A KIM Binary Calculator

BY JOHN EATON
1126 N 2nd
Vincennes, Ind. 47591

Binary math has always been a stumbling block for anyone attempting to study the inner workings of a computer. No matter how often you work problems in binary, octal or hexadecimal, you can never work as efficiently as you can in decimal. Most of us when confronted with a problem involving other number systems will have to reach for a pencil and paper in order to obtain a result.

But now there is an easier way to solve binary math problems. With this software your KIM-1 or other 6502 system can do all the work and give you the result in a form that you can understand. The program is a machine language routine that will fit in less than 1K of memory. It communicates with the user through any ASCII I/O device.

The program will work with binary numbers up to 16 bits. They can be inputted to the program in decimal, hexadecimal, octal, binary or as an ASCII literal character. The user may specify whether the results are displayed in decimal, hexadecimal, octal or binary. With these features the program is able to convert numbers between different bases.

Several logic functions are available. The program will take any two 16 bit numbers and perform the operations of AND, OR, NOR, EOR and NAND. You may also perform the arithmetic operations of addition and subtraction. The inverse and two's complement of any 16 bit number can be found.

There are two rotate commands available. The rotate commands treat two 16 bit numbers as a single 32 bit number and can shift it left or right up to 16 bits.

Using the Routines

The program requires a MOS 6502 based system and *must* have a rotate right instruction. If you have one of the earlier processors then you will have to modify the code around address 327E to perform eight ROL's for each ROR. The program is assembled starting at address 3000 but is small enough that it will fit in a KIM-1 without any additional memory. The starting addresses for the user's I/O routines go in the first 9 bytes. You will need a single character input routine, single character output routine and a routine that performs a carriage return and linefeed.

Start the program at address 3010. It should respond with a carriage return and linefeed. The first thing that you should do is set the display mode. Type either DEC, HEX, OCT, or BIN depending on what mode you want the result to be displayed in. The mode that you type will remain in effect until you specify a new one. The program will respond by displaying the current value of its display register.

The ENT command is used to enter numbers into the display register. Type ENT and a space followed by a number. If the number is other than a decimal number, it must be preceded by a % for binary, @ for octal, $ for hex and " for an ASCII literal. ASCII literals must end with a ". The number must be an integer and may be preceded by a "–" if the two's complement of the number is wanted. Once a number is displayed by the program you can use it in various ways. If you type INV or TSC then the inverse or two's complement of the number is displayed.

Several operations require the use of a second number to produce a result. Logical functions are available by typing AND, OR, EOR, NAND or NOR. Each of these commands must be followed by a space and a second number. The number can be in the same format as in the ENT command. The result of the operation will be displayed. The commands ADD and SUB may be used in a similar manner for integer arithmetic.

The rotate commands require a special format. To use them type ROL or ROR followed by an integer number in parenthesis. This number specifies the number of bits that are to be shifted. The parenthesis must be immediately followed by a 16 bit number that is to be shifted into the display to replace the bits that are shifted out.

The program does all binary/decimal conversions with counting loops. It's very easy to write but extremely slow. The worst case time for any conversion is 5 seconds so if you want fast results use a non-decimal display mode.

Isn't it nice to now have a computer that can solve problems that you never would have to solve if you didn't have a computer?

```
0000            DISL                    Display
0001            DISH                    register
0002            DECL                    dec/bin conversion
0003            DECH                    register
0004            BUFL                    Buffer 3 Bytes
0005            BUFH
0006            BUFO
0007            POIN                    input line pointer
0008            MODE                    display mode
0009            SHIFT                   # of bits per char
000A            MASK
000B            SIGN
000C            VEB
000D            VEB+1
000E            VEB+2
0010            LR                      line register 32 bytes

3000 4C 00 00   INPUT   JMP CHAIN       user input routine
3003 4C 00 00   OTPUT   JMP CHAROT      user output routine
3006 4C 00 00   CRLF    JMP CARLF       user carret/lf routine
3009 EA                 NOP
300A 20 03 30   OUT     JSR OTPUT       display number
300D 68         DISPLA  PLA
300E D0 FA              BNE OUT         characters on stack
3010 A9 01      START   LDA#01          Start program here******
3012 85 07              STA POIN
3014 20 06 30           JSR CRLF
3017 C6 07      BACK    DEC POIN
3019 20 00 30   LOOP    JSR INPUT       fetch character in Accum
301C C9 08              CMP#08          backspace?
301E F0 F7              BEQ BACK        yes
3020 A6 07              LDX POIN
3022 95 10              STA LR,X        save character in pg 0
3024 E6 07              INC POIN
3026 C9 0D              CMP#0D          carriage return?
3028 D0 EF              BNE LOOP
302A 20 06 30           JSR CRLF
302D 18                 CLD
302E A0 04              LDY#04
3030 20 95 30           JSR LOAD        load number into buffer
3033 20 D7 30           JSR OP          perform operation
3036 A2 01              LDX#01
3038 B5 00      MOVE    LDA DISL,X      move display register
303A 95 04              STA BUFL,X      into buffer and invert
303C 49 FF              EOR#FF          into dec/bin conversion
303E 95 02              STA DECL,X      register
3040 CA                 DEX
3041 10 F5              BPL MOVE
3043 A9 00              LDA#00
3045 85 06              STA BUFO
3047 48                 PHA             end of stack storage
```

```
3048 A6 08          LDX MODE      display mode
304A D0 27          BNE DIS1      non decimal
304C F8             SED
304D 20 D9 31       JSR SET9      Buffer=999999
3050 38       DEC1  SEC           add 1 to buffer
3051 A0 02          LDY#02
3053 A2 00          LDX#00
3055 B5 04    DEC2  LDA BUFL,X
3057 69 00          ADC#00
3059 95 04          STA BUFL,X
305B E8             INX
305C 88             DEY
305D 10 F6          BPL DEC2
305F E6 02          INC DECL      add 1 to DECL
3061 D0 ED          BNE DEC1
3063 E6 03    DEC3  INC DECH
3065 F0 06          BEQ DEC4      finished counting
3067 20 6A 32       JSR ADD256    add 256 to buffer
306A 4C 63 30       JMP DEC3

306D A0 14    DEC4  LDY#14        lenght= 20 bits
306F A2 04          LDX#04        bits/char= 4
3071 D0 03          BNE DIS2      unconditional branch

3073 BC 65 32 DIS1  LDY TAB2-1,X  fetch lenght
3076 86 09    DIS2  STX SHIFT
3078 D8             CLD
3079 A6 09    DIS3  LDX SHIFT     fetch LSD and save on stack
307B A9 80          LDA#80
307D 20 7E 32 DIS4  JSR ROTATE
3080 88             DEY
3081 30 8A          BMI DISPLA    display all characters
3083 CA             DEX
3084 D0 F7          BNE DIS4
3086 4A       DIS5  LSR A         char in A,shift to lo end
3087 D0 FD          BCC DIS5
3089 C9 0A          CMP#0A        convert to ASCII
308B 90 02          BCC DIS6
308D 69 06          ADC#06
308F 69 30    DIS6  ADC#30
3091 48             PHA           save char on stack
3092 4C 79 30       JMP DIS3

3095 A2 04    LOAD  LDX#04        number load routine
3097 B9 10 00 LOAD1 LDA LR,Y      Y points to start
309A DD D2 30       CMP LOAD8,X   match to %,",@,$

309D F0 15          BEQ LOAD4     match found
309F CA             DEX
30A0 D0 F5          BNE LOAD1
30A2 A2 04          LDX#04        decimal input
30A4 20 97 31       JSR LOADT     input number
30A7 20 E6 31       JSR DECHEX    convert to hex
30AA A5 0B    LOAD2 LDA SIGN      complement if "-"
30AC C9 2D          CMP#"-"
30AE D0 03          BNE LOAD3
30B0 20 58 31       JSR TSCB
30B3 60       LOAD3 RTS
30B4 E0 02    LOAD4 CPX#02        test for ASCII literal
30B6 D0 14          BNE LOAD7
30B8 C8       LOAD5 INY           ASCII section
30B9 A2 07          LDX#07
30BB B9 10 00       LDA LR,Y
30BE C9 22          CMP#""""      last character
30C0 F0 F1          BEQ LOAD3
30C2 0A       LOAD6 ASL A         shift char into buff
30C3 26 04          ROL BUFL
30C5 26 05          ROL BUFH
30C7 CA             DEX
30C8 10 F8          BPL LOAD6
30CA 30 EC          BMI LOAD5
30CC C8       LOAD7 INY           hex,oct and bin
30CD 20 97 31       JSR LOADT
30D0 4C AA 30       JMP LOAD2
30D3 25 22 40 LOAD8+1 .BYTE %,",@,
30D6 24             .BYTE $

30D7 98       OP    TYA           look up command and
30D8 48             PHA           perform
30D9 A0 00          LDY#00
30DB A2 00          LDX#00
30DD 20 F0 30       JSR OP1       match 1st letter
30E0 20 F0 30       JSR OP1       match 2nd letter
30E3 20 F0 30       JSR OP1       match 3rd letter
30E6 68             PLA
30E7 A8             TAY
30E8 A9 31          LDA#31        address hi byte
30EA 48             PHA
30EB B9 16 32       LDA TAB1,X    address lo byte-1
30EE 48             PHA
30EF 60             RTS           jump to operation
30F0 BC 16 32 OP1   LDA TAB1,X    search table for match
30F3 D9 10 00       CMP LR,Y
30F6 F0 08          BEQ OP2       match found
30F8 E8             INX           next entry
30F9 E8             INX
30FA E8             INX
30FB E8             INX
30FC C9 FF          CMP#FF        end of table?
30FE D0 F0          BNE OP1
3100 E8       OP2   INX
3101 C8             INY
3102 60             RTS

3103 20 58 31 SUB   JSR TSCB      routines that perform
3106 A9 75    ADD   LDA#75        operations
3108 D0 19          BNE EXEC
310A A9 31    NAD   LDA#31
310C 48             PHA
310D A9 13          LDA#13        INV address-1
310F 48             PHA
3110 A9 35    AND   LDA#35
3112 D0 0F          BNE EXEC
3114 20 DD 31 INV   JSR SETF
3117 A9 55    EOR   LDA#55
3119 D0 08          BNE EXEC
311B A9 31    NOR   LDA#31
311D 48             PHA
311E A9 13          LDA#13        INV address-1
3120 48             PHA
3121 A9 15    ORR   LDA#15
3123 85 0C    EXEC  STA VEB
3125 A9 00          LDA#00
3127 85 0D          STA VEB+1
3129 A9 60          LDA#60
312B 85 0E          STA VEB+2
312D 18             CLC
312E A2 00          LDX#00
3130 A0 01          LDY#01
3132 B5 04    EXEC1 LDA BUFL,X
3134 20 0C 00       JSR VEB
3137 95 00          STA DISL,X
3139 E8             INX
313A 88             DEY
313B 10 F5          BPL EXEC1
313D 60             RTS

313E A5 04    ENT   LDA BUFL
3140 85 00          STA DISL
3142 A5 05          LDA BUFH
3144 85 01          STA DISH
3146 60             RTS

3147 A9 01    BIN   LDA#01
3149 D0 0A          BNE MOD
314B A9 03    OCT   LDA#03
314D D0 06          BNE MOD
314F A9 04    HEX   LDA#04
3151 D0 02          BNE MOD
3153 A9 00    DEC   LDA#00
3155 85 08    MOD   STA MODE
3157 60             RTS

3158 A2 04    TSCB  LDX#04        two's comp of buff
315A D0 02          BNE TSC1
315C A2 00    TSC   LDX#00        two's comp of display
315E 38       TSC1  SEC
315F 98             TYA
3160 48             PHA
3161 A0 01          LDY#01
3163 B5 00    TSC2  LDA DISL,X
3165 49 FF          EOR#FF        invert byte
3167 69 00          ADC#00        add 1
3169 95 00          STA DISL,X
316B E8             INX
316C 88             DEY
316D 10 F4          BPL TSC2
316F 68             PLA
3170 A8             TAY
3171 60             RTS

3172 20 58 31 ROR   JSR TSCB      rotate right by rotating
3175 A9 1F          LDA#1F        left 32-n times
3177 25 04          AND BUFL
3179 85 04          STA BUFL
317B C6 04    ROL   DEC BUFL
317D A9 3F          LDA#3F
317F 25 04          AND BUFL
3181 48             PHA
3182 C8             INY
3183 20 95 30       JSR LOAD
3186 68             PLA
3187 A8             TAY
3188 A5 01    ROL1  LDA DISH        set C to bit 7 of DISH
318A 0A             ASL A
318B 26 04          ROL BUFL        shift registers left
318D 26 05          ROL BUFH        Y times
318F 26 00          ROL DISL
3191 26 01          ROL DISH
3193 88             DEY
3194 10 F2           BPL ROL1
3196 60             RTS

3197 B9 10 00 LOADT LDA LR,Y      Load routine for all
319A 85 0B          STA SIGN      but ASCII
319C C9 2D          CMP#"-"       test for negative
319E D0 01          BNE LOADT1
31A0 C8             INY
31A1 86 09    LOADT1 STX SHIFT    # of bits per char
31A3 BD D0 31       LDA LOADT5,X
31A6 85 0A          STA MASK
31A8 20 D5 31       JSR SETZ      buff=000000
31AB B9 10 00 LOADT2 LDA LR,Y
31AE C9 30          CMP#30
31B0 90 1E          BCC LOADT5    ending character
31B2 C9 3A          CMP#3A
```

TINY BASIC FOR COSMAC

Dear Sir:

I have read letters in your magazine (and others) seeking a source of basic for the RCA COSMAC CDP 1802 microprocessor. Infinite Incorporated offers tiny basic 3K on paper tape, listing or cassette tape. Their address is: Infinite Incorporated, 1924 Waverley Place, Melbourne, FL 32901.

Yours truly,

Dana S. Majors

28 La Cienega Way
Yuba City, CA 95991

'WHIZ' LOADER/DUMPER

news release *Received: 77 Dec 12*

Shifting Sands Microcomputer Products Corporation has begun shipping WHIZ, a software program for recording and loading MC6800 based programs on the SWTPC M68 computer with MIKBUG and the SWTPC AC-30 cassette interface. WHIZ operates at nine times the standard MIKBUG format 300 baud speed and three times that of the binary format. This higher speed capability is provided without any modifications to the standard computer or AC-30. Load a 1K program in 14 seconds, 4K in 48 seconds, and 8K BASIC in 85 seconds. WHIZ includes an interactive front-end which allows the specification of a header and program start address to be placed on tape for read-back later. For $15.95, WHIZ is supplied on Kansas City standard cassette in MIKBUG format and includes a built-in relocater to place your own copy of WHIZ in RAM memory wherever you wish. WHIZ is optionally available in 2708 EPROM from Shifting Sands Microcomputer Products Corporation, Box 441, Fairborn, OH 45324.

DATAFORCE MAINTENANCE SERVICE FOR COMPUTER VENDORS

News Release *Received: 77 Sept 1*

A national service organization is seeking the business of independent vendors of small business computers including computer stores. The general trouble shooting of machines in this market has been generally viewed as unprofitable and the major drawback to handling the bottom end of small computer lines. But the Dataforce Service Company is now offering to perform warranty or out of warranty work and field service for vendors of small computers and computer kits anywhere in the United States.

Dataforce has 115 service locations in the country and is expanding with the goal of having a service outlet within 50 miles of every computer sold.

A central diagnostic center is at Dataforce's headquarters, 2807-F Oregon Court, Torrance, CA 90503. It opens at 5 a.m. for trouble calls from the East Coast and reamins open until 5 p.m., Monday through Friday. The company stresses its desire to help first time computer users with the simple problems arising from unfamiliarity with their equipment as well as the sophisticated user who needs a highly skilled diagnosis of a computer problem.

Dataforce is first contacted by calling its Torrance diagnostic center. In California the number is (213) 328-2950. Outside the state, call (800) 421-1858. A technician qualifies the nature of the problem and takes the necessary steps to resolve it. When on-site assistance is needed, Dataforce dispatches parts and test equipment from the closest service organization to the trouble.

The company is wholly owned by Randal Data Systems, Inc., but operates independently. It was created to meet the requirements of Randal distributors, outlets for the company's Link 100, Link 200, and Link 500 series, small business computers.

```
31B4 90 06          BCC LOADT3      number
31B6 C9 47          CMP#47          ending character
31B8 B0 16          BCS LOADT5
31BA E9 06          SBC#06
31BC 25 0A    LOADT3 AND MASK       clear unneeded bits
31BE A6 09          LDX SHIFT
31C0 06 04    LOADT4 ASL BUFL       shift buffer X times
31C2 26 05          ROL BUFH
31C4 26 06          ROL BUFO
31C6 CA             DEX
31C7 D0 F7          BNE LOADT4
31C9 05 04          ORA BUFL        combine new character
31CB 85 04          STA BUFL
31CD C8             INY
31CE D0 DB          BNE LOADT2      unconditional
31D0 60       LOADT5  RTS
31D1 01 00 07       .BYTE 01,00,07
31D4 0F             .BYTE 0F

31D5 A9 00    SETZ  LDA#00
31D7 F0 06          BEQ SET
31D9 A9 99    SET9  LDA#99
31DB D0 02          BNE SET
31DD A9 FF    SETF  LDA#FF
31DF 85 04    SET   STA BUFL
31E1 85 05          STA BUFH
31E3 85 06          STA BUFO
31E5 60             RTS

31E6 98       DECHEX TYA            decimal to binary conv
31E7 48             PHA
31E8 F8             SED
31E9 A9 FF          LDA#FF
31EB 85 02          STA DECL
31ED 85 03          STA DECH
31EF E6 02    DH1   INC DECL
31F1 D0 02          BNE DH2
31F3 E6 03          INC DECH
31F5 A2 00    DH2   LDX#00
31F7 A0 02          LDY#02
31F9 18             CLC
31FA B5 04    DH3   LDA BUFL,X
31FC E9 00          SBC#00
31FE 95 04          STA BUFL,X
3200 E8             INX
3201 88             DEY
3202 10 F6          BPL DH3         keep looping until
3204 B0 E9          BCS DH1         buffer is 0000
3206 D8             CLD
3207 A5 02          LDA DECL        move into buffer
3209 85 04          STA BUFL
320B A5 03          LDA DECH
320D 85 05          STA BUFH
320F A9 00          LDA#00
3211 85 06          STA BUFO
3213 68             PLA
3214 A8             TAY
3215 60             RTS

3216 41 44 44 05    TAB1 consists of 3 ASCII chars and the
321A 41 4E 44 0F         lo byte-1 of the operations add-
321E 42 49 4E 46         ress. Hi byte = 31.
3222 44 45 43 52
3226 45 4E 54 3D
322A 45 4F 52 16
322E 48 45 58 4E
3232 49 4E 56 13
3236 4E 41 44 09
323A 4E 4F 52 1A
323E 4F 43 54 4A
3242 4F 52 52 20
3246 52 4F 4C 7A
324A 52 4F 52 71
324E 53 55 42 02
3252 54 53 43 5B
3256 FF FF FF 3C
325A FF FF FF 3C
325E FF FF FF 3C
3262 FF FF FF 3C
3266 10 10 12 10    TAB2+1 consists of lenghts for non dec dis

326A 18       ADD256 CLC            add 256 to buffer
326B A5 04          LDA BUFL
326D 69 56          ADC#56
326F 85 04          STA BUFL
3271 A5 05          LDA BUFH
3273 69 02          ADC#02
3275 85 05          STA BUFH
3277 A5 06          LDA BUFO
3279 69 00          ADC#00
327B 85 06          STA BUFO
327D 60             RTS

327E 46 06    ROTATE LSR BUFO       perform rotate rights
3280 66 05          ROR BUFH
3282 66 04          ROR BUFL
3284 6A             ROR A
3285 60             RTS
```