NATIONAL RADIO ASTRONOMY OBSERVATORY
CHARLOTTESVILLE, VIRGINIA

ELECTRONICS DIVISION INTERNAL REPORT No. 224

IMPROVED SOFTWARE FOR CONTROLLING THE ADIOS MODULE

L. R. D'ADDARIO

JANUARY 1982

NUMBER OF COPIES: 150

IMPROVED SOFTWARE FOR CONTROLLING THE ADIOS MODULE

Table of Contents

Tables

Figures

IMPROVED SOFTWARE FOR CONTROLLING THE ADIOS MODULE

L. R. D'Addario

I. Introduction

The ADIOS module (Weinreb and Weinreb 1981: Electronics Division Internal Report No. 212) provides a convenient interface between an Apple II computer and laboratory instruments. The referenced report describes the hardware fairly thoroughly, but it includes suggested software which is deficient in several respects: (1) the BASIC programs for analog input and output are unnecessarily complicated; (2) useful features of the AM9513 counter chip, which forms the heart of ADIOS, are not utilized; and (3) only a "free running" mode is discussed, allowing the possibility of "missed data" if the user's program is not fast enough.

In this note, I present several programs which provide improved performance. For a thorough understanding of the details, the reader must study the AM9513 data sheet in some depth. But this is quite complicated and subtle, so such knowledge will not be assumed here. Familiarity with the Apple II and with Internal Report No. 212 will be assumed.

II. Changes to Internal Report No. 212

Sections II and III of the report contain some misleading statements which are clarified here. Also, some significant omissions are included here.

(1) The "obscure code" said to be required to turn on the indicator lights is not necessary. Instead, use the following:

```
POKE 49342,237  :  REM  SET "CALL" BIT
POKE 49342,229  :  REM  CLEAR "CALL" BIT
POKE 49342,235  :  REM  SET "MISS" BIT
POKE 49342,227  :  REM  CLEAR "MISS" BIT
```

For this to work, however, the modes of the counters must be set differently than the report suggests in Appendix A. The better modes are obtained by changing two bytes in ADIOS INITB, or by using the new initialization program described below. In ADIOS INITB, the changes are:

Address	Old Value	New Value	
$1F3E	$A8	$AA	Counter 3 mode
$1F4A	$A8	$AA	Counter 5 mode.

(2) The operation of these indicator LED's is only vaguely described in the report, although it can be deduced by careful study of the schematic. The situation is this: The CALL and MISS bits are outputs of the AM9513, and each may be set and cleared by the commands given above. When MISS is set, the "miss" LED lights for 0.5 s; MISS has no other effect. If CALL is set during COUNT, then the "wait" LED lights until not-COUNT; on the other hand, if not-COUNT occurs before CALL is set, the "dead" LED lights until CALL. Thus, "wait" indicates that Apple is waiting and "dead" indicates that ADIOS is waiting (or was not called when expected). Finally, when CALL is cleared, the "transfer" LED lights for 0.5 s. For proper operation, then, CALL should be set just before the computer begins waiting for COUNT to finish, and should be cleared as soon as all operations required prior to the next count cycle have been completed.

(3) To read in the analog data (two 32-bit values), one can PEEK at all eight bytes sequentially without re-setting the AM9513's data pointer, contrary to what the report says. One must merely POKE the correct readout code into the AM9513:

```
CMD = 49342              : REM  9513 COMMAND PORT
DTA = 49334              : REM  9513 DATA PORT
MK = 256 : MM = 65536    : REM  CONSTANTS
POKE CMD,26              : REM  SET DATA POINTER IN 9513
A = PEEK(DTA) + MK*PEEK(DTA) + MM*(PEEK(DTA) + MK*PEEK(DTA))
B = PEEK(DTA) + MK*PEEK(DTA) + MM*(PEEK(DTA) + MK*PEEK(DTA))
```

(4) The sense of some of the digital outputs is not made clear in the report. Each of the direct outputs, DOØØ thru DO15, is a TTL high for logic "1" and low for logic "0". But the solid state relays connected to DO11 and DO12 are *closed* for logic "0" and *open* for logic "1", whereas the relay drivers connected to DO08-DO11 are *on* (relay energized) for logic "1" and *off* for logic "0".

III. New Software

A. *Initialization*. The assembly language program listed in Fig. 1 will initialize the AM9513 chip and set the count and blank times. Unlike the similar program in Report 212 (Appendix A), all three modes are supported (free-run, software trigger, and external trigger) and the parameters are passed in the CALL statement, rather than having to be POKE'd into place. The code fits into the portion of page 3 which is not used by Applesoft. After BLOADing the code, it can be used from Applesoft by executing

```
CALL 768,COUNT%,BLANK%,MODE%
```

All the parameters must be integer variables, not constants or expressions. Allowable values of MODE% are given in Table I. In the single-cycle modes, the blank/count cycle occurs only once, returning to the non-counting state until a command to start another cycle is sent by the computer. This allows software triggering and external triggering. The different modes also allow control of the clock period used for blank/count timing; COUNT% and BLANK% are in units of this clock period. MODE% may be omitted (but its preceding comma must be included), in which case the default mode is used.

TABLE I. Allowed Values of MODE%

Clock Period (units of CO%,BL%)	Free-Run		Single-Cycle		
10 ms	$0F62 =	3938	$0F42 =	3906	
1	0E62	3682	0E42*	3650*	*=default
100 μs	0D62 =	3426	0D42	3394	
10	0B62	3170	0B42	3138	
1	0A62	2914	0A42	2882	

This program has another major difference from the scheme used in Report 212. The modes of the integrating counters are set for gating directly from the COUNT signal, bypassing the circuitry in ADIOS which introduces an extra clock pulse at the end of count time. This was necessary in order to make the single-cycle modes possible without erroneous extra counts, and it turns out that the extra pulses are not needed in free-run mode either. However, one consequence is that the contents of the integrators are not automatically saved at the end of count time, but this may be done by a simple command (see below). In the free-running mode only, if the program fails to issue a "save" command during blank time, then the next integration will begin adding to the present one. Note that this imposes no additional speed requirements on the program compared with automatic saving, since in either case the program must complete processing of a measurement in less than one blank+count cycle to avoid losing data. In the single-cycle modes, ADIOS does not begin a new integration until commanded, so lost data is not possible no matter how slow the program is.

Unfortunately, a slight hardware modification is necessary to make this work. The COUNT signal must be made available to counters 3, 4 and 5 of the AM9513. To do this, jumper chip pins 4 and 34 to pin 39 and disconnect the wire originally on pin 39, as shown in Fig. 2 (wire wrap pins are D12, B15, and B10). This will not affect any systems which rely on the software described in Report 212.

B. <u>Output</u>. ADIOS provides three outputs to external equipment: two 16-b D-to-A converters and a 16-b digital word. (Some of the latter bits also have special functions.) Since it is awkward and slow to set these values from Applesoft, a simple assembly language routine to do it is included in Fig. 1. To use it,

```
               CALL 909,C%,D%,DO%  : REM  SET ALL THREE OUTPUTS
or             CALL 909,C%,D%,     : REM  SET ANALOG OUTPUTS
or             CALL 909,C%,        : REM  SET OUTPUT "C" ONLY
```

With the latter two forms, the unspecified outputs remain at their previous settings.

C. BASIC Subroutines. The listing of Fig. 3 shows suggested subroutines for using ADIOS from Applesoft, and a sample measurement loop which calls these subroutines. These programs should be mostly self-explanatory, but a few notes are given here.

The various POKE's to CMD are commands to the AM9513 chip, and are described in REMarks. The full command set, along with other important reference data on the AM9513, is given in the Appendix; this is the most useful material in the AM9513 data sheet, but was not among the pages reproduced in Report 212.

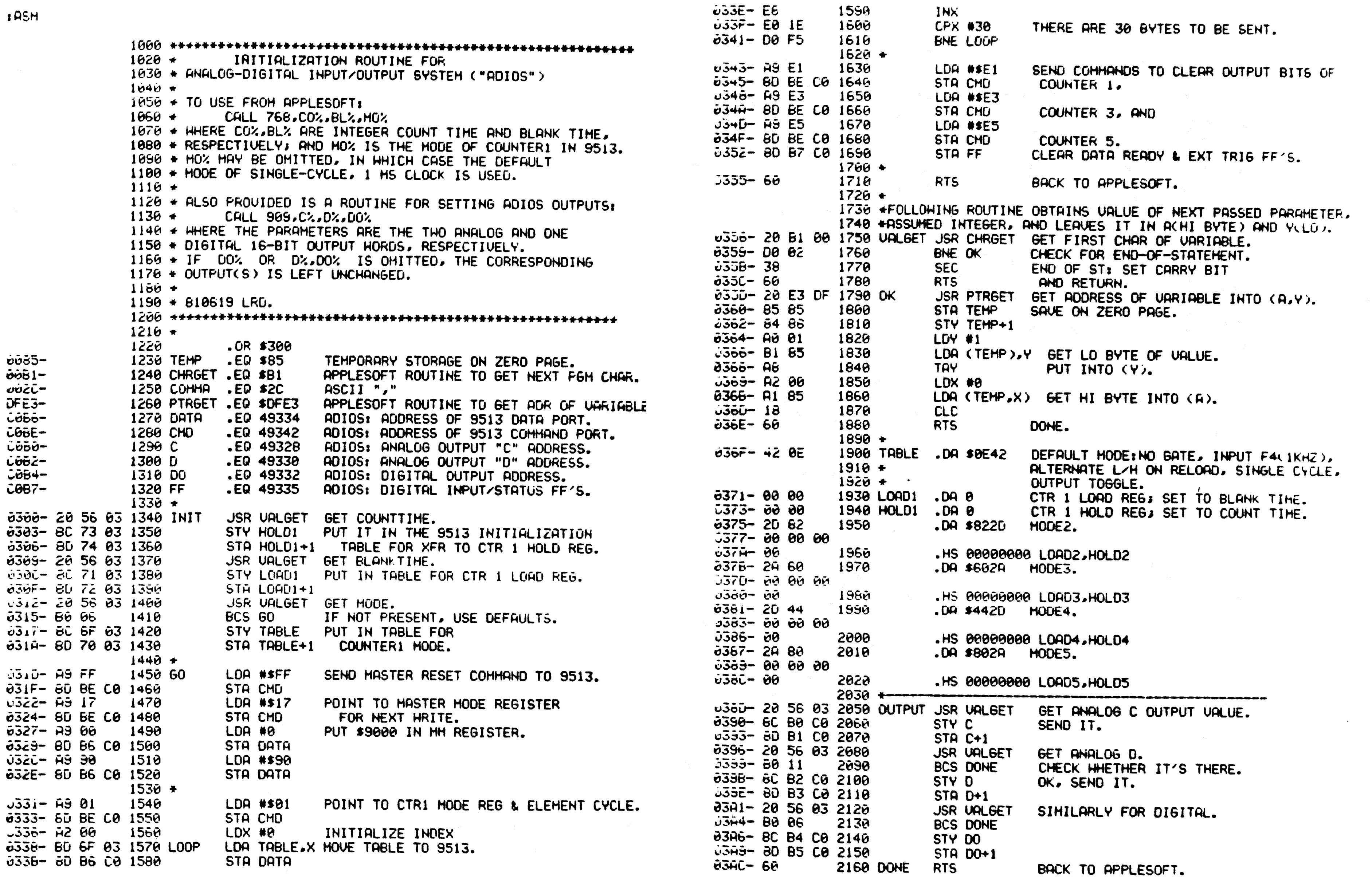
Line 260 operates the "miss" light. Report 212 used ten lines of code to do the same thing. This line can be included for all three modes, but is only meaningful in free-run. Line 320 is all that one needs to implement external trigger mode, but one might want to precede it by a line similar to 260 to flash the "miss" light if an external trigger is missed. A low-to-high transition at the "ext trig" connector sets the EXT TRIG flip flop, connected to bit 6 of the digital input byte at address FF=49335.

In the single-cycle modes, the cycle starts when line 350 is executed. This line must be omitted for free-run mode. Blank time occurs first, followed by count time. Thus, blank time can be set to allow settling of the output signals just sent, and need not be any longer. One warning, however: due to a quirk in the AM9513, BLANK% and COUNT% may be no smaller than 2; using 0 or 1 gives strange results. This allows times as small as 2 microseconds.

Further details of the timing of the various modes are given in Fig. 4.

IV. Further Improvements

To obtain the highest possible speed of execution, both of the BASIC subroutines in Fig. 3 should be implemented in assembly language. This should be fairly easy to do.



```
:ASM

                 1000 ***************************************************
                 1020 *        INITIALIZATION ROUTINE FOR
                 1030 * ANALOG-DIGITAL INPUT/OUTPUT SYSTEM ("ADIOS")
                 1040 *
                 1050 * TO USE FROM APPLESOFT:
                 1060 *        CALL 768,CO%,BL%,MO%
                 1070 * WHERE CO%,BL% ARE INTEGER COUNT TIME AND BLANK TIME,
                 1080 * RESPECTIVELY; AND MO% IS THE MODE OF COUNTER1 IN 9513.
                 1090 * MO% MAY BE OMITTED, IN WHICH CASE THE DEFAULT
                 1100 * MODE OF SINGLE-CYCLE, 1 MS CLOCK IS USED.
                 1110 *
                 1120 * ALSO PROVIDED IS A ROUTINE FOR SETTING ADIOS OUTPUTS:
                 1130 *        CALL 909,C%,D%,DO%
                 1140 * WHERE THE PARAMETERS ARE THE TWO ANALOG AND ONE
                 1150 * DIGITAL 16-BIT OUTPUT WORDS, RESPECTIVELY.
                 1160 * IF  DO%  OR  D%,DO%  IS OMITTED, THE CORRESPONDING
                 1170 * OUTPUT(S) IS LEFT UNCHANGED.
                 1180 *
                 1190 * 810619 LRD.
                 1200 **************************************************
                 1210 *
                 1220        .OR $300
0085-            1230 TEMP   .EQ $85      TEMPORARY STORAGE ON ZERO PAGE.
00B1-            1240 CHRGET .EQ $B1      APPLESOFT ROUTINE TO GET NEXT PGM CHAR.
002C-            1250 COMMA  .EQ $2C      ASCII ","
DFE3-            1260 PTRGET .EQ $DFE3    APPLESOFT ROUTINE TO GET ADR OF VARIABLE
C086-            1270 DATA   .EQ 49334    ADIOS: ADDRESS OF 9513 DATA PORT.
C08E-            1280 CMD    .EQ 49342    ADIOS: ADDRESS OF 9513 COMMAND PORT.
C080-            1290 C      .EQ 49328    ADIOS: ANALOG OUTPUT "C" ADDRESS.
C082-            1300 D      .EQ 49330    ADIOS: ANALOG OUTPUT "D" ADDRESS.
C084-            1310 DO     .EQ 49332    ADIOS: DIGITAL OUTPUT ADDRESS.
C087-            1320 FF     .EQ 49335    ADIOS: DIGITAL INPUT/STATUS FF'S.
                 1330 *
0300- 20 56 03 1340 INIT    JSR VALGET   GET COUNTTIME.
0303- 8C 73 03 1350         STY HOLD1    PUT IT IN THE 9513 INITIALIZATION
0306- 8D 74 03 1360         STA HOLD1+1    TABLE FOR XFR TO CTR 1 HOLD REG.
0309- 20 56 03 1370         JSR VALGET   GET BLANKTIME.
030C- 8C 71 03 1380         STY LOAD1    PUT IN TABLE FOR CTR 1 LOAD REG.
030F- 8D 72 03 1390         STA LOAD1+1
0312- 20 56 03 1400         JSR VALGET   GET MODE.
0315- B0 06    1410         BCS GO       IF NOT PRESENT, USE DEFAULTS.
0317- 8C 6F 03 1420         STY TABLE    PUT IN TABLE FOR
031A- 8D 70 03 1430         STA TABLE+1    COUNTER1 MODE.
               1440 *
031D- A9 FF    1450 GO      LDA #$FF     SEND MASTER RESET COMMAND TO 9513.
031F- 8D BE C0 1460         STA CMD
0322- A9 17    1470         LDA #$17     POINT TO MASTER MODE REGISTER
0324- 8D BE C0 1480         STA CMD        FOR NEXT WRITE.
0327- A9 00    1490         LDA #0       PUT $9000 IN MM REGISTER.
0329- 8D B6 C0 1500         STA DATA
032C- A9 90    1510         LDA #$90
032E- 8D B6 C0 1520         STA DATA
               1530 *
0331- A9 01    1540         LDA #$01     POINT TO CTR1 MODE REG & ELEMENT CYCLE.
0333- 8D BE C0 1550         STA CMD
0336- A2 00    1560         LDX #0       INITIALIZE INDEX
0338- BD 6F 03 1570 LOOP    LDA TABLE,X MOVE TABLE TO 9513.
033B- 8D B6 C0 1580         STA DATA
033E- E8       1590         INX
033F- E0 1E    1600         CPX #30      THERE ARE 30 BYTES TO BE SENT.
0341- D0 F5    1610         BNE LOOP
               1620 *
0343- A9 E1    1630         LDA #$E1     SEND COMMANDS TO CLEAR OUTPUT BITS OF
0345- 8D BE C0 1640         STA CMD       COUNTER 1,
0348- A9 E3    1650         LDA #$E3
034A- 8D BE C0 1660         STA CMD       COUNTER 3, AND
034D- A9 E5    1670         LDA #$E5
034F- 8D BE C0 1680         STA CMD       COUNTER 5.
0352- 8D B7 C0 1690         STA FF       CLEAR DATA READY & EXT TRIG FF'S.
               1700 *
0355- 60       1710         RTS          BACK TO APPLESOFT.
               1720 *
               1730 *FOLLOWING ROUTINE OBTAINS VALUE OF NEXT PASSED PARAMETER.
               1740 *ASSUMED INTEGER, AND LEAVES IT IN A(HI BYTE) AND Y(LO).
0356- 20 B1 00 1750 VALGET JSR CHRGET  GET FIRST CHAR OF VARIABLE.
0359- D0 02    1760         BNE OK       CHECK FOR END-OF-STATEMENT.
035B- 38       1770         SEC          END OF ST: SET CARRY BIT
035C- 60       1780         RTS           AND RETURN.
035D- 20 E3 DF 1790 OK      JSR PTRGET  GET ADDRESS OF VARIABLE INTO (A,Y).
0360- 85 85    1800         STA TEMP     SAVE ON ZERO PAGE.
0362- 84 86    1810         STY TEMP+1
0364- A0 01    1820         LDY #1
0366- B1 85    1830         LDA (TEMP),Y  GET LO BYTE OF VALUE.
0368- A8       1840         TAY             PUT INTO (Y).
0369- A2 00    1850         LDX #0
036B- A1 85    1860         LDA (TEMP,X)  GET HI BYTE INTO (A).
036D- 18       1870         CLC
036E- 60       1880         RTS          DONE.
               1890 *
036F- 42 0E    1900 TABLE   .DA $0E42    DEFAULT MODE:NO GATE, INPUT F4(1KHZ),
               1910 *                    ALTERNATE L/H ON RELOAD, SINGLE CYCLE,
               1920 *                    OUTPUT TOGGLE.
0371- 00 00    1930 LOAD1   .DA 0        CTR 1 LOAD REG; SET TO BLANK TIME.
0373- 00 00    1940 HOLD1   .DA 0        CTR 1 HOLD REG; SET TO COUNT TIME.
0375- 2D 82    1950         .DA $822D    MODE2.
0377- 00 00 00
037A- 00       1960         .HS 00000000 LOAD2,HOLD2
037B- 2A 60    1970         .DA $602A    MODE3.
037D- 00 00 00
0380- 00       1980         .HS 00000000 LOAD3,HOLD3
0381- 2D 44    1990         .DA $442D    MODE4.
0383- 00 00 00
0386- 00       2000         .HS 00000000 LOAD4,HOLD4
0387- 2A 80    2010         .DA $802A    MODE5.
0389- 00 00 00
038C- 00       2020         .HS 00000000 LOAD5,HOLD5
               2030 *-----------------------------------------
038D- 20 56 03 2050 OUTPUT JSR VALGET   GET ANALOG C OUTPUT VALUE.
0390- 8C B0 C0 2060         STY C        SEND IT.
0393- 8D B1 C0 2070         STA C+1
0396- 20 56 03 2080         JSR VALGET   GET ANALOG D.
0399- B0 11    2090         BCS DONE     CHECK WHETHER IT'S THERE.
039B- 8C B2 C0 2100         STY D        OK, SEND IT.
039E- 8D B3 C0 2110         STA D+1
03A1- 20 56 03 2120         JSR VALGET   SIMILARLY FOR DIGITAL.
03A4- B0 06    2130         BCS DONE
03A6- 8C B4 C0 2140         STY DO
03A9- 8D B5 C0 2150         STA DO+1
03AC- 60       2160 DONE    RTS          BACK TO APPLESOFT.
```

Fig. 1. Listing of assembly language subroutines.

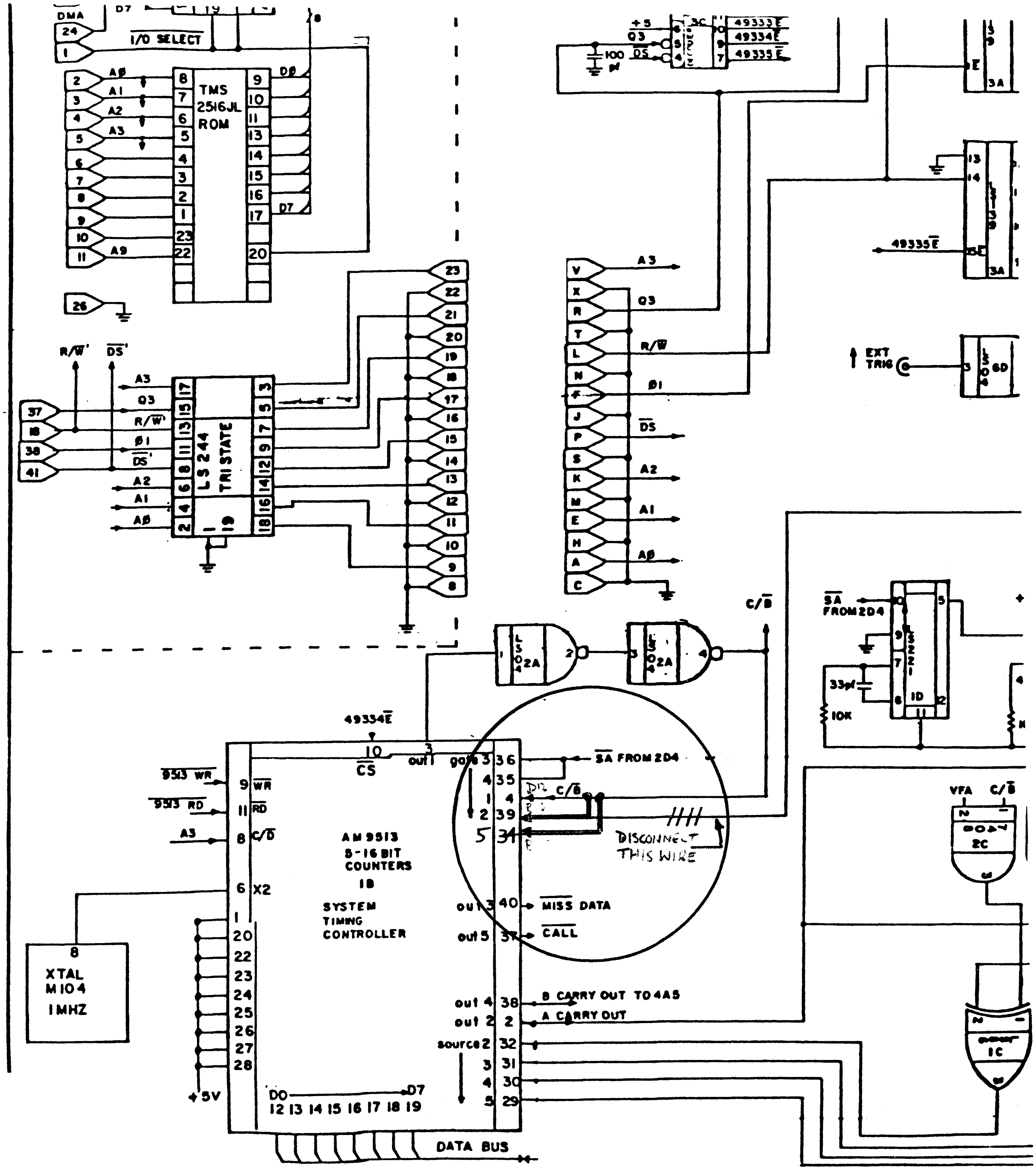
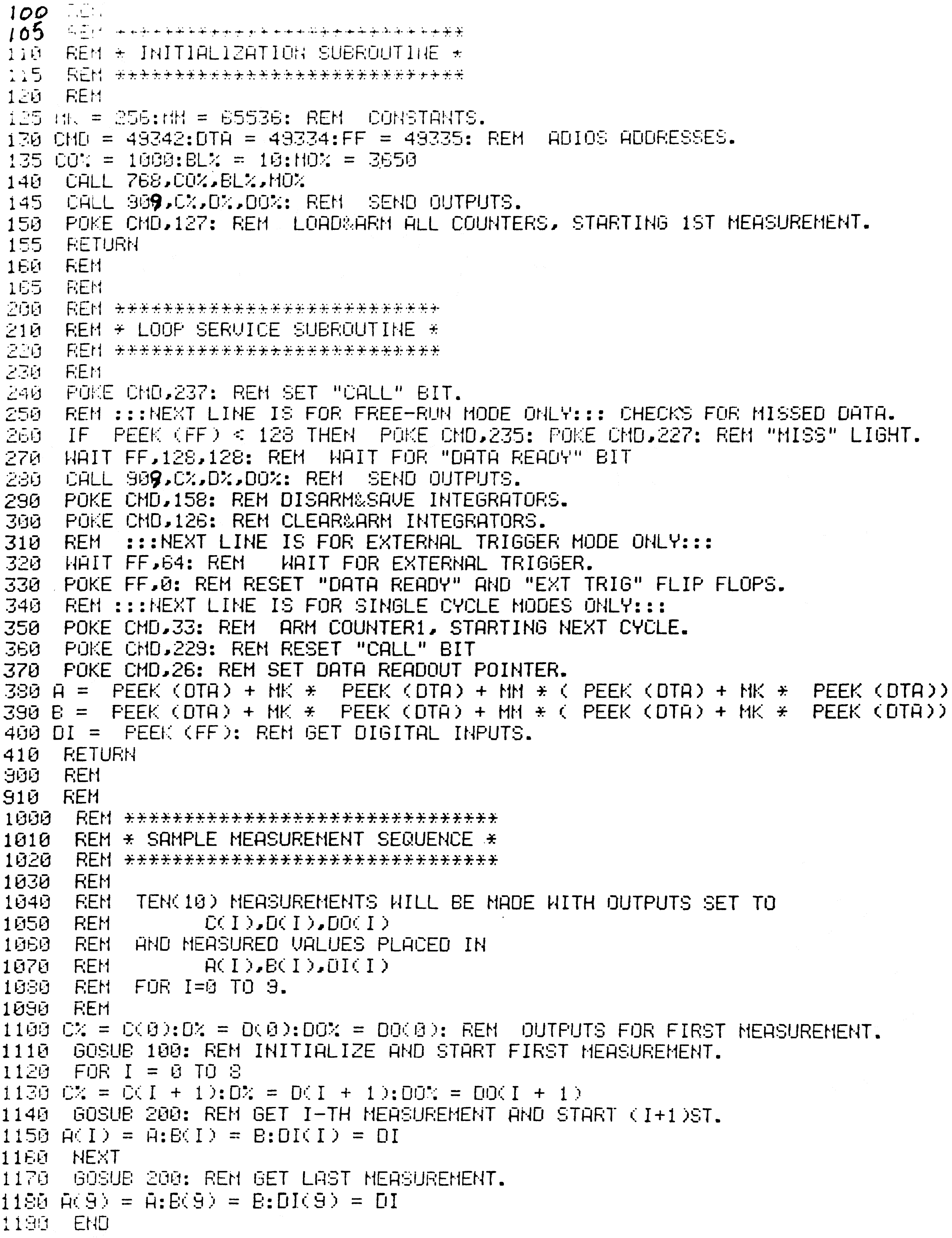
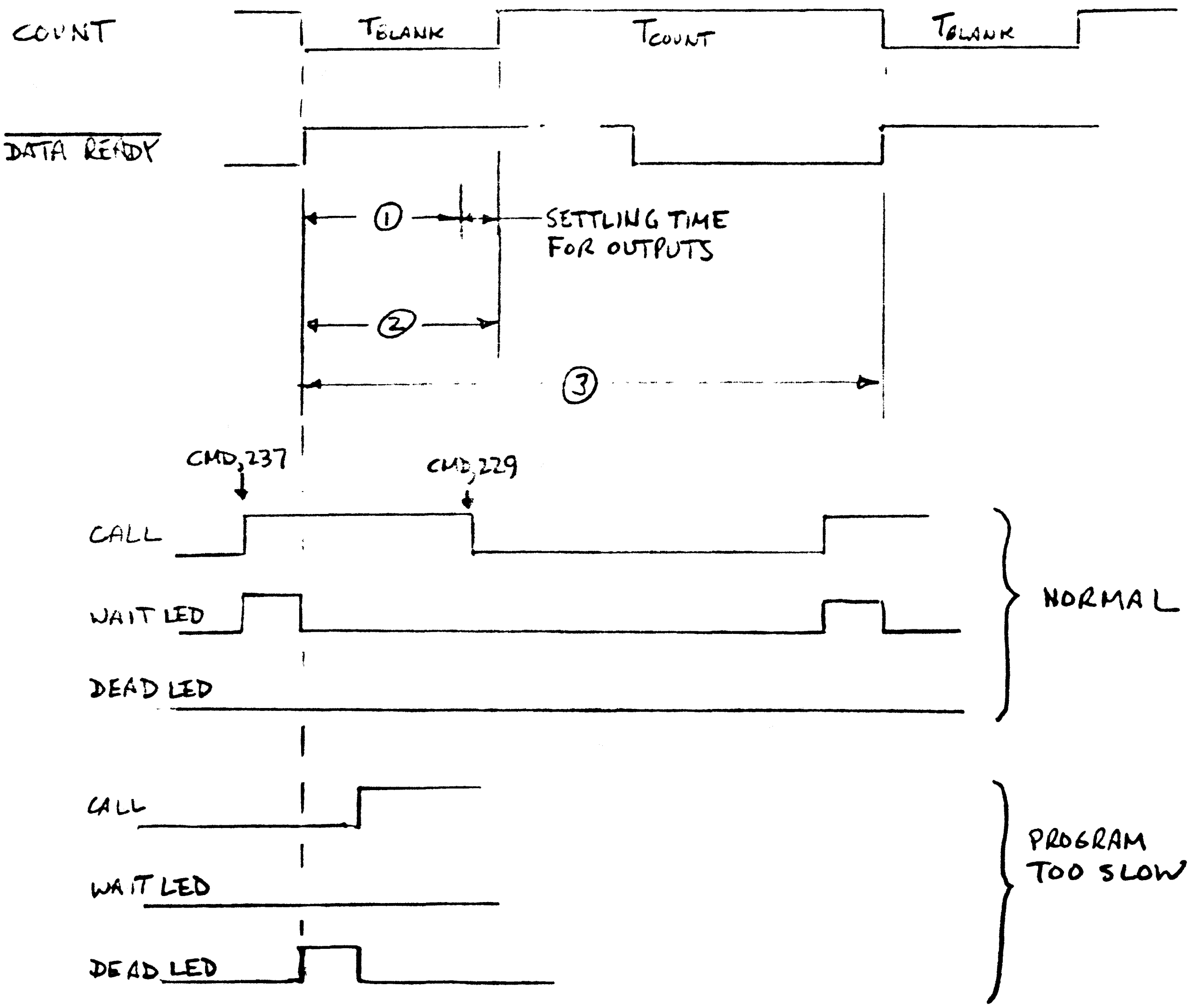


Figure 2. Wiring change required in ADIOS module.



```
100  REM
105  REM ***************************
110  REM * INITIALIZATION SUBROUTINE *
115  REM ***************************
120  REM
125 MK = 256:MM = 65536: REM  CONSTANTS.
130 CMD = 49342:DTA = 49334:FF = 49335: REM  ADIOS ADDRESSES.
135 CO% = 1000:BL% = 10:MO% = 3650
140  CALL 768,CO%,BL%,MO%
145  CALL 909,C%,D%,DO%: REM  SEND OUTPUTS.
150  POKE CMD,127: REM  LOAD&ARM ALL COUNTERS, STARTING 1ST MEASUREMENT.
155  RETURN
160  REM
165  REM
200  REM *************************
210  REM * LOOP SERVICE SUBROUTINE *
220  REM *************************
230  REM
240  POKE CMD,237: REM SET "CALL" BIT.
250  REM :::NEXT LINE IS FOR FREE-RUN MODE ONLY::: CHECKS FOR MISSED DATA.
260  IF  PEEK (FF) < 128 THEN  POKE CMD,235: POKE CMD,227: REM "MISS" LIGHT.
270  WAIT FF,128,128: REM  WAIT FOR "DATA READY" BIT
280  CALL 909,C%,D%,DO%: REM  SEND OUTPUTS.
290  POKE CMD,158: REM DISARM&SAVE INTEGRATORS.
300  POKE CMD,126: REM CLEAR&ARM INTEGRATORS.
310  REM  :::NEXT LINE IS FOR EXTERNAL TRIGGER MODE ONLY:::
320  WAIT FF,64: REM   WAIT FOR EXTERNAL TRIGGER.
330  POKE FF,0: REM RESET "DATA READY" AND "EXT TRIG" FLIP FLOPS.
340  REM :::NEXT LINE IS FOR SINGLE CYCLE MODES ONLY:::
350  POKE CMD,33: REM  ARM COUNTER1, STARTING NEXT CYCLE.
360  POKE CMD,229: REM RESET "CALL" BIT
370  POKE CMD,26: REM SET DATA READOUT POINTER.
380 A =  PEEK (DTA) + MK *  PEEK (DTA) + MM * ( PEEK (DTA) + MK *  PEEK (DTA))
390 B =  PEEK (DTA) + MK *  PEEK (DTA) + MM * ( PEEK (DTA) + MK *  PEEK (DTA))
400 DI =  PEEK (FF): REM GET DIGITAL INPUTS.
410  RETURN
900  REM
910  REM
1000  REM *****************************
1010  REM * SAMPLE MEASUREMENT SEQUENCE *
1020  REM *****************************
1030  REM
1040  REM  TEN(10) MEASUREMENTS WILL BE MADE WITH OUTPUTS SET TO
1050  REM          C(I),D(I),DO(I)
1060  REM  AND MEASURED VALUES PLACED IN
1070  REM          A(I),B(I),DI(I)
1080  REM  FOR I=0 TO 9.
1090  REM
1100 C% = C(0):D% = D(0):DO% = DO(0): REM  OUTPUTS FOR FIRST MEASUREMENT.
1110  GOSUB 100: REM INITIALIZE AND START FIRST MEASUREMENT.
1120  FOR I = 0 TO 8
1130 C% = C(I + 1):D% = D(I + 1):DO% = DO(I + 1)
1140  GOSUB 200: REM GET I-TH MEASUREMENT AND START (I+1)ST.
1150 A(I) = A:B(I) = B:DI(I) = DI
1160  NEXT
1170  GOSUB 200: REM GET LAST MEASUREMENT.
1180 A(9) = A:B(9) = B:DI(9) = DI
1190  END
```

Fig. 3. BASIC subroutines.



① Program must set outputs for next measurement.

② Program must issue command to save integrator contents from last measurement.

③ Program must clear the $\overline{\text{DATA READY}}$ bit and complete processing of last measurement.

Fig. 4(a). Free-run mode timing.

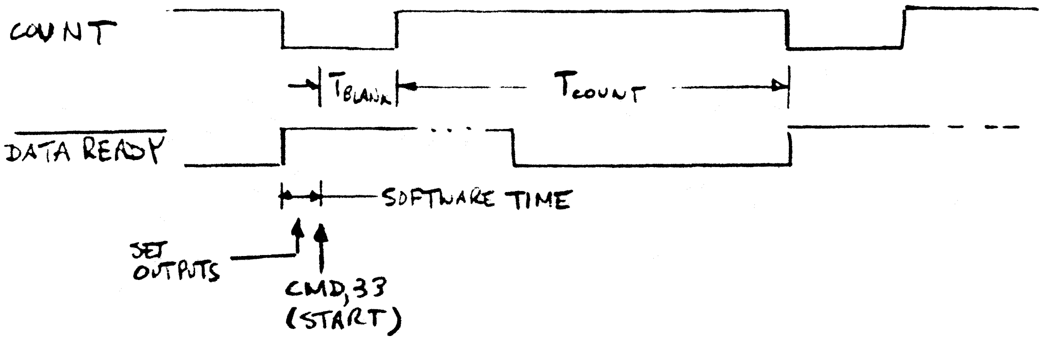


Fig. 4(b). Software trigger mode timing.

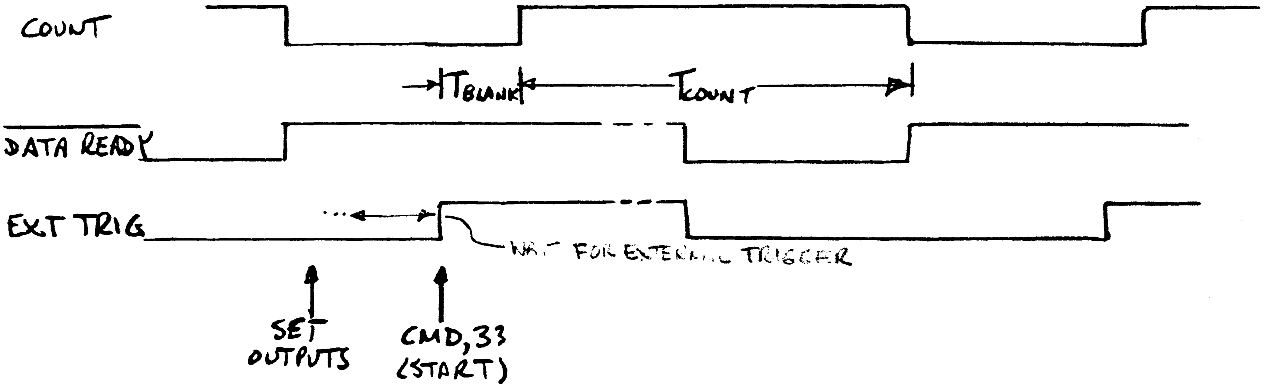
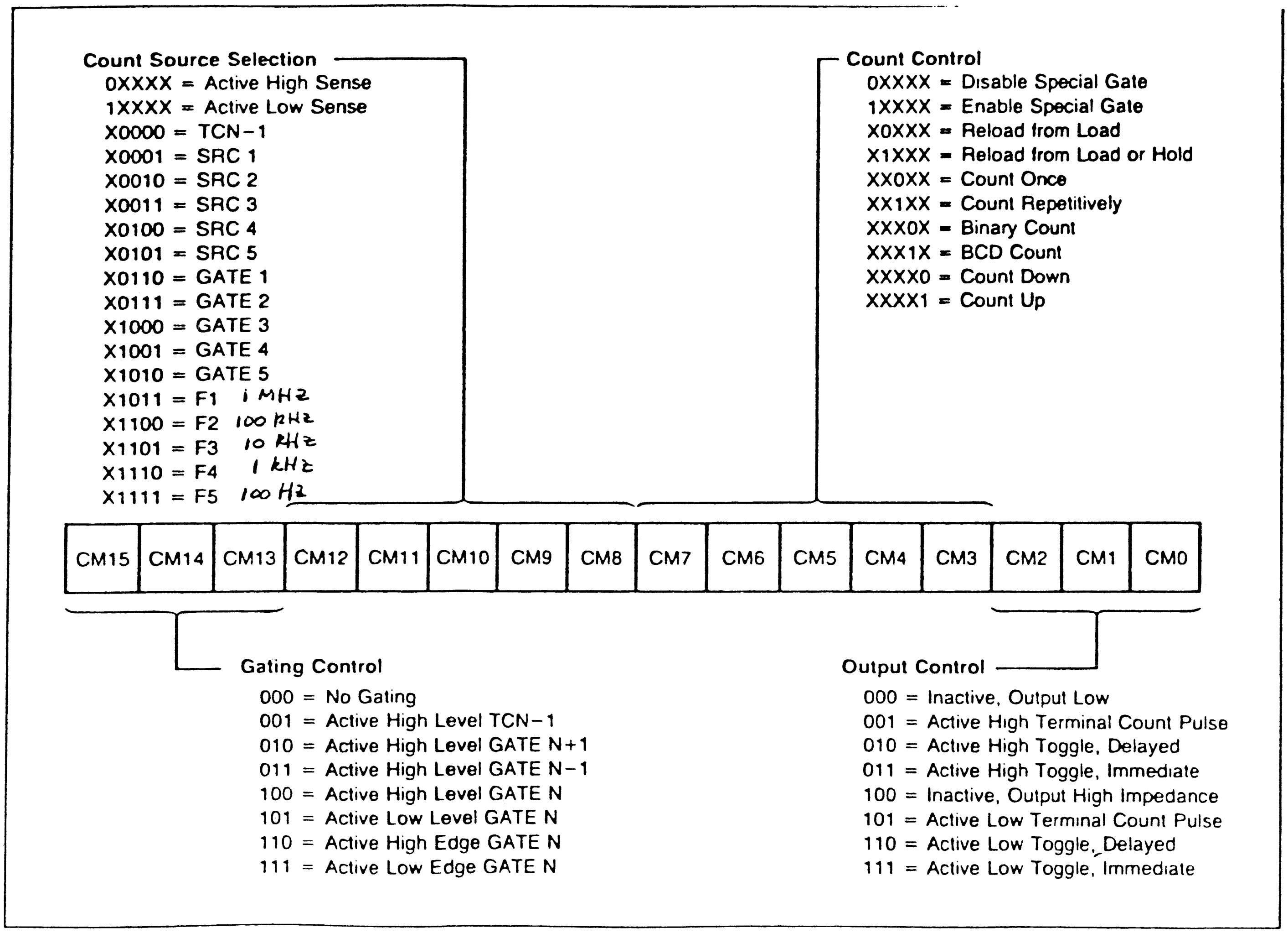


Fig. 4 (c). External trigger mode timing.

[illegible]								APPENDIX
C7	C6	C5	C4	C3	C2	C1	C0	Command Description
0	0	0	E2	E1	G4	G2	G1	Load Data Pointer register with contents of E and G fields. (G ≠ 000, G ≠ 110)
0	0	1	S5	S4	S3	S2	S1	Arm counting for all selected counters
0	1	0	S5	S4	S3	S2	S1	Load contents of specified source into all selected counters
0	1	1	S5	S4	S3	S2	S1	Load and Arm all selected counters
1	0	0	S5	S4	S3	S2	S1	Disarm and Save all selected counters
1	0	1	S5	S4	S3	S2	S1	Save all selected counters in hold register
1	1	0	S5	S4	S3	S2	S1	Disarm all selected counters
1	1	1	0	1	N4	N2	N1	Set output bit N (001 ≤ N ≤ 101)
1	1	1	0	0	N4	N2	N1	Clear output bit N (001 ≤ N ≤ 101)
1	1	1	1	0	N4	N2	N1	Step counter N (001 ≤ N ≤ 101)
1	1	1	0	1	0	0	0	Set MM14 (Disable Data Pointer Sequencing)
1	1	1	0	1	1	1	0	Set MM12 (Gate off FOUT)
1	1	1	0	1	1	1	1	Set MM13 (Enter 16-bit bus mode)
1	1	1	0	0	0	0	0	Clear MM14 (Enable Data Pointer Sequencing)
1	1	1	0	0	1	1	0	Clear MM12 (Gate on FOUT)
1	1	1	0	0	1	1	1	Clear MM13 (Enter 8-bit bus mode)
1	1	1	1	1	1	1	1	Master reset

Am9513 Command Summary.



Count Source Selection

0XXXX = Active High Sense
1XXXX = Active Low Sense
X0000 = TCN−1
X0001 = SRC 1
X0010 = SRC 2
X0011 = SRC 3
X0100 = SRC 4
X0101 = SRC 5
X0110 = GATE 1
X0111 = GATE 2
X1000 = GATE 3
X1001 = GATE 4
X1010 = GATE 5
X1011 = F1 1 MHz
X1100 = F2 100 kHz
X1101 = F3 10 kHz
X1110 = F4 1 kHz
X1111 = F5 100 Hz

Count Control

0XXXX = Disable Special Gate
1XXXX = Enable Special Gate
X0XXX = Reload from Load
X1XXX = Reload from Load or Hold
XX0XX = Count Once
XX1XX = Count Repetitively
XXX0X = Binary Count
XXX1X = BCD Count
XXXX0 = Count Down
XXXX1 = Count Up

CM15	CM14	CM13	CM12	CM11	CM10	CM9	CM8	CM7	CM6	CM5	CM4	CM3	CM2	CM1	CM0

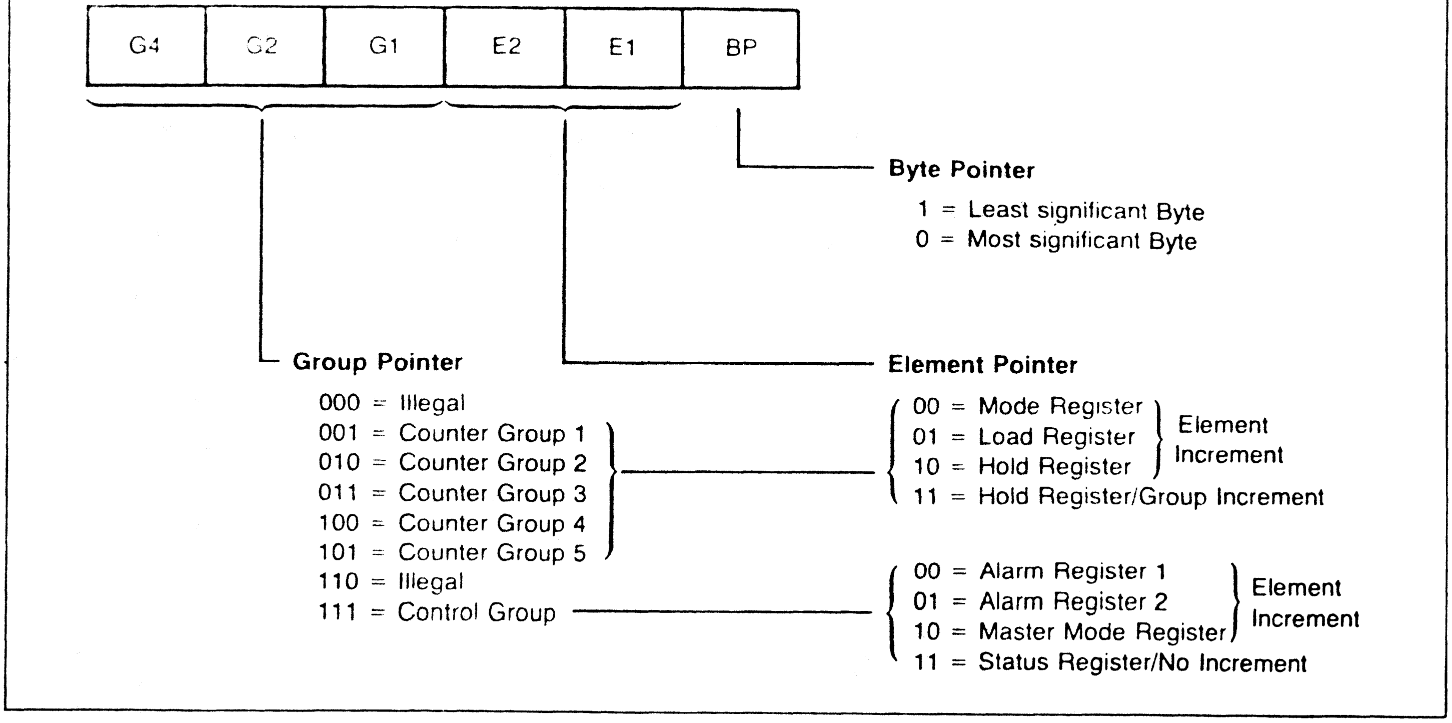
Gating Control

000 = No Gating
001 = Active High Level TCN−1
010 = Active High Level GATE N+1
011 = Active High Level GATE N−1
100 = Active High Level GATE N
101 = Active Low Level GATE N
110 = Active High Edge GATE N
111 = Active Low Edge GATE N

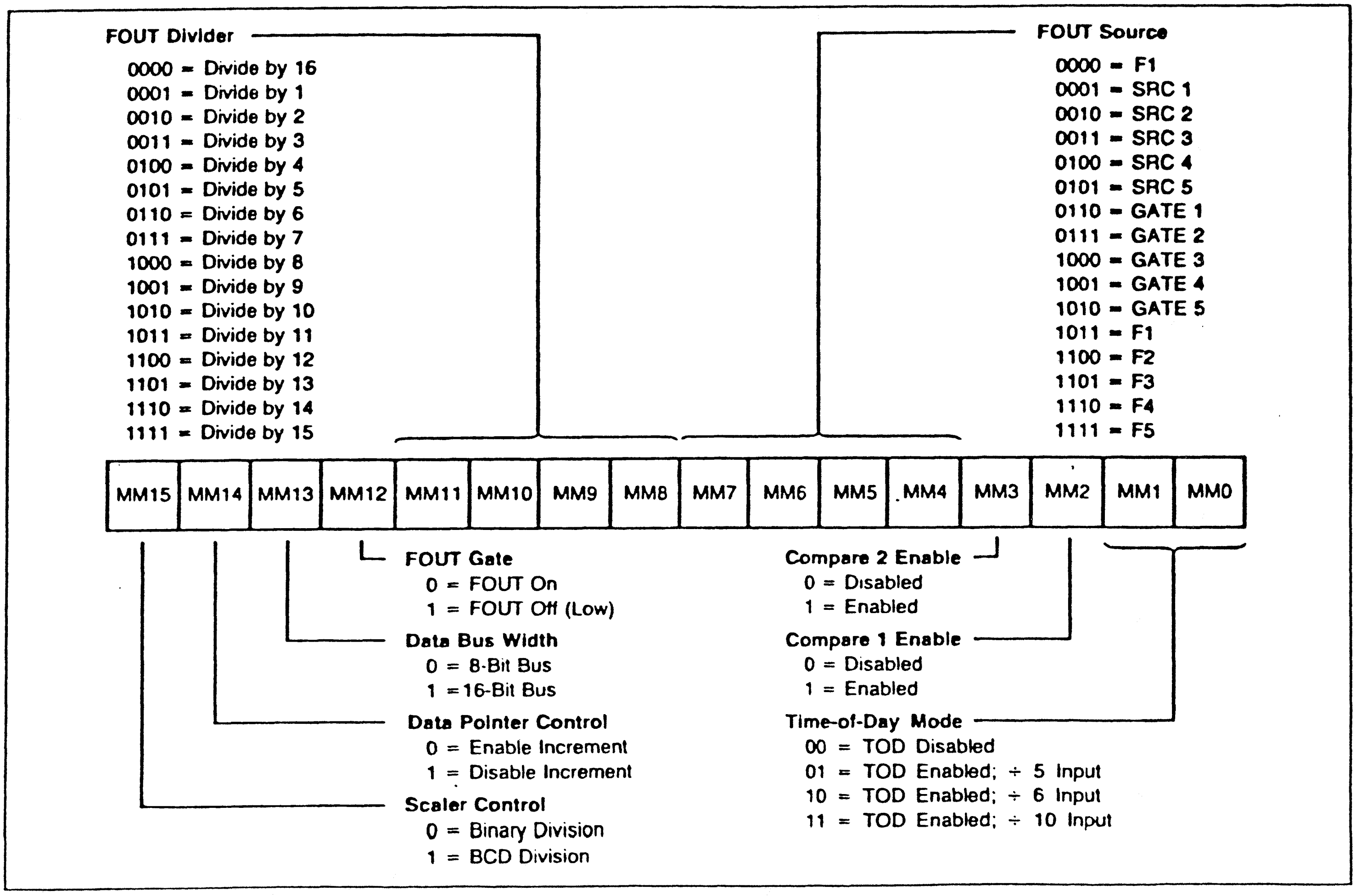
Output Control

000 = Inactive, Output Low
001 = Active High Terminal Count Pulse
010 = Active High Toggle, Delayed
011 = Active High Toggle, Immediate
100 = Inactive, Output High Impedance
101 = Active Low Terminal Count Pulse
110 = Active Low Toggle, Delayed
111 = Active Low Toggle, Immediate

Counter Mode Register Bit Assignments.



Data Pointer Counter.



Master Mode Register Bit Assignments.

NATIONAL RADIO ASTRONOMY OBSERVATORY

Addition to EDIR No. 224

IMPROVED SOFTWARE FOR CONTROLLING THE ADIOS MODULE

L. R. D'Addario
S. Keller

March 30, 1982

I. New ADIOS Subroutines

The BASIC subroutines described in Report 224 for initialization and loop service of the ADIOS module have now been implemented in assembly language. These new subroutines have been incorporated in the binary library (see Report No. 225), versions 3.0 and 3.1, replacing subroutines "ADIOS" and "ADOUT" of earlier versions.

Initialization

Call thru indirect entry: AINIT=5126: CALL AINIT,MODE%(M),OUT%(N)

Call thru LIBENT: LIB=5171 : CALL LIB"AINIT"MODE%(M),OUT%(N)

Parameters:

MODE%(M) = mode from table below, or zero for default mode.

MODE%(M+1) = blank time*, in units dependent on mode (default: msec)

MODE%(M+2) = count time*, in units dependent on mode (default: msec)

OUT%(N) = 16 bit value for analog output "C"

OUT%(N+1) = 16 bit value for analog output "D"

OUT%(N+2) = 16 bit value for digital outputs

Description:

Any measurement in progress is stopped, the counters are initialized in the specified mode, count and blank times are set, the outputs are set to the specified values, and a new measurement is started (beginning with blank time).

*These values must be ≥ 3.

Loop Service

Call thru indirect entry: ASERV=5129: CALL ASERV,IN(I),OUT%(N)

Call thru LIBENT: LIB=5171 : CALL LIB"ASERV"IN(I),OUT%(N)

Parameters:

IN(I) = analog channel "A" count (32 bits, converted to floating point)

IN(I+1) = analog channel "B" count

IN(I+2) = digital input byte (8 bits, converted to floating point)

OUT%(M...M+2) = same as "AINIT"

Description:

Waits until the current measurement is complete, if necessary. In external trigger modes, waits for the external trigger, if necessary. Reads inputs for measurement just completed and sets specified outputs for next measurement. Starts next measurement (except free run modes). Operates "transfer," "wait," and "dead" lights as appropriate in all modes, and "miss" light in free run and external trigger modes. Executes in less than 9 msec.

Allowed Values of Mode Code

Clock Period	Code Values					
	Software Trigger		External Trigger		Free Run	
10 ms	$F42 =	3906	$F43 =	3907	$F62 =	3938
1 ms	E42*	3650*	E43	3651	E62	3682
100 μs	D42	3394	D43	3395	D62	3426
10 μs	C42	3138	C43	3139	C62	3170
1 μs	B42	2882	B43	2883	B62	2914

*Code=0 also selects this mode.