Questdata

Volume 2 Issue #12

©

RELOCATOR

by
Enos Jones

It is a near certainty that many Elf owners have found themselves in the position where it is necessary to add some additional code inside an existing machine language program either because during development an ommission occurred or just to bring about improvements in an existing program. At that point two options generally exist:

1. Replace two or more bytes with a jump or branch to the new routine which is somewhere else in memory, then restore the last bytes at the end of the new routine and jump back at a point past the patch. This is not always acceptable.

2. Manually relocate the existing code to make space for the new routine. This involves rewriting the program out on paper and adjusting the branch instructions, and then reentering it. Again it is far from optimal.

The solution to this dilemma was to write a machine language program which performs the second type of relocation quickly and painlessly. It requires 11 bytes of information as follows:

A. Function select 00 – fix references only

   01 – move block and fix references

   02 – move block only

B. Starting address of (2 bytes A1–Hi address code to be relocated A2–Lo address).

C. Ending address of (2 bytes B1–Hi address code to be relocated B2–Lo address).

D. Destination address (2 bytes D1–Hi address for code to be relocated to D2–Lo address).

E. First address to (2 bytes F1–Hi address have references fixed F2–Lo address).

F. Last address to have (2 bytes F3–Hi address references fixed F4–Lo address).

The relocator uses the technique of immediate data display to prompt for each of the various parameters it needs. That is, the output display of the Super Elf will display 'FC' when a function code is required at which time the appropriate keys are pressed followed by 'I'. Elf will accept it and delay before prompting for the next parameter.

The relocator performs a block move if function code 00 is entered. The relocator both moves the block and fixes branch references if code 01 is selected. If 02 is selected, only the branch references are fixed (i.e. no block move).

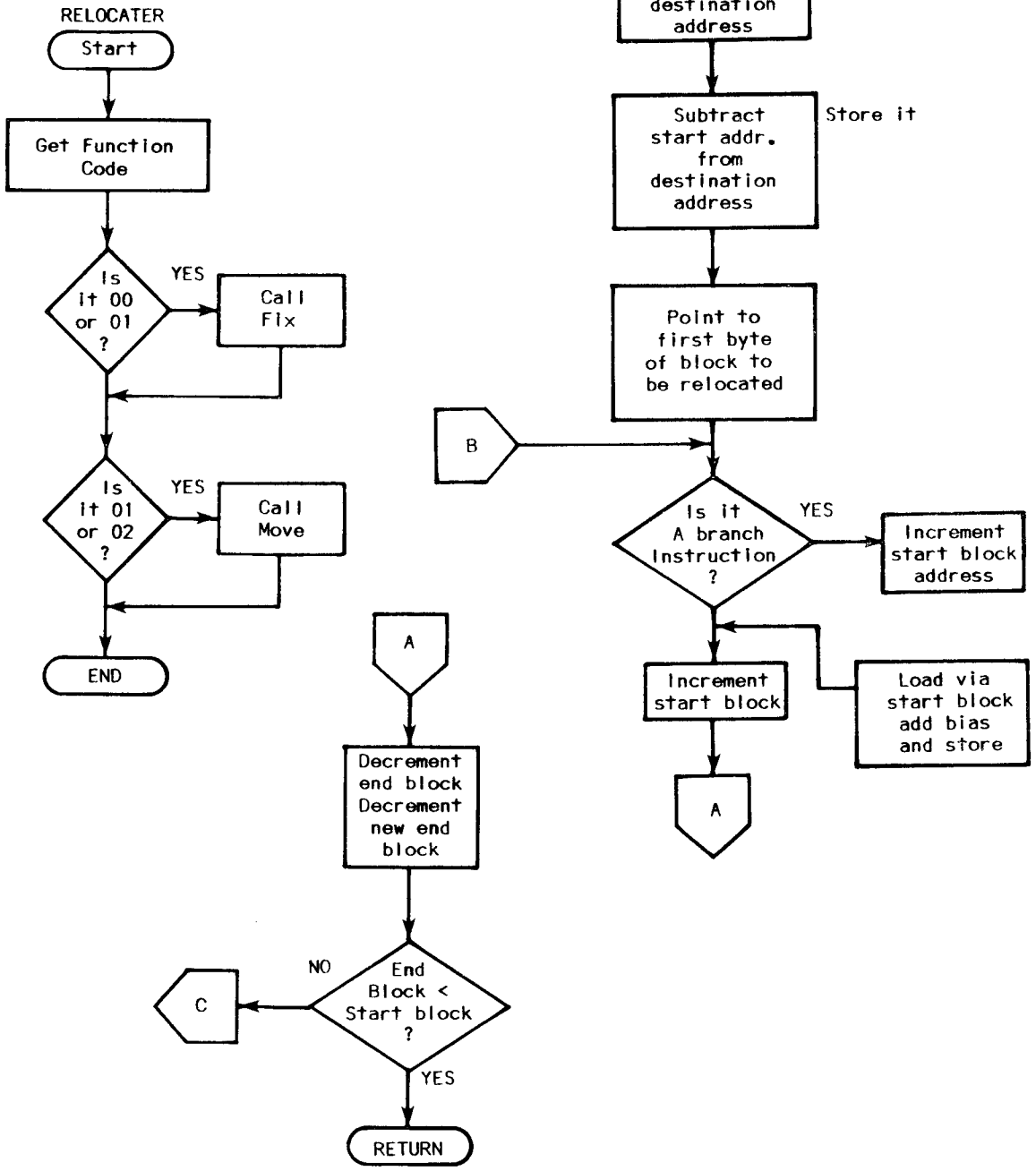
The relocator fixes references by first calculating the offset to add to the branch reference by computing offset equals (b–d). Then, it searches through the specified fix reference range (e through f) for branch instructions and when one is found it adds the offset to the next byte and replaces that byte.

It must be noted that the relocator does not adjust load immediate (F8) instructions so register set–ups using the technique must be manually changed. Additionally, the relocator performs a block move by moving a block end first.

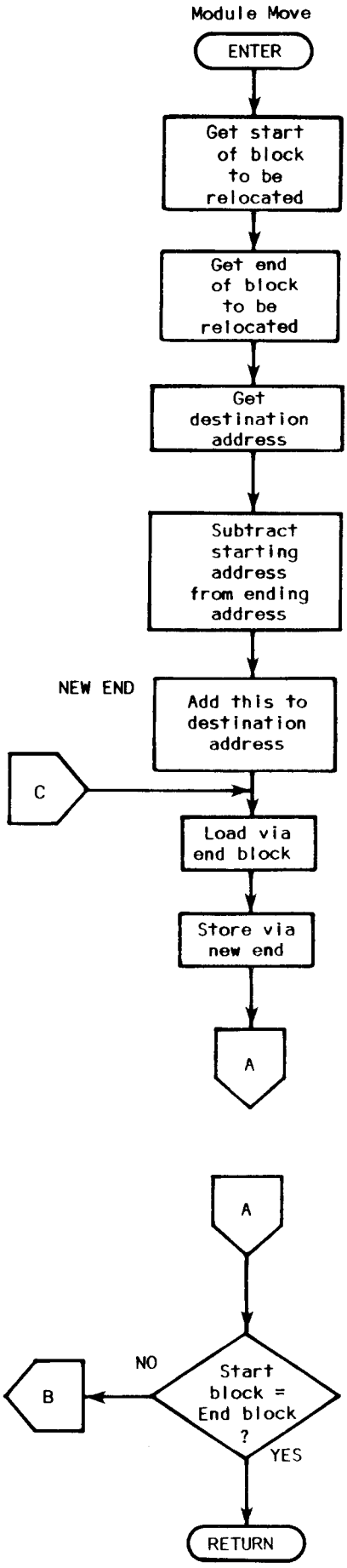
Quest Electronics Documentation and Software by Roger Pitton is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License.

Thus block moves which result in moving the block from a lower address to a higher address will always work, while moving a block from a higher address to a lower address will have to be done with caution. In the latter case, we must specify a destination address which is a lower address than the start of the original block to be moved. Finally, when relocating blocks of code which are data, function code 02 should be used so that no data would be inadvertently changed by being mistakened for a branch type instruction.

The program begins at 0400 (assumes that the PC is R3). When all the indicated parameters have been entered, the program does the indicated function and terminates by display 'AA' and going into an endless loop.



Quest Electronics Documentation and Software by Roger Pitkin is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License.



REGISTER USAGE:

R2 - Stack
R3 - PC
R4 - First address of block to be relocated
R5 - Last address of block to be relocated
R6 - Destination Addr.
R7 - First address to have references fixed
R8 - Last addr. to have references fixed.
R9 - Function Code.
RA - 2 byte code table
RB - 3 byte code table
RC - CALL FIX
RD - CALL MOVE
RE - INPUT ROUTINE

```
ADDR CODE     LABEL    OPCODE OPERAND COMMENT
0400 F8 04    START:                  R2 - Stack
0402 B2
0403 F8 F0 A2                         *
0406 F8 05 BE                         RE-Input routine
0409 F8 01 AE                         *
040C F8 05 BC                         RC-Fix reference
040F F8 14 AC                         *
0412 F8 04 BD                         RD-Move routine
0415 F8 5F AD                         *
0418 E3 64 FC                         Get Function code
041B DE A9                            *
041D E3 64 A1                         Get address of 1st
0420 DE B4                            Block to be
0422 E3 64 A2                         relocated. . .
0425 DE A4
0427 E3 64 B1                         Get last addr. of
042A DE B5                            block to be
042C E3 64 B2                         relocated.
042F DE A5                            *
0431 E3 64 D1                         Get destination
0434 DE B6                            address
0436 E3 69 D2                         *
0439 DE A6                            *
043B E3 64 F1                         Get first address
043E DE B7                            to have ref.
0440 E3 64 F2                         fixed and last
0443 DE A7
0445 E3 64 F3
0448 DE B8
044A E3 64 F4
044D DE A8
044F 89                               Get function code
0450 FF 02
0452 33 5B              BGE    CONT
0454 F8 05 BA                         CALL FIX
0457 F8 D0 AA
045A DC
095B 89                 GLO    R9     Get function code
045C 30 87              BR     FINISH
045E DE       RETURN:   SEP    R3
045F E2 85    MOVE:     SEX    R2     Get low end
                                      address
0461 52 84              STR    R2     Get low start
                                      address
0463 F5                 SD            Subtract
0464 AF                 PLO    RF     Store in RF.0
0465 95 52              GHI R5 STR 2
0467 94 75              GHI R4 SUD B  Sub. high start
                                      from high GHI
0469 BF                 PHI    RF     Store in RF.0
046A 86                 GLO    R6     Get low destin-
                                      ation addr.
046B 52                 STR 2         Store
046C 8F                 GLO RF
046D F4                 ADD
046E AF                 PLO    RF
046F 96                 GHI    R6     Get high destin.
```

Quest Electronics Documentation and Software by Roger Pitkin is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License.

```
ADDR CODE     LABEL    OPCODE OPERAND COMMENT
0470 52                STR  via R2
0471 9F 74             GHI    RF ADC
0473 BF                PHI    RF
0474 26                DEC    R6
0475 05       ANOTHER  LD5             Load via end addr.
0476 5F       STORE    RF              Store via mod.
                                       "TO" address
0477 25 2F             DEC R5 DEC RF   Dec. to address
0479 96 52             GHI R6 STR 2
047B 9F F5             GHI RF SUB
047D 3A 75             BNZ    ANOTHER
047F 86 52             GLO R6 STR 2
0481 8F F5             GLO RF SUB
0483 3A 75             BNZ    ANOTHER
0485 30 5E             BR     RETURN
0487 32 8A    FINISH   BZ     SKIP
0489 DD                CALL MOVE
048A E3                SEX    R3       Display end of
048B 64 AA             OUT4            program
048D 30 8D    LOOPA:   BR     LOOPA    'AA'
0500 D3       RETURN:  SEP    R3
0501 3F 01    INPUT:   BN4            Wait for
0503 37 03             B4              *Input
0505 32 6C 64          SEX R2 INP4 OUT4  Display input
0508 22                DEC    R2       *
0509 F8 FF B1          LDI FF PHI R1   Delay before
0506 91 21             GHI R1 DEC R1   Returning
050E 3A 0C             BNZ             *
0510 02                LD2             Put input in D
                                       register
0511 30 00             BR     RETURN   Return
0513 D3       RETURN   SEP    R3
0514 86 52    FIX:     GLO R6 STR 2    Get destination
                                       low address
0516 87                GLO    R7       Get fix start(low)
0517 F5 AF             SD     PLO RF   Subtract and store
                                       in F.0
0519 96 52             GHI R6 STR R2   Get destination
                                       address (high)
051B 97 75             GHI R7 SDB      Get fix start
                                       (high)
051D BF                PHI    RF       Subtract and store
                                       in F.1
051E 87 73             GLO R7 STX D    Push fix start on
0520 97 73             GHI R7 STX D    Stack
0522 F8 0F A1 CYCLE:   LDI 0F PLO R1   Load number
                                       affected 2 byte
                                       ops
0525 EA       ANOTHER: SEX    RA       RX is now A
0526 07                LD7             Load from block
0527 F3                XOR             Check if 2 byte
                                       (Aff.) code
0528 3A 44             BNZ    CHECK2   No, go check other
                                       tables
052A E7 17             SEX R7 INC R7   Yes! Point to
                                       branch value
052C 8F                GLO    RF       Get bias value
052D F4 57             ADD    STR7     Add to value,
                                       change value
052F E2       BLOCK END: SEX   R2       R2-Index
0530 98 52             GHI R8 STR R2   All block bytes
0532 97 F5             GHI R7 SUB      Looked at?
0534 33 3E             BNZ    MORE     *
0536 88 52             GLO R8 STR 2    *
0538 87 F5             GLO R7 XOR
053A 33 3E             BNZ    MORE
053C 30 13             BR     ENDFX    Yes, return from
                                       routine
053E 17       MORE:    INC    R7       Increment block
                                       pointer
053F F8 D0 AA          LDI    PLO RA   Restore table
                                       pointer
0542 30 22             BR     CYCLE    Go process another
```

```
ADDR CODE     LABEL    OPCODE OPERAND COMMENT
0544 21 1A    CHECK2:  DEC R1 INC RA   All table entries
0546 81                GLO    R1       Checked?
0547 3A 25             BNZ    ANOTHER  No branch
0549 F8 0A A1 2 BYTE   LDI 0A PLO R1   Load number no
              (NON)                    effect 2 byte
                                       codes
054C EA       AGAIN:   SEX    RA       RA is index
054D 07                LD7             Load from block
054E F3                XOR             Check if 2 byte
                                       number
054F 3A 67             BNZ    CHECK3   No go check other
                                       codes
0551 17                INC    R7       Yes, just
                                       increment
0552 E2 98             SEX R2 GHI R8   At end of
0554 52 97             STR R2 GHI R7   reference fixing?
0556 F5                XOR             *
0557 33 61             BNZ    N FIN.
0559 88 52             GLO RS STR R2
055B 87 F5             GLO R7 XOR
055D 33 61             BNZ    N FIN
055F 30 13             BR END FIX      *Yes return from
                                        routine
0561 17                INC    R7       Inc. block pointer
0562 F8 D0 AA          LDI XX PLO RA   Reset table
                                       pointer
0565 30 22             BR     CYCLE    Go process
                                       (through table)
0567 21 1A    CHECK3:  DEC R1 INC RA   All table entries
0569 81                GLO    R1       checked?
056A 3A 4C             BNZ    AGAIN    No! Branch
056C F8 07 A1 3 BYTE   LDI 07 PLO R1   Yes, check 3 byte
              (ACT)
056F EA       MANY     SEX    RA       Opcode tables
0570 07                LD7             Load from clock
0571 F3                XOR             Check if 3 byte
                                       opcode
0572 3A 81             BNZ    CHECK4   No! Branch
0574 E7 17 17          INC R7 INC 17   Yes! Point to low
                                       address
0577 8F F4 73          GLO RF ADD      Add bias to it
                                       and change it
057A 9F 74 57          GHI RF ADC ST7  Add bias to high
                                       and change it
057D 17 C4             SEX    R2
057F 30 2F             BR  BLOCKEND
0581 21 1A 81 CHECK4:  DEC R1 IN
0584 3A 6F             BNZ    MANY
0586 17                INC    R7
0587 F8 D0 AA          INC    R7       Reset table
058A 30 22             BR     CYCLE
```

```
ADDR CODE                              LABEL
05D0 30 31 32 33 34 35 36 37           TWO BYTE
05D8 39 3A 3B 3C 3D 3E 3F              *
05DF 7C 7D 7F F8 F9 FA FB FC           TWO BYTE
05E7 FD FF                             (No affect)
05E9 C0 C1 C2 C3 C9 CA
```

Quest Electronics Documentation and Software by Roger Pitkin is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License.

Q*BUG

Welcome to the QUEST BASIC USERS GROUP.
(Here after known as QBUG)

Hopefully, this will be a regular feature of QUEST DATA devoted to users of the QUEST SUPER BASIC VERSION 5.0 program. Emphasis will be on machine language enhancements to SUPER BASIC. My aim will be to pass along any improvements that you, the readers of this column, contribute and a few of my own. Since this is a USERS group column, it is your input that will support future columns. Please send in any ideas, suggestions, questions, or what have you, so that we can keep the column rolling.

I am a self-taught machine language tinkerer and am not a real hotshot programmer. However, with a little patience and some hand disassemblying of SUPER BASIC, I feel anyone can learn the fundamental workings of SUPER BASIC and come up with some useful gimmicks even if they only satisfy some personal whim.

First thing, let me explain the format in which I will write numbers in this and any future columns.

By now, hopefully, you know that the internal workings of the 1802 microprocessor are expressed in hexidecimal format numbers (hex). This means that any reference to register contents, memory contents, addresses, etc. should and will henceforth be in hex form. If I write "Page 3500" or "Address 2345", these will always be hex numbers. I will not show an "M" prefix for an address or an "H" suffix for a number.

Of course, any number within a Basic program, unless preceeded by a # or a @, will be in decimal format.

Also, SUPER BASIC VER. 5.0 will henceforth be called simply "Super" which it surely is!

Because my setup consists of an ELF II with the Netronics Video Board and terminal, some of my changes to SUPER are made to satisfy the limitations imposed by the quirks in the ELF II equipment. I do not do much original programming in BASIC and usually concentrate on converting published programs for TRS or other computers to a format usable in SUPER or machine language programming.

Now, let me present a method for providing a little quicker start up for SUPER.

SUPER 5.0, as presently configured and written, requires that the user respond with a C/R or 'M' when first bringing up SUPER. This is to allow SUPER to measure the terminal time constants and 'stuff' them into work page 0000 at locations 00E7 and 00E8. This routine is included so that SUPER will be somewhat hardware independent and can be used on any terminal.

Once the user has run SUPER on his terminal, these constants are available for reuse and the initial C/R or'M' response can be eliminated. Also, the initialization routine occupies memory location 3493 to 34CC and this program space can be used for other routines.

The key to eliminating the initialization routine is the inclusion of work page 0000 to 00FF in the taped master SUPER program. This will insure that the work page and the time constants and other data recorded on the work page will be loaded into memory when SUPER is loaded. Thus, SUPER, when first re-recorded to your backup tapes, should be recorded from memory location 0000 to the end of your particular version of SUPER. This will include the time constant location 00E7 and 00E8 and allow us to completely eliminate the initialization routine at 3493.

SUPER vectors to the Serial I/O initialization routine with a long branch at location 3300 to location 3400. The final long branch to the initialization routine is C0 34 93 at location 3400. For ELF II users, this should be changed to C0 31 48 (the actual CLS routine) to clear the screen and return to the C/W prompt. I presume that non ELF II owners can merely change location 3400 to a D5 instruction but cannot guarantee the results.

Finally, for this column, we should discuss procedures for providing cassette tape deck control on an ELF II using the Netronics motor control board.

Although the manual for SUPER 5.0 states that cassette tape deck control can only be accomplished with a hardware change, I find that a simple software change in SUPER will provide motor control without any hardware changes. Page 3200 of SUPER contains most of the cassette in/out routines. Simply stated, although not exactly step by step, non-ELF II deck control uses the byte 00 to turn off the input and output decks, byte 01 to turn on the input deck,

Quest Electronics Documentation and Software by Roger Pitkin is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License.

and byte 02 to turn on the output deck. The ELF II uses byte 00 to turn on both decks, byte 01 to turn on the output deck, and byte 02 to turn on the input deck. Additionally, byte 03 will turn off both decks.

The first change to be made will insure that both decks are turned off when SUPER is first entered. Since we will be using and saving work page 0000 as part of the SUPER program, we can put a simple little routine at location 0000 to turn the decks off and then jump to the Cold Start routine at location 0100. This change is:

0000 – E0 67 03 (Output byte 03,turn off decks)
0003 – C0 01 00 (Jump to Cold Start)

The next change is to the byte at location 32A0. This is the byte to turn off both decks after a Pload, Psave, Dload, or Dsave and is presently '00'. Change this byte to '03' and you now turn off the ELF II decks.

Next, we will change the bytes which turn on the input or output decks. This is presently controlled by a tricky routine at location 321E thru 3225.

Apparently, SUPER enters the cassette I/O routines with the DF register set to 1 if it is in the Dsave or Psave mode. DF is tested and if it is 1, execution jumps to location M 321EH. SUPER will then load D with 01 (F8 01 at 321E), add (7C) D, DF, and the 00 byte at 3221, and put the results in D and on the stack(52). D is then shifted to the right (F6) and DF is tested for 0.

If SUPER did come in with DF set to 1, adding the 01 in D and the 1 in DF and putting the result in D will make D=2. Shifting 02 (0000 0010 in binary) to the right would leave D=01 and force DF=0 (0000 0001(0 to DF)). The DF test at 3224 (3B) would be true and execution would branch to location 32B5. This is the cassette output routine and the byte 02 on the stack would eventually be sent to the output deck by the 63 instruction at location 32C1.

If SUPER entered the cassette I/O routines in the Pload or Dload mode, DF will be set to 0. The 01 at 321E would be added to DF (0) and the result put in D. Shifting 01 right would force DF=1 and the DF test at 3224 would fail and execution would continue at 3226. This is the cassette input routine and the 01 would be sent to the input deck by the 63 instruction at location 3226.

To make the changes necessary for the ELF II, we will simplify the routines somewhat. Since locations 3200 to 3205 are actually unused, we will use them for part of our changes. First, at location 321E, we will put a short branch to location 3200 (33 00). This branch is conditional, based on the state of DF. If DF=1, the test is true and execution will branch to 3200. Thus, if SUPER enters in the Dsave or Psave mode, we will branch to 3200. At location 3200, we will load D with 01 (F8 01), put 01 on the stack (52) and unconditionally branch (30 B5) to location 32B5, where the 01 byte will be sent to the output deck and execution in the output mode will continue.

If SUPER entered in the input mode, the conditional branch at 320E will not take effect since DF will equal 0. We will then want to load D with 02 (F8 02) at location 3210, put 02 on the stack, and continue execution at 3226, the cassette input routine. The 02 in D will be sent to, and turn on, the input deck and execution in the input mode will continue.

Incidentally, ELF II owners, don't forget to change the 63 instructions at 3226, 32A2, and 32C1 to 67.

In summary, the cassette changes are:

```
0000    E0 67 03
0003    C0 01 00
3200    F8 01 52 30 B5 C4
321E    33 00
3220    F8 02 52 C4 C4 C4 67
32A0    03 52 67
32C1    67
```

EDITORS NOTE:
This will be a regular feature as we have 6 more ready to publish!
CONGRATULATIONS FRED

QUESTDATA
P.O. Box 4430
Santa Clara, CA 95054

Publisher.....Quest Electronics
Editor.....Paul Messinger
QBUG Editor.....Fred Hannan
Proof Reading.....Judy Pitkin
Production.....John Larimer

The contents of this publication are copyright and shall not be reproduced without permission of QUESTDATA. Permission is granted to quote short sections of articles when used in reviews of this publication. QUESTDATA welcomes contributions from its readers. Manuscripts will be returned only when accompanied by a self addressed stamped envelope. Articles or programs submitted will appear with the authors name unless the contributor wishes otherwise. Payment is at the rate of $15 per published page. QUESTDATA exists for the purpose of exchanging information about the RCA 1802 microcomputer.

Quest Electronics Documentation and Software by Roger Pitkin is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License.

THE COSMAC KID

By Mark Wendell

It had been a great day. I had just finished killing off seventeen Klingons in the computer of a local college. When I got home, I checked the mail. Waiting for me was Popular Electronics and I read the article and was completely blown away, being relatively new to micros and the language, but my interest had been caught, the seed was planted. I read up as much as I could on micrcomputers in general, and I got out all of my back issues of Popular Electronics, Radio-Electronics, and Elementary Electronics, trying to find as much as I could about the elusive 1802.

Having just graduated from junior high, my finances were almost non-existant (have you ever seen a rich just-graduated junior high schooler?). So I had to start scrimping and saving, working my fingers to the bone, and weaseling as much as I could from my parents. And then came my birthday. Since I had been such a good kid all year (heh, heh), my dad decided to split the cost with me. I was almost there.

Now I had to figure out which kit to get. After price-comparing and writing some letters, I decided on Quest.

There was only one more obstacle to overcome: my mother, who hates machines with a passion. The following is a conversation between myself and the matriarch of the household on the day I finally had all the money raised.

"Yes, but what can it do? What is it good for?!"
"Well...er...it can count. And it can make sounds!" I gulped, the slow terror of rejection rising in my gut.
"It can count and make noise, and it costs over a hundred bucks!" Oh.
"Well, it also can...uh...make neat pictures on the TV!"
"I just don't know, it just doesn't sound like its worth it..."
And suddenly I had it, I'd stick her with education!
"Its also educational! It can teach me logic, and Boolean algebra, and electronics, and number systems, and programming!" I had her by the... er...I sure had her now.

"Well, if you think its worth it, but...ah... what can it do?" You get the picture.

Three days later, weary and worn, but happy at my eventual victory, I sent out my order for a Super Elf kit.

A little less than a week later I got it via UPS and eagerly looked it over. After going through two rolls of solder, I viewed my creation and took the big step: I plugged it in. And...nothing happened. Completely crushed and at the point of suicide, I wrapped it up and sent it back the next day for repairs. Time munched onward.

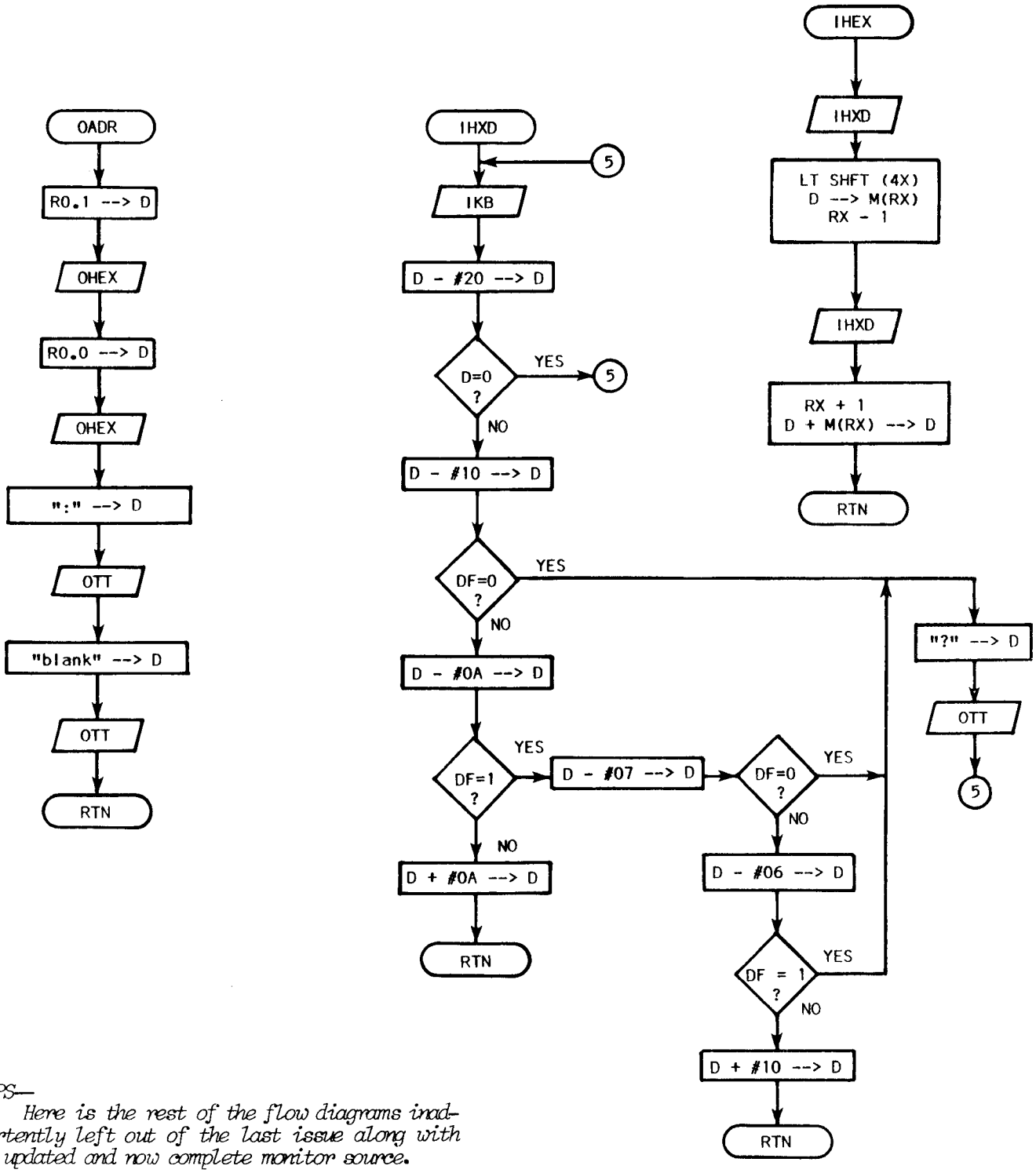
A month later, my Elf came back home with a doctor's bill attached. I paid the bill and looked it over. Apparently, I had made a few construction errors (I'm only human), but it was better now. Again I plugged it in, only this time I saw the most glorious pair of red zeroes that I had ever seen in my whole life, and these were flanked by magnificient stars that shone brilliantly, expressing the 1802's mode and state of mind. Since I had read up on programming, I loaded my first program; 7B00! Boy! Another star blinked into life! Totally overjoyed, I ran the gamut of programs listed in Popular Electronics. When I came to the sound generators, I became ecstatic-it was the most incredibly fantastic thing in the whole world! But even that didn't last forever. My software was definitely sparse, so I began looking around. Suddenly, there was a glimmering light-a ray of hope befell my eyes as I received the first complimentary issue of QUESTDATA. This was where it was at! I readily subscribed, and am now happily feeding my Elf wondrous programs. The little Elf greedily accepts all programs with open RAM and churns out results perfectly. The COSMAC is a willing jinni and joyfully serves any lucky soul fortunate enough to own the microprocessor of microprocessors: the 1802.

Quest Electronics Documentation and Software by Roger Pitkin is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License.

E~BUG··2

BY

PHILLIP LIESCHESKI



OOPS—

Here is the rest of the flow diagrams inadvertently left out of the last issue along with an updated and now complete monitor source.

Quest Electronics Documentation and Software by Roger Pitkin is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License.

Quest Electronics Documentation and Software by Roger Pitkin is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License.


```
                        * EBUG5 SYSTEM MONITOR
                        *
                        *   PHILLIP B. LIESCHESKI III
                        *
                        *
                                ORG     @4700
4700:   300D    START   BR      EBUG5
                        *
                        * SCRATCH-PAD
4702:   00      PAD     DATA    #00
                        *
                        * I/O PATCH
4703:   BF      OTT     PHI     RF
4704:   C08303          LBR     @8303
4707:   C08300  IKB     LBR     @8300
                        *
                        * BREAKPOINT RE-ENTRY PATCH
470A:   C04860  BENT    LBR     BENT1
                        *
                PAGE    EQU     @0047
                STKPG   EQU     @004E
                ZERO    EQU     @0000
                CR      EQU     @000D
                LF      EQU     @000A
                PERIOD  EQU     @002E
                PROMPT  EQU     @003E
                BLANK   EQU     @0020
                WHAT    EQU     @003F
                COLON   EQU     @003A
                PARITY  EQU     @0080
                DASH    EQU     @003C
                *
                * MAIN ROUTINE
470D:   E2      EBUG5   SEX     R2      PREPARE STACK PNTR R2
470E:   F8FF            LDI     #FF
4710:   A2              PLO     R2
4711:   F847            LDI     PAGE    PREPARE SCRT ROUTINE
4713:   B4              PHI     R4
4714:   B5              PHI     R5
4715:   F84E            LDI     STKPG   FINISH WITH STACK PNTR
4717:   B2              PHI     R2
4718:   F84E            LDI     SCALL
471A:   A4              PLO     R4
471B:   F860            LDI     SRETN
471D:   A5              PLO     R5
471E:   F80D    PROMP   LDI     CR      SIGNAL WITH A PROMPT
4720:   D44703          CALL    OTT
4723:   F80A            LDI     LF
4725:   D44703          CALL    OTT
4728:   F83E            LDI     PROMPT
472A:   D44703          CALL    OTT
472D:   F800            LDI     ZERO    LOAD INPUT REGISTER WITH
472F:   A1              PLO     R1
4730:   B1              PHI     R1
4731:   D44707          CALL    IKB     GET COMMAND
4734:   FF42            SMI     #42     CHECK FOR B COMMAND
4736:   3268            BZ      BRKP
4738:   FF02            SMI     #02     CHECK FOR D COMMAND
473A:   3246            BZ      DUMP
473C:   FF01            SMI     #01     CHECK FOR E COMMAND
473E:   329E            BZ      EXEC
4740:   FF04            SMI     #04     CHECK FOR I COMMAND
4742:   328F            BZ      INSP
4744:   FF03            SMI     #03     CHECK FOR L COMMAND
4746:   3281            BZ      LOAD
4748:   FF01            SMI     #01     CHECK FOR M COMMAND
474A:   32C1            BZ      MODI
474C:   301E            BR      PROMP
                        *
                        * CALL SCRT ROUTINE
474E:   E2      SCALL   SEX     R2
474F:   A6              PLO     R6      SAVE ACCM IN R6.0
4750:   13              INC     R3      GET RETURN ADDRESS
4751:   13              INC     R3
4752:   83              GLO     R3      SAVE RETURN ADDRESS ON STACK
4753:   73              STXD
4754:   93              GHI     R3
4755:   73              STXD
4756:   23              DEC     R3      GET SUBR ADDRESS
4757:   03              LDN     R3      PUT IT ON STACK
4758:   73              STXD
4759:   23              DEC     R3
475A:   03              LDN     R3
475B:   73              STXD
475C:   86              GLO     R6
475D:   D5              RETN            GO TO IT VIA SRETN
475E:   304E            BR      SCALL
                        *
                        * RETURN SCRT ROUTINE
4760:   E2      SRETN   SEX     R2
4761:   A6              PLO     R6      SAVE ACCM IN R6.0
4762:   60              IRX             LOAD R3 WITH CONTENTS OF STACK
4763:   72              LDXA
4764:   B3              PHI     R3
4765:   F0              LDX
4766:   A3              PLO     R3
4767:   86              GLO     R6      UNSAVE ACCM
4768:   D3              SEP     R3      BRANCH TO CONTENTS OF R3
4769:   3060            BR      SRETN
                        *
                        * BREAKPOINT SET-UP ROUTINE
476B:   D44865  BRKP    CALL    IADR    GET BREAKPOINT ADDRESS
476E:   F847            LDI     PAGE
4770:   B6              PHI     R6
4771:   BE              PHI     RE
4772:   F802            LDI     PAD
4774:   A6              PLO     R6
4775:   40              LDA     R0
4776:   20              DEC     R0
4777:   56              STR     R6      SAVE CONTENTS OF MEMORY
4778:   F8DE            LDI     #DE
477A:   50              STR     R0      INSERT A SEP RE INSTRUCTION
477B:   F80A            LDI     BENT    LOAD RE WITH MONITOR RE-ENTRY ADD
477D:   AE              PLO     RE
477E:   C0470D          LBR     EBUG5
                        *
                        * LOAD CODE ROUTINE
4781:   D44865  LOAD    CALL    IADR    GET ADDRESS
```

P.O. Box 4430, Santa Clara, CA 95054

Quest Electronics Documentation and Software by Roger Pitkin is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License.


```
478A:  61                    GLO   R1
478B:  50                    STR   R0          STORE IN MEMORY
478C:  10                    INC   R0
478D:  3084                  BR    LOAD1
                     *
                     * INSPECT CODE ROUTINE
478F:  D44865  INSP  CALL    IADR
4792:  D4486D  INSP1 CALL    OADR
4795:  40            LDA     R0
4796:  D44847        CALL    OHEX        DISPLAY MEM CONTENTS
4799:  D447E5        CALL    INCR        WAIT FOR A CR
479C:  3092          BR      INSP1       TRY AGAIN
                     *
                     * EXECUTE CODE ROUTINE
479E:  D44865  EXEC  CALL    IADR        GET ADDR
47A1:  80            GLO     R0          STORE IT ON STACK
47A2:  73            STXD
47A3:  90            GHI     R0
47A4:  73            STXD
47A5:  D5            RETN                GO TO IT VIA SRETN
                     *
                     * DUMP CODE ROUTINE
47A6:  D44865  DUMP  CALL    IADR        GET ADDR
47A9:  F808    DUMP1 LDI     #08         PREP 8 CNTR
47AB:  AB            PLO     RB
47AC:  D4486D        CALL    OADR        DISPLAY ADDR
47AF:  40      DUMP2 LDA     R0
47B0:  D44847        CALL    OHEX        DISPLAY 8 BYTES OF
47B3:  F820          LDI     BLANK
47B5:  D44703        CALL    OTT
47B8:  2B            DEC     RB
47B9:  8B            GLO     RB
47BA:  3AAF          BNZ     DUMP2
47BC:  D447E5        CALL    INCR        WAIT FOR A CR
47BF:  30A9          BR      DUMP1
                     *
                     * MODIFY CODE ROUTINE
47C1:  D44865  MODI  CALL    IADR        GET ADDR
47C4:  D447EC  MODI1 CALL    OLF         OUTPUT A LINEFEED
47C7:  D4486D  MODI2 CALL    OADR
47CA:  40            LDA     R0
47CB:  D44847        CALL    OHEX        DISPLAY MEM CONTENTS
47CE:  D44707        CALL    IKB
47D1:  52            STR     R2
47D2:  FF0D          SMI     CR
47D4:  32C4          BZ      MODI1
47D6:  F0            LDX                 CHECK FOR AN INSERT
47D7:  FF3C          SMI     DASH
47D9:  CA470D        LBNZ    EBUG5       ABORT OP
47DC:  D44824        CALL    IHEX        GET NEW BYTE
47DF:  81            GLO     R1          STORE IT IN MEMORY
47E0:  20            DEC     R0
47E1:  50            STR     R0
47E2:  10            INC     R0
47E3:  30C7          BR      MODI2
                     *
                     * INPUT CR SUBR
47E5:  D44707  INCR  CALL    IKB         WAIT FOR A CR
47E8:  FF0D          SMI     CR
47EA:  3A00          BNZ     EBUG5       ABORT IF NOT
47EC:  F80A    OLF   LDI     LF
47EE:  D44703        CALL    OTT
47F1:  D5            RETN
                     *
                     * INPUT HEX DIGIT SUBR
47F2:  D44707  IHXD  CALL    IKB         GET A HEX DIGIT
47F5:  52            STR     R2
47F6:  FF2E          SMI     PERIOD
47F8:  C2470D        LBZ     EBUG5       ABORT IF PERIOD
47FB:  F0            LDX
47FC:  FB0D          XRI     CR
47FE:  C24820        LBZ     IHXD3       RETURN IF CR
4801:  F0            LDX
4802:  FF30          SMI     #30         CONVERT ASCII TO HEX
4804:  3B18          BL      IHXD2
4806:  FF0A          SMI     #0A
4808:  3300          BGE     IHXD1
480A:  FC0A          ADI     #0A
480C:  D5            RETN
480D:  FF07  IHXD1   SMI     #07
480F:  3B18          BL      IHXD2
4811:  FF06          SMI     #06
4813:  3318          BGE     IHXD2
4815:  FC10          ADI     #10
4817:  D5            RETN
4818:  F83F  IHXD2   LDI     WHAT
481A:  D44703        CALL    OTT
481D:  C047F2        LBR     IHXD
4820:  F0      IHXD3 LDX
4821:  F980          ORI     PARITY
4823:  D5            RETN
                     *
                     * INPUT HEX NUMBER SUBR
4824:  D447F2  IHEX  CALL    IHXD        GET A HEX DIGIT
4827:  52            STR     R2
4828:  FB0D          XRI     #0D         CHECK FOR A CR RETURN
482A:  C247EC        LBZ     OLF
482D:  22            DEC     R2          SHIFT NEW NIBBLE INTO R1
482E:  91            GHI     R1
482F:  FE            SHL
4830:  FE            SHL
4831:  FE            SHL
4832:  FE            SHL
4833:  B1            PHI     R1
4834:  81            GLO     R1
4835:  F6            SHR
4836:  F6            SHR
4837:  F6            SHR
4838:  F6            SHR
4839:  52            STR     R2
483A:  91            GHI     R1
483B:  F1            OR
483C:  B1            PHI     R1
483D:  81            GLO     R1
483E:  FE            SHL
483F:  FE            SHL
4840:  FE            SHL
4841:  FE            SHL
```

```
4842:   60                  IRX
4843:   F1                  OR
4844:   A1                  PLO      R1
4845:   3024                BR       IHEX
                *
                * OUTPUT HEX NUMBER SUBR
4847:   73      OHEX        STXD              CONVERT HEX IN ACCM INTO ASCII
4848:   F6                  SHR
4849:   F6                  SHR
484A:   F6                  SHR
484B:   F6                  SHR
484C:   FF0A                SMI      #0A
484E:   C7                  LSNF
484F:   FC07                ADI      #07
4851:   FC3A                ADI      #3A
4853:   D44703              CALL     OTT
4856:   60                  IRX
4857:   F0                  LDX
4858:   FA0F                ANI      #0F
485A:   FF0A                SMI      #0A
485C:   C7                  LSNF
485D:   FC07                ADI      #07
485F:   FC3A                ADI      #3A
4861:   D44703              CALL     OTT
4864:   D5                  RETN
                *
                * INPUT ADDRESS SUBR
4865:   D44824  IADR        CALL     IHEX
4868:   81                  GLO      R1       PUSH CONTENT OF R1 INTO R0
4869:   A0                  PLO      R0
486A:   91                  GHI      R1
486B:   B0                  PHI      R0
486C:   D5                  RETN
                *
                * OUTPUT ADDRESS SUBR
486D:   90      OADR        GHI      R0
486E:   D44847              CALL     OHEX
4871:   80                  GLO      R0
4872:   D44847              CALL     OHEX
4875:   F83A                LDI      COLON
4877:   D44703              CALL     OTT
487A:   F820                LDI      BLANK
487C:   D44703              CALL     OTT
487F:   D5                  RETN
                *
                * RE-ENTRY ROUTINE
4880:   A1      BENT1       PLO      R1       SAVE ACCM IN R1.0
4881:   F847                LDI      PAGE     RESTORE CONTENTS OF MEMORY
4883:   B6                  PHI      R6
4884:   F802                LDI      PAD
4886:   A6                  PLO      R6
4887:   06                  LDN      R6
4888:   23                  DEC      R3
4889:   53                  STR      R3
488A:   9E                  GHI      RE       RESTORE R3 AS PC
488B:   B3                  PHI      R3
488C:   F88F                LDI      BENT2
488E:   A3                  PLO      R3
488F:   D3      BENT2       SEP      R3
4890:   81                  GLO      R1
4891:   D44847              CALL     OHEX     DISPLAY CONTENTS OF ACCM
4894:   C0470D              LBR      EBUG5
```


QUESTDATA
P.O. Box 4430
Santa Clara, CA 95054

A 12 issue subscription to QUESTDATA, the publication devoted entirely to the COSMAC 1802 is $12.
(Add $6.00 for airmail postage to all foreign countries except Canada and Mexico add $2.00)
Your comments are always welcome and appreciated. We want to be your 1802's best friend.

Payment
☐ Check or Money Order Enclosed
Made payable to Quest Electronics
☐ Master Charge No
☐ Bank Americard No
☐ Visa Card No
Expiration Date: ____________

Signature ____________

☐ Renewal ☐ New Subscription

NAME ____________

ADDRESS ____________

CITY ____________ STATE ____________ ZIP ____________

Quest Electronics Documentation and Software by Roger Pitkin is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License.

FROM THE PUBLISHER

SEND IN YOUR PROGRAMS

We have been very encouraged by the content, quality and quanity of programs submitted for publication to QUESTDATA. Please keep them coming. We are interested in all types of programs including software and hardware large and small, Basic and machine language covering all subjects. Programs of 256 bytes or less are particularly needed. Also we will be publishing more hardware oriented articles, with your support naturally. The format we would like you to use would include a typed write-up with annotated code listing and a flow chart. We will continue to pay at the rate of $15.00 per published page. Programs submitted in machine readable form (Basic, Basic editor, or Editor-Assembler cassettes) and or "camera ready" will get higher payment and are likely to be published sooner. Write for our free "How to write for QUESTDATA" instructions. Please enclose a self addressed stamped business size envelope for your copy by return mail.

TIME TO RESUBSCRIBE

Since many of you began with issue #1 of volume II, many subscriptions are up for renewal. The price remains at $12.00 (with the exception of Canada and Mexico) for 12 issues in spite of increased mailing costs. It is an extremely attractive value for any 1802 computer user. We expect to have some very interesting issues during 1982 that will make your computer more usefull as well as entertaining. After several major staffing changes and production improvements it appears that issues can be published regulary and frequently with an objective of at least ten per year. An exciting new series on QUEST SUPER BASIC begins with this issue and will continue on through volume III. New subscriptions or renewals may be accomplished relatively painlessly by filling in the form at the end of this issue. If you haven't renewed yet this may be your last issue. Thank you for your support in the past.

With this issue we complete the second volume of QUESTDATA. This volume will be bound and offered as we have the first volume and will do again when we complete the succeeding volumes.

COSMAC CLUB COSMAC CLUB COSMAC CLUB COSMAC CLUB COSMAC CLUB COSMAC

24 QUESTDATA
P.O. Box 4430
Santa Clara, CA 95054

ADDRESS CORRECTION REQUESTED

BULK RATE
U.S. Postage Paid
QUEST
Electronics
Permit No. 549
Santa Clara, CA

Quest Electronics Documentation and Software by Roger Pitkin is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License.