurements are to be made. Note that in this operation mode, the timer/event counter starts counting not according to the logic level but according to the transient edges. In the case of counter overflows, the counter is reloaded from the timer/event counter preload register and issues an interrupt request just like the other three modes.



Timer/Event Counter

Bit No.	Label	Function
0~2	—	Unused bit, read as "0"
3	TE	Defines the TMR active edge of the timer/event counter (0=active on low to high edge; 1=active on high to low edge)
4	TON	Enable or disable timer counting (0=disable; 1=enable)
5 6 7	TM0 TM1 TM2	Defines the operating mode (TM2, TM1, TM0) 000=Timer mode (system clock/RTC OSC). The system clock or RTC OSC is selected as the timer source by configuration option. 001=Timer mode (system clock/4) 010=Timer mode (RTC output) 011=A/D clock mode (RC oscillation decided by ADCR register) 100=Event counter mode (external clock) 101=Pulse width measurement mode (system clock/4) 110=Unused 111=Unused

TMRC (22H) Register

To enable the counting operation, the timer TON bit, should be set to 1. In the pulse width measurement mode, TON will be automatically cleared after the measurement cycle is completed. But in the other three modes, the TON can only be reset by instructions. The timer/event counter overflow is one of the wake-up sources.

If the timer/event counter is not running, 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 preload register is kept only in the timer/event counter preload register. The timer/event counter will continue to operate normally until an overflow occurs, at which point the new data will be transferred to the timer .

When the timer/event counter (reading TMRAH) is read, the clock will be blocked to avoid errors. As this may result in a counting error, this must be taken into consideration by the programmer.

It is strongly recommended to load first the desired value for TMRBL, TMRBH, TMRAL, and TMRAH registers, before turning on the related timer/event counter to ensure proper operation. This is because the initial values of TMRBL, TMRBH, TMRAL and TMRAH are unknown.

If the timer/event counter is on, the TMRAH, TMRAL, TMRBH and TMRBL registers cannot be read or written to. Only when the timer/event counter is off and when the instruction "MOV" is used can these four registers be read or written to.

Timer/event counter mode example (disable interrupt):

```
clr tmrc
clr adcr.1                  ; set timer mode
clr intc.5                  ; clear timer/event counter interrupt request flag
mov a, low (65536-1000)     ; give timer initial value
mov tmrbl, a                ; count 1000 time and then overflow
mov a, high (65536-1000)
mov tmrbh, a

mov a, 00110000b            ; timer clock source is fSYS/4 and timer on
mov tmrc, a

p10:
clr wdt
snz intc.5                  ; polling timer/event counter interrupt request flag
jmp p10
clr intc.5                  ; clear timer/event counter interrupt request flag
                            ; program continue
```

A/D Converter

A single channel RC type A/D converter is implemented in the HT47R10A-1/HT47C10-1. The A/D converter contains two 16-bit programmable count-up counters. The Timer A clock source is sourced from the system clock/RTC OSC, instruction clock or RTC output. The Timer B clock source is sourced from the external RC oscillator. The TMRAL, TMRAH, TMRBL, TMRBH registers will form an A/D converter when the ADC/$\overline{TM}$, which is bit 1 of the ADRC register, is set to ″1″.

The A/D converter Timer B clock source may come from the IN external clock input pin, RS~CS oscillation, RT~CS oscillation, CRT~CS oscillation (CRT is a resistor). The Timer A clock source is sourced from the system clock/RTC OSC, instruction clock or RTC prescaler clock output determined by the TMRC register.

There are six registers related to the A/D converter, i.e., TMRAH, TMRAL, TMRC, TMRBH, TMRBL and ADCR. The internal timer clock is the input to TMRAH and TMRAL, the A/D clock is the input to TMRBH and TMRBL. The OVB/$\overline{OVA}$ bit, which is bit 0 of the ADCR register, decides whether Timer A overflows or Timer B overflows. When this occurs, the TF bit is set and a timer interrupt occurs. When the A/D converter mode Timer A or Timer B overflows, the TON bit is reset and the timer stops counting. Writing to TMRAH/TMRBH places the start value in Timer A/Timer B and reading TMRAH/TMRBH retrieves the contents of Timer A/Timer B. Writing to TMRAL/TMRBL only writes the data into a low byte buffer. Writing to TMRAH/TMRBH will write the data and the contents of the low byte buffer into the Timer A/Timer B (16-bit) simultaneously. The Timer A/Timer B is changed by writing to TMRAH/TMRBH operations while writing to TMRAL/TMRBL operations will keep Timer A/Timer B unchanged.

Reading TMRAL/TMRBH will also latch TMRAL/TMRBL into the low byte buffer to avoid false timing problems. Reading TMRAL/TMRBL only returns the contents of the low byte buffer and not the actual timer value. Therefore the low byte of Timer A/Timer B cannot be read directly. It must first read TMRAH/TMRBH to transfer the low byte contents of Timer A/Timer B into the buffer.

If the A/D converter Timer A and Timer B are counting, the TMRAH, TMRAL, TMRBH and TMRBL cannot be read or written to. To avoid an overlap between Timer A and Timer B, the TMRAH, TMRAL, TMRBH and TMRBL registers should be accessed with the ″MOV″ instruction when Timer A and Timer B are not running.

Bits 4~7 of the ADCR register decides which resistor and capacitor comprise an oscillation circuit and input to TMRBH and TMRBL.

The TM0, TM1 and TM2 bits of TMRC define the Timer A clock source. It is recommended that the clock source of Timer A uses the system clock/RTC OSC, instruction clock or the RTC prescaler clock.

If the TON bit is set to ″1″ then Timer A and Timer B will start counting until Timer A or Timer B overflows. The timer/event counter will then generate an interrupt request flag, TF - bit 5 of INTC, and the Timer A and Timer B will stop counting and reset the TON bit to ″0″ at the same time.

If the TON bit is ″1″, TMRAH, TMRAL, TMRBH and TMRBL cannot be read or written to. Only when the timer/event counter is off and when the instruction ″MOV″ is used can these four registers be read or written to.

Bit No.	Label	Function
0	OVB/$\overline{OVA}$	In the RC type A/D converter mode, this bit is used to define the timer/event counter interrupt which comes from Timer A overflow or Timer B overflow. (0= Timer A overflow; 1= Timer B overflow) In the timer/event counter mode, this bit is void.
1	ADC/$\overline{TM}$	Defines 16 timer/event counters or RC type A/D converter is enabled. (0= timer/event counter enable; 1= A/D converter is enabled)
2~3	—	Unused bit, read as ″0″.
4 5 6 7	M0 M1 M2 M3	Defines the A/D converter operating mode (M3, M2, M1, M0) 0000= IN external clock input mode 0001= RS~CS oscillation (reference resistor and reference capacitor) 0010= RT~CS oscillation (resistor sensor and reference capacitor) 0011= CRT~CS oscillation (resistor sensor and reference capacitor) 0100= RS~CRT oscillation (reference resistor and sensor capacitor) 0101= Unused mode 0110= Unused mode 0111= Unused mode 1XXX= Unused mode

ADCR (25H) Register

RC type AD converter mode example (Timer A overflow):

```
clr tmrc
clr adcr.1                  ; set timer mode
clr intc.5                  ; clear timer/event counter interrupt request flag
mov a, low (65536-1000)     ; give Timer A initial value
mov tmrbl, a                ; count 1000 time and then overflow
mov a, high (65536-1000)
mov tmrbh, a

mov a, 00010010b            ; RS~CS; set RC type ADC mode; set Timer A overflow
mov adcr,a
mov a, 00h                  ; give Timer B initial value
mov tmrbl, a
mov a, 00h
mov tmrbh, a

mov a, 00110000b            ; Timer A clock source is fSYS/4 and timer on
mov tmrc, a

p10:
clr wdt
snz intc.5                  ; polling timer/event counter interrupt request flag
jmp p10
clr intc.5                  ; clear timer/event counter interrupt request flag
                            ; program continue
```

Example for RC type AD converter mode (Timer B overflow):

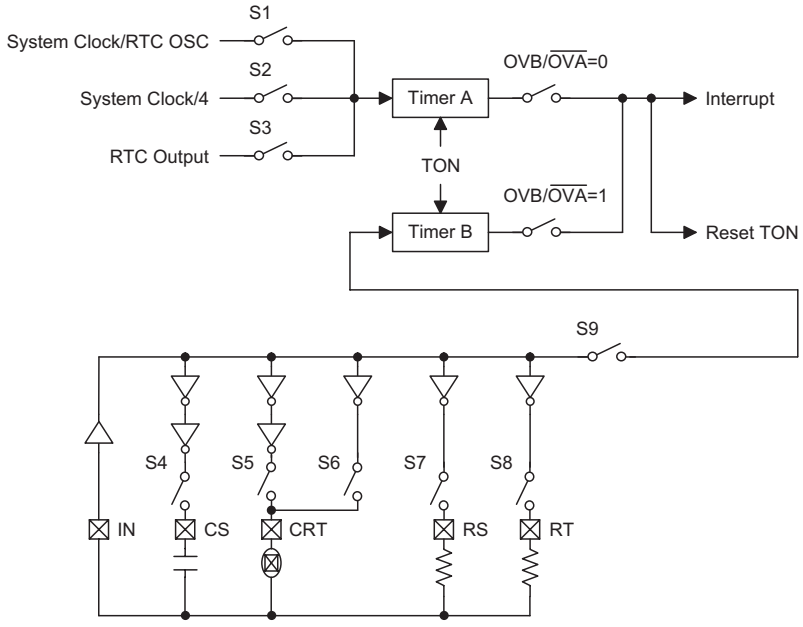
```
clr tmrc
clr adcr.1                  ; set timer mode
clr intc.5                  ; clear timer/event counter interrupt request flag
mov a, 00h                  ; give Timer A initial value
mov tmrbl, a
mov a, 00h
mov tmrbh, a

mov a, 00010011b            ; RS~CS; set RC type ADC mode; set Timer B overflow
mov adcr, a

mov a, low (65536-1000)     ; give Timer B initial value
mov tmrbl, a                ; count 1000 time and then overflow
mov a, high (65536-1000)
mov tmrbh, a

mov a, 00110000b            ; Timer A clock source is fSYS/4 and timer on
mov tmrc, a

p10:
clr wdt
snz intc.5                  ; polling timer/event counter interrupt request flag
jmp p10
clr intc.5                  ; clear timer/event counter interrupt request flag
                            ; program continue
```



TN2	TN1	TN0	S1	S2	S3
0	0	0	1	0	0
0	0	1	0	1	0
0	1	0	0	0	1
Other			0	0	0

Note: 0=off, 1=on

M3	M2	M1	M0	S4	S5	S6	S7	S8	S9
0	0	0	0	0	0	0	0	0	1
0	0	0	1	1	0	0	1	0	1
0	0	1	0	1	0	0	0	1	1
0	0	1	1	1	0	1	0	0	1
0	1	0	0	0	1	0	1	0	1
Other				0	0	0	0	0	0

Note: 0=off, 1=on

RC Type A/D Converter

Input/Output Ports

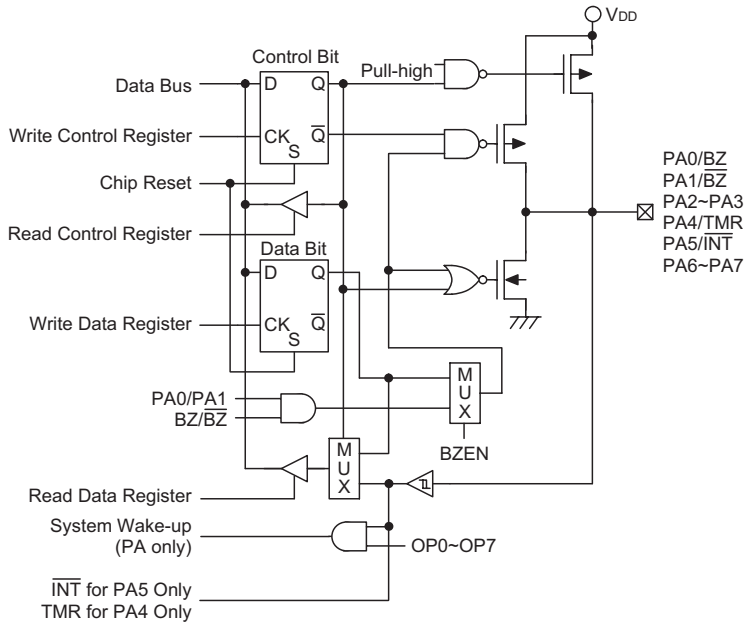
There are 8 bidirectional input/output lines in the microcontroller, confined into a single port known as PA, which is mapped to the data memory at [12H]. These I/O lines can be used for input and output operations. For input operation, these lines are non-latching, that is, these inputs must be ready at the T2 rising edge of instruction "MOV A,[m]" (m=12H). For output operation, all the data is latched and remains unchanged until the output latch is rewritten. Port PA, has its own Port Control Register known as PAC, which controls the input/output configuration of each I/O line.

With this control register, each I/O pin can be configured to be either a CMOS output or a Schmitt trigger input. Configuration options exist to connect pull-high resistors to the inputs. The I/O pins can be reconfigured dynamically (i.e. on-the-fly) under software control. To function as an input, the corresponding latch of the control register must be written with 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 pin will be setup as an output and the contents of the latches will move to the internal bus.

The latter is possible in the "read-modify-write" instruction. For an output function, CMOS is the only configuration. These control register is mapped to locations 13H. After a chip reset,the I/Os default to an input condition and will remain at a high level, or floating state, depending upon the pull-high configuration options. Each bit of these input/output latches can be set or cleared by "SET [m].i" and "CLR [m].i" (m=12H) instructions. Some instructions first input data and then follow the output operations. For example, "SET [m].i", "CLR [m].i", "CPL [m]", "CPLA [m]" read the entire port states into the CPU, execute the defined operations (bit-operation), and then write the results back to the latches or the accumulator.

Each line of port A has the capability of waking-up the device. Each I/O pin has a pull-high option where individual pull-high resistors can be connected to each pin. Note that a non- pull-high input will result in a floating state.

The PA0, PA1, PA5, PA4 are pin-shared with BZ, $\overline{BZ}$, $\overline{INT}$, TMR pins, respectively.



Input/Output Ports

The PA0 and PA1 are pin-shared with the BZ and $\overline{BZ}$ pinsl, respectively. If the BZ/$\overline{BZ}$ option is selected, the output signal, if PA0/PA1 are setup as outputs, will be the buzzer signal generated by the Multi-function timer. If setup as inputs these pins will retain their I/O operation status. Once the BZ/$\overline{BZ}$ option is selected, the buzzer output signals are controlled by the PA0/PA1 data register only.

The PA0/PA1 I/O functions are shown below.

PA0 I/O	I	I	O	O	O	O	O	O	O	O
PA1 I/O	I	O	I	I	I	O	O	O	O	O
PA0 Mode	x	x	C	B	B	C	B	B	B	B
PA1 mode	x	C	x	x	x	C	C	C	B	B
PA0 Data	x	x	D	0	1	D_0	0	1	0	1
PA1 Data	x	D	x	x	x	D_1	D	D	x	x
PA0 Pad Status	I	I	D	0	B	D_0	0	B	0	B
PA1 Pad Status	I	D	I	I	I	D_1	D	D	0	B

Note: ″I″ input,″O″ output, ″D, D0, D1″ data

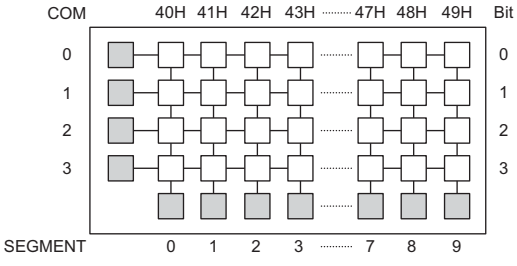
″B″ buzzer option, BZ or $\overline{BZ}$, ″x″ don′t care

″C″ CMOS output

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 if in floating input states.

LCD Display Memory

The HT47R10A-1/HT47C10-1 provides an area of embedded data memory for the LCD display. The LCD display memory has a structure of 10×4 bits. Note that if the configuration options select the LCD to have 9×4 segment outputs, then the 49H address area of the LCD display memory cannot be accessed. The LCD data memory area is located from 40H to 49H in Bank 1of the Data Memory. The bank pointer, BP- located at 04H of the data memory, will switch between the general purpose data memory and the LCD display memory. When the BP is set to the value ″01H″ any data written into the area 40H~49H will effect the LCD display. Data must be written indirectly using MP1. When BP is cleared to ″00H″, any data written into the area 40H~49H will access the general purpose data memory. The LCD display memory can be read and written to only using the indirect addressing mode via MP1. When data is written into the display data area, it is automatically read by the LCD driver which then generates the corresponding LCD driving signals. To turn the display on or off, a ″1″ or a ″0″ is written to the corresponding bit of the display memory, respectively. The figure illustrates the mapping between the display memory and LCD pattern for the HT47R10A-1/HT47C10-1.

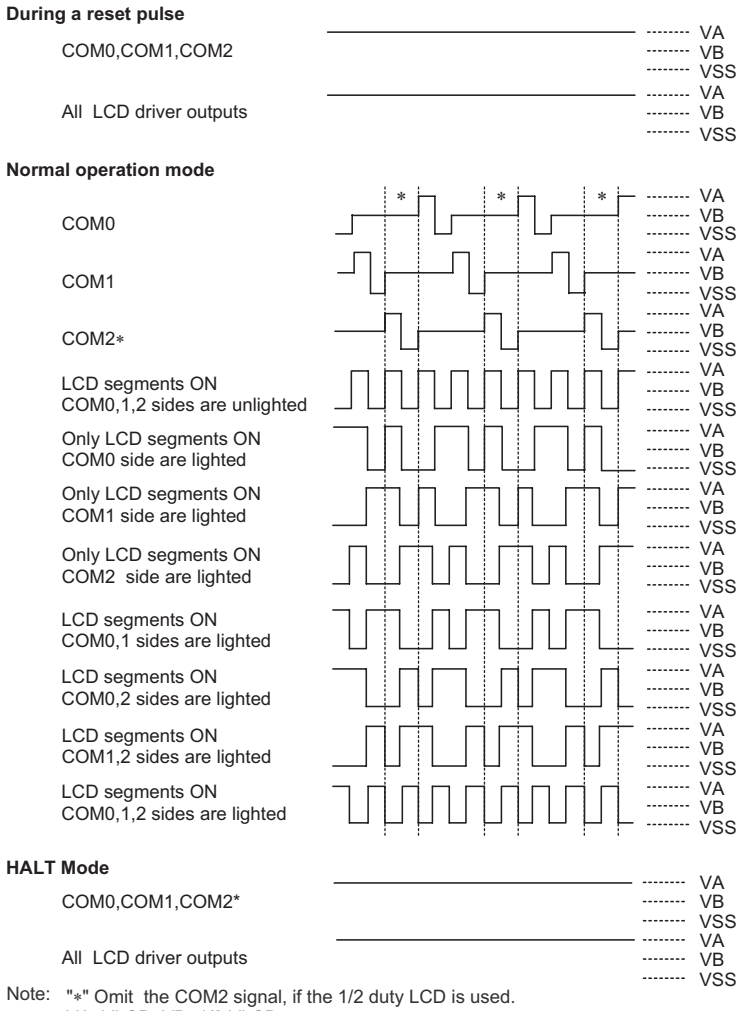


Display Memory (Bank 1)

LCD Driver Output

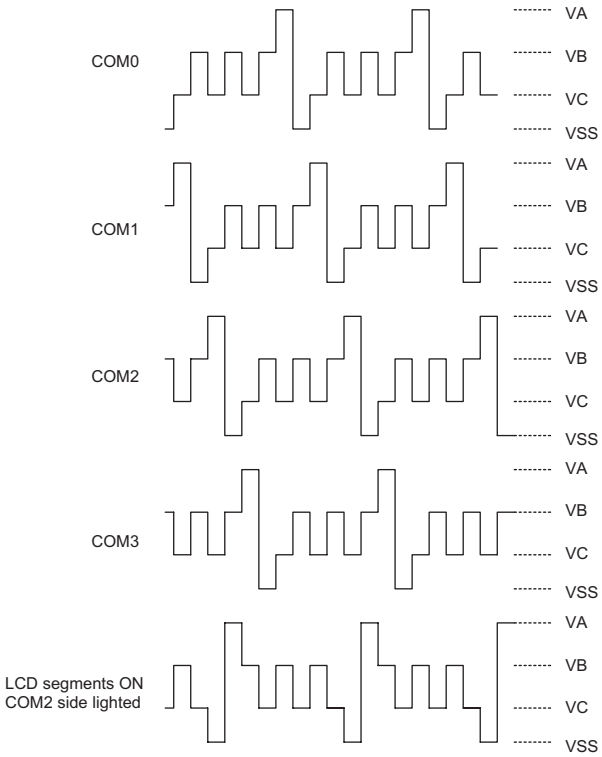
The LCD output number of the HT47R10A-1/ HT47C10-1 LCD driver can be 10×2, 10×3 or 9×4, the choice of which is chosen via a configuration option, i.e., 1/2 duty, 1/3 duty or 1/4 duty.

The bias type of the LCD driver can be C- type or R- type. For C-type biasing, a capacitor connected between C1 and C2 pins is needed. The bias voltage of the LCD driver can be either 1/2 bias or 1/3 bias, chosen via a configuration option. If 1/2 bias is selected, a capacitor mounted between the V2 pin and ground is required. If 1/3 bias is selected, two capacitors are needed on each of the V1 and V2 pins. Refer to the application diagram. If the "R" bias type is selected, no external capacitors are required.



Note: "*" Omit the COM2 signal, if the 1/2 duty LCD is used.
VA=VLCD, VB=1/2 VLCD

LCD Driver Output (1/3 Duty, 1/2 Bias, R/C Type)



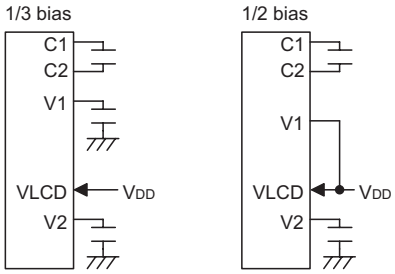
Note: 1/4 duty, 1/3 bias, C type: "VA" 3/2 VLCD, "VB" VLCD, "VC" 1/2 VLCD
1/4 duty, 1/3 bias, R type: "VA" VLCD, "VB" 2/3 VLCD, "VC" 1/3 VLCD

LCD Driver Output

Low Voltage Reset/Detector Functions

There is a low voltage detector, LVD, and a low voltage reset circuit, LVR, implemented in the microcontroller. These two functions can be enabled/disabled by configuration options. Once the configuration options for the LVD is enabled, bit RTCC.3 can be used to enable or disable the LVD circuit. The LVD detector status can be monitored via bit RTCC.5.

The LVR has the same effect or function as the external $\overline{RES}$ signal which performs a chip reset. During the Power Down mode the LVR is disabled.



V1, V2, VLCD Application Diagram C Type)

The RTCC register definitions are listed in the following table.

Bit No.	Label	Read/Write	Function
0~2	RT0~RT2	R/W	8 to 1 multiplexer control inputs to select the real time clock prescaler output
3	LVDC*	R/W	LVD enable/disable (1/0)
4	QOSC	R/W	32768Hz OSC quick start-up oscillating function 0/1: quick/slow start
5	LVDO	R	LVD detection output (1/0) 1: low voltage detected
6~7	—	—	Unused bit, read as ″0″

RTCC (09H) Register

Buzzer

The Buzzer function provides a means of producing a variable frequency output, suitable for applications such as Piezo-buzzer driving or other external circuits that require a precise frequency generator. The BZ and $\overline{BZ}$ pins form a complimentary pair, and are pin-shared with I/O pins, PA0 and PA1. A configuration option is used to select from one of the three buzzer options. The first option is for both pins PA0 and PA1 to be used as normal I/Os, the second option is for both pins to be configured as BZ and $\overline{BZ}$ buzzer pins, the third option selects only the PA0 pin to be used as a BZ buzzer pin with the PA1 pin retaining its normal I/O pin function. Note that the $\overline{BZ}$ pin is the inverse of the BZ pin which when together generate a differential output that can supply more power to connected interfaces such as buzzers.

The buzzer is driven by the internal clock source, f_S, which then passes through a divider, the division ratio of which is selected by configuration options to provide a range of buzzer frequencies from $f_S/2^2$ to $f_S/2^9$. The clock source that generates f_S, which in turn controls the buzzer frequency, can originate from three different sources, the RTC oscillator, the WDT oscillator or the system clock/4, the choice of which is determined by the f_S clock source configuration option. It is important to note that if the RTC oscillator is selected as the system clock, then f_S, and correspondingly the buzzer, will also have the RTC oscillator as its clock source. Note that the buzzer frequency is controlled by configuration options, which select both the source clock for the internal clock f_S and the internal division ratio. There are no internal registers associated with the buzzer frequency.

If the configuration options have selected both pins PA0 and PA1 to function as a BZ and $\overline{BZ}$ complementary pair of the buzzer outputs, then for correct buzzer operation it is essential that both pins must be setup as outputs by setting bits PAC0 and PAC1 of the PAC port control register to zero. The PA0 data bit in the PA data register must also be set high to enable the buzzer outputs, if set low, both pins PA0 and PA1 will remain low. In this way the single bit PA0 of the PA register can be used as an on/off control for both the BZ and $\overline{BZ}$ buzzer pin outputs. Note that the PA1 data bit in the PA register has no control over the $\overline{BZ}$ buzzer pin PA1.

If configuration options have selected that only the PA0 pin is to function as a BZ buzzer pin, then the PA1 pin can be used as a normal I/O pin. For the PA0 pin to function as a BZ buzzer pin, PA0 must be setup as an output by setting bit PAC0 of the PAC port control register to zero. The PA0 data bit in the PA data register must also be set high to enable the buzzer output, if set low, pin PA0 will remain low. In this way the PA0 bit can be used as an on/off control for the BZ buzzer pin PA0. If the PAC0 bit of the PAC port control register is set high, then pin PA0 can still be used as an input even though the configuration option has configured it as a BZ buzzer output.

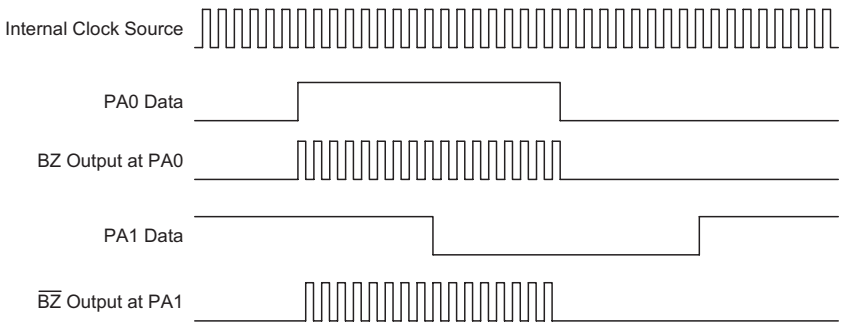
PAC Register PAC0	PAC Register PAC1	PA Data Register PA0	PA Data Register PA1	Output Function
0	0	0	x	PA0=0 PA1=0
0	0	1	x	PA0=$\overline{BZ}$ PA1=$\overline{BZ}$
0	1	0	x	PA0=0 PA1=Input line
0	1	1	x	PA0=BZ PA1=Input line
1	0	1	x	PA0=Input line PA1=$\overline{BZ}$
1	0	0	x	PA0=Input line PA1=0
1	1	x	x	PA0=Input line PA1=Input line

PA0/PA1 Pin Function Control

Note: ″x″ stand for don′t care

″D″ stand for data ″0″ or ″1″

Note that no matter what configuration option is chosen for the buzzer, if the port control register has setup the pin to function as an input, then this will override the configuration option selection and force the pin to always function as an input pin. This arrangement enables the pin to be used as both a buzzer pin and as an input pin, so regardless of the configuration option chosen; the actual function of the pin can be changed dynamically by the application program by programming the appropriate port control register bit.



Buzzer Output Pin Control

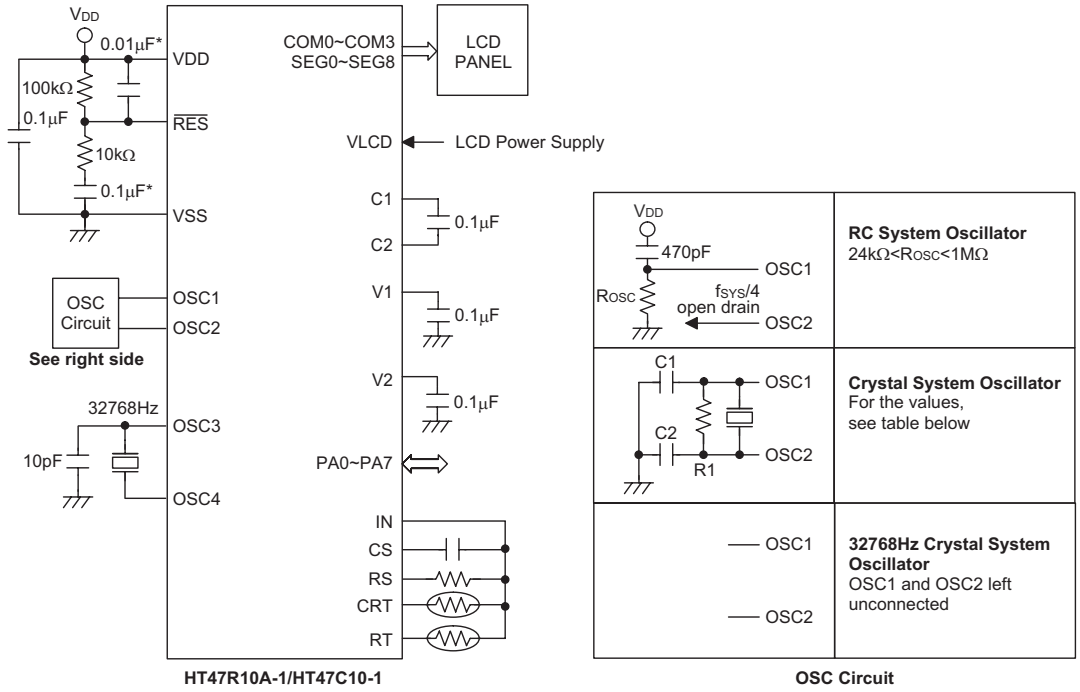
Note: The above diagram shows the situation where both pins PA0 and PA1 are selected by configuration option to be BZ and $\overline{BZ}$ buzzer output pins. The Port Control Register of both pins must have already been setup as output. The data setup on pin PA1 has no effect on the buzzer outputs.

Option

The following shows the various options in the HT47R10A-1/HT47C10-1. All these options should be defined in order to ensure having a properly functioning system.

No.	Option
1	OSC type selection. This option is to determine if an RC, a crystal oscillator or an RTC oscillator is chosen as system clock.
2	Clock source selection for WDT, RTC and Time Base. There are three types of selection: system clock/4 or RTC OSC or WDT OSC.
3	WDT enable or disable selection. WDT can be enabled or disabled.
4	CLR WDT times selection. This option defines how to clear the WDT by instruction. One time means that the ″CLR WDT″ can clear the WDT. ″Two times″ means that only if both of the ″CLR WDT1″ and ″CLR WDT2″ have been executed, then WDT can be cleared.
5	Buzzer output frequency selection. There are eight types of frequency signals for the buzzer output: $f_S/2^2$~$f_S/2^9$. ″f_S″ means the WDT clock source.
6	Wake-up selection. This option defines the wake-up function activity. External I/O pins all have the capability to wake-up the chip from a HALT mode by a following edge.
7	Pull-high selection. This option is to determine whether the pull-high resistance is viable or not on the each bits of PA.
8	I/O pins share with other function selection. PA0/BZ, PA1/$\overline{BZ}$: PA0 and PA1 can be set as I/O pins or buzzer outputs.
9	LCD common selection. There are three types of selection: 2 common (1/2 duty) or 3 common (1/3 duty) or 4 common (1/4 duty). If the 4 common is selected, the segment output pin COM3/SEG9 will be set as a common output.
10	LCD driver clock selection. There are seven types of frequency signals for the LCD driver circuits: $f_S/2^2$~$f_S/2^8$. ″f_S″ means the WDT clock source.
11	LCD on or LCD off at the HALT mode selection. The LCD can be enabled or disabled at the HALT mode.
12	LVD enable or disable
13	LVR enable or disable
14	System clock or RTC OSC selection. The timer/event counter source is from system clock or from RTC OSC in timer mode, and Timer A source is from system clock or RTC OSC in the A/D mode.

Application Circuits



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

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

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

Crystal or Resonator	C1, C2	R1
4MHz Crystal	0pF	14kΩ
4MHz Resonator	10pF	12kΩ
3.58MHz Crystal	0pF	15kΩ
3.58MHz Resonator	25pF	10kΩ
2MHz Crystal & Resonator	25pF	12kΩ
1MHz Crystal	35pF	12.5kΩ
480kHz Resonator	300pF	9.1kΩ
455kHz Resonator	300pF	10kΩ
429kHz Resonator	300pF	10kΩ
400kHz Resonator	300pF	10kΩ
The function of the resistor R1 is to ensure that the oscillator will switch off should low voltage conditions occur. Such a low voltage, as mentioned here, is one which is less than the lowest value of the MCU operating voltage.		

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.

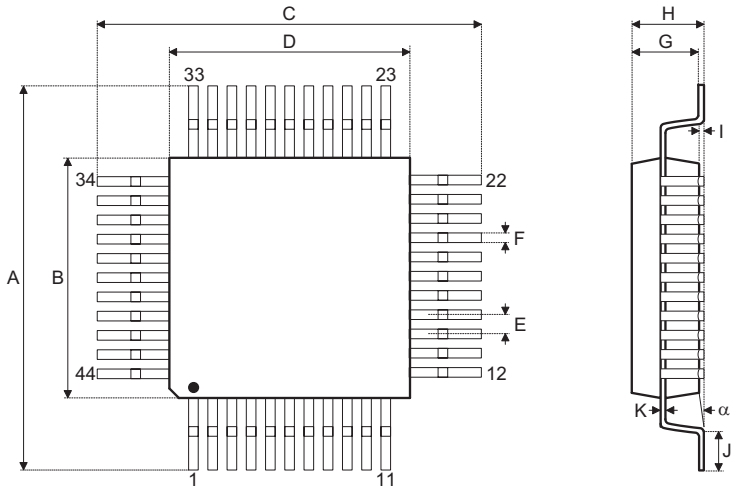
Operation ACC ← ACC ″XOR″ x

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

Package Information

44-pin QFP (10×10) Outline Dimensions



Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	13	—	13.4
B	9.9	—	10.1
C	13	—	13.4
D	9.9	—	10.1
E	—	0.8	—
F	—	0.3	—
G	1.9	—	2.2
H	—	—	2.7
I	—	0.1	—
J	0.73	—	0.93
K	0.1	—	0.2
α	0°	—	7°

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