6802 Serial Terminal

https://hackaday.io/project/176175-6802-serial-terminal

An RS232C serial terminal using MC6802 & LED dot matrix display modules.

Mitsuru Yamada

Description

In 2007, I made a computer named PERSEUS-3[1] and start to use it. As a command input and output serial terminal, I had used PC terminal software from that days but I began to desire the development of an independent terminal machine that works without a PC in 2018. Its features are as follows.
(1) For a character display, former HP dot matrix alphanumeric LED displays were used.
(2) A key scanning, sending characters by serial interface, receiving characters by serial interface and setting to character fonts to LED display module are performed by 6802 single task machine codes simultaneously.
(3) PERSEUS-3 was used as a hardware emulator.
(4) Built in an aluminum die-cast enclosure to make it small and rugged, battery operation is possible.

Details

1. Hardware configuration

The circuit is made up of two blocks shown in Fig. 1. One is a minimally configured computer block with a 6802 CPU. The other is a key matrix and LED display modules block. An 8-bit parallel read/write port connects the blocks to each other. This parallel port has the same specifications as the parallel port of the PERSEUS-3 6802 homebrew computer. The schematic is shown in the Appendix 1 and 6802_SERIAL_TERMINAL_schematic_1.pdf[2].

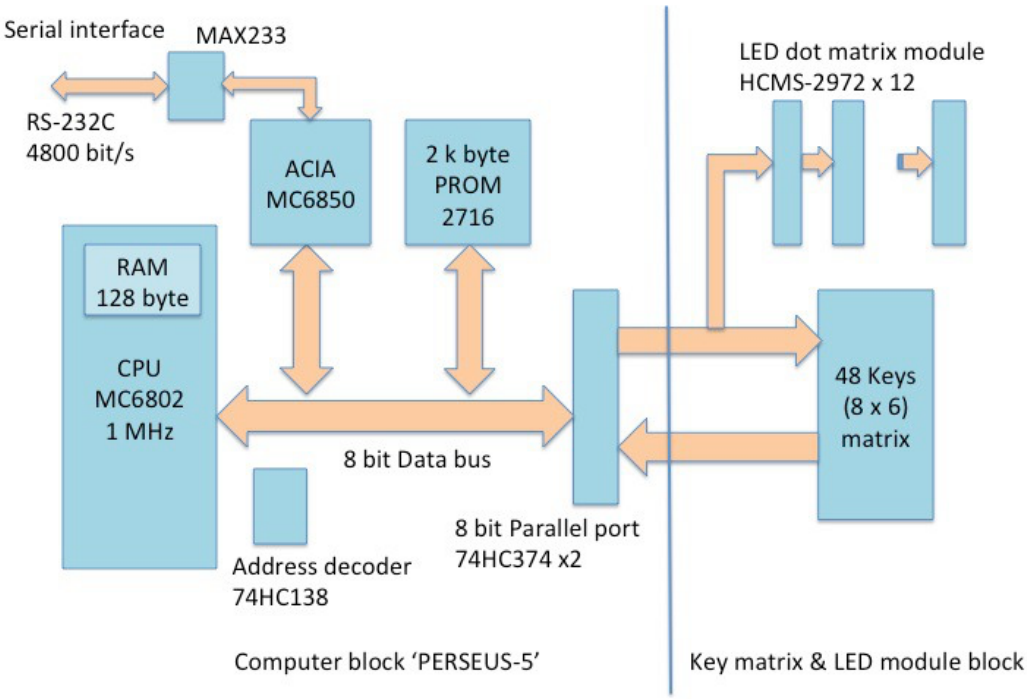
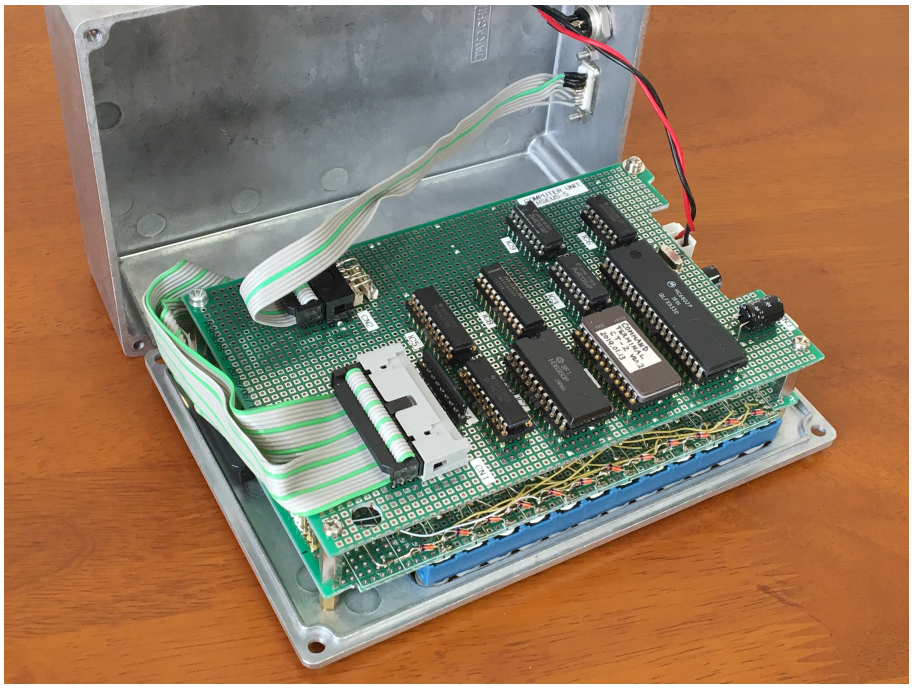


Fig 1 6802 Serial Terminal hardware block diagram

2. Computer block

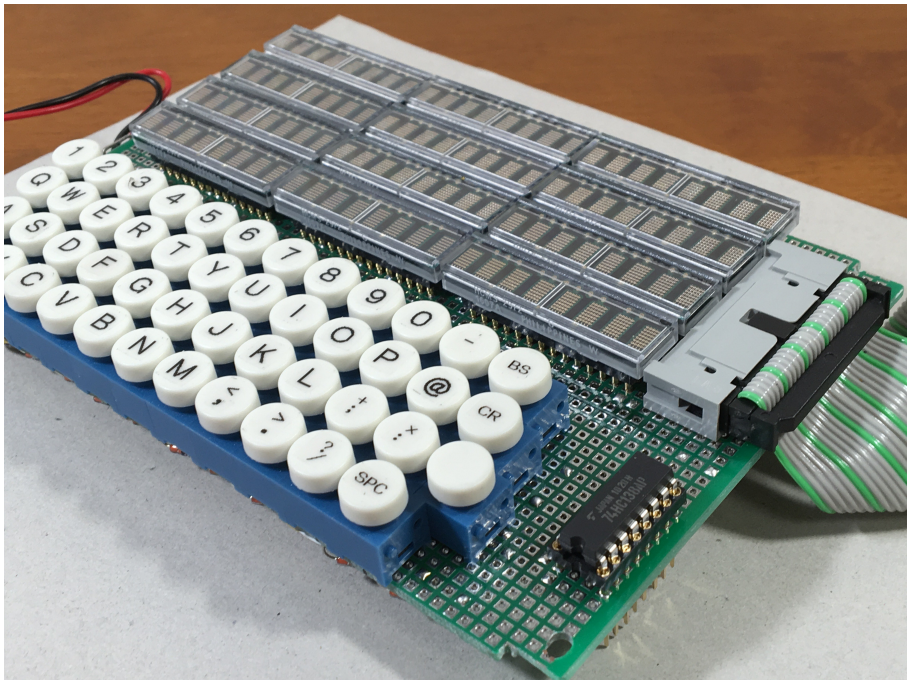
I chose the Motorola's MC6802 as the CPU for this computer same as PERSEUS-3. A RAM is used of 128 bytes embedded in MC6802 and no external RAM device. A ROM is 2 k byte 2716 PROM. A serial interface is the Asynchronous Communication Interface Adapter (ACIA) MC6850 (HD6350 Hitachi). Parallel interface to the KEY & LED block is configured with a standard logic 74HC374. I named this computer block PERSEUS-5.



3. Key matrix and LED display mod

The 48 keys full keyboard is configured 8 x 6 key matrix. Three bits of output port signals are decoded to 6 lines key scanning signal. 8 bits of input port are used to read the key scan result. For the switches, 48 Miyama electric DS-660RC-W(Top color white) pushbutton switch are used.

The terminal has 24 x 4 total 96 characters alphanumeric display. So, 12 LED modules HCMS-2972[3] are connected by daisy chain. The HCMS-2972 is an 8 characters 5 x 7 dot matrix alphanumeric display. This device requires serial input of character font patterns. For the serial input of the display, 4 bits of the output port of the parallel interface are used. The data obtained by referencing the font table from the ASCII code is serialized and sent to the LED module.



4. Debugging

The assembling machine code is shown in the Appendix 2 and COMMAND_TERMINAL-2_1_1.pdf[4]. As emulation, I connected the key matrix and LED display module block to PERSEUS-3 computer's parallel interface. I debugged the software on the PERSEUS-3 computer by step executions. After completed debugging, I programmed PROM by using my PROM programmer[5] and implemented to the computer block. Finally, the computer block was connected to the key matrix and LED display module block.

5. Build

The terminal uses a 18 cm x 12 cm x 5 cm aluminum die-cast enclosure. The key matrix & LED module block and the computer block are stacked in two layers. A power connector is provided so that it can be used with an external battery (1.2 V x 4) or 5 V DC power supply.

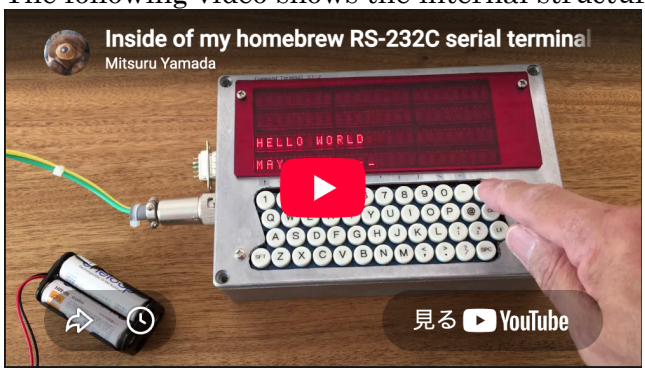
6. Results

The LED module does not work at all without initialization, so I had quite a hard time until my homemade serial communication program succeeded in initializing it. But this program code was later useful to me when I was recoding the calculator program on my project 6502 Computer runs calculator[6]. The key scan program could also be recoded and used in the 6502 calculator software.

In addition, I had a hard time concurrently processing the character input from the keys and the display process of the characters received by the serial interface. The current refresh interval of the display is 0.2 seconds. Therefore, any more number of character displays will be feel delayed in this architecture. So, I struggled some, but I am satisfied with the operation after completion.

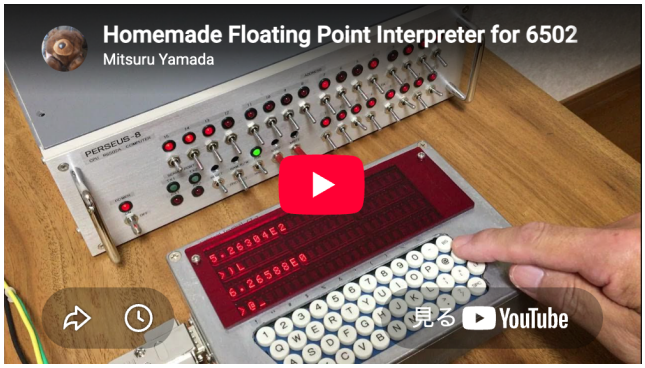
7. Video

The following video shows the internal structure.



https://www.youtube.com/watch?v=JVC7F1cIqUo

The following video shows the operation of a home-made floating-point interpreter on this terminal. There is no audio commentary, so please turn on subtitles.



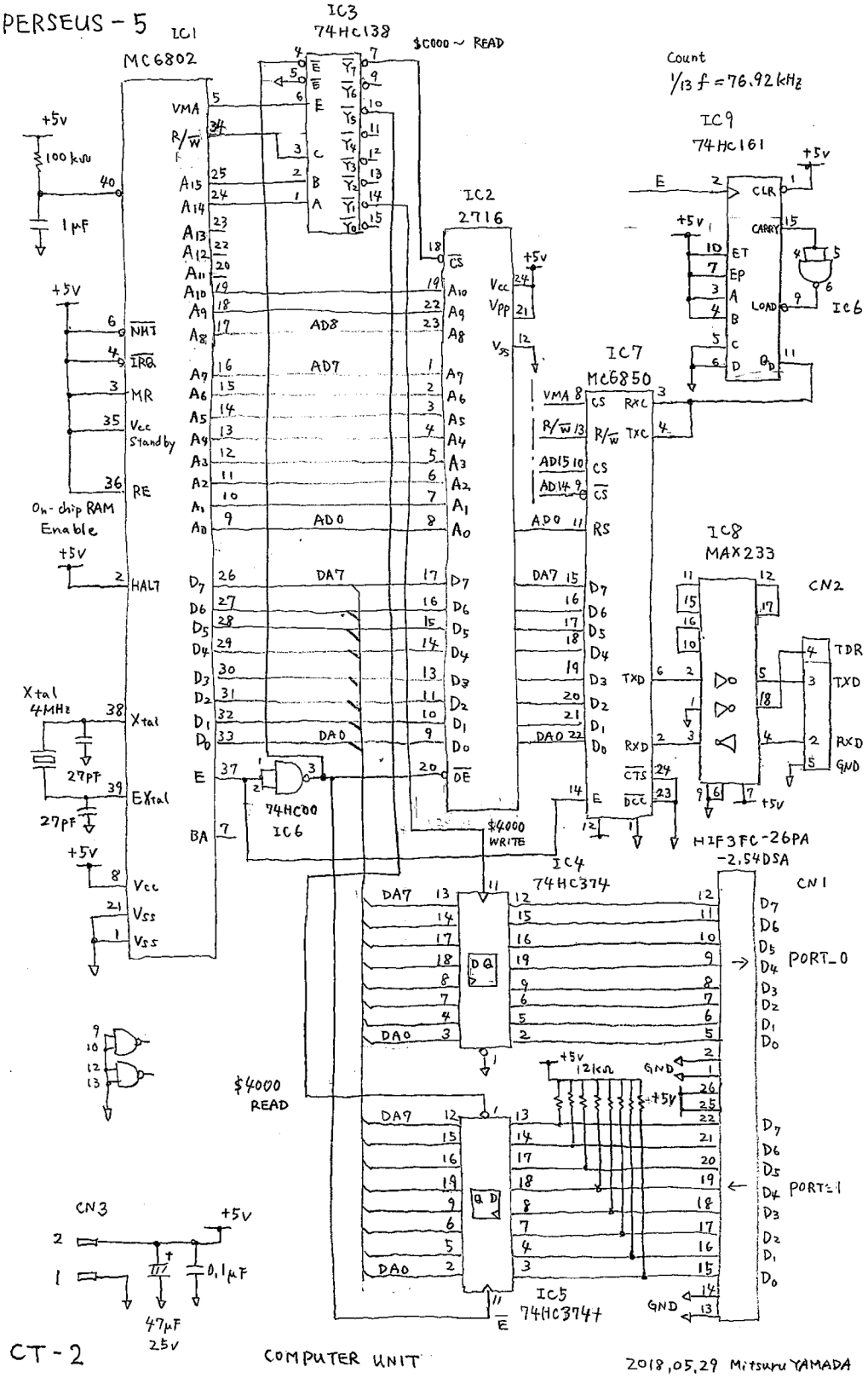
https://www.youtube.com/watch?v=ImJoRcZ9xC8&t=7s

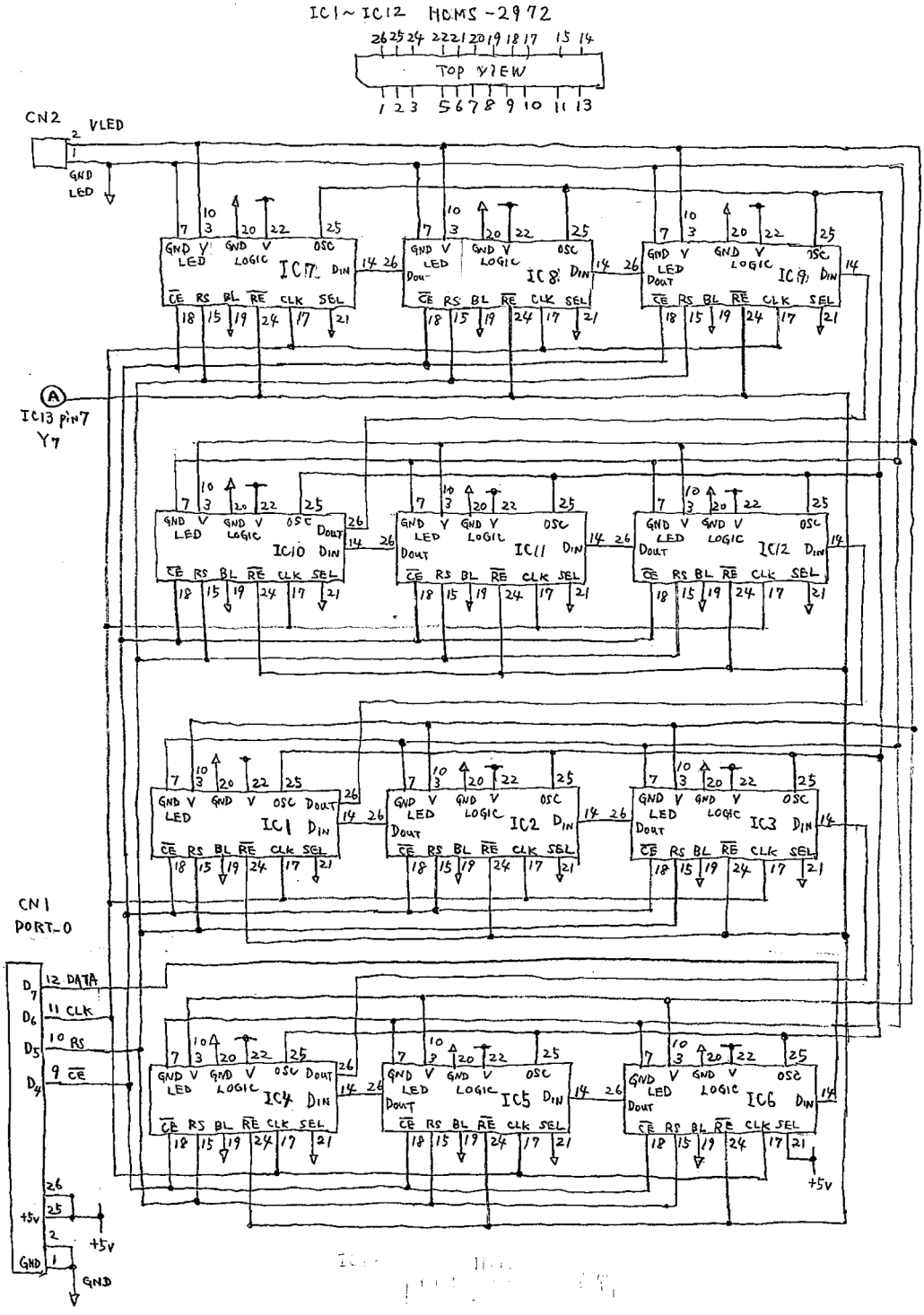
References

[1] PERSEUS-3 6802 homebrew computer, https://hackaday.io/project/176137-perseus-3-6802-homebrew-computer
[2] https://cdn.hackaday.io/files/1761757537471200/6802_SERIAL_TERMINAL_schematic_1.pdf
[3] HCMS-29xx Series High Performance CMOS 5x7 Alphanumeric Displays Data Sheet, AVAGO TECHNOLOGIES
[4] https://cdn.hackaday.io/files/1761757537471200/COMMAND_TERMINAL-2_1_1.pdf
[5] Fully manual PROM programmer, https://hackaday.io/project/175924-fully-manual-prom-programmer
[6] 6502 computer runs calculator, https://hackaday.io/project/175893-6502-computer-runs-calculator

(Posted on Dec. 05, 2020)
(Latest revision on Aug. 19, 2026)

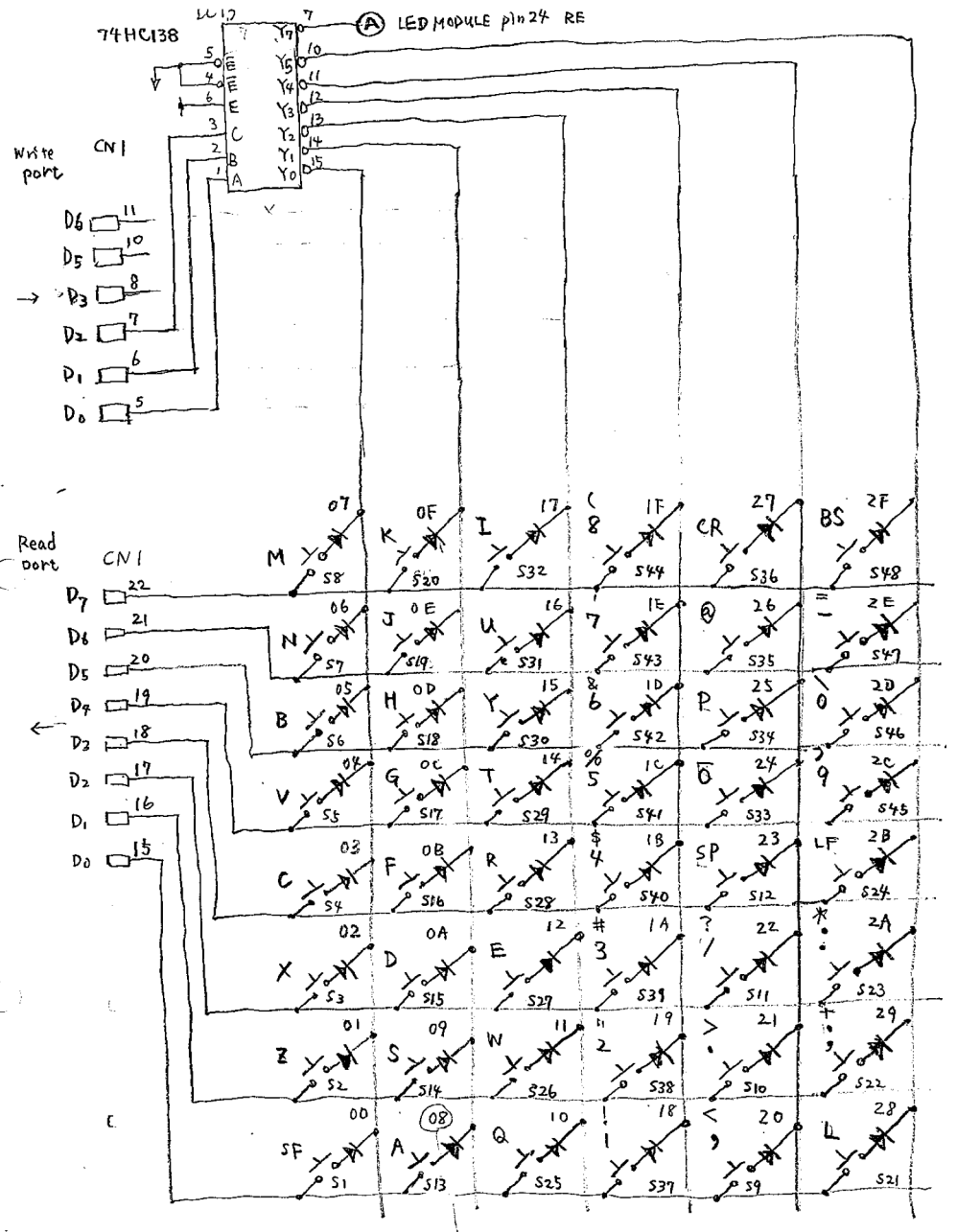
Appendix 1 Schematic of the 6802 Serial Terminal.





CT-2 KEYBOAD DISPLAY UNIT 1/2

2018. 08. 14



Key scan code table ; $FB~

Key scanning detection code

(↑ $38~ ↑ $40~ ↑ $48~
; Detection code; shift key pushed)

CT-2 KEYBOARD DISPLAY UNIT 2/2

2018.04.17

Appendix 2 Assembly machine code of the 6802 Serial Terminal.

```
%-------------------------------------------------------------------------------
% COMMAND_TERMINAL-2
% VER 2.1.1
%
% SERIAL TERMINAL PROGRAM FOR PERSEUS-5 (CPU:MC6802)
%
% HAND ASSEMBLED    MITSURU YAMADA
%
% COMPLETED                        08/JUN/2018
% CLEAN COPY                VER 2.0   05/DEC/2020
% TYPOGRAPHICAL CORRECTION  VER 2.1.1 18/MAR/2025
%
%-------------------------------------------------------------------------------
% COPYRIGHT (C) 2020 MITSURU YAMADA. ALL RIGHTS RESERVED.
%
%-------------------------------------------------------------------------------
% ADDRESS MAPPING:  24 X 4 CHARACTERS DISPLAY BUFFER (ADDRESS  $0000 - $005F)
%
% MAPPING TABLE, LOWER 8BIT ADDRESS(HEX)
%
% LEFT                                                                  RIGHT
%
% 00 01 02 03 04 05 06 07 : 08 09 0A 0B 0C 0D 0E 0F : 10 11 12 13 14 15 16 17   TOP
%
% 18 19 1A 1B 1C 1D 1E 1F : 20 21 22 23 24 25 26 27 : 28 29 2A 2B 2C 2D 2E 2F
%
% 30 31 32 33 34 35 36 37 : 38 39 3A 3B 3C 3D 3E 3F : 40 41 42 43 44 45 46 47
%
% 48 49 4A 4B 4C 4D 4E 4F : 50 51 52 53 54 55 56 57 : 58 59 5A 5B 5C 5D 5E 5F   BOTTOM
%
% -------------------------------------------------------------------------------
% VARIABLES
             SYMBOL                    DATA
             REG_0                     $60          LED FONT DATA EVACUATION
             REG_1                     $61          LED CONTROL PATTERN EVACUATION
             REG_2                     $62
             REG_3                     $63
             BUFFER_POINTER            $64          DISPLAY BUFFER POINTER (16 BIT)
             BUFFER_POINTER2           $66          DISPLAY BUFFER POINTER 2 (16 BIT)
             FONT_POINTER_H            $68          FONT TABLE POINITER UPPER 8 BIT
             FONT_POINTER_L            $69          FONT TABLE POINITER LOWER 8 BIT
             SCAN_PATTERN              $6A          SCAN PATTERN EVACUATION
             SHIFT_FLAG                $6B          SHIFT KEY PRESSED FLAG
             PUSH_FLAG_1               $6C          KEY PRESSED FLAG
             PUSH_FLAG_2               $6D          KEY PRESSED LAST TIME FLAG
             TABL_POINTER_H            $6E          ASCII TABLE POINITER UPPER 8 BIT
             TABL_POINTER_L            $6F          ASCII TABLE POINITER LOWER 8 BIT
             SCAN_PATTERN_2            $70
%
% ------------------------------------------------------------------------------
             BUFFER_START_1            $0000        DISPLAY BUFFER HEAD ADDRESS
             BUFFER_END_1              $0018        DISPLAY BUFFER LEFT END
             BUFFER_END_2              $0060        DISPLAY BUFFER END
             PORT                      $4000        PARALLEL INTERFACE ADDRESS (R/W)
             ACIA_STATUS               $A000        ACIA STATUS REGISTER
             ACIA_DATA                 $A001        ACIA DATA REGISTER
% ------------------------------------------------------------------------------
%
%                          ADDRESS(HEX)  DATA(HEX)
%-------------------------------------------------------------------------------
% PROGRAM START ADDRESS           MAIN_1       $FC00
%
% RESET VECTOR
             .ORIGIN  $FFFE
                                  FFFE   FC 00
%-------------------------------------------------------------------------------
% TERMINAL BUFFER 1 CHARACTER PROCESS
% INPUT PARAMETER          ACCA:               ASCII CODE
                           BUFFER_POINTER:     CURRENT POSITION OF BUFFER POINTER
%
             .ORIGIN  $F920
%
```

```
OUT_1_CHA     LDX  BUFFER_POINTER F920   DE 64        RETURN BUFFER POINTER
              CMPA #$61           F922   81 61        ASCII CODE > $60
              BCS  L11            F924   25 02
              SUBA #$20           F926   80 20        CONVERT LOWERCASE TO UPPERCASE
L11           CMPA #$0A           F928   81 0A        LF CODE?
              BEQ  SCROLL         F92A   27 1D        GO TO SCROLL PROCESS
              CMPA #$0D           F92C   81 0D        CR CODE?
              BEQ  LEFT_CURSOR_1  F92E   27 46        MOVE CURSOR TO LEFT END
              CMPA #$08           F930   81 08        BS CODE?
              BNE  L32            F932   26 0D
BACK_SPACE    LDAA #$20           F934   86 20        BACK SPACE PROCESS
              STAA X,$48          F936   A7 48        CLEAR CURRENT POINTER POSITION
              DEX                 F938   09          POINTER -1
              CPX  #$FFFF         F939   8C FF FF     KEEP POINTER NO EARLIER THAN TOP
              BNE  L31            F93C   26 06
              INX                 F93E   08
              BRA  L31            F93F   20 03
L32           STAA X,$48          F941   A7 48        WRITE 1 CHARACTER TO BUFFER
              INX                 F943   08          POINTER +1
L31           CPX  BUFFER_END_1   F944   8C 00 18     BUFFER RIGHT END?
              BNE  L01            F947   26 26
SCROLL        CPX  BUFFER_START_1 F949   8C 00 00     BUFFER LEFT END?
              BEQ  SCROLL_1       F94C   27 05
              LDAA #$20           F94E   86 20        IF POINTER IS NOT LEFT END,
              STAA X,$48          F950   A7 48              CLEAR POINITER POSITION
              NOP                 F952   01
SCROLL_1      LDX  BUFFER_START_1 F953   CE 00 00
L02           LDAA X,$18          F956   A6 18        COPY 1 CHARACTER TO UPPER LINE
              STAA X,$00          F958   A7 00
              LDAA X,$30          F95A   A6 30
              STAA X,$18          F95C   A7 18
              LDAA X,$48          F960   A7 30
              LDAA #$20           F962   86 20        CLEAR 1 CHARACTER LOWER LINE
              STAA X,$48          F964   A7 48
              INX                 F966   08          POINTER +1
              CPX  BUFFER_END_1   F967   8C 00 18     COMPLETED UNTIL RIGHT END?
              BNE  L02            F96A   26 EA
LEFT_CURSOR   LDX  BUFFER_START_1 F96C   CE 00 00     SET POINTER LEFT END
L01           LDAA #$5F           F96F   86 5F
              STAA X,$48          F971   A7 48        DISPLAY UNDERSCORE
              STX  BUFFER_POINITER F973  DF 64        EVACUATE POINTER
              RTS                 F975   39
%
LEFT_CURSOR_1 LDAA #$20           F976   86 20        MOVE CURSOR TO LEFT END
              STAA X,$48          F978   A7 48        CLEAR UNDERSCORE
              LDX  BUFFER_START_1 F97A   CE 00 00     SET POINTER TO LEFT END
              STX  BUFFER_POINTER F97D   DF 64        EVACUATE POINTER
              RTS                 F97F   39
%
%-------------------------------------------------------------------------------
% KEY SCANNING
              .ORIGIN  $FA00
%
KEY_SCAN      LDAA #$00           FA00   86 00        INIT. SCAN PATTERN NUMBER
              STAA SCAN_PATTERN_2 FA02   97 70        EVACUATE SCAN PATTERN NUMBER
              LDAA #$1B           FA04   86 FB        ASCII TABLE HEAD ADDRESS UPPER
              STAA TABL_POINTER_H FA06   97 6E
              LDAB #$00           FA08   C6 00        INIT. SCAN PATTERN ($00)
              STAB SCAN_PATTERN   FA0A   D7 6A        EVACUATE SCAN PATTERN
              CLRA                FA0C   4F
              STAA SHIFT_FLAG     FA0D   97 6B        CLEAR SHIFT KEY FLAG
L20           STAA PUSH_FLAG_1    FA0F   97 6C        CLEAR KEY PUSHED FLAG
L21           JSR  L28            FA11   BD FA 78     OUTPUT SCAN PATTERN
              LDAA PORT           FA14   B6 40 00     INPUT SCAN RESULT
              COMA                FA17   43          INVERT SCAN RESULT
              BEQ  L26            FA18   27 3A        $00? (NOT PRESSED)
              LDAB #$00           FA1A   C6 00        CONVERT SCAN RESULT TO 3BIT DATA
L22           ASRA                FA1C   47                SHIFT RIGHT 1BIT
              BCS  L23            FA1D   25 03
              INCB                FA1F   5C                 LOOP COUNTER +1
              BRA  L22            FA20   20 FA
%
```

```
L23          LDAA SCAN_PATTERN_2 FA22   96 70
             NOP                 FA24   01
             ABA                 FA25   1B          ADD 3BIT DATA AND KEY SCAN NUMBER
             LDAB #$01           FA26   C6 01
             STAB PUSH_FLAG_1    FA28   D7 6C       SET PRESSED FLAG TO '1'
             LDAB PUSH_FLAG_2    FA2A   D6 6D       LAST PRESSED FLAG = '1' ?
             BNE  L26            FA2C   26 26       IGNORE HOLDING DOWN
             CMPA #$00           FA2E   81 00       SHIFT KEY PRESSED?
             BEQ  L25            FA30   27 32
             STAA TABL_POINTER_L FA32   97 6F       IF SHIFT KEY IS NOT PRESSED,
             LDX  TABL_POINTER_H FA34   DE 6E             STORE TABLE INDEX TO IX
             LDAA X,$00          FA36   A6 00       GET ASCII CODE BY USING TABLE
%
SEND_1_CHA   LDAB ACIA_STATUS    FA38   F6 A0 00    SEND 1 CHARACTER THROUGH ACIA
             LSRB                FA3B   54
             LSRB                FA3C   54
             BCC  SEND_1_CHA     FA3D   24 F9
             STAA ACIA_DATA      FA3F   B7 A0 01
L27          LDAB PUSH_FLAG_1    FA42   D6 6C       UPDATE KEY PRESSED LAST TIME FLAG
             STAB PUSH_FLAG_2    FA44   D7 6D
             RTS                 FA46   39          END OF KEY SCAN SUBROUTNE
%
             .ORIGIN  $FA54
%
L26          LDAA SCAN_PATTERN_2 FA54   96 70
             ADDA #$08           FA56   8B 08       UPDATE SCAN PATTERN NUMBER(INCREMENT D3)
             STAA SCAN_PATTERN_2 FA58   97 70
             LDAB SCAN_PATTERN   FA5A   D6 6A
             INCB                FA5C   5C          SCAN PATTERN +1
             CMPB #$06           FA5D   C1 06       FINISH SCAN PATTERN $05 ?
             BNE L21             FA5F   26 B0       IF NOT FINISH RETURN TO OUT SCAN PATTERN
             BRA L27             FA61   20 DF       IF FINISH GO TO UPDATE PRESSED FLAG
%
             .ORIGIN  $FA64
%
L25          LDAA #$01           FA64   86 01       SHIFT KEY PRESSED PROCESS
             STAA SHIFT_FLAG     FA66   97 6B       SET SHIFT FLAG TO '1'
             LDAB #$03           FA68   C6 03       SET SCAN PATTREN TO $03 (SHIFT VALID RANGE)
             STAB SCAN_PATTERN_2 FA6A   D7 6A
             LDAA #$38           FA6C   86 38       SET SCAN PATTERN NUMBER TO $38
             STAA SCAN_PATTERN_2 FA6E   97 70
             CLRA                FA70   4F
             JMP  L20            FA71   7E FA 0F    RETURN TO DETECTING PRESSED
%
             .ORIGIN  $FA78
%
L28          STAB SCAN_PATTERN   FA78   D7 6A       EVACUATE SCAN PATTERN
             ANDB #$0F           FA7A   C4 0F       D4-D7 FORCED ZERO
             STAB PORT           FA7C   F7 40 00
             LDAB SCAN_PATTERN   FA7F   D6 6A       OUTPUT SCAN PATTERN
             RTS                 FA81   39
%
%-------------------------------------------------------------------------------
% SERIAL TERMNAL MAIN PROGRAM
             .ORIGIN  $FC00
%
MAIN         LDS  $007F          FC00   8E 00 7F    INIT. STACK POINTER
             JSR  INIT_DISP_2    FC03   BD FC 16    INIT. LED MODULE
             JSR  INIT_ACIA      FC06   BD FC 68    INIT. SERIAL INTERFACE
             NOP                 FC09   01
             NOP                 FC0A   01
             NOP                 FC0B   01
L10          JSR  DISP_BUFFER    FC0C   BD FC 80    DISPLAY UPDATE ENTIRE BUFFER
             BRA  L10            FC0F   20 FB       REPEAT
%
%-------------------------------------------------------------------------------
% INITIALIZE LED MODULE
             .ORIGIN  $FC16
%
INIT_DISP_2  LDAA #$07           FC16   86 07
             STAA PORT           FC18   B7 40 00    RESET LED MODULE *RE=L
             LDAA #$00           FC1B   86 00
```

```
             STAA PORT           FC1D   B7 40 00     REREASE RESET LED MODULE *RE=H
INIT_DISP    LDAA #$30           FC20   86 30
             STAA PORT           FC22   B7 40 00     (CLK=L,RS=H,*CE=H)
             LDAA #$20           FC25   86 20
             STAA PORT           FC27   B7 40 00     (CLK=L,RS=H,*CE=L)
             STAA REG_1          FC2A   B7 00 61     EVACUATE CONTROL PATTERN
             LDAB #$18           FC2D   C6 18        LOOP COUNTER = 24(DEC)
L06          STAB REG_3          FC2F   F7 00 63     EVACUATE LOOP COUNTER
             LDAA #$4A           FC32   86 4A        SET CONTROL WORD 0
             JSR  OUT_FNT_SERL   FC34   BD FC F0     SEND SERIAL DATA
             LDAB REG_3          FC37   F6 00 63     RETURN LOOP COUNTER
             DECB                FC3A   5A           LOOP COUNTER -1
             BNE   L06           FC3B   26 F2        REPEAT UNTIL LOOP COUNTER=0
             LDAA #$70           FC3D   86 70
             STAA PORT           FC3F   B7 40 00     (CLK=H,RS=H,*CE=L)
             LDAA #$30           FC42   86 30
             STAA PORT           FC44   B7 40 00     (CLK=L,RS=H,*CE=L)
             STAA REG_1          FC47   B7 00 61     EVACUATE CONTROL PATTERN
             NOP                 FC4A   01
%
CLEAR_BUFFER LDX  #$0000         FC4B   CE 00 00     INIT. BUFFER ADDRESS
             LDAA #$20           FC4E   86 20        SPACE CODE
L40          STAA X,$00          FC50   A7 00
             INX                 FC52   08           ADDRESS POINITER +1
             CPX  #$0060         FC53   8C 00 60     BUFFER END?
             BNE  L40            FC56   26 F8
             LDAA #$5F           FC58   86 5F        UNDERSCORE CODE
             STAA $48            FC5A   97 48        WRITE UNDERSCORE LOWER LEFT
             LDX  #$0000         FC5C   CE 00 00     INIT. BUFFER ADDRESS
             STX  BUFFER_POINTER FC5F   FF 00 64     EVACUATE BUFFER POINTER
             RTS                 FC62   39
%
%-------------------------------------------------------------------------------
% INITIALIZE ACIA
             .ORIGIN  $FC68
%
INIT_ACIA    LDAA #$03           FC68   86 03
             STAA ACIA_STATUS    FC6A   B7 A0 00
             LDAA #$15           FC6D   86 15        4800 BIT/S 8BIT NO PARITY
             STAA ACIA_STATUS    FC6F   B7 A0 00
             RTS                 FC72   39
%
%-------------------------------------------------------------------------------
% DISPLAY UPDATE OF ENTIRE BUFFER
             .ORIGIN  $FC80
%
DISP_BUFFER  LDAA REG_1          FC80   96 61
             ANDA #$D0           FC82   84 D0
             STAA PORT           FC84   B7 40 00     (DATA=,CLK=,RS=L,*CE=)
             ANDA #$C0           FC87   84 C0
             STAA PORT           FC89   B7 40 00     (DATA=,CLK=,RS=L,*CE=L)
             STAA REG_1          FC8C   97 61        EVACUATE CONTROL PATTERN
             LDX  BUFFER_START_1 FC8E   CE 00 00     BUFFER HEAD ADDRESS
L03          LDAA X,$00          FC91   A6 00        GET 1 CHARACTER FROM BUFFER
             STX  BUFFER_POINTER2 FC93  DF 66        EVACUATE BUFFER POINTER
             JSR  OUT_FONT       FC95   BD FC C0     SEND 1 CHARACTER FONT TO LED
             JSR  KEY_SCAN       FC98   BD FA 00     KEY SCAN
             LDX  BUFFER_POINTER2 FC9B  DE 66        RETURN BUFFER POINTER
             INX                 FC9D   08           POINTER +1
             CPX  BUFFER_END_2   FC9E   8C 00 60     BUFFER END?
             BNE  L03            FCA1   26 EE
             LDAA REG_1          FCA3   96 61
             ORAA #$10           FCA5   8A 10
             STAA PORT           FCA7   B7 40 00     (DATA=,CLK=,RS=,*CE=H)
             ANDA #$B0           FCAA   84 B0
             STAA PORT           FCAC   B7 40 00     (DATA=,CLK=L,RS=,*CE=)
             STAA REG_1          FCAF   97 61
             RTS                 FCB1   39
%
%-------------------------------------------------------------------------------
% OUTPUT 1 CHARACTER FONT (5 BYTE)
% INPUT PARAMETER   ACCA:  ASCII CODE
```

```
%
% FONT POINTER:     (ASCII CODE - $20) AND SHIFT LEFT 3BIT = (D4 TO D8) OF 16BIT
%
             .ORIGIN  $FCC0
%
OUT_FONT     SUBA #$20           FCC0   80 20        ASCII CODE -$20
             ANDA #$3F           FCC2   84 3F        CLEAR UPPER 2 BIT
             LDAB FONT_DATA_H    FCC4   C6 FE        FONT DATA HEAD ADDRESS
             ASLA                FCC6   48           SHIFT LEFT 3 BIT
             ASLA                FCC7   48
             ASLA                FCC8   48
             ADCB #$00           FCC9   C9 00        IF CARRY=1, UPPER 8BIT + 1
             STAB FONT_POINTER_H FCCB   D7 68
             STAA FONT_POINTER_L FCCD   97 69        SET CODE TO FONT DATA POINITER 9 BIT
L51          LDX  FONT_POINTER   FCCF   DE 68
             LDAB #$05           FCD1   C6 05        ACCB: LOOP COUNTER FOR 5 BYTE FONT
L04          LDAA X,$00          FCD3   A6 00        EXTRACTION FONT DATA   (2025/03/18)
             NOP                 FCD5   01
             BRA  L52            FCD6   20 09        PATCH TO PREVENT IX BREAK ON SERIAL INPUT
             NOP                 FCD8   01
             NOP                 FCD9   01
             NOP                 FCDA   01
L53          LDAB REG_2          FCDB   D6 62        RETURN LOOP COUNTER
             DECB                FCDD   5A           LOOP COUNTER -1
             BNE  L04            FCDE   26 F3
             RTS                 FCE0   39
L52          STAB REG_2          FCE1   D7 62        EVACUATE LOOP COUNTER
             INX                 FCE3   08           FONT POINTER +1
             STX  FONT_POINITER  FCE4   DF 68        EVACUATE FONT POINTER
             JSR  OUT_FNT_SERL   FCE6   BD FC F0     OUTPUT FONT BY SERIAL
             LDX  FONT_POINTER   FCE9   DE 68        RETURN FONT POINTER
             JMP  L53            FCEB   7E FC DB
%
%-------------------------------------------------------------------------------
% OUTPUT FONT 1 BYTE TO LED DISPLAY MODULE BY SERIAL
% INPUT PARAMETER   ACCA:  FONT DATA AND CONTROL REGISTER VALUE
                    REG_1: CURRENT CONTROL LINE LEVEL
%
             .ORIGIN  $FCF0
%
OUT_FNT_SERL LDAB #$08           FCF0   C6 08        SET LOOP COUNTER (8BIT)
             STAA REG_0          FCF2   97 60        EVACUATE FONT DATA
             LDAA REG_1          FCF4   96 61        RETURN CONTROL DATA
             ANDA #$3F           FCF6   84 3F        (DATA=L,CLK=L,RS=,*CE=)
             STAA REG_1          FCF8   97 61
L05          LDAA REG_0          FCFA   96 60        RETURN FONT DATA
             ANDA #$80           FCAC   84 80        EXTRACTION D7
             ORAA REG_1          FCFE   9A 61        SYNTHESIZE CONTROL DATA
             STAA PORT           FD00   B7 40 00     OUTPUT TO LED MODULE
             ORAA #$40           FD03   8A 40        (DATA=L,CLK=H,RS=,*CE=)
             STAA PORT           FD05   B7 40 00     OUTPUT TO LED MODULE
             ASL  REG_0          FD08   78 00 60     SHIFT LEFT FONT DATA
             DECB                FD0B   5A           LOOP COUNTER -1
             BNE  L05            FD0C   26 EC
             LDAA ACIA_STATUS    FD0E   B6 A0 00     SERIAL INPUT RECEIVED?
             LSRA                FD11   44
             BCC  L14            FD12   24 06        IF NOT RECEIVED, EXIT
             LDAA ACIA_DATA      FD14   B6 A0 01
             JSR  OUT_1_CHA      FD17   BD F9 20     IF RECEIVED, 1 CHARACTER PROCESS
L14          RTS                 FD1A   39

%
%-------------------------------------------------------------------------------
% KEY SCAN ASCII CODE TABLE
             .ORIGIN  $FB00
%                         ADDRESS(HEX)  DATA(HEX)

                          FB00   FF 5A 58 43 56 42 4E 4D
                          FB08   41 53 44 46 47 48 4A 4B
                          FB10   51 57 45 52 54 59 55 49
                          FB18   31 32 33 34 35 36 37 38
                          FB20   2C 2E 2F 20 4F 50 40 0D
```

```
                                FB28   4C 3B 3A 0A 39 30 2D 08
                                FB30   FF FF FF FF FF FF FF FF
                                FB38   21 22 23 24 25 26 27 28
                                FB40   3C 3E 3F FF FF FF 5E FF    (Shift-@ '^' 2025/03/18)
                                FB48   FF 2B 2A FF 29 5C 3D FF
%-------------------------------------------------------------------------------
% FONT DATA TABLE
% 5 X 7 DOT MATRIX CHARACTER FONT      (CORRECTED ERRONEUOUS DESCRIPTION 2025/03/18)
             .ORIGIN  $FE00
%                         ADDRESS(HEX) DATA(HEX)           CHARACTER

                                FE00   00 00 00 00 00      ' '
                                FE08   00 5F 00 00 00      '!'
                                FE10   00 03 00 03 00      '"'
                                FE18   14 7F 14 7F 14      '#'
                                FE20   24 2A 7F 2A 12      '$'
                                FE28   23 13 08 64 62      '%'
                                FE30   36 49 56 20 50      '&'
                                FE38   00 0B 07 00 00      '''
                                FE40   00 00 3E 41 00      '('
                                FE48   00 41 3E 00 00      ')'
                                FE50   08 2A 1C 2A 08      '*'
                                FE58   08 08 3E 08 08      '+'
                                FE60   00 58 38 00 00      ','
                                FE68   08 08 08 08 08      '-'         (2025/03/18)
                                FE70   00 30 30 00 00      '.'
                                FE78   20 10 08 04 02      '/'
                                FE80   3E 51 49 45 3E      '0'
                                FE88   00 42 7F 40 00      '1'
                                FE90   62 51 49 49 46      '2'
                                FE98   22 41 49 49 36      '3'
                                FEA0   18 14 12 7F 10      '4'
                                FEA8   27 45 45 45 39      '5'
                                FEB0   3C 4A 49 49 30      '6'
                                FEB8   01 71 09 05 03      '7'
                                FEC0   36 49 49 49 36      '8'
                                FEC8   06 49 49 29 1E      '9'
                                FED0   00 36 36 00 00      ':'
                                FED8   00 5B 3B 00 00      ';'
                                FEE0   00 08 14 22 41      '<'
                                FEE8   14 14 14 14 14      '='
                                FEF0   41 22 14 08 00      '>'
                                FEF8   02 01 51 09 06      '?'
                                FF00   3E 41 5D 55 1E      '@'
                                FF08   7E 09 09 09 7E      'A'
                                FF10   7F 49 49 49 36      'B'
                                FF18   3E 41 41 41 22      'C'
                                FF20   7F 41 41 41 3E      'D'
                                FF28   7F 49 49 49 41      'E'
                                FF30   7F 09 09 09 01      'F'
                                FF38   3E 41 41 51 72      'G'
                                FF40   7F 08 08 08 7F      'H'
                                FF48   00 41 7F 41 00      'I'
                                FF50   20 40 40 40 3F      'J'
                                FF58   7F 08 14 22 41      'K'
                                FF60   7F 40 40 40 40      'L'
                                FF68   7F 02 0C 02 7F      'M'
                                FF70   7F 04 08 10 7F      'N'
                                FF78   3E 41 41 41 3E      'O'
                                FF80   7F 09 09 09 06      'P'
                                FF88   3E 41 51 21 5E      'Q'
                                FF90   7F 09 19 29 46      'R'
                                FF98   26 49 49 49 32      'S'
                                FFA0   01 01 7F 01 01      'T'
                                FFA8   3F 40 40 40 3F      'U'
                                FFB0   07 18 60 18 07      'V'
                                FFB8   7F 20 18 20 7F      'W'
                                FFC0   63 14 08 14 63      'X'
                                FFC8   03 04 78 04 03      'Y'
                                FFD0   61 51 49 45 43      'Z'
                                FFD8   00 00 7F 41 41      '['
                                FFE0   02 04 08 10 20      '¥'
```

```
                                FFE8   41 41 7F 00 00      ‘]‘
                                FFF0   04 02 01 02 04      ‘^’        (2025/03/18)
                                FFF8   40 40 40 40 40      ‘_‘

%
%-------------------------------------------------------------------------------
% END OF PROGRAM
```