

## ELECTRONICS-2

### Microprocessors: Lab 1- Introduction

**Aims:**

- Familiarize yourself with HC-COM hardware,
- Find out about the BUFFALO monitor program,
- Use BUFFALO commands to view the contents of memory and processor registers.
- Examine the directly addressable memory space of MC68HC11 microprocessor and find out how it is partitioned,
- Explore various instructions and addressing modes of the M68HC11.

**Equipment:** HC-COM, power supply, video terminal.

**References:** For lab classes you will need:

- Instruction Set Reference and the HC-COM schematic diagram handed out during lecture 3.
- Bring a bound A4 log book for writing your observations and keeping your notes.

A textbook (“Microcomputer Technology: The 68HC11” by Peter Spasov)  
And a HC11 Reference Manual would also be useful.  
You should also refer to lecture notes, hand-outs and other references.

### Introduction

Computers are composed of many components. They may have RAM and ROM (just about all computers have these, but there are exceptions !). RAM is Random Access Memory. It is the memory where the computer uses to store data and programs. RAM can be written-to, or read-from, by the processor.

ROM stands for Read Only Memory. In contrast, ROM retains its contents when power goes off, but its contents cannot be easily changed as RAM. On your HC-COM, the RAM is IC U2, Part number 43256 static RAM containing 32K or 32768 bytes of storage. The ROM on the HC-COM is U3, Part number 27C256, EPROM (Erasable Programmable ROM), also containing 32K or 32768 bytes of non-volatile storage.

EPROMs can be erased by shining UV light through the little window on the top of the package and onto the integrated circuit inside. They can be reprogrammed using a special programmer. EPROMs typically contain programs that the computer requires to run. The ROM on the HC-COM contains the BUFFALO monitor program which manages the operation of the HC-COM.

An instruction in a computer program is a number. Note that the processor makes no distinction between instructions and data. The only difference between the two is whether the processor happens to be trying to execute it or not. A program that the processor can execute looks something like this:

86 41 B7 01 00 BD 02 00 ...

Humans find this very hard to write and even harder to understand. To make this easier to cope with, we use a notation called assembly language. Assembly language instructions equate directly to their machine code counterparts.

So, the above machine code program becomes:

Machine Code	Assembly
86 41	LDAA #41
B7 01 00	STAA 0100
BD 02 00	JSR 0200

Note: we almost always use hexadecimal for machine codes.

LDAA #41 means

**LoaD Accumulator A** with the **immediate** value (signified by #) of **41** (HEXadecimal).  
**Immediate** means the value that immediately follows the OPCODE in the program.

STAA 0100 means

**STore** the contents of **Accumulator A** to the Memory location 0100 (HEX) (note: no # this time).

JSR 0200 means

**Jump to SubRoutine** located at address **0200** (HEX)  
(JSR is used for function/subroutine/procedure/method **calls**)

Computers do not understand high level language such as C or BASIC directly. Computers do not understand assembly language either. Computers understand only machine codes. To convert a program written in an assembly language into machine codes, we have a special program known as an assembler.

There is a simple built-in assembler in BUFFALO that will convert your assembly language program into machine codes for you. In this case, the machine codes are those that can be understood by the MC68HC11 processor. A different assembler will be required for a different processor, e.g. the Intel 8051 processor.

### Be considerate of others

It is important that you work in pairs and are considerate of others. Students caught cheating or interfering with other students will be asked to leave and disciplinary action will be taken.

### Laboratory Practices

**Always ensure that no harm will come to any person or equipment.**

HC-COM computer is based on HCMOS technology. Always carry electronic components in their supplied anti-static packaging. Ensure you discharge any electro-static charge on your body by touching an earthed metallic part or anti-static mat, before touching any electronics. **HC-COM** is an acronym for LaTrobe University's 68HC11 **COM**puter.

Electric charge builds up on the human body. It is typically collected from free charge built up on the surface of carpet, seating or clothing. [light can dislodge electron charge from the surface of insulating material. The charge is removed by air movement on atoms & air particles. This leaves residual positive charge on the surface. Contacting the charged surface field attracts electrons from our body leaving a net positive charge on our body. This static charge is then discharged - often via a spark - to ground when contact to ground is made].

If static electricity discharges through CMOS electronics, the discharge (spark) can cause a pin hole through the semiconductor, and may cause malfunction, latch-up or complete failure of the device.

### **Setting up HCCOM**

The 68HC11 requires 5 volt power supplies. However a regulator on the top board will accept any voltage from +8 to +15 Volts and regulates that to 5Volts via a 7805 linear regulator. Using the banana plugs connect +8 volts to the Red wire and 0 volts to the black wire. The Blue wire is for a -8 to -15V negative supply; it is only needed if using the Analog input OP Amps - the negative supply need not be connected today.

### **Lab Report:**

**Answer all questions, perform all the exercises and prepare your report in your lab notebook as you proceed.**

**Have your completed report marked by a lab demonstrator before departing the lab.**

**1. Like any other electronic device HC-COM needs a power source to run.  
What voltage do you need to run HC-COM?**

**Why?**

**Q2. How do you connect a power supply to HC-COM?**

**Write a description in your log book.**

Double check the connection (check and cross check).

Check for other stray wires and metal objects that may be touching the circuit boards unintentionally. Then check once more. If all is safe, correct, and all reasonable double-checks have been made Switch the power supply On.

All components should stay reasonably cool except for the 5Volt regulator.

### 3. HOW TO START HYPERTERMINAL TO COMMUNICATE WITH HCCOM

In order to communicate with HC-COM we need a terminal program and keyboard so that outputs can be displayed and commands can be entered. We will use Hyperterminal

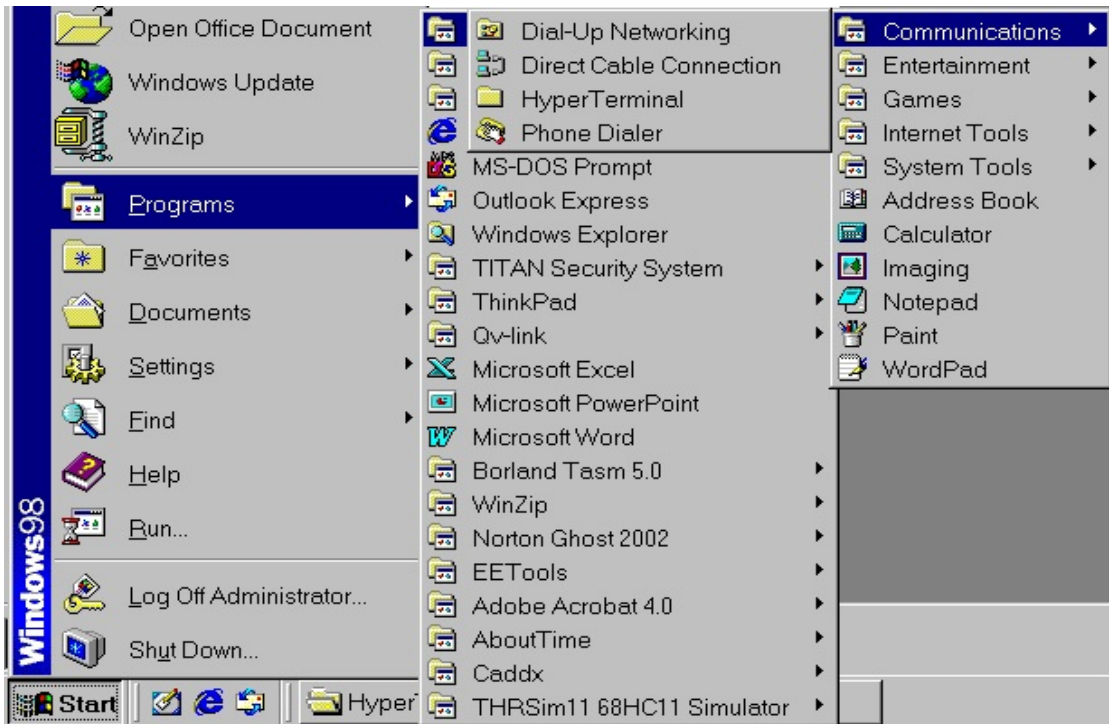


Figure 1 - Start->Programs->Accessories->Communications->HyperTerminal

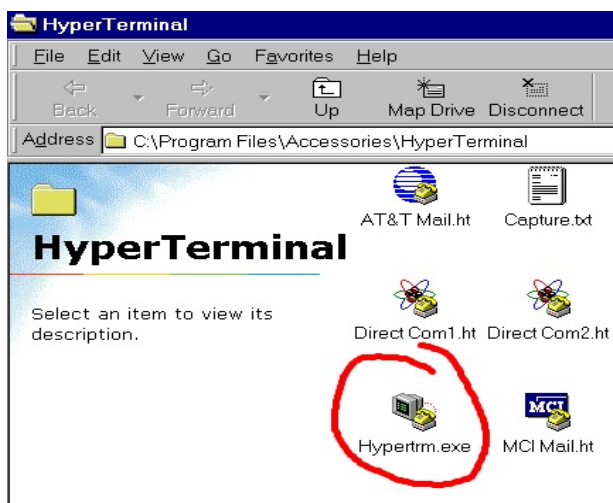


Figure 2 DoubleClick on Hyperterminal.exe

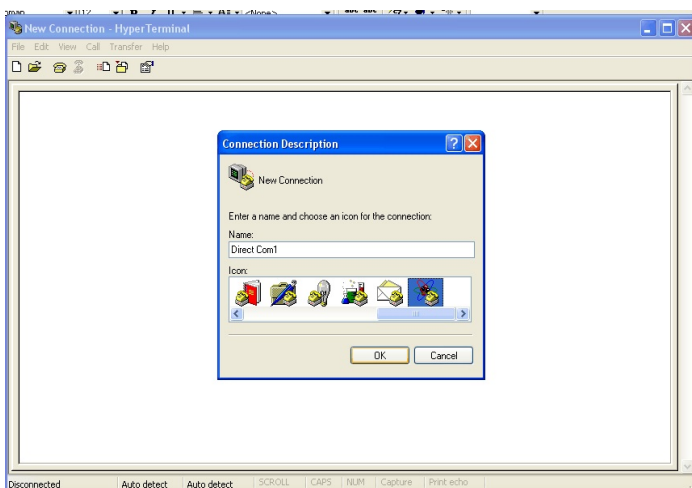
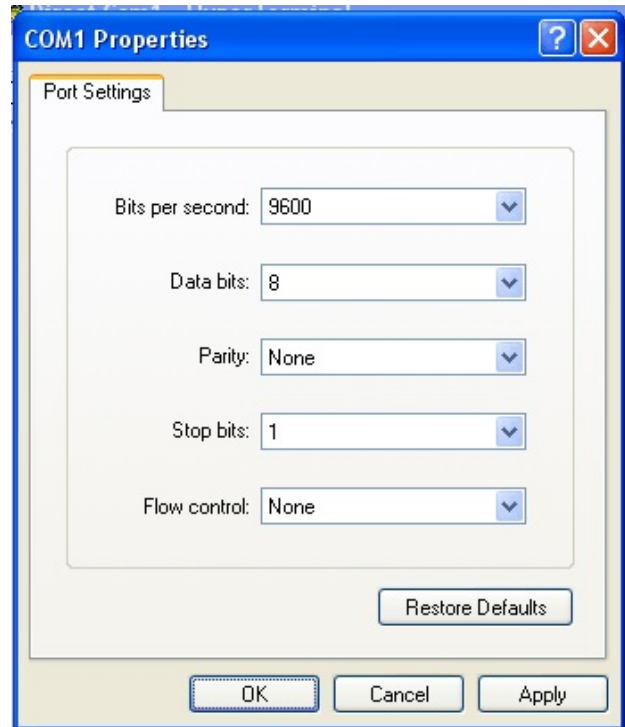


Figure 3 Name the connection: Direct Com1



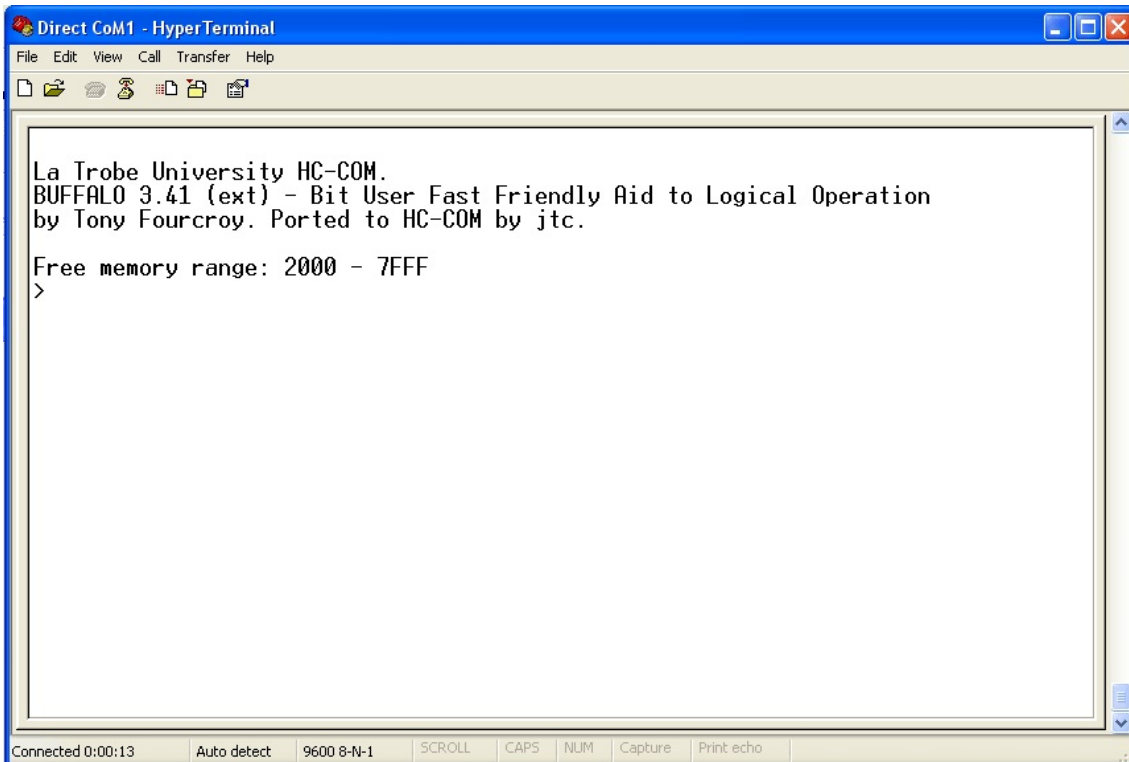
**Figure 4** Select serial port: Com1



**Figure 5** Choose Communications Options: 9600bps, No Parity, 8 Data bits, 1 Stop Bit, Flow Control:None

Once the program is started, use the **file->connect** option to connect to HC-COM. Press the reset switch and check that HC-COM boots correctly.

**Q3. What is the name of the "interface" we use to connect HC-COM to the IBM-PC?**



**Figure 6** Connect Up HCCOM and switch the power on. You should see message something like this when HCCOM boots.

**Q4. What is the size of the directly addressable memory space?**

**Q5. How is memory space partitioned in the HC-COM design?**

**Q6. How much RAM does HC-COM have ?**

**Q7. What is the address range for these RAM ?**

**Q8. How much ROM does HC-COM have ?**

**Q9. What is the address range for the ROM ?**

**Q10.** BUFFALO has **14** commands.

The most useful one for this instance is the "Memory Dump" command.

It has the format: "**MD** <address1> <address2>".

For example to dump the memory contents between 0 and FF type: **MD 0 FF <enter>**

We can use it to display the contents of memory between address1 and address2.

**Examine and record the contents of memory between addresses 0000 and 00FF.**

**Q11.** Another command which is also very useful for this instance is the "**MM** address" command. MM stands for modify memory. With this command you can change the contents of the memory at "address", one location at a time.

**Q12.** However, if we are dealing with a block of memory, we can use a more powerful command "**BF** [address] address? data", where BF stands for block fill. In other words we can fill the memory range from address] to address2 with data in one sweep.

**Q13.** The RAM space available for your program/experimentation is between **2000 and 7FFF**. Fill this RAM'space with "EF" and verify that "EF" has indeed been written to these RAM locations. **Describe how this is done.**

**Q14.** Write "00" to part of the ROM space (from 8000 to 80FF) and verify the operation. **Record what you have found and explain your results.**

**Q15.** On certain occasions it is desirable to know the contents of the various registers of MC68HC11. The "RM" command allows you to view or modify all registers (p, x, y, a, b, c, s). What do these letters stand for ? What are they used for? Record and analyze their contents.

P =

X =

Y =

A =

B =

C =

S =

For example, enter: **RM X**

**1234**

**R**

**What is the new value of X ?**

**Q16.** The BUFFALO command to commence assembling code starting from an address is "A address". Use this command to enter the following three lines of assembly codes into HC-COM starting at address 2000:

```
LDAA #41  
STAA 0100  
JSR 0200
```

(Remembering that the **RAM** space available for programming is between **2000 and 7FFF**.) After you have entered each line of codes, BUFFALO responds with the appropriate machine codes, also called opcodes. **What are the machine codes for the above 3-line program ?** (To get out of the assembly mode, type **Ctrl-C**.)

**Q17.** Use the command "MD 2000 2007" to display the contents of the memory locations from 2000 to 2007. What did you find ?

**Q18.** Fill the memory locations between 0100 and 010F with 00, and verify the operation.

**Q19.** Run the above 3-line program with the command "G 2000", and the computer will crash on you. Push the reset button on HC-COM to re-enter BUFFALO. Why should the computer crash ?

**Q20.** Change the third line of codes to SWI (It stands for SoftWare Interrupt), which will allow you to get back into BUFFALO.

Run the new program.

What was the response from BUFFALO?

**Q21.** Examine the memory locations between 0100 and 010F again.. What did you find ?

Addition using two 8-bit accumulators:

**Q 22.** Assemble the following code snippets starting at location \$2000

```
LDAA #21  
LDAB #34  
ABA  
SWI
```

Which register is the destination ?

What is the result?

If any other registers changed, describe which ones and why.

Subtraction using two 8-bit accumulators:

```
Q23. LDAA #32  
LDAB #64  
SBA  
SWI
```

Which register is the destination ?

What is the result?

If any other registers changed, describe which ones and why.

Shift Left = **Multiply by 2**

**Q24.**

```
LDAA    #AA
LDAB    #0
ASLD
SWI
```

Which register is the destination ?

What is the result?

If any other registers changed, describe which ones and why.

**Q25.** Assemble the program ASCII.Z.ASM

In MSDOS Type **AS11 ASCiiz.ASM -I > ASCII.Z.LST**

Load the program ascii.z.s19 at location \$6000

eg: Type **L 6000**

then in hyperterminal select **send->textFile->ascii.z.s19**

(need to select file type **all files not the default txt**).

The program should load and indicate success with the response 'done'

Run it at location 6000.Type to BUFFALO **G 6000**

What happened?

**Q26.** Modify the ASCII.Z.ASM code, or write your own code, to display one of your student numbers on the LED display.

Hint: Modify the program ASCII.Z.ASM code starting at "TESTSTART".

### **BUFFALO COMMANDS:**

BF <addr1> <addr2> [<data>]	Block fill memory
BR [-][<addr>]	Set up bkpt table
BULK	Erase EEPROM,
BULKALL	Erase EEPROM and CONFIG
CALL [<addr>]	Call subroutine
GO [<addr>]	Execute code at addr,
PROCEED	Continue execution
EEMOD [<addr> [<addr>]]	Modify EEPROM range
LOAD, VERIFY [T] <host dwnld command>	Load or verify S-records
MD [<addr1> [<addr2>]]	Memory dump
MM [<addr>] or [<addr>]/	Memory Modify
[/,=]	Same addr,
[^,-,CTLH]	Prev addr,
[+,CTLJ,SPACE]	Next addr
<addr>O	Compute offset,
[CR]	Quit
MOVE <s1> <s2> [<d>]	Block move
OFFSET [-]<arg>	Offset for download
RM [P,Y,X,A,B,C,S]	Register modify
STOPAT <addr>	Trace until addr
T [<n>]	Trace n instructions
TM	Transparent mode (CTLA = exit, CTLB = send brk)
[CTLW]	Wait,
[CTLX,DEL]	Abort
[CR]	Repeat last command

```

6811 assembler version 2.1 10-Aug-91
  please send bugs to Randy Sargent (rsargent@athena.mit.edu)
  original program by Motorola.
0001      * as11 asciiz.asm -l > asciiz.lst
0002      *
0003
0004      *** HCCOM Hardware equates
0005 2000  RAM      EQU $2000      * Start of RAM
0006
0007      *** RAM Variable Assignments
0008 6000  ORG      $6000      * Start variables in EVB RAM (upper half)
0009 6000 7e 60 b2  JMP      TESTSTART * Jump over variables to test program...
0010 1000  REGBAS   EQU $1000    * Starting address for 68HC11 configuration register
block
0011 6000  PROGSTART EQU $6000  * Address to load this program
0012
0013 0008  PORTD   EQU $08      * Port D data register --,SS#,SCK,MOSI,MISO,TxD,RxD
0014 0009  DDRD    EQU $09      * Port D data direction
0015 0028  SPCR    EQU $28      * SPI control register
"SPIE,SPE,DWOM,MSTR;CPOL,CPHA,SPR1,SPRO"
0016 0029  SPSR    EQU $29      * SPI status register "SPIF,WCOL,-,MODF;-,-,-"
0017 002a  SPDR    EQU $2A      * SPI data register On Read-Buffer; On Write-Shifter
0018
0019      * The LED Driver chip MAX7219 (or MAX7221) do not naturally
0020      * contain a hexadecimal font, so we create a font here.
0021      * Each Bit that is set to a "1" causes the LED to be illuminated, and
0022      * each Bit that is set to a "0" causes the LED to be extinguished.
0023      *
0024      * The MAX7219 and MAX7221 are functionally equivalent. The main
0025      * difference between them is the MAX7221 will generate less RF/
0026      * electrical noise due to slew rate limiting on its outputs.
0027      * Throughout this document we will refer only to the MAX7219
0028      * and you may assume identical operation for the MAX7221.
0029      *
0030      * The LED Driver chip is controlled using a serial peripheral
0031      * interface. We will use the supplied routine to access the
0032      * display.
0033      *
0034      * The bit format used below is: DP,a,b,c,d,e,f,g
0035 6003 7e  HEXFONT  FCB $01111110 * 0
0036 6004 30  FCB $00110000 * 1
0037 6005 6d  FCB $01101101 * 2
0038 6006 79  FCB $01111001 * 3
0039 6007 33  FCB $00110011 * 4
0040 6008 5b  FCB $01011011 * 5
0041 6009 5f  FCB $01011111 * 6
0042 600a 70  FCB $01110000 * 7
0043 600b 7f  FCB $01111111 * 8
0044 600c 7b  FCB $01111011 * 9
0045 600d 77  FCB $01110111 * A
0046 600e 1f  FCB $00011111 * b
0047 600f 0d  FCB $00001101 * c
0048 6010 3d  FCB $00111101 * d
0049 6011 4f  FCB $01001111 * E
0050 6012 47  FCB $01000111 * F
0051      *
0052      * Where the seven-segment display LEDs are labelled as follows:
0053      * --a-
0054      * f| |b
0055      * | |
0056      * --g-
0057      * e| |c
0058      * | |
0059      * --d-
0060      * (dp) - There is no decimal point in HCCOM displays.
0061      *
0062 6013  ledIndex  rmb 1
0063
0064      *
-----
0065      * SOFTWARE PROGRAMMED 16 BIT SERIAL PERIPHERAL INTERFACE (SPI) TRANSFER
0066      * THE 68HC11 can only perform 8 bit SPI natively, so we do SPI in software.
0067      *
-----
0068 0004  SDIN    EQU #$04
0069 0008  SCLOCK  EQU #$08
0070 0010  SENBAR  EQU #$10
0071 0020  SDOUT   EQU #$20
0072
0073      * LED Data Bit Format:
0074      * 15 0
0075      * x x x x ADR3 ADR2 ADR1 ADR0 D7 D6 D5 D4 D3 D2 D1 D0
0076      * ADDRESS = 0 = 0 = NOP
0077      * ADDRESS = 1..8 = Digit 0..7
0078      * ADDRESS = 9 = Decode Mode
0079      * ADDRESS = 10 = Intensity
0080      * ADDRESS = 11 = Scan Limit
0081      * ADDRESS = 12 = Shutdown
0082      * ADDRESS = 15 = Display Test
0083      *
0084      * -----
0085      * Initialise MAX7219 LED DISPLAY DRIVER
0086      * Initialise to use a software generated font
0087      * Set the brightness tp half intensity (using pulse width modulation)
0088      * -----
0089  initLeds
0090 6014 36  psha
0091 6015 37  pshb
0092
0093      * Select 7 Seg decode for all digits, 00=no decode (use HEXFONT above)
0094 6016 cc 09 00  LDD  #$0900
0095 6019 8d 17  BSR  SPI

```

```

0096          * Intensity Register - Set displays to half brightness using PWM...
0097 601b cc 0a 07      LDD  #0A07
0098 601e 8d 12      BSR  SPI
0099          * Scan limit - Set to 8 LED displays
0100 6020 cc 0b 07      LDD  #0B07
0101 6023 8d 0d      BSR  SPI
0102          * Shutdown Mode = OFF = Normal Operation
0103 6025 cc 0c ff      LDD  #0CFF
0104 6028 8d 08      BSR  SPI
0105          * Display Test
0106          * LDD  #0FFF
0107          * BSR  SPI
0108          * Exit Display test -> Normal Mode
0109 602a cc 0f 00      LDD  #0F00
0110 602d 8d 03      BSR  SPI
0111
0112 602f 33          pulb
0113 6030 32          pula
0114 6031 39          RTS
0115
0116
0117          *
-----
0118          * Transmit the 16 bit data, passed in Accumulator D, via software SPI
0119          * Accumulator D = Accumulator A concatenated with Accumulator B.
0120          * Accumulator: A = Address; Accumulator B = Data
0121          * Returns: 16 bits from SPI device in D
0122          *
-----
0123          SPI:
0124 6032 3c          PSHX
0125 6033 18 3c      PSHY
0126          * Use X to point to the 68HC11 configuration registers
0127 6035 ce 10 00    LDX  #REGBAS
0128          *
0129          * Initialise 68HC11 port D directions:
0130          * ON HCCOM Interface Adapter The MAX7219
0131          * LED display driver is connected as follows
0132          * Port D Pin 0 <- RS485 SCI Receive Data
0133          * Port D Pin 1 -> RS485 SCI Transmit Data
0134          * Port D Pin 2 <- Pin 24 = Data Out from MAX7219
0135          * Port D Pin 3 -> Pin 13 = Serial Clock - Normally low, rising edge to latch
data bit ->MAX7219
0136          * Port D Pin 4 -> Pin 12 = CS# - Active Low Chip Select, so assert high asap ->
MAX7219
0137          * Port D Pin 5 -> Pin 1 = Data in to MAX7219
0138 6038 36          PSHA
0139 6039 86 3a      LDAA  #%00111010
0140 603b a7 09      STAA  DDRD,X
0141          * Make sure that 8-bit native SPI is switched off
0142          * SPCR config bits: "SPIE,SPE,DWOM,MSTR;CPOL,CPHA,SPR1,SPR0"
0143 603d 86 00      LDAA  #00
0144 603f a7 28      STAA  SPCR,X
0145 6041 32          PULA
0146
0147          * PERFORM SPI TRANSFER VIA PROGRAM CONTROL
0148          * SET MAX7219 CS# active
0149 6042 1d 08 10    BCLR  PORTD,X SENBAR      * BCLR = BIT CLEAR
0150          * SET MAX7219 SCLOCK to 0
0151 6045 1d 08 08    BCLR  PORTD,X SCLOCK
0152          * We need to send 16 bits, so use a counter...
0153          * Use Y as a counter from 16..0
0154 6048 18 ce 00 10  LDY  #16      * Y register = 16
0155
0156          * Now Send the data bit by bit
0157          NEXT_SPI
0158          *** Now set MAX7219 SPI Clock line low on LED Driver
0159 604c 1d 08 08    BCLR  PORTD,X SCLOCK      * Set the SPI Clock clock low
0160
0161          * Shift D by 1 bit
0162          * Acc.B has least significant byte
0163 604f 58          ASLB  * 0 -> Least Significant Bit, MSBit ->CF
0164          * Acc.A has most significant byte
0165 6050 49          ROLA  * CarryIn -> LSB, MSB -> CarryOut
0166          * Now carry flag has (next) most significant bit
0167 6051 24 05      BCC   CLR_BIT      * if carry clear then goto CLR_BIT
0168          * else Bit is set so send a "1"
0169 6053 1c 08 20    BSET  PORTD,X SDOUT   * 1008 = Port D = Write Serial Data Out = 1
0170 6056 20 03      BRA   SET_BIT      * Skip next bit
0171          CLR_BIT
0172          *
0173 6058 1d 08 20    BCLR  PORTD,X SDOUT   * Bit was clear so send a "0"
0174          * 1008 = Port D = Write Serial Data Out = 0
0175          SET_BIT
0176          *
0177 605b 01          NOP      * ENDIF
0178 605c 01          NOP      * Do nothing
0179
0180          * Now strobe Clock line high on MAX7219 to shift the data
0181 605d 1c 08 08    BSET  PORTD,X SCLOCK   * 8 Cycles
0182 6060 01          NOP
0183 6061 01          NOP
0184          * and strobe Clock line low on 7219
0185 6062 1d 08 08    BCLR  PORTD,X SCLOCK
0186          *
0187          * read the serial data shifted in to the 68HC11 -> Z flag
0188          * Data is clocked out of MAX7219 on falling edge
0189 6065 1f 08 04 20 BRCLR  PORTD,X SDIN NOBITS
0190 6069 ca 01      ORB  #1      * Set bit 0=1 if serial data =1
0191          NOBITS
0192          * Bit 0 is clear already from the ASLB instruction
0193 606b 18 09      DEY

```

```

0194 606d 26 dd          BNE    NEXT_SPI
0195                    *
0196                    *
0197                    DONE_TX:
0198                    * Now set Clock line low on 7219
0199 606f 01            NOP
0200 6070 01            NOP
0201 6071 1d 08 08      BCLR   PORTD,X SCLOCK
0202                    *
0203 6074 01            NOP
0204 6075 01            NOP
0205                    * SET 7219 CS# inactive
0206 6076 1c 08 10      BSET   PORTD,X SENBAR
0207                    *
0208                    * Recover stuff from stack
0209 6079 18 38          PULY
0210 607b 38            PULX
0211 607c 39            RTS
0212
0213
0214                    outHexLedA:
0215                    * -----
0216                    * A = Hex Byte Data to display, ledIndex=led pair
0217                    * -----
0218                    * Save registers on stack...
0219 607d 36            psha
0220 607e 37            pshb
0221 607f 18 3c          pshy
0222
0223                    tab
0224 6082 b6 60 13        ldaa  ledIndex
0225 6085 7c 60 13        inc  ledIndex
0226                    * then mult by 2 as they are LED pairs
0227 6088 48            asla
0228                    * Led registers are 1 through 8 (not 0..7)
0229 6089 4c            inca
0230
0231                    * Get most significant hex nibble
0232 608a 36            psha
0233 608b 37            pshb
0234 608c 54            lsrb
0235 608d 54            lsrb
0236 608e 54            lsrb
0237 608f 54            lsrb
0238                    * Lookup font table for bits to set
0239 6090 18 ce 60 03    ldy   #HEXFONT
0240 6094 18 3a          aby
0241 6096 18 e6 00      ldab  0,y
0242 6099 bd 60 32      jsr   SPI
0243 609c 33            pulb
0244 609d 32            pula
0245
0246                    * Point to next display bit
0247 609e 4c            inca
0248                    * Get most significant hex nibble
0249 609f c4 0f          andb  #%00001111
0250 60a1 18 ce 60 03    ldy   #HEXFONT
0251 60a5 18 3a          aby
0252 60a7 18 e6 00      ldab  0,y
0253 60aa bd 60 32      jsr   SPI
0254
0255                    * Recover saved registers
0256 60ad 18 38          puly
0257 60af 33            pulb
0258 60b0 32            pula
0259 60b1 39            rts
0260
0261                    TESTSTART:
0262 60b2 bd 60 14        JSR   initLeds
0263
0264 60b5 7f 60 13        clr  ledIndex
0265
0266 60b8 86 01          ldaa  #$01
0267 60ba bd 60 7d        jsr  outHexLedA
0268 60bd 86 23          ldaa  #$23
0269 60bf bd 60 7d        jsr  outHexLedA
0270 60c2 86 45          ldaa  #$45
0271 60c4 bd 60 7d        jsr  outHexLedA
0272 60c7 86 67          ldaa  #$67
0273 60c9 bd 60 7d        jsr  outHexLedA
0274
0275 60cc 3f            swi

```