Memory Map

HEXLOC	NAME	DESCRIPTION
0000-00FF		HARDWARE PAGE 0
0000-0002	WARM	Initially Jump to Cold Start ($BD11) then becomes Jump to Warm Start ($A274)
0003-0005	JPRNT	Jump to Message printer at $A8C3
0006-0007	USRINP	USR input argument vector
0008-0009	USROUT	USR output argument vector
000A-000C	USRVEC	USR Vector
000D	NULFLG	NULL FLAG-number of NULL's output to ACIA in addition to BASIC's normal 10.
000E	TRMCNT	Terminal character count-# of chars printed since last CR - also used as Line Buffer Ptr
000F	LINLEN	Output line length (default is $48 - 72) Length of output line before auto CR/LF '0' gives automatic double spacing.
0010	TRWDTH	Terminal width for comma spaced columns
0011-0012	BINARG	Misc Args of stmts like PEEK, POKE, etc.
0013-005A	BUFFER	BASIC's Line Input Buffer
005B	DELIM1	Used by decimal to binary converter etc.
005C	DELIM2	Scan between quotes flag, etc.
005D	CHRCNT	# of characters in BUFFER & Subscript Flag
005E	DIMLET	Default DIM flag,LET flag
005F	VARTYP	Variable Type ($FF=String $00=Numeric)
0060	MFLAG	Misc flag (DATA, LIST, QUOTE FLAG, Etc.)
0061		Subscript Flag=0, FN Flag=$80
0062	IRFLAG	READ/INPUT flag ($00=INPUT $98=READ)
0063	COMPFL	Comparison evaluation flag
0064	CTRLO	Control O flag ($80=Discard output)
0065-0067		Temp String Descriptor Stack Pointer
0068-0070		Temp String Descriptor Stack
0071-0072	TMPTR1	Temporary Pointer for Garbage Collector, Dec to Bin converter, etc.
0073-0074	TMPTR2	Temporary Pointer for NEW LINE, DELETE LINE, VAL, Etc.
0075-0078	RESACC	Reserve FP Accumulator Staging area for MULTIPLY
0079-007A	TXTTAB	Start of BASIC Text Pointer ($0301)
007B-007C	VARTAB	Start of Variable Table pointer (End of Program + 1)
007D-007E	ARRTAB	Start of Array Table pointer
007F-0080	ENDTAB	End of Array Table Pointer
0081-0082	STRSPC	Start of String Space pointer goes from here to end of memory.

(continued)

Memory Map *(continued)*

HEXLOC	NAME	DESCRIPTION
0083-0084	STRPTR	Work pointer into String Space
0085-0086	MEMSIZ	End of memory + 1
0087-0088	XQTLIN	Current Line # - ($88=##$FF if Direct Mode)
0089-008A	OLDLIN	Line # at 'BREAK' or 'STOP'
008B-008C	OLDPTR	Pointer to BASIC Code for 'CONT'
008D-008E	DATLIN	Line # for Current DATA statement
008F-0090	DATPTR	Address of next DATA statement
0091-0092	INPPTR	Address of next value in Current DATA stmt
0093-0094	VARNAM	ASCII name of present variable
0095-0096	VARPNT	Address of present variable
0097-0098	FORPTR	Address of Variable to be assigned value by LET - also FOR/NEXT pointer
0099-00A0		Various work pointers etc.
00A1-00A3	JUMPER	General purpose JMP instruction - Put target Address in $A2-$A3
00A4-00A9		Various work and storage area
00AA-00AB	VARPTR	Pointer to next line after LIST
00AC-00B0	ACCUM1	Floating Point Accumulator # 1 - Format is 1 byte Exponent-3 bytes Mantissa-1 byte Sign
00AD-00AE		Contents are printed in Decimal by $B962
00AE-00AF	FAC	Where INVAR Rtne at $AE05 puts it's Argument
00B0		Sign of Floating Accumulator #1
00B1		Series evaluation Constant pointer
00B2		ACCUM1 high order (overflow) word
00B3-00B7	ACCUM2	Floating accumulator #2 (Format = EMMMS)
00B8		ACCUM1/ACCUM2 sign comparison result flag
00B9		ACCUM1 low order (rounding) word
00BA-00BB		Series pointer
00BC-00D3	CHRGET	PARSER subroutine-gets next byte from ($C3) Returns with character in 'A' CARRY clear if value is ASCII 0-9 else CARRY set. 'A' will equal zero if end of line. Ignores spaces. Copied from $BCEE at Cold Start.
00C2	CHRGOT	Entry to get current character
00C3-00C4	CHRPTR	Code Address pointer for CHRGET routine
00D1-00D7		Used by both BASIC & Extended Monitor
00D4-00D7	RNDX	Random number Seed for RND
00D8-00FA		Not used by BASIC or MONITOR
00FB		ACIA / KEYBOARD flag for MONITOR
00FC		Temporary storage for MONITOR
00FD		Temporary Data storage for MONITOR
00FE-00FF		Temporary Address storage for MONITOR
0100-01FF		HARDWARE PAGE 1
0100-010C		Number conversion to ASCII storage area
0130-0131	NMI	NMI interrupt vectored here
01C0-01C1	IRQ	IRQ interrupt vectored here
0133-01FC	STACK	BASIC's stack area

(continued)

Memory Map *(continued)*

```
  HEXLOC    NAME     DESCRIPTION
-------------------------------------------------------------

0200-02FF           HARDWARE PAGE 2

0200        CURSOR  Current Cursor offset
0201        SAVER   Character to be printed
0202        CRTWRK  CRT emulator work byte
0203        LOADFL  LOAD flag  (0=OFF  1=ON)
0204                CRT temporary
0205        SAVEFL  SAVE flag  (0=OFF  1=ON)
0206        BAUD    CRT Emulator BAUD rate (0=FAST  255=SLOW)
0207-020E   VEB     Volatile Execution Block  For screen scroll
                       NOT re-entrant

020F-0211           Not used by BASIC
0212        CFLAG   CTRL C flag  <>0 is disable
0213        KBWORK  Keyboard Poll work byte
0214        OLDKEY  Keyboard Poll last key
0215        NEWKEY  Keyboard Poll this key
0216        DBOUNC  Keyboard Poll debounce
0217                apparently un-used
0218-0219   C1INP    C1 INPUT  vector     (=$FFBA)
021A-021B   C1OUT    C1 OUTPUT vector     (=$FF69)
021C-021D   C1CTLC   C1 CTRL C vector     (=$FF9B)
021E-021F   C1LOAD   C1 LOAD   vector     (=$FF8B)
0220-0221   C1SAVE   C1 SAVE   vector     (=$FF69)
0222-02FF           Not used by BASIC

0300-09FF           BASIC workspace  (User RAM)



A000-BFFF           ** ROM BASIC **

A000-A037   INIDIS  Initial Keywords Dispatch table
                       (= Address of routine minus 1)

A038-A065   FUNDIS  Functions Dispatch table
                       (= actual address of routine)

A066-A083   ARITHD  Arithmetic Operation Table
                       3 bytes (1=precedence   2 and 3=Address)

A084-A163   KEYTBL  Keyword Tables - in Token order
                       in ASCII with last byte of each Keyword
                       bit 7 on. End of table marked by Null.
```

(continued)

Memory Map *(continued)*

HEXLOC	NAME	DESCRIPTION
A084-A0F0	INITLK	Initial Keywords
A0F1-A114	SECNDK	Secondary Keywords
A115-A163	FUNCTK	Functions Keywords
A164-A185	ERRTBL	Error message table-High Bit of 2nd character set
A186-A18C	ERRMSG	ASCII Msg-' ERROR ',$00
A18D-A191	INMSG	ASCII Msg-' IN ',$00
A192-A198	OKMSG	ASCII Msg-CR,LF,'OK',CR,LF,$00
A199-A1A0	BRKMSG	ASCII Msg-CR,LF,'BREAK',$00
A1A1-A1CE		Look back thru BASIC Stack for most recent GOSUB or FOR
A1CF-A211		Open space in memory
A212-A21E	CKSTAK	Check for 'OM' and Stack overflow
A21F-A24B	CKMEM	Check free memory available
A24C	OMERR	Out of Memory error
A24E	ERRPRT	Error message printer-enter with X=offset of message from table at $A164
A274	WARMST	Warm start location
A27D	INMAIN	BASIC input routine
A284-A34A	INSERT	Inserts tokenized line into Text area and Adjusts all forward ptrs exits by JMP INMAIN
A295		Tokenize Buffer and store in text area
A2A2	DELLIN	Deletes line from Text area
A31C-A34D	CHAIN	Rebuild chaining of BASIC lines in Memory
A357	LININP	Input and fill buffer-put null at end
A386	CHRINP	Input a character - calls $FFEB
A399	TOGGLO	Toggle CTRLO - Output flag
A3A6-A431		Tokenize in Buffer
A432-A460	FINDLN	Find BASIC line whose # <= contents of BINARG and place address in $AA-AB Carry Flag Set if line found, Clear if not ending with a NULL ($00) or a colon
A461-A479	NEWCMD	NEW routine
A463		Put $0 in Start Text ($0301-0302) - then
A46B		Reset VARTAB to TXTTAB+2 - then
A477		Reset CHRPTR - then
A47A	CLEAR	Reset STRSTC to equal MEMSIZ - then
A482		Reset ARRTAB & ENDTAB to = VARTAB - then
A48E		Do RESTORE - then
A491		Put #$68 into Address $65 - then
A495		Reset BASIC Stack to #$FC - then
A4A0		Disable 'CONT' & zero Subscript Flag
A4A7		Initialize Code Pointer (CHRPTR) to the beginning of program ($0301)

(continued)

Memory Map *(continued)*

HEXLOC	NAME	DESCRIPTION
A4B5	LSTCMD	LIST routine
A556-A5FE	FORCMD	FOR routine
A5C2-A619		Main BASIC execution loop
A5FC	XQT	Entry to BASIC execute loop
A61A-A628	RSTCMD	RESTORE routine
A629	CTLC	CTRL C routine
A636	CTLCMD	CTRL C entry point
A638	STPMCD	STOP routine
A63A	ENDCMD	END routine
A661-A67A	CNTCMD	CONT routine
A67B	NULCMD	NULL routine
A68C	CLRCMD	CLEAR routine
A691-A69B	RUNCMD	RUN routine
A69C-A6B8	GSBCMD	GOSUB routine
A6B9-A6E5	GTOCMD	GOTO routine
A6E6-A70B	RETCMD	RETURN routine
A6F4	RGERR	Return W/O Gosub error
A6F7	USERR	Undefined Statement error
A70C-A719	DATCMD	DATA routine
A71A-A73B	NXSTMT	Scan for next BASIC Statement
A71D-A73B	NXLINE	Scan for next BASIC Line
A73C-A74E	IFCMD	IF routine
A74F-A75E	REMCMD	REM routine
A75F-A77E	ONCMD	ON routine
A77F-A7B8	EVAL	Evaluate expression whose beginning address is in CHRPTR. Convert ASCII to fixed with result appearing in BINARG.
A785		Same as EVAL without zeroing the result field first (BINARG)
A7B9-A828	LETCMD	LET routine
A829-A8C2	PRTCMD	PRINT routine - Entry at $A82F
A866		Puts null at end of buffer - then
A86C	DOCRLF	Output CR/LF - then
A878	DONULL	Output Nulls from NULFLG and RTS
A88B	COMCOL	Handle comma separators in PRINT routine
A8A2	SPCTAB	Do SPC and TAB in PRINT routine
A8C3-A8DF	MSGPRT	Print ASCII message. Enter with ADDR HI in Y, ADDR LO in A. Message is ASCII ending with a NULL ($00)
A8E0	SPCOUT	Outputs one space (' ')
A8E3	QMOUT	Outputs question mark ('?')
A8E5	AOUT	Outputs character in A updates TRMCNT and checks for Line Length overflow
A904	BADINP	Handle bad input data
A923-A945	INPCMD	INPUT routine - Clears CTRL O
A925		INPUT without clear CTRL O
A946-A94E	QINPLN	Prompt with '? ' then receive INPUT via Jump to LININP ($A357)

(continued)

Memory Map *(continued)*

HEXLOC	NAME	DESCRIPTION
A94F	REACMD	READ routine
AA1C-AA2C	XTRMSG	ASCII Msg-'?EXTRA IGNORED',CR,LF,$00
AA2D-AA3F	RDOMSG	ASCII Msg-'?REDO FROM START',CR,LF,$00
AA40	NXTCMD	NEXT routine
AAAD	FRMEVL	Formula Expression Evaluator Gets Value from BASIC line (Evaulates Literals, Variables or Expressions). Puts value in ACCUM1, does TM check.
AABC	TMERR	Type Mis-match error
AAC1	FRMEV2	Same as FRMEVL without TM check
ABA0	NUMEVL	Numeric Expression Handler
ABAC	STREVL	String Expression Handler
ABD8	NOTCMD	NOT routine
ABF5-ABFA		Evaluate expression within parentheses
ABFB	CKRPAR	SN Error if next character not ')'
ABFE	CKLPAR	SN Error if next character not '('
AC01	CKCOMA	SN Error if next character not ','
AC03	CKCHR	SN Error if next char not = contents of A
AC0C	SNERR	Syntax Error
AC18	GETVAR	Find Variable, put addr in $AD-AE, check Variable type, and if Numeric transfer Value to ACCUM1.
AC27-AC65		Setup and JMP to Functions
AC66-AC93	ORCMD	OR routine
AC69-AC93	ANDCMD	AND routine
AC96-ACFD		Perform Comparisons
AD01-AD0A	DIMCMD	DIM routine
AD0B-AD80	FNDVAR	Get variable name from BASIC line, put name in VARNAM, find and put address of variable in VARPTR and in A & Y
AD53	FNDSIM	Find or Create Simple Variable Expects variable name in VARNAM, finds and puts address in VARPTR and A & Y
AD81-AD8A	CKLETR	Check if char in A is A-Z, Set carry if yes
AD8B-ADE5	CRESIM	Create Simple variable in table
ADE6-ADF6		Array pointer subroutine
ADF7-ADFA		FP Constant (-32768)
ADFB-AE16	INTEVL	Evaluate Integer Expressions
AE05	INVAR	Convert ACCUM1 to integer (+/-32768) in $AE,$AF

(continued)

Memory Map *(continued)*

HEXLOC	NAME	DESCRIPTION
AE17-AFAC	FNDARR	Find or Create Array
AE85	BSERR	Bad Subscript Error
AE88	FCERR	Function Call Error
AEA1	CREARR	Create Array in Table
AF7C		Compute array subscript size
AFAD-AFC0	FRECMD	FRE(X) routine - Calls Garbage Collector
AFC1-AFCD	OUTVAR	Assigns value to Variable ???
AFCE-AFD3	POSCMD	POS routine
AFD4-AFDD	IDCHK	Check for ID Error
AFD9	IDERR	Illegal Direct command error
AFDE-B00A	DEFCMD	DEF routine
B00B-B01D	FNCHK	Check FN syntax
B01E-B08B	FNEVAL	Evaluate FNx and Store in Stack
B08C-B114	STRCMD	STR$ routine
B0AE-B114		Scan and set-up String & Find Length
B0F3	STERR	String Too complex error
B115-B146		Allocate room in String storage area and build Descriptor for String
B147-B24C	GARCOL	Garbage Collector routine
B1D4-B217		Check for string most eligible for collection
B218		Collect a string
B24D-B289		Perform Concatenation
B268	LSERR	String too Long error
B28A-B2B2		Store String in String Area
B2B3-B2EA		Discard unwanted String
B2EB-B2FB		Clean String Descriptor Stack
B2FC-B30F	CHRCMD	CHR$ routine
B310-B33B	LFTCMD	LEFT$ routine
B33C-B344	RGTCMD	RIGHT$ routine
B347-B38B	MIDCMD	MID$ routine
B38C-B391	LENCMD	LEN routine
B392-B39A		Find String & Length
B39B-B3A7	ASCCMD	ASC routine
B3AB-B3BC	GETBYT	Evaluate Integer (<256) from line into X
B3BD-B3FB	VALCMD	VAL routine
B3FC-B407		Gets 16 bit value from line-puts in BINARG, checks for comma, then gets 8 bit argument in X and then RTS
B408-B41D	FIX	Convert contents of ACCUM1 to two byte Fixed binary number and put in BINARG
B41E-B428	PEKCMD	PEEK routine
B429-B431	POKCMD	POKE routine
B432-B44D	WAITCD	WAIT routine
B44E-BCED		6 DIGIT FLOATING POINT MATH PACKAGE
B44E-B454		Add 0.5 TO ACCUM1
B455	MINUS	Perform Subtraction
B467	PLUS	Perform Addition
B4BB		Subtract ACCUM2 from ACCUM1
B4DO		Arithmetic to normalize floating point
B4F1		Set ACCUM1 to zero
B4F8		Add ACCUM1 to ACCUM2

(continued)

Memory Map *(continued)*

HEXLOC	NAME	DESCRIPTION
B537		Complement ACCUM1
B564-B568	OVERR	Overflow Error
B569-B59B		Multiply a byte
B59C-B5BC		LOG Coefficients Constants
B5BD	LOGCMD	LOG routine
B5FB	MULT	Perform Multiplication ('*')
B62D		Add RESACC to ACCUM1
B64D-B672		Unpack Memory into ACCUM2
B673-B68F		Test and Adjust ACCUM1 and ACCUM2
B690-B69D		Handle underflow and overflow
B69E-B6B4		Multiply ACCUM1 by 2
B6B5-B6B8		Floating Point constant (2)
B6B9-B6C7		Divide by 2
B6CA	DIV	Perform Divide by
B6CF		Perform Divide into
B711		Subtract ACCUM1 from ACCUM2 result in ACCUM1
B737	D/0ERR	Divide by zero error
B74B-B76A		Unpack memory into ACCUM1
B768-B79A		Pack ACCUM1 into memory
B79B-B7AA		Move ACCUM2 to ACCUM1
B7AB-B7B9		Move ACCUM1 to ACCUM2
B7BA-B7C7		See if need to Normalize ACCUM1
B7CA-B7D7		Get sign of ACCUM1
B7D8-B7F4	SGNCMD	SGN routine
B7E8	FLOAT	Convert contents of $AD(Hi)-$AE(Lo) from Fixed Binary to floating point and put in ACCUM1. Enter with X=#$90 and Carry Set
B7F5-B7F7	ABSCMD	ABS routine
B7F8-B830		Compare ACCUM1 to Mem at (A,Y)
B831-B861		Convert Floating to Fixed
B862-B886	INTCMD	INT routine
B887-B946		Convert ASCII String to Floating Point
B947-B952		constants used with ASCII conversion
B953-B96D		Prints ' IN ' and the Line Number
B95A		Prints current line number
B95E-B96D	NUMPRT	Print decimal integer whose value is in A (lo) and X (hi)
B962		Print contents of $AD(Hi)-$AE(Lo) as Dec. integer
B96E-BA95	ASCII	Convert floating number in ACCUM1 to ASCII string at $0100-0107. $0100 is sign (space or -) - String terminated by NULL
BA96-BAAB		Constants
BAAC-BCED		FUNCTIONS PACKAGE
BAAC	SQRCMD	SQR routine
BAB6	POWER	Perform ^ (exponentiation)
BAEF-BAF9		Perform negation

(continued)

Memory Map *(continued)*

HEXLOC	NAME	DESCRIPTION
BAFA-BB1A		Constants EXP Coefficients
BB1B-BB6D	EXPCMD	EXP routine
BB6E-BBB7		Series Summations
BBB8-BBBF		RND Constants
BBC0-BBFB	RNDCMD	RND routine
BBFC	COSCMD	COS routine
BC03	SINCMD	SIN routine
BC4C	TANCMD	TAN routine
BC78-BC98		SIN & COS Constants & Coefficients
BC99-BCC8	ATNCMD	ATN routine
BCC9-BCED		ATN Coefficients
BCEE-BD09	PARSER	CHRGET - Transferred to $BC
BD0A-BD10		Prints Author's Name
BD11	COLD	Cold Start routines
BE39-BE4D		ASCII msg-'WANT SIN-COS-TAN-ATN',$00 Left over from when Tape BASIC
BE4E-BE71		ASCII msg- Author's Name
BE72-BE7D		ASCII msg'MEMORY SIZE',$00
BE7E-BE8C		ASCII msg-'TERMINAL WIDTH',$00
BE8D-BEE1		ASCII msg-Version & Copyright Notice
BEE4		UART input routine (430 Board) Contains error (BF05 should be FB05)
BEF3		UART output routine (430 Board)
BEFE		UART initialization routine (430 Board)
BF07		C2 ACIA at $FC00 input routine
BF15		C2 ACIA at $FC00 output routine
BF22		C2 ACIA at $FC00 initialization set to 8 bits data, 2 stop bits no parity, divide by 16
BF2D-BFF2	CRTEMU	CRT emulator - prints char in 'A' to screen, scrolls etc.
BFC2		Prints char in 'A' to screen - but doesn't update Cursor pointer
BFF3-BFFA		Code transferred to VEB for Scroll
C000-C01F		Disk Controler PIA and ACIA
CE00-CEFF	MULTIP	C3 Multi User Ports
CF00-CFFF	PORT8	CA-10-X Board ACIA's
D000-DFFF		C3 Hard disk buffer
D000-D3FF		C1 Video memory
D000-D7FF		C2 Video memory
DE00		C2 Screen size/Color/Sound Latch
DF00	KBPORT	Polled Keyboard Port
E000-EFFF		C3 Level 3 & CP/M Memory
E000-E7FF		Color Memory (Low 4 bits)

(continued)

Memory Map *(continued)*

HEXLOC	NAME	DESCRIPTION
E800-EFFF		Not presently used in C1/C2
F000-F001		C1 ACIA Port #1
F800-FBFF		C1 65V ROM extra pages for other machines
FC00-FC01		C2 ACIA Port #1
FC00-FCFF		C1 Floppy Bootstrap Routines
FC00		C1 Auto bootstrap entry
FC06		C1 Load track zero into $2200 up
FC8B		C1 Unload Floppy head
FC91		C1 Time delay routine - delay equals 1.25 MS times value of X at 1 MHZ
FC9C		Load next byte from disk to A
FCA6		C1 ACIA at $F000 initialization
FCB1		C1 ACIA at $F000 output routine
FCBE		Write complement of A to keyboard
FCC6		Load complement of keyboard into X
FCFF		Load complement of keyboard into A
FD00-FDFF	KBPOLL	Keyboard Polling Routine - Polls keyboard and returns with the ASCII value of key depressed in A
FE00-FEFF		65V PROM MONITOR
FE00	MONITR	Entry-clears screen, reset ACIA and Stack
FE0C	MONWRM	Entry-without Stack initialization
FE43	MONADR	Entry to address mode
FE80	OTHER	Input ASCII character, returns result in A, bit 7 cleared
FE93	LEGAL	Convert ASCII hex to binary, result in A (A=$80 if not ASCII Hex 0-F)
FEAC	MONOUT	Output Address & Data in Monitor format (ADDR from $FE-FF DATA from $FC)
FEB0	MOUT1	Output X bytes from $FC+X to screen at $D0C6+Y. Set X and Y before entering X decreases and Y increases.
FECA	DIGIT	Output LSD (HEX) from A to screen at $D0C6+Y. Set Y before calling.
FEDA	ROLL	Move LSD (HEX) in A to 2 byte number at ($FC) +X. Set X before calling.
FEE9	MONINP	Return Character in A from Keyboard or ACIA Port 1, depending on LDFLAG

(continued)

Memory Map *(continued)*

```
  HEXLOC    NAME     DESCRIPTION
-------------------------------------------------------------
FF00-FFFF            BASIC I/O SUPPORT

FF00        RESET    Reset Entry Point
FF67        BASOUT   BASIC's output vector. Outputs 1 byte to
                        screen, and if SAVEFL on, outputs to Port #1
                        (also ouput 10 NULLS & <LF> if char is <CR>)

FF89        LOADRT   LOAD flag routine
FF94        SAVERT   SAVE flag routine
FF99        CTLCRT   Control C check routine
FFB8        BASINP   BASIC's Input character routine
FFE0        HOME     Home position of cursor (C1=$64 C2=$40)
FFE1        LEN      Line Length default value
FFE2        SIZE     Screen Size type (C1=0  C2=1)
FFE3-FFEA            Misc work pointer default values
FFEB-FFED   INPUT    BASIC INPUT vector
                        C1=JMP($0218)      C2=JMP $FFB8

FFEE-FFF0   OUTPUT   BASIC OUTPUT vector
                        C1=JMP($021A)      C2=JMP $FF67

FFF1-FFF3   CTRLC    Control C check vector
                        C1=JMP($021C)      C2=JMP $FF99

FFF4-FFF6   LOAD     BASIC LOAD vector
                        C1=JMP($021E)      C2=JMP $FF89

FFF7-FFF9   SAVE     BASIC SAVE vector
                        C1=JMP($0220)      C2=JMP $FF94

FFFA-FFFB   NMIVEC   NMI Vector   (= $0130 )
FFFC-FFFD   RESETV   RESET Vector  (= $FF00 )
FFFE-FFFF   IRQVEC   IRQ Vector   (= $01C0 )
```