High-Grade

DATA CONVERTER FOR THE APPLE II

By Robert C. Nicklin

Part 2: Checkout, calibration, and use

LAST month, we discussed the principles of A/D and D/A conversion and described a converter for the Apple II. Here are the software and instructions needed to operate the converter.

Checkout & Calibration. Plug one end of a ribbon cable with DIP headers on each end into Port A at *SO4* on the interface adapter. Then, making sure the Apple computer is powered down, install the VIA in slot 5 of the computer. Plug the other end of the cable into a solderless-socket breadboard and use wire jumpers to connect D0 through D7 to ground at pin 11. Switch on the computer and wait for the prompt to appear on-screen. Measure the voltages at pins 50, 33, and 25 of the computer's bus connector; they should read +12, −12, and +5 V, respectively.

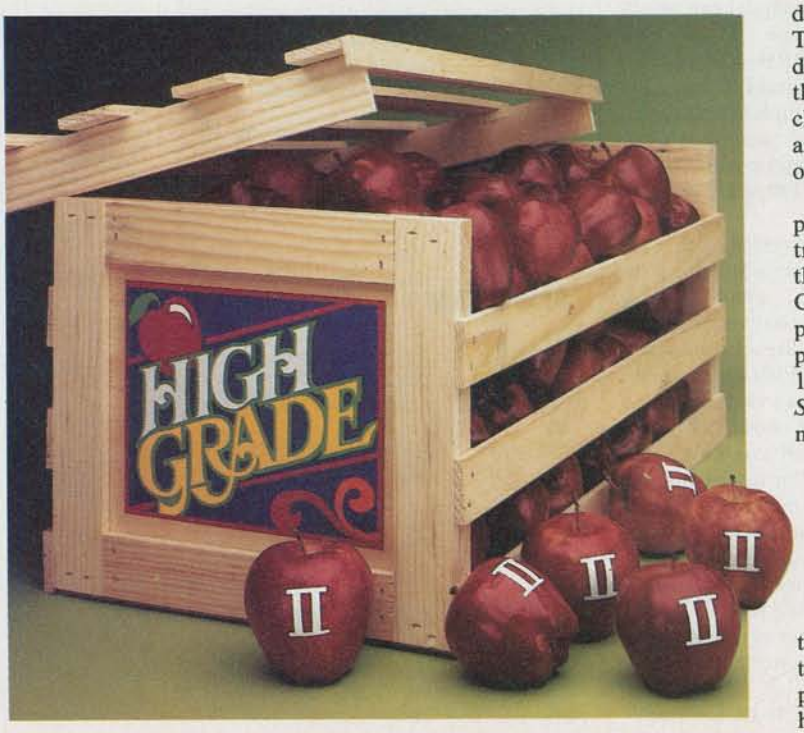
For the following discussion, the interface is assumed to be plugged into slot 5 of the computer and all data lines are grounded.

Enter and run Program 1 to test the input to Port A. Displayed on-screen should be a running column of 0s read in from Port A. One at a time, remove the data line jumpers from ground and note the change in status displayed on-screen. As each data line is ungrounded, it should display a 1 on the screen. If it doesn't, the interface adapter most likely isn't plugged into slot 5 of the computer and/or the program hasn't been correctly entered. If the problem still persists after making sure that the adapter is in slot 5 and the program has been properly entered, use a dual-trace oscilloscope to check the *IC7* delay circuit. For proper operation, the signal observed at pin 6 of *IC7* should be delayed by about 180 ns with respect to the signal at pin 1.

Remove the ground connections from D0 through D7 and load and run Program 2 to test the output. For Byte=0, there should be no voltage (0 volt) between each data line and ground; for Byte=255, each data line should be at +5 volts; for Byte=55, D0 through D7 should be 10101010, etc.

Load and run Program 3 to test control line CA2. Making measurements at *J6*, CA2 should be 0 V for PC=12 and +5 V for PC=14.

To test Port B, power down the computer and move the DIP cable to *SO3*. Repeat the above three tests, changing the POKE and PEEK addresses for DDRB and ORB as detailed in Table I. The PCR address doesn't change, but the values for PC change to 192 for 0 V and 224 for +5 V to operate CB2.



Having ascertained proper port and control operation, remove the ICs from the Data Converter assembly, power down the computer, and plug the 16-pin DIP cable into *SO3*, the Port B connector. Power up the computer and confirm that +12, −12, and +5 V are present at the appropriate IC socket pins. (Bear in mind that under these no-load conditions some non-power pins will measure high.) Having con-

firmed that the proper voltages are present at the appropriate pins, power down the computer, insert the ICs in their respective sockets (take care to observe proper orientation), and power up once again.

Connect a voltmeter to *J2* and ground the center connector of *J1* to begin testing the analog amplifier. Set the project's GAIN switch to 1 and adjust trimmer *R3* for a 0-V output. Set the GAIN switch to 500 and adjust *R3* for minimum positive output. To check calibration of the GAIN control, remove the short from across *J1* and replace it with a 10,000-ohm potentiometer connected across a D cell (use the wiper and one end lug of the pot to make connections to the converter circuit) to provide the dc signal input. Make up a table of input vs output for each of the six positions of the GAIN switch for use in making software corrections to compensate for nonlinearities in gain.

If you have access to an audio oscillator and an oscilloscope, feed sine waves into *J1* and monitor *J2* to observe that the amplifier has reasonably flat response from dc to 15 kHz.

Enter and run Program 4 to check D/A conversion. A voltmeter connected between DAC *J5* and ground should indicate 0 V when DB is 0 and 2.55 V when DB is 255.

Use Program 5 to generate a square wave whose pitch is controlled by the value of P for checking the audio amplifier. This square-wave signal can be observed at *J5* with a scope.

Program 6 sets DDRB=0 for all inputs for testing the digital I/O lines. D1 through D6 will float high while D0 and D7 are checked. The running display produced by this program will change according to whether D0 or D7 is shorted (see Table II).

A/D conversion is tested with Program 7. This simple diagnostic program is written in BASIC, which is slow enough to complete a conversion before being read. Results are continuously displayed on-screen. With *J1* grounded and the GAIN switch set to 1, the reading should be 0 on dc and about 127 on ac. Connect the pot/D-cell assembly to *J1* and slowly vary the setting of the pot (dc input level). The on-screen display should follow the changes. Make sure the −9-volt reference is present at pin 2 of *IC2*. A scope connected to TRIG *J6* should reveal CB2 going high for about one-third or one-quarter of the pulse cycle. Without this pulse, no A/D conversion is possible.

Set AC/DC switch *S2* to DC for ampli-

PROGRAM 1

```
100 REM     PROGRAM #1
120 REM     PORT A INPUT TEST
130 REM     6522 VIA IN SLOT 5
150 :
160 REM     SET PORT A DDRA
            FOR INPUT
170 D=0
180 POKE 49363, D
190 :
200 REM     READ ORA WHILE
            CHANGING
210 REM     JUMPERS FROM D0-D7
            TO GROUND
220 PRINT PEEK (49375)
230 :
240 FOR PAUSE = 1 TO 200: NEXT
250 GOTO 220
```

PROGRAM 2

```
100 REM     PROGRAM #2
120 REM     PORT A OUTPUT TEST
130 REM     6522 VIA IN SLOT 5
150 :
160 REM     SET PORT A DDRA
            FOR OUTPUT
170 D=255
180 POKE    49363,D
190 :
200 INPUT   "BYTE"; B
210 REM     OUTPUT BYTE TO
            PORT A
220 REM     TEST DO-D7 WITH
            VOLTMETER
230 POKE    49375, B
240 GOTO    200
```

PROGRAM 3

```
100 REM     PROGRAM #3
120 REM     PORT A CONTROL
125 REM     LINE CA2 TEST
130 REM     6522 VIA IN SLOT 5
150 :
160 REM     CA2 GOES LOW ON 12
170 REM     AND HIGH ON 14
180 :
190 INPUT   "PCR BYTE=?"; PB
200 POKE    49372, PB
210 GOTO    190
```

PROGRAM 4

```
100 REM     PROGRAM #4
120 REM     DAC  TEST
130 REM     6522 VIA IN SLOT 5
150 :
160 REM     SET DDRB FOR OUT
170 POKE    49362,255
180 :
190 REM     SET CB2 HIGH SO DAC
200 REM     CHIP ENABLE IS LOW
210 POKE    49372,224
220 :
230 REM     OUTPUT DB TO DAC
            AR ORB
240 INPUT   "DAC BYTE=?";DB
250 POKE    49360,DB
260 GOTO    240
```

PROGRAM 5

```
100 REM     PROGRAM #5
120 REM     AUDIO AMP TEST
130 REM     6522 VIA IN SLOT 5
150 :
160 REM     SET DDRB FOR OUTPUT
170 POKE    49362,255
180 :
190 REM     SET CB2 HIGH SO DAC
200 REM     CHIP ENABLE IS LOW
210 POKE    49372,224
220 :
230 INPUT   "PAUSE  BYTE=?";P
240 POKE    49360,0: GOSUB 290
250 POKE    49360,255: GOSUB 290
260 GOTO    240
270 END
290 FOR PAUSE = 0 TO P:
            NEXT: RETURN
```

PROGRAM 6

```
100 REM     PROGRAM #6
120 REM     DIGITAL I/O TEST
130 REM     6522 VIA IN SLOT 5
150 :
160 REM     SET DDRB FOR INPUT
170 POKE    49362,0
180 :
190 REM     GROUND D0 OR D7
            (4 WAYS)
200 PRINT   PEEK (49360)
210 GOTO    200
```

PROGRAM 7

```
100 REM     PROGRAM #7
120 REM     ADC  TEST
130 REM     6522 VIA IN SLOT 5
150 :
160 REM     SET DDRB FOR INPUT
170 POKE    49362,0
180 :
190 REM     MAKE READ HIGH
            (WITH CB2)
200 REM     AND START
            CONVERSION
210 POKE    49372,224
220 :
230 REM     MAKE READ LOW
            (WITH CB2)
240 REM     PREPARE TO READ
            CONVERSION
250 POKE    49372,192
260 :
270 REM     READ ADC
280 PRINT   PEEK (49360)
290 :
300 GOTO    210
```

tude calibration. Apply the full potential from the D cell to *J1*. Assume this to be 1.46 V, the voltmeter will indicate 7.01 V with the GAIN switch set to 5. This means that gain is actually 4.80 (7.01/1.46), although the A/D converter actually "sees" 7.01 V. Measure reference voltage V_{REF} between *J3* and ground. Assume it to be 8.56 volts. With full scale for the converter being 255 for any V_{REF}, the A/D converter will produce 209 (7.01/8.56). Working backward, we find that V_{INPUT} = (A/D output/gain) × (V_{REF}/255). This means that conversion accuracy depends on voltmeter accuracy. Often, true V_{INPUT} isn't needed, since relative amplitude will be sufficient.

Frequency accuracy requires use of a frequency counter to set a square-wave generator to 1000 Hz (the actual frequency isn't important as long as it is known). Feed the square waves into *J1* and use Table III to run the A/D converter fast enough to digitize the waveform. The Apple monitor program can be used to "look at" the bytes in memory. Another way to do this is to use a short BASIC program written to PEEK the bytes onto the screen. In either case, count the number of bytes per complete cycle of the square wave and multiply by the frequency to obtain A/D conversion sampling rate. If the A/D timing bytes are changed, this frequency calibration can be performed and recorded for a range of digitizing rates.

Operation Under Assembly Language. For A/D or D/A at rates of less than 50 samples per second, such BASIC programs as those in Programs 1 through 7 are adequate. To realize rates up to 17,000 samples per second for A/D and output rates to 33,000 bytes per second with D/A, both converters must be operated with assembly-language programs. Table III is for A/D, while Table IV is for D/A code.

Operation of Table III code is similar to operation of Program 7. Port A DDRA is set for input with the A/D converter getting a high on its Read line via CA2 to initiate conversion. A loop of at least 20 μs allows the A/D section to convert the input signal. When the Read line goes low, via CA2, conversion

TABLE I—6522 VIA ADDRESSES, SLOT 5

Starting address	49360 ($C0D0)
DDRA	49363
ORA	49375
DDRB	49362
ORB	49360
PCR	49372
ACR	49371

TABLE II—DIGITAL I/O TEST

D7	D6	D5	D4	D3	D2	D1	D0	Screen
0	1	1	1	1	1	1	0	126
0	1	1	1	1	1	1	1	127
1	1	1	1	1	1	1	0	254
1	1	1	1	1	1	1	1	255

TABLE III—ADC ASSEMBLY LANGUAGE

```
$9300   A9 00     START   LDA   $00
  02    A8              TAY           INITIALIZE Y
  03    85 FE           STA   MEM     SET START OF DATA STORAGE—
  05    A9 88           LDA   $88     ALTER FROM BASIC
  07    85 FF           STA   MEM+
  09    A9 00           LDA   $00     MAKE PORT A ALL INPUTS
  0B    8D D3 C0        STA   DDRA
  0E    78 EA EA        SEI
  11    A9 0E     ADC   LDA   $0E     MAKE READ HIGH, START CONVERSION
  13    8D DC CO        STA   PCR
  16    A2 05           LDX   $05     GIVE ADC > 20 MICROSECONDS
  18    CA        WAIT1 DEX             TO CONVERT
  19    D0 FD           BNE   WAIT1
  1B    A9 0C           LDA   $0C     MAKE READ LOW, PREPARE TO
  1D    8D DC C0        STA   PCR           READ ADC
  20    AD DF C0        LDA   ORA     READ ADC
  23    91 FE           STA   (MEM), Y STORE DATA
  25    20 40 93        JSR   MSEC    DELAY BETWEEN SAMPLES
  28    C8              INY
  29    D0 E6           BNE   ADC
  2B    E6 FF           INC   MEM+    PREPARE TO FILL NEXT MEMORY PAGE
  2D    A6 FF           LDX   MEM+    WITH DATA
  2F    E0 92           CPX   $92     CHECK FOR END OF MEMORY
  31    D0 DD           BNE   ADC
  33    58              CLI
$9334   60              RTS

$9340   A9 00     MSEC  LDA   $00     SET VIA TIMER 2 FOR ONE-SHOT
  42    8D DB C0        STA   ACR           TIME INTERVAL
  45    A9 E8           LDA   $E8     $03E8 GIVES ABOUT 1 msec
  47    8D D8 C0        STA   T2L           INTERVAL—POKE CHANGES
  4A    A9 03           LDA   $03           FROM BASIC
  4C    8D D9 C0        STA   T2H
  4F    A9 20           LDA   $20     TIMER 2 INTERRUPT FLAG MASK
  51    2C DD C0  WAIT2 BIT   IFR     IS FLAG SET?
  54    F0 FB           BEQ   WAIT2   IF NOT, THEN LOOP
  56    AD D8 C0        LDA   T2L     TIMED OUT—CLEAR FLAG
$9359   60              RTS
```

TABLE IV—DAC ASSEMBLY LANGUAGE

```
$9200   A9 FF          LDA   $FF     SET UP PORT A FOR OUTPUT
  02    8D D3 CO       STA   DDRA       TO DAC
  05    A5 FE          LDA   FE      SAVE CONTENTS OF TWO ZERO PAGE
  07    48             PHA             ADDRESSES USED FOR MEM AND MEM+
  08    A5 FF          LDA   FF           IN (IND), Y ADDRESSING
  0A    48             PHA
  0B    EA             NOP
  0C    A9 00    START LDA   $00     INITIALIZE Y FOR (IND), Y MEMORY SCAN
  0E    A8             TAY
  0F    85 FE          STA   MEM     START DATA READOUT AT $8800
  11    A9 88          LDA   $88
  13    85 FF          STA   MEM+
  15    EA             NOP
  16    78             SEI           MAKE DAC OUTPUT GLITCHLESS
  17    A9 0C          LDA   $0C     MAKE CA2 LOW TO TURN OFF DAC
  19    8D DC C0       STA   PCR
  1C    A9 0E          LDA   $0E     MAKE CA2 HIGH TO TRIGGER SCOPE
  1E    8D DC C0       STA   PCR        AND ENABLE DAC
  21    EA             NOP
```

NOTE: CA2 GETS INVERTED BEFORE IT GETS TO DAC, BY 7416 HEX INVERTER

```
  22    B1 FE    DAC   LDY   (MEM),Y  DISPLAY MEMORY CONTENTS
  24    8D DF C0       STA   ORA          ON SCOPE
  27    20 50 92       JSR   DELAY
  2A    C8             INY           DAC GETS ONE PAGE AT A TIME
  2B    D0 F5          BNE   DAC
  2D    E6 FF          INC   MEM+    MOVE TO NEXT PAGE
  2F    A6 FF          LDX   MEM+       TO CHECK FOR TOP LIMIT
  31    E0 92          CPX   $92
  33    D0 ED          BNE   DAC
  35    EA             NOP
  36    58             CLI
  37    C6 FD          DEC   NSCAN   HAVE N SCANS BEEN COMPLETED?
  39    A5 FD          LDA   NSCAN
  3B    D0 CF          BNE   START
  3D    EA             NOP
  3E    68             PLA
  3F    85 FF          STA   FF      REPLACE ZERO PAGE CONTENTS
  41    68             PLA
  42    85 FE          STA   FE
$9244   60             RTS
$9250   A2 02    DELAY LDX   $N      DELAY BYTE
  52    CA             DEX
  53    D0 FD          BNE
  55    60             RTS
```

TABLE V—CLEAR MEMORY ASSEMBLY LANGUAGE

```
$9280   A0 00          LDY   $00     INITIALIZE Y FOR (MEM),Y
  82    A5 FE          LDA   FE      SAVE CONTENTS OF TWO ZERO PAGE
  84    48             PHA             MEMORY LOCATIONS USED FOR
  85    A5 FF          LDA   FF        (IND), Y ADDRESSING OF
  87    48             PHA             MEM AND MEM+
  88    A9 88          LDA   $88     START CLEARED AREA AT PAGE $88
  8A    85 FF          STA   MEM+
  8C    A9 00          LDA   $00
  8E    85 FE          STA   MEM
  90    A9 00          LDA   $00     BYTE TO FILL MEMORY
  92    91 FE    CLEAR STA   (MEM),Y
  94    C8             INY           FILL MEMORY
  95    D0 FB          BNE   CLEAR
  97    E6 FF          INC   MEM+
  99    A6 FF          LDX   MEM+
  9B    E0 92          CPX   $92
  9D    D0 F3          BNE   CLEAR
  9F    68             PLA           REPLACE ZERO PAGE CONTENTS
  A0    85 FF          STA   FF
  A2    68             PLA
  A3    85 FE          STA   FE
  A5    60             RTS
```

TABLE VI—APPLE ADC LOCATIONS (SLOT 5 ASSUMED)

Hex	Address Decimal	Explanation
$9300	37632	Start of ADC Assembly language program
9306	37638	Set beginning of data storage pages $88-$91 (136-145)
930C	37644	DDRA LOW $D3 (211) DDRB LOW $D2 (210)
930D	37645	DDRA high $C0 (192) DDRB HIGH $C0 (192)
9312	37650	READ HIGH (CA2 OR CB2) START CONVERSION*
931C	37660	READ LOW PREPARE TO READ CONVERSION*
9321	37665	ORA LOW $DF (223) ORB LOW $D0 (208)
9322	37666	ORA HIGH $C0 (192) ORB HIGH $C0 (192)
9330	37680	SET END OF DATA STORAGE MEMORY
9346	37702	T2L BYTE
934B	37707	T2H BYTE
9359	37721	LAST OP CODE (90 BYTES)

	low	high
CA2	$0C	$0E
	12	14
CB2	$C0	$E0
	192	224

TABLE VII—APPLE DAC LOCATIONS (SLOT 5 ASSUMED)

Hex	Address Decimal	Explanation
$9200	37376	BEGINNING
9203	37379	DDRA LOW BYTE (211) DDRB LOW BYTE (210)
9204	37380	DDRA HIGH BYTE (192) DDRB HIGH BYTE (192)
9212	37394	SET BEGINNING OF SCAN-PAGE 136 TO 145
9218	37400	SET CA2 LOW (12) SET CB2 LOW (192)
921D	37405	SET CA2 HIGH (14) SET CB2 HIGH (224)
9225	37413	ORA LOW BYTE (223) ORB LOW BYTE (208)
9226	37414	ORA HIGH BYTE (192) ORB HIGH BYTE (192)
9232	37426	SET END OF SCAN (SCAN UP TO THIS PAGE)
9251	37457	SCAN DELAY BYTE (0 TO 225)
9257	37463	LAST OP CODE
00FD	253	NUMBER OF SCANS BEFORE RETURN KEY IS CHECKED

TABLE VIII—CLEAR MEMORY LOCATIONS (SLOT 5 ASSUMED)

Hex	Address Decimal	Explanation
$9280	37504	BEGINNING OF PROGRAM
9289	37513	SET HIGH BYTE, START OF CLEARED AREA
928D	37517	SET LOW BYTE, START OF CLEARED AREA
9291	37521	BYTE TO FILL MEMORY WITH
929C	37532	PAGE NUMBER FOR END OF CLEARED AREA —DON'T CLEAR BEYOND THIS (PAGE 146 MAXIMUM) —PAGE 146 ($92) MEANS $91FF IS LAST LOCATION THAT WILL BE CLEARED
92A5	37541	LAST BYTE OF PROGRAM

NOTE: $8800-$91FF ARE SET ASIDE FOR ADC, DAC. DATA STORAGE $9200-$93FF ARE SET ASIDE FOR DAC, CLEAR MEMORY AND ADC ASSEMBLY LANGUAGE PROGRAMS

is latched so it can be read out from ORA. The sample is stored and a delay subroutine is called. The program determines if alotted memory is full; if it isn't, it sends a Read high for another conversion.

Operation of the D/A converter under the listing given in Table IV proceeds as in the BASIC listing in Program 7. Port A DDRA is set for output. A high on CA2 triggers a scope connected to *J6* and, after inversion by *IC4*, pulls low the D/A converter's Chip Enable line. A byte is then fetched from memory and sent to the D/A converter via ORA. The program then checks whether or not the memory specified has been scanned; if it hasn't, another byte is sent to ORA. After completing a set number of scans, the program returns to the routine that called it.

Table IV can be used to clear a specific block of memory before filling it with data from the A/D converter, or before storing the output from the D/A converter. Although memory clearing can be performed from BASIC, it's too slow for more than a 1K-byte block of memory used for repeated measurements.

Operation Under BASIC. The various assembly-language programs presented in this article are best used as subroutines called from a BASIC driver program. It's easy to set memory limits and data rates and change to the alternate port by POKEing new values into the assembly-language code. This permits use of the fast assembly-language routines without having to be familiar with assembly-language programming. The BASIC listings for Apple DAC, Apple ADC, and Clear Memory contain the three assembly-language programs discussed. Tables VI, VII, and VIII list the POKE values required to tailor the routines to your purposes.

These programs all assume you're using an Apple II Plus computer with 48K of user RAM and slot 5 in the computer. The memory map in Table IX reveals where everything is stored. If you're using a disk drive, you can keep the DOS from overwriting the machine-language program by booting the DOS, setting HIMEM:34815 and then loading the BASIC program. This protects operation by putting data storage and machine language above BASIC and below the DOS. Memory locations 34816 through 37375 are for data storage; 37376 through 37721 for the D/A, Clear, and A/D; and 37722 through 38399 for data or expansion.

The three assembly-language routines can be easily relocated, since

except Apple ADC, they contain no absolute addresses. Because delay routine MSEC calls $9326 (memory location 37670) and $9327 (location 37671), Table III must be changed to fit the new location. If slot 5 in the computer isn't used, the correct VIA address for the slot must be POKEd.

Capturing Waveforms. The Waveform Program combines D/A, A/D, and Clear routines and illustrates use of the VIA and Data Converter. The purpose here is to capture an audio waveform and display it on the CRT screen of an oscilloscope.

Load the Waveform Program, connect a microphone to *J1* and a scope to *J2*, and set *S2* to AC. Sing a steady "o-o-o-h-h-h" tone into the mic and set the GAIN switch for a peak-to-peak signal level of less than 9 volts. Connect a lead from TRIG *J6* to the scope's trigger input and adjust the scope for a triggering. RUN the program while holding a note.

The computer will capture and display 1280 samples of the waveform. Since these samples are stored in memory, they can be SAVEd on disk, written to a printer, or plotted on a strip chart, or the waveform could be Fourier analyzed if desired.

With some minor changes in the BASIC program, 256 samples of each of 10 different waveforms could be captured and selectively analyzed. Alternatively, they could be plotted in HIRES with about a 25% loss in vertical resolution, on 192 vertical points for 256 from the A/D converter. ◇

TABLE IX—APPLE DATA CONVERTER MEMORY MAP

HEX		DECIMAL
$FFFF	APPLE USES	65535
$C100		49408
$C0FF	PERIPHERAL CARD I/O SPACE FOR	49407
$C080	6522 VIA (8 SLOTS)	49280
	APPLE USES	
$BFFF	DOS	49151
$9600		38400
$95FF	SPACE FOR MACHINE LANGUAGE	38399
$935A	PROGRAMS (MLP) OR CONVERTER DATA	37722
$9359	ADC MLP	37721
$9300		37632
	80 BYTES FOR MLP	
$92A5	CLEAR MEMORY MLP	37541
$9280		37504
	40 BYTES FOR MLP	
$9257	DAC MLP	37463
$9200		37376
$91FF	ADC/DAC DATA STORAGE	37375
$8800	(PAGES 136-145)	34816
$87FF	FOR BASIC PROGRAMS	34815
$6000	(SET HIMEM:34815)	24576
$5FFF	HIGH RESOLUTION GRAPHICS	24575
$2000		8192
$1FFF	FOR BASIC PROGRAMS	8191
$0C00		3072
$0BFF	APPLE USES	3071
$0000		0