Rockwell

R6500 Microcomputer System
APPLICATION NOTE

AIM 65 Program Timer

PURPOSE

Often, while designing a microcomputer system, it is useful to know just how long it takes for a program (or program segment) to execute. As an alternative to actually going through the program counting up instruction cycle times, here is a routine which will measure the execution time of a program up to 65,535 microseconds long.

PROCEDURE

Since the program to be measured gets called like a subroutine from the timing program, it must terminate at the proper point with an RTS instruction ($60). This terminating RTS instruction does not get included in the elapsed execution time measurement.

After the RTS instruction has been placed at the end of the program, or program segment to be measured, start the timing program by pressing the F1 key. When the START ADDRESS? prompt is displayed, enter the hexadecimal starting address of the program you'll be measuring (up to four digits) and then press the RETURN key. If you make a mistake entering the address, you can correct it by backing up the cursor with the DEL (delete) key just as in normal AIM 65 operations.

If the program takes longer than 65,535 microseconds to execute, the error message ***TOO LONG*** will be displayed. Otherwise the execution time will be displayed in microseconds.

OPERATION

The timing program sets up the interval timer (T1) in the R6522 to operate in the free running mode and then puts the address of the return entry point, AFTER, onto the stack before jumping to the program which is to be measured. The NOP instruction at the label AFTER allows for the fact that whenever an RTS instruction is executed, the return address on the stack is incremented by one (see section 8.2 of the R6500 PROGRAMMING MANUAL for an explanation).

The R6522 timer gets read immediately upon return from the program segment that's being measured. If the low byte of the timer is less than 4, then the high byte is corrected because it rolled over while we were reading the low byte. When we apparently add $12 to the low byte of the timer (location $0240), we are really subtracting it because we will be exclusive-ORing both high and low timer values with $FF a few instructions later. The rest of the routine is doing the hexadecimal to decimal conversion on the timer values and printing the elapsed time.

```
2000                ZERO    =$0148              ; IND JMP LOCATION
2000                ;
2000                ; RAM LOCATIONS FOR HEX TO DECIMAL
2000                ; CONVERSION TEMPORARY STORAGE.
2000                ;
2000                        *=$014A
014A                DCMLO   *=*+1
014B                DCMMI   *=*+1
014C                DCMHI   *=*+1
014D                LOWB    *=*+1
014E                HIB     *=*+1
014F                ;
014F                ; AIM MONITOR 6522 LOCATIONS
014F                ;
014F                T1L     =$A804
014F                T1H     =$A805
014F                ACR     =$A80B
014F                IFR     =$A80D
014F                ; STARTING ADDRESS GETS STORED HERE
014F                NUMIN   =$EAB1
```


Specifications subject to change without notice
Document No. R6500 N23
March 1981

```
014F                    NUMLO   =$A41C
014F                    NUMHI   =$A41D
014F                    ; OUTPUT CR AND LF
014F                    CRLF    =$E9F0
014F                    ; OUT HEX NUMBERS IN ASCII
014F                    NUMA    =$EA46
014F                    ;
014F                            *=$010C
010C
010C  4C 00 02                  JMP START
010F                    ;
010F                            *=$0200
0200                    ;
0200  D8                START   CLD
0201                    ; CLEAR DISPLAY
0201  20 F0 E9                  JSR CRLF
0204                    ; PROMPT FOR START ADDRESS
0204  A0 20                     LDY #MSG3-MSG1
0206  20 AC 02                  JSR DISP
0209                    ; GET ADDRESS
0209  20 B1 EA                  JSR NUMIN
020C                    ; PUT START ADDRESS IN INDIRECT JUMP
020C  AD 1C A4                  LDA NUMLO
020F  8D 48 01                  STA ZERO
0212  AD 1D A4                  LDA NUMHI
0215  8D 49 01                  STA ZERO+1
0218                    ; SET UP TIMER TO FREE-RUN
0218  A9 00                     LDA #$00
021A  8D 0B A8                  STA ACR
021D  A2 FF                     LDX #$FF
021F  8E 04 A8                  STX T1L
0222                    ; SET UP STACK SO USER PROGRAM
0222                    ; CAN END WITH AN " RTS "
0222  A9 02                     LDA #>AFTER
0224  48                       PHA
0225  A9 2E                     LDA #<AFTER
0227  48                       PHA
0228                    ; START TIMER
0228  8E 05 A8                  STX T1H
022B                    ; EXECUTE USER PROGRAM
022B  6C 48 01                  JMP (ZERO)
022E                    ;
022E                    ;
022E                    ; USER PROGRAM COMES BACK HERE
022E                    ; AFTER EXECUTING AN " RTS "
022E                    ;
022E  EA                AFTER   NOP
022F  2C 0D A8                  BIT IFR
0232                    ; READ TIMERS
0232  AD 04 A8                  LDA T1L
0235  AC 05 A8                  LDY T1H
0238                    ; ERROR OF TIMER UNDERFLOW
0238                    ; ( > $FFFF MICROSECONDS)
0238  70 39                     BVS ERROR
023A                    ; IF TIMER LOW BYTE IS < 4
023A                    ; THEN WE NEED TO FIX TIMER HIGH BYTE
023A  C9 04                     CMP #$04
023C  B0 01                     BCS OK
023E  C8                        INY
023F  18                OK      CLC
0240                    ; ADD $12 TO FUDGE VALUE TO
0240                    ; IGNORE OVERHEAD
0240  69 12                     ADC #$12
0242  90 01                     BCC NOC
0244  C8                        INY
0245                    ; COMPLIMENT TIMER VALUES
0245  49 FF             NOC     EOR #$FF
0247  48                        PHA
```

```
0248  20 F0 E9          JSR CRLF
024B                  ; OUTPUT A SPACE
024B  20 3E E8          JSR $E83E
024E  98                TYA
024F  49 FF             EOR #$FF
0251  8D 4E 01          STA HIB
0254  68                PLA
0255  8D 4D 01          STA LOWB
0258  20 C2 02          JSR CONVRT
025B  AD 4C 01          LDA DCMHI
025E  20 46 EA          JSR NUMA
0261  AD 4B 01          LDA DCMMI
0264  20 46 EA          JSR NUMA
0267  AD 4A 01          LDA DCMLO
026A  20 46 EA          JSR NUMA
026D  A0 00             LDY #$00
026F  20 AC 02          JSR DISP
0272  60                RTS
0273                  ; DISPLAY ERROR MESSAGE
0273  A0 0E           ERROR  LDY #MSG2-MSG1
0275  20 F0 E9          JSR CRLF
0278  20 AC 02          JSR DISP
027B  60                RTS
027C  20 4D           MSG1   .BYTE ' MICROSECONDS',$A0
0289  A0
028A  2A 2A           MSG2   .BYTE '*** TOO LONG *** ',$A0
029B  A0
029C  53 54           MSG3   .BYTE 'START ADDRESS? ',$A4
02AB  A4
02AC                  ;
02AC                  ;
02AC                  ;
02AC                  ;
02AC                  ;
02AC                  ;
02AC                  ; OUTPUT MESSAGES TO DISPLAY
02AC                  ;
02AC  B9 7C 02        DISP   LDA MSG1,Y
02AF  48                PHA
02B0  29 7F             AND #$7F
02B2  20 7A E9          JSR $E97A
02B5  C8                INY
02B6  68                PLA
02B7  10 F3             BPL DISP
02B9  60                RTS
02BA                  ;
02BA                  ;
02BA                  ; LOOK-UP TABLE FOR HEX TO DECIMAL CONVERSION
02BA                  ;
02BA                  ;
02BA  96              CNVLO  .BYTE $96,$56,$16,$01
02BB  56
02BC  16
02BD  01
02BE  40              CNVHI  .BYTE $40,$02,$00,$00
02BF  02
02C0  00
02C1  00
02C2                  ;
02C2                  ;
02C2                  ;
02C2                  ;
02C2                  ; THIS ROUTINE CONVERTS HEX TO DECIMAL
02C2                  ; FOR THE DISPLAY.
02C2                  ;
02C2  A9 00           CONVRT LDA #$00
02C4  8D 4A 01          STA DCMLO
02C7  8D 4B 01          STA DCMMI
```

```
02CA  8D 4C 01          STA DCMHI
02CD  A2 03             LDX #$03
02CF  AD 4D 01    MORE  LDA LOWB
02D2  29 0F             AND #$0F
02D4  F0 20             BEQ ARND
02D6  A8                TAY
02D7  18                CLC
02D8  F8                SED
02D9  AD 4A 01    HERE  LDA DCMLO
02DC  7D BA 02          ADC CNVLO,X
02DF  8D 4A 01          STA DCMLO
02E2  AD 4B 01          LDA DCMMI
02E5  7D BE 02          ADC CNVHI,X
02E8  8D 4B 01          STA DCMMI
02EB  AD 4C 01          LDA DCMHI
02EE  69 00             ADC #$00
02F0  8D 4C 01          STA DCMHI
02F3  88                DEY
02F4  D0 E3             BNE HERE
02F6  CA          ARND  DEX
02F7  30 0E             BMI FINISH
02F9  A0 04             LDY #$04
02FB  4E 4E 01    NIBRO LSR HIB
02FE  6E 4D 01          ROR LOWB
0301  88                DEY
0302  D0 F7             BNE NIBRO
0304  4C CF 02          JMP MORE
0307  D8          FINISH CLD
0308  60                RTS
0309                    .END
```