APU-H
ARITHMETIC PROCESSOR CARD

CCM, INC. P.O. BOX 2308 RESTON, VA. 22091

Introduction

The APU-H is a high performance math processor card that adds a wide range of 16 and 32 bit fixed point and 32 bit floating point operations to the H8 system capabilities. The APU-H is well suited for virtually any application, scientific or business, where a computational capability is required.

Specifications

Fixed point operations	16 and 32 bit
Floating point operations	32 bit
Fixed point range (16 bit)	-32768 to +32767
Fixed point range (32 bit)	-2147483648 to +2147483647
Floating point range	$\pm$ (2.7×10^{-20} to 9.2×10^{18})
Card size	6.25" x 12" (standard)
Clock frequency	2 Mhz but will accept up to 8 Mhz
Power requirements	100 ma at 18v DC 250 ma at 8v DC

Circuit operation

The APU-H is built around the Advanced Micro Devices AM9511A. This processor handles 16 and 32 bit fixed point and 32 bit floating point numbers in the basic arithmetic operations of addition, subtraction, multiplication and division as well as trigonometric, logarithmic and other complex operations.

The APU-H is viewed as 2 I/O ports by the H8. The operands for an operation (e.g., 2 addends in an addition) are first placed on a data stack addressable as one of the I/O ports. The command (addition, multiplication, etc) is placed on the other port, the operation is performed by the 9511 and the result in placed on the data stack. The result may be read in or left on the stack to be used in another operation.

Two methods are possible for determining when an operation is complete. The program may query a status bit on the APU-H and read in the result when completion of the operation is indicated. Or the end indication in the 9511 may be tied to one of the interrupt lines on the H8 and an interrupt service routine may input the result.

Operation of the APU-H is now described.

The APU-H is configured to appear as port 200 (octal) for data and port 201 for commands and status. U1 functions as an address decoder, activating the chip select (CS pin on the 9511) on I/O operations to address 200 or 201. The A0, or least significant address bit is not decoded, but tied directly to the command/data (C/D) pin on the 9511. This allows the 9511 to distinguish between commands and data. U2, along with U4 and U5 also serves an address decoding function for the inverting buffers U6 and U7 allowing data to be passed from the H8 bus to the 9511.

The PAUSE signal (pin 17 on the 9511) is tied to the READY line on the H8 bus and allows the 9511 to place the 8080A in a wait state while the data from the 9511 is placed on the data bus. This occurs after completion of a 9511 operation, an event which is marked by the END signal (pin 24 on the 9511) going from high to low. The RD and WR signals for the 9511 are taken (after inversion) from the IOR and IOW signals. The clock signal for the 9511 is taken from the H8 Ø2 signal.

A typical operation, such as an addition, begins by outputting the operands (2 addends) to port 200. For each operand the data should be output in the order of least significant to most significant byte. The operands will then sit on the data stack of the 9511. The addition command is then output to port 201. It is now necessary to determine when the operation is complete. Status from an operation is available by inputting the status byte on port 201 and completion may be tested by inspecting the high order bit. When this bit is 0 the operation is complete and the result may be input or another operation initiated. The result may be input by executing the appropriate number of IN 200 commands. The most significant byte of the result will be the top byte available in the data stack of the 9511 on 200. As indicated earlier the END signal may be used to generate an interrupt on one of the H8 interrupt lines. However, an interrupt service routine such as the sample in Appendix B must be provided.

Data considerations

Three data formats are supported on the APU-H: 32 bit floating point, 32 bit fixed point and 16 bit fixed point. Operands, consisting of data in these formats, for an operation must be placed in the data stack (port 200) in the order least significant byte first, most significant byte last. After an operation, data may be retrieved from the stack in the order most significant byte first, least significant byte last. The size of the stack is such that it will accommodate 8 of the 2 byte operands and 4 of the 4 byte operands.

The fixed point data operands are signed integers in binary 2's complement notation. The most significant or high order bit is the sign and 0 represents a positive number and 1 represents negative.

For floating point operands, the 32 bit operand is broken up into a 24 bit mantissa and 8 bits for exponent and sign bits. The mantissa is normalized which means that the most significant bit of the mantissa must be a 1 except in the special case of a 0 value, where all 32 bits are 0. The least significant (right most) 7 bits of the remaining 8 bits are devoted to the exponent and its sign. The exponent is an unbiased 2's complement number having a value between -64 and +63. The sign of the mantissa occupies the remaining and most significant bit and 0 indicates positive and 1 represents negative.

Command set

The 9511 IC on the APU-H has an extensive command set. These commands can generally be broken into arithmetic (add, subtract, sine, cosine, etc) and manipulation (housekeeping such as push or pop stack, sign change, etc). We strongly recommend the programmer become familiar with the commands as explained in pages 9-21 of the enclosed 9511 manual. All commands use data previously placed in the operand (data) stack, in the top of stack (and next on stack position). The result will always be placed on the top of stack position and will have the same precision and format as the data used as the operands.

Addressing the APU-H

As indicated previously, the APU-H responds to port addresses 200/201 for data and commands, respectively. However, the address decoding logic includes jumpers to alter the port addresses if necessary.

The least significant bit of the 8 bit port address is directly connected to the C/D pin on the 9511 and, of course, is not subject to alteration or jumpering. The 7 most significant bits of the port address may be altered through the use of jumper pads below U10 and U11.

When rejumpering to make the APU-H respond to an address other than the 200/201, keep in mind that for any address, all inputs to U1 must be high for a select signal to be generated and enable the 9511 data lines. Also, remember that because of inverted buffering on the address lines of the H8 CPU board, all address signals are presented to the APU-H in an inverted state.

The jumper pads are in groups of 3 and each group of 3 is in a triangular pattern. The left most group represents the A7 line and the right most group represents the A1 line. Within each group of pads the pad at the top of the triangle will always be jumpered to 1 of the 2 pads forming the base of this triangle. Connecting to the lefthand pad passes the signal to U1 without inversion. Connecting the top pad of any group to the right hand bottom pad inverts the signal before before it reaches U1

The address furnished, 200/201, is configured as shown below:

To make the APU-H respond, for example, to the 202/203 address range, change the right most group of pads to and leave the others the same.

Basic interface

The source listings to assembly language code that allows two different interfaces with the APU-H for BASIC are provided at Appendix A.

The first interface method involves replacement of the Heath provided math routines in the Extended BASIC. This method of APU-H usage involves no differences in operation or programming aside from entering a high memory limit at BASIC initialization time. The second method utilizes the USR function of BASIC and any (not just the BASIC arithmetic operations) APU-H operation may be executed through USR. The USR method is substantially slower than the replacement method in terms of execution time.

The use of the 2 interfaces is described below. The discussion assumes a 16K memory configuration with the BASIC interface code near the top of available memory.

Replacement

This method simply "front ends" all calls to the Heath BASIC arithmetic routines and transfers control to APU-H based routines. The code at Appendix A should be keyed in or loaded. This code takes approximately 530 bytes of memory. The "patches" listed in Appendix A1 should be entered after BASIC is loaded in. Three bytes of the existing code for BASIC are included under the "Existing Code" column as a check to ensure the proper code is being overlaid.

The "patches" are provided for version 10.05.00 of the Extended BASIC. CCM will assist users in determining the appropriate locations for patches for other versions of Extended BASIC and the regular BASIC. CCM will require a cassette of the version of BASIC in question as well as a list of the utility routine entry points as generally provided in the BASIC manual. All material will be returned. Users wishing to develop their own patch lists can do so with little difficulty by using the BASIC source listing now available from Heath.

Once the patches are correctly implemented and the APU-H routines are in place, the operation of the APU-H should be transparent except for

faster execution. Error codes, though taken from the 9511, remain essentially the same as in the BASIC math routines.

A high memory limit of 23839 (decimal) is appropriate for an H8 with 16K of memory.

USR interface

This interface accommodates floating point operations and is executed through the USR and POKE functions. As before, this discussion also assumes a 16K configuration with the BASIC interface code near the top of available memory. Usage of this method requires the UINT routines as well as the other code required for the replacement method.

Floating point numbers are transmitted to the APU-H data stack as arguments to the USR function. Only 1 value can be transmitted per function execution. Commands to the APU-H are transmitted by "POKE"ing them into location 23840 (decimal) or 135040 (octal) prior to the USR execution.

There are basically 2 types of operations which are callable from BASIC: those which use 1 data operand, such as square root and exponential; and those which require 2 data operands such as addition, subtraction, etc. (Note that all succeeding examples will assume that the service request bit in the command is off).

For those command operations involving 1 data element the BASIC program should POKE the decimal value of the appropriate command (e.g., 1 for square root and 9 for natural log) into 23840. (See the 9511 manual, page 4, for a command summary.) Then execute the USR function in the form X=USR(ARG) where ARG is the data (e.g., ARG would be a 2 when the square root of 2 was desired). When control is returned to BASIC from USR, X will contain the square root of 2. ARG may be a variable or the actual value of the number to be passed. The sequence

```
POKE  23840,1
X=USR(2.00)
```

will perform the previously mentioned square root operation.

For those operations involving 2 data elements the BASIC program should POKE a 0 into location 23840 and then execute X=USR(ARG) where ARG contains the appropriate variable or data value. (Note that for certain operations a specific operand such as the dividend or minuend must be passed first. The 9511 manual contains the details on this.) With a 0 in location 23840 the interface will simply place the data in the argument on the 9511 stack. When control returns to BASIC from USR, POKE the appropriate command into 23840 and place the other data value in ARG and execute X=USR(ARG). When control returns this time

from BASIC, X will contain the desired result. For example, to execute a 32 bit floating point add of 5.22 and 6.86 perform the following sequence of instructions:

```
POKE 23840,0
X=USR(5.22)
POKE 23840,16
X=USR(6.86)
```

After the second USR call is executed X should contain the value 12.08.

To use the BASIC USR interface, key or load in the Appendix A subroutines. BASIC requires that the location of the routine to be called by execution of USR be placed in USRFCN. In version 10.05.00 of Extended BASIC, USRFCN is at 111303, but will be different for other versions of BASIC. The address of the first instruction in the BASIC interface is 135052. Place this value in 111303, least significant byte first. Therefore, the contents of 111303 will be 052135. Loading in the Appendix A routines and placing the address at USRFCN should take place after loading in BASIC but before starting it. Care should be taken to place the high memory limit for BASIC (at BASIC initialization time) below the address of the interface routines (below 135040 octal).

The BASIC interfaces were written specifically for the Heath implementations and are highly dependent on their method of representing floating point numbers. Their representation includes a mantissa that uses 2's complement notation and has 24 bits with the most significant bit being the sign bit. Other implementations of the BASIC language for the H8 may not use this structure and the furnished BASIC interfaces may not function correctly with them.

Assembly language usage

Unlike the BASIC usage of the APU-H, assembly language usage is not limited to just 32 bit floating point operations; fixed point 16 and 32 bit operations are also available.

Developing the code to use the APU-H is relatively straightforward. The numbers, or operands, must be placed on the data stack of the 9511 on the APU-H using the OUT instruction. Once the data (1 or 2 operands) has been placed on the stack the command may then be output, again using the OUT instruction.

As sold, the APU-H comes with the data stack configured as I/O port 200 and the command stack as port 201.

The following sequence of instructions illustrates assembler usage of the APU-H. Assume the address of the least significant byte of the second operand is in register pair BC. This sequence multiplies 2 16 bit

fixed point numbers:

```
        LDAX    B     LSB of multiplier
        OUT     200Q  to 9511
        DCX     B     MSB of multiplier
        LDAX    B
        OUT     200Q  to 9511
        DCX     B     LSB of multiplicand
        LDAX    B
        OUT     200Q  to 9511
        DCX     B     MSB of multiplicand
        LDAX    B
        OUT     200Q  to 9511
        MVI     A,156Q SMUL CMD
        OUT     201Q
```

The next step is to determine when the operation is complete. The 9511 contains a status register, accessible through port 201. The high order bit of the register indicates if the 9511 is busy (1=busy). The following code determines when the results may be read off the stack:

```
IN      IN      201Q  read in status
        ANI     200Q  busy?
        JNZ     IN    jmp to IN if not thru
```

The data is now ready to be removed from the stack, most significant byte first. Assume register pair BC points to the location where the most significant byte of the result is to be placed.

```
        IN      200Q  read MSB
        STAX    B     store it
        INX     B     bump to next store place
        IN      200Q  read LSB
        STAX    B
```

Appendix C contains a listing of a subroutine for performing a multiply.

Interrupts

When the 9511 has completed an operation a high to low transition occurs on one of its pins, $\overline{\text{END}}$ (pin 24). This pin may be tied to one of the interrupt lines on the H8 thus generating an interrupt every time an operation completes. In this case, the interrupt is cleared by any read or write operation.

As sold, the APU-H has all interrupt logic disabled. To provide for interrupt usage, the following steps should be taken.

Jumper pad W to $\overline{\text{END}}$
Jumper pad Y to X
Jumper pad INT to the desired interrupt line (1 to 7)

Jumper pad Z is provided to invert the interrupt transition if necessary. Jumper pad Y to pad Z to have a low to high transition generate an interrupt. Use of the service request facility (explained below) of the 9511 may require the use of the inverter.

An interrupt service routine must be furnished and a JMP instruction to it placed in the proper UIVEC location (see the H8 manual for the description and listing of the panel monitor code and UIVEC). For example, to use the interrupt number 7 with a service routine at 42300, place a 303,300,042 at location 040061.

An interrupt service routine which begins at 042300 is listed in Appendix B.

Another use of interrupts which can control operation of the APU-H involves the use of the service request and acknowledge facilities. The high order bit in the command issued to the APU-H, if turned on, causes a low to high transition to occur on the SVREC pin at the completion of an operation. The SVREC pad is available for connection to pad W to allow it to generate an interrupt. The SVREC can be cleared by driving the SVACK line low or issuing another command where the high order bit is 0.

Interrupts should only be used where the input of the results must truly be asynchronous. For those applications where the program must have the results of an operation to continue, the routines of the Assembly language section are faster. Consult the H8 manuals for further discussions of interrupts.

Other selectable features

U8, the 9511, operates at a clock frequency of 2 Mhz. The source for this signal is the Ø2 clock on the system bus. The APU-H as sold is set for 2Mhz systems. However, should an H8 CPU upgrade occur which includes higher frequencies, the jumpers in the vicinity of U9 should be reconfigured as follows:

4 Mhz	jumper E to C jumper D to A
8 Mhz	jumper E to C jumper D to B jumper F to A

Other APU-H operating considerations

The APU-H has a floating point range less than that supported by the Heath BASIC (see Specifications). In most applications this will not be a consideration, but it will impact computations involving very large numbers.

Slightly different results from those returned by the Heath BASIC may be noted when raising numbers to a power or using the LOG or EXP functions. Discussion of the accuracy of PWR (used by APU-H for exponentiation) LOG and EXP may be found in the 9511 manual under the respective headings.

References

A number of interesting and informative references exist on the AMD9511. Included are articles appearing in the April 24, 1980 issue of Electronics, the May 1978 issue of Kilobaud, and the September 1980 issue of Interface Age.

COMPONENTS

Integrated Circuits

U1	74LS30
U2	74LS00
U3-U5	74LS04
U6,U7	74LS240
U8	AMD9511A
U9	7474
U10	7805
U11	7812

Capacitors

C1,C3	2.2 ufd 35v tantalum
C2,C4	10 ufd 35v electrolytic
C5-C14	.1 ufd 25v disc ceramic

Resistors

R1	3.3K ¼watt
R2	10K ¼watt

Miscellaneous

Printed circuit board 6.25"x12"
Molex edge connectors (2) 22-15-2251
Bracket
6-32 ¼ screw (4)
6-32 nut (2)
#6 lockwasher (2)
4-40 ¼ screw (2)
4-40 nut (2)
Connector key (1)

The following code of Appendix A performs both the USR functions as well as replacing the BASIC math routines.

The tasks of each routine (or group of routines) are now described.

Routines UINT to loc 135167 are branched to by a USR call in BASIC and and set things up for the 9511 operation.

Routine STUP takes a 4 byte floating point number, loads it into ACCX and calls APU.

Routine SAVE saves and restores register contents upon entry/exit to the APU based routines (not used by USR).

Routine PUSHA saves 4 bytes in a specified work area.

Routines PWR through FPADD are APU based routines that take the place of the original BASIC routines.

Routine CMDIN reads the result from a 9511 operation and checks err status.

Routines APU through PLS perform the interfacing with the 9511 as well as reformatting data between 9511 and BASIC formats. APU through loc 137030 is the driver for this section of code.

Routine ROUND rounds the 9511 answer to the Heath BASIC precision.

```
* BASIC USR interface routine
135040    000             FLG       DB    000Q          9511 CMD area
135041    000             FLGB      DB    000Q          1st time flg
135042    000,000,000     DATAA     DB    0,0,0,0       Store 1st oprnd
135046    000,000,000     DATAB     DB    0,0,0,0       Store 2nd oprnd
135052    072,040,135     UINT      LDA   FLG           See if cmd there
135055    376,000                   CPI   000Q
135057    302,076,135               JNZ   SEC           CMD, take branch
135062    076,100                   MVI   A,100Q        Set flag to indicate
135064    062,041,135               STA   FLGB           1st oprnd there
135067    001,042,135               LXI   B,DATAA       Store oprnd
135072    315,236,135               CALL  PUSH
135075    311                       RET
135076    072,041,135     SEC       LDA   FLGB          1st or sec pass
135101    376,100                   CPI   100Q
135103    312,125,135               JE    TWO           2nd, take brnch
135106    072,040,135               LDA   FLG
135111    062,000,137               STA   137000A       Single op cmd (SQRT)
135114    315,001,137               CALL  APU
135117    076,000                   MVI   A,000Q        Clr flg before leave
135121    062,041,135               STA   FLGB
135124    311                       RET
135125    001,046,135     TWO       LXI   B,DATAB       Save 2nd oprnd
135130    315,236,135               CALL  PUSH
135133    076,000                   MVI   A,000Q        Set APU routine
135135    062,000,137               STA   137000A        for operation
135140    041,042,135               LXI   H,DATAA       Get 1st oprnd
135143    315,172,135               CALL  STUP          Put oprnd on 9511 stck
135146    072,040,135               LDA   FLG           Do second oprnd
135151    062,000,137               STA   137000A
135154    041,046,135               LXI   H,DATAB
135157    315,172,135               CALL  STUP          Do cmd
135162    076,000                   MVI   A,000Q        Clr flg
135164    062,041,135               STA   FLGB
135167    311                       RET                 Leave
```

```
* Move an operand to ACCX and call APU
135172   353              STUP        XCHG                 H to D
135173   001,066,040                  LXI   B,ACCX         Work area
135176   046,004                      MOV   H,004Q
135200   032              LDA         LDAX  D
135201   002                          STAX  B
135202   003                          INX   B
135203   023                          INX   D
135204   045                          DCR   H              Thru?
135205   302,200,135                  JNZ   LDA            No
135210   001,066,040                  LXI   B,ACCX         Repoint to wk area
135213   315,001,137                  CALL  APU
135216   311                          RET                  Thru
135217   343              SAVE        XTHL                 Save reg status
135220   325                          PUSH  D
135221   305                          PUSH  B
135222   001,227,135                  LXI   LVA            Set for future exit
135225   305                          PUSH  B
135226   351                          PCHL                 Status saved
135227   301              LVA         POP   B              Restore exit addr
135230   321                          POP   D
135231   341                          POP   H
135232   311                          RET                  Status restd for BASIC
* Save 4 bytes of data in a work area
135233   001,210,137      PUSHA       LXI   B,WORK        Get work addr
135236   021,066,040      PUSH        LXI   D,ACCX        Get accx addr
135241   046,004                      MOV   H,004Q         4 bytes
135243   032              LDAA        LDAX  D
135244   002                          STAX  B
135245   003                          INX   B
135246   023                          INX   D
135247   045                          DCR   H
135250   302,243,135                  JNZ   LDAA
135253   311                          RET
```

```
135254  325             PWR     PUSH  D           Save D
135255  315,233,135             CALL  PUSHA       Save curr ACCX
135260  341                     POP   H           Bring D to H
135261  076,000                 MOV   A,000Q      Put op on stck
135263  062,000,137             STA   137000A
135266  315,172,135             CALL  STUP
135271  041,210,137             LXI   H,WORK      Get prev ACCX
135274  076,013                 MOV   A,013Q
135276  062,000,137             STA   137000A     Store cmd
135301  315,171,135             CALL  STUP        Do PWR
135304  311                     RET               Bk to BASIC
135305  001,066,040     TAN     LXI   B,ACCX      Accx addr
135310  076,004                 MOV   A,004Q      Tan cmd
135312  062,000,137             STA   FLAG
135315  315,001,137             CALL  APU         Do tan
135320  311                     RET               Bk to BASIC
135321  001,066,040     COS     LXI   B,ACCX      Accx addr
135324  076,003                 MOV   A,003Q      Cos cmd
135326  062,000,137             STA   FLAG
135331  315,001,137             CALL  APU         Do cos
135334  311                     RET               Bk to BASIC
135335  001,066,040     SIN     LXI   B,ACCX      Accx addr
135340  076,002                 MOV   A,002Q      Sin cmd
135342  062,000,137             STA   FLAG
135345  315,001,137             CALL  APU         Do sin
135350  311                     RET               Bk to BASIC
135351  001,066,040     LOG     LXI   B,ACCX      Accx addr
135354  076,011                 MOV   A,011Q      Ln cmd
135356  062,000,137             STA   FLAG
135361  315,001,137             CALL  APU         Do ln
135364  311                     RET               Bk to BASIC
135365  001,066,040     EXP     LXI   B,ACCX      Accx addr
135370  076,012                 MOV   A,012Q      Exp cmd
135372  062,000,137             STA   FLAG
135375  315,001,137             CALL  APU         Do exp
136000  311                     RET               Bk to BASIC
136001  001,066,040     SQRT    LXI   B,ACCX      Accx addr
136004  076,001                 MOV   A,001Q      Sqr cmd
136006  062,000,137             STA   FLAG
136011  315,001,137             CALL  APU         Do sqr
136014  311                     RET               Bk to BASIC
```

```
136015  345             FPSUB   PUSH  H
136016  315,233,135             CALL  PUSHA       Put current ACCX on wk
136021  341                     POP   H
136022  076,000                 MOV   A,000Q      Indic operand on stck
136024  062,000,137             STA   FLAG
136027  315,172,135             CALL  STUP        Plac op on stck
136032  041,210,137             LXI   H,WORK      Get orig ACCX
136035  076,021                 MOV   A,021Q      Sub cmd
136037  062,000,137             STA   FLAG
136042  315,172,135             CALL  STUP        Do sub
136045  311                     RET
136046  001,066,040     ATN     LXI   B,ACCX      Accum Addr
136051  076,007                 MVI   A,007Q      Atan cmd
136053  062,000,137             STA   FLAG        Store cmd
136056  315,001,137             CALL  APU         Do atan
136061  311                     RET               Leave
136062  000                     NOP
136063  001,066,040     FPMUL   LXI   B,ACCX      Accx addr
136066  345                     PUSH  H           Save H
136067  076,000                 MOV   A,000Q      Multiplicand
136071  062,000,137             STA   FLAG         on 9511
136074  315,001,137             CALL  APU
136077  076,022                 MOV   A,022Q      Mul cmd
136101  062,000,137             STA   FLAG
136104  341                     POP   H
136105  315,172,135             CALL  STUP        Do mul
136110  311                     RET               Bk to BASIC
136111  345             FPDIV   PUSH  H           Save H
136112  001,066,040             LXI   B,ACCX      Get Accx addr
136115  076,000                 MOV   A,000Q
136117  062,000,137             STA   FLAG        Put operand on 9511
136122  315,001,137             CALL  APU
136125  341                     POP   H           Restore H
136126  076,023                 MOV   A,023Q      Div cmd
136130  062,000,137             STA   FLAG
136133  315,172,135             CALL  STUP        Do div
136136  311
136137  001,066,040     FPADD   LXI   B,ACCX      Accx addr
136142  345                     PUSH  H           Save H
136143  076,000                 MOV   A,000Q      Put
136145  062,000,137             STA   FLAG          Addend on
136150  315,001,137             CALL  APU            9511
136153  076,020                 MOV   A,020Q      Add cmd
136155  062,000,137             STA   FLAG
136160  341                     POP   H
136161  315,172,135             CALL  STUP        Do add
136164  311                     RET               Bk to BASIC
```

```
* This code rounds 9511 answer before passing it back to BASIC
136165   346,001          ROUND      ANI   001Q          Need round?
136167   312,235,136                 JZ    NORD          No
136172   140                         MOV   H,B           Set
136173   151                         MOV   L,C            to use H and L
136174   006,002                     MVI   B,002Q        Set to
136176   076,001                     MVI   A,001Q         Add 001
136200   206                         ADD   M              In lsb
136201   167              RN         MOV   M,A
136202   043                         INX   H
136203   076,000                     MOV   A,000Q        Propogate
136205   216                         ADC   M              Carry
136206   005                         DCR   B               If there
136207   302,201,136                 JNZ   RN               Thru 3 bytes
136212   167                         MOV   M,A
136213   334,243,136                 CC    SO             Leave if carry on
136216   104                         MOV   B,H           Back to BC rp
136217   115                         MOV   C,L
136220   067              SHF        STC                 Clear carry
136221   077                         CMC
136222   037                         RAR                 Back to 23 bits
136223   002                         STAX  B             Save first byte
136224   013                         DCX   B
136225   012                         LDAX  B
136226   037                         RAR                 Next byte
136227   002                         STAX  B
136230   013                         DCX   B
136231   012                         LDAX  B
136232   037                         RAR                 Next byte
136233   002                         STAX  B
136234   311                         RET                 Leave
136235   003              NORD       INX   B             No rnd requ'd; set
136236   003                         INX   B
136237   012                         LDAX  B
136240   303,220,136                 JMP   SHF
136243   043              SO         INX   H             This code handles
136244   176                         MOV   A,M            Mantissa ovfl caused
136245   346,100                     ANI   100Q            By round
136247   064                         INR   M
136250   206                         ADD   M
136251   346,100                     ANI   100Q          Ovflo?
136253   302,220,070                 JNZ   ERR.OV
136256   053                         DCX   H             Set to ret to mainline
136257   066,200                     MVI   M,200Q
136261   076,200                     MVI   A,200Q
136263   311                         RET
.
.        Patch area
136277
```

```
136300   333,201          CMDIN      IN     201Q          Read status
*This code determines when APU op complete and checks for errors
136302   147                         MOV    H,A           Save A
136303   346,200                     ANI    200Q          Busy?
136305   302,300,136                 JNZ    CMDIN         No
136310   174                         MOV    A,H           Restore a
136311   376,000                     CPI    000Q          Err free?
136313   310                         RZ                   Yes
136314   346,077                     ANI    077Q          Only err bits
136316   376,020                     CPI    020Q          Div by 0
136320   312,161,070                 JZ     ERR.DD        Yes
136323   376,010                     CPI    010Q          Invalid num?
136325   312,166,070                 JZ     ERR.IN        Yes
136330   376,030                     CPI    030Q          Ovrflw?
136332   312,220,070                 JZ     ERR.OV        Yes
136335   346,002                     ANI    002Q          Ovrflw?
136337   302,220,070                 JNZ    ERR.OV        Yes
136342   174                         MOV    A,H           Restore a
136343   346,004                     ANI    004Q          Undrflw?
136345   310                         RZ                   No
136346   001,000,000                 LXI    B,0           Yes, make 0
136351   120                         MOV    D,B
136352   130                         MOV    E,B
136353   041,066,040                 LXI    H,ACCX
136356   163                         MOV    M,E           Move in 0
136357   043                         INX    H             Nxt
136360   162                         MOV    M,D
136361   043                         INX    H
136362   161                         MOV    M,C
136363   043                         INX    H             Nxt
136364   160                         MOV    M,B
136365   341                         POP    H             Simulate ret
136366   303,030,137                 JMP    EN            Leave
137000   000              FLAG       DB     000Q
137001   315,035,137      APU        CALL   NEGA          Chk 2's compl
137004   315,066,137                 CALL   NFRM          Output operands
137007   072,000,137                 LDA    FLAG          Chk for presence
137012   376,000                     CPI    000Q           Of cmd
137014   310                         RZ                   No cmd, leave
137015   323,201                     OUT    201Q          Output cmd
137017   315,300,136                 CALL   CMDIN         Wait to finish
137022   315,115,137                 CALL   NFRA          Read in data result
137025   315,145,137                 CALL   NNGA          Chk for 2's compl
```

```
137030   311              EN          RET              Thru APU op, leave
137035   003              NEGA        INX   B          Bump to MSB
137036   003                          INX   B           Of mantissa
137037   012                          LDAX  B          Get MSB
137040   346,200                      ANI   200Q       Heath minus?
137042   312,057,137                  JZ    PLUS       No
137045   315,123,102                  CALL  FPNEG      Call BASIC neg
137050   003                          INX   B           To make pos
137051   012                          LDAX  B          Now at exponent
137052   366,200                      OR    200Q       Make 9511 neg
137054   002                          STAX  B          Put in accx
137055   013                          DCX   B
137056   311                          RET              Thru
137057   003              PLUS        INX   B          Handle pos case
137060   012                          LDAX  B
137061   346,177                      ANI   177Q       Conform to 9511
137063   002                          STAX  B
137064   013                          DCX   B          Leave ptr at MSB
137065   311                          RET              Leave
137066   013              NFRM        DCX   B          Get to LSB
*Upon entry to NFRM, BC points to MSB in ACCX
137067   013                          DCX   B
137070   012                          LDAX  B          Get LSB in a
*Set to shft mantissa to conform to 9511
137071   067                          STC              Set carry to 0
137072   077                          CMC
137073   027                          RAL              Left shft LSB
137074   323,200                      OUT   200Q       Put LSB on 9511 stac
137076   003                          INX   B          Next byte
137077   012                          LDAX  B
137100   027                          RAL              Left shft
137101   323,200                      OUT   200Q       Put on stack
137103   003                          INX   B          MSB
137104   012                          LDAX  B
137105   027                          RAL              Left shft
137106   323,200                      OUT   200Q       MSB on stack
137110   003                          INX   B          Now do exp
137111   012                          LDAX  B
137112   323,200                      OUT   200Q       Exp on stack
137114   311                          RET              Leave
137115   333,200          NFRA        IN    200Q       Read result
137117   002                          STAX  B          Have exp
*BC pair points to exp address in ACCX
137120   333,200                      IN    200Q       Get MSB
137122   013                          DCX   B           Of mantissa
137123   002                          STAX  B          Store in ACCX
137124   013                          DCX   B          Next to MSB
137125   333,200                      IN    200Q
```

```
137127   002                          STAX  B          Store in accx
137130   013                          DCX   B          LSB
137131   333,200                      IN    200Q
137133   002                          STAX  B          ACCX
137134   315,165,136                  CALL  ROUND      Round off
137137   003                          INX   B          Point to
137140   003                          INX   B           Exp
137141   003                          INX   B
137142   311                          RET              Leave
137145   012              NNGA        LDAX  B          Get exp
*Assume BC points to exponent of mantissa
137146   346,200                      ANI   200Q       Neg?
137150   312,174,137                  JE    PLS        No
137153   012                          LDAX  B          Get exp
137154   346,177                      ANI   177Q       Turn off 9511 sign
137156   002                          STAX  B          Store in accx
137157   346,100                      ANI   100Q       Neg exp
137161   302,170,137                  JNE   NNG        Yes
137164   012                          LDAX  B
137165   306,200                      ADI   200Q       Put into BASIC fmt
137167   002                          STAX  B
137170   315,123,102      NNG         CALL  FPNEG      Make neg
137173   311                          RET              Leave
137174   012              PLS         LDAX  B          Get exp
137175   346,100                      ANI   100Q       Neg exp?
137177   302,206,137                  JNZ   END        Yes
137202   012                          LDAX  B          No
137203   306,200                      ADI   200Q       Make BASIC fmt
137205   002                          STAX  B
137206   311              END         RET              Return to mainline
137210                    WORK        DS    4
```

BASIC 10.05.00 Replacements

MODULE	LOC	EXISTING CODE	PATCH
ATAN	064163	305,072,070	315,217,135,303,046,136
PWR	062003	315,015,100	315,217,135,303,254,135
TAN	064000	315,007,065	315,217,135,303,305,135
COS	063262	305,315,007	315,217,135,303,321,135
SIN	063254	021,336,111	315,217,135,303,335,135
LOG	062362	305,041,070	315,217,135,303,351,135
EXP	062232	305,072,070	315,217,135,303,365,135
SQRT	063115	305,315,175	315,217,135,303,001,136
FPDIV	103101	315,036,104	315,217,135,353,303,111,136
FPMUL	102144	315,036,104	315,217,135,353,303,063,136
FPSUB	102007	315,036,104	315,217,135,353,303,015,136
FPADD	101201	315,036,104	315,217,135,353,303,137,136

```
042300   365              IRP        PUSH  PSW          Save status
042301   363                         DI                 Lock others out
042302   333,200                     IN    200Q         Read MSB
042304   062,XXX,XXX                 STA                Store it
042307   333,200                     IN    200Q         Read LSB
042311   062,XXX,XXX                 STA                Store it
042314   373                         EI                 Unlock
042315   361                         POP   PSW          Restore
042316   311                         RET                Leave
```

This interrupt service routine inputs a 16 bit result after the APU-H has completed the requested operation and generated an interrupt to the CPU

```
040100   052,152,040     SMUL        LHLD   040152A        Get op addr
040103   353                         XCHG                  Use de
040104   032                         LDAX   D              Get lsb
040105   323,200                     OUT    200Q           Output
040107   033                         DCX    D              Get to msb
040110   032                         LDAX   D              Get msb
040111   323,200                     OUT    200Q           Output
040113   033                         DCX    D              Next op lsb
040114   032                         LDAX   D              Get lsb
040115   323,200                     OUT    200Q           Output
040117   033                         DCX    D              Msb
040120   032                         LDAX   D              Get msb
040121   323,200                     OUT    200Q
040123   076,156                     MVI    A,156Q         Get cmd
040125   323,201                     OUT    201Q           Output
040127   333,201         IN          IN     201Q           Now get status
040131   346,200                     ANI    200Q           Status thru?
040133   302,127,040                 JNZ    IN             No
040136   333,200                     IN     200Q           Msb of result
040140   022                         STAX   D              Store
040141   333,200                     IN     200Q           LSB of result
040143   023                         INX    D
040144   022                         STAX   D              Store it
040145   311                         RET
040146   000,000                     DB     0,0            1st oprnd
040150   000,000                     DB     0,0            2nd oprnd
040152   151,040                     DB     151Q,040Q      Pointer
```

This software performs a fixed point multiplication between 2 16 bit numbers. The least significant byte of the second number is pointed to by the value in location 040152