MATHPAC: A KIMATH SUPPLEMENT

by John Eaton
435 W. Padre #W10
Santa Barbara, CA 93105

Received: 77 Nov 28

The MOS Technology KIMATH program is an arithmetic program that handles 16 digit floating point operations. It can add, subtract, multiply and divide any two 16 digit numbers. KIMATH handles numbers in a BCD format; there is no conversion to any type of binary number. The numbers are stored in registers that are 18 bytes wide. The first byte in each register is the sign byte. Bit 7 is the sign of the mantissa and bit 6 is the sign of the exponent (0=+,1=–). The next 16 bytes contain the mantissa with one digit per byte. Each mantissa byte contains a BCD digit in the lower four bits of the byte with 0 in the upper four. The 18th byte contains the exponent in BCD which can be from 00 to 99. The entire register is in scientific notation so that the mantissa must be between 1 and 10.

The user has three registers in page 02: Rx, Ry, and Rz. When you call any operation, KIMATH will take the values from Rx and Ry and perform the operation and return the result to Rz.

KIMATH provides several routines for moving data from these registers to others in page 02. The Move routines are all labeled after the general format MVSD. The S is the source register and the D is the destination register. If you see a "JSR MVZX" it means that the number in Rz is moved into Rx.

The 18 byte format is a very inefficient way to store a large number of variables in memory. KIMATH has routines (PSTRES, PGTARG) that can store or recall a 16 digit number from user memory and only require 10 bytes per number.

KIMATH has several functions but most of them are limited to a range of 0 to 1. MATHPAC expands these functions to a far greater range and adds several other useful functions.

KIMATH is designed to increase the power of a 6502 system. Although capable of 16 digit operations, it is not able to directly drive a user's I/O device. Extra routines are required to take the power of KIMATH and give it to the user. MATHPAC does just that. It takes the user's ASCII I/O device and turns it into a scientific calculator.

Using MATHPAC

MATHPAC was designed to conform to the user's format instead of forcing the user to conform to the computer's. To use MATHPAC you simply type in commands in the form of assignment statements. For example if you type: @=1.234/56.78 followed by a carriage return then MATHPAC will figure out the value of the right side of the "=" and display it. (@ indicates display.) There are no restrictions on entering data since the program is designed to accept data in the same manner as a scientific calculator. You can use a "=" to enter a negative number and an "E" if you want to use scientific notation. An example is if you want to multiply 250 microamps by 11.75K ohms you type: @=250E–6*11.75E3. The program will respond by outputting 2.9375. With MATHPAC you can add (+), subtract (–), multiply (*), divide (/) or raise to a power (↑) any two 16 digit numbers.

Most scientific calculators have some form of memory where you can store results for later use. MATHPAC can store up to 26 sixteen digit numbers that can be identified by the letters A–Z. This is done simply by using a letter instead of the "@" in an assignment statement. Once a letter is defined it can be used in other calculations wherever a number is called for. For example if you type:

```
A=1234    cr*
B=345     cr
C=A–B     cr
@=C       cr          *carriage return
```

The computer will respond by outputting 889. It will also leave the numbers stored under the labels of A, B, And C. These letters can now be used in place of numbers in any other calculations.

Table 1. MATHPAC Functions:

Code	Address	Result
ABS	3513	Absolute value of Arg is found. No limitation on size.
ACS	343E	Arc cosine of arg is found. Result is in degrees from 0 to 180. Arg should be less than or equal to ±1.
ALG	32F9	Antilog base 10 is found. Arg should be greater than -99 and less than +100.
ASN	3454	Arcsin of arg is found. Result in degrees from -90 to +90. Arg should be less than or equal to ±1.
ATN	346D	Arc tangent of arg is found. Result is in degrees. No limit on size of arg. Result is from -90 to +90.
COS	3399	Cosine of arg(degrees) is found. No limit on size of arg.
DEG	351E	Argument in Radians is converted into degrees. No limit on the size of arg.
INV	354A	1/Arg is found. No limit on size of arg.
LOG	3210	Log base 10 of arg is found. Arg must be positive and non zero.
RAD	3526	Argument in Degrees is converted into radians. No limit on the size of arg.
SIN	3354	Sin of arg(degrees) is found. No limit on the size of arg.
SQR	3272	Square root of the absolute value of arg is found. No limit on the size of arg.
TAN	3370	Tangent of arg(degrees) is found. No limit to size of arg.

MATHPAC also has internal functions. Table 1 lists 13 functions and their ranges. You will notice that most of them are extended over the entire range of the KIMATH registers whenever possible (0 to ± 9.999999999999999 E±99). To call a function you simply type its three letter code and the argument in parenthesis. The argument can be either a number or a letter. Example:

```
@=SQR (23.45) cr
A=TAN (9.789E23) cr
B=LOG(A) cr
```

Each line typed into MATHPAC must contain one assignment per line. It can be one of three types: (1) SIMPLE assignment such as @=A, A=34 or B=A. (2) FUNCTION assignment such as B=SIN (A) (3) OPERATIVE assignment such as @=34.5/67, A=56+89, or C=A–3.

Two variable operative assignments cannot be mixed with functions on the same line. Use letters to store the results of calculations if mixed operations are required. If the program does not understand or is unable to carry out your command then it will respond with a "WHAT".

Placing MATHPAC in Your System

Your system must have at least 5K of memory in addition to I/O routines. 1K of RAM is required from 0000 to 03FF. MATHPAC itself needs 2K from 3000 to 37FF. KIMATH needs 2K from F800 to FFFF. Refer to Table 2 for a breakdown of the memory used. The entire system will work in a KIM-1 with an additional 4K of memory. The user must obtain his own copy of MOS Technology's KIMATH program and have single character input and output routines that pass data thru the accumulator.

Table 2. Memory Requirements:

0000--001C	Page zero use
0040--007F	I/O buffer for ASCII characters
0200--029A	KIMATH Page 02 requirements
0300--03FF	Number storage
3000--37FF	MATHPAC
F800--FFFF	KIMATH

All the codes used by MATHPAC are ASCII. Place the address of your character input routine in the jump command at 3600. The address of the character output routine will go in the jump command at 3603. Address 3606 must contain either an OD (carriage return) or an OA (linefeed). If your terminal does not have an automatic linefeed with carriage return then use an OA, otherwise use an OD.

Page 03 is where MATHPAC stores its data and must be cleared to 00. The amount of memory used is variable. Place a block of 11 bytes of FF where you want the memory to end. If you fill the last 11 bytes of page 03 with FF then MATHPAC will be able to store 22 sixteen digit numbers.

The byte at address 0000 must be set to 10. This sets the length of all operations to 16 digits. All functions are automatically rounded off to 8 digits and all other operations are rounded off to 14 digits. You must start the MATHPAC program at 3607.

Expanding the Functions

You may want to add some of your own special functions to MATHPAC. All functions take their argument from KIMATH's Rx register and leave the result in Rz. If you have a routine that does this then you may add it to MATHPAC by placing its starting address in TAB2. If you read through TAB1 and TAB2 you will see that there are three functions (FNA, FNB and FNC) that call the KIMATH routine MVXZ(FCFO). If you substitute your starting address for the first address of FCFO then calling FNA will call your function. If you want to get fancy and give it its own three letter code then you will have to reassemble both tables and insert your code in alphabetical order.

Extra Uses for MATHPAC

KIMATH is useful when it can be called by other programs to perform arithmetic operations. It consists of a series of routines and is useful to any of your other programs. MATHPAC has many similar uses when called on as subroutines. Tables 1 and 3 show many of the different routines that can be called by the user programs to perform operations on the KIMATH registers.

Table 3. MATHPAC support routines:

Name	Address	Result
PACKER	3000	Packs the ASCII data at ARGYL,ARGYH into Ry. No restrictions on format.
UNPACK	30F9	Converts Rz into readable number and stores it at RES,RES+1
STORE	3182	Stores Rz in memory under the ID in the accumulator. Returns with FF in accumulator if there is not enough room.
RECALL	31BB	Finds number in memory with ID in accumulator. Loads it into Ry.Sets accumulator to FF if number not in memory.
FORGET	31D8	Erases number from memory,ID from accumulator.
INT	329D	Largest interger less than or equal to Rx is found.
ONEX	350A	Rx is set to one.
PIE	3553	Ry is set equal to Pi
HEXDEC	3558	CNT (0003) is converted from a HEX number to a BCD number.
SETCON	3568	Constant from table at 37C0 is loaded into Ry. Accumulator determines which one.
CHOPIT	3575	Rz is scanned and PREC is set to cover only non zero digits. -0 is also corrected for.
PACADD	3589	Y index is added to ARGYL,ARGYH
RNDF	3597	Rx is rounded off the the lenght in the X index register.

After you use MATHPAC and KIMATH for a while you may notice a quirk in the system. If you type @=.5–0 the computer will respond with –9.5. Not quite the right answer. This is caused by an error in KIMATH that affects the subtraction of zero from a positive number that is less than one. If you have KIMATH in RAM then you can correct it by changing FCBB to DO and FCBD to FO.

Table 4. Assignment Statement Format:

@ (display) A-Z (Save in memory)	=	Simple assignment single letter or number.
		Function Arg in parenthesis can be number or letter
		Operation two variables can be either letter, number or both

```
                ; Calculator supplememnt for KIMATH
                ; see KIMATH manual for undefined labels
0000 10         N                       set to 10 or lenght
0017            PER
0018            QUADCT
0019            ID
001A            SIGN
001B            CAL1
001C            CAL2
0040            LR                      I/O buffer 64 Bytes
0300            ; Page 03 used for numeric storage.
                ; clear all bytes to 00. Set last 11
                ; bytes of page 03 ( or first 11 of
                ; page 04)to FF.

3000 20 7C FD   PACKER  JSR CLRY        routine to load  raw
3003 A2 00              LDX#00          number at (ARGYL,
3005 A0 00              LDY#00          ARGYH) into Ry.
3007 84 17              STY PER
3009 84 03              STY CNT
300B 84 1A              STY SIGN
300D B1 08              LDA(ARGYL),Y    1st character
300F C9 2B              CMP#2B          "+"
3011 F0 08              BEQ PACK1
3013 C9 2D              CMP#2D          "-"
3015 D0 07              BNE PACK2
3017 A9 80              LDA#80
3019 85 1A              STA SIGN        set sign neg
301B C8         PACK1   INY
301C B1 08              LDA(ARGYL),Y
301E C9 2E      PACK2   CMP#2E          "."
3020 D0 0F              BNE PACK4
3022 A9 40              LDA#40          decimal point found
3024 24 17              BIT PER
3026 30 05              BMI PACK3       stop counting exponent
3028 05 1A              ORA SIGN        start counting down
302A 85 1A              STA SIGN
302C 0A                 ASL A
302D 85 17      PACK3   STA PER
302F D0 EA              BNE PACK1       unconditional
3031 C9 30      PACK4   CMP#30          test for 0-9
3033 90 2F              BCC PACK8       non-digit
3035 C9 3A              CMP#3A
3037 B0 2B              BCS PACK8       non-digit
3039 24 17              BIT PER
303B 10 0D              BPL PACK5       not counting exp
303D E6 03              INC CNT
303F 70 15              BVS PACK7       counting up
3041 C9 30              CMP#30          zero?
3043 F0 D6              BEQ PACK1       place setting zero
3045 48                 PHA
3046 A9 40              LDA#40          stop counting
3048 D0 09              BNE PACK6       unconditional
304A 70 0A      PACK5   BVS PACK7       counting stopped
304C C9 30              CMP#30
304E F0 CB              BEQ PACK1       leading zero
3050 48                 PHA
3051 A9 C0              LDA#C0          start counting up
3053 85 17      PACK6   STA PER
3055 68                 PLA
3056 29 0F      PACK7   AND#0F          mask off digit
3058 9D 48 02           STA SY+1,X      store in Ry
305B E8                 INX
305C E0 11              CPX#11          16 digits?
305E 90 BB              BCC PACK1       not yet
3060 A2 10              LDX#10          clamp X to 16
3062 D0 B7              BNE PACK1       unconditional
3064 8A         PACK8   TXA             X=0?
3065 D0 04              BNE PACK9       no
3067 86 1A              STX SIGN
3069 86 03              STX CNT
306B 20 58 35   PACK9   JSR HEXDEC      convert exp to BCD
306E 8D 58 02   EXOT    STA EY
3071 B1 08              LDA(ARGYL),Y
3073 C9 45              CMP#45          "E"
3075 F0 06              BEQ EXP
3077 A5 1A              LDA SIGN
3079 8D 47 02           STA SY
307C 60                 RTS
307D A5 03      EXP     LDA CNT         old exp
307F 48                 PHA
3080 A5 1A              LDA SIGN
3082 48                 PHA
3083 29 80              AND#80          preserve man sign
3085 85 1A              STA SIGN        new sign
3087 A9 00              LDA#00
3089 85 03              STA CNT         new exp
308B C8                 INY
308C B1 08              LDA(ARGYL),Y
308E C9 2B              CMP#2B          "+"
3090 F0 0A              BEQ EXP1
3092 C9 2D              CMP#2D          "-"
3094 D0 09              BNE EXP2
3096 A9 40              LDA#40          new exp sign neg
3098 05 1A              ORA SIGN
309A 85 1A              STA SIGN
309C C8         EXP1    INY
309D B1 08              LDA(ARGYL),Y
309F C9 30              CMP#30          test for 0-9
30A1 90 15              BCC EXP3        non digit
30A3 C9 3A              CMP#3A
30A5 B0 11              BCS EXP3        non digit
30A7 29 0F              AND#0F          mask off digit
30A9 06 03              ASL CNT
30AB 06 03              ASL CNT
30AD 06 03              ASL CNT
30AF 06 03              ASL CNT         shift exponent
30B1 05 03              ORA CNT         combine with digit
30B3 85 03              STA CNT
30B5 38                 SEC
30B6 B0 E4              BCS EXP1        unconditional
30B8 F8         EXP3    SED             adjust sign and exp
30B9 68                 PLA             old sign
30BA 48                 PHA
30BB 45 1A              EOR SIGN        test signs of the
30BD 85 17              STA PER         two exp's to see
30BF 24 17              BIT PER         if they are the same
30C1 50 21              BVC EXP6        sign's same
30C3 68                 PLA             old sign
30C4 85 17              STA PER
30C6 68                 PLA             old exp
30C7 C5 03              CMP CNT
30C9 90 09              BCC EXP4        new exp gtr
30CB E5 03              SBC CNT         difference of exp's
30CD 48                 PHA             adjusted exp
30CE A5 17              LDA PER         old sign
30D0 48                 PHA             adjusted sign
30D1 38                 SEC
30D2 B0 0C              BCS EXP5        unconditional
30D4 E5 03      EXP4    SBC CNT         difference of exp's
30D6 85 03              STA CNT         compensate subtracting
30D8 A9 00              LDA#00          larger number from
30DA E5 03              SBC CNT         small by subtracting
30DC 48                 PHA             from zero
30DD A5 1A              LDA SIGN
30DF 48                 PHA             adjusted sign
30E0 A9 00      EXP5    LDA#00
30E2 85 03              STA CNT
30E4 18         EXP6    CLC
30E5 68                 PLA             sign
30E6 85 1A              STA SIGN
30E8 68                 PLA             exponent
30E9 65 03              ADC CNT
30EB 48                 PHA
30EC D8                 CLD
30ED D0 06              BNE EXP7        exp not zero
30EF A9 BF              LDA#BF
30F1 25 1A              AND SIGN
30F3 85 1A              STA SIGN
30F5 68         EXP7    PLA
30F6 4C 8E 30           JMP EXOT
30F9 A0 6A 02   UNPACK  LDA EZ          routine to unpack
30FC 85 03              STA CNT         Rz and store at
30FE 20 C3 FB           JSR DECHEX      (RES,RES+1)
3101 A0 00              LDY#00
3103 2C 59 02           BIT SZ
3106 10 05              BPL UNPAC1      positive number
3108 A9 2D              LDA#2D          "-"
310A 91 0A              STA(RES),Y
310C C8                 INY
310D A2 00      UNPAC1  LDX#00
310F A5 03              LDA CNT
3111 C9 10              CMP#10          exp gtr 15
3113 B0 3B              BCS UNPAC7      use scientific notation
3115 2C 59 02           BIT SZ
3118 50 0E              BVC UNPAC3      exp is positive
311A A9 2E              LDA#2E          decimal point
311C 91 0A              STA(RES),Y
311E A9 30              LDA#30          zero
3120 C8         UNPAC2  INY             display place setting 0's
3121 91 0A              STA(RES),Y
3123 C6 03              DEC CNT
3125 10 F9              BPL UNPAC2
3127 88                 DEY
3128 BD 5A 02   UNPAC3  LDA SZ+1,X      fetch digit
312B 09 30              ORA#30          convert to ASCII
312D 91 0A              STA(RES),Y
312F E8                 INX
3130 C8                 INY
3131 24 03              BIT CNT
3133 30 09              BMI UNPAC4
3135 C6 03              DEC CNT
3137 10 05              BPL UNPAC4
3139 A9 2E              LDA#2E          decimal point
313B 91 0A              STA(RES),Y
313D C8                 INY
313E E4 10      UNPAC4  CPX PREC        all digits moved?
3140 D0 E6              BNE UNPAC3
3142 24 03              BIT CNT
3144 30 09              BMI UNPAC6
3146 A9 30              LDA#30
3148 91 0A      UNPAC5  STA(RES),Y      trailing zero's
314A C8                 INY
314B C6 03              DEC CNT
314D 10 F9              BPL UNPAC5
314F 60         UNPAC6  RTS
3150 A9 00      UNPAC7  LDA#00          scientific notation
3152 85 03              STA CNT
3154 20 28 31           JSR UNPAC3
3157 A9 20              LDA#20          blank
3159 91 0A              STA(RES),Y
315B C8                 INY
315C A9 45              LDA#45          "E"
315E 91 0A              STA(RES),Y
3160 C8                 INY
3161 2C 59 02           BIT SZ
```

```
3164 50 05            BVC UNPAC8      positive exponent
3166 A9 2D            LDA#2D          "-"
3168 91 0A            STA(RES),Y
316A C8               INY
316B AD 6A 02  UNPAC8 LDA EZ
316E 4A               LSR A
316F 4A               LSR A
3170 4A               LSR A
3171 4A               LSR A
3172 09 30            ORA#30          convert to ASCII
3174 91 0A            STA(RES),Y
3176 C8               INY
3177 AD 6A 02         LDA EZ
317A 29 0F            AND#0F
317C 09 30            ORA#30          convert to ASCII
317E 91 0A            STA(RES).Y
3180 C8               INY
3181 60               RTS

                ; routines to store ad recall numbers.
                ; numbers are taken from Rz and stored
                ; in page 03. Numbers are recalled to Ry.
3182 20 E2 31 STORE   JSR SRCH
3185 D0 0D            BNE STOR1       ID already in memory
3187 A5 19            LDA ID
3189 48               PHA
318A A9 00            LDA#00
318C 20 E2 31         JSR SRCH        look for empty cell
318F F0 26            BEQ STOR2       no room in page 03
3191 68               PLA
3192 91 0C            STA(PTR),Y      set ID in pg 03
3194 A5 0A     STOR1  LDA RES
3196 48               PHA
3197 A5 0B            LDA RES+1
3199 48               PHA
319A A9 01            LDA#01
319C 20 04 32         JSR ADDM        add one to address
319F A5 0C            LDA PTR
31A1 85 0A            STA RES
31A3 A5 0D            LDA PTR+1
31A5 85 0B            STA RES+1
31A7 A5 00            LDA N
31A9 85 10            STA PREC
31AB 20 3C FE         JSR PSTRES      Move Rz into Pg 03
31AE 68               PLA
31AF 85 0B            STA RES+1
31B1 68               PLA
31B2 85 0A            STA RES
31B4 A5 19            LDA ID
31B6 60               RTS
31B7 68        STOR2  PLA
31B8 A9 FF            LDA#FF          No room in pg 3
31BA 60               RTS
31BB 20 E2 31  RECALL JSR SRCH
31BE F0 17            BEQ RECAL1       not in memory
31C0 A9 01            LDA#01
31C2 20 04 32         JSR ADDM         add one to address
31C5 A5 00            LDA N            recall number into Ry
31C7 4A               LSR A
31C8 69 01            ADC#01
31CA 85 04            STA LENGHT
31CC 20 87 FD         JSR CLRZ
31CF 20 E1 FD         JSR PGTARG
31D2 20 10 FD         JSR MVZY
31D5 A5 19            LDA ID
31D7 60        RECAL1 RTS
31D8 20 E2 31  FORGET JSR SRCH
31DB F0 04            BEQ FORGE1
31DD A9 00            LDA#00
31DF 91 0C            STA(PTR),Y
31E1 60        FORGE1 RTS
31E2 D8        SRCH   CLD              search page 03 for
31E3 85 19            STA ID           ID or FF
31E5 A0 00            LDY#00
31E7 A9 02            LDA#02
31E9 85 0D            STA PTR+1
31EB A9 F5            LDA#F5
31ED 85 0C            STA PTR
31EF 20 FF 31  SRCH1  JSR ADDL
31F2 B1 0C            LDA(PTR),Y
31F4 C5 19            CMP ID
31F6 F0 04            BEQ SRCH2
31F8 C9 FF            CMP#FF
31FA D0 F3            BNE SRCH1
31FC C9 FF     SRCH2  CMP#FF
31FE 60               RTS
31FF A5 00     ADDL   LDA N            Add  lenght to address
3201 4A               LSR A
3202 69 03            ADC#03
3204 18        ADDM   CLC              add A to address
3205 65 0C            ADC PTR
3207 85 0C            STA PTR
3209 A9 00            LDA#00
320B 65 0D            ADC PTR+1
320D 85 0D            STA PTR+1
320F 60               RTS

                ; LOG base 10 of Rx is found and stored
                ; in Rz. Rx must be positive and non zero
3210 A5 00     LOGT   LDA N
3212 48               PHA              save lenght
3213 AD 35 02         LDA SX
3216 48               PHA              save sign
3217 AD 46 02         LDA EX

321A 48               PHA              save exponent
321B A9 00            LDA#00
321D 8D 35 02         STA SX
3220 8D 46 02         STA EX
3223 A9 09            LDA#09
3225 20 68 35         JSR SETCON       Ry=1/SQR(10)
3228 20 0B F9         JSR MUL
322B 20 0C FD         JSR MVZX
322E 20 E7 FA         JSR LOG
3231 20 0C FD         JSR MVZX
3234 20 7C FD         JSR CLRY
3237 A9 05            LDA#05
3239 8D 49 02         STA SY+2          Ry=+.5
323C 20 08 F8         JSR ADD
323F 20 0C FD         JSR MVZX
3242 20 7C FD         JSR CLRY
3245 68               PLA              exponent
3246 C9 10            CMP#10
3248 B0 09            BCS LOGT1        exp gtr 9
324A 29 0F            AND#0F
324C 8D 48 02         STA SY+1
324F A9 00            LDA#00
3251 F0 10            BEQ LOGT2        unconditional
3253 48        LOGT1  PHA
3254 4A               LSR A
3255 4A               LSR A
3256 4A               LSR A
3257 4A               LSR A
3258 8D 48 02         STA SY+1
325B 68               PLA
325C 29 0F            AND#0F
325E 8D 49 02         STA SY+2
3261 A9 01            LDA#01
3263 8D 58 02  LOGT2  STA EY           Ry now contains exp
3266 68               PLA
3267 0A               ASL A            adjust sign
3268 8D 47 02         STA SY
326B 20 08 F8         JSR ADD
326E 68               PLA
326F 85 00            STA N            lenght
3271 60               RTS
3272 20 13 35  SQRT    JSR ABS          square root routine
3275 20 A6 FC         JSR XZTST
3278 D0 01            BNE SQRT1
327A 60               RTS
327B 20 18 FD  SQRT1  JSR MVZN
327E 20 14 FD         JSR MVZM
3281 AD 46 02         LDA EX
3284 85 03            STA CNT
3286 20 C3 FB         JSR DECHEX       exp now hex
3289 4A               LSR A            divide by two
328A D0 02            BNE SQRT2
328C A9 01            LDA#01
328E 85 03     SQRT2  STA CNT
3290 20 58 35         JSR HEXDEC       exp now BCD
3293 8D 7C 02         STA EM
3296 A9 07            LDA#07
3298 85 01            STA NKON
329A 4C B5 FA         JMP SQRT0
                ; routine to find the largest interger
                ; less than or equal to Rx.
329D A5 00     INT    LDA N
329F 48               PHA              save lenght
32A0 AD 35 02         LDA SX
32A3 48               PHA              save sign
32A4 29 7F            AND#7F
32A6 8D 35 02         STA SX           set positive
32A9 20 F4 FC         JSR MVXM
32AC 2C 35 02         BIT SX
32AF 50 03            BVC INT1         Rx gtr than one
32B1 20 71 FD         JSR CLRX         Rx=0
32B4 AD 46 02  INT1   LDA EX
32B7 C9 15            CMP#15
32B9 90 02            BCC INT2         exp less 15
32BB A9 15            LDA#15
32BD 85 03     INT2   STA CNT
32BF 20 C3 FB         JSR DECHEX       exp now hex
32C2 85 00            STA N
32C4 E6 00            INC N
32C6 20 7C FD         JSR CLRY
32C9 20 87 FD         JSR CLRZ
32CC 20 08 F8         JSR ADD
32CF 68               PLA              sign
32D0 10 23            BPL INT4
32D2 20 0C FD         JSR MVZX
32D5 20 20 FD         JSR MVMY
32D8 A9 10            LDA#10
32DA 85 00            STA N
32DC 20 00 F8         JSR SUB
32DF 20 0C FD         JSR MVZX
32E2 20 00 FD         JSR MVYZ
32E5 20 A6 FC         JSR XZTST
32E8 F0 06            BEQ INT3
32EA 20 0A 35         JSR ONEX
32ED 20 08 F8         JSR ADD
32F0 A9 80     INT3   LDA#80
32F2 8D 59 02         STA SZ
32F5 68        INT4   PLA
32F6 85 00            STA N
32F8 60               RTS
                ; antilog base 10 routine. Rx must be
                ; gtr than -99 and less than +100
32F9 2C 35 02  ALOG   BIT SX
```

```
32FC 70 12          BVS ALOG2       Rx less than 1
32FE A5 46 02       LDA EX
3301 C9 02          CMP#02
3303 90 0B          BCC ALOG2       Exp less 2
3305 20 D2 FC ALOG1 JSR INFIN
3308 A5 35 02       LDA SX
330B 4A             LSR A
330C 8D 59 02       STA SZ
330F 60             RTS
3310 20 F8 FC ALOG2 JSR MVXN
3313 20 9D 32       JSR INT
3316 20 0C FD       JSR MVZX
3319 AE 6A 02       LDX EZ
331C E0 02          CPX#02
331E F0 E5          BEQ ALOG1       X=-100
3320 A5 00          LDA N
3322 48             PHA
3323 B5 59 02       LDA SZ,X
3326 0A             ASL A
3327 0A             ASL A
3328 0A             ASL A
3329 0A             ASL A
332A 1D 5A 02       ORA SZ+1,X
332D 85 17          STA PER
332F 48             PHA             save exponent
3330 A5 59 02       LDA SZ
3333 4A             LSR A           adjust sign
3334 48             PHA             save sign
3335 20 2C FD       JSR MVNX
3338 A5 17          LDA PER
333A F0 09          BEQ ALOG3       exp=00
333C 20 10 FD       JSR MVZY
333F 20 00 F8       JSR SUB
3342 20 0C FD       JSR MVZX
3345 20 41 FB ALOG3 JSR TENX
3348 68             PLA
3349 8D 59 02       STA SZ
334C 68             PLA
334D 8D 6A 02       STA EZ
3350 68             PLA
3351 85 00          STA N
3353 60             RTS
3354 20 A8 33 SIN   JSR TRIG5       SIN(Rx) found and
3357 20 08 F8       JSR ADD         placed in Rz
335A 20 76 33 TRIG1 JSR TRIG4
335D A5 18    TRIG2 LDA QUADCT
335F F0 0C          BEQ TRIG3
3361 C9 03          CMP#03
3363 F0 08          BEQ TRIG3
3365 A5 59 02       LDA SZ
3368 49 80          EOR#80
336A 8D 59 02       STA SZ
336D 4C 75 35 TRIG3 JMP CHOPIT
3370 20 A8 33 TAN   JSR TRIG5       TAN(Rx) found and
3373 20 00 F8       JSR SUB         placed in Rz
3376 20 10 FD TRIG4 JSR MVZY
3379 20 1C FD       JSR MVMX
337C 20 16 FA       JSR DIVIDE
337F A5 6A 02       LDA EZ
3382 C9 06          CMP#06
3384 90 E7          BCC TRIG3
3386 2C 59 02       BIT SZ
3389 70 E2          BVS TRIG3
338B A5 59 02       LDA SZ
338E 48             PHA
338F 20 D2 FC       JSR INFIN
3392 68             PLA
3393 8D 59 02       STA SZ
3396 4C 75 35       JMP CHOPIT
3399 20 A8 33 COS   JSR TRIG5       COS(Rx) found and
339C 20 00 F8       JSR SUB         placed in Rz
339F 20 14 FD       JSR MVZM
33A2 20 00 F8       JSR SUB
33A5 4C 5A 33       JMP TRIG1
33A8 A9 FF    TRIG5 LDA#FF          Rx can be any value
33AA 85 18          STA QUADCT
33AC 2C 35 02 TRIG6 BIT SX
33AF 30 0C          BMI TRIG7
33B1 20 18 34       JSR Y360        angle is pos
33B4 20 00 F8       JSR SUB
33B7 20 0C FD       JSR MVZX
33BA 4C AC 33       JMP TRIG6
33BD 20 18 34 TRIG7 JSR Y360        angle is neg
33C0 20 08 F8       JSR ADD
33C3 20 0C FD       JSR MVZX
33C6 2C 35 02       BIT SX
33C9 30 F2          BMI TRIG7
33CB 20 13 34 TRIG8 JSR Y90
33CE 20 00 F8       JSR SUB
33D1 20 0C FD       JSR MVZX
33D4 E6 18          INC QUADCT
33D6 2C 35 02       BIT SX
33D9 10 F0          BPL TRIG8
33DB A5 18          LDA QUADCT      angle between -90 and 0
33DD 4A             LSR A
33DE B0 09          BCS TRIG9
33E0 20 13 34       JSR Y90
33E3 20 08 F8       JSR ADD
33E6 20 0C FD       JSR MVZX
33E9 20 13 34 TRIG9 JSR Y90
33EC 20 16 FA       JSR DIVIDE
33EF 20 0C FD       JSR MVZX
33F2 A5 00          LDA N
33F4 48             PHA
33F5 20 5C FB       JSR TANX        Rz=TAN(X/2)
33F8 68             PLA
33F9 85 00          STA N
33FB 20 0C FD       JSR MVZX
33FE 20 10 FD       JSR MVZY
3401 20 0B F9       JSR MUL
3404 20 14 FD       JSR MVZM
3407 20 08 F8       JSR ADD
340A 20 20 FD       JSR MVMY
340D 20 14 FD       JSR MVZM
3410 4C 0A 35       JMP ONEX
3413 A9 1E    Y90   LDA#1E
3415 4C 68 35       JMP SETCON
3418 20 7C FD Y360  JSR CLRY
341B A9 03          LDA#03
341D 8D 48 02       STA SY+1
3420 A9 06          LDA#06
3422 8D 49 02       STA SY+2
3425 2C 35 02       BIT SX
3428 70 0D          BVS Y360A
342A F8             SED
342B AD 46 02       LDA EX
342E F0 07          BEQ Y360A
3430 38             SEC
3431 E9 01          SBC#01
3433 C9 02          CMP#02
3435 B0 02          BCS Y360B
3437 A9 02    Y360A LDA#02
3439 8D 58 02 Y360B STA EY
343C D8             CLD
343D 60             RTS
                    ; arctrig routines give results in degrees
343E 20 C9 34 ACOS  JSR ARCSET
3441 2C 47 02       BIT SY
3444 10 14          BPL ASIN1       angle in 1st quad
3446 20 5A 34       JSR ASIN1       angle in 2nd quad
3449 20 0C FD       JSR MVZX
344C A9 1B          LDA#1B
344E 20 68 35       JSR SETCON      Ry=180
3451 4C 08 F8       JMP ADD
3454 20 C9 34 ASIN  JSR ARCSET
3457 20 BF FC       JSR XSY
345A AD 35 02 ASIN1 LDA SX
345D 29 80          AND#80
345F 48             PHA
3460 20 16 FA       JSR DIVIDE
3463 20 0C FD       JSR MVZX
3466 68             PLA
3467 0D 35 02       ORA SX
346A 8D 35 02       STA SX
346D A5 00    ATAN  LDA N
346F 48             PHA
3470 AD 35 02       LDA SX
3473 48             PHA
3474 29 7F          AND#7F
3476 8D 35 02       STA SX
3479 20 BF FC       JSR XSY
347C 20 0A 35       JSR ONEX
347F 20 08 F8       JSR ADD
3482 20 BF FC       JSR XSY
3485 20 16 FA       JSR DIVIDE
3488 20 0C FD       JSR MVZX
348B 20 A6 FC       JSR XZTST
348E F0 26          BEQ ATAN2
3490 2C 35 02       BIT SX
3493 50 0A          BVC ATAN1
3495 AD 46 02       LDA EX
3498 D0 05          BNE ATAN1
349A A9 99          LDA#99
349C 8D 46 02       STA EX
349F 20 78 FB ATAN1 JSR ATANX
34A2 68             PLA
34A3 48             PHA
34A4 29 40          AND#40
34A6 D0 0E          BNE ATAN2
34A8 20 0C FD       JSR MVZX
34AB A9 12          LDA#12
34AD 20 68 35       JSR SETCON      Ry=Pi/2
34B0 20 BF FC       JSR XSY
34B3 20 00 F8       JSR SUB
34B6 68       ATAN2 PLA             sign
34B7 29 80          AND#80
34B9 0D 59 02       ORA SZ
34BC 8D 59 02       STA SZ
34BF 20 0C FD       JSR MVZX
34C2 20 1E 35       JSR DEG         convert to degrees
34C5 68             PLA
34C6 85 00          STA N
34C8 60             RTS
34C9 2C 35 02 ARCSET BIT SX
34CC 70 17          BVS ARC2        Rx less one
34CE AD 36 02       LDA SX+1
34D1 48             PHA
34D2 AD 35 02       LDA SX
34D5 48             PHA
34D6 20 71 FD       JSR CLRX
34D9 68             PLA
34DA 8D 35 02       STA SX
34DD 68             PLA
34DE F0 02          BEQ ARC1
34E0 A9 01          LDA#01
34E2 8D 36 02 ARC1  STA SX+1
```

```
34E5 20 EC FC   ARC2    JSR MVXY        -1 ls X ls +1
34E8 20 F0 FC           JSR MVXZ
34EB A9 01              LDA#01
34ED 20 82 31           JSR STORE
34F0 20 0B F9           JSR MUL         X^2
34F3 20 10 FD           JSR MVZY
34F6 20 0A 35           JSR ONEX
34F9 20 00 F8           JSR SUB         1-X^2
34FC 20 0C FD           JSR MVZX
34FF 20 72 32           JSR SQRT        SQR(1-X^2)
3502 20 0C FD           JSR MVZX
3505 A9 01              LDA#01
3507 4C BB 31           JMP RECALL       Ry * ARG
                ;
350A 20 71 FD   ONEX    JSR CLRX
350D A9 01              LDA#01
350F 8D 36 02           STA RX+1        Rx=1.000
3512 60                 RTS
                ;
3513 AD 35 02   ABS     LDA SX          Absolute value
3516 29 7F              AND#7F
3518 8D 35 02           STA SX
351B 4C F0 FC           JMP MVXZ
                ;
351E A9 00      DEG     LDA#00          convert to deg
3520 20 68 35           JSR SETCON      Pi/180
3523 4C 16 FA           JMP DIVIDE
                ;
3526 A9 00      RAD     LDA#00          convert to rad
3528 20 68 35           JSR SETCON      Pi/180
352B 4C 0B F9           JMP MUL
                ;
352E 20 00 FD   XRY     JSR MVYZ        raise Rx to Ry
3531 A9 01              LDA#01
3533 20 82 31           JSR STORE
3536 20 10 32           JSR LOGT
3539 20 0C FD           JSR MVZX
353C A9 01              LDA#01
353E 20 BB 31           JSR RECALL
3541 20 0B F9           JSR MUL
3544 20 0C FD           JSR MVZX
3547 4C F9 32           JMP ALOG
                ;
354A 20 EC FC   INV     JSR MVXY        find 1/Rx
354D 20 0A 35           JSR ONEX
3550 4C 16 FA           JMP DIVIDE
                ;
3553 A9 21      PIE     LDA#21          set Ry=Pi
3555 4C 68 35           JMP SETCON
                ;
3558 F8         HEXDEC  SED             convert CNT from
3559 E6 03              INC CNT         HEX to BCD
355B A9 99              LDA#99
355D 18         HEX1    CLC
355E 69 01              ADC#01
3560 C6 03              DEC CNT
3562 D0 F9              BNE HEX1
3564 85 03              STA CNT
3566 D8                 CLD
3567 60                 RTS
                ;
3568 85 01      SETCON  STA NKON        load constant in Ry
356A A9 C0              LDA#C0
356C 85 0E              STA KON
356E A9 37              LDA#37
3570 85 0F              STA KONH
3572 4C 92 FD           JMP LOOKUP
                ;
3575 A6 00      CHOPIT  LDX N           remove unnedded 0's
3577 Bd 59 02   CHOP1   LDA SZ,X        by adjusting PREC
357A D0 0A              BNE CHOP2
357C CA                 DEX
357D D0 F8              BNE CHOP1
357F 8E 59 02           STX SZ          man=0,clear sign,exp
3582 8E 6A 02           STX EZ
3585 E8                 INX
3586 86 10      CHOP2   STX PREC
3588 60                 RTS
                ;
3589 98         PACADD  TYA             add Y to ARGY
358A D8                 CLD
358B 18                 CLC
358C 65 08              ADC ARGYL
358E 85 08              STA ARGYL
3590 A9 00              LDA#00
3592 65 09              ADC ARGYH
3594 85 09              STA ARGYH
3596 60                 RTS
3597 A5 00      RNDF    LDA N           round off routine
3599 48                 PHA             round off to X
359A A9 10              LDA#10
359C 85 00              STA N
359E 2C 35 02           BIT SX
35A1 70 05              BVS RNDF1
35A3 CD 46 02           CMP EX
35A6 90 43              BCC RNDF3
35A8 AD 35 02   RNDF1   LDA SX
35AB 48                 PHA
35AC 29 7F              ANF#7F
35AE 8D 35 02           STA SX
35B1 8A                 TXA
35B2 48                 PHA
35B3 20 EC FC           JSR MVXY
35B6 20 0A 35           JSR ONEX
35B9 68                 PLA
35BA 48                 PHA
35BB AA                 TAX
35BC A9 05              LDA#05
35BE 9D 37 02           STA SX+2,X
35C1 20 08 F8           JSR ADD
35C4 20 10 FD           JSR MVZY
35C7 20 87 FD           JSR CLRZ
35CA 20 0A 35           JSR ONEX
35CD 20 BF FC           JSR XSY
35D0 68                 PLA
35D1 AA                 TAX
35D2 E8                 INX
35D3 86 00              STX N
35D5 20 00 F8           JSR SUB
35D8 20 0C FD           JSR MVZX
35DB 20 A6 FC           JSR XZTST
35DE F0 07              BEQ RNDF2
35E0 68                 PLA
35E1 48                 PHA
35E2 29 80              AND#80
35E4 0D 35 02           ORA SX
35E7 8D 35 02   RNDF2   STA SX
35EA 68                 PLA
35EB 20 F0 FC   RNDF3   JSR MVXZ
35EE 68                 PLA
35EF 85 00              STA N
35F1 60                 RTS
3600 4C 00 00   INVEC   JMP CHARIN      user input routine
3603 4C 00 00   OTVEC   JMP CHAROT      user output routine
3606 0D         ECHO    BYTE 0D         echo character
3607 A9 01      SCICAL  LDA#01          START OF ROUTINE
3609 85 1B              STA CAL1
360B C6 1B      BACK    DEC CAL1        backspace routine
360D 20 00 36   LOOP1   JSR INVEC
3610 C9 08              CMP#08          backspace?
3612 F0 F7              BEQ BACK        yes
3614 A6 1B              LDX CAL1        X points to open cell
3616 95 40              STA LR,X        store chars at 0040
3618 E6 1B              INC CAL1
361A C9 0D              CMP#0D          carriage return?
361C D0 EF              BNE LOOP1
361E AD 06 36           LDA ECHO
3621 20 03 36           JSR OTVEC       Echo character
3624 A5 40              LDA LR          assignment char
3626 48                 PHA
3627 A9 40              LDA#40
3629 85 08              STA ARGYL
362B 85 0A              STA RES
362D A9 00              LDA#00
362F 85 09              STA ARGYH
3631 85 0B              STA RES+1
3633 A0 02              LDY#02
3635 20 0C 37           JSR LOAD
3638 B0 12              BCS HAV1        number loaded
363A A5 43              LDA LR+3        letter found,test function
363C 20 1B 37           JSR LTRTST
363F 90 6D              BCC FUNCTN      function found
3641 A5 42              LDA LR+2
3643 20 BB 31           JSR RECALL      fetch number into Ry
3646 C9 FF              CMP#FF
3648 F0 17              BEQ WHATC       number not in memory
364A A0 03              LDY#03
364C 20 FC FC    HAV1   JSR MVYX
364F B1 08              LDA(ARGYL),Y
3651 48                 PHA             operation
3652 C9 0D              CMP#0D          carriage return
3654 F0 0D              BEQ OPS
3656 C8                 INY
3657 20 0C 37           JSR LOAD
365A B0 07              BCS OPS
365C 20 BB 31           JSR RECALL
365F C9 FF              CMP#FF
3661 F0 20      WHATC   BEQ WHAT
3663 68         OPS     PLA
3664 20 25 37           JSR OPERT       two number op
3667 20 0C FD           JSR MVZX
366A A6 00              LDX N
366C CA                 DEX
366D CA                 DEX
366E 20 97 35   OUT     JSR RNDF        result in Rz
3671 20 75 35           JSR CHOPIT      remove unwanted zero's
3674 68                 PLA             assignment
3675 C9 40              CMP#40          @?
3677 F0 0E              BEQ OUT1        display result
3679 20 1B 37           JSR LTRTST      assignment a letter?
367C B0 7B              BCS WHAT        non letter
367E 20 82 31           JSR STORE       save result
3681 C9 FF              CMP#FF
3683 F0 74      WHATD   BEQ WHAT        no room in pg 03
3685 D0 80              BNE SCICAL      unconditional
3687 20 F9 30   OUT1    JSR UNPACK      display Rz
368A A9 0D              LDA#0D          car ret
368C 91 0A              STA(RES),Y
368E C8                 INY
368F AD 06 36           LDA ECHO        echo character
3692 91 0A              STA(RES),Y
3694 A2 00      DISP    LDX#00
3696 86 1B              STX CAL1
3698 A6 1B      DISP1   LDX CAL1
369A B5 40              LDA LR,X
369C CD 06 36           CMP ECHO         last character?
```

MICROSOFT-MITS EXCLUSIVE LICENSE TERMINATED

News Release *Received: 77 Nov 23*

Microsoft's BASIC for the 8080 and Z-80, the first resident high-level language for a microprocessor, is now generally available on both a single copy and OEM basis. The BASIC became the subject of an extended legal dispute which resulted in the termination of an exclusive license to MITS, Inc.

The BASIC, best known in the field as Altair[tm] BASIC has been in use for 1½ years and has a user base of over 5000. Several software firms offer applications software written in Microsoft's BASIC. Current OEM users of the BASIC include General Electric, National Cash Register, Applied Digital Data Systems, Radio Shack and Data Terminals and Communications.

FIX FOR ELDERLY EDITOR/ASSEMBLER

Dear Dr. Dobbs, *Dated: 77 Sep 6*

Here is more data on that outstanding piece of junk, the M-T free Editor-Assembler. My version is the one distributed to clubs by Processor Technology about two years ago. I have found PCHL assembles as DF, not E9. The fix for that is easy; change the code in the table. In my listing it is location F978.

I have also found that the pseudo-op DB is counted as three bytes in the first pass; this makes all values in the symbol table be off by 2 x n (where n is the number of DB statements preceding the symbol). The fix for that is more subtle. In locations F78D (DATA1), change JMP OPZ to JMP patch (a four-byte area). At patch, put XRA A (AF), JMP OPZ (C3 6E FA).

Jim Kaufman
2890 15th St.
Boulder, CO 80302

```
369F F0 07              BEQ DISP2       yes
36A1 20 03 36           JSR OTVEC
36A4 E6 1B              INC CAL1
36A6 D0 F0              BNE DISP1       unconditional
36A8 20 03 36   DISP2   JSR OUTVEC
36AB 4C 07 36           JMP SCICAL
36AE A0 00      FUNCTN  LDY#00          function found
36B0 A2 00              LDX#00
36B2 20 E4 36           JSR LOOK        match 1st letter
36B5 20 E4 36           JSR LOOK        match 2nd letter
36B8 20 E4 36           JSR LOOK        match 3rd letter
36BB B9 86 37           LDA TAB2-1,Y    Add Hi byte
36BE 85 1C              STA CAL2
36C0 B9 85 37           LDA TAB2-2,Y    Add Lo byte
36C3 85 1B              STA CAL1
36C5 A0 06              LDY#06
36C7 20 0C 37           JSR LOAD        load arg
36CA B0 07              BCS FUN1
36CC 20 BB 31           JSR RECALL
36CF C9 FF              CMP#FF
36D1 F0 26              BEQ WHAT        number not in mem
36D3 20 FC FC   FUN1    JSR MVYX
36D6 20 E1 36           JSR FUN         perform function
36D9 20 0C FD           JSR MVZX
36DC A2 08              LDX#08          round off to 8 digits
36DE 4C 6E 36           JMP OUT         display result
36E1 6C 1B 00   FUN     JMP(CAL1)
36E4 B9 4B 37   LOOK    LDA TAB1,Y      compare letter to table
36E7 D5 42              CMP LR+2,X
36E9 F0 0B              BEQ FOUND
36EB C9 FF              CMP#FF          end of table
36ED F0 05              BEQ NTFND
36EF C8                 INY             next position
36F0 C8                 INY
36F1 C8                 INY
36F2 D0 F0              BNE LOOK
36F4 F0 03      NTFND   BEQ WHAT        function not there
36F6 E8         FOUND   INX
36F7 C8                 INY
36F8 60                 RTS
36F9 A2 05      WHAT    LDX#05          output "WHAT"
36FB BD 06 37   WHAT1   LDA WHAT2,X
36FE 95 40              STA LR,X
3700 CA                 DEX
3701 10 F8              BPL WHAT1
3703 4C 94 36           JMP DISP
3706 57 48 41           BYTE 57 48 41   "WHAT cr lf"
3709 54 0D 0A           BYTE 54 0D 0A
370C B1 08      LOAD    LDA(ARGYL),Y    load varible into Ry
370E 20 1B 37           JSR LTRTST
3711 90 07              BCC LOAD1
3713 20 89 35           JSR PACADD      adjust address
3716 20 00 30           JSR PACKER      load number
3719 38                 SEC
371A 60         LOAD1   RTS
371B C9 41      LTRTST  CMP#41          test for letter
371D 90 04              BCC BAD
371F C9 5B              CMP#5B
3721 90 01              BCC OUTL
3723 38         BAD     SEC             C=1,non letter
3724 60         OUTL    RTS
3725 C9 5E      OPERT   CMP# 5E         raise to power
3727 D0 03              BNE OP1
3729 4C 2E 35           JMP XRY

372C C9 2A      OP1     CMP#2A          *
372E D0 03              BNE OP2
3730 4C 0B F9           JMP MUL
3733 C9 2F      OP2     CMP#2F          /
3735 D0 03              BNE OP3
3737 4C 16 FA           JMP DIVIDE
373A C9 2B      OP3     CMP#2B          +
373C D0 03              BNE OP4
373E 4C 08 F8           JMP ADD
3741 C9 2D      OP4     CMP#2D          -
3743 D0 03              BNE OP5
3745 4C 00 F8           JMP SUB
3748 4C F0 FC   OP5     JMP MVXZ
```

Tables:

```
374B 41 42 53   ABS             TAB1    function code names
374E 41 43 53   ACS
3751 41 4C 47   ALG
3754 41 53 4E   ASN
3757 41 54 4E   ATN
375A 43 4F 53   COS
375D 44 45 47   DEG
3760 46 4E 41   FNA
3763 46 4E 42   FNB
3766 46 4E 43   FNC
3769 49 4E 56   INV
376C 4C 4F 47   LOG
376F 52 41 44   RAD
3772 53 49 4E   SIN
3775 53 51 52   SQR
3778 54 41 4E   TAN
377B FF FF FF
377E FF FF FF
3781 FF FF FF
3784 FF FF FF
3787 FF 13 35                   TAB2    function addresses
378A FF 3E 34
378D FF F9 32
3790 FF 54 34
3793 FF 6D 34
3796 FF 99 33
3799 FF 1E 35
379C FF F0 FC
379F FF F0 FC
37A2 FF F0 FC
37A5 FF 4A 35
37A8 FF 10 32
37AB FF 26 35
37AE FF 54 33
37B1 FF 72 32
37B4 FF 70 33
37B7 FF FF FF
37BA FF FF FF
37BD FF FF FF
37C0 40 17 45 32 92 51 99 40 F2   Pi/180       00
37C9 40 31 62 27 76 60 16 70 F1   1/SQR(10)    09
37D2 00 15 70 79 63 26 79 50 F0   Pi/2         12
37DB 00 18 F2                     180           1B
37DE 00 90 F1                      90           1E
37E1 00 31 41 59 26 53 59 F0       Pi           21
```