Experiment 1

Familiarization with MDA8086 and MTS86C Microprocessor Trainers

1.1 Objective

The objectives of this experiment are

- Familiarization with Microprocessor Trainers
- Using MDA8086 or MTS86C trainers for Assembly Programming and Testing
- Making absolute machine codes from assembly program using assembler, loader
- Communicating MDA8086 or MTS86C with a PC through serial communication

1.2 Learning Outcome

At the end of the experiment the students will be able to

- Make machine code (ABS or HEX) files and run on a microprocessor system
- Use MDA8086 and MTS86C trainers for low level programming
- Communicate between a PC and a microprocessor trainer

1.3 Trainer Kits

We will use two different types of 8086 microprocessor trainers during experiments. The two trainer kits are, MTS-86C manufactured by K&H, Taiwan and MDA-8086 manufactured by Midas Engineering, Korea.

1.3.1 Features

The features of the two trainers are summarized in Table 1.1.

Table 1.1 Features of MDA-8086 and MTS-86C Trainers (comparison chart)

Specification	MDA-8086	MTS-86C
8086 CPU	4.9152MHz	4.9152MHz
Display	LCD (16x2 Line)	LCD (16x2 Line)
Monitor ROM	27C256 x 2 (64KB) F0000～FFFFFH.	27C256 x 2 (64KB) F0000～FFFFFH.
Main RAM	62256 x 2 (64KB) 00000H ～0FFFFH	62256 x 2 (64KB) 00000H ～0FFFFH
User memory space	Nil	27256 x 2 or 62256 x 2 (64KB) (Empty Socket)
Clock Generator	8284A	8284A
Keyboard	16 of hexadecimal keys and 8 function keys	16 of hexadecimal keys and 8 function keys
Speaker interface	SPEAKER: Able to test sound using with speaker and further more able to test synthesizer	SPEAKER: Able to test sound using with speaker and further more able to test synthesizer
Serial Port	Serial Port 8251 (RS-232C)	Serial Port 2 X 8251 (RS-232C x 2Port, 25Pin)
I/O Port	Two 8255	Three 8255
Programmable Timer	One 8253	One 8253
ADC	Single channel ADC (ADC 0804)	8 BIT x 8 Channel, ADC 0809

DA	D/A Converter DAC 0808 (8BIT x 1 Channel)	D/A Converter DAC 0808 (8BIT x 1 Channel)
Interrupt Controller	One 8259	One 8259
Keyboard and display controller	CD45328	8279
Experiment device	**Four Experiment Devices** 1. **DOTMATRIX LED**: Interfaced to 8255A (PPI) 2. **FND LED**: Interfaced to 8255A (PPI) 3. **LEVEL METER**: Connected to DAC 4. **STEPPING MOTOR INTERFACE:** So as to control stepping mo tor driver circuit of stepping motor is interfaced.	**Eight Experimental Devices** 1. **A/D Experiments** Contain (VR, Photo TR, Thermistor, MIC) 2. **D/A Experiments** Contain (2W AMP, Speaker) Speaker, MIC, Function 3. Battery Back-up Function 4. Thermistor Sensor Control Function 5. Photo Sensor Control Function 6. AMP, Recording Function 7. I/O Simulation LED x 8ea, Button Switch x 8ea 8. FND Control Function
Power Supply	POWER: AC 110～220V, D C +5V 3A , +12V 1A , -12V 0.5A SMPS	Power Supply (Switching) ±12V, +5V (90-260V) Free Volt

1.3.2 Trainer Outlooks

Fig. 1.1 MTS-86C microprocessor trainer kit picture

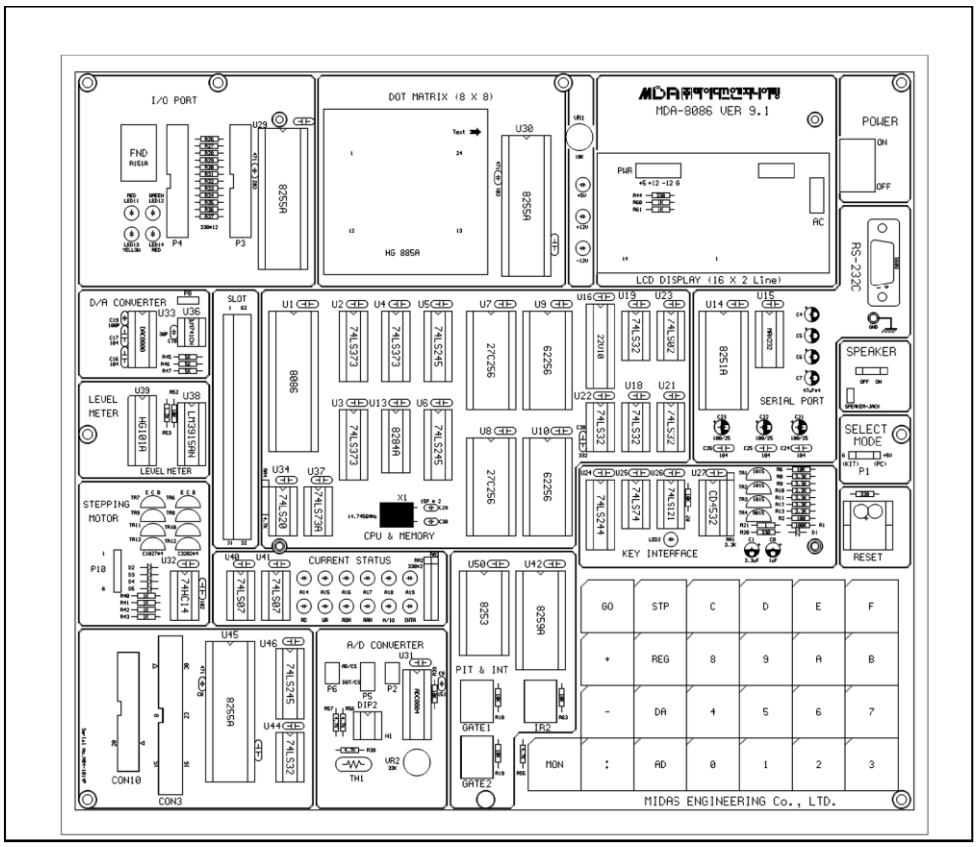


Fig. 1.2 The MDA-8086 trainer kit picture.

1.3.3 Memory Map and I/O Addresses for MDA-8086 and MTS-86C

1.3.3.1 Memory Mapping

Table 1.2 Memory mapping in MTS-86C and MDA8086 trainers.

Memory	MTS-86C	MDA-8086
Program and Data Memory (RAM)	00000H ~ 0FFFFH	00000H ~ 0FFFFH
Monitor ROM	F0000H ~ FFFFFH	F0000H ~ FFFFFH
User Free Space (RAM or ROM)	10000H ~ EFFFFH 27256 x 2 or 62256 x 2 (64KB) (Empty Socket)	No slot

Table 1.3 MDA-8086 I/O Address Map

Address	I/O Port Functions	Device
00H 02H 04H	INSTRUCTION REGISTER STATUS REGISTER DATA REGISTER	LCD Display
01H 03H	KEYBOARD REGISTER (Only read) KEYBOARD FLAG (Only write)	Keyboard

08H 0AH	DATA REGISTER INSTRUCTION /STATUS REGISTER	8251(Using to data communication)
09H 0BH 0DH 0FH	TIMER 0 REGISTER TIMER 1 REGISTER TIMER 2 REGISTER CONTROL REGISTER	8253(TIMER/COUNTER)
10H 12H	COMMAND REGISTER DATA REGISTER	8259(Interrupt controller)
11H	SPEAKER	SPEAKER
18H 1AH 1CH 1EH	A PORT DATA REGISTER B PORT DATA REGISTER C PORT DATA REGISTER CONTROL REGISTER	8255A-CS1(DOT & ADC INTERFACE)
19H 1BH 1DH 1FH	A PORT DATA REGISTER B PORT DATA REGISTER C PORT DATA REGISTER CONTROL REGISTER	8255-CS2(LED & STEPPING MOTOR)

Table 1.4 MTS-86C I/O Address Map

Address	I/O Port Function	Device
FFFFH FFFDH FFFBH FFF9H	CONTROL REGISTER C PORT DATA REGISTER B PORT DATA REGISTER A PORT DATA REGISTER	8255 - 1
FFFEH FFFCH FFFAH FFF8H	CONTROL REGISTER C PORT DATA REGISTER B PORT DATA REGISTER A PORT DATA REGISTER	8255 - 2
FFF2H FFF0H	INSTRUCTION /STATUS REGISTER DATA REGISTER	8251 (RS232C PORT 1)
FFEAH FFE8H	8279 Status/ Command 8279 Data	Keypad Control
FFDEH FFDCH FFDAH FFD8H	8253 Command 8253 Count2 8253 Count1 8253 Count0	Counter & Timer
FFD2H FFD0H	8251-2 Command 8251-2 Data	RS232C PORT2
FFCAH FFC8H	COMMAND REGISTER DATA REGISTER	8259 - Interrupt Control
3FF0H	FND Display	
3FD8H	D/A Converter	8 Bit D/A Converter
3FD6H 3FD4H 3FD2H 3FD0H	CONTROL REGISTER C PORT DATA REGISTER B PORT DATA REGISTER A PORT DATA REGISTER	<For experiments> Connect to speaker Experiment for Output(8bits) Experiment for Input(8bits)
3FCEH 3FCCH 3FCAH 3FC8H	A/D Converter IN3 or IN7 A/D Converter IN2 or IN6 A/D Converter IN1 or IN5 A/D Converter IN0 or IN4	8 Bit A/D Converter IN0~3, IN4-IN7 can be modified by DIP8 of S/W 3

1.4 Working with MDA-8086

1.4.1 Keyboard Functions

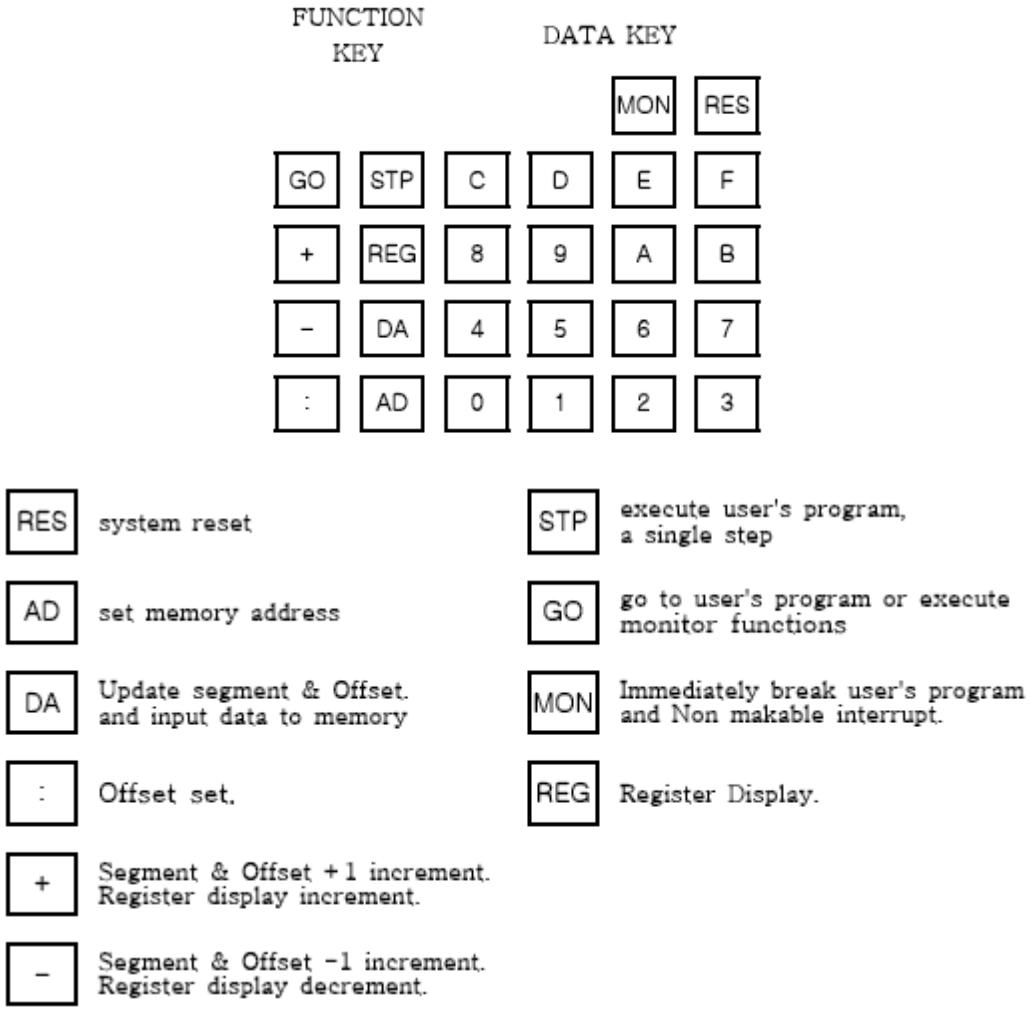


Fig. 1.3 MDA-8086 Keyboard and their functions

1.4.2 Serial Communication with a PC

a) Connect the serial cable as shown in Fig. 1.4 between the PC COM1 port and MDA-8086 serial port.

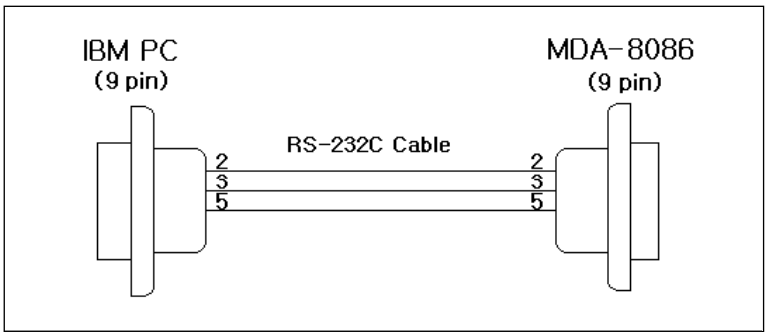


Fig. 1.4 Serial communication cable to be connected between PC and MDA-8086

a) Set the jumper of P1 into right position as shown in Fig. 1.5.

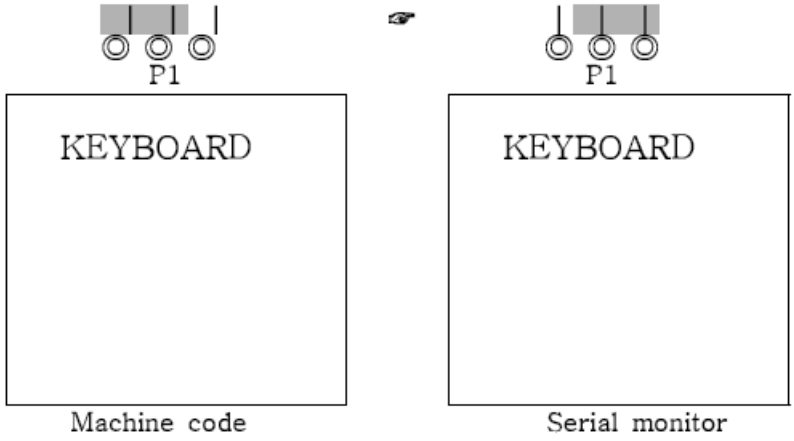


Fig. 1.5 Put jumper in right position for serial monitor

b) Run WinComm program from Windows. A blank window will open as shown in Fig. 1.6.

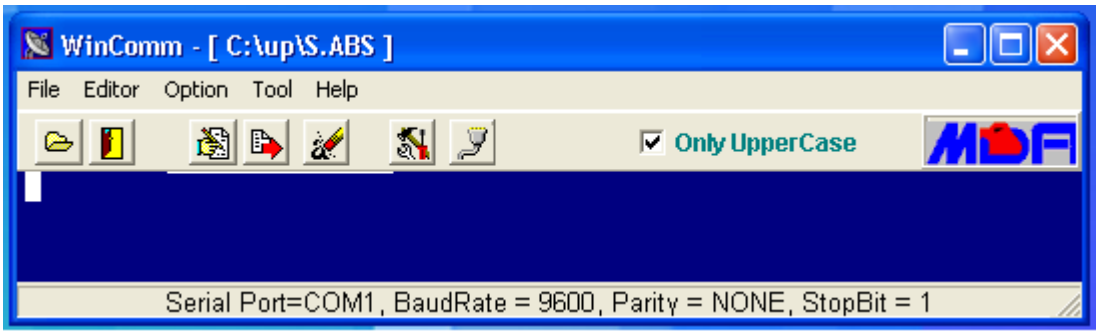


Fig. 1.6 WinComm Window.

c) Press the RESET button of MDA-8086 trainer. Some text lines as shown in Fig. Fig. 1.7 will appear in the WinComm window.

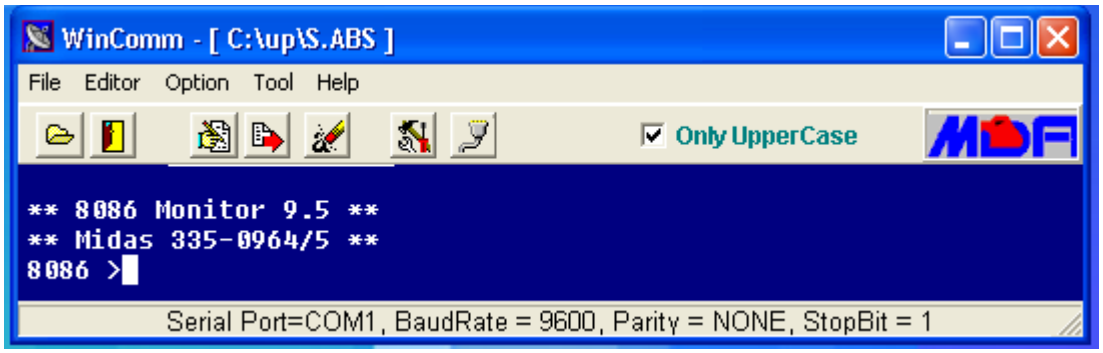


Fig. 1.7 WinComm window after RESET button of MDA-8086 is pressed

d) The communication is established and the user can put command in the WinComm window prompt for controlling the MDA-8086 trainer.

1.4.3 MDA 8086 Command List

In the "WinComm" windows, the MDA 8086 accepts the following commands:

Command	Summary	Usage
?	Help Command	
E	Enter Data To Memory	E segment : offset
D	Dump Memory Contents	D segment : offset length
R	Register Display & Change	R [register name]
M	Move Memory From 1 to 2	M address1, length, address2
F	Fill Memory With Any Data	F address, length, data
L	Program Down Load	L Return key
G	Execute Program	G segment : offset
T	Program 1 step execute	

To get the command help, type **?** and press enter in the WinComm window prompt. A Help lisiting will appear as shown in Fig. 1.8.

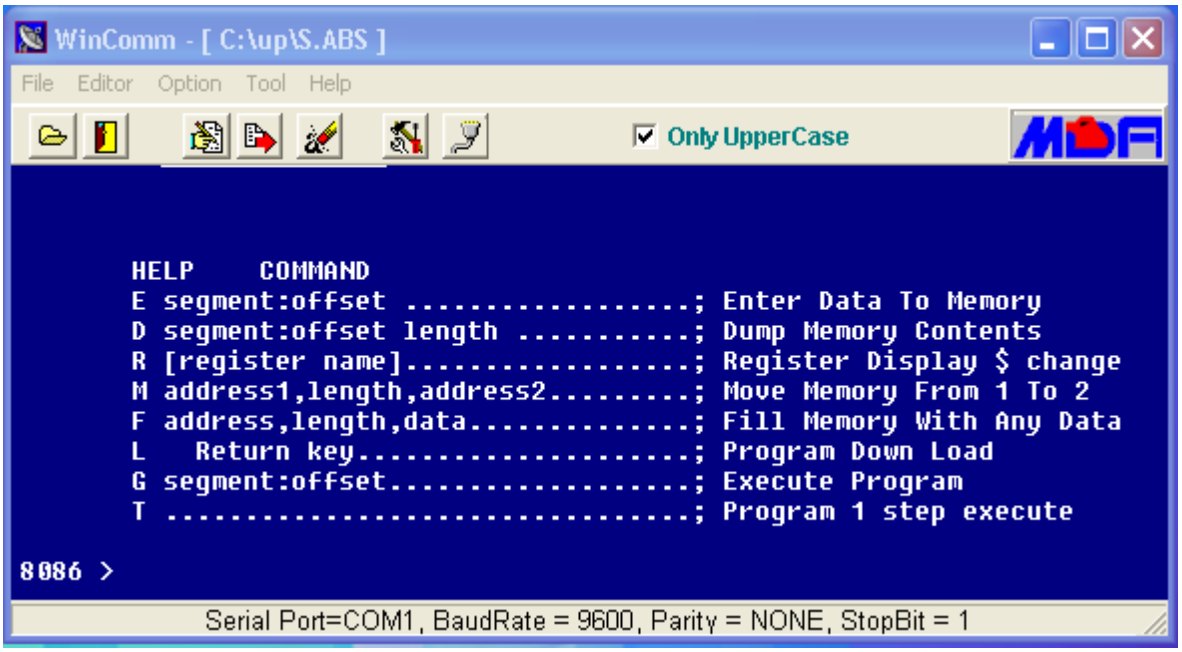


Fig. 1.8 Help window in WinComm

1.4.4 Program Download and RUN

In the WinComm window command prompt, do the following for loading a program into the MDA-8086 trainer and running the program from the command prompt:

a) Type L and press Enter. A message will appear in the window "Down load start !!" as shown in Fig. 1.9.

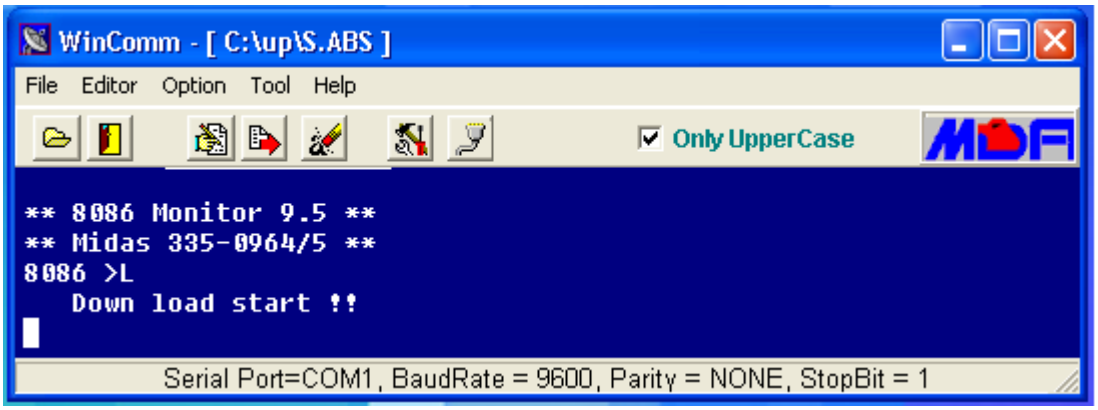


Fig. 1.9 Loading an ABS File

b) Press the "Page UP" button. The user will be asked to enter the filename. Browse and choose the ABS file and press ok. The HEX file content will be echoed in the WinComm window and at the same time the HEX file will be downloaded into the RAM of the MDA-8086 trainer.

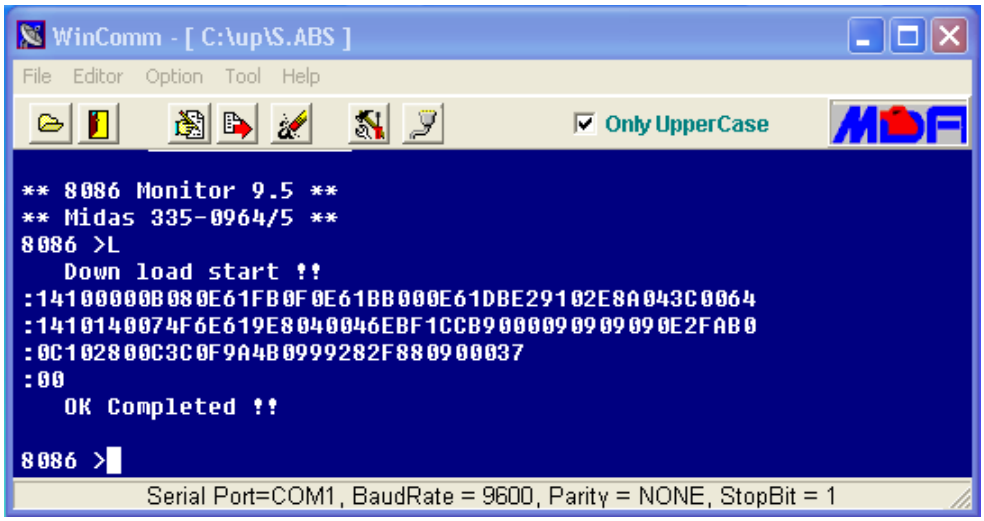


Fig. 1.10 Downloading an ABS file into MDA-8086 trainer

c) Type G and Press Enter. The HEX program will run from 0000:1000H location on the MDA-8086 trainer and echoed on the WinComm window as shown in Fig. 1.11.

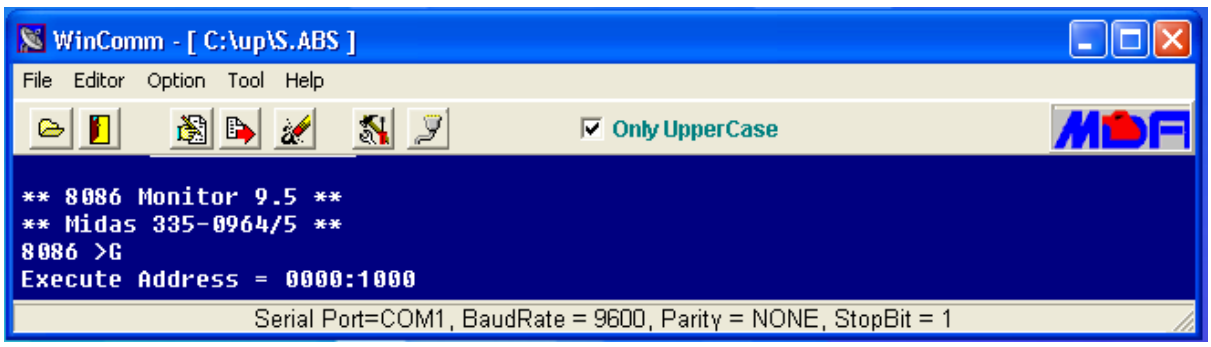
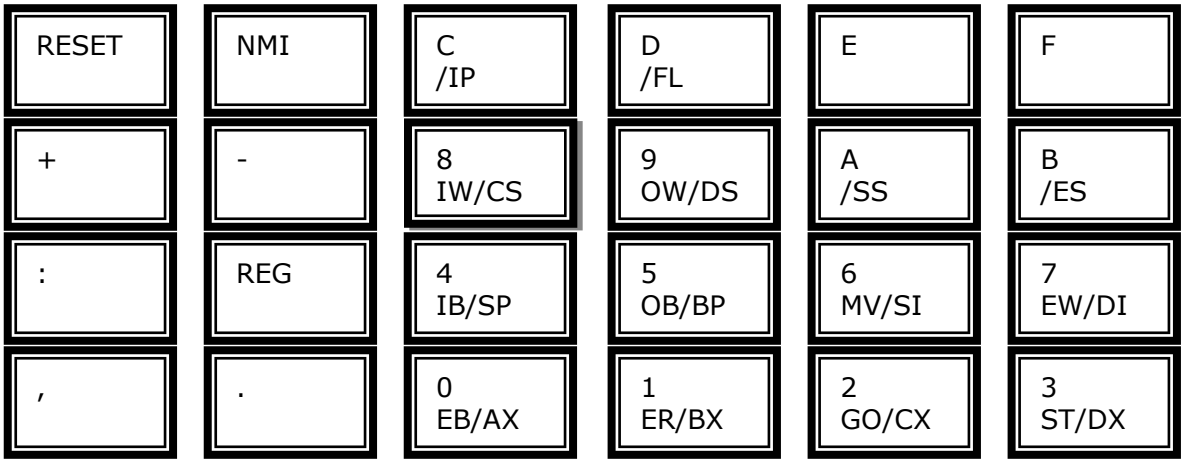


Fig. 1.11 Executing a program into the MDA-8086 trainer from WinComm

1.5 Working with MTS-86C

1.5.1 MTS-86C Keypad Functions

Function Keys Hexadecimal Keys



1.5.2 Key Functions of MTS-86C

Function Keys	Function
RESET	Master reset, terminates any recent activity and displays MTS86C message on the LCD
NMI	Used to generate an immediate type 2 interrupt (NMI)
+	Allows the user to add two hexadecimal values
-	Allows the user to subtract one hex number from another
:	Used to separate an address to be entered into two parts: SEGMENT and OFFSET value
REG	Allows the user to use the contents of any of the 8086's registers as an address or data entry
,	Used to separate keypad entries and to increment the address field to the next consecutive memory locations
•	The command terminator

Function	Hexadecimal Keys	Example	Usage
Examine Byte	0 [EB/EX]	Examine the memory location 0000:0011	RESET → EB → 11 → ,
Examine Register	1 [ER/BX]	Change the content of ES to 0010	RESET → ER → ES → 10 → •
Examine Word	7 [EW/DI]	Examine a word in memory location 0100:0000	RESET → EW → 0100 →: → 0000 → ,
Executing a Program	2 [GO/CX]	Run a program stored in location F000:0000	GO → F000 → : → 0000 → •

1.5.3 Program Development for MTS-86C

The assembly program, MASM.EXE (assembler), LINK.EXE (linker), EXE2BIN.EXE (EXE to BIN converter), BIN2HEX.EXE (BIN to HEX converter) should be located in the same folder.

At the beginning of the experiment, students should copy all these utility files along with V.BAT from C:\MTS86C to the working folder.

1.5.3.1 Composing an assembly program

Step 1. To compose program and text editor may be used. You can use Windows DOS text editor "EDIT.COM'.
   a. Goto windows task bar. Click **start** and then click **RUN**.
   b. Type **edit** on the RUN edit box and then press **Enter** button on the keyboard.
   c. The **EDIT** text editor window will be opened where the user can write the assembly program.

Step 2. Save the assembly program with an extension ASM, say **prog1.asm**

1.5.3.2 Assemble

Step 1. Microsoft assembler MASM.EXE would assemble the assembly program. MASM should be run from DOS command prompt.

Step 2. Go to the DOS command prompt C:\ (**start →All Programs → Accessories → Command Prompt**)

Step 3. Type MASM and press **Enter**. MASM will run and ask for the assembly filename. Once the assembly filename is entered, MASM will produce object file '.OBJ', listing file `.LST', and cross reference file '.CRF'.

1.5.3.3 Making HEX file from Object File

Step 1. Run **LINK** program to change from OBJ to EXE file
   a. Usage: "FOLDER PATH"\LINK PROG1

Step 2. Run EXE2BIN program to convert EXE to BIN file
   a. Usage: "FOLDER PATH"\EXE2BIN PROG1.EXE

Step 3. Run BIN2HEX program to convert BIN to HEX file
   a. Usage: "FOLDER PATH"\BIN2HEX

Step 4. Finally delete the PROG1.EXE file
   a. Usage: "FOLDER PATH"\DEL PROG1.EXE

1.5.3.4 Making the HEX file from ASM file directly

The user can run the V.BAT file where all the utility programs are commanded sequentially.

Step 1. Run the V.BAT file on the assembly program
   a. Usage: "FOLDER PATH"\V PROG1

Step 2. To perform BIN2HEX, the user will be asked to '**Input BIN filename'.** Type PROG1.BIN and press enter. The **PROG1.HEX** will be generated.

1.6 Intel HEX or ABS file format

Intel's Hex-record format allows program or data files to be encoded in a printable (ASCII) format. This allows viewing of the object file with standard tools and easy file transfer from one computer to another, or between a host and target. An individual Hex-record is a single line in a file composed of many Hex-records.

1.6.1 HEX-record content

Hex-Records are character strings made of several fields which specify the record type, record length, memory address, data, and checksum. Each byte of binary data is encoded as a 2-character hexadecimal number: the first ASCII character representing the high-order 4 bits, and the second the low-order 4 bits of the byte.

The 6 fields which comprise a Hex-record are defined as follows:

Field		Characters	Description
1	Start code	1	An ASCII colon, ":".
2	Byte count	2	The count of the character pairs in the data field.
3	Address	4	The 2-byte address at which the data field is to be loaded into memory.
4	Type	2	00, 01, or 02.
5	Data	0-2n	From 0 to n bytes of executable code, or memory loadable data. n is normally 20 hex (32 decimal) or less.
6	Checksum	2	The least significant byte of the two's complement sum of the values represented by all the pairs of characters in the record except the start code and checksum.

Each record may be terminated with a CR/LF/NULL. Accuracy of transmission is ensured by the byte count and checksum fields.

1.6.2 HEX-record types

There are three possible types of Hex-records.

00	A record containing data and the 2-byte address at which the data is to reside.
01	A termination record for a file of Hex-records. Only one termination record is allowed per file and it must be the last line of the file. There is no data field.
02	A segment base address record. This type of record is ignored by Lucid programmers.

1.6.3 HEX-record example

Following is a typical Hex-record module consisting of four data records and a termination record.

```
:1001000021460136012147013600 7EFE09D2190140
:100110002146017EB7C20001FF5F16002148011988
:10012000194E79234623965778239EDA3F01B2CAA7
:100130003F0156702B5E712B722B732146013421C7
:00000001FF
```

The first data record is explained as follows:

:	Start code.
10	Hex 10 (decimal 16), indicating 16 data character pairs, 16 bytes of binary data, in this record.
01	Four-character 2-byte address field: hex address 0100,
00	indicates location where the following data is to be loaded.
00	Record type indicating a data record.

The next 16 character pairs are the ASCII bytes of the actual program data.

40	Checksum of the first Hex-record.

The termination record is explained as follows:

:	Start code.
00	Byte count is zero, no data in termination record.
00	Four-character 2-byte address field, zeros.
00	
01	Record type 01 is termination.
FF	Checksum of termination record

1.6.4 Calculating the Checksum

As mentioned in the format table above, the last two characters represent a checksum of the data in the line. Since the checksum is a two-digit hexadecimal value, it may represent a value of 0 to 255, inclusive.
The checksum is calculated by summing the value of the data on the line, excluding the leading colon and checksum byte itself, and taking its two's complement. For example, the line:

:0300300002337A1E

Breaking this line into it's components we have:

Record Length: 03 (3 bytes of data)
Address: 0030 (the 3 bytes will be stored at 0030, 0031, and 0032)
Record Type: 00 (normal data)
Data: 02, 33, 7A
Checksum: 1E

Taking all the data bytes above, we have to calculate the checksum based on the following hexadecimal values:

03 + 00 + 30 + 00 + 02 + 33 + 7A = E2

The two's complement of E2 is 1E which is, as you can, the checksum value.
For those unfamiliar with calculating a two's complement, it's quite simple: The two's complement of a number if the value which must be added to the number to reach the value 256 (decimal). That is to say, E2 + 1E = 100.
You may also calculate the two's complement by subtracting the value from 100h. In other words, 100h - E2h = 1Eh -- which is the checksum.
If the value in question is greater than FFh, simply take the part which is less than 100h. For example, if you want the two's complement of the value 494h, simply drop the leading "4" which leaves you with 94h. The two's complement of 94h is 6Ch.

1.6.5 File Transmission between PC to MTS-86C

a) Power on the MTS-86C trainer and connect the RS -232 cable between the PC (Com port) and the MTS-86C (RS232-1 port)

Important: You must use the RS232 cable from K&H Company -- One side with 9 -Pin D type connector and the other side with 25 -Pin D type connector.

b) Launch "Hyper Terminal" in your windows. The program path is located at [Start / Accessories / Communications / Hyper Terminal]
c) Enter a name and choose an icon for connection as shown below (Fig. 1.12) press ok.

Fig. 1.12 Hyper terminal window

d) Select proper com port, COM1 for example, and press ok.

Fig. 1.13 Port selection in a hyper terminal.

e) Select the parameters for COM port as shown below (Fig. 1.14) and press OK.

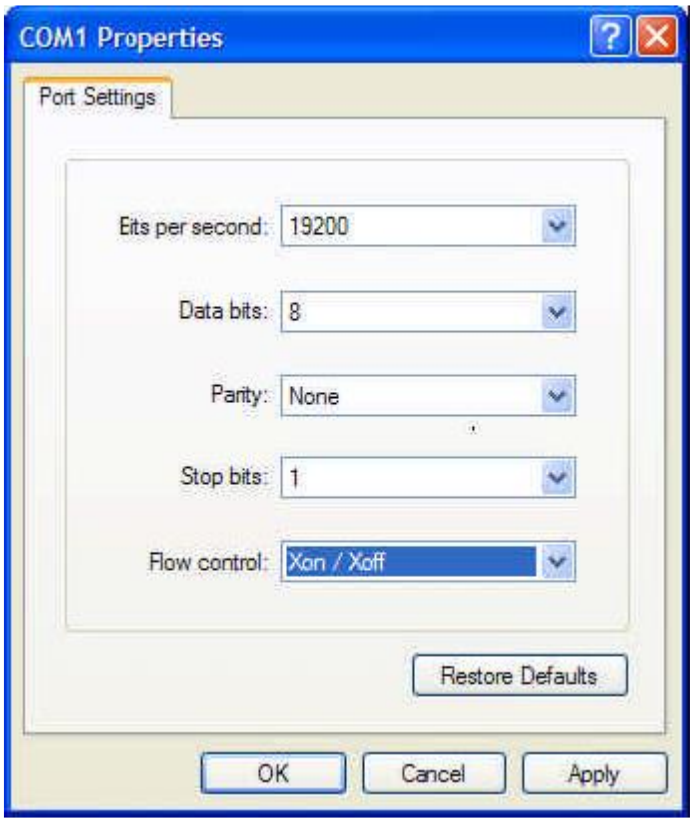


Fig. 1.14 Configuring the hyper terminal

f) Press "RESET" button on MTS-86C and then press any key among "A~F". The following message will be shown on "Hyper Terminal", which means that connection is established through RS-232 cable. (Fig. 1.15)

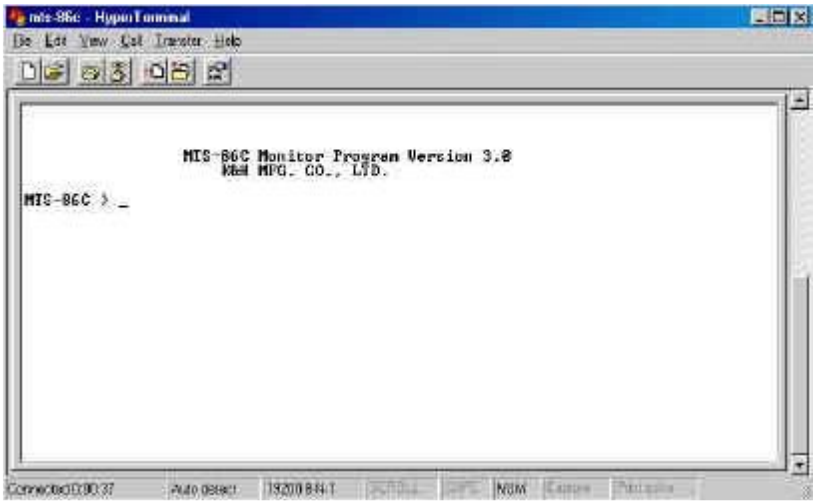


Fig. 1.15 Pressing any key from A-Z in MTS-86C trainer establish connection.

g) You can Press "H" and press "Enter" from PC to ask for command help (Fig. 1.16).

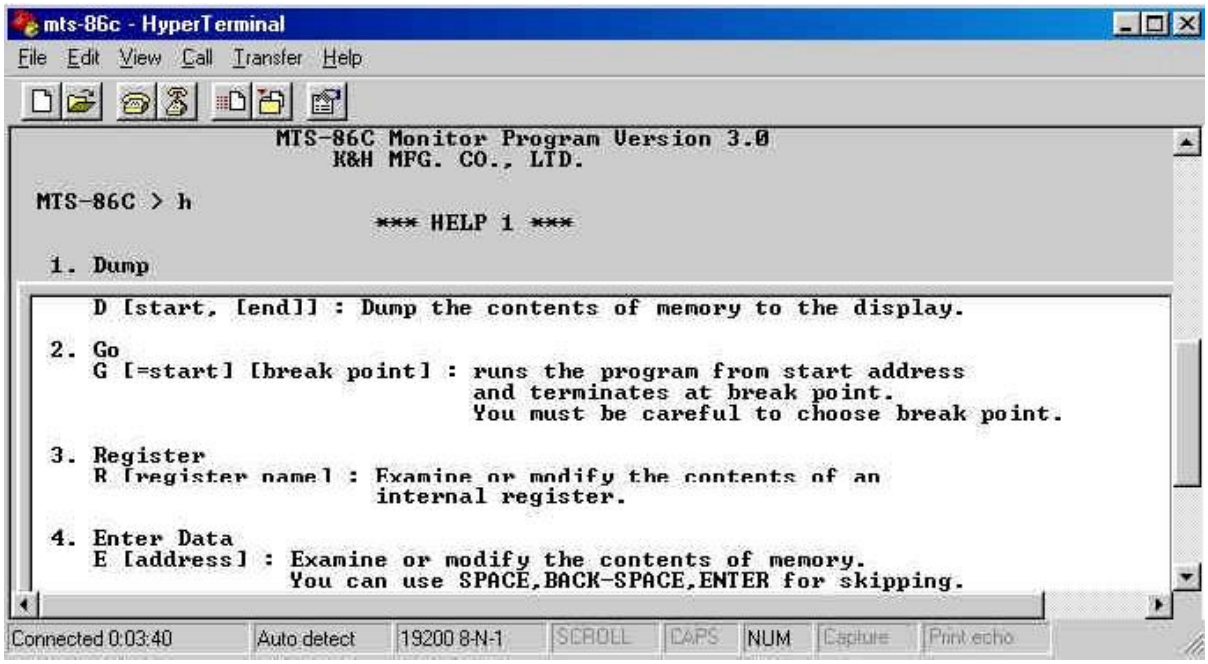


Fig. 1.16 Help command output in the hyper terminal window.

h) Example for running a program by using Hyper Terminal (Fig. 1.17)
(1) Press "L" and "Enter", as shown in Fig. 1.17
(2) Select "Transfer >> Send Text File " from the menu bar

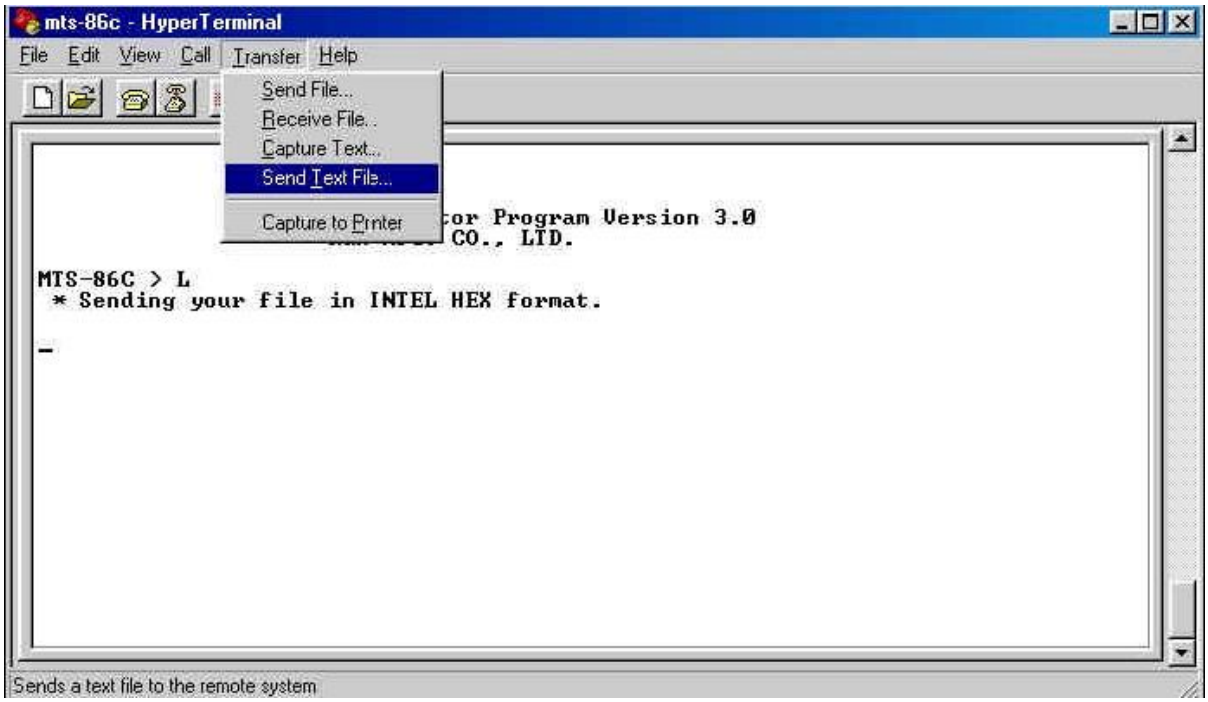


Fig. 1.17 Loading a HEX file.

i) Browse the *.hex file, as shown in Fig. 1.18

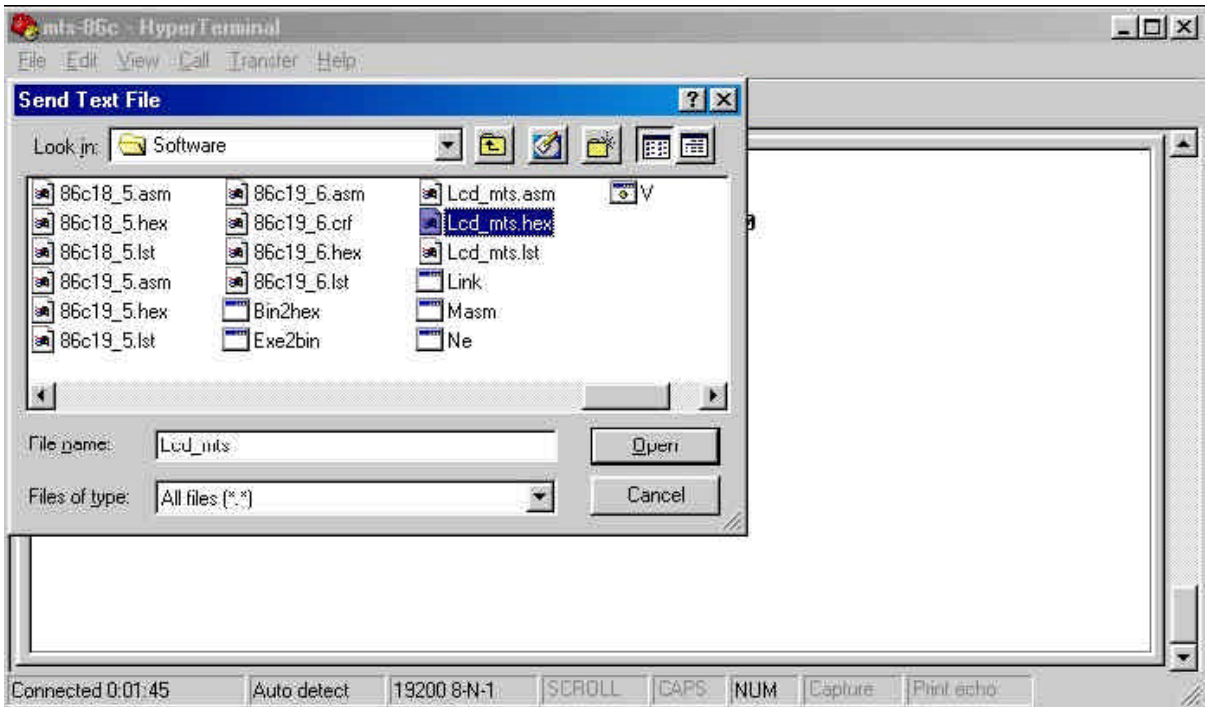


Fig. 1.18 Selecting the HEX file

j) Press "G" and "Enter" and the program will be performed on MTS-86C trainer, as shown in Fig. 1.19.

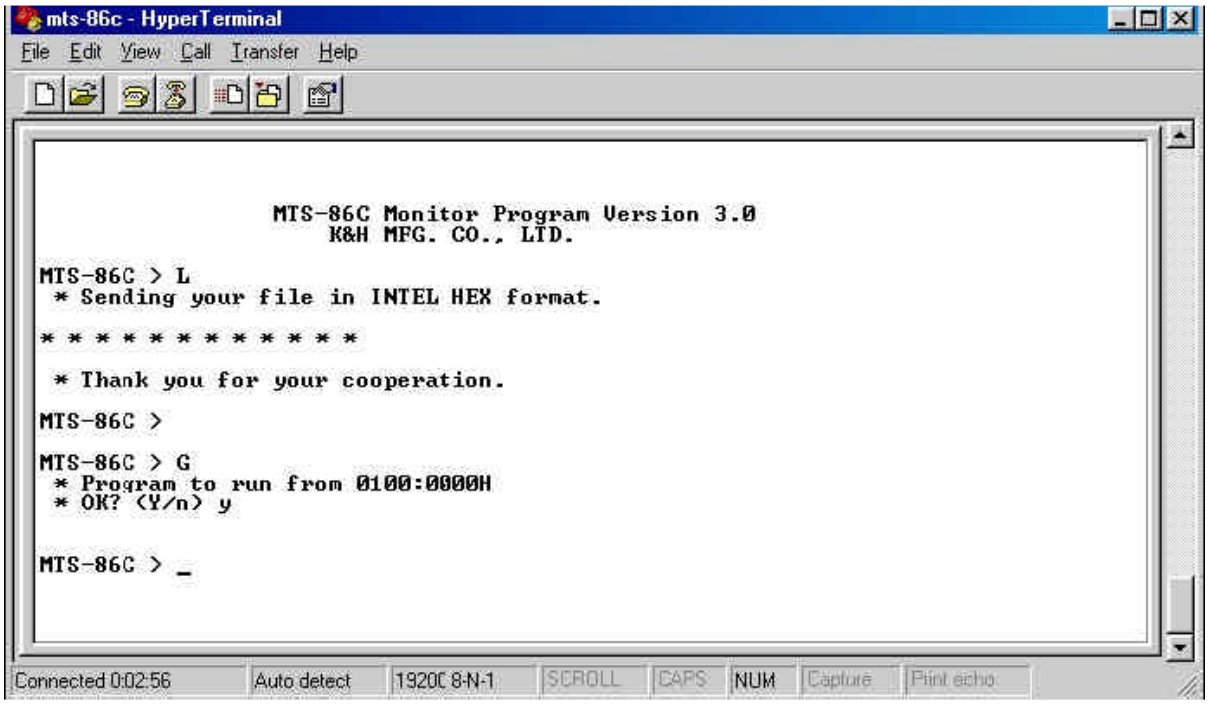


Fig. 1.19 Executing program using G command

1.7 Experimenting with MTS-86C and MDA-8086

In the previous sections procedures for using MTS-86C and MDA-8086 trainers are given in details. Here the students are asked to write an assembly program, make machine codes (HEX or ABS file), and insert the program into the trainer kits both manually and from PC. Finally, run and examine results.

1.7.1 Default Values after Serial COMM Link

When the serial COMM link between the PC and trainer is established (using either Hyper Terminal or WinComm), the default values for the CS, DS and IP of MTS-86C and MDA-8086 are different.

Table 1.5 Register values after serial COMM link

Register	MTS-86C	MDA8086
CS	0100H	0000H
DS	0100H	0000H
IP	0000H	1000H

Although in both cases, the starting physical memory location is 01000H, the segment bases are different. Students should be aware of this information.

1.7.2 Assembly Program Build and RUN

1. Type the following assembly program using Windows "EDIT" editor.

```
;*****************************************************
; Program name: DAY1.ASM
; Moving a word data from one memory location to another location
; Original data = 2048H
; Original data location = 0100:0150H
; Data to be stored in location = 0100:0160H
```

```
;*****************************************************

CODE  SEGMENT
        ASSUME        CS:CODE, DS:CODE
        ;
        ORG    100H           ; Program Effective Address, IP = 100H
        MOV    AX, CS
        MOV    DS, AX         ; Make DS = CS
        MOV    AX, DATA1
        MOV    DATA2, AX
        HLT

        ORG    150H           ; Location of source data at 150H
DATA1 DW     2048H

        ORG    160H           ; Location of final data in 160H
DATA2 DW     ?

CODE  ENDS                    ; Segment end
        END                   ; End of Assembly Program
;*****************************************************
```

2. Save it using the filename DAY1.ASM
3. Go to the DOS prompt and make the HEX (ABS) file and the listing file
4. Note down the HEX (ABS) file in your note sheet along with the listing file
5. Download the HEX (ABS) file from the PC to the microprocessor trainer and run it from the PC
6. Examine the contents of the original and final data locations, and note down the results
7. Examine and note down the content of AX register
8. Disconnect the serial connection between the PC and trainer
9. Restart the trainer
10. Manually key in the HEX (ABS) file codes into the trainer from location 0100:0100H
11. Manually put data 2048H in location 0100:0150H
12. Reset the trainer
13. Examine the HEX (ABS) file codes in the trainer to be sure that the program stored in the RAM properly
14. Run the program
15. Examine and note down the program result (original data, final data and AX register)

1.7.3 Program Running using Single Stepping

1. Type the following assembly program using Windows "EDIT" editor.

```
;*****************************************************
; Program name: DAY1ST.ASM
; Moving a word data from one memory location to another location
; Original data = 2048H
; Original data location = 0100:0150H
; Data to be stored in location = 0100:0152H
;*****************************************************

CODE  SEGMENT
        ASSUME        CS:CODE, DS:CODE
        ;
        ORG    100H           ; Program Effective Address, IP = 100H
        MOV    AX, CS
        MOV    DS, AX
```

```
        MOV   AX, DATA1
        MOV   BX, AX
        MOV   CX, BX
        MOV   DX, CX
        MOV   DATA2, DX
        HLT

        ORG   150H          ; Location of source data at 150H
DATA1   DW    2048H
DATA2   DW    ?

CODE    ENDS                ; Segment end
        END                 ; End of Assembly Program
;************************************************
```

2. Save it using the filename DAY1ST.ASM
3. Go to the DOS prompt and make the HEX (ABS) file and also the listing file
4. Note down the HEX (ABS) file in your note sheet as well as the listing file
5. Download the HEX (ABS) file from the PC to the microprocessor trainer
6. Run the program into the trainer from the PC using single stepping mode
7. Examine and note down the results in every step.
8. Disconnect the serial connection between the PC and trainer
9. Restart the trainer
10. Manually key in the HEX (ABS) file codes into the trainer from location 0100:0100H
11. Manually put data 2048H in location 0100:0150H
12. Reset the trainer
13. Examine the HEX (ABS) file codes in the trainer to be sure that the program stored in the RAM properly
14. Run the program using single stepping mode from the trainer
15. Examine and note down the program result in every step

1.8 Report

The students should write a report (individual submission) on the performed experiment (s). The report should contain the experiment number, experiment name, objective, apparatus, methodology etc along with the followings:

1.8.1 Result and Discussion

Write the following in the report

1. Key features of MTs-86C and MDA-8086 trainers
2. The assembly program and the results
3. Listing file generated during programming

1.8.2 Question Answer

a) What are the physical location of DATA1 and DATA2 in the assembly programs
b) Why should you need to copy contents of the CS to DS

1.8.3 Conclusion

Write a conclusion on this experiment highlighting the trainer features and programming using the trainers.