Application Note 52

Using the Dallas Phantom Real Time Clocks

DESCRIPTION

The Dallas Phantom Real Time Clocks are a family of devices that offer the combination of a transparent CMOS timekeeper and a nonvolatile static RAM meeting the standard JEDEC bytewide pinouts. Some varieties of the Dallas Phantom Real Time Clocks also provide a transparent CMOS timekeeper for use with ROM. The timekeeper is transparent to the RAM/ROM memory map because it does not occupy any of the existing RAM/ROM locations. These devices are termed “Phantom” because the timekeeper is accessed only when a predetermined 64–bit pattern has been received by the device. When the timekeeper is not being accessed, the RAM/ROM can be accessed normally. The timekeeper keeps track of hundredths of seconds, seconds, minutes, hours, day, date, month, and year information. In the absence of power, a lithium energy source maintains the timekeeping operation and retains data in the CMOS static RAM.

FAMILY OVERVIEW

DS1215

The heart of the Dallas Phantom Real Time Clock family is the DS1215 Phantom Time Chip. This integrated circuit is a combination of a CMOS timekeeper and a nonvolatile memory controller. In the absence of power, an external battery maintains the timekeeping operation and retains data in the CMOS static RAM. The watch keeps track of hundredths of seconds, seconds, minutes, hours, day, date, month, and year information. The last day of the month is automatically adjusted for months with less than 31 days, including correction for leap year every four years. The real time clock operates in one of two formats: 12–hour mode with an AM/PM indicator or a 24–hour mode. The nonvolatile controller supplies all the necessary support circuitry to convert a CMOS RAM to a nonvolatile memory. The DS1215 can also be used to provide timekeeping functions with ROM.

DS1216

Stemming from the DS1215 are the DS1216 SmartWatch Intelligent Sockets. The SmartWatch is a 600-mil wide DIP socket with a built–in DS1215 (providing timekeeping functions and a nonvolatile RAM controller), an embedded lithium energy source, and a 32.768 kHz crystal. When the socket is mated with a bytewide CMOS static RAM, it provides a complete solution to problems associated with memory volatility and uses a common energy source to maintain time and date. The DS1216 can also be mated with a ROM to provide timekeeping capability only. Figures 1 and 2 show the basic interface of a SmartWatch with RAM inserted and a SmartWatch with ROM inserted, respectively.

DS1243Y, DS1244Y, DS1248Y

The DS124x Nonvolatile SRAM with Phantom Time Clock modules are the final members of the Dallas Phantom Real Time Clock family. These devices are fully nonvolatile static RAM with a built–in Phantom clock, embedded lithium energy source, and 32.768 kHz crystal. These devices operate identical to a DS1216 with a RAM inserted. The DS124x Nonvolatile SRAM with Phantom Time Clock modules will maintain over 10 years of data retention in the absence of power.

Perhaps the best way to sum up the Dallas Phantom Real Time Clock family is as follows. The DS1215 Phantom Time Chip is the basic building block that provides timekeeping and a nonvolatile memory controller. The DS1216 then adds a crystal and lithium energy source to the DS1215 and encapsulates them all in a socket that will accept either a RAM or a ROM. Finally, the DS124x modules contain both nonvolatile RAM and timekeeping features in a ready–to–use package.

The entire Dallas Phantom Real Time Clock family is shown in Table 1.

Table 1

DS1215	Phantom Time Chip
DS1216B	SmartWatch/RAM 16K/64K
DS1216C	SmartWatch/RAM 64K/256K
DS1216D	SmartWatch/RAM 256K/1M
DS1216E	SmartWatch/ROM 64K/256K
DS1216F	SmartWatch/ROM 64K/256K/1M
DS1243Y	64K NV SRAM with Phantom Clock
DS1244Y	256K NV SRAM with Phantom Clock
DS1248Y	1024K NV SRAM with Phantom Clock

APPLICATION

The Dallas Phantom Real Time Clock family offers two features that will greatly enhance a system. The first feature is nonvolatile RAM capability. The second feature is that the Phantom Clock is transparent to the RAM and therefore timekeeping capacity can be added to a system without changing the existing hardware. All that is required is an existing bytewide memory socket. The retrofit capability is maximized through the transparent interfaces supported by the Phantom Time Chip. Also advantageous to the designer is that an upgrade path is provided to higher RAM densities with the DS124x modules or to higher RAM/ROM densities with the DS1216.

It should be mentioned that there is some software overhead that is associated with having timekeeping functions that are transparent to RAM as will be discussed in detail below. If it is determined that a transparent clock is not necessary, then the DS164x Nonvolatile Timekeeping RAM family could offer an excellent solution to your timekeeping and nonvolatile SRAM needs. These offer nonvolatile SRAM with the Real Time Clock registers located in the RAM address space. Another possible solution is the DS1386 or DS1486 RAMified Watchdog Timekeepers which offer nonvolatile RAM and Real Time Clock as well as a few extra features including alarm function and Watchdog timer.

OPERATION

Nonvolatile RAM Operation

One important feature of the Dallas Phantom Real Time Clocks is that the nonvolatile RAM can be used to store system configuration data and, since the clock is transparent to the RAM, no memory is lost to timekeeping needs. When the Phantom Clock is not accessed, the $\overline{CE}$ signal is passed on to the chip enable of the memory. Reading and writing to the RAM is identical to that of a standard RAM chip. Figure 1 illustrates a typical RAM/Time Chip interface. Note that this is the basic interface used for the DS1216 SmartWatch/RAM and the DS124x.

The Phantom Real Time Clock family performs circuit functions required to make a CMOS RAM nonvolatile. First, a switch is provided to direct power from the battery or V_{CC} supply, depending on which is greater. The second function provided is power-fail detection. Power–fail detection occurs when V_{CCI} falls below V_{TP}, which is equal to 1.26 x V_{BAT}. When V_{CCI} goes out of tolerance, a comparator outputs a power–fail signal to the chip enable logic. The third function accomplishes write protection by holding the chip enable signal to the memory ($\overline{CEO}$) within 0.2 volts of V_{CC} or battery as long as V_{CC} is out of tolerance. During nominal power supply conditions the memory chip enable signal ($\overline{CEO}$) will track the chip enable signal ($\overline{CEI}$) sent to the socket with a maximum propagation delay of 20 ns.

Finally, an important consideration when using the DS1216 is to select a RAM that draws no more than a maximum of 1 µA. If the RAM data retention current is larger than 1 µA, the device will not meet the data retention expectations of more than 10 years at 25°C. In the 10-year data retention calculation for the DS1216, it is assumed that system power will be on 20% of the time. Perhaps the best way to insure that a data retention of 10 years at 25°C is met is to use one of the DS124x modules with self–contained nonvolatile RAM. The DS124x modules will insure data retention for 10 years regardless of how often the system power is on.

ROM Operation

The DS1215 and DS1216(E/F) can also be used in conjunction with a ROM. A typical ROM/Time Chip interface is illustrated in Figure 2. In this configuration, the ROM/$\overline{RAM}$ pin is connected to V_{CCO} to select the ROM mode of operation. Since ROM is a read–only device that retains data in the absence of power, battery back–up and write protection is not required. As a result, the chip enable logic will force $\overline{CEO}$ low when power fails. The real time clock does retain the same internal nonvolatility and write protection as described in the RAM mode.

Real Time Clock Operation

The block diagram of Figure 3 illustrates the main elements of the Phantom Clock. Communication with the Phantom Clock is established by pattern recognition of a serial bit stream of 64 bits which must be matched by executing 64 consecutive write cycles containing the proper write data as shown in Figure 4. All accesses which occur prior to recognition of the 64–bit pattern are directed to memory via the chip enable output pin ($\overline{CEO}$). After recognition is established, the next 64 read or write cycles either extract or update data in the Phantom Clock and $\overline{CEO}$ remains high during this time, disabling the connected memory.

Data transfer to and from the Phantom Clock is accomplished with a serial bit stream under the control of chip enable input ($\overline{CEI}$), output enable ($\overline{OE}$), and write enable ($\overline{WE}$). Initially, a read cycle using the $\overline{CEI}$ and $\overline{OE}$ control of the Phantom Clock starts the pattern recognition sequence by moving a pointer to the first bit of the 64–bit comparison register. Next, 64 consecutive write cycles are executed using the $\overline{CEI}$ and $\overline{WE}$ control of the Phantom Clock. These 64 write cycles are used only to gain access to the Phantom Clock. However, the write cycles generated to gain access to the Phantom Clock are also writing data to a location in the mated RAM. The preferred way to manage this requirement is to set aside just one address location in RAM as a scratch pad for the Phantom Clock.

When the first write cycle is executed, it is compared to bit 0 of the 64–bit comparison register. If a match is found, the pointer increments to the next location of the comparison register and awaits the next write cycle. If a match is not found, the pointer does not advance and all subsequent write cycles are ignored until a read cycle is encountered which resets the comparison register pointer to the beginning of the 64–

bit comparison register. If a read cycle occurs at any time during the pattern recognition process, the present sequence is aborted and the comparison register pointer is reset. Pattern recognition continues for a total of 64 write cycles as described above until all the bits in the comparison register have been matched (this bit pattern is shown in Figure 4). With a correct match of the 64 bits, the Phantom Clock is enabled and data transfer to or from the timekeeping registers can proceed.

The next 64 cycles will cause the Phantom Clock to either receive or transmit data, depending on the level of the $\overline{OE}$ pin or the $\overline{WE}$ pin. Data will either be written to or read from the eight Phantom Clock registers shown in Figure 5. Cycles to other locations outside the memory block can be interleaved with $\overline{CE}$ cycles without interrupting the pattern recognition sequence or data transfer sequence to the Phantom Clock.

Figure 6 offers an example of pseudo code for both accessing the Phantom Clock embedded in a RAM through pattern recognition and interfacing with the clock registers. Another source code example is given in Figure 7. This code is used for interfacing with the 8051 microcontroller. Also, refer to the data book for timing diagrams for both read and write cycles.

Interfacing the Phantom Time Clock with a ROM is somewhat different from that of a RAM. This is due to the fact that no writes are made to a ROM. Since there are no $\overline{WE}$ or data in signals associated with the ROM, the Phantom Time Clock instead uses two address lines to access the real time clock as can be seen in Figure 2.

In summary, the operation of the Phantom Clocks is best defined as operating in two different modes. The first is the pattern match mode. In this mode, the Phantom Clock is transparent to the system yet monitors communication to the RAM waiting for a match of its 64–bit access pattern. When the 64–bit access pattern has been written, the Phantom Clocks enter the clock access mode. In this mode, the eight phantom clock registers are available to be written or read and will stay in this mode until all eight registers have been accessed, until a reset has been executed, or until a power-fail.

TROUBLESHOOTING

The Dallas Phantom Real Time Clocks have proven to be highly reliable and meet the published specifications. However, in the course of development, several common difficulties could be experienced. To reduce these difficulties, Dallas Semiconductor has gathered the common difficulties and pitfalls into a troubleshooting guide to assist users.

COMMON DIFFICULTIES

Cannot Access Clock Registers

Several items can cause this phenomenon.

1. Comparison register pointer has not been set to the first bit. The Phantom Real Time Clock hides behind the SRAM and waits for a match to its 64–bit access pattern. In this mode (the pattern match mode), every write operation to the RAM will be interpreted as an attempt to match the access pattern by matching the value written to DQ0 (D for the DS1215) to the pattern bit pointed to by the pattern match pointer. It is possible that a partial match of the pattern can occur during normal operation of a system. It is best to assume that there is a partial match of the access pattern and that the comparison register pointer is not pointing to the first bit of the match pattern. Therefore, the comparison register

pointer must be reset to the first pattern bit before writing the match pattern. This is accomplished by performing one read operation of the RAM before writing the match pattern.

2. Device is in clock access mode after system reset or interrupt. It is possible that during the course of previous operation, the Phantom Clock had been accessed, but had not gone back into pattern match mode before a system reset or interrupt occurred. In other words, data bits would be written to or read from the Phantom Clock registers rather than the RAM. A solution to this problem is to execute 65 consecutive read cycles immediately after an interrupt or system reset. This will insure that the device is taken out of the clock access mode (by reading a maximum of 64 bits) and will reset the comparison register pointer.

3. Access pattern has been input in reverse order. Insure that the pattern is input in the following order. Start with bit 0 of byte 0 continuing to bit 7 of byte 7.

4. Device is being reset. Insure that the device is not accidentally being reset. This can especially be a problem with the DS1216C, DS1216D, and DS1244Y where the reset pin is shared with an address pin. In this situation, that particular address line must never be taken low unless the reset bit (byte 4, bit 4) of the Phantom Clock is disabled, otherwise the device will be reset and the data transfer will be aborted.

5. Device is in constant write protect mode. If only one battery is being used for the DS1215, ensure that the BAT2 pin is grounded. If this pin remains floating it is possible that the device will think that a power−fail condition has occurred. This is due to the method in which power−fail is detected. A power−fail is determined to have occurred when V_{CC} is less than V_{TP}, which is equal to 1.26 x V_{BAT}. If V_{BAT2} is floating, it is possible that the node could float to a value such that V_{TP} is equal to V_{CC} and thus the device will always be in a write-protect mode.

Device Will Not Oscillate

1. Oscillator enable bit is disabled. Insure that the oscillator enable bit (bit 5 of byte 4) is at logic 0.

2. Wrong crystal used (DS1215). Insure that the correct crystal is being used. It is very important that a crystal with a load capacitance of 6 pF is used. Dallas Semiconductor recommends Seiko part number DS−VT−200, Daiwa part number DT−26S, or equivalent.

3. Poor crystal connection (DS1215). To insure the greatest performance, insure that the crystal is placed as close as possible to the crystal input pins. It should also be mentioned that the DS1215 does not require load capacitors or feedback resistors.

Note: It should also be mentioned that it is difficult to determine if the device is oscillating by trying to measure the frequency with an oscilloscope probe. This is because of the loading caused by the scope probe which can kill the oscillator.

Timekeeping Inaccurate

1. Input pins driven higher than V_{CC}. It is very important to insure that input pins never go above V_{CC}. If any input is allowed to go above V_{CC}, it is possible that the oscillator may be briefly stopped, which will cause the device to lose time.

2. Wrong crystal used (DS1215). For best accuracy, insure that the correct crystal is used.

RAM is Losing Data when Powered Down

This problem can occur especially in NMOS processors which become unstable at a higher voltage than CMOS processors. This problem manifests itself in the method in which power−fail is determined. Write protection is asserted when V_{CC} drops below V_{TP}, which is equal to 1.26 x V_{BAT}. Typically, the battery has a voltage of ~3.0V which leads to a V_{TP} of 3.78V. Therefore, in a power−down situation, if the processor becomes unstable at a V_{CC} of greater than ~3.78V (which is often the case for an NMOS processor), a spurious write cycle could corrupt the data in the Phantom Clock. The solution to this problem is to ensure that the processor is reset before it becomes unstable and thus prevent any unwanted writes from being executed. This can be accomplished by monitoring V_{CC} by one of Dallas Semiconductor's power monitors (the DS123x family) which generate a reset signal when V_{CC} is out of tolerance.

Cannot Read Consecutive Hundredths of Seconds

It is not possible to read consecutive hundredths of seconds because the access time to read the clock registers is too long.

COMMON PITFALLS

1. Device needs separate read and write signals. The Dallas Phantom Real Time Clocks were designed with Intel timing in mind. Therefore it is necessary to have separate read and write signals. It should be further stressed that simply complementing one signal to arrive at the other is not sufficient because this will cause the pattern match pointer to be reset during each write cycle since the $\overline{OE}$ signal will toggle whenever the $\overline{WE}$ signal toggles.

2. Battery attachment (DS1215). Any battery attached to the BAT1 or BAT2 pins must be connected directly to the pin. It should be noted that a diode should not be connected between the battery input pin and the battery. This is not necessary because internal reverse charging current protection circuitry is provided and is UL recognized (#E99151).

3. ROM/$\overline{RAM}$ pin (DS1215). Insure that the ROM/RAM pin is set to the correct value.

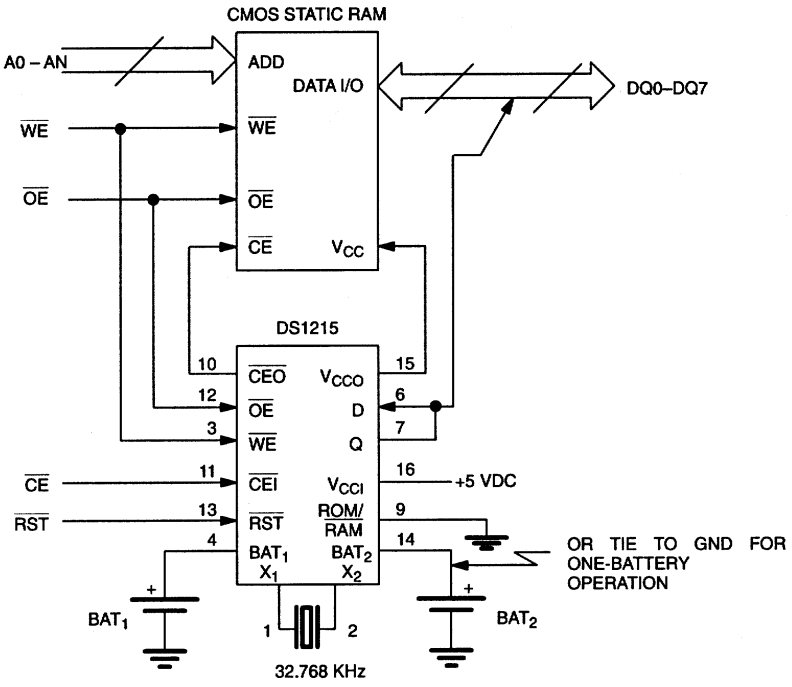
4. Reading and writing to clock registers. It is important that all 64 bits be read or written to when the clock registers have been accessed. If this is not done, the device will remain in the clock access mode.

5. Do not water wash DS1216 Intelligent Sockets. Water washing for flux removal must not be performed on the DS1216 Intelligent Sockets because contaminants in the water can cause discharging of the internal lithium energy source.

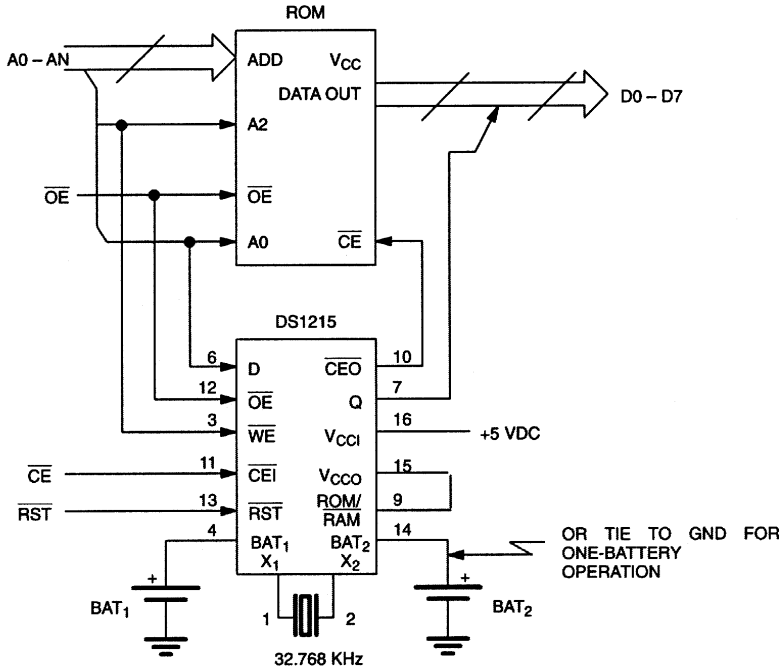
5. Crystal selection (DS1215). A 32.768 kHz quartz crystal, Seiko part number DS−VT−200, Daiwa part number DT−26S, or equivalent should be used. The crystal selected for use must have a specified load capacitance of 6 pF. The use of an incorrect crystal can kill the oscillator or cause accuracy problems. Also, the use of an external trim capacitor to adjust the oscillator is discouraged.

It is recommended that one of the Dallas SmartWatch or Nonvolatile SRAM with Phantom Time Clock devices be selected for the highest accuracy (±1 minute/month).

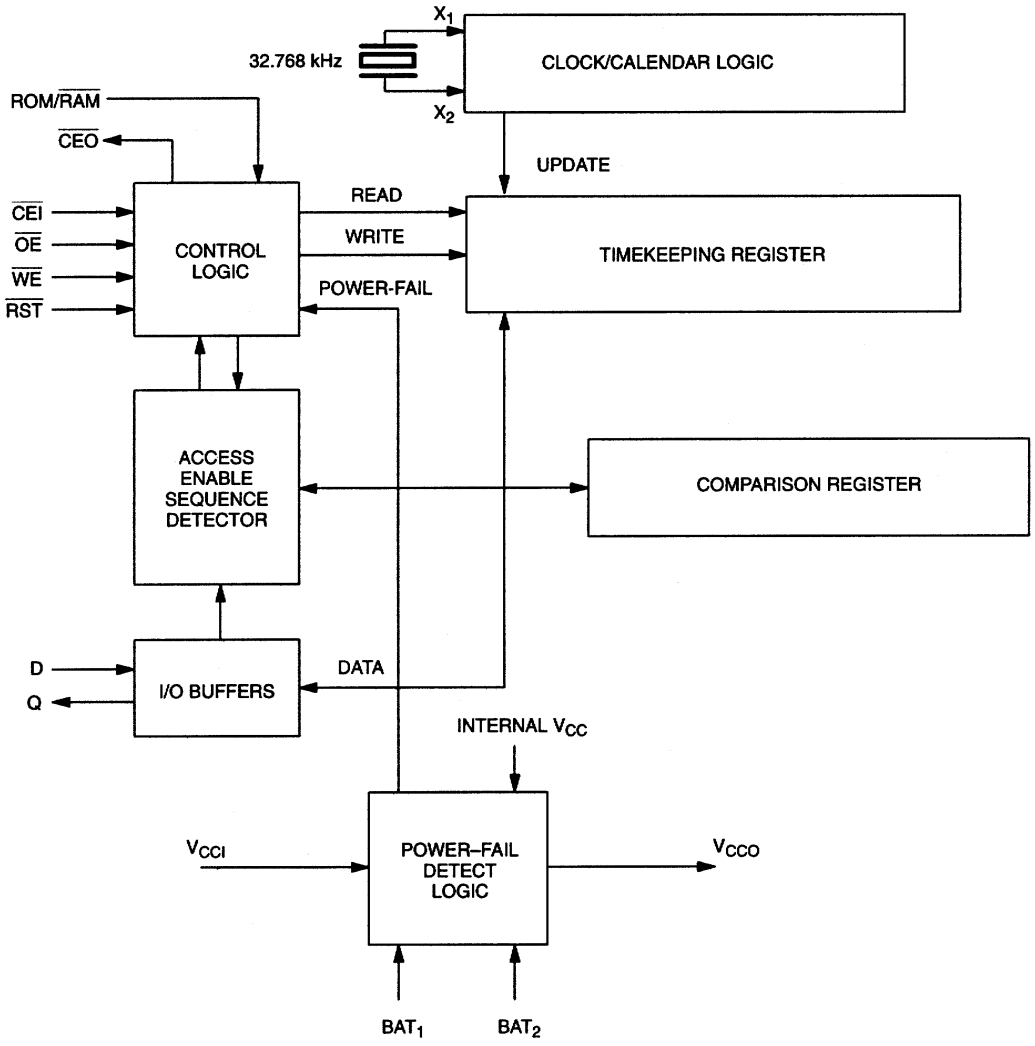
RAM/TIME CHIP INTERFACE Figure 1



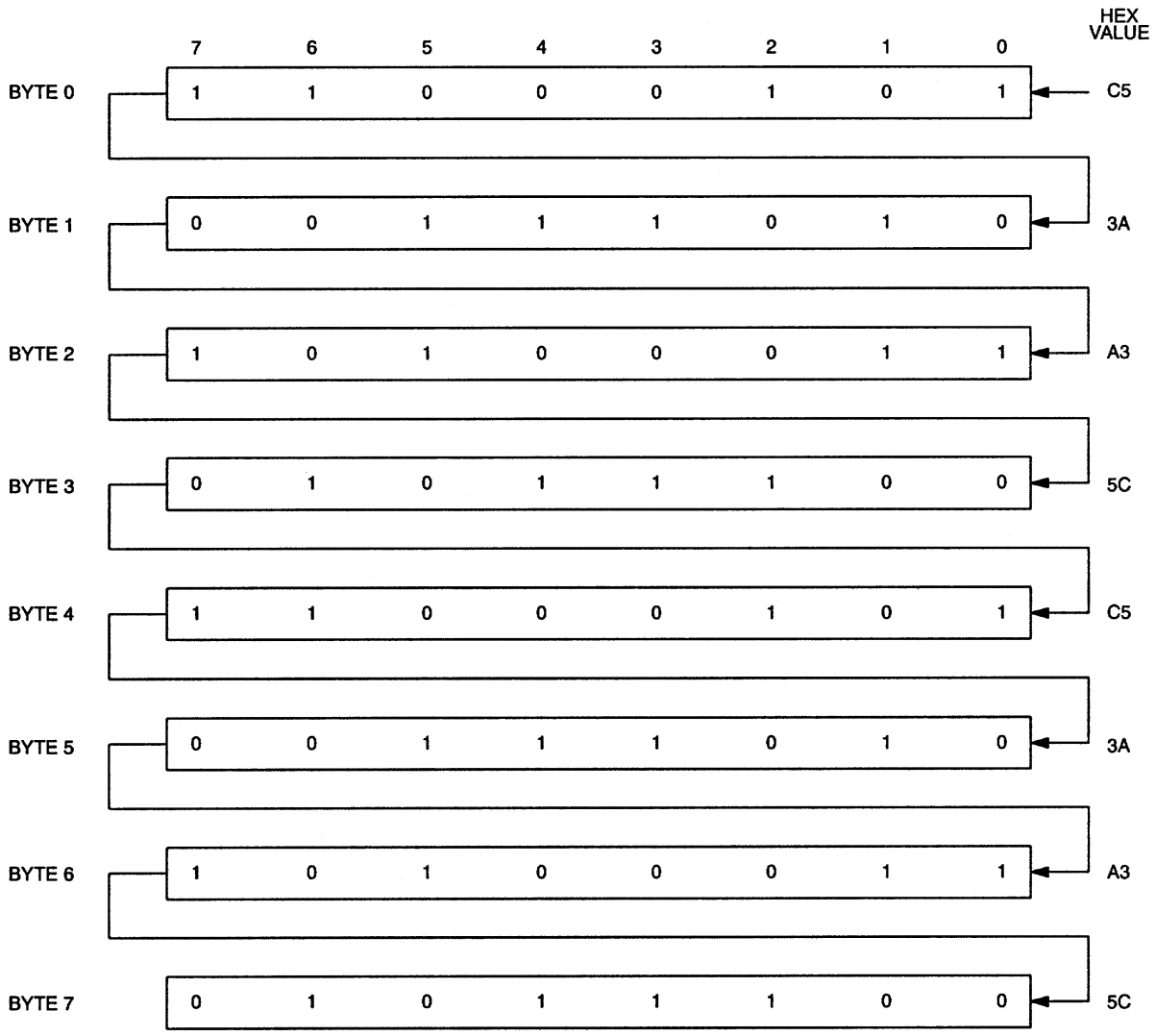
ROM/TIME CHIP INTERFACE Figure 2



TIMING BLOCK DIAGRAM Figure 3



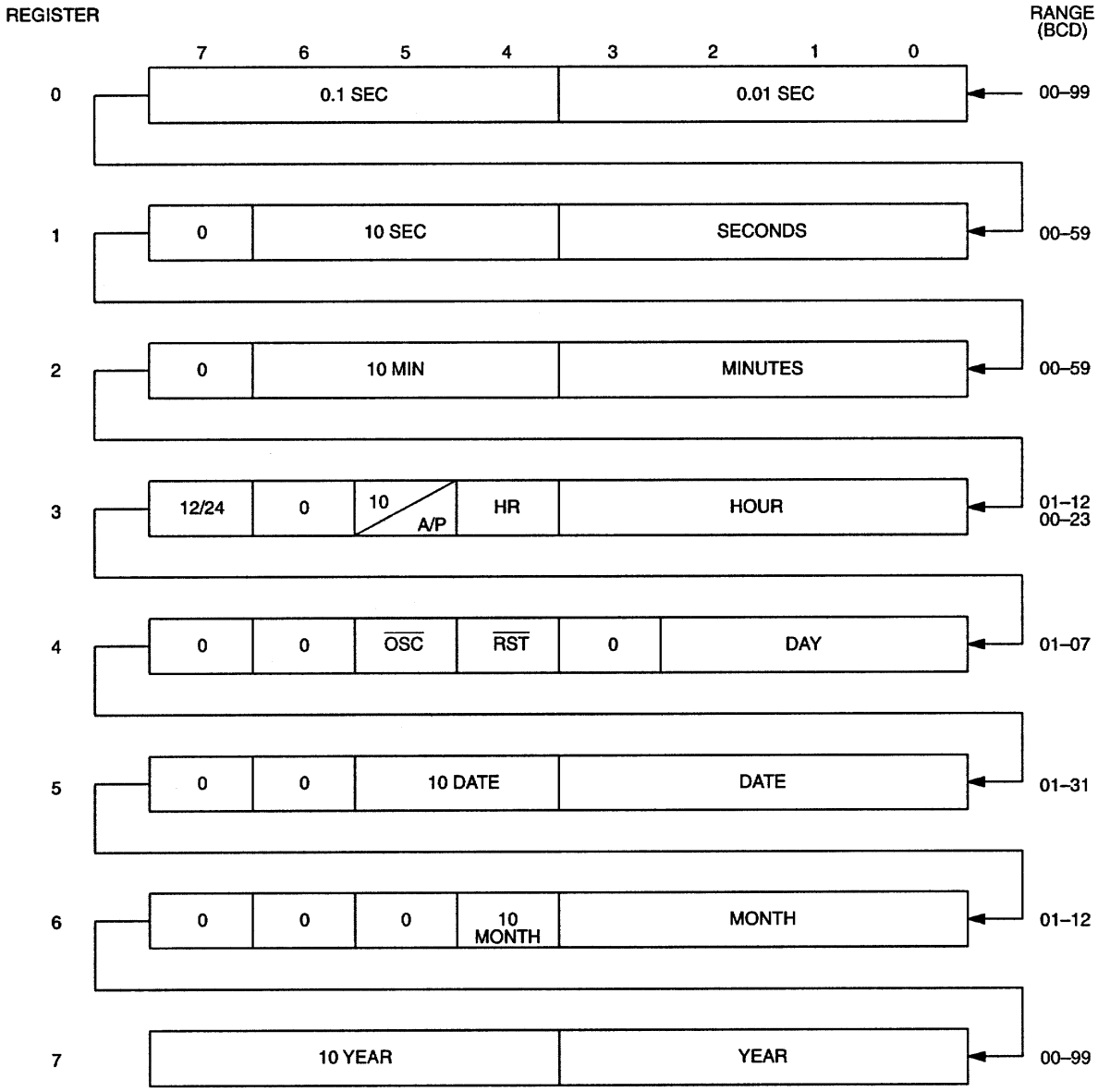
TIME CHIP COMPARISON REGISTER DEFINITION Figure 4



NOTE:

The pattern recognition in Hex is C5, 3A, A3, 5C, C5, 3A, A3, 5C. The odds of this pattern being accidentally duplicated and causing inadvertent entry to the Time Chip are less than 1 in 10^{19}. This pattern is sent to the Phantom Clock LSB to MSB.

TIME CHIP REGISTER DEFINITION Figure 5



PSEUDO CODE Figure 6

```
* This code will access the Phantom Time Clock by sending the 64-bit   *
* access pattern.  Then the time data will be written to the clock and *
* finally the Phantom Time Clock will be accessed again and it will be *
* read. The time information to be written is 12:00 PM Wednesday,      *
* January 1, 1992. Also note that the oscillator has been enabled and  *
* reset has been disabled.                                             *

A : Array[0..7] = (C5, 3A, A3, 5C, C5, 3A, A3, 5C)  (access pattern)
T : Array[0..7] = (00, 00, 00, B2, 14, 01, 01, 92)  (time data)
X : Byte at 1000                                    (memory location 1000H)
D : Array [0..7]
S : Byte

* Send access pattern to Phantom Time Clock *
   FOR I = 0 TO 64 S = x                   (perform 65 consecutive reads from x)
   FOR I = 0 TO 7                             (loop for 8 bytes)
      FOR J = 0 TO 7                          (loop for 8 bits)
         X = A[I] SHR J                       (write to X, shift bits right J)
      NEXT J
   NEXT I

* Write time data to Phantom Time Clock registers *
   FOR I = 0 TO 7                             (loop for 8 bytes)
      FOR J = 0 TO 7                          (loop for 8 bits)
         X = T[I] SHR J                       (write to X, shift bits right J)
      NEXT J

* Send access pattern to Phantom Time Clock *
   FOR I = 0 TO 64 S = X                   (perform 65 consecutive reads from X)
   FOR I = 0 TO 7                             (loop for 8 bytes)
      FOR J = 0 TO 7                          (loop for 8 bits)
         X = A[I] SHR J
      NEXT J
   NEXT I

* Read Phantom Time Clock registers *
   FOR I = 0 TO 7                             (loop for 8 bytes)
      D[I] = 0                             (initiate the byte)
      FOR J = 0 TO 7                          (loop for 8 bits)
         D[I] = D[I] or (X and 1) SHL J    (position bits in byte)
      NEXT J
   NEXT I
```

EXAMPLE SOURCE CODE FOR 8051 MICROCONTROLLER Figure 7

```
; 8051CODE.DOC
; RTC procedure to access the DS1215 Serial Timekeeper, or DS1216
; SmartWatch using 8031, 8051 or 80C196
;
BIT_SEG SEGMENT BIT
        RSEG    BITSEG
WF:     DBIT    1
BYTE_SEG        SEGMENT DATA
        RSEG    BYTE_SEG
BUFF:   DS      8           ;Centi-sec: 00-99
;                           ;Seconds: 00-59
;                           ;Minutes: 00-59
;                           ;Hours: 01-12 / 00-23
;                           ;Day:1nHEX % RST off, n=DAY# 01-07
;                           ;Date: 01-31
;                           ;Month: 01-12
;                           ;Year: 00-99
CODESEG SEGMENT CODE
        RSEG    CODE_SEG
;*************************
;*** MAIN PROGRAM GOES HERE
;*************************
;
; Main program SETS WF for Read Mode and on return from RTC the BUFF will
; contain the 8 bytes of data read from the clock.  If WF is CLEARED then
; RTC will return after writing the 8 byte BUFF to the clock.
;
;   NOTE !!! : Refer to the DS1215 (RAM MODE) or DS1216 data sheet.
;
RTC:    PUSH        PSW                 ;Save user registers.
        PUSH        ACC
        PUSH        B
        MOV         B, RO
        PUSH        B
        MOV         RO, #BUFF           ;Load pointer to start of table.
        LCALL       OPEN                ;Set up to open the DS1216.
        MOV         B, #8H              ;Load loop counter for 8 bytes.
        JNB         WF, WRITETIME       ;Read/Write mode check.
;
READTIME:           LCALL   RBYTE       ;Read one byte.
        MOV         @RO, A              ;Save in RTn temporary register.
        INC         R0                  ;Temporary data register pointer.
        DJNZ        B, READTIME         ;Loop to read 8 bytes.
        SJMP        ENDTIME             ;Done reading goto finish.
;
WRITETIME:          MOV     A, @R0      ;Load byte of data to be written.
        LCALL       WBYTE               ;Write one byte.
        INC         R0                  ;Temporary data register pointer.
        DJNZ        B, WRITETIME        ;Loop to write 8 bytes.
;
ENDTIME:            POP     B           ;Restore registers.
        MOV         R0, B
        POP         B
        POP         ACC
        POP         PSW
        RET                             ;Return to main calling program.
;
;
;
;
;************************************
; SUBROUTINE TO OPEN THE CLOCK/CALENDAR
;************************************
;
;
; This subroutine executes the sequence of reads and writes which
; is required in order to open communication with the timekeeper.
;
OPEN:   LCALL       CLOSE               ;Make sure it is closed.
        MOV         B, #4               ;Set pattern period count.
        MOV         A, #0C5H            ;Load first byte of pattern.
OPENA:  LCALL       WBYTE               ;Send out the byte.
        XRL         A, #0FFH            ;Generate next pattern byte.
        LCALL       WBYTE               ;Send out the byte.
```

```
        SWAP        A                       ;Generate next pattern byte.
        DJNZ        B, OPENA                ;Repeat until 8 bytes sent.
        RET                                 ;Return.
;
;*****************************
;*** SUBROUTINE TO CLOSE CLOCK
;*****************************
;
; This subroutine insures that the registers of the timekeeper
; are closed by executing 72 successive reads of the date and time
; registers.
;
CLOSE:  MOV         B, #9                   ;Set up to read 9 bytes.
CLOSEA: LCALL       RBYTE                   ;Read a byte.
        DJNZ        B, CLOSEA               ;Loop for 9 byte reads.
        RET                                 ;Return
;
;*********************************
:*** SUBROUTINE TO READ A DATA BYTE
;*********************************
;
RBYTE:  PUSH        DPL                     ;Save the data
        PUSH        DPH                     ;pointer on stack.
        PUSH        B                       ;Save the B register.
        MOV         DPTR, #RTCADDR              ;Enable the clock.
        MOV         B, #8                   ;Set the bit count.
LI:     PUSH        ACC                     ;Save the accumulator.
        MOVX        A, @DPTR                ;Input the data bit.
        RRC         A                       ;Move it to carry.
        POP         ACC                     ;Get the accumulator.
        RRC         A                       ;Save the data bit.
        DJNZ        B, LI                   ;Loop for a whole byte.
        POP         B                       ;Restore the B register.
        POP         DPH                     ;Restore the data
        POP         DPL                     ;pointer from stack.
        RET                                 ;Return.
;
;
;
;*********************************
;*** SUBROUTINE TO WRITE A DATA BYTE
;*********************************
;
WBYTE:  PUSH        DPL                     ;Save the data
        PUSH        DPH                     ;pointer on stack.
        PUSH        B                       ;Save the B register.
        MOV         DPTR, #RTCADDR              ;Enable the clock.
        MOV         B, #8                   ;Set the bit count.
LO:     PUSH        ACC                     ;Save the accumulator.
        ANL         A, #1                   ;Set up bit for output.
        MOVX        @DPTR, A                ;Output the data bit.
        POP         ACC                     ;Restore the accumulator.
        RR          A                       ;Position next bit.
        DJNZ        B, LO                   ;Loop for a whole byte.
        POP         B                       ;Restore the B register.
        POP         DPH                     ;Restore the data
        POP         DPL                     ;pointer from stack.
        RET                                 ;Return.
;
;***************
;END OF PROGRAM
;***************
;
        END                                 ;End of program.
```