Appendix F

Variations for Revision Level 2 ROMs

This appendix describes the differences between the Revision Level 3 ROMs, as presented in the chapter material, and the Revision Level 2 ROMs.

Chapter 1: STARTUP

Asterisks (*) appear in place of the pound signs (#) in the initial display line of the Revision Level 2 ROMs:

```
*** COMMODORE BASIC ***
```

You can use this as an indicator of which ROMs your CBM computer has.

Chapter 4: ARRAYS

On the Revision Level 2 ROMs, the total number of array elements in any one array is limited to 256. For example, for a one-dimensional array, elements may go from 0 to 255. For a two-dimensional array with dimension 2 in the second subscript, elements may go from (0,0), (1,0) to (127,0) or (0,1), (1,1) to (127,1), etc.

An example of programming within this restriction is given below under "Chapter 5: Generating Random Numbers."

Chapter 5: DEVELOPING A PROGRAM, Interactive Programming

In the Revision Level 2 ROMs, the system location that enables the cursor to blink is location 548. To enable the cursor, you would use the statement:

```
80 POKE 548,0     Enable cursor (Revision Level 2 ROMs)
```

instead of:

```
80 POKE 167,0     Enable cursor (Revision Level 3 ROMs)
```

Chapter 5: RND

RND(0) is non-functional. An argument of zero returns a value that is constant, or nearly constant, and that may vary from CBM to CBM computer.

You will have to use −TI to generate random seeds. This is the method used in all of the examples in Chapter 5 under "Generating Random Numbers."

Chapter 5: GENERATING RANDOM NUMBERS

Do not try to use RND(−RND(0)) to generate random seeds; it will not work. Instead, use −TI as shown in all of the examples.

The RANDOM VERSION 2 sample program in Chapter 5 will not work on the Revision Level 2 ROMs because of the 256-element array limitation. A second version of the program is shown below. It shows the lengths you have to go to in order to program with the 256-element array limitation. In this program the 1000-element table is divided into four quarters of 250 elements each.

```
5 REM RANDOM VERSION 2A
10 REM *******  B L A N K E T  *******
20 REM RANDOM DISPLAY OF ONE
30 REM   CHARACTER ENTERED FROM THE
40 REM   KEYBOARD
50 REM ********************************
70 DIM T1(249),T2(249),T3(249),T4(249)
75 T=4    :REM NUMBER OF TABLES
76 N=250  :REM NO OF ELEMENTS
80 GOSUB 200     :REM INITIALIZE TABLES
90 PRINT"HIT A KEY OR <R> TO END";
95 N1=N:N2=N:N3=N:N4=N
100 GET C$:IF C$="" GOTO 100
105 IF C$=CHR$(13) GOTO 170
110 PRINT"□";          :REM CLEAR SCREEN
120 X=RND(-TI)         :REM START NEW SEED
125 C=(ASC(C$)AND128)/2 OR (ASC(C$)AND63)
126 FOR L=1TO1000  :REM 1 FOR EACH SPOT
127 T%=T*RND(1)+1  :REM PICK A TABLE
128 ON T% GOSUB 300,400,500,600  :REM GO PICK AN ELEMENT
130 POKE A,C           :REM DISPLAY CHAR
140 NEXT L
160 GOTO 95
170 END
199 REM **SUBR TO INITIALIZE TABLES**
200 FOR I=0 TO N-1:T1(I)=I:NEXT
210 FOR I=0 TO N-1:T2(I)=I+250:NEXT
220 FOR I=0 TO N-1:T3(I)=I+500:NEXT
230 FOR I=0 TO N-1:T4(I)=I+750:NEXT
240 RETURN
299 REM **SUBROUTINE FOR TABLE T1**
```

```
300 N1=N1-1
305 REM IF EMPTY, GO TO ANOTHER TABLE
310 IF N1<0 THEN ON INT(3*RND(1)+1) GOTO 400,500,600
320 A%=(N1+1)*RND(1) :REM PICK AN ELEM
330 A=T1(A%)+32768 :REM FORM POKE ADDR
340 TP=T1(A%):T1(A%)=T1(N1):T1(N1)=TP :REM SWAP ELEMENTS
350 RETURN
399 REM **SUBROUTINE FOR TABLE T2**
400 N2=N2-1
410 IF N2<0 THEN ON INT(3*RND(1)+1) GOTO 300,500,600
420 A%=(N2+1)*RND(1)
430 A=T2(A%)+32768
440 TP=T2(A%):T2(A%)=T2(N2):T2(N2)=TP
450 RETURN
499 REM **SUBROUTINE FOR TABLE T3**
500 N3=N3-1
510 IF N3<0 THEN ON INT(3*RND(1)+1) GOTO 300,400,600
520 A%=(N3+1)*RND(1)
530 A=T3(A%)+32768
540 TP=T3(A%):T3(A%)=T3(N3):T3(N3)=TP
550 RETURN
599 REM **SUBROUTINE FOR TABLE T4**
600 N4=N4-1
610 IF N4<0 THEN ON INT(3*RND(1)+1) GOTO 300,400,500
620 A%=(N4+1)*RND(1)
630 A=T4(A%)+32768
640 TP=T4(A%):T4(A%)=T4(N4):T4(N4)=TP
650 RETURN
```

Chapter 6: FILES

This section is for CBM users who are having problems reading cassette data files using the old ROMs. If your CBM has the Revision Level 2 ROMs and you intend to use data files frequently, you should seriously consider replacing the Revision Level 2 ROMs with the Revision Level 3 ROMs, as the Revision Level 3 ROMs ensure greater reliability when reading and writing data files.

If you do plan to use the Revision Level 2 ROMs, you must do a little extra programming to get around these problems. When writing data to the data tape, the Revision Level 2 ROMs neglect to initialize the pointer to the start address of the cassette tape buffer, and also fail to leave enough blank space on the tape between physical records.[1] Consequently, when the CBM attempts to read the data back from the data tape, the problems may result in lost or garbled data. Here are a few precautions you can take to overcome these obstacles.

1. Initialize the pointer of the cassette buffer start address. Because the Revision Level 2 ROMs fail to initialize the start address to the cassette tape buffer before a file is OPENed, you must be sure to do so before opening a file with a series of POKEs:

```
Cassette #1:POKE 243,122:POKE 244,2:OPEN 1,1,1
Cassette #2:POKE 243,58 :POKE 244,3:OPEN 2,2,1
```

Memory address locations 243 and 244 point to the start address of the current tape buffer. By POKEing in the above values the pointer will be initialized properly.

2. Force interrecord gaps. The Revision Level 2 ROMs do not leave enough blank space on the tape between physical records. When the CBM attempts to read back the data with an INPUT# or GET#, if the physical records are too close together the data cannot be read, resulting in read errors and lost data. To prevent this, you can force larger gaps to be written between records by calling a routine to advance the tape each time the cassette buffer is emptied.

Before forcing an interrecord gap you must detect when the cassette buffer has written out a "physical record" or "block" of data to the tape. The buffer holds 191 characters (or 191 bytes). A full buffer is a signal that a block of data was just written to the tape, since the contents of the buffer are dumped only after it has reached its capacity. By detecting a full buffer, you can infer that a block of data was just written to the tape and an interrecord gap is needed.

How to Detect a Full Buffer

When writing data out to a tape, following each PRINT# statement the length of each data item is calculated and kept in an accumulator, which is then compared to the buffer limit (191 characters). When the accumulator equals 191 the writing to the tape is stopped until an interrecord gap is written on the tape. Below is a sample program:

```
10 POKE 243,122:POKE 244,2:OPEN1,1,1
20 FOR X=1 TO 100
30 PRINT#1,X
40 A=LEN(STR$(X))+1
50 IF (QT+A)>=191 GOSUB 1000    :REM *IF BUFFER FULL CALL SUB. TO ADVANCE TAPE*
60 QT=QT+A
70 NEXT X
80 CLOSE1
90 END
```

Line 20 prints a variable. If the variable printed (in this case, X) is numeric it must be converted to string form so the LEN function may be used to determine X's length, as shown in line 40:

```
40 A=LEN(STR$(X))+1
```

One is added to the lengths of the strings to include the carriage returns that are written on the tape following each data item. Line 50 accumulates the number of characters in the previous strings, (QT), plus A, and compares the total to 191 (the buffer limit). If the number of characters written to the tape (QT + A) is greater than or equal to 191 the entire buffer is written to the tape, and it is time to force an interrecord gap by calling the subroutine at 1000. However, if QT+A is less than 191 (QT+A<191), the buffer is not yet full. Line 60 increments QT by A, and the process keeps repeating until the buffer is full, and all the data is written from the buffer to the tape, interspersed with the interrecord gaps.

Advancing the Cassette Tape

There are three necessary steps in the routine to advance the tape:

1. Turn on the cassette tape motor (POKE 59411,53).
2. Use a wait loop to the stall program while the tape is advancing.
3. Turn off the cassette tape motor (POKE 59411,61).

POKE 59411,53 pokes "53" into memory address location 59411, which controls the cassette motor. Value 53 turns on the motor to advance the tape. Once the motor is on, a wait loop lets the tape advance for a few jiffies. The wait loop will be discussed shortly. To stop the tape, a POKE 59411,61 turns off the cassette motor. The length of the wait loop may be varied or altered, but these two POKEs are absolutely necessary to turn the cassette motor on and off.

Following is a sample wait loop inserted between the two POKE statements:

```
1000 POKE 59411,53              :REM *START TAPE MOTOR*
1010 T=TI
1020 IF (TI-T)<10 GOTO 1020     :REM *WAIT 10 JIFFIES*
1030 POKE 59411,61              :REM *STOP TAPE MOTOR*
1040 QT=0
1050 RETURN
```

Lines 1010 to 1020 make up the wait loop. Line 1010 sets variable T to the current value of TI. TI is the number of jiffies since the PET was powered up or the clock was zeroed. (A jiffy is 1/60 of a second.) TI is incremented once every jiffy, or 60 times a second. By subtracting T from TI, the elapsed time is calculated. The program must wait until ten jiffies (1/6 of a second) has elapsed before the program can continue. While TI increments, until the difference between TI and T equals ten jiffies the program is stalled, letting the cassette tape advance. This blank space on the tape is the interrecord gap. Once (TI−T) equals ten, the next statement turns off the cassette motor with a POKE 59411,61.

The routine calculates the space between each record. The tape is advanced exactly the same amount between each physical record because the time between POKEing on and off the cassette motor will always be ten jiffies. The length of the wait loop may be adjusted by changing the constant of the condition expression:

```
TI-T<X
```

The larger the value of X, the larger the interrecord gap will be. If you're unsure how long the interrecord gap should be, keep the wait loop between 5 and 30 jiffies. It is always better to have the interrecord gap too long than too short.

There is one potential problem with this routine, though it is doubtful you will ever encounter the problem. If the CBM computer has been powered up for close to twenty-four hours, or you have set the internal clock close to the twenty-fourth hour, the routine might hang up during the wait loop. At 24:00:00 the jiffy clock is reset from 5184000 jiffies to zero. If T is assigned within a few jiffies of 5184000 both TI and the jiffy clock will be reset to zero. The result is that the condition TI−T<10 will always be true (0000008−5183998 < 10) and the wait loop will hang up infinitely because TI−T will never be greater than nine. It is very improbable that this will ever happen to you, but you should use caution if the jiffy clock is nearing the twenty-fourth hour.

Here is another way to advance the tape:

```
POKE 59411,53                   :REM *START TAPE MOTOR*
POKE 514,0                      :REM *ZERO JIFFY CLOCK*
WAIT 514,16                     :REM *WAITS 16 JIFFIES*
POKE 59411,61                   :REM *STOP TAPE MOTOR*
```

POKE 514,0 pokes a zero into the low-order byte of the internal clock at memory address 514, wiping out the current jiffy time and resetting the clock to zero. The WAIT 514,16 inhibits further program action until the clock has incremented 16 jiffies. Meanwhile, the tape advances until memory address location 514 contains 16 and the following POKE turns the cassette motor off.

There is one drawback with this wait loop. Every time the jiffy clock is reset to zero the CBM loses track of time. Therefore, this routine should *not* be used if it is important within the program that real time be kept or used in any way.

Here is yet another way to implement a wait loop during the data tape advance:

```
POKE 59411,53
FOR I=1 TO 60:NEXT I
POKE 59411,61
```

This method is simple but less accurate than the previous two. Using a FOR-NEXT loop, the program is stalled as the loop increments to the maximum value of I before turning off the motor. However, the time it takes to increment through a FOR-NEXT loop cannot be measured as accurately as time measured in jiffies, and thus the interrecord gaps cannot be precise. One advantage with this method is that it does not alter or inhibit the use of the jiffy clock in any way.

Let's go back to the original wait loop and combine it with the routine that detects a full buffer. Below is a sample program which writes 100 numbers to a data tape with a FOR-NEXT loop. Within the loop is a check for a full buffer. If the buffer is full the data is written to the tape, and the subroutine at 1000 is called to create an interrecord gap:

```
10 POKE 243,122:POKE 244,2:OPEN1,1,1
20 FOR X=1 TO 100
30 PRINT#1,X
40 A=LEN(STR$(X))+1
50 IF (QT+A)>=191 GOSUB 1000      :REM *IF BUFFER FULL CALL SUB. TO ADVANCE TAPE
60 QT=QT+A
70 NEXT X
80 CLOSE1
90 END
1000 POKE 59411,53                :REM *START TAPE MOTOR*
1010 T=TI
1020 IF (TI-T)<10 GOTO 1020       :REM *WAIT 10 JIFFIES*
1030 POKE 59411,61                :REM *STOP TAPE MOTOR
1040 QT=0                         :REM *RESET ACCUMULATOR
1050 RETURN
```

where:

A	is the length of the printed string plus 1 for carriage return
QT	is the accumulator to add lengths of printed strings.

If you follow these suggestions and routines you should have little or no trouble writing and reading data files. But, if you find that you cannot get the files to work even with these routines, you should install the Revision Level 3 ROMs in your CBM computer.

Chapter 7: MEMORY MAP

All of the changes in Chapter 7 are based on the fact that the memory map for the Revision Level 2 ROMs was reorganized for the Revision Level 3 ROMs.

The detailed memory maps used by the different versions of CBM BASIC are shown in the back of this appendix.

Table F-1 describes the Revision Level 2 ROMs used in the original PET computers. Table F-2 shows the Revision Level 3 ROMs used in BASIC 3.0 CBM computers. Table F-3 shows the most recent memory map for the BASIC 4.0 CBM computers.

Tables F-1 and F-2 have a similar format; the Table F-3 format differs. Tables F-1 and F-2 show the memory address in decimal and hexadecimal, and also show sample decimal and hexadecimal equivalent values in memory locations. Table F-3 compares the BASIC 4.0 memory map with the BASIC 3.0 revision shown in Table F-2. The DESCRIPTION column provides the location description as currently used by Commodore; the LABEL column shows the assembly language label currently assigned to the location by Commodore. The BASIC 4.0 column gives the hexadecimal address of each location, while the BASIC 3.0 column gives the equivalent BASIC 3.0 hexadecimal

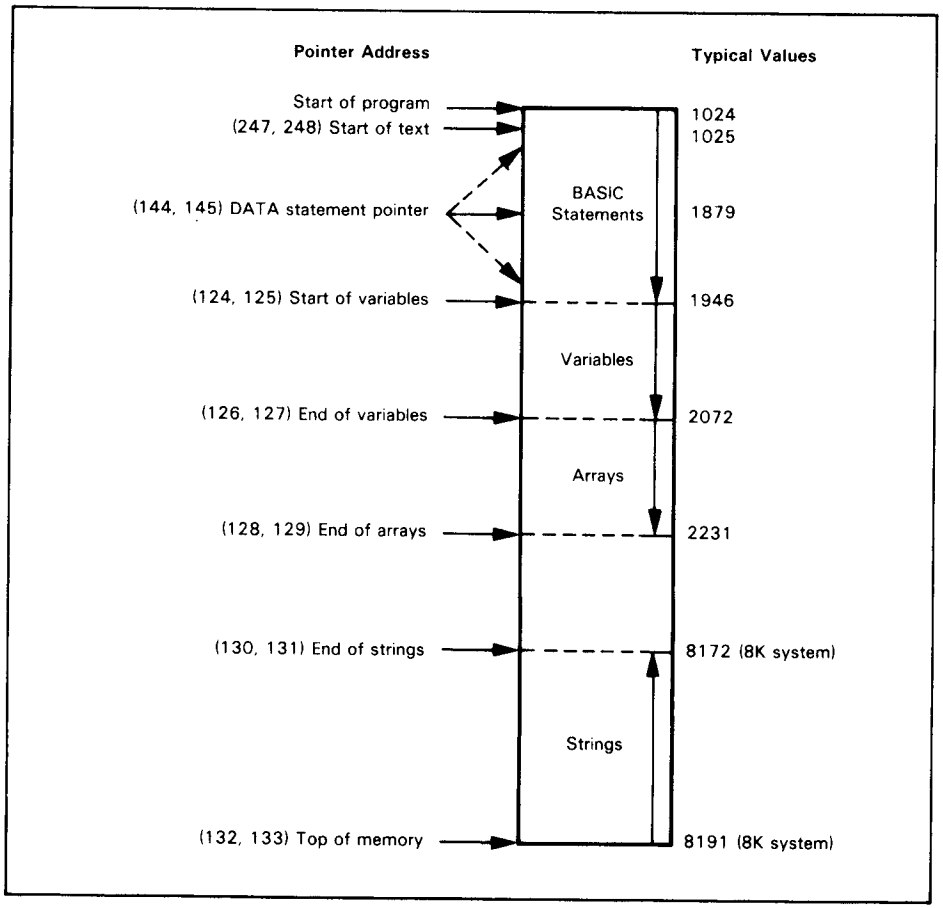


Figure F-1. Principal Pointers in User Program Area

address. To find any BASIC 4.0 location, first find the hexadecimal address given in Table F-2. Find this hexadecimal address in the BASIC 3.0 column of Table F-3 and the comparable BASIC 4.0 hexadecimal address is in the adjacent column.

With the exception of the first two entries in Table F-3, which actually represent memory address 0000, all subsequent 0000 addresses identify entries which do not exist in one version of BASIC or the other. For example, if you see an address in the BASIC 3.0 column with 0000 in the BASIC 4.0 column, then BASIC 4.0 has no equivalent location in its memory map. Conversely, a 0000 address in the BASIC 3.0 column identifies a new entry in the BASIC 4.0 memory map for which there is no BASIC 3.0 equivalent.

Chapter 7: CBM BASIC INTERPRETER

The system locations holding principal pointers in the user program area are different for the Revision Level 2 ROMs. Your pointers, in place of Figure 7-2, are as shown in Figure F-1. Figure F-2, replacing Figure 7-4, also reflects these changes.

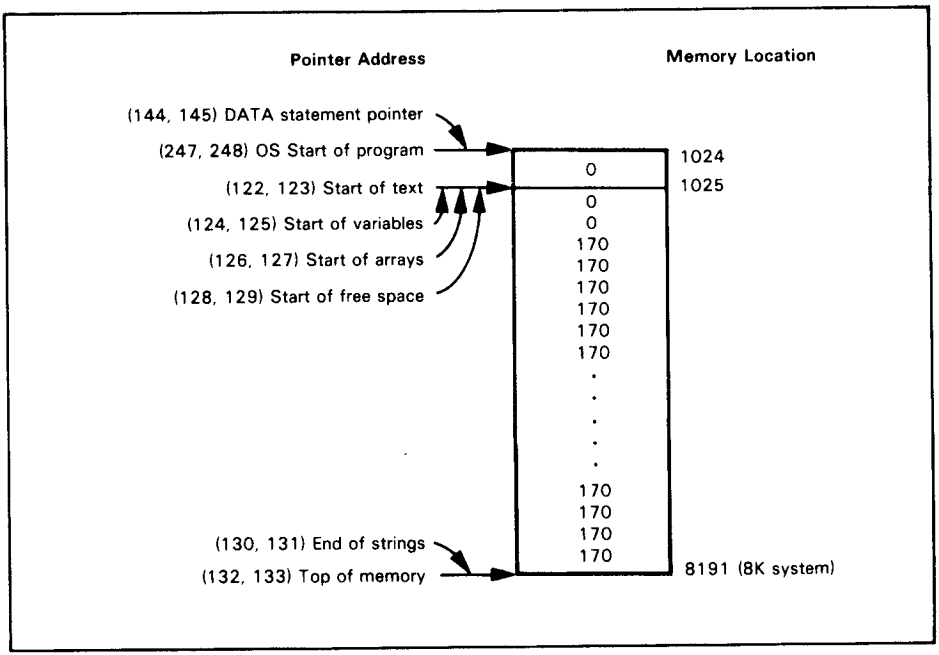


Figure F-2. User Program Area on Power-up

Chapter 7: VARIABLES, Floating Point Variable Format

Use the following program to examine floating point representations:

```
10 INPUT A
20 X=PEEK(125)*256+PEEK(124)+2
30 PRINT A;"=";PEEK(X);PEEK(X+1);PEK(X+2);PEEK(X+3);PEEK(X+4)
40 GOTO 10
```

This is the same one given in Chapter 7 except for the system locations at line 20 being PEEKed.

Chapter 7: CONSTANTS

Instead of pointer (42,43), the pointer in the diagrams is (124,125).

Chapter 7: ARRAY STORAGE FORMAT

Use the following program for viewing sample Array Area entries:

```
10 DIM A(5),B%(2,2),C$(10)      :REM SAMPLE ARRAYS
20 FOR I=0 TO 5:A(I)=I:NEXT
30 FOR I=0 TO 2:FOR J=0 TO 2:B%(J,I)=100+3*I+J:NEXT J,I
40 FOR I=0 TO 10:C$(I)=CHR$(ASC("A")+I):NEXT
50 X=PEEK(127)*256+PEEK(126)      :REM POINT TO ARRAY AREA
60 Y=PEEK(129)*256+PEEK(128)      :REM END OF ARRAYS
70 FOR I=X TO Y
80 PRINT I,PEEK(I)
90 GET D$:IF D$="" THEN GOTO 90:REM HIT KEY FOR NEXT ELEMENT
100 NEXT
```

This is the same as the program in Chapter 7 except for the system locations accessed in lines 50 and 60.

Chapter 7: ASSEMBLY LANGUAGE PROGRAMMING

For the Revision Level 2 ROMs, item 2, Top of Core discussion should read as follows:

2. Top of MEMORY. Memory locations 134 and 135 contain the pointer to the top of memory. On 8K CBMs this value is 8192. You can temporarily set the top of memory pointer to a lower address, thereby reserving a number of bytes from the new pointer value to the actual top of memory for storage of an assembly language program. To set the pointer, say, down 1000 bytes, you will need to store the value 7192 (8192-1000) converted into low, high address order, e.g.:

High Low

$7192_{10} = 1C18_{16} \rightarrow 1C_{16} = 28_{10}$ and $18_{16} = 24_{10}$

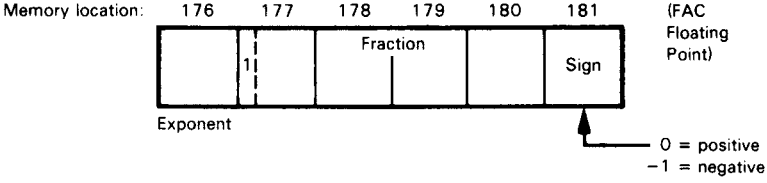
So 24 is to be stored at location 134 (low byte), and 28 is to be stored at location 135 (high byte). The following instructions can be used:

```
10 AL=PEEK(134):AH=PEEK(135):    REM SAVE CURRENT POINTER
20 POKE 132,24:POKE 135,28:      REM TOP OF CORE NOW = 7192
:
:
100 POKE 134,AL:POKE 135,AH:     REM RESTORE POINTER
110 END
```

Chapter 7: USR

Since the accumulator is maintained in different system locations on the Revision Level 2 ROMs, the accumulator description will read as described below.

The parameter value is passed to the USR subroutine in system locations that function as a floating point accumulator (FAC) for all functions. The FAC resides in six bytes from memory locations 176 to 181 ($B0_{16}-B5_{16}$). The FAC has the following format:



Like floating point variables, the exponent is stored in excess 128 format, and the fraction is normalized with the high-order bit of byte 177 (the high-order byte of the fraction) set to 1. The difference between this format and the variable format is that the high-order 1 bit is present in byte 177 of the FAC. An extra byte (181) is used to hold the sign of the fraction. (This is done for ease of manipulation by the functions that use the FAC.)

1. *PET User Notes*, Volume 1, Issue 6, Sept.-Oct. 1978, p. 14, "Cassette File Usage Summary" by Jim Butterfield.
2. *Best of the PET Gazette*, p. 38, "On Data Files" by Michael Richter.

Table F-1. CBM Memory Map (Rev. 2 ROMs)

Memory Address		Sample Value		Description
Decimal	Hexadecimal	Decimal	Hexadecimal	
			Page 0 (0-255)	
				USR Function Locations
0	0000	76	4C	Constant 6502 JMP instruction
1-2	0001-0002	826	033A	User address jump vector
				Terminal I/O Maintenance
3	0003	0	00	Active input device number (0=keyboard)
4	0004	0	00	No. of nulls to print after CR/LF (0=normal)
5	0005	0	00	Cursor position for POS function (0-255)
6	0006	127	7F	Terminal width (unused)
7	0007	127	7F	Limit for scanning source columns (unused)
8	0008	60	3C	Line number storage preceding buffer
9	0009	3	03	Constant
10-89	000A-0059	48	30	BASIC input line buffer (80 bytes)
90	005A	0	00	General counter for BASIC
91	005B	0	00	Delimiter flag for quote mode scan
92	005C	255	FF	Input buffer pointer, general counter
				Evaluation of Variables
93	005D	0	00	Flag for dimensioned variables
94	005E	0	00	Flag for variable type: 00=numeric FF=string
95	005F	0	00	Flag for numeric variable type: 00=floating point 80=integer
96	0060	0	00	Flag to allow reserved words in strings and remarks
97	0061	0	00	Flag to allow subscripted variable
98	0062	0	00	Flag for input type: 0=INPUT 64=GET 152=READ
99	0063	0	00	Flag sign of TAN function
100	0064	0	00	Flag to suppress output: + normal – suppressed
101	0065	104	68	Index to next available descriptor
102-103	0066-0067	101	0065	Pointer to last string temporary
104-111	0068-006F	2	0002	Table of double-byte descriptors that point to variables (8 bytes)
112-113	0070-0071	14525	38BD	Indirect index #1
114-115	0072-0073	62983	F607	Indirect index #2
116	0074	1	01	Pseudo-register for function operands (6 bytes)
117	0075	234	EA	
118	0076	0	00	
119	0077	0	00	
120	0078	0	00	
121	0079	0	00	

Table F-1. CBM Memory Map (Rev. 2 ROMs) (Continued)

Memory Address		Sample Value		Description
Decimal	Hexadecimal	Decimal	Hexadecimal	
				Data BASIC Storage Maintenance
122-123	007A-007B	1025	0401	Pointer to start of text
124-125	007C-007D	1946	079A	Pointer to start of variables
126-127	007E-007F	2072	0818	Pointer to end of variables
128-129	0080-0081	2231	08B7	Pointer to end of arrays
130-131	0082-0083	8192	2000	Pointer to start of strings (moving down)
132-133	0084-0085	8191	1FFF	Pointer to end of strings (top of available RAM)
134-135	0086-0087	8192	2000	Pointer to limit of BASIC memory
136-137	0088-0089	2000	07D0	Line number of current line being executed −1 in 137=direct mode statement
138-139	008A-008B	110	006E	Line number for last line executed before CONT
140-141	008C-008D	1922	0782	Pointer to next line to be executed after CONT
142-143	008E-008F	1150	047E	Line number of current DATA line
144-145	0090-0091	1879	0757	Pointer to current DATA line
146-147	0092-0093	13	000D	Next DATA item within line
148-149	0094-0095	89	0059	Current variable name
150-151	0096-0097	2032	07F0	Pointer to current variable
152-153	0098-0099	2032	07F0	Pointer to next FOR. . . NEXT variable
154-155	009A-009B	31999	7CFF	Pointer to current operator in ROM table
156	009C	0	00	Mask for current logical operator
157-158	009D-009E	898	0382	Pointer to user function FN definition
159-160	009F-00A0	104	0068	Pointer to a string description
161	00A1	221	DD	Length of string
162	00A2	3	03	Constant used by garbage collection routine
163	00A3	76	4C	Constant 6502 JMP instruction
164-165	00A4-00A5	0	0000	Jump vector for user function FN
166-171	00A6-00AB	129	81	Floating point accumulator #3 (6 bytes)
172-173	00AC-00AD	0	00	Block transfer pointer #1
174-175	00AE-00AF	0	00	Block transfer pointer #2
176-181	00B0-00B5			Floating point accumulator (FAC) #1 (6 bytes)
		0	00	176 00B0 Exponent +128
		0	00	177 00B1 Fraction MSB Floating Point
		0	00	178 00B2 Fraction
		0	00	179 00B3 Fraction MSB Integer
		0	00	180 00B4 Fraction LSB
		0	00	181 00B5 Sign of fraction (0 if zero or positive, −1 if negative)
182	00B6	0	00	Copy of FAC #1 sign of fraction
183	00B7	0	00	Counter for number of bits to shift to normalize FAC #1
184-189	00B8-00BD	0	00	Floating point accumulator #2 (6 bytes)
190	00BE	0	00	Overflow byte for floating argument
191	00BF	0	00	Copy of FAC #2 sign of fraction
192-193	00C0-00C1	258	0102	Conversion pointer

Table F-1. CBM Memory Map (Rev. 2 ROMs) (Continued)

Memory Address		Sample Value		Description
Decimal	Hexadecimal	Decimal	Hexadecimal	
				RAM Subroutines
194-199	00C2-00C7	230	E6	Routine to fetch next BASIC character
200	00C8	173	AD	Entry to refetch current character
201-202	00C9-00CA	1929	0789	Pointer to source text
203-223	00CB-00DF	201	C9	Work area for RND function
				OS Page Zero Storage
224-225	00E0-00E1	33728	83C0	Pointer to start of line where cursor is flashing
226	00E2	0	00	Column position where cursor is flashing (0-79)
227-228	00E3-00E4	33792	8400	Utility pointer
229-230	00E5-00E6	1929	0789	End of current program
231-233	00E7-00E9	254	FE	Utility
234	00EA	0	00	Flag for quote mode. 0=not quote mode
235-237	00EB-00ED	192	C0	Utility
238	00EE	0	00	No. of characters in current file name
239	00EF	5	05	Current logical file number
240	00F0	255	FF	GPIB primary address
241	00F1	63	3F	GPIB device number
242	00F2	39	27	Max. no. of characters on current line (39,79)
243-244	00F3-00F4	634	027A	Pointer to start of current tape buffer (634 or 826)
245	00F5	23	17	Line number where cursor is flashing (0-24)
246	00F6	10	0A	I/O storage
247-248	00F7-00F8	1024	0400	OS pointer to program
249-250	00F9-00FA	3100	0C1C	Pointer to current file name
251	00FB	0	00	Number of Insert keys pushed to go
252	00FC	9	09	Serial bit shift word
253	00FD	0	00	Number of blocks remaining to read/write
254	00FE	0	09	Serial word buffer
255	00FF	243	F3	Overflow byte for binary to ASCII conversions
				Page 1 (256-511)
256-up	0100-up	32	20	Tape read working storage (up to 511) and conversion stg. 256-318 For error correction in tape reads (62 bytes) 256-266 Binary to ASCII conversion (11 bytes)
511-down	01FF-down	0	00	Stack (down to 256)
				Page 2-3 (512-1023)
				OS Working Storage
512-514	0200-0202	3801352	3A0108	24-hour clock incremented every 1/60 second (jiffy). Resets every 5,184,000 jiffies (24 hours). Stored in low to high order.

Table F-1. CBM Memory Map (Rev. 2 ROMs) (Continued)

Memory Address		Sample Value		Description
Decimal	Hexadecimal	Decimal	Hexadecimal	
515	0203	255	FF	Matrix coordinate of key depressed at current jiffy. 1-80=key 255=no key
516	0204	0	00	Status of SHIFT key: 0=unshifted (up) 1=shifted (down)
517-518	0205-0206	37916	941C	Secondary jiffy clock
519	0207	52	34	Interrupt driver flag for cassette #1 ON switch
520	0208	0	00	Interrupt driver flag for cassette #2 ON switch
521	0209	255	FF	Keyswitch PIA
522	020A	0	00	Utility
523	020B	0	00	I/O flag: 0=LOAD 1=VERIFY
524	020C	0	00	I/O status byte
525	020D	0	00	Number of characters in keyboard buffer (0 to 9)
526	020E	0	00	Flag to indicate reverse field on (0=normal)
527-536	020F-0218	85	55	Keyboard buffer (10 bytes)
537-538	0219-021A	34048	8500	Hardware interrupt vector
539-540	021B-021C	0	0000	6502 BRK instruction interrupt vector
541-546	021D-0222			Input routine storage (6 bytes)
		13	0D	542 021E No. of characters on screen line
547	0223	255	FF	Key image
548	0224	1	01	Flag for cursor enable: 0=Enable 1=Disable
549	0225	11	0B	Counter to flip cursor (20 to 1)
550	0226	32	20	Copy of character at current cursor position
551	0227	0	00	Flag for cursor on/off: 0=cursor moved 1=blink started
552	0228	0	00	Flag for tape write
553-577	0229-0241			High byte of screen line addresses 553-559=128 (lines 1-7) 560-565=129 (lines 8-13) 566-572=130 (lines 14-20) 573-577=131 (lines 21-25)
578-587	0242-024B	5	05	Table of logical numbers of open files
588-597	024C-0255	5	05	Table of device numbers of open files
598-607	0256-025F	255	FF	Table of secondary address modes of open files
608	0260	0	00	Flag for input source: 0=keyboard buffer 1=screen memory
609	0261	0	00	I/O utility
610	0262	1	01	Number of open files (index into tables)

Table F-1. CBM Memory Map (Rev. 2 ROMs) (Continued)

Memory Address		Sample Value		Description
Decimal	Hexadecimal	Decimal	Hexadecimal	
611	0263	0	00	Default input device number (0=keyboard)
612	0264	3	03	Default output device number (3=screen)
613	0265	0	00	Tape parity byte
614	0266	0	00	I/O utility
615	0267	0	00	I/O utility
616	0268	0	00	Byte pointer in filename transfer
617	0269	0	00	I/O utility
618	026A	255	FF	I/O utility
619	026B	0	00	I/O utility
620	026C	8	08	Serial bit count
621	026D	0	00	Count of redundant tape blocks
622	026E	0	00	Tape utility
623	026F	0	00	Cycle counter flip for each bit read from tape
624	0270	0	00	Countdown synchronization on tape write
625	0271	0	00	Tape buffer 1 index to next character
626	0272	0	00	Tape buffer 2 index to next character
627	0273	0	00	Countdown synchronization on tape read
628	0274	0	00	Flag to indicate bit/byte tape error
629	0275	0	00	Flag to indicate tape error 0=first half-byte marker not written
630	0276	0	00	Flag to indicate tape error 0=2nd half-byte marker not written /Tape dropout counter
631	0277	0	00	Tape dropout counter
632	0278	128	80	Flag for tape read current function
633	0279	9	09	Checksum utility
634-825	027A-0339	1	01	Tape buffer for cassette #1 (192 bytes)
826-1017	033A-03F9	173	AD	Tape buffer for cassette #2 (192 bytes)
1018-1023	03FA-03FF	28	1C	Utility space/unused.
		Page 4-32 (1024-8191)		
1024-8191	0400-1FFF	0	00	User program area
		Page 33-128 (8192-32767)		
8192-32767	2000-7FFF	0	00	Expansion RAM
		Page 129-144 (32768-36863)		
32768-36863	8000-8FFF	12	0C	TV RAM 32768-33767 Display memory (1000 bytes)
		Page 145-192 (36864-49151)		
36864-49151	9000-BFFF	0	00	Expansion ROM
		Page 193-232 BASIC (49152-59391)		
				Pointers to BASIC Routines
49152-49153	C000-C001	50973	C71D	Pointer −1 to END*
49154-49155	C002-C003	50760	C648	Pointer −1 to FOR
49156-49157	C004-C005	52277	CC35	Pointer −1 to NEXT

* These memory locations contain the address of the byte preceding the specified BASIC routines

Table F-1. CBM Memory Map (Rev. 2 ROMs) (Continued)

Memory Address		Sample Value		Description
Decimal	Hexadecimal	Decimal	Hexadecimal	
49158-49159	C006-C007	51183	C73F	Pointer −1 to DATA
49160-49161	C008-C009	51909	CAC5	Pointer −1 to INPUT#
49162-49163	C00A-C00B	51935	CADF	Pointer −1 to INPUT
49164-49165	C00C-C00D	53104	CF70	Pointer −1 to DIM
49166-49167	C00E-C00F	52003	CB23	Pointer −1 to READ
49168-49169	C010-C011	51356	C89C	Pointer −1 to LET
49170-49171	C012-C013	51100	C79C	Pointer −1 to GOTO
49172-49173	C014-C015	51060	C774	Pointer −1 to RUN
49174-49175	C016-C017	51231	C81F	Pointer −1 to IF
49176-49177	C018-C019	50956	C70C	Pointer −1 to RESTORE
49178-49179	C01A-C01B	51071	C77F	Pointer −1 to GOSUB
49180-49181	C01C-C01D	51145	C7C9	Pointer −1 to RETURN
49182-49183	C01E-C01F	51250	C832	Pointer −1 to REM
49184-49185	C020-C021	50971	C71B	Pointer −1 to STOP
49186-49187	C022-C023	51266	C842	Pointer −1 to ON
49188-49189	C024-C025	55041	D701	Pointer −1 to WAIT
49190-49191	C026-C027	65492	FFD4	Pointer −1 to LOAD
49192-49193	C028-C029	65495	FFD7	Pointer −1 to SAVE
49194-49195	C02A-C02B	65498	FFDA	Pointer −1 to VERIFY
49196-49197	C02C-C02D	53908	D294	Pointer −1 to DEF
49198-49199	C02E-C02F	55032	D6F8	Pointer −1 to POKE
49200-49201	C030-C031	51582	C97E	Pointer −1 to PRINT#
49202-49203	C032-C033	51614	C99E	Pointer −1 to PRINT
49204-49205	C034-C035	51012	C744	Pointer −1 to CONT
49206-49207	C036-C037	50599	C5A7	Pointer −1 to LIST
49208-49209	C038-C039	51055	C76F	Pointer −1 to CLR
49210-49211	C03A-C03B	51588	C984	Pointer −1 to CMD
49212-49213	C03C-C03D	65501	FFDD	Pointer −1 to SYS
49214-49215	C03E-C03F	65471	FFBF	Pointer −1 to OPEN
49216-49217	C040-C041	65474	FFC2	Pointer −1 to CLOSE
49218-49219	C042-C043	51870	CA9E	Pointer −1 to GET
49220-49221	C044-C045	50512	C550	Pointer −1 to NEW
49222-49223	C046-C047	56075	DB0B	Pointer to SGN**
49224-49225	C048-C049	56222	DB9E	Pointer to INT
49226-49227	C04A-C04B	56106	DB2A	Pointer to ABS
49228-49229	C04C-C04D	0	0000	Pointer to USR pointer
49230-49231	C04E-C04F	53860	D264	Pointer to FRE
49232-49233	C050-C051	53893	D285	Pointer to POS
49234-49235	C052-C053	56868	DE24	Pointer to SQR
40236-49237	C054-C055	57157	DF45	Pointer to RND
49238-49239	C056-C057	55487	D8BF	Pointer to LOG
49240-49241	C058-C059	56992	DEA0	Pointer to EXP
49242-49243	C05A-C05B	57246	DF9E	Pointer to COS
49244-49245	C05C-C05D	57253	DFA5	Pointer to SIN
49246-49247	C05E-C05F	57326	DFEE	Pointer to TAN
49248-49249	C060-C061	57416	E048	Pointer to ATN
49250-49251	C062-C063	55014	D6E6	Pointer to PEEK
49252-49253	C064-C065	54868	D654	Pointer to LEN
49254-49255	C066-C067	54089	D349	Pointer to STR$
49256-49257	C068-C069	54917	D685	Pointer to VAL
49258-49259	C06A-C06B	54883	D663	Pointer to ASC
49260-49261	C06C-C06D	54724	D5C4	Pointer to CHR$
49262-49263	C06E-C06F	54744	D5D8	Pointer to LEFT$

** These memory locations contain the address of the first byte of the specified BASIC routines

Table F-1. CBM Memory Map (Rev. 2 ROMs) (Continued)

Memory Address		Sample Value		Description
Decimal	Hexadecimal	Decimal	Hexadecimal	
49264-49265	C070-C071	54788	D604	Pointer to RIGHT$
49266-49267	C072-C073	54799	D60F	Pointer to MID$
49268-57343	C074-DFFF			**BASIC Routines**

Starting Address		Function
49836	C2AC	FOR...NEXT stack check
49882	C2DA	Insert line space marker
49949	C31D	Stack overflow check
50007	C357	Error message abort
50057	C389	READY
50068	C394	Execute line
50092	C3AC	Handle new line
50224	C430	Rechain lines after insert/delete
50274	C462	Input line
50297	C479	Get character from input line
50317	C48D	Keyword encoder
50466	C522	Line number search
50513	C551	NEW
50586	C59A	Set pointer to start of program
50600	C5A8	LIST
50761	C649	FOR...NEXT
50869	C6B5	Statement processor
50930	C6F2	Statement execute
50957	C70D	RESTORE
50972	C71C	STOP
50974	C71E	END
51013	C745	CONT
51056	C770	CLR
51061	C775	RUN
51072	C780	GOSUB
51101	C79D	GOTO
51146	C7CA	RETURN
51184	C7F0	DATA
51198	C7FE	Next line scan
51232	C820	IF
51251	C833	REM
51267	C843	ON...GOTO/GOSUB
51299	C863	Number fetch
51357	C89D	LET=
51484	C91C	Digit check
51583	C97F	PRINT#
51589	C985	CMD
51615	C99F	PRINT
51751	CA27	Print string
51780	CA44	Print character
51831	CA77	Input data error
51871	CA9F	GET

Table F-1. CBM Memory Map (Rev. 2 ROMs) (Continued)

Memory Address		Sample Value		Description
Decimal	Hexadecimal	Decimal	Hexadecimal	
				51910 CAC6 INPUT#
				51936 CAED INPUT
				51991 CB17 Input prompt
				52004 CB24 READ
				52242 CC12 Error messages
				52278 CC36 NEXT
				52370 CC92 Format checker
				52408 CCB8 Expression evaluator
				52538 CD3A Stack argument
				52637 CD9D Symbol evaluator
				52668 CDBC Pi
				53105 CF71 DIM
				53207 CFD7 Variable table look-up
				53415 D0A7 Floating-to-integer
				53860 D264 FRE
				53880 D278 Integer-to-floating
				53893 D285 POS
				53909 D295 DEF
				54089 D349 STR$
				54724 D5C4 CHR$
				54744 D5D8 LEFT$
				54788 D604 RIGHT$
				54799 D60F MID$
				54868 D654 LEN
				54883 D663 ASC
				54917 D685 VAL
				55014 D6E6 PEEK
				55033 D6F9 POKE
				55042 D702 WAIT
				55080 D728 Subtraction
				55103 D73F Addition
				55487 D8BF LOG
				55552 D900 Multiplication
				55646 D95E Load number to AFAC
				55650 D962 Load variable to AFAC
				55780 D9E4 Division
				55924 DA74 Load Accumulator (FAC)
				55928 DA78 Load variable to FAC
				55979 DAAB Store variable from FAC
				56075 DB0B SGN
				56106 DB2A ABS
				56222 DB9E INT
				56868 DE24 SQR
				56878 DE2E Raise AFAC to power FAC
				56992 DEA0 EXP
				57157 DF45 RND
				57246 DF9E COS
				57253 DFA5 SIN
				57326 DFEE TAN

Table F-1. CBM Memory Map (Rev. 2 ROMs) (Continued)

Memory Address		Sample Value		Description
Decimal	Hexadecimal	Decimal	Hexadecimal	
57344-59391	E000-E7FF			**Screen Editor** **Starting Address — Function** 57416 E048 — ATN 57525 E0B5 — Initialize BASIC system 57910 E236 — Clear screen 57981 E27D — Character fetch
58004-58986	E294-E66A			Video driver 58282 E3AA — Scroll processor 58346 E3EA — Video display routine 58185 E349 — Quote mode ($EA) switcher 58346 E3EA — Print character 58713 E559 — Scroll 1 line 58758 E586 — Interrupt Request (IRQ)
58987-59012	E66B-E684			Interrupt handler
59013-59198	E685-E73E			Clock update
59199-59227	E73F-E75B			Keyboard scan
59228-59348	E75C-E7D4			Keyboard encoding table
	Page 233-240 I/O Ports and Expansion I/O (PIA's and VIA) (59392-61439)			
				Keyboard PIA (59408-59411)
59408	E810	233	E9	I/O Port A and Data Direction register
59409	E811	60	3C	Control Register A — screen blanking 52=Screen off (blanked) 60=Screen on
59410	E812	255	FF	I/O Port B and Data Direction register 255=all keys except: 254=RVS key 253=key 251=SPACE key 247= < key
59411	E813	61	3D	Control Register B — #1 cassette motor 53=motor on 61=motor off
				IEEE Port PIA (59424-59427)
59424	E820	255	FF	I/O Port A and Data Direction register PEEK (59424) reads input data.
59425	E821	188	BC	Control Register A — set output line CA2 POKE 59425,52=low POKE 59425,60=high
59426	E822	255	FF	I/O Port B and Data Direction register POKE 59426,data writes output data POKE 59426,255 before a read to Port A
59427	E823	60	3C	Control Register B — set output line CB2 POKE 59427,52=low POKE 59427,60=high

Table F-1. CBM Memory Map (Rev. 2 ROMs) (Continued)

Memory Address		Sample Value		Description
Decimal	Hexadecimal	Decimal	Hexadecimal	
				Parallel User Port VIA (59456-59471)
59456	E840	254	FE	I/O Port B 207=#2 cassette motor on 223=#2 cassette motor off WAIT 59456,23,23 waits for vertical retrace of display Bit 1=PB1 (NFRD on IEEE connector) output line Bit 3=PB3 (ATN on IEEE connector) output line
59457	E841	255	FF	I/O Port A with handshaking
59458	E842	30	1E	Data Direction register for I/O Port B
59459	E843	0	00	Data Direction register for I/O Port A For each bit 1=output, 0=input =0 all input =255 all output
59460-59461	E844-E845	25248	62A0	(Low, high order) Read Timer 1 Counter; write to Timer 1 Latch and (high byte) initiate count
59462-59463	E846-E847	65381	FF65	(Low, high order) Read Timer 1 Latch
59464	E848	113	71	Read Timer 2 Counter low byte and reset interrupt; write to Timer 2 low byte PEEK (59464) Clock decrements every microsecond POKE 59464,n sets SR rate of shift from high (n=0) to low (n=255) for music from User Port.
59465	E849	200	C8	Read Timer 2 Counter high byte; write to Timer 2 high byte and reset interrupt. PEEK (59465) Clock decrements every 256 microseconds
59466	E84A	1	01	Serial I/O Shift register (SR) POKE 59466,15 or 51 or 85 to generate square wave output at CB2 for playing music from User Port.
59467	E84B	0	00	Auxiliary Control register. =16 Sets SR to free-running mode for music from User Port. =0 for proper operation of tape drive
59468	E84C	14	0E	Peripheral Control register =12 for graphics on shifted characters =14 for lower-case letters on shifted characters
59469	E84D	0	00	Interrupt Flag register
59470	E84E	128	80	Interrupt Enable register
59471	E84F	255	FF	I/O Port A without handshaking
		Page 241-256 Operating System (61440-65535)		
61622-61904	F0B6-F1D0			**File Control**
61905-63532	F1D1-F82C			**Starting Address** **Function** 61905 F1D1 Get a character (without cursor) 61921 F1E1 Input a character (with cursor)

Table F-1. CBM Memory Map (Rev. 2 ROMs) (Continued)

Memory Address		Sample Value		Description
Decimal	Hexadecimal	Decimal	Hexadecimal	
				62002 F232 Display a character
				62026 F24A Close all files
				62121 F2A9 CLOSE
				62250 F32A STOP search
				62278 F346 Tape playback
				62402 F3C2 LOAD
				62481 F411 Display filename
				62515 F433 Fetch file number
				62556 F45C Number fetch
				62647 F4B7 VERIFY
				62724 F504 Fetch filename
				62741 F515 Fetch tape character
				62753 F521 OPEN
				62824 F568 Record SAVE routine
				62894 F5AE Tape header search
				62947 F5E3 Clear current tape buffer
				62957 F5ED Write tape end block
				63101 F67D Set up tape end pointer
				63108 F684 SYS
				63134 F69E SAVE
				63153 F6B1 SAVE memory block on cassette
				63273 F729 Update secondary jiffy clock
63533-64789	F82D-FD15			**Tape Control**
				63582 F85E Check for cassette on
				63615 F87F Tape read to buffer
				63684 F8C4 Write block to tape
				63765 F915 Interrupt wait
64824-65458	FD38-FFB2			**Power-On Diagnostics**
				64824 FD38 System reset SYS (64824) simulates power-on reset
				64909 FD8D Reset BASIC (does not affect User Program)
				64912 FD90 EOT-buffer compare
65472-65516	FFC0-FFEC			**Jump Vectors**
65472-65474	FFC0-FFC2	76 62753	4C F521	JMP OPEN
65475-65477	FFC3-FFC5	76 62121	4C F2A9	JMP CLOSE
65487-65489	FFCF-FFD1	76 61921	4C F1E1	JMP RDT
65490-65492	FFD2-FFD4	76 62002	4C F232	JMP WRT
65493-65495	FFD5-FFD7	76 62402	4C F3C2	JMP LOAD
65496-65498	FFD8-FFDA	76 63134	4C F69E	JMP SAVE
65499-65501	FFDB-FFDD	76 62647	4C F4B7	JMP VERIFY
65502-65504	FFDE-FFED	76 63108	4C F684	JMP SYS
65508-65510	FFE4-FFE6	76 61905	4C F1D1	JMP GETC
65514-65516	FFEA-FFEC	76 63273	4C F729	JMP Clock Update
65530-65535	FFFA-FFFF			**6502 Interrupt Vectors**
65530-65531	FFFA-FFFB	51808	CA60	Non-maskable interrupt (NMI)
65532-65533	FFFC-FFFD	64824	FD38	System reset (RESET)
65534-65535	FFFE-FFFF	58987	E66B	Interrupt request, break (IRQ+BRK)

Table F-2. CBM Memory Map (Rev. 3 ROMs)

Memory Address		Sample Value		Description
Decimal	Hexadecimal	Decimal	Hexadecimal	
			Page 0 (0-255)	
				USR Function Locations
0	0000	76	4C	Constant 6502 JMP instruction
1-2	0001-0002	826	033A	User address jump vector
				Evaluation of Variables and Terminal I/O Maintenance
3	0003	0	00	Search character
4	0004	0	00	Delimiter flag for quote mode scan
5	0005	255	FF	Input buffer pointer, general counter
6	0006	0	00	Flag for dimensioned variables
7	0007	0	00	Flag for variable type: 00=numeric FF=string
8	0008	0	00	Flag for numeric variable type: 00=floating point 80=integer
9	0009	0	00	Flag for DATA scan; LIST quote; memory
10	000A	0	00	Flag to allow subscripted variable; FNx flag
11	000B	0	00	Flag for input type: 0=INPUT 64=GET 152=READ
12	000C	0	00	Flag for ATN sign; comparison evaluation
13	000D	0	00	Flag to suppress output: + normal – suppressed
14	000E	0	00	Current I/O device for prompt-suppress
15	000F	40	28	Terminal width (unused)
16	0010	30	1E	Limit for scanning source columns (unused)
17-18	0011-0012	828	033C	Basic integer address (for SYS, GOTO, etc.)
19	0013	22	16	Index to next available descriptor
20-21	0014-0015	19	13	Pointer to last string temporary
22-29	0016-001D	2	0002	Table of double-byte descriptions that point to variables (8 bytes)
30-31	001E-001F	16451	4043	Indirect index #1
32-33	0020-0021	26119	6607	Indirect index #2
34	0022	1	01	Pseudo-register for function operands (6 bytes)
35	0023	140	8C	
36	0024	0	00	
37	0025	0	00	
38	0026	0	00	
39	0027	0	00	

Table F-2. CBM Memory Map (Rev. 3 ROMs) (Continued)

Memory Address		Sample Value		Description
Decimal	Hexadecimal	Decimal	Hexadecimal	
				Data Storage Maintenance
40-41	0028-0029	1025	0401	Pointer to start of BASIC text
42-43	002A-002B	1920	0780	Pointer to start of variables
44-45	002C-002D	2032	07F0	Pointer to end of variables
46-47	002E-002F	2191	088F	Pointer to end of arrays
48-49	0030-0031	8192	2000	Pointer to start of strings (moving down)
50-51	0032-0033	8191	1FFF	Pointer to end of strings (top of available RAM)
52-53	0034-0035	8192	2000	Pointer to limit of BASIC memory
54-55	0036-0037	2000	07D0	Current line number. Loc. 55=2 if no program yet executed
56-57	0038-0039	110	006E	Previous line number
58-59	003A-003B	1897	0769	Pointer to next line to be executed (for CONT)
60-61	003C-003D	200	00C8	Line number of current DATA line
62-63	003E-003F	1855	073F	Pointer to current DATA item
				Expression Evaluation
64-65	0040-0041	514	0202	INPUT vector
66-67	0042-0043	89	0059	Current variable name.
68-69	0044-0045	2006	07D6	Pointer to current variable
70-71	0046-0047	2006	07D6	Pointer to current FOR. . . NEXT variable
72-73	0048-0049	1279	04FF	Pointer to current operator in ROM table
74	004A	0	00	Mask for current logical operator
75-76	004B-004C	62268	F33C	Pointer to user function FN definition
77-78	0040-004E	26531	67A3	Pointer to a string description
79	004F	243	F3	Length of string
80	0050	3	03	Constant used by garbage collection routine
81	0051	76	4C	Constant 6502 JMP instruction
82-83	0052-0053	0	00	Jump vector for functions
84-89	0054-0059	211	D3	Floating point accumulator #3 (6 bytes)
90-91	005A-005B	0	0000	Block transfer pointer #1
92-93	005C-005D	0	0000	Block transfer pointer #2
94-99	005E-0063			Floating point accumulator (FAC) #1 (6 bytes)
		0	00	94 005E Exponent +128
		0	00	95 005F Fraction MSB Floating Point
		0	00	96 0060 Fraction
		0	00	97 0061 Fraction MSB Integer
		0	00	98 0062 Fraction LSB
		0	00	99 0063 Sign of fraction (0 if zero or positive, −1 if negative)
100	0064	0	00	Copy of FAC #1 sign of fraction
101	0065	0	00	Counter for number of bits to shift to normalize FAC #1
102-107	0066-006B	0	00	Floating point accumulator #2 (6 bytes)
108	006C	0	00	Overflow byte for floating argument
109	006D	0	00	Copy of FAC #2 sign of fraction
110-111	006E-006F	258	0102	Conversion pointer

Table F-2. CBM Memory Map (Rev. 3 ROMs) (Continued)

Memory Address		Sample Value		Description
Decimal	Hexadecimal	Decimal	Hexadecimal	
				RAM Subroutines
112-135	0070-0087	230 173 1904	E6 AD 0770	Routine to fetch next BASIC character 118 76 Entry to refetch current character 119-120 77-78 Pointer into source text
136-140	0088-008C	128	80	Next random no. in storage and RND work area
				OS Page Zero Storage
141-143	008D-008F	398710	061576	24-hour clock incremented every 1/60 second (jiffy). Resets every 5,184,000 jiffies (24 hours). Stored in high to low order
144-145	0090-0091	58926	E62E	Hardware interrupt vector
146-147	0092-0093	64791	FD17	6502 BRK instruction interrupt vector
148-149	0094-0095	50057	C389	NMI interrupt vector
150	0096	0	00	Status word ST (1 byte)
151	0097	255	FF	Matrix coordinate of key depressed at current jiffy. 1-80=key. 255=no key
152	0098	0	00	Status of SHIFT key: 0=unshifted (up) 1=shifted (down)
153-154	0099-009A	65282	FF02	Correction factor for clock
155	009B	255	FF	Keyswitch PIA: STOP and RVS flags
156	009C	0	00	Timing constant buffer
157	009D	0	00	I/O flag: 0=LOAD 1=VERIFY
158	009E	0	00	Number of characters in keyboard buffer (0 to 9)
159	009F	0	00	Flag to indicate reverse field on (0=normal)
160	00A0	0	00	IEEE 488 output flag FF=character waiting
161	00A1	13	0D	Byte pointer to end of line for input
162	00A2	0	00	Utility
163-164	00A3-00A4	11, 13	0B, 0D	Cursor log (row, column)
165	00A5	63	3F	IEEE 488 output character buffer
166	00A6	255	FF	Key image
167	00A7	1	01	Flag for cursor enable: 0=Enable 1=Disable
168	00A8	17	11	Counter to flip cursor (20 to 1)
169	00A9	32	20	Copy of character at current cursor position
170	00AA	0	00	Flag for cursor on/off: 0=cursor moved 1=blink started
171	00AB	0	00	Flag for tape write
172	00AC	0	00	Flag for input source: 0=keyboard buffer 1=screen memory

Table F-2. CBM Memory Map (Rev. 3 ROMs) (Continued)

Memory Address		Sample Value		Description
Decimal	Hexadecimal	Decimal	Hexadecimal	
				OS Page Zero Storage (Continued)
173	00AD	0	00	I/O utility: X save flag
174	00AE	1	01	Number of open files (index into tables)
175	00AF	0	00	Default input device number (0=keyboard)
176	00B0	3	03	Default output device number (3=screen)
177	00B1	0	00	Tape parity byte
178	00B2	0	00	Flag for byte received
179	00B3	0	00	I/O utility
180	00B4	0	00	Tape buffer character
181	00B5	0	00	Byte pointer in filename transfer
182	00B6	0	00	I/O utility
183	00B7	0	00	Serial bit count
184	00B8	0	00	Tape utility
185	00B9	0	00	Cycle counter — flip for each bit read from tape
186	00BA	0	00	Countdown synchronization on tape write
187	00BB	0	00	Tape buffer 1 index to next character
188	00BC	0	00	Tape buffer 2 index to next character
189	00BD	0	00	Countdown synchronization on tape read
190	00BE	0	00	Flag to indicate bit/byte tape error
191	00BF	0	00	Flag to indicate tape error 0=first half-byte marker not written
192	00C0	0	00	Flag to indicate tape error 0=2nd half-byte marker not written
193	00C1	0	00	Tape dropout counter
194	00C2	0	00	Flag for cassette read current function 0=scan, 1-15=count, 40_{16}=load, 80_{16}=end
195	00C3	0	00	Checksum utility
196-197	00C4-00C5	33728	83CD	Pointer to start of line where cursor is flashing
198	00C6	0	00	Column position where cursor is flashing (0-79)
199-200	00C7-00C8	33792	8400	Load start address, utility pointer
201-202	00C9-00CA	0	0000	Load end address
203-204	00CB-00CC	0	00	Tape timing constants
205	00CD	0	00	Flag for quote mode 0=not quote mode
206	00CE	0	00	Flag for tape read timer enable 0=disabled
207	00CF	0	00	Flag for EOT received from tape
208	00D0	0	00	Read character error
209	00D1	0	00	No. of characters in current file name
210	00D2	4	04	Current logical file number
211	00D3	255	FF	Current secondary address
212	00D4	4	04	Current device number
213	00D5	39	27	Current screen line length (39, 79)
214-215	00D6-00D7	0	0000	Pointer to start of current tape buffer (634 or 826)

Table F-2. CBM Memory Map (Rev. 3 ROMs) (Continued)

Memory Address		Sample Value		Description
Decimal	Hexadecimal	Decimal	Hexadecimal	
216	00D8	24	18	Line number where cursor is flashing (0-24)
217	00D9	10	0A	I/O storage: last key input, buffer checksum, bit buffer
218-219	00DA-00DB	0	0000	Pointer to current file name
220	00DC	0	00	Number of Insert keys pushed to go
221	00DD	0	00	Serial bit shift word
222	00DE	0	00	Number of blocks remaining to read/write
223	00DF	0	00	Serial word buffer
224-248	00E0-00F8			High byte of screen line addresses
		128	80	224-230=128 (lines 1-7)
		129	81	231-236=129 (lines 8-13)
		130	82	237-243=130 (lines 14-20)
		131	83	244-248=131 (lines 21-25)
249	00F9	0	00	Cassette #1 status switch
250	00FA	0	00	Cassette #2 status switch
251-252	00FB-00FC	54144	D380	Tape start address
253-255	00FD-00FF	243	F3	Utility
				Page 1 (256-511)
256-up	0100-up	32	20	Tape read working storage (up to 511) and conversion storage 256-318 For error correction in tape reads (62 bytes) 256-266 Binary to ASCII conversion (11 bytes)
511-down	01FF-down	44	2C	Stack (down to 256)
				Page 2-3 (512-1023)
512-592	0200-0250			BASIC input line buffer (80 bytes)
		12597	3135	512-513 0200-0201 Program Counter
		50	32	514 0202 Processor status
		0	00	515 0203 Accumulator
		171	AB	516 0204 X index
		0	00	517 0205 Y index
		0	00	518 0206 Stack pointer
		15104	3B00	519-520 0207-0208 User modifiable IRQ
593-602	0251-025A	4	04	Table of logical numbers of open files
603-612	025B-0264	4	04	Table of device numbers of open files
613-622	0265-026E	255	FF	Table of secondary address modes of open files
623-632	026F-0278	3	03	Keyboard buffer (10 bytes)
633	0279	28	1C	Keyboard utility
634-825	027A-0339	28	1C	Tape buffer for cassette #1 (192 bytes)
826-1017	033A-03F9	173	AD	Tape buffer for cassette #2 (192 bytes)
1018-1019	03FA-03FB	59383	E7F7	Vector for Machine Language Monitor
1020-1023	03FC-03FF	195	C3	Utility space/unused

Table F-2. CBM Memory Map (Rev. 3 ROMs) (Continued)

Memory Address		Sample Value		Description
Decimal	Hexadecimal	Decimal	Hexadecimal	
				OS Page Zero Storage (Continued)
			Page 4-128 (1024-32767)	
1024-32767	0400-7FFF	0	00	User program area and Expansion RAM 4K PET: 1024-4095 0400-0FFF User program area 4096-32767 1000-7FFF Expansion RAM 8K PET: 1024-8191 0400-1FFF User program area 8192-32767 2000-7FFF Expansion RAM 16K PET: 1024-16383 0400-3FFF User program area 16384-32767 4000-7FFF Expansion RAM 32K PET: 1024-32767 0400-7FFF User program area
			Page 129-144 (32768-36863)	
32768-36863	8000-8FFF	32	20	TV RAM 32768-33767 Display memory (1000 bytes)
			Page 145-192 (36864-49151)	
36864-49151	9000-BFFF	144	90	Expansion ROM
			Page 193-232 BASIC (49152-59391)	
				Pointers to BASIC Routines
49152-49153	C000-C001	51008	C740	Pointer -1 to END*
49154-49155	C002-C003	50775	C657	Pointer -1 to FOR
49156-49157	C004-C005	52255	CC1F	Pointer -1 to NEXT
49158-49159	C006-C007	51199	C7FF	Pointer -1 to DATA
49160-49161	C008-C009	51878	CAA6	Pointer -1 to INPUT#
49162-49163	C00A-C00B	51904	CAC0	Pointer -1 to INPUT
49164-49165	C00C-C00D	53090	CF62	Pointer -1 to DIM
49166-49167	C00E-C00F	51974	CB06	Pointer -1 to READ
49168-49169	C010-C011	51372	C8AC	Pointer -1 to LET
49170-49171	C012-C013	51116	C7AC	Pointer -1 to GOTO
49172-49173	C014-C015	51076	C784	Pointer -1 to RUN
49174-49175	C016-C017	51247	C82F	Pointer -1 to IF
49176-49177	C018-C019	50991	C72F	Pointer -1 to RESTORE
49178-49179	C01A-C01B	51087	C78F	Pointer -1 to GOSUB
49180-49181	C01C-C01D	51161	C7D9	Pointer -1 to RETURN
49182-49183	C01E-C01F	51266	C842	Pointer -1 to REM
49184-49185	C020-C021	51006	C73E	Pointer -1 to STOP
49186-49187	C022-C023	51282	C852	Pointer -1 to ON
49188-49189	C024-C025	55055	D70F	Pointer -1 to WAIT
49190-49191	C026-C027	65492	FFD4	Pointer -1 to LOAD
49192-49193	C028-C029	65495	FFD7	Pointer -1 to SAVE
49194-49195	C02A-C02B	65498	FFDA	Pointer -1 to VERIFY
49196-49197	C02C-C02D	53900	D28C	Pointer -1 to DEF
49198-49199	C02E-C02F	55046	D706	Pointer -1 to POKE
49200-49201	C030-C031	51594	C98A	Pointer -1 to PRINT#

* These memory locations contain the address of the byte preceding the specified BASIC routines

Table F-2. CBM Memory Map (Rev. 3 ROMs) (Continued)

Memory Address		Sample Value		Description
Decimal	**Hexadecimal**	**Decimal**	**Hexadecimal**	
				Pointers to BASIC Routines (Continued)
49202-49203	C032-C033	51626	C9AA	Pointer −1 to PRINT
49204-49205	C034-C035	51050	C76A	Pointer −1 to CONT
49206-49207	C036-C037	50612	C5B4	Pointer −1 to LIST
49208-49209	C038-C039	50550	C576	Pointer −1 to CLR
49210-49211	C03A-C03B	51600	C990	Pointer −1 to CMD
49212-49213	C03C-C03D	65501	FFDD	Pointer −1 to SYS
49214-49215	C03E-C03F	65471	FFBF	Pointer −1 to OPEN
49216-49217	C040-C041	65474	FFC2	Pointer −1 to CLOSE
49218-49219	C042-C043	51836	CA7C	Pointer −1 to GET
49220-49221	C044-C045	50522	C55A	Pointer −1 to NEW
49222-49223	C046-C047	56133	DB45	Pointer to SGN **
49224-49225	C048-C049	56280	DBD8	Pointer to INT
49226-49227	C04A-C04B	56164	DB64	Pointer to ABS
49228-49229	C04C-C04D	0	0000	Pointer to USR pointer
49230-49231	C04E-C04F	53849	D259	Pointer to FRE
49232-49233	C050-C051	53882	D27A	Pointer to POS
49234-49235	C052-C053	56926	DE5E	Pointer to SQR
49236-49237	C054-C055	57215	DF7F	Pointer to RND
49238-49239	C056-C057	55542	D8F6	Pointer to LOG
49240-49241	C058-C059	57050	DEDA	Pointer to EXP
49242-49243	C05A-C05B	57304	DFD8	Pointer to COS
49244-49245	C05C-C05D	57311	DFDF	Pointer to SIN
49246-49247	C05E-C05F	57384	E028	Pointer to TAN
49248-49249	C060-C061	57484	E08C	Pointer to ATN
49250-49251	C062-C063	55016	D6E8	Pointer to PEEK
49252-49253	C064-C065	54870	D656	Pointer to LEN
49254-49255	C066-C067	54079	D33F	Pointer to STR$
49256-49257	C068-C069	54919	D687	Pointer to VAL
49258-49259	C06A-C06B	54885	D664	Pointer to ASC
49260-49261	C06C-C06D	54726	D5C6	Pointer to CHR$
49262-49263	C06E-C06F	54746	D5DA	Pointer to LEFT$
49264-49265	C070-C071	54790	D606	Pointer to RIGHT$
49266-49267	C072-C073	54801	D611	Pointer to MID$
49268-49297	C074-C091			Hierarchy and action addresses for operators
49298-49553	C092-C191			Table of BASIC keywords
49554-49833	C192-C2A9			BASIC error messages
				BASIC Routines
				Starting Address **Function**
49834-59343	C2AA-DFFF			49834 C2AA FOR . . NEXT stack check
				49880 C2D8 Insert line space marker
				49947 C31B Stack overflow check
				49960 C328 Error message abort
				50057 C389 READY
				50091 C3AB Handle new line

** These memory locations contain the address of the first byte of the specified BASIC routines

Table F-2. CBM Memory Map (Rev. 3 ROMs) (Continued)

Memory Address		Sample Value		Description
Decimal	Hexadecimal	Decimal	Hexadecimal	
				BASIC Routines (Continued) **Starting Address — Function** 50242 C442 Rechain lines after insert/delete 50287 C46F Input line 50325 C495 Keyword encoder 50476 C52C Line number search 50523 C55B NEW 50551 C577 CLR 50599 C5A7 Set pointer to start of program 50613 C5B5 LIST 50776 C658 FOR 50944 C700 Statement execute 50992 C730 RESTORE 51007 C73F STOP 51009 C741 END 51051 C76B CONT 51077 C785 RUN 51088 C790 GOSUB 51117 C7AD GOTO 51162 C7DA RETURN 51200 C800 DATA 51214 C80E Scan for next BASIC statement 51217 C811 Scan for next BASIC line 51248 C830 IF 51267 C843 REM 51283 C853 ON 51315 C873 Number fetch 51373 C8AD LET = 51496 C928 Add ASCII digit to Accumulator #1 51595 C98B PRINT# 51601 C991 CMD 51627 C9AB PRINT 51740 CA1C Print string 51769 CA39 Print character 51791 CA4F Input data error 51837 CA7D GET 51879 CAA7 INPUT# 51962 CAFA Input prompt 51975 CB07 READ 52220 CBFC Error messages 52256 CC20 NEXT 52345 CC79 Format checker 52383 CC9F Expression evaluator 53091 CF63 DIM 53101 CF6D Variable table lookup 53249 D001 Create new variable 53420 D0AC Array table search/ create array

Table F-2. CBM Memory Map (Rev. 3 ROMs) (Continued)

Memory Address		Sample Value		Description
Decimal	Hexadecimal	Decimal	Hexadecimal	
				BASIC Routines (Continued)

Starting Address		Function
53849	D259	FRE
53869	D26D	Integer-to-floating
53882	D27A	POS
53888	D280	Valid direct check
53901	D28D	DEF
54079	D33F	STR$
54726	D5C6	CHR$
54746	D5DA	LEFT$
54790	D606	RIGHT$
54801	D611	MID$
54870	D656	LEN
54885	D665	ASC
54919	D687	VAL
54994	D6D2	Floating-to-integer
55016	D6E8	PEEK
55047	D707	POKE
55056	D710	WAIT
55091	D733	Subtraction
55150	D76E	Addition
55542	D8F6	LOG
55607	D937	Multiplication
55704	D998	Load number to AFAC
55818	DA0A	Division
55982	DAAE	Load Accumulator (FAC)
56030	DADE	Store FAC
56072	DB08	Copy AFAC to FAC
56088	DB18	Copy FAC to AFAC
56133	DB45	SGN
56164	DB64	ABS
56280	DBD8	INT
56526	DCCE	IN line message
56553	DCE9	Numeric-to-ASCII
56319	DBFF	String-to-floating
56926	DE5E	SQR
56936	DE68	Power function
57050	DEDA	EXP
57215	DF7F	RND
57304	DFD8	COS
57311	DFDF	SIN

Table F-2. CBM Memory Map (Rev. 3 ROMs) (Continued)

Memory Address		Sample Value		Description
Decimal	Hexadecimal	Decimal	Hexadecimal	
				Screen Editor **Starting Address Function**
57344-5391	E000-E7FF			57384 E028 TAN 57484 E08C ATN 57593 E0F9 Subroutine to be moved to page 0 ($70-$87) 57617 E111 Initial RND seed (5 bytes) 57622 E116 Initialize BASIC system 57897 E229 Clear screen 57943 E257 Home cursor 57989 E285 Character fetch
58100-58906	E2F4-E61A			Video driver 58100 E2F4 Input from screen 58175 E33F Quote mode ($CD) switcher 58188 E34C Print character 58687 E53F Scroll 1 line
58907-59113	E61B-E6E9			Interrupt Handler
59114-59127	E6EA-E6F7			Keyboard Scan
59128-59241	E6F8-E769			Keyboard Encoding Table
59242-59391	E76A-E7FF			Subroutines for Machine Language Monitor
	Page 233-240 I/O Ports and Expansion I/O (PIA's and VIA) (59392-61439)			
				Keyboard PIA (59408-59411)
59408	E810	249	F9	I/O Port A and Data Direction register
59409	E811	60	3C	Control Register A — screen blanking 52=Screen off (blanked) 60=Screen on
59410	E812	255	FF	I/O Port B and Data Direction register 255=all keys except: 254=RVS key 253=| key 251=SPACE key 247=< key
59411	E813	61	3D	Control Registers B — #1 cassette motor 53=motor on 61=motor off
				IEEE Port PIA (59424-59427)
59424	E820	255	FF	I/O Port A and Data Direction register PEEK (59424) reads input data
59425	E821	188	BC	Control Register A — set output line CA2 POKE 59425,52=low POKE 59425,60=high
59426	E822	255	FF	I/O Port B and Data Direction registers POKE 59426, data writes output data POKE 59426,255 before a read to Port A
59427	E823	60	3C	Control Register B — set output line CB2 POKE 59427,52=low POKE 59427,60=high

Table F-2. CBM Memory Map (Rev. 3 ROMs) (Continued)

Memory Address		Sample Value		Description
Decimal	Hexadecimal	Decimal	Hexadecimal	
				Parallel User Port VIA (59456-59471)
59456	E840	223	DF	I/O Port B 207=#2 cassette motor on 223=#2 cassette motor off WAIT 59456,23,23 waits for vertical retrace of display Bit 1=PB1 (NFRD on IEEE connector) output line Bit 3=PB3 (ATN on IEEE connector) output line
59457	E841	255	FF	I/O Port A with handshaking
59458	E842	30	1E	Data Direction register for I/O Port B
59459	E843	0	00	Data Direction register for I/O Port A For each bit 1=output, 0=input =0 all input =255 all output
59460-59461	E844-E845	29241	7239	(Low, high order) Read Timer 1, Counter; write to Timer 1 Latch and (high byte) initiate count
59462-59463	E846-E847	65535	FFFF	(Low, high order) Read Timer 1 Latch
59464	E848	147	93	Read Timer 2 Counter low byte and reset interrupt; write to Timer 2 low byte PEEK (59464) Clock decrements every microsecond POKE 59454,n sets SR rate of shift from high (n=0) to low (n=255) for music from User Port
59465	E849	217	D9	Read Timer 2 Counter high byte; write to Timer 2 high byte and reset interrupt PEEK (59465) Clock decrements every millisecond
59466	E84A	0	00	Serial I/O Shift register (SR) POKE 59466, 15 or 85 to generate Square wave output at CB2 for playing music from User Port.
59467	E84B	0	00	Auxiliary Control register =16 Sets SR to free-running mode for music from User Port =0 for proper operation of tape drive
59468	E84C	14	0E	Peripheral Control register =12 for graphics on shifted characters =14 for lower-case letters on shifted characters
59469	E84D	0	00	Interrupt Flag register
59470	E84E	128	80	Interrupt enable register
59471	E84F	255	FF	I/O Port A without handshaking
		Page 241-256 Operating System (61440-65535)		
61440-61621	F000-F0B5			Monitor messages

Table F-2. CBM Memory Map (Rev. 3 ROMs) (Continued)

Memory Address		Sample Value		Description
Decimal	Hexadecimal	Decimal	Hexadecimal	
61622-61904	F0B6-F1D0			**GPIB Handler (IEEE 488 Bus)** **Starting Address Function** 61622 F0B6 Setup for Listen, Talk, etc. 61678 F0EE Send character 61736 F128 Output character immediate mode 61750 F136 Error messages 61796 F164 Send immediate Listen command, then secondary address 61807 F16F Output characters 61823 F17F Send Unlisten/ Untalk 61836 F18C Input character
61905-63493	F1D1-F805			**File Control** 61905 F1D1 Get a character (without cursor) 61921 F1E1 Input a character (with cursor) 62002 F232 Output a character to any device 62062 F26E Close all files 62066 F272 Restore default I/O devices 62121 F2A9 CLOSE 62209 F301 STOP search 62223 F30F STOP key 62229 F315 Direct mode test 62402 F3C2 LOAD 62474 F40A Display filename/ fetch file number 62526 F43E Fetch LOAD/SAVE parameters 62560 F460 Fetch byte paramter 62566 F466 Send program name to GPIB 62612 F494 Tape header search 62647 F4B7 VERIFY 62670 F4CE Fetch OPEN/CLOSE parameters 62753 F521 OPEN 62886 F5A6 Find any tape header 62938 F5DA Write tape header 63036 F63C Process tape header 63108 F684 SYS 63134 F69E SAVE 63273 F729 Clock update 63344 F770 Set input device 63420 F7BC Set output device

Table F-2. CBM Memory Map (Rev. 3 ROMs) (Continued)

Memory Address		Sample Value		Description
Decimal	Hexadecimal	Decimal	Hexadecimal	
				Tape Control
63494-64720	F806-FCD0			63494 F806 Advance tape buffer pointer 63541 F835 Gheck for cassette on 63573 F855 Tape read to buffer 63622 F886 Write block to tape 63716 F8E6 Interrupt wait
				Power-on Diagnostics
64721-64784	FCD1-FD10			64721 FCD1 System reset SYS(64721) simulates power-on reset 64766 FCFE NMI interrupt entry point
64785-65471	FD11-FFBF			64769 FD01 Table of interrupt vectors
				Machine Language Monitor
				Jump Vectors
65472-65474	FFC0-FFC2	76 62753	4C F521	JMP OPEN
65475-65477	FFC3-FFC5	76 62121	4C F2A9	JMP CLOSE
65478-65480	FFC6-FFC8	76 63344	4C F770	JMP Set Input Device
65481-65483	FFC9-FFCB	76 63420	4C F7BC	JMP Set Output Device
65484-65486	FFCC-FFCE	76 62066	4C F272	JMP Restore Default I/O Devices
65487-65489	FFCF-FFD1	76 61921	4C F1E1	JMP Input Character — RDT
65490-65492	FFD2-FFD4	76 62002	4C F232	JMP Output Character — WRT
65493-65495	FFD5-FFD7	76 62402	4C F3C2	JMP LOAD
65496-65498	FFD8-FFDA	76 63134	4C F69E	JMP SAVE
65499-65501	FFDB-FFDD	76 62647	4C F4B7	JMP VERIFY
65502-65504	FFDE-FFED	76 63108	4C F684	JMP SYS
65505-65507	FFE1-FFE3	76 62223	4C F30F	JMP Test STOP Key
65508-65510	FFE4-FFE6	76 61905	4C F1D1	JMP Get Character
65511-65513	FFE7-FFE9	76 62062	4C F26E	JMP Close all files
65514-65516	FFEA-FFEC	76 63273	4C F729	JMP Clock Update
				6502 Interrupt Vectors
65530-65531	FFFA-FFFB	65766	FCFE	Non-maskable interrupt (NMI)
65532-65533	FFFC-FFFD	64721	FCD1	System reset (RESET)
65534-65535	FFFE-FFFF	58907	E61B	Interrupt request break (IRQ+BRK)

Table F-3. Hex Addresses and Label References: CBM BASICs

BASIC 3.0	BASIC 4.0	Labels	Description
0000	0000	USRPOK	$4C CONSTANT AND ADDRESS TO DISPATCH USR
0000	0000	ERRNF	ERROR CALL VALUE - ECV - NEXT WITHOUT FOR
0001	0001	ADDPRC	X
0002	0002	BUFPAG	INPUT BUFFER AT $0200
0002	0002	ADDPR2	X
0003	0003	STRSIZ	NUMBER OF LOCS PER STRING DESCRIPTOR
0003	0003	INTEGR	ONE-BYTE INTEGER FROM "@INT"
0003	0003	CHARAC	STARTING DELIMITER
0004	0004	ENDCHR	ENDING DELIMITER
0004	0004	ADDPR4	X
0005	0005	COUNT	GENERAL COUNTER FOR BASIC
0006	0006	DIMFLG	FLAG TO REMEMBER DIMENSIONED VARIABLES
0007	0007	VALTYP	FLAG FOR VARIABLE TYPE 0-NUMERIC $FF-STRING
0008	0008	INTFLG	FLAG FOR INTEGER TYPE
0008	0008	ADDPR8	X
0009	0009	GARBFL	X
0009	0009	DORES	FLAG WHETHER CAN OR CAN'T CRUNCH RESERVED WORDS
000A	000A	CLMWID	SIZE OF PRINT WINDOW
000A	000A	SUBFLG	FLAG WHICH ALLOWS SUBSCRIPTS IN SYNTAX
000B	000B	INPFLG	FLAGS INPUT OR READ
000C	000C	DOMASK	MASK USED BY RELATION OPERATIONS
000C	000C	TANSGN	FLAG SIGN OF TANGENT
000D	000D	DSDESC	DS$ LENGTH AND POINTER TO DS$
000E	0010	CHANNL	ACTIVE I/O CHANNEL #
0010	0010	ERRSN	ERROR CALL VALUE - ECV - SYNTAX
0011	0011	POKER	HOLDS ADDRESS FORE POKE COMMAND
0011	0011	LINNUM	LINE NUMBER STORAGE
0012	0012	FORSIZ	AMOUNT OF BYTES USED ON STACK FOR-NEXT
0013	0013	TEMPPT	INDEX TO NEXT AVAILABLE DESCRIPTOR
0014	0014	LASTPT	POINTER TO LAST STRING TEMP LO;HI
0016	0016	TEMPST	STORAGE FOR NUMTMP TEMP DESCRIPTORS
0016	0016	ERRRG	ECV - RETURN WITHOUT GOSUB
0017	0017	NUMLEV	NUMBER OF GOSUB LEVELS ALLOWED
001E	001E	NCMPOS	X
001F	001F	INDEX	INDRIECT INDEX #1
001F	001F	INDEX1	SAME
0021	0021	INDEX2	INDIRECT INDEX #2
0023	0023	RESHO	RES -REGISTER
0024	0024	RESMOH	[
0025	0025	ADDEND	TEMP USED BY "UMULT"
0025	0025	RESMO	[
0026	0026	RESLO	[
0028	0028	LINLEN	LENGTH OF SCREEN LINE 40-COL EDITORS
0028	0028	TXTTAB	POINTER TO START OF BASIC TEXT AREA
002A	002A	VARTAB	POINTER TO START OF VARIABLES
002A	002A	ERROD	ECV - OUT OF DATA
002C	002C	ARYTAB	POINTER TO START OF ARRAY TABLE
002E	002E	STREND	POINTER TO END OF VARIABLES
0030	0030	FRETOP	POINTER TO START OF REAL STRINGS
0032	0032	FRESPC	POINTER TO TOP OF FREE STRING SPACE
0034	0034	MEMSIZ	HIGHEST RAM ADDR AVALIBLE FOR BASIC
0035	0035	ERRFC	ECV - ILLEGAL QUANTITY
0036	0036	CURLIN	CURRENT LINE BEING EXECUTED
0038	0038	OLDLIN	LAST LINE EXECUTED (FOR CONT COMMAND)
003A	003A	OLDTXT	OLD TXTPTR (FOR CONT COMMAND) AND TEMP STORAGE
003C	003C	DATLIN	DATA LINE # FOR ERRORS

Table F-3. Hex Addresses and Label References: CBM BASICs (Continued)

BASIC 3.0	BASIC 4.0	Labels	Description
003E	003E	DATPTR	DATA STATEMENT POINTER
0040	0040	INPPTR	SOURCE OF INPUT ADDRESS
0042	0042	VARNAM	CURRENT VARIABLE NAME
0044	0044	FDECPT	POINTER INTO POWERS OF TEN FOR FOUT
0044	0044	VARPNT	POINTER TO VARIABLE IN MEMORY
0045	0045	ERROV	ECV - OVERFLOW
0046	0046	LSTPNT	PNTR TO LIST STRING
0046	0046	ANDMSK	THEN MASK USED BY WAIT FOR ANDING
0046	0046	FORPNT	POINTER TO CURRENT FOR-NEXT VARIABLE REFERENCE
0047	0047	EORMSK	THE MASK FOR EORING IN WAIT
0048	0048	VARTXT	POINTER INTO LIST OF VARIABLES
0048	0048	OPPTR	POINTER TO CURRENT OPERATOR IN TABLE
004A	004A	OPMASK	MASK CREATED BY CURRENT OPERATOR
004B	004B	GRBPNT	POINTER USED IN GARBAGE COLLECTION
004B	004B	TEMPF3	A THIRD FAC TEMPORARY 4-BYTES
004B	004B	DEFPNT	POINTER USED IN FUNCTION DEFINITION
004D	004D	DSCPNT	POINTER TO A STRING DESCRIPTION
004D	004D	ERROM	ECV - OUT OF MEMORY
0050	0050	FOUR6	VARIABLE CONSTANT USED BY GARB COLLECT
0051	0051	BUFLEN	INPUT BUFFER MAX SIZE+1
0051	0051	JMPER	$4C CONSTANT AND ADDRESS USED TO DISPATCH FUNCS
0052	0052	SIZE	X
0053	0053	OLDOV	THE OLD OVERFLOW
0054	0054	TEMPF1	A FAC TEMP 4-BYTES
0055	0055	ARYPNT	A POINTER USED IN ARRAY BUILDING
0055	0055	HIGHDS	DESTINATION OF HIGHEST ELMENT IN BLT.
0057	0057	HIGHTR	SOURCE OF HIGHEST ELEMENT TO MOVE
0059	0059	TEMPF2	A FAC TEMP 4-BYTES
005A	005A	DECCNT	NUMBER OF PLACES BEFORE DECIMAL POINT
005A	005A	LOWDS	LOCATION OF LAST BYTE TRANSFERRED INTO
005A	005A	ERRUS	ECV - UNDEF'D STATEMENT
005B	005B	TENEXP	BASE TEN EXPONENT FOR FIN AND FOUT
005C	005C	GRBTOP	A POINTER USED IN GARBAGE COLLECTION
005C	005C	DPTFLG	FLAG IF A DECIMAL POINT HAS BEEN INPUT
005C	005C	LOWTR	LAST THING TO MOVE IN BLT.
005D	005D	EXPSGN	SIGN OF BASE TEN EXPONENT
005D	005D	EPSGN	X
005E	005E	DSCTMP	THIS IS WHERE TEMP DESCS ARE BUILT
005E	005E	FAC	THE MAIN FLOATING POINT ACCUMULATOR
005E	005E	FACEXP	THE EXPONENT BYTE
005F	005F	FACHO	[MOST SIGNIFICANT BYTE OF MANTISSA
0060	0060	FACMOH	[ONE MORE
0061	0061	INDICE	INDICE IS SET UP HERE BY "@INT"
0061	0061	FACMO	[MIDDLE ORDER OF MANTISSA
0062	0062	FACLO	[LEAST SIG BYTE OF MANTISSA
0063	0063	FACSGN	SIGN OF FAC (0 OR -1) WHEN UNPACKED
0064	0064	DEGREE	A CONT USED BY POLYNOMIALS
0064	0064	SGNFLG	SIGN OF FAC IS PRESERVED HERE BY FIN
0065	0065	BITS	COUNTER FOR # OF BIT SHIFTS TO NORMALIZE FAC
0066	0066	ARGEXP	THE ARG REGISTER EXPONENT
0067	0067	ARGHO	[
0068	0068	ARGMOH	[
0069	0069	ARGMO	[
006A	006A	ARGLO	[
006B	006B	ARGSGN	THE SIGN (SAME AS FAC)
006B	006B	ERRBS	ECV - BAD SUBSCRIPT

Table F-3. Hex Addresses and Label References: CBM BASICs (Continued)

BASIC 3.0	BASIC 4.0	Labels	Description
006C	006C	STRNG1	POINTER TO A STRING OR DESCRIPTOR
006C	006C	ARISGN	A SIGN REFLECTING THE RESULT
006D	006D	FACOV	OVERFLOW BYTE OF THE FAC
006E	006E	BUFPTR	POINTER TO BUF USED BY "CRUNCH ROUTINE"
006E	006E	STRNG2	POINTER TO STRING OR DESC.
006E	006E	POLYPT	POINTER INTO POLYNOMIAL COEFFICIENTS.
006E	006E	CURTOL	ABSOLUTE LINEAR INDEX IS FORMED HERE
006E	006E	FBUFPT	POINTER INTO FBUFFER USED IN FOUT.
0070	0070	CHRGET	ROUTINE - GETS NEXT CHARACTER FROM BASIC TEXT
0076	0076	CHRGOT	ROUTINE -REGETS CURRENT CHARACTER FROM BASIC TEXT
0077	0077	TXTPTR	POINTER TO CURRENT SOURCE TEXT
0078	0078	ERRDD	ECV - REDIM'D ARRAY
007D	007D	ONUM	LABEL IN CHRGET
0080	0080	ENDTK	TOKEN - END
0081	0081	FORTK	TOKEN - FOR
0083	0083	DATATK	TOKEN - DATA
0085	0085	ERRDVO	ECV - DIVISION BY ZERO
0087	0087	CHRRTS	LABEL IN CHRGET
0088	0088	RNDX	NEXT RANDOM NUMBER - INITIAL LOAD FROM ROM
0089	0089	GOTOTK	TOKEN - GOTO
008B	008B	ZZ7	X
008D	008D	CTIMR	24 HR CLOCK 1/60 OF SEC
008D	008D	GOSUTK	TOKEN - GOSUB
008F	008F	REMTK	TOKEN - REM
0095	0095	ERRID	ECV - ILLEGAL DIRECT
0096	0096	CSTAT	I/O OPERATION STATUS BYTE (VARIABLE ST)
0099	0099	PRINTK	TOKEN - PRINT
00A2	00A2	SCRATK	TOKEN - NEW
00A3	00A3	TABTK	TOKEN - TAB
00A3	00A3	ERRTM	ECV - TYPE MISMATCH
00A4	00A4	TOTK	TOKEN - TO
00A5	00A5	FNTK	TOKEN - FN
00A6	00A6	SPCTK	TOKEN - SPC
00A7	00A7	THENTK	TOKEN - THEN
00A8	00A8	NOTTK	TOKEN - NOT
00A9	00A9	STEPTK	TOKEN - STEP
00AA	00AA	PLUSTK	TOKEN - +
00AB	00AB	MINUTK	TOKEN - -
00B0	00B0	ERRLS	ECV - STRING TO LONG
00B1	00B1	GREATK	TOKEN - >
00B2	00B2	EQULTK	TOKEN - =
00B3	00B3	LESSTK	TOKEN - <
00B4	00B4	ONEFUN	TOKEN - SGN START OF SINGLE PARM FUNCTIONS
00BF	00BF	ERRBD	ECV - FILE DATA
00C6	00C6	TRMPOS	X
00C7	00C7	LASNUM	TOKEN - CHR$ LAST FUNC WITH ARITHMETIC PARMS
00C8	00C8	ERRST	ECV - FORMULA TOO COMPLEX
00CB	00CB	GOTK	TOKEN - GO (GO TO)
00DB	00DB	ERRCN	ECV - CAN'T CONTINUE
00E9	00E9	ERRUF	ECV - UNDEF'D FUNCTION
00FF	00FF	PI	VALUE OF PI SYMBOL
00FF	00FF	LOFBUF	START OF FOUT STRING FOR STRD AND TI$
0100	0100	FBUFFR	FOUT BUFFER HOLDS ASCII STRING FOR OUTPUT
01FB	01FB	STKEND	TOP OF STACK FOR BASIC
01FF	01FF	ZZ1	X
01FF	01FF	ZZ5	X

Table F-3. Hex Addresses and Label References: CBM BASICs (Continued)

BASIC 3.0	BASIC 4.0	Labels	Description
01FF	01FF	ZZ4	X
0200	0200	BUF	BASIC INPUT BUFFER (80 CHARACTERS-BYTES LONG)
0200	0200	BUFOFS	SAME AS ABOVE
0201	0201	ZZ2	X
0202	0202	ZZ3	X
0400	0400	RAMLOC	BEGINING OF RAM AVALIABLE FOR BASIC TEXT
0000	8000	OFFSET	*VALUE USED IN ASSEMBLY - ROM VERSION
0000	8B88	ZZ8	X
C000	B000	ROMLOC	BEGINING OF BASIC ROMS -V2=$C000 V4=$B000
C000	B000	STMDSP	START OF COMMAND DISPATCH TABLE
C046	B066	FUNDSP	START OF FUNCTION DISPATCH TABLE
C04C	B06C	USRLOC	X
C074	B094	OPTAB	START OF MATH OPERATORS DISPATCH TABLE
C089	B0A9	NEGTAB	UNITARY NEGATE DISPATCH (.BYTE 125,DISPATCH)
C08C	B0AC	NOTTAB	NOT OPERATOR DISPATCH (.BYTE 90,DISPATCH)
C08F	B0AF	PTDORL	COMPARISON DISPATCH (.BYTE 100,DISPATCH)
C092	B0B2	RESLST	START OF RESERVED WORD LIST (ASCII,END(OR $80))
C192	B20D	ERRTAB	START OF BASIC ERROR MESSAGE STORAGE
C28B	B306	ERR	MESSAGE - "ERROR"
C292	B30D	INTXT	MESSAGE - "IN"
C297	B312	REDDY	MESSAGE - "READY"
C2A2	B31B	BRKTXT	MESSAGE - "BREAK"
C2AA	B322	FNDFOR	PEEKS AT THE STACK FOR AN ACRTIVE "FOR" LOOP
C2AF	B327	FFLOOP	X
C2C4	B33C	CMPFOR	X
C2D0	B348	ADDFRS	X
C2D7	B34F	FFRTS	X
C2D8	B350	BLTU	"OPENS UP" A SPACE IN BASIC FOR A NEW LINE
C2DF	B357	BLTUC	X
C2FC	B374	BLT1	X
C308	B380	BLTLP	X
C30C	B384	MOREN1	X
C313	B38B	DECBLT	X
C31B	B393	GETSTK	TEST FOR STACK-TOO-DEEP ERROR
C328	B3A0	REASON	CHECKS FOR AVALIABLE MEMORY SPACE
C332	B3AA	TRYMOR	X
C336	B3AE	REASAV	X
C341	B3B9	REASTO	X
C354	B3CC	REARTS	X
C355	B3CD	OMERR	OUT OF MEMORY ERROR VECTOR
C357	B3CF	ERROR	ERROR HANDLER (ERROR TYPE IN .X)
C364	B3DA	ERRCRD	X
C36A	B3E0	GETERR	X
0000	B3ED	TYPERR	PRINTS OUT THE ERROR MESSAGE
C37E	B3F4	ERRFIN	X
C389	B3FF	READY	PRINTS "READY." GOES INTO MAIN BASIC LOOP (← NMI)
C392	B406	MAIN	MAIN BASIC LOOP, ANALYZES INPUT LINES
C3AB	B41F	MAIN1	LINES THAT START WITH A NUMBER HANDLED HERE
C3E6	B45A	QDECT1	X
C3EE	B462	MLOOP	X
C3FC	B470	NODEL	X
C417	B48B	NODELC	X
C431	B4A5	STOLOP	X
C439	B4AD	FINI	CLEANS BASIC SYSTEM UP; CLR
C442	B4B6	LNKPRG	RELINKS BASIC STATEMENTS IN TEXT AREA
C44B	B4BF	CHEAD	X

Table F-3. Hex Addresses and Label References: CBM BASICs (Continued)

BASIC 3.0	BASIC 4.0	Labels	Description
C453	B4C7	CZLOOP	X
C46E	B4E1	LNKRTS	X
C46F	B4E2	INLIN	INPUT A LINE OF INFORMATION INTO BUF (MAX 80 CHARS)
C471	B4E4	INLINC	X
C47E	B4F8	FININ1	X
C495	B4FB	CRUNCH	LOOKS UP KEYWORDS IN AN INPUT LINE
C49B	B501	KLOOP	X
C4A7	B50D	CMPSPC	X
C4BD	B523	KLOOP1	X
C4C5	B52B	MUSTCR	X
C4CF	B53D	RESER	X
C4D1	B544	RESCON	X
C4E0	B552	GETBPT	X
C4E2	B554	STUFFH	X
C4F5	B567	COLIS	X
C4F7	B569	NODATT	X
C4FE	B570	STR1	X
C507	B579	STRNG	X
C50E	B580	NTHIS	X
C512	B584	NTHIS1	X
0000	B58D	NTHIS2	X
C522	B599	CRDONE	X
C52C	B5A3	FNDLIN	SEARCHES FOR A LINE NUMBER (NUMBER IN LINNUM)
C530	B5A7	FNDLNC	X
C547	B5BE	FNDLO1	X
C550	B5C7	AFFRTS	X
C559	B5D0	FLINRT	X
C55A	B5D1	FLNRTS	X
C55B	B5D2	SCRATH	IMPLEMENTS "NEW" COMMAND - CLEARS EVERY THING
C55D	B5D4	SCRTCH	X
C572	B5E9	RUNC	X
C577	B5EE	CLEAR	CLR - ROUTINE
C579	B5F0	CLEARC	X
0000	B60B	FLOAD	X
C593	B60E	STKINI	X
C5A6	B621	STKRTS	X
C5A7	B622	STXTPT	TXTPTR=TXTTAB-1
C5B5	B630	LIST	ROUTINE - LIST
C5BD	B638	GOLST	X
C5D4	B64F	LSTEND	X
C5E2	B65D	LIST4	X
C5FF	B67A	TSTDUN	X
C601	B67C	TYPLIN	X
C608	B683	PRIT4	X
C60C	B687	PLOOP	X
C619	B694	PLOOP1	X
C62D	B6A8	GRODY	X
C630	B6AB	QPLOP	X
C642	B6C5	RESRCH	X
C645	B6C8	RESCR1	X
0000	B6CE	RESCR2	X
C64D	B6D4	PRIT3	X
0000	B6D5	PRIT3B	X
C658	B6DE	FOR	ROUTINE - FOR
C669	B6EF	NOTOL	X
C6A1	B727	LDFONE	X

Table F-3. Hex Addresses and Label References: CBM BASICs (Continued)

BASIC 3.0	BASIC 4.0	Labels	Description
C6B5	B73B	ONEON	X
C6C4	B74A	NEWSTT	MAIN STATEMENT DISPATCH LOOP (DO NEXT STATEMENT)
C6D4	B759	DIRCON	X
C6E4	B769	DIRCN1	X
C6F7	B77C	GONE	DISPATCHES NEXT BYTE CHRGET RETURNS
C700	B785	GONE3	DISPATCHES .A IF NONZERO ELSE LOOP TO NEWSTT
C702	B787	GONE2	X
0000	B795	GONE4	X
C717	B7A2	GLET	X
C71A	B7A5	MORSTS	X
C71E	B7A9	SNERR1	SYNTAX ERROR VECTOR
0000	B7AC	GO	HANDLE GO TOKEN CASE (FIND A TO)
C730	B7B7	RESTOR	ROUTINE - RESTORE
C73A	B7C1	RESFIN	X
C73E	B7C5	ISCRTS	X
C73F	B7C6	STOP	STOP - SEC END - CLC
C741	B7C8	END	ROUTINE - END
C742	B7C9	STOPC	ROUTINE - STOP
C751	B7D8	STPEND	X
C759	B7E0	DIRIS	X
C75B	B7E2	ENDCON	X
C768	B7EB	GORDY	JMP READY
C76B	B7EE	CONT	ROUTINE - CONT
C784	B807	CONTRT	X
C785	B808	RUN	ROUTINE - RUN
C790	B813	GOSUB	ROUTINE - GOSUB
C7A4	B827	RUNC2	X
C7AD	B830	GOTO	ROUTINE - GOTO
C7C4	B847	LUK4IT	X
C7C8	B84B	LUKALL	X
C7D9	B85C	GORTS	X
C7DA	B85D	RETURN	ROUTINE - RETURN
C7EB	B86E	USERR	BAD SUBSCRIPT ERROR VECTOR
C7F0	B873	SNERR2	SYNTAX ERROR VECTORY
C7F3	B876	RETU1	X
C800	B883	DATA	X
C803	B886	ADDON	X
C80D	B890	REMRTS	X
C80E	B891	DATAN	SEARCH FOR NEXT '
C811	B894	REMN	LOOK FOR EOL($00) (TXTPTR OFFSET IN .Y)
C819	B89C	EXCHQT	X
C821	B8A4	REMER	X
C830	B8B3	IF	ROUTINE - IF
C83F	B8C2	OKGOTO	X
C843	B8C6	REM	ROUTINE - REM
C848	B8CB	DOCOND	X
C850	B8D3	DOCO	X
C853	B8D6	ONGOTO	ROUTINE - ON (GOTO OR GOSUB)
C85B	B8DE	SNERR3	SYNTAX ERROR VECTOR
C85F	B8E2	ONGLOP	X
C867	B8EA	ONGLP1	X
C872	B8F5	ONGRTS	X
C873	B8F6	LINGET	INPUT A BASIC LINE NUMBER (0-63999)(VALUE IN LINNUM)
C879	B8FC	MORLIN	X
C8A7	B92A	NXTLGC	X
C8AD	B930	LET	ROUTINE - LET

Table F-3. Hex Addresses and Label References: CBM BASICs (Continued)

BASIC 3.0	BASIC 4.0	Labels	Description
C8CA	B94D	QINTGR	X
C8DE	B961	COPFLT	X
C8E1	B964	COPSTR	X
C8E2	B965	INPCOM	X
C8F5	B978	TIMELP	X
C90F	B992	NOML6	X
C91F	B9A2	TIMEST	X
C928	B9AB	TIMNUM	X
C92F	B9B2	FCERR2	ILLEGAL QUANITY ERROR VECTOR
C932	B9B5	GOTNUM	X
C937	B9BA	GETSPT	COPY STRINGS IF NEEDED
0000	B9BE	DSKX0	X
0000	B9D2	DSKX1	X
0000	B9D4	DSKX2	X
C948	B9E1	QVARIA	X
C956	B9EF	DNTCPY	X
C95D	B9F6	COPY	X
C973	BA13	COPYC	X
0000	BA2E	COPY00	X
0000	BA44	COPY01	X
0000	BA46	COPY02	X
0000	BA4E	STRADJ	POINT TO STRING FOR A COPY
0000	BA6C	ADJ	X
0000	BA70	ADJXX	X
0000	BA74	ADJ02	X
0000	BA83	ADJ00	X
0000	BA85	ADJ01	X
C98B	BA88	PRINTN	ROUTINE - PRINT#
C991	BA8E	CMD	ROUTINE - CMD
C99B	BA98	SAVEIT	X
C9A5	BAA2	STRDON	X
C9A8	BAA5	NEWCHR	X
C9AB	BAA8	PRINT	ROUTINE - PRINT
C9AD	BAAA	PRINTC	X
C9D5	BAD2	FININL	X
C9E2	BADF	CRDO	OUTPUT A CARRIAGE RETURN
C9EC	BAED	CRFIN	X
C9EE	BAEF	PRTRTS	X
C9EF	BAF0	COMPRT	X
C9F2	BAF3	MORCO1	X
C9FC	BAFD	TABER	TAB AND SPC HANDLER
CA0C	BB0D	ASPAC	X
CA0D	BB0E	XSPAC	X
CA0E	BB0F	XSPAC2	X
CA11	BB12	NOTABR	X
CA17	BB18	XSPAC1	X
CA1C	BB1D	STROUT	PRINT STRING FROM ADDRESS IN .Y AND .A
CA1F	BB20	STRPRT	PRINT STRING POINTED TO BY INDEX
CA26	BB27	STRPR2	X
CA39	BB3A	OUTSPC	OUTPUT A SPACE
CA40	BB41	CRTSKP	OUTPUT A $1D
CA43	BB44	OUTQST	OUTPUT A ?
CA45	BB46	OUTDO	OUTPUT THE CHAR IN .A
CA4C	BB49	OUTRTS	X
CA4F	BB4C	TRMNOK	HANDLES BAD INPUT DATA
CA59	BB56	GETDTL	X

Table F-3. Hex Addresses and Label References: CBM BASICs (Continued)

BASIC 3.0	BASIC 4.0	Labels	Description
CA5D	BB5A	STCURL	X
CA61	BB5E	SNERR4	SYNTAX ERROR VECTOR
CA64	BB61	TRMNO1	X
CA6D	BB6A	DOAGIN	X
CA7D	BB7A	GET	ROUTINE - GET OR GET#
CA94	BB91	GETTTY	X
CAA7	BBA4	INPUTN	ROUTINE - INPUT#
CAB7	BBB4	IODONE	RESTORE INPUT TO KEYBOARD
CAB9	BBB6	IORELE	X
CAC1	BBBE	INPUT	ROUTINE - INPUT
CAD2	BBCD	NOT@TI	X
CADA	BBD5	GETAGN	X
CAED	BBE8	BUFFUL	X
0000	BBF1	PTHRTI	X
CAFA	BBF5	@INLIN	PROMPTS AND RECEIVES THE INPUT
CB04	BBFF	GINLIN	X
CB07	BC02	READ	ROUTINE - READ
CB0E	BC09	INPCON	X
CB10	BC0B	INPCO1	X
CB16	BC11	INLOOP	X
CB42	BC3D	@DATA	X
CB4B	BC46	GETNTH	X
CB4E	BC49	DATBK	X
CB52	BC4D	DATBK1	X
CB66	BC61	SET@UT	X
CB72	BC6D	RESETC	X
CB73	BC6E	NOWGET	X
CB7E	BC79	NOWGE1	X
CB8A	BC85	NUMINS	X
CB92	BC8D	STRDN2	X
CB9E	BC99	TRMOK	X
CBB9	BCB4	DATLOP	X
CBD2	BCCD	NOWLIN.	X
CBDF	BCDA	VAREND	X
CBEA	BCE5	VARY0	PRINT "EXTRA IGNORED " IF KEYBOARD AND A SEPERATOR
CBFB	BCF6	INPRTS	X
CBFC	BCF7	EXIGNT	MESSAGE - EXTRA IGNORED
CC0D	BD07	TRYAGN	MESSAGE - ?REDO FROM START
CC20	BD19	NEXT	ROUTINE - NEXT
CC26	BD1F	GETFOR	X
CC29	BD22	STXFOR	X
CC34	BD2D	ERRGO5	X
CC36	BD2F	HAVFOR	X
CC76	BD6F	NEWSGO	X
CC79	BD72	LOOPDN	CHECKS DATA FORMAT
CC8B	BD84	FRMNUM	JMP FRMEVL
CC8E	BD87	CHKNUM	CHECK THAT CURRENT TYPE IS NUMERIC
CC90	BD89	CHKSTR	CHECK THAT CURRENT TYPE IS STRING (CHKS VALTYP)
CC91	BD8A	CHKVAL	X
CC97	BD90	CHKOK	X
CC98	BD91	DOCSTR	X
CC9A	BD93	CHKERR	TYPE MISMATCH ERROR VECTOR
CC9C	BD95	ERRGO4	X
CC9F	BD98	FRMEVL	FORMULA EVALUATOR - EVALUATES ALL FORMULAS
CCA5	BD9E	FRMEV1	X
CCAA	BDA3	LPOPER	X

Table F-3. Hex Addresses and Label References: CBM BASICs (Continued)

BASIC 3.0	BASIC 4.0	Labels	Description
CCB9	BDB2	TSTOP	X
CCBC	BDB5	LOPREL	X
CCD8	BDD1	ENDREL	X
CCF1	BDEA	QPREC	X
CCFA	BDF3	DOPREC	X
CCFB	BDF4	NEGPRC	X
CD08	BE01	FINREL	X
CD12	BE0B	FINRE2	X
CD1A	BE13	QPREC1	X
CD21	BE1A	DOPRE1	PUSHES A PARTIAL EVALUATION ON THE STACK
CD31	BE2A	SNERR5	SYNTAX ERROR VECTOR
CD34	BE2D	PUSHF1	X
CD39	BE32	PUSHF	X
CD44	BE41	FORPSH	X
CD59	BE56	QOP	X
CD5C	BE59	QOPGO	X
CD5E	BE5B	QCHNUM	X
CD65	BE62	UNPSTK	X
CD67	BE64	PULSTK	RESTORE ARG FROM STACK (PUSHED EVALUATION)
CD81	BE7E	QOPRTS	X
CD83	BE80	UNPRTS	X
CD84	BE81	EVAL	EVALUATES NUMERIC FORMULAS
CD88	BE85	EVAL0	X
CD8D	BE8A	EVAL1	X
CD90	BE8D	EVAL2	X
CDA3	BEA0	PIVAL	STORAGE - THE BINARY.VALUE OF PI
CDA8	BEA5	QDOT	X
CDB8	BEB5	STRTXT	IMMEDIATE STRINGS HANDLER
CDC1	BEBE	STRTX2	X
CDC7	BEC4	EVAL3	X
CDCF	BECC	NOTOP	EVAL - NOT
CDDE	BEDB	EVAL4	X
CDEC	BEE9	PARCHK	EVALUATE A FUNCTION WITHIN ()'S (FRMEVL)
CDF2	BEEF	CHKCLS	CHECK FOR RIGHT PARENTHESIS)
CDF5	BEF2	CHKOPN	CHECK FOR LEFT PARENTHESIS (
CDF8	BEF5	CHKCOM	CHECK FOR A COMMA
CDFA	BEF7	SYNCHR	COMPARE TXTPTR AGAINST .A IF <> THEN...
CE03	BF00	SNERR	...SYNTAX ERROR VECTOR
CE08	BF05	DOMIN	SET UP FUNCTION FOR FUTURE EVALUATION
CE0A	BF07	GONPRC	X
0000	BF0C	CKSMB0	THE CHECKSUM BYTE FOR THE $B000 ROM
0000	BF0D	ISVJMP	JMP ISVAR
0000	BF10	PABB0	PATCHES
0000	BF10	PATCHG	P
0000	BF1D	PCTH0	P
0000	BF1E	PCTH1	P
0000	BF21	PATCHH	P
0000	BF2E	PATCHI	P
CE0F	BF8C	ISVAR	SET UP A VARIABLE NAME SEARCH
CE11	BF8E	ZZ6	X
CE12	BF8F	ISVRET	X
0000	BFC1	ISVDS	DS$ TEST AND HANDLER
CE42	BFD3	STRRTS	X
CE43	BFD4	GOOO	X
CE54	BFE5	GOOOOO	X
0000	BFFC	CHKDS	CHECK FOR A DS VARIABLE

Table F-3. Hex Addresses and Label References: CBM BASICs (Continued)

BASIC 3.0	BASIC 4.0	Labels	Description
CE69	C003	GETTIM	ASSIGN TIME TO TI
CE75	C00F	GSTATV	X
0000	C01C	GDSAV	X
CE82	C040	GOMOVF	X
CE89	C047	ISFUN	DISPATCH AND EVAL IF IT'S A FUNCTION
CEB3	C071	OKNORM	X
CEB8	C076	FINGO	PLACE FUNCTIONS DISPATCH ADDRESS IN JUMPER AND GO
CEC8	C086	OROP	EVAL - OR
CECB	C089	ANDOP	EVAL - AND
CEF8	C0B6	DOREL	DO COMPARISONS
CF10	C0CE	STRCMP	X
CF38	C0F6	STASGN	X
CF3D	C0FB	NXTCMP	X
CF43	C101	GCOMP	X
CF48	C106	GETCMP	X
CF54	C112	DOCMP	X
CF5D	C11B	GOFLOT	X
CF60	C11E	DIM3	MULTIPLE DIM RE-ENTRY (CHKS FOR A COMMA)
CF63	C121	DIM	ROUTINE - DIM
CF6D	C12B	PTRGET	SEARCHES FOR A BASIC VARIABLE
CF72	C130	PTRGT1	X
CF74	C132	PTRGT2	X
CF7E	C13C	INTERR	SYNTAX ERROR VECTOR
CF81	C13F	PTRGT3	X
CF91	C14F	ISSEC	X
CF92	C150	EATEM	X
CF9C	C15A	NOSEC	X
CFA6	C164	NOTSTR	X
CFB6	C174	TURNON	X
CFBD	C17B	STRNAM	X
CFD3	C18F	STXFND	X
CFD5	C191	LOPFND	X
CFDF	C19B	LOPFN	X
0000	C1AB	NXTPTR	MOVE SEARCH TO NEXT TABLE ENTRY
CFED	C1AC	NOTIT	X
CFF7	C1B6	ISLETC	X
D000	C1BF	ISLRTS	X
D001	C1C0	NOTFNS	DID NOT FIND VARIABLE - CREATE A NEW ONE
D007	C1C6	LDZR	X
D00C	C1CB	NOTEVL	X
D01C	C1DB	GOBADV	X
D01F	C1DE	GSTAVR	CHECK FOR ST CASE
0000	C1E6	GDSVAR	CHECK FOR DS CASE
D027	C1F2	VAROK	GOOD USABLE VARIABLE
D03D	C208	NOTEVE	X
D448	C21C	ARYVA2	X
D44C	C220	ARYVA3	X
D457	C228	ARYVGO	SEARCH THE ARRAYS
D488	C259	ARYGET	MOVE THRU THE ARRAY TABLES
D492	C263	GOGO	X
0000	C281	GOGO1	X
D4D0	C290	DVARTS	X
0000	C29D	ARYDON	X
D069	C2B9	FINPTR	LOGS BASIC VARIABLE LOCATION
D073	C2C3	FINNOW	X
D078	C2C8	FMAPTR	ARRAY POINTER SUBROUTINE

Table F-3. Hex Addresses and Label References: CBM BASICs (Continued)

BASIC 3.0	BASIC 4.0	Labels	Description
D084	C2D4	JSRGM	X
D089	C2D9	N32768	STORAGE - THEN BINARY VALUE -32768
D08D	C2DD	INTIDX	EVALUATE FORMULA RESULT IS POSITIVE INTEGER VALUE
D093	C2E3	POSINT	CONVERT FLOATING BINARY TO POSITVE INTEGER
D09A	C2EA	AYINT	CONVERT FLOATING BINARY TO INTEGER
D0A7	C2F7	NONONO	ILLEGAL QUANITY ERROR VECTOR
D0A9	C2F9	QINTGO	JMP QINT
D0AC	C2FC	ISARY	LOCATES AND/OR CREATES ARRAYS
D0B6	C306	INDLOP	X
D0F7	C347	LOPFDA	X
D103	C353	LOPFDV	X
D112	C362	NMARY1	X
D120	C370	BSERR	BAD SUBSCRIPT ERROR VECTOR
D123	C373	FCERR	ILLEGAL QUANITY ERROR VECTOR
D125	C375	ERRGO3	X
D128	C378	GOTARY	X
D13C	C38C	NOTFDD	X
D150	C39F	NOTFLT	X
D159	C3A8	STOMLT	X
D162	C3B1	LOPPTA	X
D172	C3C1	NOTDIM	X
D195	C3E4	GREASE	X
D1A4	C3F3	ZERITA	X
D1A9	C3F8	DECCUR	X
D1C6	C415	GETDEF	X
D1CE	C41D	INLPNM	X
D1E4	C433	BSERR7	SYNTAX ERROR VECTOR
D1E7	C436	OMERR1	OUT OF MEMORY ERROR VECTOR
D1EA	C439	INLPN2	X
D1EB	C43A	INLPN1	X
D1FC	C44B	ADDIND	X
D20D	C45C	NOTFL1	X
D213	C462	STOML1	X
D227	C476	DIMRTS	X
D228	C477	UMULT	INTEGER ARITHMETIC ROUTINES FOR MULTI-DIM ARRAYS
D231	C480	UMULTD	X
D23B	C48A	UMULTC	X
D254	C4A3	UMLCNT	X
D258	C4A7	UMLRTS	X
D259	C4A8	FRE	ROUTINE - FRE(X)
D260	C4AF	NOFREF	X
D26D	C4BC	GIVAYF	CONVERTS INTEGER TO FLOATING BINARY
D27A	C4C9	POS	ROUTINE - POS(X)
D27C	C4CB	SNGFLT	X
D280	C4CF	ERRDIR	IF COMMAND TYPE IS INDIRECT ONLY - ILLEGAL DIRECT
D288	C4D7	ERRGUF	UNDEFINED FUNCTION ERROR VECTOR
D28D	C4DC	DEF	ROUTINE - DEF FN()=
D2BB	C50A	GETFNM	X
D2CE	C51D	FNDOER	EVALUATES FN() IN FORMULAS
D2F2	C541	DEFSTF	X
D329	C578	DEFFIN	X
D33F	C58E	STRD	ROUTINE - STR$
D349	C598	TIMSTR	MAKE A STRING OUT OF INFO AT $01FF
D34F	C59E	STRINI	MAKE A STRING OUT OF (FACMO POINTER)
D357	C5A6	STRSPA	X
D361	C5B0	STRLIT	SCANS AND SETS UP STRING ELEMENTS

Table F-3. Hex Addresses and Label References: CBM BASICs (Continued)

BASIC 3.0	BASIC 4.0	Labels	Description
D367	C5B6	STRLT2	X
D371	C5C0	STRGET	X
D37E	C5CD	STRFIN	X
D382	C5D1	STRFI1	X
D383	C5D2	STRFI2	X
D38F	C5DE	STRST2	X
D399	C5E8	STRCP	X
D3A4	C5F3	PUTNEW	CHECK STRING TEMPS PLACE DATA IN TEMPS
D3AC	C5FB	ERRGO2	X
D3AF	C5FE	PUTNW1	X
D3CE	C61D	GETSPA	BUILDS STRING VECTORS
D3D0	C61F	TRYAG2	X
D3DB	C62D	TRYAG3	X
0000	C63A	TRYAG4	X
D3E5	C644	STRFRE	X
0000	C65A	GETRTS	X
D3F0	C65B	GARBAG	X
D400	C66A	GARBA2	DOES 'GARBAGE COLLECTION' - PACKS STRINGS
0000	C67E	GLOOP	X
0000	C68A	COL00	X
0000	C693	COL00B	X
0000	C69E	COL00A	X
0000	C6A9	COL01	X
0000	C6B2	COL02	X
0000	C6CE	GLOP1	X
0000	C6D8	COL02B	X
0000	C6F0	COL02A	X
0000	C700	GRBEND	JMP ENDGRB
0000	C703	COL03	MOVES FRESPC TO FRETOP
0000	C716	ENDGRB	MOVES FRESPC TO FRETOP
0000	C71F	SKIP2	X
0000	C724	SKIP2A	X
0000	C726	MOVPNT	X
0000	C730	MOV00	X
0000	C735	MOVTOP	X
0000	C73F	MOV01	X
0000	C744	SETINX	X
0000	C746	SET00	X
D517	C74F	CAT	CONCATENATE TWO STRINGS (FAC) AND (++(TXTPTR))
D537	C76F	SIZEOK	X
D554	C78C	MOVINS	X
D562	C79A	MOVSTR	X
D566	C79E	MOVDO	X
D56A	C7A2	MOVLP	X
D573	C7AB	MVDONE	X
D57C	C7B4	MVSTRT	X
D57D	C7B5	FRESTR	X
D580	C7B8	FREFAC	X
D584	C7BC	FRETMP	FREES UP TEMPORARY STRING POINTERS
0000	C7DE	RE500	X
0000	C7F6	FRE01	X
D5AF	C7FC	FREPLA	X
0000	C7FE	FRE02	X
D5B5	C811	FRETMS	X
D5C5	C821	FRERTS	X
D5C6	C822	CHRD	ROUTINE - CHR$(VALUE) (VALUE 0-255)

Table F-3. Hex Addresses and Label References: CBM BASICs (Continued)

BASIC 3.0	BASIC 4.0	Labels	Description
D5DA	C836	LEFTD	ROUTINE - LEFT$()
D5E0	C83C	RLEFT	X
D5E6	C842	RLEFT1	X
D5E7	C843	RLEFT2	X
D5E8	C844	RLEFT3	X
D5FF	C85B	PULMOR	X
D606	C862	RIGHTD	ROUTINE - RIGHT$()
D611	C86D	MIDD	ROUTINE - MID$()
D622	C87E	MID2	X
D63B	C897	PREAM	USED BY RIGHT
D656	C8B2	LEN	ROUTINE - LEN(STRING)
D65C	C8B8	LEN1	X
D665	C8C1	ASC	ROUTINE - ASC(STRING)
D672	C8CE	GOFUC	X
D675	C8D1	GTBYTC	DOES A CHRGET AND GETBYT
D678	C8D4	GETBYT	EVALUATE THE FORMULA AND RETURN A BYTE VALUE (IN .X)
D67B	C8D7	CONINT	X
D687	C8E3	VAL	ROUTINE - VAL(STRING)
D6A7	C903	VAL2	X
D6BD	C918	ST2TXT	X
D6C5	C920	VALRTS	X
D6C6	C921	GETNUM	EVALUATE FORMULA AND RETURN INTEGER VALUE (0-65535)
D6CC	C927	COMBYT	X
D6D2	C92D	GETADR	CONVERT FAC TO VALUE(0-65535) PLACE IN POKER
D6E8	C943	PEEK	ROUTINE - PEEK(X)
D6FB	C94E	GETCON	X
D6FE	C951	DOSGFL	X
D707	C95A	POKE	ROUTINE - POKE X
D710	C963	FNWAIT	ROUTINE - WAIT
D71F	C972	STORDO	X
D723	C976	WAITER	X
D72B	C97E	ZERRTS	X
D72C	C97F	FADDH	ADD 1/2 TO FPB VALUE IN FAC
D733	C986	FSUB	UNPACKS ARGUMENT AND SUBTRACT FPB
D736	C989	FSUBT	FPB SUBRATRACTION ARG-FAC
D76E	C998	FADD5	X
D773	C99D	FADD	UNPACK ARGUMENT INTO ARG DO A FPB ADD
D776	C9A0	FADDT	FPB ADDITION FAC=FAC+ARG
D783	C9AD	FADDC	X
D79F	C9C9	FADDA	X
D7A3	C9CD	FADD1	X
D7AF	C9D9	FADD4	X
D7BB	C9E5	SUBIT	X
D7DE	CA08	FADFLT	X
D7E3	CA0D	NORMAL	NORMALIZE ADDITION AND SUBTRACTION RESULTS
D7E7	CA11	NORM3	X
D803	CA2D	ZEROFC	FAC=0
D805	CA2F	ZEROF1	X
D807	CA31	ZEROML	MAKE SIGN POSITIVE
D80A	CA34	FADD2	X
D829	CA53	NORM2	X
D835	CA5F	NORM1	X
D842	CA6C	SQUEEZ	X
D844	CA6E	RNDSHF	X
D852	CA7C	RNDRTS	X
D853	CA7D	NEGFAC	COMPLEMENT FAC ENTIRELY

Table F-3. Hex Addresses and Label References: CBM BASICs (Continued)

BASIC 3.0	BASIC 4.0	Labels	Description
D859	CA83	NEGFCH	COMPLEMENT JUST THE NUMBER IN FAC
D87B	CAA5	INCFAC	INCREMENT FAC
D889	CAB3	INCFRT	X
D88A	CAB4	OVERR	OVERFLOW ERROR VECTOR
D88F	CAB9	MULSHF	SHIFER ROUTINES
D891	CABB	SHFTR2	X
D8A5	CACF	SHIFTR	X
D8B2	CADC	SHFTR3	X
D8B8	CAE2	SHFTR4	X
D8BC	CAE6	ROLSHF	X
D8C6	CAF0	SHFTRT	X
D8C8	CAF2	FONE	FLOATING-POINT-BINARY CONSTANTS
D8CD	CAF7	LOGCN2	X
D8E2	CB0C	S0R05	X
D8E7	CB11	S0R20	X
D8EC	CB16	NEGHLF	X
D8F1	CB1B	LOG2	X
D8F6	CB20	LOG	ROUTINE - LOG(X)
D8FD	CB27	LOGERR	ILLEGAL QUANITY ERROR VECTOR
D900	CB2A	LOG1	X
0000	CB5A	MULLN2	X
D934	CB5E	FMULT	FPB MULTIPLY FAC=FAC*ARG
D937	CB61	FMULTT	FPB MULTIPLY WITH ARG AND .AC LOADED
D965	CB8F	MLTPLY	X
D96A	CB94	MLTPL1	X
D96D	CB97	MLTPL2	X
D989	CBB3	MLTPL3	X
D997	CBC1	MULTRT	X
D998	CBC2	CONUPK	UNPACK MEMORY INTO ARG
D9C3	CBED	MULDIV	CHECK AND ADJUST EXPS OF FPB MULT AND DIV
D9C5	CBEF	MLDEXP	X
D9D0	CBFA	TRYOFF	X
D9E0	CC0A	MLDVEX	X
D9E6	CC10	ZEREMV	X
D9EB	CC15	GOOVER	OVERFLOW ERROR VECTOR
D9EE	CC18	MUL10	MULTIPLY FAC BY 10
D9F9	CC23	FINML6	X
DA04	CC2E	MUL10R	X
DA05	CC2F	TENC	FPB VALUE 10
DA0A	CC34	DIV10	DIVIDE FAC BY 10
DA13	CC3D	FDIVF	X
DA1B	CC45	FDIV	UNPACK MEMORY AND DIVIDE
DA1E	CC48	FDIVT	FAC = ARG/FAC
DA35	CC5F	DIVIDE	X
DA4B	CC75	SAVQUO	X
DA58	CC82	0SHFT	X
DA5B	CC85	SHFARG	X
DA69	CC93	DIVSUB	X
DA86	CCB0	LD100	X
DA8A	CCB4	DIVNRM	X
DA96	CCC0	DV0ERR	OVERFLOW ERROR VECTOR
DA9B	CCC5	MOVFR	MOVE RES TO FAC
DAAE	CCD8	MOVFM	MOVE MEMORY TO FAC
DAD3	CCFD	MOV2F	X
DAD6	CD00	MOV1F	X
DADC	CD06	MOVVF	X

Table F-3. Hex Addresses and Label References: CBM BASICs (Continued)

BASIC 3.0	BASIC 4.0	Labels	Description
DAE0	CD0A	MOVMF	MOVE FAC TO MEMORY
DB08	CD32	MOVFA	MOVE ARG TO FAC
DB0A	CD34	MOVFA1	X
DB0E	CD38	MOVFAL	X
DB18	CD42	MOVAF	MOVE FAC TO ARG
DB1B	CD45	MOVEF	X
DB1D	CD47	MOVAFL	X
DB26	CD50	MOVRTS	X
DB27	CD51	ROUND	ROUND FAC
DB2F	CD59	INCRND	X
DB37	CD61	SIGN	EXTRACT SIGN FROM FAC IN .A
DB3B	CD65	FCSIGN	X
DB3D	CD67	FCOMPS	X
DB44	CD6E	SIGNRT	X
DB45	CD6F	SGN	ROUTINE - SGN(X)
DB48	CD72	FLOAT	FLOAT THE SIGNED INTEGER IN FAC
DB50	CD7A	FLOATS	FLOAT THE SIGNED NUMBER IN FAC
DB55	CD7F	FLOATC	X
DB5B	CD85	FLOATB	X
DB64	CD8E	ABS	ROUTINE - ABS(X)
DB67	CD91	FCOMP	COMPARE ARG AND FAC .A=1←A<F
DB69	CD93	FCOMPN	X
DB9E	CDC8	FCOMPC	X
DBA4	CDCE	FCOMPD	X
DBA7	CDD1	QINT	FAC=INT(FAC) SIGNED ROUTINE - INT(X)
DBBB	CDE5	QISHFT	X
DBC6	CDF0	QINTRT	X
DBC7	CDF1	QINT1	X
DBD8	CE02	INT	ROUTINE - INT(X)
DBF5	CE1F	CLRFAC	.A TO ALL POSISTIONS OF FAC
DBFE	CE28	INTRTS	X
DBFF	CE29	FIN	FBP INPUT, TXTPTR POINTS TO ASCII, RETURNS IN FAC
DC03	CE2D	FINZLP	X
DC12	CE3C	QPLUS	X
DC16	CE40	FINC	X
DC19	CE43	FINDG0	X
DC1B	CE45	FIN1	X
DC3A	CE64	FINEC1	X
DC3C	CE66	FINEC	X
DC3F	CE69	FNEDG1	X
DC41	CE6B	FINEC2	X
DC4D	CE77	FINDP	X
DC53	CE7D	FINE	X
DC55	CE7F	FINE1	X
DC5E	CE88	FINDIV	X
DC67	CE91	FINMUL	X
DC6E	CE98	FINONG	X
DC73	CE9D	NEGX0S	X
DC76	CEA0	FINDIG	X
DC7D	CEA7	FINDG1	X
DC8A	CEB4	FINLOG	X
DC9D	CEC7	FINEDG	X
DCAC	CED6	MLEX10	X
DCBA	CEE4	MLEXMI	X
DCBF	CEE9	N0999	FPB VALUE 99999999.90625
DCC4	CEEE	N9999	FPB VALUE 999999999.5

Table F-3. Hex Addresses and Label References: CBM BASICs (Continued)

BASIC 3.0	BASIC 4.0	Labels	Description
DCC9	CEF3	NMIL	FPB VALUE 10-9
0000	CEF8	CK3MC0	CHECKSUM BYTE $C000 ROM
DCCE	CF78	INPRT	PRINT CURRENT LINE NUMBER
DCD9	CF83	LINPRT	PRINT NUMBER IN (.A←HIGH .Y←LOW)
DCE6	CF90	STROU2	JMP STROUT
DCE9	CF93	FOUT	FPB OUTPUT
DCEB	CF95	FOUTC	X
DCF3	CF9D	FOUT1	X
DD0C	CFB6	FOUT37	X
DD15	CFBF	FOUT7	X
DD17	CFC1	FOUT4	X
DD22	CFCC	FOUT3	X
DD2D	CFD7	FOUT38	X
DD34	CFDE	FOUT9	X
DD3B	CFE5	FOUT5	X
DD3E	CFE8	BIGGES	X
DD53	CFFD	FOUTPI	X
DD54	CFFE	FOUT6	X
DD5F	D009	FOUT39	X
DD70	D01A	FOUT16	X
DD72	D01C	FOUT8	X
DD74	D01E	FOUTIM	CLOCK ENTRY INTO FOUT
DD76	D020	FOUT2	X
DD9A	D044	FOUT41	X
DD9C	D046	FOUT40	X
DDA3	D04D	FOUTYP	X
DDBE	D068	STXBUF	X
DDD0	D07A	FOULDV	X
DDD2	D07C	FOUT11	X
DDDF	D089	FOUT12	X
DDEF	D099	FOUT14	X
DDFB	D0A5	FOUT15	X
DE10	D0BA	FOUT19	X
DE13	D0BD	FOUT17	X
DE18	D0C2	FOUT20	X
DE1D	D0C7	FHALF	FPB VALUE 1/2
DE1F	D0C9	ZERO	X
DE22	D0CC	FOUTBL	TABLES OF POWERS OF -10↑X
DE46	D0F0	FDCEND	END OF POWERS TABLE
DE5E	D108	TIMEND	FPB TIME CONVERSION TABLES
DE5E	D108	SGR	ROUTINE - SGR(X)
DE68	D112	FPWRT	ROUTINE (ARG↑FAC)
DE71	D11B	FPWRT1	X
DE8B	D135	FPWR1	X
DEA1	D14B	NEGOP	NEGATE THE NUMBER IN FAC
DEAB	D155	NEGRTS	X
DEAC	D156	LOGEB2	FPB VALUE LOG(E) BASE 2
DEB1	D15B	EXPCON	LOG AND EXPONENT FPB TABLES
DEDA	D184	EXP	ROUTINE - EXP(FAC)
DEEA	D194	STOLD	X
DEF5	D19F	GOMLDV	X
DEF8	D1A2	EXP1	X
DF08	D1B2	SWAPLP	X
DF2D	D1D7	POLYX	POLYNOMIAL EVALUATOR
DF43	D1ED	POLY	POLYNOMIAL EVALUATOR
DF47	D1F1	POLY1	X

Table F-3. Hex Addresses and Label References: CBM BASICs (Continued)

BASIC 3.0	BASIC 4.0	Labels	Description
DF56	D200	POLY3	X
DF5A	D204	POLY2	X
DF67	D211	POLY4	X
DF77	D221	RMULC	X
DF7B	D225	RADDC	X
DF7F	D229	RND	ROUTINE - RND(X)
DF9D	D247	QSETNR	X
DFB2	D25C	RND1	X
DFC2	D26C	STRNEX	X
DFD5	D27F	GMOVMF	X
DFD8	D282	COS	ROTINE - COS(X)
DFDF	D289	SIN	ROUTINE - SIN(FAC)
E011	D2BB	SIN1	X
E014	D2BE	SIN2	X
E021	D2CB	SIN3	X
E028	D2D2	TAN	ROUTINE - TAN(FAC)
E050	D2FA	COSC	X
E054	D2FE	PI2	FPB VALUE PI/2
E059	D303	TWOPI	FBP VALUE 2*PI
E05E	D308	FR4	FPB VALUE 1/4
E063	D30D	SINCON	SIN TABLES FPB VALUES
E08C	D32C	ATN	ROUTINE - ATN(FAC)
E094	D334	ATN1	X
E0A2	D342	ATN2	X
E0B5	D355	ATN3	X
E0BB	D35B	ATN4	X
E0BC	D35C	ATNCON	X
E0F9	D399	INITAT	BASIC SYSTEM INITIALIZATION CODE
E0FF	D39F	CHDGOT	X
E110	D3B0	CHDRTS	X
0000	D3B6	INIT	BASIC SYSTEM INITIALIZATION ROUTINE
E131	D3C9	MOVCHG	X
E15D	D400	LOOPMM	X
E165	D408	LOOPM1	X
E174	D417	USEDEC	X
E178	D41B	USEDEF	X
E1B7	D44B	WORDS	MESSAGE - 'BYTES FREE'
E1C4	D458	FREMES	MESSAGE - '### COMMODORE BASIC ###'
E1DE	D472	LASTWR	LAST BYTE OF BASIC SYSTEM CODE+1
0000	DEA4	PATCH2	PATCHES
E844	E844	CHTIM	X
0000	FF93	CONCAT	VECTOR - CONCAT
0000	FF96	DOPEN	VECTOR - DOPEN
0000	FF99	DCLOSE	VECTOR - DCLOSE
0000	FF9C	RECORD	VECTOR - RECORD
0000	FF9F	FORMAT	VECTOR - FORMAT
0000	FFA2	COLECT	VECTOR - COLLECT
0000	FFA5	BACKUP	VECTOR - BACKUP
0000	FFA8	DCOPY	VECTOR - COPY
0000	FFAB	APPEND	VECTOR - APPEND
0000	FFAE	DSAVE	VECTOR - DSAVE
0000	FFB1	DLOAD	VECTOR - DLOAD
0000	FFB4	DIRCAT	VECTOR - DIRECTORY
0000	FFB4	DCAT	VECTOR - CATALOG
0000	FFB7	RENAME	VECTOR - RENAME
0000	FFBA	SCRATC	VECTOR - SCRATCH

Table F-3. Hex Addresses and Label References: CBM BASICs (Continued)

BASIC 3.0	BASIC 4.0	Labels	Description
0000	FFBD	READDS	VECTOR - DS AND DS$
FFC0	FFC0	COPEN	VECTOR - OPEN
FFC3	FFC3	CCLOS	VECTOR - CLOSE
FFC6	FFC6	CCOIN	VECTOR - SET INPUT DEVICE
FFC9	FFC9	CCOUT	VECTOR - SET OUTPUT DEVICE
FFCC	FFCC	CLSCHN	VECTOR - RESTORE NORMAL I/O DEVICES
FFCC	FFCC	CCCHN	SAME AS ABOVE
C481	FFCF	INCHR	VECTOR - INPUT A CHARACTER (FROM SCREEN)
FFCF	FFCF	CINCH	SAME AS ABOVE
FFD2	FFD2	OUTCH	VECTOR - OUTPUT A CHARACTER
FFD5	FFD5	CLOAD	VECTOR - LOAD
FFD8	FFD8	CSAVE	VECTOR - SAVE
FFDB	FFDB	CVERF	VECTOR - VERIFY
FFDE	FFDE	CSYS	VECTOR - SYS
FFE1	FFE1	ISCNTC	VECTOR - TEST STOP KEY
FFE4	FFE4	CGETL	VECTOR - GET CHARACTER FROM KEYBOARD BUFFER
FFE7	FFE7	CCALL	VECTOR - ABORT ALL I/O CHANNELS
000F	0000	CONTW	Z
000D	0000	CNTWFL	Z
000F	0000	LINWID	Z
0010	0000	NCMWID	Z
006C	0000	STRNGI	Z
007F	0000	0	Z
C494	0000	INCRTS	Z
C721	0000	SNERRX	Z
D404	0000	FNDVAR	Z
D41E	0000	TVAR	Z
D427	0000	SVARS	Z
D433	0000	SVAR	Z
D43B	0000	SVARGO	Z
D440	0000	ARYVAR	Z
D48A	0000	ARYSTR	Z
D497	0000	DVARS	Z
D4A1	0000	DVAR	Z
D4B6	0000	DVAR2	Z
D4C0	0000	DVAR3	Z
D4DB	0000	GRBRTS	Z
D4E0	0000	GRBPAS	Z
D5B0	0000	FRETRT	Z
D745	0000	STORD1	Z
D745	0000	STORD1	Z
D745	0000	STORD1	Z
D745	0000	STORD1	Z

Index

Other OSBORNE/McGraw-Hill Publications

An Introduction to Microcomputers: Volume 0 — The Beginner's Book
An Introduction to Microcomputers: Volume 1 — Basic Concepts, second edition
An Introduction to Microcomputers: Volume 2 — Some Real Microprocessors
An Introduction to Microcomputers: Volume 3 — Some Real Support Devices
Osborne 4 & 8-Bit Microprocessor Handbook
Osborne 16-Bit Microprocessor Handbook
8089 I/O Processor Handbook
CRT Controller Handbook
68000 Microprocessor Handbook
8080A/8085 Assembly Language Programming
6800 Assembly Language Programming
Z80 Asembly Language Programming
6502 Assembly Language Programming
Z8000 Assembly Language Programming
6809 Assembly Language Programming
Running Wild — The Next Industrial Revolution
The 8086 Book
PET and the IEEE 488 Bus(GPIB)
PET/CBM Personal Computer Guide, 2nd Edition
Business System Buyer's Guide
CP/M® II User's Guide
Apple II® User's Guide
Microprocessors for Measurement & Control
Some Common BASIC Programs
Some Common BASIC Programs — PET/CBM Edition
Practical BASIC Programs
Payroll with Cost Accounting
Accounts Payable and Accounts Receivable
General Ledger
8080 Programming for Logic Design
6800 Programming for Logic Design
Z80 Programming for Logic Design

The PET/CBM PERSONAL COMPUTER GUIDE is a step-by-step guide that assumes no prior knowledge of computers. If you can read English, you can use this book.

This revised second edition provides even more useful material than the popular first edition. It covers all the most recent CBM products: the CBM 8000 and 4000 series computers, the 2040 and 8050 disk drives, and programmable printers. Adam Osborne co-authored this new edition. He has re-written it to be a step-by-step BASIC tutorial. So if you don't know BASIC, don't worry. This book will teach you both BASIC and CBM BASIC.

If you're thinking about buying any personal computer, this book will show you what the PET can do for you. If you've just bought a PET or CBM, this is the book you must have to really understand your computer. By using the examples found in this book you'll quickly get your PET/CBM up and running. These examples are thoroughly documented so you can learn how and why the programs work. It's the "how" and "why" that are important in learning to make the PET work efficiently for you.
The PET PERSONAL COMPUTER GUIDE covers everything you'll need to be master of your PET.

Complete operating instructions for:
- keyboard
- tape cassette
- disk
- printer

Description of all CBM BASIC statements
A BASIC tutorial
Optimal programming techniques including
- input/output programming
- file handling
- screen editing

Programming problems with solutions
CBM capabilities and limitations

commodore
Approved

ISBN 0-931988-55-1