CC DLOAD

by Frank Bogardus

First, read this article and learn about the Color Computer's ROM. Then you can use the DLOAD command to link it with the Model I.

Loading and saving programs via the cassette recorder limits the Color Computer to lighter computing tasks. I needed a faster way to access programs without the expense of disks and without the major program handlers needed in the Color Computer. That amounted to an impossible set of specs, until I discovered DLOAD.

The Command

Page 146 of the Extended Color Basic manual explains the command: "The statement DLOADM can be used to download (transfer) USR functions from another computer." But when I tried DLOADM as a command, I got TM error (type mismatch).

Page 192 gives the syntax as DLOADM X,1. The explanation says, "...loads machine-language program at specified baud. 0=300 baud, 1=1200 baud."

Using this syntax also produces a TM error. The Color Computer ROM is obviously looking for a string somewhere. I tried DLOADM "TEST",I with the same results.

I soon realized that I could use the Model I as my printer. I could send ev-

Program Listing. Model I Host

```
                00100 ;************************************************
                00200 ;MODEL I HOST FOR COLOR COMPUTER DLOAD COMMAND
                00300 ;Written 02/12/82: For Color Computer Download.
                00400 ;
                00500 ;************************************************
5200            00600          ORG    5200H
                00700 ;RS232 port pointers:
00E8            00800 STATUS   EQU    0E8H            ;Modem status register
00E9            00900 CONFIG   EQU    0E9H            ;Read sense switches
00EA            01000 CONTRL   EQU    0EAH            ;UART cntrl (in)/status
00EB            01100 DATA     EQU    0EBH            ;DATA in/out
                01200 ;
4467            01300 MESAGE   EQU    4467H           ;NEWDOS80 message handler
402D            01400 DOS      EQU    402DH           ;DOS reentry
3840            01500 KBDCH    EQU    3840H           ;Break key location
                01600 ;=========================================
                01700 ;Initialize UART to 1200 bd, par=N,stop=2,word=8.
5200 D3E8       01800 INITU    OUT    (STATUS),A
5202 DBE8       01900          IN     A,(STATUS)
5204 3E7F       02000          LD     A,07FH          ;Bit: 0111 1111
5206 D3EA       02100          OUT    (CONTRL),A      ;Sets all but baud.
5208 3E77       02200          LD     A,77H
520A D3E9       02300          OUT    (CONFIG),A      ;Sets baud.
                02400 ;
520C DD211554   02500          LD     IX,CHKSUM
5210 31FFFF     02600          LD     SP,0FFFFH       ;Set to top of memory
                02700 ;=========================================================
                02800 ;General scheme:
                02900 ;  1. Get, echo, display Start Byte (STBYT=8AH).
                03000 ;  2. Get, display Name: save to NAMBUF
                03100 ;  3. Get checksum, echo 0C8H.
                03200 ;  4. Send Flag 1 and Flag 2 (FLG12)
                03300 ;  5. Calculate, send FLG12 checksum (CHKSUM).
                03400 ;  6. Get character: 9FH=Okay; 0BCH=Error.
                03500 ;  7. Get, save 2 byte block count (BLOCK)
                03600 ;  8. Get checksum, echo 0C8H.
                03700 ;  9. Send 1 byte, initialize checksum.
                03800 ; 10. Send 80 bytes, CHKSUM.
                03900 ; 11. Determine if more or end.
                04000 ;
                04100 ;=========================================
                04200 ;1.Get, echo, display start byte.
5213 CDEA53     04300 ONE      CALL   CLRBUF          ;Clear all buffers
5216 3E0E       04400          LD     A,0EH           ;Turn on cursor
5218 CDCD53     04500          CALL   H33
521B 214A54     04600          LD     HL,OKMSG        ;Main prompt
521E CD6744     04700          CALL   MESAGE          ;To screen
5221 CD9E53     04800 ONEA     CALL   IN232           ;Get byte or Break
5224 FE8A       04900          CP     8AH             ;DLOAD called?
5226 280A       05000          JR     Z,OKONE         ;Go if yes
5228 FEDB       05100          CP     0DBH            ;PROGRAM SAVE called?
522A CA1C53     05200          JP     Z,LLIST         ;Go if yes
522D CDBA53     05300          CALL   CHKERR          ;Error called from CC?
5230 18EF       05400          JR     ONEA            ;Keep trying for input
5232 CD8F53     05500 OKONE    CALL   OUT232          ;Echo character
5235 216554     05600          LD     HL,DLDMSG       ;Get DLOAD message
5238 CD6744     05700          CALL   MESAGE          ;To screen
                05800 ;2. Get, display name, save to NAMBUF
```

Listing continues

The Key Box

Model I
32K RAM
Assembly Language
One Disk Drive
RS-232
4-Wire Cable
and
Color Computer
16K RAM
Extended Color Basic
RS-232

Listing continued

```
523B 0608     05900 TWO     LD      B,8             ;Max length
523D 217B55   06000         LD      HL,NAMBUF       ;Storage
5240 CD9E53   06100 GET1    CALL    IN232           ;Get byte
5243 CDBA53   06200         CALL    CHKERR          ;Error called from CC?
5246 77       06300         LD      (HL),A          ;Save byte
5247 CDCD53   06400         CALL    B33             ;Display byte
524A 23       06500         INC     HL              ;Bump buffer pointer
524B 05       06600         DEC     B               ;Decrement byte count
524C 20F2     06700         JR      NZ,GET1         ;Get more if B>0
524E 3E0D     06800         LD      A,0DH           ;For screen clarity
5250 CDCD53   06900         CALL    B33             ;Display carraige return
              07000 ;3. Get checksum, load program, echo 0C8H.
5253 CD9E53   07100 THREE   CALL    IN232           ;Get byte
5256 CDBA53   07200         CALL    CHKERR          ;Error called from CC?
5259 CD9D55   07300         CALL    RDPRG           ;Get program into buffer
525C 3EC8     07400         LD      A,0C8H          ;"OK" for CC
525E CD8F53   07500         CALL    OUT232          ;Send
5261 216E54   07600         LD      HL,BLKMSG       ;Block message
5264 CD6744   07700         CALL    MESAGE          ;Message handler
              07800 ;4. Send Flag 1 and Flag 2 (FLG12)
5267 3A0F54   07900 FOUR    LD      A,(FLG12)       ;Get Flag 1
526A CD8F53   08000         CALL    OUT232          ;Send
526D DD360000 08100         LD      (IX),0          ;Initialize Checksum
5271 CDC653   08200         CALL    ADDCS           ;Add to Checksum
5274 3A1054   08300         LD      A,(FLG12+1)     ;Get Flag 2
5277 CD8F53   08400         CALL    OUT232          ;Send
527A CDC653   08500         CALL    ADDCS           ;Add to Checksum
              08600 ;5. Send checksum.
527D 3A1154   08700         LD      A,(CHKSUM)      ;Get Checksum value
5280 CD8F53   08800         CALL    OUT232          ;Send
5283 DD360000 08900         LD      (IX),0          ;Reset Checksum.
              09000 ;6. Get character (9FH = Okay, 0BCH = Error)
5287 CD9E53   09100 SIX     CALL    IN232           ;Get byte
528A FE97     09200         CP      97H             ;"OK" sent from CC?
528C 2805     09300         JR      Z,OKSIX         ;Go if yes
528E CDBA53   09400         CALL    CHKERR          ;Error called from CC?
5291 18F4     09500         JR      SIX             ;Keep checking
5293 CD8F53   09600 OKSIX   CALL    OUT232          ;Echo 97H
              09700 ;7. Get, save 2 bytes (BLOCK)
5296 CD9E53   09800 SEVEN   CALL    IN232           ;Get byte
5299 CDBA53   09900         CALL    CHKERR          ;Error called from CC?
529C DD360000 10000         LD      (IX),0          ;Initialize Checksum
52A0 321254   10100         LD      (BLOCK),A       ;Save
52A3 CDC653   10200         CALL    ADDCS           ;Add to checksum
52A6 CD9E53   10300         CALL    IN232           ;Get second byte
52A9 CDBA53   10400         CALL    CHKERR          ;Error called from CC?
52AC 321354   10500         LD      (BLOCK+1),A     ;Save
52AF CDC653   10600         CALL    ADDCS           ;Add to Checksum
              10700 ;8. Get checksum, echo 0C8H.
52B2 CD9E53   10800 EIGHT   CALL    IN232           ;Get checksum from CC
52B5 3EC8     10900         LD      A,0C8H          ;Load "OK" byte
52B7 CD8F53   11000         CALL    OUT232          ;Send
              11100 ;9. Send 1 byte, initialize checksum.
52BA DD360000 11200 NINE    LD      (IX),0          ;Initialize Checksum
52BE 3A7954   11300         LD      A,(MARK)        ;Get "last block" mark
52C1 FE00     11400         CP      0               ;0 means not last block
52C3 2804     11500         JR      Z,NOTEND        ;Go if not last block
52C5 3E00     11600         LD      A,0             ;Flag 3 value, last block
52C7 1803     11700         JR      ISEND           ;Skip Flag 3 load
52C9 3A7654   11800 NOTEND  LD      A,(FLG3)        ;Load normal Flag 3 value
52CC CD8F53   11900 ISEND   CALL    OUT232          ;Send
52CF CDC653   12000         CALL    ADDCS           ;Maintain Checksum
              12100 ;10. Send 80 bytes, Checksum
52D2 0680     12200 TEN     LD      B,80H           ;Byte count
52D4 2A7754   12300         LD      HL,(PRGPTR)     ;Current program byte
52D7 7E       12400 SEND1   LD      A,(HL)          ;Load to A
52D8 CD8F53   12500 NODS    CALL    OUT232          ;Send
52DB F5       12600         PUSH    AF              ;Preserve on stack
52DC CDC653   12700         CALL    ADDCS           ;Maintain Checksum
52DF F1       12800         POP     AF              ;Return to A
52E0 FEFF     12900         CP      0FFH            ;End-of-program marker?
52E2 2005     13000         JR      NZ,GOSEND       ;Go if not
52E4 3E01     13100         LD      A,1             ;Last block mark
52E6 327954   13200         LD      (MARK),A        ;Set mark
52E9 23       13300 GOSEND  INC     HL              ;Bump program pointer
52EA 05       13400         DEC     B               ;Decrement counter
52EB 20EA     13500         JR      NZ,SEND1        ;More if not done
52ED 227754   13600         LD      (PRGPTR),HL     ;Done, save pointer
52F0 3A1154   13700         LD      A,(CHKSUM)      ;Get updated Checksum
52F3 CD8F53   13800         CALL    OUT232          ;Send
52F6 3E2E     13900         LD      A,'.'           ;One "." for each block
52F8 CDCD53   14000         CALL    B33             ;To display
              14100 ;11. Determine if more or done
52FB 3A7954   14200         LD      A,(MARK)        ;Last block mark
52FE FE01     14300         CP      1               ;Last block yet?
5300 C28752   14400         JP      NZ,SIX          ;Again if not
5303 3A7A54   14500         LD      A,(MARK2)       ;End block mark
5306 FE00     14600         CP      0               ;Send end yet?
5308 C21353   14700         JP      NZ,ALLDUN       ;If yes, all done
530B 3E01     14800         LD      A,1             ;End block value
530D 327A54   14900         LD      (MARK2),A       ;Set mark
5310 C38752   15000         JP      SIX             ;Send another block
5313 211E54   15100 ALLDUN  LD      HL,DUNMSG       ;Message to display
5316 CD6744   15200         CALL    MESAGE          ;Message handler
5319 C31352   15300         JP      ONE             ;Go to main prompt
              15400 ;=========================================
531C F5       15500 LLIST   PUSH    AF              ;Save for echo back
531D D3E8     15600         OUT     (STATUS),A      ;Reinitialize UART
531F DBE8     15700         IN      A,(STATUS)
5321 3E3F     15800         LD      A,3FH           ;7-bit, 2-stop, no parity
5323 D3EA     15900         OUT     (0EAH),A
5325 3E77     16000         LD      A,77H           ;1200 baud
5327 D3E9     16100         OUT     (0E9H),A
5329 F1       16200         POP     AF
532A 217B55   16300         LD      HL,NAMBUF       ;To save name
532D CD8F53   16400         CALL    OUT232          ;Echo
5330 CD9E53   16500 GETNXT  CALL    IN232           ;Get name byte
5333 CD8F53   16600         CALL    OUT232          ;Echo
```

Listing continues

erything out the RS-232 serial printer line, receive it with 7-bit, no-parity, 600-baud protocol, save it to disk, and print it later. This capability, and a 6809 disassembler, led to a full ROM printout. The command tables included CLOAD and CLOADM, as expected, but they contained DLOAD (not DLOADM) as the sole downloading command. By dropping the M I got the computer to hand up, suggesting that it was looking for a serial input.

I wanted to write a Color Computer monitor to load machine-language programs but I needed to find the ROM locations to send and receive single bytes through the RS-232 serial port. That solution would not allow loading Basic programs, but DLOAD was not supposed to do that, anyway.

However, after experimenting, I found that DLOAD is very good at loading Basic programs. It may load machine-language programs as well, but I haven't yet cracked the code for that.

The DLOAD Command

The program requires a thorough understanding of the DLOAD command. The only version I have been able to get a response from has the syntax: DLOAD "file name," X, where X is a 0 for 300-baud operation and a 1 for 1,200-baud operation. The Color Computer stores the countdown baud value for the DLOAD command at 0E6H. The value is 2CH for 1,200 baud (the initialized value) and B0H for 300 baud. You can set other baud rates by POKEing other values. This baud counter is separate from the two-byte counter for the printer function (maintained at 95H-96H), so you can maintain two different baud rates. Also, the values for a given baud rate are slightly higher for the DLOAD command than for the printer.

Following is a description of the ROM actions upon recognition of the DLOAD command:

- Control passes to 8C18H, where the name reads into a buffer at 01D2-01D9H, and the baud rate is deciphered and stored in 00E6H. The name is left justified in an 8-byte buffer and padded on the right by spaces.
- A handshaking byte of 8AH is sent to RS-232 out, and comes back as an echo from the RS-232.
- The ROM sends out an 8-byte name block, and it maintains a checksum of the characters sent.
- The checksum goes to RS-232 out, presumably for the host to verify the name; RS-232 in is checked for C8H.
- The ROM checks RS-232 in for

Listing continued

```
5336 CDCD53   16700          CALL    H33             ;Send to screen
5339 FE0D     16800          CP      0DH             ;End of name?
533B 2804     16900          JR      Z,GETPRG        ;If yes, get program
533D 77       17000          LD      (HL),A          ;Save name
533E 23       17100          INC     HL              ;Bump pointer
533F 18EF     17200          JR      GETNXT          ;Get more
5341 21DB55   17300 GETPRG   LD      HL,PROGRM       ;Program buffer pointer
5344 CD9E53   17400 GETBYT   CALL    IN232           ;Get program byte
5347 CD8F53   17500          CALL    OUT232          ;Echo
534A CDCD53   17600          CALL    H33             ;Send to screen
534D FE5C     17700          CP      5CH             ;Shift/clear received?
534F 2008     17800          JR      NZ,GETMOR       ;If not, get more
5351 CD9E53   17900          CALL    IN232           ;Get carraige return
5354 CD8F53   18000          CALL    OUT232          ;Echo
5357 1804     18100          JR      SAVPRG          ;Go to Save routine
5359 77       18200 GETMOR   LD      (HL),A          ;Save program byte
535A 23       18300          INC     HL              ;Bump program pointer
535B 18E7     18400          JR      GETBYT          ;Get next byte
535D 36FF     18500 SAVPRG   LD      (HL),0FFH       ;Program end marker
535F 217B54   18600          LD      HL,BUFFER       ;Disk buffer
5362 117B55   18700          LD      DE,NAMBUF       ;File Control Block
5365 0600     18800          LD      B,0             ;LRL = 256
5367 CD2044   18900          CALL    OPENNW          ;Open file
536A C20554   19000          JP      NZ,FILERR       ;Check for error
536D DDE5     19100          PUSH    IX              ;Preserve IX on stack
536F DD21DB55 19200          LD      IX,PROGRM       ;IX = program pointer
5373 DD7E00   19300 SAVBYT   LD      A,(IX)          ;Get program byte
5376 FEFF     19400          CP      0FFH            ;End of program?
5378 2807     19500          JR      Z,SAVEND        ;End routine if yes
537A CD1B00   19600          CALL    WRITE           ;Byte to disk
537D DD23     19700          INC     IX              ;Bump program pointer
537F 18F2     19800          JR      SAVBYT          ;Get next byte
5381 CD2844   19900 SAVEND   CALL    CLOSE           ;Close file
5384 213F54   20000          LD      HL,SAVMSG       ;Message to display
5387 CD6744   20100          CALL    MESAGE          ;Message handler
538A DDE1     20200          POP     IX              ;Restore IX from stack
538C C30052   20300          JP      INITU           ;Reinitialize
              20400 ;=============================================
              20500 ;SUBROUTINES
              20600 ;SERIAL OUT: CHARACTER IN A.
538F F5       20700 OUT232   PUSH    AF
5390 CD9753   20800          CALL    READY
5393 F1       20900          POP     AF
5394 D3EB     21000          OUT     (0EBH),A        ;Send character out RS232
5396 C9       21100          RET
5397 DBEA     21200 READY    IN      A,(CONTRL)      ;Loop until next char
5399 CB77     21300          BIT     6,A             ;character can go out.
539B 28FA     21400          JR      Z,READY
539D C9       21500          RET
              21600 ;----------------------
              21700 ;SERIAL IN: CHARACTER IN A.
539E C5       21800 IN232    PUSH    BC              ;Preserve BC
539F CDA453   21900          CALL    IN1             ;Get byte
53A2 C1       22000          POP     BC
53A3 C9       22100          RET
53A4 0EEA     22200 IN1      LD      C,CONTRL        ;UART control register
53A6 ED40     22300 AGN1     IN      B,(C)           ;Get control status
53A8 CB78     22400          BIT     7,B             ;Test for Data Received
53AA 200A     22500          JR      NZ,GOTBYT       ;Go if set
53AC 3A4038   22600          LD      A,(KBDCH)       ;Check keyboard if not
53AF FE04     22700          CP      4               ;Test for break
53B1 20F3     22800          JR      NZ,AGN1         ;Loop til character rec'd
53B3 C32D40   22900          JP      DOS             ;To DOS if break
53B6 DBEB     23000 GOTBYT   IN      A,(DATA)        ;Get byte
53B8 B7       23100          OR      A               ;Set condition codes
53B9 C9       23200          RET
              23300 ;----------------------
53BA FEBC     23400 CHKERR   CP      0BCH            ;Error from CC?
53BC C0       23500          RET     NZ              ;Return if not
53BD 211454   23600          LD      HL,ERRMSG       ;Message to screen
53C0 CD6744   23700          CALL    MESAGE          ;Message handler
53C3 C30052   23800          JP      INITU           ;Reinitialize
              23900 ;----------------------
53C6 DDAE00   24000 ADDCS    XOR     (IX)            ;Maintain checksum
53C9 DD7700   24100          LD      (IX),A
53CC C9       24200          RET
              24300 ;----------------------
53CD D5       24400 H33      PUSH    DE              ;Preserve registers
53CE DDE5     24500          PUSH    IX
53D0 FDE5     24600          PUSH    IY
53D2 CD3300   24700          CALL    33H             ;One byte to screen
53D5 FDE1     24800          POP     IY              ;Restore registers
53D7 DDE1     24900          POP     IX
53D9 D1       25000          POP     DE
53DA C9       25100          RET
              25200 ;----------------------
53DB F5       25300 SETBUF   PUSH    AF              ;Preserve A
53DC 217B55   25400          LD      HL,NAMBUF       ;Buffer to clear
53DF 0621     25500          LD      B,33            ;Buffer length
53E1 3E20     25600          LD      A,20H           ;Load blank spaces
53E3 77       25700 SB1      LD      (HL),A          ;Begin clearing
53E4 23       25800          INC     HL
53E5 05       25900          DEC     B
53E6 20FB     26000          JR      NZ,SB1          ;Clear until countdown
53E8 F1       26100          POP     AF
53E9 C9       26200          RET
              26300 ;----------------------------
53EA 21DB55   26400 CLRBUF   LD      HL,PROGRM       ;Buffer to clear
53ED F3       26500          DI                      ;Avoid interference
53EE 3600     26600          LD      (HL),0          ;Load with nulls
53F0 11DC55   26700          LD      DE,PROGRM+1     ;Destination of nulls
53F3 010080   26800          LD      BC,8000H        ;Length of buffer
53F6 EDB0     26900          LDIR                    ;Clear buffer
53F8 FB       27000          EI                      ;Enable interrupts
53F9 CDDB53   27100          CALL    SETBUF          ;Clear name buffer
53FC 3E00     27200          LD      A,0             ;Reset:
53FE 327954   27300          LD      (MARK),A        ;- last block mark
5401 327A54   27400          LD      (MARK2),A       ;- end mark
5404 C9       27500          RET
```

Listing continues

three bytes in succession; the first two (flags 1 and 2) are stored and the third is a checksum of the first two. The two stored bytes of the program later determine the type of input (see "Flags").

- Upon successful receipt of the checksum, the ROM sends a 97H to RS-232 out, and echoes back through RS-232 in.
- Two bytes are sent to RS-232 out, then the checksum of the two is sent; a C8H must be returned to RS-232 in to indicate correct receipt of the checksum.
- The ROM requests a single byte (flag 3) from RS-232 in, and initializes the checksum (see flags).
- A block of 128 bytes is received from RS-232 in, then the checksum of the block plus flag 3; the block is stored at 01DAH, the cassette buffer.
- Depending on the flags, the Color Computer cycles for another 128-byte block declares an error, or ends successfully. Unless an error occurs, the loaded block is placed in the Basic program statement table, with all pointers correctly adjusted.

Checksum

The program maintains the checksum by performing an exclusive-or (XOR) between the previous valuc of the checksum and the ASCII of the value last received. The resulting value becomes the updated checksum. The checksum value always initializes to zero, and reinitializes each time its value goes to RS-232 out.

Flags

Flags 1 and 2 are the two bytes sent to the Color Computer when the ROM checks the RS-232. When the ROM requests a single byte from RS-232 in, flag 3 is sent. Flag 3 is figured into the data-block checksum, and sends it as the opening byte for each successive block.

I haven't yet figured out the flags' general purposes, but I have figured out the results for some of the combinations. I have also worked out a combination to successfully load a Basic program, which was my primary goal.

If flag 1 is an FFH the Color Computer returns an NE error, regardless of the other flags' values. The manual doesn't list that error. It might be the no-error error that I always feared would show up some day.

If flag 2 is an 00H, the Color Computer returns an FM error, which is a format error. This occurs in tape loading when CLOAD encounters a machine-language program, and when

Listing continued

```
                27600 ;-------------------------
5405 B7         27700 FILERR  OR      A               ;Set condition codes
5406 C8         27800         RET     Z               ;If no error
5407 CBFF       27900         SET     7,A             ;Prepare for display
5409 CD0944     28000         CALL    4409H           ;Display message
540C C30052     28100         JP      INITU           ;Reinitialize
                28200 ;=========================================================
                28300 ;MARKERS, BUFFERS
540F 0101       28400 FLG12   DEFW    0101H
5411 00         28500 CHKSUM  DEFB    0               ; = IX
5412 0000       28600 BLOCK   DEFW    0               ;Count from CC
                28700 ;---------------------------------------
                28800 ;MESSAGES
5414 49         28900 ERRMSG  DEFM    'I/O ERROR'
541D 0D         29000         DEFB    0DH
541E 0A         29100 DUNMSG  DEFB    0AH
541F 2A         29200         DEFM    '*** DONE ***'
542B 0D0A       29300         DEFW    0A0DH
542D 50         29400 LOADOK  DEFM    'PROGRAM LOAD OKAY'
543E 0D         29500         DEFB    0DH
543F 46         29600 SAVMSG  DEFM    'FILE SAVED'
5449 0D         29700         DEFB    0DH
544A 0A0A       29800 OKMSG   DEFW    0A0AH
544C 43         29900         DEFM    'Color Computer Host'
545F 0A         30000         DEFB    0AH
5460 2A         30100         DEFM    '* OK'
5464 0D         30200         DEFB    0DH
5465 0A         30300 DLDMSG  DEFB    0AH
5466 44         30400         DEFM    'DLOAD: '
546D 03         30500         DEFB    03H
546E 42         30600 BLKMSG  DEFM    'BLOCKS '
5475 03         30700         DEFB    03
                30800 ;---------------------------------
                30900 ;MORE MARKERS
5476 80         31000 FLG3    DEFB    80H
5477 DB55       31100 PRGPTR  DEFW    PROGRM
5479 00         31200 MARK    DEFB    0
547A 00         31300 MARK2   DEFB    0
                31400 ;=================================================
                31500 ;DISK READ ROUTINE
4424            31600 OPEN    EQU     4424H           ;Existing file only
0013            31700 READ1   EQU     0013H           ;One byte read
4420            31800 OPENNW  EQU     4420H           ;New or existing file
001B            31900 WRITE   EQU     001BH           ;One byte read
4428            32000 CLOSE   EQU     4428H           ;File close
0100            32100 BUFFER  DEFS    256             ;User file buffer
557B 20         32200 NAMBUF  DEFM    '                '
559D D9         32300 RDPRG   EXX                     ;Use alternate registers
559E 08         32400         EX      AF,AF'
559F DDE5       32500         PUSH    IX              ;Preserve IX
55A1 117B55     32600         LD      DE,NAMBUF       ;File Control Block
55A4 217B54     32700         LD      HL,BUFFER
55A7 0600       32800         LD      B,0             ;LRL = 256
55A9 CD2444     32900         CALL    OPEN            ;If it exists
55AC C20554     33000         JP      NZ,FILERR       ;Check for error
55AF DD21DB55   33100         LD      IX,PROGRM       ;Pointer to load buffer
55B3 CD1300     33200 RDLOOP  CALL    READ1           ;One byte from disk
55B6 2817       33300         JR      Z,RDGO          ;Go if no error
55B8 FE1C       33400         CP      1CH             ;End of File error
55BA C20554     33500         JP      NZ,FILERR       ;Display other error
55BD DD3600FF   33600         LD      (IX),0FFH       ;EOF Marker in memory
55C1 21DB55     33700         LD      HL,PROGRM       ;Reset to program start
55C4 227754     33800         LD      (PRGPTR),HL     ;Set pointer
55C7 212D54     33900         LD      HL,LOADOK       ;Message to display
55CA CD6744     34000         CALL    MESAGE          ;Message handler
55CD 1807       34100         JR      PREXX           ;Prepare to exit
55CF DD7700     34200 RDGO    LD      (IX),A          ;Save byte loaded
55D2 DD23       34300         INC     IX              ;Bump pointer
55D4 18DD       34400         JR      RDLOOP          ;Get more
55D6 DDE1       34500 PREXX   POP     IX              ;Restore IX
55D8 D9         34600         EXX                     ;Restore main registers
55D9 08         34700         EX      AF,AF'
55DA C9         34800         RET
55DB            34900 PROGRM  EQU     $               ;Program buffer
5200            35000         END     INITU
00000 TOTAL ERRORS
```

CLOADM encounters a Basic program. This byte differentiates the two for the DLOAD command.

If flags 1 and 2 form 0001H, 00FFH, 0100H, or 01FFH, depending on the value of flag 3, no error returns.

If flag 3 is 00H, one data block is sent and the command considers itself finished. The block received is not processed as a Basic program. I use this as a way of ending the program load in the host program.

If flag 3 is 01H, the command looks for continuous blocks; they are not processed as Basic programs within the Color Computer. This value can be part of the key to loading a machine-language program.

If flag 3 is FFH, the command fails with a DS error, or a Direct Statement in File error. This error also occurs when a CLOAD command encounters a data tape, suggesting that the DLOAD command may also be available for loading data from the RS-232.

If Flag 3 is 80H, the DLOAD command processes the block as a Basic program and anticipates receipt of the next block. The host program uses this value until the entire program has downloaded, and then switches to 00H to signify the end of file.

ROM Locations

In deciphering the DLOAD command I discovered many general-purpose Color Computer ROM routines useful in two-way serial communication. I'll pass these along for those interested in working toward their own machine-language routines. The host program requires no machine-language monitor in the Color Computer. With a bit of work and imagination, a small monitor could facilitate full eight-bit interchanges with any computer.

The routines use the hardware stack area to keep track of what's going on; so use the entry and exit routines.

8DBCH—Gets single characters from RS-232 input, with characters received in A register. This call keeps trying for an input for a specified number of tries. The one-byte value at 00E7H times the two-byte value at 008AH determines the number of tries. These are initialized on power-up to five and 0000, giving five times 65536 tries before a return. The Y register must point to an area where at least three bytes can be used by the subroutine. I recommend the following routine:

```
CLRA          Clear A register
PSHS A,B,X    Set Hardware Stack
LEAY 0,S      Point Y to Stack
CALL $8DBCH   Get Character
Process character received in register A.
```

8E0CH—Sends single characters in register A to RS-232 output. The value at 00E6H determines the baud rate. The Y register must point to an area where at least three bytes can be used by the subroutine. Use the following routine:

```
LDA CHAR      Load character
PSHS A,B,X    Set Hardware Stack
LEAY 0,S      Point Y to Stack
CALL 8E0CH    Send Character
```

8DB8H—Sends characters out and waits for echo back. Set up this routine the same as 8E0CH, above.

8D72H—Sends characters out, waits for echo, and maintains checksum in the XOR format. Using the routine setups above, a PULS A,B,X should result in the last character received remaining in A, with the current checksum value in B.

8D62H—Sends out current checksum; resets checksum to zero; waits for a C8H returned; declares an error if not received (IO ERROR).

The Program

Program Listing 1 is a host program for the Model 1. You can receive any program keyed into the Color Computer from the Model 1 and store them on disk.

You can send any program stored on disk in ASCII format to the Color Computer, whether originally keyed into the Color Computer or the Model I.

You can list, run or manipulate within the Color Computer any program loaded through the command as if keyed in or loaded from cassette.

All controls come from the Color Computer keyboard; you don't have to touch the Model I during program exchanges.

The last feature lets you use the Model I as a remote disk-recording device for all Color Computer programs, with the only connection needed between the two being a four-conductor wire.

The host program initializes the Model I RS-232 to eight bits, no parity, 1200 baud, then polls the RS-232 port waiting for one of two bytes: an 8AH, indicating the Color Computer is looking to download a program, or DBH, indicating a program is about to come in to be saved. If the computer receives the former, the program sets the name, loads the program from disk and runs through the remainder of the protocol.

To send a file from the Color Computer to the Model I, the file is essentially printed. A complication arises with the Color Computer 1.0 ROM: it prints a seven-bit rather than an eight-bit word, and the print protocol initializes at 600 baud. You can correct this by POKEing 29H in memory location 96H. The former problem is dealt with in the host program by treating one of the stop bits as part of the ASCII code. So, when the Color Computer sends 5BH (the left bracket, formed by shift/down arrow), the host recognizes the character by expecting the top bit to be on, to receive DBH. The Color Computer command to initiate a save is PRINT #−2, (shift/down arrow) FILENAME (enter).

The receipt of the misread 5BH causes the host to reinitialize to a seven-bit protocol, and to treat the incoming characters up to a carriage return as the file name. The Color Computer issues the command LLIST, which sends the program as an ASCII printout. When finished, a terminator signal is needed. The host program recognizes a 5CH, formed on the Color Computer with a shift/clear, and displayed as a backslash. When this byte is received, the program saves to disk under the received name and the RS-232 UART reinitializes to the eight-bit protocol.

This procedure allows an automatic save on the Model I by appending a Remark backslash to the end of any Color Computer program. It has the disadvantage of making the backslash a disallowed character in the body of a Basic program. If this proves a problem, the save command can be made more particular by checking that the most recent byte entered into the buffer was a carriage return (0DH), since the backslash can never appear as the leading character of the next Basic line.

I'm told the 1.1 ROM uses an eight-bit protocol. If that is true, the reinitialization portion of the save program (lines 15600–16100) will not be necessary. The host will be looking for 5BH, rather than DBH, to start the save process. Make the latter change at line 5100.

If you would rather run the whole thing at 300 baud, change the value loaded into the A register at lines 2200 and 16000 to 55H rather than 77H. ■

Frank Bogardus can be reached at RD #2, Box 312, Rensselaer, NY 12144.

RELIABLE LOW-COST HIGH-SPEED TAPING

STANDARD RECORDER AND CASSETTES. . . .NO REPROGRAMMING HASSLES. . . .NO EXPENSIVE MODIFICATIONS.

NO-FUSS HIGH SPEED SOFTWARE

KWICOS (Mod 1, 4k to 48k) **$26**
KWIK Cassette Operating System for Mod 1. The easy-to-use Level II enhancement for reliable fast taping (select 1000-3000 baud). Features: save, load, verify, search, chain-load, catalog, and test-read of both BASIC and machine-code programs . . . plus: long pgm names, passwords, debounce, slow 'list', self 'backup', and more.

KOS3 (Mod 3, 16k to 48k) **$26**
The KWIK Cassette Operating System for Model 3. All KWICOS features at 2200 baud, plus KWIK set of: clock display, Time, Date, Cassette high/low, I/O routing, etc.

KWIKIT (specify Model) **$12**
Mini-system for BASIC programs only. EasyLoad 1000 baud for Mod 1, 2200 baud for Mod 3. Many KWICOS features.

KWINK (Model 1, 4k-48k) **$15**
Makes stand-alone fast-loading (2x-6x) copies of any standard 500 baud "SYSTEM" program. (At 6x, 3 minute program loads in 44 sec!)

KLOAD (Model 1) **$15**
Similar to KWINK, but for BASIC pgms only. (Specify 16-32-48k)

KLOAN (Mod 3, 16k-48k) **$12**
Makes 500 or 1500 baud copy of any other standard 'system' pgm.

KWIK Software
Box 328
Bolivar, MO 65613
Phone (417) 326-7154

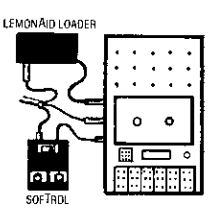
SKEPTICAL? Any doubt that KWIK model I speed-up programs work? Send $3 for DEMO tape (refund with first order).
WE FLAT GUARANTEE...If you are not satisfied with ANY product in this ad, you get your money back. No hassles. No delay.

NO-FUSS ANY-SPEED HARDWARE

The amazing LemonAid Loader takes the 'finickies' out of Model I loads, but even more amazing . . . make KWIK copies of your Model I programs and load them flawlessly at 6 < speed! Instant installation . . . simply plug between your cable and recorder. No internal batteries or external power needed. NO VOL. JUGGLING!

All Loaders for Model I, switchable to Model III
Model L80 Deluxe loader plus speaker/earphone output jack.....$19.99
Model L80M Like L80 but with built-in audio monitor.....$23.50
Model LS80M L80M plus LSS SOFTROL (below) in one.....$37.50
(Above for Radio Shack CTR80 or 80A. No modifications needed. May or may not work with other recorders.) LOADS I, III, & CoCo
Model LS81M Loader/Softrol/Monitor for new RS CCR81.
Built-in auto-powered amplifier for marginal tapes.....$44.50

Model LSS-2 Solid state SOFTROL (1) eliminates switch-hits (2) pushbutton and slide switch control of CTR Motor makes tape positioning a snap with no plug pulling (3) cushioned motor-off delay pulls end of programs past CTR pinch rollers so you get no pinch-hits, plus (4) automatically gaps between saved programs. May be used to computer-switch other DC loads up to 15 volts and 1.5 A. Works with all standard-plug recorders and computers.....$18.99



LEMONS TECH #358
325 N. Hwy 65
P.O. Drawer 429
Buffalo, MO 65622
(417) 345 7643

Call either number 'til 10pm most any day for orders or info. US ppd. $4 COD or overseas (except APO/FPO). MO res. add tax.