Fast Low Cost A/D Converter

Requirements:
'Build-it-yourself' A/D converter
Any microcomputer with a parallel port

Perhaps you are an old pro at bridging the gap between the analog world we live in and the digital domain inhabited by your microcomputer. Or perhaps you are like the rest of us, a little hesitant about plugging things into the back of your computer. The selection of A/D converters is confusing at best. One must consider such things as resolution, speed, accuracy, cost and ease of interfacing.

Recently a chip has been introduced onto the market which should lessen the nightmare of A/D selection for most applications. This chip, National Semiconductor's **ADC0820**, boasts 1.2 microsecond maximum conversion time, ± 1 LSB total unadjusted error, 35 milliwatts typical power dissipation, and an 8-bit parallel, bus compatible output. A couple of years ago an A/D converter exhibiting similar characteristics would cost about two hundred ($200) collars. The ADC0820 presently sells for about fifteen dollars in single unit quantities.

Construction

The beauty of the ADC0820 is exemplified in the ease with which it is interfaced. The circuit in Figure 1 has been used with both the PET and VIC-20 computers and can in fact be connected to any computer having a latched data bus. Power can be supplied externally, from pin 2 of the cassette port or, in the case of the VIC-20, from pin 2 of the User Port.

In the circuit of Figure 1, Vref+ is tied to 5 Vdc and Vref- to ground. This will provide counts of 0 and 255 for Vin equal to 0 and 5 volts respectively. If, however, your application requires a full scale voltage less than 5 Vdc or a count of 0 to be given for a Vin offset from ground, Vref+ and Vref- can be adjusted accordingly.

Circuit construction can be printed circuit, wire wrap or point-to-point wiring. If wire wrapping is used, hookup wire should be soldered from the analog input signal to Vin. Whichever method is used, this connection should be kept as short as possible. Finally, since, the ADC0820 is a CMOS device, care should be taken to avoid static electricity.

Software

Listings of the machine language loader routines are shown below for a variety of microcomputers. These represent two programs. Each program will load the machine code into the top of memory and move the top of memory pointer down appropriately. An assembly code version of each program is provided for the PET. These can be easily modified to run on other 6502-based systems and, with some work, 6809-based systems as well.

The first program, **USR**, will perform an analog to digital conversion and return the converted value via the USR command. Typical usage would be:

```
10  GOSUB 60000
20  PRINT USR(1)
30  GOTO 20
```

The second program, **BUFFER**, will sample and store into a buffer 256 consecutive data points. This second routine is very useful for capturing spontaneous, non-triggerable signals. Three variables have been reserved for used with this program. They are **NEM**, **BUF** and **THRESH**. **NEM**, standing for **N**ew **E**nd of **M**emory, marks the start of the machine language routine. **BUF** is the beginning of the buffer where conversion data is stored, and **THRESH** is the address of the threshold value which must be exceeded before any data will be stored. Typical usage for this program would be:

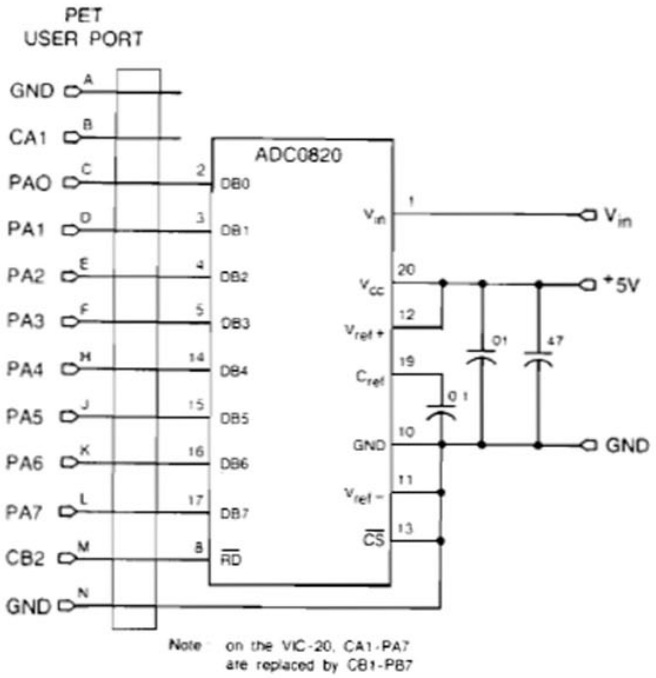
```
10  GOSUB 6000
20  POKE THRESH,50
30  SYS NEM
40  FOR I = 0 TO 256
50  PRINT PEEK(BUF + I)
60  NEXT I
70 GOTO 30
```

Keep in mind when using either of these programs that the top of memory will be moved down and new machine code deposited everytime the loader subroutine is called. Therefore, unless you call this loader only at the beginning of your program, you will soon be greeted with '**OUT OF MEMORY ERROR**'.

Conclusion

I hope I've shared some of my enthusiasm for this new A/D converter. We have been using them for about six months now and haven't run into a single problem. Most applications have been as data collectors in college laboratories, although I can easily imagine such applications as voice/music digitizers, solar controllers or even automotive efficiency monitors. Should anyone discover a unique use for the ADC0820, I hope you'll share your experiences with the rest of us.

Figure 1



Listing 1. USR Program: VIC Version

```
60000 : REM ** MACHINE LANGUAGE LOADER ROUTINE **
60010 : REM ** TO PERFORM AN A/D CONVERSION    **
60020 : REM ** VIA THE USR COMMAND FOR THE     **
60030 : REM **         VIC-20                  **
60040 : REM ** BY F. J. GENETT  8/1/83         **
60050 : REM ** MICRO, FEBRUARY 1984, #69       **
60060 NEM=PEEK(55)+256*PEEK(56)-28
60070 X=INT(NEM/256) : Y=NEM-256*X
60080 POKE 56,X : POKE 52,X : POKE 2,X
60090 POKE 55,Y : POKE 51,Y : POKE 1,Y : POKE 0,76
60100 FOR I=0 TO 27
60110  READ D:POKE NEM+I,D
60120 NEXT I
60130 RETURN
60140 DATA 169,0,141,18,145,173
60150 DATA 28,145,9,244,141,28
60160 DATA 145,41,223,141,28,145
60170 DATA 173,16,145,168,169,0
60180 DATA 32,145,211,96
```

Listing 2. Buffer Program: VIC Version

```
60000 : REM ** MACHINE LANGUAGE LOADER ROUTINE **
60010 : REM ** TO SAMPLE AND STORE 256         **
60020 : REM ** CONSECUTIVE DATA BYTES          **
60030 : REM ** WRITTEN FOR THE VIC-20          **
60040 : REM ** BY F. J. GENETT  8.1/83         **
60050 : REM ** MICRO, FEBRUARY 1984, #69       **
60060 NEM=PEEK(55);256*PEEK(56)-315
60070 BUF=NEM+59 : THRESH=NEM+29
60080 X=INT(NEM/256): Y=NEM-256*X
60090 POKE 56,X : POKE 52,X : POKE 55,Y : POKE 51,Y
60100 FOR I=0 TO 51
60110  READ D: POKE NEM+I,D
60120 NEXT I
60130 X=INT(BUF/256): POKE NEM+46,X: POKE NEM+45,BUF-256*X
60140 RETURN
60150 DATA 120,169,0,141,18,145
60160 DATA 170,173,28,145,9,224
60170 DATA 141,28,145,168,152,141
60180 DATA 28,145,41,223,141,28
60190 DATA 145,173,16,145,201,5
60200 DATA 144,240,152,141,28,145
60210 DATA 41,223,141,28,145,173
60220 DATA 16,145,157,0,48,232
60230 DATA 208,238,88,96
```

Listing 3. USR Program: PET Version

```
60000 : REM ** MACHINE LANGUAGE LOADER ROUTINE **
60010 : REM ** TO PERFORM AN A/D CONVERSION    **
60020 : REM ** VIA THE USR COMMAND FOR THE     **
60030 : REM **         PET 4.0 ROM             **
60040 : REM ** BY F. J. GENETT  8/1/83         **
60050 : REM ** MICRO, FEBRUARY 1984, #69       **
60060 NEM=PEEK(52)+256*PEEK(53)-28
60070 X=INT(NEM/256) : Y=NEM-256*X
60080 POKE 53,X : POKE 49,X : POKE 2,X
60090 POKE 52,Y : POKE 48,Y : POKE 1,Y : POKE 0,76
60100 FOR I=0 TO 27
60110  READ D:POKE NEM+I,D
60120 NEXT I
60130 RETURN
60140 DATA 169,0,141,67,232,173
60150 DATA 76,232,9,244,141,76
60160 DATA 232,41,223,141,76,232
60170 DATA 173,79,232,168,169,0
60180 DATA 32,188,196,96
```

Listing 4. Buffer Program: PET Version

```
60000 : REM ** MACHINE LANGUAGE LOADER ROUTINE **
60010 : REM ** TO SAMPLE AND STORE 256         **
60020 : REM ** CONSECUTIVE DATA BYTES          **
60030 : REM ** WRITTEN FOR THE VIC-20          **
60040 : REM ** BY F. J. GENETT  8.1/83         **
60050 : REM ** MICRO, FEBRUARY 1984, #69       **
60060 NEM=PEEK(52)+256*PEEK(53)-315
60070 BUF=NEM+59 : THRESH=NEM+29
60080 X=INT(NEM/256): Y=NEM-256*X
60090 POKE 53,X : POKE 49,X : POKE 52,Y : POKE 48,Y
60100 FOR I=0 TO 51
```

```
60110  READ D: POKE NEM+I,D
60120 NEXT I
60130 X=INT(BUF/256): POKE NEM+46,X: POKE NEM+45,BUF-256*X
60140 RETURN
60150 DATA 120,169,0,141,67,232
60160 DATA 170,173,76,232,9,224
60170 DATA 141,76,232,168,152,141
60180 DATA 76,232,41,223,141,76
60190 DATA 232,173,79,232,201,5
60200 DATA 144,240,152,141,76,232
60210 DATA 41,223,141,76,232,173
60220 DATA 79,232,157,0,48,232
60230 DATA 208,238,88,96
```

Listing 5. USR Program Assembly PET Version

```
033A A9 00    INIT  LDA  #$00  ; SET DDRA FOR INPUT
033C 8D 43 E8       STA $E843
033F AD 4C E8       LDA $E84C  ; PULL CB2 HIGH
0342 09 E0          ORA #$E0
0344 8D 4C E8       STA $E84C
0347 29 DF          AND #$DF   ; PULL CB2 LOW
0349 8D 4C E8       STA $E84C
034C AD 4F E8       LDA $E84F  ; READ CONVERSION BYTE
034F A8             TAY        ; LOAD CONVERSION BYTE
0350 A9 00          LDA #$00   ; INTO FLOATING
0353 20 BC C4       JSR $C4BC  ; POINT ACCUMULATOR
0355 60             RTS        ; RETURN FROM SUBROUTINE
```

Listing 6. BUFFER Program Assembly PET Version

```
033A 78       INIT  SEI        ; DISABLE INTERRUPTS
033B A9 00          LDA #$00   ; SET DDRA FOR INPUT
033D 8D 43 E8       STA $E843
0340 AA             TAX
0341 AD 4C E8       LDA $E84C  ; PULL CB2 HIGH
0344 09 E0          ORA #$E0
0346 8D 4C E8       STA $E84C
0349 A8             TAY        ; STORE PCR IN Y REG.
034A 98       THRSH TYA        ; RECALL PCR
034B 8D 4C E8       STA $E84C  ; PULL CB2 LOW
034E 29 DF          AND #$DF
0350 8D 4C E8       STA $E84C
0353 AD 4F E8       LDA $E84F  ; READ CONVERSION BYTE
0356 C9 05          CMP #$05   ; WAIT UNTIL INPUT
0358 90 F0          BCC THRSH  ; EXCEEDS THRESHOLD
035A 98       BUFF  TYA        ; RECALL PCR
035B 8D 4C E8       STA $E84C  ; PULL CB2 LOW
035E 29 DF          AND #$DF
0360 8D 4C E8       STA $E84C
0363 AD 4F E8       LDA $E84F  ; READ CONVERSION BYTE
0366 9D 00 30       STA $3000,X  ; FILL BUFFER
0369 E8             INX        ; BUMP POINTER AND
036A D0 EE          BNE BUFF   ; EXIT IF BUFFER FULL
036C 58             CLI        ; ENABLE INTERRUPTS
036D 60             RTS        ; RETURN FROM SUBROUTINE
```