SOFTWARE AND HARDWARE TIMERS

Examples and discussions of using Software and Hardware

Timers using BASIC with machine language routines.

1) Software duration timers
2) Software background timer
3) Hardware timers

ATARI INC.
CONSUMER PRODUCT SERVICE
PRODUCT SUPPORT GROUP
1312 Crossman Avenue
Sunnyvale, CA 94086

(800) 672-1404 inside CA
(800) 538-8543 outside CA

DEMOPAC #8
Rev. 1 5-83/JC

DISCLAIMER OF WARRANTY ON PROGRAMS CONTAINED HEREIN

All computer programs contained herein are distributed on an "as is" basis by Atari, Inc. ("Atari") without warranty of any kind. Any statements concerning the capabilities or utility of the computer programs are not to be construed as express or implied warranties. The entire risk as to the quality and performance of such programs is with the user. Should a program fail to fulfill the individual requirements of the user, or prove defective, the user (and not Atari), assumes the entire cost of all servicing, damages or liabilities which may result from the use of any such computer program.

Atari shall have no liability or responsibility to the user or any other person or entity with respect to any claim, loss, liability, or damage caused or alleged to be caused directly or indirectly by computer programs distributed by Atari. This disclaimer includes, but is not limited to, any interruption of services, loss of business, data or anticipatory profits, and/or incidental or consequential damages resulting from the purchase, use or operation of computer programs made available by Atari.

Some states do not allow the limitation or exclusion of implied warranties or of incidental or consequential damages, so the above limitations or exclusions may not apply to you.

Every effort has been made to ensure the accuracy of this document. However, because of ongoing improvements and updating of our computer software and hardware, Atari cannot guarantee the accuracy of printed material after the date of publication and disclaims any liability for changes, errors, or omissions.

Correspondence regarding this document should be forwarded to Manager of Technical Support, Consumer Product Service, Atari, Incorporated, 1312 Crossman Avenue, Sunnyvale, CA 94086.

SOFTWARE TIMERS

JC 5/10/83

The ATARI operating system maintains five software timers. These timers are actually memory locations which are updated (decremented) during the VBLANK interrupt process.

'VBLANK' refers to the interrupt which occurs each 1/60th second at the end of a complete video scan, while the video beam is shut off and returning to the upper left corner of the screen. During this period the operating system performs various system functions in two stages. Stage one VBLANK processes are always performed every VBLANK or 1/60th second. Stage two VBLANK functions may be skipped if time critical operations must be performed, as indicated by a non zero value in the 'CRITIC' flag at decimal location 66.

Timer number one is used by the operating system to time I/O operations and is updated during each stage one VBLANK. Timer number one should not be used by an applications program unless serial I/O is suspended during the period it is in use.

Timers two through five are not used by the operating system and are updated every stage two VBLANK. Since stage two VBLANK is skipped duringtime critical I/O operations, timers two through five may not always be updated every 1/60th second. This can be seen by running the background timer demonstration program and then performing an I/O operation such as accessing the disk to read or write a file. If it is essential not to miss a count while using a timer during critical I/O operations, the system clock which is updated during stage one VBLANK may be used.

There are two methods used by the timers to indicate that the countdown has been completed. Timers one and two call a subroutine. An applications program may place the address of a user subroutine into special memory locations reserved for this purpose. When timer one or two times out (decrements to zero), a 'JSR' to the user subroutine takes place. The user subroutine should end with an RTS (return) instruction. Timers two through five use special locations in memory as 'flags'. It is necessary to set the 'flag' to a non-zero value before starting the timer and then check the 'flag' location for a zero value, which indicates that time out has occured.

Since the timers are incremented in 1/60th second intervals, the minimum time value that can be set is 1/60 second (when a value of 1 is used), and the maximum time value is 18.20 minutes (when a value of 65,535 or hex $FFFF, is used).

TO SET AND START THE TIMERS USE THE FOLLOWING PROCEDURE:

1. For timer 1 or 2, store the address of the routine to be executed when time out occurs, in the vector locations specified in the Software Timer Address Table. Store the Least Significant Byte (LSB) followed by Most Significant Byte (MSB). Make sure the code to be executed at this address has been loaded into memory. For timers 3,4, or 5 set the flag location specified in the table, to the value '$FF' (this location becomes '0' when time out occurs).

2. Load the accumulator with the number (1-5), of the Software Timer.

3. Set the timer duration by loading the 'Y' register with the LSB, and the 'X' register with the MSB of the number of 1/60th second intervals to count. A value of '0' in the 'X' register and '60' in the 'Y' register would indicate a 1 second interval.

4. To begin timer execution do a JSR (Jump Subroutine) to the 'SETVBV' routine at hex location $E45C.

5. If timer 3,4, or 5 is used it is necessary to check its flag location for a zero value, to determine if time out has occured. Since timers 1 and 2 are vectored to the address set in step 1, when time out occurs, execution will automatically begin at the specified location. This makes it possible to create a program which re-initializes and re-starts the timers just before returning, giving a background effect. This is the method employed by the 'Background Software Timer' demonstration program which plays a song in the background while other activities may continue in the foreground.

The following chart lists the locations of the five software timers and their respective flags and jump vectors. All addresses are in hexadecimal.

SOFTWARE TIMER ADDRESS TABLE

Timer Number	Timer Name	Hex Addr.	Flag/ Vector	Hex Addr.	Use
1	CDTMV1 -	$0218	CDTMA1 -	$0226	2-byte vector
2	CDTMV2 -	$021A	CDTMA2 -	$0228	2-byte vector
3	CDTMV3 -	$021C	CDTMF3 -	$022A	1-byte flag
4	CDTMV4 -	$021E	CDTMF4 -	$022C	1-byte flag
5	CDTMV5 -	$0220	CDTMF5 -	$022E	1-byte flag

SOFTWARE TIMER DEMO PROGRAM

'SOFTIME.BAS' is a program which demonstrates the use of the software timers (2-5), which set a flag to indicate that the timer has completed its countdown to zero. When the timer has completed its assigned timing task, a location in memory (see timer address table), is set to zero.

The following BASIC program uses an assembly language routine, 'SOFTIME.ASM', to set and start software timer number four. The assembly language routine then continuously checks the associated timer flag until it becomes zero, at which time a return instruction is executed, and program control returns to the BASIC program.

Using the software timers in this way is an extremely simple process. The assembly language portion of this routine uses only ten instructions and takes up a compact sixteen bytes. In spite of its small size, because the time duration is passed from BASIC, it is possible to use this routine for a variety of purposes in a BASIC program. This routine can replace many 'FOR/NEXT' loops used for timing durations, when more accurate timing is important.

In the 'SOFTIME.BAS' example, the timer routine is used three times with varying values to time different functions. In line '350' the user is prompted to enter the timer duration in Jiffies (1/60th second), which is stored as the numeric variable 'D'. This value is then used in line '480' to time the delay between printing characters on the screen. The value of 'D' can vary between 1 (1/60th second) and 65535 (18.20 minutes). Before the character is printed, the timer is used to time a sound of 1/60th second duration in line '450'. After the screen border has been filled with characters, the timer is used for a third function in line '500' where a two second delay is timed prior to clearing the screen and repeating the entire process.

```
100 REM *****************************
110 REM * SOFTIME.BAS               *
120 REM *                           *
130 REM * John Clark                *
140 REM * 04/21/83                  *
150 REM * call assembly language    *
160 REM * routine to use software   *
170 REM * timer #4 to time BASIC    *
180 REM * routines                  *
190 REM *****************************
200 REM
210 REM ****************************
220 REM SET UP ARRAY FOR SCREEN POSTIONING
230 DIM X(55),Y(55)
240 FOR I=0 TO 55
250 IF I<20 THEN X(I)=I:Y(I)=0
260 IF I>19 AND I<29 THEN X(I)=19:Y(I)=I-19
270 IF I>28 AND I<48 THEN X(I)=-1*(I-29)+18:Y(I)=9
280 IF I>47 THEN X(I)=0:Y(I)=-1*(I-48)+8
290 NEXT I
300 REM ***************************
310 GOSUB 610:REM POKE ASSEMBLY LANGUAGE ROUTINE INTO MEMORY
320 REM
330 REM ***************************
340 REM CALL GRAPHICS 2 AND SET COLORS
350 GRAPHICS 2
360 POKE 712,190:REM BACKGROUND COLOR
370 POKE 710,246:REM LETTER COLOR
380 POSITION 2,5:PRINT #6;"software timers"
390 PRINT "INPUT TIMER DURATION (JIFFIES)";
400 INPUT D
410 REM ***************************
420 REM LOOP TO PRINT TIMED DISPLAY
430 FOR I=0 TO 55
440 SOUND 0,230,10,8
450 A=USR(1536,1):REM TIME SOUND DURATION 1/60th SECOND
460 SOUND 0,0,0,0
470 POSITION X(I),Y(I):PRINT #6;CHR$(176):REM PRINT BLUE ZERO CHARACTER
480 A=USR(1536,D):REM TIME DURATION OF EACH SCREEN CHARACTER DISPLAYED
490 NEXT I
500 A=USR(1536,120):REM TWO SECOND DELAY AND THEN REPEAT PROMPT
510 GOTO 350
520 REM
530 REM
540 REM ***************************
550 REM POKE ASSEMBLY LANGUAGE
560 REM TIMER PROGRAM
570 REM INTO MEMORY
580 REM BEGINNING AT LOCATION 1536
590 REM ***************************
600 REM
610 FOR I=0 TO 20
620 READ J:POKE 1536+I,J
630 NEXT I
640 RETURN
650 DATA 104,104,170,104,168,169,255,141,44,2
660 DATA 169,4,32,92,228,173,44,2,208,251,96
```

```
                 05 ;****************************
                 10 ;* SOFTWARE DURATION TIMERS  *
                 15 ;*                          *
                 20 ;*   John Clark 5/10/83     *
                 25 ;*                          *
                 30 ;* time basic operations    *
                 35 ;* with assembly language   *
                 40 ;* routines.                *
                 45 ;*                          *
                 50 ;* Called from BASIC with   *
                 55 ;* 'A=USR(1536,Duration)'   *
                 60 ;* where duration is number *
                 65 ;* of 1/60 second units     *
                 70 ;*                          *
                 75 ;****************************
                 80 ;
                 85 ;
022C             90 CDTMF4    =     $022C     ;countdown timer #4 flag
E45C             95 SETVBV    =     $E45C     ;set timer routine
                 0100 ;
0000             0105         *=    $600
                 0110 ;
0600 68          0115         PLA             ;throw away number of arguments
0601 68          0120         PLA             ;MSB of timer duration
0602 AA          0125         TAX             ;store in X register
0603 68          0130         PLA             ;get LSB of timer duration
0604 A8          0135         TAY             ;store in Y register
0605 A9FF        0140         LDA   #$FF      ;value to initialize timer #4 flag
0607 8D2C02      0145         STA   CDTMF4    ;store in timer #4 flag
060A A904        0150         LDA   #4        ;indicate timer #4 will be used
060C 205CE4      0155         JSR   SETVBV    ;set timer and start timing
060F AD2C02      0156 LOOP    LDA   CDTMF4    ;check timer #4 flag
0612 D0FB        0158         BNE   LOOP      ;if not '0' then check again
0614 60          0160         RTS             ;return to BASIC
0 ERRORS
```

BACKGROUND SOFTWARE TIMER DEMO PROGRAM

The Background Timer demonstration program uses software timer #2 to play a background tune, while other activity, such as running or editing a BASIC program, can take place in the foreground.

The BASIC portion of this program simply pokes the assembly language routine into memory and calls it with a 'USR' statement. Once the program is initiated, it will continue running until system reset is pressed.

This program takes advantage of the fact that when software timer number two has finished its countdown, a jump subroutine takes place to a designated user routine. The address of this routine is placed in the timer number two vector location at hex $0228 and $0229 (LSB,MSB). The 'background' effect is achieved by having the user routine re-initialize and start timer number two after playing a note from a note table stored in memory. Each time the timer counts down to zero, a jump subroutine occurs to the user defined interrupt service routine, which plays a note and starts the timer again. An offset to determine which note will be played from the note table is stored in location $0676. This offset is increased to point to successive notes each time the routine is called, and is reset to zero, to begin again, after all thirty two notes have been played. Software timer number three is used to time a 1/30th second duration for each note.

Since timer number two is updated during each stage two 'VBLANK', any I/O activity will interfere with the 'background operation' (stage two 'VBLANK' operations are skipped when critical I/O is taking place). Initiate the program and then do a disk file access. The background song will stop and then resume when I/O has been completed. If it is important to keep timing while I/O is taking place, the system clock may be used.

```
100 REM ************************
110 REM *** BACKGROUND TIMER ***
120 REM ***   John Clark     ***
130 REM ***    04/18/83      ***
140 REM ************************
150 REM
160 REM ****************************************
170 REM **POKE MACHINE LANGUAGE TIMER ROUTINE **
180 REM **INTO PAGE SIX MEMORY                **
190 REM **AND CALL WITH USR FUNCTION          **
200 REM ****************************************
210 REM
220 REM
230 FOR I=0 TO 118
240 READ J:POKE 1536+I,J
250 NEXT I
260 A=USR(1536)
270 END
280 REM
290 REM
300 DATA 104,32,66,6,162,0,142,118,6,96,169,0,141,8,210,169,3,141,15,210,174,118
,6,189,86,6,141,0,210,232
310 DATA 224,32,208,2,162,0,142,118,6,169,175,141,1,210,160,2,162,0,169,255,141,
42,2,169,3,32,92,228,173,42
320 DATA 2,208,251,141,1,210,169,10,141,40,2,169,6,141,41.2,169,2,160,20,162,0,3
2,92,228,96,96,108,121,108
330 DATA 96,96,96,0,108,108,108,0,96,81,81,0,96,108,121,108,96,96,96,0,108,108,9
6,108,121,0,0,0,0
```

```
              10 ;***************************
              20 ;BACKGROUND SOFTWARE TIMER
              30 ;***************************
              40 ;
              50 ;John Clark 04/18/83
              60 ;use software timer #2 interrupt
              70 ;to play background tune
              80 ;use software timer #3 to count duration of note
              82 ;called from BASIC with
              83 ;'A=USR(1536)'
              90 ;
              91 ;***************************
              92 ;
0228          0100 CDTMA2 =    $228       ;vector for timer interrupt
E45C          0110 SETVBV =    $E45C      ;set timer routine
D208          0120 AUDCTL =    $D208      ;audio control
D20F          0130 SKCTL  =    $D20F      ;serial port control
D200          0140 AUDF1  =    $D200      ;audio freq 1
D201          0150 AUDC1  =    $D201      ;audio channel 1 control
022A          0170 CDTMF3 =    $22A       ;timer 3 flag
              0180 ;
0000          0190        *=   $600
              0200 ;INITIALIZE TIMER #2 AND RETURN
              0210 ;
              0220 ;add PLA here if calling from BASIC
              0230 ;
0600 204106   0240        JSR  INIT       ;initialize timer #2
0603 A200     0250        LDX  #0         ;initial offset value for note table
0605 8E7506   0260        STX  POINT      ;save offset to note table in memory
0608 60       0270        RTS             ;return from initializing timer #2
              0280 ;*****************************
              0290 ;TIMER INTERRUPT SERVICE ROUTINE
              0300 ;
0609 A900     0310 TIME   LDA  #0         ;initialize audio control
060B 8D08D2   0320        STA  AUDCTL
060E A903     0330        LDA  #3         ;initialize POKEY to use audio registers
0610 8D0FD2   0340        STA  SKCTL
0613 AE7506   0350        LDX  POINT      ;get offset to note table from memory
0616 BD5506   0360        LDA  NOTES,X    ;get a note from note table
0619 8D00D2   0370        STA  AUDF1      ;place note in audio freq register
061C E8       0380        INX             ;point to next note in table
061D E020     0390        CPX  #32        ;is it last note in table?
061F D002     0400        BNE  STORE      ;no, store note in memory location POINT
0621 A200     0410        LDX  #0         ;if done point to first note of song
0623 8E7506   0420 STORE  STX  POINT      ;store next offset in memory
0626 A9AF     0430        LDA  #$AF       ;use pure note full volume
0628 8D01D2   0440        STA  AUDC1      ;place in audio control register
062B A002     0490        LDY  #2         ;1/30 second duration
062D A200     0520 COUNT  LDX  #0         ;MSB of note duration
062F A9FF     0540        LDA  #$FF       ;init timer 3 flag
0631 8D2A02   0550        STA  CDTMF3     ;to a non zero value
0634 A903     0551        LDA  #3         ;indicate set timer #3
0636 205CE4   0560        JSR  SETVBV     ;set and start timer #3
              0570 ;*****************************
              0580 ;loop to time duration
              0590 ;*****************************
0639 AD2A02   0600 LOOP   LDA  CDTMF3     ;check timer flag
063C D0FB     0610        BNE  LOOP       ; check until 0
063E 8D01D2   0620        STA  AUDC1      ;found 0 in A register, turn off sound
```

```
             0630 ;
             0640 ;*******************************
             0650 ;TIMER #2 INTERRUPT INITIALIZATION ROUTINE
0641 A909    0660 INIT    LDA  #TIME&255 ;LSB of background routine
0643 8D2802  0670         STA  CDTMA2    ;set vector for timer#2 interrupt
0646 A906    0680         LDA  #TIME/256 ;MSB of background routine
0648 8D2902  0690         STA  CDTMA2+1  ;set vector for timer #2 interrupt
064B A902    0700         LDA  #2        ;indicate software timer #2 is to be used
064D A014    0710         LDY  #20       ;LSB of countdown value for timer #2
064F A200    0720         LDX  #0        ;MSB of countdown value for timer #2
0651 205CE4  0730         JSR  SETVBV    ;initiate timer #2
0654 60      0740         RTS
             0750 ;
             0760 ;define notes for background song
0655 60      0770 NOTES   .BYTE $60,$6C,$79,$6C,$60,$60,$60
0656 6C
0657 79
0658 6C
0659 60
065A 60
065B 60
065C 00      0780         .BYTE $00,$6C,$6C,$6C,$00,$60,$51,$51,$00
065D 6C
065E 6C
065F 6C
0660 00
0661 60
0662 51
0663 51
0664 00
0665 60      0781         .BYTE $60,$6C,$79,$6C,$60,$60,$60
0666 6C
0667 79
0668 6C
0669 60
066A 60
066B 60
066C 00      0782         .BYTE $00,$6C,$6C,$60,$6C,$79,$00,$00,$00
066D 6C
066E 6C
066F 60
0670 6C
0671 79
0672 00
0673 00
0674 00
0675 00      0790 POINT   .BYTE 0
0676         0800         *=   $02E0
02E0 0006    0810         .WORD $600
0 ERRORS
```

HARDWARE TIMERS
JC 5/10/83

The POKEY chip contains four countdown timers which are commonly used for sound generation. Three of these hardware timers also have interrupt vectors which can point to a user defined routine. Because of the high clock speeds used with these hardware timers, the software timer limitation of 1/60th second can be overcome. It should be noted that hardware timers number two and three are used during serial I/O, for baud rate generation and cannot be altered.

The POKEY timers by default operate with a clock rate of 64 Khz (64000 cycles per second), but it is possible, by setting the correct bits in the AUDCTL register, to change this rate to 15 Khz (15000 cycles per second) or 1.79 Mhz (1,790,000 cycles per second). The number of cycles to count before generating an interrupt is stored in the audio frequency registers, AUDF1, AUDF2, or AUDF4. It is possible to merge two audio frequency registers by setting a bit in the AUDCTL register and use two bytes to store the value representing the number of cycles to count. The hardware timer demonstration program uses this technique. This makes the maximum duration that can be timed equal to 4.369 seconds and the theoretical minimum duration that can be timed equal to 1.79 millionths of a second. When using the hardware timers to time extremely short intervals, it is necessary to turn off 'DMA' and 'Vertical Blank Interrupts (VBI)' by storing a '0' in 'DMACTL' at location '$D400', and 'NMIEN' at location '$D40E'. Experiment with your particular application to determine if the timing rate generated is in conflict with the screen display.

TO SET AND START THE HARDWARE TIMERS USE THE FOLLOWING PROCEDURE:

1. Set the 'AUDCTL' ($D208) register to choose the desired clock speed. The default clock speed is 64 Khz. To enable the 15 Khz clock for all channels, set bit 0. To enable the 1.79 Mhz clock for channel number 1, set bit 6, and for channel number 2 set bit 5. 'AUDCTL' can also be used to join channels 1 and 2, or channels 3 and 4, to allow the timing duration to be defined by sixteen bits. This provides a range of from 1 to 65,535 clock cycles.

2. Set the volume level for the channel(s) to be used as timers, to '0', by storing a '0' in the 'AUDC1' ($D201), 'AUDC2' ($D203), or 'AUDC4' ($D207) register.

3. Choose the number of clock cycles to count and store this value in the frequency register for the appropriate channel, 'AUDF1' ($D200), 'AUDF2' ($D202), or 'AUDF4' ($D206). If two channels have been joined to allow a sixteen bit value to be specified for the number of clock cycles, enter the Least Significant Byte (LSB) in the lower numbered frequency register of the pair, and enter the Most Significant Byte (MSB) in the higher numbered frequency register.

4. Set up the routine to process the interrupt which will occur when the timer has completed its count. It is possible to have this routine re-initialize and re-start the hardware timer, so that the interrupt routine will be performed continuously in the background (see example program).

5. Set the timer interrupt vector to point to the routine created in step 4. Place the address of this routine (LSB,MSB) in the vector register associated with the timer selected, (VTIMR1-$0210, VTIMR2-$0212, VTIMR4-$0214). If two timer channels have been joined, place the address of the interrupt routine (LSB,MSB) in the vector register for the higher numbered channel, i.e., If channels number one and two have been joined, place the interrupt routine address in 'VTIMR2'.

6. To enable the interrupts for the hardware timers, set the appropriate bit in 'IRQEN' ($D20E), and the shadow register 'POKMSK' ($0010). Setting bit 0 enables interrupts for timer 1, bit 1 enables interrupts for timer 2, and bit 2 enables interrupts for timer 4. Use an 'OR' statement to accomplish this so the bits currently set in this register will not be altered (see example program).

7. To start the hardware timers, write any value to register 'STIMER' ($D209).

HARDWARE TIMER LOCATIONS AND DEFINITIONS

AUDCTL - $D208(53768 decimal).

Bit	Result when set
0	Switches main clock base from 64 Khz to 15 KHz.
3	Joins channel 4 to channel 3(16 bit resolution)
4	Joins channel 2 to channel 1(16 bit resolution)
5	Clocks channel 3 with 1.79 MHz.
6	Clocks channel 1 with 1.79 MHz.

IRQEN - $D20E(53774 decimal)
POKMSK - $0010(16 decimal) shaddow register for IRQEN.

Bit	Result when set
0	The POKEY timer #1 interrupt is enabled
1	The POKEY timer #2 interrupt is enabled
2	The POKEY timer #4 interrupt is enabled (OS rev 'B')

AUDIO CONTROL REGISTERS

	Hex	Dec
AUDC1	$D201	53761
AUDC2	$D203	53763
AUDC3	$D205	53765
AUDC4	$D207	53767

AUDIO FREQ. REGISTERS

	Hex	Dec
AUDF1	$D200	53760
AUDF2	$D202	53762
AUDF3	$D204	53765
AUDF4	$D206	53766

HARDWARE TIMER VECTOR LOCATIONS

Timer #	Timer Name	Hex loc	Dec loc
1	VTIMR1	$0210,$0211	528,529
2	VTIMR2	$0212,$0213	530,531
4	VTIMR4	$0214,$0215	532,533

STIMER - $D209 (53769) - Start hardware timers.
Store any non zero value in STIMER to start timers.

HARDWARE TIMER DEMONSTRATION PROGRAM
JC-5/10/83

The following demonstration program uses the hardware timer interrupt capability, to time the duration of each tone in a series of 256 tones. When all 256 tones have been played, the series is repeated. Timer number three is used to produce the tone, while a combination of timers one and two are used to generate the interrupt that causes the next tone in the series to be played, by loading it into the 'AUDF3', audio register.

The use of hardware timers number one and two joined together, provide a demonstration of a wide range of timing values. The BASIC portion of the program allows a choise of clock speed, and number of cycles to count (duration). In selecting a clock speed of 15 Khz (15,000 cycles per second), and choosing the maximum duration 65,535 cycles to count ($FFFF), it is possible to create a delay of 4.369 seconds between notes. Near the other end of the spectrum a choice of 1.79 Mhz in combination with a selection of 1791 cycles to count, will cause a delay of .0010 seconds between notes. At this rate, all 256 notes can be played in 0.26 seconds. The total number of cycles counted during this 0.26 second period is 458,496.

The purpose of this discussion is not to show how to generate sound effects, since this can be done without using the interrupt capability of the timer, but to point out the great speed that these timers are capable of and the large number of data 'samples' that could be captured, perhaps while monitoring an analog to digital converter connected to a real time control application.

To run the BASIC hardware timer demonstration program type in the program and enter 'RUN'. When prompted to enter the timer duration, enter a decimal number from 1 to 65535. You will then be prompted to enter the clock rate to be used. Enter a '1' to use the 15 Khz rate, a '2' to use the 64 Khz rate or a '3' to use the 1.79 Mhz rate. Try experimenting with different duration values, and clock speeds. Remember that all 256 notes will be played regardless of the duration or clock speed chosen. To run the program again with different time and/or clock values, press the 'System Reset' key to halt program execution, and enter 'RUN'; enter responses to prompts for duration and clock speed.

```
100 REM ***************************
110 REM *                         *
120 REM *     HARDTIME.BAS        *
130 REM *      JC 5/09/83         *
140 REM *                         *
150 REM * Hardware timer demo     *
160 REM * pokes machine language  *
170 REM * program into memory     *
180 REM * and calIs with          *
190 REM * A=USR(1536)             *
200 REM *                         *
210 REM ***************************
220 REM
230 REM
240 PRINT CHR$(125):REM CLEAR SCREEN
250 FOR I=0 TO 84
260 READ J:POKE 1536+I,J
270 NEXT I
280 POSITION 2,10
290 REM ** Get timer duration **
300 PRINT "INPUT TIMER DURATION(0-65535)";:INPUT DUR
310 IF DUR<1 OR DUR>65535 THEN PRINT CHR$(253):GOTO 280
320 REM * Calculate MSB,LSB duration *
330 HIDUR=INT(DUR/256):LODUR=DUR-(HIDUR*256)
340 POSITION 2,12
350 PRINT "ENTER CLOCK SPEED":PRINT
360 PRINT "          1 = 15 Khz"
370 PRINT "          2 = 64 Khz"
380 PRINT "          3 = 1.79 Mhz":PRINT
390 POSITION 2,20
400 PRINT "CLOCK SPEED = ";:INPUT CLOCK
410 IF CLOCK<1 OR CLOCK>3 THEN PRINT CHR$(253):GOTO 390
420 IF CLOCK=1 THEN CLOCK=17
430 IF CLOCK=2 THEN CLOCK=16
440 IF CLOCK=3 THEN CLOCK=80
450 POKE 54286,0:REM DISABLE VBI'S
460 POKE 54272,0:REM DISABLE DMA
470 POKE 1590,HIDUR:REM MSB DURATION
480 POKE 1585,LODUR:REM LSB DURATION
490 POKE 1572,CLOCK:REM SET CLOCK SPEED
500 A=USR(1536)
510 REM
520 REM
530 DATA 104,169,0,133,205,141,8,210,169,3,141,15,210,32,35,6,96,166,205,232,134
,205,142,4,210,169,175,141,5,210
540 DATA 32,35,6,104,64,169,17,141,8,210,169,0,141,3,210,141,1,210,169,255,141,0
,210,169,255,141,2,210,169,17
550 DATA 141,18,2,169,6,141,19,2,120,165,16,9,2,133,16,141,14,210,88,169,255,141
,9,210,96
```

```
            10 ;********************************
            15 ;*                              *
            20 ;*HARDWARE TIMER DEMONSTRATION  *
            25 ;*      JC-5/09/83              *
            30 ;*uses pokey hardware timer     *
            35 ;*to time duration between notes*
            40 ;*called from BASIC with        *
            45 ;* A=USR(1536)                  *
            50 ;*                              *
            55 ;********************************
            60 ;
D208        65 AUDCTL   =    $D208      ;audio control
D20F        70 SKCTL    =    $D20F      ;serial port control
D201        75 AUDC1    =    $D201      ;audio channel 1 control
D203        80 AUDC2    =    $D203      ;audio channel 2 control
D205        85 AUDC3    =    $D205      ;audio channel 3 control
D200        90 AUDF1    =    $D200      ;audio frequency #1
D202        95 AUDF2    =    $D202      ;audio frequency #2
D204        0100 AUDF3  =    $D204      ;audio frequency #3
0212        0105 VTIMR2 =    $0212      ;pokey timer #2 vector
D209        0110 STIMER =    $D209      ;start pokey timers
D20E        0115 IRQEN  =    $D20E      ;interrupt request enable
0010        0120 POKMSK =    $010       ;IRQEN shadow
00CD        0125 NOTE   =    $CD        ;store note value here
            0130 ;
            0135 ;
0000        0140        *=   $600       ;place program on page six
0600 68     0145        PLA             ;# arguments from BASIC
0601 A900   0150        LDA  #0         ;initialize note to zero
0603 85CD   0155        STA  NOTE       ;save note value
0605 8D08D2 0160        STA  AUDCTL     ;initialize audio control
0608 A903   0165        LDA  #$3        ;value to initialize POKEY
060A 8D0FD2 0170        STA  SKCTL      ;initialize POKEY
060D 202306 0175        JSR  SETUP      ;initialize and start timer
0610 60     0180        RTS             ;return to BASIC
            0185 ;
            0190 ;
            0195 ;*******************************
            0200 ;*                             *
            0205 ;*This routine will occur after*
            0210 ;*each interrupt generated     *
            0215 ;*by the hardware timer.       *
            0220 ;*It will cause the next note  *
            0225 ;*in the series to be played   *
            0230 ;*and then initialize and      *
            0235 ;*start the timer again        *
            0240 ;*                             *
            0245 ;*******************************
            0250 ;
0611 A6CD   0255 TIME   LDX  NOTE       ;get note from memory
0613 E8     0260        INX             ;increase to next note
0614 86CD   0265        STX  NOTE       ;store new note in memory
0616 8E04D2 0270        STX  AUDF3      ;place note in audio freq register
0619 A9AF   0275        LDA  #$AF       ;use pure note full volume
061B 8D05D2 0280        STA  AUDC3      ;place in audio control register
061E 202306 0285        JSR  SETUP      ;initialize and start timer
0621 68     0290        PLA
0622 40     0295        RTI             ;return from interrupt
            0300 ;
```

```
               0305 ;
               0310 ;******************************
               0315 ;*                            *
               0320 ;* Routine to set and start    *
               0325 ;* pokey hardware timer #2     *
               0330 ;* Clock is reduced to slowest *
               0335 ;* speed of 15 Khz.            *
               0340 ;* value to time is initialized*
               0345 ;* at highest value  $FFFF     *
               0350 ;* (decimal 65535)             *
               0355 ;*                            *
               0360 ;******************************
               0365 ;
               0370 ;
0623 A911      0375 SETUP  LDA  #$11       ;join channels two and 1
0625 8D08D2 0380        STA  AUDCTL     ;and switch clock base from 64 to 15 Khz
0628 A900      0385        LDA  #0
062A 8D03D2 0390        STA  AUDC2      ;set timer channel
062D 8D01D2 0395        STA  AUDC1      ;audio volumes to 0
0630 A9FF      0400        LDA  #$FF       ;LSB of timer duration
0632 8D00D2 0405        STA  AUDF1      ;audf1 holds LSB duration
0635 A9FF      0410        LDA  #$FF       ;MSB of timer duration
0637 8D02D2 0415        STA  AUDF2      ;audf2 holds MSB of duration
063A A911      0420        LDA  #TIME&255 ;LSB of address for timer interrupt
063C 8D1202 0425        STA  VTIMR2     ;pokey timer #2 vector
063F A906      0430        LDA  #TIME/256 ;MSB of address for timer interrupt
0641 8D1302 0435        STA  VTIMR2+1   ;pokey timer#2 vector
0644 78        0440        SEI             ;disable all maskable interrupts
0645 A510      0445        LDA  POKMSK     ;value of interrupt enable state
0647 0902      0450        ORA  #2         ;set bit #2 to enable pokey timer #2
0649 8510      0455        STA  POKMSK     ;place this value in pokey shaddow
064B 8D0ED2 0460        STA  IRQEN      ;also store in pokey hardware register
064E 58        0465        CLI             ;enable interrupts
064F A9FF      0470        LDA  #$FF       ;use non zero value to start timers
0651 8D09D2 0475        STA  STIMER     ;start timer
0654 60        0480        RTS             ;return to controlling routine
0 ERRORS
```

REFERENCES

SOFTWARE TIMERS:

1. Jim Clark, "Atari Timing Delays," Compute, November 1981, Issue 18, PP 104,106

2. Mike Dougherty, "Using Atari's Countdown Timers," Micro, February 1982, No. 45, PP 38-40,82.

3. De Re ATARI, The Atari Program Exchange, P.O. Box 3705, Santa Clara, CA 95055, 1982, PP 8-40 - 8-41

4. TECHNICAL REFERENCE NOTES, ATARI, Inc., Sunnyvale, CA, 94086, "Operating System Users Manual," PP 109-111.

5. Ian Chadwick, Mapping the ATARI, Compute Books, P.O. Box 5406, Greensboro, NC 27403, 1983.

REAL-TIME CLOCK :

1. RICHARD BILLS, "Real-Time Clock on the Atari," Compute May 1981, Issue 12, p 88.

2. Bill Bartlett and Judy Bogart, "Demopac #3, ATARI Product Support, May 1982.

3. Ian Chadwick, Mapping the ATARI, Compute Books, P.O. Box 5406, Greensboro, NC 27403, 1983.

HARDWARE TIMERS :

1. De Re ATARI, The Atari Program Exchange, P.O. Box 3705, Santa Clara, CA 95055, 1982, P 8-39

2. Ian Chadwick, Mapping the ATARI, Compute Books, P.O. Box 5406, Greensboro NC 27403, 1983