Computers & Chemistry Vol. 8, No. 3, pp. 169–173, 1984
Printed in the U.S.A.

0097-8485/84 $3.00 + .00
Pergamon Press Ltd.

A FAST, LOW COST DATA ACQUISITION SYSTEM

T. G. M. VAN DEN BELT* and H. ERKELENS
Department of Inorganic Chemistry, State University of Utrecht, Croesestraat 77A, 3522 AD Utrecht, The Netherlands

(*Received* 22 *June* 1983)

Abstract—A low cost data acquisition system is described for the Apple][microcomputer. Two additional cards are necessary, a commercial available timer card and a home-made 12 bits ADC card. Data can be recorded through 4 differential channels with resolutions ranging from 1.2 to 1.2×10^{-3} mV/bit. Time intervals between the measurements can range from 1/4 ms to 2 hours.

INTRODUCTION

In many experimental situations fast changing effects have to be measured that can only be analyzed later on. For this purpose transient recorders were developed and after recording, the data was transferred to a chart recorder or to a computer. The transfer to a chart recorder makes the analyzing still very cumbersome while the data transfer to a computer needs expensive interfacing. Secondly the possibilities of these systems are limited with regards to the timing. The timing interval is constant and cannot be changed during the measurement. We will describe here the hardware and software of an inexpensive data acquisition system that can tailor the timing of the measurements to the users need. The system is based on an Apple][microcomputer with a Timer card and a 12 bits ADC card. The software was written mainly in assembly language to reach the necessary speed. An example of a decay measurement with different timing intervals will be given.

DESCRIPTION

Hardware

We used a common Apple][microcomputer with two extra cards. Of course the software can easily be used on any microcomputer based on the 6502 microprocessor. Only the memory addresses have to be changed. The Timer card is the commercially available 7440A programmable timer module from California Computer systems which is designed around the 6840 IC device. This timer consists of three independent timers. Each timer can be controlled by the internal Apple clock or by an external signal. We used timers 2 and 3, timer 2 controlled by the Apple clock and timer 3 controlled by the output from timer 2. Therefore the output from timer 2 must be linked to the clock-input of timer 3. In this way time cycles from 1/4 ms to 2 hours can be obtained. The 12 bits Analog to Digital converter card is not commercially available at the moment. The card consists of a 12 bits ADC, the Analog Devices 574A, a programmable differential amplifier, the Burr Brown 3606, a 4 channel dual multiplexer, the Analog Devices 7502, a Sample & Hold and some support logic. An outline of the card is given in Fig. 1. The control of the card is done through two addresses, $C0X0 and $C0X1 where X = 8 + (slot number in Apple):

WRITE TO $C0X0—The channel number and the gain factor are loaded. The channel number ranges from 0 to 3 and the gain factor ranges from 0 to $A. The low nibble of the databyte must contain the channel number, the high nibble the gain factor. Table 1 gives a review of the gain factors, the maximum input and the resolution.

WRITE TO $C0X1—Start a conversion. The conversion time is approximately 0.030 ms.

READ FROM $C0X0—Read the 8 most significant bytes.

READ FROM $C0X1—Read the 4 least significant bytes. The low nibble has no meaning.

The software assumes that the timer card is in slot 1 and the ADC card in slot 2 of the Apple.

Software

The assembly language part of the software consists of three parts and is given in listing 1. Parts 1 and 2 regulate respectively the timer and the ADC, while part 3 does the overhead like initialisations and storing the results in its proper place. Parts 1 and 2 are made up of completely self consistent subroutines and can be used in other programs as well. They are more or less system software while part 3 is application software. Both part 1 and part 2 use codes which are stored on the zero-page of the memory. The timer subroutines only use a time code which has to be stored at memory location $1C or decimal 28. This code determines the time cycle according to Table 2. Of the three subroutines ITIME, STIME and WTIME the first initializes the timer, the second starts a time cycle while the third waits for the time cycle to end. ITIME only has to run once. Between STIME and WTIME other subroutines like those in part 2 can run when the user is sure this subroutine will finish before the time cycle ends. The sequence

```
LBL1  JSR STIME
      JSR ADC12
      JSR WTIME
      JMP LBL1
```

will result in a series of measurements with exact time intervals as determined by the time code.

The subroutines to control the ADC card can be found in part 2. Two input control codes must be present in $06 and $07, in $06 the gain number and in $07 the channel number. The results of the subroutines are to be found in $08 and $09. $08 contains the 8 most significant bytes and $09 contains the 4

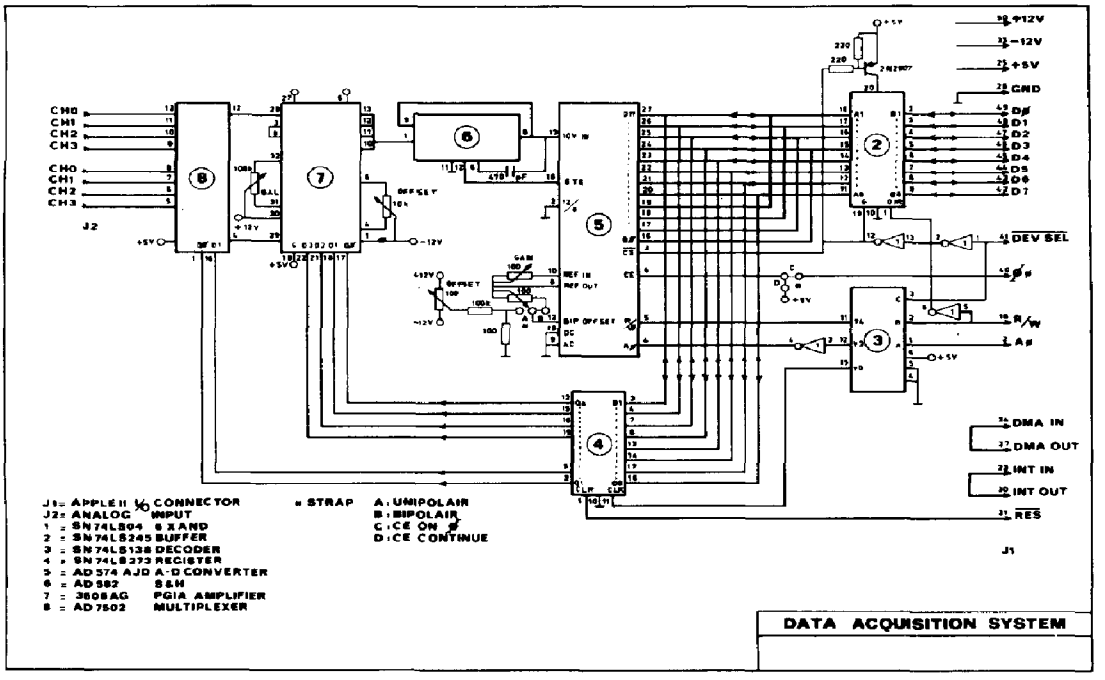

Fig. 1. Outline of the 12 bits Analog to Digital converter with multiplexer and programmable amplifier, interfaced to Apple][microcomputer.

least significant bytes in the high nibble and the gain number in the low nibble. This output format gives all the necessary information in just two bytes. The three subroutines ADC12, MAXGN and HEXAM have different functions. ADC12 simple does one measurement, while MAXGN does also only one measurement but determines as well the maximum gain number which gives a proper result. It puts this maximum gain number in $06. This subroutine is extremely useful when more than one channel is measured. The subroutine HEXAM jumps first to MAXGN and then performs 16 measurements in a row and calculates the average of them. All 16 measurements are performed with the same gain

Table 1. Specification of the 12 bits ADC card in the bipolar mode. The gainfactor is input for the card, the gain number is input for the software. The subroutines determine the factor from the number

GAIN NUMBER	GAINFACTOR dec	GAINFACTOR hex	GAIN	MAXIMUM	RESOLUTIE
0	0	0	1	5.000 V	1.22 mV
1	1	1	2	2.5	.61
2	2	2	4	1.25	.305
3	5	5	8	625 mV	152 µV
4	6	6	16	313	76.3
5	8	8	32	156	38.1
6	9	9	64	78	19.0
7	10	A	128	39	9.54
8	12	C	256	20	4.77
9	13	D	512	10	2.38
10	14	E	1024	5	1.19

Table 2. Time cycle as a function of time code

Time code dec	Time code hex	msec	Time code dec	Time code hex	sec	Time code dec	Time code hex	min
0	0	.25	9	9	.2	18	12	2
1	1	.5	10	A	.5	19	13	5
2	2	1	11	B	1	20	14	10
3	3	2	12	C	2	21	15	15
4	4	5	13	D	5	22	16	30
5	5	10	14	E	10	23	17	60
6	6	20	15	F	20	24	18	120
7	7	50	16	10	50			
8	8	100	17	11	60			

number. HEXAM needs approximately 0.5 ms to run and uses memory locations $FA through $FD on the zero-page.

Part 3 is a more variable part of the software as it contains the specific needs of the user. Part 3A shows the required software for a specific number of measurements (maximum 32,000) with regular time intervals. The number must be specified in $1A and $1B (resp. low and high byte, 26 and 27 decimal). Subroutine STORE stores the contents of $08 and $09 in a row. The start of this row is given in $1D and $1E (resp. low and high byte, 29 and 30 decimal). This will normally be $6000 for the Apple][.

DECAY MEASUREMENT

For electrochemical decay measurements it is necessary to do a large number of measurements immediately after the current interruption. Later on the number must be severely reduced to avoid memory overflow. We did 16 measurements with each time cycle from Table 2. This delivers us time intervals of 1/4 ms in the beginning till 2 s after a minute. Part 3B gives the used software for a measurement that takes a minute. Subroutine START is a userdefinable subroutine that starts the decay. In our cases a mercury wetted relay was switched through a PIA. The lines around label WAIST are used to spend time when the time code has not been changed. The Applesoft-basic counterpart of this measurement is given in listing 2. It fills the arrays TIME and ADC respectively with the time of the measurement and the measured voltage at that time. These arrays can be used for further analysis of the data.

Acknowledgement—We wish to thank Mr. Louwerse of the electronic department of our laboratory for practical help during the development of the ADC card.

Listing 1

Assembly language part of the program to control the timer card and the 12 *bits* ADC *card*

```
TIMER0  EQU $C090
TIMER1  EQU $C091
TIMER2  EQU $C092
TIMER3  EQU $C093
TIMER4  EQU $C094
TIMER5  EQU $C095
ADC0    EQU $C0A0
ADC1    EQU $C0A1
MWAIT   EQU $FCA8    APPLE MONITOR WAIT SUBROUTINE
MBELL   EQU $FBDD    APPLE MONITOR BELL SUBROUTINE
START   EQU $0000    USER DEFINABLE

************************
*  PART 1 - TIMER      *
************************
ITIME   LDA #$10     INITIALIZE TIMER CARD
        STA TIMER2
        LDA #$11
        STA TIMER0
        STA TIMER1
        LDA #$93
        STA TIMER0
        RTS
STIME   LDY $1C      START TIME CYCLE
        LDA DATA1,Y
        STA TIMER2
        LDA DATA2,Y
        STA TIMER3
        LDA #$00
        STA TIMER4
        STA $1F
        LDA DATA3,Y
        STA TIMER5
        LDA #$92
        STA TIMER0   START TIMERS
        RTS
DATA1   DFB $0,$0,$0,$0,$0,$0,$1
        DFB $3,$C,$7,$31,$31,$31,$63
DATA2   DFB $48,$9D,$C4,$3E,$8C,$EC,$71
        DFB $70,$19,$49,$EE,$EE,$EE,$DE
DATA3   DFB $2,$2,$4,$27,$23,$2A,$44
        DFB $39,$20,$88,$27,$4F,$C7,$C7
WTIME   LDA TIMER1   WAIT FOR END OF TIME CYCLE
        LSR A
        LSR A
        BCC WAITT
        LDX #$10     TIME CYCLE ALREADY FINISHED
```

```
RMBELL JSR MBELL    THEN BELL 10 TIMES
       DEX
       BNE RMBELL
       RTS
WAITT  LDA TIMER1
       LSR A
       LSR A
       BCC WAITT
       LDA #$93
       STA TIMER0   RESET TIMERS
       RTS
************************
*  PART 2 - ADC        *
************************
ADC12  LDY $06
       LDA DATA,Y
       ASL A
       ASL A
       ASL A
       ASL A
       CLC
       ADC $07
       STA ADC0
       LDA #$01
       JSR MWAIT    WAIT 30 MUSEC
CONV   STA ADC1     START CONVERSION
       LDA #$01
       JSR MWAIT
       CLC
       PHA
       PLA          WAIT 35 MUSEC
       LDA ADC0
       STA $08
       LDA ADC1
       AND #$F0
       ADC $06
       STA $09
       RTS
DATA   DFB $00,$01,$02,$05,$06,$08
       DFB $09,$0A,$0C,$0D,$0E
MAXGN  JSR ADC12
       LDA $08
       BEQ DECRE
       CLC
       ADC #$01
       BEQ DECRE
       SEC
       SBC #$42
       SEC
       SBC #$7E
       BCC INCRE
RTS    RTS
DECRE  LDA $06
       BEQ RTS
       DEC $06
       JMP MAXGN
INCRE  LDA $06
       CMP #$0A
       BEQ RTS
       INC $06
       JMP MAXGN
HEXAM  JSR MAXGN
       LDA #$00
       STA $FC
       STA $FD
       LDX #$10
RUN    LDA ADC0
       STA $FA
       LDA ADC1
       STA $FB
       STA ADC1     START NEW CONVERSION
       LDY #$04
************************
*  PART 3A             *
************************
       LDA #$00
       STA $1D
       LDA #$60
```

```
ROLL   CLC
       ROR $FA
       ROR $FB
       DEY
       BNE ROLL
       CLC
       LDA $FD
       ADC $FB
       STA $FD
       LDA $FC
       ADC $FA
       STA $FC
       DEX
       BNE RUN
       LDA $FD
       AND #$F0
       CLC
       ADC $06
       STA $09
       LDA $FC
       STA $08
       RTS
```

```
************************
*  PART 3B - DECAY     *
************************
DECAY  LDA #$10
       STA $1B
       LDA #$00
```

```
        STA $1E
        JSR ITIME
NEXT    JSR STIME
        DEC $1A
        BNE NOEND
        DEC $1B
        BMI END
NOEND   JSR MAXGN
        JSR STORE
        JSR WTIME
        JMP NEXT
END     RTS
STORE   LDY #$00
        LDA $08
        STA ($1D),Y
        INC $1D
        LDA $09
        STA ($1D),Y
        INC $1D
        BNE LBL1
        INC $1E
LBL1    RTS
```

```
        STA $1C
        STA $1D
        LDA #$60
        STA $1E
        JSR ITIME
        JSR START
LBL2    DEC $1B
        BNE LBL3
        INC $1C
        LDA #$10
        STA $1B
        LDA #$0C
        CMP $1C
        BNE LBL4
        JMP LBL5
LBL3    LDX #$03
        LDY #$00
WAIST   DEX
        BNE WAIST
LBL4    JSR STIME
        JSR ADC12
        JSR STORE
        JSR WTIME
        JMP LBL2
LBL5    JSR STIME
        RTS
```

Listing 2

Applesoft-basic program to fill the arrays **TIME** *and* **ADC** *with the time of the measurement and the measured voltage at that time*

```
  10  DIM TC(11),TIME(200),ADC(200)
 100  DATA  0.5,1,2,5,10,20,50,100,200,500,1000,2000
 110  FOR I= 0 TO 11 :  READ TC(I) :  NEXT

 300  I= 0 : C= 0 : MEM= 24576 : TT= 0
 310    FOR K= 0 TO 11
 320    T= TC(K)
 330      FOR KK= 0 TO 15
 340      TT= TT + T : MEM= MEM + 2
 350      GOSUB 1000
 360      I= I + 1 : TIME(I)= TT : ADC(I)= UD
 380      NEXT
 390    NEXT
 400  END

1000  UA= PEEK (MEM) : UB=  PEEK (MEM + 1)
1010  UC=  INT (UB / 16)
1020  VF= UB - 16 * UC
1030  UD= 16 * UA + UC - 2048
1040  UD= UD * 5000 / 2048
1050  UD= UD / (2 ^ VF)
1060  RETURN
```