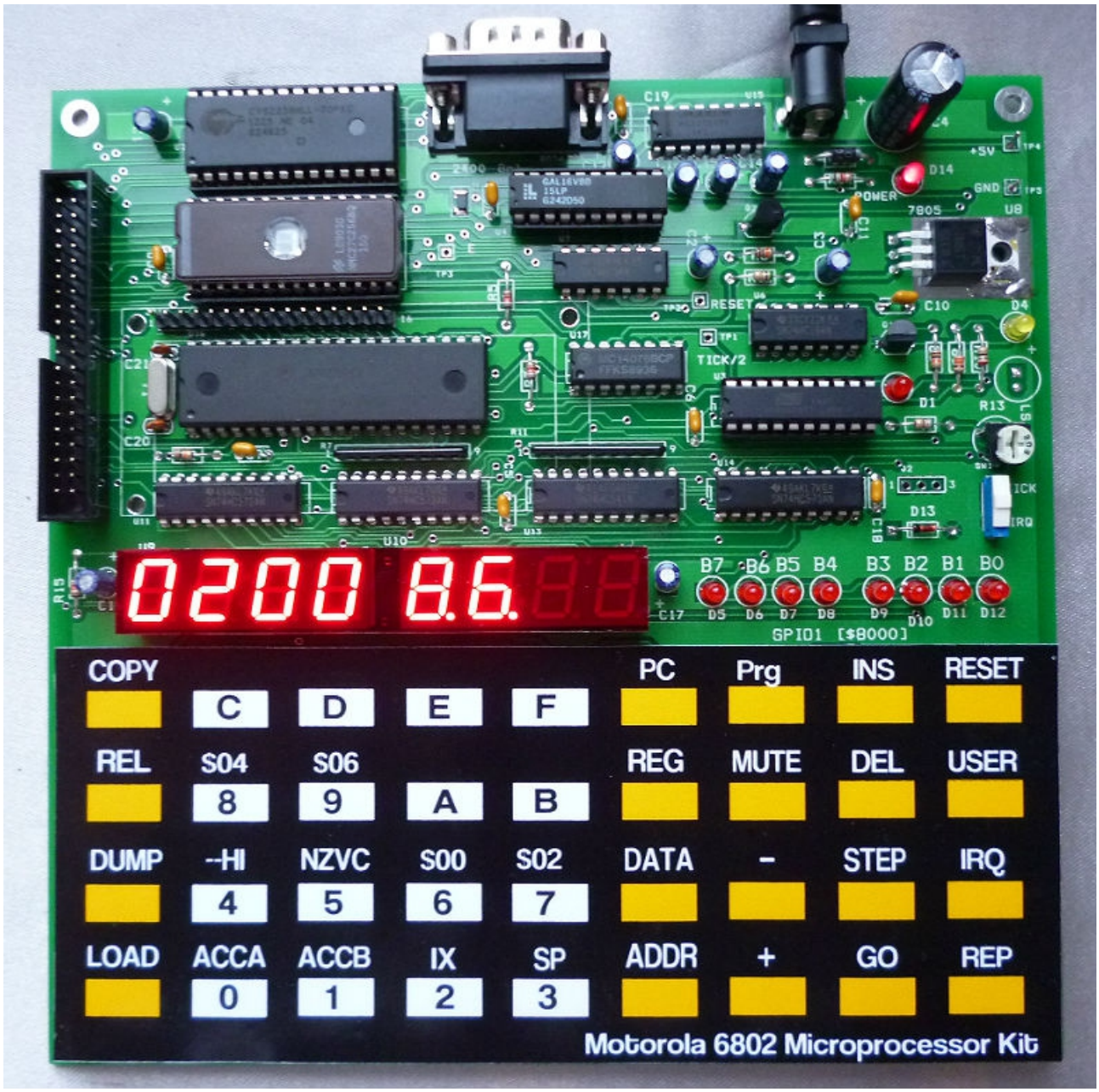
6802 Microprocessor Kit
User's Manual



Rev 1.1, October 2015

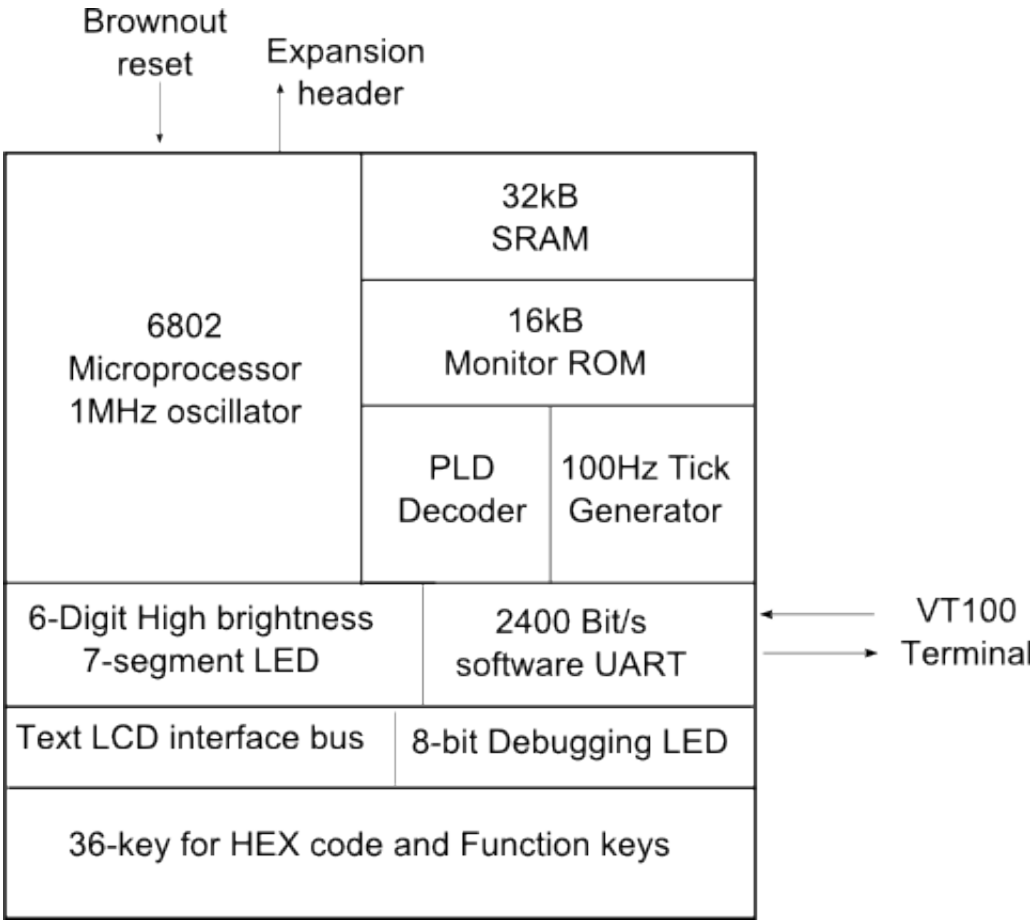
6802 MICROPROCESSOR KIT

CONTENTS

OVERVIEW

The 6802 Microprocessor kit is a new design single board computer using the early 1976 Motorola 6802 microprocessor. This single board computer is a basic learning tool for programming the 6802 with low level instructions hex code. The board has hex keypad and 7-segment display for entering the instruction hex code and test it directly. Students will learn basic of the computer hardware and software of the 6802 easily. The 6802 is software compatible with the 6800 microprocessor. The instructions and addressing modes of 6802 microprocessor are good for learning the basic of microprocessor operations.

FUNCTIONAL BLOCK DIAGRAM



Notes

1. UART is software control for low speed asynchronous communication.

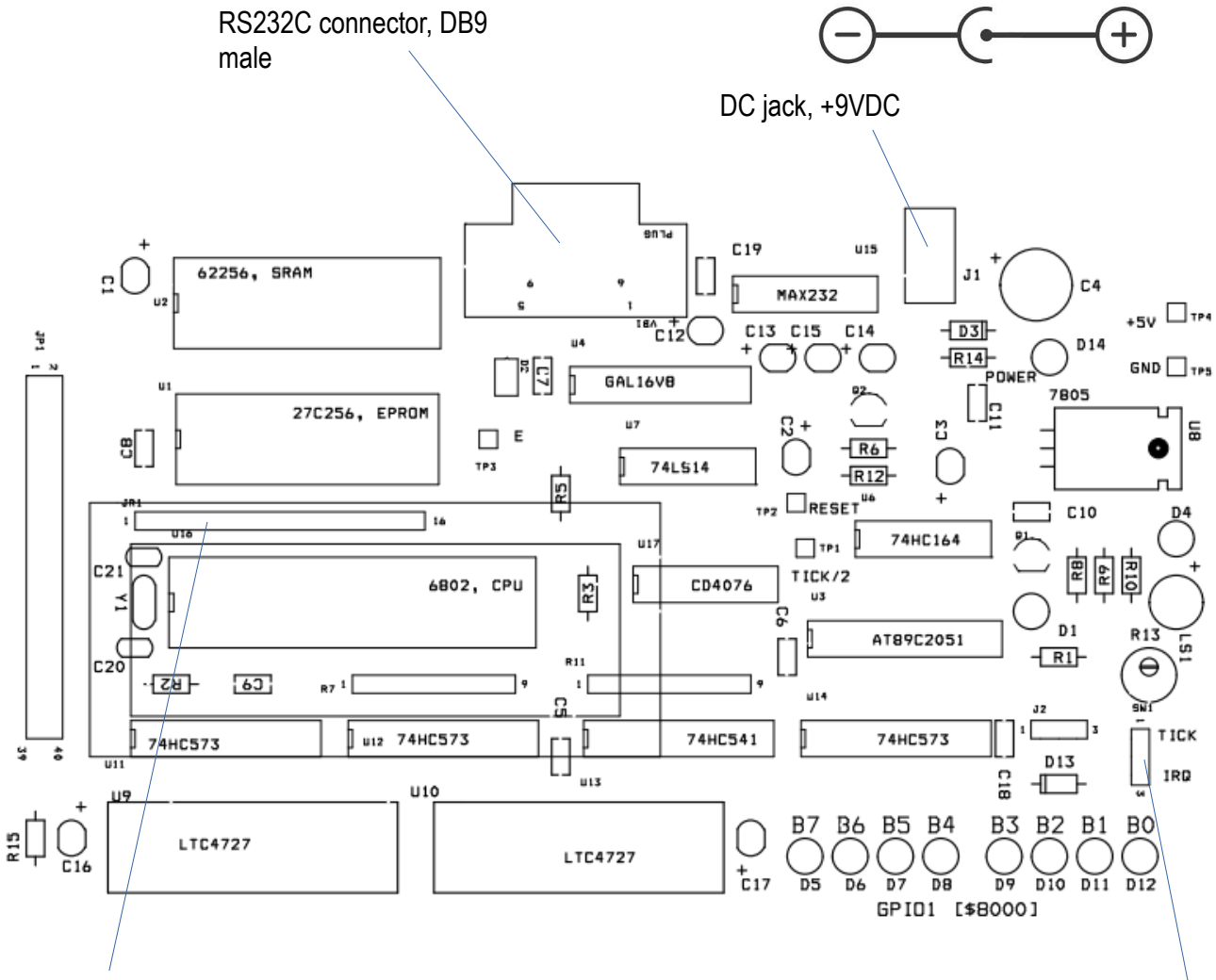
2. The kit has 8-bit LCD module interfacing bus.

3. 100Hz Tick generator is for interrupt experiment.

4. Ports for display and keypad interfacing were built with discrete logic IC chips.

5. Memory and Port decoders are made with Programmable Logic Device, PLD. 4

HARDWARE LAYOUT



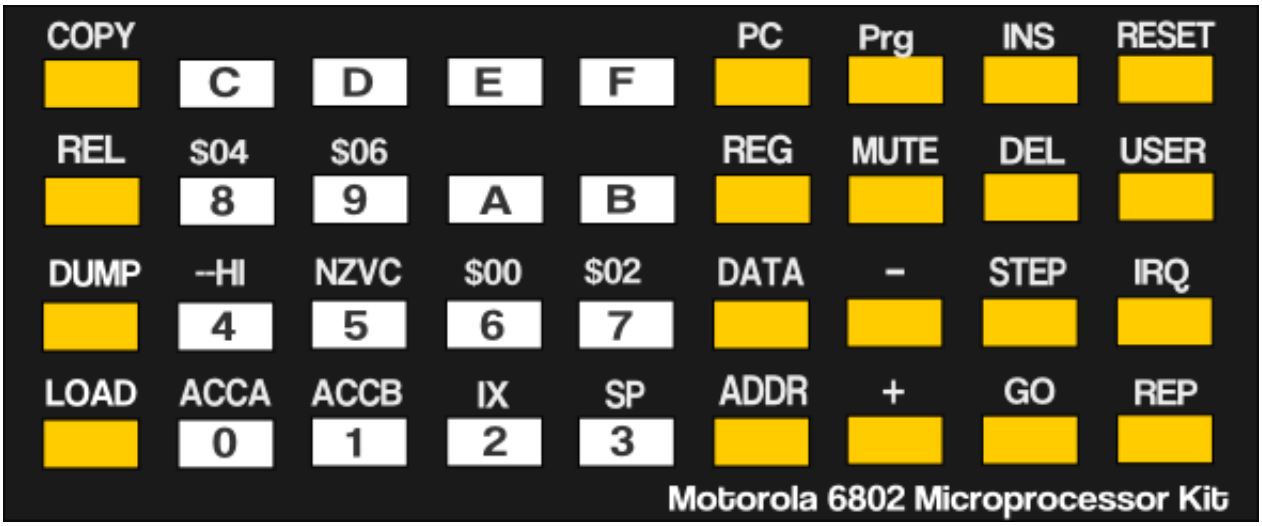
Important Notes

1. Plugging or removing the LCD module must be done when the kit is powered off!

2. AC adapter should provide approx. +9VDC, higher voltage will cause the voltage regulator chip becomes hot.

3. The kit has diode protection for wrong polarity of adapter jack. If the center pin is not the positive (+), the diode will be reverse bias, preventing wrong polarity feeding to voltage regulator.

KEYBOARD LAYOUT



HEX keys Hexadecimal number 0 to F with associated user registers, flag bits and page zero memory $00 to $06 (use with key REG)

CPU control keys

RESET Reset the CPU, the 6802 will get reset vector from location FFFE and FFFF

USER User key, no connection

IRQ Make IRQ pin to logic low, used for experimenting with interrupt process

Monitor function keys

REP Repeat the key that pressed, must be pressed together with REP key.

INS Insert one byte to the next location, the 256 bytes will be shifted down.

DEL Delete one byte at current display, the next 256 bytes will be moved up.

STEP Execute user code only single instruction and return to save CPU registers

GO Jump from monitor program to user code

- Decrement current display address by one

+ Increment current display address by one

PC Set current display address with user Program Counter

REG Display user registers, flags or page zero $00 to $06 with HEX key.

ACCA, Accumulator A

ACCB, Accumulator B

IX, 16-bit Index register

SP, 16-bit Stack Pointer register

--HI, upper 4-bit contents of the Condition Code registers, for Half Carry Flag and Interrupt flag

NZVC, lower 4-bit contents of the Condition Code register, for Negative, Zero, Overflow and Carry flags.

$00, display 16-bit data location $00 (high byte) and $01 (low byte)

$02, display 16-bit data location $02 (high byte) and $03 (low byte)

$04, display 16-bit data location $04 (high byte) and $05 (low byte)

$06, display 16-bit data location $06 (high byte) and $07 (low byte)

DATA Set entry mode of hex keys to Data field

ADDR Set entry mode of hex keys to Address field

COPY Copy block of memory used with key + for Start, End, Destination and with key GO

REL Compute relative byte, used with key + for Start, Destination and key GO

DUMP Display 16 bytes x 16 lines memory contents by using 2400 bit/s display terminal.

LOAD Load Motorola S-record or Intel hex file at 2400 bit/s using serial port

MUTE Turn beep ON/OFF

Prg Demonstration program 00-09, program number can be selected with key + or -. Then press key GO to run it. Available sample programs are,

00 Rotate bit with ROLA instruction

01 Binary number counting from 00 to FF

02 BCD number counting from 00 to 99

03 Simple clock using 10ms tick for IRQ interrupt. SW1 must select to Tick. Key + will clear the clock.

04 Test LCD module. Display message on the LCD display. Caution! Plugging/removing the LCD module must do when the kit was POWER OFF!

05-09 N/A for this version. User may add it in the monitor source code.

HARDWARE FEATURES

The hardware features:

- **CPU: Motorola 6802, NMOS 8-bit Microprocessor @1MHz clock**
- **Memory: 32kB RAM, 16kB EPROM**
- **Memory and I/O Decoder chip: Programmable Logic Device GAL16V8D**
- **Display: high brightness 6-digit 7-segment LED**
- **Keyboard: 36 keys**
- **RS232 port: software controlled UART 2400 bit/s 8n1**
- **Debugging LED: 8-bit GPIO1 LED at location $8000**
- **Tick: 10ms tick produced by 89C2051 for time trigger experiment**
- **Text LCD interface: direct CPU bus interface text LCD**
- **Brownout reset: KIA7042 reset chip for power brownout reset**
- **Expansion header: 40-pin header**

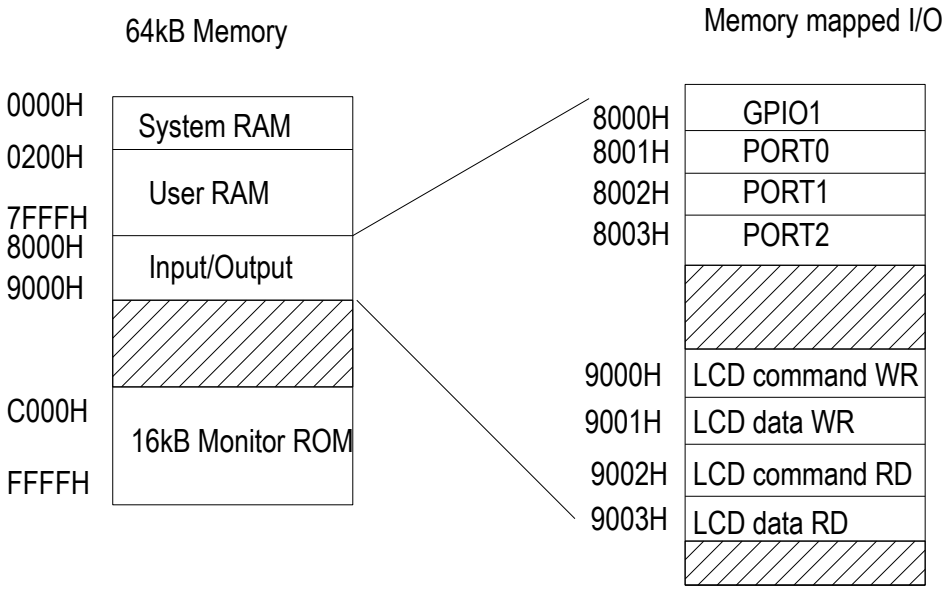
MONITOR PROGRAM FEATURES

The MONITOR program features:

- **Simple hex code entering**
- **Insert and Delete byte**
- **Single step running**
- **User registers: ACCA, ACCB, IX, SP, and Condition code registers for storing CPU status after program execution**
- **Page zero memory display**
- **Easy offset byte calculation for Relative addressing mode**
- **Copy block of memory**
- **Motorola s-record or Intel hex file downloading**
- **Memory dump**
- **Beep ON/OFF**
- **Demo programs**

MEMORY AND I/O MAPS

The first 32kB is RAM space from 0000-7FFFH. Zero page is location 00-FFH. User Stack space is set at 7F00H. System Stack is set at 7FFFH. User space is from 200H to 7000H. The 6802 CPU uses memory space from 8000H-BFFFH for I/O ports. Available user I/O for lab experiment is A000H-BFFFH. The monitor ROM is located at C000H-FFFFH



INTERRUPT and RESET VECTORS

The vectors for RESET and Interrupts are shown below.

Vector		Description
MS	LS	
$FFF8	$FFF9	IRQ
$FFFA	$FFFB	Software Interrupt
$FFFC	$FFFD	NMI
$FFFE	$FFFF	RESET

Software interrupt vector is for SWI instruction. Hex code for SWI instruction is 3FH.

User can put 3FH (SWI instruction) at the end of user program. It will generate interrupt process, return to the monitor program that saves CPU registers to user registers. We can examine the program running easily.

NMI is reserved for Single Step running. When press STEP key, the NMI interrupt will be generated after user instruction was executed. The service routine is the same as software interrupt, saving CPU registers to user registers for later checking.

IRQ is prepared for user interrupt experiment. The RAM vector, 00F8H is put the IRQ vector in ROM. CPU will jump to location 00F8H. User can place the service routine for

IRQ at that location! **NOT THE VECTOR ITSELF!** See the program example of using IRQ in the 6802 Programming Book.

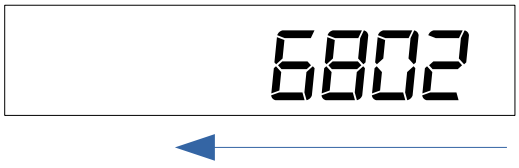
GETTING STARTED

The kit accepts DC power supply with minimum voltage +7.5V. It draws DC current approx. 200mA. However we can use +9VDC from any AC adapter. The example of AC adapter is shown below.

The center pin is positive.

If your adapter is adjustable output voltage, try with approx. +9V. Higher voltage will make higher power loss at the voltage regulator, 7805. Dropping voltage across 7805 is approx. +2V. To get +5VDC for the kit, we thus need DC input >+7.5V.

When power up, we will see the cold boot message 6802 running.



Press RESET it will be warm boot. When RESET key has been released it will show,

If there is no LED light up, check the adapter polarity with multimeter.

HOW TO ENTER PROGRAM USING HEX CODE

Let us try enter HEX CODE of the example program to memory and test it. We write the program using 6802 instructions.

Address	Hex code	Label	Instruction	comment
0200	86 01	MAIN	LDAA #1	Load register A with 1
0202	B7 80 00		STAA $8000	Write A to GPIO1@ 8000H

Our test program has only two instructions.

The first instruction is LDAA #1, Load Accumulator A with 8-bit constant, 1.

This instruction has two bytes hex code i.e., 86 is LDAA #n and #n is 01.

The 2nd instruction is STAA $8000. The instruction's machine code is B7. The location of GPIO1 is 8000, 16-bit data.

The total of hex codes for this small program is 5 bytes that are, 86, 01, B7, 80, and 00.

The first byte will be entered to location 0200. And the following bytes will be entered at 0201, 0202, 0203, 0204.

Let us see how to enter these codes into the memory.

Step 1 Press key PC, the display will show current memory address and its contents.

Shown the location 0200 has data A4. There are small dots at the data field indicating the active field, ready for modifying the hex contents.

Step 2 Press key 8 and key 6. The new hex code 86 will be entered to the location 0200.

Step 3 Press key + to increment the location from 0200 to 0201. Then enter hex key 1,

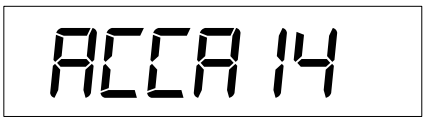
Repeat Step 3 until completed for the last location. We can verify the hex code with key + or key -.

To change the display location, press key ADDR. The dots will move to Address field. Any hex key pressed will change the address.

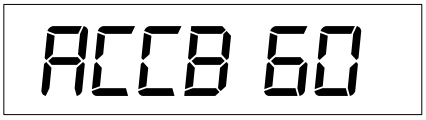
USER REGISTERS DISPLAY

Before we test the code running, let us see how to use the user registers. User registers are the memory block in RAM that used to save the contents of CPU registers after completing a given program running. We can examine the user registers for checking our code running then.

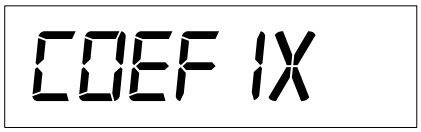
Press key REG, then press key 0, it will show **Accumulator A has 14**.



Press key 1 for Accumulator B.



Press key 2 for Index register, shown **IX has 16-bit data= C0EF**



Press key 3 for Stack Pointer register, shown **SP has 16-bit data= 7F00**

Press key 4 for upper half 4-bit of the Condition Code Register for Half carry flag and Interrupt flag.

Press key 5 for lower half 4-bit of the Condition Code Register for Negative, Zero, Overflow and Carry flags.

Press key 6, the 16-bit data at location 00 and 01 will show. Shown 16-bit data, 174C.

17 is MS byte stored at address 0000.

4C is LS byte stored at address 0001.

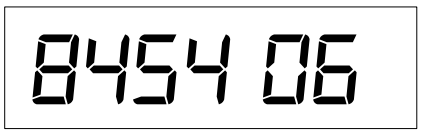
Similarly for key 7, 8 and 9.

16 is MS byte stored at address 0002

4A is LS byte stored at address 0003

0C is MS byte stored at address 0004

82 is LS byte stored at address 0005



84 is MS byte stored at address 0006

54 is LS byte stored at address 0007

These (user) registers will be useful for program testing. The page zero memory display, each 16-bit will also be useful for checking Arithmetic and logical operations. We will see the examples in the programming book.

TEST CODE RUNNING WITH SINGLE STEPPING

Now get back to the program we have just entered. Let us take a look again.

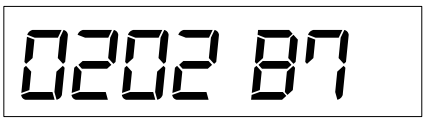
Address	Hex code	Label	Instruction	comment
0200	86 01	MAIN	LDAA #1	Load register A with 1
0202	B7 80 00		STAA $8000	Write A to GPIO1@ 8000H

We will try test the program using single step running.

Step 1 Press key PC. We see that the location 0200 has 86.

Step 2 Press REG key, then key 0, to check the Accumulator A.

Step 3 Press PC then STEP key. The instruction LDAA #1 will be executed.



The display will show next instruction to be executed, that is STAA $8000.

Step 4 Let check again register A with key REG and 0. We see that now the Accumulator A has 01.

Step 5 Press PC then STEP, the instruction STAA $800 will be executed. The display will show next instruction to be executed.

We will see the contents of Accumulator register will be stored at GPIO1 LED!

To get back key PC to the location 0200, press RESET key, then press PC.

We see that the single step running is useful for learning the operation of each instruction. We can check the result with user registers easily.

HOW TO FIND OFFSET BYTE

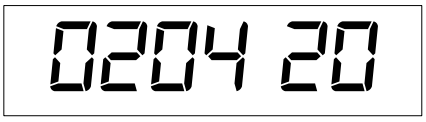
The offset byte is signed number using two's complement. It is the distant in number of byte length between destination address and current Program Counter. The 6802 CPU uses it for relative addressing of the Branch instructions. The monitor program has function key for finding the offset byte by entering the start and destination addresses.

Let us see the example, how to find the offset byte, XX from below program.

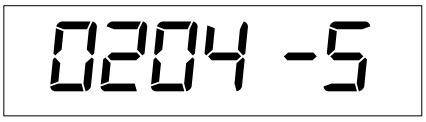
This program will increment the Accumulator A, store it to GPIO1 LED, then branch back to loop. The BRA LOOP instruction has machine code 20. The 2nd byte is OFFSET byte.

Address	Hex code	Label	Instruction	comment
0200	4C	LOOP	INCA	Increment Accumulator A
0201	B7 80 00		STAA $8000	Write A to GPIO1@ 8000H
0204	20 XX		BRA LOOP	Jump back to loop

Suppose we have entered the hex code from beginning to the location 0204. The display will show.



To find the offset byte, XX, Press key REL.



The display address will be start location. Press key +, and enter destination address, 200.

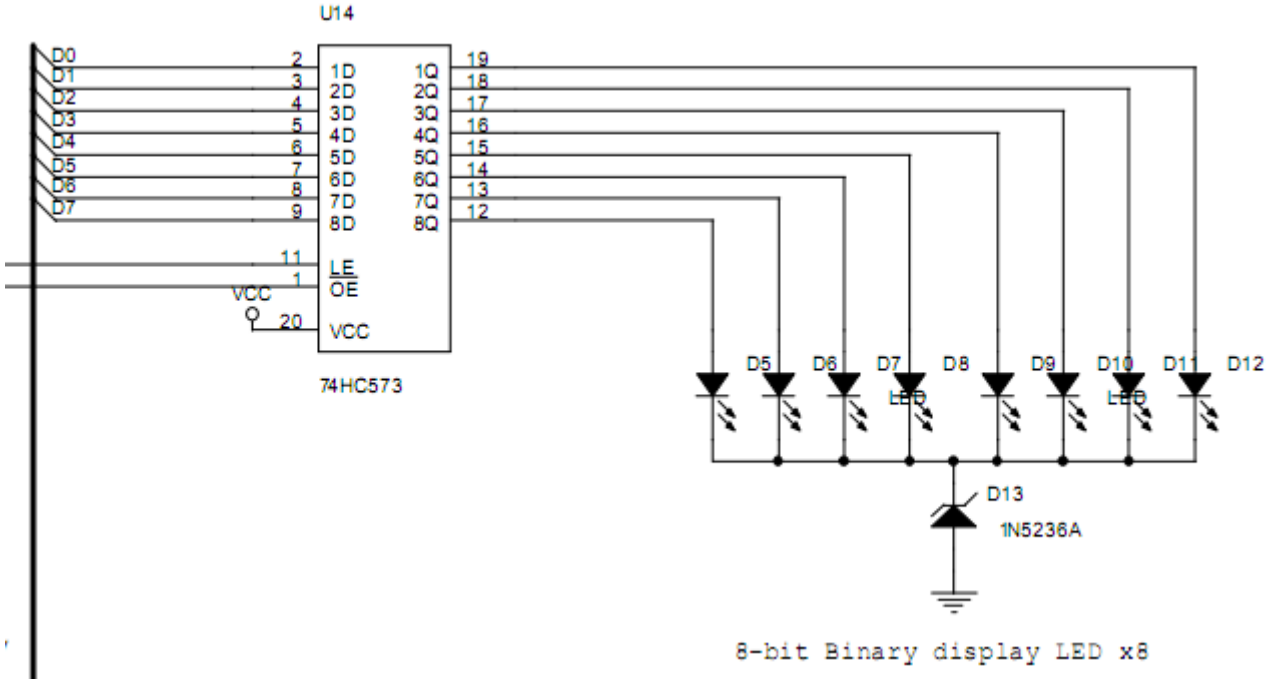
Press GO, the offset byte, FA or -6 will be entered to location 0205 automatically.

If we count by hand, we will see that the byte length will be 6 bytes backward.

We can test this program with key STEP and REP together. We will see binary incrementing on the GPIO1 LED. The program will jump back to loop with Branch Always instruction.

GPIO1 LED

The 6802 kit provides a useful 8-bit binary display. It can be used to debug the program or code running demonstration. The I/O address is 8000H. U14 is 8-bit data latch. Logic 1 at the output will make LED lit.



The GPIO1 LED can be used to display accumulator register easily. Let us take a look the sample code below.

Address	Hex code	Label	Instruction	comment
0200	C6 01	MAIN	LDAB #1	Load register B with 1
0202	F7 80 00		STAB $8000	Write B to GPIO1@ 8000H

The test code has only two instructions. The first instruction has two bytes machine code, C6 and 01. The second instruction has three bytes F7, 80 and 00.

Enter the hex code to memory from 0200 to 0204. Then press PC, and execute the instruction with single step by pressing key STEP. The 2nd press STEP key that executes instruction STAB $8000 will make the GPIO1 LED showing the contents of register B. Try change the load value to register B and test the code.

Another example is with JUMP instruction. The JUMP instruction will change the Program Counter to 0200, to repeat program running. Now we use instruction that increments the Accumulator A. After incrementing, we write register A to location of GPIO1 at 8000H. And with JMP LOOP instruction, the program will be repeated.

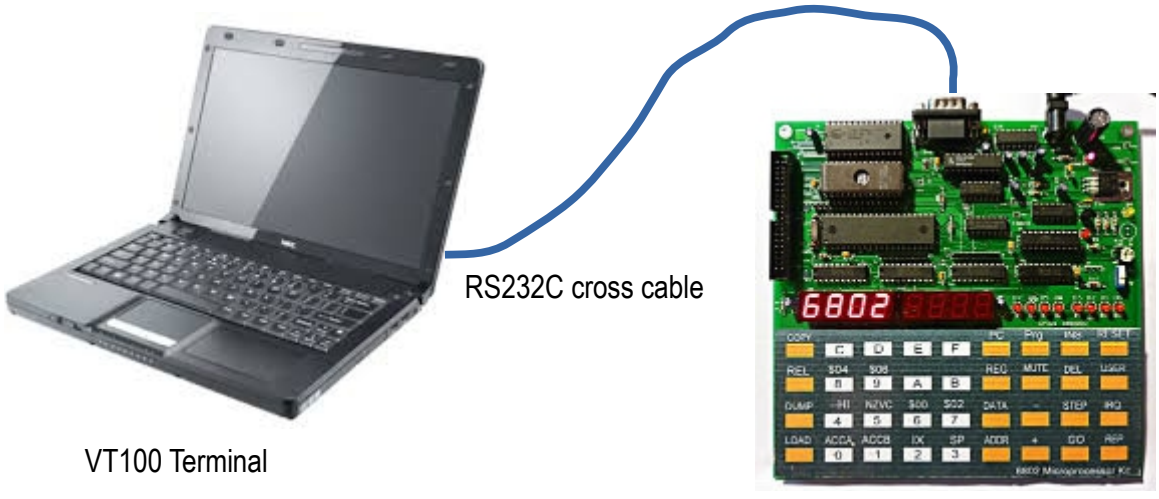
Address	Hex code	Label	Instruction	comment
0200	4C	LOOP	INCA	Increment Accumulator A
0201	B7 80 00		STAA $8000	Write A to GPIO1@ 8000H
0204	7E 02 00		JMP LOOP	Jump back to loop

Again enter the hex code to memory and test it with single step. Now press key STEP and key REP together. Every time when instruction STAA $8000 was executed, did you see the binary number counting up?

Note the JUMP instruction has 3 bytes hex code, but the BRA instruction has only 2 bytes.

CONNECTING 6802 KIT TO TERMINAL

For LOAD key, we can connect the 6802 kit to a terminal by RS232C cross cable. You may download free terminal program, teraterm from this URL,
http://ttssh2.sourceforge.jp/index.html.en



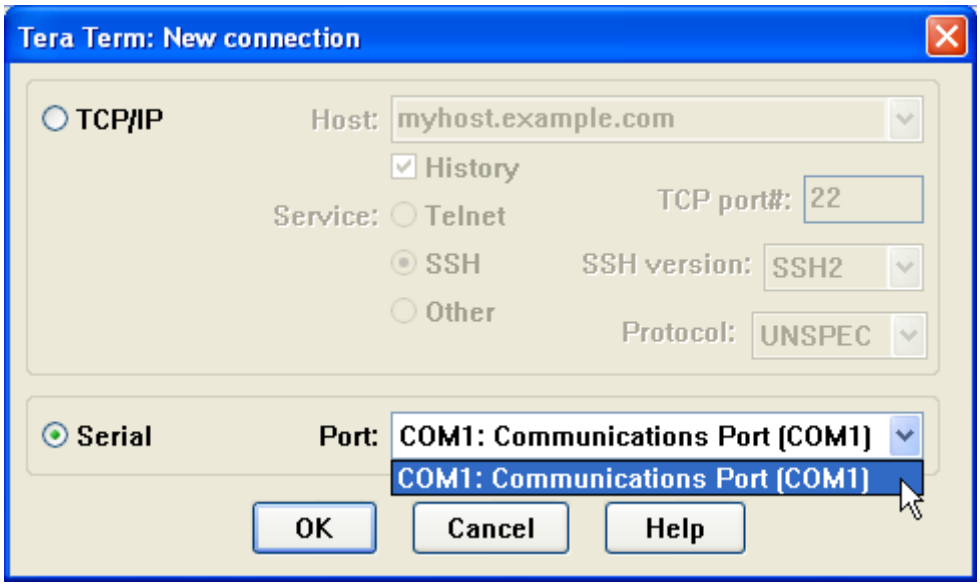
VT100 Terminal

6802 Kit

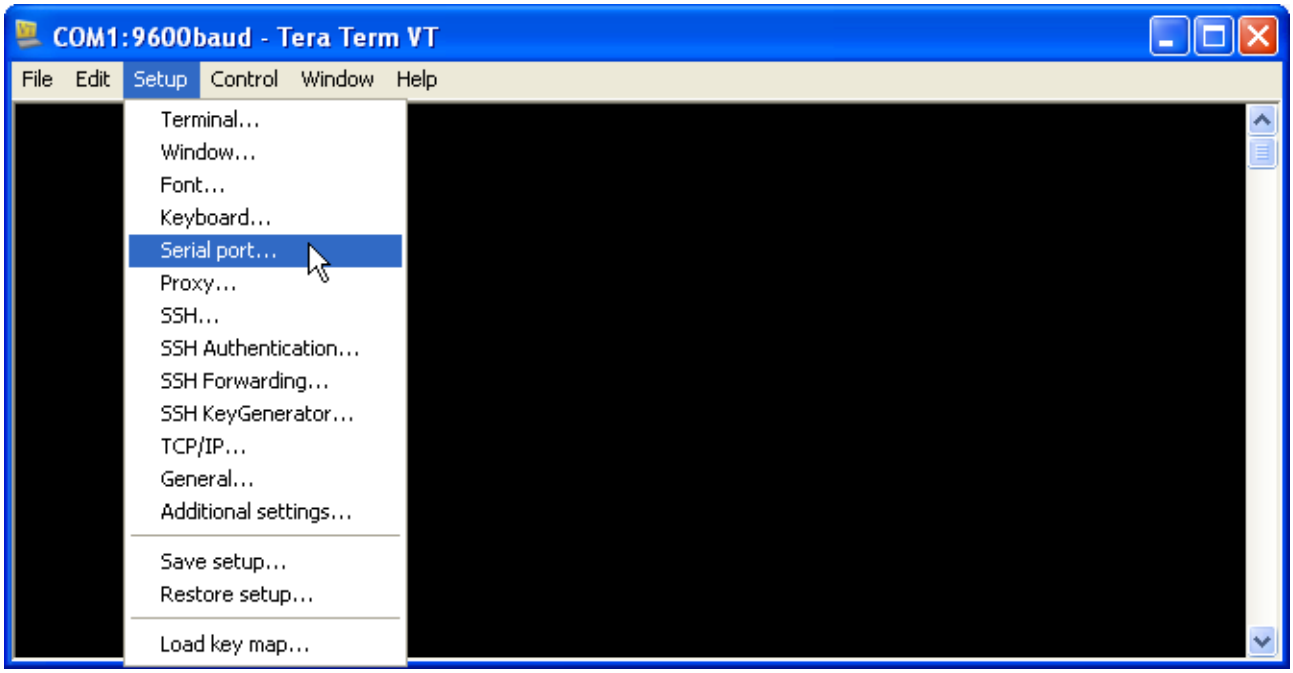
The example shows connecting laptop with COM1 port to the RS232C port of the 6802 kit. New laptop has no COM port, we may use the USB-RS232 adapter for converting the USB port to RS232 port.

To download Motorola s-record or Intel hex file that generated from the assembler or c compiler, set serial port speed to 2400 bit/s, 8-data bit, no parity, no flow control, one stop bit.

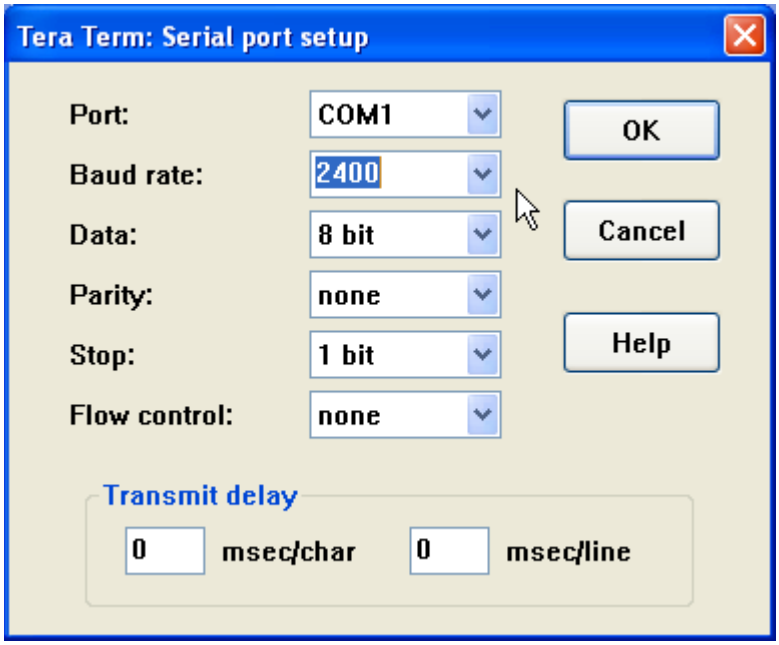
Step 1 Run teraterm, then click at Serial connection.



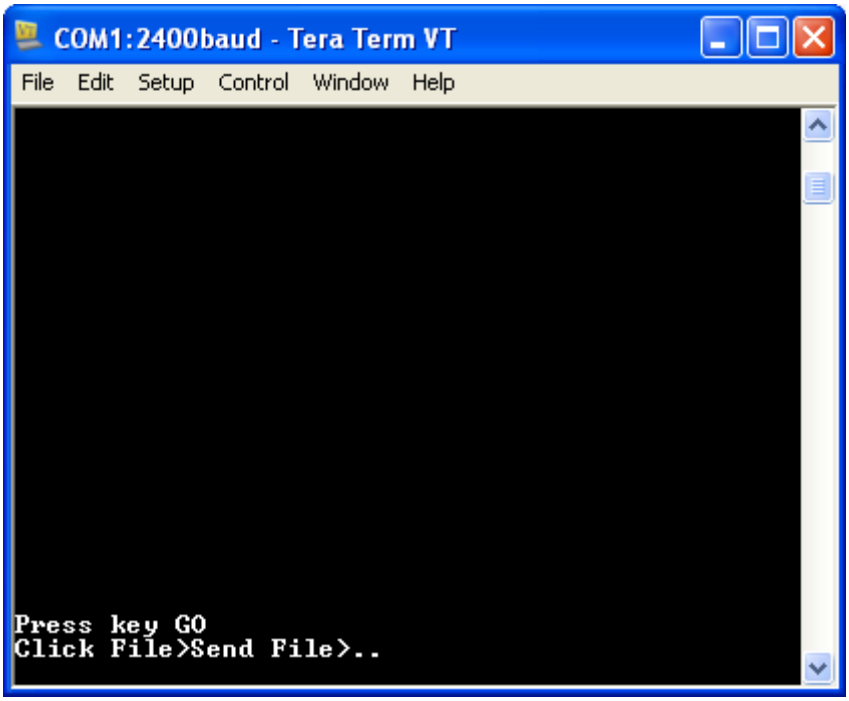
Step 2 Click setup>Serial port.



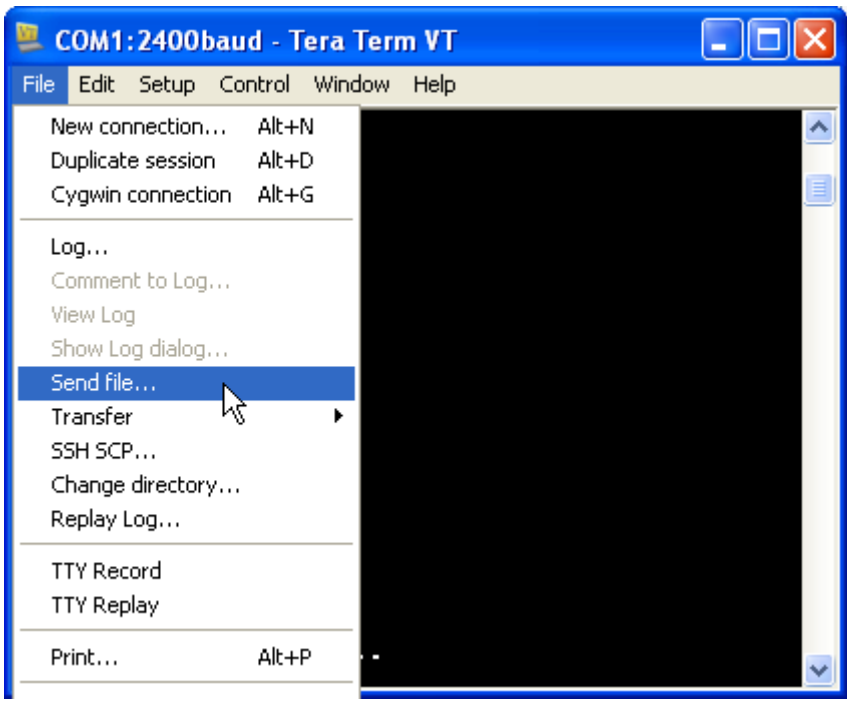
Step 3 Set serial port speed to 2400 and format as shown below.



Step 4 Press key LOAD, then key GO. The kit will wait for the data stream from terminal.



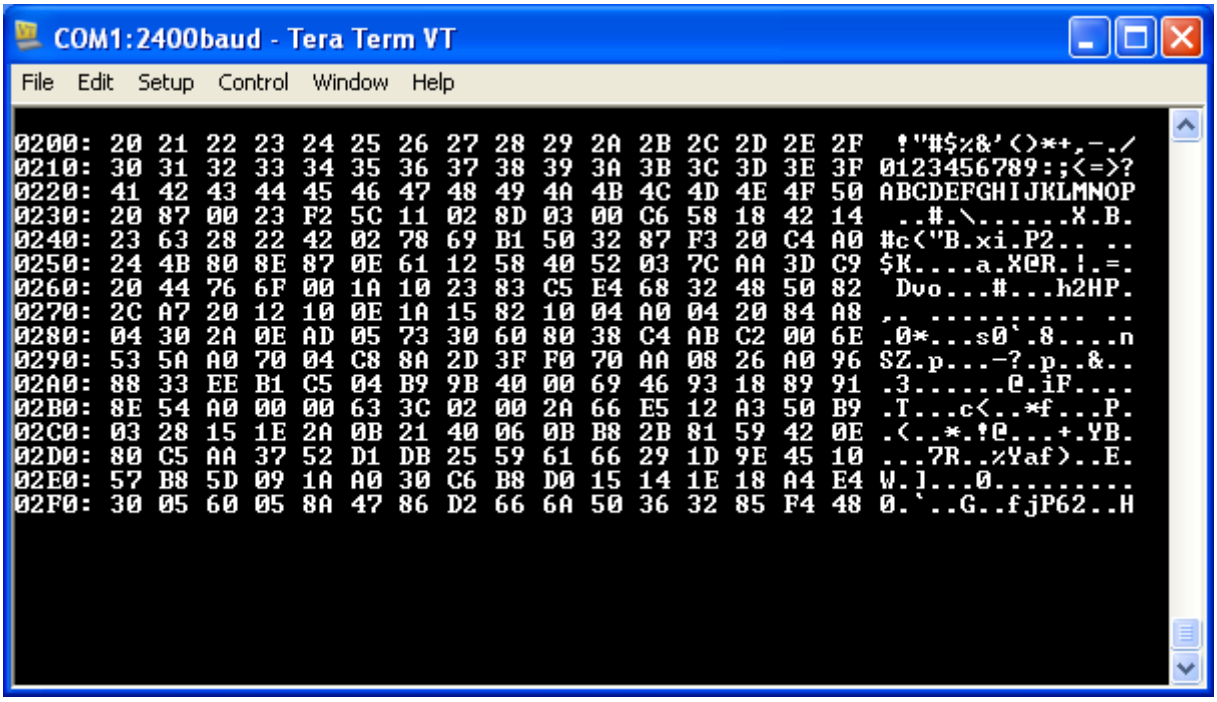
Step 5 On PC, Click file>Send File>LED.HEX.



The kit will read the hex file, write to memory, when completed if no checksum error, the display will show the current address. The kit accepts for both Motorola s-record or Intel hex files automatically.

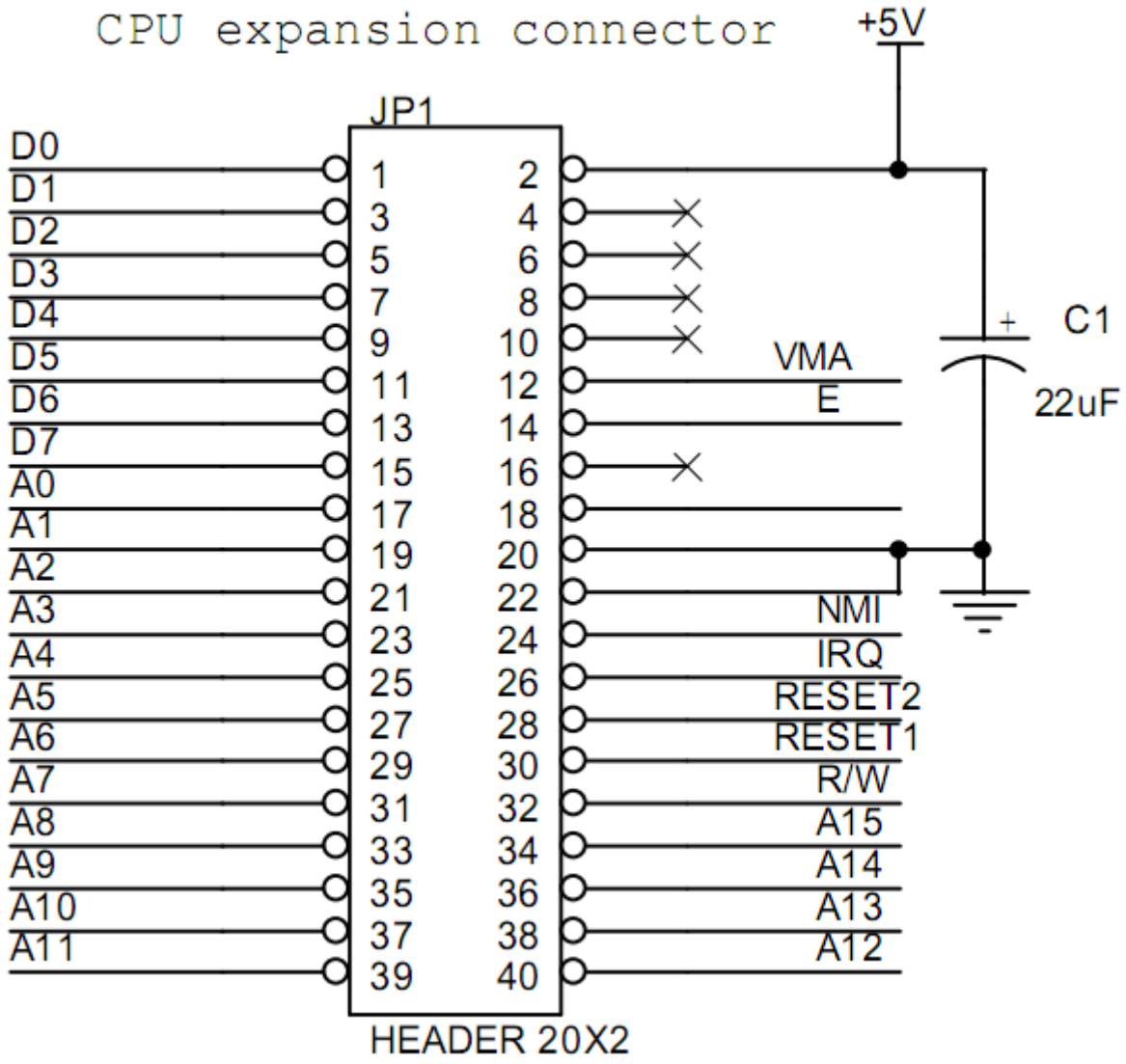
DUMP MEMORY CONTENTS

With terminal connection, we can use it to display the memory contents, 256 bytes easily. The monitor command is DUMP key. When pressed, the kit will send data stream to display the memory contents from current program counter. In addition, the ASCII code is also decoded for each line.



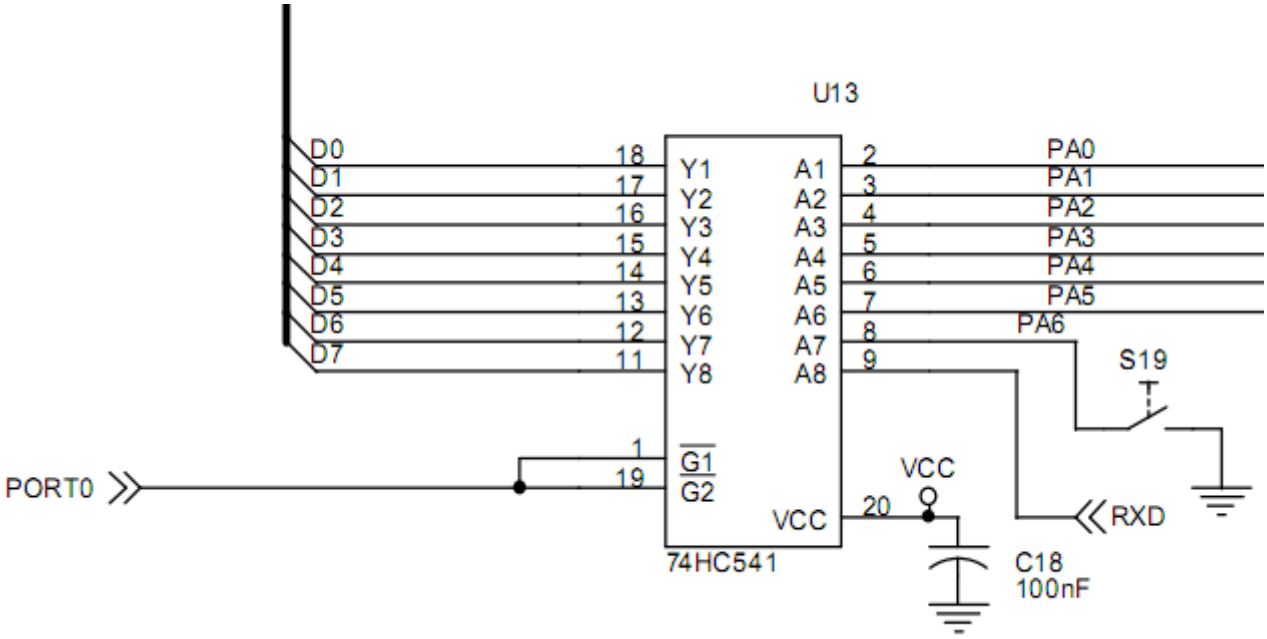
EXPANSION BUS HEADER

JP1, 40-pin header provides CPU bus signals for expansion or I/O interfacing. Students may learn how to make the simple I/O port, interfacing to Analog-to-Digital Converter, interfacing to stepper motor or AC power circuits.



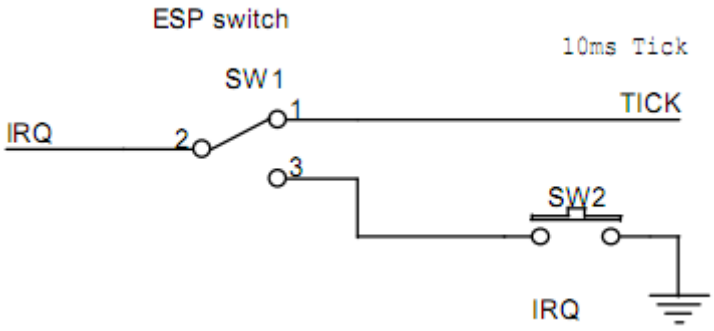
REP KEY

REP(repeat) key, S19 is one bit active low key switch connected to bit 6 of Port 0. To test the logic of S19, we can use instruction LDAA $0000 and check bit 6 of the accumulator with test bit instruction. REP key is used in monitor program together with key STEP, + or – to provide automatic repeating.

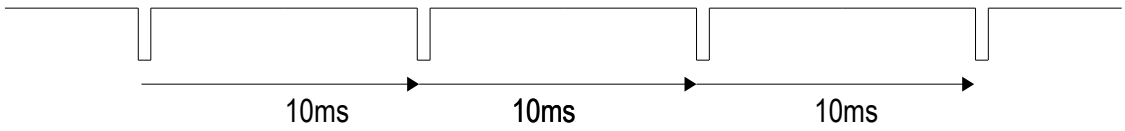


10ms TICK GENERATOR

SW1 is a selector for interrupt source between key IRQ or 10ms tick produced by 89C2051 microcontroller. Tick generator is software controlled using timer0 interrupt in the 89C2051 chip. The active low tick signal is sent to P3.7. For tick running indicator, P1.7 drives D1 LED.

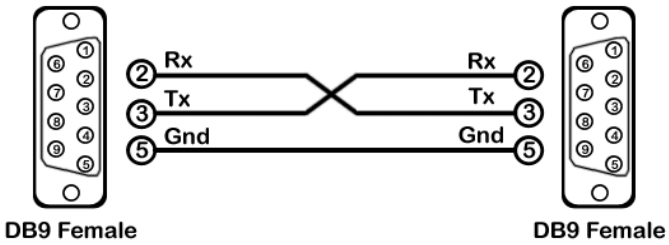


Tick is a 10ms periodic signal for triggering the 6802 IRQ pin. When select SW1 to Tick, the 6802 CPU can be triggered by a maskable interrupt. The 100Hz tick or 10ms tick can be used to produce tasks that executed with multiple of tick. The 6802 kit lab look will show how to use 10ms tick to make a digital timer.



RS232C PORT

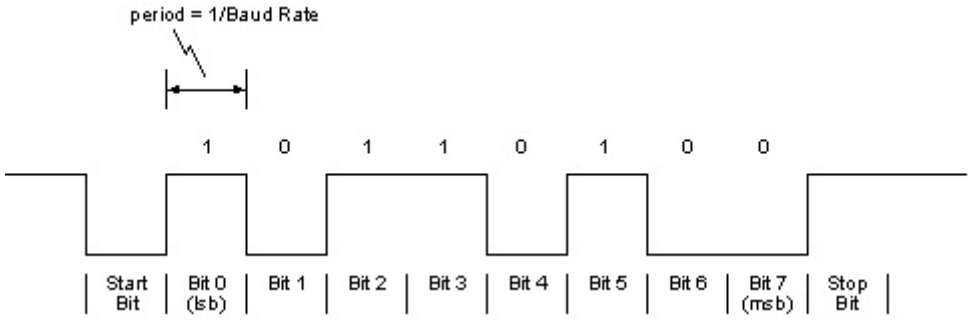
The RS232C port is for serial communication. We can use a cross cable or null MODEM cable to connect between the kit and terminal. The connector for both sides are DB9 female. We may build it or buying from computer stores.



For new PC or laptop computer without the RS232 port. It has only USB port, we may have the RS232C port by using the USB to RS232 converter.

DATA FRAME for UART COMMUNICATION

Serial data that communicated between kit and terminal is asynchronous format. The 6802 kit has no UART chip, instead it uses software controlled to produce bit rate of 2400 bit/s. The data frame is composed of start bit, 8-data bit and stop bit. For our kit, period = 1/2400 = 417 microseconds.



Since bit period is provided by machine cycle delay. Thus to send/receive serial data correctly, all interrupts must be disabled.

CONNECTING LCD MODULE

JR1 is 20-pin header for connecting the LCD module. The example shows connecting the 16x2 lines text LCD module. R12 is a current limit resistor for back-light. R13 is trimmer POT for contrast adjustment. The LCD module is interfaced to the 6802 bus directly. The command and data registers are located in I/O space having address from 9000H to 9003H.

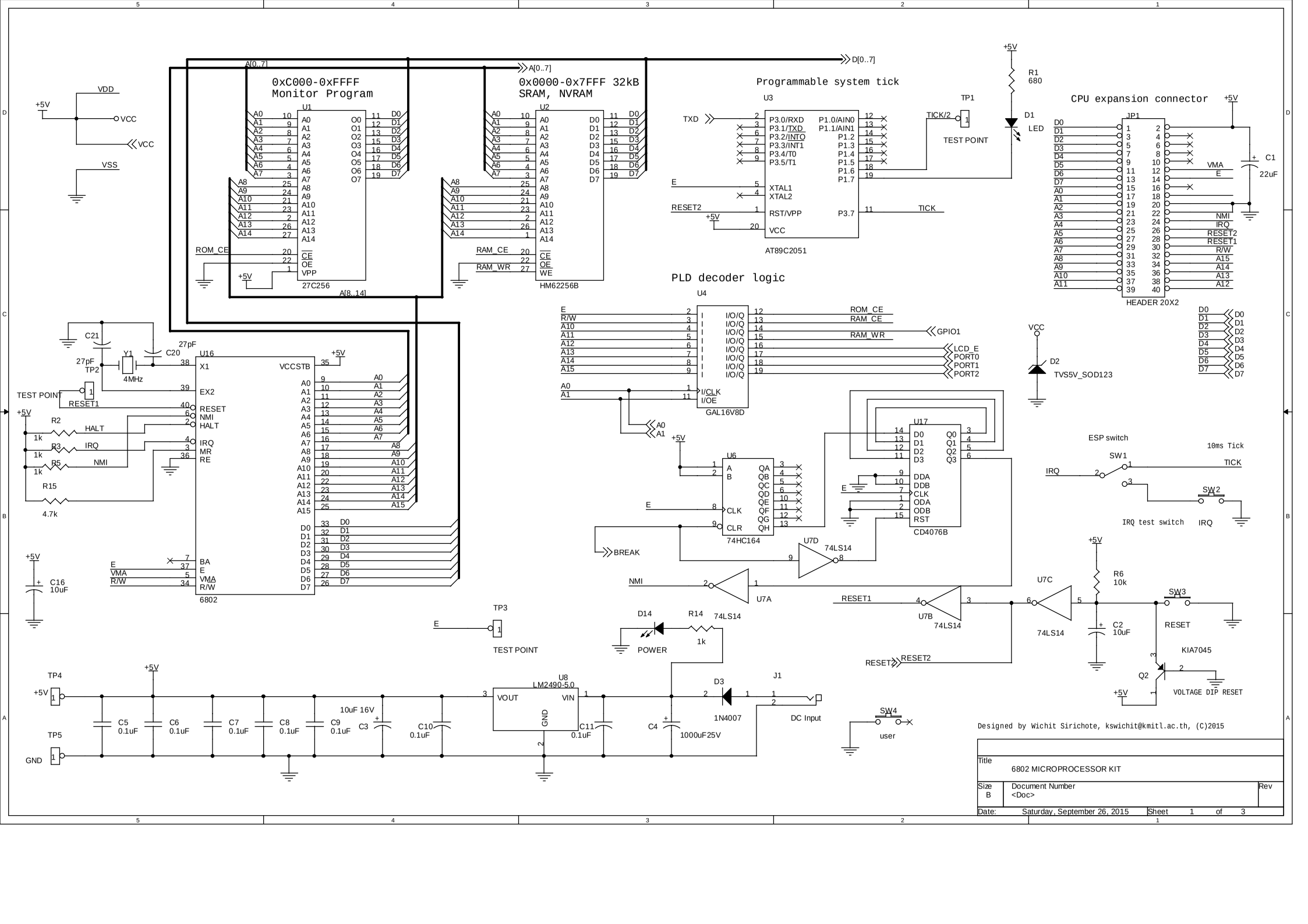
Be advised that plugging or removing the LCD module must be done when the kit is powered off.

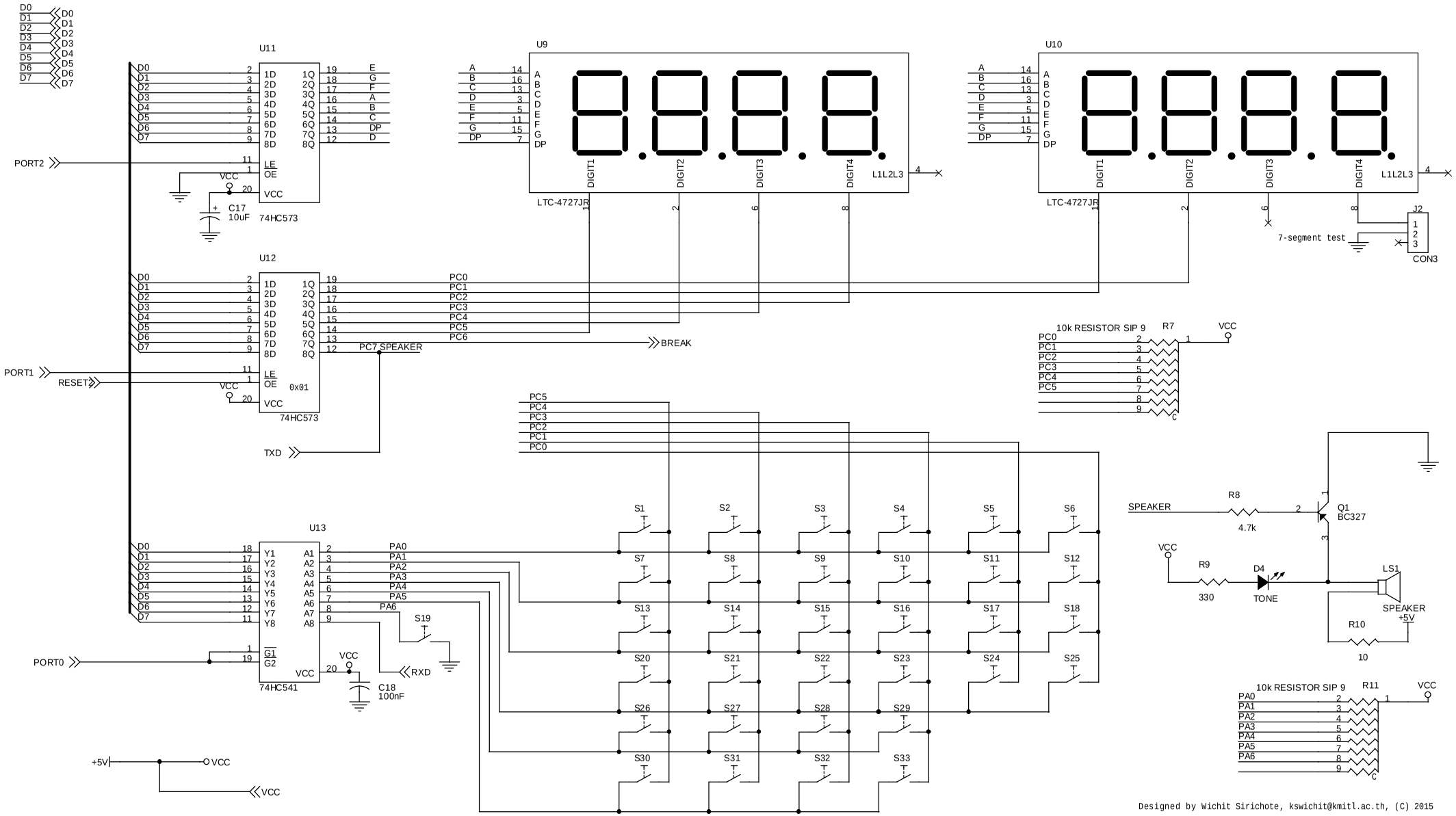
Text LCD module accepts ASCII codes for displaying the message on screen.
Without settings the LCD by software, no characters will be displayed. The first line will be black line by adjusting the R13 for contrast adjustment.

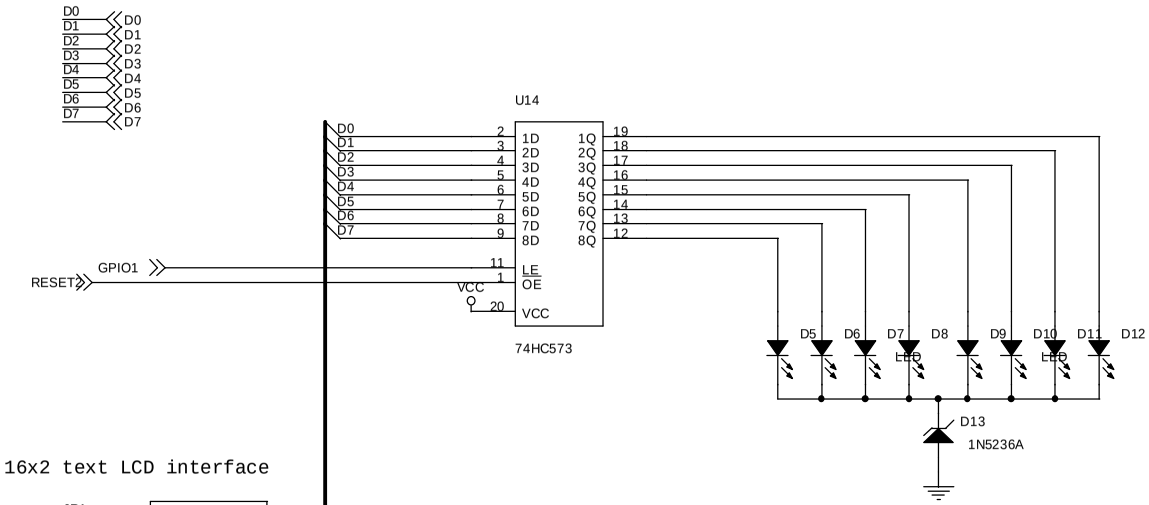
LOGIC PROBE POWER SUPPLY

The kit provides test points TP4(+5V) and TP5(GND) for using the logic probe. Students may learn digital logic signals with logic probe easily. The important signals are RESET (TP2) and E clock (TP3). Tick signal, however indicated by D1 LED blinking. Logic probe can test it at P3.7 of the 89C2051 microcontroller directly. Red clip is for +5V and Black clip for GND.

HARDWARE SCHEMATIC, PARTS LIST and PCB layout

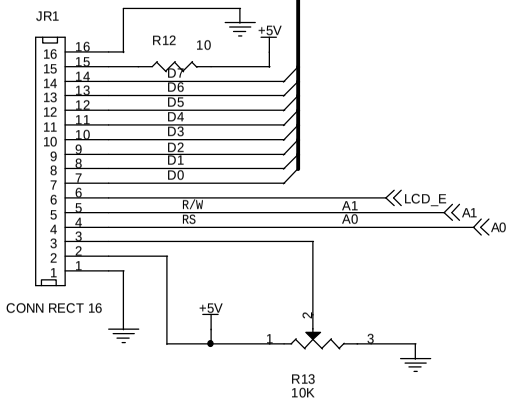

0xC000-0xFFFF
Monitor Program
U1
27C256
0x0000-0x7FFF 32kB
SRAM, NVRAM
U2
HM62256B
Programmable system tick
U3
AT89C2051
TP1
TEST POINT
TICK/2
TICK
R1
680
D1
LED
CPU expansion connector
JP1
HEADER 20X2
C1
22uF
PLD decoder logic
U4
GAL16V8D
D2
TVS5V_SOD123
U16
6802
C21
27pF
C20
27pF
Y1
4MHz
TP2
RESET1
R2
1k
HALT
R3
1k
IRQ
R5
1k
NMI
R15
4.7k
U6
74HC164
U17
CD4076B
U7D
74LS14
U7A
U7B
74LS14
U7C
74LS14
ESP switch
SW1
SW2
10ms Tick
IRQ test switch
R6
10k
SW3
RESET
C2
10uF
KIA7045
Q2
VOLTAGE DIP RESET
TP3
TEST POINT
D14
POWER
R14
1k
U8
LM2490-5.0
D3
1N4007
J1
DC Input
SW4
user
TP4
TP5
C5
0.1uF
C6
0.1uF
C7
0.1uF
C8
0.1uF
C9
0.1uF
C3
10uF 16V
C10
0.1uF
C11
0.1uF
C4
1000uF25V
C16
10uF
Designed by Wichit Sirichote, kswichit@kmitl.ac.th, (C)2015
Title
6802 MICROPROCESSOR KIT
Size
B
Document Number
<Doc>
Rev
Date:
Saturday, September 26, 2015
Sheet
1
of
3

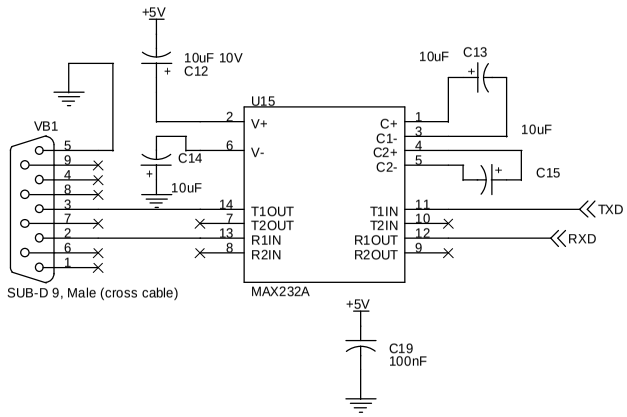

U11
74HC573
U12
74HC573
U13
74HC541
U9
LTC-4727JR
U10
LTC-4727JR
L1L2L3
DIGIT1
DIGIT2
DIGIT3
DIGIT4
PORT2
PORT1
RESET
PORT0
VCC
C17
10uF
C18
100nF
0x01
PC7 SPEAKER
BREAK
TXD
RXD
7-segment test
CON3
10k RESISTOR SIP 9
R7
R8
4.7k
Q1
BC327
SPEAKER
R9
330
D4
TONE
LS1
SPEAKER
+5V
R10
10
R11
S1
S2
S3
S4
S5
S6
S7
S8
S9
S10
S11
S12
S13
S14
S15
S16
S17
S18
S19
S20
S21
S22
S23
S24
S25
S26
S27
S28
S29
S30
S31
S32
S33
+5V
Designed by Wichit Sirichote, kswichit@kmitl.ac.th, (C) 2015
Title
6802 MICROPROCESSOR KIT
Size
B
Document Number
<Doc>
Rev
1
Date:
Saturday, September 26, 2015
Sheet
2
of
3



16x2 text LCD interface

8-bit Binary display LED x8





Designed by Wichit Sirichote, kswichit@kmitl.ac.th, (C)2015

Title	6802 MICROPROCESSOR KIT	
Size B	Document Number <Doc>	Rev 1
Date: Saturday, September 26, 2015	Sheet 3 of 3	

PARTS LIST

Semiconductors

U1 27C256, 32kB EPROM
U2 HM62256B, 32kB SRAM
U3 AT89C2051, 2kB Microcontroller
U4 GAL16V8D, PLD
U6 74HC164, shift register
U7 74LS14, hex inverter
U8 LM7805, voltage reguator
U10,U9 LTC-4727JR, common cathode LED
U11,U12,U14 74HC573, 8-bit Latch
U13 74HC541, tri-state buffer
U15 MAX232A, RS232 level converter
U16 6802 Motorola 8-bit Microprocessor
U17 CD4076B, D-FF
D2 TVS5V_SOD123
D3 1N4007, rectifier diode
D4 TONE, LED
D13 1N5236A, zener diode
Q1 BC327, PNP transistor
Q2 KIA7045, reset chip

Resistors (all resistors are 1/8W +/-5%)

R1 680
R2,R3,R5 1k
R8,R4 4.7k
R13,R6 10k
R11,R7 10k RESISTOR SIP 9
R9 330
R12,R10 10

Capacitors

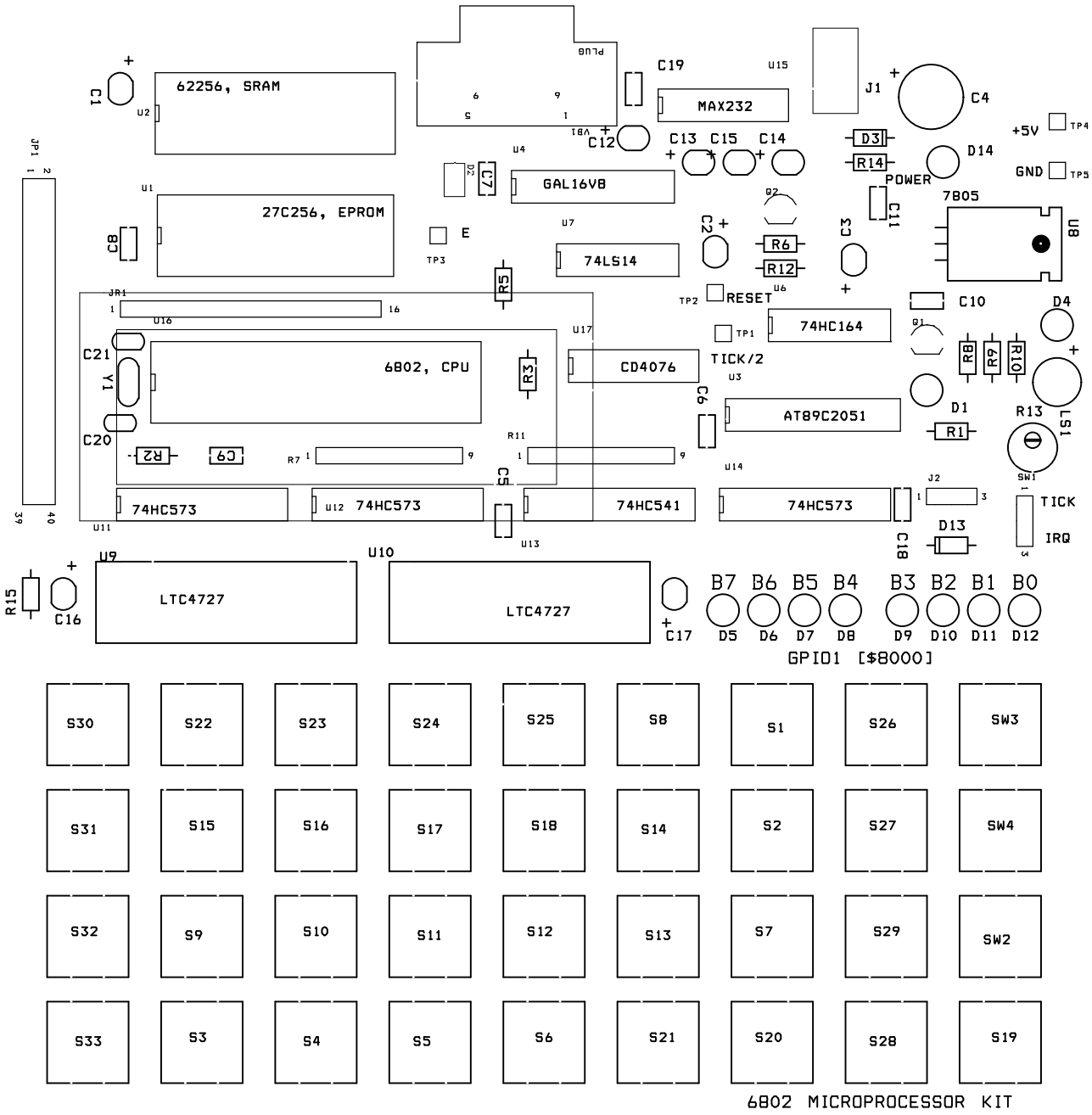
C1 22uF electrolytic
C2,C13,C14,C15,C16,C17 10uF electrolytic
C3 10uF 16V electrolytic
C4 1000uF25V electrolytic
C5,C6,C7,C8,C9 0.1uF disc ceramic
C10,C11 0.1uF disc ceramic
C12 10uF 10V electrolytic
C19,C18 100nF disc ceramic

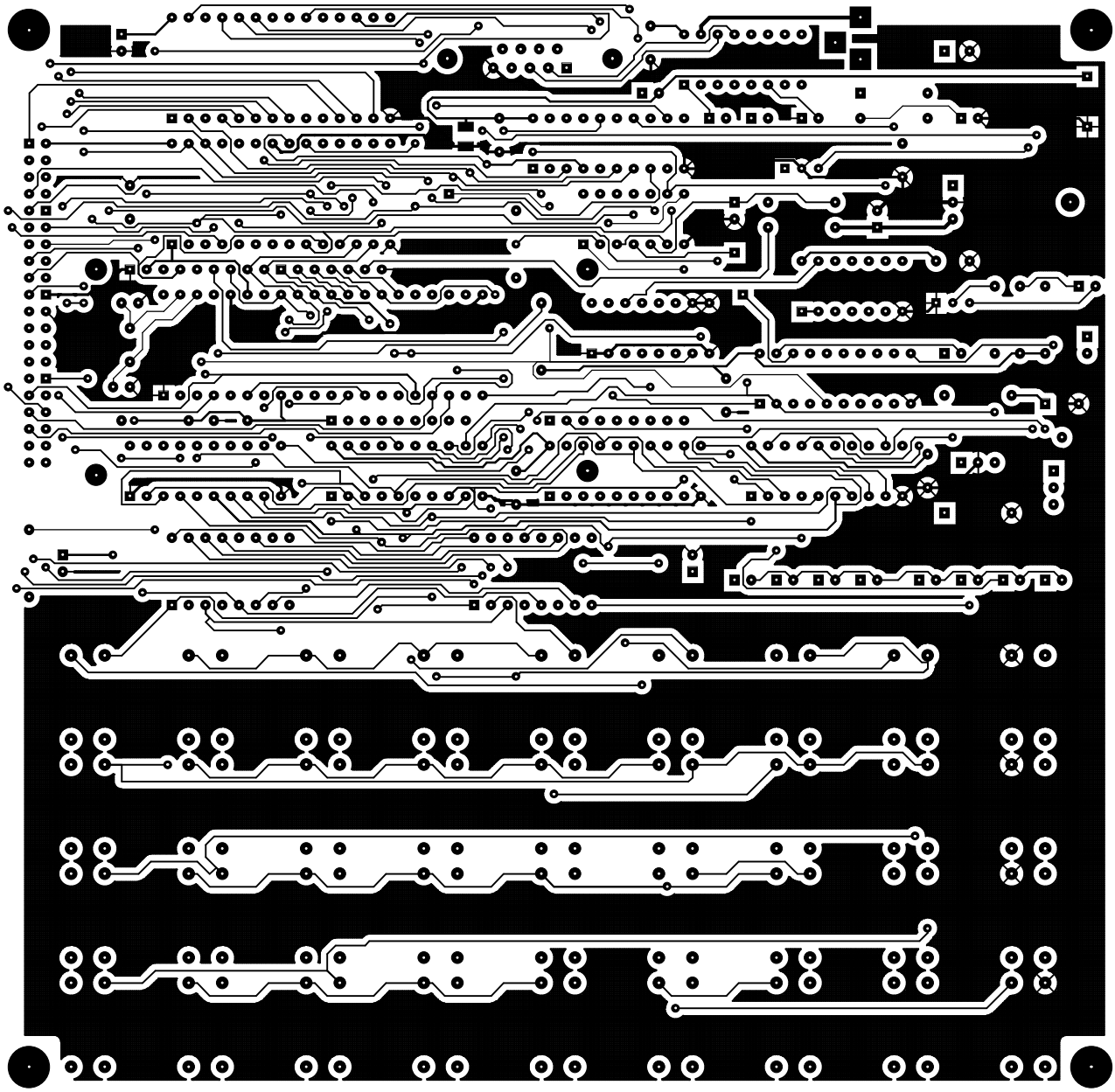
Additional parts

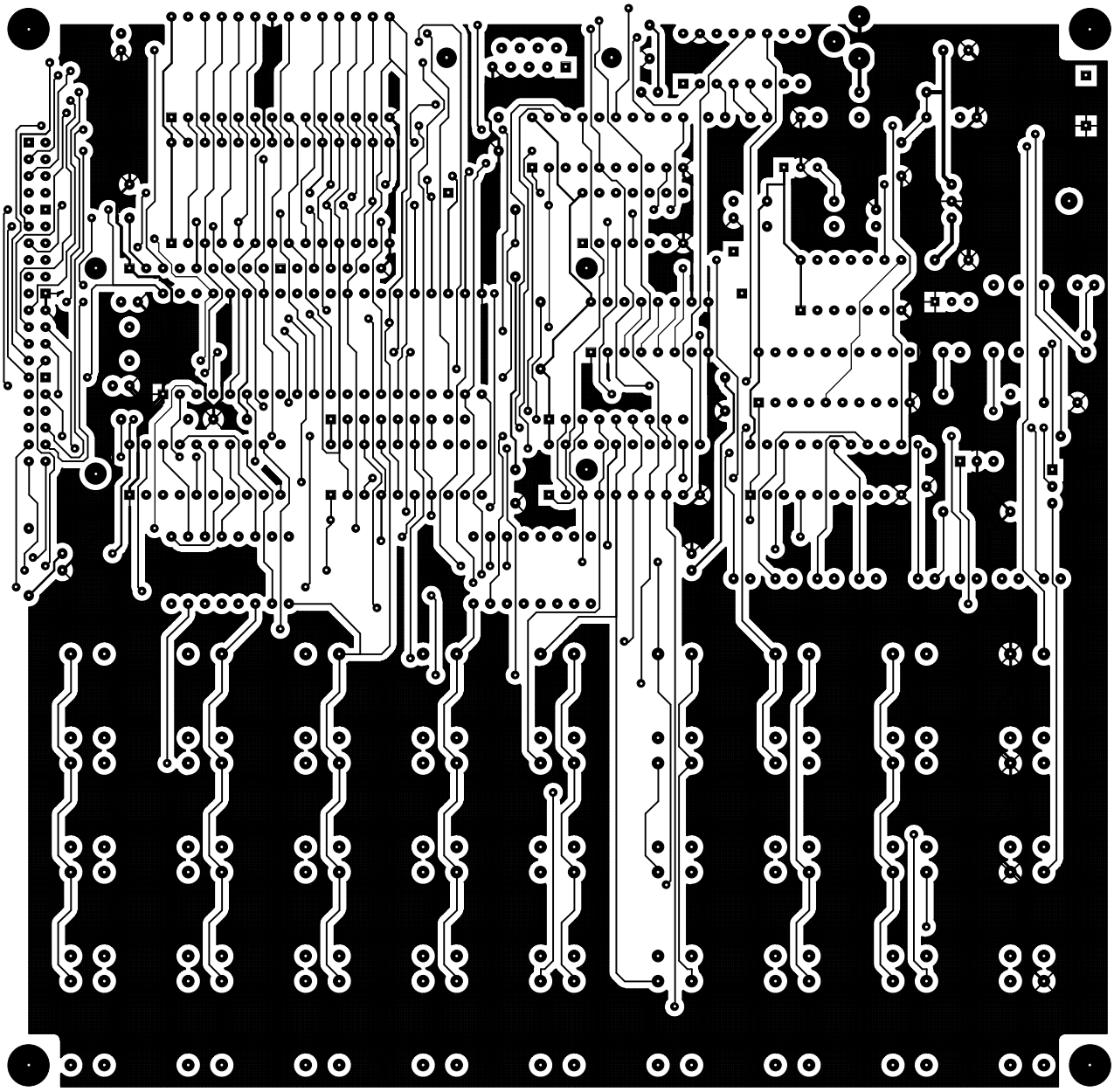
JJP1 HEADER 20X2
JR1 CONN RECT 16 text LCD connector
J1 DC Input
J2 CON3 3-pin header
LS1 SPEAKER

SW1 ESP switch
SW2 IRQ
SW3 RESET
SW4 user
S1,S2,S3,S4,S5,S6,S7,S8,
S9,S10,S11,S12,S13,S14,Tact switch
S15,S16,S17,S18,S19,S20,
S21,S22,S23,S24,S25,S26,
S27,S28,S29,S30,S31,S32,
S33
TP1,TP2,TP3 TEST POINT
TP4 +5V
TP5 GND

VB1 SUB-D 9, Male (cross cable)
Y1 4MHz Xtal
PCB double side plate through hole
LED cover Clear RED color acrylic plastic
Keyboard sticker printable SVG file


62256, SRAM
27C256, EPROM
GAL16V8
74LS14
6802, CPU
CD4076
74HC164
AT89C2051
MAX232
7805
74HC573
74HC573
74HC541
74HC573
LTC4727
LTC4727
RESET
TICK/2
POWER
+5V
GND
TICK
IRQ
B7 B6 B5 B4 B3 B2 B1 B0
D5 D6 D7 D8 D9 D10 D11 D12
GPIO1 [$8000]
S30 S22 S23 S24 S25 S8 S1 S26 SW3
S31 S15 S16 S17 S18 S14 S2 S27 SW4
S32 S9 S10 S11 S12 S13 S7 S29 SW2
S33 S3 S4 S5 S6 S21 S20 S28 S19
6802 MICROPROCESSOR KIT





MONITOR PROGRAM LISTINGS

```
0001   0000             ; MONITOR PROGRAM FOR 6802 MICROPROCESSOR KIT
0002   0000             ; (C) COPYRIGHT 2015 BY WICHIT SIRICHOTE
0003   0000             ;
0004   0000             ; Source code was translated by tasm assembler with command
0005   0000             ; d:>tasm -68 mon68.asm mon68.hex
0006   0000
0007   0000             ; 12 September 2015 fix 16-bit variable allocation
0008   0000             ;                   HIGH BYTE DISPLAY
0009   0000             ;                   LOW BYTE  DISPLAY+1
0010   0000
0011   0000             ; 13 September 2015 test single step with 74LS164
0012   0000             ;                   8th clock will make NMI low!
0013   0000             ;                   The number of clock must be 7 after the break sign
0014   0000             ;                   has been set. So the 8th clock will make NMI low
0015   0000             ;                   User instruction will be fetched and executed at t
0016   0000
0017   0000             ;                   Add key INS and DEL
0018   0000
0019   0000             ; 14 September 2015 provide RAM vector for IRQ at location 00F8H
0020   0000             ;                   CPU will jump to 00F8H if IRQ was triggered and In
0021   0000             ;                   mask has been cleared.
0022   0000
0023   0000
0024   0000             ; 16 September 2015 Testing single step with 4076
0025   0000             ;                   add 4-bit D-FF 4076 to provide 4-bit shifting
0026   0000             ;                   use with RTI 10 cycles to make NMI service
0027   0000
0028   0000             ; 17 September 2015 Add register display, page zero display
0029   0000
0030   0000             ; 18 September 2015 test software UART at 2400 Hz, found half bit 1207
0031   0000             ;                   or 2414Hz, % error = 0.6%
0032   0000             ;                   test receive hex file
0033   0000             ; 19 September 2015 add LCD drivers to monitor program
0034   0000             ;                   begin test monitor code in RAM @4000h
0035   0000
0036   0000             ; 20 September 2015 add Motorola S record for hex file download comman
0037   0000             ;                   tested with tasm -68 -g2 led.asm
0038   0000
0039   0000             ; 21 September 2015 fix s-record lodaing, check number of byte include
0040   0000             ;                    s-record loading was tested with tasm -g2 option
0041   0000
0042   0000             ;                   add key mute for beep on/off
0043   0000
0044   0000             ;                   prepare to backup the source code!
0045   0000
0046   0000             ; 23 September 2015 add cold/warm boot setup
0047   0000
0048   0000             ; 25 September 2015 add function copy block of memory and cold message
0049   0000             ;                   add hex dump function for displaying hex contents
0050   0000             ;                   on 2400 terminal
0051   0000             ; 28 Spetember 2015 fix rep key bouncing! when pressed together with k
0052   0000             ;                   fix ASCII printing with key DUMP
0053   0000
0054   0000             ; address of the I/O ports
0055   0000
0056   0000
0057   0000             GPIO1   .EQU 8000H
0058   0000             PORT0   .EQU 8001H
0059   0000             PORT1   .EQU 8002H
0060   0000             PORT2   .EQU 8003H
0061   0000
0062   0000             DIGIT   .EQU 8002H
0063   0000             SEG7    .EQU 8003H
0064   0000             KIN     .EQU 8001H
0065   0000
0066   0000
0067   0000             ONE     .EQU 30H
0068   0000             ZERO    .EQU 0BDH
0069   0000
0070   0000
0071   0000             BUSY              .EQU 80H
0072   0000
0073   0000             ; below LCD's registers are mapped into memory space
0074   0000
0075   0000             command_write     .EQU 9000H
0076   0000             data_write        .EQU 9001H
```

```
0077   0000             command_read      .EQU 9002H
0078   0000             data_read         .EQU 9003H
0079   0000
0080   0000
0081   0000
0082   0000              ; page zero register definition
0083   0000              ; LOCATION $00 TO $7F ARE 128 BYTES FOR USER PROGRAM USE
0084   0000
0085   0000              .DSEG
0086   0080              .ORG 80H
0087   0080
0088   0080             ; zero page memory definitions for monitor use
0089   0080             REG_E   .BLOCK 1
0090   0081             REG_D   .BLOCK 1
0091   0082             REG_B   .BLOCK 1
0092   0083             REG_C   .BLOCK 1
0093   0084             HL      .BLOCK 2           ; 84H = L 85H = H
0094   0086             DE      .BLOCK 2
0095   0088             REG_A   .BLOCK 1
0096   0089
0097   0089             _ERROR  .BLOCK 1           ; ERROR FLAG FOR INTEL HEX FILE DOWNLOADING
0098   008A             BCC     .BLOCK 2           ; BYTE CHECK SUM
0099   008C             BUFFER  .BLOCK 6          ; 8BH - 90H PAGE ZERO DISPLAY BUFFER
0100   0092             INVALID .BLOCK 1          ; INVALID KEY HAS BEEN PRESSED FLAG BIT
0101   0093                                       ; 0 VALID
0102   0093                    ; 1 INVALID
0103   0093
0104   0093             KEY     .BLOCK 1
0105   0094             STATE   .BLOCK 1
0106   0095             ZERO_FLAG .BLOCK 1          ; ZERO WHEN HEX KEY PRESSED FOR ADDRESS OR
0107   0096
0108   0096             DISPLAY .BLOCK 2         ; display address
0109   0098
0110   0098             USER_PC  .BLOCK 2       ; FOR SAVING CURRENT PC, ON RESET, IT SETS TO
0111   009A             USER_A  .BLOCK 1
0112   009B             USER_B  .BLOCK 1
0113   009C             USER_IX  .BLOCK 2
0114   009E             USER_SP  .BLOCK 2         ; USER STACK POINTER
0115   00A0             USER_P  .BLOCK 1         ; PROGRAM STATUS REGISTER
0116   00A1             SAVE_SP .BLOCK 2         ; SAVE SYSTEM STACK
0117   00A3
0118   00A3             START_ADDRESS .BLOCK 2
0119   00A5             DESTINATION  .BLOCK 2    ; FOR OFFSET BYTE CALCULATION
0120   00A7             OFFSET_BYTE .BLOCK 2     ; OFFSET BYTE = DESTINATION - START_ADDRESS
0121   00A9
0122   00A9             END_ADDRESS .BLOCK 2      ; FOR COPY MEMORY FUNCTION
0123   00AB
0124   00AB             COLD    .BLOCK 2         ; COLD BOOT OR WARM BOOT
0125   00AD
0126   00AD             REPDELAY .BLOCK 1
0127   00AE
0128   00AE             DEBUG    .BLOCK 2      ; FOR PROGRAM DEBUGGING
0129   00B0             TEMP16   .BLOCK 2      ; REGISTER 16 BITS
0130   00B2
0131   00B2             IX2      .BLOCK 2
0132   00B4
0133   00B4             OFFSET   .BLOCK 2      ; FOR X INDEXT REGISTER USE
0134   00B6             BIN2SEG  .BLOCK 16     ; TABLE FOR 4-BIT BINARY TO 7-SEGMENT PATTERN
0135   00C6             BEEP_FREQ .BLOCK 1     ; STORE BEEP FREQUENCY
0136   00C7             BEEP_PERIOD .BLOCK 2   ; STORE BEEP PERIOD LOAD WITH IX
0137   00C9             DEMO_NO   .BLOCK 1     ; FOR DEMO 0-9
0138   00CA             MUTE      .BLOCK 1     ; TOGGLE MUTE FOR BEEP ON/OFF 0=BEEP ON
0139   00CB             BIT1_COUNTER .BLOCK 1  ; COUNTER FOR BIT 1 TEST
0140   00CC
0141   00CC
0142   00CC             SEC100    .BLOCK 1     ; VARIABLES FOR CLOCK PROGRAM
0143   00CD             SEC       .BLOCK 1
0144   00CE             MIN       .BLOCK 1
0145   00CF             HOUR      .BLOCK 1
0146   00D0             RUNSTOP   .BLOCK 1
0147   00D1
0148   00D1             K         .BLOCK 1     ; GENERAL PURPOSE REGISTERS
0149   00D2             J         .BLOCK 1
0150   00D3
0151   00D3
0152   00D3
```

```
0153   00D3                        .CSEG
0154   00D3
0155   00D3
0156   C000                        .ORG 0C000H    ; START ADDRESS FOR ROM
0157   C000
0158   C000                  ;       .ORG 4000H     ; TEST IN RAM
0159   C000
0160   C000
0161   C000 86 BF       START      LDAA #$BF
0162   C002 B7 80 02               STAA PORT1     ; DISABLE NMI BREAK SIGNAL
0163   C005 86 00                  LDAA #0
0164   C007 B7 80 00               STAA GPIO1     ; TEST DISPLAY
0165   C00A
0166   C00A
0167   C00A              ; power up delay
0168   C00A
0169   C00A CE 01 2C                    LDX #300
0170   C00D 09           POWER_UP_DELAY DEX
0171   C00E 26 FD                BNE POWER_UP_DELAY
0172   C010
0173   C010
0174   C010 8E 7F FF             LDS #$7FFF     ; SYSTEM STACK
0175   C013
0176   C013 CE 7F 00             LDX #$7F00
0177   C016 DF 9E                STX USER_SP   ; STORE USER STACK
0178   C018
0179   C018
0180   C018 86 00                LDAA #0
0181   C01A 97 8C                STAA BUFFER
0182   C01C 97 8D        STAA BUFFER+1
0183   C01E 97 92        STAA INVALID   ; CLEAR INVALID FLAG
0184   C020
0185   C020              ; INSERT 6502 TEXT
0186   C020
0187   C020 86 AF        LDAA #0AFH
0188   C022 97 91                STAA BUFFER+5
0189   C024 86 BF                LDAA #0BFH
0190   C026 97 90        STAA BUFFER+4
0191   C028 86 BD        LDAA #0BDH
0192   C02A 97 8F        STAA BUFFER+3
0193   C02C 86 9B        LDAA #9BH
0194   C02E 97 8E        STAA BUFFER+2
0195   C030
0196   C030
0197   C030 86 00        LDAA #0
0198   C032 97 94                STAA STATE    ; INITIAL STATE
0199   C034 97 95        STAA ZERO_FLAG
0200   C036
0201   C036 86 02                LDAA #02
0202   C038 97 96        STAA DISPLAY
0203   C03A 97 98                STAA USER_PC
0204   C03C 86 00                LDAA #0
0205   C03E 97 97        STAA DISPLAY+1
0206   C040 97 99                STAA USER_PC+1
0207   C042
0208   C042 07                   TPA
0209   C043 97 A0                STAA USER_P
0210   C045
0211   C045
0212   C045 96 96                LDAA DISPLAY
0213   C047 97 84        STAA HL
0214   C049 96 97        LDAA DISPLAY+1
0215   C04B 97 85        STAA HL+1
0216   C04D
0217   C04D
0218   C04D CE 00 35             LDX #35H
0219   C050 97 C7                STAA BEEP_PERIOD
0220   C052
0221   C052 86 52                LDAA #52H
0222   C054 97 C6                STAA BEEP_FREQ
0223   C056
0224   C056 86 00                LDAA #0
0225   C058 97 C9                STAA DEMO_NO
0226   C05A
0227   C05A DE AB                LDX COLD
0228   C05C 8C AA 55             CPX #0AA55H
```

```
0229   C05F 27 18              BEQ SKIP_COLD
0230   C061
0231   C061             ; COLD START
0232   C061
0233   C061 CE AA 55            LDX #0AA55H
0234   C064 DF AB               STX COLD
0235   C066
0236   C066 86 00               LDAA #0
0237   C068 97 CA               STAA MUTE      ; BEEP IS ON
0238   C06A
0239   C06A 86 FF               LDAA #0FFH
0240   C06C B7 80 00            STAA GPIO1
0241   C06F
0242   C06F
0243   C06F BD C9 AD            JSR BEEP
0244   C072
0245   C072 BD CB 43            JSR DISPLAY_START_MSG
0246   C075
0247   C075
0248   C075 4F                  CLRA
0249   C076 B7 80 00            STAA GPIO1
0250   C079
0251   C079
0252   C079
0253   C079             SKIP_COLD
0254   C079
0255   C079                  ;   LDX #TEXT3
0256   C079                  ;   JSR PSTR       ; PRINT TEXT3 TESTING 2400 TERMINAL
0257   C079
0258   C079
0259   C079
0260   C079 7E C9 A2             JMP MAIN
0261   C07C
0262   C07C
0263   C07C
0264   C07C             ;----------------------- 2400 BIT/S SOFTWARE UART --------------------
0265   C07C             ; one bit delay for 2400 bit/s UART
0266   C07C
0267   C07C C6 3D       BIT_DELAY LDAB #61     ; 1207 TESTED WITH 55H @1MHz (4MHz xtal)
0268   C07E 5A          LOOP      DECB
0269   C07F 26 FD                 BNE LOOP
0270   C081 39            RTS
0271   C082
0272   C082             ; 1.5 bit delay
0273   C082
0274   C082 C6 5C       BIT1_5_DELAY LDAB #92      ; DELAY 1.5 BIT
0275   C084 5A          LOOP1      DECB
0276   C085 26 FD                 BNE LOOP1
0277   C087 39            RTS
0278   C088
0279   C088             ; SEND ASCII LETTER TO TERMINAL
0280   C088             ; ENTRY: A
0281   C088
0282   C088 97 80       SEND_BYTE: STAA REG_E  ; SAVE ACCUMULATOR
0283   C08A
0284   C08A 86 3F                 LDAA #3FH    ; start bit is zero
0285   C08C B7 80 02      STAA PORT1
0286   C08F 8D EB                 BSR BIT_DELAY    ; delay one bit
0287   C091
0288   C091 86 08                 LDAA #8      ; 8-data bit wil be sent
0289   C093 97 81         STAA REG_D
0290   C095
0291   C095 96 80       CHK_BIT:   LDAA REG_E
0292   C097 84 01                  ANDA #1
0293   C099 27 07           BEQ SEND_ZERO
0294   C09B
0295   C09B 86 BF           LDAA #0BFH
0296   C09D B7 80 02        STAA PORT1
0297   C0A0
0298   C0A0 20 07                  BRA NEXT_BIT
0299   C0A2
0300   C0A2
0301   C0A2 86 3F       SEND_ZERO: LDAA #3FH
0302   C0A4 B7 80 02               STAA PORT1
0303   C0A7 20 00                  BRA NEXT_BIT
0304   C0A9
```

```
0305   C0A9 8D D1       NEXT_BIT:  BSR BIT_DELAY
0306   C0AB
0307   C0AB 74 00 80               LSR REG_E
0308   C0AE 7A 00 81          DEC REG_D
0309   C0B1 26 E2                  BNE CHK_BIT
0310   C0B3
0311   C0B3 86 BF                  LDAA #0BFH
0312   C0B5 B7 80 02          STAA PORT1
0313   C0B8 8D C2                  BSR BIT_DELAY
0314   C0BA
0315   C0BA 39                     RTS
0316   C0BB
0317   C0BB
0318   C0BB
0319   C0BB             ; RECEIVE BYTE FROM 2400 BIT/S TERMINAL
0320   C0BB             ; EXIT: A
0321   C0BB
0322   C0BB B6 80 01    CIN   LDAA PORT0
0323   C0BE 84 80               ANDA #80H
0324   C0C0 26 F9         BNE CIN
0325   C0C2
0326   C0C2 8D BE                BSR BIT1_5_DELAY
0327   C0C4
0328   C0C4 86 07         LDAA #7
0329   C0C6 97 81         STAA REG_D
0330   C0C8 86 00         LDAA #0
0331   C0CA 97 80         STAA REG_E
0332   C0CC
0333   C0CC
0334   C0CC
0335   C0CC B6 80 01    CHK_BIT_RX   LDAA PORT0
0336   C0CF 84 80                  ANDA #80H
0337   C0D1 26 08                  BNE BIT_IS_ONE
0338   C0D3
0339   C0D3 96 80                  LDAA REG_E
0340   C0D5 84 7F                  ANDA #7FH
0341   C0D7 97 80          STAA REG_E
0342   C0D9 20 08                  BRA NEXT_BIT_RX
0343   C0DB
0344   C0DB 96 80       BIT_IS_ONE LDAA REG_E
0345   C0DD 8A 80                  ORAA #80H
0346   C0DF 97 80          STAA REG_E
0347   C0E1 20 00                  BRA NEXT_BIT_RX
0348   C0E3
0349   C0E3 8D 97       NEXT_BIT_RX  BSR BIT_DELAY
0350   C0E5
0351   C0E5 74 00 80               LSR REG_E
0352   C0E8
0353   C0E8 7A 00 81               DEC REG_D
0354   C0EB 26 DF          BNE CHK_BIT_RX
0355   C0ED
0356   C0ED 8D 8D                  BSR BIT_DELAY    ; CENTER OF STOP BIT
0357   C0EF
0358   C0EF 96 80          LDAA REG_E
0359   C0F1
0360   C0F1 39             RTS
0361   C0F2
0362   C0F2
0363   C0F2
0364   C0F2             CR        .EQU 0DH
0365   C0F2             LF        .EQU 0AH
0366   C0F2             EOS       .EQU 0
0367   C0F2
0368   C0F2             ;NEW LINE
0369   C0F2             ; PRINT CR, LF
0370   C0F2
0371   C0F2 86 0D       NEW_LINE     LDAA #0DH
0372   C0F4 BD C0 88         JSR SEND_BYTE
0373   C0F7 86 0A            LDAA #0AH
0374   C0F9 BD C0 88         JSR SEND_BYTE
0375   C0FC 39               RTS
0376   C0FD
0377   C0FD
0378   C0FD             ; WRITE NIBBLE TO TERMINAL
0379   C0FD
0380   C0FD 84 0F       OUT1X        ANDA #0FH
```

```
0381   C0FF
0382   C0FF 8B 30                    ADDA #30H
0383   C101 81 3A                    CMPA  #3AH
0384   C103 2D 02                    BLT OUT1X1
0385   C105 8B 07                    ADDA #7
0386   C107 BD C0 88    OUT1X1       JSR SEND_BYTE
0387   C10A 39               RTS
0388   C10B
0389   C10B
0390   C10B 36          OUT2X        PSHA
0391   C10C
0392   C10C 44                       LSRA
0393   C10D 44                       LSRA
0394   C10E 44                       LSRA
0395   C10F 44                       LSRA
0396   C110
0397   C110                ;  STAA GPIO1
0398   C110
0399   C110 BD C0 FD          JSR OUT1X
0400   C113 32                       PULA
0401   C114 BD C0 FD          JSR OUT1X
0402   C117 39                RTS
0403   C118
0404   C118              ; PRINT LINE OF MEMORY POINTED TO HL
0405   C118
0406   C118 86 10       PRINT_LINE   LDAA #16
0407   C11A 97 D2                    STAA J
0408   C11C
0409   C11C              PRINT_LINE1
0410   C11C BD C0 F2                 JSR NEW_LINE
0411   C11F 86 10                    LDAA #16
0412   C121 97 83                    STAA REG_C
0413   C123
0414   C123 DE 84                    LDX HL
0415   C125 DF B0                    STX TEMP16  ; FOR ASCII PRINTING
0416   C127
0417   C127 96 84                    LDAA HL
0418   C129 BD C1 0B          JSR OUT2X
0419   C12C 96 85                    LDAA HL+1
0420   C12E BD C1 0B          JSR OUT2X
0421   C131
0422   C131 86 3A             LDAA #':'
0423   C133 BD C0 88          JSR SEND_BYTE
0424   C136 86 20                    LDAA #' '
0425   C138 BD C0 88                 JSR SEND_BYTE
0426   C13B
0427   C13B
0428   C13B DE 84       PRINT_LINE2  LDX HL
0429   C13D
0430   C13D A6 00                    LDAA 0,X
0431   C13F
0432   C13F BD C1 0B          JSR OUT2X
0433   C142
0434   C142 86 20             LDAA #' '
0435   C144 BD C0 88          JSR SEND_BYTE
0436   C147
0437   C147 DE 84                    LDX HL
0438   C149 08                       INX
0439   C14A DF 84                    STX HL
0440   C14C
0441   C14C 7A 00 83          DEC REG_C
0442   C14F
0443   C14F 26 EA             BNE PRINT_LINE2
0444   C151              ;-----------------------------------
0445   C151
0446   C151              ; PRINT ASCII CODE
0447   C151
0448   C151
0449   C151 86 10                    LDAA #16
0450   C153 97 83                    STAA REG_C
0451   C155              PRINT_IT1
0452   C155 DE B0                    LDX TEMP16
0453   C157
0454   C157 A6 00                    LDAA 0,X
0455   C159
0456   C159 81 20                    CMPA #20H
```

```
0457   C15B 2C 02                  BGE  PRINT_IT
0458   C15D
0459   C15D 86 2E                  LDAA #'.'
0460   C15F
0461   C15F BD C0 88    PRINT_IT   JSR SEND_BYTE
0462   C162
0463   C162 DE B0                  LDX TEMP16
0464   C164 08                     INX
0465   C165 DF B0                  STX TEMP16
0466   C167
0467   C167 7A 00 83               DEC REG_C
0468   C16A 26 E9                  BNE PRINT_IT1
0469   C16C
0470   C16C             ;---------------------------------
0471   C16C
0472   C16C 7A 00 D2               DEC J
0473   C16F 26 AB                  BNE PRINT_LINE1
0474   C171
0475   C171
0476   C171 39               RTS
0477   C172
0478   C172
0479   C172 86 FF       KEY_DUMP   LDAA #$FF
0480   C174 97 CA                  STAA MUTE      ;TURN BEEP OFF
0481   C176
0482   C176
0483   C176 DE 96                  LDX DISPLAY
0484   C178 DF 84                  STX HL
0485   C17A
0486   C17A BD C1 18               JSR PRINT_LINE
0487   C17D
0488   C17D BD C0 F2               JSR NEW_LINE
0489   C180
0490   C180 DE 84                  LDX HL
0491   C182 DF 96                  STX DISPLAY
0492   C184 BD C4 06               JSR STILL_DATA
0493   C187
0494   C187 39                     RTS
0495   C188
0496   C188
0497   C188
0498   C188
0499   C188
0500   C188             ;------------------------------SOFTWARE UART -----------------------
0501   C188
0502   C188
0503   C188 96 00       LOOP2      LDAA $0
0504   C18A B7 80 00               STAA GPIO1
0505   C18D BD C1 95               JSR DELAY
0506   C190 7C 00 00               INC $0
0507   C193 20 F3                  BRA LOOP2
0508   C195
0509   C195 CE 27 10    DELAY      LDX #10000
0510   C198 09          DELAY1     DEX
0511   C199 26 FD                  BNE DELAY1
0512   C19B 39                     RTS
0513   C19C
0514   C19C
0515   C19C
0516   C19C             ; CONVERT LOW NIBBLE IN ACCUMULATOR TO 7-SEGMENT PATTERN
0517   C19C             ; ENTRY: A
0518   C19C             ; EXIT: A
0519   C19C
0520   C19C             NIBBLE_7SEG
0521   C19C
0522   C19C CE CB FB             LDX  #SEGTAB
0523   C19F
0524   C19F DF B0                STX   TEMP16
0525   C1A1
0526   C1A1 9B B1                ADDA TEMP16+1
0527   C1A3 97 B1                STAA TEMP16+1
0528   C1A5
0529   C1A5 86 00                LDAA #0
0530   C1A7 99 B0                ADCA TEMP16
0531   C1A9 97 B0                STAA TEMP16
0532   C1AB
```

```
0533   C1AB DE B0              LDX TEMP16
0534   C1AD
0535   C1AD A6 00              LDAA  0,X          ; GET 7-SEGMENT PATTERN
0536   C1AF
0537   C1AF 39                 RTS
0538   C1B0
0539   C1B0             ; CONVERT BYTE TO 7-SEGMENT PATTERN
0540   C1B0             ; ENTRY: A
0541   C1B0             ; EXIT: DE
0542   C1B0
0543   C1B0 36          BYTE_7SEG   PSHA
0544   C1B1
0545   C1B1 84 0F                   ANDA #0FH
0546   C1B3 BD C1 9C                JSR NIBBLE_7SEG
0547   C1B6 97 86           STAA DE
0548   C1B8
0549   C1B8 32                      PULA
0550   C1B9
0551   C1B9 44                      LSRA
0552   C1BA 44                      LSRA
0553   C1BB 44                      LSRA
0554   C1BC 44                      LSRA
0555   C1BD
0556   C1BD BD C1 9C        JSR NIBBLE_7SEG
0557   C1C0 97 87           STAA DE+1
0558   C1C2
0559   C1C2 39                      RTS
0560   C1C3
0561   C1C3
0562   C1C3             ; CONVERT BYTE TO 7-SEGMENT PATTERN AND SAVE TO DISPLAY BUFFER DATA FI
0563   C1C3             ; ENTRY: A
0564   C1C3
0565   C1C3 36          DATA_DISPLAY PSHA   ; SAVE ACCUMULATOR
0566   C1C4 BD C1 B0                JSR BYTE_7SEG
0567   C1C7 96 86            LDAA DE
0568   C1C9 97 8C            STAA BUFFER
0569   C1CB 96 87            LDAA DE+1
0570   C1CD 97 8D            STAA BUFFER+1
0571   C1CF 32                      PULA
0572   C1D0 39               RTS
0573   C1D1
0574   C1D1             ; CONVERT 16-BIT ADDRESS IN HL AND SAVE IT TO ADDRESS FILED DISPLAY BU
0575   C1D1             ; ENTRY: HL
0576   C1D1
0577   C1D1             ADDRESS_DISPLAY
0578   C1D1
0579   C1D1 96 84                   LDAA HL
0580   C1D3 BD C1 B0        JSR BYTE_7SEG
0581   C1D6 96 86           LDAA DE
0582   C1D8 97 90                   STAA BUFFER+4
0583   C1DA 96 87           LDAA DE+1
0584   C1DC 97 91                   STAA BUFFER+5
0585   C1DE 96 85           LDAA HL+1
0586   C1E0 BD C1 B0        JSR BYTE_7SEG
0587   C1E3 96 86           LDAA DE
0588   C1E5 97 8E                   STAA BUFFER+2
0589   C1E7 96 87           LDAA DE+1
0590   C1E9 97 8F                   STAA BUFFER+3
0591   C1EB 39              RTS
0592   C1EC
0593   C1EC
0594   C1EC             ; CONVERT ASCII TO HEX
0595   C1EC             ; ENTRY: A
0596   C1EC
0597   C1EC             TO_HEX
0598   C1EC 80 30                SUBA #30H
0599   C1EE 81 10                CMPA #10H
0600   C1F0 2D 04                BLT ZERO_NINE
0601   C1F2 84 DF                ANDA #11011111B
0602   C1F4 80 07                SUBA #7
0603   C1F6
0604   C1F6 39          ZERO_NINE  RTS
0605   C1F7
0606   C1F7             ; CONVERT TWO ASCII LETTERS  TO SINGLE BYTE
0607   C1F7             ; EXIT: A
0608   C1F7
```

```
0609   C1F7 BD C0 BB    GET_HEX    JSR CIN
0610   C1FA BD C1 EC       JSR TO_HEX
0611   C1FD 48                    ASLA
0612   C1FE 48                    ASLA
0613   C1FF 48                    ASLA
0614   C200 48                    ASLA
0615   C201
0616   C201 B7 80 00       STAA GPIO1
0617   C204
0618   C204 97 88          STAA REG_A
0619   C206
0620   C206 BD C0 BB       JSR CIN
0621   C209 BD C1 EC       JSR TO_HEX
0622   C20C 9A 88                 ORAA REG_A
0623   C20E
0624   C20E 39             RTS
0625   C20F
0626   C20F
0627   C20F 9B 8A       ADD_BCC   ADDA BCC
0628   C211 97 8A          STAA BCC
0629   C213 39             RTS
0630   C214
0631   C214
0632   C214
0633   C214
0634   C214             ; get hex file for both Intel and Motorola s record automatically
0635   C214             ;
0636   C214             ; GET_RECORD READS INTEL HEX FILE AND SAVE TO MEMORY
0637   C214
0638   C214 86 00       GET_RECORD LDAA #0
0639   C216 97 89          STAA _ERROR
0640   C218
0641   C218 BD C0 BB    GET_RECORD1 JSR CIN
0642   C21B 81 3A                 CMPA #':'
0643   C21D 27 07          BEQ GET_RECORD2
0644   C21F
0645   C21F 81 53                 CMPA #'S'          ; if it was Motorola s record
0646   C221 26 F5                 BNE GET_RECORD1
0647   C223
0648   C223 7E C2 92              JMP GET_S_RECORD2
0649   C226
0650   C226
0651   C226             GET_RECORD2
0652   C226
0653   C226 86 00          LDAA #0
0654   C228 97 8A          STAA BCC
0655   C22A
0656   C22A BD C1 F7       JSR GET_HEX
0657   C22D 97 83          STAA REG_C     ; GET NUMBER OF BYTE
0658   C22F
0659   C22F BD C2 0F       JSR ADD_BCC
0660   C232
0661   C232 BD C1 F7       JSR GET_HEX
0662   C235 97 84                 STAA HL
0663   C237
0664   C237 BD C2 0F       JSR ADD_BCC
0665   C23A
0666   C23A BD C1 F7       JSR GET_HEX
0667   C23D 97 85                 STAA HL+1        ; GET LOAD ADDRESS
0668   C23F
0669   C23F BD C2 0F       JSR ADD_BCC
0670   C242
0671   C242 BD C1 F7       JSR GET_HEX
0672   C245
0673   C245 81 00                 CMPA #0
0674   C247
0675   C247 27 14          BEQ DATA_RECORD
0676   C249
0677   C249 BD C0 BB    WAIT_CR    JSR CIN
0678   C24C 81 0D                 CMPA #0DH
0679   C24E 26 F9          BNE WAIT_CR
0680   C250
0681   C250 B7 80 00       STAA GPIO1
0682   C253
0683   C253 96 89          LDAA _ERROR
0684   C255 81 01                 CMPA #1
```

```
0685   C257 26 03          BNE NOERROR
0686   C259
0687   C259             ; SHOW ERROR ON LED
0688   C259             ;  JSR OUT_OFF_RANGE
0689   C259
0690   C259 B7 80 00       STAA GPIO1
0691   C25C
0692   C25C             NOERROR
0693   C25C 39                  RTS
0694   C25D
0695   C25D             DATA_RECORD
0696   C25D
0697   C25D BD C1 F7       JSR GET_HEX
0698   C260 DE 84               LDX  HL
0699   C262 A7 00               STAA 0,X  ; WRITE TO MEMORY
0700   C264
0701   C264 BD C2 0F       JSR ADD_BCC
0702   C267
0703   C267 B7 80 00       STAA GPIO1
0704   C26A
0705   C26A 08                  INX
0706   C26B DF 84               STX HL
0707   C26D
0708   C26D 7A 00 83       DEC REG_C
0709   C270 26 EB          BNE DATA_RECORD ; UNTIL C=0
0710   C272
0711   C272 96 8A               LDAA BCC
0712   C274 88 FF               EORA #0FFH    ; ONE'S COMPLEMENT
0713   C276 4C                  INCA
0714   C277                                   ; TWO'S COMPLEMENT
0715   C277 97 8A          STAA BCC
0716   C279
0717   C279
0718   C279 BD C1 F7       JSR GET_HEX    ; GET BYTE CHECK SUM
0719   C27C
0720   C27C 91 8A               CMPA BCC      ; COMPARE WITH BYTE CHECK SUM
0721   C27E 27 04          BEQ SKIP11
0722   C280
0723   C280 86 01          LDAA #1
0724   C282 97 89          STAA _ERROR    ; ERROR FLAG =1
0725   C284
0726   C284
0727   C284             SKIP11
0728   C284
0729   C284 7E C2 18       JMP GET_RECORD1     ; NEXT LINE
0730   C287
0731   C287
0732   C287
0733   C287
0734   C287             ; Motorola s record downloading subrouitne
0735   C287
0736   C287             ; S1 14 0200 8601B780008D034920F8CE07D00926FD39 30
0737   C287             ;
0738   C287             ; S1 0A 0218 CE13880926FD39 0D
0739   C287             ;
0740   C287             ; S9 03 0000 FC
0741   C287             ;
0742   C287             ; 14 is pair count for remaining pair
0743   C287
0744   C287 86 00       GET_S_RECORD LDAA #0
0745   C289 97 89          STAA _ERROR
0746   C28B
0747   C28B
0748   C28B
0749   C28B             GET_S_RECORD1
0750   C28B
0751   C28B BD C0 BB            JSR CIN
0752   C28E 81 53               CMPA #'S'
0753   C290 26 F9               BNE GET_S_RECORD1
0754   C292
0755   C292             GET_S_RECORD2
0756   C292
0757   C292 BD C0 BB            JSR CIN
0758   C295 81 31               CMPA #'1'
0759   C297 27 06               BEQ GET_S1
0760   C299
```

```
0761   C299 81 39                  CMPA #'9'
0762   C29B 27 29                  BEQ END_OF_RECORD
0763   C29D
0764   C29D 20 EC                  BRA GET_S_RECORD1
0765   C29F
0766   C29F             GET_S1
0767   C29F
0768   C29F 86 00          LDAA #0
0769   C2A1 97 8A          STAA BCC
0770   C2A3
0771   C2A3 BD C1 F7       JSR GET_HEX
0772   C2A6 97 83                  STAA REG_C     ; GET NUMBER OF pairs
0773   C2A8
0774   C2A8 BD C2 0F       JSR ADD_BCC
0775   C2AB
0776   C2AB BD C1 F7       JSR GET_HEX
0777   C2AE 97 84                  STAA HL
0778   C2B0
0779   C2B0 7A 00 83               DEC REG_C
0780   C2B3
0781   C2B3 BD C2 0F       JSR ADD_BCC
0782   C2B6
0783   C2B6 BD C1 F7       JSR GET_HEX
0784   C2B9 97 85                  STAA HL+1        ; GET LOAD ADDRESS
0785   C2BB
0786   C2BB 7A 00 83               DEC REG_C
0787   C2BE
0788   C2BE BD C2 0F       JSR ADD_BCC
0789   C2C1
0790   C2C1 7A 00 83               DEC REG_C        ; BYTE COUNT-1 FOR BCC
0791   C2C4
0792   C2C4 20 14                  BRA DATA_S_RECORD
0793   C2C6
0794   C2C6             END_OF_RECORD
0795   C2C6
0796   C2C6 BD C0 BB               JSR CIN
0797   C2C9 81 0D                  CMPA #0DH         ; END OF LINE 0D,0A
0798   C2CB 26 F9                  BNE END_OF_RECORD
0799   C2CD
0800   C2CD B7 80 00       STAA GPIO1
0801   C2D0
0802   C2D0 96 89          LDAA _ERROR
0803   C2D2 81 01                  CMPA #1
0804   C2D4 26 03                  BNE NOERROR2
0805   C2D6
0806   C2D6             ; SHOW ERROR ON LED
0807   C2D6             ;  JSR OUT_OFF_RANGE
0808   C2D6
0809   C2D6 B7 80 00       STAA GPIO1
0810   C2D9
0811   C2D9             NOERROR2
0812   C2D9 39                     RTS
0813   C2DA
0814   C2DA
0815   C2DA             DATA_S_RECORD
0816   C2DA
0817   C2DA BD C1 F7       JSR GET_HEX
0818   C2DD DE 84                  LDX  HL
0819   C2DF A7 00                  STAA 0,X  ; WRITE TO MEMORY
0820   C2E1
0821   C2E1 BD C2 0F       JSR ADD_BCC
0822   C2E4
0823   C2E4 B7 80 00       STAA GPIO1
0824   C2E7
0825   C2E7 08                     INX
0826   C2E8 DF 84                  STX HL
0827   C2EA
0828   C2EA 7A 00 83       DEC REG_C
0829   C2ED 26 EB                  BNE DATA_S_RECORD ; UNTIL C=0
0830   C2EF
0831   C2EF 96 8A                  LDAA BCC
0832   C2F1 88 FF                  EORA #0FFH    ; ONE'S COMPLEMENT
0833   C2F3 97 8A          STAA BCC
0834   C2F5
0835   C2F5
0836   C2F5 BD C1 F7       JSR GET_HEX    ; GET BYTE CHECK SUM
```

```
0837   C2F8
0838   C2F8 91 8A                  CMPA BCC      ; COMPARE WITH BYTE CHECK SUM
0839   C2FA 27 04                  BEQ SKIP21
0840   C2FC
0841   C2FC 86 01             LDAA #1
0842   C2FE 97 89             STAA _ERROR    ; ERROR FLAG =1
0843   C300
0844   C300
0845   C300              SKIP21
0846   C300
0847   C300 7E C2 8B              JMP GET_S_RECORD1     ; NEXT LINE
0848   C303
0849   C303
0850   C303
0851   C303
0852   C303
0853   C303
0854   C303
0855   C303
0856   C303
0857   C303
0858   C303
0859   C303
0860   C303
0861   C303
0862   C303
0863   C303
0864   C303
0865   C303
0866   C303
0867   C303
0868   C303
0869   C303
0870   C303
0871   C303
0872   C303
0873   C303
0874   C303
0875   C303
0876   C303
0877   C303
0878   C303
0879   C303
0880   C303
0881   C303
0882   C303
0883   C303
0884   C303
0885   C303
0886   C303
0887   C303
0888   C303              ;*******************************************************************
0889   C303              ;
0890   C303              ; EXECUTE FUNCTIONS OR HEX KEY ENTERED
0891   C303              ; CHECK HEX KEY OR FUNCTIONS KEY
0892   C303              ; ENTRY: A
0893   C303
0894   C303 81 10        KEYEXE     CMPA #10H
0895   C305 2C 56                   BGE FUNCTION_KEY
0896   C307
0897   C307              ;HHHHHHHHHHHHHHH   KEY HEX ENTERED HHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHH
0898   C307
0899   C307 97 83                   STAA REG_C     ; SAVE HEX KEY
0900   C309 96 94                   LDAA STATE
0901   C30B
0902   C30B 81 01                   CMPA #1
0903   C30D 26 05                   BNE CHK_STATE2
0904   C30F
0905   C30F 96 83             LDAA REG_C
0906   C311 7E C7 15          JMP HEX_ADDR
0907   C314
0908   C314 81 02        CHK_STATE2 CMPA #2
0909   C316 26 03                   BNE CHK_STATE3
0910   C318 7E C7 4B          JMP HEX_DATA
0911   C31B
0912   C31B 81 03        CHK_STATE3 CMPA #3
```

```
0913   C31D 26 03                 BNE CHK_STATE5
0914   C31F 7E C7 6B          JMP HEX_REG
0915   C322
0916   C322 81 05         CHK_STATE5 CMPA #5
0917   C324 26 03                 BNE CHK_STATE6
0918   C326 7E C8 9C          JMP HEX_REL
0919   C329
0920   C329 81 06         CHK_STATE6 CMPA #6
0921   C32B 26 03                 BNE CHK_STATE7
0922   C32D 7E C8 A8          JMP HEX_REL6
0923   C330
0924   C330 81 07         CHK_STATE7 CMPA #7
0925   C332 26 03                 BNE CHK_STATE8
0926   C334 7E C8 D8              JMP HEX_SEND_FILE
0927   C337
0928   C337 81 08         CHK_STATE8  CMPA #8
0929   C339 26 03                  BNE CHK_STATE9
0930   C33B 7E C8 D9           JMP HEX_SEND_FILE2
0931   C33E
0932   C33E 81 1E         CHK_STATE9  CMPA #30
0933   C340 26 03                  BNE CHK_STATE10
0934   C342 7E C8 B4               JMP HEX_COPY31
0935   C345
0936   C345 81 1F         CHK_STATE10 CMPA #31
0937   C347 26 03                  BNE CHK_STATE11
0938   C349 7E C8 C0               JMP HEX_COPY32
0939   C34C
0940   C34C 81 20         CHK_STATE11 CMPA #32
0941   C34E 26 03                  BNE CHK_STATE12
0942   C350 7E C8 CC               JMP HEX_COPY33
0943   C353
0944   C353
0945   C353               CHK_STATE12
0946   C353
0947   C353
0948   C353 96 83                  LDAA REG_C
0949   C355 B7 80 00               STAA GPIO1
0950   C358 86 01                  LDAA #1         ; INVALID KEY PRESSED
0951   C35A 97 92            STAA INVALID
0952   C35C
0953   C35C
0954   C35C
0955   C35C
0956   C35C
0957   C35C               ; HEX KEY WAS PRESSED
0958   C35C
0959   C35C
0960   C35C 39               RTS
0961   C35D
0962   C35D
0963   C35D
0964   C35D
0965   C35D
0966   C35D
0967   C35D
0968   C35D
0969   C35D
0970   C35D
0971   C35D
0972   C35D
0973   C35D
0974   C35D
0975   C35D
0976   C35D
0977   C35D
0978   C35D
0979   C35D
0980   C35D
0981   C35D
0982   C35D               ;FFFFFFFFFFFFFFFFFFFFF FUNCTION KEY FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
0983   C35D
0984   C35D               FUNCTION_KEY
0985   C35D
0986   C35D 81 19                 CMPA #19H    ; KEY ADDR
0987   C35F 26 03           BNE CHK_FUNC1
0988   C361 7E C3 CE        JMP KEY_ADDR
```

```
0989   C364
0990   C364 81 14       CHK_FUNC1 CMPA #14H   ; KEY DATA
0991   C366 26 03                 BNE CHK_FUNC2
0992   C368 7E C3 FE       JMP KEY_DATA
0993   C36B
0994   C36B
0995   C36B
0996   C36B 81 10       CHK_FUNC2 CMPA #10H   ; KEY +
0997   C36D 26 03                 BNE CHK_FUNC3
0998   C36F 7E C4 70       JMP KEY_INC
0999   C372
1000   C372 81 11       CHK_FUNC3 CMPA #11H    ; KEY -
1001   C374 26 03                 BNE CHK_FUNC4
1002   C376 7E C5 0A       JMP KEY_DEC
1003   C379
1004   C379 81 18       CHK_FUNC4  CMPA #18H
1005   C37B 26 03                  BNE CHK_FUNC5
1006   C37D 7E C4 41        JMP KEY_PC
1007   C380
1008   C380 81 1B       CHK_FUNC5  CMPA #1BH
1009   C382 26 03                  BNE CHK_FUNC6
1010   C384 7E C4 55        JMP KEY_REG
1011   C387
1012   C387 81 12       CHK_FUNC6  CMPA #12H
1013   C389 26 03                  BNE CHK_FUNC7
1014   C38B 7E C5 33        JMP KEY_GO
1015   C38E
1016   C38E 81 1D       CHK_FUNC7  CMPA #1DH
1017   C390 26 03                  BNE CHK_FUNC8
1018   C392 7E C6 79        JMP KEY_REL
1019   C395
1020   C395 81 1F       CHK_FUNC8  CMPA #1FH
1021   C397 26 03                  BNE CHK_FUNC9
1022   C399 7E C6 8D        JMP KEY_DOWNLOAD_HEX
1023   C39C
1024   C39C 81 13       CHK_FUNC9 CMPA #13H
1025   C39E 26 03                 BNE CHK_FUNC10
1026   C3A0 7E C6 36       JMP KEY_STEP
1027   C3A3
1028   C3A3
1029   C3A3 81 16       CHK_FUNC10 CMPA #16H
1030   C3A5 26 03                  BNE CHK_FUNC11
1031   C3A7 7E C6 0D        JMP KEY_INS
1032   C3AA
1033   C3AA 81 17       CHK_FUNC11 CMPA #17H
1034   C3AC 26 03                  BNE CHK_FUNC12
1035   C3AE 7E C6 26        JMP KEY_DEL
1036   C3B1
1037   C3B1 81 15       CHK_FUNC12 CMPA #15H
1038   C3B3 26 03                  BNE CHK_FUNC13
1039   C3B5 7E C6 E3               JMP KEY_DEMO   ; CHANGE SBR KEY TO DEMO
1040   C3B8
1041   C3B8
1042   C3B8 81 1A       CHK_FUNC13 CMPA #1AH
1043   C3BA 26 03                  BNE CHK_FUNC14
1044   C3BC 7E C6 FD               JMP KEY_MUTE   ; TURN BEEP ON/OFF
1045   C3BF
1046   C3BF 81 1C       CHK_FUNC14 CMPA #1CH      ;
1047   C3C1 26 03                  BNE CHK_FUNC15
1048   C3C3 7E C7 01               JMP KEY_COPY
1049   C3C6
1050   C3C6 81 1E       CHK_FUNC15 CMPA #1EH
1051   C3C8 26 03                  BNE CHK_FUNC16
1052   C3CA 7E C1 72               JMP KEY_DUMP
1053   C3CD
1054   C3CD             CHK_FUNC16
1055   C3CD
1056   C3CD
1057   C3CD 39                     RTS
1058   C3CE
1059   C3CE
1060   C3CE             ;-------------------------------------------------------
1061   C3CE 86 01       KEY_ADDR     LDAA #1
1062   C3D0 97 94                    STAA STATE     ; STAATE =1 FOR ADDRESS MODE
1063   C3D2
1064   C3D2 86 00            LDAA #0
```

```
1065   C3D4 97 95            STAA ZERO_FLAG
1066   C3D6
1067   C3D6             STILL_ADDRESS
1068   C3D6 BD C4 2E                JSR READ_MEMORY
1069   C3D9
1070   C3D9 96 91            LDAA BUFFER+5
1071   C3DB 8A 40                   ORAA #40H
1072   C3DD 97 91            STAA BUFFER+5
1073   C3DF
1074   C3DF 96 90            LDAA BUFFER+4
1075   C3E1 8A 40                   ORAA #40H
1076   C3E3 97 90            STAA BUFFER+4
1077   C3E5
1078   C3E5 96 8F            LDAA BUFFER+3
1079   C3E7 8A 40                   ORAA #40H
1080   C3E9 97 8F            STAA BUFFER+3
1081   C3EB
1082   C3EB 96 8E                   LDAA BUFFER+2
1083   C3ED 8A 40                   ORAA #40H
1084   C3EF 97 8E            STAA BUFFER+2
1085   C3F1
1086   C3F1 96 8D                   LDAA BUFFER+1
1087   C3F3 84 BF                   ANDA #~40H
1088   C3F5 97 8D            STAA BUFFER+1
1089   C3F7
1090   C3F7 96 8C            LDAA BUFFER
1091   C3F9 84 BF                   ANDA #~40H
1092   C3FB 97 8C            STAA BUFFER
1093   C3FD
1094   C3FD 39                      RTS
1095   C3FE
1096   C3FE             ;-------------------------------------------------------
1097   C3FE
1098   C3FE 86 02       KEY_DATA     LDAA #2
1099   C400 97 94                    STAA STATE     ; STATE =2 FOR DATA MODE
1100   C402
1101   C402 86 00             LDAA #0
1102   C404 97 95            STAA ZERO_FLAG
1103   C406
1104   C406 BD C4 2E    STILL_DATA   JSR READ_MEMORY
1105   C409
1106   C409 96 91            LDAA BUFFER+5
1107   C40B 84 BF                   ANDA #~40H
1108   C40D 97 91            STAA BUFFER+5
1109   C40F
1110   C40F 96 90            LDAA BUFFER+4
1111   C411 84 BF                   ANDA #~40H
1112   C413 97 90            STAA BUFFER+4
1113   C415
1114   C415 96 8F            LDAA BUFFER+3
1115   C417 84 BF                   ANDA #~40H
1116   C419 97 8F            STAA BUFFER+3
1117   C41B
1118   C41B 96 8E                   LDAA BUFFER+2
1119   C41D 84 BF                   ANDA #~40H
1120   C41F 97 8E            STAA BUFFER+2
1121   C421
1122   C421 96 8D                   LDAA BUFFER+1
1123   C423 8A 40                   ORAA #40H
1124   C425 97 8D            STAA BUFFER+1
1125   C427
1126   C427 96 8C            LDAA BUFFER
1127   C429 8A 40                   ORAA #40H
1128   C42B 97 8C            STAA BUFFER
1129   C42D
1130   C42D 39                      RTS
1131   C42E
1132   C42E
1133   C42E             ; READ MEMORY
1134   C42E
1135   C42E             READ_MEMORY
1136   C42E
1137   C42E 96 96              LDAA DISPLAY
1138   C430 97 84       STAA HL
1139   C432 96 97       LDAA DISPLAY+1
1140   C434 97 85       STAA HL+1
```

```
1141   C436 BD C1 D1    JSR ADDRESS_DISPLAY
1142   C439
1143   C439 DE 84              LDX   HL
1144   C43B A6 00              LDAA 0,X
1145   C43D
1146   C43D             ;STAA GPIO1
1147   C43D
1148   C43D BD C1 C3    JSR DATA_DISPLAY
1149   C440 39          RTS
1150   C441
1151   C441
1152   C441
1153   C441
1154   C441             ; KEY PC, SET CURRENT USER ADDRESS
1155   C441
1156   C441 86 02       KEY_PC      LDAA #2
1157   C443 97 94                   STAA STATE     ; STAATE =2 FOR DATA MODE
1158   C445
1159   C445 86 00           LDAA #0
1160   C447 97 95           STAA ZERO_FLAG
1161   C449
1162   C449 96 98                   LDAA USER_PC
1163   C44B 97 96           STAA DISPLAY
1164   C44D 96 99                   LDAA USER_PC+1
1165   C44F 97 97           STAA DISPLAY+1
1166   C451                        ; JSR READ_MEMORY
1167   C451 BD C4 06        JSR STILL_DATA
1168   C454 39              RTS
1169   C455
1170   C455             ; KEY REGISTER
1171   C455             ; SET STATE TO 3 FOR REGISTER INPUT WITH HEX KEY
1172   C455
1173   C455 86 03       KEY_REG    LDAA #3
1174   C457 97 94                  STAA STATE  ; STAATE = 3 FOR REGISTER DISPLAY
1175   C459
1176   C459 86 03          LDAA #3
1177   C45B 97 91          STAA BUFFER+5
1178   C45D 86 8F          LDAA #8FH
1179   C45F 97 90          STAA BUFFER+4
1180   C461 86 BE          LDAA #0BEH
1181   C463 97 8F          STAA BUFFER+3
1182   C465 86 00                 LDAA #0
1183   C467 97 8E          STAA BUFFER+2
1184   C469 86 00          LDAA #0
1185   C46B 97 8D          STAA BUFFER+1
1186   C46D 97 8C          STAA BUFFER
1187   C46F
1188   C46F 39             RTS
1189   C470
1190   C470             ; INCREMENT CURRENT ADDRESS BY ONE
1191   C470             ;
1192   C470
1193   C470 96 94       KEY_INC     LDAA STATE
1194   C472 81 14                   CMPA #20
1195   C474 26 11                   BNE SKIP_INC1
1196   C476
1197   C476 7C 00 C9                INC DEMO_NO
1198   C479 96 C9                   LDAA DEMO_NO
1199   C47B 81 0A                   CMPA #10
1200   C47D 2D 04                   BLT SKIP_INC2
1201   C47F 86 00                   LDAA #0
1202   C481 97 C9                   STAA DEMO_NO
1203   C483
1204   C483
1205   C483 BD C1 C3    SKIP_INC2   JSR DATA_DISPLAY
1206   C486 39                      RTS
1207   C487
1208   C487
1209   C487
1210   C487
1211   C487
1212   C487 96 94       SKIP_INC1   LDAA STATE
1213   C489 81 05                   CMPA #5
1214   C48B 27 4D           BEQ REL_KEY_PRESSED
1215   C48D
1216   C48D 81 07                   CMPA #7
```

```
1217   C48F 27 61           BEQ SEND_INC1
1218   C491
1219   C491
1220   C491 81 1E                  CMPA #30
1221   C493 27 2D                  BEQ COPY_KEY_PRESSED
1222   C495
1223   C495 81 1F                  CMPA #31
1224   C497 27 11                  BEQ COPY_KEY_PRESSED1
1225   C499
1226   C499
1227   C499
1228   C499             ; NORMAL INCREMENT CURRENT ADDRESS BY ONE
1229   C499
1230   C499 86 02                  LDAA #2
1231   C49B 97 94                  STAA STATE     ; STATE =2 FOR DATA MODE
1232   C49D
1233   C49D 86 00           LDAA #0
1234   C49F 97 95           STAA ZERO_FLAG
1235   C4A1
1236   C4A1
1237   C4A1 DE 96                   LDX DISPLAY
1238   C4A3 08                      INX
1239   C4A4 DF 96                   STX DISPLAY
1240   C4A6
1241   C4A6
1242   C4A6                        ; JSR READ_MEMORY
1243   C4A6 BD C4 06         JSR STILL_DATA
1244   C4A9 39               RTS
1245   C4AA
1246   C4AA
1247   C4AA
1248   C4AA             COPY_KEY_PRESSED1
1249   C4AA
1250   C4AA                         ; Save ENDING address
1251   C4AA
1252   C4AA
1253   C4AA DE 96                   LDX DISPLAY
1254   C4AC DF A9                   STX END_ADDRESS
1255   C4AE
1256   C4AE 86 20                   LDAA #32           ; STATE 32 FOR KEY GO COPYING MEMORY
1257   C4B0 97 94                   STAA STATE
1258   C4B2 86 00                   LDAA #0
1259   C4B4 97 95            STAA ZERO_FLAG
1260   C4B6
1261   C4B6 BD C3 D6         JSR STILL_ADDRESS
1262   C4B9 86 B3                   LDAA #0B3H
1263   C4BB 97 8C            STAA BUFFER
1264   C4BD 86 02            LDAA #2
1265   C4BF 97 8D            STAA BUFFER+1
1266   C4C1 39               RTS
1267   C4C2
1268   C4C2
1269   C4C2             COPY_KEY_PRESSED
1270   C4C2
1271   C4C2                  ; Save start address
1272   C4C2
1273   C4C2
1274   C4C2 DE 96                   LDX DISPLAY
1275   C4C4 DF A3                   STX START_ADDRESS
1276   C4C6
1277   C4C6 86 1F                   LDAA #31
1278   C4C8 97 94                   STAA STATE
1279   C4CA 86 00                   LDAA #0
1280   C4CC 97 95            STAA ZERO_FLAG
1281   C4CE
1282   C4CE BD C3 D6         JSR STILL_ADDRESS
1283   C4D1 86 8F                   LDAA #08FH
1284   C4D3 97 8C            STAA BUFFER
1285   C4D5 86 02            LDAA #2
1286   C4D7 97 8D            STAA BUFFER+1
1287   C4D9 39               RTS
1288   C4DA
1289   C4DA
1290   C4DA
1291   C4DA             REL_KEY_PRESSED
1292   C4DA
```

```
1293   C4DA                  ; Save start address
1294   C4DA
1295   C4DA
1296   C4DA DE 96                    LDX DISPLAY
1297   C4DC DF A3                    STX START_ADDRESS
1298   C4DE
1299   C4DE 86 06                    LDAA #6
1300   C4E0 97 94                    STAA STATE
1301   C4E2 86 00                    LDAA #0
1302   C4E4 97 95             STAA ZERO_FLAG
1303   C4E6
1304   C4E6 BD C3 D6          JSR STILL_ADDRESS
1305   C4E9 86 B3             LDAA #0B3H
1306   C4EB 97 8C             STAA BUFFER
1307   C4ED 86 02             LDAA #2
1308   C4EF 97 8D             STAA BUFFER+1
1309   C4F1 39                RTS
1310   C4F2
1311   C4F2
1312   C4F2              SEND_INC1   ; Save start address
1313   C4F2
1314   C4F2
1315   C4F2 DE 96                    LDX DISPLAY
1316   C4F4 DF A3                    STX START_ADDRESS
1317   C4F6
1318   C4F6 86 08                    LDAA #8
1319   C4F8 97 94                    STAA STATE
1320   C4FA 86 00                    LDAA #0
1321   C4FC 97 95             STAA ZERO_FLAG
1322   C4FE
1323   C4FE BD C3 D6          JSR STILL_ADDRESS
1324   C501 86 8F             LDAA #08FH
1325   C503 97 8C             STAA BUFFER
1326   C505 86 02             LDAA #2
1327   C507 97 8D             STAA BUFFER+1
1328   C509 39                RTS
1329   C50A
1330   C50A
1331   C50A
1332   C50A
1333   C50A
1334   C50A
1335   C50A
1336   C50A              ; DECREMENT CURRENT ADDRESS BY ONE
1337   C50A              ;
1338   C50A
1339   C50A 96 94        KEY_DEC     LDAA STATE
1340   C50C 81 14                    CMPA #20
1341   C50E 26 12                    BNE SKIP_DEC1
1342   C510
1343   C510 7A 00 C9                 DEC DEMO_NO
1344   C513 96 C9                    LDAA DEMO_NO
1345   C515 97 C9                    STAA DEMO_NO
1346   C517 81 00                    CMPA #0
1347   C519 2C 03                    BGE SKIP_DEC2
1348   C51B 4F                       CLRA
1349   C51C 97 C9                    STAA DEMO_NO
1350   C51E
1351   C51E BD C1 C3     SKIP_DEC2   JSR DATA_DISPLAY
1352   C521
1353   C521
1354   C521 39                       RTS
1355   C522
1356   C522
1357   C522              SKIP_DEC1
1358   C522
1359   C522
1360   C522
1361   C522 86 02                    LDAA #2
1362   C524 97 94                    STAA STATE     ; STAATE =2 FOR DATA MODE
1363   C526
1364   C526 86 00            LDAA #0
1365   C528 97 95            STAA ZERO_FLAG
1366   C52A
1367   C52A
1368   C52A DE 96                    LDX DISPLAY
```

```
1369   C52C 09                      DEX
1370   C52D DF 96                   STX DISPLAY
1371   C52F                        ;  JSR READ_MEMORY
1372   C52F BD C4 06          JSR STILL_DATA
1373   C532 39                RTS
1374   C533
1375   C533
1376   C533              ; KEY GO WRITE USER REGISTERS TO STACK AND USE RTI TO JUMP TO USER PRO
1377   C533              ;
1378   C533
1379   C533              KEY_GO
1380   C533 96 94               LDAA STATE
1381   C535 81 20               CMPA #32
1382   C537 27 38               BEQ GO_STATE32   ; GO FOR COPYING DATA
1383   C539
1384   C539
1385   C539 81 06               CMPA #6
1386   C53B 27 51               BEQ GO_STATE6
1387   C53D
1388   C53D 81 08               CMPA #8
1389   C53F 27 34               BEQ GO_STATE8
1390   C541
1391   C541 81 0A               CMPA #10
1392   C543 27 31               BEQ SHORT_GO_STATE10
1393   C545
1394   C545 81 14               CMPA #20
1395   C547 26 31               BNE SKIP_GO1
1396   C549
1397   C549              ; EXECUTE KEY GO STATE 20 DEMO PROGRAMS
1398   C549
1399   C549 96 C9               LDAA DEMO_NO      ; CHECK SUB FUNCTION FOR STATE=20
1400   C54B 81 04               CMPA #4
1401   C54D 26 04               BNE DEMO2
1402   C54F BD CA 81            JSR DEMO_LCD
1403   C552 39                  RTS
1404   C553
1405   C553 81 01        DEMO2  CMPA #1
1406   C555 26 04               BNE DEMO3
1407   C557 BD CA 98            JSR DEMO_LED
1408   C55A 39                  RTS
1409   C55B
1410   C55B 81 00        DEMO3  CMPA #0
1411   C55D 26 03               BNE DEMO4
1412   C55F 7E CA AA            JMP RUN_DOT
1413   C562
1414   C562 81 02        DEMO4  CMPA #2
1415   C564 26 03               BNE DEMO5
1416   C566 7E CA B4            JMP BCD_COUNTING
1417   C569
1418   C569 81 03        DEMO5  CMPA #3
1419   C56B 26 03               BNE DEMO6
1420   C56D 7E CA F7            JMP CLOCK_PGM
1421   C570
1422   C570              DEMO6
1423   C570
1424   C570
1425   C570
1426   C570 39                  RTS
1427   C571
1428   C571
1429   C571 7E C5 E3     GO_STATE32  JMP GO_COPY_MEMORY
1430   C574
1431   C574 39                       RTS
1432   C575
1433   C575
1434   C575 39           GO_STATE8   RTS
1435   C576
1436   C576
1437   C576
1438   C576
1439   C576              SHORT_GO_STATE10
1440   C576
1441   C576 BD C6 B9                 JSR GO_STATE10
1442   C579
1443   C579 39                       RTS
1444   C57A
```

```
1445   C57A
1446   C57A
1447   C57A
1448   C57A             SKIP_GO1
1449   C57A
1450   C57A
1451   C57A
1452   C57A
1453   C57A
1454   C57A
1455   C57A 9F A1               STS SAVE_SP   ; SAVE SYSTEM STACK
1456   C57C
1457   C57C             ; NOW SWITCH TO USER STACK
1458   C57C
1459   C57C 9E 9E               LDS USER_SP   ; NOW LOAD WITH USER STACK
1460   C57E
1461   C57E 96 97               LDAA DISPLAY+1
1462   C580 36                  PSHA
1463   C581 96 96               LDAA DISPLAY
1464   C583 36                  PSHA
1465   C584 96 A0       LDAA USER_P
1466   C586 06                  TAP
1467   C587 DE 9C               LDX USER_IX
1468   C589 D6 9B               LDAB USER_B
1469   C58B 96 9A       LDAA USER_A
1470   C58D 39                  RTS
1471   C58E
1472   C58E
1473   C58E
1474   C58E             ; KEY GO WITH RELATIVE CALCULATION
1475   C58E             ; FIND OFFSET BYTE
1476   C58E
1477   C58E             GO_STATE6
1478   C58E
1479   C58E
1480   C58E DE 96               LDX DISPLAY
1481   C590 DF A5               STX DESTINATION
1482   C592
1483   C592             ; NOW COMPUTE OFFSET_BYTE = DESTINATION - START_ADDRESS
1484   C592
1485   C592             ; THE REAL PC WILL BE NEXT INTSRUCTION ADDRESS (+2 FROM BRANCH INSTRUC
1486   C592
1487   C592
1488   C592 DE A3               LDX START_ADDRESS
1489   C594 08                  INX
1490   C595 08                  INX
1491   C596 DF B0               STX TEMP16        ; SAVE CURRENT PC TO TEMP16
1492   C598
1493   C598
1494   C598
1495   C598 96 A6               LDAA DESTINATION+1
1496   C59A 90 B1               SUBA TEMP16+1
1497   C59C 97 A8               STAA OFFSET_BYTE+1
1498   C59E
1499   C59E 96 A5               LDAA DESTINATION
1500   C5A0 92 B0               SBCA TEMP16
1501   C5A2 97 A7               STAA OFFSET_BYTE
1502   C5A4
1503   C5A4
1504   C5A4
1505   C5A4             ; CHECK IF THE OFFSET BYTE WAS BETWEEN -128 (FF80) TO +127 (007F)
1506   C5A4             ; IF BIT 7 OF THE OFFSET BYTE IS 0, THE HIGH BYTE MUST BE ZERO
1507   C5A4             ; IF BIT 7 OF THE OFFSET BYTE IS 1, THE HIGH BYTE MUST BE FF
1508   C5A4             ; OTHERWISE, THE OFFSET BYTE WAS OUT OF RANGE, SHOW ERROR THEN
1509   C5A4
1510   C5A4 96 A8               LDAA OFFSET_BYTE+1
1511   C5A6 84 80               ANDA #80H
1512   C5A8 27 09       BEQ CHK_OFFSET_HIGH
1513   C5AA
1514   C5AA             ; CHECK HIGH BYTE MUST BE FF (-1)
1515   C5AA
1516   C5AA 96 A7               LDAA OFFSET_BYTE
1517   C5AC 81 FF               CMPA #0FFH
1518   C5AE 26 18       BNE OUT_OFF_RANGE
1519   C5B0
1520   C5B0 7E C5 B7    JMP IN_RANGE
```

```
1521   C5B3
1522   C5B3                CHK_OFFSET_HIGH
1523   C5B3 96 A7                  LDAA OFFSET_BYTE
1524   C5B5 26 11          BNE OUT_OFF_RANGE
1525   C5B7
1526   C5B7                ; STORE OFFSET TO THE 2ND BYTE OF BRANCH INSTRUCTION
1527   C5B7
1528   C5B7 DE A3          IN_RANGE LDX START_ADDRESS
1529   C5B9 08                       INX
1530   C5BA
1531   C5BA 96 A8                  LDAA OFFSET_BYTE+1
1532   C5BC A7 00                  STAA 0,X
1533   C5BE
1534   C5BE DF 96                  STX DISPLAY
1535   C5C0
1536   C5C0 BD C4 06       JSR STILL_DATA
1537   C5C3
1538   C5C3 86 02          LDAA #2
1539   C5C5 97 94                  STAA STATE
1540   C5C7 39             RTS
1541   C5C8
1542   C5C8                OUT_OFF_RANGE
1543   C5C8
1544   C5C8 86 02                  LDAA #2
1545   C5CA 97 91          STAA BUFFER+5
1546   C5CC 86 8F          LDAA #8FH
1547   C5CE 97 90          STAA BUFFER+4
1548   C5D0 86 03          LDAA #3
1549   C5D2 97 8F          STAA BUFFER+3
1550   C5D4 86 03          LDAA #3
1551   C5D6 97 8E          STAA BUFFER+2
1552   C5D8 86 00          LDAA #0
1553   C5DA 97 8D          STAA BUFFER+1
1554   C5DC 97 8C          STAA BUFFER
1555   C5DE
1556   C5DE 86 02          LDAA #2
1557   C5E0 97 94                  STAA STATE
1558   C5E2
1559   C5E2 39             RTS
1560   C5E3
1561   C5E3
1562   C5E3
1563   C5E3
1564   C5E3
1565   C5E3                ; SERVICE KEY GO FOR COPY KEY
1566   C5E3
1567   C5E3                GO_COPY_MEMORY
1568   C5E3
1569   C5E3 DE 96                  LDX DISPLAY
1570   C5E5 DF A5                  STX DESTINATION
1571   C5E7
1572   C5E7                COPY1
1573   C5E7 DE A3                  LDX START_ADDRESS
1574   C5E9 A6 00                  LDAA 0,X
1575   C5EB DE A5                  LDX DESTINATION
1576   C5ED A7 00                  STAA 0,X
1577   C5EF
1578   C5EF
1579   C5EF DE A5                  LDX DESTINATION
1580   C5F1 08                     INX
1581   C5F2 DF A5                  STX DESTINATION
1582   C5F4
1583   C5F4
1584   C5F4 DE A3                  LDX START_ADDRESS
1585   C5F6 08                     INX
1586   C5F7 DF A3                  STX START_ADDRESS
1587   C5F9
1588   C5F9
1589   C5F9 9C A9                  CPX END_ADDRESS
1590   C5FB
1591   C5FB 26 EA                  BNE COPY1
1592   C5FD
1593   C5FD                ; STORE LAST BYTE AS WELL
1594   C5FD
1595   C5FD DE A3                  LDX START_ADDRESS
1596   C5FF A6 00                  LDAA 0,X
```

```
1597   C601 DE A5              LDX DESTINATION
1598   C603 A7 00              STAA 0,X
1599   C605
1600   C605
1601   C605
1602   C605 BD C4 06     JSR STILL_DATA
1603   C608
1604   C608 86 02        LDAA #2
1605   C60A 97 94              STAA STATE
1606   C60C 39           RTS
1607   C60D
1608   C60D
1609   C60D
1610   C60D
1611   C60D
1612   C60D
1613   C60D
1614   C60D
1615   C60D
1616   C60D
1617   C60D
1618   C60D
1619   C60D
1620   C60D
1621   C60D
1622   C60D
1623   C60D
1624   C60D              ; INSERT ONE BYTE WITHIN 256 BYTES
1625   C60D              ; CURRENT LOCATION WILL BE DISPLAY+1
1626   C60D
1627   C60D
1628   C60D DE 96        KEY_INS  LDX DISPLAY ; GET CURRENT DISPLAY ADDRESS
1629   C60F 08                    INX
1630   C610 DF 96                 STX DISPLAY
1631   C612
1632   C612 C6 00                 LDAB #0
1633   C614
1634   C614 A6 FE        INSERT   LDAA 0FEH,X
1635   C616 A7 FF                 STAA 0FFH,X
1636   C618 09                    DEX
1637   C619
1638   C619 5A                    DECB
1639   C61A 26 F8                 BNE INSERT
1640   C61C
1641   C61C
1642   C61C 86 00                 LDAA #0
1643   C61E
1644   C61E DE 96                 LDX DISPLAY
1645   C620 A7 00                 STAA 0,X
1646   C622
1647   C622 BD C4 06              JSR STILL_DATA
1648   C625
1649   C625 39                    RTS
1650   C626
1651   C626              ; DELETE ONE BYTE AT CURRENT DISPLAY ADDRESS
1652   C626              ; SHIFTED UP 256 BYTES
1653   C626
1654   C626 DE 96        KEY_DEL  LDX DISPLAY ; GET CURRENT DISPLAY ADDRESS
1655   C628
1656   C628 C6 00                 LDAB #0
1657   C62A
1658   C62A A6 01        DELETE   LDAA 1,X
1659   C62C A7 00                 STAA 0,X
1660   C62E 08                    INX
1661   C62F
1662   C62F 5A                    DECB
1663   C630 26 F8                 BNE DELETE
1664   C632
1665   C632
1666   C632 BD C4 06              JSR STILL_DATA
1667   C635
1668   C635 39                    RTS
1669   C636
1670   C636
1671   C636              ; SINGLE STEP KEY
1672   C636              ; ENABLE BREAK SIGNAL SO AT THE 8TH OF E CLOCK, THE NMI WILL PRODUCE
```

```
1673   C636
1674   C636             KEY_STEP
1675   C636
1676   C636 9F A1               STS SAVE_SP   ; SAVE SYSTEM STACK
1677   C638
1678   C638             ; NOW SWITCH TO USER STACK
1679   C638
1680   C638 9E 9E               LDS USER_SP   ; NOW LOAD WITH USER STACK
1681   C63A
1682   C63A 96 97               LDAA DISPLAY+1
1683   C63C 36                  PSHA
1684   C63D 96 96               LDAA DISPLAY  ; PUSH PC
1685   C63F 36                  PSHA
1686   C640
1687   C640 96 9D               LDAA USER_IX+1
1688   C642 36                  PSHA
1689   C643                                   ; PUSH IX
1690   C643 96 9C               LDAA USER_IX
1691   C645 36                  PSHA
1692   C646
1693   C646 96 9A               LDAA USER_A    ; PUSH A
1694   C648 36                  PSHA
1695   C649
1696   C649 96 9B               LDAA USER_B    ; PUSH B
1697   C64B 36                  PSHA
1698   C64C
1699   C64C 96 A0               LDAA USER_P   ; PUSH CONDITION CODE REGISTER
1700   C64E 36                  PSHA
1701   C64F
1702   C64F C6 FF               LDAB #$FF
1703   C651 F7 80 02            STAB PORT1    ; NOW ENABLE 74LS164
1704   C654
1705   C654
1706   C654 3B                  RTI           ; 10 CYCLES
1707   C655
1708   C655
1709   C655
1710   C655
1711   C655             NMI_SERVICE
1712   C655
1713   C655                    ; STAA GPIO1
1714   C655
1715   C655 86 BF               LDAA #0BFH
1716   C657 B7 80 02            STAA PORT1    ; STOP BREAK SIGNAL
1717   C65A
1718   C65A 20 00               BRA  SWI_SERVICE
1719   C65C
1720   C65C                    ; RTI
1721   C65C
1722   C65C
1723   C65C             ; SAVE CPU REGISTERS TO USER REGISTERS
1724   C65C             ;  SWI INSTRUCTION 3FH
1725   C65C
1726   C65C
1727   C65C             SWI_SERVICE
1728   C65C
1729   C65C                  ;   POP CC
1730   C65C                  ;   POP ACCB
1731   C65C                  ;   POP ACCA
1732   C65C                  ;   POP IXH
1733   C65C                  ;   POP IXL
1734   C65C                  ;   POP PCH
1735   C65C                  ;   POP PCL
1736   C65C
1737   C65C 32                  PULA
1738   C65D 97 A0               STAA USER_P  ; SAVE CONDITION CODE REGISTER
1739   C65F
1740   C65F 32                  PULA
1741   C660 97 9B               STAA USER_B
1742   C662
1743   C662 32                  PULA
1744   C663 97 9A               STAA USER_A
1745   C665
1746   C665 32                  PULA
1747   C666 97 9C               STAA USER_IX
1748   C668
```

```
1749   C668 32                 PULA
1750   C669 97 9D              STAA USER_IX+1
1751   C66B 32                 PULA
1752   C66C
1753   C66C 97 98              STAA USER_PC
1754   C66E 32                 PULA
1755   C66F 97 99              STAA USER_PC+1
1756   C671
1757   C671
1758   C671
1759   C671 9F 9E              STS USER_SP
1760   C673
1761   C673 9E A1              LDS SAVE_SP    ; RESTORE SYSTEM STACK AND GET BACK TO MONITOR
1762   C675
1763   C675 BD C4 41           JSR KEY_PC     ; DISPLAY CURRENT PC
1764   C678
1765   C678 39                 RTS
1766   C679
1767   C679             ;---------------------------------------------------------------
1768   C679
1769   C679 86 05       KEY_REL      LDAA #5
1770   C67B 97 94                    STAA STATE ; STATE = 5 FOR RELATIVE BYTE CALCULATION
1771   C67D
1772   C67D 86 00                    LDAA #0
1773   C67F 97 95            STAA ZERO_FLAG
1774   C681 BD C3 D6         JSR STILL_ADDRESS
1775   C684 86 AE            LDAA #0AEH
1776   C686 97 8C            STAA BUFFER
1777   C688 86 02            LDAA #2
1778   C68A 97 8D            STAA BUFFER+1
1779   C68C 39               RTS
1780   C68D
1781   C68D
1782   C68D
1783   C68D             KEY_DOWNLOAD_HEX
1784   C68D
1785   C68D 86 B3             LDAA #0B3H    ; PRINT LOAD
1786   C68F 97 91             STAA BUFFER+5
1787   C691 86 85             LDAA #85H
1788   C693 97 91             STAA BUFFER+5
1789   C695 86 A3             LDAA #0A3H
1790   C697 97 90             STAA BUFFER+4
1791   C699 86 3F             LDAA #3FH
1792   C69B 97 8F             STAA BUFFER+3
1793   C69D 86 B3             LDAA #0B3H
1794   C69F 97 8E             STAA BUFFER+2
1795   C6A1 86 00             LDAA #0
1796   C6A3 97 8D             STAA BUFFER+1
1797   C6A5 97 8C             STAA BUFFER
1798   C6A7
1799   C6A7 86 0A                    LDAA #10
1800   C6A9 97 94                    STAA STATE
1801   C6AB
1802   C6AB BD C9 FF                 JSR CLR_SCREEN
1803   C6AE CE CB A7                 LDX #DOWNLOAD
1804   C6B1 BD CA 14                 JSR PSTR
1805   C6B4
1806   C6B4 86 FF                    LDAA #$FF
1807   C6B6 97 CA                    STAA MUTE   ; TURN OFF BEEP
1808   C6B8
1809   C6B8 39               RTS
1810   C6B9
1811   C6B9
1812   C6B9 86 01       GO_STATE10   LDAA #1
1813   C6BB B7 80 00         STAA GPIO1
1814   C6BE
1815   C6BE CE CB B6                 LDX #DOWNLOAD2
1816   C6C1 BD CA 14                 JSR PSTR
1817   C6C4
1818   C6C4
1819   C6C4 BD C2 14                 JSR GET_RECORD  ; GET INTEL HEX FILE
1820   C6C7
1821   C6C7 96 89                    LDAA _ERROR
1822   C6C9 81 00                    CMPA #0
1823   C6CB 26 08                    BNE FOUND_ERROR
1824   C6CD
```

```
1825   C6CD CE CB D0                LDX #COMPLETE
1826   C6D0 BD CA 14                JSR PSTR
1827   C6D3 20 06                   BRA SKIP_ERROR
1828   C6D5
1829   C6D5 CE CB DD    FOUND_ERROR   LDX #ERROR_FOUND
1830   C6D8 BD CA 14                JSR PSTR
1831   C6DB
1832   C6DB             SKIP_ERROR
1833   C6DB
1834   C6DB 86 02             LDAA #2
1835   C6DD 97 94                    STAA STATE
1836   C6DF BD C4 06          JSR STILL_DATA
1837   C6E2
1838   C6E2 39                RTS
1839   C6E3
1840   C6E3
1841   C6E3
1842   C6E3 86 14       KEY_DEMO      LDAA #20   ; STATE 20 FOR KEY DEMO
1843   C6E5 97 94                     STAA STATE
1844   C6E7
1845   C6E7 86 1F                     LDAA #1FH    ; PRINT Prg
1846   C6E9 97 91             STAA BUFFER+5
1847   C6EB 86 03                     LDAA #03H
1848   C6ED 97 90                     STAA BUFFER+4
1849   C6EF
1850   C6EF 86 BE                     LDAA #0BEH
1851   C6F1 97 8F                     STAA BUFFER+3
1852   C6F3 86 00                     LDAA #0
1853   C6F5 97 8E                     STAA BUFFER+2
1854   C6F7
1855   C6F7 96 C9                     LDAA DEMO_NO
1856   C6F9 BD C1 C3                  JSR DATA_DISPLAY
1857   C6FC 39                        RTS
1858   C6FD
1859   C6FD
1860   C6FD
1861   C6FD
1862   C6FD
1863   C6FD 73 00 CA    KEY_MUTE      COM MUTE
1864   C700 39                        RTS
1865   C701
1866   C701
1867   C701
1868   C701 86 1E       KEY_COPY      LDAA #30
1869   C703 97 94                     STAA STATE    ; STATE = 30
1870   C705
1871   C705 86 00                     LDAA #0
1872   C707 97 95                     STAA ZERO_FLAG
1873   C709
1874   C709 BD C3 D6                  JSR STILL_ADDRESS
1875   C70C 86 AE                     LDAA #0AEH
1876   C70E 97 8C                     STAA BUFFER
1877   C710 86 02                     LDAA #2
1878   C712 97 8D                     STAA BUFFER+1
1879   C714 39                        RTS
1880   C715
1881   C715
1882   C715
1883   C715
1884   C715
1885   C715
1886   C715             ;------------------- HEX KEY FOR ADDRESS ------------------
1887   C715
1888   C715 96 95       HEX_ADDR     LDAA ZERO_FLAG
1889   C717 81 00                    CMPA #0
1890   C719 26 0A            BNE SHIFT_ADDRESS
1891   C71B
1892   C71B 86 01            LDAA #1
1893   C71D 97 95            STAA ZERO_FLAG
1894   C71F 86 00                    LDAA #0
1895   C721 97 96            STAA DISPLAY
1896   C723 97 97            STAA DISPLAY+1
1897   C725
1898   C725 0C          SHIFT_ADDRESS CLC
1899   C726 79 00 97                 ROL DISPLAY+1
1900   C729 79 00 96                 ROL DISPLAY
```

```
1901   C72C
1902   C72C 0C                      CLC
1903   C72D 79 00 97                ROL DISPLAY+1
1904   C730 79 00 96                ROL DISPLAY
1905   C733
1906   C733 0C               CLC
1907   C734 79 00 97                ROL DISPLAY+1
1908   C737 79 00 96                ROL DISPLAY
1909   C73A
1910   C73A 0C               CLC
1911   C73B 79 00 97                ROL DISPLAY+1
1912   C73E 79 00 96                ROL DISPLAY
1913   C741
1914   C741 96 97                   LDAA DISPLAY+1
1915   C743 9A 83                   ORAA REG_C
1916   C745 97 97                   STAA DISPLAY+1
1917   C747
1918   C747                        ; JSR READ_MEMORY
1919   C747
1920   C747 BD C3 D6         JSR STILL_ADDRESS
1921   C74A
1922   C74A 39               RTS
1923   C74B             ;------------------------- HEX KEY FOR DATA MODE --------------------
1924   C74B
1925   C74B
1926   C74B 96 95       HEX_DATA     LDAA ZERO_FLAG
1927   C74D 81 00                    CMPA #0
1928   C74F 26 0A            BNE SHIFT_DATA
1929   C751
1930   C751 86 01            LDAA #1
1931   C753 97 95            STAA ZERO_FLAG
1932   C755
1933   C755 86 00            LDAA #0
1934   C757 DE 96                    LDX  DISPLAY
1935   C759 A7 00                    STAA 0,X
1936   C75B
1937   C75B DE 96       SHIFT_DATA   LDX  DISPLAY
1938   C75D A6 00                    LDAA 0,X
1939   C75F 48                       ASLA
1940   C760 48                       ASLA
1941   C761 48                       ASLA
1942   C762 48                       ASLA
1943   C763 9A 83                    ORAA REG_C
1944   C765 A7 00                    STAA 0,X
1945   C767
1946   C767                    ; JSR READ_MEMORY
1947   C767 BD C4 06            JSR STILL_DATA
1948   C76A 39                  RTS
1949   C76B
1950   C76B
1951   C76B             ;************************ HEX REGSITERS DISPLAY **********************
1952   C76B
1953   C76B             ; DISPLAY USER REGSITERS
1954   C76B
1955   C76B 96 83       HEX_REG  LDAA REG_C
1956   C76D 81 00                CMPA #0
1957   C76F 26 16        BNE CHK_REG1
1958   C771
1959   C771 96 9A        LDAA USER_A
1960   C773             ; STAA GPIO1
1961   C773 BD C1 C3     JSR DATA_DISPLAY
1962   C776
1963   C776 86 3F                LDAA #3FH
1964   C778 97 8E        STAA BUFFER+2
1965   C77A 86 8D                LDAA #8DH        ; REGISTER ACCA
1966   C77C 97 8F        STAA BUFFER+3
1967   C77E 86 8D                LDAA #8DH
1968   C780 97 90        STAA BUFFER+4
1969   C782 86 3F                LDAA #3FH
1970   C784 97 91                STAA BUFFER+5
1971   C786 39           RTS
1972   C787
1973   C787             CHK_REG1
1974   C787 81 01                CMPA #1
1975   C789 26 16        BNE CHK_REG2
1976   C78B
```

```
1977   C78B 96 9B               LDAA USER_B
1978   C78D             ; STAA GPIO1
1979   C78D
1980   C78D BD C1 C3     JSR DATA_DISPLAY
1981   C790 86 A7               LDAA #0A7H
1982   C792 97 8E        STAA BUFFER+2
1983   C794 86 8D               LDAA #8DH         ; REGISTER ACCB
1984   C796 97 8F        STAA BUFFER+3
1985   C798 86 8D               LDAA #8DH
1986   C79A 97 90        STAA BUFFER+4
1987   C79C 86 3F               LDAA #3FH
1988   C79E 97 91               STAA BUFFER+5
1989   C7A0 39           RTS
1990   C7A1
1991   C7A1 81 02       CHK_REG2  CMPA #2
1992   C7A3 26 10        BNE CHK_REG3
1993   C7A5
1994   C7A5 DE 9C               LDX USER_IX
1995   C7A7             ; STAA GPIO1
1996   C7A7
1997   C7A7 DF 84               STX HL
1998   C7A9
1999   C7A9 BD C1 D1            JSR ADDRESS_DISPLAY
2000   C7AC
2001   C7AC 86 30               LDAA #30H
2002   C7AE 97 8D               STAA BUFFER+1
2003   C7B0 86 26               LDAA #26H        ; REGISTER IX
2004   C7B2 97 8C               STAA BUFFER
2005   C7B4 39           RTS
2006   C7B5
2007   C7B5
2008   C7B5 81 03       CHK_REG3 CMPA #3
2009   C7B7 26 10        BNE CHK_REG4
2010   C7B9
2011   C7B9 DE 9E               LDX USER_SP
2012   C7BB             ; STAA GPIO1
2013   C7BB DF 84               STX HL
2014   C7BD
2015   C7BD BD C1 D1            JSR ADDRESS_DISPLAY
2016   C7C0 86 AE               LDAA #0AEH
2017   C7C2 97 8D               STAA BUFFER+1
2018   C7C4 86 1F               LDAA #1FH        ; REGISTER SP
2019   C7C6 97 8C               STAA BUFFER
2020   C7C8 39           RTS
2021   C7C9
2022   C7C9
2023   C7C9
2024   C7C9 81 04       CHK_REG4  CMPA #4
2025   C7CB 26 2F               BNE CHK_REG5
2026   C7CD
2027   C7CD                                      ; CONDITION CODE DISPLAY FOR HIGH NIBBLE 11HI
2028   C7CD
2029   C7CD 96 A0               LDAA USER_P
2030   C7CF 84 10               ANDA #$10
2031   C7D1 26 06               BNE  SHOW_BIT4
2032   C7D3
2033   C7D3 86 BD               LDAA #ZERO
2034   C7D5 97 8E               STAA BUFFER+2
2035   C7D7 20 04               BRA  NEXT_BIT1
2036   C7D9
2037   C7D9 86 30       SHOW_BIT4 LDAA #ONE
2038   C7DB 97 8E               STAA BUFFER+2
2039   C7DD
2040   C7DD 96 A0       NEXT_BIT1 LDAA USER_P
2041   C7DF
2042   C7DF 84 20                ANDA #$20
2043   C7E1 26 06                BNE  SHOW_BIT5
2044   C7E3
2045   C7E3 86 BD                LDAA #ZERO
2046   C7E5 97 8F                STAA BUFFER+3
2047   C7E7 20 04                BRA NEXT_BIT2
2048   C7E9
2049   C7E9 86 30       SHOW_BIT5 LDAA #ONE
2050   C7EB 97 8F                STAA BUFFER+3
2051   C7ED
2052   C7ED 86 30       NEXT_BIT2 LDAA #ONE
```

```
2053   C7EF 97 90                STAA BUFFER+4
2054   C7F1 97 91                STAA BUFFER+5
2055   C7F3
2056   C7F3 86 8D                LDAA #8DH     ; STORE CH, CONDITION CODE HIGH NIBBLE
2057   C7F5 97 8D                STAA BUFFER+1
2058   C7F7 86 37                LDAA #37H
2059   C7F9 97 8C                STAA BUFFER
2060   C7FB
2061   C7FB
2062   C7FB 39                   RTS
2063   C7FC
2064   C7FC
2065   C7FC 81 05       CHK_REG5  CMPA #5
2066   C7FE 26 49                BNE CHK_REG6
2067   C800
2068   C800                                    ; CONDITION CODE DISPLAY FOR LOW NIBBLE NZVC
2069   C800
2070   C800 96 A0                LDAA USER_P
2071   C802 84 80                ANDA #$80
2072   C804 26 06                BNE  SHOW_BIT6
2073   C806
2074   C806 86 BD                LDAA #ZERO
2075   C808 97 91                STAA BUFFER+5
2076   C80A 20 04                BRA  NEXT_BIT3
2077   C80C
2078   C80C 86 30       SHOW_BIT6 LDAA #ONE
2079   C80E 97 91                STAA BUFFER+5
2080   C810
2081   C810 96 A0       NEXT_BIT3 LDAA USER_P
2082   C812 84 04                 ANDA #4
2083   C814 26 06                 BNE  SHOW_BIT7
2084   C816
2085   C816 86 BD                 LDAA #ZERO
2086   C818 97 90                 STAA BUFFER+4
2087   C81A 20 04                 BRA NEXT_BIT4
2088   C81C
2089   C81C 86 30       SHOW_BIT7 LDAA #ONE
2090   C81E 97 90                 STAA BUFFER+4
2091   C820
2092   C820 96 A0       NEXT_BIT4 LDAA USER_P
2093   C822 84 02                 ANDA #2
2094   C824 26 06                 BNE SHOW_BIT8
2095   C826
2096   C826 86 BD                 LDAA #ZERO
2097   C828 97 8F                 STAA BUFFER+3
2098   C82A 20 04                 BRA  NEXT_BIT5
2099   C82C
2100   C82C 86 30       SHOW_BIT8 LDAA #ONE
2101   C82E 97 8F                 STAA BUFFER+3
2102   C830
2103   C830 96 A0       NEXT_BIT5 LDAA USER_P
2104   C832 84 01                 ANDA #1
2105   C834 26 06                 BNE SHOW_BIT9
2106   C836
2107   C836 86 BD                 LDAA #ZERO
2108   C838 97 8E                 STAA BUFFER+2
2109   C83A 20 04                 BRA NEXT_BIT6
2110   C83C
2111   C83C 86 30       SHOW_BIT9 LDAA #ONE
2112   C83E 97 8E                 STAA BUFFER+2
2113   C840
2114   C840             NEXT_BIT6
2115   C840 86 8D                 LDAA #8DH
2116   C842 97 8D                 STAA BUFFER+1
2117   C844 86 85                 LDAA #85H
2118   C846 97 8C                 STAA BUFFER
2119   C848 39                    RTS
2120   C849
2121   C849
2122   C849 81 06       CHK_REG6  CMPA #6
2123   C84B 26 10                 BNE CHK_REG7
2124   C84D
2125   C84D CE 00 00              LDX #0
2126   C850 EE 00                 LDX 0,X
2127   C852 DF 84                 STX HL
2128   C854 BD C1 D1              JSR ADDRESS_DISPLAY
```

```
2129   C857
2130   C857 86 00               LDAA #0
2131   C859 BD C1 C3            JSR DATA_DISPLAY
2132   C85C 39                  RTS
2133   C85D
2134   C85D
2135   C85D 81 07       CHK_REG7  CMPA #7
2136   C85F 26 10                 BNE CHK_REG8
2137   C861
2138   C861 CE 00 02            LDX #2
2139   C864 EE 00               LDX 0,X
2140   C866 DF 84               STX HL
2141   C868 BD C1 D1            JSR ADDRESS_DISPLAY
2142   C86B
2143   C86B 86 02               LDAA #2
2144   C86D BD C1 C3            JSR DATA_DISPLAY
2145   C870 39                  RTS
2146   C871
2147   C871
2148   C871 81 08       CHK_REG8  CMPA #8
2149   C873 26 10                 BNE CHK_REG9
2150   C875
2151   C875 CE 00 04            LDX #4
2152   C878 EE 00               LDX 0,X
2153   C87A DF 84               STX HL
2154   C87C BD C1 D1            JSR ADDRESS_DISPLAY
2155   C87F
2156   C87F 86 04               LDAA #4
2157   C881 BD C1 C3            JSR DATA_DISPLAY
2158   C884 39                  RTS
2159   C885
2160   C885 81 09       CHK_REG9  CMPA #9
2161   C887 26 10                 BNE CHK_REGA
2162   C889
2163   C889 CE 00 06            LDX #6
2164   C88C EE 00               LDX 0,X
2165   C88E DF 84               STX HL
2166   C890 BD C1 D1            JSR ADDRESS_DISPLAY
2167   C893
2168   C893 86 06               LDAA #6
2169   C895 BD C1 C3            JSR DATA_DISPLAY
2170   C898 39                  RTS
2171   C899
2172   C899             CHK_REGA
2173   C899
2174   C899 39                  RTS
2175   C89A
2176   C89A
2177   C89A
2178   C89A
2179   C89A
2180   C89A
2181   C89A
2182   C89A
2183   C89A
2184   C89A
2185   C89A 39                  RTS
2186   C89B
2187   C89B
2188   C89B
2189   C89B
2190   C89B
2191   C89B
2192   C89B
2193   C89B
2194   C89B
2195   C89B
2196   C89B
2197   C89B
2198   C89B
2199   C89B
2200   C89B
2201   C89B
2202   C89B
2203   C89B 39                  RTS
2204   C89C
```

```
2205   C89C
2206   C89C             ;-------------------------------------------------------
2207   C89C
2208   C89C BD C7 15    HEX_REL     JSR HEX_ADDR
2209   C89F 86 AE                   LDAA #0AEH
2210   C8A1 97 8C           STAA BUFFER
2211   C8A3 86 02           LDAA #2
2212   C8A5 97 8D           STAA BUFFER+1
2213   C8A7 39              RTS
2214   C8A8
2215   C8A8
2216   C8A8
2217   C8A8 BD C7 15    HEX_REL6    JSR HEX_ADDR
2218   C8AB 86 B3                   LDAA #0B3H
2219   C8AD 97 8C           STAA BUFFER
2220   C8AF 86 02           LDAA #2
2221   C8B1 97 8D           STAA BUFFER+1
2222   C8B3 39              RTS
2223   C8B4
2224   C8B4
2225   C8B4             HEX_COPY31
2226   C8B4
2227   C8B4 BD C7 15                JSR HEX_ADDR
2228   C8B7 86 AE                   LDAA #0AEH
2229   C8B9 97 8C           STAA BUFFER
2230   C8BB 86 02           LDAA #2
2231   C8BD 97 8D           STAA BUFFER+1
2232   C8BF 39              RTS
2233   C8C0
2234   C8C0
2235   C8C0             HEX_COPY32
2236   C8C0
2237   C8C0 BD C7 15                JSR HEX_ADDR
2238   C8C3 86 8F                   LDAA #08FH
2239   C8C5 97 8C           STAA BUFFER
2240   C8C7 86 02           LDAA #2
2241   C8C9 97 8D           STAA BUFFER+1
2242   C8CB 39              RTS
2243   C8CC
2244   C8CC             HEX_COPY33
2245   C8CC
2246   C8CC BD C7 15                JSR HEX_ADDR
2247   C8CF 86 B3                   LDAA #0B3H
2248   C8D1 97 8C           STAA BUFFER
2249   C8D3 86 02           LDAA #2
2250   C8D5 97 8D           STAA BUFFER+1
2251   C8D7 39              RTS
2252   C8D8
2253   C8D8
2254   C8D8
2255   C8D8
2256   C8D8
2257   C8D8 39          HEX_SEND_FILE   RTS
2258   C8D9
2259   C8D9 39          HEX_SEND_FILE2  RTS
2260   C8DA
2261   C8DA
2262   C8DA
2263   C8DA
2264   C8DA
2265   C8DA
2266   C8DA
2267   C8DA             ; SCAN DISPLAY AND KEYBOARD
2268   C8DA             ; ENTRY: DISPLAY BUFFER IN PAGE 0
2269   C8DA             ; EXIT: KEY = -1 NO KEY PRESSED
2270   C8DA             ;       KEY >=0 KEY POSITION
2271   C8DA             ; REGSITERS USED: X,A,Y
2272   C8DA
2273   C8DA             SCAN1
2274   C8DA CE 00 8C            LDX #BUFFER
2275   C8DD
2276   C8DD 86 00       SCAN2   LDAA #0
2277   C8DF 97 83       STAA REG_C
2278   C8E1
2279   C8E1 86 FF       LDAA #-1
2280   C8E3 97 93       STAA KEY
```

```
2281   C8E5
2282   C8E5 86 01       LDAA #1
2283   C8E7 97 80       STAA REG_E
2284   C8E9
2285   C8E9 86 06       LDAA #6
2286   C8EB 97 84       STAA HL
2287   C8ED
2288   C8ED             ;to the active column.
2289   C8ED 96 80       KCOL    LDAA REG_E
2290   C8EF
2291   C8EF 88 FF               EORA #0FFH              ; COMPLEMENT IT
2292   C8F1
2293   C8F1 84 BF               ANDA #0BFH              ; MUST BE LOW FOR BREAK
2294   C8F3
2295   C8F3 B7 80 02    STAA DIGIT
2296   C8F6
2297   C8F6 A6 00               LDAA 0,X
2298   C8F8 B7 80 03    STAA SEG7
2299   C8FB
2300   C8FB                     ; automatic adjust for 1-3 segment display
2301   C8FB
2302   C8FB BD C9 34            JSR AUTOBRIGHTNESS
2303   C8FE
2304   C8FE             ;        LDAB #$30
2305   C8FE             ;DELAY3  DECB
2306   C8FE             ;        BNE DELAY3
2307   C8FE
2308   C8FE
2309   C8FE 86 00       LDAA #0                 ; TURN LED OFF
2310   C900 B7 80 03    STAA SEG7
2311   C903
2312   C903 C6 32               LDAB #50
2313   C905 5A          DELAY10 DECB
2314   C906 26 FD       BNE DELAY10
2315   C908
2316   C908
2317   C908 86 06       LDAA #6
2318   C90A 97 82       STAA REG_B
2319   C90C
2320   C90C B6 80 01    LDAA KIN
2321   C90F
2322   C90F 97 81       STAA  REG_D
2323   C911
2324   C911
2325   C911 74 00 81    KROW LSR  REG_D    ;Rotate D 1 bit right, bit 0
2326   C914                     ;of D will be rotated into
2327   C914 25 04       BCS NOKEY      ;carry flag.
2328   C916
2329   C916 96 83       LDAA REG_C
2330   C918 97 93       STAA KEY
2331   C91A
2332   C91A 7C 00 83    NOKEYINC REG_C   ;Increase current key-code by 1.
2333   C91D
2334   C91D 7A 00 82    DEC REG_B
2335   C920 26 EF       BNE KROW
2336   C922
2337   C922 08          INX
2338   C923
2339   C923 96 80       LDAA REG_E
2340   C925 48                  ASLA
2341   C926 97 80       STAA REG_E
2342   C928
2343   C928
2344   C928 7A 00 84    DEC  HL
2345   C92B 26 C0       BNE  KCOL
2346   C92D 39          RTS
2347   C92E
2348   C92E
2349   C92E C6 C8       DEBOUNCE LDAB #200
2350   C930 5A          DELAY4   DECB
2351   C931 26 FD         BNE DELAY4
2352   C933 39            RTS
2353   C934
2354   C934             ; ADJUST TIME ON IF EQUAL OR LESS THAN 3 BITS ON
2355   C934
2356   C934             AUTOBRIGHTNESS
```

```
2357   C934
2358   C934 5F                  CLRB
2359   C935 D7 CB               STAB BIT1_COUNTER
2360   C937
2361   C937 C6 08               LDAB #8
2362   C939
2363   C939 49          CHECK_BIT1 ROLA
2364   C93A 24 03                BCC _BIT1
2365   C93C 7C 00 CB            INC BIT1_COUNTER
2366   C93F
2367   C93F             _BIT1
2368   C93F 5A                  DECB
2369   C940 26 F7               BNE CHECK_BIT1
2370   C942
2371   C942 96 CB               LDAA BIT1_COUNTER
2372   C944
2373   C944 81 03               CMPA #3     ; DIMMING FOR <=4 BITS THAT ON
2374   C946 2E 06               BGT NORMAL
2375   C948
2376   C948 C6 03               LDAB #3
2377   C94A 5A          DELAY20  DECB
2378   C94B 26 FD               BNE DELAY20
2379   C94D 39                  RTS
2380   C94E
2381   C94E
2382   C94E             NORMAL
2383   C94E
2384   C94E
2385   C94E C6 30               LDAB #$30    ; tested with 20h 30
2386   C950 5A          DELAY3  DECB
2387   C951 26 FD               BNE DELAY3
2388   C953 39                  RTS
2389   C954
2390   C954
2391   C954
2392   C954
2393   C954
2394   C954
2395   C954
2396   C954
2397   C954
2398   C954             ;---------------------------------------------------------------------
2399   C954
2400   C954 BD C8 DA    SCANKEY JSR SCAN1
2401   C957 96 93               LDAA KEY
2402   C959 81 FF               CMPA #-1
2403   C95B 27 1D       BEQ KEY_RELEASED
2404   C95D
2405   C95D B6 80 01    LDAA PORT0
2406   C960 84 40               ANDA #40H
2407   C962 27 05               BEQ SKIP_DISPLAY5
2408   C964
2409   C964 BD C9 2E            JSR DEBOUNCE
2410   C967 20 EB               BRA SCANKEY
2411   C969
2412   C969             SKIP_DISPLAY5
2413   C969
2414   C969
2415   C969             ; IF REPEAT KEY WAS PRESSED, SLOW DOWN IT
2416   C969 86 14                 LDAA #20
2417   C96B 97 AD                 STAA REPDELAY
2418   C96D
2419   C96D BD C8 DA    DISPLAY4  JSR SCAN1
2420   C970 7A 00 AD      DEC REPDELAY
2421   C973 26 F8         BNE DISPLAY4
2422   C975
2423   C975
2424   C975 CE 00 00              LDX #0         ; THEN REPEAT KEY PRESS
2425   C978 DF 92                 STX INVALID    ; RESET INVALID FLAG
2426   C97A             KEY_RELEASED
2427   C97A
2428   C97A BD C9 2E    JSR DEBOUNCE
2429   C97D
2430   C97D             UNTIL_PRESS
2431   C97D
2432   C97D BD C8 DA    JSR SCAN1
```

```
2433   C980 96 93              LDAA KEY
2434   C982 81 FF              CMPA #-1
2435   C984 27 F7       BEQ UNTIL_PRESS
2436   C986
2437   C986 BD C9 2E    JSR DEBOUNCE
2438   C989
2439   C989 BD C8 DA    JSR SCAN1
2440   C98C
2441   C98C 96 93       LDAA KEY
2442   C98E CE CC 0B            LDX  #KEYTAB
2443   C991
2444   C991 DF B0               STX   TEMP16
2445   C993
2446   C993 9B B1               ADDA TEMP16+1
2447   C995 97 B1               STAA TEMP16+1
2448   C997
2449   C997 86 00               LDAA #0
2450   C999 99 B0               ADCA TEMP16
2451   C99B 97 B0               STAA TEMP16
2452   C99D
2453   C99D DE B0               LDX TEMP16
2454   C99F
2455   C99F A6 00               LDAA  0,X          ; OPEN TABLE
2456   C9A1
2457   C9A1                 ;    STAA GPIO1       ; TEST NOW A IS INTERNAL CODE
2458   C9A1
2459   C9A1 39                  RTS
2460   C9A2
2461   C9A2
2462   C9A2 BD C9 54    MAIN    JSR SCANKEY        ; scan display and keypad
2463   C9A5
2464   C9A5 BD C3 03            JSR KEYEXE         ; execute key that pressed
2465   C9A8
2466   C9A8 BD C9 AD            JSR BEEP           ; beep/no beep after key executed
2467   C9AB
2468   C9AB 20 F5               BRA MAIN           ; repeat forever
2469   C9AD
2470   C9AD
2471   C9AD             ;-------------------------------------------------------
2472   C9AD             ; BEEP WHEN KEY PRESSED
2473   C9AD             ;
2474   C9AD             ; CALIBRATED TO 523Hz
2475   C9AD             ; NEW FREQUENCY 673Hz, TESTED 670Hz
2476   C9AD
2477   C9AD B6 80 01    BEEP     LDAA PORT0
2478   C9B0 84 40               ANDA #40H
2479   C9B2 27 1D        BEQ NO_BEEP    ; CHECK IF REPEAT KEY IS PRESSED, THEN NO BEEP
2480   C9B4
2481   C9B4 96 CA               LDAA MUTE
2482   C9B6 81 00               CMPA #0
2483   C9B8 27 01               BEQ BEEPON
2484   C9BA 39                  RTS           ; IF MUTE=1 THEN NO BEEP
2485   C9BB
2486   C9BB
2487   C9BB
2488   C9BB CE 00 35    BEEPON   LDX #35h      ;    BEEP_PERIOD
2489   C9BE
2490   C9BE 86 3F       BEEP2    LDAA #3FH
2491   C9C0 B7 80 02     STAA PORT1
2492   C9C3 BD C9 D2     JSR BEEP_DELAY
2493   C9C6 86 BF        LDAA #0BFH
2494   C9C8 B7 80 02     STAA PORT1
2495   C9CB BD C9 D2     JSR BEEP_DELAY
2496   C9CE
2497   C9CE 09           DEX
2498   C9CF 26 ED        BNE BEEP2
2499   C9D1
2500   C9D1             NO_BEEP
2501   C9D1
2502   C9D1 39                  RTS
2503   C9D2
2504   C9D2
2505   C9D2
2506   C9D2
2507   C9D2
2508   C9D2 C6 52       BEEP_DELAY LDAB #52H  ;  #78H   ; #09BH CALIBRATED 524Hz
```

```
2509   C9D4 5A          BEEP_LOOP DECB
2510   C9D5 26 FD                 BNE BEEP_LOOP
2511   C9D7 39              RTS
2512   C9D8
2513   C9D8
2514   C9D8
2515   C9D8
2516   C9D8             ; TEST CODE
2517   C9D8
2518   C9D8
2519   C9D8 4C          TEST       INCA
2520   C9D9 B7 80 00              STAA GPIO1
2521   C9DC 8D 02                 BSR DELAY_LED
2522   C9DE 20 F8                 BRA TEST
2523   C9E0
2524   C9E0 CE 07 D0    DELAY_LED LDX #2000
2525   C9E3 09          DELAY_LED1 DEX
2526   C9E4 26 FD                 BNE DELAY_LED1
2527   C9E6 39                    RTS
2528   C9E7
2529   C9E7
2530   C9E7             ; TEST BEEP
2531   C9E7
2532   C9E7 86 3F       BEEP3    LDAA #3FH
2533   C9E9 B7 80 02     STAA PORT1
2534   C9EC 8D 0B                 BSR BEEP_DELAY1
2535   C9EE 86 BF        LDAA #0BFH
2536   C9F0 B7 80 02     STAA PORT1
2537   C9F3 8D 04                 BSR BEEP_DELAY1
2538   C9F5 26 F0                 BNE BEEP3
2539   C9F7 20 EE                 BRA BEEP3
2540   C9F9
2541   C9F9
2542   C9F9 C6 A0       BEEP_DELAY1 LDAB #0A0H      ;
2543   C9FB 5A          BEEP_LOOP1 DECB
2544   C9FC 26 FD                  BNE BEEP_LOOP1
2545   C9FE 39              RTS
2546   C9FF
2547   C9FF
2548   C9FF
2549   C9FF
2550   C9FF             ; CLEAR DISPLAY
2551   C9FF
2552   C9FF CE 00 32    CLR_SCREEN LDX #50
2553   CA02
2554   CA02 DF B0       CLR_SCREEN1 STX TEMP16
2555   CA04
2556   CA04 86 0D                 LDAA #13
2557   CA06 BD C0 88              JSR SEND_BYTE
2558   CA09 86 0A                 LDAA #10
2559   CA0B BD C0 88              JSR SEND_BYTE
2560   CA0E
2561   CA0E DE B0                 LDX TEMP16
2562   CA10 09                    DEX
2563   CA11 26 EF                 BNE CLR_SCREEN1
2564   CA13 39                    RTS
2565   CA14
2566   CA14
2567   CA14
2568   CA14             ; PRINT STRING TO TERMINAL USING 2400 UART
2569   CA14             ; ENTRY: IX POINTED TO STRING
2570   CA14
2571   CA14 A6 00       PSTR     LDAA 0,X
2572   CA16 26 01                 BNE SEND_STRING
2573   CA18
2574   CA18 39                    RTS
2575   CA19
2576   CA19             SEND_STRING
2577   CA19
2578   CA19 DF B2                 STX IX2
2579   CA1B
2580   CA1B BD C0 88              JSR SEND_BYTE
2581   CA1E
2582   CA1E DE B2                 LDX IX2
2583   CA20 08                    INX
2584   CA21 20 F1                 BRA PSTR
```

```
2585   CA23
2586   CA23
2587   CA23
2588   CA23
2589   CA23
2590   CA23
2591   CA23
2592   CA23
2593   CA23
2594   CA23
2595   CA23
2596   CA23             ;-------------------- HARDWARE DRIVERS-------------------------
2597   CA23             ; TEXT MODE LCD DRIVERS
2598   CA23
2599   CA23             ; wait until LCD ready bit set
2600   CA23
2601   CA23
2602   CA23 36          LcdReady         PSHA
2603   CA24 B6 90 02    ready            LDAA command_read
2604   CA27 84 80                        ANDA #BUSY
2605   CA29 26 F9                        BNE ready   ; loop if busy flag = 1
2606   CA2B 32                           PULA
2607   CA2C 39             RTS
2608   CA2D
2609   CA2D
2610   CA2D             LCD_command_write
2611   CA2D BD CA 23                      JSR LcdReady
2612   CA30 B7 90 00                      STAA command_write
2613   CA33 39               RTS
2614   CA34
2615   CA34
2616   CA34 BD CA 23    LCD_data_write  JSR LcdReady
2617   CA37 B7 90 01                    STAA data_write
2618   CA3A 39                          RTS
2619   CA3B
2620   CA3B
2621   CA3B BD CA 23    clr_screen       JSR LcdReady
2622   CA3E 86 01                        LDAA #1
2623   CA40 BD CA 2D        JSR LCD_command_write
2624   CA43 39                          RTS
2625   CA44
2626   CA44 86 38       InitLcd          LDAA #38H
2627   CA46 BD CA 2D                     JSR LCD_command_write
2628   CA49 86 0C                        LDAA #0CH
2629   CA4B BD CA 2D              JSR LCD_command_write
2630   CA4E BD CA 3B                     JSR clr_screen
2631   CA51 86 00                        LDAA #0
2632   CA53 C6 00                        LDAB #0
2633   CA55 BD CA 59                     JSR goto_xy
2634   CA58 39                           RTS
2635   CA59
2636   CA59
2637   CA59             ; goto_xy(x,y)
2638   CA59             ; entry: A = x position
2639   CA59             ;        B = y position
2640   CA59
2641   CA59 C1 00       goto_xy          CMPB #0
2642   CA5B 26 06           BNE case1
2643   CA5D 8B 80                        ADDA #80H
2644   CA5F BD CA 2D                     JSR LCD_command_write
2645   CA62 39              RTS
2646   CA63
2647   CA63 C1 01       case1            CMPB #1
2648   CA65 26 06                        BNE case2
2649   CA67 8B C0                        ADDA #0C0H
2650   CA69 BD CA 2D        JSR LCD_command_write
2651   CA6C 39              RTS
2652   CA6D
2653   CA6D 39          case2           RTS
2654   CA6E
2655   CA6E
2656   CA6E
2657   CA6E             ; write ASCII code to LCD at current position
2658   CA6E             ; entry: A
2659   CA6E
2660   CA6E BD CA 34    putch_lcd       JSR LCD_data_write
```

```
2661   CA71 39                          RTS
2662   CA72
2663   CA72
2664   CA72
2665   CA72
2666   CA72             ; print strings on the LCD
2667   CA72             ; entry: IX pointed to string
2668   CA72
2669   CA72
2670   CA72 A6 00       pstring         LDAA 0,X
2671   CA74 26 01                       BNE PSTRING1
2672   CA76 39                          RTS
2673   CA77
2674   CA77 DF B2       PSTRING1        STX IX2
2675   CA79 BD CA 6E                    JSR putch_lcd
2676   CA7C DE B2                       LDX IX2
2677   CA7E 08                          INX
2678   CA7F 20 F1                       BRA pstring
2679   CA81
2680   CA81
2681   CA81             ; DEMO LCD 20X2 BIG LETTERS LCD
2682   CA81
2683   CA81 BD CA 44    DEMO_LCD        JSR InitLcd
2684   CA84
2685   CA84 CE CB 65                    LDX #TEXT1
2686   CA87 BD CA 72                    JSR pstring
2687   CA8A
2688   CA8A 86 00                       LDAA #0
2689   CA8C C6 01                       LDAB #1
2690   CA8E BD CA 59                    JSR goto_xy
2691   CA91 CE CB 75                    LDX #TEXT2
2692   CA94 BD CA 72                    JSR pstring
2693   CA97
2694   CA97
2695   CA97 39                          RTS
2696   CA98
2697   CA98
2698   CA98
2699   CA98             ; BINARY COUNTING  #1
2700   CA98
2701   CA98 86 00       DEMO_LED    LDAA #0
2702   CA9A
2703   CA9A B7 80 00    DEMOLOOP STAA GPIO1
2704   CA9D 8D 04                  BSR DEMODELAY
2705   CA9F 8B 01                  ADDA #1
2706   CAA1 20 F7                  BRA DEMOLOOP
2707   CAA3
2708   CAA3
2709   CAA3 CE 00 00    DEMODELAY    LDX #0000
2710   CAA6 09          DEMODELAY1   DEX
2711   CAA7 26 FD                  BNE DEMODELAY1
2712   CAA9 39                     RTS
2713   CAAA
2714   CAAA
2715   CAAA             ; RUNNING DOT LED  PRM #0
2716   CAAA
2717   CAAA 86 01       RUN_DOT  LDAA #1
2718   CAAC
2719   CAAC B7 80 00    RUN_DOT1 STAA GPIO1
2720   CAAF 8D F2                  BSR DEMODELAY
2721   CAB1 49                     ROLA
2722   CAB2 20 F8                  BRA RUN_DOT1
2723   CAB4
2724   CAB4
2725   CAB4             ; BCD COUNTING  #2
2726   CAB4
2727   CAB4 86 00       BCD_COUNTING LDAA #0
2728   CAB6
2729   CAB6 B7 80 00    BCD_COUNT1   STAA GPIO1
2730   CAB9 8D E8                    BSR DEMODELAY
2731   CABB 8B 01                    ADDA #1
2732   CABD 19                       DAA
2733   CABE 20 F6                    BRA BCD_COUNT1
2734   CAC0
2735   CAC0
2736   CAC0
```

```
2737   CAC0
2738   CAC0             ; UPDATE CLOCK VARIABLES
2739   CAC0             ; ENTER EVERY 10ms FROM IRQ INTERRUPT
2740   CAC0
2741   CAC0             SERVICE_IRQ
2742   CAC0
2743   CAC0
2744   CAC0 96 CC                   LDAA SEC100
2745   CAC2 8B 01                   ADDA #1
2746   CAC4 97 CC                   STAA SEC100
2747   CAC6 81 64                   CMPA #100
2748   CAC8 26 2C                   BNE  EXIT_CLOCK
2749   CACA
2750   CACA 4F                      CLRA
2751   CACB 97 CC                   STAA SEC100
2752   CACD
2753   CACD 96 CD                   LDAA SEC
2754   CACF 8B 01                   ADDA #1
2755   CAD1 19                      DAA
2756   CAD2 97 CD                   STAA SEC
2757   CAD4 81 60                   CMPA #60H
2758   CAD6 26 1E                   BNE EXIT_CLOCK
2759   CAD8
2760   CAD8 4F                      CLRA
2761   CAD9 97 CD                   STAA SEC
2762   CADB
2763   CADB 96 CE                   LDAA MIN
2764   CADD 8B 01                   ADDA #1
2765   CADF 19                      DAA
2766   CAE0 97 CE                   STAA MIN
2767   CAE2 81 60                   CMPA #60H
2768   CAE4 26 10                   BNE EXIT_CLOCK
2769   CAE6 4F                      CLRA
2770   CAE7 97 CE                   STAA MIN
2771   CAE9
2772   CAE9 96 CF                   LDAA HOUR
2773   CAEB 8B 01                   ADDA #1
2774   CAED 97 CF                   STAA HOUR
2775   CAEF 81 24                   CMPA #24H
2776   CAF1 26 03                   BNE EXIT_CLOCK
2777   CAF3
2778   CAF3 4F                      CLRA
2779   CAF4 97 CF                   STAA HOUR
2780   CAF6
2781   CAF6             EXIT_CLOCK
2782   CAF6
2783   CAF6 3B                      RTI
2784   CAF7
2785   CAF7
2786   CAF7             ; CLOCK PROGRAM #3 TEST 10ms TICK MAKING CLOCK DISPLAY
2787   CAF7
2788   CAF7
2789   CAF7 86 7E       CLOCK_PGM  LDAA #7EH    ; INSERT HEX CODE FOR JMP SERVICE_IRQ
2790   CAF9 97 F8                  STAA $F8
2791   CAFB
2792   CAFB 86 CA                  LDAA #SERVICE_IRQ>>8
2793   CAFD 97 F9                  STAA $F9
2794   CAFF 86 C0                  LDAA #SERVICE_IRQ&$FF
2795   CB01 97 FA                  STAA $FA
2796   CB03
2797   CB03 4F                     CLRA
2798   CB04 97 CC                  STAA SEC100
2799   CB06 97 CD                  STAA SEC
2800   CB08 97 CE                  STAA MIN
2801   CB0A 97 CF                  STAA HOUR
2802   CB0C 97 D0                  STAA RUNSTOP
2803   CB0E
2804   CB0E
2805   CB0E 0E                      CLI       ; ENABLE IRQ
2806   CB0F
2807   CB0F             DISPLAY_CLOCK
2808   CB0F
2809   CB0F
2810   CB0F 96 CC                LDAA SEC100
2811   CB11 BD C1 C3             JSR DATA_DISPLAY
2812   CB14
```

```
2813   CB14 96 CD                LDAA SEC
2814   CB16 97 85                STAA HL+1
2815   CB18 96 CE                LDAA MIN
2816   CB1A 97 84                STAA HL
2817   CB1C BD C1 D1             JSR ADDRESS_DISPLAY
2818   CB1F
2819   CB1F 96 8E                LDAA BUFFER+2
2820   CB21 8A 40                ORAA #$40
2821   CB23 97 8E                STAA BUFFER+2
2822   CB25
2823   CB25 96 90                LDAA BUFFER+4
2824   CB27 8A 40                ORAA #$40
2825   CB29 97 90                STAA BUFFER+4
2826   CB2B
2827   CB2B
2828   CB2B
2829   CB2B BD C8 DA             JSR SCAN1
2830   CB2E
2831   CB2E 96 93                LDAA KEY
2832   CB30 81 FF                CMPA #-1
2833   CB32 27 0D                BEQ SKIP_SHOW_KEY
2834   CB34
2835   CB34 81 21                CMPA #21H
2836   CB36 26 09                BNE SKIP_SHOW_KEY
2837   CB38
2838   CB38 4F                   CLRA
2839   CB39 97 CC                STAA SEC100
2840   CB3B 97 CD                STAA SEC
2841   CB3D 97 CE                STAA MIN
2842   CB3F 97 CF                STAA HOUR
2843   CB41
2844   CB41                    ;   STAA GPIO1
2845   CB41
2846   CB41             SKIP_SHOW_KEY
2847   CB41
2848   CB41 20 CC                BRA DISPLAY_CLOCK
2849   CB43
2850   CB43
2851   CB43
2852   CB43             ; CHECK KEY PRESSED
2853   CB43             ; +    CODE 21H
2854   CB43             ; ADDR CODE 1BH
2855   CB43             ; GO   CODE 16H
2856   CB43
2857   CB43
2858   CB43             DISPLAY_START_MSG
2859   CB43
2860   CB43 CE CB F5             LDX #START_MSG+6
2861   CB46 DF B0                STX TEMP16
2862   CB48
2863   CB48
2864   CB48 86 06                LDAA #6
2865   CB4A 97 D1                STAA K
2866   CB4C
2867   CB4C             DISPLAY_MSG1
2868   CB4C
2869   CB4C 86 20                LDAA #$20
2870   CB4E
2871   CB4E 97 D2                STAA J
2872   CB50
2873   CB50
2874   CB50             DISPLAY_START1
2875   CB50
2876   CB50 DE B0                LDX TEMP16
2877   CB52 BD C8 DD             JSR SCAN2
2878   CB55
2879   CB55 7A 00 D2             DEC J
2880   CB58 26 F6                BNE DISPLAY_START1
2881   CB5A
2882   CB5A DE B0                LDX TEMP16
2883   CB5C 09                   DEX
2884   CB5D DF B0                STX TEMP16
2885   CB5F
2886   CB5F 7A 00 D1             DEC K
2887   CB62 26 E8                BNE DISPLAY_MSG1
2888   CB64
```

```
2889   CB64 39                    RTS
2890   CB65
2891   CB65
2892   CB65
2893   CB65
2894   CB65
2895   CB65
2896   CB65
2897   CB65
2898   CB65
2899   CB65
2900   CB65
2901   CB65
2902   CB65
2903   CB65
2904   CB65
2905   CB65
2906   CB65
2907   CB65
2908   CB65
2909   CB65             ;sssssssssssssssssssssssss STRINGS CONSTANT sssssssssssssssssssssssss
2910   CB65             ;
2911   CB65
2912   CB65 20204D6F746FTEXT1           .BYTE "  Motorola 6802",0
2912   CB6B 726F6C61203638303200
2913   CB75 4D6963726F70TEXT2           .BYTE "Microprocessor Kit",0
2913   CB7B 726F636573736F72204B697400
2914   CB88
2915   CB88 0D0A36383032TEXT3           .BYTE 13,10,"6802 MICROPROCESSOR KIT V1.0",0
2915   CB8E 204D4943524F50524F434553534F52204B49542056312E3000
2916   CBA7
2917   CBA7
2918   CBA7
2919   CBA7
2920   CBA7
2921   CBA7 0D0A50726573DOWNLOAD .BYTE 13,10,"Press key GO",0
2921   CBAD 73206B657920474F00
2922   CBB6 0D0A436C6963DOWNLOAD2 .BYTE 13,10,"Click File>Send File>..",0
2922   CBBC 6B2046696C653E53656E642046696C653E2E2E00
2923   CBD0 0D0A4E6F2065COMPLETE .BYTE 13,10,"No error..",0
2923   CBD6 72726F722E2E00
2924   CBDD 0D0A43686563ERROR_FOUND .BYTE 13,10,"Checksum error!",0
2924   CBE3 6B73756D206572726F722100
2925   CBEF
2926   CBEF 00          START_MSG .BYTE 0
2927   CBF0 00                    .BYTE 0
2928   CBF1 9B                    .BYTE 9BH
2929   CBF2 BD                    .BYTE 0BDH
2930   CBF3 BF                    .BYTE 0BFH
2931   CBF4 AF              .BYTE 0AFH
2932   CBF5 00                    .BYTE 0
2933   CBF6 00              .BYTE 0
2934   CBF7 00              .BYTE 0
2935   CBF8 00              .BYTE 0
2936   CBF9 00              .BYTE 0
2937   CBFA 00                    .BYTE 0
2938   CBFB
2939   CBFB
2940   CBFB BD          SEGTAB  .BYTE   0BDH    ;'0'
2941   CBFC 30           .BYTE   030H    ;'1'
2942   CBFD 9B           .BYTE   09BH    ;'2'
2943   CBFE BA           .BYTE   0BAH    ;'3'
2944   CBFF 36           .BYTE   036H    ;'4'
2945   CC00 AE           .BYTE   0AEH    ;'5'
2946   CC01 AF           .BYTE   0AFH    ;'6'
2947   CC02 38           .BYTE   038H    ;'7'
2948   CC03 BF           .BYTE   0BFH    ;'8'
2949   CC04 BE           .BYTE   0BEH    ;'9'
2950   CC05 3F           .BYTE   03FH    ;'A'
2951   CC06 A7           .BYTE   0A7H    ;'B'
2952   CC07 8D           .BYTE   08DH    ;'C'
2953   CC08 B3           .BYTE   0B3H    ;'D'
2954   CC09 8F           .BYTE   08FH    ;'E'
2955   CC0A 0F           .BYTE   00FH    ;'F'
2956   CC0B
2957   CC0B
```

```
2958   CC0B
2959   CC0B             ; Key-posistion-code to key-internal-code conversion table.
2960   CC0B
2961   CC0B             KEYTAB:
2962   CC0B 03          K0 .BYTE  03H  ;HEX_3
2963   CC0C 07          K1 .BYTE  07H  ;HEX_7
2964   CC0D 0B          K2 .BYTE  0BH  ;HEX_B
2965   CC0E 0F          K3 .BYTE  0FH  ;HEX_F
2966   CC0F 20          K4 .BYTE  20H  ;NOT USED
2967   CC10 21          K5 .BYTE  21H  ;NOT USED
2968   CC11 02          K6 .BYTE  02H  ;HEX_2
2969   CC12 06          K7 .BYTE  06H  ;HEX_6
2970   CC13 0A          K8 .BYTE  0AH  ;HEX_A
2971   CC14 0E          K9 .BYTE  0EH  ;HEX_E
2972   CC15 22          K0A  .BYTE  22H  ;NOT USED
2973   CC16 23          K0B  .BYTE  23H  ;NOT USED
2974   CC17 01          K0C  .BYTE  01H  ;HEX_1
2975   CC18 05          K0D  .BYTE  05H  ;HEX_5
2976   CC19 09          K0E  .BYTE  09H  ;HEX_9
2977   CC1A 0D          K0F  .BYTE  0DH  ;HEX_D
2978   CC1B 13          K10  .BYTE  13H  ;STEP
2979   CC1C 1F          K11  .BYTE  1FH  ;TAPERD
2980   CC1D 00          K12  .BYTE  00H  ;HEX_0
2981   CC1E 04          K13  .BYTE  04H  ;HEX_4
2982   CC1F 08          K14  .BYTE  08H  ;HEX_8
2983   CC20 0C          K15  .BYTE  0CH  ;HEX_C
2984   CC21 12          K16  .BYTE  12H  ;GO
2985   CC22 1E          K17  .BYTE  1EH  ;TAPEWR
2986   CC23 1A          K18  .BYTE  1AH  ;CBR
2987   CC24 18          K19  .BYTE  18H  ;PC
2988   CC25 1B          K1A  .BYTE  1BH  ;REG
2989   CC26 19          K1B  .BYTE  19H  ;ADDR
2990   CC27 17          K1C  .BYTE  17H  ;DEL
2991   CC28 1D          K1D  .BYTE  1DH  ;RELA
2992   CC29 15          K1E  .BYTE  15H  ;SBR
2993   CC2A 11          K1F  .BYTE  11H  ;-
2994   CC2B 14          K20  .BYTE  14H  ;DATA
2995   CC2C 10          K21  .BYTE  10H  ;+
2996   CC2D 16          K22  .BYTE  16H  ;INS
2997   CC2E 1C          K23  .BYTE  1CH  ;MOVE
2998   CC2F
2999   CC2F
3000   CC2F
3001   CC2F                   ; VECTOR NMI,RESET AND IRQ
3002   CC2F
3003   CC2F
3004   FFF8                        .ORG 0FFF8H
3005   FFF8
3006   FFF8 00 F8                  .BYTE 0,0F8H ; RELOCATE IRQ TO RAM VECTOR AT 00F8H
3007   FFFA
3008   FFFA
3009   FFFA                        .ORG 0FFFAH     ; SWI VECTOR
3010   FFFA
3011   FFFA C6 5C                  .BYTE SWI_SERVICE>>8,SWI_SERVICE&$FF
3012   FFFC
3013   FFFC
3014   FFFC                        .ORG 0FFFCH     ; NMI VECTOR
3015   FFFC
3016   FFFC C6 55                  .BYTE NMI_SERVICE>>8,NMI_SERVICE&$FF
3017   FFFE
3018   FFFE
3019   FFFE                        .ORG 0FFFEH
3020   FFFE
3021   FFFE C0 00                  .BYTE $C0,00   ; RESTART
3022   0000
3023   0000
3024   0000
3025   0000
3026   0000                 .END
3027   0000
3028   0000
3029   0000
tasm: Number of errors = 0
```

NOTE